HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL
FHO Fachhochschule Ostschweiz

ins institute for networked solutions

# Kursanmeldungssystem mit Silverlight und Dynamics CRM

## Bachelorarbeit

Abteilung Informatik
Hochschule für Technik Rapperswil

Frühjahrssemester 2011


Autoren: Clemens Meier und Silvan Gacond
Betreuer: Prof. Hansjörg Huser
Gegenleser: Prof. Stefan F. Keller
Experte: Stefan Zettel, Ascentiv AG

# Einleitung

# Impressum

**Kontakt**
Clemens Meier, cmeier@hsr.ch
Silvan Gacond, sgacond@hsr.ch



Druckdatum, 16.6.2011

# Erklärung

Wir erklären hiermit,

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

Rapperswil, den 16.06.2011

**Clemens Meier** **Silvan Gacond**

# Danksagung

Für die Unterstützung während der Bachelorarbeit möchten wir folgenden Personen einen besonderen Dank ausrichten:

**Prof. Hansjörg Huser** für seine Unterstützung und sein wertvolles und konstruktives Feedback.

**Jürg Jucker** für die fachliche Unterstützung und die gute Zusammenarbeit.

**Andrea Moschin** für die inhaltlichen Inputs aus Anwendersicht.

**Dynamix Solutions GmbH** für das Bereitstellen der nötigen Infrastruktur.

**Marguerite & Peter Meier** für das Korrekturlesen.

**Lydia Mosberger** für den Usability Test.

**Mischa Trecco** für die super LaTeX Vorlage.

# Abstract

Das Microsoft Innovation Center, Teil des Instituts für vernetzte Systeme (INS) der Hochschule für Technik Rapperswil, unterstützt Softwarehersteller mit Schulungs- und Beratungsangeboten. Das bestehende Kursanmeldetool bietet potentiellen Kursteilnehmern die Möglichkeit, sich online über ausgeschriebene Kurse zu informieren, sowie sich für diese anzumelden. Die Kursanmeldung läuft in einer isolierten Umgebung und hat einen separaten Datenstamm. Es soll nun eine Möglichkeit geschaffen werden, das Kursverwaltungssystem in das bestehende Customer Relation Management System Microsoft Dynamics CRM zu integrieren.

Die Kursverwaltung ist ein typischer Anwendungsfall für eine Spezialisierung von Dynamics CRM. Dynamics CRM liefert eine umfangreiche Basisfunktionalität zur Verwaltung von Kundenbeziehungen, hat aber keine Funktionen zur Verwaltung von Kursen, Schulungen oder Workshops. Um die Anforderungen an die Kursverwaltung zu erfüllen, muss Dynamics CRM in vielerlei Hinsicht ausgebaut werden. Um die neuen Daten einzugliedern, muss die Datenstruktur erweitert werden, sowie die Funktionen für die Administration dieser Daten geschaffen werden. Die neu zu entwickelnde Web-Anmeldeapplikation soll als eigenständige Komponente über eine definierte Schnittstelle mit dem CRM kommunizieren.

Die Architektur der Web-Applikation berücksichtigt neben der guten Integration in Dynamics CRM auch andere Aspekte. Aus Gründen der Sicherheit und der Austauschbarkeit kommt eine Zwischenschicht zum Einsatz, welche nur einen limitierten externen Zugriff auf die Daten des Dynamics CRM, sowie die Anbindung an ein anderes Backend System ermöglicht. Ausserdem wird mit Microsoft Silverlight auf eine Frontend Technologie gesetzt, die eine ansprechende Benutzeroberfläche sowie eine saubere Architektur ermöglicht.

Durch den konsequenten Einsatz moderner Technologien und Frameworks wurde eine ausbaubare Lösung geschaffen, welche sich gut in Dynamics CRM integriert und trotzdem soweit abgekoppelt ist, dass sich alle Komponenten gut austauschen lassen. Die Administration der Kurse lässt sich vollumfänglich in Dynamics CRM erledigen. Die dazugehörigen Masken und Workflows sind aus Bedienersicht gleich aufgebaut, wie die restliche CRM Funktionalität. Dadurch findet sich ein geübter CRM Nutzer schnell zurecht und kann wie im restlichen CRM den Arbeitsablauf selbst anpassen oder automatisieren.

# Management Summary

## Ausgangslage

Das Microsoft Innovation Center (MIC) Rapperswil ist Teil des Instituts für vernetzte Systeme (INS). Als ideale Verbindung von Forschung und Praxis unterstützt das Zentrum Softwarehersteller mit Schulungs- und Beratungsangeboten. Monatlich werden Kurse zu verschiedenen Microsoft Technologien durchgeführt.

Zur Verwaltung von Kundenkontakten wird am INS das Customer Relation Management System Microsoft Dynamics CRM verwendet, dessen Kenfunktionalität Vertrieb, Kundendienst und Marketing beinhaltet.

Um Interessenten die Anmeldung an ausgeschriebene Kurse zu ermöglichen, existiert ein Kursanmeldungssystem, welches über das Internet zugänglich ist. Diese beiden Systeme sind komplett unabhängig voneinander und haben separate Datenstämme. Die Daten müssen manuell vom Kursanmeldungssystem in das CRM übertragen werden, was fehleranfällige und mühsame Handarbeit verursacht.

Es soll nun mit einem neuen Kursanmeldungssystem eine Möglichkeit geschaffen werden, den Datenstamm des CRM im Kursanmeldungssystem direkt zu verwenden. Die Administration der kompletten Kurs- und Teilnehmerdaten kann damit im CRM stattfinden. Die manuelle Übertragung der Daten entfällt.

## Vorgehen / Technologien

Microsoft Dynamics CRM bietet neben seiner Kernfunktionalität sehr flexible Anpassungs- und Erweiterungsmöglichkeiten. Die Kursverwaltung ist ein typischer Anwendungsfall einer Spezialisierung von Dynamics CRM. Die Basisfunktionalität zur Verwaltung von Kundenbeziehungen besteht und kann nach nur kleinen Anpassungen direkt für die Administration der Kursteilnehmer verwendet werden. Für die Verwaltung von Kursen, Schulungen oder Workshops, besteht jedoch noch keine fertige Lösung.

Um die Anforderungen an das Kursanmeldungs- und Kursverwaltungssystem zu erfüllen, können die Erweiterungsmöglichkeiten des CRM verwendet werden. Durch benutzerdefinierte Entitäten kann der Datenstamm um beliebige neue Informationsstrukturen erweitert werden. Für die Administration dieser Daten generiert Dynamics CRM automatisch Formulare, welche für den jeweiligen Anwendungsfall anpassbar sind. Durch CRM Plugins können unter anderem komplexere Konsistenz- und Plausibilitätsprüfungen realisiert werden.

Die über das Internet erreichbare Kursanmeldeapplikation soll als eigenständige Komponente über eine definierte Schnittstelle mit dem CRM kommunizieren. Für den externen Zugriff auf die CRM Daten, bietet Dynamics CRM einen Webservice. Aus Gründen der Sicherheit, Performance und Lizenzierung, soll dieser Service aber nicht direkt aus dem Internet erreichbar sein.

Durch diese Anforderungen ist eine Zwischenschicht nötig, welche in einer demilitarisierten Zone des Netzwetzwerks betrieben wird und nur die erforderlichen Daten aus dem CRM für die Kursanmeldungsoberfläche auf dem Internet zur Verfügung stellt. Durch die verwendete Architektur innerhalb der Zwischenschicht, könnte diese mit wenig Anpassungsaufwand auf ein anderes Backend System angepasst werden.

Um auf dem Internet eine ansprechende Benutzeroberfläche und eine saubere Architektur zu ermöglichen, wird mit der Clienttechnologie Microsoft Silverlight 4.0 gearbeitet.

Verwendete Technologien und Frameworks:

- Microsoft Dynamics CRM 2011
  - Custom Entities
  - Frontend Customizations
  - Plugins
  - Processes
- Microsoft Silverlight 4.0
  - WCF RIA Services
  - Prism
  - MEF
  - Silverlight 4 Unit Testing Framework

## Ergebnisse

Durch den konsequenten Einsatz moderner Technologien und Frameworks wurde eine ausbaubare Lösung geschaffen, welche sich gut in Dynamics CRM integriert und trotzdem soweit abgekoppelt ist, dass sich alle Komponenten gut austauschen lassen.

Die Administration der Kurse lässt sich vollumfänglich in Dynamics CRM erledigen. Die dazugehörigen Masken und Workflows sind aus Bedienersicht gleich aufgebaut, wie die restliche CRM Funktionalität. Dadurch findet sich ein geübter CRM Nutzer schnell zurecht und kann wie im restlichen CRM den Arbeitsablauf selbst anpassen oder automatisieren.

Die Weboberfläche ermöglicht den Kursinteressenten eine komfortable Information und Anmeldung für die angebotenen Kurse. Durch die direkte Verbindung mit dem CRM fällt keine Arbeit für das Übertragen der Daten mehr an.

Abschliessend kann gesagt werden, dass das System einen Stand erreicht hat, der die Funktionalität des bestehenden Systems abdeckt und eine viel bessere Integration in das bestehende CRM System bietet.

## Ausblick

Das entwickelte System kann in einem nächsten Schritt an das produktiv genutzte CRM angeschlossen und in Betrieb genommen werden.

Durch die flexible Architektur, kann das System problemlos weiter ausgebaut werden. Denkbare Erweiterungen sind zum Beispiel eine Bereitstellung von Dokumenten für Kursteilnehmer oder über die Weboberfläche zugängliche Zufriedenheitsumfragen.

# Inhaltsverzeichnis

# Teil I

# Technischer Bericht

# 1 Einführung

## 1.1 Kurse des Microsoft Innovation Center Rapperswil

> „Das Microsoft Innovation Center (MIC) Rapperswil ist eine gemeinsame Initiative von Microsoft Schweiz und der HSR Hochschule für Technik Rapperswil. Als ideale Verbindung von Forschung und Praxis unterstützt das Zentrum Softwarehersteller mit Schulungs- und Beratungsangeboten. Die Leistungen unterstützen Firmen in allen Stadien Ihrer Produkteentwicklung: vom ersten strategischen Ansatz, über den Wissensaufbau bis hin zur individuellen Beratung, Projektbegleitung und Umsetzung.“ [MICR11]

Es wird bereits ein Kursanmeldungssystem eingesetzt, mit dessen Hilfe sich potentielle Interessenten auf dem Internet über Kurse informieren und sich auch gleich dafür anmelden können. Die aktuelle Version dieses Anmeldungssystems ist ein komplett abgeschlossenes System, bei welchem die Stammdaten zusätzlich gepflegt werden müssen. Dies führt zu einem redundanten Datenstamm innerhalb des MIC Rapperswil und zu erhöhtem Verwaltungsaufwand. Ausserdem ist die Benutzeroberfläche des bestehenden Systems eher spartanisch gehalten.

Das bestehende System soll nun durch ein, in den angesprochenen Punkten verbessertes System ersetzt werden. Um vor allem dem Hauptproblem, der fehlenden Integration in Umsysteme begegnen zu können, soll Microsoft Dynamics CRM als Grundlage für die Verwaltung des Datenstamms eingesetzt werden. Das Frontend für Kursinteressenten und Kursteilnehmer soll nach wie vor im Internet zur Verfügung stehen.

## 1.2 Microsoft Dynamics CRM

Mit der Dynamics Reihe hat Microsoft umfassende Enterprise Lösungen auf den Markt gebracht, um die Abläufe in Betrieben zu unterstützen. Mit Dynamics CRM, einem Produkt aus der Dynamics Reihe, sollen „Customer-Relations“ also Kundenbeziehungen (z.B.: Marketingaktivitäten, Service- und Supportfälle, etc.) gepflegt und verwaltet werden.

Die Grundfunktionalität von Dynamics CRM beinhaltet Abläufe aus dem normalen Geschäftsleben. Zusätzlich lässt sich das System flexibel erweitern und anpassen. Es bietet eine ideale Plattform um den Anwendungsfall einer Kursverwaltung zu implementieren. Ausserdem stellt es Schnittstellen für die Anbindung an eine Web-Applikation zur Verfügung und ermöglicht so die Kursanmeldung über das Internet. Details zur Funktionalität und Erweiterbarkeit siehe *Microsoft Dynamics CRM 2011* (auf Seite 5) .

# 2 Systemüberblick

## 2.1 Bestehendes System

Das bestehende Kursverwaltungssystem, welches aktuell am Institut für Vernetzte Systeme (INS) eingesetzt wird, ist eine klassische Web-Applikation auf Basis von ASP.NET. Die Datenhaltung erfolgt in einer eigenen, abgetrennten Datenbank und ist nicht in andere Systeme integriert. Daraus entsteht eine grosse Redundanz im Datenstamm, beispielsweise bei den Adressdaten der Teilnehmer.

## 2.2 Microsoft Dynamics CRM 2011

Microsoft Dynamics CRM 2011 enthält Grundfunktionen zur Verwaltung von Geschäftsprozessen und Abläufen innerhalb eines Betriebs. Die Kernfunktionalität beinhaltet Vertrieb, Kundendienst und Marketing. Durch die hohe Flexibilität und Anpassbarkeit lässt sich das System optimal auf bestehende Prozesse und auf die jeweilige Geschäftssituation anpassen.


**Abbildung (2.1)** Übersicht Dynamics CRM [Mic11]

Alle typischen Kundeninteraktionen können in Dynamics CRM abgebildet und überwacht werden. Es bietet viele Möglichkeiten zur Zurückverfolgung von Interaktionen sowie der Übersicht der geplanten oder offenen Interaktionen mit einem Kunden (E-Mail, Telefon, Fax, etc.). Kommt auf einem beliebigen Kanal eine Kundenanfrage, kann der entsprechende

Bearbeiter sofort erkennen, welche Interaktionen auch von anderen Mitarbeitern oder Teams in der Zwischenzeit unternommen wurden.

### 2.2.1 Vertrieb

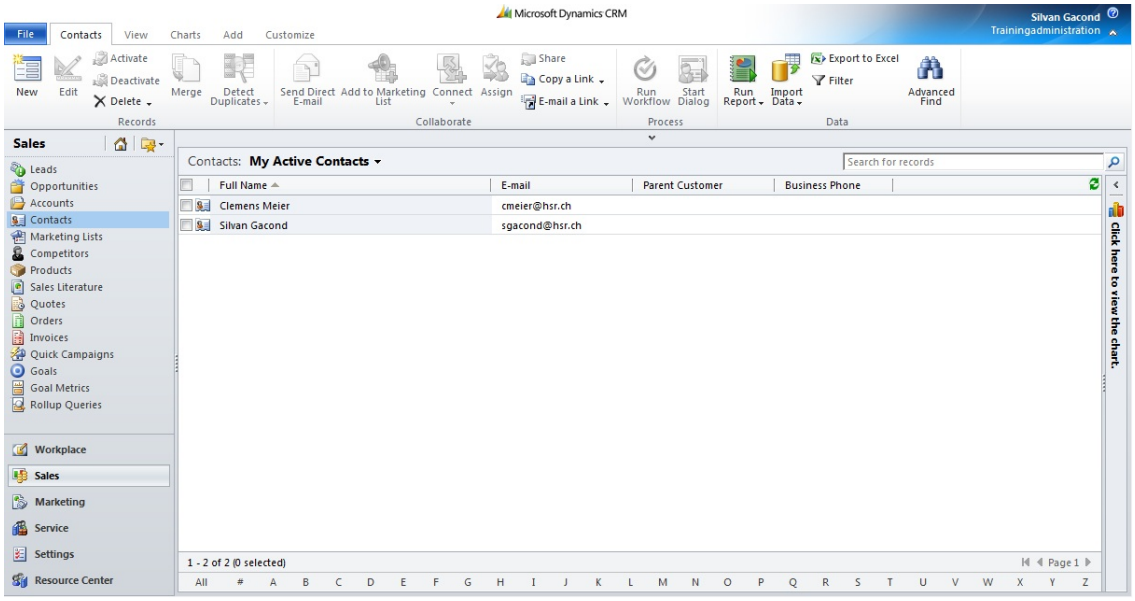



**Abbildung (2.2)** Screenshot Vertriebsmodul

Um Vertriebsaktivitäten systematisch zu erfassen, nach zu verfolgen und zu überprüfen, bietet Dynamics CRM eine eingebaute Verwaltung von Leads, Geschäftschancen, Kunden-Accounts und Kontakten. Ausserdem wird die Verwaltung von Angeboten, Verkaufschancen und Aufträgen ermöglicht. Alle diese Bereiche profitieren vom oben angesprochenen Nachverfolgungssystem.

Zur Integration der Kursverwaltung bietet sich die Verwendung der Kontaktdaten und Zuordnung in Interessegruppen an. Meldet sich z.B. ein Teilnehmer für einen Kurs an, kann sein Kontakt der entsprechenden Firma zugeordnet werden und der Kontakt mit dem gewählten Kursthema der gewünschten Technologiesparte.

### 2.2.2 Kundendienst

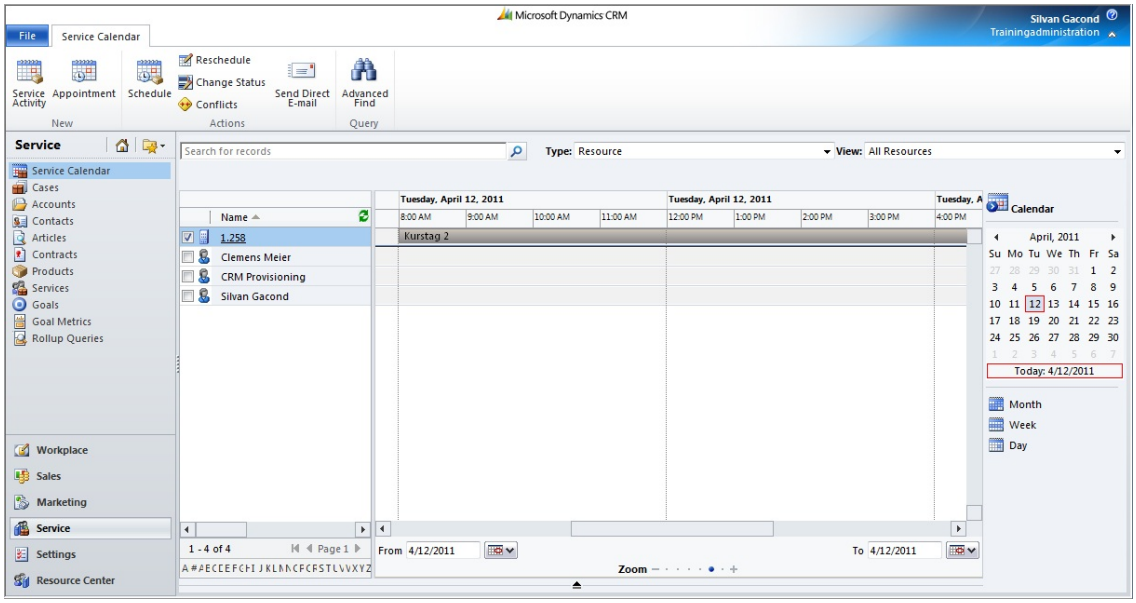



**Abbildung (2.3)** Screenshot Kundendienstmodul

Im Kundendienst-, respektive im Servicemodul des Dynamics CRM werden Service-Fälle und entsprechende Service-Aktivitäten verwaltet. Ausserdem verfügt das Modul über eine Ressourcenplanung, welche die Administration von Personalressourcen sowie die Verwaltung von Standorten und Räumen ermöglicht.

Für den Fall der Kursverwaltung soll die Standort- und Raumverwaltung verwendet werden. Die einzelnen Kurstage werden dafür als Service-Aktivität eingetragen, was die Verwendung der Ressourcenplanung ermöglicht.

### 2.2.3 Marketing

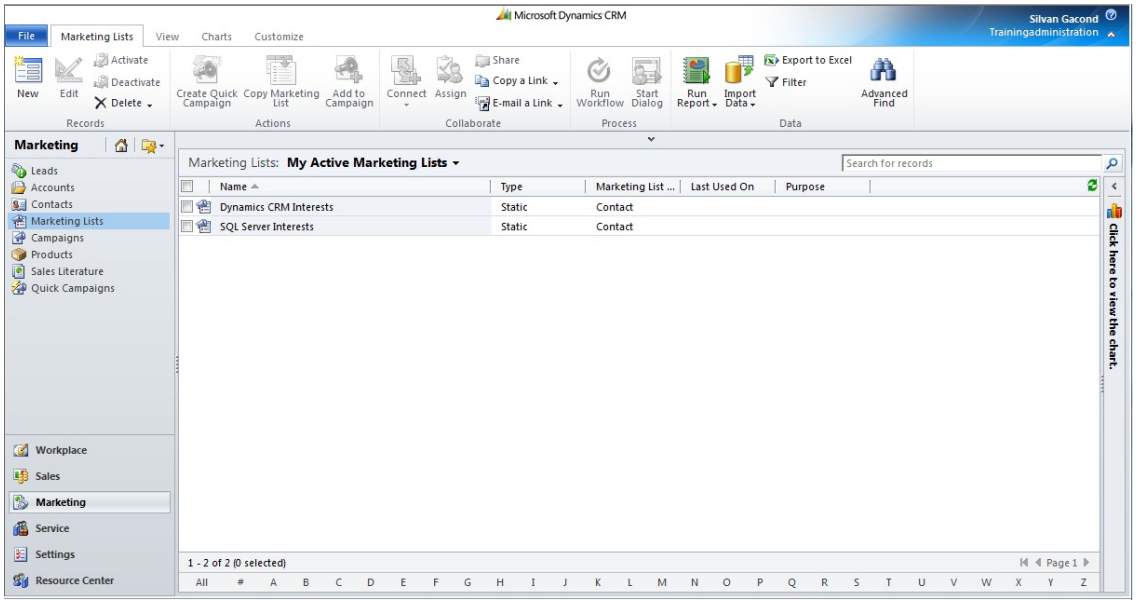



**Abbildung (2.4)** Screenshot Kundendienstmodul

Der breite Datenstamm aus dem Vertriebs- und Kundendienstmodul lässt sich natürlich auch hervorragend für Marketingzwecke einsetzen. Dazu können Kontakte, Kundenaccounts oder Leads in sogenannten Marketinglisten verwaltet werden. Diese beziehen sich auf potentielle Zielgruppen für ein bestimmtes Produkt oder eine bestimmte Produktkategorie. Mit diesen Marketinglisten kann man anschliessend Kampagnen auslösen, welche über die verschiedenen Kommunikationskanäle laufen.

Für die Verwaltung von Kursen lassen sich diese Marketingmechanismen gut nutzen.

### 2.2.4 xRM - Erweiterbarkeit

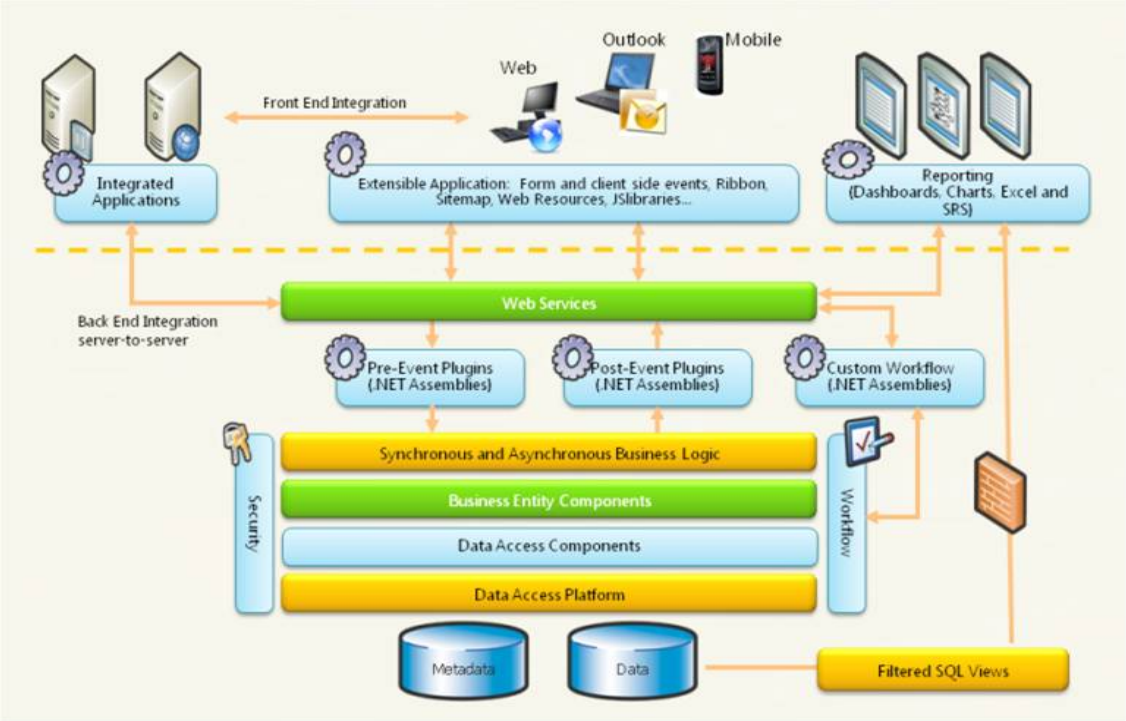



**Abbildung (2.5)** Übersicht xRM - Erweiterbarkeit [Mic11]

Durch die Erweiterbarkeit von Dynamics CRM (xRM) wird die Anwendung für die Verwaltung von Kursen überhaupt möglich. Einerseits lassen sich die bestehenden Entitäten nach entsprechenden Bedürfnissen anpassen und erweitern. Andererseits ist durch die Web-Service Schnittstelle möglich, ausserhalb des CRM Clients auf die Datenstruktur des CRM zuzugreifen.

Ausserdem bietet Dynamics CRM die Möglichkeit mit eigenen Plugin-Assemblies Geschäftslogik im Kern des CRM einzubinden. So lassen sich zusätzliche Automatismen oder Validierungen einbauen, welche sowohl im CRM Client wie auch beim Zugriff über die Web-Service Schnittstelle eingreifen können.

## 2.3 Microsoft Silverlight

Mit Microsoft Silverlight steht eine Technologie zur Verfügung, welche die Entwicklung von umfangreichen Benutzeroberflächen innerhalb einer Webseite erlaubt. Die Silverlight-

Anwendung läuft innerhalb eines Browser-Plugins, welches die Benutzer vorher installieren müssen. Obwohl nicht alle Anwender im Internet das Silverlight-Plugin installiert haben, darf von einem Interessent eines Microsoft-Kurses durchaus erwartet werden, dieses Plugin auf seinem Rechner zu installieren.

## 2.4 Gesamtüberblick

Das neue Kursanmeldungssystem soll nun in das bestehende System mit Microsoft Dynamics CRM integriert werden. Dabei darf Microsoft Dynamics CRM aus Sicherheits- und lizenzrechtlichen Gründen nur intern, also nicht direkt über das Internet erreichbar sein. Es ist dazu eine Zwischenschicht innerhalb der demilitarisierten Zone (DMZ) nötig, welche nur die gewünschten Daten aus dem CRM über das Internet zur Verfügung stellt. Eine Informationsplattform soll Interessenten über das aktuelle Kursangebot informieren. Ausserdem soll sich ein Interessent direkt über diese Plattform anmelden können.

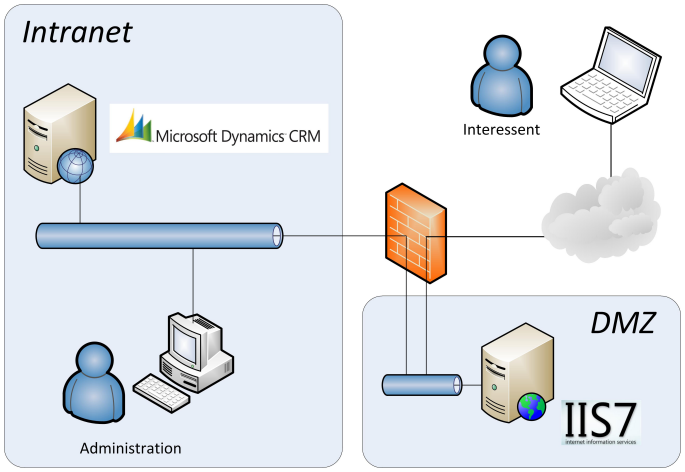



**Abbildung (2.6)** Übersicht des Gesamtsystems

# 3 Technologieevaluation

## 3.1 Interne Administration der Kurse

### 3.1.1 Microsoft Dynamics CRM

Mit Microsoft Dynamics CRM ist die Basis für die serverseitige Umsetzung bereits festgelegt. Eine Abklärung welche Funktionalitäten direkt von Dynamics verwendet werden können und welche noch zusätzlich implementiert werden müssen, ist im Kapitel *Verwendbare Funktionalität von Microsoft Dynamics CRM* (auf Seite 45) zu finden.

## 3.2 Über das Internet erreichbare Oberfläche

### 3.2.1 Silverlight 4

Microsoft bietet mit Silverlight 4 eine sehr ausgereifte Clienttechnologie. Mit Silverlight ist es möglich, Webapplikationen auf ähnliche Art zu entwickeln, wie man sich dies bei Desktopapplikationen gewohnt ist. Um eine Silverlight Applikation zu verwenden, ist allerdings neben einem marktüblichen Webbrowser noch ein Silverlight Plugin notwendig, welches für Betriebssysteme Microsoft Windows und MacOS X zur Verfügung steht.

Die Clientapplikation wird über eine übliche HTML Webseite zur Verfügung gestellt. Wird diese Webseite über einen Webbrowser geladen, so wird (sofern bereits installiert) das Silverlight Plugin gestartet, welches die Silverlight Applikation ausführt. Die Applikation kommuniziert nun mit einem Webservice, welcher die benötigten Daten liefert.

#### Vorteile

Silverlight bietet eine nahtlose Integration mit anderen .NET Technologien wie zum Beispiel WCF Services. Die standardmässig integrierte Bibliothek für grafische Benutzeroberflächen kann mittels Bibliotheken von Drittherstellern beliebig erweitert werden. Silverlight eignet sich damit sehr gut für Webapplikationen mit viel Benutzerinteraktion und Datenkommunikation. Es ermöglicht die Umsetzung des MVVM Patterns, welches das automatisierte Testen der Applikation mit Unit Test wesentlich erleichtert.

#### Nachteile

Das Silverlight Plugin ist nur auf etwa 50% aller Rechner am Internet installiert. Es könnte deshalb sein, dass ein Interessent zuerst das Silverlight Plugin installieren muss, bevor er die Kursinformationen betrachten könnte. Allerdings kann man vor allem bei potentiellen Kursteilnehmern, welche sich für Microsoft Kurse interessieren annehmen, dass sie bei der Installation eines Silverlight Plugins nicht überfordert sind.

### 3.2.2 Asp.NET und HTML 5

ASP.NET ist die etablierte Technologie für klassische Webapplikationen von Microsoft. Mit ASP.NET Webforms oder ASP.NET MVC gibt es verschiedene Herangehensweisen um Webapplikationen zu entwickeln, die sich vor allem in der Architektur der Software unterscheiden. Webforms ist der traditionelle Ansatz von Microsoft um Webapplikationen zu erstellen. ASP.NET MVC hingegen verfolgt einen leichtgewichtigeren Ansatz, welcher von anderen modernen MVC (Model View Controller) Frameworks wie Ruby on Rails oder Spring MVC inspiriert ist. Die neuen Features von HTML 5 ermöglichen eine einfache Einbindung von Multimediainhalten für die direkte Verwendung im Webbrowser ohne die Notwendigkeit weiterer Browserplugins.

#### Vorteile

Zur Betrachtung der Webseite wäre kein Browserplugin nötig.

#### Nachteile

Erfahrungsgemäss ist vor allem die Entwicklung von Benutzeroberflächen mit HTML und JavaScript einiges aufwändiger und fehleranfälliger als mit Desktop- oder Clienttechnologien wie Silverlight. Vor allem Browserinkompatibilitäten erschweren die Entwicklung von komplexeren grafischen Benutzeroberflächen. Durch den Einsatz der benötigten unterschiedlichen Technologien (ASP.NET / HTML & CSS / JavaScript), erhöht sich ausserdem der Testaufwand. Der HTML 5 Standard ist noch nicht endgültig verabschiedet und nicht in allen marktüblichen Webbrowsern zuverlässig umgesetzt, was die Entwicklung zusätzlich erschwert.

### 3.2.3 Weitere Alternativen

Natürlich könnten als alternative Client Technologien zu Silverlight auch Adobe Flash / Adobe Air oder JavaFX in Betracht gezogen werden. Zum Beispiel Flash hätte eine deutlich höhere Verbreitung als Silverlight. Auch serverseitig könnten andere Technologien eingesetzt werden. Da mit Microsoft Dynamics CRM aber auf eine Microsoft-Technologie gesetzt wird, macht es hier wenig Sinn nicht die verwandten Technologien aus gleichem Hause zu benützen.

### 3.2.4 Entscheidung

Silverlight wird für die Kursanmeldung verwendet. Für kleine Übersichten ohne Interaktivität könnte ASP.NET MVC als einfache Alternative dienen.

## 3.3 Webservice

Der Webservice soll bestimmte Daten vom bestehenden Microsoft Dynamics CRM System auslesen und stellt diese in aufbereiteter Form über einen eigenständigen Webservice für die Anzeige im Internet zur Verfügung. Neben dem Lesen müssen auch gewisse Daten geschrieben werden können. Ausserdem ist über diesen Webservice eine Registration und Authentifizierung von Benutzern notwendig.

### 3.3.1 Verbindung zu Microsoft Dynamics CRM

Microsoft Dynamics kann Daten über einen WCF-Service zur Verfügung stellen. Diese Schnittstelle bietet dank WCF 4.0 ein vollständiges Linq (Language Integrated Query) Interface, welches es einfach und effizient ermöglicht Daten aus dem CRM-System für die weitere Verwendung zu laden.

### 3.3.2 Kommunikationsmöglichkeiten zur Bereitstellung der Daten im Internet

Silverlight bietet Möglichkeiten um Daten mit verschiedensten Technologien zu senden oder zu empfangen. Für die Übermittlung von komplexen Datentypen über das Internet ist davon in erster Linie SOAP (nach inzwischen veralteter Definition eine Abkürzung von Simple Object Access Protokoll) interessant.

### 3.3.3 Windows Communication Foundation (WCF)

Mit Windows Communication Foundation steht in .NET ein Framework für die Erstellung von Services zur Übermittlung von komplexen Datentypen zur Verfügung. WCF ist nicht nur für Webservices ausgelegt, sondern ermöglicht auch Interprozesskommunikation. WCF kann mit Hilfe von WSDL (Web Service Description Language) per Knopfdruck Code für die Clientseitige Verwendung eines SOAP-Services erstellen. Auch die Serverseitige Implementierung des Services gestaltet sich durch WCF eher einfach.

#### Vorteile

WCF ist effizient und einfach.

#### Nachteile

Die Verwendung von reinem WCF wird kompliziert, sobald ein komplettes Domainmodell mit vielen Beziehungen zwischen den Daten auf der Clientseite verwendet werden soll. Sobald ein Objekt mehrmals vom Service auf den Client geladen wird, besteht die Gefahr von „Lost Update“. Daten können auf der einen Instanz des Datenobjektes geändert werden, ohne das diese auf der anderen Instanz angepasst werden. Bei der Speicherung des Objektes geht darum die vorherige Änderung verloren.

### 3.3.4 WCF RIA Services

WCF RIA Services geht einen Schritt weiter und fügt dem WCF Service praktische Funktionalitäten für verschiedene Problemlösungen hinzu. RIA Service verwaltet zum Beispiel die übermittelten Objekte auf der Clientseite. Wird auf demselben Client nun ein Objekt ein zweites Mal verlangt, so wird dieses nicht neu vom Webservice geladen, sondern aus dem Objektpool genommen. Ausserdem unterstützt RIA Services sowohl server- als auch clientseitige Validation von Properties der übermittelten Datentypen.

WCF RIA Services wurde ursprünglich für die Verwendung mit Silverlight entwickelt, kann aber auch mit anderen .NET Technologien verwendet werden. Wird als untere Schicht das Microsoft ADO.NET Entity Framework verwendet, so gestaltet sich die Erstellung eines RIA Services durch einen einfachen Knopfdruck.

Wie man einen einfachen WCF RIA Service mit einer anderen Datenquelle implementiert, ist im in der Software Projektdokumentation im Kapitel Technologien und Konzepte (siehe *WCF RIA Services* auf Seite 56) erklärt.

### Vorteile

RIA Services erleichtert die Verwendung eines komplexeren Domainmodelles auf der Clientseite wesentlich. Explizites Nachladen von abhängigen Objekten ist nicht nötig. Ausserdem gestalten sich die asynchronen Aufrufe zur Verwendung des Services mit RIA Services sehr einfach und übersichtlich. WCF RIA Services wurde von Anfang an für die Verwendung mit Silverlight ausgelegt.

### Nachteile

Durch die Verwendung von WCF RIA Services entsteht eine weitere Abhängigkeit von einer Technologie welche noch neu und deren Zukunft ungewiss ist. RIA Services kann mit Entity Framework sehr einfach verwendet werden. Bei der Verwendung anderer Datenquellen, ist aber ein Zusatzaufwand nötig.

## 3.3.5 Entscheidung

Authentifizierung, eingebaute Validation, Nachladen von abhängigen Daten und die Verwendung eines Clientkontextes vereinfachen die Handhabung des Webservices. Die Entwicklung des Services verkompliziert sich durch die Verwendung von RIA Services, allerdings wird dafür die Entwicklung des Web-Frontends einfacher. Darum wurde die Entscheidung gefällt, für die Kommunikation zwischen der Zwischenschicht und dem Web-Frontend WCF RIA Services einzusetzen.

# 4 Umsetzungskonzept

## 4.1 Architekturübersicht

Die Implementierung des Kursadministrationssystems teilt sich in drei physikalische Layer auf:

**CRM-Layer** Die Datenhaltung sowie die ganze Back-End Funktionalität ist innerhalt von Dynamics CRM implementiert. Logische Prüfungen und Konsistenzchecks sollten bereits auf dieser Stufe mit Hilfe von Plugins erfolgen, damit diesen auch Eingriffe aus dem Back-End unterliegen.

**Service-Layer** Zwischen CRM und Web-Frontend wird aus Security- und Entkopplungsgründen eine Zwischenschicht eingesetzt.

**Web-Layer** Die Kursanmeldung ist im Web in einer Silverlight-Applikation eingegliedert. Sie konsumiert den WCF-Service, der durch die Zwischenschicht bereitgestellt wird. Grundsätzlich können auch andere Web-Komponenten angeschlossen werden, dies wäre denkbar um z.B. eine kleine Komponente in eine andere Website einzugliedern.

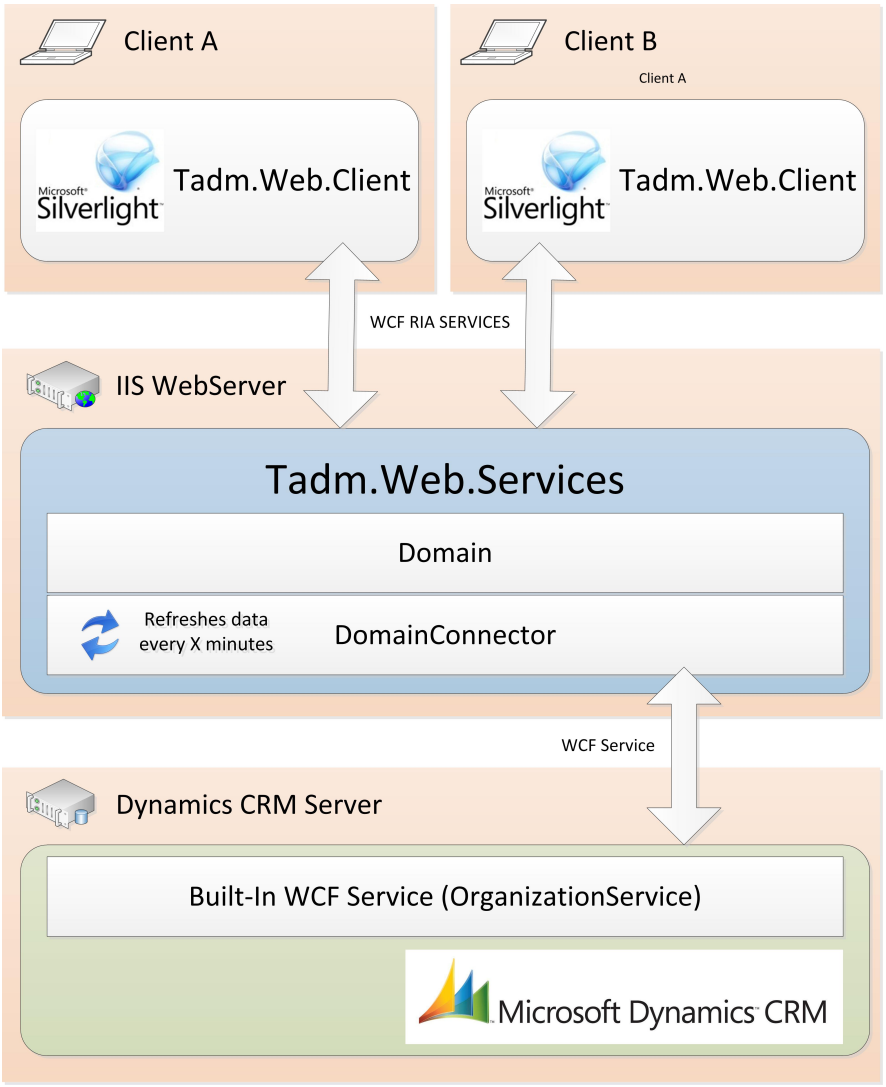



**Abbildung (4.1)** Überblick über die Komponenten (schematisch)

## 4.2 CRM - Integration

Die Integration der Kursverwaltung in das Dynamics CRM ist eine klassische Anwendung der xRM Erweiterungsmöglichkeiten des Systems. Der vorliegende Anwendungsfall stellt einen Fall einer Kundenbeziehung dar, nämlich das Konsumieren einer Dienstleistung durch Kunden. Konkret: Die Teilnahme an einem ausgeschriebenen Kurs.

