
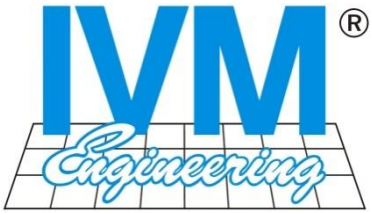


**Masterarbeit**

# Webbasierte Überwachung und Visualisierung von Anlagedaten

Studienrichtung Softwareentwicklung und Wirtschaft
Fakultät für Informatik
Sommersemester 2010

Thomas Gebhard
thomas.gebhard@student.tugraz.at

**Betreuer:** Dipl.-Ing. Dr.techn. Granitzer Michael
**Institut:** Institut für Wissensmanagement

---

**Betreuer:** Dipl.-Ing. Andreas Hafellner
**Firma:** IVM Technical Consultants

# Kurzfassung

Das 21. Jahrhundert ist durch Energie- und Ressourcenverschwendung gekennzeichnet. Die Auswirkungen dieses Verhaltens können in der ganzen Welt wahrgenommen werden. Der Wandel der Gesellschaft hin zur nachhaltigen Nutzung der zur Verfügung stehenden Ressourcen kann als Anstoß für diese Arbeit gesehen werden.
Diese Masterarbeit beschäftigt sich mit der Erstellung eines webbasierten Ansatzes zur Überwachung und Visualisierung von Anlagedaten. Die zu überwachenden Informationen kommen aus dem Bereich der Energietechnik. Einleitend werden die Rahmenbedingungen dieser Arbeit definiert. Des Weiteren werden die funktionalen und nichtfunktionalen Anforderungen erläutert. Aufbauend darauf werden softwareentwicklungstechnische Entscheidungen getroffen, welche bei der Implementierung der RIA-Applikation erforderlich waren. Die Applikation ist Teil eines Gesamtproduktes, welches als verteiltes System realisiert ist. Der agile Softwareentwicklungsprozess Scrum wird als Prozess eingesetzt. Technologisch wurde die RIA-Applikation mit Silverlight realisiert.
Der praktische Teil dieser Arbeit veranschaulicht die Realisierung der Visualisierung der Anlagedaten im zeitlichen, domain- und geospezifischen Kontext. Zu Beginn des praktischen Abschnittes wird ein Überblick über die RIA-Applikation gewährt, darauf aufbauend veranschaulichen Code-Ausschnitte zum besseren Verständnis die verschiedenen Visualisierungen. Abschluss dieses Teils bildet die Erklärung des Zusammenspiels der einzelnen Module der RIA-Applikation. Das Resümee über die eingesetzten Technologien und Praktiken bildet den Gesamtabschluss dieser Masterarbeit.

# Abstract

The 21st century is characterized by a tremendous waste of energy and non-renewable resources. The effects of this behavior can be noticed throughout the world. Our society has already started to some extent to change their way of consumption towards a more sustainable utilization of available resources. The increasing importance of this noticeable change has been a major inspiration for this master thesis.
In a nutshell, this master thesis focuses on the creation of a distinctive web-based approach with the purpose of monitoring and visualizing complex data. As the topic implies, the data of interest derives from the energy sector. For a better understanding of the issue discussed, the paper starts out with a clear definition of the basic conditions and framework requirements. In that vein, the functional and non-functional requisites are explained. The information provided is inevitable for the chapters to follow, where critical decisions regarding the technical development and advancement of the software are taken. These decisions are crucial for a successful implementation of the RIA application. The RIA application is an integral part of an already implemented and complete distributed system. For the realization the agile Scrum Software Development tool is used as a process. From a technological point of view, the RIA application has been created with Silverlight.
Following, the paper focuses on the practical part which, in essence, demonstrates how the visualization of the system data in a time, domain and geo-specific context can be realized. First off, there will be provided an overview of the RIA application. In a second step, for a better illustration and understanding of the various visualizations some code snippets will be showcased. The last chapter concentrates on a thorough explanation of the interplay of the individual modules of the RIA application. As a final step, the master thesis concludes with a roundup and summary of the various technologies and methods applied.

Deutsche Fassung:
Beschluss der Curricula-Kommission für Bachelor-, Master- und Diplomstudien vom 10.11.2008
Genehmigung des Senates am 1.12.2008

## EIDESSTATTLICHE ERKLÄRUNG

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommene Stellen als solche kenntlich gemacht habe.

Graz, am ……..… ……..
(Unterschrift)

Englische Fassung:

## STATUTORY DECLARATION

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

…… ……..
date (signature)

# Danksagung

Ich bedanke mich auf diesem Wege bei meinen Eltern, die mir meine schulische Ausbildung und mein Studium ermöglicht haben.

Des Weiteren danke ich meinem Betreuer Michael Granitzer für die Freiheit bei der Themenauswahl. Auch der Firma IVM möchte ich meinen Dank aussprechen, da sie es mir ermöglicht hat Arbeit und Studium erfolgreich zu vereinen. Insbesondere bedanke ich mich bei Andreas Hafellner, der mir viel Freiraum und Möglichkeiten bei der Zusammenarbeit eingeräumt hat.

Meiner Freundin Sabrina danke ich für ihre Geduld, die aufmunternden Worte und ihre Unterstützung während meiner Studienzeit. Für die Unterstützung danke ich des Weiteren meiner Schwester Christine und meinem Studienkollegen Markus.

Thomas Gebhard

Graz, September 2010

# Inhaltsverzeichnis

# 1. Einleitung

## 1.1. Motivation

> *„Zahlreichen Untersuchungen zufolge (2) könnte die EU mindestens 20 % ihres derzeitigen Energieverbrauchs auf kostengünstige Weise einsparen; dies entspricht einem Gegenwert von 60 Mrd. EUR pro Jahr oder dem gegenwärtigen gemeinsamen Energieverbrauch von Deutschland und Finnland.“ (Kommission, 2005)*

> *„Die verbrannten Milliarden: Deutschland knausert, aber ein großes Potenzial bleibt unberührt: Haushalte und Betriebe könnten viel Geld sparen, wenn sie Energie intelligenter nutzten“ (Vorholz, 2003)*

Diese Zitate und der Wandel der Gesellschaft hin zur nachhaltigen Nutzung der zur Verfügung stehenden Ressourcen kann als Anstoß für diese Masterarbeit gesehen werden. Die Gesellschaft ist bereit, für Energieeinsparungen und Nachhaltigkeit Investitionen zu tätigen.

> *„Verlässliche, umweltfreundliche und kostengünstige Energieversorgung wird zunehmend zur Schlüsselfrage für Gesellschaft und Wirtschaft. Forschung und Entwicklung leisten einen zentralen Beitrag zur Sicherung und Weiterentwicklung unseres Energiesystems und stehen auf der Agenda internationaler Aktivitäten ganz oben.“ (Strategieprozess Energie 2050)*

Aus den angeführten Zitaten kann man erkennen, dass die Domäne Energie ein wesentlicher Bestandteil dieser Masterarbeit ist. Die Überwachung und Visualisierung des Energiebedarfs von diversen Anlagen, privat als auch industriell, in Kombination mit der Materie Internet ist der Schwerpunkt dieser Arbeit.

## 1.2. Zielsetzung der vorliegenden Arbeit

Warum die Thematik Internet?

Zum einen tendieren die Menschen in Zeiten von Web 2.0 immer mehr dazu, Informationen in das Internet zu transferieren (Murugesan, 2007) und zum anderen ist es aus Sicht der Green IT vorteilhaft, wenn für das Aufzeichnen von Daten ein zentraler Knotenpunkt (Aikebaier, Yang, Enokido, & Takizawa, 2009) verwendet wird. Ansonsten würde jeder Nutzer eines Überwachungssystems seine eigene Infrastruktur aufbauen müssen, dies verbraucht Ressourcen und Energie.

Ziel dieser Arbeit ist es, der Ressourcen- und Energieverschwendung vorbeugend, ein webbasiertes System zur Überwachung des Energiebedarfs zu verwirklichen.

Der Begriff „webbasiert“ kann sehr weitläufig (Tamm & Günther, 2005) definiert werden, daher wird dieser nochmals mittels des Schlagwortes Rich Internet Application(RIA[1]) spezifiziert. Die aktualisierte Definition der Zielsetzung ist eine Rich Internet Applikation zur Überwachung des Energiebedarfs von Anlagen. Grundsätzlich ergeben sich hierbei folgende Fragen:

- Welche Technologien ermöglichen es eine RIA-Applikation zu realisieren?
- Welche Technologie ist für dieses Projekt am besten geeignet?

Eine weitere Zielsetzung dieser Masterarbeit ist die Verwendung eines agilen Softwareentwicklungsprozesses zur effizienten Umsetzung von Rich Internet Applikationen. Die Verwendung eines agilen Softwareentwicklungsprozesses wurde durch das hoch spezialisierte Umfeld der beteiligten Domänen Energietechnik und Softwareentwicklung notwendig. Dies ermöglicht es, in kurzen Iterationen auf die speziellen Bedürfnisse der Domäne der Energietechniker zu reagieren. Zusätzlich ergeben sich hierbei folgende Fragen:

- Welcher agile Softwareentwicklungsprozess ist am besten für die Rahmenbedingungen des Projektes geeignet?
- Was sind die Vor- und Nachteile eines nicht agilen Softwareentwicklungsprozesses in Bezug auf dieses Projekt?

Zu guter Letzt ist der Bereich der Informationsvisualisierung die dritte Zielsetzung dieser Arbeit. Dabei sollen verschiedene Visualisierungstechniken von Energiedaten unter Anwendung von Mash-Ups im zeitlichen-, geospezifischen- und domainspezifischen-Kontext realisiert werden.

Zusammenfassend kann man sagen, dass die Erstellung einer Rich Internet Applikation unter Verwendung eines agilen Softwareentwicklungsprozesses zur Überwachung von energetischen Anlageinformationen inklusive fortführender Informationsvisualisierung das Hauptziel dieser Arbeit ist.

## 1.3. Herangehensweise

Diese Arbeit ist wie folgt aufgebaut:

Das Kapitel 2 definiert die Anwendungsszenarien und die zu erwartenden Herausforderungen, welche die Applikation zu bewältigen hat. Es werden funktionale und nichtfunktionale Anforderungen angeführt. Aufgrund der Interaktionen mit dem Auftraggeber wurden fortlaufend neue bzw. abweichend priorisierte Anforderungen der Applikation hinzugefügt. Dies führte zur Notwendigkeit eines Softwareentwicklungsprozesses, welcher in Kapitel 3 beschrieben wird.

[1] „*Der Begriff Rich Internet Applikation beschreibt eine Anwendung, die Internet Techniken benutzt und eine intuitive Benutzeroberfläche bietet.*“ http://de.wikipedia.org/wiki/Rich_Internet_Application

Da diese Arbeit Teil eines Gesamtprojektes ist, musste für die Implementierung eine geeignete Technologie gewählt werden. Die Auswahl der einzusetzenden Technologien und diverser Graphical User Interface (GUI) Architekturen wird in Kapitel 4 und Kapitel 5 beschrieben.

Das Kapitel 6 beinhaltet ausgewählte Details zur Implementierung, um den praktischen Einsatz der Software zu verdeutlichen. Dieses Kapitel schließt auch realisierte Usecases und Klassendiagramme der RIA-Applikation mit ein.

Die Kapitel 7 und 8 bilden den Abschluss dieser Masterarbeit. Zum einen wird dies einen Erfahrungsbericht der eingesetzten Technologien beinhalten, zum anderen einen Ausblick über den weiteren Verlauf der Implementierung geben.

# 2. Anwendungsszenarien

In diesem Kapitel werden die Anwendungsszenarien mithilfe von Usecases dargelegt. Aus den Usecases werden funktionale und nichtfunktionale Anforderungen abgeleitet, die im praktischen Teil dieser Arbeit realisiert werden.

## 2.1. Usecases

Die hier beschriebenen Anwendungsszenarien beinhalten funktionale und nichtfunktionale Anforderungen eines Realsystems, das in weiterer Folge realisiert wird. Die Spezifikationen der Usecases wurden in Zusammenarbeit mit der Firma IVM im Auftrag der Firma NTE Systems erstellt.

### 2.1.1. Basisdefinitionen

Einleitend werden die Begriffe

- Anlagen-Monitoring,
- webbasiertes Monitoring und
- Informationsvisualisierung

in Kontext dieser Masterarbeit spezifiziert.

Anlagen im Sinne dieser Arbeit sind Immobilien die privat als auch industriell in Verwendung sind. Dementsprechend reicht die Zielgruppe dieser Anwendung von privaten Einzelpersonen bis hin zu institutionellen Körperschaften. Im weiteren Verlauf wird in diesem Zusammenhang von Benutzer/n bzw. Anwender/n gesprochen.

Der Begriff Anlagen-Monitoring umfasst in dieser Arbeit keine Video- oder Gebäudeüberwachung. Vielmehr handelt es sich hierbei um Anlagedaten, die über Sensoren aufgezeichnet werden. Sensordaten im Kontext dieser Arbeit können beispielsweise Durchflussmenge, Vorlauf- bzw. Rücklauftemperatur der Wärmepumpe aus dem Bereich der Energietechnik sein. Nachfolgende Abbildung verdeutlicht dies schematisch.

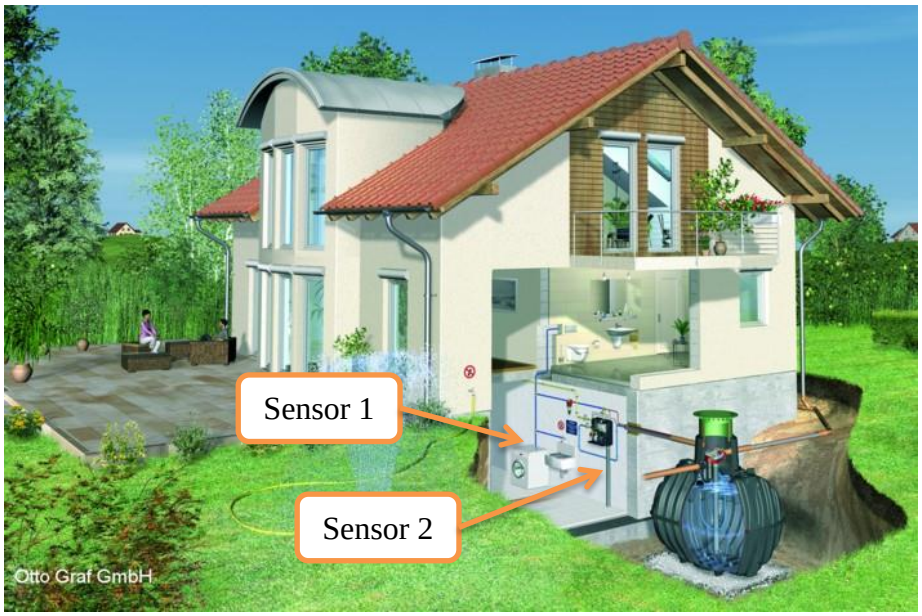


**Abbildung 1: Monitoring exemplarische Sensoren (Carat Paket Haus und Garten)**

Ausgehend von den Sensoren werden diese energietechnik-spezifischen Daten in einem zentralen Punkt gesammelt und gespeichert. Für die Auswertung der Daten bedarf es eines entsprechenden Clients[2], der die gesammelten Daten aufbereitet und für den Menschen in verständlicher Darstellungsform visualisiert. Beispielsweise ist aus „Abbildung 1: Monitoring exemplarische Sensoren“ ein Einfamilienhaus als Anlage erkennbar. Die beiden Sensoren „Sensor 1“ und „Sensor 2“ repräsentieren die Datenpunkte, welche an einer zentralen Stelle gesammelt und überwacht werden. Diese beiden Sensoren können in Intervallen von wenigen Sekunden abgefragt werden. Werden nun in einem XY-Diagramm, auf der X-Achse die Zeit (t) und auf der Y-Achse der dazugehörige Wert der Sensoren aufgetragen, würde dies eine historische Darstellung der aufgezeichneten Daten wiederspiegeln. Dies ist schematisch aus „Abbildung 2: XY-Diagramm“ ersichtlich:

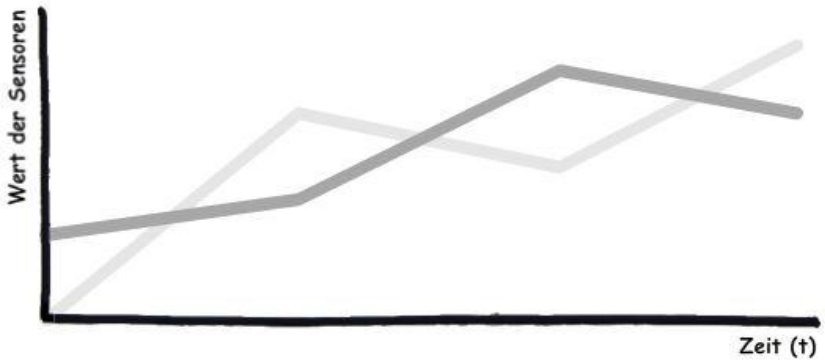


**Abbildung 2: XY-Diagramm**

[2] Weitere Informationen zu Client findet man unter: http://de.wikipedia.org/wiki/Client

Die Daten-Aggregation, Speicherung und Bereitstellung ist nicht Teil dieser Arbeit. Diese Arbeit beschränkt sich auf den Bereich der Aufbereitung und Visualisierung der Daten. Dies entspricht dem Client Bereich, also die Schnittelle zum Anwender, eines Monitoring Systems.

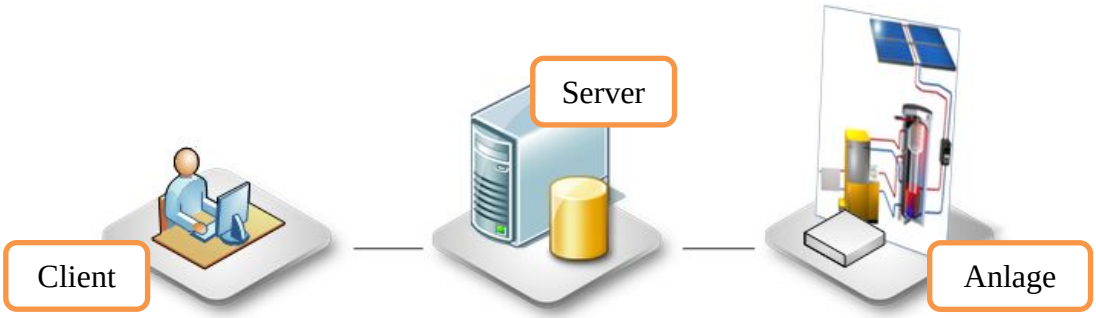


**Abbildung 3: einfache Form eines Monitoring Systems**

„Abbildung 3: einfache Form eines Monitoring Systems“ veranschaulicht die zuvor präsentierte Form des Monitoring Systems. Im linken Teil ist der Client-Bereich zu Aufbereitung der Daten ersichtlich, welcher mit dem mittleren Bereich (Server), dem zentralen Datenpunkt, kommuniziert. Der zentrale Bereich (Server) wiederum kommuniziert direkt mit den Anlagen, welche die Sensoren beinhalten. Eine direkte Verbindung vom Client-Bereich zum Anlagen-Bereich ist in dieser Masterarbeit nicht vorgesehen.

Im Gegensatz zum einfachen Monitoring wird das webbasierte Monitoring, wie bereits in Kapitel 1 erwähnt, durch die Komponente Internet erweitert. Nunmehr bedeutet dies, dass die Anlagen nicht mehr lokal gebunden sind oder sich im selben Netzwerk befinden müssen. Vielmehr können die Anlagen Systemgrenzen überschreitend verteilt und dennoch überwacht werden. Dies bringt eine Reihe an Herausforderungen mit sich, deren Lösung nicht Teil dieser Arbeit ist.

Teil dieser Arbeit ist es, dass die Endanwender der Auswertungssoftware verteilt im Internet auf die Informationen der Anlagen zugreifen können. Dies sollte ohne Installation einer Software erfolgen.

Unter Informationsvisualisierung im Kontext dieser Arbeit wird die grafische Repräsentation der aufgezeichneten Sensordaten verstanden. Den grafischen Formen der Darstellung sind dabei keine Grenzen gesetzt. Wie aus „Abbildung 4: Variante der Informationsvisualisierung“ ersichtlich ist, kann für die grafische Darstellung eines Temperaturwertes ein Thermometer, ein Drehknopf oder einfach nur ein Textfeld für die Repräsentation ausgewählt werden. Dies ist natürlich nur ein Bruchteil dessen, gemessen an den tatsächlichen Visualisierungsmöglichkeiten, die es heute für die diversesten Domänen gibt.

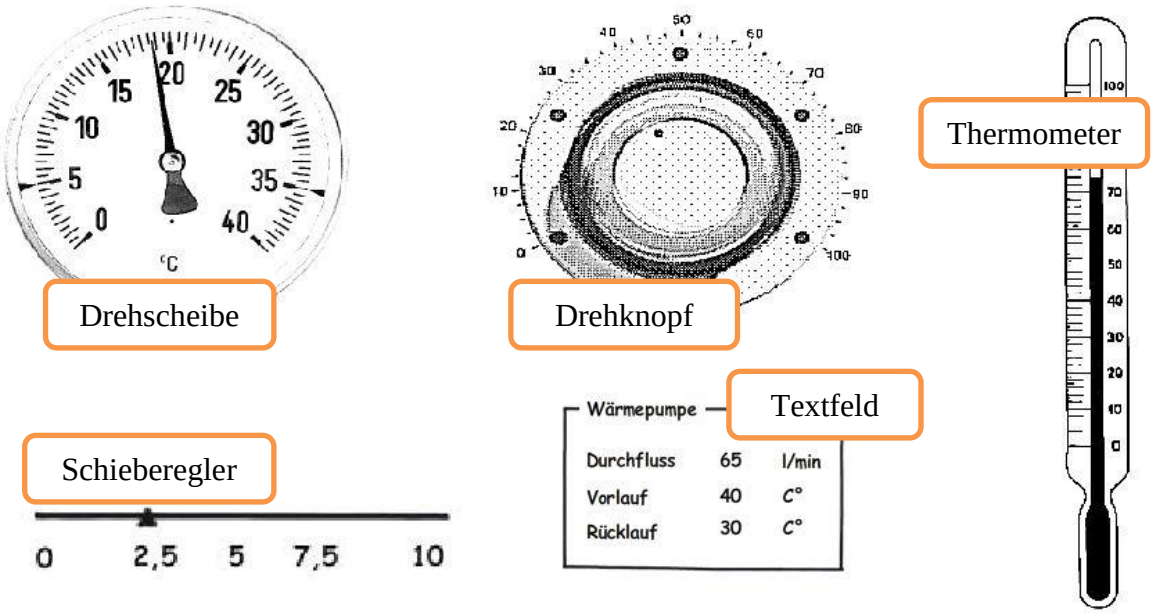


**Abbildung 4: Variante der Informationsvisualisierung**

Die „Abbildung 4: Variante der Informationsvisualisierung“ beinhaltet exemplarisch eine Auswahl der möglichen Visualisierungskomponenten, die in der Domäne Energietechnik allgegenwärtig sind.

Bis dato wurde der notwendige Kontext der Arbeit aufbereitet. Im weiteren Verlauf werden ein Systemüberblick und die einzelnen Usecases angeführt.

### 2.1.2. Systemüberblick

Diese Masterarbeit ist Teil eines Softwaresystems der Firma NTE Systems, lautend auf den Namen CLM[3] Plattform.

Die CLM Plattform ermöglicht es, Rohdaten von diversen Systemen zu lesen, zu schreiben, zu interpretieren und zu visualisieren. Eine schematische Darstellung der CLM Plattform ist aus „Abbildung 5: Die CLM Plattform“ ersichtlich.

[3] Control Loop and Monitoring Plattform, kurz CLM

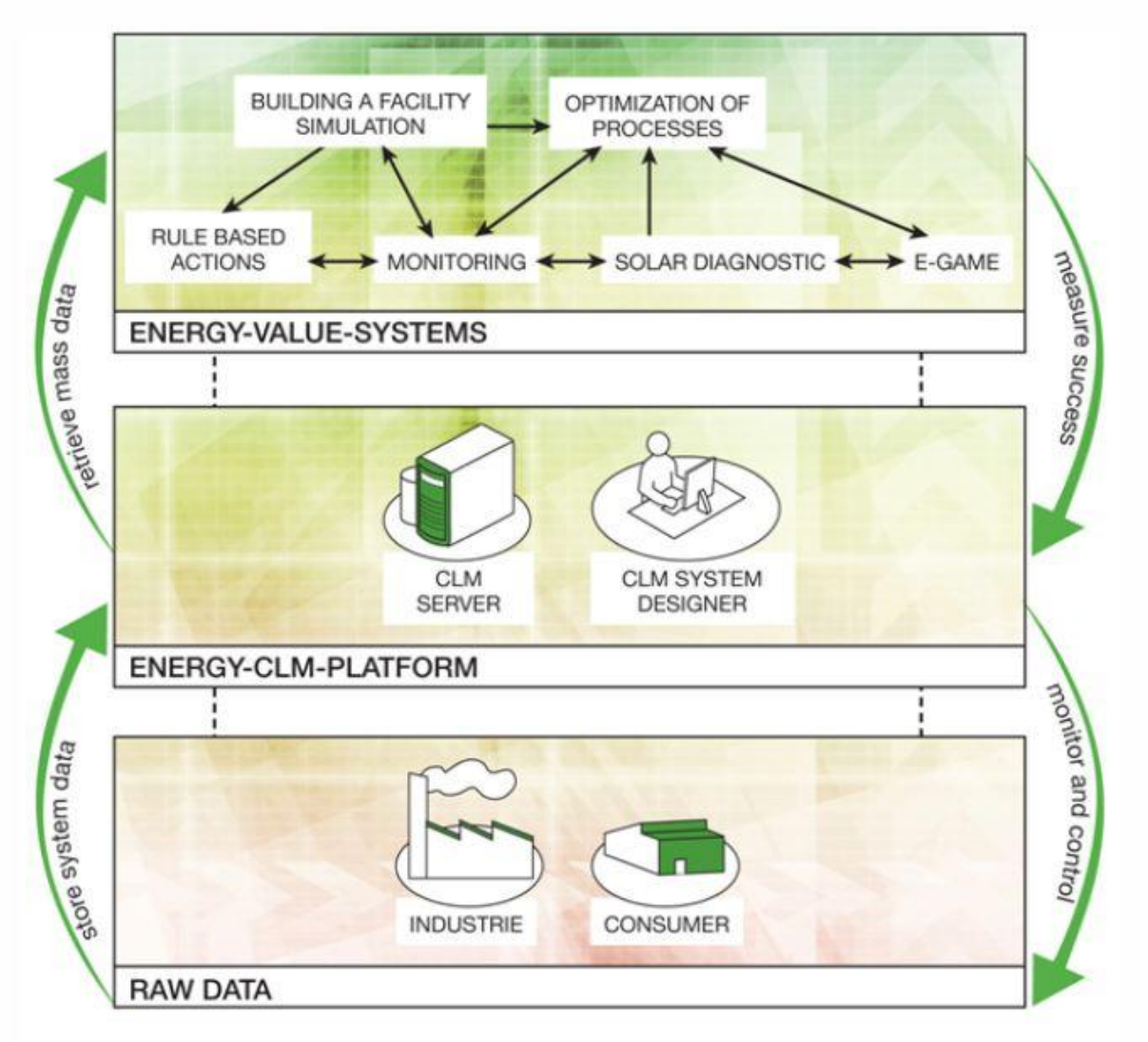


Abbildung 5: Die CLM Plattform

Die Rohdaten werden vorzugsweise von speicherprogrammierbaren Steuerungen (SPS) gelesen, welche sich in Industrie- bzw. Endkundengebäuden befinden. Die Rohdaten benötigen keine spezielle Domänenzugehörigkeit, sodass Daten aller Art überwacht werden können.

#### *2.1.2..1 CLM Server*

Der CLM Server ist die Kernkomponente der CLM Plattform. Diese Komponente ist einerseits zur Datenbeschaffung und –bereitstellung, andererseits für die Speicherung diverser Konfigurationen des Systems zuständig. Des Weiteren besitzt diese Komponente die Aufgabe, die Sicherheit (Rechte- und Gruppenmanagement) im System zu gewährleisten. Bei einem Webservice-Call werden nur Methoden und Daten verwendet, die der User-Berechtigung entsprechen. Dies bewirkt, dass sich der Desktop- und Web-Client nicht mit der Interpretation der User-Rechte befassen muss.

Die Kommunikation des CLM Servers zu den SPS erfolgt über diverse SPS-Hersteller-Protokolle. Des Weiteren unterstützt die Plattform bereits die Kommunikation mit dem OPC UA[4] Standard, die neueste Spezifikation der OPC[5] Foundation. Somit ist die CLM Plattform bestens für zukünftige Herausforderungen gerüstet.

#### *2.1.2..2 CLM System Designer*

Der CLM System Designer ist die Endanwender-Verwaltungseinheit der CLM Plattform. Hier hat der Benutzer die Möglichkeit, Anlagen zu verwalten, zu überwachen, Auswertungen zu erstellen und die Anlagen zu steuern; all dies mittels Drag and Drop.

Der CLM System Designer ist eine Desktop-Applikation, welche zuzüglich zu den Verwaltungsaufgaben auch einen Anlagendesigner besitzt, der das reale Anlagenschema mit aufgezeichneten Livewerten verknüpfen kann (Konrad, 2010). Zudem können zum Zeitpunkt dieser Masterarbeit Historiendaten konfiguriert und zu Diagrammen zusammengestellt werden. Zusätzlich zu den bisherig genannten Funktionalitäten ist im System Designer die Zuweisung der Rechte und Gruppen von einzelnen Anlagen und Usern implementiert.

All diese Konfigurationen und Einstellungen werden an den zuvor erwähnten CLM Server transferiert und zentral gespeichert.

#### *2.1.2..3 CLM WebControl*

Aufgabe der CLM WebControl Applikation ist es, die Stufe des „Energy Value Systems“ aus „Abbildung 5: Die CLM Plattform“ zu realisieren. Dem Benutzer der Plattform soll es ermöglicht werden, von überall auf der Welt auf seine Anlage zuzugreifen und deren Korrektheit zu überprüfen.

Schwerpunkt des praktischen Teils dieser Masterarbeit ist der Bereich „Monitoring“. Die Bereiche „Rule Based Actions“, „Building Facility Simulation“, „Solar Diagnostic“, „Optimization of Processes“ und „E-Game“ sind nicht Teil dieser Arbeit.

Zum Zeitpunkt dieser Arbeit sind die Konfigurationsmöglichkeiten der Anlagen aufgrund der Prioritäten des Auftraggebers im CLM System Designer integriert. Daher sind die Schwerpunkte des CLM WebControls die Interpretation der Konfigurationen, die Performance-Optimierung bei der Übertragung der Daten vom CLM Server bis hin zum CLM WebControl und die Erfüllung der nachfolgenden Usecases.

### 2.1.3. Auflistung Usecases

Im weiteren Verlauf werden die relevanten Usecases zum Zeitpunkt dieser Arbeit angeführt und erläutert.

