HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

vSphere Life-Cycle-Management für virtuelle Server an der HSR

Studienarbeit

Abteilung Informatik
Hochschule für Technik Rapperswil

Herbstsemester 2013

Autor(en): Raphael Faes
Tobias Büchel
Betreuer: Prof. Hansjörg Huser
Projektpartner: HSR Abt. Informatik

Änderungsgeschichte

Datum	Version	Änderung	Autor
12.12.13	0.1	Zusammenführen der Dokumente in Projektdokumentation	Tobias Büchel
17.12.13	0.2	Designentscheide, Domain Model, database model, UI	Tobias Büchel
18.12.13	0.3	Architektur	Tobias Büchel
18.12.13	0.4	Fazit	Raphael Faes
20.12.13	0.5	Überarbeitung Zeitauswertung	Tobias Büchel
20.12.13	1.0	Final	Raphael Faes, Tobias Büchel

Inhaltsverzeichnis

1 Abstract

Das Projekt beinhaltet die Ablösung des aktuellen Systems zur Bestellung und Verwaltung von virtuellen Maschinen für die Abteilung I (Informatik) der HSR.

Dozenten können über das Web-Frontend virtuelle Maschinen bestellen und Benutzern zuweisen. Der Administrations-Zugang ermöglicht es der verantwortlichen Person diese Aufträge anzusehen und vor dem Deployment erforderliche Informationen, wie zum Beispiel den Hostname und die IP-Adresse, anzupassen. Dabei kann der Deploy- und Destroy-Prozess an die vSphere-Umgebung direkt aus der Webapplikation getätigt werden. Auf der Overview-Page erkennt der Administrator auf einen Blick, bei welchen VMs Handlungsbedarf besteht. Farben und Icons markieren Zustände, wie zum Beispiel, dass eine neue Bestellung eingetroffen ist oder eine Maschine sich dem End-Of-Life-Datum nähert. Aktionen (Deploy, Destroy und Versand von E-Mails) können als Batch-Prozesse in Auftrag gegeben werden. Das Tool unterstützt ebenfalls das Parsen von Dokumenten und Versenden von E-Mails an die Benutzer und Dozenten der virtuellen Maschine, um diese über den Status der VM zu informieren. Durch die Anbindung an das Active Directory der HSR können Berechtigungen über Security Groups vergeben werden, was eine nahtlose Integration in die bestehende Infrastruktur ermöglicht.

Die VLCM-Applikation automatisiert zeitintensive Aufgaben, die bis anhin manuell getätigt werden mussten, und ermöglicht eine höhere Qualität des vSphere-Life-Cycle-Management-Prozesses für die Kunden der Abteilung I an der HSR.

2 Problemanalyse

2.1 Aufgabenstellung

2.1.1 Prozess

Zu den wesentlichen Prozessen gehört der Deploy- und Destroy-Prozess von virtuellen Maschinen. In den zwei folgenden Unterkapiteln ist der Zustand des zu ablösenden Systems festgehalten. Die roten Markierungen kennzeichnen zeitlich aufwändige Tasks, welche das neue System durch Automation verringern soll.

2.1.1.1 Deploy-Prozess

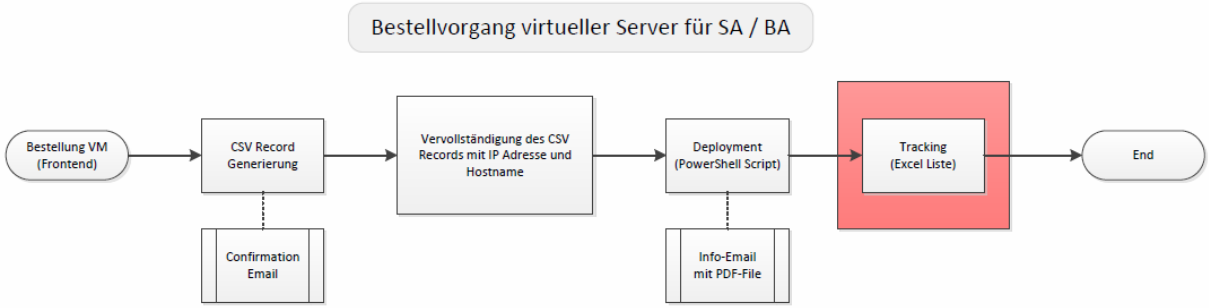

Abbildung 1: Deploy-Prozess

Beim Deploy-Prozess ist der aufwändigste Teil das Nachführen der VM-Details in der Excel-Liste. Datums-Felder wie CreateDate, DestroyDate, OrderDate, Semester usw. werden soweit technisch möglich automatisch nachgeführt, so dass der Teilprozess „Tracking“ fast ganz entfällt.