In einem ersten Schritt musste das Datenmodell im CRM abgelegt werden. Hier wurde

mit benutzerdefinierten Entitäten gearbeitet. Diese sind aus technischer Sicht im weitesten Sinne mit Entitäten einer Datenbank zu vergleichen und werden schlussendlich von Dynamics CRM auch als solche abgelegt. Hier kann nun von der Zusatzfunktionalität des Dynamics CRM gegenüber einer normalen Datenbank Gebrauch gemacht werden. Es stehen standardmässig bereits Felder für Ownership- und Change Tracking der einzelnen Einträge zur Verfügung. Diese benutzerdefinierten Entitäten können analog einer normalen Datenbank auch Beziehungen enthalten, welche auf referentielle Integrität geprüft werden können.

Um die angelegten Daten im Back-End bearbeiten zu können, generiert Dynamics CRM automatisch Listen und Formulare zum Abfragen und Bearbeiten der erstellten Entitäten. Diese sind für den jeweiligen Anwendungsfall anpassbar. Ausserdem lassen sie sich mit eigenen Komponenten ergänzen. So kann beispielsweise ein Rich-Text Editor eingebaut werden, um Kursberschreibungen komfortabel im HTML Format erfassen zu können.

Komplexere Plausibilitätsprüfungen von Einträgen können mit Plugins realisiert werden. Diese haben den entscheidenden Vorteil, dass sie sowohl beim Bearbeiten einer Entität übers Formular im CRM, wie auch beim Update via Web-Service Schnittstelle eingreifen und so Prüfungen vornehmen können. Ein Beispiel für den Einsatz dieses Konzeptes, ist die zwingende Erfassung einer einmaligen primären E-Mail Adresse eines Kunden, damit diese als Login-Benutzername für den Web-Client verwendet werden kann. Wird diese Prüfung als Plugin realisiert, wird ein Duplikat sowohl dann vermieden wenn der Kunde seine E-Mail Adresse im Web-Client ändert, wie auch wenn er beispielsweise mit jemandem aus der Administration des MIC telefoniert, welcher ihm die E-Mail Adresse direkt im CRM Backend abändert.

## 4.3 Zwischenschicht

Die Zwischenschicht entkoppelt die Web-Oberfläche vom CRM System. Sie kommuniziert über die eigenen CRM-Serviceschnittstellen mit dem CRM und über einen losgelösten WCF RIA-Service mit dem Web Front-End.

Um eine möglichst geringe Kopplung zum CRM zu erreichen, existiert auf der Zwischenschicht ein zweites, logisch losgelöstes Domänenmodell. Diese einfachen Domänenobjekte (Data Transfer Objects, DTO's) werden durch ein Connector-Interface abgefüllt, welches als Adapter an das angebundene System fungiert. Dies würde eine Loslösung vom CRM in ein anderes Back-End System ermöglichen. Ausserdem werden die Domänenobjekte des CRM durch Entfernen der CRM eigenen Prefixe, welche Konflikte zwischen verschiedenen Herausgebern verhindern sollen, in eine neutrale Form gebracht.

Zusätzlich übernimmt die Zwischenschicht eine Caching-Funktion. Die durch den Connector übertragenen Entitäten werden auf der Zwischenschicht zwischengespeichert, damit nicht

jeder Aufruf bis in das CRM-System geleitet werden muss. Dies wurde hauptsächlich aus Performancegründen so umgesetzt. Die Gefahr von Synchonisationsproblemen besteht nur in sehr wenigen Fällen (z.B.: nur noch ein freier Platz im Kurs und zwei Benutzer möchten sich gleichzeitig anmelden). Um auch diese Fehler zu vermeiden, lässt sich der Cache überbrücken und die Überprüfung erfolgt direkt im CRM.

Ein weiteres Ziel der Zwischenschicht ist die Entkoppelung der Systeme aus Sicherheitsgründen. Würde der Web-Client direkt auf das CRM System zugreifen, müsste der CRM Server seine Schnittstellen gegenüber dem Internet zur Verfügung stellen. So entsteht einerseits ein Lizenzierungsproblem, da jeder zugreifende Client die CRM Funktionalität benutzt und dadurch Lizenzpflichtig wird, andererseits aber auch ein Sicherheitsproblem, da über diese CRM Serviceschnittstellen der volle Zugriff auf die ganze Funktionalität des CRM gewährt wird.

## 4.4 Web-Frontend

Um das Web-Frontend der Kursanmeldung zu realisieren, wird auf die Microsoft Silverlight Technologie zurückgegriffen. Diese erlaubt die Entwicklung von komplexen und interaktiven Web-Oberflächen. Ausserdem lässt sie sich durch die gute Unterstützung von WCF (inkl. RIA-Services) sehr gut in die verwendete Technologielandschaft eingliedern.

Die Kommunikation mit der Zwischenschicht erfolgt über die bereits erwähnten „Rich Internet Application Services“ (RIA Services). Dies ist eine weitgehend automatisierte Integration der Windows Communication Foundation (WCF) Services in Silverlight. RIA Services erlaubt die Kommunikation des Silverlight Clients mit einem Objektmodell auf dem Webserver in dem von der Entwicklungsumgebung Zwischencode generiert wird, der bereits Caching und Lazy Loading Funktionalitäten umfasst und so sehr einfach und schnell anzuwenden ist. Dadurch wird neben dem reduzierten Entwicklungsaufwand auch viel unnötiger Datenverkehr über das Internet vermieden. Ausserdem sind diverse Funktionalitäten wie z.B. die Benutzerauthentifizierung bereits angedacht oder eingebaut.

Um auch innerhalb der Silverlight Applikation des Frontends eine geringe Kopplung der Komponenten sowie eine gute Testbarkeit zu erreichen, wird mit dem Microsoft Prism Framework und konsequent mit dem Model-View-Viewmodel Pattern (MVVM) gearbeitet. Micorosft Prism besteht aus einem Framework kombiniert mit einer Sammlung von „Best Practices“ von diversen Anwendungsbeispielen (Bekanntmachen von Service-Referenzen, Modularisierung, Navigation, etc.).

# 5 Resultate

Das Projekt ist nach Ende der Bachelorarbeit auf einem Stand, auf dem es aus Sicht der Funktionalität grundsätzlich das bestehende System ablösen könnte. Die grundlegenden Use Cases sind umgesetzt und getestet.

Allfällige Anpassungen oder Erweiterungen nach Wunsch der Endanwender sind durch die flexible Architektur verhältnismässig einfach umsetzbar. Durch die Erweiterbarkeit des CRM im laufenden Betrieb lassen sich auch auf der Seite der Administration Verbesserungen und Vereinfachungen des Arbeitsablaufes praktisch jederzeit realisieren.

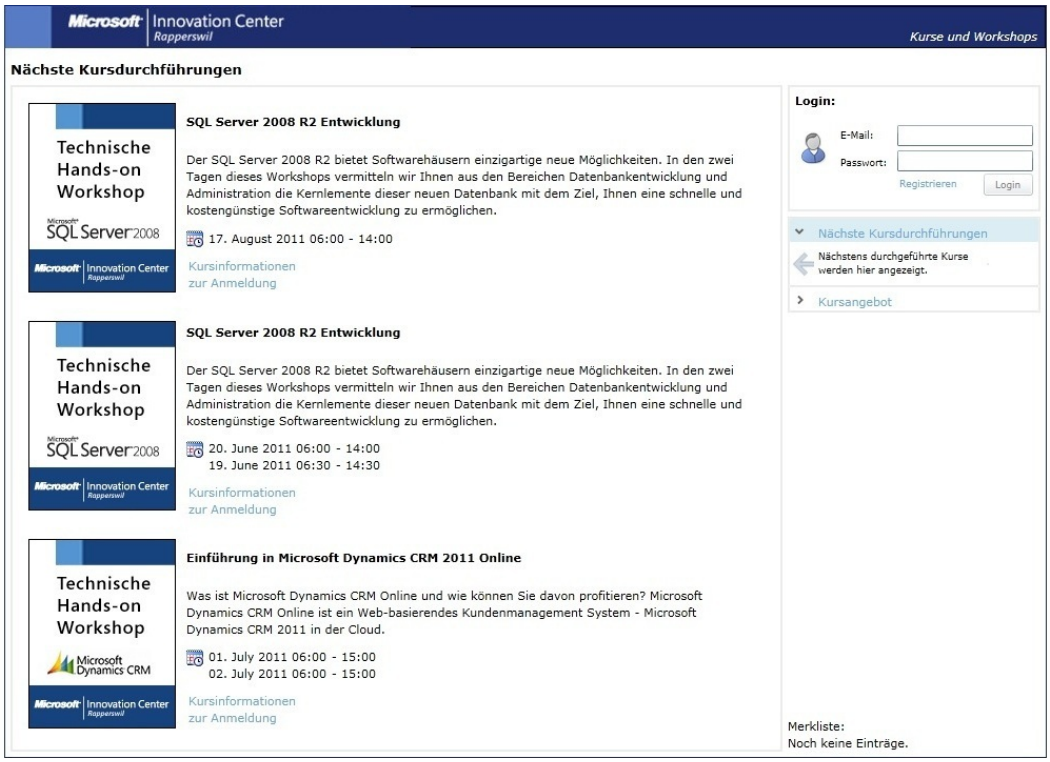



**Abbildung (5.1)** Screenshot der Weboberfläche

Mit der Realisierung des Projektes kann nun eine merkliche Verbesserung der Integration des Dynamics CRM in die Abläufe der Kursadministration verzeichnet werden.

Rückblickend kann zudem gesagt werden, dass einige der eingesetzten Technologien zur Umsetzung des verhältnismässig einfachen Anwendungsfalls der Kursverwaltung nicht zwingend nötig gewesen wären. Trotzdem, oder vielleicht gerade deshalb, kann die eingesetzte Architektur nun aber für künftige Erweiterungen oder andere Anwendungsfälle von

CRM Anbindungen auf dem Web als Vorlage dienen. Das Projekt steht auf einer soliden Basis, um allfällige Weiterentwicklungen oder Erweiterungen ohne grossen Konzeptaufwand umzusetzen.

# Teil II

# SW-Projektdokumentation

# 6 Projekt Management

## 6.1 Projektorganisation

Das Projektteam besteht aus zwei einander gleichgestellten Personen: Clemens Meier und Silvan Gacond. Prof. Hansjörg Huser betreut das Projekt. Ausserdem wird das Team von Jürg Jucker, einem Mitarbeiter des INS mit grosser CRM Erfahrung unterstützt. Er hat zusätzlich auch die Sichtweise eines potentiellen Endanwenders da er selber als Kursleiter für Kurse des INS fungiert.

Der Projektverlauf soll nach einer agilen Methode, angelehnt an die bekannten Vorgehensweisen von „Scrum“ überwacht und unterstützt werden. Da bei einer Teamgrösse von zwei Personen die konsequente Umsetzung von Scrum keinen Sinn macht, werden nur ausgewählte Ansätze und Artefakte daraus übernommen.

Obwohl beide Teammitglieder in allen Bereichen mitarbeiten, wurden folgende Hauptverantwortlichkeiten definiert:

**Clemens Meier** Service Architektur, Domain-Schicht, Projekt Management

**Silvan Gacond** CRM Erweiterungen, Silverlight Architektur, Infrastruktur

Die Bachelorarbeit wird mit 12 ECTS an den Studiengang angerechnet. Dies ergibt eine Sollarbeitszeit von 360 Stunden pro Teammitglied. Geteilt durch die 16 Wochen, die dem Projekt zur Verfügung stehen, ergibt dies eine durchschnittliche Arbeitszeit von 22.5 Stunden pro Woche. Da beide Teammitglieder neben der Bachelorarbeit einem Nebenerwerb nachgehen, wird diese Arbeitszeit flexibel auf die Projektzeit aufgeteilt und regelmässig Bilanz gezogen. Für die Planung wird mit 22.5 Stunden pro Teammitglied und Woche geplant.

## 6.2 Management Abläufe

### 6.2.1 Projektplanung

Die Planung erfolgt nach dem Vorbild von Scrum nach Sprints. Diese wurden am Anfang des Projektes auf die verfügbare Zeit eingeteilt und enden jeweils mit einem Meilenstein. Jedem Sprint werden zu Beginn die zu erledigenden Aufgaben zugewiesen und das Ziel des Sprints festgelegt.

### 6.2.2 Fortschrittsüberwachung

Da beide Teammitglieder vorwiegend an den Arbeitsplätzen an der HSR am Projekt arbeiten, werden tägliche kurze Sitzungen (angelehnt an „Daily Scrum“) abgehalten an denen der Stand der einzelnen Aufgaben kurz durch besprochen und die nächsten Aufgaben definiert werden.

Wöchentlich, jeweils montags, findet zudem eine Sitzung mit dem Betreuer statt, wo er über den aktuellen Stand sowie allfällige Probleme orientiert wird.

Am Ende der letzteren Sprints wurde jeweils der aktuelle Stand mit Jürg Jucker besprochen. Einerseits aus Sicht des Endanwenders, andererseits auch als technischer Ratgeber bezüglich CRM und Architektur.

## 6.3 Artefakte

**Sprintplanung** Grobe Planung der einzelnen Sprints innerhalb der ganzen Projektdauer.

**Product Backlog** Übersicht der Work Items über das gesamte Projekt.

**Burndown Chart** Grafische Übersicht des Projektfortschritts.

**Zeiterfassungen** Aufstellung der tatsächlichen Arbeitszeit der einzelnen Teammitlieder.

Die Artefakte sind im Anhang dieser Dokumentation zu finden (siehe *Projektmanagement* auf Seite 143).

# 7 Anforderungen

## 7.1 Szenarios

### 7.1.1 Suche nach Kursen

Peter Müller, ein potentieller Kursteilnehmer sucht im Internet nach Kursen über Microsoft SQL Server. Nach der Suche bei Google stösst er auf die Kurswebseite des Microsoft Innovation Centers der HSR und betrachtet das dort ausgeschriebenen Kursangebot. Der angebotene Kurs „Microsoft SQL Server 2008 R2“ ist schnell gefunden. Durch die sachliche und übersichtliche Darstellung überzeugt, registriert er sich und meldet sich zur zweiten Durchführung des Kurses an.

### 7.1.2 Information über neue Kurse im Interessebereich

Hans Meier (.NET Softwareentwickler) will immer auf dem neusten Stand der Technik bleiben. Da er bereits ein Jahr zuvor an einem anderen Kurs an des Microsoft Innovation Centers teilgenommen hat, ist er schon im Kursanmeldungssystem des Microsoft Innovation Centers registriert. Wegen des bereits besuchten Kurses ist er in einer Marketingliste eingetragen und wird automatisch mittels Newsletter auf den neu ausgeschriebenen Kurs über Smart/Rich Client Development (Windows Presentation Foundation WPF) aufmerksam gemacht.

### 7.1.3 Erfassen und Ausschreibung eines Kurses

Paul Mäder ist Berater am Microsoft Innovation Center und zuständig für Weiterbildung in Microsoft SharePoint und Microsoft Dynamics. In einem halben Jahr sollen die geplanten Kurse für Microsoft Dynamics CRM 2011 beginnen. Der Kurs wird nur durchgeführt, wenn sich auch genügend Interessenten melden. Er will deshalb potentielle Kursteilnehmer so schnell wie möglich erreichen. Die Ausschreibung des Kurses soll die potentiellen Kursteilnehmer sofort ansprechen, damit sie sich für den Kurs anmelden.

### 7.1.4 Verwaltung eines Kurses

Damit die Vorbereitungen für den Kurs optimal koordiniert werden können, erstellt Paul Mäder eine ToDo-Liste mit Hilfe der Aktivitäten im CRM. Verschiedenste Aufgaben

kann er anhand dieser ToDo-Liste an seine Mitarbeiter delegieren. Ausserdem kann er die angemeldeten Kursteilnehmer über die definitive Nichtdurchführung oder eine Raumänderung benachrichtigen.

### 7.1.5 Feedback nach dem Kurs

Nach Abschluss einer Kursdurchführung kann Paul Mäder eine Zufriedenheitsumfrage an die Teilnehmer senden. Das Feedback ermöglicht ihm, seinen Kurs dauernd zu verbessern.

### 7.1.6 Information während des Kurses

Klara Tobler hat sich für den Kurs „Einführung in Microsoft Dynamics CRM 2011“ angemeldet. Damit sich die Kursteilnehmer im Vorfeld des Kurses bereits mit dem Thema befassen können, hat der Kursleiter Paul Mäder die Folien bereits online gestellt. Diese kann Klara Tobler nun im exklusiv für die Kursteilnehmer zugänglichen Bereich herunterladen. Bei einer Änderung wird sie per Email benachrichtigt.

### 7.1.7 Feedback nach dem Kurs

Nach dem Kurs möchte Klara Tobler die Zufriedenheitsumfrage ausfüllen, da sie hell begeistert ist vom soeben absolvierten Kurs und dies Paul Mäder unbedingt mitteilen will. Dennoch würde sie gerne das nächste Mal etwas mehr über die Anpassungsmöglichkeiten von Dynamics CRM erfahren und vermerkt dies im Feld "Weiteres".

### 7.1.8 Marketing und Administration

Maria Widmer ist zuständig für das Marketing der Kurse am INS. Sie will so viele potentielle Kursteilnehmer wie möglich erreichen, ohne das sich diese über SPAM beklagen. Die öffentliche Ausschreibung der Kurse ermöglicht ihr auf eine bequeme Art und Weise neue Interessenten zu erreichen. Die Kursinformationen werden von den Kursleitern erstellt und müssen von Maria Widmer nur noch kontrolliert werden. In gewissen Zeitabständen sendet Maria Widmer Werbung mittels eines Newsletters an Interessenten. Registrierte Interessenten/Kursteilnehmer erhalten gezielt nur die Kursinformationen, welche für sie auch interessant sein könnten.

## 7.2 Aktoren

### 7.2.1 Interessent

Der Interessent informiert sich über das Internet über die angebotenen Kurse. Er ist ein potentieller Kursteilnehmer.

#### Profil

Da die Kurse am INS ein fortgeschrittes Informatikverständnis voraussetzen, ist ein Kursinteressent typischerweise ein fortgeschrittener Computeranwender oder Informatiker. Es können also überdurchschnittliche Kenntnisse im Umgang mit Computern vorausgesetzt werden.

#### Erwartungen

Der Interessent will sich so effizient wie möglich Informationen über die angebotenen Kurse beschaffen, was eine übersichtliche Oberfläche voraussetzt.

### 7.2.2 Kursteilnehmer

Der Kursteilnehmer hat sich bereits für einen Kurs angemeldet und wird an einem Kurs teilnehmen. Er kann Interessent für weitere Kurse sein.

#### Profil

Für den Kursteilnehmer gelten die gleichen Voraussetzungen wie für den Interessenten.

#### Erwartungen

Die Weboberfläche des Kursadministrationstools dient dem Kursteilnehmer zur weiterführenden Information während des Kurses.

### 7.2.3 Administration

Der Administrationsmitarbeiter arbeitet innerhalb Dynamics CRM und erledigt sämtliche Verwaltungsaufgaben im Kursverwaltungssystem. Er kann die Registrierungsdaten der Teilnehmer wie auch der Interessenten jederzeit anpassen. Dies ist nötig wenn ein Teilnehmer sich z.B. telefonisch meldet und eine Anmeldung abändern möchte.

#### Profil

Der Administrator arbeitet täglich mit Microsoft Dynamics CRM.

#### Erwartungen

Der Administrator kümmert sich um die Koordination von Kursen, nimmt Anmeldungen telefonisch entgegennehmen welche er ins System eintragen will oder kümmert sich um Werbekampagnen. Er will seine Arbeit in erster Linie so effizient wie möglich erledigen. Da er sich die Abläufe innerhalb Dynamics CRM bereits gewohnt ist, sollen sich die neuen Funktionalitäten aus seiner Sicht weitgehend daran anlehnen.

### 7.2.4 Kursleiter

Der Kursteileiter leitet mindestens einen Kurs. Soll er auch Administrationsaufgaben übernehmen, ist er gleichzeitig auch Administrationsmitarbeiter. Das muss aber nicht zwingend der Fall sein.

#### Profil

Beim Kursleiter können sehr gute Informatikkenntnisse vorausgesetzt werden. Falls nötig, kann der Umgang mit dem bestehenden Microsoft Dynamics CRM anhand einer kurzen Schulung angeeignet werden.

#### Erwartungen

Der Kursleiter will über das System möglichst viele Interessenten für seine Kurse zu erreichen. Die Plattform soll ihm die Organisation des Kurses erleichtern. Die Kursteilnehmer will er effizient informieren können.

# 7.3 Use Cases

## 7.3.1 Übersicht

Nachfolgend eine Übersicht über die maximal implementierten Use Cases. Die optional deklarierten Use Cases sind lediglich „Brief“, also in Kurzform beschrieben.

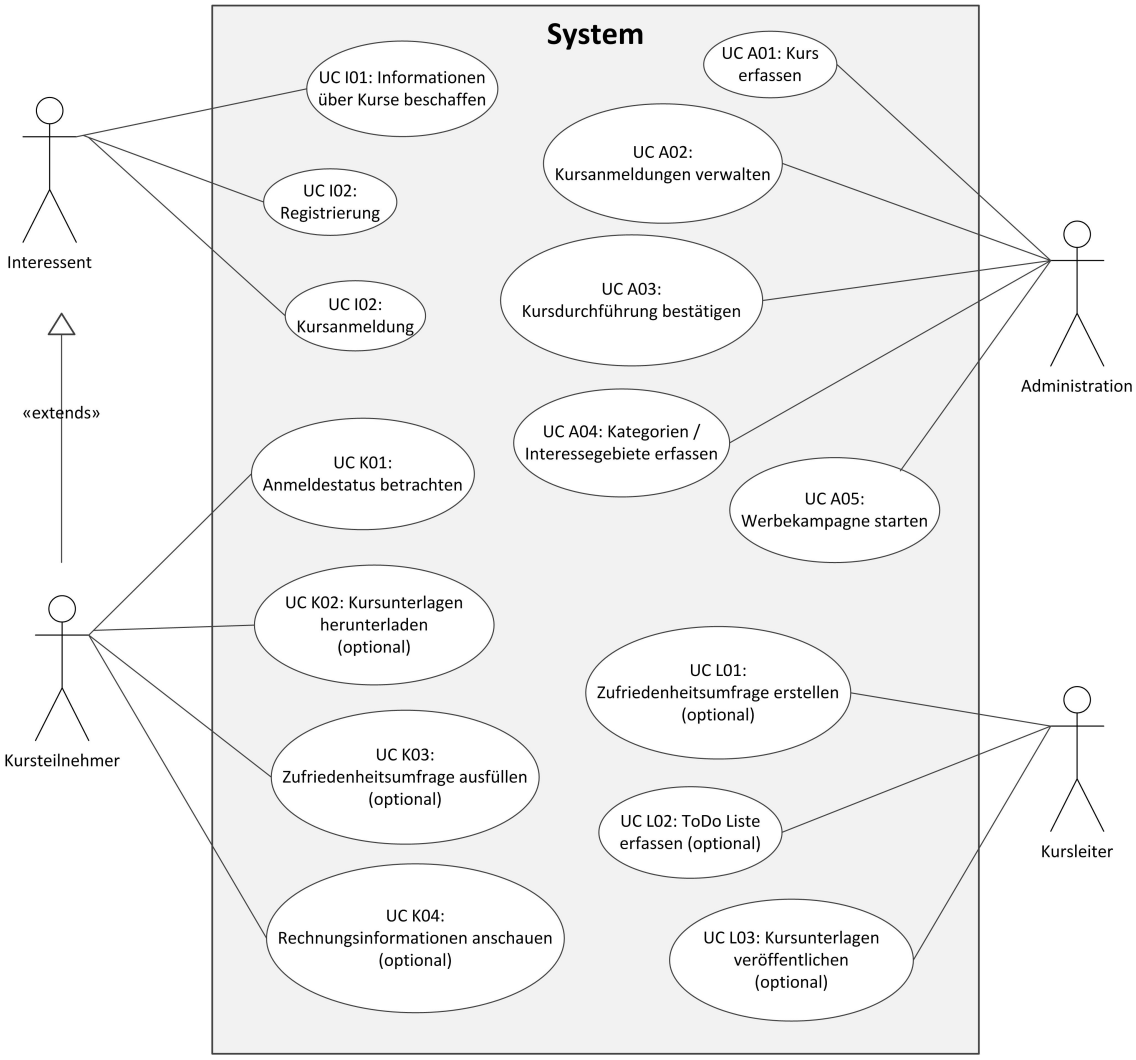



**Abbildung (7.1)** Übersicht über die Use Cases

### 7.3.2 Interessent

#### UC I01: Informationen über Kurse beschaffen

Der Interessent beschafft sich Informationen über die anstehenden Kurse, indem er die Webseite betrachtet. Er kann Kurse nach Zeitraum und Interessegebiet suchen.

| | |
|---|---|
| **Beschreibung** | Der Interessent möchte Informationen über einen Kurs einsehen. |
| **Primäraktor** | Interessent |
| **Vorbedingungen** | • Der Kurs ist bereits im System erfasst und veröffentlicht. |
| **Ablauf** | 1. Der Interessent besucht die Website des INS.<br>2. Er navigiert zum entsprechenden Kursthema.<br>3. Die Kursinformationen werden angezeigt. |

**Tabelle (7.1)** Use Case I01

#### UC I02: Registrierung/Registrierungsinformationen bearbeiten

Der Interessent registriert sich, damit er sich für einen Kurs anmelden kann, bei welchem eine Registrierung erforderlich ist.

| | |
|---|---|
| **Beschreibung** | Der Interessent hat sich im System registriert. |
| **Primäraktor** | Interessent |
| **Vorbedingungen** | • Der Interessent ist anhand seiner E-Mail Adresse noch nicht registriert. |

| | |
|---|---|
| **Ablauf** | 1. Der Interessent navigiert zur Eingabemaske für die Registrierungsdaten.<br>2. Die Eingabemaske um die persönlichen Daten einzugeben erscheint.<br>3. Er gibt seine persönlichen Daten ein.<br>a) Alle Felder werden korrekt ausgefüllt.<br>b) Falls eines oder mehrere Felder nicht korrekt ausgefüllt wird, erscheint eine Warnung.<br>4. Der Interessent bestätigt die Registrierung.<br>5. Das System trägt den Interessenten als Kontakt im CRM ein. |

**Tabelle (7.2)** Use Case I02

### UC I03: Kursanmeldung

Der Interessent meldet sich für einen Kurs an. Die meisten Kurse setzen eine vorherige Registrierung voraus. Um sich für diese anzumelden, muss der Interessent registriert und im System angemeldet sein. Es gibt aber auch Kurse, welche keine Registrierung erfordern.

| | |
|---|---|
| **Beschreibung** | Der Interessent meldet sich für eine Kursdurchführung an. |
| **Primäraktor** | Interessent |
| **Vorbedingungen** | • Der Kurs ist im System eingetragen und publiziert.<br>• Eine entsprechende Kursdurchführung ist im System eingetragen und publiziert.<br>• Die Kursdurchführung verfügt noch über freie Plätze. |

| **Ablauf** | 1. Der Interessent navigiert zum gewünschten Kurs und der gewünschten Durchführung.<br>2. Er navigiert weiter zur Anmeldung der Kursdurchführung.<br>a) Er ist noch nicht eingeloggt: das System fragt nach, ob sich bereits registriert hat und leitet ihn automatisch zur Anmeldeseite oder hebt den Login-Bereich grafisch hervor.<br>b) Der Interessent loggt sich entweder am System ein oder registriert sich neu (UC I02).<br>3. Das System erstellt die Kursanmeldung, generiert das Bestätigungsmail und schiebt den Status der Anmeldung auf „Bestätigt“ (confirmed). |
|---|---|

**Tabelle (7.3)** Use Case I03

### 7.3.3 Kursteilnehmer

#### UC K01: Anmeldestatus betrachten

Der Kursteilnehmer möchte den Status der angemeldeten Kursdurchführung einsehen, um Gewissheit über den Durchführungsentscheid zu haben.

| **Beschreibung** | Der Kursteilnehmer möchte den Durchführungsstatus einsehen. |
|---|---|
| **Primäraktor** | Interessent |
| **Vorbedingungen** | • Der Kursteilnehmer ist bereits für den Kurs angemeldet.<br>• Der Kursteilnehmer ist im System eingeloggt.<br>• Eine entsprechende Kursdurchführung ist im System eingetragen und publiziert. |

| **Ablauf** | 1. Der Kursteilnehmer navigiert zum gewünschten Kurs und der gewünschten Durchführung.<br>2. Das System zeigt den Status der Kursdurchführung an. |
|---|---|

**Tabelle (7.4)** Use Case K01

### UC K02: Kursunterlagen herunterladen (optional)

Vorbedingung: Kursteilnehmer ist registriert und eingeloggt. Die Teilnahme wurde für den betreffenden Kurs bestätigt.

Der Kursteilnehmer kann die durch den Kursleiter bereitgestellten Kursunterlagen herunterladen und einsehen. Damit kann er sich schon vor Kursbeginn auf den Kurs vorbereiten, oder nach dem Kurs das Gelernte nochmals auffrischen.

### UC K03: Zufriedenheitsumfrage ausfüllen (optional)

Vorbedingung: Kursteilnehmer hat einen Kurs bereits besucht, ist registriert und eingeloggt.

Der Kursteilnehmer erhält nach dem Kurs eine Anfrage an der Zufriedenheitsumfrage teilzunehmen. Die automatische Auswertung der Rückmeldungen möglichst vieler Kursteilnehmer ermöglicht dem Kursleiter und den zuständigen Administratoren eine Qualitätskontrolle der Kursdurchführung.

### UC K04: Rechnungsinformationen anschauen (optional)

Vorbedingung: Der Kursteilnehmer ist registriert und eingeloggt. Die Teilnahme für mindestens einen Kurs wurde bestätigt.

Der Kursteilnehmer kann Rechnungsinformationen betrachten.

### 7.3.4 Administration

#### UC A01: Kurs und Kursdurchführung erfassen

Der Administrationsmitarbeiter erfasst die Randdaten für einen Kurs und dessen Durchführungsdaten. Dazu gehören eine kurze Beschreibung, Ort und Zeit der Durchführung und weitere Informationen. Anschliessend publiziert er den Kurs und die Durchführung im Internet.

| | |
|---|---|
| **Beschreibung** | Der Administrationsmitarbeiter möchte einen Kurs und eine zugehörige Durchführung erfassen. |
| **Primäraktor** | Administration |
| **Vorbedingungen** | • Der Mitarbeiter ist an Dynamics CRM angemeldet. |
| **Ablauf** | 1. Der Mitarbeiter navigiert zur Kursliste.<br>2. Er eröffnet einen neuen Kurs und gibt die Informationen ein.<br>3. Er speichert den Kurs ab und öffnet die Durchführungsliste dazu.<br>4. Er eröffnet eine neue Kursdurchführung, erfasst die nötigen Informationen und Durchführungstage.<br>5. Er navigiert zurück zum Kurs und stellt den Status auf „veröffentlicht“.<br>6. Er öffnet nochmals die Kursdurchführung und schiebt auch deren Status auf „veröffentlicht“. |

**Tabelle (7.5)** Use Case A01

#### UC A02: Kursanmeldungen verwalten

Der Administrationsmitarbeiter kann sich Informationen über die angemeldeten Interessenten beschaffen. Er kann Anmeldungen streichen, oder zusätzlich Anmeldungen eintragen. Zum Beispiel wenn sich ein Interessent per Telefon meldet.

| | |
|---|---|
| **Beschreibung** | Der Administrationsmitarbeiter möchte einen Kurs und eine zugehörige Durchführung erfassen. |
| **Primäraktor** | Administration |
| **Vorbedingungen** | • Der Mitarbeiter ist an Dynamics CRM angemeldet.<br>• Ein Kurs ist bereits erfasst und publiziert.<br>• Es haben sich bereits Teilnehmer für den Kurs angemeldet. |
| **Ablauf** | 1. Der Mitarbeiter navigiert zur Kursliste.<br>2. Er öffnet den entsprechenden Kurs und die entsprechende Durchführung.<br>3. Er navigiert zur Teilnehmerliste und nimmt da die entsprechenden Manipulationen vor. |

**Tabelle (7.6)** Use Case A02

### UC A03: Kursdurchführung bestätigen/absagen

Vor der Durchführung eines Kurses, wird der Kurs manuell oder automatisch bestätigt. Anhand der Anzahl der Anmeldungen muss der Kursleiter entscheiden, ob der Kurs definitiv stattfindet. Er kann auf einfache Weise die Kursteilnehmer benachrichtigen.

| | |
|---|---|
| **Beschreibung** | Der Administrationsmitarbeiter möchte den Status einer Kursdurchführung ändern. |
| **Primäraktor** | Administration |
| **Vorbedingungen** | • Der Mitarbeiter ist an Dynamics CRM angemeldet.<br>• Ein Kurs ist bereits erfasst. |

| | |
|---|---|
| **Ablauf** | 1. Der Mitarbeiter navigiert zur Kursliste.<br>2. Er öffnet den entsprechenden Kurs und die entsprechende Durchführung.<br>3. Er schiebt den Status der aktuellen Durchführung auf den gewünschten Status.<br>4. (Optional) Mit Hilfe der CRM Funktionalität kann er die Teilnehmer mittels Massen E-Mail benachrichtigen. |

**Tabelle (7.7)** Use Case A03

### UC A04: Kategorien/Interessengebiete erfassen

Vorbedingung: Der Administrator ist am System angemeldet.
Der Administrator kann Kategorien und Interessegebiete erfassen welche zur leichteren Gruppierung und besserer Übersicht über die angebotenen Kurse dienen sollen.

| | |
|---|---|
| **Beschreibung** | Der Administrationsmitarbeiter möchte weitere Interessengebiete erfassen. |
| **Primäraktor** | Administration |
| **Vorbedingungen** | • Der Mitarbeiter ist an Dynamics CRM angemeldet. |
| **Ablauf** | 1. Der Mitarbeiter navigiert zu den Interessegebieten.<br>2. Er nimmt an der entsprechenden Stelle die Änderung vor. |

**Tabelle (7.8)** Use Case A04

### UC A05: Werbekampagne starten

Der Administrator kann Informationen über anstehende Kurse an die registrierten Interessenten verschicken. Die registrierten Interessenten erhalten nur diejenigen Informationen,

welche zu den Interessegebieten gehören welche sie bei der Registrierung angegeben haben.

| | |
|---|---|
| **Beschreibung** | Der Administrationsmitarbeiter möchte eine Werbekampagne starten. |
| **Primäraktor** | Administration |
| **Vorbedingungen** | • Der Mitarbeiter ist an Dynamics CRM angemeldet. |
| **Ablauf** | 1. Der Mitarbeiter navigiert zu den Marketinglisten, die nach Kursdurchführung erstellt wurden.<br>2. Er startet eine Marketingkampagne mit den gewünschten Listen. |

**Tabelle (7.9)** Use Case A05

### 7.3.5 Kursleiter

#### UC L01: Zufriedenheitsumfrage erstellen (optional)

Vorbedingung: Ein Kurs ist bereits erfasst. Der Kursleiter ist am System angemeldet.
Der Kursleiter erstellt eine Zufriedenheitsumfrage, anhand welcher er Rückmeldungen der Kursteilnehmer zur Qualitätskontrolle erhalten will.

#### UC L02: ToDo-Liste erfassen (optional)

Vorbedingung: Ein Kurs ist bereits erfasst. Der Kursleiter ist am System angemeldet.
Der Kursleiter kann eine ToDo erfassen um die Vorbereitungen für den Kurs zu koordinieren.

#### UC L03: Kursunterlagen veröffentlichen (optional)

Vorbedingung: Ein Kurs ist bereits erfasst. Der Kursleiter ist am System angemeldet.
Der Kursleiter kann Präsentationsfolien oder weitere Unterrichtsunterlagen für alle Kursteilnehmer über die Plattform verfügbar machen.

## 7.4 Nichtfunktionale Anforderungen

Die folgenden nichtfunktionalen Anforderungen wurden ausgehend vom bekannten „FURPS Modell“ [PE05] ausgelegt und leicht erweitert.

### 7.4.1 Functionality (Funktionalität)

Die oben mit Use Cases und User Stories definierte Funktionalität soll grundsätzlich erweiterbar sein. Das Design und die Implementierung soll so weit offen sein, dass Erweiterungen möglich sind, ohne die bestehende Funktionalität weitgehend umstellen zu müssen.

### 7.4.2 Usability (Benutzbarkeit)

Da die ganze Anwendung von zwei Plattformen aus bedient wird, sind die Anforderungen an die Benutzbarkeit der beiden Plattformen unterschiedlich: Das Web-Frontend in Silverlight soll grundsätzlich ohne weitere Erklärungen zu bedienen sein. Da das Zielpublikum des Web-Frontends beliebige Benutzer aus dem Internet sein können, kann man hier nicht von Vorkenntnissen ausgehen. Die Back-End Applikation innerhalb Dynamics CRM soll sich möglichst gut in die bestehenden Workflows des Dynamics CRM eingliedern. Geübte CRM Benutzer sollen die gewünschten Funktionalitäten an den Stellen finden, wo sie von der restlichen CRM Funktionalität her zu erwarten wären.

### 7.4.3 Reliability (Zuverlässigkeit)

Die Applikation soll im normalen Betrieb ohne aussergewöhnliche Einflüsse (Netzwerkprobleme, Serverfehler, etc.) fehlerfrei funktionieren. Fehleingaben von Benutzern sollen mit aussagekräftigen Meldungen quittiert werden und nicht zu unbekannten Fehlern im System führen.

### 7.4.4 Performance (Effizienz)

Um die Web Applikation ohne Einschränkungen bedienen zu können, soll eine übliche Performance für Internetanwendungen realisiert werden können. Da diese Aussage stark von weiteren, unkontrollierbaren Parametern abhängt, kann sie leider nicht genauer quantisiert werden. Auf die sinnvollen Grundsätze wie z.B. asynchrone Aufrufe, Lazy Loading, wiederverwenden von Views, etc. ist aber zu achten. Die Back-End Applikation

im CRM soll sich von der Performance her nicht merklich von den restlichen CRM Komponenten unterscheiden.

### 7.4.5 Supportability (Wartbarkeit)

Die Komponenten der Software sollen grundsätzlich so aufgebaut sein, dass sie wiederverwendet werden können, sofern der Anwendungsfall Sinn macht. Ausserdem sollen alle Texte grundsätzlich in separaten Ressourcenfiles abgelegt sein, um in einer späteren Entwicklung eine allfällige Mehrsprachigkeit integrieren zu können. Alle Erweiterungen an Dynamics CRM sollen ausserdem sinnvoll gekapselt sein, um eine schnelle Installation oder Deinstallation auf einem anderen CRM System zu ermöglichen.

### 7.4.6 Security (Sicherheit)

Bei der Architektur der Software ist darauf zu achten, dass die CRM Web-Services, welche grundsätzlich einen vollen Zugriff auf das gesamte System ermöglichen, nicht vom Internet aus erreichbar sein müssen. Nur erforderliche Operationen zur Kursinformation und Kursanmeldung sollen über das Internet ausgeführt werden können.

# 8 Analyse

## 8.1 Domain Analyse

### 8.1.1 Domain Model

Für die Aktoren (siehe *Aktoren* auf Seite 29) Interessent und Kursteilnehmer müssen Daten über das Internet zugänglich gemacht werden. Die erforderlichen Daten sind im folgenden Domain Model dargestellt.

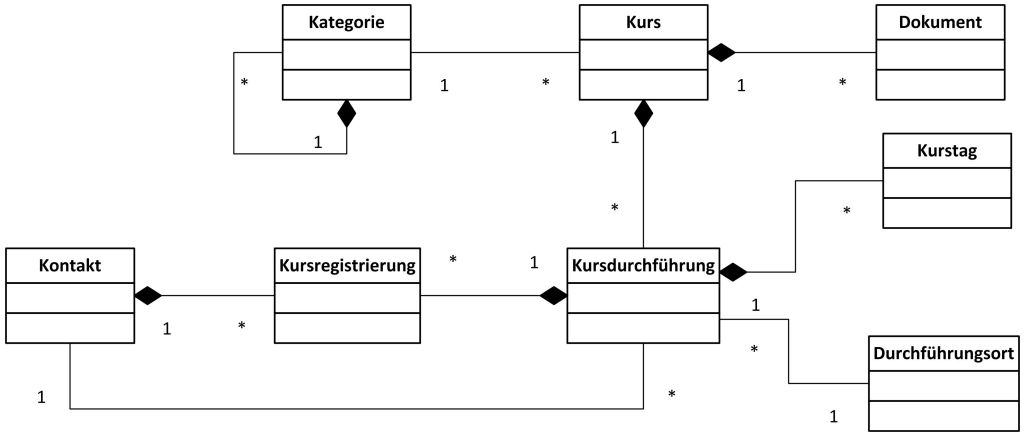



**Abbildung (8.1)** Domain Model für die über das Internet zugängliche Plattform.

Neben einfachen Beziehungen und Kompositionen sticht die zusätzliche Verbindung zwischen „Kursdurchführung“ und „Kontakt“ hervor. Diese zeigt die Zuordnung eines Kursleiters an eine Kursdurchführung.

### 8.1.2 Beschreibung der Domain Objekte

#### Kurs

Ein Kurs beinhaltet einen Titel für den Kurs, die Kursbeschreibung und weitere allgemeine Informationen über einen Kurs.