---

[4]Weitere Informationen zum OPC UA Standard findet man unter: http://www.opcfoundation.org/Default.aspx/01_about/UA.asp?MID=AboutOPC

[5]Weitere Information zur OPC Foundation findet man unter: http://www.opcfoundation.org/Default.aspx

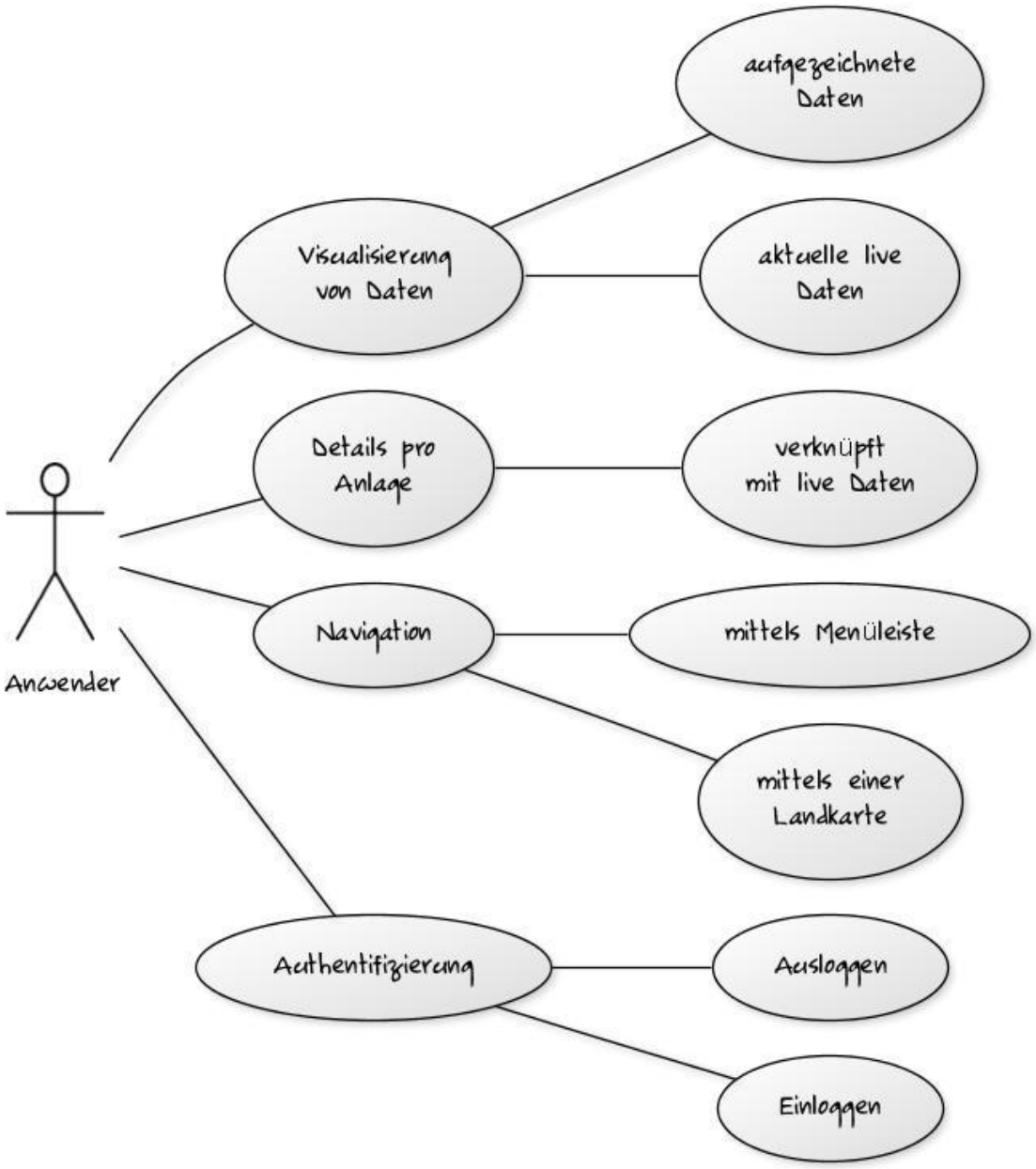


**Abbildung 6: Usecases Überblick**

Aus „Abbildung 6: Usecases Überblick“ sind vier wesentliche Usecases ersichtlich:

1. Der Benutzer möchte aufgezeichnete Daten visualisieren.
   Der Nutzer des webbasierten Monitoring Systems möchte die aufgezeichneten Daten visualisieren. Der Zeitraum, in dem Daten visualisiert werden, soll durch den Anwender frei konfigurierbar sein, jedoch sollen ihm unterstützend Short-Input Optionen zur Verfügung stehen. Beispielsweise könnte dies die Verschiebung eines Zeitfensters um einen Tag oder eine Stunde sein, wie es aus „Abbildung 7: eine mögliche Short-Input Leiste“ ersichtlich ist.


Abbildung 7: eine mögliche Short-Input Leiste

Des Weiteren möchte der Anwender Live-Werte der Anlage visualisiert bekommen. Dies sollte wenn möglich im domänenspezifischen Kontext erfolgen. Das würde bedeuten, dass der Anwender ein schematische Darstellung, wie z. B. aus „Abbildung 8: Schematische Darstellung einer Anlage aus dem Blickwinkel der Energietechnik“ ersichtlich, verwendet.

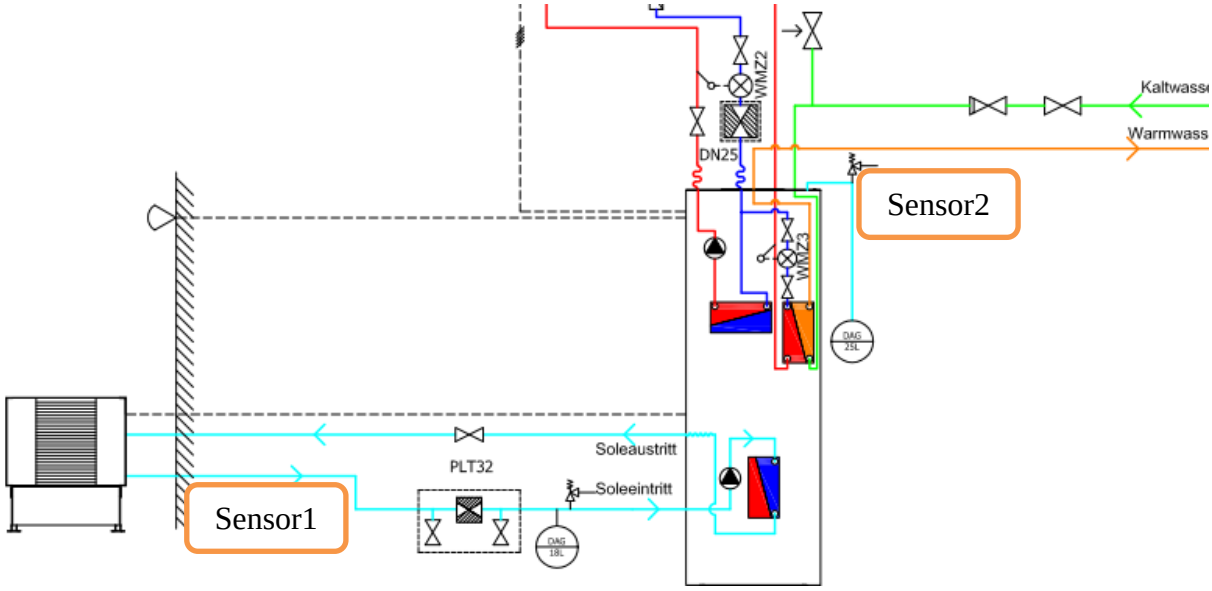


Abbildung 8: Schematische Darstellung einer Anlage aus dem Blickwinkel der Energietechnik

2. Der Benutzer möchte detailliert seine Anlagen betrachten.
   Zusätzlich zur Verknüpfung von Livedaten mit einer im Kontext spezifischen Visualisierung möchte der Anwender weiteren Einblick bekommen. Hierfür sollte es ihm möglich sein, die aufgezeichneten Anlagedaten auf seinen Rechner zu transferieren. Dies sollte in einem Dateiformat geschehen, das mit gängigen Systemen wiederverwendet werden kann.

3. Der Benutzer möchte zwischen seinen Anlagen navigieren.
   Der Endanwender möchte zwischen seine Anlagen in diversen Optionen navigieren. Zum einen möchte er dies über eine altbewährte Navigationsleiste (De Chiara & Fish, 2007), zum anderen über eine geografische Landkarte erreichen. Die „Abbildung 9: Mock-up einer geografischen Navigation“ verdeutlicht die geografische Navigation in Form eines Mock-ups.

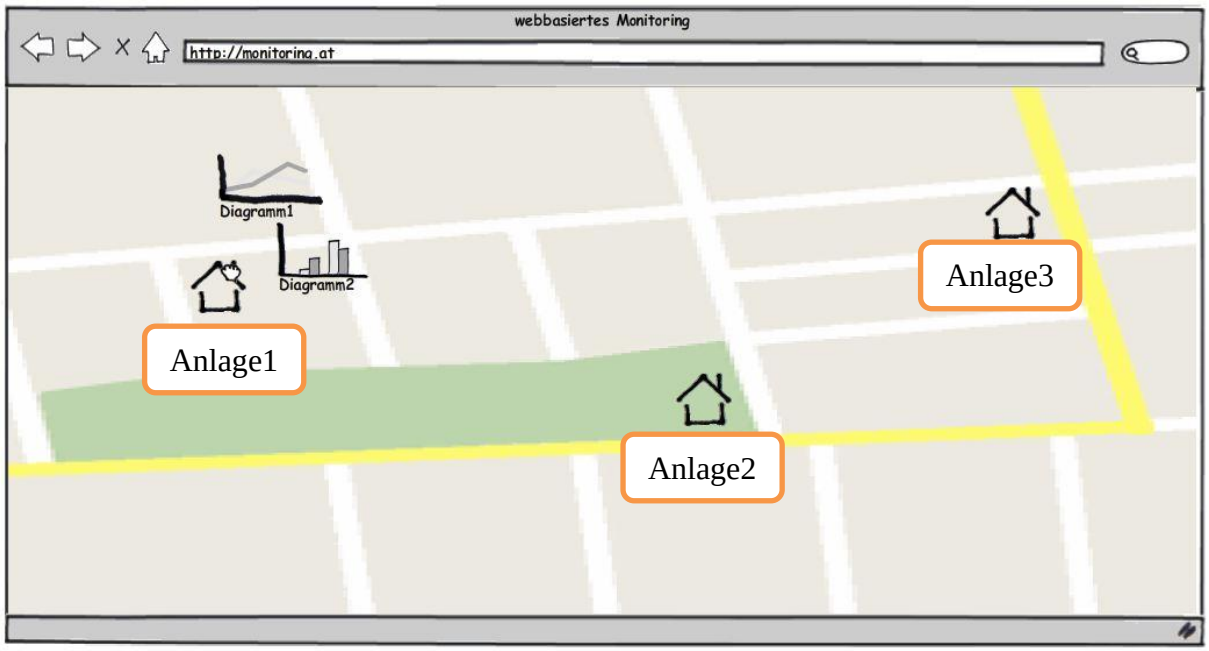


Abbildung 9: Mock-up einer geografischen Navigation

4. Der Benutzer möchte seine Anlageinformationen mit einer Authentifizierung absichern.
   Der Anwender möchte den unerlaubten Zugriff auf die Informationen seiner Anlagen durch eine Authentifizierung absichern. Es muss verhindert werden, dass der Anwender des Monitoring Systems Zugriff auf fremde Anlagedaten bekommt. Des Weiteren dürfen die Informationen nicht im Klartext (Plain text[6]) über das Internet versendet werden, jegliche Kommunikation muss über eine gesicherte Verbindung erfolgen.

## 2.2. Anforderungen

Aus der zuvor angeführten Auflistung von Usecases lassen sich funktionale und nichtfunktionale Anforderungen (im weiteren Verlauf „Requirements“ genannt) ableiten.

### 2.2.1. Funktionale Anforderungen

Folgend werden die funktionalen Anforderungen angeführt. Diese Punkte sind nach Priorität sortiert: Anforderungen mit hoher Priorität sind am Anfang angeführt, Anforderungen mit niedriger Priorität am Ende der Aufzählung.

1. Die Applikation muss durch eine Authentifizierung geschützt sein.
2. Die Applikation soll eine Authentifizierungsmöglichkeit besitzen.
3. Die Applikation soll die aufgezeichneten Daten im zeitlichen Kontext visualisieren.

[6] Weitere Informationen zu Plain text findet man unter: http://de.wikipedia.org/wiki/Plain_text

4. Verlaufsdiagramme sollen interaktiv sein.
5. Die Interaktion von Verlaufsdiagrammen soll den zeitlichen Visualisierungsbereich umfassen.
6. Der Inhalt der Verlaufsdiagramme soll in Form einer Tabelle für erste Sortier- und Filterfunktionalitäten dargestellt werden.
7. Die Applikation soll eine detaillierte Ansicht pro Anlage ermöglichen.
8. Die Applikation soll die aufgezeichneten Daten im energetischen Kontext visualisieren.
9. Die detaillierte Ansicht soll Livewerte der Anlage darstellen.
10. Die detaillierte Ansicht soll frei konfigurierbar sein.
11. Die Applikation soll eine Navigation zwischen den Anlagen ermöglichen.
12. Die Navigation soll auch einzelne Verlaufsdiagramme einer Anlage beinhalten.
13. Der Einstiegspunkt der Applikation soll eine Übersicht der eigenen Anlagen bieten.
14. Die Applikation soll Informationen der Anlage im geografischen Kontext visualisieren.
15. Die Verlaufsdiagramme sollen exportierbar sein.
16. Die gesamten aufgezeichneten Daten sollen exportierbar sein.
17. Verlaufsdiagramme sollen die Möglichkeit besitzen, mehr als eine Y-Achse darzustellen.
18. Es soll die Möglichkeit bestehen, den Achsen Titel zu vergeben.
19. Verlaufsdiagramme sollen als Bar- bzw. Linienchart darstellbar sein.
20. Das Aussehen der Verlaufsdiagramme soll individuell einstellbar sein.

### 2.2.2. Nichtfunktionale Anforderungen

Zusätzlich zu den funktionalen Anforderungen gibt es noch nichtfunktionale Anforderungen, diese wurden teilweise vom Auftraggeber angefordert bzw. haben sich aus den Usecases als logische Schlussfolgerung ergeben. Diese Punkte sind nach Priorität sortiert: Anforderungen mit hoher Priorität sind am Anfang angeführt, Anforderungen mit niedriger Priorität am Ende der Aufzählung.

1. Die Applikation muss als RIA (Rich Internet Application) realisiert werden.
2. Die RIA-Applikation muss erweiterbar sein.
3. Die Applikation soll modular aufgebaut sein.
4. Die Verstrickung von Modulen soll auf ein Minimum reduziert werden.
5. Die Applikation soll erweiterbar sein.
6. Es müssen Webservices zum Datenaustausch verwendet werden.
7. Es muss ein verschlüsselter Datenaustausch zwischen Server- und Client-Applikation gewährleistet werden.
8. Der verschlüsselte Datenaustausch muss per SSL erfolgen.
9. Es muss ein Rechte- und Gruppenmanagement geben.
10. Die Qualität der Anwendung muss mit System- und Regressionstests belegt werden.
11. Kernfeatures müssen mittels Unit-Tests geprüft werden.

12. Der Austausch von Konfigurationen zwischen Desktop Client und Web Client muss mittels XML Format erfolgen.
13. Die Applikation soll bei auftretenden Fehlern eine gut definierte Fehlerbehandlung besitzen.
14. Die Applikation soll Fehler protokollieren, damit sie für die Entwicklung nachvollziehbar sind.
15. Die RIA Anwendung muss ein ansprechendes „Look and Feel“ besitzen.
16. Die RIA Anwendung muss mit minimalem Aufwand ein neues Corporate Identity Theme[7] nachziehen können.
17. Der Datenexport muss im CSV Dateiformat erfolgen.
18. Der Datenexport muss eine Fortschrittsanzeige beinhalten.
19. Der Datenexport muss unterbrechbar sein.
20. Der Code muss dokumentiert sein.

[7] Weitere Informationen zum Thema Theme findet man unter: http://de.wikipedia.org/wiki/Skin_%28Computer%29

# 3. Softwareentwicklungsprozess

Dieses Themengebiet wird in der Masterarbeit angeführt, da Scrum als Softwareentwicklungsprozess im praktischen Teil dieser Arbeit eingesetzt wurde. Ohne einen Softwareentwicklungsprozess ist die sinnvolle Koordination eines Entwicklungsteams in der Größe von sechs Softwareentwicklern schwierig. Dies ist die gesamte Anzahl der Entwickler die an der CLM[8] Plattform der Firma NTE Systems arbeiten. Im weiteren Verlauf wird eine Auswahl von Softwareentwicklungsprozessen und deren Kernelemente präsentiert. Daraufhin wird eine Auswahl getroffen, welcher Prozess den projektspezifischen Anforderungen und den sich schnell ändernden Kundenwüschen am besten gewachsen ist.

Es gibt in der heutigen Zeit eine Unmenge an Softwareentwicklungsprozessen. Daher werden in der nachfolgenden Auflistung und Evaluierung das V-Modell XT als Vertreter der klassischen Softwareentwicklungsprozesse und Scrum als Vertreter der agilen Softwareentwicklungsprozesse angeführt.

Das V-Modell XT wurde ausgewählt, da viele große Firmen[9] für die tägliche Arbeit diesen Softwareentwicklungsprozess einsetzen.

> „*Der Vorstand von WEIT e.V. erwartet, dass sich das V-Modell® XT in Deutschland weiter durchsetzt und sich zudem über Deutschlands Grenzen hinaus europaweit etabliert.“* (Informationsdienst Wissenschaft , 2008)

Der Scrum-Prozess wurde ausgewählt, da einschlägige Fachlektüre den Eindruck erweckt, dass dieser Prozess in der agilen[10] Softwareentwicklung laut (Scherer, Beim Rugby abgeschaut, 2008) und (West, Grant, Gerush, & D´Silva, 2010) „State-of-the-Art“ ist.

> *„Als verbreitetste agile Methode geht Scrum aus der Studie hervor, das nahezu 11 Prozent der Befragten verwenden.“ (Developer, 2010)*

Eine vollständige Auflistung und Evaluierung aller Softwareentwicklungsprozesse ist nicht das Kernthema dieser Masterarbeit.

---

[8] Control Loop and Monitoring, kurz CLM. Dies ist ein Produkt der Firma NTE Systems.

[9] „*Am 31. März 2008 gründeten Vertreter der 4Soft GmbH, von EADS Deutschland GmbH, des Fraunhofer IESE, der Industrieanlagen Betriebsgesellschaft (IABG) mbH, der Siemens AG, der Technischen Universität Clausthal sowie der Technischen Universität München in Potsdam den WEIT e.V. Unmittelbar nach Vereinsgründung erklärte die Bundesrepublik Deutschland, vertreten durch das Bundesverwaltungsamt (BVA), ihren Beitritt. WEIT e.V. koordiniert die Weiterentwicklung und Pflege des V-Modell® XT sowie die Unterstützung der qualifizierten Anwendung durch das V-Modell®-XT-Zertifizierungsprogramm.*“ (Informationsdienst Wissenschaft , 2008)

[10] Unter Agiler Softwareentwicklung werden die Prozesse zusammengefasst, die als flink, beweglich Bezeichnet werden können. Beispielsweise gehören folgende Prozesse auch zur Gruppe der Agilen Softwareentwicklung: Extreme Programming(XP), Feature Driven Development(FDD), Text Driven Development(TDD). Weitere Informationen hierzu findet man unter: http://de.wikipedia.org/wiki/Agile_Softwareentwicklung

# 3.1. V-Modell XT

Das nachfolgende Kapitel verschafft einen kurzen Überblick über das V-Modell XT. Dieser Abschnitt beruht auf den Dokumenten (Höhn, Rausch, & Höppner, Das V-Modell XT: Grundlagen, Methodik und Anwendungen, 2008) und (Friedrich, Hammerschall, Kuhmann, & Sihling, 2009).

Laut (Höhn, Rausch, & Höppner, Das V-Modell XT: Grundlagen, Methodik und Anwendungen, 2008) ist das V-Modell XT ein Referenzmodell für den Prozess eines Softwareentwicklungsprojektes, ein Katalog zur Konfiguration projektspezifischer Vorgehensmodelle, ein Vorgehensmodell für Vorgehensmodelle.

Die Kernkonzepte des V-Modell XT sind gemäß (Friedrich, Hammerschall, Kuhmann, & Sihling, 2009) zum einen die Beschreibung der Abläufe im Verlauf eines Entwicklungsprojektes über Produkte, Rollen und Aktivitäten, zum anderen die Vorgehensbausteine, die eine Modularisierung der Abläufe und eine flexible Zusammenstellung erlauben.

## 3.1.1. Produkte, Rollen, Aktivitäten

Ein Produkt im Sinne des V-Modell XT ist ein Ergebnis in einem Projekt. Diese können sich im Laufe des Projektes ergeben, oder sie existieren bereits und stehen dem Projekt als externe Produkte zur Verfügung. Produkte sind in den meisten Fällen diverse Dokumente, beispielsweise das Pflichtenheft, oder sie existieren in Form von Systemelementen, beispielsweise eine Softwarekomponente.

Die beteiligten Personen, deren Fähigkeiten und Verantwortlichkeiten in einem Projekt werden als Rollen beschrieben. Eine Rolle kann im V-Modell XT von mehreren Personen eingenommen werden. Beispielsweise vermag der Projektleiter auch die Rolle eines Entwicklers einzunehmen. Eine der wenigen Regeln bei den Rollen besagt, dass der Projektleiter nicht gleichzeitig auch für die Qualitätssicherung zuständig sein kann.

Die Vorgehensweise, wie ein Produkt zu produzieren ist, wird als Aktivität bezeichnet. Jedes Produkt wird durch eine einzige Aktivität beschrieben. Externe Produkte besitzen keine Aktivität, da sie bereits existieren.

## 3.1.2. Vorgehensbausteine

Da es eine Unmenge an Produkten und Aktivitäten im V-Modell XT geben kann, werden einzelne Produkte und Aktivitäten thematisch organisiert. Diese thematische Organisation kann auch als Modul bezeichnet werden. Einzelne Module wiederum werden als Vorgehensbausteine angesehen. Kernbausteine des Modells sind die Module Projektmanagement, Qualitätssicherung, Konfigurationsmanagement sowie das Problem- und Änderungsmanagement.

Der Ablauf der Projektsteuerung und Projektplanung wird im Vorgehensbaustein Projektmanagement behandelt. Die Prozesse der Qualitätssicherung werden im Vorgehensbaustein Qualitätssicherung angeführt. Die Erstellung und Pflege der Produktbibliothek wird im Vorgehensbaustein Konfigurationsmanagement erklärt. Der Vorgehensbaustein Problem- und Änderungsmanagement umfasst die Verwaltung von Problem- und Änderungsanträgen.

### 3.1.3. Durchführungsrahmen

Der V-Modell XT Prozess sieht es vor, dass es diverse Durchführungsstrategien gibt. In den Durchführungsstrategien werden Entscheidungspunkte in einer bestimmten Reihenfolge festgelegt. Das V-Modell XT gibt folgenden grundlegenden Durchführungsrahmen vor:

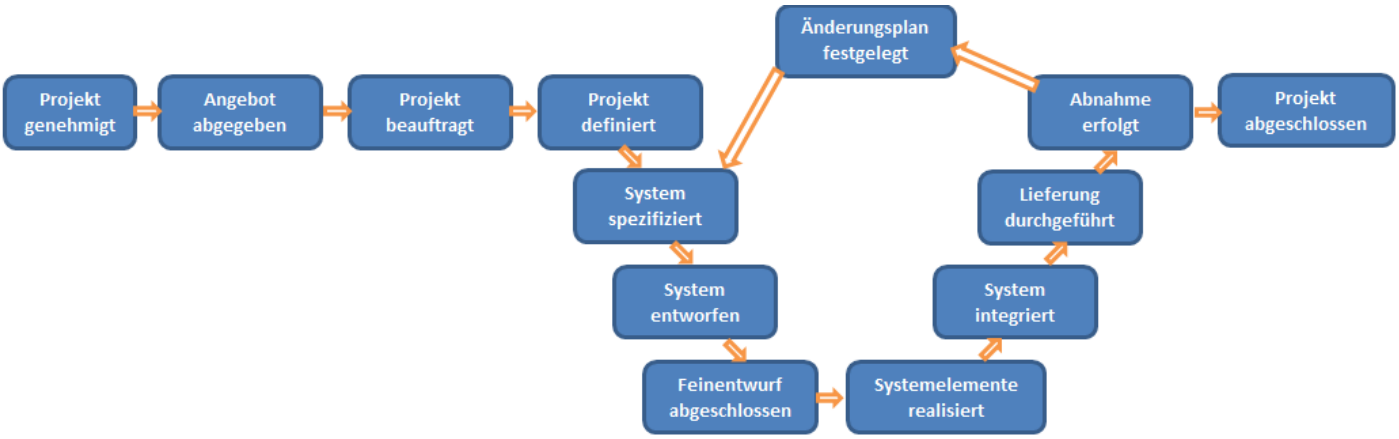


Abbildung 10: Entscheidungspunkte des V-Modells XT

Die Entscheidungspunkte „Projekt genehmigt“, „Angebot abgegeben“, „Projekt beauftragt“, „Projekt definiert“, „Änderungsplan festgelegt“, „Abnahme erfolgt“ und „Projekt abgeschlossen“, welche aus „Abbildung 10: Entscheidungspunkte des V-Modells XT“ ersichtlich sind, dienen zur Kommunikation mit dem Auftraggeber.

Die Entscheidungspunkte „System spezifiziert“, „System entworfen“, „Feinentwurf abgeschlossen“, „Systemelemente realisiert“, „System integriert“ und „Lieferung durchgeführt“ sind die eigentlichen Entscheidungspunkte der Systementwicklung.

Einstiegspunkt des V-Modell XT ist es, wenn auf Seiten des Auftraggebers ein Projekt genehmigt wurde. Aufbauend darauf wird ein möglicher Auftragnehmer ein Angebot für das Projekt vorlegen. Sofern der Auftragnehmer den Zuschlag erhält, wird er mit dem Projekt beauftragt und es beginnt die Phase der Projektdefinition. Nachfolgend wird das zu realisierende System spezifiziert und in einem Entwurf realisiert. Darauf aufbauend werden die Systemelemente realisiert, ins Gesamtsystem integriert und am Ende dem Auftraggeber ausgeliefert. Sofern dieser das System akzeptiert und keine weiteren Änderungen gewünscht werden, ist das Projekt abgeschlossen. Sofern Änderungen notwendig sind, werden diese im Änderungsplan festgehalten und iterativ wird wiederum bei der Spezifikation des Systems begonnen.

### 3.1.4. Einsatzgebiet

Einsatz findet das V-Modell XT vor allem in der Planung und Durchführung von deutschen IT-Projekten im Regierungsauftrag. Die Verwendung des V-Modell XT ist für öffentliche Aufträge in Deutschland zwingend vorgeschrieben. (IABG), (IT-Beauftragten, 2009) und (Höhn & Höppner, Das V-Modell XT, Anwendungen, Werkzeug, Standards, 2008, S. 3)

## 3.2. Scrum

Dieser Abschnitt soll einen Überblick darüber gewähren, was Scrum ist und welche Gegebenheiten den Scrum-Prozess charakterisieren.

> *„Ein Team zieht aus, eine Mission zu erfüllen. Ein Team aus Spezialisten, von einer Vision geleitet, arbeitet diszipliniert und hoch professionell, von einem Teilprodukt zum nächsten, bis es das Endprodukt abliefern kann. Scrum ist eine Grundüberzeugung, eine Philosophie und eine Arbeitsweise mit klar definierten Rollen, einem sehr einfachen Prozessmodell und einem klaren und einfachen Regelwerk, das es diesem Missionsteam ermöglicht, seine Ziele hocheffektiv zu erreichen.“ (Gloger, Scrum - Produkte zuverlässig und schnell entwickeln, 2009, S. 8)*

> *„Scrum ist ein Framework für das Management komplexer Projekte. Komplexe Projekte sind dadurch charakterisiert, dass nicht exakt vorhersehbar ist, welchen Verlauf das Projekt nehmen wird und was in Zukunft alles passiert.“ (Wirdemann, 2009, S. 26)*

Scrum gehört zur Klasse der agilen Softwareentwicklungsprozesse.

Das Prozessmodell von Scrum beschreibt den Rahmen, in dem sich alle Aktivitäten für die Produktentwicklung wiederfinden. In diesem Modell gibt es sechs Rollen, sechs Meetings und neun Artefakte. Der nachfolgende Abschnitt beruht auf dem Dokument (Wirdemann, 2009).

Im Scrum-Prozess werden die Anforderungen eines Projektes zentral im Product Backlog verwaltet. Das Product Backlog entspricht dem Anforderungskatalog. Die Anforderungen im Product Backlog werden als User Stories bezeichnet, diese beschreiben aus dem Blickwinkel des Anwenders eine Funktionalität. Idealerweise werden diese wie folgt aufgebaut: Wer, will was, warum.

Das Scrum Team realisiert die User Stories aus dem Product Backlog. Die Entwicklungsperiode, auch Sprint genannt, hat eine Länge von ein bis vier Wochen. Das Team setzt selbständig einen Teil der User Stories des Product Backlog in einen Sprint um.

Die „Abbildung 11: Die vier Phasen von Scrum“ beschreibt den iterativen Scrum-Prozess.

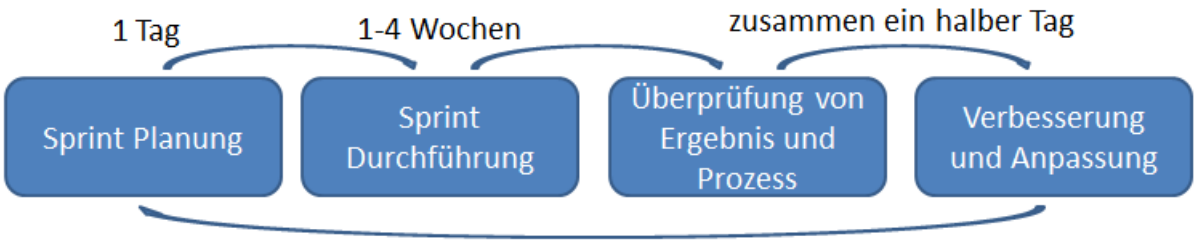


**Abbildung 11: Die vier Phasen von Scrum**

Die Sprint-Planung dauert in der Regel einen Tag. Diese Phase wird zur Planung des Sprints verwendet. Es werden jene User Stories aus dem Product Backlog ausgewählt, die realistisch im Sprint umgesetzt werden können. Die Entwicklungsperiode, Sprint-Durchführung, umfasst

einen Zeitraum von ein bis vier Wochen und wird von den jeweiligen Teams festgelegt. Dieser Zeitraum wird zur Realisierung der User Stories benötigt. Am Ende werden die realisierten User Stories dem Kunden präsentiert und von diesem überprüft. Das Team evaluiert am Ende den Prozess. Mögliche Verbesserungsvorschläge werden im nächsten Sprint berücksichtigt. Dieser iterative Prozess wird die gesamte Laufzeit des Projektes beibehalten.

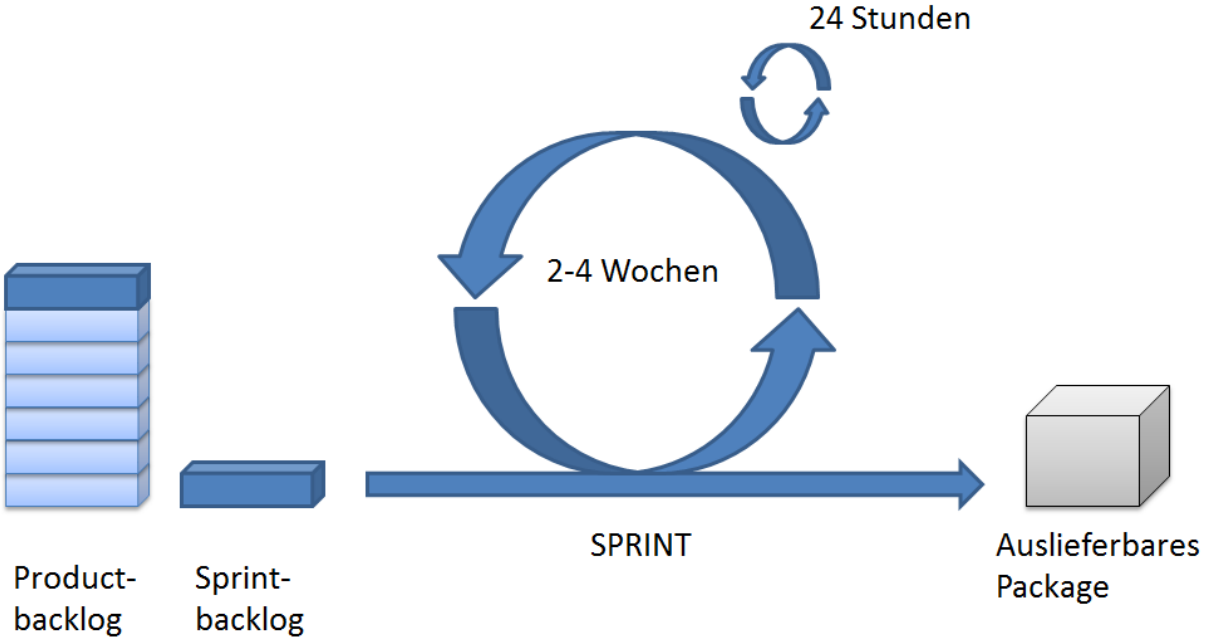


Abbildung 12: Scrum Entwicklungsprozess Schematische Darstellung

### 3.2.1. Rollen

Dieser Abschnitt beruht auf dem Dokument (Gloger, Scrum - Produkte zuverlässig und schnell entwickeln, 2009, S. 12-15).

Der Scrum-Prozess beinhaltet folgende Rollen:

- Die Kernaufgabe des **Product Owner** ist es, die Produktentwicklung zu planen und zu lenken. Des Weiteren ist es die Aufgabe des Produkt Owners, die Kundenwünsche in User Stories zu formulieren und diese dem Product Backlog hinzuzufügen. Der Product Owner ist des Weiteren für die Priorisierung des Product Backlog und den Release-Plan verantwortlich. Er hat auch dafür Sorge zu tragen, dass der finanzielle Aufwand dem Projektergebnis entspricht.

- Das **Team** ist für die Realisierung des Produktes verantwortlich. Es beinhaltet alle notwendigen Komponenten, die für die Realisierung eines Projektes notwendig sind. Dies können Spezialwissen der Teammitglieder oder notwendige Kompetenzen für die Realisierung sein. Des Weiteren regelt das Team seine Aufgaben selbst, jedoch erfolgt dies unter Anwendung der Standards und Prozesse, die vom Unternehmen vorgegeben werden. Die Arbeitsmenge, das Aufgabenvolumen, eines Teams wird nicht von außen

bestimmt. Das Team bestimmt dies selbst und trägt die Verantwortung für die Qualität des Ergebnisses.

- Der **Scrum Master** ist kein Teammitglied, somit auch nicht weisungsbefugt. Er ist als Bindeglied zwischen Product Owner und Team anzusehen. Seine Aufgabe ist es, das Team zu unterstützen, sodass dieses die Ziele erreichen kann. Er ist dafür verantwortlich, Herausforderungen, Schwierigkeiten, Probleme und Blockaden, die das Team beim Arbeiten behindern, zu beseitigen. Dazu gehört auch, dass er die gesamten am Projekt beteiligten Personen schult und ihnen erläutert, welchen Aufgaben sie in den einzelnen Scrum-Rollen zugeteilt sind. Die einzelnen Personen müssen diese verstehen, sodass die Aufgaben korrekt ausgeführt werden können.

- Der **Kunde** ist der Financier des Projektes.

- Der **Anwender** ist der Benutzer des Projektergebnisses. Er ist somit eine wichtige Informationsquelle für das Scrum Team. Er definiert auch mit dem Product Owner den Funktionsumfang des Produktes. Gleichzeitig ist er der Ansprechpartner für das Team, um das Produkt bedienbarer zu machen.

- Der **Manager** ist der Leiter der Unternehmung, in der das Entwicklerteam, der Scrum Master und der Product Owner beschäftigt sind. Er stellt finanzielle Mittel, Ressourcen und Richtlinien innerhalb des Unternehmens bereit.

### 3.2.2. Meetings

Dieser Abschnitt beruht auf dem Dokument (Gloger, Scrum - Produkte zuverlässig und schnell entwickeln, 2009, S. 12-15).

Im Scrum-Prozess gibt es idealerweise folgende Meetings:

- **Sprint Planning Meeting 1**: In diesem Meeting werden dem Team vom Product Owner die Backlog Items für den geplanten Sprint präsentiert. Idealerweise ist bei diesem Meeting auch der Anwender anwesend und erklärt dem Team offene Fragen. Die genaue Anzahl der Backlog Items für den neuen Sprint werden vom Team bestimmt bzw. akzeptiert und in das Selected Product Backlog aufgenommen.

- **Sprint Planning Meeting 2**: In diesem Meeting werden die ausgewählten Items aus dem Selected Product Backlog durch das Team geplant und in Tasks zerlegt. Bei diesem Meeting sind der Anwender und der Product Owner nicht notwendigerweise anwesend. Ein Task definiert eine Aufgabe, die zur Lösung eines Product Backlog Items notwendig ist. Somit ist für die Entwickler ersichtlich, wie viel Aufwand zur Lösung dieses Items erforderlich ist. Alle Backlog Items und die dazugehörigen Tasks werden in das Sprint Backlog aufgenommen.

- **Daily Scrum**: Täglich trifft sich das Team zu einem vereinbarten Zeitpunkt, um ein Meeting abzuhalten, das durch den Scrum Master moderiert wird. Dabei wird die Arbeitszeit vom Vortag auf einzelne Tasks verbucht und dem Team kurz der Status der einzelnen Tasks mitgeteilt. Des Weiteren werden für den neuen Tag Aufgaben (Tasks) ausgewählt, die man sich vornimmt zu lösen. Sofern ein Teammitglied Probleme mit einem Task hat, dient das Meeting zum Informationsaustausch mit den anderen Mitgliedern, sodass Probleme schnell erkannt und frühzeitig darauf reagiert werden kann. Sonstige Blockaden oder Probleme müssen dem Scrum Master mitgeteilt werden, damit sie so schnell wie möglich in Angriff genommen werden können.

- **Estimation Meeting**: Sollten sich im Laufe eines Sprints die Rahmenbedingungen durch zusätzliche Informationen verändern, besteht die Möglichkeit, im Rahmen des Estimation Meetings User Stories neu zu bewerten, oder aber das Sprint Backlog neu zu priorisieren.

- **Sprint Review**: Dieses Meeting dient zur Präsentation der erarbeiteten Funktionalitäten. Diese müssen voll funktionsfähig und getestet sein, sodass sie produktiv eingesetzt werden können. Instabile Funktionalitäten und/oder nicht getestete Funktionalitäten werden nicht als abgeschlossen angesehen. Diese müssen somit in den neuen Sprint übernommen werden.

- **Sprint Retrospektive**: Dieses Meeting dient dazu, den Prozess und den vergangenen Sprint zu analysieren und zu verbessern, damit das Team im neuem Sprint ohne Hindernisse arbeiten kann.

### 3.2.3. Artefakte

> *„Artefakte sind die Ergebnisse von Aktivitäten im Softwareentwicklungsprozess. In der agilen Softwareentwicklung gilt auslieferbarer Code als das einzige Artefakt, auf das nicht verzichtet werden kann.“ (Scherer, Scrum Artefakte)*

Dieser Abschnitt beruht auf dem Dokument (Gloger, Scrum - Produkte zuverlässig und schnell entwickeln, 2009, S. 12-15).

Der Scrum-Prozess kennt folgende Artefakte:

- Eine **Vision** beinhaltet eine klare Vorstellung über die Realisierung des Produkts. Es ist Aufgabe des Product Owners die Visionen für das Team zu sammeln und zu definieren.

- **Product Backlog Item**: Dies ist eine Funktionalität, die im auszuliefernden Produkt enthalten sein muss. Product Backlog Items werden im Product Backlog gesammelt.

- **Product Backlog**: Es beinhaltet eine Sammlung von Product Backlog Items, welche die Funktionalitäten des Produktes definieren. Innerhalb des Product Backlog werden die Items vom Product Owner priorisiert.

- **Sprint Goal**: Dies ist eine Ansammlung von Product Backlog Items, die in einem Sprint realisiert werden sollten.

- **Selected Product Backlog**: Dieses orientiert sich am Sprint Goal. Es werden aber nur jene Product Backlog Items übernommen, auf welche sich das Team auch einigt. Alleinig das Team entscheidet, wie viele Items vom Sprint Goal in das Selected Product Backlog übernommen werden.

- Als **Aufgaben,** auch Tasks genannt, wird all jenes bezeichnet, das zur Erreichung des Ziels notwendig ist.

- **Sprint Backlog**: Dies ist die Liste der Product Backlog Items, die im Sprint Goal definiert werden. Diese Liste wird im Sprint Planning Meeting 2 erarbeitet.

- **Release Plan**: Dieser Plan gibt einen Überblick darüber, in welchem Sprint welche Backlog Items geliefert werden können. Dies ist ein Mittel, um dem Team anzuzeigen, was noch erledigt werden muss. Es ist somit ein reines Informationsinstrument.

- **Impediment Backlog**: Das ist die Liste an Problemen, die vom Scrum Master gelöst werden müssen.

- **Product Increment**: Dies ist ein Teil des Gesamtproduktes, welches für die Auslieferung an den Kunden bereit ist.

## 3.3. Scrum vs. V-Modell XT

Man kann sagen, dass das V-Modell XT der Standard öffentlicher IT Projekte in Deutschland ist. Dieses Modell hat einen hohen Organisationsaufwand und benötigt enorm viele Ressourcen für die Erstellung diverser Dokumente, die aufgrund dieses Prozesses notwendig sind. Für ein kleines Softwareentwicklungsunternehmen mit einem neuen Produkt kann diese Überregulierung in der Vorgehensweise zum Scheitern der Unternehmung führen. Der Mehraufwand des Prozesses verhindert es, dass man sich nicht auf die wesentlichen Punkte des Projektes, die Entwicklung, fokussieren kann.

Nachteilig für den Scrum-Prozess ist die iterative Vorgehensweise bei der Berechnung der Kosten für Festpreis-Projekte. Der Einsatz von Scrum bei diesen speziellen Projekten benötigt einen großen Sicherheitspuffer und erzielt andererseits eine höhere Gewinnmarge, die gegebenenfalls zeitliche Verzögerungen abfedern muss. Dieser Umstand führt zu teureren Angeboten. Als Dienstleistungsunternehmen wird man durch den Einsatz von Scrum mehr Pufferzeiten einrechnen als notwendig sind, damit man keine Strafzahlungen tätigen muss. Diese

Strafzahlungen sind meist vertraglich fixiert und würden bei Nicht-Einhaltung des Liefertermins fällig werden. Wird hingegen eine nutzenbasierte Preisgestaltung (Value-based pricing[11]) verwendet, werden die Kosten auf Seiten des Auftraggebers minimiert und zusätzlich bekommt er wirklich das, was er braucht. (Rüssel, 2009) und (Gloger, Nutzen-basierende Preisgestaltung: Zwei Vorschläge, 2010)

| | **Scrum** | **V-Modell XT** |
|---|---|---|
| **Vorgehensweise** | iterativ | vorhersehbar |
| **Budget** | gering | hoch |
| **Dokumentation** | gering | unabdingbar |
| **Komplexität** | gering | hoch |

**Tabelle 1: Gegenüberstellung Scrum vs. V-Modell XT**

Die „Tabelle 1: Gegenüberstellung Scrum vs. “ stellt zusammenfassend die einzelnen Kategorien Vorgehensweise, Budget, Dokumentation und Komplexität gegenüber.

- **Vorgehensweise**: Dieser Punkt beschreibt den Verlauf des Softwareentwicklungsprozesses.
- **Budget**: Dieser Punkt beschreibt das notwendige Kapital, das benötigt wird, um den Prozess zu ermöglichen.
- **Dokumentation**: Dieser Punkt beschreibt den Dokumentationsaufwand, der mit dem ausgewählten Prozessmodell einhergeht.
- **Komplexität**: Dieser Punkt beschreibt die Komplexität des Prozesses. Darin sind die Regeln und Richtlinien angeführt, die im Prozess enthalten sind.

Betrachtet man die projektspezifischen Anforderungen aus Kapitel 2 so erkennt man, dass für fortlaufende Projekte, die nicht in sich abgeschlossen sind, das Scrum-Modell besser auf die sich ständig verändernden Anforderungen reagieren kann. Des Weiteren sprechen die Faktoren Komplexitätsgrad und Dokumentationsaufwand für den Scrum-Prozess.

Somit ist die vom Management getroffene Entscheidung für den Scrum Softwareentwicklungsprozess zum aktuellen Zeitpunkt des Projektes auch die richtige Entscheidung.

[11] Weitere Informationen zu „Value-based pricing“ findet man unter: http://en.wikipedia.org/wiki/Value-based_pricing

# 4. Technologievergleich und Auswahl

Einer der Punkte aus Kapitel 2 ist es, die Überwachungsplattform webbasiert zu realisieren.

Hierfür existieren von statischen Webseiten über dynamische Content Management Systemen (CMS[12]) bis hin zu hoch modernen Rich-Internet-Applikationen eine Unmenge an Technologien, die für die Realisierung in Frage kommen würden. Gegenständlicher Technologievergleich beschäftigt sich nur mit RIA Technologien.

Die Technologien wurden auf den RIA Teilbereich der Möglichkeiten beschränkt, da RIA-Applikationen State-of-the-Art in interaktiven Webapplikationen sind. Durch den Einsatz neuer Möglichkeiten, beispielsweise die Drag & Drop Funktionalität, werden die Applikationen benutzerfreundlicher und bedienbarer. Die Applikationen werden interaktiver und erhöhen die User Experience (Hans-Dirk, 2008). Des Weiteren können so die ständig wachsenden Erwartungen auf der Anwenderseite erfüllt werden. Eine größere Zielgruppe kann mit dem Einsatz von RIA-Applikationen auf mobilen Endgeräten erreicht werden (Fraternali, Comai, Bozzon, & Toffetti Carughi, April 2010). Ständig wachsende Erwartungen auf der Anwenderseite können unter anderem das Bedürfnis nach neuen Informationen innerhalb des Browsers ohne neues Laden sein, oder neue Möglichkeiten um zwischen Informationen zu navigieren (Alor-Hernandez, et al., 2009).

Dieser Abschnitt beschäftigt sich mit diversen Technologien, die es ermöglichen, die Webplattform als Rich-Internet-Applikation zu realisieren. Beginnend mit einer Auflistung der Möglichkeiten und den Unterschieden der verschiedenen Technologien, wird dieser Abschnitt mit der Auswahl für eine Technologie abgeschlossen.

## 4.1. Vergleichskriterien

Es werden nun einzelne Vergleichskriterien aufgelistet, die zur Verwendung einer Technologie für das CLM WebControl unerlässlich sind. Entscheidend ist die Unterstützung von Webservices jeglicher Art, da zur Informationsbeschaffung rein auf diese gesetzt wird. Das bedeutet, dass die Webplattform nur über Webservices mit der Serverkomponente für das Überwachungssystem kommunizieren wird. Technologisch wurde dies im Server-Team so fixiert. Des Weiteren wurde die Verbreitung einer Technologie als wesentliches Entscheidungsmerkmal festgelegt. Abschließend wird diese Liste durch die zusätzlichen Tools zur Unterstützung der Entwicklung konkretisiert. Diese drei Punkte sind ein kleiner Auszug jener Punkte, die für eine Entscheidungsgrundlage zwischen verschiedenen Technologien existieren. Diese Liste erhebt keinen Anspruch auf Vollständigkeit, sie ist jedoch für diverse Anforderungen aus Kapitel 2 ausreichend.

### 4.1.1. Webservice

Laut Booth (Booth, Haas, & McCabe, 2004) ist ein Webservice im übertragenen Sinne eine Softwarekomponente die es ermöglicht, plattformübergreifende Interaktionen zwischen Maschinen netzwerkübergreifend zu realisieren. Ein Webservice wird anhand einer Schnittstelle

[12] Weitere Informationen zur Thematik CMS finden Sie hier: http://de.wikipedia.org/wiki/Content-Management-System

(WSDL – Web Service Description Language) spezifiziert. Fremde Systeme kommunizieren mit dem Webservice über SOAP (Simple Object Access Protocol) basierte Nachrichten. Dies sind XML serialisierte Objekte, welche mittels HTTP (Hypertext Transfer Protocol) übertragen werden. Man kann somit sagen, dass ein Webservice das Bindeglied zwischen Systemgrenzen und Technologien für verschiedenste Anwendungen sein kann.

Aufbauend auf die Definition von Webservices stellt sich die Frage, warum die Verwendbarkeit von Webservices für die Technologie-Entscheidung notwendig ist. Sie ist notwendig, da diese als Basistechnologie gesehen werden kann. Im Sinne der Wiederverwendbarkeit wird das Rad nicht neu erfunden und es muss somit auch nicht der Umgang mit Webservices auf Low-Level Ebene implementiert werden. Eine Grundunterstützung für Webservices muss von der verwendeten Technologie zur Verfügung gestellt werden. Auf die verschiedenen Protokolle, die Webservices benutzen können, wird bei der Technologie-Auswahl noch nicht eingegangen. Detailliertere Informationen hierzu werden im Kapitel 6 „Ausgewählte Details der Implementierung“ betrachtet.

### 4.1.2. Erreichbarkeit / Verfügbarkeit

Ein weiterer Punkt für die Auswahl der Technologie ist eine möglichst hohe Unterstützung dieser über Systemgrenzen hinweg. Dies muss nicht zwingend notwendig sein, jedoch sollte diese Technologie den zu erwartenden Markt größtenteils abdecken.

### 4.1.3. Entwicklungsumgebung

Des Weiteren sollte die Technologie eine möglichst gut integrierte Entwicklungsumgebung, auch IDE (Integrated Development Environment) genannt, besitzen. Beispielsweise kann durch eine Code-Vervollständigung der Prozess der Softwareentwicklung vereinfacht und dadurch Fehler vermieden werden.

### 4.1.4. Dokumentation

Ein weiteres Kriterium sind die Dokumentation der Technologie, sowie die entsprechenden Update-Zyklen. Sprich in welchen Intervallen gibt es Aktualisierungen bzw. wie aktiv ist die Community bei Open Source Frameworks.

Die Anforderungen

- Webservice
- Erreichbarkeit / Verfügbarkeit
- IDE
- Dokumentation

bilden im weiteren Verlauf dieses Kapitels die Entscheidungsgrundlage für eine Technologie.

## 4.2. Überblick Technologien

Welche Technologien würden für die Entwicklung einer Webapplikation generell in Frage kommen?

Nachfolgend diesbezüglich eine vorselektierte Liste:

### 4.2.1. Silverlight[13]

Silverlight ist ein Browser Plug-In für Rich Internet Applikationen. Es ist ein proprietäres Framework aus dem Hause Microsoft und beinhaltet ausgewählte Funktionalitäten des .NET Frameworks. Silverlight beinhaltet wesentliche Konzepte für die Trennung von Design und Applikationslogik, wie sie auch im .NET Framework mit Windows Presentation Foundation (WPF[14]) vorhanden sind. Das Mittel zur Trennung von Design und Logik ist die Extensible Application Markup Language (XAML[15]). Detailliertere Informationen über die Verwendung von XAML befinden sich im Kapitel 6, welches ausgewählte Details der Implementierung beinhaltet.

Es folgt die Evaluierung der technologischen Anforderungen.

- **Webservice 4.1.1**
  Das Silverlight Framework bietet bereits mit Auslieferung des Frameworks eine sehr gute Webservice-Unterstützung. Diese Unterstützung basiert auf dem Windows Communication Framework (WCF[16]), das ein fixer Bestandteil des .NET Frameworks ist.

- **Verfügbarkeit / Erreichbarkeit 4.1.2**
  Das Silverlight Browser Plug-In wird für alle gängigen Windows-Betriebssysteme und dem Internet Explorer ab Version 7 angeboten. Des Weiteren funktionieren die Browser Firefox ab Version 3 und Chrome ab Version 4 offiziell mit Silverlight. Für Macintosh ab Version 10.4, Firefox ab Version 3 und Safari ab Version 3 gibt es ebenfalls eine offizielle Unterstützung für Silverlight (Compatible Operating Systems and Browsers). Inoffiziell unterstützt auch der Browser Opera mit der Version 10 das Silverlight Plug-In, wird aber nicht offiziell auf der Kompatibilitätsseite für das Plug-In angeführt. Keine offizielle Unterstützung gibt es zurzeit für Unix und Linux basierte Betriebssysteme, welche aber eine Open Source Nachbildung mit dem Projekt Moonlight besitzen. Moonlight bietet aktuell eine komplette Unterstützung für Silverlight 2,

[13] Weitere Informationen zu Silverlight findet man unter: http://www.silverlight.net/
[14] Weitere Informationen zu WPF findet man unter: http://de.wikipedia.org/wiki/Windows_Presentation_Foundation
[15] Weitere Informationen zu XAML findet man unter: http://de.wikipedia.org/wiki/Extensible_Application_Markup_Language
[16] Weitere Informationen zu WCF findet man unter: http://de.wikipedia.org/wiki/Windows_Communication_Foundation

Silverlight 3 ist in Entwicklung. Es existiert noch keine Unterstützung für Silverlight 4, das Anfang April dieses Jahres offiziell von Microsoft veröffentlicht wurde (Novell). Laut (Rich Internet Application Statistics, 2010) liegt die Verbreitung von Silverlight in den diversen Versionen bei ca. 63%.

- **IDE 4.1.3**
  Für die Entwicklung von Silverlight Applikationen gibt es eine Reihe von Entwicklungsumgebungen. Aus dem Hause Microsoft sind dies Visual Studio[17]und Expression Blend[18]. Visual Studio besitzt einen integrierten WYSIWYG[19]Designer und Code Editor, welcher eine komfortable IntelliSense-Unterstützung für die Entwicklung bietet. IntelliSense bedeutet nichts anderes, als dass dem Entwickler eine automatische Code-Vervollständigung zur Verfügung gestellt wird. Dies unterstützt den Entwickler wesentlich bei seiner täglichen Programmierarbeit.

- **Dokumentation 4.1.4**
  Unerlässlich für die Entwicklung mit einer Technologie sind die Dokumentation derselben und eine möglichst aktive Online Community dieser Technologie. Eine vollständige Dokumentation von Silverlight ist auf der MSDN[20] Plattform zu finden. *„Our forums have 249,862 threads and 288,435 posts, contributed by 83,870 members from around the world. In the past day, we had 107 new threads, 491 new posts, and 54 new users."* (The Official Microsoft Silverlight Forum)

### 4.2.2. JavaFX[21]

JavaFX ist ein Framework für RIA[22] Applikationen. Es hat den Vorteil, dass es plattformübergreifend verfügbar ist und verwendet werden kann. Es ist ein proprietäres Format und gehört zur Oracle[23] Firmen Gruppe. Programmiert wird es anhand von JavaFX Script[24]. Wie bereits im vorherigen Abschnitt betrachtet, besteht auch in JavaFX die Möglichkeit, die Logik vom Design zu trennen. In JavaFX ist hierfür der Einsatz von Cascading Style Sheets (CSS[25])

---

[17]Weitere Informationen zu Visual Studio findet man unter:
http://www.microsoft.com/germany/visualstudio/products/default.aspx
[18]Weiter Informationen zu Expression Blend findet man unter:
http://www.microsoft.com/expression/products/blend_overview.aspx
[19]WYSIWYG bedeutet „*What You See Is What You Get*", weitere Informationen hierfür finden Sie unter:
http://de.wikipedia.org/wiki/WYSIWYG
[20] Weiter Informationen zum Microsoft Developer Network(MSDN) findet man unter:
http://msdn.microsoft.com/de-de/library/cc838158(VS.95).aspx
[21] Weitere Informationen zu Java FX findet man unter: http://javafx.com/
[22] Rich Internet Applikation, kurz RIA weitere Informationen findet man unter:
http://de.wikipedia.org/wiki/Rich_Internet_Application
[23] Weiter Informationen zu Oracle findet man unter: http://www.oracle.com/index.html
[24] Weitere Informationen zu JavaFX Script findet man unter: http://java.sun.com/javafx/1.3/docs/api/index.html
[25] Weitere Informationen zu Cascading Style Sheets findet man unter:
http://de.wikipedia.org/wiki/Cascading_Style_Sheets

notwendig. Aus „Listing 1: JavaFX Hello-World-Programm“ ist exemplarisch ein Hello-World-Programm ersichtlich, das mit JavaFX realisiert wurde. Zu Beginn des Code-Ausschnitts werden durch „import“ die notwendigen Komponenten referenziert. Darauf aufbauend wird mit „Stage“ der Beginn des Programm-Containers definiert. Mit „Scene“ wird innerhalb des Programm-Containers der visuelle Container der Applikation definiert. In diesem speziellen Beispiel entspricht dies einem Text mit dem Inhalt „Hello World“.

```
package javafx_hello_world;

import javafx.stage.Stage;
import javafx.scene.Scene;
import javafx.scene.text.Text;
import javafx.scene.text.Font;

Stage {
    title: "Application title"
    scene: Scene {
        width: 100
        height: 50
        content: [
            Text {
                font: Font {
                    size: 16
                }
                x: 10
                y: 30
                content: "Hello World"
            }
        ]
    }
}
```

**Listing 1: JavaFX Hello-World-Programm**

Evaluierung der technologischen Anforderungen:

- **Webservice 4.1.1**
  Wie Silverlight bietet auch JavaFX eine Unterstützung für Webservices. Es kann mit Representational State Transfer (REST[26]) basierten Webservices, aber auch mit Simple Object Access Protocol (SOAP[27]) basierten Webservices umgehen.

- **Erreichbarkeit / Verfügbarkeit 4.1.2**

[26] Weitere Informationen zu REST findet man unter: http://de.wikipedia.org/wiki/Representational_State_Transfer
[27] Weitere Informationen zu SOAP findet man unter: http://de.wikipedia.org/wiki/SOAP

Potentiell ist die Verfügbarkeit von JavaFX überall dort gewährleistet, wo sich bereits eine Java Runtime Environment (JRE[28]) befindet. Laut (Rich Internet Application Statistics, 2010) liegt die Verbreitung von Java in den diversen Versionen bei ca. 75%.

- **IDE 4.1.3**
  Für die Entwicklung von JavaFX wird vom Hersteller Oracle ein Plug-In für die NetBeans[29] IDE angeboten, ebenso ein Plug-In für die Eclipse[30] IDE. Die NetBeans IDE besitzt eine IntelliSense-Unterstützung für JavaFX im Code Editor, ein integrierter Designer wird von Haus aus nicht mitgeliefert, jedoch gibt es eine Preview (JavaFX Composer) des JavaFX Composers[31], der für die visuelle Gestaltung der JavaFX Applikationen gedacht ist.

- **Dokumentation 4.1.4**
  Auch Oracle veröffentlichte eine vollständige Dokumentation[32] des JavaFX Frameworks. „*Messages: 9073, Topics: 2359, Views: 218:649*“ sind die statistischen Daten des offiziellen JavaFX Forums (JavaFX General Forum).

### 4.2.3. Adobe Flex[33]

Adobe Flex ist ein Framework aus dem Hause Adobe[34] zur Erstellung von RIA-Applikationen. Ein Vorteil von Flex ist, dass das Framework als Open Source veröffentlicht wurde. Zur Interpretation von Adobe Flex-Anwendungen wird für Browsers der Adobe Flash Player benötigt bzw. für Desktop-Applikationen die Adobe Integrated Runtime (AIR[35]). Als Programmiersprache wird MXML[36] und ActionScript[37] verwendet. MXML ist ein auf XML basierende Beschreibungssprache, anhand derer das Aussehen, das Layout der Benutzeroberfläche, gestaltet werden kann. Sind in der Beschreibung MXML-Tags enthalten, werden diese bei der Programmkompilierung in ActionScript Klassen umgewandelt, da ein MXML-Tag einer ActionScript Klasse entspricht. Aus „Listing 2: Adobe Flex Hello-World-Programm“ ist ein Hello-World-Programm ersichtlich, das mittels Adobe Flex realisiert wurde.

---

[28] Mehr Information zu JRE findet man unter: http://de.wikipedia.org/wiki/Java-Laufzeitumgebung
[29] Mehr Informationen zu NetBeans findet man unter: http://netbeans.org/
[30] Mehr Informationen zu Eclipse findet man unter: http://www.eclipse.org/
[31] Weitere Informationen zu JavaFX Composer findet man unter: http://wiki.netbeans.org/JavaFXComposer
[32] Weitere Informationen zu Dokumentation von JavaFX findet man unter: http://download.oracle.com/docs/cd/E17411_01/javafx/index.html
[33] Weitere Informationen zu Adobe Flex findet man unter: http://www.adobe.com/de/products/flex/
[34] Weitere Informationen zu Adobe findet man unter: http://www.adobe.com/de/
[35] Weitere Informationen zu Adobe AIR findet man unter: http://www.adobe.com/products/air/
[36] Weitere Informationen zu MXML findet man unter: http://en.wikipedia.org/wiki/MXML
[37] Weitere Informationen zu ActionScript findet man unter: http://de.wikipedia.org/wiki/ActionScript

```
<?xml version="1.0" encoding="utf-8"?>
<s:Application xmlns:fx="http://ns.adobe.com/mxml/2009"
                        xmlns:s="library://ns.adobe.com/flex/spark"
                        xmlns:mx="library://ns.adobe.com/flex/mx" minWidth="100" minHeight="100">
    <s:Label text="Hello World" paddingLeft="10" paddingTop="10"/>
</s:Application>
```

**Listing 2: Adobe Flex Hello-World-Programm**

Der MXML-Tag "<s:Application ...>" ist der Container für das Adobe Flex Programm. Die Attribute „xmlns" ermöglichen es, weitere Softwarekomponenten zu dieser Applikation zu referenzieren. Durch den MXML-Tag „<s:Label …>" wird der Applikation ein Textfeld hinzugefügt, das keine Eingabeoption besitzt. Dieses „Label" dient lediglich zur Darstellung des Inhaltes, welcher dem Text „Hello World" entspricht.

Evaluierung der technologischen Anforderungen:

- **Webservice 4.1.1**
  Ebenso wie Silverlight und JavaFX kann Adobe Flex mit einer Unterstützung für Webservices von Haus aus aufwarten.