### Kursdurchführung

Eine Kursdurführung enthält die Daten, welche zu einer einmaligen Durchführung eines Kurses benötigt werden. Ein Kurs kann mehrere Kursdurchführungen haben.

### Kontakt

Ein Kontakt beinhaltet Informationen über einen Kursteilnehmer oder einen Kursleiter.

### Kurstag

Ein Kurstag enthält Informationen zum Beginn und Ende des Kurses an einem bestimmten Tag. Eine Kursdurchführung kann sich über mehrere Kurstage erstrecken.

### Durchführungsort

Pro Kursdurchführung ist nur ein Durchführungsort anzugeben. Dieser ist im CRM der Durchführungsort des ersten Kurstages der Kursdurchführung.

### Kursregistrierung

Eine Kursregistrierung verknüpft einen Kontakt (Kursteilnehmer) mit einer Kursdurchführung. Sie kann als ungültig deklariert oder bestätigt werden.

### Dokument

Dokumente sind Unterlagen, welche der Kursleiter seinen Kursteilnehmern zu Verfügung stellen kann.

### Kategorie

Kategorien dienen zur Kategorisierung von Kursen. Damit können Kurse nach Interessegebiet gesucht werden. Kategorien sind hierarchisch gegliedert.

## 8.2 Verwendbare Funktionalität von Microsoft Dynamics CRM

Einige eingebaute Funktionen des Dynamics CRM können „out of the box“ direkt verwendet werden. Durch die Anpassbarkeit der Entitäten können diese leicht abgeändert werden und dadurch das eigene Datenmodell ergänzen. Folgende Entitäten lassen sich so für die Anwendung direkt einsetzen:

**Contact (Kontakt)** Die eingebauten Kontakte des Dynamics CRM können für die Kursteilnehmer wie auch für die Kursleiter eingesetzt werden. Da teilweise externe Kursleiter zum Einsatz kommen können und die Administration innerhalb des CRM von Angestellten der Administration erledigt werden kann, kann nicht davon ausgegangen werden dass jeder Kursleiter als Benutzer innerhalb Dynamics CRM geführt wird.

**Site (Gebäude)** Die CRM-eigene Verwaltung von Gebäuden und Räumen (inkl. Ressourcenplanung) kann sehr gut direkt für die Reservation von Kursräumen verwendet werden. Dies bedingt, dass einzelne Kurstage als Service-Aktivität erfasst werden, um die Ressource auch aktiv zu belegen.

**Service Activity (Service-Aktivität)** Wie bereits erwähnt, werden einzelne Kurstage sinnvollerweise als Service-Aktivitäten erfasst. Diese werden direkt in die Ressourcenverwaltung eingebunden und sorgen so dafür, dass die Ressourcen (Räume und Kursleiter) nicht mehrfach belegt werden können. Dies geht ohne grosse Anpassungen der Entitäten, lediglich die Beziehung zwischen Kursdurchführung und Service-Aktivität muss erfasst werden können. Ausserdem wird ein Service-Typ für einen ganzen Kurstag erfasst werden müssen.

**Subject (Betreff)** Innerhalb Dynamics CRM können sogenannte „Subjects“ oder Betreffs erfasst werden, welche global über das gesamte System Gültigkeit haben. Sie sind hierarchisch aufgebaut und ermöglichen die Zuordnung verschiedenster Entitäten zu den jeweiligen Betreffs. Für die Kursverwaltung wird von einem Basisbetreff ausgegangen, welcher z.B. „Kurse“ heisst und sämtliche Kursthemen als Unterbetreffs listet.

Um die internen Abläufe umzusetzen, kann die Plugin Architektur von Dynamics CRM verwendet werden. Die Backend Abläufe sollen mit normalen Formularen und Ansichten umgesetzt werden, damit diese Abläufe ähnlich aussehen, wie die normalen bekannten CRM Abläufe.

## 8.3 Architekturziel

Nach Analyse der Anforderungen der Domain- und der CRM Ebene, wurde als Ziel definiert, diese beiden Schichten physikalisch zu trennen.

### 8.3.1 Abtrennung aus Sicherheits- und Lizensierungsgründen

Die Clients die aus dem Internet den Service aufrufen, sollen Netzwerktechnisch sauber von der eigentlichen CRM Instanz abgetrennt werden. Der Server der die Zwischenschicht betreibt, steht dabei in einer demilitarisierten Zone (DMZ) und ist von aussen erreichbar. Einzig dieser Server sollte Zugriff auf die eigentlichen CRM Services haben, auf keinen Fall die Clients aus dem Internet.

Ein weiterer Vorteil besteht darin, dass durch die physikalische Abtrennung sozusagen die Zwischenschicht einen weiteren CRM Client darstellt. Es wird also nur eine zusätzliche „Client Access Licence“ (CAL) benötigt.

### 8.3.2 Abtrennung zur Reduzierung der Kopplung

Um trotz der Abhängigkeit von den Objekten des Dynamics CRM eine möglichst geringe Kopplung zu erreichen, wurde das Datenmodell des CRM ganz abgetrennt. In der Zwischenschicht existiert ein weiteres Datenmodell (`Tadm.Web.Domain`) welches mit Hilfe einer Connector Klasse (`Tadm.Web.CrmConnector`) an die CRM Services und die CRM Typen angekoppelt wird.

Durch diese Abtrennung ist es möglich, mit automatisierten Tests isoliert die CRM Verbindung zu testen. Oder aber die CRM Verbindung durch eine andere Klasse zu ersetzen, welche das System für die Tests mit Dummydaten versorgt. So kann die Kommunikation von der Webserviceschnittstelle des Frontends mit der Zwischenschicht isoliert getestet werden.

# 9 Technologien und Konzepte

## 9.1 Übersicht

Dieses Kapitel beschreibt Technologien und Konzepte, die für die Umsetzung des Kursanmeldungssystemes verwendet wurden.

## 9.2 Dynamics CRM Erweiterungen (xRM)

### 9.2.1 Microsoft Dynamics CRM / xRM

Neben den Grundfunktionen verfügt Dynamics CRM über viel Funktionalität zur Anpassung des Systems. So kann mit dem frei erhältlichen CRM-SDK auf verschiedenen Ebenen in das System eingegriffen werden, um es so den Bedürfnissen und internen Abläufen im Kunden-Management des entsprechenden Mandanten anzupassen. Diese Anpassbarkeit lässt sich aber auch nutzen um Funktionalitäten zu implementieren, die nicht der klassischen CRM Funktionalität entsprechen, sich aber weitgehend in den bestehenden Datenstamm eingliedern lassen. Natürlich liessen sich auch ganz CRM fremde Funktionalitäten implementieren, dies macht aber in den meisten Fällen keinen Sinn. Die ganze Anpassbarkeit wird unter dem Überbegriff xRM (e**x**tensible **r**elationship **m**anagement) geführt.

### 9.2.2 Möglichkeiten zur Erweiterung des CRM

Microsoft Dynamics CRM bietet grundsätzlich drei Ebenen an, an denen in die CRM Funktionalität eingegriffen werden kann. Als tiefste Ebene kann man die Plugins bezeichnen, welche synchron oder asynchron direkt in die Kerntransaktionen des Basissystems eingreifen können. Weiterer eigener Servercode lässt sich in eigenen Workflow-Aktivitäten einbetten, diese laufen immer asynchron und können vom Benutzer des CRM's einfach selber parametriert oder gesteuert werden. Um auf höchster Ebene in die Präsentationsschicht des CRM einzugreifen, lässt sich das Web-Frontend Clientseitig mit sogenannten Web-Ressourcen beeinflussen.

### 9.2.3 xRM Web Services

Auf allen drei angesprochenen Ebenen zum Eingriff ins CRM lassen sich die xRM Web Services ansprechen:

**Discovery Service** Da ein CRM Server mehrere Mandanten resp. Organisationen betreiben kann, muss der Client zuerst ermitteln können, welche Organisationen überhaupt auf dem entsprechenden Server existieren und wie die URL's der einzelnen Organization Services sind. Dazu kann der sogenannte Discovery Service eingesetzt werden. Er wird über einen spezifischen URL angesprochen, der normalerweise bei jeder CRM Installation gleich ist.

**Organization Service** Der Organization Service ist der umfangreichste der angebotenen Web Services. Er bietet die gesamte CRM Funktionalität die auch über das Web-Frontend angeboten wird. Natürlich lassen sich nur die Operationen ausführen, zu denen der angemeldete Benutzer berechtigt ist. Theoretisch lässt sich mit Hilfe dieses Services die gesamte CRM Oberfläche durch eine eigene ersetzen.

**Deployment Service** Mit dem Deployment Service lassen sich die verschiedenen Mandanten des CRM verwalten. So ist es möglich zur Laufzeit einen Mandanten hinzuzufügen oder zu löschen. Dies ermöglicht z.B. eine Demoplattform auf dem Internet, wo sich potentielle CRM Kunden einen Demo-Mandant einrichten können, welcher nach einer gewissen Zeit automatisch wieder gesperrt wird.

### 9.2.4 Plugin - Architektur

Wie bereits angesprochen lassen sich Plugins entweder asynchron oder synchron in die Event-Pipeline im Dynamics CRM einbinden. Das folgende Bild zeigt die Möglichkeiten, eigenen Code innerhalb der Transaktion anzubringen. Mögliche Punkte sind rot markiert:

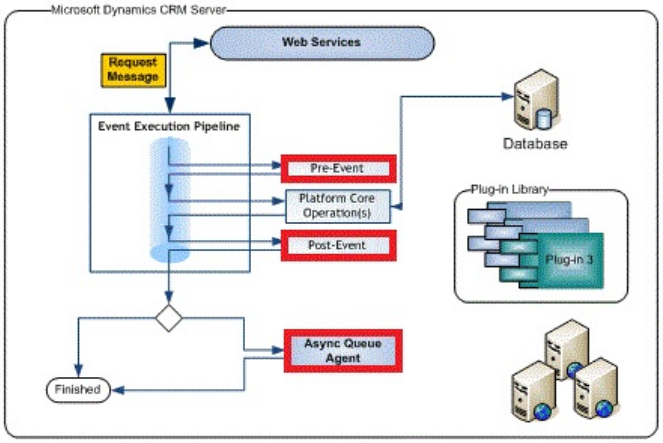



**Abbildung (9.1)** Dynamics CRM Event-Pipeline (vereinfacht) [Mic11]

Bei synchron ausgeführten Plugins lässt sich die Transaktion abbrechen. Dies ermöglicht also die Integration von komplexeren Validierungsmechanismen als die eingebauten Möglichkeiten des CRM zulassen. Eine andere praktische Methode ist die Statusänderung der verschiedenen Entitäten innerhalb der Synchronen Event-Pipeline zu implementieren. So wird verhindert, dass ein inkonsistenter Stand abgespeichert werden kann. Egal ob die Änderung nun vom Organization-Service oder vom normalen Web-Frontend ausgelöst wird.

Plugins bestehen grundsätzlich aus dem kompilierten Code einer .NET Assembly und können über die CRM Web-Services registriert und auf bestimmte Events gebunden werden. Um diesen Ablauf zu vereinfachen beinhaltet das CRM SDK das sogenannte „Plugin Registration Tool“. Es ermöglicht die grafische Registrierung von CRM Plugins.

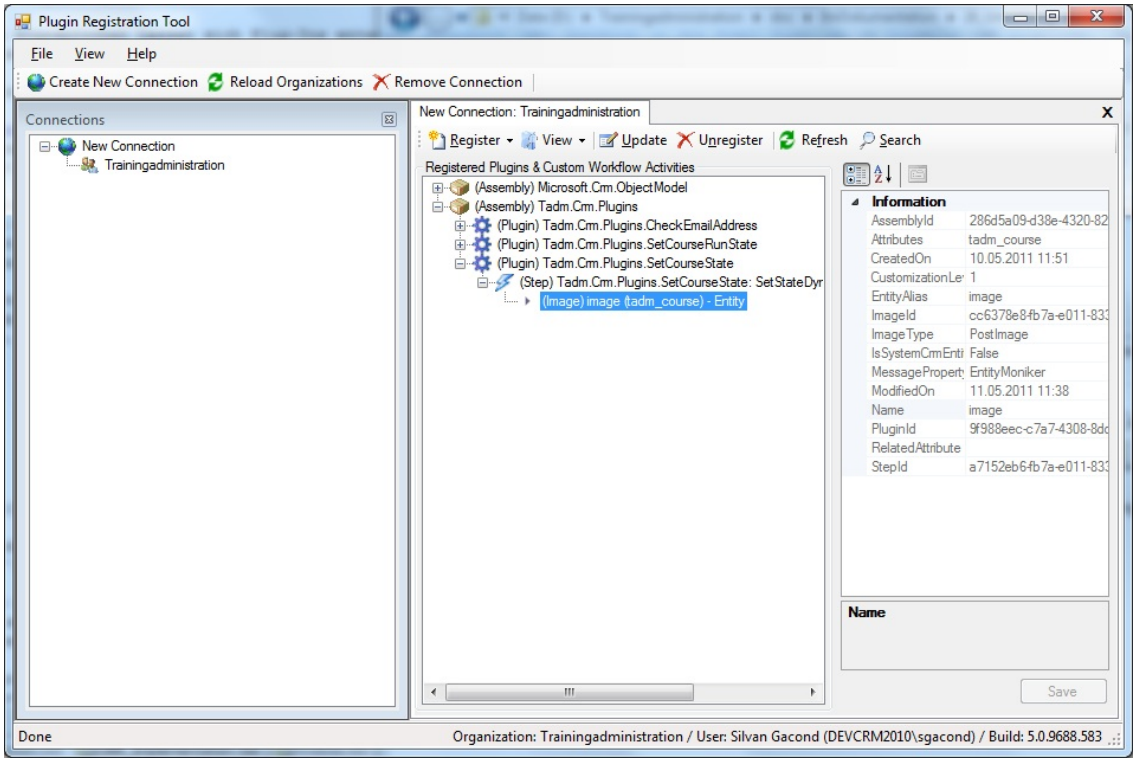



**Abbildung (9.2)** Screenshot des Plugin Registraion Tools

Neben der Registrierung der Plugin-Assemblies können die Plugins mit Hilfe dieses Tools auch konfiguriert werden. Damit das Plugin ausgeführt wird, ist mindesten ein sogenannter Plugin-Step notwendig. Er konfiguriert den Event, wann das Plugin ausgelöst wird und bestimmt an welcher Stelle der Event-Pipeline das Plugin ausgeführt wird. Ausserdem könnte noch ein String zur Konfiguration sowie der Ausführungskontext angegeben werden.

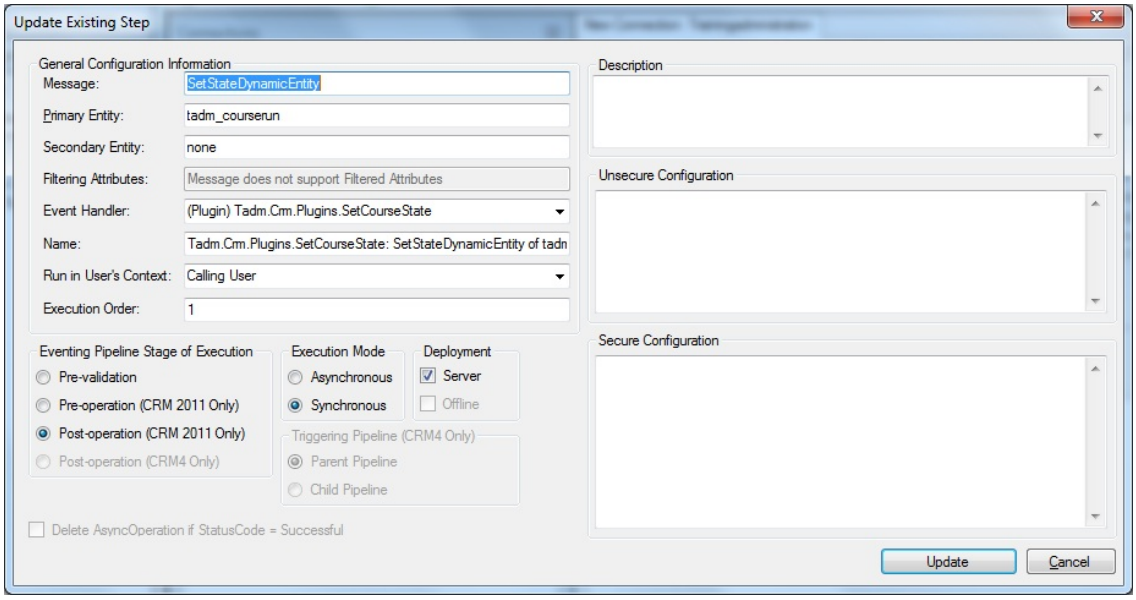



**Abbildung (9.3)** Screenshot der Konfigrationsmöglichkeiten eines Plugin-Steps)

Zusätzlich kann jedem Plugin-Step noch eines oder mehrere „Images“ mitgegeben werden. Dies sind „Schnappschüsse“ einer bestimmten Entität vor- oder nach der Ausführung der Core-Operation (Pre-Image oder Post-Image). Diese werden auch richtig abgefüllt, wenn die Core-Operation abgebrochen wird (bei synchroner Ausführung des Plugins).

### 9.2.5 Statusverwaltung

Das Datenmodell des Dynamics CRM sieht für jede Entität eine zweistufige Statusverwaltung vor. Die beiden verwendeten Felder heissen „State“ und „Status“, oder übersetzt „Status“ und „Statusgrund“.

Beim Status wird in aller Regel nur „Aktiv“ und „Inaktiv“ unterschieden. Dies lässt auch bei komplexeren Statusvergaben zu, grundsätzlich aktive und inaktive Datensätze einfach zu selektieren und anzuzeigen. Eine einfache View im CRM muss also die komplizierteren Statuswechsel nicht unbedingt kennen, sondern kann grundsätzlich alle aktiven Datensätze anzeigen.

Der Statusgrund ist eine feinere Beschreibung des genauen Status. Bei der Entität „Kursdurchführung“ im Beispiel der Kursverwaltung, gibt es neben dem Status „Aktiv“ und „Inaktiv“ noch die Möglichkeiten „Geplant“, „Veröffentlicht“, „Bestätigt“ und „Ausgebucht“ für den aktiven Status sowie „Durchgeführt“ und „Abgebrochen“ für den inaktiven Status.

### 9.2.6 Prozesse und Workflows

Mit Hilfe von Prozessen können asynchrone Automatismen auch nach der Installation vom User manipuliert werden. Einfache Abläufe (inkl. Bedingungen) welche von vordefinierten Events (Erstellen, Ändern, Löschen, Status ändern von allen Entitäten) ausgelöst werden, können so grafisch zusammengestellt werden. Einzelne Schritte dieser Abläufe können entweder eingebaute Schritte sein (neuer Datensatz erstellen, etc.) oder aber eigene, programmierte Aktivitäten. Diese lassen sich, genau wie Plugins, in C# mit Hilfe des CRM SDK entwickeln und ins CRM einbinden.

Da die meisten Abläufe im Anwendungsfall der Kursadministration weitgehend vorgegeben und nicht durch den Benutzer veränderbar sein müssen, sollen nur sehr selten Prozesse zum Einsatz kommen. Ein Fall ist das Absenden der E-Mail Bestätigung nach erfolgter Anmeldung an einen Kurs. Grundsätzlich kann dieser Prozess soweit angepasst werden, dass das zu versendende E-Mail ein anderes Format besitzt oder die Bestätigung mit einem Dialog manuell ausgelöst werden muss.

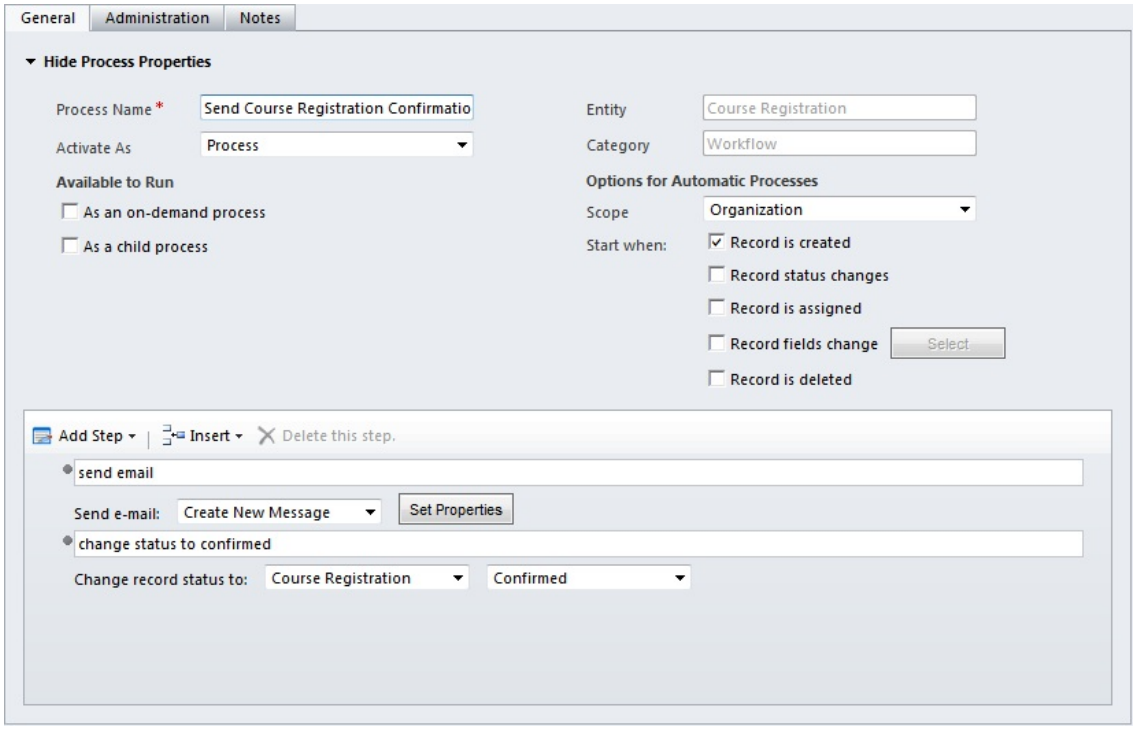



**Abbildung (9.4)** Design-Ansicht des Workflows um die Kursbestätigung zu versenden und im System zu bestätigen.

Neben den automatisch ausgelösten Workflow-Prozessen existiert in Dynamics-CRM 2011 noch die Möglichkeit, Workflows als On-Demand Workflows, also auf Abruf einzurichten.

Ausserdem können Dialogprozesse erstellt werden, welche gleich wie Workflows definiert werden, aber nicht im Hintergrund ablaufen sondern mit dem Benutzer interagieren. Beide dieser Möglichkeiten werden aber im Kursverwaltungssystem selten bis gar nie genutzt, können natürlich aber zur Laufzeit von den CRM Benutzern hinzugefügt oder bearbeitet werden, falls sich die internen Abläufe ändern würden.

### 9.2.7 Frontend Erweiterungen

Für einige eigene Funktionalität in der Kursverwaltung, mussten Frontend Komponenten des CRM angepasst werden. Dazu können sogenannte „Web-Resources“ eingesetzt werden, welche in die jeweiligen Masken und Formulare eingefügt werden können. Web-Ressourcen können clientseitige Web-Komponenten beinhalten, also Bilder, einfache HTML Seiten und JavaScript-Files. Um die CRM Funktionalität aus JavaScript ansprechen zu können, stellt das CRM diverse Funktionen zur Verfügung, die nach Integration des JavaScripts als Web-Ressource automatisch dazu geladen werden. Grundsätzlich hat man über diese Funktionen direkten Zugriff auf die Funktionalität, die das aktuell geöffnete Formular sowieso bietet. Man kann aber, über asynchrone Abfragen auf den Organization-Service (siehe *xRM Web Services* auf Seite 48) Zugriff auf die weiteren Komponenten bekommen.

### 9.2.8 CRM Solutions

Mit Dynamics CRM 2011 wurde unter dem Überbegriff Solution eine Kapselung von Anpassungen („customizations“) im Dynamics CRM ermöglicht. Diese Kapselung erlaubt es, eine Solution, also quasi ein Anpassungsset, als Packet auszuliefern, zu installieren oder wieder zu entfernen. Diese Solution kann neben eigenen Komponenten auch angepasste Standardkomponenten beinhalten.

Während der Entwicklung wird die Solution als „unmanaged“ angesehen, sie kann also noch ändern, kann aber nicht als Ganzes deinstalliert werden. Trotzdem erlaubt sie das Zusammenfassen der Anpassungen sowie gelegentliche Backups derselben.

Eine Solution hat immer einen sogenannten „Publisher“ zugeordnet, also den Herausgeber der Änderungen. Dieser Publisher bestimmt auch einen Prefix, welcher an alle eigenen Entitäten, Beziehungen und Felder angehängt wird (`tadm_`). So ist jederzeit ersichtlich, wer welche Änderungen vorgenommen hat. Kommen mehrere Solutions vom gleichen Publisher, wird davon ausgegangen, dass die Verantwortung beim Publisher selber liegt, dass sich die Anpassungen der beiden Solutions nicht gegenseitig stören. Durch die verschiedenen Prefixes der verschiedenen Publisher sind Konflikte unter deren Anpassungen nicht mehr sehr häufig. Einzig wenn vorgefertigte Entitäten auf verschiedene Weisen angepasst werden, wird anhand der Priorität, mit welcher die Solution installiert ist, eine Änderung der

Anderen vorgezogen.

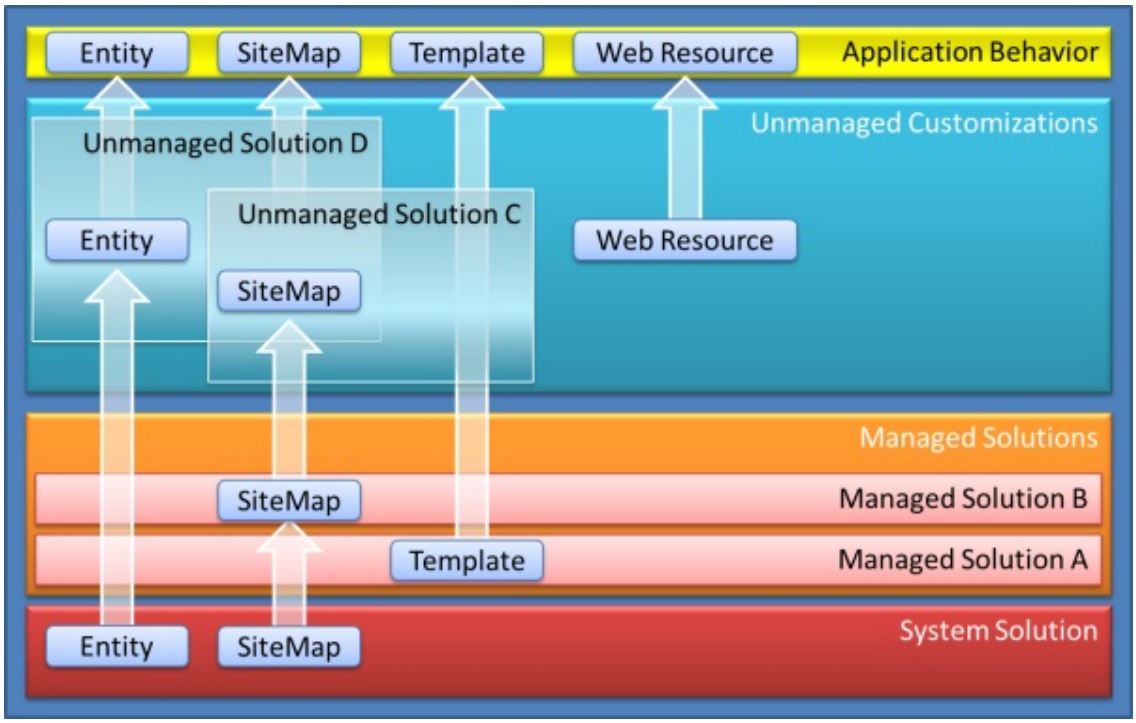



**Abbildung (9.5)** Solution-Vererbungshierarchie [Mic11]

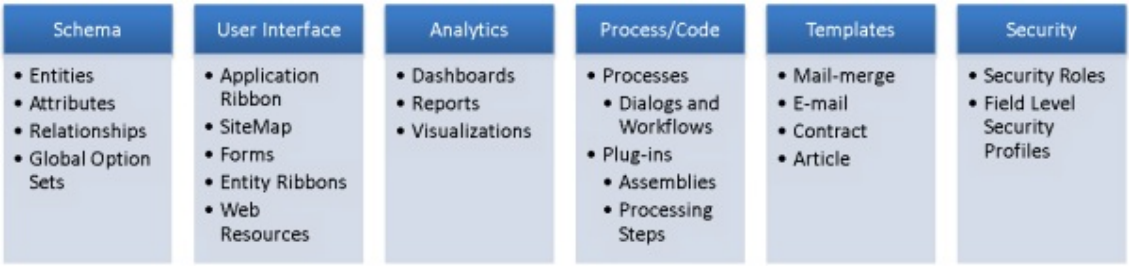



**Abbildung (9.6)** Mögliche Komponenten innerhalb Solutions [Mic11]

Beeinflussen mehrere „managed Solutions“ die selben Entitäten, werden Teile der Anpassungen wie folgt zusammengefügt:

**Anpassungen an Formularen** Formularerweiterungen werden zusammengefügt, sofern die anzupassende Entität bereits in der Organisation besteht. Dies ist vor allem bei den eingebauten Systementitäten der Fall. Wird eine Solution als „managed Solution“ exportiert, werden ausschliesslich die Änderungen zur bestehenden Entität abgespeichert und bei einer Installation der „managed Solution“ auch ausschliesslich

diese installiert. Um solche Zusammenführungen in einer eigenen Solution zu verhindern empfiehlt es sich, eigne Formulare zu erstellen und nicht die Standardformulare des CRM („Information“ und „Information (Mobile)“) anzupassen. Werden z.B. neue Felder einem bestehenden Formular hinzugefügt, erscheinen diese automatisch zuunterst. Dies lässt sich mit einem eigenen Formular verhindern.

**Anpassungen an der Sitemap (Navigation innerhalb CRM)** Um Anpassungen der Navigation vorzunehmen, muss das sogenannte „SiteMap XML“ angepasst werden. Durch den Aufbau als XML kann Dynamics CRM sehr gut die Anpassungen extrahieren und nur diese in die „managed Solution“ integrieren. Wird diese dann Installiert, werden ausschliesslich die neu hinzugekommenen Punkte importiert. Dadurch dass die SiteMap nach wie vor vom Benutzer anpassbar ist, können diese durch ihn natürlich nach wie vor verschoben oder umsortiert werden.

**Anpassungen an bestehenden Option Sets** Option Sets (Enumerationen zur Auswahl eines Wertes innerhalb einer Entität) werden innerhalb Dynamics CRM mit einem Zahlenwert und einer Beschriftung abgespeichert. Der Zahlenwert wird zusammengerechnet aus einem fortlaufenden Wert, welcher die Option selber repräsentiert sowie einem Prefix, der dem „Publisher“ also dem Herausgeber der Solution zugewiesen ist. So soll verhindert werden, dass Kollisionen unter verschiedenen Publishern auftreten. Innerhalb seiner verschiedenen Solutions liegt es aber in der Verantwortung des Publishers, keine Konflikte zu verursachen. Überschreiben also zwei Solutions ein Option Set, können durch diese Prefix-Lösung die möglichen Optionen in der Regel ohne weiteres zusammengeführt werden.

### 9.2.9 Kern-Datenmodell innerhalb des CRM

Innerhalb Dynamics CRM lassen sich zur Umsetzung eines eignen Datenmodells eigene Entitäten erstellen. Ausserdem lassen sich die eingebauten Entitäten soweit anpassen, dass eingebaute Funktionalität auch in eigenen Abläufen oder zusammen mit eigener Geschäftslogik eingesetzt werden kann.

Um den angepassten Kern des CRM auch im Code typensicher verwenden zu können, können aus einer bestehenden CRM Instanz die passenden Businessklassen generiert werden.

Die detaillierte Implementation des Datenmodells ist im Kapitel Umsetzung genau beschrieben (siehe *Kern-Datenmodell innerhalb des CRM* auf Seite 73).

### 9.2.10 Weiterführende Informationen

Weitere Informationen sind auf Microsofts Dynamics CRM SDK Referenz [Mic11] zu finden, welche als Grundlage für dieses Kapitel diente.

## 9.3 Zwischenschicht

### 9.3.1 WCF RIA Services

Da die Benutzeroberfläche beim Nachladen von Daten nicht blockieren soll, erlaubt Silverlight keine synchronen Aufrufe auf WCF Webservices. Durch die erforderlichen asynchronen Aufrufe und das Nachladen von untereinander abhängigen Daten entsteht oft recht komplizierter Code. Ausserdem kann es recht aufwändig sein, Daten zwischen Client und Server zu synchronisieren. Windows Communication Foundation Rich Internet Application Services (WCF RIA Services) vereinfacht die Erstellung von mehrschichtigen, verteilten, datenorientierten Applikationen. RIA Services wurde in erster Linie für Silverlight entwickelt, kann aber auch ohne Silverlight verwendet werden.

WCF RIA Services bietet folgende Hilfsmittel:

**CRUD** Automatische Generierung von Create, Read, Update und Delete Methoden für Entitäten bei der Verwendung von Microsoft Entity Framework. Zugriff auf andere serverseitige Datenquellen kann einigermassen einfach selbst implementiert werden.

**Client Proxies für Servicemethoden** Automatische Generierung von Client Proxy Methoden zur Ausführung von Methoden auf dem Server.

**Clientseitige Verwaltung von geladenen Objekten** Über den Service geladene Objekte werden auf der Clientseite in einem Objektpool verwaltet und nur dann über den Service nachgeladen, falls sie noch nicht vorhanden sind.

**Automatisches Nachladen von abhängigen Daten** Abhängige Daten können automatisch nachgeladen werden. Da die clientseitig verwendbaren Objekte automatisch das Interface `INotifyPropertyChanged` erfüllen, kann die View durch RIA Services automatisch aktualisiert werden, sobald die Daten nachgeladen sind.

**Verteilte Validation und Businesslogik** Validationslogik und frei definierbare Businesslogik kann sowohl auf der Client- als auch auf der Serverseite verwendet werden. Dies ist durch die Angabe von Attributen möglich.

**Direkte Integration in grafische Bedienungselemente** Benutzeroberflächenelemente wie `DataGrid` oder `DataForm` erlauben die direkte Verwendung der CRUD Funktionalität von RIA-Services.

**Integration der ASP.NET Security** Durch die Erstellung eines `AuthenticationService` kann ASP.NET Security verwendet werden.

**Projekt Vorlagen** Es bestehen Projektvorlagen für die Erstellung von Silverlight Applikationen mit Visual Studio 2010 für die Verwendung von RIA-Services.

**Transaktionen** RIA Services unterstützt Transaktionen und Hilfsmittel zur Behandlung von Datenkonsistenzproblemen bei Nebenläufigkeit.

### Domain Service

Um CRUD Funktionalität über WCF RIA Services mit einer beliebigen Datenquelle zur Verfügung zu stellen, muss ein Domain Service erstellt werden, welcher von `System.ServiceModel.DomainServices.Server.DomainService` erbt. Für jede Entität wird ein Set von Funktionen (Query,Insert,Update,Delete) benötigt. Das folgende Beispiel zeigt eine einfache Implementierung eines Domainservices.

```
1  [EnableClientAccess()]
   public class MyDomainService : DomainService
   {
       public IQueryable<Contact> Contacts { get; set; }
5
       [Query()]
       public IQueryable<Contact> GetContacts()
       {
           return Contacts;
10     }

       [Insert()]
       public void InsertContact(Contact contact)
       {
15         insertContact(contact);
       }

       [Update()]
       [RequiresAuthentication]
20     public void UpdateContact(Contact contact)
       {
           updateContact(contact);
       }

25     ....

   }
```

Die Delete Methode wurde absichtlich weggelassen. Damit entfällt die Möglichkeit Daten zu löschen, was bei einem öffentlichen Service durchaus Sinn macht.

Auf dem Client wird der Service durch die Verwendung eines `DomainContext` Objektes benützt, welches die bereits geladenen Daten in einem Objektpool verwaltet. Die CRUD Methoden `Insert` und `Update` werden automatisch verwendet, nachdem auf dem Client `SubmitChanges()` auf den `DomainContext` aufgerufen wurde und Änderungen der Daten im Objektpool bestehen. Sie können nicht explizit verwendet werden.

```
1  domainContext.Contacts.Add(contact);
   domainContext.SubmitChanges(
       sop =>
       {
5          if (sop.HasError)
           {
               doSomethingOnError();
               return;
           }
10
           doSomethingAfterSaving();
       },
       null);
```

Mit dem Attribut `[Invoke]` oder der Anweisung `UsingCustomMethod` auf einer `Update` Methode, können Methoden definiert werden, welche explizit vom Client aus aufgerufen werden können.

Die zur Verfügung gestellten Daten eines WCF RIA Services können durch die Angabe spezieller Endpunkte in der `Web.config` des Service Projektes auch mit anderen Technologien verwendet werden. Unterstützt werden OData, JSON und SOAP. Damit wird es zum Beispiel möglich Daten direkt in Excel (mit OData) oder per JavaScript (mit JSON) zu verwenden.

### 9.3.2 Weiterführende Informationen

Beispiele zur konkreten Verwendung von WCF RIA Services werden im Kapitel Umsetzung (siehe *WCF RIA Services* auf Seite 88) beschrieben.

Weitere Informationen sind auf im MSDN (Microsoft Developer Network) zu finden. Für die Verfassung dieses Kapitels wurde ausserdem das Buch „Silverlight 4 in Action“ verwendet [Bro10].

# 9.4 Web Frontend mit Silverlight

Für die Implementierung des Web-Frontends wurde Silverlight 4 eingesetzt, welches für die Oberflächenprogrammierung eine Untermenge von Microsofts GUI-Framework Windows Presentation Foundation (WPF) 4.0 unterstützt. Die einfache Integration von Webservices erleichtert die Entwicklung von datenlastigen Client-Anwendungen wesentlich. In diesem Kapitel sollen die wichtigsten für das Web-Frontend verwendeten Technologien und Konzepte genauer erklärt werden.

## 9.4.1 Model-View-ViewModel (MVVM)

Model-View-ViewModel (MVVM) ist ein Architekturpattern, welches Microsoft für die Entwicklung von Benutzeroberflächen mit WPF empfiehlt und durch die Hilfsmittel von WPF auch immer mehr fördert. Es ermöglicht eine saubere Trennung von Daten, Logik, Zustand und Darstellung einer Applikation. Dank dieser Trennung ermöglicht man eine flexible Anpassung der Benutzeroberfläche mit Microsofts UI-Designsoftware „Expression Blend“ und eine erhöhte automatische Testbarkeit der Software.

### Andere Pattern für die Erstellung von Benutzeroberflächen

Um MVVM zu erklären und dessen Vorzüge besser aufzeigen zu können, sollen nun zunächst andere Architekturpattern für die Erstellung von Benutzeroberflächen und Trennung von Logik und Darstellung kurz erläutert werden. Die folgenden Beschreibungen sind stark angelehnt, an die beiden Artikel GUI Architectures [Fow06] und Presentation Model [Fow04b] von Martin Fowler.

**Forms and Controls** Der klassische Ansatz Bunutzeroberflächen zu erstellen, welcher Martin Fowler als „Forms and Controls“ bezeichnet, wurde stark durch visuelle UI-Editoren geprägt. Diese Tools erlauben es per Drag&Drop wiederverwendbare Elemente (Controls) zu einem Formular zusammenzustellen. Das Formular ist schliesslich für die Darstellung verschiedener Elemente und die Logik zuständig. Ein Synchronisationsmechanismus zwischen dem sichtbaren Zustand der GUI-Elemente und dem Zustand der (z.B. aus einer Datenbank) geladenen Daten im Speicher stellt sicher, dass der aktuelle Stand der Daten angezeigt wird.

„Forms and Controls“ ist der am weitesten verbreitete Ansatz Rich-Client Applikationen zu erstellen, hat aber leider wesentliche Nachteile. Die fehlende Trennung von Logik und Darstellung macht es schwierig die Benutzeroberfläche anzupassen, ohne die Logik neu implementieren zu müssen. Logik muss oft an mehreren Orten platziert werden und wird dadurch unübersichtlich und fehleranfällig. Ausserdem ist

die automatische Testbarkeit stark eingeschränkt.

Mit WPF kann dieser Ansatz ganz einfach umgesetzt werden, indem man die ganze Logik im Code Behind (siehe auf Seite 63) platziert, was aber aus den oben genannten Gründen nicht zu empfehlen ist.

**Model-View-Controller (MVC)** Das Model-View-Controller Pattern (MVC) teilt die Benutzeroberfläche einer Applikation in drei verschiedene Komponenten. Das Model kann als Facade auf die Geschäftslogik und die Daten gesehen werden und enthält die Grundfunktionalität der Software. Die View zeigt dem Benutzer die Informationen an, während der Controller für die Verarbeitung von Benutzereingaben zuständig ist. Der Controller manipuliert anhand der Benutzereingaben das Model und ist für das Laden der richtigen View zuständig.

Im klassischen MVC kommuniziert der Controller möglichst wenig mit der View. Ein Benachrichtigungsmechanismus (meist nach dem Observerpattern implementiert) stellt die Konsistenz zwischen View und Model sicher. Das Model benachrichtigt den Observer, dass sich Daten geändert haben. Dieser meldet der View, dass die Daten vom Model neu geladen werden müssen. Es können mehrere Views vom gleichen Controller kontrolliert werden.