- **Erreichbarkeit / Verfügbarkeit 4.1.2**
  Potentiell ist die Verfügbarkeit von Adobe Flex überall dort gewährleistet, wo sich bereits eine Installation des Adobe Flash Players in der Version 10 oder größer befindet. Laut (Rich Internet Application Statistics, 2010) liegt die Verbreitung des Flash Players[38] diverser Versionen bei ca. 92%.

- **IDE 4.1.3**
  Für die Entwicklung von Adobe Flex-Anwendungen wird vom Hersteller eine IDE kostenpflichtig zur Verfügung gestellt, dies ist der Adobe Flash Builder[39]. Der Adobe Flash Builder basiert auf der Open Source IDE Eclipse und besitzt einen Code Editor und einen grafischen Designer mit WYSIWYG Editor. Ebenso wie die Visual Studio IDE und NetBeans IDE besitzt der Adobe Flash Builder eine IntelliSense-Unterstützung für die Programmierung.

[38] Weitere Informationen zu Adobe Flash Player findet man unter: http://www.adobe.com/de/products/flashplayer/
[39] Weitere Informationen zum Adobe Flash Builder findet man unter: http://www.adobe.com/products/flashbuilder/

- **Dokumentation 4.1.4**
  Auch Adobe veröffentlichte eine Dokumentation[40] zum Flex Framework. „*Discussions: ~44000*“ sind die statistischen Daten aus dem offiziellen Adobe Flex Forum (Forums - Flex).

## 4.3. Fazit Technologievergleich und Auswahl

Dieser Abschnitt behandelt die Entscheidungsfindung der Technolgie-Auswahl. Die drei Konkurrenten Adobe Flex, JavaFX und Silverlight verfügen alle über eine vollständige Dokumentation und über eine hinreichende IDE Unterstützung inklusive IntelliSense-Funktionalität. Bei der Erreichbarkeit und Aktivität der Community gibt es grobe Unterschiede. Auf der einen Seite Adobe Flex mit einer breiten Marktdurchdringung von über 90% dank des Flash Players, jedoch einer geringen Aktivität der Community, wenn man sich auf die spärlichen Zahlen (siehe Kapitel 4.2.2) des offiziellen Forums bezieht. Silverlight hingegen hat eine sehr aktive Community, jedoch erst eine Marktdurchdringung von ca. 60%. JavaFX liegt im Mittelfeld, muss aber Abstriche bei der IDE Unterstützung durch einen fehlenden Designer hinnehmen.

Zusätzlich zur Marktdurchdringung der RIA Technologien laut (Rich Internet Application Statistics, 2010) gibt es noch einen weiteren Anbieter der ähnliche Ergebnisse erzielt wie (Rich Internet Application Statistics, 2010). Dies ist aus „Abbildung 13: Rich Internet Application Usage “ ersichtlich.

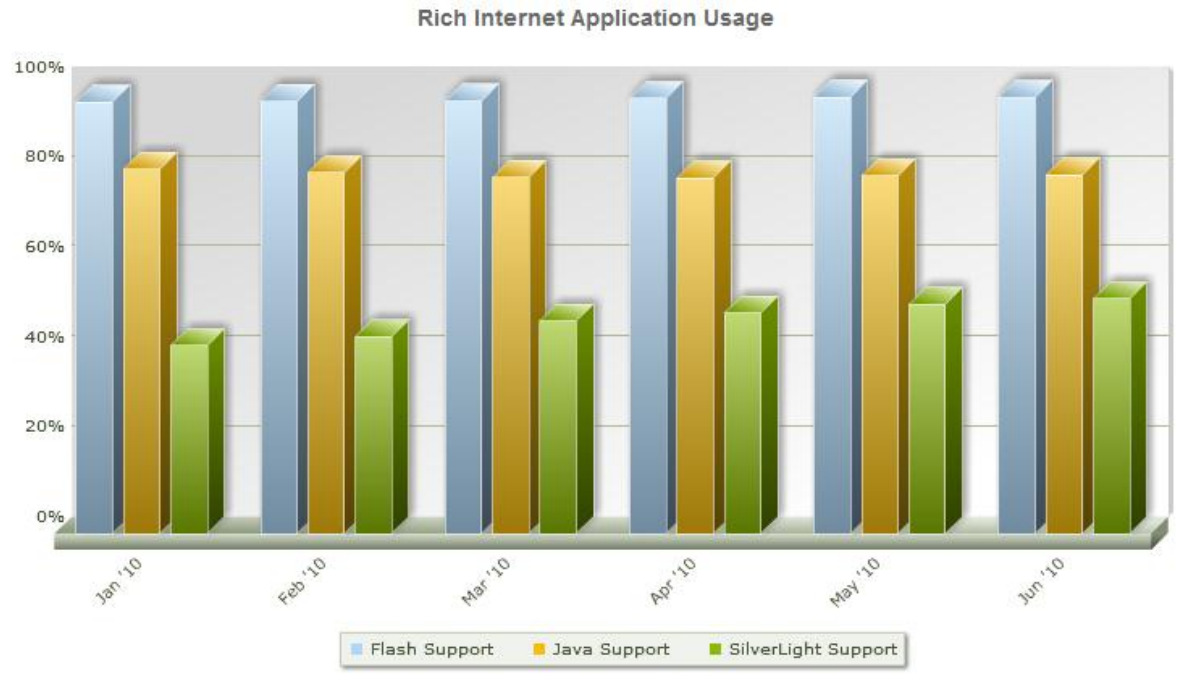


**Abbildung 13: Rich Internet Application Usage (Rich Internet Application Market Share, 2010)**

[40] Weitere Informationen zur Dokumentation von Adobe Flex findet man unter: http://www.adobe.com/livedocs/flex/2/langref/

Im folgenden Absatz werden weitere Parameter für die Entscheidungsfindung betrachtet.

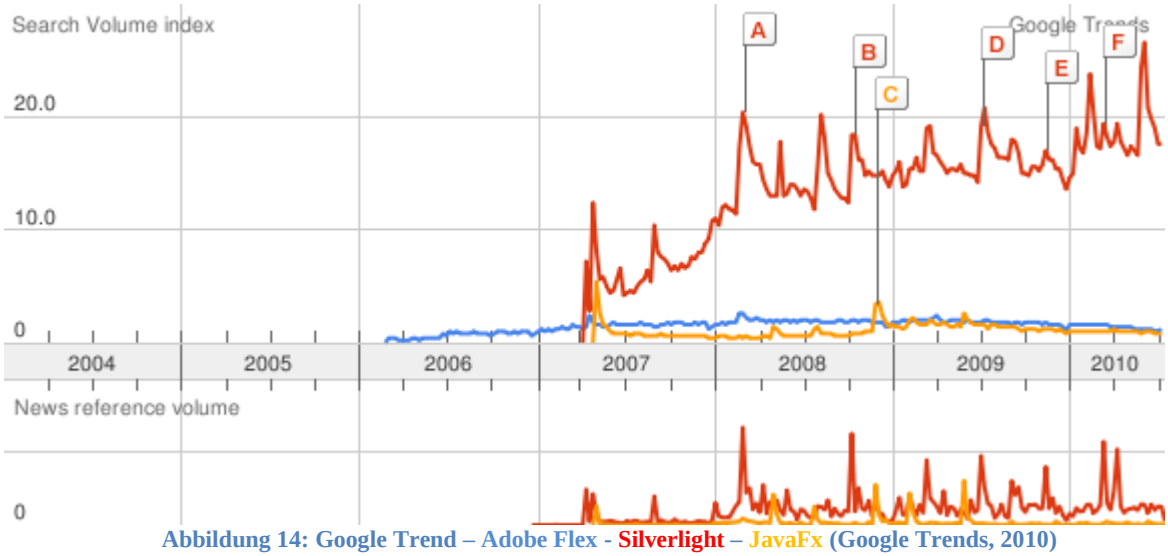


**Abbildung 14: Google Trend – Adobe Flex - Silverlight – JavaFx (Google Trends, 2010)**

Die „Abbildung 14: Google Trend – Adobe Flex - Silverlight – JavaFx“ ist eine Google Trend[41] Analyse mit den Suchbegriffen:

- Adobe Flex
- Silverlight
- JavaFX

Daraus ist ersichtlich, dass es einen signifikanten Unterschied beim Suchverhalten des am meist verwendeten Suchmaschinenanbieters der Welt Google (Alexa Site Info, 2010) gibt. Demnach wurde der Suchbegriff Silverlight weit öfter verwendet als Adobe Flex oder JavaFX.

Ein weiterer Vergleich der drei Technologien erfolgt anhand der Jobangebote des Anbieters Indeed[42].

[41] Weitere Informationen zu Google Trend findet man unter: http://www.google.com/trends
[42] Weitere Informationen zu Indeed JobTrend findet man unter: http://www.indeed.com/jobtrends

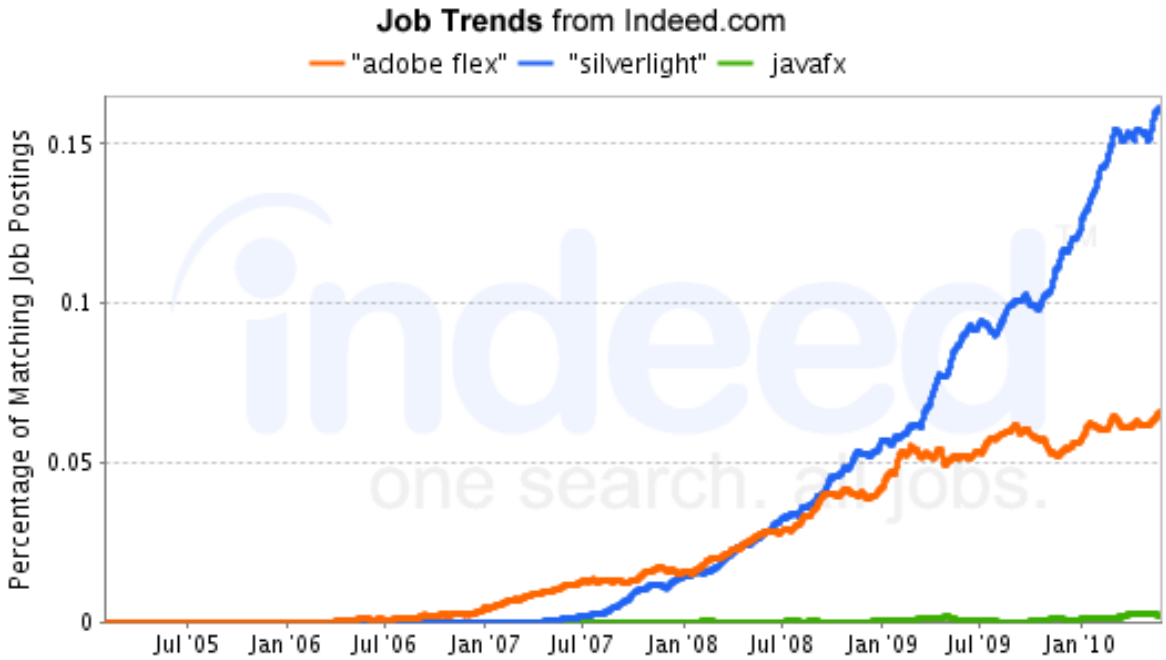


Abbildung 15: Job Trend von Indeed.com (Indeed Job Trend, 2010)

Aus der „Abbildung 15: Job Trend von Indeed.com“ ist klar ersichtlich, dass Silverlight einen höheren Prozentanteil zu passenden Jobs angeboten hat, als seine Konkurrenten.

| | **Silverlight** | **Adobe Flex** | **JavaFX** |
|---|---|---|---|
| **Marktdurchdringung** | ~55% | ~90% | ~75% |
| **Webservice Unterstützung** | ja | ja | ja |
| **IDE Unterstützung** | ja, inkl. WYSIWYG Designer und IntelliSense | ja, inkl. WYSIWYG Designer und IntelliSense | ja, ohne WYSIWYG Designer, aber mit IntelliSense |
| **Dokumentation** | ja | ja | ja |
| **Community** | groß | gering | mittel |
| **RIA Google Trend** | hoch | gering | gering |
| **Job Trend** | hoch | mittel | gering |

Tabelle 2: Vergleich RIA Technologien

Zusammenfassend wird festgehalten, dass in Anbetracht der Job-Aussichten, der aktiven Community und der sehr guten IDE Unterstützung die Wahl für Silverlight mit bestem Gewissen vertretbar ist. Des Weiteren wurden bei der Evaluierung keine technischen Hindernisse wahrgenommen, die die Entscheidung für Silverlight gravierend verändert hätten.

Ein weiterer Punkt für die Entscheidung der Silverlight-Technologie ist, dass die restlichen Komponenten des Gesamtsystems aus dem Kapitel 2.1.2 auch in den entsprechenden .NET

Technologien entwickelt wurden. Somit sind Synergieeffekte aus technologischen Entscheidungen zu erwarten. Beispielswiese ist die Silverlight-Technologie sehr stark an den Konzepten der WPF-Technologie angelehnt, sodass sich hier erste Anknüpfungspunkte ergeben.

# 5. User Interface Design Patterns

Dieses Kapitel beschäftigt sich mit diversen Softwarekonzepten zur Erstellung von Graphical User Interfaces (GUIs).

Bei der Entwicklung von Benutzeroberflächen kann es mehrere Herausforderungen geben, die eine flexible Architektur der Software benötigen. Gerade bei raschen Kundenwunschänderungen muss die Benutzeroberfläche in der Lage sein, diese auch in einem verträglichen Kosten-Nutzen-Verhältnis zu realisieren. Dies ist idealerweise dort der Fall, wo es eine ersichtliche Trennung zwischen Daten, Interaktionen und visuellen Komponenten gibt.

Um diese Herausforderungen zu meistern, gibt es im Wesentlichen drei Designer Patterns, die sich mit dieser Thematik beschäftigen.

1. Model View Controller (MVC)
2. Model View Presenter (MVP)
3. Model View ViewModel (MVVM)

Nachfolgend werden die einzelnen Patterns erläutert und abschließend miteinander verglichen.

## 5.1. MVC

Das Model View Controller Pattern wurde erstmals von Trygve Teenskaug 1979 (Reenskaug, 1979) beschrieben. Er definierte die Schichten wie folgt:

- **Model**: Diese Schicht repräsentiert die Daten, die visualisiert werden sollen. Es kann ein einzelnes Objekt oder eine Struktur von Objekten sein.

- **View**: Die View-Schicht dient zur Präsentation der Daten aus der Model-Schicht. Des Weiteren wird diese Schicht zur Annahme der Benutzerinteraktionen benötigt. Diese Schicht kennt die Controller- und die Model-Schicht. Die Interpretation der Benutzerinteraktionen wird an die Controller-Schicht weitergereicht.

- **Controller**: Diese Schicht ist für die Verwaltung der View-Schicht zuständig. Die Benutzerinteraktionen werden von der Controller-Schicht ausgewertet. Die Controller-Schicht ist nicht für die Manipulation der Daten zuständig.

Grundsätzlich kann man sagen, dass die Model-Schicht die Informationen beinhaltet, die visualisiert werden sollen, die View-Schicht, die Informationen des Models präsentiert und die Controller-Schicht, welche das Verhalten der Applikation definiert. (Chlebek, 2006)

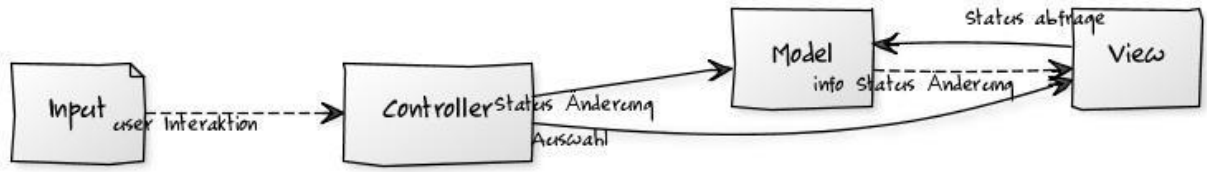


Abbildung 16: MVC Architektur (Curry & Grace, 2008)

Die „Abbildung 16: MVC Architektur“ veranschaulicht nochmals die Zusammenhänge zwischen den drei Schichten der MVC Architektur. Ausgehend vom Input wird der Controller über eine Userinteraktion informiert. Der Controller entscheidet aufgrund der Art der Userinteraktion, welche Schichten er darüber informieren soll. Beispielsweise kann diese Interaktion die Änderung eines Models zur Folge haben, oder aber eine andere View wird in den Vordergrund gebracht. Sofern die Userinteraktion eine Model-Änderung verlangt, wird der Controller die entsprechende Methode des Models zur Datenmanipulation aufrufen. Das Model hat keinerlei Kenntnis der View. Die View ist jedoch vom Model abhängig. Wird nun eine Änderung im Model notwendig, wird der Controller auch die View informieren, dass diese die Informationen aus dem Model aktualisiert. Nachteil dieses Design Patterns ist, dass die View sehr eng mit dem Model verknüpft ist. Wird das Model geändert, ist zwingend auch die Änderung der View erforderlich, umgekehrt ist dem nicht so.

## 5.2. MVP

Das Model View Presenter Pattern wurde erstmals 1990 von IBM und Taligent erwähnt und eingesetzt. (Fowler, 2006)

Der nachfolgende Abschnitt beruht auf den Arbeiten von (Potel, 1996) und (Alles, et al., 2006).

Das Model View Presenter Pattern besteht aus den folgenden drei Schichten:

- **Model**: Die Model-Schicht beinhaltet die Daten, die zur Visualisierung benötigt werden.
- **View**: Die View-Schicht ist für die Visualisierung der Daten verantwortlich und für die Interaktion mit dem Endanwender zuständig.
- **Presenter**: Die Presenter-Schicht ist für das Verhalten der Applikation zuständig. Sie steuert die Kommunikation zwischen der Model-Schicht und der View-Schicht.

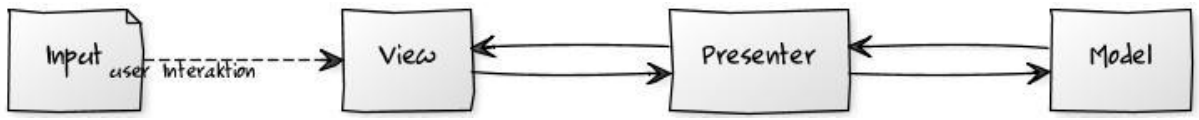


Abbildung 17: MVP Architektur (Microsoft, 2008)

Die „Abbildung 17: MVP Architektur veranschaulicht das Zusammenspiel der drei Schichten der MVP Architektur. Im Vergleich zum MVC Pattern wird die Userinteraktion nicht in der

Controller/Presenter Schicht behandelt, sondern in der View-Schicht. Sofern die Userinteraktion eine Model-Änderung erfordert, wird dies an den Presenter weitergereicht. Die Änderung wird vom Presenter an das Model weitergereicht. Darauf aufbauend wird die View mit den neuen Model-Informationen aktualisiert.

Vorteil des MVP Patterns zum MVC Pattern ist, dass die einzelnen Komponenten stärker voneinander getrennt sind und die View komplett vom Model getrennt wurde. Durch diese erhöhte Trennung der einzelnen Komponenten wird eine verbesserte Testbarkeit der Einzelteile erreicht. Jedoch ist in diesem Pattern nun die View stärker mit dem Presenter verbunden, sodass die Komponenten auch weiterhin sehr stark miteinander kombiniert sind.

## 5.3. MVVM

Das Model View Viewmodel Pattern wurde von John Gossman als Spezialisierung des MVP Pattern für den Einsatz in der WPF und Silverlight-Technologie definiert (Gossman, 2005).

Der nachfolgende Abschnitt beruht auf dem Dokument von (Gossman, 2005).

Das MVVM Pattern besteht aus den drei Schichten:

- **Model**: Die Model-Schicht des MVVM Patterns ist wie beim MVC Pattern definiert. Diese Schicht ist die Daten- bzw. Business-Logikschicht. Sie ist komplett UI unabhängig. Die Model-Schicht kann Daten aus einer relationalen Tabelle, einer XML Datei oder in Code geschriebener Form entsprechen.

- **View**: Die View-Schicht beinhaltet die visuellen Komponenten des Programms. Des Weiteren implementiert diese Schicht die Interaktionen mit den diversen Eingabeoptionen, die beispielsweise beim MVC Pattern im Controller abgebildet sind.

- **Viewmodel**: Diese Schicht kann auch als Model der View definiert werden bzw. als Abstraktion der View-Schicht. Diese Schicht beinhaltet die Transformationen der Models, sodass diese durch das Databinding[43] in der View verwendet werden können. Des Weiteren werden in dieser Schicht die Commands[44] implementiert, welche die View verwendet um mit den Models zu interagieren.

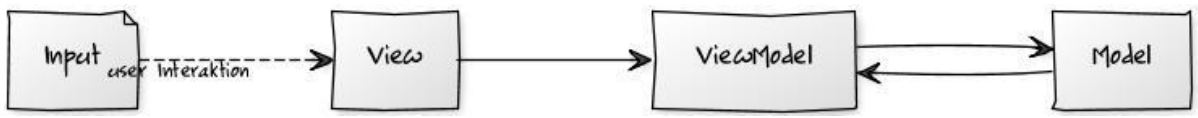


**Abbildung 18: MVVM Architektur**

[43] Weitere Informationen zum Databinding in Silverlight findet man unter: http://msdn.microsoft.com/en-us/library/cc278072%28VS.95%29.aspx

[44] Weitere Informationen zu Commands in Silverlight findet man unter: http://msdn.microsoft.com/en-us/library/ff921126%28PandP.20%29.aspx

Die „Abbildung 18: MVVM Architektur“ veranschaulicht die Interaktion der drei Schichten der MVVM Architektur. Die View-Schicht hat durch das spezielle Sprachfeature Data-Context[45] einseitigen Zugriff auf die Viewmodel-Schicht. Dieser Zugriff beschränkt sich auf die Möglichkeit, öffentliche Eigenschaften mit speziellen View-Eigenschaften über das Databinding zu verbinden. Wird nun eine Userinteraktion zur Model-Änderung ausgelöst, wird die Viewmodel-Schicht über Command-Binding benachrichtigt. Da die Viewmodel-Schicht selbst eine Referenz zur Model-Schicht hält, hat es die Möglichkeit, die Daten nun direkt zu manipulieren. Durch das Sprachfeature INotifyPropertyChanged[46] ist es der Viewmodel-Schicht möglich, die View-Schicht über die Änderung der Daten zu informieren ohne davon Kenntnis zu haben.

Im Vergleich zum MVC und MVP Pattern hat das MVVM Pattern die loseste Kopplung der Komponenten. Dieser Vorteil gepaart mit den Sprachfeatures für die Informationsübermittlung bei Zustandsänderungen des Models und des Viewmodels ist der ausschlaggebende Grund, weshalb dieses Pattern bei den WPF/Silverlight Entwicklern so gerne eingesetzt wird.

Nachteil dieses Patterns ist es, dass bei Refactoring[47]-Tätigkeiten des Viewmodels auch die entsprechenden Änderungen in der View eingetragen werden müssen. Aufgrund der besonderen Gegebenheiten der Silverlight und WPF Technologie werden fehlerhafte Bindings erst zur Laufzeit erkennbar. Diese können nicht zur Kompilierzeit erkannt werden.

## 5.4. Fazit User Interface Design Patterns

Diese drei Design Patterns repräsentieren eine Entwicklungsevolution von GUI spezifischen Design Patterns. Mit dem Aufkommen neuer Technologien und Mechanismen wurden auch neue Anforderungen an die zu verwendenden Entwurfsmuster (Design Pattern) gestellt, welche die neuen technologischen Features berücksichtigen.

Das MVVM Pattern ist die jüngste Entwicklung, welche genau für die Gegebenheiten von WPF und Silverlight definiert worden ist (Smith, 2009). Somit ist dieses Design Pattern ein ideales Muster, an das es sich bei der der Entwicklung von WPF bzw. Silverlight Applikationen zu halten lohnt.

Sofern man keine moderne Technologie verwendet, die den Einsatz von MVVM ermöglicht, muss man sich zwischen MVC und MVP entscheiden. Wobei es eine Unmenge an Abwandlungen dieses Patterns gibt. Ist die Testbarkeit ein wichtiges Thema, so haben die Pattern MVVM und MVP einen Vorteil gegenüber dem MVC Pattern. Da diese eine lose Koppelung verfolgen und somit einzelne Teile leichter durch Mock-ups[48] ersetzt werden können, ergo die Testbarkeit der einzelnen Bestandteile erhöht wird.

---

[45] Weiter Informationen zu DataContext in Silverlight findet man unter: http://msdn.microsoft.com/de-de/library/system.windows.frameworkelement.datacontext%28VS.95%29.aspx

[46] Weitere Informationen zum INotifyPropertyChanged Interface findet man unter: http://msdn.microsoft.com/en-us/library/system.componentmodel.inotifypropertychanged.aspx

[47] Weitere Informationen zum Thema Refactoring findet man unter: http://de.wikipedia.org/wiki/Refactoring

[48] Der Begriff Mock-up bedeutet so viel wie Attrappe. Weitere Informationen zum Thema Mock-up findet man unter: http://de.wikipedia.org/wiki/Mock-up

# 6. Ausgewählte Details der Implementierung

Aufgabe des praktischen Teils dieser Arbeit war es, die Anforderungen des Kapitel 2 zu realisieren.

Die Implementierung hat den größten Teil des Zeitbudgets für diese Masterarbeit in Anspruch genommen.

## 6.1. Eingesetzte Technologien

Wie bereits in Kapitel 4 erwähnt, wurde die gesamte CLM Plattform mit Microsoft-Technologien realisiert. Es wurden dabei die neuesten Versionen und die bestmöglichsten Tools verwendet.

Nachfolgend eine kurze Auflistung der verwendeten Technologien, gegliedert in die einzelnen Bereiche der CLM Plattform:

- **CLM Server**: WCF, Entity Framework[49], IIS & AppFabric[50], Enterprise Library[51]
- **CLM System Designer**: WCF, WPF, Prism[52]
- **CLM WebControl**: Silverlight, WCF, Prism

Als Unterstützung zur Entwicklung wurden folgende Tools verwendet: Visual Studio, Team Foundation[53] Server und Expression Blend.

Die Implementierung des CLM WebControls wird anhand eines reduzierten Beispiels erläutert. Dieses Beispiel wird nicht den vollen Umfang des realen Codes besitzen, da dieser die Lesbarkeit stark beinträchtigen würde. Um einen Überblick über den Umfang des CLM WebControls zu gewähren, folgen nun diverse Kennzahlen. Diese Kennzahlen wurden mithilfe des NDepend[54] Visual Studio Plugins erstellt.

- Anzahl der Klassen: 270
- Anzahl der Interfaces: 34
- Anzahl der Assemblies: 12
- Lines of Code (kurz LOC[55]): circa 9000
- Verhältnis C# Code zu XAML Code: circa 50%

[49] Weitere Informationen zum Entity Framework findet man unter: http://msdn.microsoft.com/en-us/library/aa697427(VS.80).aspx
[50] Weitere Informationen zur AppFabric findet man unter: http://msdn.microsoft.com/en-us/windowsserver/ee695849.aspx
[51] Weitere Informationen zur Enterprise Library findet man unter: http://entlib.codeplex.com/
[52] Weitere Informationen zu Prism findet man unter: http://compositewpf.codeplex.com/
[53] Weitere Informationen zum Team Foundation Server (TFS) findet man unter: http://msdn.microsoft.com/de-de/library/ms242904(VS.80).aspx
[54] Weitere Informationen zu NDepend findet man unter: http://www.ndepend.com/
[55] Weitere Informationen zu Lines of Code findet man unter: http://de.wikipedia.org/wiki/Lines_of_Code

Diese Kennzahlen verdeutlichen rein den Umfang des Projektes und geben keine Auskunft über die Qualität der Software.

## 6.2. CLM WebControl

Wie schon einleitend erwähnt, ist die CLM WebControl-Applikation der praktische Teil dieser Masterarbeit. Nachfolgend werden ausgewählte Details der Implementierung erläutert.

Bei der Implementierung des CLM WebControls wurde des Weiteren besonders auf die Erweiterbarkeit und Modularität der Applikation geachtet. Erweiterbarkeit dahingehend, dass neue Funktionalitäten mit relativ geringem Aufwand in die Applikation eingefügt werden können und Modularität, dass logische als auch funktionale Bereiche abgekapselt werden können, sodass die Module (Bausteine) so wenig wie möglich miteinander verstrickt sind.

### 6.2.1. Prism

Zur Lösung dieser Anforderungen wurde das Prism Framework verwendet. Prism wird unter der Rubrik „Microsoft Patterns & Practices“ auf der Open Source Initiative von Microsoft www.codeplex.com gehostet (Project Hosting for Open Source Software).

> *„Prism is designed to help you more easily build modular Windows Presentation Foundation (WPF) and Silverlight client applications. These types of applications typically feature multiple screens, rich, flexible user interaction and data visualization, and role-determined behavior. They are "built to last" and "built for change." This means that the application's expected lifetime is measured in years and that it will change in response to new, unforeseen requirements. This application may start small and over time evolve into a composite client—composite applications use loosely coupled, independently evolvable pieces that work together in the overall application.“ (Project Hosting for Open Source Software)*

Ein wesentliches Konzept von Prism ist, eine Applikation in Regionen (Bereiche) aufzugliedern. In diese Regionen kann zur Laufzeit ein Modul geladen oder entfernt werden. Für die Realisierung des CLM WebControls wurde eine schematische Untergliederung in Bereiche (Regionen) verwendet, welche aus der „Abbildung 19: CLM WebControl Regions“ ersichtlich ist.