Bei vielen MVC Implementierungen (z.B. nach POSA 1 [FRH+96]) ist die Synchronistation zwischen Model und View so implementiert, dass die View das Model referenziert. Dies schränkt die Wiederverwendbarkeit der View stark ein. Ausserdem wird es bei komplexeren Anwendungen oft schwierig die Synchronisation zwischen View und Model nachzuvollziehen, da sie nicht über den Controller geht. Variationen von MVC werden heute vor allem bei Web-Applikationen verwendet, wo die Synchronisation zwischen Model und View auf ein Minimum (normalerweise beim erneutem Request) beschränkt ist.

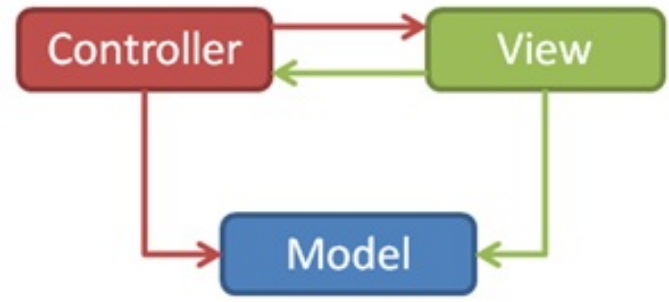



**Abbildung (9.7)** Darstellung der Beziehungen zwischen den drei Komponenten von MVC (Quelle: http://msdn.microsoft.com). Sowohl die Verbindung von der View zum Controller als auch vom Model zur View (hier nicht angezeigt) wird meistens über einen Observer hergestellt, welcher für die Benachrichtigungen bei Datenänderungen zuständig ist.

**Model-View-Presenter (MVP)** Das Model-View-Presenter Pattern (MVP) kappt im Vergleich zu MVC die direkte Kommunikation von der View zum Model. Alle Aufrufe und Daten werden von der View empfangen und an den Presenter weitergeleitet, welcher die Logik enthält. Somit ist hier die View der Einstiegspunkt, während bei MVC Controller den Einstiegspunkt bildet.

Pro View gibt es normalerweise einen Presenter, welcher meist von der View referenziert wird. Der Presenter führt Aktionen auf das Model aus, muss aber auch die View kennen, um im Model geänderte Daten an die View zurückschicken zu können. MVP hält sich wie MVC an die strikte Trennung von Datenlogik (Model) und Darstellung (View und Presenter). Die Abhängigkeit der View vom Model entfällt aber. Um Austauschbarkeit der View (vom Presenter) zu ermöglichen, wird bei MVP ein Interface für die View deklariert. Durch die Deklaration des Interfaces der View kann die View für Tests ausgetauscht werden.

Komplexere Views können mehrere Presenter enthalten. Ein Presenter verwendet aber nur eine View. Diese 1:1 Beziehung macht es schwierig gleiche Daten in mehreren Views zu synchronisieren, was z.B. bei MVC durch die Verwendung mehrerer Views mit einem Controller möglich ist.

Die oben beschriebene Variante von MVP nennt sich „Passive View“. „Supervising Controller“ ist eine weitere relativ oft verwendete Variante, welche eine Kommunikation zwischen Model und View durch Data Binding erlaubt. Damit wird die Synchronisation vereinfacht, aber die Testbarkeit verringert, da nicht mehr die ganze Kommunikation über den Presenter geht.

```
1 public class DomainView: IDomainView
```

```
{
    private IDomainPresenter domainPresenter;

    public DomainView()
    {
        this.domainPresenter = new ConcreteDomainPresenter(this);
    }
}
```

Das obenstehende Codebeispiel zeigt die Instanziierung eines Presenters im Code Behind einer View bei MVP mit WPF. Die 1:1 Beziehung wird hier ersichtlich.

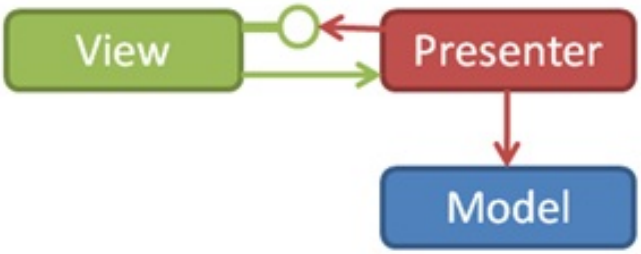



**Abbildung (9.8)** Darstellung der Beziehungen zwischen den drei Komponenten von MVP der Variante Passive-View (Quelle: http://msdn.microsoft.com)

**Presentation-Model** Das von Martin Fowler beschriebene Presentation-Model Pattern führt eine Komponente zwischen View und Model ein, welche den kompletten Zustand der View repräsentiert. Diese Komponente wird Presentation-Model genannt. Im Gegensatz zu MVP wird der ganze Zustand der View und deren gesamtes Verhalten durch das Presentation Model bestimmt. Das Presentation-Model wird ausserdem von der View als Facade für den Zugriff auf das Model benützt.

Es kann entweder die View das Presentation-Model referenzieren oder umgekehrt. Im ersten Fall ist die View ohne eine Änderung am Presentation Model beliebig austauschbar. Ausserdem ist es möglich, mehrere Views mit dem gleichen Presentation Model zu verwenden. Die Applikation kann ausserdem ohne die Verwendung einer View getestet werden.


**Abbildung (9.9)** Darstellung der Beziehungen zwischen den drei Komponenten von Presentation-Model (Quelle: http://msdn.microsoft.com)

### Genauere Betrachtung von MVVM

Das Model-View-ViewModel Pattern (MVVM) ist eine Spezialisierung des von Martin Folwer beschriebenen Presentation-Model Pattern unter Verwendung von Microsofts GUI Framework Windows Presentation Foundation (WPF). MVVM wird so implementiert, dass die View das Presentation-Model referenziert, welches bei MVVM ViewModel genannt wird. Es werden dazu Konzepte verwendet, die in den folgenden Abschnitten erklärt werden.

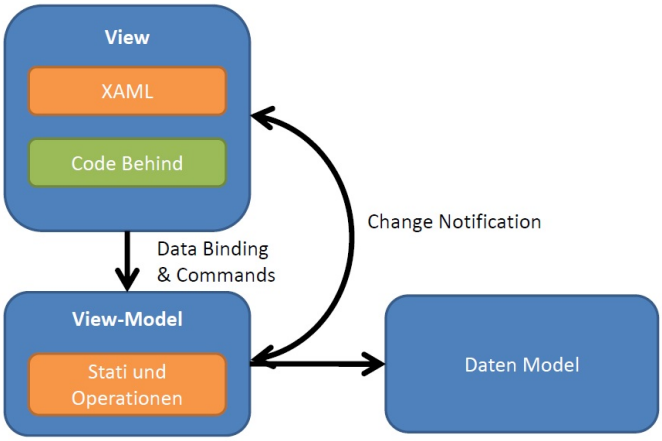



**Abbildung (9.10)** Schematische Darstellung von MVVM (Quelle: Vorlesungsfolien Microsoft Technologien HS2010 an der HSR)

**Extensible Application Markup Language (XAML)** XAML ist eine deklarative XML-Beschreibungssprache für die Oberflächengestaltung mit WPF, welche Microsoft in .NET 3.0 eingeführt hat. Durch die Angabe von XML Namespaces ist XAML beliebig erweiterbar. XAML kann in Visual Studio 2010 direkt als XML bearbeitet werden. Ausserdem bietet Visual Studio ein Tool zur visuellen Bearbeitung der Views, welches aber nicht immer befriedigend funktioniert. Als Alternative bietet Microsoft mit Expression Blend (siehe *Microsoft Expression Blend* auf Seite 103) ein Werkzeug, in welchem Benutzeroberflächen einiges komfortabler visuell bearbeitet werden können.

```
1  <UserControl x:Class="MyNamespace.MyView"
       xmlns=
           "http://schemas.microsoft.com/winfx/2006/xaml/presentation"
       xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
5      xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
       xmlns:mc=
           "http://schemas.openxmlformats.org/markup-compatibility/2006"
       mc:Ignorable="d"
       d:DesignHeight="300" d:DesignWidth="400">
```

```
10
    <Grid x:Name="LayoutRoot" Background="White">

    </Grid>
</UserControl>
```

Dieses Beispiel zeigt eine minimale Definition eines eigenen Silverlight UserControls in XAML.

**Code Behind** Grafische Benutzerelemente werden in WPF in zwei Teilen (Partial Classes) definiert. Der visuelle Teil wird deklarativ durch XAML beschrieben, der logische Teil im Code Behind. Im Code Behind können unter anderem Eventhandler definiert werden, welche bei bestimmten GUI-Ereignissen ausgelöst werden. Bei einer konsequenten Implementierung des MVVM Patterns, besteht der Code Behind jedoch nur aus einem Konstruktor, welcher das Element initialisiert und einer Möglichkeit zum Setzen des ViewModels.

```
1   [Export("MyView")]
    public partial class MyView : UserControl
    {
        public MyView()
5       {
            InitializeComponent();
        }

        [Import]
10      public MyViewModel ViewModel
        {
            set { this.DataContext = value; }
        }
    }
```

Dieses Beispiel zeigt den minimalen Code Behind mit Verwendung von Managed Extensibility Framework (MEF) (siehe *Managed Extensibility Framework (MEF)* auf Seite 66) für das Setzen des ViewModels.

**ViewModel** Das ViewModel enthält alle Zustände und Daten und ausführbare Commands welche von der View verwendet werden können und bildet eine Fassade für die verwendbaren Daten und Operationen des Models, in welchem die eigentliche Businesslogik implementiert ist. Verschiedene Views können das gleiche ViewModel referenzieren. Unittests können direkt auf das ViewModel, ohne die Verwendung einer View ausgeführt werden.

**Model** Das Model kapselt die eigentliche Businesslogik und die Daten und kann zum

Beispiel einen Webservice oder eine Datenbank verwenden. Da das ViewModel komplett von der View entkoppelt und damit auch bequem testbar ist, ist es für eine einfachere Aktualisierung der Daten in der View oft sinnvoll unter Verwendung von Funktionalität des Models einen kleinen Teil der Logik direkt im ViewModel zu implementieren.

**Data Binding & Change Notification** Damit das ViewModel die View nicht referenzieren muss, wird für die Synchronisation der Daten Data Binding verwendet. Es wird dazu für jedes Binding ein Objekt instanziiert, welches Daten in der View und die entsprechenden Daten im ViewModel referenziert und synchronisieren kann. Um Änderungen der Daten im ViewModel an das Data Binding melden zu können, muss das ViewModel das Interface `INotifyPropertyChanged` implementieren. Die Best Practices Sammlung Prism von Microsoft (siehe *Prism* auf Seite 68) enthält die Basisklasse `NotificationObject`, eine fertige Implementation des `INotifyPropertyChanged` Interfaces, welche sich ausgezeichnet für ViewModels eignet. Bei einer Änderung der Daten muss im ViewModel die Methode `RaisePropertyChanged("`PropertyName"')` aufgerufen werden, welche über das Data Binding die View benachrichtigt, dass sich Daten geändert haben. Data Binding kann in den vier verschiedenen Varianten OneWay, TwoWay, OneWayToSource und OneTime angegeben werden.

**Commands** Ohne die Implementation von Commands im ViewModel und die Möglichkeit diese an GUI-Elemente zu binden, wäre MVVM nicht umsetzbar. Commands implementieren das Interface ICommand, welches die Methoden `Execute()` und `CanExecute()` und den Event `CanExecuteChanged` verlangt. Durch die Verwendung einer `DelegateCommand` Implementation, welche es ermöglicht direkt im Konstruktor ein Delegate oder eine Lambdafunktion für die Execute-Methode und die CanExecute-Methode anzugeben, wird die Definition von Commands wesentlich erleichtert. Prism (siehe *Prism* auf Seite 68) bietet bereits eine sehr flexible `DelegateCommand` Implementation.

Konstruktor des DelegateCommands von Prism:

```
1  public DelegateCommand(Action<T> executeMethod,
                          Func<T, bool> canExecuteMethod);
```

Beispiel einer Verwendung des DelegateCommands mit einem String als Parameter und Lambdafunktionen für `Execute()` und `CanExecute()`:

```
1  private DelegateCommand<string> myCommand;
   public DelegateCommand<string> MyCommand
   {
```

```
    get
    {
        if (myCommand == null)
            myCommand = new DelegateCommand<string>(
                                        (string s) =>
                                {
                                    // implement execute here
                                    MyCommand.RaiseCanExecuteChanged();
                                },
                                (string s) =>
                                {
                                    // implement can execute here
                                    return !string.IsNullOrEmpty(s);
                                }));

        return myCommand;
    }

}
```

Binding des Commands in XAML:

```
<Button Content="Button Command" Command="{Binding MyCommand}" />
```


**Abbildung (9.11)** Durch die Auslösung des CanExecuteChanged Events durch RaiseCanExecuteChanged() des gebundenen Commands, wird durch das erneute Aufrufen von CanExecute() eine Schaltfläche automatisch aktiviert (oben) oder deaktiviert.

### 9.4.2 Managed Extensibility Framework (MEF)

Das „Managed Extensibility Framework“ (MEF) ist ein Framework für die Umsetzung leichtgewichtiger, erweiterbarer Applikationen. Es erlaubt einerseits das Auffinden und Verwenden von Erweiterungen ohne Neukompilierung der Applikation und ohne weitere Konfiguration, andererseits Dependency Injection für die Zusammensetzung von Komponenten aus verschiedenen zusammensetzbaren Teilen (sinngemäss übersetzt aus [Mic10]).

### Dependency Injection

Als Dependency Injection wird allgemein eine Architekturart bezeichnet, welche die Zusammensetzung von verschiedenen Bausteinen nicht durch ausgeschriebenen Code, sondern durch ein gegebenes Framework zur Laufzeit ermöglicht.

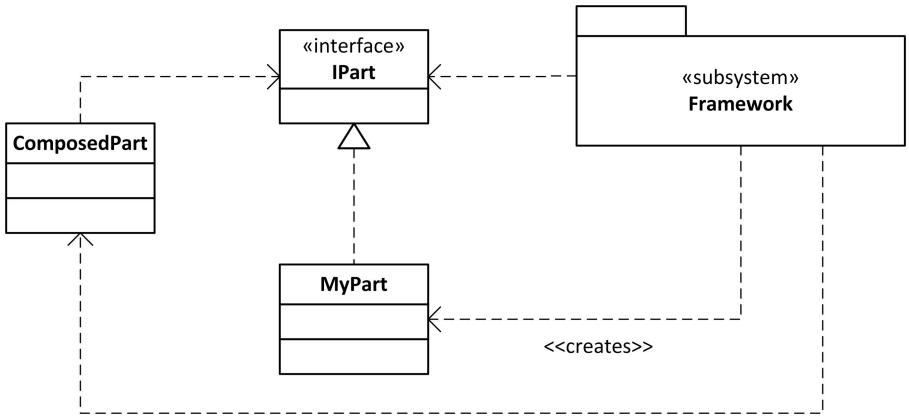



**Abbildung (9.12)** Dependency Injection nach Martin Fowler.

Martin Fowler hat auf seiner Webseite einen ausführlichen Artikel über Dependency Injection veröffentlicht [Fow04a].

### Funktionsweise von MEF

Zusammensetzbare Teile (Composable Parts) werden durch das Export Attribut `[System.ComponentModel.Composition.Export]` deklariert. Abhängigkeiten zwischen verschiedenen Composable Parts können durch das Import Attribut `[System.ComponentModel.Composition.Import]` angegeben werden. Composable Parts sind aber nicht direkt voneinander abhängig, sondern über Verträge (Contracts), welche aus einem einfachen Namen bestehen. Wird kein Name angegeben, so verwendet MEF den Namen des Typen, welcher exportiert wird. Um die Abhängigkeit so klein wie möglich zu halten, werden normalerweise Interfaces oder abstrakte Klassen exportiert. Composable Parts können entweder manuell bei einem Katalog mit einem Namen registriert oder durch verschiedene Arten von automatischen Katalogen gesucht werden. Nach dem Kompilierschritt sind die Export und Importanweisungen als Metadaten der Assemblies verfügbar, welche ohne das Laden der Assembly durchsucht werden können. Durch MEF standardmässig verwendbar sind zum Besipiel `AggregateCatalog` für die Verwendung von mehreren Katalogen gleichzeitig, `AssemblyCatalog` für die Registrierung von Assemblies oder `DirectoryCatalog` für die Registrierung ganzer Verzeichnisse, welche Assemblies beinhalten. Ein Katalog wird schussendlich vom `CompositionContainer` verwendet um

die Composable Parts zusammenzusetzen.

```
1  [Export(typeof(IMyComponent))]
   public class MyComponent : IMyComponent
   {
       [Import]
5      public IMyOtherComponent OtherComponent;

       ....
   }
```

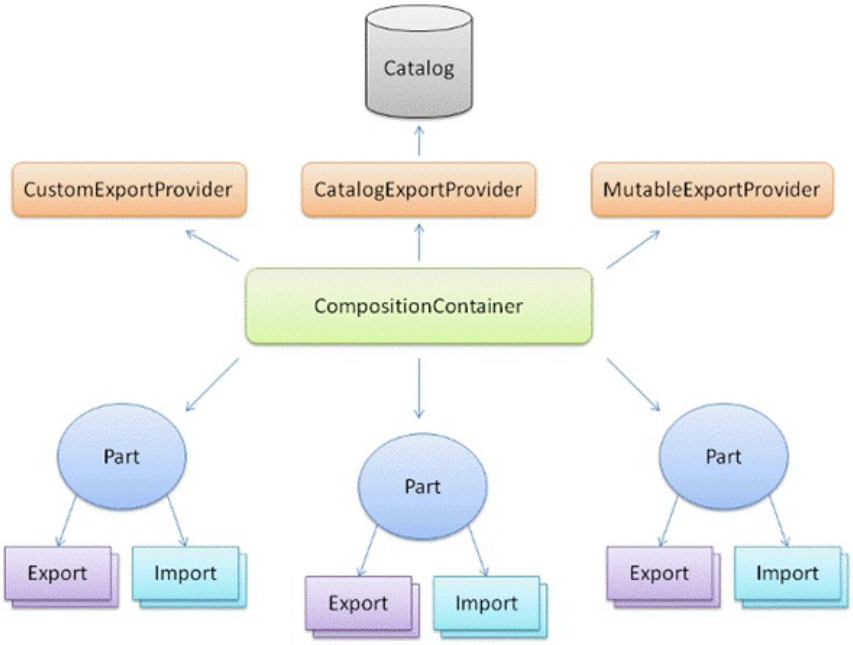



**Abbildung (9.13)** Vereinfachte Darstellung der Architektur von MEF [Mic10]

### 9.4.3 Prism

Microsofts Best Practices Sammlung Prism ist ein Leitfaden, welcher den Entwickler unterstützen soll, grosse, flexibel anpassbare und einfach wartbare Desktop Applikationen mit Windows Presentation Foundation (WPF), Silverlight oder dem Windows Phone 7 zu entwickeln. Unter der Verwendung von Design Pattern welche wichtige Architekturprinzipien wie „Separation of Concern“ und „Loose Coupling“ ermöglichen, können einzelne Komponenten unabhängig voneinander entwickelt und später ohne grosse Aufwände in

die Applikation integriert werden. Die daraus resultierende Architektur wird „Composite Application“ genannt (sinngemäss übersetzt aus [Mic10]).

Prism enthält Referenzimplementationen, Quickstarts, eine wiederverwendbare Codebibliothek und eine ausführliche Dokumentation. Die neuste Version 4.0 von Prism unterstützt das .NET Framework 4.0 und Silverlight 4 und beinhaltet neue Hilfsmittel für die konsequente Umsetzung des MVVM-Pattern (siehe *Model-View-ViewModel (MVVM)* auf Seite 59), für Navigation zwischen dynamisch nachladbaren Views und unterstützt das Managed Extensibility Framework (MEF) (siehe *Managed Extensibility Framework (MEF)* auf Seite 66).

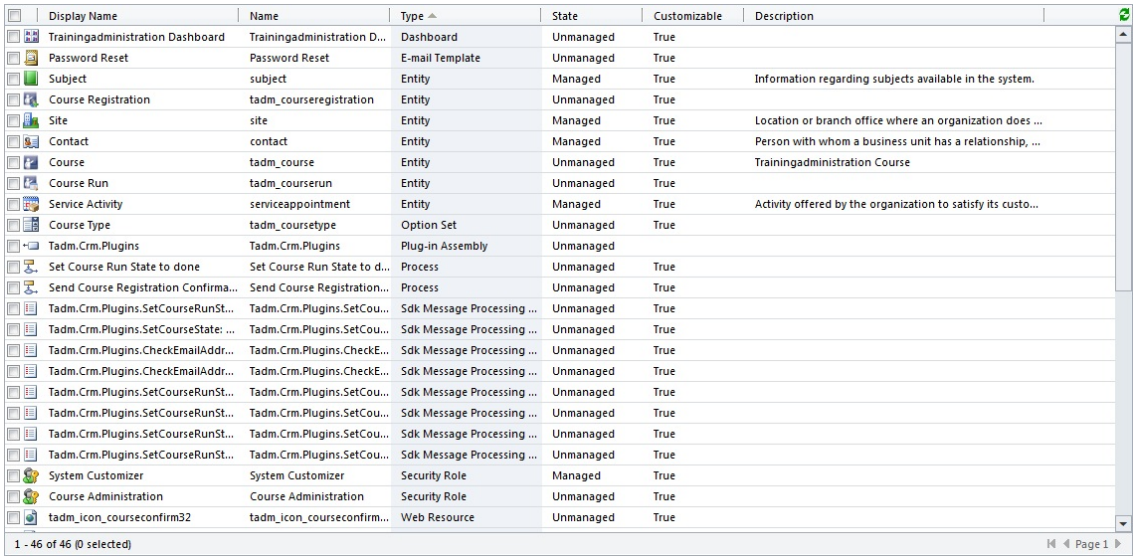
# 10 Umsetzung

## 10.1 Übersicht

Dieses Kapitel beschreibt konkrete Umsetzungen der in Kapitel (siehe *Technologien und Konzepte* auf Seite 47) erklärten Konzepte, wie sie für das Kursanmeldungssystem verwendet wurden.

## 10.2 CRM - Implementation

Die Solution (siehe *CRM Solutions* auf Seite 53) „Trainingadministration“, welche die Anpassungen des CRM beinhaltet, besteht aus verschiedensten Komponenten, welche an verschiedenen Orten in die CRM Funktionalität eingreifen. Folgendes Bild zeigt einen Überblick derjenigen:



| Display Name | Name | Type | State | Customizable | Description |
|---|---|---|---|---|---|
| Trainingadministration Dashboard | Trainingadministration D... | Dashboard | Unmanaged | True | |
| Password Reset | Password Reset | E-mail Template | Unmanaged | True | |
| Subject | subject | Entity | Managed | True | Information regarding subjects available in the system. |
| Course Registration | tadm_courseregistration | Entity | Unmanaged | True | |
| Site | site | Entity | Managed | True | Location or branch office where an organization does ... |
| Contact | contact | Entity | Managed | True | Person with whom a business unit has a relationship, ... |
| Course | tadm_course | Entity | Unmanaged | True | Trainingadministration Course |
| Course Run | tadm_courserun | Entity | Unmanaged | True | |
| Service Activity | serviceappointment | Entity | Managed | True | Activity offered by the organization to satisfy its custo... |
| Course Type | tadm_coursetype | Option Set | Unmanaged | True | |
| Tadm.Crm.Plugins | Tadm.Crm.Plugins | Plug-in Assembly | Unmanaged | | |
| Set Course Run State to done | Set Course Run State to d... | Process | Unmanaged | True | |
| Send Course Registration Confirma... | Send Course Registration... | Process | Unmanaged | True | |
| Tadm.Crm.Plugins.SetCourseRunSt... | Tadm.Crm.Plugins.SetCou... | Sdk Message Processing ... | Unmanaged | True | |
| Tadm.Crm.Plugins.SetCourseState: ... | Tadm.Crm.Plugins.SetCou... | Sdk Message Processing ... | Unmanaged | True | |
| Tadm.Crm.Plugins.CheckEmailAddr... | Tadm.Crm.Plugins.CheckE... | Sdk Message Processing ... | Unmanaged | True | |
| Tadm.Crm.Plugins.CheckEmailAddr... | Tadm.Crm.Plugins.CheckE... | Sdk Message Processing ... | Unmanaged | True | |
| Tadm.Crm.Plugins.SetCourseRunSt... | Tadm.Crm.Plugins.SetCou... | Sdk Message Processing ... | Unmanaged | True | |
| Tadm.Crm.Plugins.SetCourseRunSt... | Tadm.Crm.Plugins.SetCou... | Sdk Message Processing ... | Unmanaged | True | |
| Tadm.Crm.Plugins.SetCourseRunSt... | Tadm.Crm.Plugins.SetCou... | Sdk Message Processing ... | Unmanaged | True | |
| Tadm.Crm.Plugins.SetCourseRunSt... | Tadm.Crm.Plugins.SetCou... | Sdk Message Processing ... | Unmanaged | True | |
| System Customizer | System Customizer | Security Role | Managed | True | |
| Course Administration | Course Administration | Security Role | Unmanaged | True | |
| tadm_icon_courseconfirm32 | tadm_icon_courseconfirm... | Web Resource | Unmanaged | True | |

1 - 46 of 46 (0 selected) Page 1

**Abbildung (10.1)** Screenshot Solution Trainingadministration

Bei den unten nicht komplett dargestellten Komponenten auf dem Bild handelt es sich um weitere Web-Ressourcen die nicht zwingend einzeln aufgelistet werden müssen (Ribbon Icons, etc.). Folgende Auflistung zeigt die angepassten Stellen innerhalb des CRM und beschreibt wo welcher Teil der CRM Logik untergebracht wurde:

**Angepasste Entitäten** Die Systementitäten, welche im Kursadministrationssystem benötigt werden, werden ebenfalls in die Solution eingebettet. Dies um bei einer späteren Installation ohne separate Konfiguration bereits die benötigten Beziehungen mit installieren zu können. Ausserdem werden so die Abhängigkeiten erkannt derentwegen diese Entitäten nicht ohne weiteres gelöscht werden können.

**Eigene Entitäten** Um die eigentlichen Kursinformationen zu speichern, sind drei eigene Entitäten notwendig. Dies sind der eigentliche Kurs (Course), die Kursdurchführung (Course Run) sowie die Anmeldung eines Teilnehmers an einen Kurs (Course Registration). Eingebettet in diese Entitäten sind auch die dazugehörigen Formulare und Abfragen, um die Kurse einfach administrieren zu können.

**Dashboards** Mit Hilfe des hinzugefügten Dashboards lässt sich eine komfortable Übersicht über das Kursverwaltungssystem realisieren. Es kombiniert verschiedene Ansichten der betreffenden Entitäten und ermöglicht so einen schnellen Zugriff auf alle relevanten Einträge.

**E-Mail Vorlagen** Einfache, anpassbare Abläufe bedienen sich E-Mail Vorlagen, welche vor dem abschicken mit den Daten der entsprechenden Entität abgefüllt werden. Beispielsweise wird für die Bestätigung für das Rücksetzen des Passworts eine mit der Solution mitgelieferte Mail-Vorlage eingesetzt.

**Plugin Assemblies & SDK Message Processing Steps** Plugins innerhalb CRM müssen über die bereitgestellten Web-Services registriert werden. Sind sie einmal registriert, können sie in eine bestehende Solution integriert werden. „SDK Message Processing Steps“ sind sozusagen die Trigger, die das Plugin bei der entsprechenden Interaktion auslösen. Auch diese werden über den Web-Service (resp. mit Hilfe des Plugin Registration Tools) konfiguriert.

**Prozesse** Eigene Workflows oder Dialoge werden in Dynamics CRM 2011 unter dem Überbegriff „Prozess“ abgelegt. Konkret sind hier einige einfache Workflows abgelegt, welche asynchrone Aufgaben übernehmen (z.B.: Marketingliste bei Anmeldeschluss erstellen).

**Web Ressourcen** Zur Anpassung des Web-Frontends sind diverse Web Ressourcen in der Solution integriert: Alle Komponenten des Rich-Text Editors, welcher im Kursformular eingebaut ist, Formularweiterungen um die Funktionalität der Ribbons auszulösen, sowie alle benötigten Bilder und Icons, welche neu hinzugekommen sind.

**Sicherheitsrollen** Damit die Berechtigungen isoliert gesetzt werden können, wird mit der Solution eine neue Sicherheitsrolle „Course Administration“ mitgeliefert. Die Rolle „System Customizer“ wird zudem angepasst, damit diese ebenfalls Zugriff auf die entsprechenden Entitäten besitzt.

### 10.2.1 Kern-Datenmodell innerhalb des CRM

Das Kursverwaltungssystem kommt komplett ohne externe Datenbank aus. Sämtliche Daten werden innerhalb des CRM abgelegt. Teilweise mittels eigener Entitäten, teilweise mittels Gebrauch von eingebauten, angepassten Systementitäten.

Auf dem nachfolgenden Diagramm sind die erstellten, wie auch die verwendeten Systementitäten ersichtlich. Im Sinne der Übersichtlichkeit wurden die automatisch generierten CRM Attribute zum Change- und Ownershiptracking weg gelassen. Bei den angepassten Systementitäten sind ebenfalls nur die neu erstellten resp. geänderten Attribute ersichtlich:

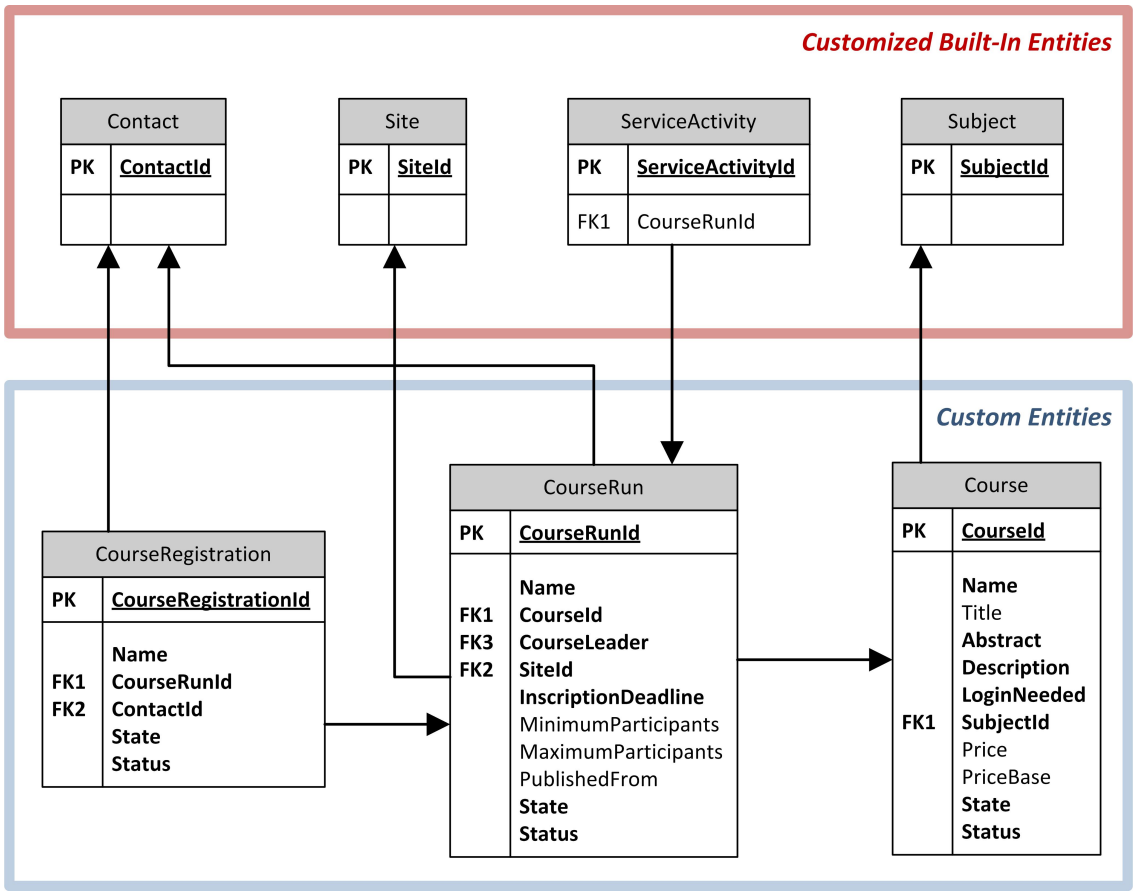



**Abbildung (10.2)** Kern-Datenmodell innerhalb Dynamics CRM (Notation gemäss Microsoft Visio 2010)

### 10.2.2 Statusverwaltung

Die Statusverwaltung innerhalb der Kursverwaltung wird weitgehend durch Plugins und CRM Web-Frontend Erweiterungen sichergestellt. Die untenstehende Grafik zeigt die vorhandenen Status und deren mögliche Wechsel innerhalb des Systems:

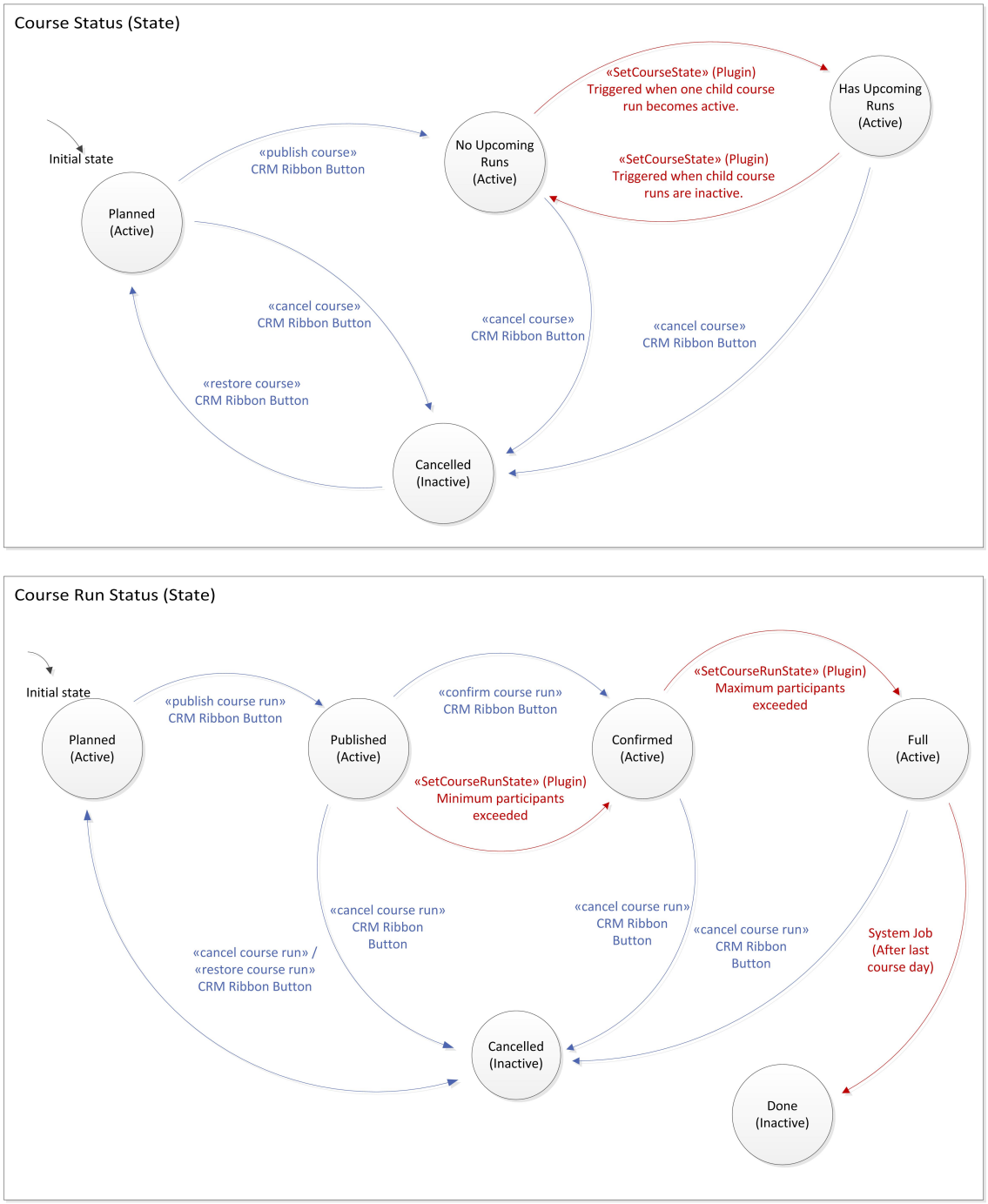



**Abbildung (10.3)** Statusdiagramm für die Entitäten „Kurs“ und „Kursdurchführung“

### 10.2.3 Abläufe innerhalb des CRM

Die administrativen Vorgänge laufen allesamt innerhalb des CRM ab. Der Zugriff über die Web-Oberfläche erfolgt ausschliesslich durch die Teilnehmer oder Interessenten, welche sich für die Kurse interessieren und / oder sich für diese anmelden möchten.

#### Erfassen und Publizieren

Um einen neuen Kurs zu erfassen, wird ein entsprechender Datensatz im CRM angelegt. Das entsprechende Formular bietet die Möglichkeit über den eingebauten Rich-Text Editor (Silverlight Komponente) eine formatierte Beschreibung des Kurses zu erfassen. Ein neu erfasster Kurs hat nach der Erfassung den Status „geplant“ (planned) und wird noch nicht auf der Web-Oberfläche angezeigt. Sind alle Informationen erfasst, kann der Status manuell geschoben werden. Dies unabhängig davon, ob bereits Durchführungen geplant sind oder nicht.

#### Automatische Statuswechsel

Einige Statuswechsel innerhalb der Applikation müssen automatisch ausgelöst werden. Diese werden durch bestimmte Ereignisse ausgelöst, welche bestimmte Vorbedingungen erfüllen. Folgende Tabellen zeigen die Übergänge, die zugehörigen Events und deren Bedingungen:

| Course Run Status | | |
|---|---|---|
| **Event** | **Condition** | **To Status** |
| CourseRegistration: Create, Delete, SetState | Participants >= Max.Participants, Status = Confirmed | full |
| CourseRegistration: Create, Delete, SetState | Participants >= Min.Participants, Status = Published | Confirmed |
| CourseRegistration: Create, Delete, SetState | Participants < Max.Participants, Status = Full | Confirmed |
| CourseRegistration: Create, Delete, SetState | Participants < Min.Participants, Status = Full | Confirmed |

**Tabelle (10.1)** Statuswechsel der Entität „Kursdurchführung“ (Course Run)

| Course Status | | |
|---|---|---|
| **Event** | **Condition** | **To Status** |
| CourseRun: Create, Delete, SetState | >= 1 Related Course Run with Status Published, Confirmed, Full | Has Upcoming Runs |
| CourseRun: Create, Delete, SetState | < 1 Related Course Run with Status Published, Confirmed, Full | No Upcoming Runs |

**Tabelle (10.2)** Statuswechsel der Entität „Kurs“ (Course)

Diese automatischen Statuswechsel werden durch Plugins eingeleitet. Diese Plugins sind synchron in die Event-Pipeline des CRM eingebaut und benutzen den OrganizationService um die entsprechenden Daten abzufragen. Näheres zu den eingesetzten Plugins im nächsten Kapitel.

### 10.2.4 CRM Erweiterungen zur Realisierung der Abläufe

Um die beschriebenen Abläufe zu realisieren, muss Client- und Serverseitiger Code im CRM Verfügbar gemacht werden:

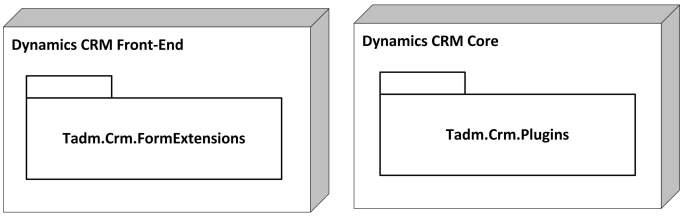



**Abbildung (10.4)** Deployment der Namespaces innerhalb des CRM

Grundsätzlich wird hier zwischen Server- und Clientcode unterschieden. Der Clientcode beinhaltet die Erweiterungen für das CRM Frontend, welche in die entsprechenden Formulare geladen werden können. Sie können von Formularevents (OnLoad, OnSave, ...) oder Ribbon Buttons ausgelöst werden. Diese Erweiterungen bestehen hauptsächlich aus JavaScript Libraries, zusätzlich kommt eine Silverlight-Komponente zum Einsatz, welche den Rich-Text Editor des Kursfomulares bereitstellt.

#### Ribbon Buttons für die Statuswechsel

Die manuellen Statuswechsel werden vom CRM-Benutzer von Hand ausgelöst und werden über neu hinzugefügte Ribbon-Buttons realisiert, welche in das jeweilige Formular

eingebunden wurden.

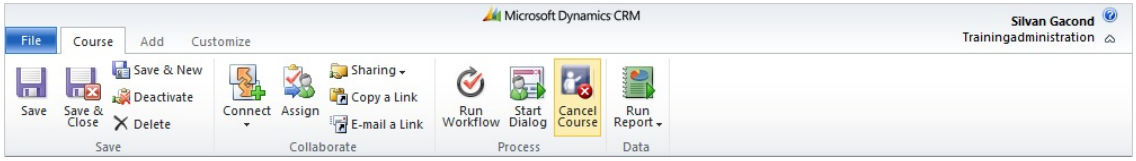



**Abbildung (10.5)** Beispielscreenshot eines Ribbon-Button zum manuellen Statuswechsel

Der Klick auf den Ribbon Button löst eine Javascript Funktion aus, die per SOAP asynchron mit dem OrganizationService kommuniziert und so den Statuswechsel auslöst.