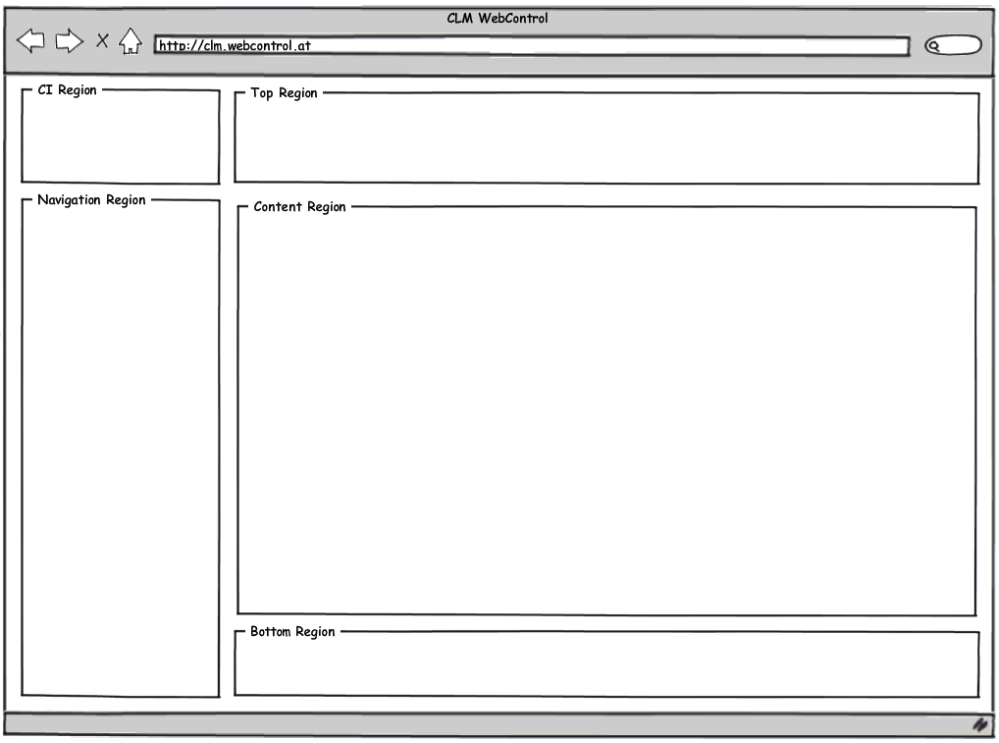


**Abbildung 19: CLM WebControl Regions**

Diese einzelnen Bereiche

- CI Region,
- Navigation Region,
- Top Region,
- Content Region und
- Bottom Region

werden je nach Bedarf mit Prism-Modulen erweitert. Exemplarisch könnte dies wie folgt aussehen:

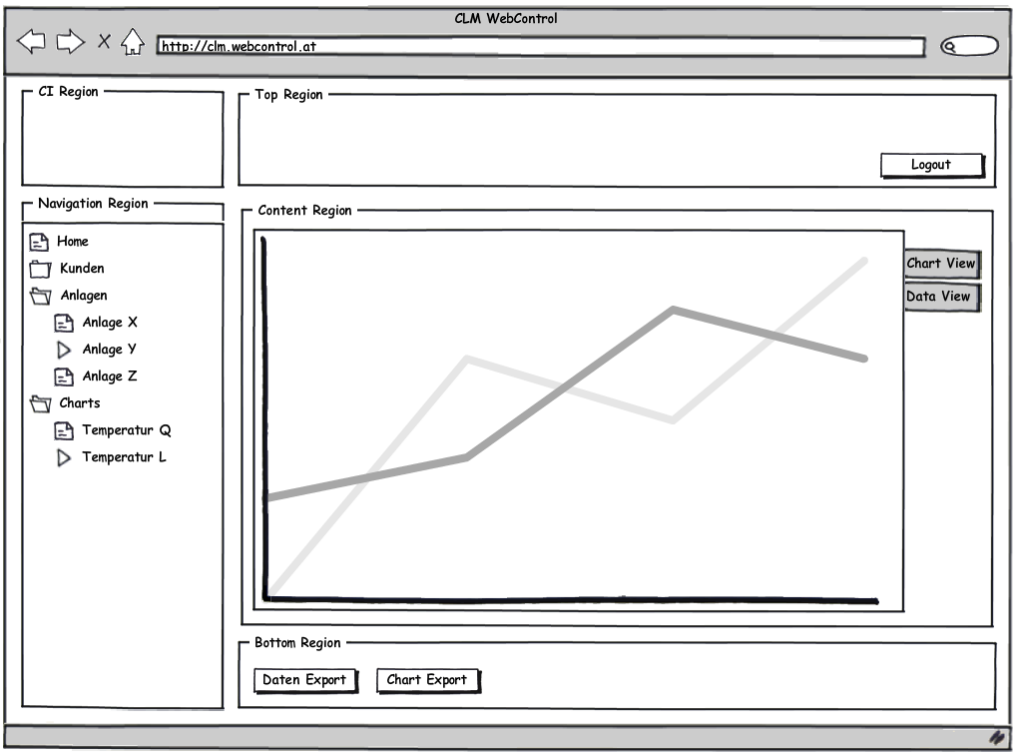


Abbildung 20: CLM Webcontrol Regions Mockup

Die „Navigation Region“ wurde mit einer baumstrukturartigen Navigation beladen, die „Content Region“ beinhaltet nun eine Chart-Ansicht, die „Top Region“ die Möglichkeit, sich vom System abzumelden und die „Bottom Region“ beinhaltet diverse Export-Funktionalitäten.

### 6.2.2. Realisierte Usecases

Nun folgt die Liste der gesamt realisierten Usecases.

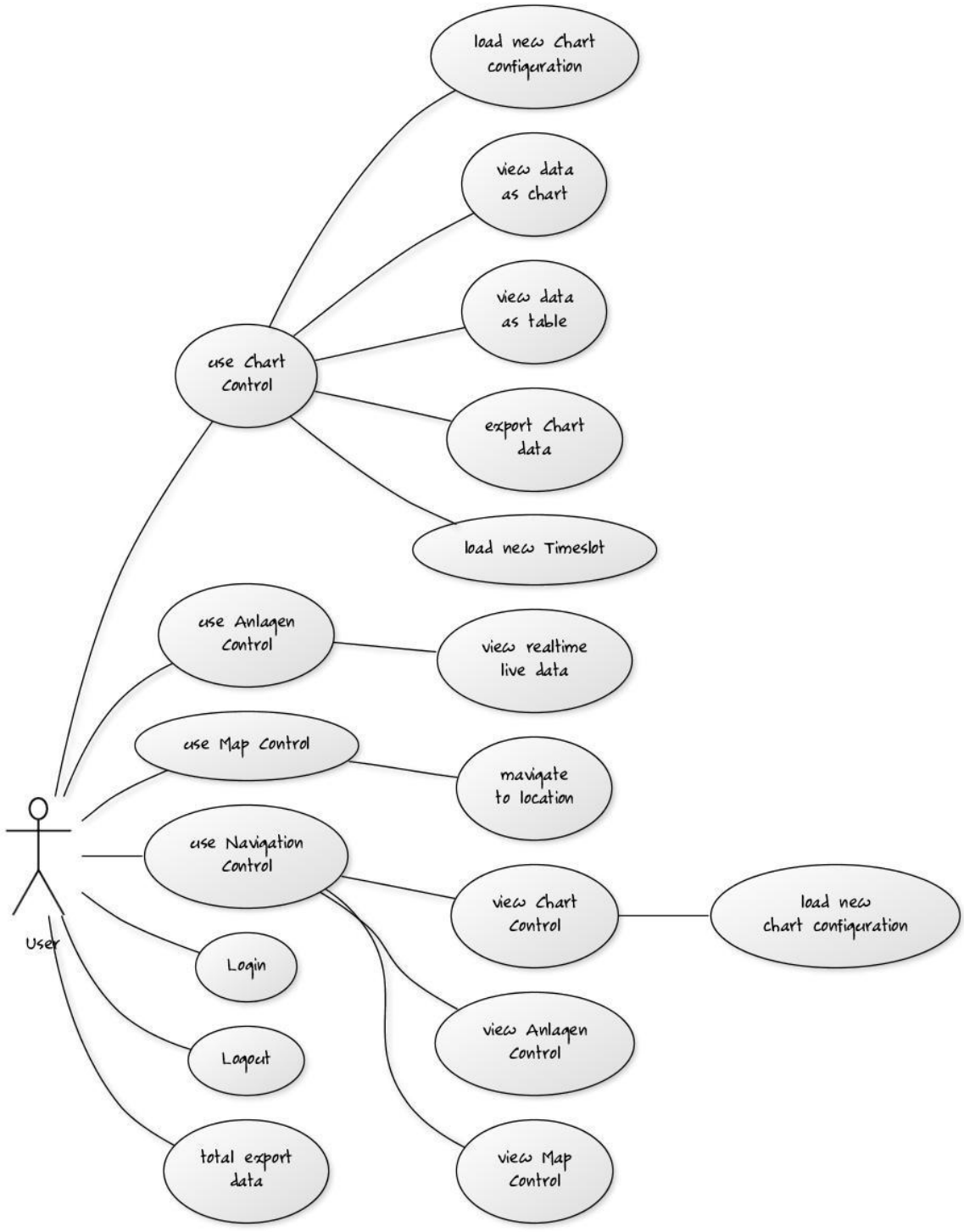


Abbildung 21: Gesamte Liste der Usecases

Die „Abbildung 21: Gesamte Liste der Usecases“ beinhaltet einige Usecases mehr als nachfolgend erläutert werden. Diese Reduzierung hat sich dadurch ergeben, dass die ausgewählten Usecases die Zusammenhänge und Implikationen bei der Realisierung des praktischen Teils dieser Masterarbeit einfacher darstellen konnten. Eine vollständige Anführung der Usecases inklusive Codebeispiele hätte den Umfang dieser Masterarbeit um ein Vielfaches erhöht.


**Abbildung 22: Gesamt Integration Chart Control Screenshot**

Die „Abbildung 22: Gesamt Integration Chart Control “ visualisiert die Integration des Chart Controls, sowie des Navigation Controls in das Gesamtsystem. Auf der linken Seite der Abbildung ist das Navigation Control ersichtlich, im rechten mittleren Bereich das Chart Control. Das Navigation Control beinhaltet eine Anlage und dazugehörig sechs Verlaufsdiagramme. Im Chart Control wird das erste Verlaufsdiagramm visualisiert, das zwei verschiedene Y-Achsenskalierungen beinhaltet. Zum einen die Vor- und Rücklauftemperatur eines Gaskessels mit der Einheit Grad Celsius (°C) und zum anderen den Durchfluss des Gaskessels mit der Einheit Liter pro Minute (l/min). Dies ist der aktuelle Status der funktionalen Anforderung, aufgezeichnete Daten im zeitlichen Kontext zu visualisieren. Im linken oberen Bereich der Abbildung ist das Corporate Identity Logo erkennbar. Im rechten oberen Bereich ist die Abmelde-Funktionalität der Applikation ersichtlich. Links unterhalb des Chart Controls ist die Rohdaten Export Funktionalität zu sehen. Mittels dieser Funktionalität hat der User die Möglichkeit, die gesamten aufgezeichneten Daten auf seinen lokalen Rechner zu transferieren.

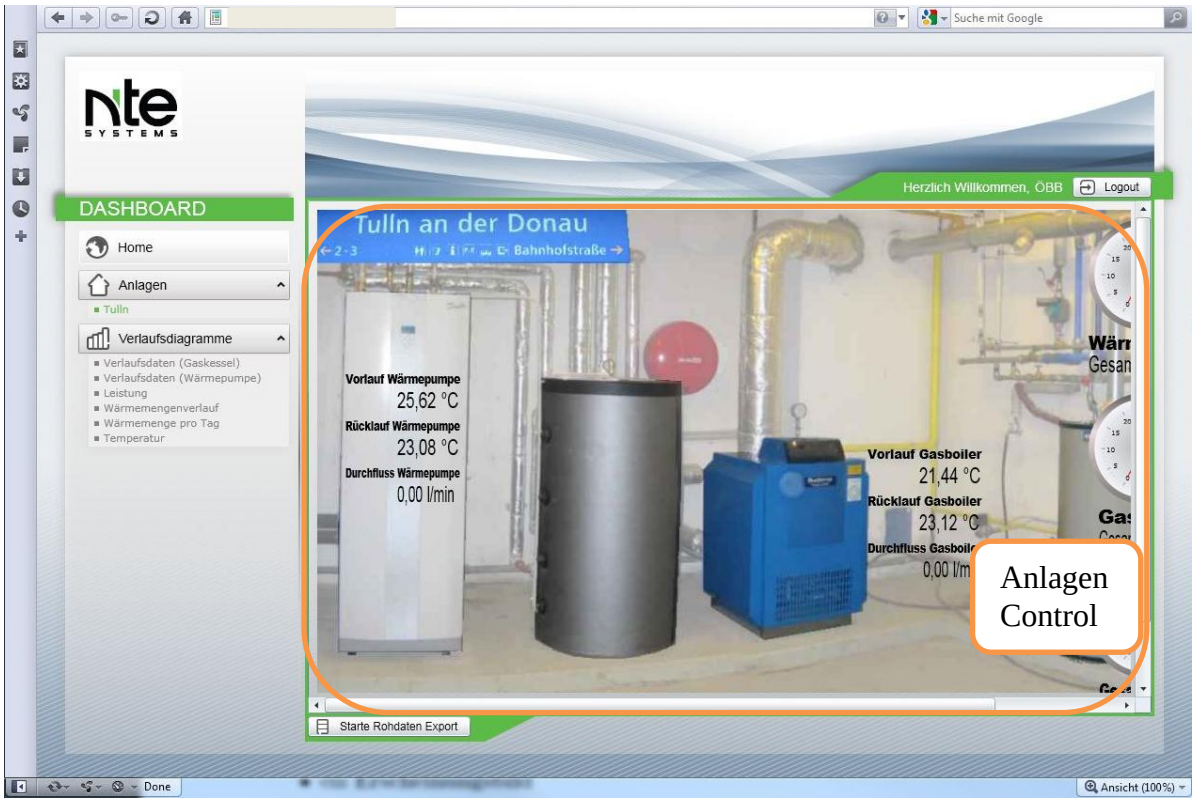


**Abbildung 23: Gesamt Integration Anlagen Control Screenshot**

Die „Abbildung 23: Gesamt Integration Anlagen Control Screenshot“ visualisiert das Anlagen Control, eingebettet in das gesamte System der CLM WebControl Applikation. Das Anlagen Control beinhaltet das Anlagenschema der Anlage, welche im Navigation Control ausgewählt ist. Im Anlagenschema ist eine fotorealistische Darstellung ersichtlich, diese beinhaltet eine Wärmepumpe und einen Gaskessel (Gasboiler). Die einzelnen Objekte Wärmepumpe und Gasboiler wurden mit Textfeldern versehen, die den aktuellen Wert eines Sensors repräsentieren. Diese Darstellungsform repräsentiert die Realisierung der funktionalen Anforderung, aufgezeichnete Daten im energetischen Kontext zu visualisieren.

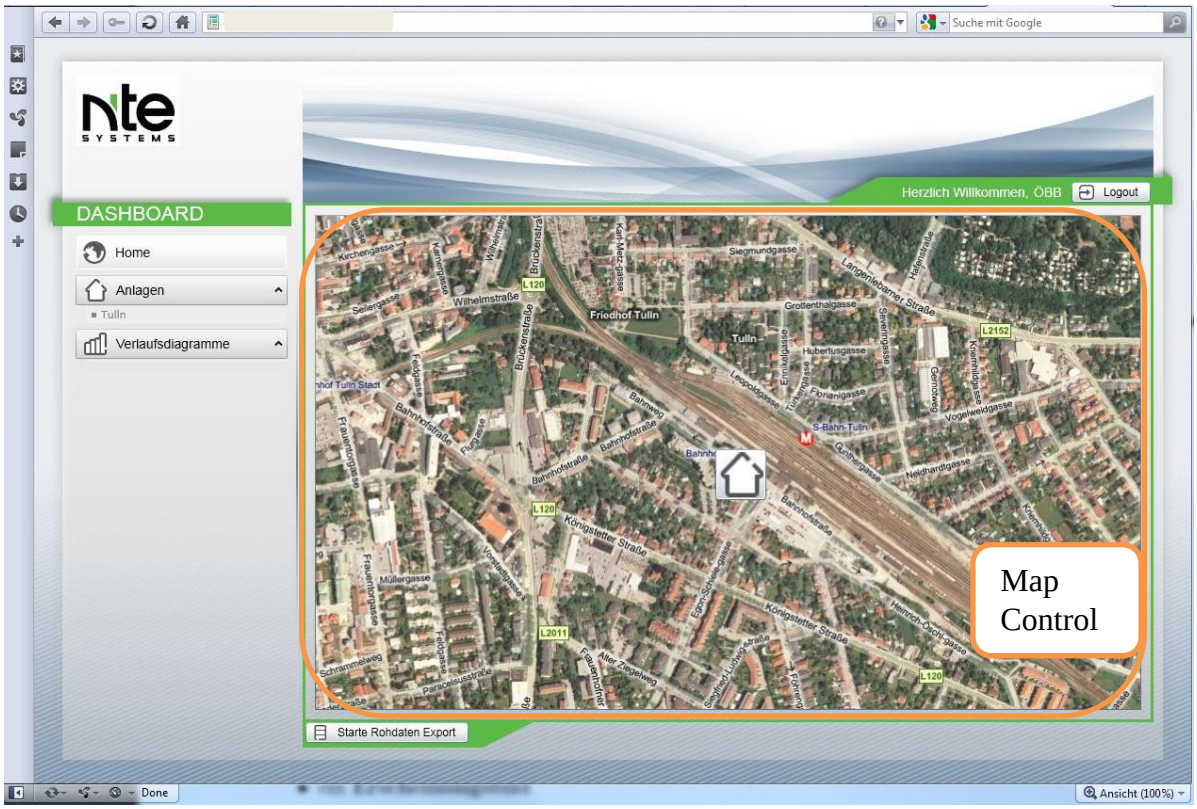


Abbildung 24: Gesamt Integration Map Control Screenshot

In „Abbildung 24: Gesamt Integration Map Control Screenshot“ ist ersichtlich, wie die einzelnen Prism-Module zu einer gesamten Applikation zusammengesetzt wurden. Kernpunkt dieser Abbildung ist das Map Control im zentralen Bereich des Screenshots. Das Map Control visualisiert im geografischen Kontext die Lage der zur Verfügung stehenden Anlagen, dies ist auch der Einstiegspunkt der RIA-Applikation nach einer erfolgreichen Authentifizierung. Im aktuellen Screenshot ist nur eine Anlage für den User zugangsberechtigt, sodass das Map Control auf die Zoomstufe einer Stadt gestellt wird. Bei mehreren Anlagen wird die Zoomstufe so eingestellt, dass alle Anlagen im Map Control beim Einstieg ersichtlich sind.

Bis dato wurde anhand von Screenshots der Aufbau der RIA-Applikation erklärt. Nachfolgende Abbildungen geben einen schematischen Überblick über den Aufbau der realisierten RIA-Applikation.

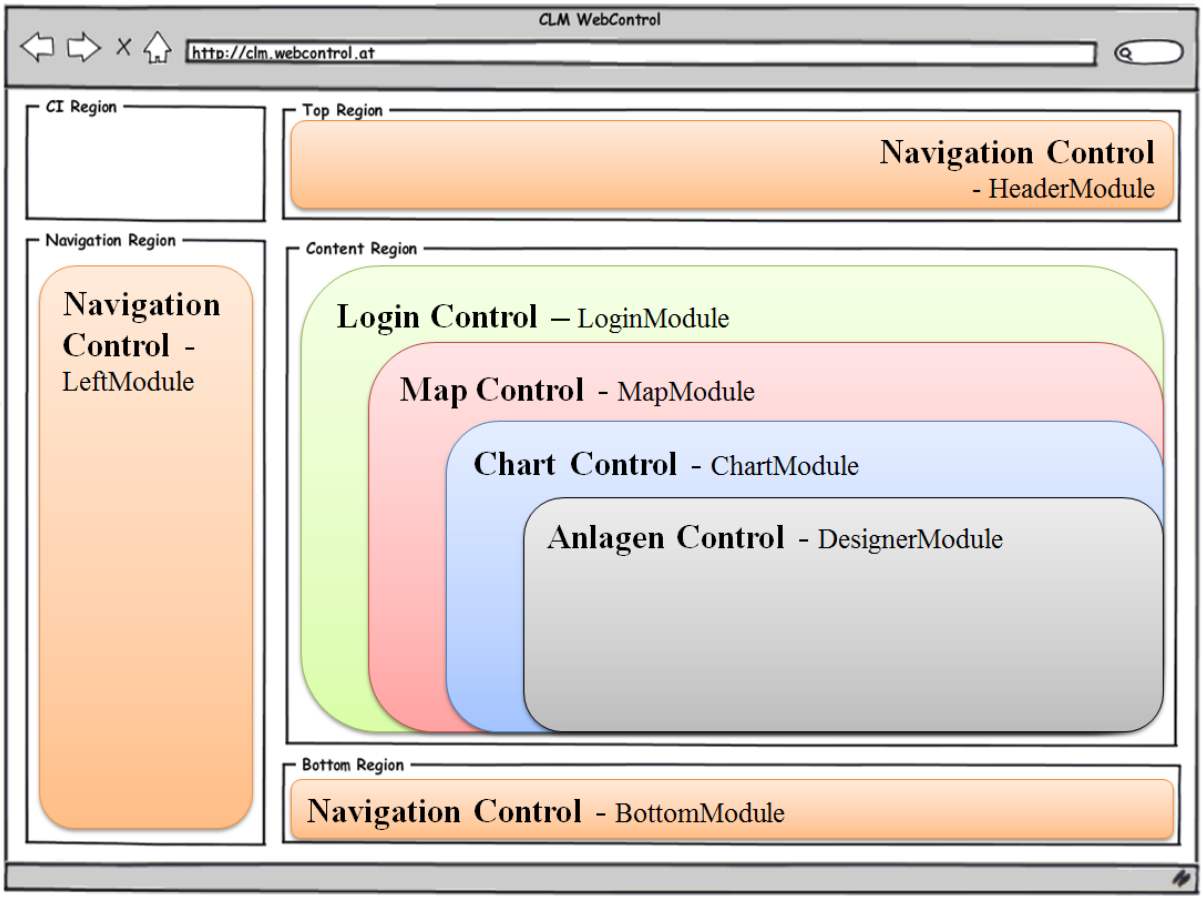


Abbildung 25: RIA-Applikation Control Überblicksdiagramm

Wie bereits in Kapitel „6.2.1 Prism“ erwähnt, wurde die Applikation in Regionen aufgegliedert. Mittels Prism ist es möglich einzelnen Regionen zur Laufzeit Prism-Module hinzuzufügen oder zu entfernen. Der Aufbau von Prism-Modulen ist aus „Abbildung 26: Prism-Modul Detailansicht“ erkennbar. Aus „Abbildung 25: RIA-Applikation Control Überblicksdiagramm“ ist ersichtlich, wie die realisierten Controls und deren Module positioniert sind.

Es ist erkennbar, dass das Navigation Control drei Module besitzt, die wie der Name bereits sagt, zur Navigation benutzt werden. In der „Navigation Region“ ist das Left-Module des Navigation Control integriert. Dieses Modul hat die Aufgabe wichtige Navigationsmöglichkeiten anzubieten. Das Header-Modul ist in der „Top Region“ angesiedelt und ermöglicht dem User sich vom System abzumelden. Im Bottom-Modul das in der „Bottom Region“ integriert ist, wird dem User die Möglichkeit geboten, seine gesamten aufgezeichneten Daten zu exportieren.

Nun folgt die „Content Region“. Wie aus der „Abbildung 25: RIA-Applikation Control Überblicksdiagramm“ sichtbar ist, sind in dieser Region vier verschiedene Controls und deren Module integriert. Programmübergreifend existiert ein Controller Objekt, das je nach Bedarf das entsprechende Modul in die Content Region lädt. Zum Einstieg dieser Applikation ist keines dieser angeführten Module geladen, außer dem Login-Modul. Dieses beinhaltet die Authentifizierungsfunktionalität. Hat der Anwender gültige Zugangsdaten eingegeben, werden bei Akzeptierung vom Server, das Navigation Control und das Map Control geladen, das Login Control wird von der Content Region entfernt, da es nicht mehr benötigt wird.

Des Weiteren ist aus „Abbildung 25: RIA-Applikation Control Überblicksdiagramm“ die CI Region erkennbar. Wie man sieht, hat diese jedoch kein Control das zur Laufzeit geladen wird. Diese Region dient vielmehr als Platzhalter für ein CI Logo. Dieses CI Logo wird zum aktuellen Zeitpunkt der Applikation statisch über das entsprechende CI Theme in die Applikation eingebunden.

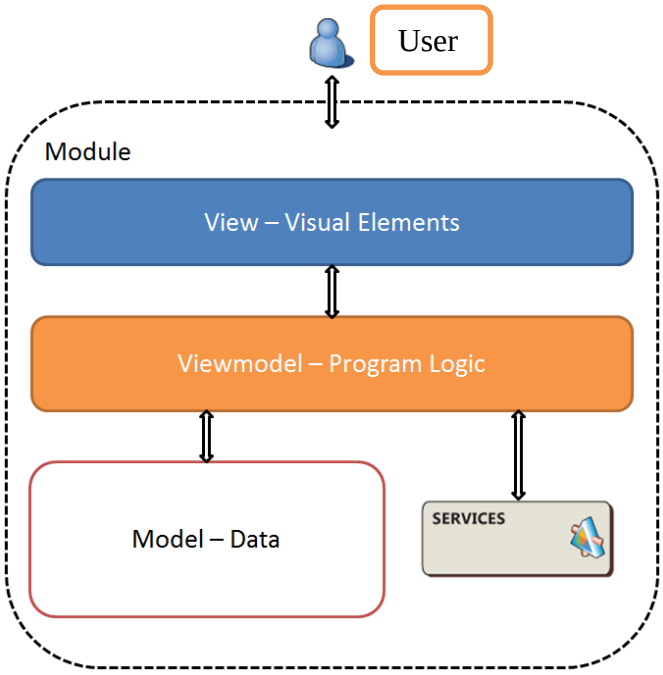


Abbildung 26: Prism-Modul Detailansicht

Aus „Abbildung 26: Prism-Modul Detailansicht“ ist der Aufbau von Prism Modulen ersichtlich. Für den User/Endanwender ist von einem Prism Module nur die Präsentationsschicht (View) sichtbar. Über diese View erfolgt die gesamte Visualisierung der Daten und die Interkation des Users mit dem Modul. Für eine klare Trennung der Logik von der Präsentation ist die View Schicht über das WPF/Silverlight Feature Databinding mit dem Viewmodel verbunden. Dies bedeutet, dass beispielsweise in der View auf einen Button gedrückt wird, die Interpretationslogik sich jedoch in der Viewmodel-Schicht befindet. Die Viewmodel-Schicht wiederum holt sich entweder über Webservice Calls die benötigten Model Instanzen, oder erzeugt diese selbstständig. Eine Referenzimplementierung eines Prism-Moduls mit der entsprechenden View-, Viewmodel- und Model-Schicht ist aus Kapitel „6.2.4 Chart Control“ ersichtlich.

### 6.2.3. Datenmodell

Dieser Abschnitt befasst sich mit dem Datenmodell des CLM WebControls. Wie bereits mehrfach erwähnt, werden die Konfigurationen des Anlagenschemas und der Verlaufsdiagramme im CLM System Designer erstellt. Aufgabe des CLM WebControls ist es, diese zu interpretieren und visualisieren. Einleitend wird exemplarisch die Interpretation eines Anlagenschemas erläutert, dies entspricht dem Model des Model-View-Viewmodel (MVVM) Pattern. Darauf aufbauend werden Klassendiagramme weitere Zusammenhänge zwischen der

Datenabholung am Server und der Aufbereitung in einem Control auf Seiten des Silverlight Clients erläutern.

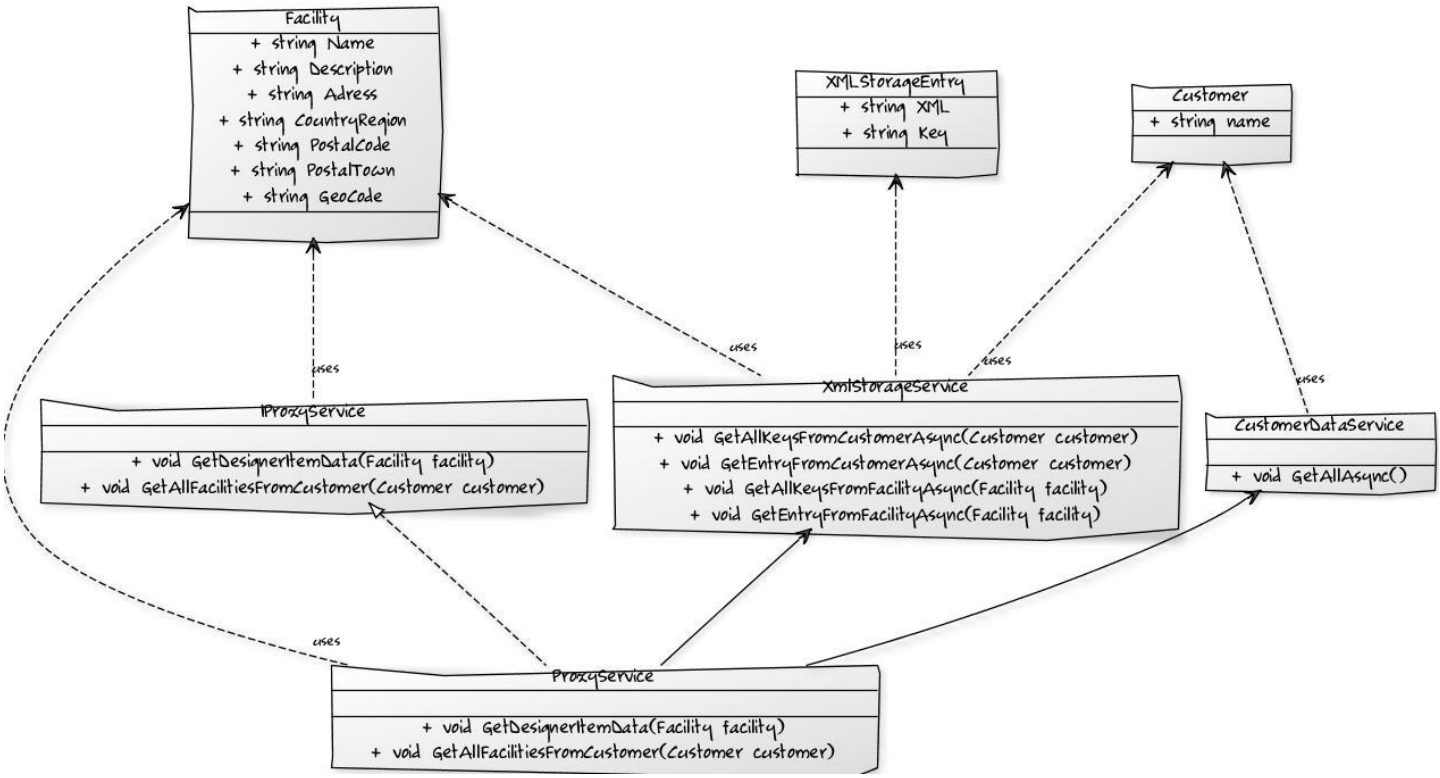


**Abbildung 27: ProxyService Klassendiagramm**

Aus „Abbildung 27: ProxyService Klassendiagramm“ sind folgende Klassen und Interfaces ersichtlich:

- **Customer**: Diese Klasse repräsentiert einen Kunden. Ein Kunde hat die Möglichkeit mehrere Anlagen zu besitzen. Über den Kunden erreicht man die den Anlagen.

- **CustomerDataService**: Diese Webservice Instanz stellt eine Methode zur Ermittlung der zugangsberechtigten Customer dar. Berechtigte Customer werden anhand des Rechte- und Gruppenmanagement auf Seiten des CLM Servers ermittelt.

- **Facility**: Diese Klasse repräsentiert eine Anlage. Eine Anlage ist eindeutig zu einem Kunden zugewiesen.

- **XMLStorageEntry**: Diese Klasse kann über den XMLStorageService (Webservice-Instanz) Informationen am Server abfragen. Die spezifischen Konfigurationen sind in der Eigenschaft XML gespeichert. Die Eigenschaft Key ermöglicht es, verschiedene XML Instanzen zu einer Facility oder einem Customer hinzuzufügen.

- **XMLStorageService**: Dies ist die Webservice-Instanz, welche anhand der WSDL Beschreibung generiert wurde. Mittels Credentials (Berechtigungsnachweis), welche bei jedem Aufruf einer Methode mitgegeben werden, wird die Identität des Anwenders überprüft. Für die Absicherung des Transportes der Credentials wird als Transportprotokoll SSL zwischen dem Client und dem Server verwendet.

- **IProxyService**: Dieses Interface dient zur Kapselung der notwendigen Methoden für die einzelnen Service Calls. Da in Silverlight jegliche Aufrufe von Webservices asynchron erfolgen, ist die Kapselung der Aufrufe für die weitere Vorgehensweise in der Entwicklung sehr hilfreich. Des Weiteren wird durch den Einsatz des Interfaces die Verwendung des integrierten Dependency Injection[56] Framework in Prism ermöglicht, sodass eine Minimierung der Abhängigkeiten innerhalb des CLM WebControls ermöglicht wird.

- **ProxyService**: Diese Klasse repräsentiert die Implementierung des IProxyServices. Sie kapselt die Aufrufe des XMLStorage- und CustomerDataService und reagiert auf die einkommenden asynchronen Antworten entsprechend der implementierten Programmlogik.

Erfolgt aus der Client-Seite der Anwendung eine Authentifizierung, werden einleitend alle befugten Customer ermittelt. Aufbauend auf die Customer werden die dazugehörigen Anlagen (im weiteren Verlauf Facility genannt) angefragt. Diese Informationen sind für die Aufbereitung des Navigation- und des Map-Controls essentiell. Sofern über eine der beiden Navigationsmöglichkeiten, geografisch- oder alt bewährte Navigations-Leiste, eine Auswahl getroffen wird, wird ein neuer Service Call abgerufen. Dieser holt sich für die ausgewählte Facility das entsprechende Facility Schema.

```
public void GetDesignerItemData(Facility facility)
    {
      this.XmlStorageClient.GetFromFacilityAsync("facilitySchema", facility);
    }
```

**Listing 3: ProxyService.cs GetDesignerItemData**

Der Abholaufruf des Anlagenschemas ist aus „Listing 3: ProxyService.cs GetDesignerItemData" ersichtlich. Die Weiterverarbeitung der asynchronen Antwort erfolgt in „Listing 4: ProxyService.cs XmlStorageClient_GetEntryFromFacilityCompleted".

[56] Weitere Informationen zu Dependency Injection findet man unter: http://de.wikipedia.org/wiki/Dependency_Injection

```
void XmlStorageClient_GetEntryFromFacilityCompleted(object sender, GetFromFacilityCompletedEventArgs
e)
    {

      try
      {
        if ((e.Result != null) && (e.Error == null))
        {
          XMLStorageEntry entry = e.Result;
          if (entry.Key == "facilitySchema")
            this.eventAggregator.GetEvent<XMLStorageEntryComplete>().Publish(entry);
        }
        else if (e.Error != null)
        {
          throw e.Error;
        }
      }
      catch (Exception ee)
      {
        this.logger.Log(ee, Category.Exception, Priority.Low);
      }
    }
```

**Listing 4: ProxyService.cs XmlStorageClient_GetEntryFromFacilityCompleted**

Aus „Listing 4: ProxyService.cs XmlStorageClient_GetEntryFromFacilityCompleted" ist ersichtlich, dass beim Eintreffen des gültigen Keys "facilityschema" ein Event veröffentlicht wird, dass das Ergebnisobjekt weiterleitet. Die Erklärung zu Events und der Kommunikation zwischen den einzelnen Modulen mithilfe von Prism erfolgt in Kapitel 6.2.8. Das Ergebnisobjekt dieses Aufrufes ist vom Typ XMLStorageEntry. Der Inhalt der XML Eigenschaft ist aus „Listing 5: Facility Schema XML Konfiguration exemplarische Darstellung" ersichtlich.

```
<FacilitySchema >
  <DesignerItems >
    <DesignerItem >
      <!-- …. -- >
      <Content   Type =" Nte . SolarChecker . Client . Wpf . Controls .
            DesignerItems . NumberDesignerItem ">
        <NumberDesignerItem >
          <Bindings >
            <DesignerItemBinding >
              <Property >Number</ Property >
              <ID >38</ ID >
            </ DesignerItemBinding >
            <!-- …. -- >
            </ Binding >
          <!-- …. -- >
        </ NumberDesignerItem >
      </ Content >
    </ DesignerItem >
    ...
  </ DesignerItems >
</ FacilitySchema >
```

**Listing 5: Facility Schema XML Konfiguration exemplarische Darstellung**

Das Anlagenschema oder auch Facilityschema genannt, welches aus „Listing 5: Facility Schema XML Konfiguration exemplarische Darstellung“ ersichtlich ist, wird im Anlagen Control interpretiert und für die visuelle Aufbereitung der Anlage angewandt.
Nun folgt die Modelbasis des Chart Controls.

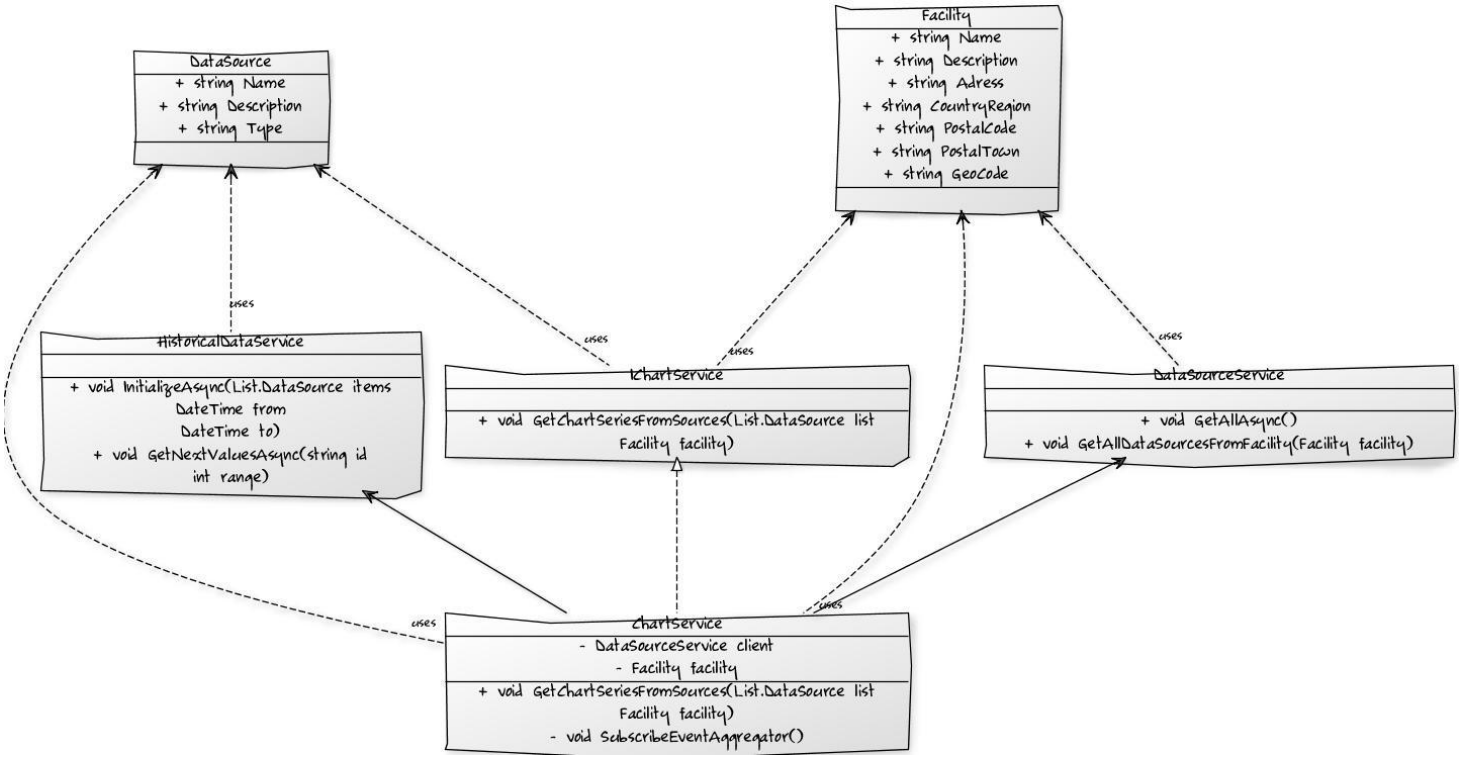


**Abbildung 28: ChartService Klassendiagramm**

Aus „Abbildung 28: ChartService Klassendiagramm“ sind folgende Objekte ersichtlich:

- **Facility**: Diese Klasse repräsentiert eine Anlage.

- **DataSource**: Diese Klasse repräsentiert einen Sensor einer physikalischen Anlage. Diese Klasse beinhaltet wesentliche Informationen des Sensors, beispielsweise den Datentyp eines Sensors.

- **DataSourceService**: Dieser Webservice ermöglicht es, alle DataSources einer Facility für die Weiterverarbeitung abzufragen. Die Konfiguration eines Verlaufsdiagramms, welches auf Seiten der CLM System Designer erstellt wird, beinhaltet nur eine ID, die eine logische Verbindung mit einer DataSource ermöglicht, jedoch nicht das serialisierte Objekt an sich.

- **HistoricalDataService**: Dieser Service ermöglicht es, aufgezeichnete Daten eines Sensors in visualisierbarer Form vom Server zum Client zu transferieren.

- **IChartService**: Dieses Interface dient wiederum zur Spezifikation der Methoden der Service Kapselungsklasse ChartService.

- **ChartService**: Diese Klasse ist die reale Implementierung des IChartService Interfaces. Der Service ist für die Steuerung der vom CLM Server zu ladenden Daten zuständig.

Im weiteren Verlauf wird die Modelbasis des Anlagen Controls für die Livewerte erläutert.

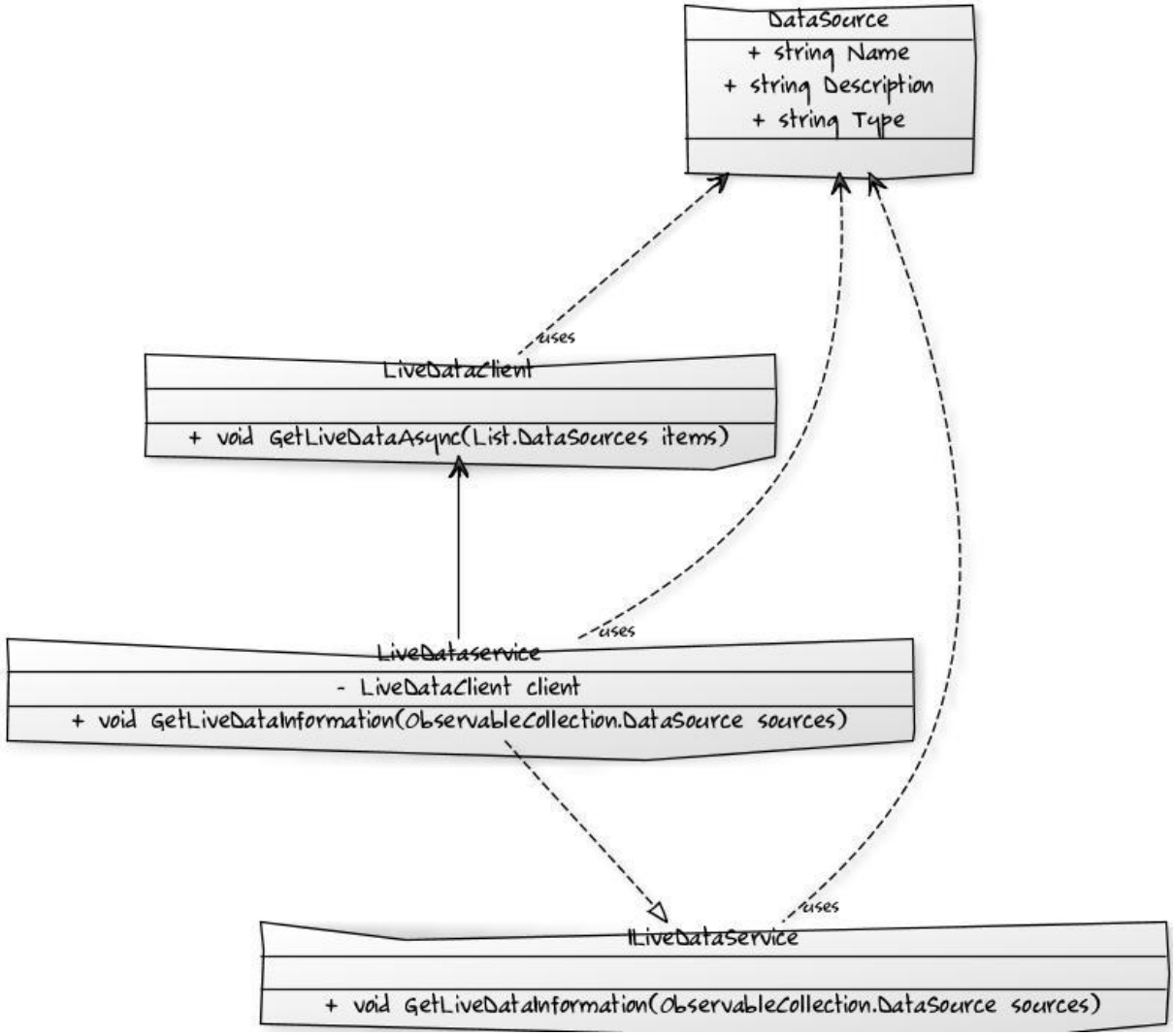


**Abbildung 29: LiveDataService Klassendiagramm**

Die Modelbasis aus „Abbildung 29: LiveDataService Klassendiagramm“ beinhaltet folgende Objekte:

- **DataSource**: Wie bereits erwähnt, repräsentiert diese Klasse einen Datensensor einer physikalischen Anlage.

- **LiveDataClient**: Diese Klasse ermöglicht es, die Methode GetLiveDataAsync aufzurufen.

- **ILiveDataService**: Dieses Interface dient wiederum zur Spezifikation der LiveDataService Klasse.

- **LiveDataService**: Diese Implementierung dient wiederum zur Kapselung der Webservice Funktionalitäten aus der LiveDataClient Klasse.

Nachfolgend werden gesammelt jene Klassen erläutert, die bei den einzelnen Services übertragen werden. Diese werden im CLM System als Data Transfer Objects (kurz DTOs) bezeichnet.

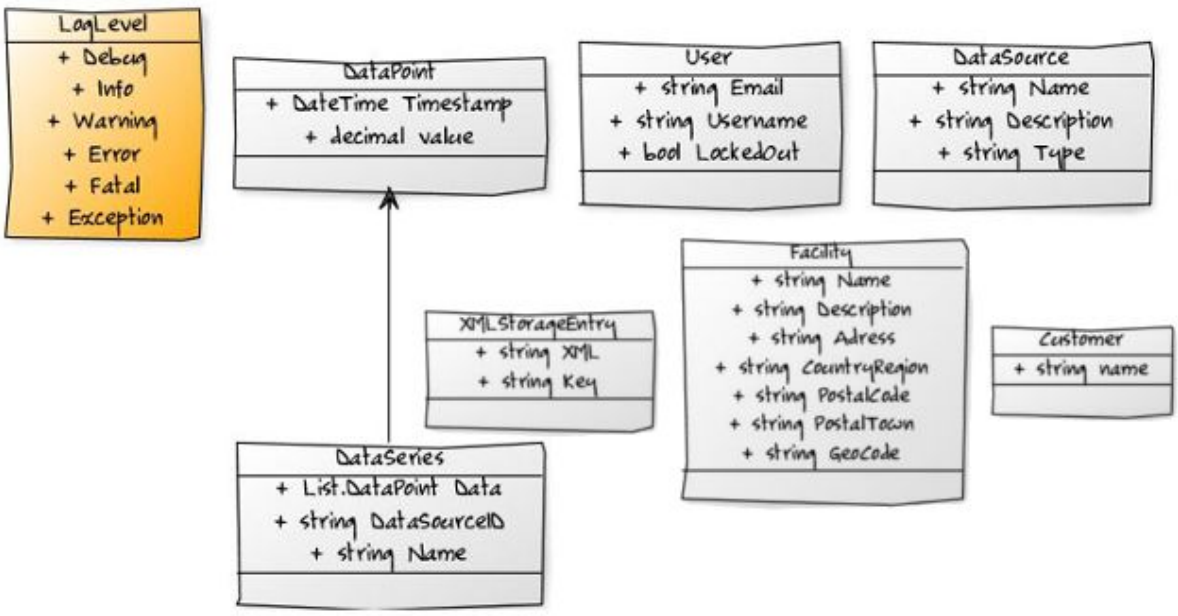


**Abbildung 30: CLM Data Transfer Objects**

Nun werden die einzelnen DTO Klassen aus der „Abbildung 30: CLM Data Transfer Objects“ beschrieben.

- **Customer**: Die Klasse Customer repräsentiert einen realen Kunden, welcher die CLM Plattform verwendet.

- **User**: Ein Kunde (Customer) hat die Möglichkeit über den CLM System Designer eigene User einzurichten. Dieser User kann mit diversen Rechten versehen werden. Beispielsweise obliegt es dem Kunden für einzelne User Administrator- oder Besucherrechte zu vergeben. Jeglicher User kann das CLM WebControl verwenden, jedoch

werden ihm in Abhängigkeit seiner Autorisierung die entsprechenden Funktionalitäten zur Verfügung gestellt. Hat der User beispielsweise keinen Schreibzugriff auf ein Anlagenschema, kann er dieses auch nicht verändern.

- **Facility**: Die Klasse Facility repräsentiert eine Anlage. Eine Anlage kann zu einem Customer hinzugefügt werden. Jedoch können mehrere User einer Anlage zugeordnet werden.

- **DataSource**: Datasources repräsentieren reale Sensoren einer Facility. Diese Klasse ermöglicht es, Sensoren einen leserlichen Namen zu vergeben und diverse Parameter für die Aufzeichnung einzustellen. Beispielsweise das Aufzeichnungsintervall.

- **XMLStorageEntry**: Diese Klasse ermöglicht es, eine XML Konfiguration mit einem eindeutigen Key am Server abzuspeichern.

- **DataSeries**: Für die Darstellung von DataSources in Verlaufsdiagrammen werden am Server bei einer entsprechenden Anfrage Dataseries generiert. Dieses Objekt gewährleistet es, dass der Inhalt in den geläufigen XY Diagrammen darstellbar ist.

- **DataPoint**: Diese Klasse repräsentiert einen Wert einer Datasource zu einem bestimmten Zeitpunkt (Timestamp). Die Liste von DataPoints in der Eigenschaft Data aus der Klasse DataSeries ermöglicht die Darstellung der Werte in den Verlaufsdiagrammen.

- **LogLevel:** Diese Enum-Klasse ermöglicht es, bei der Protokollierung auf der Clientseite entsprechende Informationen beim Transfer zum Server zuzuweisen. Beispielsweise sind Exceptions und Logeinträge mit dem LogLevel „Fatal“ von größter Wichtigkeit, um Fehler beim Anwender zu reproduzieren. Wie aus dem Try Catch Block aus „Listing 4: ProxyService.cs XmlStorageClient_GetEntryFromFacilityCompleted“ ersichtlich, werden jegliche Webservice Fehler protokolliert und zum Server transferiert.

### 6.2.4. Chart Control

Das Chart Control besitzt die Aufgabe, anhand einer speziellen XML-basierten Konfiguration verschiedene Messpunkte zu visualisieren. Die Referenzimplementierung des Chart Controls besteht aus folgenden Dateien:

- ChartControlModule.cs
- ChartViewModel.cs
- ChartView.xaml

- DataSource.cs
- ChartConfigurationModel.cs

```
namespace Nte.Clm.Client.Silverlight.Modules.ChartControl
{
   public class ChartControlModule : IModule
   {
      private IRegionManager regionManager;
      private IUnityContainer container;

      public ChartControlModule(IRegionManager regionManager, IUnityContainer container)
      {
         this.regionManager = regionManager;
         this.container = container;
      }

      public void Initialize()
      {
            ChartViewModel viewModel = this.container.Resolve<ChartViewModel>();
            this.container.RegisterInstance(typeof(ChartViewModel), viewModel);

            IRegion chartregion = this.regionManager.Regions["ContentRegion"];
            chartregion.Add(viewModel.View, "ChartView");        }
   }
}
```

**Listing 6: ChartControlModule.cs**

Die Klasse ChartControlModule stellt eine einfache Implementierung eines Prism-Modules dar. Auf diese Art und Weise ist es möglich, einzelne Klassen als Modul zu definieren, sodass diese von Prism verwendet werden können. Zur Instanziierung einer ChartViewModel-Instanz wird das IUnityContainer[57] Interface zur Hilfe genommen, dies erzeugt mittels Dependency Injection[58] die korrekte Instanz der benötigen Klasse.

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.Collections.Specialized;
using Nte.Clm.Shared.Utils.Reflection;

namespace Nte.Clm.Shared.Dto
{
   [System.Runtime.Serialization.DataContract]
   public partial class DataSource : IEntity
   {
      [System.Runtime.Serialization.DataMember]
      public virtual long Id { get; set; }
```

[57] Weitere Informationen zu IUnityContainer findet man unter: http://msdn.microsoft.com/en-us/library/microsoft.practices.unity.iunitycontainer.aspx

[58] Weitere Informationen zum Entwurfsmuster Dependency Injection findet man unter: http://de.wikipedia.org/wiki/Dependency_Injection

```
        [System.Runtime.Serialization.DataMember]
        public virtual string ReadableName { get; set; }

        [System.Runtime.Serialization.DataMember]
        public virtual string Type { get; set; }
    }
}
```

**Listing 7: DataSource.cs**

Die Klasse DataSource ist eine Klasse, anhand derer einzelne Messpunkte von Anlagen beschrieben werden. Sofern für einen Messpunkt historische Daten existieren, und diese numerisch interpretierbar sind, ist dieser Messpunkt auch für Charts geeignet.

- Die Eigenschaft ID ist eine im System einzigartige ID, sodass DataSources eindeutig zuordenbar sind.
- Die Eigenschaft ReadableName stellt der Klasse DataSource einen leserlichen Namen zu Verfügung.
- Die Eigenschaft Type beschreibt den Datentyp der DataSource.

```
using System;
using System.Collections.Generic;
using Nte.Clm.Shared.Dto;
using System.Windows.Media;
using System.Reflection;

namespace Nte.Clm.Client.Wpf.ServiceManager.Modules.Workspace.Models
{
    public class ChartConfigurationItem : INotifyPropertyChanged
    {
        private String name;
        public String Name { get; set; }

        private String chartType;
        public String ChartType { get; set; }

        private Dictionary<string, Color> idColorlist = new Dictionary<string, Color>();
        public Dictionary<string, Color> IDColorList { get; set; }

        private Dictionary<string, string> idAxisLabel = new Dictionary<string, string>();
        public Dictionary<string, string> IDAxisLabel { get; set; }

        private List<DataSource> selectedDataSources = new List<DataSource>();
        public List<DataSource> SelectedDataSources { get; set; }
    }
}
```

**Listing 8: ChartConfigurationModel.cs**

```
public List<DataSource> SelectedDataSources
    {
      get
      {
        return selectedDataSources;
      }
      set
      {
        if (this.selectedDataSources != value)
        {
          selectedDataSources = value;
          OnPropertyChanged(MethodInfo.GetCurrentMethod());
        }
      }
    }
```

**Listing 9: Implizite OnPropertyChanged Benachrichtigung**

Die Klasse ChartConfigurationModel stellt eine einfache Datenklasse für die Chart-Konfiguration dar. Das Interface INotifyPropertyChanged stellt ein Event und eine Methode, siehe „Listing 9: Implizite OnPropertyChanged Benachrichtigung“, dar, anhand derer man bei Änderungen in der Model Klasse, die höchste Ebene, die View, benachrichtigen kann, dass sich die Daten verändert haben. Dieser Mechanismus ermöglicht es, ohne Kenntnis der View Instanz, die View zu informieren, dass sich etwas verändert hat.

Die einzelnen Eigenschaften (Properties) der ChartConfigurationModel Datenklasse:

- **Name**: Dies ist der Name der Chart-Konfiguration. Dieser wird in der Navigation bei den verschiedenen Chart-Konfigurationen benötigt, um eine Unterscheidungsmöglichkeit zu haben.
- **ChartType**: Der Chart Typ wird als String in die Konfiguration geschrieben, da somit eine größere Unabhängigkeit von möglichen Drittherstellern von Softwarekomponenten erreicht wird. Es wurde für die Chart-Visualisierung auf eine Library eines Drittherstellers zurückgegriffen, welcher bereits viele der gängigen Charttypen (Barchart, LineChart, Columnchart, 3DChart uvm.) unterstützt. Zusätzlich zur Herstellerunabhängigkeit wurde auch für den Austausch via Webservice eine Fehlerquelle minimiert. Da diese Chart-Konfiguration über SOAP vom CLM Server zum CLM ServiceDesigner und zum CLM WebControl transportiert wird, wird diese Konfiguration anhand eines XML Serializers[59] transportfähig aufbereitet. Da es für den .NET XML Serializer diverse Einschränkungen gibt, z.B. öffentliche Getter und Setter Methoden von Properties, und dieser mit Klassen von Drittherstellern Probleme haben könnte, wurde vorbeugend auf einen String Datentyp zurückgegriffen.
- **SelectedDataSources:** Dies ist eine Liste von DataSourcen, welche eine Chart-Konfiguration charakterisieren.

[59] Weitere Informationen zur XmlSerializer Klasse findet man unter: http://msdn.microsoft.com/de-de/library/system.xml.serialization.xmlserializer(VS.80).aspx

- **IDColorList:** Ein Wörterbuch, das als Suchschlüssel die eindeutige ID der DataSource besitzt. Das passende Gegenstück dazu ist ein Farbton, welcher für diese Datasource bei der Visualisierung verwendet werden soll.
- **IDAxisLabel:** Ein Wörterbuch, das als Suchschlüssel die eindeutige ID der DataSource besitzt. Das passende Gegenstück dazu ist die Beschriftung für diese eine DataSource, welche im Frontend visualisiert werden soll.

```
<UserControl xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
       x:Class="Nte.Clm.Client.Silverlight.Modules.ChartControl.CustomControls.ChartControl.ChartControl">
   <Grid x:Name="LayoutRoot">
   <!-- …  -- >
      <Chart  x:Name="chart"
                    Data="{Binding ChartModelData}"
                    View="{Binding ChartModelView}">
      </Chart>
   <!-- …  -- >
   </Grid>
</UserControl>
```

**Listing 10: ChartView.xaml**

Die ChartView Klasse ist exemplarisch in „Listing 10: ChartView.xaml“ implementiert. Kernelement der View Klasse ist die Chart Komponente; diese ist anhand des <Chart /> gekennzeichnet. Die Klasse Chart besitzt zwei Eigenschaften die interessant sind. Dies sind die Data und die View Eigenschaft. Diese beiden Eigenschaften werden auch über das XAML typische Data Binding[60] auf zwei Eigenschaften zum DataContext[61] des UserControls verbunden. Die Data Eigenschaft ist dafür verantwortlich, dass das Chart Datenpunkte visualisieren kann. Die View Eigenschaft ist für die Beschreibung und das Aussehen der Datenpunkte verantwortlich.

```
public ChartViewModel(
        Nte.Clm.Client.Silverlight.Modules.ChartControl.CustomControls.ChartControl.ChartControl view)
    {
        this.ChartModelData = new C1.Silverlight.Chart.ChartData();
        this.ChartModelView = new C1.Silverlight.Chart.ChartView();

        this.View = view;
        this.View.DataContext = this;
    }
```

**Listing 11: ChartViewModel.cs Constructor**

Aus „Listing 11: ChartViewModel.cs Constructor“ ist ersichtlich, dass der ChartView Instanz eine ChartViewModel Instanz als DataContext zugewiesen wird. Wie man erkennen kann, ist dies einer der kritischen Punkte beim Einsatz der XAML Technologie. Aus Unachtsamkeit

[60] Weitere Informationen zur Binding Klasse im .net Framework findet man unter: http://msdn.microsoft.com/en-us/library/system.windows.data.binding.aspx

[61] Weitere Informationen zum DataContext Eigenschaft(Property) findet man unter: http://msdn.microsoft.com/en-us/library/system.windows.frameworkelement.datacontext.aspx

oder beim Refactoring von diversen Eigenschaften (Properties) kann es passieren, dass diese zwar im ViewModel einer Klasse geändert werden, die dazugehörige View aber nicht abgeändert wird. Ergo kann ein funktionierendes Binding fehlschlagen, da es die entsprechende Eigenschaft in der ViewModel Klasse nicht mehr gibt. Das Fehlschlagen dieses Bindings ist beim Endanwender nicht ersichtlich, da dies nicht als Fehler visualisiert wird. Es führt jedoch zu einem Fehlverhalten der Software, das nicht zwingend mit einem Absturz der Software einhergeht, und somit auch schwer nachvollzieh- und auffindbar ist.