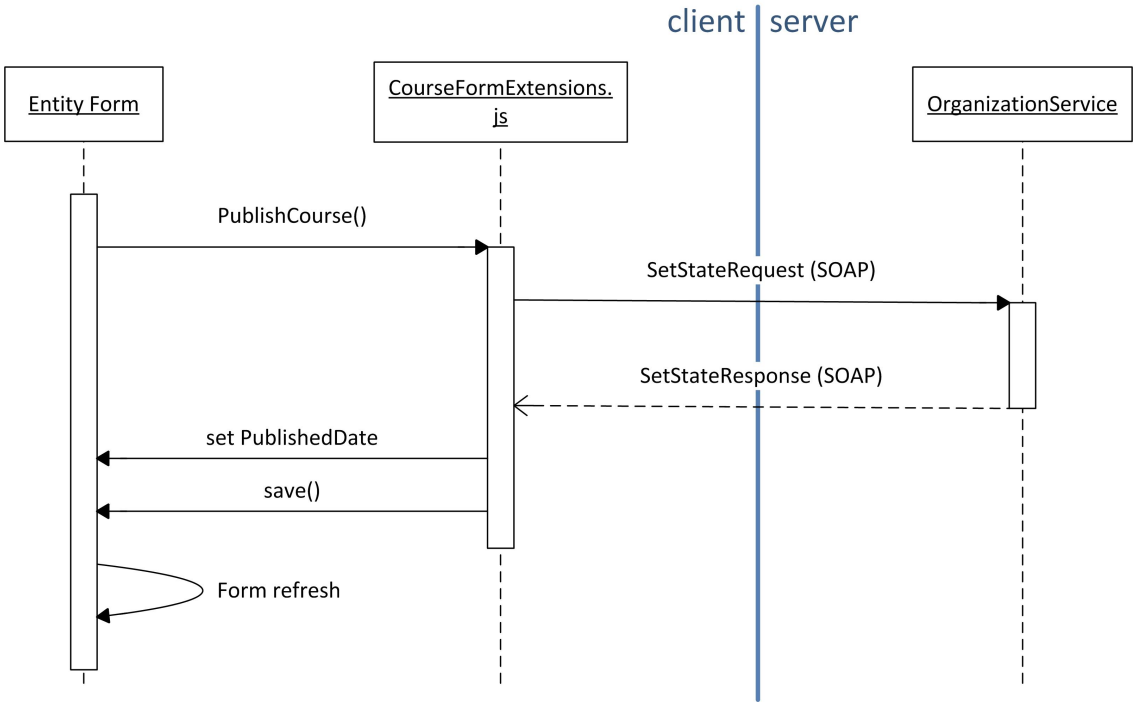



**Abbildung (10.6)** Ablauf eines manuellen Statuswechels via CRM Form

Die JavaScript Funktion muss also in der Lage sein, einen SOAP-Request abzusetzen. Dieser muss ziemlich mühsam als Zeichenkette zusammengesetzt werden. Um den Request fehlerfrei formatieren zu können, wird ein Online-Tool angeboten, um die verschiedenen „Organization Requests“ einfach zusammenstellen zu können und so im Code zu verwenden. Folgendes Codebeispiel zeigt eine solche Abfrage:

```
1  var soapXml = "<s:Envelope xmlns:s=\".../\">" +
               "    <s:Body>" +
```

```
                "      <Execute xmlns=\"...\" xmlns:i=\"...\">" +
                "        <request i:type=\"b:SetStateRequest\"" +
                "       xmlns:a=\"...\" xmlns:b=\"...\">" +
                "          <a:Parameters xmlns:c=\"...\">" +
                "            <a:KeyValuePairOfstringanyType>" +
                "              <c:key>EntityMoniker</c:key>" +
                "              <c:value i:type=\"a:EntityReference\">" +
                "                <a:Id>" + entityId + "</a:Id>" +
                "                <a:LogicalName>tadm_course</a:LogicalName>" +
                "                <a:Name i:nil=\"true\" />" +
                "              </c:value>" +
                "            </a:KeyValuePairOfstringanyType>" +
                "            <a:KeyValuePairOfstringanyType>" +
                "              <c:key>State</c:key>" +
                "              <c:value i:type=\"a:OptionSetValue\">" +
                "                <a:Value>" + statecode + "</a:Value>" +
                "              </c:value>" +
                "            </a:KeyValuePairOfstringanyType>" +
                "            <a:KeyValuePairOfstringanyType>" +
                "              <c:key>Status</c:key>" +
                "              <c:value i:type=\"a:OptionSetValue\">" +
                "                <a:Value>" + statuscode + "</a:Value>" +
                "              </c:value>" +
                "            </a:KeyValuePairOfstringanyType>" +
                "          </a:Parameters>" +
                "          <a:RequestId i:nil=\"true\" />" +
                "          <a:RequestName>SetState</a:RequestName>" +
                "        </request>" +
                "      </Execute>" +
                "    </s:Body>" +
                "  </s:Envelope>";

var req = new XMLHttpRequest();
req.open("POST", orgServiceUrl, false);
req.setRequestHeader("Accept", "application/xml, text/xml, */*");
req.setRequestHeader("Content-Type", "text/xml; charset=utf-8");
req.setRequestHeader("SOAPAction", ".../IOrganizationService/Execute");
req.send(soapXml);
```

Bemerkung: Um dieses Codebeispiel übersichtlich zu halten, wurden die Schema-URLs gekürzt oder ganz weggelassen.

Für reine Abfragen könnte unter CRM 2011 der neue OData (siehe http://www.odata.org) Endpoint eingesetzt werden, was den Javascript Code sehr vereinfachen würde. Leider ist der OData Endpoint nur in der Lage, Abfragen auszulösen. Schreibende Requests sind nicht möglich.

### Rich Text Editor

Um Kursbeschreibungen zur Anzeige im Web-Frontend formatieren zu können, wurde ins Kursformular im CRM ein Rich-Text Editor eingebaut. Dieser Editor ist grundsätzlich eine Silverlight Komponente, die via JavaScript ein verstecktes Feld im CRM Formular abfüllt. Zusätzlich wird dieses Feld mit dem Attribut versehen, dass es beim Speichern zwingend persistiert wird.

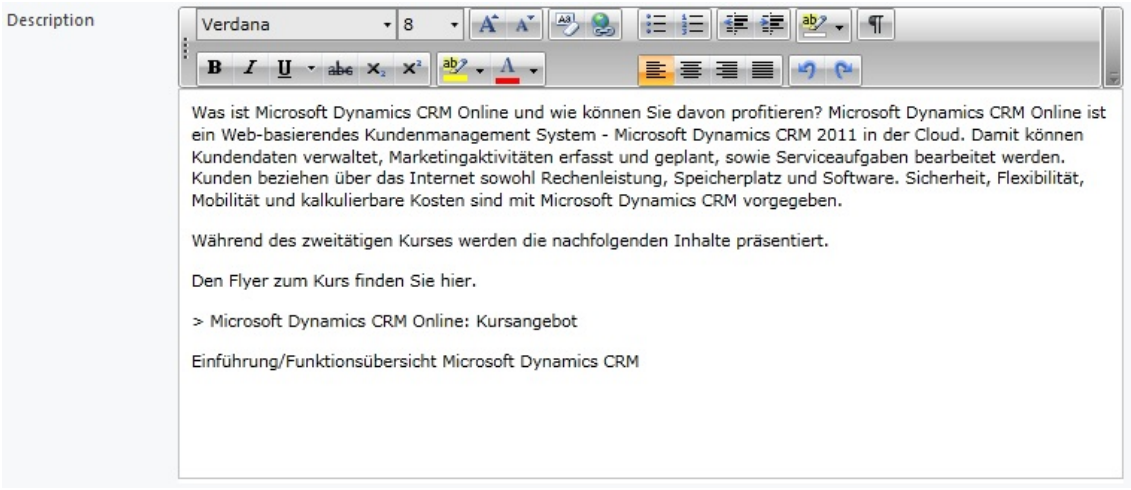



**Abbildung (10.7)** Screenshot des eingebetteten Rich-Text Editor

Innerhalb der Silverlight Komponente wurde ein Telerik Steuerelement eingesetzt, welches einen bedeutend grösseren Funktionsumfang aufweist als eigentlich benötigt wird. Um diese Funktionalität ein wenig einzugrenzen, wurden die Toolbars eingeschränkt. Dies damit nur einfache Formatierungen möglich sind, um das Web-Frontend nicht zu beinträchtigen.

### Plugins zur Validierung und zur automatischen Statusverwaltung

Um die automatischen Statuswechsel zu ermöglichen wurden Plugin Assemblies eingebunden. Diese werden bei den entsprechenden Events von der CRM Event-Pipeline selber ausgelöst und nehmen innerhalb des Plugin-Codes die entsprechende Validierung der Vorbedingung vor.

Wird ein Plugin zur Validierung eingesetzt, muss es im Falle einer negativen Validierung eine bestimmte Exception werfen (`InvalidPluginExecutionException`), welche vom CRM bis ans Frontend weitergeleitet wird.

Das Plugin zur Validierung wie auch die Plugins zur automatischen Statusverwaltung werden synchron ausgeführt. Beim Validierungsplugin ist dies ein Muss, da andernfalls

im Falle einer negativen Validierung die Transaktion nicht mehr unterbrochen werden könnte. Bei den Plugins zur automatischen Statusverwaltung soll mit der synchronen Ausführung die Konsistenz gewährleistet werden. Die Performanceeinbusse, welche durch den synchronen Aufruf auftritt, ist vernachlässigbar.

Das folgende Sequenzdiagramm zeigt den Ablauf des Validierungsplugins zur Vermeidung von doppelten E-Mail Adressen. Auf detaillierte Abbildung der anderen Plugins wird verzichtet, da die Abläufe sehr ähnlich sind.

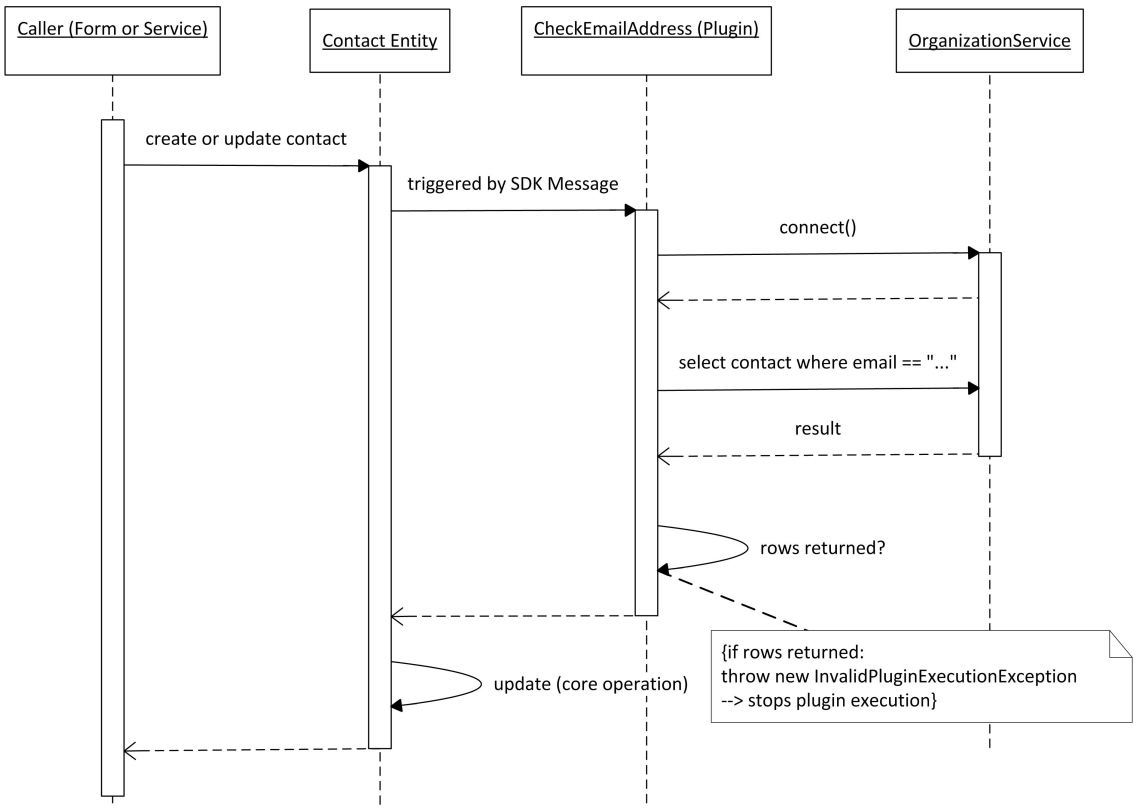



**Abbildung (10.8)** Sequenzdiagramm des Plugins zur Validierung der E-Mail Adresse eines Kontaktes

### Workflow zur Erstellung der Marketingliste bei Registrierungs-Deadline

Um eine Aktion auf einen bestimmten Zeitpunkt zu planen, gibt es in Dynamics CRM die Möglichkeit einen laufenden Workflow auf einen bestimmten Zeitpunkt warten zu lassen.

Am Anmeldeschluss einer Kursdurchführung wird nun eine eigene Workflow-Activity gestartet, welche die Marketingliste erstellt und abfüllt. Durch die Implementation als Workflow kann nun dieser im laufenden Betrieb angepasst werden (Auslösezeitpunkt etc).


**Abbildung (10.9)** Workflow mit Wait-Condition

## 10.3 Zwischenschicht

Wie im Kapitel *Architekturziel* (auf Seite 46) beschrieben, verfolgt die Zwischenschicht mehrere Ziele. Grundsätzlich soll sie aber die CRM-Ebene möglichst gut vom Web-Frontend abtrennen. So kann die Gegenseitige Abhängigkeit möglichst klein gehalten werden und auch die Abtrennung bezüglich Sicherheit ist gewährleistet.

### 10.3.1 Architekturübersicht Zwischenschicht

Da die Zwischenschicht das Backend System abtrennt, geschieht darin eine Konvertierung der ursprünglichen CRM Objekte in die neuen, in der Frontend Schicht verwendeten, Domain-Objekte. Diese Konvertierung trennt die CRM-Funktionalität gänzlich vom Rest der Applikation ab und würde so das vollständige Ersetzen des CRM durch ein anderes Back-End System ermöglichen.

Folgende Grafik zeigt die Abhängigkeiten unter den verschiedenen Komponenten der Zwischenschicht. Um die Übersicht zu wahren, ist die Grafik auf das Minimum reduziert:

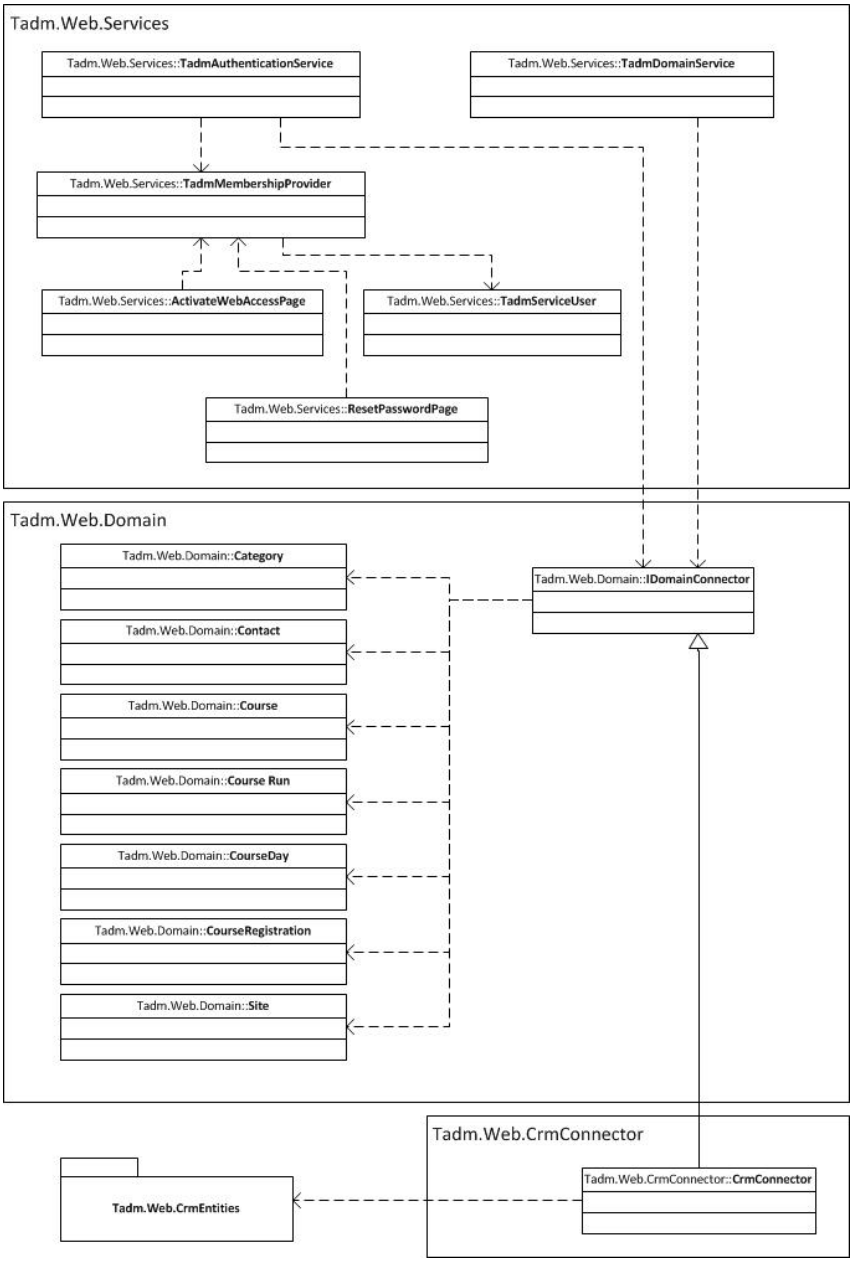



**Abbildung (10.10)** Abhängigkeiten innerhalb der Zwischenschicht (UML gemäss Visio)

### 10.3.2 Austauschbarer Domain Connector

Jegliche Operationen von Seiten des Web Frontends werden auf ein Interface (IDomainConnector) ausgeführt. Die konkrete Implementation des Interfaces führt dann die Anfragen aus. Sollte jemals das CRM durch ein anderes Backend System ausgetauscht werden, kann ein anderer Connector implementiert werden. So ist es also durchaus denkbar mit verhältnismässig wenig Aufwand zum Beispiel einen DatabaseConnector, einen SAPConnector oder einen WebServiceConnector zu schreiben.

### 10.3.3 CRM Connector

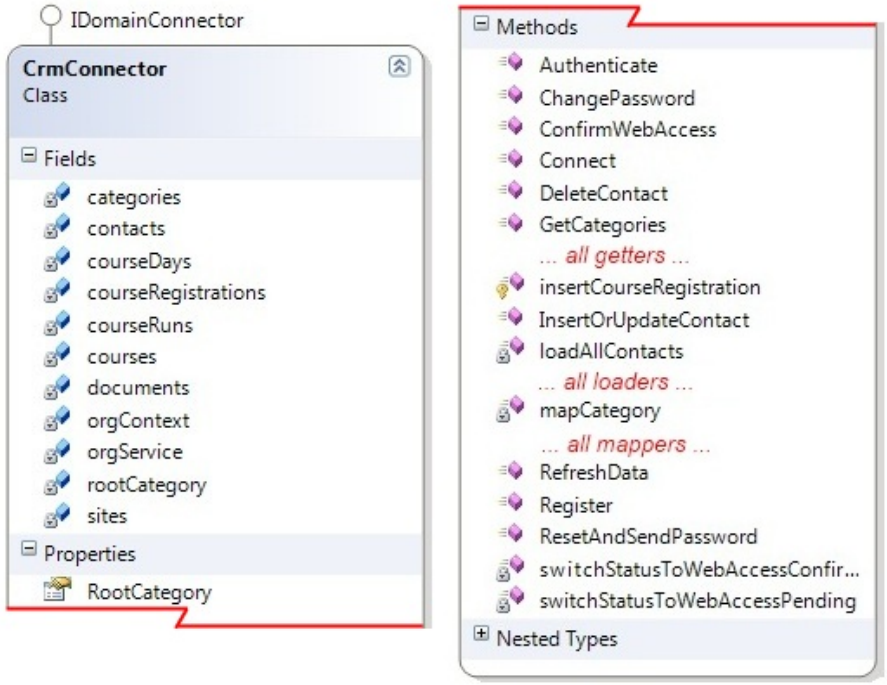



**Abbildung (10.11)** CRM Connector, Übersicht Methoden und Attribute (gekürzt)

Innerhalb des CRM Connector, einer konkreten Implementierung des IDomainConnectors, werden die geladenen Objekte aus dem CRM in die normalen Domain Objekte konvertiert und zwischengespeichert. Der CRM Connector ist auch die einzige Klasse, welche CRM Funktionalität ausführt, resp. mit den CRM Web-Services kommuniziert. Dies aus oben erwähnten Gründen. Der CRM Connector besteht hauptsächlich aus loader, mapper und getter Methoden. Die Ladeoperation wird mit `RefreshData()` ausgelöst, diese ruft die jeweiligen `load...()` Methoden in der richtigen Reihenfolge auf. Diese holen

die entsprechenden Entitäten vom Organization Service und rufen für alle Entitäten die `map...()` Methoden auf, welche das Objekt in das entsprechende Domänen-Objekt konvertieren. Diese werden jeweils in die entsprechenden Collections abgefüllt.

### 10.3.4 Caching

Da die Verbindung zum CRM unter Umständen einige Sekunden in Anspruch nehmen kann und um nicht bei jedem Web-Requenst mehrere CRM Zugriffe auszulösen, werden die Entitäten innerhalb des CRM Connectors zwischengespeichert. Lesende Zugriffe, also praktisch alle Zugriffe beim Anwendungsfall der Kursverwaltung, werden so schlussendlich nur auf dem Cache ausgeführt. Schreibende Zugriffe werden direkt ins CRM geleitet, um allfällige Validierungen direkt durchführen zu können.

Der Cache soll global für die ganze Applikation bestehen, also nicht bei jedem Request oder jeder Session neu geladen werden. Dazu wird die „.NET Global Application Class“ (`global.asax`) und der dazugehörige „Application“ Container eingesetzt. In regelmässigen Abständen wird durch die Global Application Class eine Aktualisierung der Daten ausgelöst.

Folgendes Diagramm zeigt den Ablauf des Verbindungsaufbaus und zwei mögliche Beispiele von Anfragen auf den CRM Connector:

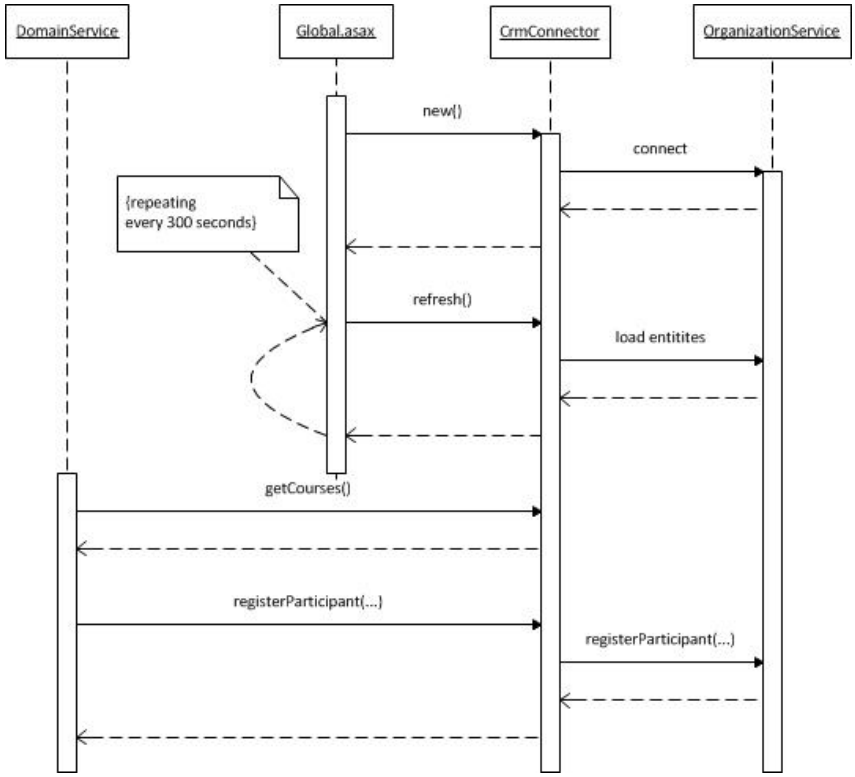



**Abbildung (10.12)** Ablauf Verbindungsaufbau und mögliche Beispielanfragen (State Diagram gemäss Visio)

Eine mögliche Auswirkung dieses Cachings ist, dass eventuell die Zahlen der freien Kursplätze nicht ganz aktuell sind, resp. eine Überbuchung möglich wird. Dies ist aber im Anwendungsfall des Microsoft Innovation Centers Rapperswil kein Problem, da in diesem Fall meist eine Individuelle Lösung gesucht wird, eine Überbuchung vom System her also nicht um jeden Preis verhindert werden muss. Falls dies in einem anderen Anwendungsfall zu einem Problem führen sollte, müsste die Registrierung im CRM validiert und abgebrochen werden.

### 10.3.5 Daten Transfer Objekte

Um kleinstmögliche Abhängigkeit vom Backendsystem zu gewährleisten wurde das gesamte auf dem Client verfügbare Datebmodell mit POCOs („Plain Old CLR Object“) implementiert. Diese Objekte sind reine Datenobjekte und werden im `DomainConnector` aus den Informationen des Backendsystemes erstellt und über WCF RIA Services an den Client übermittelt. Es wurde das in der Analyse ermittelte Domain Model (siehe *Beschreibung der Domain Objekte* auf Seite 43) umgesetzt.

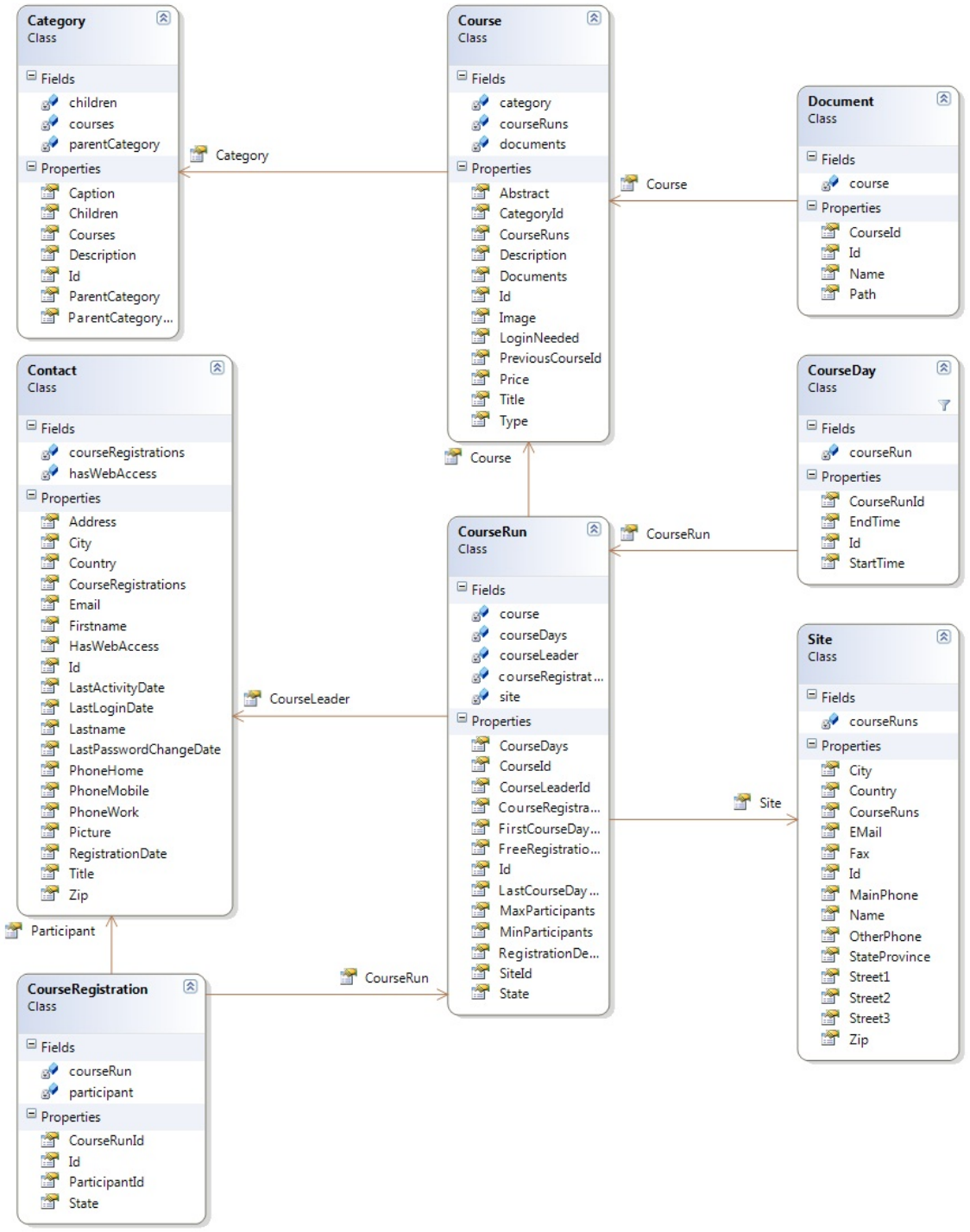



**Abbildung (10.13)** Darstellung der Daten Transfer Objekte.

Wie auf dem Diagramm ersichtlich, müssen diese „POCOs“ im Falle einer Beziehung zusätzlich zu den Objektreferenzen die ID des Gegenübers speichern (dessen Attribut, welches als `[Key]` annotiert ist). Zusätzlich stellt das referenzierte Objekt eine Collection bereit, in der alle referenzierenden Objekte abgespeichert werden. So sind die modellierten Beziehungen immer in beide Richtungen navigierbar.

### 10.3.6 WCF RIA Services

#### Domain Service

Die Klasse `TadmDomainService` ist von `System.ServiceModel.DomainServices.Server.DomainService` abgeleitet und definiert die RIA Serviceschnittstelle. Sie verwendet das Interface `IDomainConnector`, enthält Query Methoden für alle auf dem Client verwendbaren Entitäten und stellt Insert und Update Methoden für die Domain Klasse Contact zur Verfügung.

#### Verwendung eines DomainServices

Um den DomainService in einem Silverlight Projekt zu verwenden, muss in den Projekteinstellungen das Projekt des RIA Services angegeben werden. Dies kann durch ein einfaches Auswählen des Projektes mit dem Service per Mausklick gemacht werden.


**Abbildung (10.14)** Auswahl des Service Projektes.

Die Entwicklungstools, welche das RIA Services Framework zur Verfügung stellt, generieren automatisch Code für die Verwendung des Services im Client Projekt.

#### Authentication Service

Um die vollen Funktionalitäten von WCF RIA Services zur Authentifizierung auszuschöpfen, muss ein `AuthenticationService` implementiert werden. Verwendet man Microsoft Entity Framework ist dies sehr einfach. Da wir aber aufgrund der Architektur innerhalb des Services nur das Interface `IDomainConnector` für Datenoperationen zur Verfügung haben, gestaltete sich die Implementierung eines eigenen `AuthenticationService` etwas schwieriger. Die Basisklasse `AuthenticationService` verwendet einen `MembershipProvider` für die Benutzerverwaltung. Dieser musste für die Verwendung mit `IDomain-`

Connector fast komplett überschrieben werden. Der Typ des Membership Providers (TadmMembershipProvider) wird im Web.config des Service Projektes angegeben und dadurch automatisch durch TadmAuthenticationService verwendet.

```
<configuration>
  ...
  <system.web>
      ...
      <authentication mode="Forms" />
      <membership defaultProvider="TadmMembershipProvider">
        <providers>
          <add name="TadmMembershipProvider"
          type="Tadm.Web.Services.Security.TadmMembershipProvider,
                Tadm.Web.Services"/>
        </providers>
      </membership>
      ...
  </system.web>
  ...
</configuration>
```

## 10.4 Web Frontend mit Silverlight

In diesem Kapitel sollen die Architektur und konkrete Umsetzungen der Silverlightapplikation für die Kursanmeldung beschrieben werden. Die Silverlightapplikation benützt einen RIA-Service welcher bereits im vorherigen Kapitel beschrieben wurde.

### 10.4.1 Projektstruktur

Das Web-Frontend wurde als modularisierte Composite Application mit der Verwendung von Prism (siehe *Prism* auf Seite 68) und MEF (siehe *Managed Extensibility Framework (MEF)* auf Seite 66) umgesetzt. Schlussendlich haben wir nur ein einziges Modul implementiert, das Kursanmeldungsmodul (CoursesModule). Es sind aber weitere Module in Zukunft denkbar. Zum Beispiel ein Modul welches den Kursteilnehmern eine Möglichkeit bieten würde, die Dokumente der Kurse anzuschauen.

Die für das Frontend relevanten Projekte sind Tadm.Web.Client, Tadm.Web.Client.Infrastructure und Tadm.Web.Client.CoursesModule. Tadm.Web.Services und Tadm.Web.Domain werden zur Kompilierzeit benötigt, da der in Tadm.Web.Client.CoursesModule verwendete RIA-Service durch Codegenerierung hinzugefügt wird.

Die kompilierte Silverlight Applikation besteht schlussendlich aus den XAP-Files `Tadm.Web.Client.xap`, `Tadm.Web.Client.Infrastructure.xap` und pro Module je einem weiteren XAP. In unserem Fall ist dies nur das Modul für die Kursanmeldung `Tadm.Web.Client.CoursesModule.xap`.

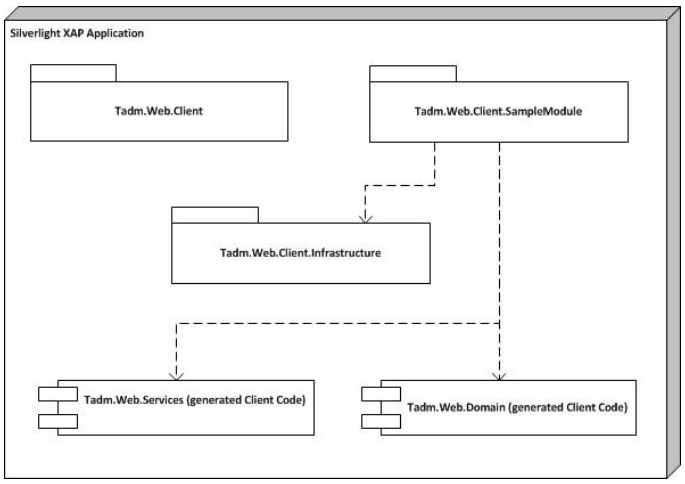



**Abbildung (10.15)** Abhängigkeiten zwischen den Komponenten.

### Tadm.Web.Client

Tadm.Web.Client stellt das Gerüst der Silverlight-Applikation dar. Die Komponente kümmert sich um das Laden der Module und die Anordnung der Views.

**Bootsrapper** Prism bietet mit `Microsoft.Practices.Prism.MefExtensions` eine abstrakte Basisklasse für einen Bootstrapper an, welcher die automatische Initialisierung und Verwendung von MEF (siehe *Managed Extensibility Framework (MEF)* auf Seite 66) für das Nachladen der Module ermöglicht. Diese Basisklasse wurde für unseren Bootstrapper erweitert. Der Bootstrapper initialisiert die Shell und den `ModulesCatalog`.

**Shell** Die Shell stellt das grafische Gerüst der gesamten Applikation dar. Im nächsten Kapitel *Zusammensetzung der Benutzeroberfläche* (auf Seite 93) wird dies genauer erläutert.

**ModulesCatalog** Beim ModulesCatalog werden die verwendbaren Module registriert. Er wird im Bootstrapper durch das Konfigurationsfile ModulesCatalog.xaml gefüllt, wo auch gegenseitige Abhängigkeiten zwischen den Modulen angegeben werden können.

```
1 <Modularity:ModuleCatalog
      xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
```

```
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:sys="clr-namespace:System;assembly=mscorlib"
        xmlns:Modularity=
          "clr-namespace:Microsoft.Practices.Prism.Modularity;
           assembly=Microsoft.Practices.Prism">

    <Modularity:ModuleInfoGroup
                    Ref="Tadm.Web.Client.CoursesModule.xap"
                    InitializationMode="WhenAvailable">
        <Modularity:ModuleInfo ModuleName="CoursesModule" />
    </Modularity:ModuleInfoGroup>

</Modularity:ModuleCatalog>
```

In unserem Fall wird nur das Modul CoursesModule nachgeladen, welches sich im XAP-File `Tadm.Web.Client.CoursesModule.xap` befindet.

**Module** Module werden im `ModulesCatalog` registriert und durch MEF geladen. Sie sind in keiner Weise abhängig von `Tadm.Web.Client` und befinden sich in anderen Assemblies. Da im `ModulesCatalog` nur der Name und das XAP-File des Modules angegeben wird, ist auch `Tadm.Web.Client` nicht direkt von den Modulen abhängig. Durch MEF wäre es sogar möglich einfach nur ein Verzeichnis anzugeben wo sich die nachladbaren Module befinden. Damit liesse sich die Applikation durch ein einfaches Platzieren des XAP-Files eines neuen Modules im entsprechenden Verzeichnis erweitern.

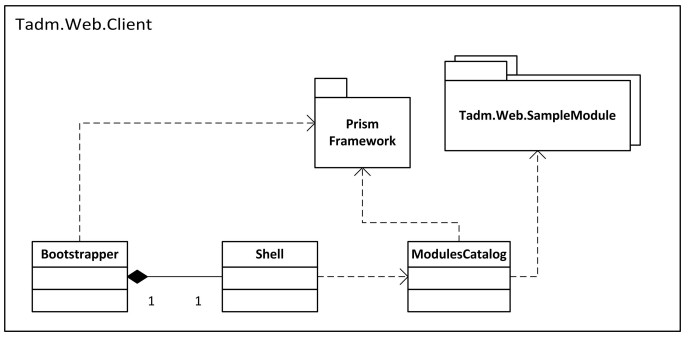



**Abbildung (10.16)** Vereinfachte Darstellung der Abhängigkeiten im Projekt Tadm.Web.Client.

### Tadm.Web.Client.Infrastructure

Das Projekt `Tadm.Web.Client.Infrastructure` kann als gemeinsame Bibliothek für die Module angesehen werden. Hier können immer wieder benötigte Komponenten wie zum Beispiel Converter oder Commands für verschiedene Zwecke zur Wiederverwendbarkeit

platziert werden.

### Tadm.Web.Client.SampleModule

Ausser der Angabe im `ModulesCatalog` muss der Client die einzelnen Module nicht kennen. Die Applikation kann so um beliebig viele weitere Module ergänzt werden, ohne das das Projekt Client neu kompiliert werden muss.

**Module Klasse** Ein Modul beinhaltet eine Klasse die das Interface `Microsoft.Practices.Prism.Modularity.IModule` erfüllt, welche eine Methode Initialize() enthält und wird durch das ExportModule Attribut durch MEF gefunden. Nach dem Laden des Modules sind durch MEF alle exportierten Views des Modules erreichbar. Innerhalb der Initialize() Methode kann angegeben werden welche Views bereits nach dem Laden verwendet werden sollen. Dies ist zum Beispiel bei Ergänzungen für die Hauptnavigation der Applikation sinnvoll. Die Komposition zwischen View und ViewModel wird durch MEF wegen den angegebenen Export und Import Attributen automatisch erledigt.

**View** Ein Modul kann mehrere Views enthalten, welche durch das Export Attribut (siehe *Zusammensetzung der Benutzeroberfläche* auf Seite 93) mit Hilfe von MEF exportiert werden und somit frei zugänglich sind. Ein Modul kann aber auch ohne Views definiert werden.

**ViewModel** Das View Model wird durch MEF per Dependency Injection (siehe *Managed Extensibility Framework (MEF)* auf Seite 66) der View zugänglich gemacht.

**NavigationItemView** NavigationItemViews sind spezielle Views, welche in der Initialize() Methode des Moduls automatisch der Hauptnavigation der Seite zugänglich gemacht werden. In unserem Fall wurde als Navigation ein `Accordion` (eine aufklappbare Navigationsstruktur) definiert. Die NavigationItemViews müssen deshalb vom Typ `AccordionItem` sein.

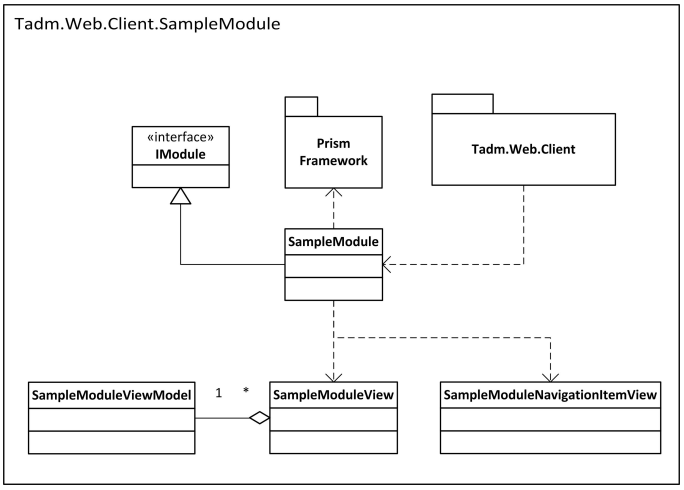



**Abbildung (10.17)** Vereinfachte Darstellung der Abhängigkeiten im Projekt Tadm.Web.Client.

## 10.4.2 Zusammensetzung der Benutzeroberfläche

Prism und MEF bieten Hilfsmittel Views zur Laufzeit nachzuladen und diese dynamisch in sogenannten Regions anzuordnen.

Die Shell bildet das Grundgerüst der Benutzeroberfläche der Applikation und teilt die zur Verfügung stehende Fläche in verschiedene Regionen (Regions). Regions sind Platzhalter für eine oder mehrere Views, welche zur Laufzeit geladen und angezeigt werden können. MEF bietet bereits die Möglichkeit Views durch das Export Attribut automatisch bei einem Container zu registrieren und sobald diese benötigt werden, zu laden. Mit Prism / MEF können diese Views nun durch den RegionManager in den definierten Regions der Shell platziert werden. Wo sich diese Regions befinden, ist nur in der Shell definiert. Keine andere Komponente kennt die Anordnung der Regions. Nachgeladene Module können neue Views in die Regions laden, ohne zu wissen wo sich diese befinden. Damit wird es möglich das Layout zu ändern, ohne das dies eine Änderung in den einzelnen Modulen zur Folge hätte.

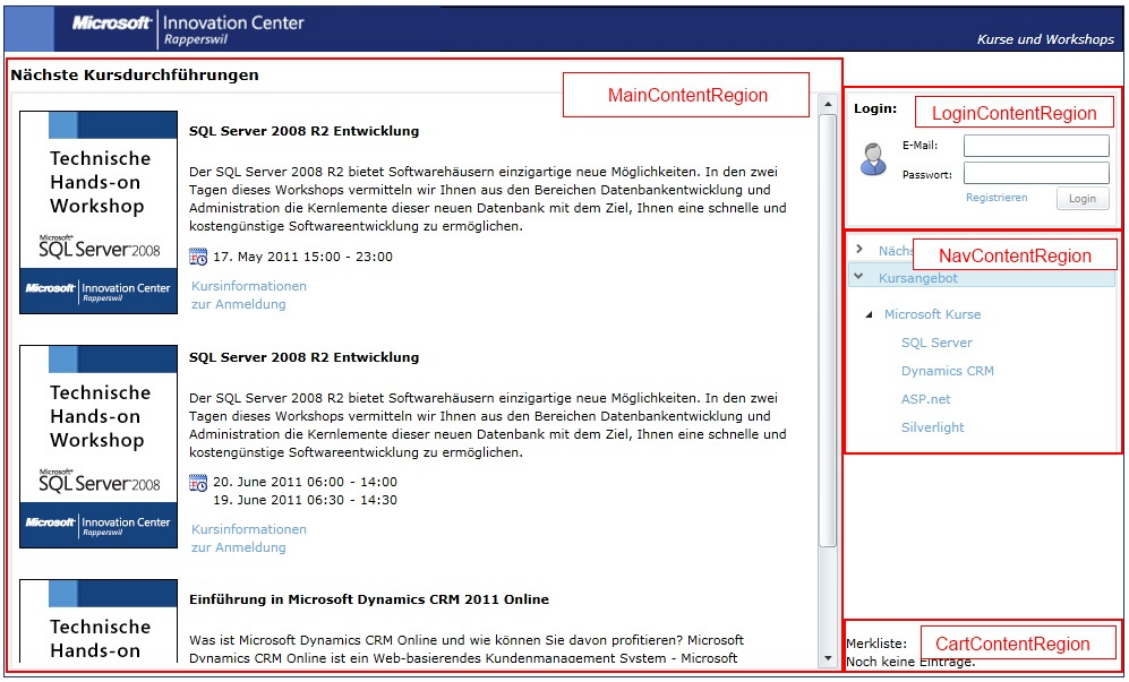



**Abbildung (10.18)** Regions-Aufteilung des Web-Frontends.

Die Deklaration einer Region im XAML erfolgt mittels eines `ContentControl`.

```
1 <ContentControl prism:RegionManager.RegionName="MainContentRegion"/>
```

Der RegionManager kann durch MEF per Importanweisung an die zu verwendenden Komponenten (meist ViewModels) gegeben werden.

```
1 [Import(AllowRecomposition = false)]
  public IRegionManager RegionManager;
```

Nach dem Import des Region Managers kann eine View durch Angabe des Names der View und des Namens der Region folgendermassen in geladen werden.

```
1 RegionManager.RegisterViewWithRegion("MainContentRegion", "MyView");
```

Der folgende Aufruf bewirkt ein Austauschen der angezeigten View mit MyOtherView.

```
1 RegionManager.RequestNavigate("MainContentRegion", "MyOtherView");
```

### 10.4.3 Navigation

Betrachtet man das Silverlight Web-Frontend, so befindet sich auf der rechten Seite die Navigationsleiste mit den Menupunkten „Nächste Kursdurchführungen“ und „Kursangebot“. Ein Klick auf einen dieser Menupunkte bewirkt ein Austauschen der View in der MainContentRegion (siehe *Zusammensetzung der Benutzeroberfläche* auf Seite 93) auf der linken Seite.

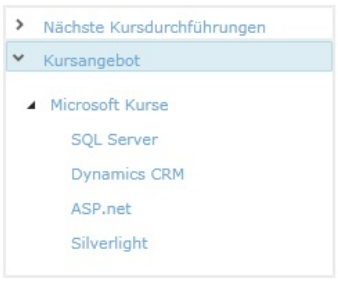



**Abbildung (10.19)** Darstellung der Navigationsleiste.

Eine elegante Art eine Navigation zu erstellen, beschreibt das Prism QuickStart Projekt „View Switching Navigation“ [Mic10]. Dieses Beispiel diente als Grundlage für die Implementierung der Navigation des Kursanmeldungs-Web-Frontends. Weil die Menupunkte der Navigation in der `Initialize()` Methode des Moduls beim Laden des Moduls durch den RegionManager registriert werden, können ohne Neukompilierung weitere Module automatisch in die Navigation integriert werden.

```
1  [ModuleExport(typeof(CoursesModule))]
   public class CoursesModule : IModule
   {
       [Import]
5      public IRegionManager RegionManager;

       public void Initialize()
       {
           // place own navigation items
10         RegionManager.RegisterViewWithRegion(
               RegionNames.NavContentRegion,
               typeof(CourseRunsNavigationItemView)
           );
           RegionManager.RegisterViewWithRegion(
```

```
15              RegionNames.NavContentRegion,
                typeof(CoursesNavigationItemView)
            );
        }
}
```

Durch das Attribut `ViewSortHint` kann bei der `NavigationsItemView` eine gewünschte Sortierung angegeben werden. Diese Art Sortierung funktioniert allgemein beim Laden von Views in der gleichen Region. Wird zweimal der gleiche `ViewSortHint` angegeben, so werden die Views in der Reihenfolge angezeigt in welcher sie geladen wurden.

```
1  [Export]
   [ViewSortHint("02")]
   public partial class CoursesNavigationItemView
```

Es gibt mehrere Möglichkeiten Logik an verschiedenen Punkten der Navigationsereignisse einzuhaken. Durch das `Microsoft.Practices.Prism.Regions.INavigationAware` Interface können Views und ViewModels von automatisch benachrichtigt werden, wenn ein Navigationsrequest auf den RegionManager ausgelöst wird. Es werden dazu die Methoden `IsNavigationTarget()`, `OnNavigatedFrom()` und `OnNavigatedTo()` angeboten, welche beim Aufruf `RegionManager.RequestNavigate()` automatisch aufgerufen werden. Durch das Interface `IConfirmNavigationRequest` kann ausserdem ein Wegnavigieren von einer View zuerst vom Benutzer bestätigt oder abgelehnt werden.

### 10.4.4 Automatische Aktualisierung des aktuell gewählten Menupunktes

In unserem Kursanmeldungs-Frontend kann man direkt auf eine bestimmte Kursdurchführung navigieren, wenn man einen Kurs betrachtet. Bei der Navigation wird dabei automatisch der Menupunkt „Nächste Kursdurchführungen“ aktiviert. Um dies so einfach wie möglich zu bewerkstelligen, wurde etwas ViewLogik im Code Behind eingefügt, wie dies auch im Prism QuickStart gemacht wurde.

```
1  void IPartImportsSatisfiedNotification.OnImportsSatisfied()
   {
      IRegion mainContentRegion =
       regionManager.Regions[RegionNames.MainContentRegion];
5
      if (mainContentRegion != null &&
       mainContentRegion.NavigationService != null)
      {
```

```
        mainContentRegion.NavigationService.Navigated +=
            this.MainContentRegion_Navigated;
    }
}

public void MainContentRegion_Navigated(object sender,
                          RegionNavigationEventArgs e)
{
    this.UpdateNavigationButtonState(e.Uri);
}

private void UpdateNavigationButtonState(Uri currentUri)
{
    this.IsSelected = (uri == currentUri);
}
```

Durch die Implementierung des `System.ComponentModel.Composition.IPartImportsSatisfiedNotification` Interfaces der View kann zum Zeitpunkt nach dem Erstellen der NavigationItemView ein Eventhandler Navigated Event der `MainContentRegion` angehängt werden. Dieser betrachtet die aktuelle URI der `MainContentRegion`, welche die aktuell geladenen View bezeichnet und kann somit feststellen, ob das `NavigationItem` als selektiert gesetzt werden soll.

### 10.4.5 Validation

Sowohl das Registrationsformular, als auch das Loginformular enthält Validationslogik. Die Logik dazu ist in der Basisklasse `Tadm.Web.Client.Infrastructure.Util.ValidationObject` gekapselt, welche die Interfaces `INotifyPropertyChanged` und `INotifyDataErrorInfo` implementiert und `System.ComponentModel.INotifyDataErrorInfo` verlangt das Property `HasErrors`, den Event `ErrorsChanged` und die Methode `GetErrors(string PropertyName)` und ermöglicht es ein Objekt automatisch zu validieren, sobald ein Property gesetzt wird Fehlermeldungen anzuzeigen.

Die Domain Objekte, welche mit WCF RIA Services benützt werden, würden eigentlich `INotifyPropertyChanged` und `INotifyDataErrorInfo` schon erfüllen. Da wir aber in unseren DTO's keine Passworte übermitteln, wurden für die Registrierung und das Loginformular eigene Clientseitig verwendbare Objekte erstellt.


**Abbildung (10.20)** Screenshot des Registrationsformulares.

```
1  [CustomValidation(typeof(Validators), "Email")]
   [Required(ErrorMessage = "Bitte geben Sie Ihre Email Adresse an!")]
   public string Email
   {
5      get
       {
           return Contact.Email;
       }
       set
10     {
           ValidateProperty("Email", value);
           Contact.Email = value;
           RaisePropertyChanged("Email");
       }
15  }
```

Um die Gültigkeit der Email-Adresse zu ermitteln, wurde im vorherigen Codebeispiel das Attribut `[CustomValidation(typeof(Validators), "Email")]` verwendet. Dieses weist das `ValidationObject` an, die Klasse Validators und deren statische Methode `Email()` zu verwenden. Diese wurde folgendermassen implementiert:

```
1  public static ValidationResult Email(string email,
       ValidationContext validationContext)
   {
       Regex regEmail = new Regex(matchEmailPattern);
5      if (regEmail.IsMatch(email))
```

```
        return ValidationResult.Success;
    return new ValidationResult(
        invalidEmailMessage,
        new[] { validationContext.MemberName });
}
```

Neben Validatoren für einzelne Properties können auch Validatoren für ein ganzes ValidationObject angegeben werden. Dies wurde zum Beispiel nötig um zu überprüfen, ob beim Profil zweimal das gleich Passwort eingegeben wurde.

### 10.4.6 Behaviors

Commands können bei Buttons oder Hyperlinks auf Klickereignisse gebunden werden. Um Logik aber auch bei anderen Ereignissen der Benutzeroberfläche auszulösen, kann das Handling des Ereignisses im Code Behind der View implementiert werden. Damit wird aber das MVVM-Pattern verletzt. Die View sollte ja möglichst keine Logik enthalten.

Für die konsequente Umsetzung von MVVM, können Elemente der Benutzeroberfläche mit zusätzlichem Verhalten (Behavior) ausgestattet werden. Prism bietet dazu zum Beispiel das Behavior `CommandBehaviorBase<T>` wobei T von `System.Windows.Controls.Control` abgeleitet sein muss. Es ermöglicht die Bindung eines Commands an ein Control. Da das Control im Kunstruktor des Behaviors and das Behavior übergeben wird, kann dort auf die definierten Events des Controls zugegriffen werden.

Im folgenden Codebeispiel wird ein `KeyEventHandler` auf den KeyUp Event eines `TextBox` Controls gehängt. Dieser KeyDown Event führt das gebundene Command aus. Wenn das Control ausgewählt ist wird das Command also ausgelöst, sobald der Benutzer auf eine Taste gedrückt hat.

```
public class TextBoxValueChangedBehavior : CommandBehaviorBase<TextBox>
{
    public TextBoxValueChangedBehavior(TextBox element) : base(element)
    {
        element.KeyUp += new KeyEventHandler(element_KeyUp);
    }

    void element_KeyUp(object sender, KeyEventArgs e)
    {
        TextBox control = sender as TextBox;
        base.CommandParameter = control.Text;
        base.ExecuteCommand();
    }

}
```

Es muss aber noch ein „Attached Property“ für die Injektion des Behaviors in das Control definiert werden. Dieses kann in XAML verwendet werden um das Command zu binden, nach dem der Namespace deklariert wurde in welchem sich das „Attached Property“ befindet.

Die schlussendliche Verwendung des Behaviors durch die Angabe des „Attached Property“ und das Binden des Commands in XAML:

```
<UserControl x:Class="Tadm.Web.Client.CoursesModule.Views.LoginView"
    ...
    xmlns:commands="clr-namespace:Tadm.Web.Client.Infrastructure.Commands;
    assembly=Tadm.Web.Client.Infrastructure"
    ...
<TextBox Text="{Binding Email, Mode=TwoWay}"
commands:TextBoxValueChanged.Command="{Binding EmailValueChangedCommand}"/>
        ...
</UserControl>
```

### 10.4.7 Interaction Requests

Mit `NotificationInteractionRequest` oder `ConfirmationInteractionRequest` kann ein Programmablauf mit einfachen Benutzerinteraktionen ganz ohne Code Behind gesteuert werden. `NotificationInteractionRequest` löst einen Request aus, der nach Bestätigung durch den Benutzer eine weitere Aktion auslöst. `ConfirmationInteractionRequest` kann je nach Entscheid des Benutzers zwei verschiedene Aktionen auslösen.

Im folgenden Codebeispiel wird auf einem `ConfirmationInteractionRequest` die Methode `Raise()` aufgerufen, welche ein `Confirmation` Objekt und eine Aktion verlangt:

```
Confirmation confirmation = new Confirmation()
{
    Title = "Login Fehler",
    Content = "...Wollen sie ihr Passwort zurücksetzen?..."
};

ConfirmationInteractionRequest.Raise(
    confirmation,
    c =>
    {
        if (!c.Confirmed)
        {
            focusOnLogin();
```

```
                return;
            }
            resetPassword();
            return;
        }
);
```

Für die Interaktion mit dem Benutzer wird in der View ein Trigger benötigt, welcher durch den `InteractionRequest` ausgelöst wird.

Folgendes XAML-Beispiel zeigt die Angabe eines Triggers, an welchen ein `ConfirmationInteractionRequest` gebunden wird. Dieser ist für das Auslösen des View Ereignisses verantwortlich, sobald `Raise()` auf den `ConfirmationInteractionRequest` aufgerufen wird.

```
<UserControl.Resources>
    <DataTemplate x:Name="ConfirmationDialogTemplate">
        <TextBlock
            HorizontalAlignment="Center"
            VerticalAlignment="Center" Text="{Binding}"/>
    </DataTemplate>
    ...
</UserControl.Resources>

<ei:Interaction.Triggers>
    <prism:InteractionRequestTrigger
    SourceObject="{Binding ConfirmationInteractionRequest}">
        <prism:PopupChildWindowAction
        ContentTemplate="{StaticResource ConfirmationDialogTemplate}"/>
    </prism:InteractionRequestTrigger>
    ...
</ei:Interaction.Triggers>
```


**Abbildung (10.21)** Screenshot einer Rückmeldung an den Benutzer.

## 10.5 Zusammenfassung

Folgende Grafik zeigt den Aufbau des Gesamtsystems mit den verschiedenen Stellen an welchen Anpassungen gemacht oder eigene Komponenten implementiert wurden:

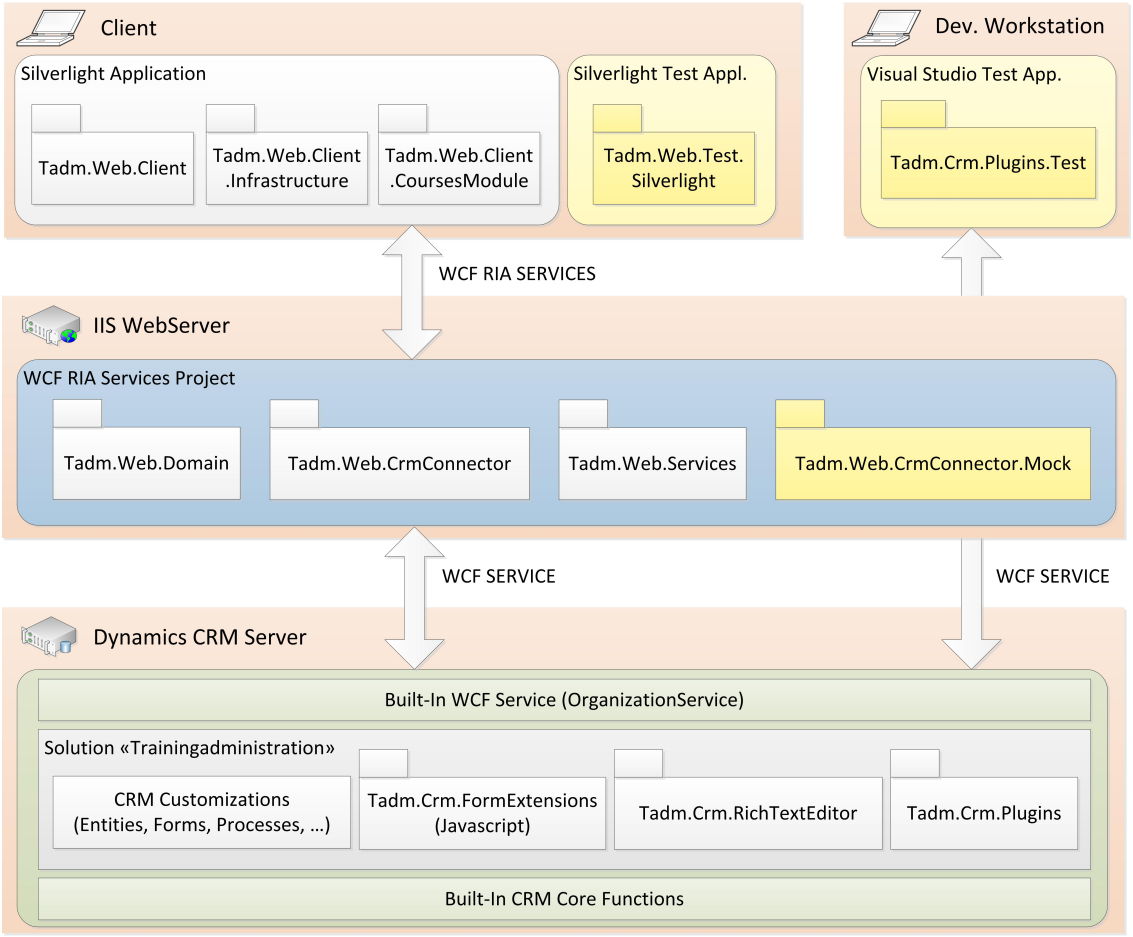



**Abbildung (10.22)** Gesamtüberblick (schematisch)

# 11 Werkzeuge und Infrastruktur

## 11.1 Entwicklungswerkzeuge und Ressourcen

### 11.1.1 Microsoft Visual Studio 2010

Die Entwicklung der Lösung findet fast ausschliesslich innerhalb Visual Studio 2010 (Ultimate Edition) statt. Dieses Werkzeug bietet eine optimale Unterstützung für die verwendeten Technologien und ist über das „MSDN Academic Alliance Software Center“ für Studienzwecke an der HSR verfügbar.

### 11.1.2 Microsoft Expression Blend

Um das Layout der Silverlight-Komponenten anzupassen, wurde mit Expression Blend gearbeitet. Dieses Werkzeug bietet eine sehr schöne grafische Unterstützung der Layout Prozesse und vereinfacht die Verwaltung von Layout Ressourcen erheblich. Ausserdem ist es sehr gut in Visual Studio integriert und lässt sich auf Knopfdruck aus Visual Studio dazu schalten.

### 11.1.3 CRM-SDK

Neben den erforderlichen Klassenbibliotheken bietet das „Dynamics CRM Software Development Kit“ einige Beispielprojekte, sehr ausführliche Dokumentationen, sowie kleine Tools um administrative Abläufe wie z.B. das registrieren von Plugins massiv zu vereinfachen.

### 11.1.4 Microsoft Prism

Prism ist eine frei erhältliche Sammlung von „Patterns & Practices“ für Silverlight und WPF Applikationen. Neben einigen Klassenbibliotheken bietet es umfangreiche Beispielprojekte und eine ausführliche Dokumentation. Es ist auf der Microsoft Open-Source Plattform „Codeplex“ frei verfügbar.

## 11.2 Entwicklungsinfrastruktur

### 11.2.1 CRM-Entwicklungsserver

Um die CRM Erweiterungen zu entwickeln und sie zugleich als Testdatenbank zu verwenden, steht eine Instanz auf dem virtuellen CRM2011 Entwicklungsserver der Firma Dynamix Solutions GmbH zur Verfügung (Windows 2008 Server, SQL Server 2008 R2, Dynamics CRM 2011).

### 11.2.2 Team Foundation Server

Als Source Control und Work-Item Server, steht ebenfalls ein virtueller Server der Firma Dynamix Solutions GmbH zur Verfügung (Windows 2008 Server, Team Foundation Server 2010).

### 11.2.3 Build- und Testrechner

Zusätzlich zum normalen Team Foundation Server steht für Testzwecke ein virtueller Rechner mit Windows 7 und Visual Studio 2010 zur Verfügung. Dieser holt sich per geplantem Script jeweils in der Nacht den aktuellen Stand vom Team Foundation Server, kompiliert diesen und stellt ihn auf seinem lokalen IIS als Testinstanz zur Verfügung.

### 11.2.4 Entwicklungsrechner

Für die Entwicklung stehen zwei Arbeitsplatzrechner der HSR zur Verfügung (Windows 7, Visual Studio 2010).

## 11.3 Dokumentation

Die Dokumentation wird weitgehend mit Hilfe der LaTeX Technologie erstellt. Sie eignet sich sehr gut um parallel am gleichen Dokument zu arbeiten und dieses mit Hilfe einer Versionsverwaltung beidseitig aktuell zu halten.

# 12 Testing

## 12.1 Testing der CRM Komponenten

Durch die Implementation der automatischen Statuswechsel und der zusätzlichen Validierungen auf dem CRM, müssen alle Fälle gemäss dem Statusdiagramm (siehe *Statusverwaltung* auf Seite 74) einfach getestet werden können. Um dies gewährleisten zu können, wurden Unit Tests erstellt, welche die installierten Plugins auf der CRM Testinstanz per `OrganizationService` ansprechen.

Die restlichen CRM Erweiterungen (Formulare, Entitäten, etc.) lassen sich leider nicht sinnvoll automatisiert testen. Sie werden mit Hilfe der manuellen Systemtests anhand der UseCases in den Testablauf mit einbezogen.

Da sich die CRM Core Events (Entität erzeugen, Status schieben, etc.) nicht durch einen Mock ersetzen lassen, müssen die erwähnten Unit Tests für jeden Test Daten erzeugen und diese nach durchlaufenem Test wieder löschen. Dies ist grundsätzlich möglich ohne die restlichen Daten im CRM zu tangieren. Bricht aber ein Test während dem Durchlauf ab, sollten die restlichen Testdaten manuell wieder gelöscht werden. Leider funktioniert das nicht in `finally` oder `cleanup` Blöcken, da für die Entitätsreferenzen innerhalb des Tests der aktive Kontext benötigt wird, der beim Abbruch mit Exception verloren geht.

Die Tests sind alle eigenständig, erzeugen also alle die Testdaten und löschen sie anschliessend wieder. Aufbauend von neuen Entitäten, werden die Initialstatus sowie die automatischen Statuswechsel geprüft. Zusätzlich wird die Verhinderung von doppelten E-Mail Adressen geprüft.

### 12.1.1 Testklasse: ContactPluginTests

**CheckDuplicatedEmailAddress** Zwei Kontakte werden mit der gleichen E-Mail Adresse erstellt. Geprüft wird, ob tatsächlich die erwartete Exception auftritt.

**CheckNoEmailAddress** Es werden ebenfalls zwei Kontakte erstellt. Beide ohne E-Mail Adresse. Dies soll so funktionieren, da die restliche CRM Funktionalität nicht weiter beeinträchtigt werden soll (abgesehen von der Verhinderung von doppelten E-Mail Adressen im restlichen CRM, dies lässt sich leider nicht vermeiden).

### 12.1.2 Testklasse: StatePluginsTests

**TestInitialStates** Als erstes soll geprüft werden, ob neue Kurs- und Kursdurchführungsentitäten automatisch im richtigen Status erfasst werden. Dies sind die beiden Entitäten welche von den automatischen Statuswechseln betroffen sind.

**TestPublishedNoUpcomingRunsState** Ein Kurs und eine Durchführung dazu wird erstellt, nur der Kurs publiziert. Der Kurs soll im Status „Keine offenen Durchführungen“ verbleiben.

**TestPublishedUpcomingRuns** Nun werden der Kurs und die dazugehörige Durchführung publiziert. Der Kurs sollte sich automatisch in den Status „Hat offene Kursdurchführungen“ schieben. Die Durchführung im Status „publiziert“ muss verbleiben.

**TestUnpublishedUpcomingRuns** Der Kurs und die dazugehörige Durchführung werden beide publiziert, der Kurs hat sich automatisch in den Status „Hat offene Durchführungen“ geschoben. Nun wird die Kursdurchführung wieder abgebrochen und es wird überprüft ob sich der Kurs selber zurück in den Status „Keine offenen Kursdurchführungen“ geschoben hat.

**TestCancelledAndCancelRuns** Bei publiziertem Kurs und publizierter Durchführung wird der Kurs abgebrochen. Es wird getestet, ob die Durchführung auch korrekt abgebrochen wurde.

**TestRegisterOneParticipant** Es wird einer publizierten Kursdurchführung eine Person angemeldet. Da die minimale Anzahl Teilnehmer für die Test-Kursdurchführung bei zwei Personen liegt, sollte der Status der Durchführung nicht geändert werden, sondern auf „veröffentlicht“ verbleiben.

**TestRegisterTwoParticipants** Nun werden an der publizierten Kursdurchführung zwei Personen angemeldet. Das Limit der minimalen Teilnehmerzahl wird dadurch überschritten und die Durchführung soll automatisch bestätigt werden.

**TestRegisterFourParticipants** Werden an der publizierten Kursdurchführung nun vier Personen angemeldet, ist die maximale Teilnehmerzahl erreicht. Die Durchführung sollte nun automatisch den Status „voll“ anehmen.

**TestRegisterFourParticipantsAndCancelOne** Wird eine Kursteilnahme an einem vollen Kurs abgebrochen, soll der Kurs automatisch wieder den Status „bestätigt“ annehmen.

**TestRegisterFourParticipantsAndCancelThree** Werden mehrere Kursteilnahmen an einem vollen Kurs abgebrochen und die Anzahl bestätigter Teilnehmer sinkt wieder unter das Minimum, soll der Kurs zwar in den Status „bestätigt“ zurückfallen, aber nicht wieder in den Status „geplant“. Einmal bestätigte Kurse sollen nur im

Ausnahmefall wieder abgebrochen werden müssen.

### 12.1.3 Testergebnisse der automatischen CRM-Plugin Tests

Alle Unit-Tests laufen fehlerfrei durch.

Da die Tests auf die installierten Plugins innerhalb der CRM Testumgebung wirken, kann die Codeabdeckung leider nicht automatisch berechnet werden. Querchecks haben aber ergeben dass sie praktisch die gesamte Plugin-Funktionalität abdecken.

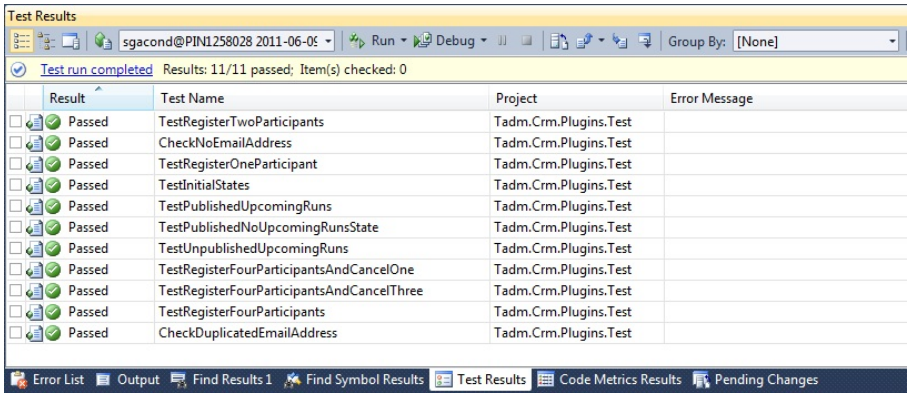



**Abbildung (12.1)** Testergebnis der automatisierten CRM Plugin Tests

## 12.2 Testing des Web Frontends

Um das Web Frontend zu testen wurde das Silverlight Unit Testing Framework verwendet. Dieses ist im Silverlight 4.0 Toolkit enthalten. http://silverlight.codeplex.com/

Durch die weitgehende Einhaltung des MVVM-Patterns konnten alle Tests direkt auf die ViewModels gemacht werden, welche den RIA Service benützen. Auf weitere automatisierte Tests des Web Frontends wurde verzichtet.

Alle implementierten Silverlight Tests befinden sich in Projekt `Tadm.Web.Test.Silverlight`.

### 12.2.1 Vorgetäuschter Domain Connector für die Silverlight Tests

Um nicht nochmal die ganze Funktionalität im CRM zu testen, wurde für die Tests ein eigener `CrmConnectorMock` geschrieben welcher das `IDomainConnector` implemen-

tiert. Dieser täuscht die Funktionalität des CRM's vor und erstellt Domain Objekte welche für die Tests verwendet werden können.

Damit die Tests den `CrmConnectorMock` verwenden, muss im Web.config des Service Projektes `Tadm.Web.Services` zuerst der Eintrag `UseDummyData` auf `True` gesetzt werden. Für die Verwendung des richtigen `CrmConnector` bei produktiven Einsatz muss diese Einstellung auf `False` gesetzt sein.

```
<configuration>
    <appSettings>
        ...
      <add key="UseDummyData" value="true"/>
        ...
    </appSettings>
    ...
</configuration>
```

### 12.2.2 Testen von asynchronen Aufrufen

Da im Web-Frontend praktisch alle Aktionen abhängig von Daten aus dem Webservice sind und diese Aufrufe immer asynchron sein müssen, muss auch in den Tests auf diese asynchronität geachtet werden. Das Silverlight Unit Testing Framework bietet hier einige Tools um asynchrone Testaufrufe aneinander zu hängen. `EnqueueConditional()` kann verwendet werden um auf ein bestimmtes Ereignis zu warten, während man mit `EnqueueCallback()` Testcode für eine spätere Ausführung definieren kann.

Die meisten Ereignisse auf welche man beim Testen von ViewModels warten muss, sind Aktualisierungen von Properties des ViewModels. Da die implementierten ViewModels alle das Interface `INotifyPropertyChanged` erfüllen, wurde für das Warten auf den `PropertyChanged` Event eines definierten Properties, in einer wiederverwendbaren Basisklasse `ViewModelTestBase` eine eigene Methode definiert.

Folgende Methode wurde für das Warten auf die Aktualisierung eines Properties implementiert:

```
protected void testAfterPropertyChanged(
    string propertyName,
    INotifyPropertyChanged viewModel,
    Action testCode,
    Action afterTest = null)
{
  bool propChanged = false;
  viewModel.PropertyChanged += (s, e) =>
```

```
    {
        if (e.PropertyName == propertyName)
            propChanged = true;
    };

    EnqueueConditional(() => propChanged);

    EnqueueCallback(() =>
    {
        testCode.Invoke();
        if (afterTest != null)
            afterTest.Invoke();
        else
            EnqueueTestComplete();
    });
}
```

Da alle Daten Transfer Objekte auf der Clientseite auch `INotifyPropertyChanged` erfüllen, kann diese Hilfsmethode auch für das Warten auf das Nachladen von abhängigen Daten verwendet werden. Da dies aber nur die Funktionalität von WCF RIA Services testen würde, wurde auf solche Tests verzichtet.

### 12.2.3 Testergebnisse der automatischen Silverlight Tests

Alle Unit-Tests laufen fehlerfrei durch.

## 12.3 Manuelle Tests des Gesamtsystems

Anhand der UseCases wird das ganze System nochmals systematisch durchgetestet.

### 12.3.1 Testumgebung

**Dynamics CRM** Entwicklungsinstanz „Trainingadministration“ auf CRM 2011 Entcklungsserver (mit Testdaten)

**Web Oberfläche** Virtueller Build- und Testrechner (Windows 7, IIS 7), nightly build vom 7.6.2011

**Client** Arbeitsplatzrechner, HSR Domäne, Windows 7, Internet Explorer 9

**Datum** 7.6.2011

**Durchgeführt von** Silvan Gacond

### 12.3.2 Testfall 1: UC I01 A

| **UseCase** | UC I01 |
|---|---|
| **Ausgangslage** | Es sind bereits Kurse und Durchführungen im System erfasst und nach Kategorien zugeordnet. |
| **Ablauf** | • Zugriff auf die Weboberfläche.<br>• Nächste Kursdurchführungen werden angezeigt.<br>• Navigation durch Kursbaum auf der rechten Seite.<br>• Die entsprechenden Kurse werden angezeigt. |
| **Erwartet** | Die publizierten Kurse resp. Durchführungen werden korrekt angezeigt. |
| **Eingetroffen** | Drei publizierte Testkurse werden in den richtigen Kategorien angezeigt. Die nächsten Kursdurchführungen werden korrekt angezeigt. Unter den Kursdetails werden die korrekten Durchführungen angezeigt. |
| **Testresultat** | OK |

**Tabelle (12.1)** Testfall 1: UC I01 A

### 12.3.3 Testfall 2: UC I02 A

| **UseCase** | UC I02 |
|---|---|
| **Ausgangslage** | Der Interessent ist mit der entsprechenden E-Mail Adresse noch nicht registriert. |
| **Ablauf** | • Registrieren eines neuen Benutzers mit einer neuen E-Mail Adresse. |
| **Erwartet** | Der Benutzer wird erfolgreich registriert und freigeschaltet. |
| **Eingetroffen** | Der Benutzer kann sich normal anmelden und ist dann bereits am System eingeloggt. Der Kontakt im CRM besitzt den richtigen Status und das Login nach manuellem Logout funktioniert. |
| **Testresultat** | OK |

**Tabelle (12.2)** Testfall 2: UC I02 A

### 12.3.4 Testfall 3: UC I02 B

| | |
|---|---|
| **UseCase** | UC I02 |
| **Ausgangslage** | Der Interessent ist mit der entsprechenden E-Mail Adresse bereits registriert. |
| **Ablauf** | • Versuch einen Benutzer mit bestehender E-Mail Adresse zu Registrieren. |
| **Erwartet** | Der Benutzer kann sich nicht anmelden. Das System fragt nach ob das Passwort vergessen wurde. |
| **Eingetroffen** | Korrektes Verhalten, entsprechende Meldung erscheint. |
| **Testresultat** | OK |

**Tabelle (12.3)** Testfall 3: UC I02 B

### 12.3.5 Testfall 4: UC I03 A

| | |
|---|---|
| **UseCase** | UC I03 |
| **Ausgangslage** | Der Interessent ist registriert und eingeloggt. |
| **Ablauf** | • Navigation zur entsprechenden Kursdurchführung.<br>• Anmeldung an die entsprechenden Kursdurchführung auslösen. |
| **Erwartet** | Anmeldung erfolgt, Bestätigung im CRM wird gesetzt und E-Mail versandt. |
| **Eingetroffen** | Anmeldung korrekt erfolgt und wurde im CRM durch den Workflow asynchron bestätigt. E-Mail geht wegen der Konfiguration des Entwicklungsservers nicht raus (fehlende Komponente), ist aber in den Systemjobs des CRM zum Versand bereit. |
| **Testresultat** | Nicht komplett Testbar, testbare Teile OK. |

**Tabelle (12.4)** Testfall 4: UC I03 A

### 12.3.6 Testfall 5: UC K01 A

| | |
|---|---|
| **UseCase** | UC K01 |
| **Ausgangslage** | Der Kursteilnehmer hat sich an eine publizierte Kursdurchführung angemeldet und ist eingeloggt. |
| **Ablauf** | • Navigieren zur entsprechenden Kursdurchführung.<br>• Anzeige kontrollieren. |
| **Erwartet** | Status der Durchführung und Status der Anmeldung wird angezeigt. |
| **Eingetroffen** | Status wird korrekt angezeigt. Anmeldestatus nur wenn eine Anmeldung erfolgt ist. |
| **Testresultat** | OK |

**Tabelle (12.5)** Testfall 5: UC K01 A

### 12.3.7 Testfall 6: UC A01 A

| | |
|---|---|
| **UseCase** | UC A01 |
| **Ausgangslage** | Der Mitarbeiter hat CRM Zugang und die erforderlichen Rechte. |
| **Ablauf** | • Navigation zum Kursverwaltungs-Dashboard.<br>• Neuer Kurs erstellen.<br>• Bild des Kurses erfassen.<br>• Neue Kursdurchführung erstellen.<br>• Entsprechende Kurstage (Service - Aktivitäten) erstellen.<br>• Status von Kurs und Durchführung auf „publiziert“ resp. „keine Offenen Durchführungen“ schieben. |
| **Erwartet** | Kurs und Durchführung erfolgreich erfasst, erscheint in Kursverwaltungssystem. |
| **Eingetroffen** | Abwarten bis der Cache der Weboberfläche die Daten neu geladen hat. Dann Kurs korrekt erschienen. |
| **Testresultat** | OK |

**Tabelle (12.6)** Testfall 6: UC A01 A

### 12.3.8 Testfall 7: UC A02 A

| | |
|---|---|
| **UseCase** | UC A02 |
| **Ausgangslage** | Der Mitarbeiter hat CRM Zugang, die erforderlichen Rechte und die entsprechende Durchführung hat bereits Anmeldungen. |
| **Ablauf** | • Neue Anmeldung innerhalb des CRM erfassen (für bestehenden Benutzer mit Webzugriff).<br>• Anzeige in der Weboberfläche kontrollieren.<br>• Anmeldung wieder löschen.<br>• Anzeige in der Weboberfläche kontrollieren. |
| **Erwartet** | Die Manipulationen lassen sich fehlerfrei ausführen und die Anzeige auf der Weboberfläche ist korrekt. |
| **Eingetroffen** | Nach Erneuerung des Cache (abwarten der Ablaufzeit), wurden die Änderungen korrekt dargestellt. |
| **Testresultat** | OK |

**Tabelle (12.7)** Testfall 7: UC A02 A

### 12.3.9 Testfall 8: UC A04 A

| | |
|---|---|
| **UseCase** | UC A04 |
| **Ausgangslage** | Der Mitarbeiter hat CRM Zugang und die erforderlichen Rechte um die Themen innerhalb Dynamics CRM zu verwalten. |
| **Ablauf** | • Navigation zu den CRM Einstellungen / Themen.<br>• Erfassen eines neuen Themas unterhalb des Vaterknotens der Kursadministration in der Baumstruktur.<br>• Überprüfen der Anzeige auf der Weboberfläche (Kategoriebaum in de Navigation). |
| **Erwartet** | Die neue Kategorie wird im Baum sichtbar. |
| **Eingetroffen** | Nach Erneuerung des Cache (abwarten der Ablaufzeit), wurde die neue Kategorie korrekt dargestellt. |
| **Testresultat** | OK |

**Tabelle (12.8)** Testfall 8: UC A04 A

### 12.3.10 Testfall 9: UC A05 A

| | |
|---|---|
| **UseCase** | UC A05 |
| **Ausgangslage** | Der Benutzer hat CRM Zugang und die erforderlichen Rechte. |
| **Ablauf** | • Der Benutzer navigiert zur automatisch erstellten Marketingliste der Kursanmeldungen.<br>• Er initiiert eine Schnellkampagne mit den Mitgliedern der Marketingliste. |
| **Erwartet** | Die Angemeldeten Teilnehmer erscheinen alle in der entsprechenden Marketingliste. Die Schnellkampagne ist eine Basisfunktionalität des CRM und muss hier nicht weiter getestet werden. |
| **Eingetroffen** | Die Marketingliste wurde nach Anmeldeschluss mit den Teilnehmern gefüllt. Sie kann verwendet werden. |
| **Testresultat** | OK |

**Tabelle (12.9)** Testfall 9: UC A05 A

## 12.4 Tests durch Drittpersonen

### 12.4.1 Einfacher Usabilitytest mit einer unbeteiligten Person

Um die Bedienbarkeit der Weboberfläche zu testen, wurde eine unbeteiligte Person beigezogen, welche in die Zielgruppe der ausgeschriebenen Kurse passen könnte. Lydia Mosberger, 27 Jährig, arbeitet in der Studierendenadministration einer Hochschule in Zürich und hat unter Anderem die Betreuung des Verwaltungssystems der Studierenden unter sich. Sie verfügt über eine kaufmännische Grundausbildung mit zusätzlichen Diplomen in der Informatik.

Die Testperson wurden ohne weitere Erklärung Aufgaben gegeben (siehe unten) und den Weg zum Ziel beobachtet. Zusätzlich wurde de Kommentar der Testperson notiert.