```
public void UpdateChartInformation(ChartConfigurationItem items)
    {
        ChartConfigurationItem parameter = items;
        List<DataSource> list = parameter.SelectedDataSources;
        this.chartconfig = parameter;

       // …
        foreach (var key in this.chartconfig.IDAxisLabel.Keys)
        {
           var ax = new Axis();
           //…
           ax.Title = this.chartconfig.IDAxisLabel[key];
           this.ChartModelView.Axes.Add(ax);
        }
        // …
        OnPropertyChanged(„ChartModelView“);
    }
```

**Listing 12: ChartViewModel.cs UpdateChartInformation**

Aus „Listing 12: ChartViewModel.cs UpdateChartInformation“ ist ein Teil der Logik ersichtlich, die beim Laden einer neuen Chart-Konfiguration abgearbeitet wird. Unter anderem ein Teilbereich für die Achsenbeschriftung, welcher anhand des Data Binding an die Benutzeroberfläche propagiert wird.

Das Listing der ChartViewModel Klasse beinhaltet nur einen Bruchteil dessen, was die reale Implementierung dieser ViewModel Klasse beinhaltet. Zusätzlich zur Interpretation der ChartConfiguration Klasse, beinhaltet diese Klasse auch alle notwendigen Methoden und Commands, anhand derer man eine neues Zeitintervall, oder aber einen Datenexport veranlassen kann.

Die Integration dieses ChartModules und die Kommunikation dieses Moduls mit anderen Teilstücken der RIA-Applikation werden im Kapitel„6.2.8 Gesamt Integration“ behandelt.

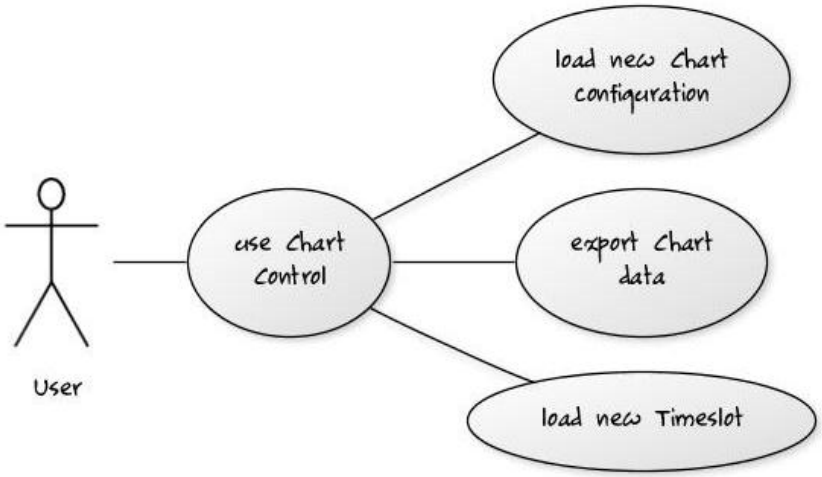


Abbildung 31: Chart Control Usecases

Die „Abbildung 31: Chart Control Usecases“ veranschaulicht die wichtigsten Usecases welche anhand des Chart Controls realisiert wurden.

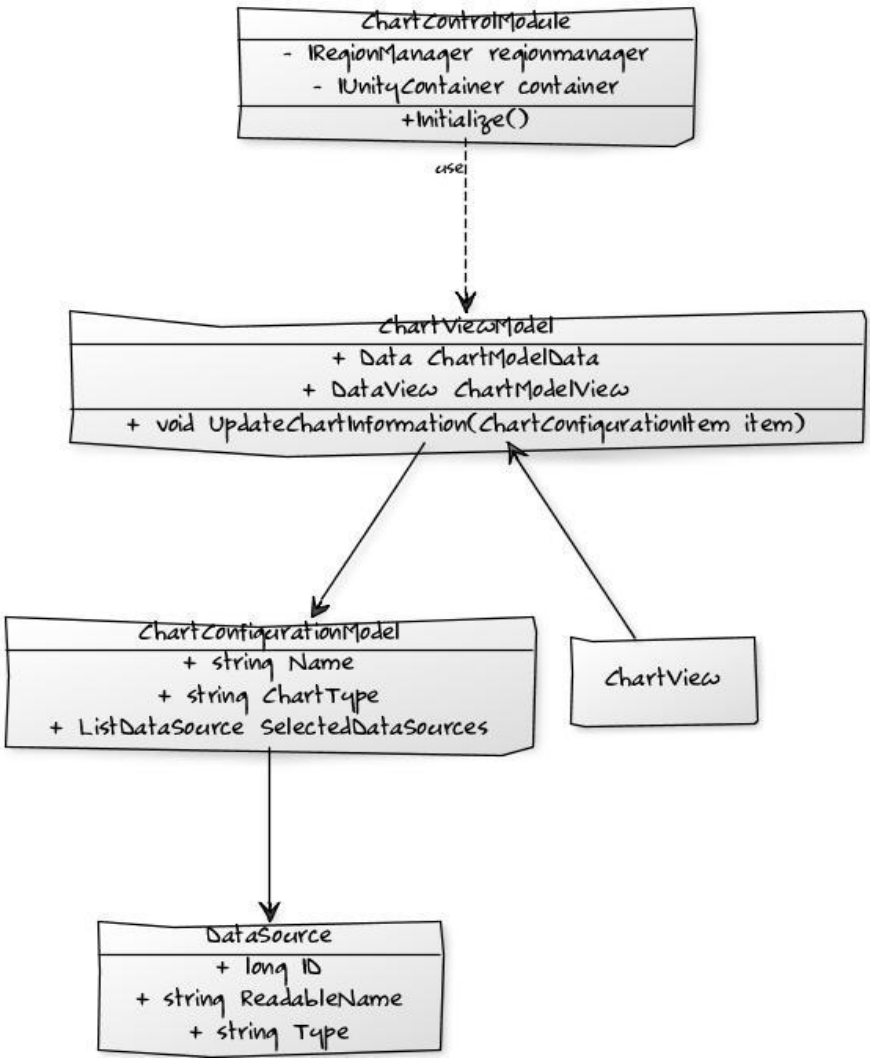


Abbildung 32: Chart Control ClassDiagramm

Die „Abbildung 32: Chart Control ClassDiagramm“ veranschaulicht eine grobe Visualisierung des Klassendiagramms der Chart Control Komponenten.


**Abbildung 33: Chart Control Screenshot**

Die „Abbildung 33: Chart Control Screenshot“ zeigt die aktuelle Implementierung des Chart Control.

## 6.2.5. Anlagen Control

Dieser Abschnitt beschäftigt sich mit der Realisierung des Anlagen Controls. Dieser Baustein dient zur Visualisierung der Live-Daten in Verbindung mit grafischen Symbolen. Im nachfolgenden Abschnitt gibt es Bereiche, die sich inhaltlich mit (Konrad, 2010) überschneiden. Dieser beschäftigt sich mit der Erstellung der Konfiguration des Anlagenschemas seitens des CLM Service Designers.

Die Implementierung des Anlagen Controls besteht aus folgenden Dateien:

- DesignerControlModule.cs
- DesignerViewModel.cs
- DesignerView.xaml
- DesignerModelBase.cs
- TextDesignerModel.cs

- LiveDataWorker.cs

Wie bereits im vorherigen Abschnitt angeführt, wird die Datei DesignerControlModule für Prism benötigt, damit der Anlagen Control Baustein ins Gesamtsystem integriert werden kann.

```
<UserControl x:Class="Nte.Clm.Client.Silverlight.Modules.DesignerItemControl.Views.DesignerItemView"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation">
  <Canvas x:Name="DesignerItemCanvas" Children="{Binding Items }">
  </Canvas>
</UserControl>
```

**Listing 13: DesignerView.xaml**

Das „Listing 13: DesignerView.xaml“ besteht im Wesentlichen aus einem Canvas[62] Control. Ein Canvas Control besitzt die Möglichkeit, Objekte frei zu positionieren. Diese Objekte werden zur Laufzeit über ein Konfigurationsschema, welches zentral am CLM Server gespeichert ist, geladen. Die zu visualisierenden Objekte werden aufbereitet und in eine Liste gegeben. Diese Liste ist wiederum über das Data Binding mit der View verbunden.

```
private DesignerModelBase DeserializeItem(String xml, String type)
      {
         if (String.Compare(type, typeof(TextDesignerModel).ToString()) == 0)
         {
            Type t = typeof(TextDesignerModel);
            XmlSerializer xs = new XmlSerializer(t);
            byte[] buf = System.Text.Encoding.UTF8.GetBytes(xml);
            MemoryStream mem = new MemoryStream(buf);
            TextDesignerModel item = (TextDesignerModel)xs.Deserialize(mem);
            return item;
         }
         // ...
      }
```

**Listing 14: DesignerViewModel.cs TextDesignerModel Deserialization**

Die Hauptaufgabe der DesignerViewModel Klasse ist es, eine Anlagenkonfiguration zu interpretieren und die entsprechenden grafischen Symbole zu erzeugen. Jedes grafische Symbol, welches in einer Anlagen-Konfiguration verwendet werden kann, ist von DesignerModelBase abgeleitet. Die Klasse DesignerModelBase besitzt Properties, die jedes Objekt zur Positionierung in der DesignerView Klasse benötigt. Die Klasse TextDesignerModel ist eine Spezialisierung der DesignerModelBase Klasse, diese erweitert die Base Klasse um die Eigenschaft Text. Jene Eigenschaft kann zur Laufzeit mit Werten verbunden und grafisch für den Endanwender aufbereitet werden.

[62] Weitere Informationen zum Canvas Control findet man unter: http://msdn.microsoft.com/en-us/library/system.windows.controls.canvas.aspx

```
<DataTemplate x:Name="TextDesignerModel"
              x:Key="TextDesignerModel">
    <Viewbox Stretch="Fill"
             StretchDirection="Both">
      <Label Content="{Binding Text }"
                  Foreground="{Binding Colour }"
                  FontFamily="{Binding Font }"
                  HorizontalContentAlignment="Stretch"
                  VerticalContentAlignment="Stretch">
      </Label>
    </Viewbox>
  </DataTemplate>
```

**Listing 15: DataTemplate**

Die grafische Aufbereitung der einzelnen Symbole erfolgt über DataTemplates[63]. Da die Konfiguration des Anlagenschemas anhand des CLM ServiceDesigneres erfolgt und das Schema äquivalent im Web dargestellt werden soll, muss vor allem bei den DataTemplates darauf geachtet werden, dass die grafischen Symbole und Objekte zur Definition des Templates auch in Silverlight existieren. Ansonsten kann es passieren, dass der Endkunde das grafische Symbol nicht mehr wiedererkennt. Das „Listing 15: DataTemplate“ veranschaulicht ein Silverlight und WPF kompatibles DataTemplate für die Klasse TextDesignerModel. Nun muss das aufbereitete Konfigurationsobjekt in die entsprechende Liste für die View gelegt und auch schon automatisch visualisiert werden.

Nun wurden, ausgehend vom Prism Modul, die View und das ViewModel des Anlagen Controls betrachtet, exemplarisch ein Anlagen-Konfigurationsobjekt deserialisiert und mit einem DataTemplate versehen. Jetzt fehlt noch die Verbindung der Objekte mit Live-Daten.

```
public LiveDataWorker(ILiveDataService service)
    {
      this.service = service;

      this.timer = new DispatcherTimer();
      timer.Tick += new EventHandler(updateLiveData);
      timer.Interval = new TimeSpan(0,0,5);
      timer.Start();

      this.Items = new List<DataSource>();
    }
```

**Listing 16: LiveDataWorker.cs Constructor**

Die Aktualisierung der Live-Daten ist nicht Event-getrieben sondern benutzt einen Polling-Mechanismus, der beim Server abfragt, ob für ein bestimmtes Set von Messpunkten neue Informationen vorhanden sind. Dieser Polling-Mechanismus wird mittels des DispatcherTimers[64] realisiert. Aus „Listing 16: LiveDataWorker.cs Constructor“ ist eine exemplarische

[63] Weitere Informationen zu DataTemplate findet man unter: http://msdn.microsoft.com/en-us/library/ms742521.aspx

[64] Weitere Informationen zur DispatcherTimer Klasse findet man unter: http://msdn.microsoft.com/de-de/library/system.windows.threading.dispatchertimer.aspx

DispatcherTimer-Instanzierung ersichtlich, welche im 5-Sekunden-Takt die Methode „updateLiveData“ aufruft. UpdateLiveData wiederum holt für die Elemente der Items Liste den letzten Wert.

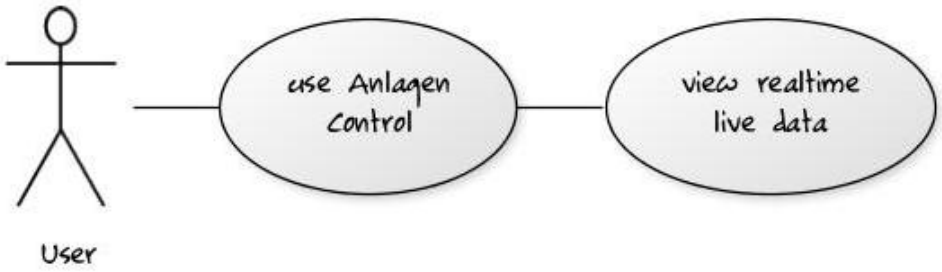


**Abbildung 34: Anlagen Control Usecase**

Die „Abbildung 34: Anlagen Control Usecases“ veranschaulicht den Usecase, welcher anhand des Chart Controls realisiert wurde.

Das Anlagen Control wurde für zukünftige Anforderungen geplant und implementiert, sodass es ohne größere Umbauarbeiten (Refactoring) möglich ist, weitere Funktionalitäten hinzuzufügen.

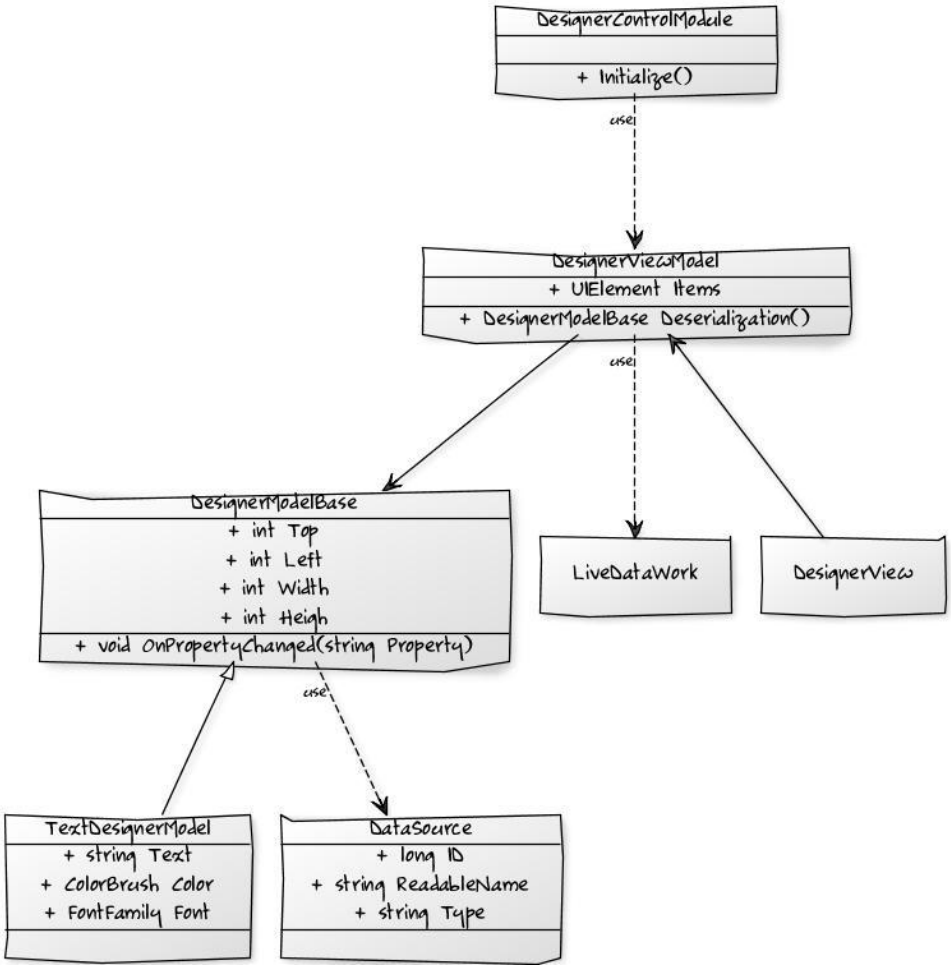


**Abbildung 35: Anlagen Control ClassDiagramm**

Die „Abbildung 35: Anlagen Control ClassDiagramm“ veranschaulicht in einer Skizze das Klassendiagramm des Anlagen Controls. Aufgrund der Reduktion der Komplexität zur besseren Lesbarkeit, wurde auf das Anführen von Methoden, Properties und Fields verzichtet.


Abbildung 36: Anlagen Control Screenshot

Die „Abbildung 36: Anlagen Control Screenshot“ veranschaulicht den aktuellen Stand des Anlagen Controls zum Zeitpunkt dieser Masterarbeit. Wie man sehen kann, ist der Einsatz von grafischen Symbolen nicht nur auf Texte limitiert, sondern es können auch Bilder, und bereits erste komplexere Objekte wie z.B. das Circular Gauge Control verwendet werden.

### 6.2.6. Map Control

Das Map Control ist als erweiterte Navigationshilfe für den Endanwender im CLM WebControl angedacht. Kernpunkt dieses Bausteins ist es, dass alle Anlagen eines Kunden in eine Weltkarte eingetragen werden und bei der Aktivierung mittels Click auf einer Anlage, alle essentiellen Daten dieser Anlage geladen werden. Dies wären z.B. das Anlagenschema mit den Live-Daten oder die Chart-Konfigurationen dieser einen Anlage.

Dieses Control besteht aus den Dateien:

- MapControlModule.cs
- MapViewModel.cs
- MapView.xaml

Wiederum wird die Klasse MapControlModule für das Prism Framework benötigt.

```
<UserControl xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
          x:Class="Nte.Clm.Client.Silverlight.Modules.MapControl.Views.MapView">
      <Maps     ShowTools="False"
                x:Name="c1map"
                Zoom="{Binding Zoom, Mode=TwoWay}"
                TilesMode="DeepZoom"
                Source="{Binding Source}">

         <c1maps:C1Maps.Layers>
            <MapItemsLayer x:Name="ItemsLayer" Items="{Binding Items}"
      </Maps>
</UserControl>
```

**Listing 17: MapView.xaml**

Das „Listing 17: MapView.xaml“ verwendet ein Maps Control eines Drittherstellers. Es werden die Zoomstufe, die Datenquelle der Weltkarte und die zu visualisierenden Objekte mittels DataBinding an das UI gebunden.

```
public void GeoCodeRe-
sponse(Nte.Clm.Client.Silverlight.Infrastructure.ProxyLibrary.ClientProxyGeocodeService.GeocodeResponse
parameter)
      {
         // …
         Point pt = new Point();
         pt.X = parameter.Results[0].BestView.Northeast.Longitude;
         pt.Y = parameter.Results[0].BestView.Northeast.Latitude;
         this.View.c1map.Center = pt;
         this.Zoom = this.zoom_country;
         CreateNewLayerItem();
      }
```

**Listing 18: MapViewModel.cs GeoCodeResponse**

Als nächstes stellt sich die Frage, wie kommt diese Control jedoch zu den Daten, die benötigt werden, damit die Anlagen auf der richtigen Position der Weltkarte positioniert werden können. Es wurde definiert, dass beim Laden des Map Controls folgende Informationen vorhanden sein müssen:

- Ein Set von Elementen, die eine Anlage beschreiben.

Eine Anlage wird durch den Namen und die Adresse beschrieben. Jene Adresse dient als Hilfsmittel für die Positionierung auf der Weltkarte. Im Sinne des Future Internet (Service Web 3.0 Video, 2009) Gedankens wurde auf einen existierenden Webservice zurückgegriffen, der eine reale Adresse in Geokoordinaten transformieren kann. Das Ergebnis des Webservice wird in „Listing 18: MapViewModel.cs GeoCodeResponse“ dahingehend verwendet, dass

1. eine Anlage richtig auf der Weltkarte positioniert wird und
2. sie mit einer Logik versehen wird.

Aufbauend darauf, kann der Endanwender durch Aktivieren der Logik, das Laden von weiteren Anlagedaten aktivieren. Diese Daten werden unter anderem im Navigation Control benötigt. Dazu wird im nächsten Abschnitt mehr erläutert.


**Abbildung 37: Map Control Usecase**

Die „Abbildung 37: Map Control Usecase“ visualisiert den Usecase des Map Controls, welcher zum Zeitpunkt der Masterarbeit implementiert wurde. Fortführend könnte man dieses Control noch für weitere Anwenderszenarien einsetzen. Es wäre durchaus realisierbar, dass z.B. bei einem Mouse Over Effekt, starre Kennzahlen der Anlage visualisiert werden, sodass der Anwender direkt sieht ob es sich lohnt die Anlage genauer zu betrachten, oder aber mittels eines Blickes erkennen kann, dass alles in Ordnung ist.

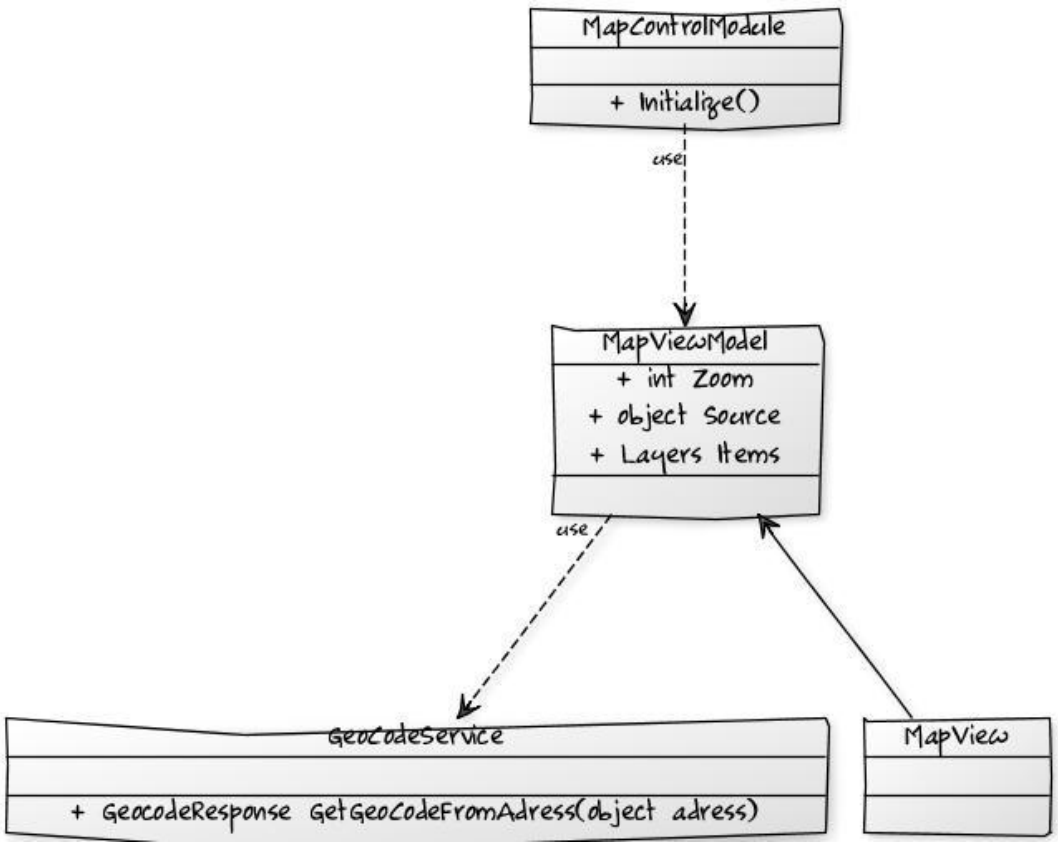


**Abbildung 38: Map Control ClassDiagramm**

Aus der „Abbildung 38: Map Control ClassDiagramm“ ist ein grober Überblick des Map Control in Form eines Klassendiagramms ersichtlich.


**Abbildung 39: Map Control Screenshot**

Die „Abbildung 39: Map Control Screenshot“ visualisiert den finalen Stand des Map Controls zum Zeitpunkt dieser Masterarbeit.

## 6.2.7. Navigation Control

Das Navigation Control besitzt die Aufgabe, dem Endanwender verschiedene Navigationsmöglichkeiten anzubieten. Dem Endanwender soll es möglich sein, zwischen seinen Anlagen zu navigieren, anlagenspezifische Charts zu öffnen, oder die Live-Daten einer Anlage einzusehen.

Das Navigation Control besteht aus folgenden Dateien:

- NavigationControlModule.cs
- NavigationViewModel.cs
- NavigationView.xaml

```
<UserControl xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        x:Class="Nte.Clm.Client.Silverlight.Modules.NavigationControl.Views.NavigationView">

        <!-- Map -->
        <StackPanel Orientation="Vertical">
          <Button  prism:Click.Command="{Binding MapCommand, Mode=OneWay}"/>
        </StackPanel>

        <!-- Facility -->
        <StackPanel Orientation="Vertical">
          <Expander >
            <ListBox x:Name="FacilityList"
                  ItemsSource="{Binding FacilityList, Mode=OneWay}"
                  SelectionMode="Single"/>
          </ Expander>
        </StackPanel>

        <!-- Charts -->
        <StackPanel Orientation="Vertical">
          <Expander >
              <ListBox x:Name="ChartList"
                    ItemsSource="{Binding ChartList, Mode=OneWay}"
                    SelectionMode="Single"/>
          </ Expander>
        </StackPanel>
</UserControl>
```

**Listing 19: NavigationView.xaml**

Aus „Listing 19: NavigationView.xaml“ ist ersichtlich, dass das Navigationselement sehr stark in Verbindung mit den zuvor präsentierten Controls steht. Zum einen gibt es ein grafisches Element für das Map Control, welches auf das MapCommand gebunden wird. Zum anderen gibt es diverse Listen, welche für Anlagen und Chart zur Verfügung stehen.

Nun zum MapCommand. Sofern dieses ausgelöst wird, wird „Listing 20: NavigationViewModel.cs MapCommand Executed“ ausgeführt.

```
private void MapCommand_Executed(object o)
      {

         this.CheckModulesIfInitialized();

         IRegion region = this.regionManager.Regions["ContentRegion"];

         DesignerItemViewModel anlagenvm =
                  this.container.Resolve(typeof(DesignerItemViewModel)) as DesignerItemViewModel;
         if (region.Views.Contains(anlagenvm.View) && region.ActiveViews.Contains(anlagenvm.View))
         {
            region.Deactivate(anlagenvm.View);
         }

         ChartViewModel chartvm = this.container.Resolve(typeof(ChartViewModel)) as ChartViewModel;
         if (region.Views.Contains(chartvm.View) && region.ActiveViews.Contains(chartvm.View))
         {
            region.Deactivate(chartvm.View);
         }

         MapViewModel mapvm = this.container.Resolve(typeof(MapViewModel)) as MapViewModel;
         if (region.Views.Contains(mapvm.View) && !region.ActiveViews.Contains(mapvm.View))
         {
            region.Activate(mapvm.View);
         }

      }
```

**Listing 20: NavigationViewModel.cs MapCommand Executed**

Innerhalb der MapCommand_Executed Methode werden im Content Bereich die entsprechenden Vorbereitungen getroffen, sodass das MapControlModule geladen werden kann. Sobald diese abgeschlossen sind, wird jegliches andere Modul in diesem Bereich deaktiviert und nur mehr das Map Control visualisiert.

Ob es sich im weiteren Verlauf der Interaktion des Users um das Laden einer Chart Konfiguration oder die Anlagenansicht handelt, ist nicht von Belang, da die Vorgehensweise der Aktivierung und Deaktivierung des entsprechenden Moduls immer auf dieselbe Art und Weise erfolgt.

Laden der Anlagedaten: Wählt der Endanwender eine Anlage aus, werden im Hintergrund diverse Aktionen gestartet. Zum einen wird für diese eine Anlage das Anlagenschema vom CLM Server geladen und für die Visualisierung aufbereitet. Zum anderen wird die Chart-Konfiguration, passend zu dieser Anlage, vom Server geholt und in der Navigationsliste für Charts visualisiert.

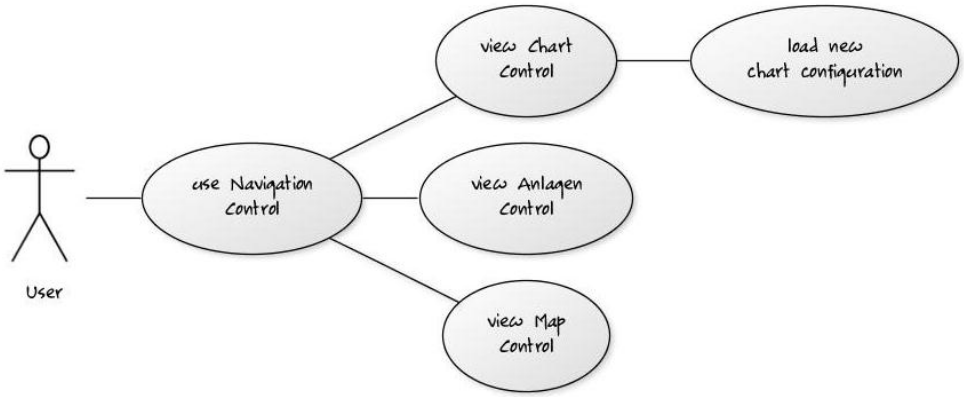


Abbildung 40: Navigation Control Usecase

Die „Abbildung 40: Navigation Control “ veranschaulicht die Usecases, welche in das CLM WebControl integriert wurden.

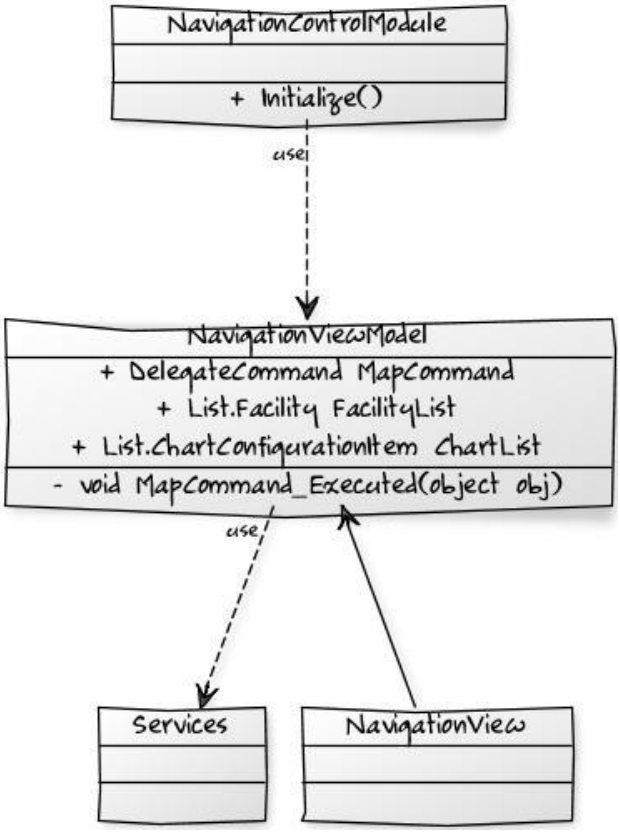


Abbildung 41: Navigation Control ClassDiagramm

Die „Abbildung 41: Navigation Control ClassDiagramm“ veranschaulicht das Klassendiagramm des Navigations Control Modules.


**Abbildung 42: Navigation Control Screenshot**

Die „Abbildung 42: Navigation Control Screenshot“ visualisiert das Navigation Control zum Zeitpunkt dieser Masterarbeit.

### 6.2.8. Gesamtintegration

In Folge geht es um die gesamte Integration der Module in das CLM WebControl. In den zuvor angeführten Abschnitten werden exemplarisch diverse Module für das CLM WebControl erstellt. Der Datenaustausch zwischen den einzelnen Modulen wird nun erläutert.

Hierfür sind Events bzw. der Event Aggregator[65] des Prism Frameworks eine ideale Lösung.

[65] Weitere Informationen zur Event Aggregator Klasse findet man unter: http://msdn.microsoft.com/en-us/library/ff649187.aspx

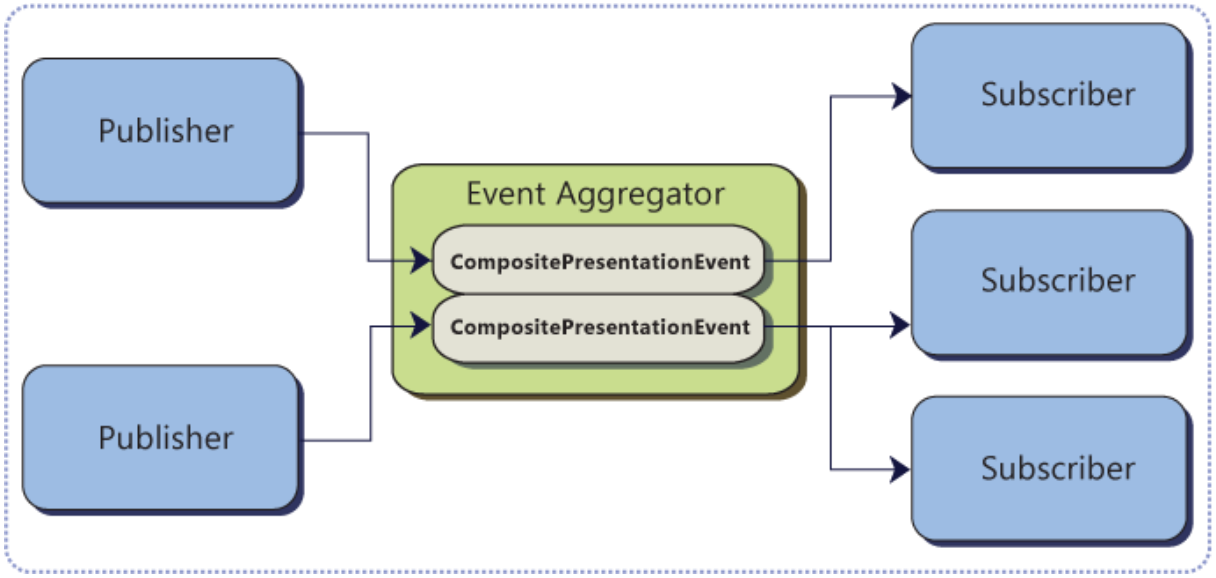


Abbildung 43: Event Aggregator Funktionsweise (Event Aggregator)

In „Abbildung 43: Event Aggregator Funktionsweise “ wird schematisch die Funktionsweise des EventAggregators dargestellt. Einzelne Objekte können den EventAggregator verwenden, um Events zu veröffentlichen (Publisher). Jedes Objekt, das sich auf bestimmte Events registriert (Subscriber) hat, wird bei diesen Events informiert.

Exemplarisch wird ein Event wie folgt definiert:

```
public class UpdateChartInformationEvent: CompositePresentationEvent<ChartConfigurationItem> {}
```

Listing 21: UpdateChartInformationEvent.cs

Das UpdateChartInformationEvent besitzt als Parameter eine Instanz vom Typ ChartConfigurationItem, welche für die Chart-Visualisierung die notwendigen Parameter beinhaltet. Veröffentlicht (gepublished) wird das Event im NavigationViewModel sobald der Endanwender eine Chart-Konfiguration auswählt.