### Testumgebung

**Dynamics CRM** Entwicklungsinstanz „Trainingadministration“ auf CRM 2011 Entcklungsserver (mit Testdaten)

**Web Oberfläche** Virtueller Build- und Testrechner (Windows 7, IIS 7), nightly build vom 13.6.2011

**Client** MacBook Pro, Windows 7 (virtualisiert, VMware Fusion 3.1), Internet Explorer 8

**Datum** 13.6.2011

### Test 1

| | |
|---|---|
| **Aufgabe** | Information über Durchführungsdaten und -orte des Kurses „SQL Server 2008 R2“ beschaffen. |
| **Herangehensweis** | Hat die Kursdurchführung auf der ersten Ansicht der nächsten Kursdürchführungen sofort entdeckt und die Durchführungsdaten gefunden. Mit einem Klick auf die Kursinformationen wurde noch die fehlende Information über den Durchführungsort beschafft. |
| **Kommentar** | Die Ansicht ist übersichtlich und die Information ist auf den ersten Blick ersichtlich. |

**Tabelle (12.10)** Usability Test 1

### Test 2

| | |
|---|---|
| **Aufgabe** | Sind noch Plätze in der Kursdurchführung „SQL Server 2008 R2“ diesen Juni frei? |
| **Herangehensweis** | Auch hier wurde in der Ansicht „nächste Kursdürchführungen“ gestartet. Mit einem Klick auf die Kursinformationen wurde die Information über die Belegung des Kurses gefunden. |
| **Kommentar** | Auch dies war einfach. |

**Tabelle (12.11)** Usability Test 2

### Test 3

| | |
|---|---|
| **Aufgabe** | Anmeldung an den Kurs „SQL Server 2008 R2“ (Durchführung vom Juni 2011). |
| **Herangehensweis** | Aus der vom vorherigen Test noch geöffneten Ansicht wurde die Anmeldeinformation geöffnet (Schaltfläche: „zur Anmeldung“). Dort war ersichtlich, dass dieser Kurs eine Registrierung erfordert. Diese wurde erledigt, dann mit der „zurück“ Schaltfläche zurück zur Anmeldeinformation navigiert. Dort wurde die Anmeldung mit einem Klick erledigt. |
| **Kommentar** | Kein Problem. Vielleicht könnte die Applikation nach der Registrierung direkt zurück zur letzten Ansicht wechseln. |

**Tabelle (12.12)** Usability Test 3

### Ergebnis

Alle Aufgaben konnten, auch von einer unbeteiligten Person, einfach und ohne grosse Umwege erledigt werden. Eine kleine Verbesserung wurde notiert.

# 13 Weiterentwicklung

## 13.1 Ablösung des aktuellen Systems

Mit der vorliegenden Bachelorarbeit wurde hauptsächlich das Ziel verfolgt, das bestehende Kursanmeldungssystem auf der Webseite des Instituts für vernetzte Systeme abzulösen. Wird das System in der aktuellen Form so von den zukünftigen Benutzern abgenommen, lässt sich dies innert Kürze realisieren. Der aktuell erfasste Adressstamm sowie die entsprechenden Kursdaten müssten dazu übernommen werden. Dies wäre z.B. mit den SQL Server Integration Services, mit Hilfe eines Plugins für Dynamics CRM möglich.

## 13.2 Implementierung der optionalen Use Cases

Die als optional gekennzeichneten Use Cases könnten mit der aktuellen Architektur nachträglich gut umgesetzt werden. Da diese im bestehenden Kursanmeldungssystem auf der Website des INS auch nicht implementiert sind, könnte dies in einem zweiten Schritt erfolgen. Auch die Integration auf CRM Ebene muss sich in der Praxis erst beweisen und könnte danach im laufenden Betrieb angepasst werden. Durch die Solution-Kapselung und die Möglichkeiten zur Anpassung des CRM während dem Betrieb liessen dies zu.

## 13.3 Anpassungen an andere Mandanten

Durch die konsequente Verwendung von String- und Layoutressourcen, könnte das System mit verhältnismässig wenig Aufwand an andere Mandanten angepasst werden.

# 14 Installationsanleitung

## 14.1 Installation der CRM Komponenten

Als erstes muss die Basis der Lösung innerhalb des CRM geschaffen werden. Dazu kann die Solution „Trainingadministration“ importiert und veröffentlicht werden. Diese Solution beinhaltet lediglich Anpassungen, keine Daten.

Anschliessend müssen die Themenbereiche erfasst werden. Dazu werden die „Subjects“ innerhalb Dynamics CRM (Settings -> Business Management -> Subjects) verwendet. Diese können beliebig innerhalb des Themenbaums von Dynamics CRM angeordnet werden, sollten aber von einem einzelnen Vaterknoten aus organisiert sein.

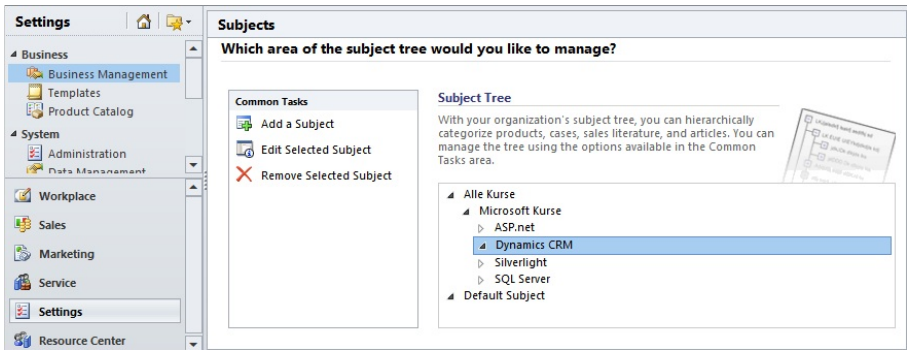



**Abbildung (14.1)** Themenbaum mit hierarchischer Struktur

Die ID (Guid) dieses Vaterknotens muss später in der `web.config` Datei des Tadm.Web Packages eingetragen werden. Die GUID lässt sich herausfinden, in dem der entsprechende Vaterknoten der Themenbereiche editiert wird:

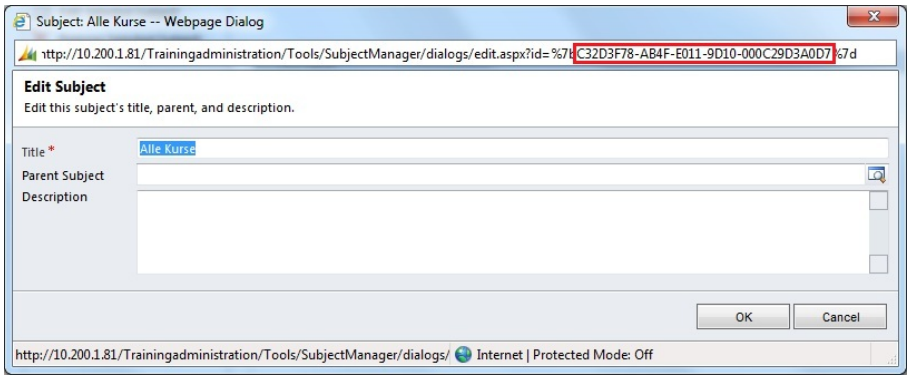



**Abbildung (14.2)** Subject-ID aus URL der Editiermaske

Anschliessend ist die Grundinstallation innerhalb des CRM komplett. Bevor jedoch Kurse und Durchführungen erfasst werden können sollten noch folgende Schritte erledigt werden:

**Gebäude und Räume** Um die Kursdurchführungen Gebäuden und Räumen zuteilen zu können, sollten solche bereits vorerfasst sein. Der Stamm an Gebäuden und Räumen sollte also um die Kursräumlichkeiten ergänzt werden.

**Services** Da die Kurstage als Service-Aktivitäten erfasst werden, bietet sich an sogenannte „Services“ vor zu erfassen. Ein Beispiel eines solch vorerfassten Service ist der Service „Ganzer Kurstag“ der bereits die Randzeiten vorerfasst hat. Dies beschleunigt die Erfassung der einzelnen Kurstage.

**Benutzer und Rollen** Zuletzt sollten die entsprechenden Benutzer noch der Rolle „Course Administration“ zugeteilt werden, damit die Berechtigungen auf den Formularen und Entitäten stimmen. Zusätzlich muss ein Systembenutzer erfasst werden, welcher nachher die Webkomponente repräsentiert. Auch dieser (Domain, Benutzername und Passwort) wird zur Konfiguration der Webkomponente benötigt.

## 14.2 Installation der Webkomponente

Die Webkomponente kann als Web-Applikation in einen IIS 7 Server mit .NET Framework 4 integriert werden. Damit die Kommunikation mit dem CRM funktioniert sollte der Rechner Zugriff auf die WCF Services des Dynamics CRM Servers haben (Organization Service).

Die Konfiguration der Webkomponente muss bei der Installation anschliessend noch auf den entsprechenden CRM Server angepasst werden:

```
<appSettings>

      <add key="CrmOrganizationServiceURI"
           value="http://10.200.1.81/Trainingadministration
                                    /XRMServices/2011/Organization.svc" />

      <add key="CrmDomain" value="s3cc" />
      <add key="CrmUsername" value="sgacond" />
      <add key="CrmPassword" value="*******" />

      <add key="RootCategory"
             value="c32d3f78-ab4f-e011-9d10-000c29d3a0d7"/>

      <add key="UseDummyData" value="false"/>

</appSettings>
```

**CrmOrganizationServiceURI** Die URI des OrganizationServices. Diese ist standardmässig die URI des Mandants + `/XRMServices/2011/Organization.svc`.

**CrmDomain / CrmUsername / CrmPassword** Die Identität des Service-Benutzers welche die Webkomponente nutzt um aufs CRM zuzugreifen.

**RootCategory** Vaterknoten im Kategoriebaum, unter dem die Kurskategorien zugeordnet sind.

**UseDummyData** Um losgelöst vom CRM zu arbeiten, kann die Webkomponente mit Dummydaten versorgt werden. Dies wird vor allem für Unit Tests verwendet. Sollte im normalen Betrieb immer auf `False` stehen oder gar nicht eingetragen sein.

Die Installation der entsprechenden Assemblies (inkl. die des CRM SDK) kann wahlweise im Global Assembly Cache oder in der Web-Applikation selbst erfolgen.

# 15 Schlussfolgerungen und Resultate

## 15.1 Erreichte Resultate

Das urpsrüngliche Ziel der Arbeit, die Ablösung des bestehenden Kursverwaltungssystems des Microsoft Innovation Center, kann als erreicht betrachtet werden. Das System ist noch nicht migriert und installiert, aber aus Anforderungssicht auf einem Stand der durchaus der Funktionalität des bestehenden Systems entspricht.

Neben den funktionellen Anforderungen ist die technische Sicht des Ergebnisses hervorzuheben. Die Architektur ist durchdacht, skalierbar und sauber getrennt. Mit wenigen Anpassungen lässt sich das System für andere Mandanten einrichten oder an ein anderes Backendsystem anschliessen. Ausserdem kann die Applikation an den entscheidenden Stellen automatisiert getestet werden.

Mit der aktuellen Integration in Dynamics CRM lassen sich administrative Abläufe im Backend zur Laufzeit anpassen oder zusätzlich automatisieren. Ausserdem kann die entwickelte Anbindung mit dem Connector-Interface, dem Caching und der Integration der WCF RIA Services als Referenz für andere Silverlight-Applikationen verwendet werden, die an das Dynamics CRM angeschlossen werden sollen.

## 15.2 Nicht oder nur teilweise erreichte Resultate

Einige der Ideen, welche in der Anforderungsanalyse entstanden, wurden nicht umgesetzt. In der entscheidenden Phase des Projekts wurde mit allen beteiligten Vereinbart, das Schwergewicht auf einen sauberen Abschluss der grundlegenden Use Cases zu legen. So blieb dafür genügend Zeit, die Architektur und die eigentlichen Schlüsselstellen derselben sauber zu dokumentieren.

## 15.3 Schlussfolgerungen

Die Arbeit hat gezeigt, dass Microsoft Dynamics CRM mit seinen Erweiterungsmöglichkeiten durchaus als Backend oder Datenhaltung für externe Applikationen eingesetzt werden kann. Trotzdem ist der Einsatz von Dynamics CRM als Backend nur sinnvoll, wenn es sich beim Anwendungsfall wirklich um ein Problem handelt, dass mit der Verwaltung von Kundenbeziehungen zusammenhängt. Nur so kann die eingebaute Funktionalität sinnvoll genutzt und der Mehrwert der eingebauten Dynamics CRM Funktionen verwendet werden.

## 15.4 Erfahrungsbericht Clemens Meier

### Projektverlauf

Durch die schon am Anfang des Projektes sehr klare Aufgabenstellung, konnten wir uns bereits in den ersten Wochen mit den zu verwendenden Technologien befassen. Es ergab sich dabei, dass Silvan sich in den ersten Wochen viel mehr in Microsoft Dynamics CRM einarbeitete, während ich mit WCF und WCF RIA Services beschäftigte.

Eigentlich hatten wir schon sehr schnell eine funktionierende Zwischenschicht, welche die Daten über den CRM OrganizationService laden und per WCF für die weitere Verwendung zur Verfügung stellen konnte. Doch auf die Möglichkeiten von RIA Services einen eigenen Clientkontext zu arbeiten und auf das automatische Nachladen von abhängigen Daten in Views wollten wir nicht verzichten. Die Verwendung von RIA Services mit dem CRM als Datenquelle wurde aber zu einer Herausforderung die uns etwas Zeit kostete. Vor allem die Umsetzung der Authentifikation war anspruchsvoll.

In der Zwischenzeit konnte Silvan sich tief in Dynamics CRM einarbeiten, wo ich etwas in einen Wissensrückstand geriet. In den weiteren Phasen des Projektes kümmerte ich mich deshalb mehr um die Umsetzung des Silverlight Web-Frontends und den RIA Service, während Silvan mehr für die CRM Angelegenheiten zuständig war. Trotz den technischen Herausforderungen konnten wir immer im Plan bleiben. Wir gerieten während des ganzen Projektes nie wirklich in Rückstand.

### Rückblick

Da ich mich mit Silvan sehr gut verstehe und weil wir uns gut ergänzen, verlief die Zusammenarbeit mit ihm immer optimal. Auch die gute Betreuung durch Prof. Hansjörg Huser und Jürg Jucker ermöglichte es uns, zielgerichtet und effizient zu arbeiten. Obwohl wir uns mit Technologien beschäftigten, die für uns beide neu waren, konnten wir schon nach wenigen Wochen einen funktionierenden Prototypen vorweisen.

Das schlussendlich entstandene Produkt ist funktionsfähig und erfüllt die wichtigsten Anforderungen. Hätten wir etwas weniger „Overengineering“ betrieben, wäre die Implementation von weiteren Features sicher möglich gewesen. Aber schlussendlich geht es ja bei einer Bachelorarbeit auch darum, neue Technologien zu erlernen.

## Gelerntes

Da ich im Vergleich zu anderen Technologien noch nicht sehr viel Erfahrung mit .NET hatte, konnte ich während des Projektes allgemein mein Wissen über .NET stark erweitern. Vor allem die konsequente Umsetzung von MVVM unter Verwendung von Prism und MEF und die ausführliche Beschäftigung mit WCF RIA Services brachten mir sehr viele neue Erkenntnisse. Auch durch die Beschäftigung mit Dynamics CRM konnte ich mir wertvolles Wissen aneignen.

Anstatt MEF hätten wir für die Dependency Injection auch das leichtgewichtigere Unity verwenden können. Die Modularisierung ist zwar schön, wäre aber für unser Projekt nicht zwingend nötig gewesen. Trotzdem sind dadurch keine Nachteile entstanden.

RIA Services zeigen ihre Stärken vor allem dann, wenn man sie soviel wie möglich direkt in der View verwendet. Dies konnten wir teilweise ausnutzen. Leider gestaltet sich das automatische Testen bei der Verwendung von RIA Services recht schwierig. Allerdings scheint RIA-Services auch noch etwas „in den Kinderschuhen zu stecken“. Die Dokumentation auf MSDN ist noch nicht sehr ausführlich. Ausserdem sollte es offizielle Möglichkeiten geben den Service für Tests zu mocken. Im Internet gibt es dazu bereits inoffizielle Lösungen, welche aber alle mehrere tausend Zeilen Code umfassen.

## Fazit

Nach der etwas harzigen Semesterarbeit mit nur wenigen Resultaten, war diese Bachelorarbeit sehr motivierend. Die Anforderungen an das Endprodukt waren schon zu Beginn sehr klar. Die Herausforderung bestand in der Aneignung des Know-Hows über die verwendeten Technologien.

Ich habe während dieser Arbeit sehr viel gelernt und bin stolz auf das entstandene Produkt.

## 15.5 Erfahrungsbericht Silvan Gacond

### Projektverlauf

Als das definitive Thema der Arbeit fest stand, war die Motivation von Anfang an sehr gross. Ich wusste dass wir mit Technologien arbeiten werden, die einerseits modern und spannend sind, andererseits aber auch in der Wirtschaft sehr gefragt sind. Bereits bei der letzten Arbeit mit Sharepoint mussten wir aber lernen das Technologien, die zwar als Wundermittel angepriesen werden, doch nicht immer die universelle Lösung aller Probleme sein können und das man sich sehr gut überlegen sollte, eine solch spezifische Lösung als Basis zu verwenden.

Einlesen in das CRM SDK war also angesagt, Silverlight und MVVM waren bereits zu Projektbeginn keine Fremdwörter, darum legte ich meinen Fokus zuerst in Richtung CRM Anpassung und Anbindung. Schnell wurde klar, dass diese Anwendung durchaus eine sinnvolle Erweiterung des Dynamics CRM ist und das die Anpassungsmöglichkeiten des CRM genügend flexibel für diesen Anwendungsfall sind.

Neben der Technologie waren die funktionalen Anforderungen sehr klar und einfach. Es gab ein abzulösendes System und der Anwendungsfall war nicht wirklich komplex, so konnte der Fokus auf die technologischen Herausforderungen gelegt werden. Die eigentliche Anforderungsanalyse hielt sich durchaus in Grenzen.

Wir begannen also ausgehend von Prototypen die Architektur der Applikation in allen Schichten aufzubauen. Obwohl wir beide unsere Kerngebiete hatten, fand ein guter Austausch statt und alle grundsätzlichen Entscheide wurden zusammen getroffen. Während diesem Aufbau standen wir manchmal vor dem Problem die richtige Lösung für unseren Anwendungsfall zu finden. Oft sind die eingesetzten Technologien für den kleinen Anwendungsfall nicht unbedingt nötig, Beispiele dafür sind die eingesetzten RIA Services oder die GUI Modularisierung mit MEF. Trotzdem haben wir uns entschieden diese Technologien einzusetzen, da sie stark dazu beitragen die Architektur sauber zu halten und so einen späteren Ausbau der Applikation sehr vereinfachen. Ausserdem kann die Applikation so als Beispiel für ähnliche Projekte (CRM mit Webanbindung) eingesetzt werden kann.

### Rückblick

Rückblickend bin ich mit dem Projektverlauf sehr zufrieden. Wir waren beide sehr motiviert und hatten beide neben der Arbeit keine Vorlesungen mehr zu besuchen, so hatten wir trotz Nebenjob genügend Zeit um das Projekt von Anfang an vorwärts zu treiben. Die gute Betreuung und die zahlreichen guten Inputs durch unseren Betreuer Hansjörg Huser sowie Jürg Jucker vom INS hat dazu beigetragen, dass wir sehr effizient vorwärts kamen

und den Aufwand am richtigen Ort einsetzen konnten. Die Zusammenarbeit mit Clemens verlief wie gewohnt sehr gut. Wir haben oft verschiedene Ansichten, Herangehensweisen und auch verschiedene Stärken. Dies führte dazu dass wir uns auch in dieser Arbeit sehr gut ergänzen konnten.

## Gelerntes

Wir haben in dieser Arbeit einiges Neuland beschritten. Die Kerntechnologie Dynamics CRM, dessen Oberfläche ich am ersten Tag der Arbeit das erste Mal gesehen habe, hat einiges an Einarbeitung gefordert. Auch den speziellen Einsatz der WCF RIA Services ohne Entity Framework sowie den Aufbau des GUI mit Prism und MEF waren neue Herausforderungen.

## Fazit

Ich bin mit dem Ergebnis der Arbeit sehr zufrieden. Obwohl man in der zur Verfügung stehenden Zeit wahrscheinlich mehr von den optionalen UseCases hätte umsetzen können, bin ich froh, dass wir uns dagegen entschieden haben. So hatten wir die Möglichkeit Dokumentation und Testing gründlich zu beenden. So steht das Projekt nun auf soliden Beinen, ist gut Dokumentiert und ausbaufähig.

# Literaturverzeichnis


[Bro10] Brown, Pete: *Silverlight 4 in action*, Manning Publications Co., Stamford (2010)

[Fow04a] Fowler, Martin: Inversion of Control Containers and the Dependency Injection pattern, Website (2004), http://martinfowler.com/articles/injection.html; letzter Zugriff: 14.6.2011.

[Fow04b] Fowler, Martin: Presentation Model, Website (2004), http://martinfowler.com/eaaDev/PresentationModel.html; letzter Zugriff: 14.6.2011.

[Fow06] Fowler, Martin: GUI Architectures, Website (2006), http://martinfowler.com/eaaDev/uiArchs.html; letzter Zugriff: 14.6.2011.

[FRH+96] F., Buschmann; R., Meunier; H., Rohnert; P., Sommerlad und M., Stal: *Pattern Oriented Software Architecture, Volume 1*, Wiley (1996)

[Mic10] Microsoft: Developer's Guide to Microsoft Prism (2010), http://msdn.microsoft.com/en-us/library/gg406140.aspx; letzter Zugriff: 14.6.2011.

[Mic11] Microsoft: Learn About Development for Microsoft Dynamics CRM (2011), http://msdn.microsoft.com/en-us/library/gg334635.aspx; letzter Zugriff: 14.6.2011.

[MICR11] Microsoft Innovation Center Rapperswil, Institut fi¿½r vernetzte Systeme: Website (2011), http://ins.hsr.ch; letzter Zugriff: 14.6.2011.

[PE05] Peter Eeles, IBM, Senior IT Architect: Capturing Architectural Requirements, Website (2005), http://www.ibm.com/developerworks/rational/library/4706.html; letzter Zugriff: 6.6.2011.

# Glossar

| | |
|---|---|
| **ASP.NET** | 'Active Server Pages .NET', Microsoft Technologie von Webseiten mit serverbasierter Logik. |
| **Assembly** | Klassenbibliothek (meistens .dll), beinhaltet .NET Common Language Runtime (CLR) Zwischencode (Intermediate Language, IL). |
| | |
| **CRM** | 'Customer Relationship Management', System zur Verwaltung von Kundenbeziehungen. |
| | |
| **Dependency Injection** | Design-Pattern welches das Prinzip des umgekehrten Kontrollflusses anwendet (Inversion of Control). |
| **Dynamics CRM** | CRM Produkt von Microsoft. |
| | |
| **Entity Framework** | Microsoft Technologie zur Kommunikation mit einer Datenbank (OR-Mapper). |
| | |
| **MEF** | 'Managed Extensibility Framework', Microsoft Framework um Module in GUI Applikationen zur Laufzeit zu laden. |
| **Microsoft Dynamics** | Enterprise Produktlinie von Microsoft. |
| **MSDN** | 'Microsoft Developer Network', Resource im Internet mit Refernzen, Best Practices und Beispielen. |
| **MVC** | Model-View-Controller Design Pattern. |
| **MVP** | Model-View-Presenter Design Pattern. |
| **MVVM** | Model-View-Viewmodel Design Pattern. |
| | |
| **Plugin** | Zusätzliche Komponente für ein bestehendes System, welche geladen und aktiviert werden kann. |
| **Prism** | Sammlung von Best Practices und Klassenbibliothek von Microsoft für GUI Applikationen. |
| | |
| **Scrum** | Verbreitete Projektmanagement Methode, meistens für Software Projekte benutzt. |
| **Silverlight** | Microsoft Technologie um erweiterte GUI Applikationen im Browser zu ermöchlichen. |
| | |
| **WCF RIA Services** | 'Windows Communication Foundation, Rich Internat Application Services', Technologie zur Vereinfachung der Kommunikation mit einem Web Service. |

| | |
|---|---|
| **WPF** | 'Windows Presentation Foundation', Microsoft Technologie für Windows-Benutzeroberflächen. |
| **WSDL** | 'Web Service Description Language', Beschreibungssprache für Web Services. |

# Abbildungsverzeichnis

# Tabellenverzeichnis

# Anhang A

# Aufgabenstellung

# Kursadministration mit Silverlight

## Aufgabenstellung für Clemens Meier und Silvan Gacond

### Einführung

Das INS und das MIC bieten Workshops und Kurse an. Für die Ausschreibungs-, Anmelde- und Verwaltunsgprozesse wird seit einigen Jahren eine eigenentwickelte Webapplikation eingesetzt. Diese Applikation hat funktionelle Mängel und ist auch bezügliche Benutzeroberfläche nicht mehr zeitgemäss und zudem wartungsintensiv.
Das Ziel dieser Arbeit ist die Neuentwicklung einer verbesserten Applikation basierend auf einem Silverlight-Client und einem Backend basierend auf MS-CRM.

### Aufgabenstellung

- Analyse der Anforderungen
  - Analyse des bestehenden Systems
  - Erfassen der erweiterten Anforderungen
  - Resultat: Use Case-Spezifikation, Domain-Modell
- Definition der Systemarchitektur
  - Silverlight-Client
  - Service-Schnittstelle zum Backend
  - Backend mit CRM und evtl SQL-Server
- Design und Implementation des Clients
  - UI-Design
  - Internes Design: MVVM; evtl basierend auf Prism oder einem anderen Framework
  - Implementation eines Kernsystems für das Backend
- Systemtests

# Resultate

- Analyse-und Design-Dokumente
- Ausführbare Software mit Dokumentation

## Auftraggeber

INS
Ansprechpartner: Jürg Jucker (CRM, Anforderungen), Peter Nedic (Anforderungen), Andrea Moschin (Anforderungen), Marc Müller (Architektur)

## Projektteam

Clemens Meier (cmeier@hsr.ch)
Silvan Gacond (sgacond@hsr.ch)

## Betreuung HSR

Hansjörg Huser, hhuser@hsr.ch, Tel: 055 222 49 12 (HSR Raum 6.010)

# Projektabwicklung

## Termine:

- Beginn der Arbeit: M**o., 21. Feb. 2011**

- Abgabetermin Kurzfassung/Poster/Mgmt-Summary zum Review: 10.06.2011
- Abgabetermin (ink. Poster): 17.06.2011, 12.00 Uhr
- **HSR-Forum, Vorträge und Präsentation der Bachelor- und Diplomarbeiten, 17.6.2011 nachmittags**
- Mündliche BA-Prüfung : genaues Datum folgt (8.8.-27.8.)
- Zwischenbesprechung/Review mit Auftraggeber nach Projektplan

## Arbeitsaufwand

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

### Hinweise für die Gliederung und Abwicklung des Projektes:

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

Folgende Teilschritte bzw. Meilensteine sollten Sie in Ihrer Planung vorsehen:

Schritt 1: Projektauftrag inkl. Projektplan (mit Meilensteinen),

- Meilenstein 1: Review des Projektauftrages abgeschlossen. Projektauftrag von Auftraggeber und Dozent genehmigt
  Letzter Meilenstein: Systemtest abgeschlossen
  Termin: ca. eine Woche vor Abgabe
- Entwickeln Sie Ihre SW in einem iterativen, inkrementellen Prozess: Planen Sie möglichst früh einen ersten lauffähigen Prototypen mit den wichtigsten und kritischsten Kernfunktionen. In die folgenden Phasen können Sie dieses Kernsystem schrittweise ausbauen und testen.
- Falls Sie in Ihrer Arbeit neue oder Ihnen unbekannte Technologien einsetzen, sollten Sie parallel zum Erarbeiten des Projektauftrages mit dem Technologiestudium beginnen.
- Setzen Sie konsequent Unit-Tests ein! Verwalten Sie ihre Software und Dokumente auf einem SVN-Repository. Stellen Sie sicher, dass der/die Betreuer jederzeit Zugriff auf das Repository haben und dass das Projekt anhand des Repositories jederzeit wiederhergestellt werden kann.
- Achten Sie auf die Einhaltung guter Programmier- und Designprinzipien
  - DRY, high cohesion, loose coupling, etc.
  - Clean Code!
- Halten Sie sich im Übrigen an die Vorgaben aus dem Modul SE-Projekt.

Projektadministration

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

### Inhalt der Dokumentation

Bei der Abgabe muss jede Arbeit folgende Inhalte haben:

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

- Projekttagebuch, Dokumentation des Projektverlaufes, Planung etc.

## Fortschrittsbesprechung:

Regelmässig findet zu einem fixen Zeitpunkt eine Fortschrittsbesprechung statt.
Teilnehmer: Dozent und Studenten, bei Bedarf auch Vertreter der Auftraggeber
Termin: jeweils Montag, 14h, Raum 6.010 (Abweichungen werden rechtzeitig kommuniziert)
Traktanden

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

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

Rapperswil, 21. Feb. 2011
Hansjörg Huser

Anhang: Notizen zu den Anforderungen

Backend
Als Basis für das Backend (inkl. Datenhaltung) sehe ich das CRM, da diese viele Grundfunktionen unterstützt wie Newsletter erstellen und verschicken, , Marketinglisten erstellen, Notizen zu telefonischen Anfragen hinterlegen. Mit der Workflow-Engine können automatisierte Prozesse wie E-Mail Versand für die Bestätigung der Kursdurchführung einfach implementiert werden.
Das CRM unterstützt zudem Mandantenfähigkeit (pro Mandant eine CRM-Instanz).

Primär aus Security-Gründen darf die Web-Applikation für die Kursteilnehmer nicht direkt auf das CRM zugreifen, sondern nur über eine vorgeschaltete Service-Schnittstelle. Dies erlabut zudem den Ersatz der CRM-Funktionalität mit einer Eigenentwicklung,

Hier die Requirements Ergänzungen zum bestehenden Tool:

- Kurse
  - Einführung von Kurskategorien (Entwicklung, Datenbanken, Programmierung etc.), entspricht im CRM den Marketinglisten, damit kann jeder Teilnehmer nach dem Besuch resp. der Anmeldung gleich der Marketingliste hinzugefügt werden. Für den späteren Versand der Newsletter
  - Kursbeschreibung (Details etc.) muss innerhalb vom Tool definiert werden können
  - Kriterien resp. Themen für Zufriedenheitsumfrage
  - Checkliste Vorbereitung (Unterlagen etc.)
- Kursdurchführung (siehe auch Details z.B. Migros Klubschule)
  - Zeitraum des Kurses z.B. 27.08.2010 – 31.01.2011
  - Kursdaten eff. Daten an dem der Kurs stattfindet z.B. beim Lehrgang
  - Anzahl der Teilnehmer max. Platzanzahl des Schulungsraumes (Ressourcen, Facility im CRM) z.B. 1.213 = 18 Teilnehmer max. Teilnehmeranzahl für das Web.-Tool (z.B. 16 Teilnehmer)
  - Ablage der Unterlagen für Druck resp. Download durch den Trainer (sofern Intern)
- Verwaltung von Events (Kurse welchen nur eine Durchführung haben)
- Workflows / Prozesse
  - Anmeldung auch ohne Registrierung (Account-Erstellung )
  - Anmeldung nach Warenkorb-Prinzip
  - Abmeldung für registrierte Teilnehmer
  - Vereinfachte Anmeldung für Events (z.B. auch für MIC Events / Stadt Rapperswil) bedingt ohne Registrierung
  - Abmelde Formular für nicht registrierte Teilnehmer generiert einen Service-Request im CRM (mit entsprechender Benachteiligung des Sekretariats)
  - autom. Benachrichtigung dass ein Kurs für einen Trainer eingetragen ist (Termin und E-Mail)
  - je nach Kurs/Event müssen andere E-Mail Templates (Workflow) definiert werden können (CRM Standard sollte da reichen)
  - Je nach Kurs/Event sollten auch zusätzliche Felder ausgefüllt werden müssen (z.B. Fahrzeugangaben)
- Dokumente
  - Kursbestätigungen
  - Download (Upload) der elektr. Kursunterlagen
  - Rechnungsgenerierung (Excel-Liste und allenfalls auch SSRS-Dokument)
  - Zufriedenheitsumfrage Teilnehmer

Hier noch sep. Requirements and Zusatztools:

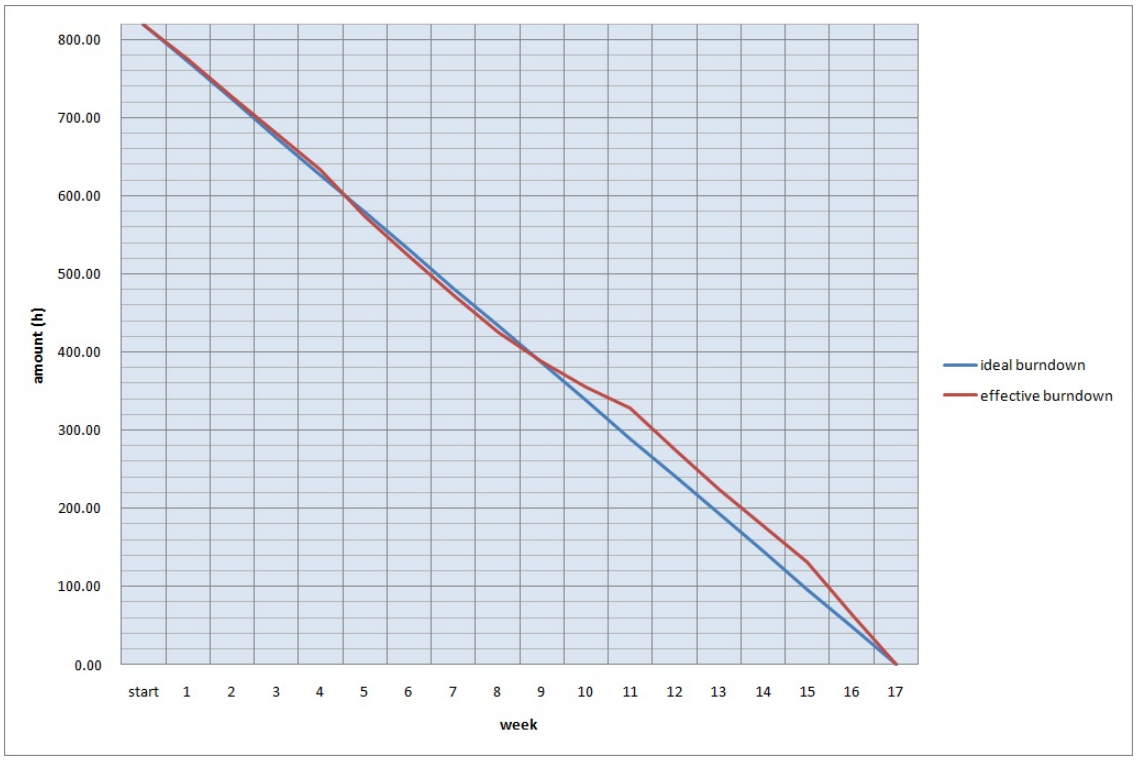
- Newsletter Abonierung

    - Abmeldung über E-Mail Adresse
    - Anmeldung mit vollständiger Andresse inkl. E-Mail
- Zufriedenheitsumfrage
- Buchungstool (wie z.B. Event-in-a-Box von Microsoft)
- Wizard-Tool für die Erstellung von Anmeldungen oder Service-Requests (CRM) mit dynamischen Felder.

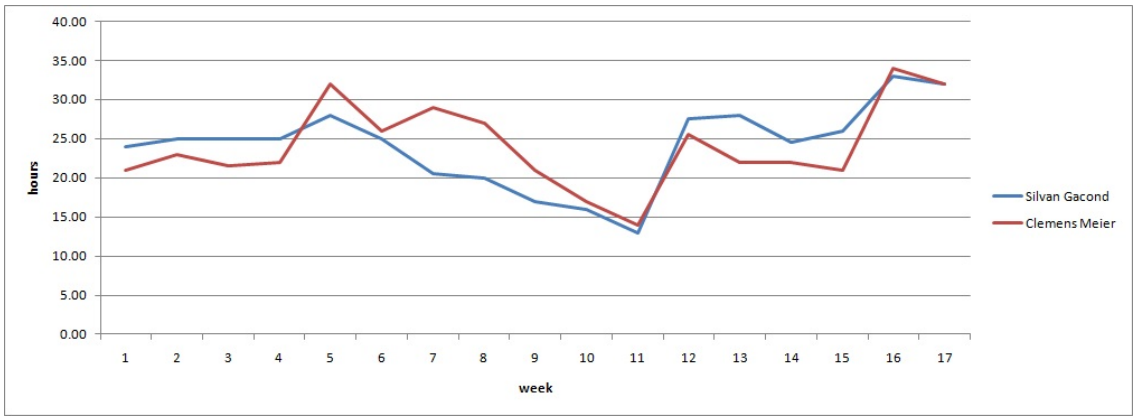
# Anhang B

# Projektmanagement

## Project Burndown Chart



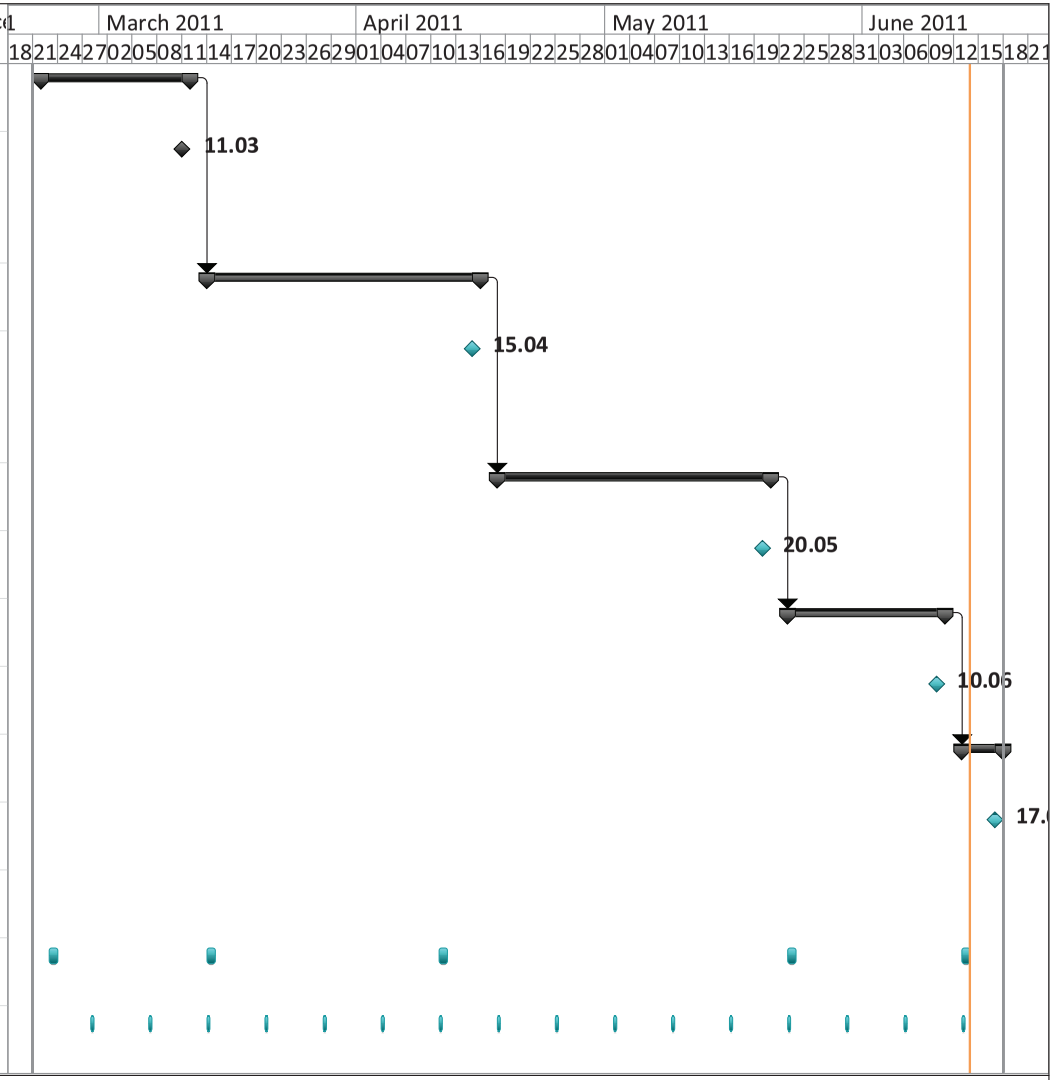
# Übersicht Arbeitszeit pro Woche




Genaue Zeiterfassungen siehe *Zeiterfassungen* (auf Seite 147) .

| ID | Task Mod | Task Name | Duration | Start | Finish | Predec |
|---|---|---|---|---|---|---|
| 1 | | **Sprint 0** | **14 days** | **Tue 22.02.11** | **Fri 11.03.11** | |
| 2 | | Anforderungen für Sprint 1 klar | 0 days | Fri 11.03.11 | Fri 11.03.11 | |
| 3 | | **Sprint 1** | **5 wks** | **Mon 14.03.11** | **Fri 15.04.11** | **1** |
| 4 | | Erster Prototyp durch alle Layers | 0 days | Fri 15.04.11 | Fri 15.04.11 | |
| 5 | | **Sprint 2** | **5 wks** | **Mon 18.04.11** | **Fri 20.05.11** | **3** |
| 6 | | volle Funktionalität | 0 days | Fri 20.05.11 | Fri 20.05.11 | |
| 7 | | **Sprint 3** | **3 wks** | **Mon 23.05.11** | **Fri 10.06.11** | **5** |
| 8 | | Test abgeschlossen | 0 days | Fri 10.06.11 | Fri 10.06.11 | |
| 9 | | **Sprint 4** | **5 days** | **Mon 13.06.11** | **Fri 17.06.11** | **7** |
| 10 | | Abgabe | 0 days | Fri 17.06.11 | Fri 17.06.11 | |
| 11 | | | | | | |
| 12 | | **Projektplan** | **79 days** | **Wed 23.02.11** | **Mon 13.06.11** | |
| 18 | | **Sitzung** | **76 days** | **Mon 28.02.11** | **Mon 13.06.11** | |