```
void ChartList_SelectionChanged(object sender, SelectionChangedEventArgs e)
      {
        if (e.AddedItems.Count > 0)
        {
          ChartConfigurationItem conf = e.AddedItems[0] as ChartConfigurationItem;

         this.eventaggregator.GetEvent<UpdateChartInformationEvent>().Publish(conf);
        }
      }
```

Listing 22: NavigationViewModel.cs Event Publisher

Damit das Event jedoch verwertet werden kann, muss es auch einen Subscriber dafür geben. Dieser befindet sich in der ChartViewModel Klasse, die sich im Chart Control befindet.

```
private void SubscribeAggregator()
      {
         try
         {
            this.eventAggregator.GetEvent<UpdateChartInformationEvent>().Subscribe(UpdateChartInformation);
         }
         catch (Exception ex)
         {
            // …
         }
      }
```

**Listing 23: ChartViewModel.cs Event Subscription**

Nun wurde eine Verbindung ausgehend vom Navigation Control zum Chart Control definiert. Die Auswertung des Events erfolgt in der Methode UpdateChartInformation, welche sich im „Listing 12: ChartViewModel.cs UpdateChartInformation“ wiederspiegelt.

Es wurden der Datenaustausch zwischen den einzelnen Prism Modulen besprochen, die einzelnen Module und deren Views, ViewModels und Module, jedoch fehlt noch zur Gänze der Model Bereich.

Der Model Bereich wurde größtenteils anhand von WebServices realisiert, welche durch asynchrone Aufrufe das ViewModel mit Daten versorgen. Exemplarisch wird die Handhabung des Webservices anhand des LiveDataService veranschaulicht.

Wie im „Listing 16: LiveDataWorker.cs Constructor“ ersichtlich ist, wird in einem bestimmten Zeitintervall die Methode updateLiveData aufgerufen.

Nachfolgend wird die Implementierung der updateLiveData Methode angeführt.

```
private void updateLiveData(object sender, EventArgs e)
      {
         if (this.Items.Count > 0)
         {
            ObservableCollection<DataSource> list = new ObservableCollection<DataSource>(this.Items);
            if (list.Count > 0)
               this.service.GetLiveDataInformation(list);
         }
      }
```

**Listing 24: LiveDataWorker.cs updateLiveData**

Sofern in der Instanz von LiveDataWorker in der Eigenschaft Items Elemente enthalten sind, werden diese an den LiveDataService übergeben. Der LiveDataService stellt die Methode GetLiveDataInformation zur Verfügung, welche eine Liste von DataSources als Parameter erwartet. Die Implementierung dieser Methode sieht wie folgt aus:

```
public void GetLiveDataInformation(ObservableCollection<DataSource> sources)
    {
      this.getLiveDataCompletedEventArgs_eventhandler =
        new EventHandler<GetLiveDataCompletedEventArgs>(LiveDataClient_GetLiveDataCompleted);

      this.LiveDataClient.GetLiveDataCompleted += this.getLiveDataCompletedEventArgs_eventhandler;

      this.LiveDataClient.GetLiveDataAsync(sources);
    }
```

**Listing 25: LiveDataService.cs GetLiveDataInformation**

Die Implementierung der GetLiveDataInformation Methode wiederum ruft die asynchrone Methode GetLiveDataAsync des LiveData Services auf. Da ein synchroner Aufruf dieser Methode die Benutzeroberfläche blockieren würde, gibt es laut Vorgabe des Silverlight Frameworks keine synchronen Webservice-Methoden, in WCF hingegen gibt es diese sehr wohl. Sofern der abgesetzte Methodenaufruf eine Antwort erhält, wird nachfolgende Methode zum Zuge kommen:

```
void LiveDataClient_GetLiveDataCompleted(object sender, GetLiveDataCompletedEventArgs e)
    {
      try
      {
        if (e.Result != null && e.Error == null)
        {
          Dictionary<DataSource, DataPoint<object>> series = e.Result;

          this.eventAggregator.GetEvent<LiveDataWorkerCompleteEvent>().Publish(series);
        }
        else if (e.Error != null)
          throw e.Error;
      }
      catch (Exception ee)
      {
        Dictionary<DataSource, DataPoint<object>> dic = new Dictionary<DataSource,
                                                              DataPoint<object>>();
        this.eventAggregator.GetEvent<LiveDataWorkerCompleteEvent>().Publish(dic);
      }
    }
```

**Listing 26: LiveDataService.cs LiveDataClient_GetLiveDataCompleted**

Aus dem „Listing 26: LiveDataService.cs LiveDataClient_GetLiveDataCompleted“ ist ersichtlich, dass bei fehlerfreier Übertragung der Daten vom CLM Server zum CLM WebControl, die Daten anhand des Event-Mechanismus ihren Bestimmungsort finden.

# 7. System Evaluierung

In diesem Kapitel folgt ein Erfahrungsbericht über die eingesetzten Technologien und Praktiken zur Realisierung der Anforderungen aus Kapitel 2.

## 7.1. Softwareentwicklungsprozess

In diesem Abschnitt wird auf die Erfahrungen eingegangen, die während der gesamten Softwareentwicklung mit Scrum gemacht wurden. Zu Beginn gab es einen Analyse- und Recherche-Sprint, sodass sich das Team einen Überblick verschaffen konnte, welche Möglichkeiten und State-of-the-Art Funktionalitäten im Bereich Monitoring existieren. Somit konnte den Teammitgliedern ein Gesamtbild des zu realisierenden Projektes vermittelt werden. Aufbauend auf den Analyse- und Recherche-Sprint konnten mithilfe des Product Owners Product Backlog Items für das Projekt erstellt werden. Diese Items erhielten auch eine Priorisierung des Product Owners. Zur Halbzeit dieser Masterarbeit haben sich beim Auftraggeber neue Anforderungen ergeben, welche teilweise neue und höher priorisierte Items aus dem Product Backlog für die CLM Plattform implizierten. Dank des Scrum-Prozesses konnte sehr flexibel auf die neuen Gegebenheiten reagiert werden, sodass die neuen Anforderungen vollends erfüllt wurden. Aus heutiger Sicht war die Entscheidung den Scrum-Prozess zu verwenden, die richtige.

Nun folgen die negativen Aspekte des Einsatzes von Scrum. Wirtschaftliche Rahmenbedingungen gestatten es oft nicht, dass Rollenmodelle vollständig eingehalten werden können. Einschläge Literatur (Gloger, Scrum - Produkte zuverlässig und schnell entwickeln, 2009) empfiehlt es strengstens, dass Teammitglieder keine weiteren Rollen des Scrum-Prozesses einnehmen. Dies konnte im aktuellen Team nicht realisiert werden, sodass ein Teammitglied gleichzeitig die Rolle eines Scrum Masters hatte.

Einerseits ist die lockere Definition des Scrum-Prozesses und dessen Prozessstrukturen verlockend für Unternehmen diesen für diverse Verbesserungen einzuführen. Andererseits birgt die teilweise Einführung der Vorgehensweise die Gefahr, dass sich Enttäuschung in den einzelnen Teams ausbreitet. Eine vollständige Einhaltung von Scrum ist empfehlenswert, denn einer der großen Vorteile des Scrum-Prozesses ist die Zufriedenheit der Entwickler und die einhergehende Produktivitätssteigerung, welche wiederum dem Unternehmen zugutekommt.

Zusammenfassend kann man sagen, dass die Einsetzung des Scrum-Prozesses die richtige Wahl für dieses Projekt war. Denn trotz der angeführten negativen Aspekte überwiegen doch die positiven Punkte des Prozesses. Diverse Meetings ermöglichen es, Probleme und Hindernisse des Prozesses aufzuzeigen, sodass frühzeitig Korrekturen vorgenommen werden können. All das führt langfristig zu einem Prozess, der genau auf die Bedürfnisse des Teams zugeschnitten ist und deren Zufriedenheit steigert.

## 7.2. Technologien und Architekturkonzepte

Dieses Kapitel beinhaltet den Erfahrungsbericht über die Silverlight Technologie und des MVVM Patterns im praktischen Einsatz dieser Arbeit. Des Weiteren wird am Ende dieses Abschnittes die Verwendung der Entwicklungstools bewertet.

### 7.2.1. MVVM Pattern

Das MVVM Pattern war von größter Bedeutung für die Entwicklung des CLM WebControls. Dank der durchgängigen Trennung der View-Schicht von der Model-Schicht und Verbindung der View-Schicht mit der Viewmodel-Schicht über reinem Databinding, war es eine Leichtigkeit, neue Anforderungen und Funktionalitäten in das System zu integrieren. Des Weiteren konnte in kurzen Iterationszyklen der Umfang der GUI verändert oder aber mit neuen Eingabeoptionen versehen werden. Dies geschah ohne Veränderung der Viewmodel-Schicht, geschweige denn der Model-Schicht. Spätestens zu diesem Zeitpunkt hatte sich der Einsatz des MVVM Patterns bewährt.

Als weiterer Vorteil wird die erhöhte Testbarkeit der einzelnen Module angeführt, sodass wichtige Kernfeatures mittels Unit- und Regressionstests die Qualität des CLM Web Controls auch im weiteren Verlauf der Entwicklung gewährleisten. Zusätzlich ist zu erwähnen, dass im fortgestrittenen Stadium der Entwicklung GUI Capture Replay Applikationen verwendet werden konnten, um Systemtests zu realisieren und diese wiederholbar zu machen. In der frühen Phase der Entwicklung war dies nicht sinnvoll, da sich der Aufbau der GUI andauernd verändert hat und damit auch die Aufzeichnungen der Capture Replay Software nicht mehr replizierbar waren.

Ein Nachteil bei der Verwendung des MVVM Patterns ist, dass eine Unmenge an Frameworks existiert, welche die Unterstützung des MVVM Patterns anpreisen. Bis Dato ist noch immer das Prism Framework im Einsatz, da es sich für die Entwicklung als sehr stabil und als die richtige Entscheidung erwiesen hat.

### 7.2.2. Silverlight

Nun folgt der Erfahrungsbericht über die Silverlight Technologie als RIA Entwicklungstechnologie. Einleitend ist zu sagen, dass Silverlight als Stiefkind der WPF Technologie gesehen werden muss. Durch Realisierung von Projekten mit der WPF Technologie hat man viele Vorteile der Sprache kennen gelernt. Beispielsweise ist die Klasse Datatrigger einer dieser Vorteile. Beim Einsatz der Silverlight Technologie existieren diese Sprachfeatures teilweise gar nicht oder nur mit einer rudimentären Unterstützung. Beispielsweise existieren in Silverlight keine Datatriggers, sondern nur Eventtriggers. Datatriggers ermöglichen es, in Abhängigkeit eines Datentyps entsprechende Datatemplates für die Visualisierung eines Models zu verwenden. Diese Möglichkeit wäre sehr hilfreich beim Anlagen Control gewesen. Man hätte vollständig aus dem XAML Code heraus die entsprechenden Templates für das Anlagenschema setzen können, wie es im CLM System Designer (Konrad, 2010) möglich ist. Da dies nicht der Fall war, musste von Seiten des CLM Web Controls in der Viewmodel Klasse in Abhängigkeit des Datentyps das entsprechende Template mittels C# Code gesetzt werden.

Ein weiterer negativer Punkt von Silverlight ist die Tatsache, dass es eine abweichende Implementierung von Datentypen im Vergleich zum .NET Framework gibt. Beispielsweise existiert in .NET Framework der Datentyp Tuple[66]<T,T> (T für generisch). Dieser Datentyp existiert auch in Silverlight. Von Seiten des .NET Frameworks ist es möglich, diesen Datentyp

[66] Weitere Informationen zur Klasse Tuple findet man unter: http://msdn.microsoft.com/en-us/library/system.tuple%28v=VS.95%29.aspx

mittels Webservice zu verwenden. Das Silverlight Framework besitzt bei der Implementierung des Datentyps Tuple<T,T> eine unzulängliche Ausprägungsstufe, sodass es dem XML Serializer nicht möglich ist, diesen Datentyp zu deserialisieren. Vorbeugend musste somit bei der Verwendung von Datentypen auf Basisdatentypen zurückgegriffen werden.

Silverlight hat eine Unmenge an Feinheiten, die den Missbrauch der Silverlight Applikationen vorbeugen sollen. Beispielsweise ist es nicht möglich, einen Benutzerdialog aus dem Programmcode zu starten, sofern dieser keiner bewussten Interaktion des Anwenders zuordenbar ist. Sofern man dies trotzdem versucht, wird an der entsprechenden Codestelle eine Security-Exception[67] geworfen. Des Weiteren müssen für den Zugriff auf Webservices die Datei „crossdomain.xml“ mit den Inhalt aus „Listing 27: crossdomain.xml“ und die Datei „clientaccesspolicy.xml“ mit dem Inhalt aus „Listing 28: clientacesspolicy.xml“ auf der Root-Ebene des Webservice-Hosting-Servers erreichbar sein, ansonsten ist kein Zugriff auf diese Services durch den Silverlight Client möglich.

```
<! DOCTYPE cross-domain-policy SYSTEM "http://www.macromedia.com/xml/dtds/cross-domain-policy.dtd">
  <cross-domain-policy>
    <allow-access-from domain="*" secure="false"/>
    <allow-http-request-headers-from domain="*" headers="SOAPAction" secure="true"/>
  </cross-domain-policy>
```

**Listing 27: crossdomain.xml**

```
<access-policy>
    <cross-domain-access>
      <policy>
        <allow-from http-request-headers="SOAPAction">
          <domain uri="http://*"/>
          <domain uri="https://*"/>
        </allow-from>
        <grant-to>
          <resource path="/" include-subpaths="true"/>
        </grant-to>
      </policy>
    </cross-domain-access>
  </access-policy>
```

**Listing 28: clientacesspolicy.xml**

Weitere Komplikationen während der Entwicklung hat es beim Einsatz von „Self-signed certificates[68]“in Verbindung mit Windows 7 und dem neuesten Internet Explorer 8 gegeben. Die Installation des Zertifikates am Entwicklungsrechner hatte nicht das Ziel erreicht, eine sichere Verbindung zum Server aufzubauen. Erst der Einsatz der Protokollsoftware Fiddler[69] ermöglichte es, im Debug-Modus von Visual Studio das Zertifikat beim Verbindungsaufbau anzunehmen, das Zertifikat musste manuell akzeptiert werden. Sobald das CLM Web Control am Hosting Server veröffentlicht wird, wird dem Anwender automatisch der Akzeptier-Dialog ähnlich zu „Abbildung 44: Sicherheitsabfrage Self-signed Certificate“ angezeigt.

[67] Weitere Informationen zur Klasse Security Exception findet man unter: http://msdn.microsoft.com/en-us/library/system.security.securityexception.aspx

[68] Weiter Informationen zu „Self-signed certificates“ findet man unter: http://en.wikipedia.org/wiki/Self-signed_certificate

[69] Weitere Informationen zur Fiddler Anwendung findet man unter: http://www.fiddler2.com/fiddler2/


Abbildung 44: Sicherheitsabfrage Self-signed Certificate

Als eine weitere Herausforderung im Einsatz der Silverlight Technologie kann die Verwendung der Webservices gesehen werden. Als Desktop Entwickler hat man die Möglichkeit synchrone Methodenaufrufe der Webservices zu benutzen. Dies ist in Silverlight nicht möglich. Silverlight bietet einen rein asynchronen Zugriff auf Webservice Methoden an. Dies bedarf auf Seiten des Entwicklers einer völlig neuen Denkweise, da das Eintreffen von Nachrichten nicht zu einem bestimmten Zeitpunkt vorherseh-, geschweige denn, bestimmbar ist.

Zusammenfassend kann man sagen, dass beim Einsatz von Silverlight doch einige Probleme aufgetreten sind. Diese konnten jedoch alle mit geeigneten Mitteln gelöst werden, sodass die Entwicklung ohne weitere Probleme verlaufen ist. Viele dieser Hindernisse sind aus dem Blickwinkel eines WPF Entwicklers entstanden, da diese in der WPF Technologie nicht bestehen. Dank dieses Vorwissens konnte jedoch die Einarbeitungszeit in die neue Technologie auf ein Minimum reduziert werden. Trotz dieser Auflistung an Kritikpunkten der Silverlight Technologie kann die Entscheidung für Silverlight als positiv erachtet werden. Dank der Ähnlichkeiten zur WPF Technologie ist das Technologie-Sharing innerhalb des Entwicklungsteams um ein vielfaches einfacher, andernfalls hätte eine komplett andere Technologie eingesetzt werden müssen.

### 7.2.3. Entwicklungstools

Nun folgt eine kurze Bewertung der eingesetzten Entwicklungstools. Für die Entwicklung der CLM Plattform wurde der Team Foundation Server und das Visual Studio 2010 Ultimate eingesetzt. Der Team Foundation Server bietet für Softwareentwicklungsteams eine Menge an

praktischen Funktionen. Gated Checkin[70] und Nightly Builds[71] sind nur einige die an dieser Stelle erwähnt werden sollten.

Die Visual Studio 2010 Ultimate IDE ist die neueste Entwicklung aus dem Hause Microsoft. Diese Entwicklungsumgebung ermöglichte es, auf hohem Niveau und mit vielen praktischen Funktionen die tägliche Arbeit eines Entwicklers zu erfüllen.

Auf Seiten des CLM Web Controls wurde zusätzlich für die Gestaltung des View-Layers die Anwendung Expression Blend 4[72] verwendet. Da sich diese Applikation zum Zeitpunkt der Erstellung dieser Arbeit in der Beta Phase befand, hatten einige Features diverse Probleme, sodass bei Detailarbeiten auf den alt bewährten XAML Editor zurückgegriffen wurde.

Abschließend kann man sagen, dass die Verwendung aktuellster Microsoft Entwicklungsanwendungen eine Steigerung der Team Produktivität ermöglichte. Diese Anwendungen sind für die neuesten Technologien entwickelt worden und haben somit einen wesentlichen Beitrag zum Erfolg dieses Projektes beigetragen. Ein Werkzeug bietet jedoch nur so viel Mehrwert, wie der Anwender imstande ist, aus ihm herauszuholen. Daher war es in der Umsetzung wichtig, dass die Entwickler die Werkzeuge richtig kennengelernt haben. Nur so konnte der Mehrwert richtig ausgenutzt werden.

[70] Weitere Informationen zu Gated Checkin findet man unter: http://msdn.microsoft.com/en-us/library/dd787631.aspx

[71] Weitere Informationen zu Nightly Builds findet man unter: http://msdn.microsoft.com/de-de/library/bb399135.aspx

[72] Weitere Informationen zu Expression Blend 4 findet man unter: http://www.microsoft.com/expression/products/Blend_WhatIsExpressionBlend.aspx

# 8. Zusammenfassung und Ausblick

## 8.1. Zusammenfassung

Die vorliegende Arbeit versucht einen Einblick in das Thema State-of-the-Art Monitoring von Anlagedaten zu gewähren. Eingangs wurden die Begriffe „webbasiertes Monitoring“ und „Informationsvisualisierung“ für diese Arbeit spezifiziert. Anschließend wurden die zu realisierenden Requirements definiert. In weiterer Folge wurde Scrum als geeigneter Softwareentwicklungsprozess ausgewählt. Im Verlauf der Umsetzung haben sich die Anforderungen des Kunden geändert, sodass zu diesem Zeitpunkt die Usecases neu priorisiert wurden. Dank des Einsatzes von Scrum konnte sehr flexibel auf die neuen Gegebenheiten reagiert werden. Die Entwicklung der RIA-Applikation wurde dadurch nicht auffallend gehemmt. Des Weiteren wurden jegliche Anforderungen aus Kapitel 2 erfolgreich realisiert und exemplarisch in Kapitel 6 erklärt.

Die Technologieentscheidung ist auf Silverlight gefallen, welche inklusive des MVVM Patterns Einfluss auf das Kapitel 6 (Ausgewählte Details Implementierung) hatte. In Kapitel 6 wurde anhand mehrerer Beispiele der Aufbau und die Implementierung der RIA-Applikation dargelegt. Focus dieser Arbeit ist der praktische Teil, sodass dieser sehr detailliert ist.

## 8.2. Ausblick

Mit Abschluss dieser Masterarbeit ist das Projekt CLM WebControl keineswegs beendet. Es werden kontinuierlich neue Funktionalitäten integriert und bestehende erweitert. Als nächstes werden aus dem Scrum Product Backlog die Funktionalitäten „Fernwirken einzelner Sensoren“ und „regelbasierte Aktionen“ in Angriff genommen.

Der Einsatz der Silverlight Technologie ermöglicht es, diese Applikation auch für das Windows 7 Phone zu portieren, da die Silverlight Technologie für das Windows 7 Phone komplett kompatibel ist. Einzig der Aufwand, eine neue Oberfläche zu realisieren, könnte eine noch zu erledigende Tätigkeit werden. Diese stellt dank der Trennung der Logik vom Design keine Herausforderung mehr dar.

# Anhang A: Tabellenverzeichnis

# Anhang B: Abbildungsverzeichnis

# Anhang C: Listingverzeichnis

# Anhang D: Referenzen

*Informationsdienst Wissenschaft* . (02. 04 2008). Abgerufen am 09. 08 2010 von http://idw-online.de/pages/de/news253306

*Service Web 3.0 Video*. (2009). Abgerufen am 03. 08 2010 von http://www.future-internet.eu/publications/media.html

*Alexa Site Info*. (21. 07 2010). Abgerufen am 21. 07 2010 von http://www.alexa.com/siteinfo/google.com

*Google Trends*. (21. 07 2010). Abgerufen am 21. 07 2010 von http://www.google.com/trends

*Indeed Job Trend*. (21. 07 2010). Abgerufen am 21. 07 2010 von http://www.indeed.com/jobtrends

*Rich Internet Application Market Share*. (21. 07 2010). Abgerufen am 21. 07 2010 von http://www.statowl.com/custom_ria_market_penetration.php

*Rich Internet Application Statistics*. (20. 07 2010). Abgerufen am 20. 07 2010 von http://www.riastats.com/#

*Simply Hired Employment Trends*. (21. 07 2010). Abgerufen am 21. 07 2010 von http://www.simplyhired.com/a/jobtrends/home

Agency, A. E. (kein Datum). *Effizienzziele können erreicht werden*. Abgerufen am 31. 07 2010 von http://www.monitoringstelle.at/Einsparpotentiale.486.0.html

Aikebaier, A., Yang, Y., Enokido, T., & Takizawa, M. (2009). *Energy-Efficient Computation Models for Distributed Systems.* IEEE Xplore Digital Library: IEEE Computer Society.

Alles, M., Crosby, D., Erickson, C., Harleton, B., Marsiglia, M., Pattison, G., et al. (2006). *Presenter First: Organizing Complex GUI Applocations for Test-Driven Development.* IEEE Xplore: IEEE Computer Society.

Alor-Hernandez, G., Vasquez-Ramirez, R., Posada-Gomez, R., Juarez-Martinez, U., Gomez, J., Mencke, M., et al. (2009). *Towards Dynamic Representation of Rich Internet Applications through Web service Invocation.* IEEE Xplore Digital Library: IEEE Computer Society.

Booth, D., Haas, H., & McCabe, F. (11. 02 2004). *Web Services Architecture*. Abgerufen am 11. 07 2010 von W3C Working Group Note: http://www.w3.org/TR/ws-arch/#whatis

*Carat Paket Haus und Garten*. (kein Datum). Abgerufen am 01. 08 2010 von http://www.graf-online.de

Chlebek, P. (2006). *User Interfaceorientierte Softwarearchitektur.* Wiesbaden: Friedr.Vieweg & Sohn Verlag.

*Compatible Operating Systems and Browsers*. (kein Datum). Abgerufen am 19. 07 2010 von http://www.microsoft.com/getsilverlight/get-started/install/default.aspx?reason=unsupportedbrowser

Curry, E., & Grace, P. (2008). *Flexible Self-Management Using the Model-View-Controller Pattern.* IEEE XPlore: IEEE Computer Society.

De Chiara, R., & Fish, A. (2007). *EulerView: a non-hierarchical visualization component.* IEEE Xplore Digital Library: IEEE Computer Society.

Developer, H. (25. 01 2010). *Studie: Agile Softwareentwicklung ist Mainstream*. Abgerufen am 09. 08 2010 von http://www.heise.de/developer/meldung/Studie-Agile-Softwareentwicklung-ist-Mainstream-912207.html

*Event Aggregator*. (kein Datum). Abgerufen am 30. 07 2010 von http://msdn.microsoft.com/en-us/library/ff649187.aspx

*Forums - Flex*. (kein Datum). Abgerufen am 21. 07 2010 von http://forums.adobe.com/community/flex

Fowler, M. (18. 07 2006). *GUI Architectures*. Abgerufen am 30. 07 2010 von http://www.martinfowler.com/eaaDev/uiArchs.html#Model-view-presentermvp

Fraternali, P., Comai, S., Bozzon, A., & Toffetti Carughi, G. (April 2010). *Engineering Rich Internet Applications with a Model-Driven Approach.* ACM Transactions on the Web: ACM.

Friedrich, J., Hammerschall, U., Kuhmann, M., & Sihling, M. (2009). *Das V-modell XT: Für Projektleiter und QS-verantwortliche.* (2. Ausg.). Dordrecht Heidelberg London New York: Springer.

Gloger, B. (2009). *Scrum - Produkte zuverlässig und schnell entwickeln* (2. Ausg.). München Wien: Carl Hanser Verlag.

Gloger, B. (17. 08 2010). *Nutzen-basierende Preisgestaltung: Zwei Vorschläge*. Abgerufen am 29. 08 2010 von Nutzen-basierende Preisgestaltung: Zwei Vorschläge: http://borisgloger.com/2010/08/17/nutzen-basierende-preisgestaltung-zwei-vorschlaege/

Gossman, J. (05. 10 2005). *Introduction to Model/View/ViewModel pattern for building WPF apps*. Abgerufen am 30. 07 2010 von http://blogs.msdn.com/b/johngossman/archive/2005/10/08/478683.aspx

Hans-Dirk, W. (2008). „Rich Internet Applications“ – Eine perfekte Kombination benutzerfreundlicher Schnittstellen mit Webtechnologie. *Informatik Spektrum*, 333-343.

Höhn, R., & Höppner, S. (2008). *Das V-Modell XT, Anwendungen, Werkzeug, Standards.* Berlin Heidelberg New York: Springer Verlag.

Höhn, R., Rausch, A., & Höppner, S. (2008). *Das V-Modell XT: Grundlagen, Methodik und Anwendungen.* Berlin Heidelberg New York: Springer Verlag.

IABG. (kein Datum). *Das V-Modell*. Abgerufen am 18. 07 2010 von Das V-Modell: http://www.v-modell.iabg.de/

IT-Beauftragten, R. d. (04. 11 2009). *Das V-Modell XT*. Abgerufen am 18. 07 2010 von Das V-Modell XT: http://www.v-modell-xt.de/

*JavaFX Composer*. (kein Datum). Abgerufen am 14. 07 2010 von http://wiki.netbeans.org/JavaFXComposer

*JavaFX General Forum*. (kein Datum). Abgerufen am 20. 07 2010 von http://forums.sun.com/forum.jspa?forumID=932

Kommission, E. (2005). *Weniger kann mehr sein - Grünbuch über Energieeffizienz.* Luxemburg: Amt für Veröffentlichungen der Europäischen Gemeinschaft.

Konrad, J. (2010). *Masterarbeit: Informationsvisualisierung und Fernsteuerung von verteilten Anlagen.* Graz: Tu Graz.

Microsoft. (02 2008). *Model-View-Presenter Pattern*. Abgerufen am 30. 07 2010 von http://msdn.microsoft.com/en-us/library/cc304760.aspx

Murugesan, S. (2007). *Understanding Web 2.0.* IEEE Xplore Digital Library: IEEE Computer Society.

Novell. (kein Datum). *Moonlight*. Abgerufen am 19. 07 2010 von http://mono-project.com/Moonlight

Potel, M. (1996). *MVP: Model-View-Presenter.* Taligent Inc.

*Project Hosting for Open Source Software*. (kein Datum). Abgerufen am 28. 07 2010 von http://www.codeplex.com/

Reenskaug, T. (10. 12 1979). *Models - Views - Controllers.* Abgerufen am 30. 07 2010 von http://heim.ifi.uio.no/~trygver/1979/mvc-2/1979-12-MVC.pdf

Rüssel, F. (11. 12 2009). *Festpreisprojekte und Scrum*. Abgerufen am 29. 08 2010 von Festpreisprojekte und Scrum: http://www.armerkater.de/2009/12/festpreisprojekte-und-scrum/

Scherer, J. (2008). Beim Rugby abgeschaut. *dotnet pro*, 16-19.

Scherer, J. (kein Datum). *Scrum Artefakte*. Abgerufen am 29. 07 2010 von http://scrum-fibel.de/artefakte/scrum%20artefakte.html

Smith, J. (02 2009). *WPF-Anwendungen mit dem Model-View-ViewModel-Entwurfsmuster*. Abgerufen am 30. 07 2010 von http://msdn.microsoft.com/de-de/magazine/dd419663.aspx

*Strategieprozess Energie 2050*. (kein Datum). Abgerufen am 31. 07 2010 von http://www.e2050.at/

Tamm, G., & Günther, O. (2005). *Webbasierte Dienste: Technologien, Märkte und Geschäftsmodelle.* Heidelberg: Physica-Verlag.

*The Official Microsoft Silverlight Forum*. (kein Datum). Abgerufen am 20. 07 2010 von http://forums.silverlight.net/forums/

Vorholz. (07 2003). *Die verbrannten Milliarden*. Abgerufen am 03. 08 2010 von http://www.zeit.de/2003/07/Energie

West, D., Grant, T., Gerush, M., & D´Silva, D. (20. 01 2010). *Agile Development: Mainstream Adoption has changed agility.* Abgerufen am 09. 08 2010 von http://www.forrester.com/rb/Research/agile_development_mainstream_adoption_has_changed_agility/q/id/56100/t/2?src=RSS_2&cm_mmc=Forrester-_-RSS-_-Document-_-56100

Wirdemann, R. (2009). *Scrum mit User Stories.* München Wien: Carl Hanser Verlag.