Project: ProjektplanBA
Date: Tue 14.06.11

| | | | | | |
|---|---|---|---|---|---|
| Task | | External Milestone | | Manual Summary Rollup | |
| Split | | Inactive Task | | Manual Summary | |
| Milestone | | Inactive Milestone | | Start-only | |
| Summary | | Inactive Summary | | Finish-only | |
| Project Summary | | Manual Task | | Deadline | |
| External Tasks | | Duration-only | | Progress | |

| ID | Iteration Path | Work Item ID | Title | Duration | Start | Finish |
|---|---|---|---|---|---|---|
| 1 | \Sprint 0 | 1 | Kickoff Meeting und erste Ideen dazu | 1 day | Tue 22.02.11 | Tue 22.02.11 |
| 2 | \Sprint 0 | 11 | Aufsetzen Latex Template für Doku | 4 hrs | Tue 22.02.11 | Tue 22.02.11 |
| 3 | \Sprint 0 | 9 | Grobplanung Sprints und Setup WorkItems | 4 hrs | Thu 24.02.11 | Thu 24.02.11 |
| 4 | \Sprint 0 | 3 | Anforderungen v1 | 5 days | Thu 24.02.11 | Wed 02.03.11 |
| 5 | \Sprint 0 | 4 | Szenarios / UseCases v1 | 5 days | Thu 03.03.11 | Wed 09.03.11 |
| 6 | \Sprint 0 | 5 | Workflows v1 spezifizieren | 1 day | Wed 09.03.11 | Wed 09.03.11 |
| 7 | \Sprint 0 | 6 | Analyse / Ausprobieren Möglichkeiten in Dynamics CRM für ersten S | 2.5 days | Fri 25.02.11 | Tue 01.03.11 |
| 8 | \Sprint 0 | 7 | Analyse Funktionalität Dynamics oder Standalone | 1 day | Fri 11.03.11 | Fri 11.03.11 |
| 9 | \Sprint 0 | 8 | Anforderungen für Sprint 1 klar | 0 days | Mon 14.03.11 | Mon 14.03.11 |
| 36 | \Sprint 0 | 12 | Zwischenschicht: Solution Vorbereiten | 0.5 days | Mon 28.02.11 | Mon 28.02.11 |
| 37 | \Sprint 0 | 10 | Einfacher Technologieprototyp für Verbindung SL / CRM | 8 days | Wed 02.03.11 | Fri 11.03.11 |
| 38 | \Sprint 0 | 16 | Basis-Datenmodell und passende Forms im CRM | 3 days? | Wed 02.03.11 | Fri 04.03.11 |
| 39 | \Sprint 0 | 13 | Solution Tadm.Web auf allen Schichten aufsetzen | 2 days? | Wed 02.03.11 | Mon 07.03.11 |
| 40 | \Sprint 0 | 15 | Service Interface Zwischenschicht v1 definieren | 2 days? | Tue 08.03.11 | Wed 09.03.11 |
| 41 | \Sprint 0 | 17 | Evaluation Architektur für Silverlight | 2.5 days? | Tue 08.03.11 | Thu 10.03.11 |
| 42 | \Sprint 0 | 18 | Einlesen Prism / MEF | 3 days? | Wed 09.03.11 | Fri 11.03.11 |
| 43 | \Sprint 0 | 19 | Aufbau Architektur Silverlight (Prism / MEF) | 6 days? | Wed 09.03.11 | Wed 16.03.11 |
| 10 | **\Sprint 1** | **20** | **Implementation UseCases v1: Grundfunktionalität gemäss aktuelle** | **35 days** | **Mon 14.03.11** | **Fri 29.04.11** |
| 11 | \Sprint 1 | 14 | CrmConnector und MockConnector für ersten einfachen Prototyp | 3 days | Mon 14.03.11 | Wed 16.03.11 |
| 12 | \Sprint 1 | 24 | Testdaten ins CRM | 1 day | Mon 14.03.11 | Mon 14.03.11 |
| 13 | \Sprint 1 | 25 | CRM RichText Editor für Kurse | 1 day | Tue 15.03.11 | Tue 15.03.11 |
| 14 | \Sprint 1 | 21 | Kursübersicht v1 (einfaches UI) | 10 days | Mon 14.03.11 | Fri 25.03.11 |
| 15 | \Sprint 1 | 22 | Kursdetail v1 (einfaches UI) | 8 days | Wed 16.03.11 | Fri 25.03.11 |
| 16 | \Sprint 1 | 23 | Connector Refresh via Timer, Propagation via Session Cache | 0.5 days | Mon 21.03.11 | Mon 21.03.11 |
| 17 | \Sprint 1 | 26 | Login / Membership Provider (Authentifizierung via CRM) | 2 days | Mon 28.03.11 | Tue 29.03.11 |
| 18 | \Sprint 1 | 27 | Registrierung bis ins CRM (einfaches UI) | 1.5 days | Wed 30.03.11 | Thu 31.03.11 |
| 19 | \Sprint 1 | 28 | Ablaufdiagramm für Registrierungsvorgang | 0.5 days | Fri 01.04.11 | Fri 01.04.11 |
| 20 | \Sprint 1 | 29 | Registrierungsformular User v1 | 1 day | Mon 04.04.11 | Mon 04.04.11 |
| 21 | \Sprint 1 | 30 | CRM Workflow für E-Mail Bestätigung (Kursregistrierung) | 1 day | Mon 04.04.11 | Mon 04.04.11 |
| 22 | \Sprint 1 | 31 | CRM Plugin für E-Mail Duplikat-Detection | 1 day | Tue 05.04.11 | Tue 05.04.11 |
| 23 | \Sprint 1 | 32 | Dialog "Passwort vergessen?" bei Versuch Anm. mit gleicher E-Ma | 0.5 days | Tue 05.04.11 | Tue 05.04.11 |
| 24 | \Sprint 1 | 33 | Refactoring Membership Provider (sinnvollere Lösung RIA Service | 1 day | Wed 06.04.11 | Wed 06.04.11 |
| 25 | \Sprint 1 | 40 | Dokumentation Architektur Schichten / Technologien / Übersicht | 2 days | Fri 08.04.11 | Mon 11.04.11 |
| 26 | \Sprint 1 | 34 | Dokumentation Architektur Ria-Services | 3 days | Tue 12.04.11 | Thu 14.04.11 |
| 27 | \Sprint 1 | 35 | Dokumentation Architektur Prism / MVVM / Dependency Injectio | 3 days | Thu 14.04.11 | Sat 16.04.11 |
| 28 | \Sprint 1 | 36 | Dokumentation CRM Komponenten | 3 days | Sat 16.04.11 | Tue 19.04.11 |
| 29 | \Sprint 1 | 37 | Dokumentation CRM Plugin / Workflow Business Logik | 3 days | Wed 20.04.11 | Fri 22.04.11 |
| 30 | \Sprint 1 | 38 | Vorbereiten Zwischenpräsentation | 3 days | Mon 25.04.11 | Wed 27.04.11 |
| 31 | \Sprint 1 | 39 | Aufräumen Solution für sauberen Stand nach Sprint 1 | 2 days | Wed 27.04.11 | Thu 28.04.11 |
| 32 | **\Sprint 2** | **41** | **Kursliste und bessere Kurs-Details auf UI** | **15 days** | **Mon 02.05.11** | **Fri 20.05.11** |
| 33 | \Sprint 2 | 42 | Kursliste implementieren | 5 days | Mon 02.05.11 | Fri 06.05.11 |
| 34 | \Sprint 2 | 51 | Kursdetailview verbessern | 4 days | Fri 06.05.11 | Wed 11.05.11 |
| 35 | \Sprint 2 | 43 | Verbessern des Silverlight UI | 8 days | Wed 11.05.11 | Fri 20.05.11 |
| 44 | **\Sprint 2** | **45** | **Crm Funktionalität automatisieren** | **15 days** | **Mon 02.05.11** | **Fri 20.05.11** |
| 45 | \Sprint 2 | 46 | Plugin SetCourseRunState | 2 days | Fri 06.05.11 | Mon 09.05.11 |
| 46 | \Sprint 2 | 47 | Plugin SetCourseState | 2 days | Mon 09.05.11 | Tue 10.05.11 |
| 47 | \Sprint 2 | 48 | CRM Ribbon Buttons um CourseRun Status schieben zu können | 2 days | Mon 09.05.11 | Tue 10.05.11 |
| 48 | \Sprint 2 | 49 | Unit Testing für CRM Plugin Funktionalität | 4 days | Wed 11.05.11 | Mon 16.05.11 |
| 49 | \Sprint 2 | 50 | Dokumentation nachziehen (CRM Funktionalität) | 4 days | Tue 17.05.11 | Fri 20.05.11 |
| 50 | **\Sprint 3** | **54** | **Testing / Dokumentation** | **10 days** | **Mon 23.05.11** | **Fri 03.06.11** |
| 51 | \Sprint 3 | 44 | Unit Testing für Web-Komponente | 5 days | Mon 23.05.11 | Fri 27.05.11 |
| 52 | \Sprint 3 | 55 | Dokumentation Technologien und Konzepte | 10 days | Mon 23.05.11 | Fri 03.06.11 |
| 53 | \Sprint 3 | 56 | Dokumentation Umsetzung | 3 days | Wed 01.06.11 | Fri 03.06.11 |
| 54 | **\Sprint 4** | **63** | **Abschlussarbeiten** | **10 days** | **Mon 06.06.11** | **Fri 17.06.11** |
| 55 | \Sprint 4 | 64 | Dokumentation Umsetzung fertig | 5 days | Mon 06.06.11 | Fri 10.06.11 |
| 56 | \Sprint 4 | 57 | Manuelle Tests (Schreiben & Durchführen) | 2 days | Mon 06.06.11 | Tue 07.06.11 |
| 57 | \Sprint 4 | 58 | Abstract / Management Summary | 2 days | Thu 09.06.11 | Fri 10.06.11 |
| 58 | \Sprint 4 | 59 | Poster | 1 day | Fri 10.06.11 | Fri 10.06.11 |
| 59 | \Sprint 4 | 60 | Kleinere Bugfixes | 2 days | Fri 10.06.11 | Mon 13.06.11 |
| 60 | \Sprint 4 | 61 | Projektmanagement Charts und Doku | 3 days | Tue 14.06.11 | Thu 16.06.11 |
| 61 | \Sprint 4 | 62 | Dokumentation korrigieren / verfollständigen | 5 days | Mon 13.06.11 | Fri 17.06.11 |


Project: ProjektplanBA
Date: Tue 14.06.11

Task | Milestone | Project Summary | External Milestone | Inactive Milestone | Manual Task | Manual Summary Rollup | Start-only | Deadline
Split | Summary | External Tasks | Inactive Task | Inactive Summary | Duration-only | Manual Summary | Finish-only | Progress

# Anhang C

# Zeiterfassungen

# Zeiterfassung

**Projekt:** **Trainingsadministration**
**Teammitglied:** **Clemens Meier**

**Durchschnittliche Soll-Arbeitszeit:** **21.2**
**(gerechnet mit 17 Wochen, Feiertage nicht miteinbezogen)**

| Woche | Tag | Datum | Tätigkeit | Zeit | Zeit/Wo | Total | Soll |
|---|---|---|---|---|---|---|---|
| Woche 1 | Di | 22. 2. 11 | Startsitzung H. Huser | 1.0 | | | |
| | | | Einrichtung Entwicklungsumgebung | 3.0 | | | |
| | | | Technologiestudium | 1.0 | | | |
| | | | Studium Aufgabenstellung | 1.0 | | | |
| | Mi | 23. 2. 11 | Domainanalyse | 2.0 | | | |
| | | | Einrichten Entwicklungsumgebung | 2.0 | | | |
| | | | Dokumentevorlagen | 2.0 | | | |
| | | | Projektplanung | 2.0 | | | |
| | Fr | 25. 2. 11 | Vorbereitung Sitzung (Anforderungen) | 2.0 | | | |
| | | | Sitzung J.Jucker | 1.5 | | | |
| | | | Anforderungen | 3.0 | | | |
| | | | Use Cases | 0.5 | | | |
| | | | | | **21.0** | **21.0** | 21.2 |
| Woche 2 | Mo | 28. 2. 11 | Anforderungen + Domainanalyse | 4.0 | | | |
| | | | Umsetzung Domain (DTO's) | 2.0 | | | |
| | | | Vorbereitung Sitzung | 1.0 | | | |
| | | | Sitzung H. Huser | 1.0 | | | |
| | Di | 01. 3. 11 | Technologiestudium WCF Service | 5.0 | | | |
| | | | Anforderungen | 3.0 | | | |
| | Mi | 02. 3. 11 | Implementation + Konfiguration WCF Service | 4.0 | | | |
| | | | Technologiestudium | 3.0 | | | |
| | | | | | **23.0** | **44.0** | 42.4 |
| Woche 3 | Mo | 07. 3. 11 | WCF Service + CRM Connector Prototyp | 7.0 | | | |
| | Di | 08. 3. 11 | WCF Service + CRM Connector Prototyp | 5.0 | | | |
| | Mi | 09. 3. 11 | WCF Service + CRM Connector Prototyp | 5.0 | | | |
| | Fr | 11. 3. 11 | Grafiken Service WCF+RIA | 2.0 | | | |
| | | | Vorbereitung Sitzung | 1.0 | | | |
| | | | Sitzung H. Huser / J. Jucker | 1.5 | | | |
| | | | | | **21.5** | **65.5** | 63.6 |
| Woche 4 | Mo | 14. 3. 11 | WCF RIA Service | 3.0 | | | |
| | | | Technologiestudium RIA-Services | 3.0 | | | |
| | | | Definition Interface für Service | 2.0 | | | |
| | Di | 15. 3. 11 | WCF RIA Service + CRM Connector | 7.0 | | | |
| | Mi | 16. 3. 11 | WCF RIA Service | 7.0 | | | |
| | | | | | **22.0** | **87.5** | 84.8 |
| Woche 5 | Mo | 21. 3. 11 | Austauschbarer DomainConnector | 4 | | | |
| | | | ConnectorMock | 4 | | | |
| | Di | 22. 3. 11 | Einlesen Prism/ MEF | 5.0 | | | |
| | | | ConnectorMock | 4.0 | | | |
| | Mi | 23. 3. 11 | RIA DomainService | 8.0 | | | |
| | Fr | 25. 3. 11 | Sitzung | 1.0 | | | |
| | | | Loginformular + Authentication | 6.0 | | | |
| | | | | | **32.0** | **119.5** | 106.0 |
| Woche 6 | Mo | 28. 3. 11 | Authentication, einlesen ausprobieren | 7.0 | | | |
| | Di | 29. 3. 11 | RIA AuthenticationService | 6.0 | | | |
| | Mi | 30. 3. 11 | Login / Membership Provider | 7.0 | | | |
| | Fr | 01. 4. 11 | Authentication RIA Services | 5.0 | | | |
| | | | Sitzung | 1.0 | | | |
| | | | | | **26.0** | **145.5** | 127.2 |
| Woche 7 | Mo | 04. 4. 11 | Login / Membership Provider / Alternativen? | 9.0 | | | |
| | Di | 05. 4. 11 | Login / Membership Provider | 6.0 | | | |
| | Mi | 06. 4. 11 | Login / Membership Provider Verbesserungen | 7.0 | | | |
| | Fr | 08. 4. 11 | Ablauf Registrierug im Webfrontend | 6.0 | | | |
| | | | Sitzung (H.Huser und J.Jucker) | 1.0 | | | |
| | | | | | **29.0** | **174.5** | 148.4 |
| e 8 | Mo | 11. 4. 11 | Behaviors, Technologiestudium + Implementation | 7.0 | | | |
| | Di | 12. 4. 11 | Notification+Confirmation in Silverlight | 6.0 | | | |
| | Mi | 13. 4. 11 | Refactoring View Models | 4.0 | | | |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Woch | | | Notification+Confirmation in Silverlight | 3.0 | | | |
| | Fr | 15. 4. 11 | CRMException Handling Silverlight (Duplicated Email) | 3.0 | | | |
| | | | Kursansicht | 4.0 | | | |
| | | | | | **27.0** | **201.5** | 169.6 |
| Woche 9 | Mo | 18. 4. 11 | View Switching Navigation | 7.0 | | | |
| | Di | 19. 4. 11 | View Switching Navigation | 7.0 | | | |
| | Mi | 20. 4. 11 | Dokumentation Techinischer Bericht | 5.0 | | | |
| | | | Dokumentation Review | 2.0 | | | |
| | | | | | **21.0** | **222.5** | 190.8 |
| Woche 10 | Di | 26. 4. 11 | Vorbereitung Zwischenpräsentation | 6.0 | | | |
| | Mi | 27. 4. 11 | Vorbereitung Zwischenpräsentation | 4.0 | | | |
| | | | Zwischenpräsentation | 2.0 | | | |
| | Fr. | 29. 4. 11 | UML-Diagramme für Dokumentation | 3.0 | | | |
| | | | Abschlussarbeiten Spring | 2.0 | | | |
| | | | | | **17.0** | **239.5** | 212.0 |
| Woche 11 | Mi | 04. 5. 11 | View Switching Navigation | 7.0 | | | |
| | Fr | 06. 5. 11 | Dokumentation Navigation | 7.0 | | | |
| | | | | | **14.0** | **253.5** | 233.2 |
| Woche 12 | Mo | 09. 5. 11 | Besprechung Zwischenstand | 0.5 | | | |
| | | | ValidationObjects | 6.0 | | | |
| | Di | 10. 5. 11 | Dokumentation | 6.0 | | | |
| | Mi | 11. 5. 11 | Registration Silverlight | 7.0 | | | |
| | Fr | 13. 5. 11 | Studium / Dokumentation GUI Patterns | 6.0 | | | |
| | | | | | **25.5** | **279.0** | 254.4 |
| Woche 13 | Mo | 16. 5. 11 | Verbesserung Registration Silverlight UI | 6.0 | | | |
| | Di | 17. 5. 11 | Sitzung mit J.Jucker | 1.0 | | | |
| | | | Dokumentation Konzepte und Technologien | 6.0 | | | |
| | Mi | 18. 5. 11 | Dokumentation RIA Services | 6.0 | | | |
| | Fr | 20. 5. 11 | Bug Fixing | 3.0 | | | |
| | | | | | **22.0** | **301.0** | 275.6 |
| Woche 14 | Mo | 23. 5. 11 | Dokumentation GUI Patterns | 7.0 | | | |
| | Di | 24. 5. 11 | Dokumentation MEF / Prism / Dependency Injection | 6.0 | | | |
| | Mi | 25. 5. 11 | Unit Tests Silverlight/View Model | 2.0 | | | |
| | | | Dokumentation Technologien und Konzepte | 4.0 | | | |
| | Fr | 27. 5. 11 | String Resources überall einbauen | 3.0 | | | |
| | | | | | **22.0** | **323.0** | 296.8 |
| Woche 15 | Di | 31. 5. 11 | Dokumentation | 7.0 | | | |
| | Mi | 01. 6. 11 | Dokumentation (Überarbeitung) | 6.0 | | | |
| | Do | 02. 6. 11 | Dokumentation Umsetzung Silverlight | 4.0 | | | |
| | | | Dokumentation Technologie RIA Service | 4.0 | | | |
| | | | | | **21.0** | **344.0** | 318.0 |
| Woche 16 | Mo | 06. 6. 11 | CrmConnectorMock Unit Tests Silverlight | 6.0 | | | |
| | Di | 07. 6. 11 | Unit Tests Silverlight/View Model | 6.0 | | | |
| | Mi | 08. 6. 11 | Dokumentation Umsetzung (Serviceschicht) | 7.0 | | | |
| | Do | 09. 6. 11 | Dokumentation Tests (Silverlight) | 6.0 | | | |
| | | | Review Abstract | 2.0 | | | |
| | Fr | 10. 6. 11 | Dokumentation fertigstellen | 7.0 | | | |
| | | | | | **34.0** | **378.0** | 339.2 |
| Woche 17 | Mo | 13. 6. 11 | Dokumentation fertigstellen | 1.0 | | | |
| | | | Review Dokumentation | 2.0 | | | |
| | | | Korrektur Dokumentation | 5.0 | | | |
| | Di | 14. 6. 11 | Dokumentation fertigstellen | 2.0 | | | |
| | | | Korrektur Dokumentation | 4.0 | | | |
| | Mi | 15. 6. 11 | Korrektur Dokumentation | 5.0 | | | |
| | | | Erfahrungsbericht | 1.0 | | | |
| | Do | 16. 6. 11 | vorbereiten zur Abgabe | 6.0 | | | |
| | Fr | 17. 6. 11 | Abgabe, HSR Forum | 6.0 | | | |
| | | | | | **32.0** | **410.0** | 360.4 |

**Total Ist - Arbeitszeit:** **410.0**

**Total Soll - Arbeitszeit:** **360.4**

**Differenz:** **410.0**

# Zeiterfassung

**Projekt:** **Trainingsadministration**
**Teammitglied:** **Silvan Gacond**

**Durchschnittliche Soll-Arbeitszeit:** **21.2**
**(gerechnet mit 17 Wochen, Feiertage nicht miteinbezogen)**

| Woche | Tag | Datum | Tätigkeit | Zeit | Zeit/Wo | Total | Soll |
|---|---|---|---|---|---|---|---|
| Woche 1 | Di | 22.2. | Setup Doku, einlesen CRM | 6.0 | | | |
| | Mi | 23.2. | Technologiestudium CRM | 6.0 | | | |
| | Do | 24.2. | Technologiestudium CRM / SL / Services | 6.0 | | | |
| | Fr | 25.2. | Sitzung Anforderungen | 2.0 | | | |
| | | | Analyse | 4.0 | **24.0** | **24.0** | 21.2 |
| Woche 2 | Mo | 28.2. | Zwischensitzung | 1.0 | | | |
| | | | Anforderungen | 5.0 | | | |
| | Di | 1.3. | Anforderungen / Analyse | 6.0 | | | |
| | Mi | 2.3. | Anf / Analyse / Technologie | 6.0 | | | |
| | Do | 3.3. | Technologie / Solution aufsetzen | 7.0 | **25.0** | **49.0** | 42.4 |
| Woche 3 | Mo | 7.3. | Prototyp Dynamics Web Services | 6.5 | | | |
| | Di | 8.3. | Einlesen / Ausprobieren Prism | 7.0 | | | |
| | Mi | 9.3. | Einlesen / Ausprobieren Prism | 3.0 | | | |
| | | | Silverlight Architektur evaluieren | 3.0 | | | |
| | Do | 11.3. | Sitzung Zwischenstand (H.Huser & | 1.5 | | | |
| | | | Aufsetzen SL Architektur mit Prism | 4.0 | **25.0** | **74.0** | 63.6 |
| Woche 4 | Mo | 14.3. | CRM Solution aufsetzen und mit | 7.0 | | | |
| | Di | 15.3. | Testdaten befüllen | 6.0 | | | |
| | | | RichText Editor für CRM | 2.0 | | | |
| | Mi | 16.3. | CRM Forms | 6.0 | | | |
| | Fr | 18.3. | Kursübersicht v1 | 4.0 | **25.0** | **99.0** | 84.8 |
| Woche 5 | Mo | 21.3. | Kursübersicht v1 | 7.0 | | | |
| | Di | 22.3. | Kursübersicht v1, Kursdetail v1, Caching, | 8.0 | | | |
| | Mi | 23.3. | Connector, Mock (initial Abfüllen und | 6.0 | | | |
| | Fr | 25.3. | ausprobieren) | 6.0 | | | |
| | | | Sitzung | 1.0 | **28.0** | **127.0** | 106.0 |
| Woche 6 | Mo | 28.3. | RIA & Membership, einlesen, | 6.0 | | | |
| | Di | 29.3. | Registrierung, CRM Vorgang | 6.0 | | | |
| | Mi | 30.3. | Registrierung, mehr Testdaten für CRM, | 7.0 | | | |
| | Fr | 1.4. | CRM UI | 5.0 | | | |
| | | | Sitzung | 1.0 | **25.0** | **152.0** | 127.2 |
| Woche 7 | Mo | 4.4. | Registrierungsformular, Ablaufdiagramm | 7.0 | | | |
| | Di | 5.4. | CRM Automatismen für Registrierung | 8.0 | | | |
| | Fr | 8.4. | (Konzept, ausprobieren) | 4.0 | | | |
| | | | Sitzung (H.Huser & J.Jucker) | 1.5 | **20.5** | **172.5** | 148.4 |
| Woche 8 | Mo | 11.4. | CRM Workflow für Registrierung, | 6.5 | | | |
| | Di | 12.4. | CRM Duplikaterkennung E-Mail | 6.0 | | | |
| | Mi | 13.4. | Doku CRM, Architektur | 7.5 | **20.0** | **192.5** | 169.6 |
| Woche 9 | Mo | 18.4. | Dokumentation nachziehen (CRM, RIA, Architektur, Plug-Ins, Businesslogik), Vorbereiten Zwischenpräsentation | 4.0 | | | |
| | | | Techn. Bericht | 2.0 | | | |
| | Di | 19.4. | Techn. Bericht | 6.0 | | | |
| | Mi | 20.4. | Techn. Bericht | 5.0 | **17.0** | **209.5** | 190.8 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Woche 10 | Di | 26.4. | Vorbereiten Zwischenpräsentation | 6.0 | | | |
| | Mi | 27.4. | Vorbereiten Zwischenpräsentation | 4.0 | | | |
| | | | Zwischenpräsentation | 2.0 | | | |
| | Fr | 29.4. | Feedback Zw.Präsentation notieren, | 2.0 | | | |
| | | | Solution Clean-Up, sauberer Stand nach | 2.0 | | | |
| | | | | | **16.0** | **225.5** | 212.0 |
| W 11 | Mi | 4.5. | Kursliste (mit Auswahlbaum) | 7.0 | | | |
| | Fr | 6.5. | Kursliste (mit Auswahlbaum) | 6.0 | | | |
| | | | | | **13.0** | **238.5** | 233.2 |
| Woche 12 | Mo | 9.5. | Besprechung Zwischenstand | 0.5 | | | |
| | | | Plugins Statusverwaltung | 7.0 | | | |
| | Di | 10.5. | Plugins Statusverwaltung | 6.5 | | | |
| | Mi | 11.5. | Plugins Statusverwaltung | 6.5 | | | |
| | Do | 12.5. | CRM Frontend-Buttons & Script für | 7.0 | | | |
| | | | | | **27.5** | **266.0** | 254.4 |
| Woche 13 | Mo | 16.5. | UnitTesting CRM Statusverwaltung | 7.5 | | | |
| | Di | 17.5. | UnitTesting CRM Statusverwaltung | 7.5 | | | |
| | | | Sitzung mit J.Jucker | 1.0 | | | |
| | Mi | 18.5. | CRM Testing Doku beginnen | 7.0 | | | |
| | Fr | 20.5. | CRM Testing Doku | 5.0 | | | |
| | | | | | **28.0** | **294.0** | 275.6 |
| Woche 14 | Mo | 23.5. | UnitTesting Webkomponente, Probleme | 7.0 | | | |
| | Di | 24.5. | mit MockConnector / RIA | 3.0 | | | |
| | | | Dokustruktur überarbeiten | 3.0 | | | |
| | Mi | 25.5. | Dokustruktur überarbeiten | 2.0 | | | |
| | | | Doku Technologien und Konzepte | 4.0 | | | |
| | Fr | 27.5. | Doku Technologien und Konzepte | 5.5 | | | |
| | | | | | **24.5** | **318.5** | 296.8 |
| Woche 15 | Mo | 30.5. | Silverlight UI verbessern (Blend) | 6.0 | | | |
| | Di | 31.5. | Silverlight UI verbessern (Blend) | 7.0 | | | |
| | Mi | 1.6. | Doku Technologien und Konzepte | 6.0 | | | |
| | Fr | 3.6. | Doku Umsetzung | 3.0 | | | |
| | | | Doku überarbeiten / korrekturlesen | 4.0 | | | |
| | | | | | **26.0** | **344.5** | 318.0 |
| Woche 16 | Mo | 6.6. | Manuelle Tests vorbereiten / | 6.0 | | | |
| | Di | 7.6. | Mannelle Tests durchführen | 2.0 | | | |
| | | | Doku allgemeine Teile, Infrastruktur, | 4.0 | | | |
| | Mi | 8.6. | Doku allgemeine Teile, Infrastruktur, | 5.5 | | | |
| | Do | 9.6. | Abstract, Poster, allgemeine Teile, | 5.0 | | | |
| | Fr | 10.6. | Korrekturen | 6.5 | | | |
| | Sa | 11.6. | Gegenlesen zuhause, Korrekturen mit | 4.0 | | | |
| | | | | | **33.0** | **377.5** | 339.2 |
| Woche 17 | Mo | 13.6. | Handkorrekturen einbauen | 5.0 | | | |
| | | | Solution durchgehen: Filestruktur, | 3.0 | | | |
| | Di | 14.6. | Usings, Referenzen aufräumen, | 2.0 | | | |
| | | | Projektmanagement Charts, Doku | 3.5 | | | |
| | Mi | 15.6. | Letzte Korrekturen, Pendenzen | 3.5 | | | |
| | | | Sitzungsprotokolle reinschreiben | 2.0 | | | |
| | | | Erfahrungsbericht | 1.0 | | | |
| | Do | 16.6. | Drucken, binden, brennen, Poster | 6.0 | | | |
| | Fr | 17.6. | Abgabe, HSR Forum | 6.0 | | | |
| | | | | | **32.0** | **409.5** | 360.4 |

**Total Ist - Arbeitszeit:** **409.5**
**Total Soll - Arbeitszeit:** **360.4**
**Differenz:** **49.1**

# Anhang D

# Sitzungsprotokolle

## Sitzungsprotokoll 25.2.2011

- J.Jucker, INS
- A.Moschin, INS
- C.Meier, Student HSR
- S.Gacond, Student HSR

### Anforderungen

Die funktionalen Anforderungen sind in der Aufgabenstellung schon recht ausführlich beschrieben. Zusätzliche Ideen sind reichlich vorhanden und werden im Verlauf des Projektes genauer beschrieben und implementiert, sofern genügend Zeit vorhanden.

Grundsätzlich sollen alle Backend-Arbeiten innerhalb des CRM erledigt werden können. Eine Administrationsschnittstelle auf dem Web ist nicht erforderlich.

Im alten System ist vor allem die öffentliche Darstellung der Kursbeschreibungen ungenügend. Deshalb ist beim neuen System die Erfassung von HTML Elementen innerhalb des CRM ein wichtiger Punkt. Auch die Anpassung von Workflows soll im Vergleich zum alten System vereinfacht werden.

Zusätzlich zur Anmeldekomponente welche im Netz erscheint, sollen diverse kleinere Frontend Schnittstellen möglich sein, welche z.B. in die einzelnen Webseiten (Microsoft Schweiz, MIC Rapperswil, INS, ...) eingebunden werden können.

Es existieren verschiedene Kurstypen welche im Frontend unterschiedlich behandelt werden. Folgende Beispiele wurden angesprochen:

- normale Kurse mit mehreren Durchführungen
- Prüfungen (z.B. Cisco) mit Auswahl der entsprechenden Prüfung
- Events (ein Tag resp. eine Durchführung) ohne Registrierungszwang

## Architektur

Die ganze Datenhaltung wird vollumfänglich im CRM eingegliedert. Dies geht mit den Custom Entities die eingerichtet werden können sehr komfortabel und erspart eine weitere SQL Server Instanz.

Zusätzlich wird eine Zwischenschicht implementiert, welche die CRM Schnittstelle in ein normales Datenmodell "parst". Dies ermöglicht die Abtrennung des CRM und den Ersatz durch ein anderes Backend System. Ausserdem wird aus Sicht der Security der Webservice welcher mit der Silverlight Komponente (oder anderen Webtechnologie) kommuniziert sauber von dem Service auf dem CRM getrennt. Dies ermöglicht eine bessere Kontrolle und nur eine Verbindung aus der DMZ zum internen CRM Server. Auf dem CRM kann so ein Web-User eingerichtet werden, welcher auch in den jeweiligen Änderungsgeschichten erscheint. Diese Zwischenschicht hat zusätzlich den Vorteil, dass sie separat testbar ist.

## CRM Komponenten

Grundsätzlich können einige Komponenten des Datenmodells gut mit bestehenden CRM Komponenten realisiert werden.

Ein Kurstag sollte demnach zum Beispiel als *Service Activity* erfasst werden, so sind sämtliche Ressourcenzuordnungen bereits richtig erfolgt und die integrierte Planung der Ressourcen kann genutzt werden. Für die Interessengruppen von Kunden können *Marketinglisten* verwendet werden.

Die *Workflow-Engine* eignet sich bestens um Abläufe wie z.B. der E-Mail Versand oder Tracking des Status des Kurses zu verwalten.

Auch das eingebaute *Reporting* im CRM wird für einzelne Teile verwendet. z.B.: Teilnehmerlisten, Namensschilder, ...

## Diverses

Zwischenstände sollten per Nightly Build auf Testserver zur Verfügung stehen.

## Nächste Schritte

- Kurze Übersicht und Gliederung der Anforderungen
- Reduktion der Anforderungen für ersten Prototyp

- Architekturkonzept
- Prototyp durch alle Schichten
- Nächste Sitzung: Fr. 11.2.2011, 14:00, 6.010

# Sitzungsprotokoll 28.2.2011

- Prof. H.Huser, Projektbetreuer HSR
- C.Meier, Student HSR
- S.Gacond, Student HSR

## Besprechung Anforderungen

Kurze Besprechung der Sitzung vom 25.2.2011:

### Anforderungen

Eigentlich gemäss Aufgabenstellung.
Zusätzlich: HTML Editor für Kursbeschreibungen.
Ausserdem soll möglichst viel mit der eingebauten CRM Funktionalität abgedeckt werden (Workflows, etc.).

### Technologie

CRM Schicht ist klar.
Webserviceschicht unbedingt abgetrennt. Vor allem aus Sicherheitsgründen, aber auch wegen Lizenzen und Kapselung (Austauschbarkeit). Zwischenschicht wahrscheinlich mit WCF und eignen DTO's. Die CRM Objekte sollen nicht bis ins Silverlight verwendet werden.

### Organisation

Sitzung mit J.Jucker jeweils ca. alle zwei Wochen (wenn er Zeit hat und es Sinn macht).
Zwischenstand auf dem Testserver sollte aktuell sein (nightly build).
Am 2. und 3. Mai sind die Microsoft TechDays. Jürg Jucker hält dort einen Vortrag über CRM Erweiterbarkeit und möchte einen Stand zum zeigen mitnehmen.

## Nächste Schritte

Requirementsanalyse, evtl. Requirementsliste. Tasks für ersten Sprint priorisieren.
Nächste Sitzung: 7.3.2011

# Sitzungsprotokoll 7.3.2011

- Prof. H.Huser, Projektbetreuer HSR
- C.Meier, Student HSR
- S.Gacond, Student HSR

## Zusammenfassung aktueller Stand

Die Anforderungen wurden als Szenarios und Brief UseCases niedergeschrieben. Technologieprototyp durch alle Schichten ist in Entstehung. Zwischenschicht mit Zugriff auf CRM und WCF Schnittstelle für Silverlight.
Domainmodell für ersten UseCase (Kursinformationen beschaffen) ist vorhanden.

## Plan für diese Woche

- Ausformulieren der wichtigsten UseCases
- 1. Prototyp weiter vorantreiben
- Dokumentation: Gerüst vorbereiten, Nichtfunktionale Anforderungen schreiben

## Nächste Schritte

Es wird abgeklärt ob eine gemeinsame Sitzung mit J.Jucker am Freitag (11.3.2011) möglich sei. Ansonsten normale Zwischenstandssitzung am Montag (14.3.2011).

# Sitzungsprotokoll 11.3.2011

- Prof. H.Huser, Projektbetreuer HSR
- J.Jucker, INS
- C.Meier, Student HSR
- S.Gacond, Student HSR

## Zusammenfassung aktueller Stand

### Anforderungen

Anforderungen durchbesprochen (Szenarios, UseCases, Domain Modell). Alles grundsätzlich ok, einige kleine Änderungen angebracht.

### Prototyp

Prototyp durch alle Schichten funktioniert, momentan noch mit WCF.
Architektur und Dependency Injection: sehr schön, aber für diesen UseCase nicht unbedingt nötig. Trotzdem Architektonisch sehr sauber und interessant.

## Entscheid

Kommunikation Silverlight - Zwischenschicht mit RIA Services:

+ Authentication
+ Filter, Validierung, etc.
- Domain Service, Mehraufwand

HTML Rich Text Editor Komponente im CRM mit Telerik Control.
Zusätzliche Validierungen im CRM mit Plugins.

## Nächste Sitzung

21.3.2011 14:00

# Sitzungsprotokoll 21.3.2011

- Prof. H.Huser, Projektbetreuer HSR
- C.Meier, Student HSR
- S.Gacond, Student HSR

## Zusammenfassung aktueller Stand

Architektur mit RIA Services und POCO's.
UI Architektur, Prism / MEF.
Domain Schicht (anhand Domain Model auf Zwischenschicht).
Telerik HTML Editor.

## Ausblick auf nächste Woche

Loginprototyp, nachher Implementierung gemäss UseCases (Architektur OK).
Nächste Sitzung: 1.4.2011, 11:00

# Sitzungsprotokoll 1.4.2011

- Prof. H.Huser, Projektbetreuer HSR
- J.Jucker, INS
- C.Meier, Student HSR
- S.Gacond, Student HSR

## Zusammenfassung aktueller Stand

Nochmals zeigen der Architektur und Funktionalität im aktuellen Prototyp: RIA Services
UI mit Prism / MEF
Warenkorb / Merkliste
Infrastruktur für Unit Tests (Dummydaten für Service, SL Unit Testing Projekt)

## Anforderungen

UseCases teilweise nochmal durchbesprochen (E-Mail Benachrichtigung wann, etc.).

- Zahlung für Kurse unterschiedlich, mal im System noch nicht vorsehen.
- Passwort vergessen per E-Mail regeln.
- falls schon registriert, eventuell Kontakt zuordnen.

CRM Plugins: ev. freie Plätze überprüfen? ev. Email Adressduplikate verhindern?
Eventuell muss mit Read-Only Feldern gearbeitet werden, die Redundanzen enthalten.

## Ziele für nächste Wochen

Gestaltung UI für erste UseCases.
Vollständige Implementation der Grundfunktionen.

## Nächste Sitzung

8.4.2011, Zeit per Mail

# Sitzungsprotokoll 8.4.2011

- Prof. H.Huser, Projektbetreuer HSR
- J.Jucker, INS
- C.Meier, Student HSR
- S.Gacond, Student HSR

## Aktueller Stand

Umsetzung der UseCases schreitet voran.
CRM Plugin um Email Duplikate zu verhindern ist fertig.
Best Practice für Domain Context abgeklärt: pro ViewModel oder pro Applikation (in unserem Fall: Session).
Membership Provider in erster Version fertig, braucht noch Refactoring.

## Zwischenpräsentation und Gegenleser

Termin für Präsentation ausmachen.
Abklären was genau erwartet wird, Mail schreiben.

## Tech Days

Jürg Jucker möchte unsere Applikation an den TechDays zeigen: wir brauchen einen lauffähigen Stand nach Ostern.
Was man zeigen können sollte: Plugin, Anbindung an Web-Frontend, evtl. Teile vom Code und Doku.

## Name definitiv

„Kursanmeldungssystem mit Silverlight und Dynamics CRM“

## Nächste Sitzung

18.4.2011, 14:00

# Sitzungsprotokoll 18.4.2011

- Prof. H.Huser, Projektbetreuer HSR
- C.Meier, Student HSR
- S.Gacond, Student HSR

## Aktueller Stand

Dokumentation vorangetrieben.
Membership Provider Refactoring, via Authentication Service.

## Zwischenpräsentation

Termin: 27.4.2011
Inhalt:

- Ausgangslage
- Aufgabenstellung: Basisfunktionalität, Erweiterungen
- CRM Erweiterungen: Grundfunktionalität CRM, Erweiterungen, Schnittstellen
- Architektur: Evaluation, CRM-Seite, RIA Services, Prism / MVVM, Testing
- Stand: Planung, Umsetzung, Demo
- Probleme / Herausforderungen
- Nächste Schritte

## Nächste Woche

Vorbereitung Zwischenpräsentation.
UI Gestaltung / Vorbereitung TechDays.
Dokumentation nachziehen:

- Techn. Bericht: Einstieg erleichtern, ca 20 Seiten.
- Struktur grundsätzlich OK. Eventuell zusätzliches Kapitel über Konzepte?

Nächste Sitzung: ad Hoc, H. Huser kommt vorbei (anfangs Woche).

# Sitzungsprotokoll 16.5.2011

- Prof. H.Huser, Projektbetreuer HSR
- C.Meier, Student HSR
- S.Gacond, Student HSR

## Aktueller Stand

Neue Unit Tests für CRM-Plugins.
GUI-Funktionalität fast komplett.
UnitTesting des Services macht noch Probleme (Dummydaten für DomainService generieren nicht nachvollziehbare Exception).

## Nächste Woche

Dokumentation weitermachen.
Unit Tests reparieren.
GUI-Kosmetik.
Test durch J.Jucker.

## Präsentation / Mündliche Prüfung

Anfangs August. H. Huser klärt mit Experte ab.

## Nächste Sitzung

Zwischenbesprechungen im kleinen Rahmen ad Hoc, H. Huser kommt vorbei (jeweils anfangs Woche).