2.1.1.2 Destroy-Prozess

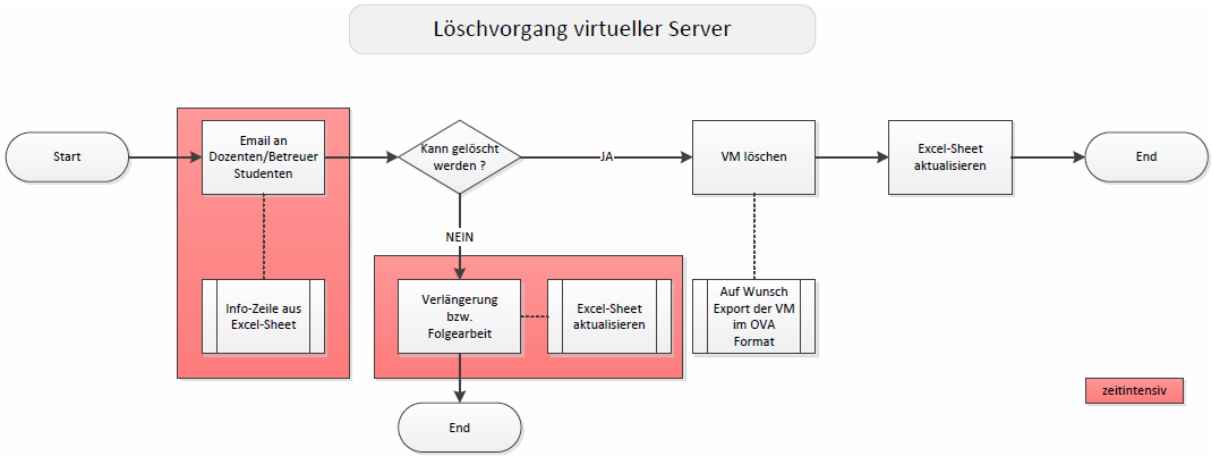

Abbildung 2: Destroy-Prozess

E-Mail-Benachrichtigungen sollen per Knopfdruck aus dem Administrationsbereich versendet werden können. „Excel-Sheet aktualisieren“ entspricht etwa dem „Tracking“ aus dem Deploy-Prozess und bietet viele Möglichkeiten zur Automation. Gewisse Aspekte sind aber organisatorischer Natur (z.B. wenn jemand eine VM länger benötigt) und können nicht vollständig automatisiert werden.

2.1.2 Ziel der Arbeit

Web Applikation für das Erfassen und Verwalten von VMs

- Konzeption eines benutzerfreundlichen Interfaces
- Definition der Schnittstellen zu den Datenlieferanten
- Erzeugen des Virtual Infrastructure Guide PDFs und versenden via E-Mail
- Design und Implementation auf MVC 4
- Konzeptionelle Abklärungen
    - Persistente Speicherung der Daten

2.1.3 Resultate

- Lauffähige Software gemäss Spezifikation
- Projekt- und Produktdokumentation
- Dokumente gemäss Vorgaben des Studiengangs

2.1.4 Auftraggeber

Abteilung Informatik der HSR

Oliver Rehmann

2.1.5 Projektteam

Tobias Büchel

Raphael Faes

2.1.6 Betreuung HSR

Hansjörg Huser

2.2 Projektabwicklung

2.2.1 Termine

- Beginn der Arbeit: Mo, 16. Sept. 2013
- Abgabetermin Kurzfassung/Mgmt-Summary zum Review: 16. Dez. 2013
- Abgabetermin 20. Dez. 2013, 12.00 Uhr
- Zwischenbesprechung/Review mit Auftraggeber nach Projektplan

2.2.2 Arbeitsaufwand

Für die erfolgreich abgeschlossene Arbeit werden 8 ECTS angerechnet. Dies entspricht einer Arbeitsleistung von mind. 240 Stunden pro Student. (14 Wochen zu ca. 17h)

2.2.3 Dokumentation und Sourcen

Projektdokumente sind im HSR git-Repository abgelegt

Das **Zeitmanagement** wird in Redmine nachgeführt

Der Source Code wird auf dem TFS-Server eingecheckt

2.3 Anforderungsanalyse

Folgende spezifische Anforderungen wurden während der Projektphase vom Auftraggeber als wichtige Features definiert.

2.3.1 Funktionale Anforderungen

- Anmeldung gegen Active Directory
- Anhand von AD Gruppen wird die Berechtigung gesteuert
- Automatisches Mail Versand bei Events
- Jede VM hat ein Start und ein Enddatum
- PDFs und Word Dokumente müssen generiert werden können mit den Infos der VM und den dazugehörenden Benutzern
- Die Deploy Funktionalität gegen ein Interface implementieren, damit dieses ausgetauscht werden kann
- Die VMs müssen auf der vSphere Plattform erstellt und wieder gelöscht werden können
- Die VMs sollen sich anhand der verbleibenden Laufzeit farblich unterscheiden
- Das System soll vor dem Deployen überprüfen ob die Pflichtfelder ausgefüllt sind (IP Adresse und Hostname)
- Tags um die PDFs und Word Dokumente zu parsen, müssen leicht erweiterbar sein
- Die Namen der VMs müssen sich jedes Semester wiederholen können
- History der VMs muss gegeben sein
- Die Aktionen auf dem Userinterface müssen immer für eine oder mehrere VMs möglich sein
- Beim Textfeldern soll das System wenn möglich eine Auto Complete Funktion anbieten
- Die Datengrids sollen mit Search-as-you-type gefiltert werden können
- Möglichst viel soll über die Datenbank konfiguriert werden können
- Das System muss sich merken können, ob und wann Mails versendet wurden
- Email Texte müssen konfigurierbar sein
- Email Texte müssen geparst werden können mit den Tags

2.3.2 Nichtfunktionale Anforderungen

- Die Datenhaltung muss in einer Datenbank erfolgen
- IE ab Version 8 aufwärts müssen unterstützt werden
- MVC 4 wird als Implementationstechnologie vorausgesetzt

2.3.3 Optionale Anforderungen

- Überprüfung der IP Adressen gegen einen DHCP Server (QIP)
- Export der VM
- Erstellen der VM erfolgt direkt aus dem System, anfangs wird nur das bereits bestehende Skript gefüttert
- Einen Admin Read Modus für Stellvertretung

2.3.4 Use Case

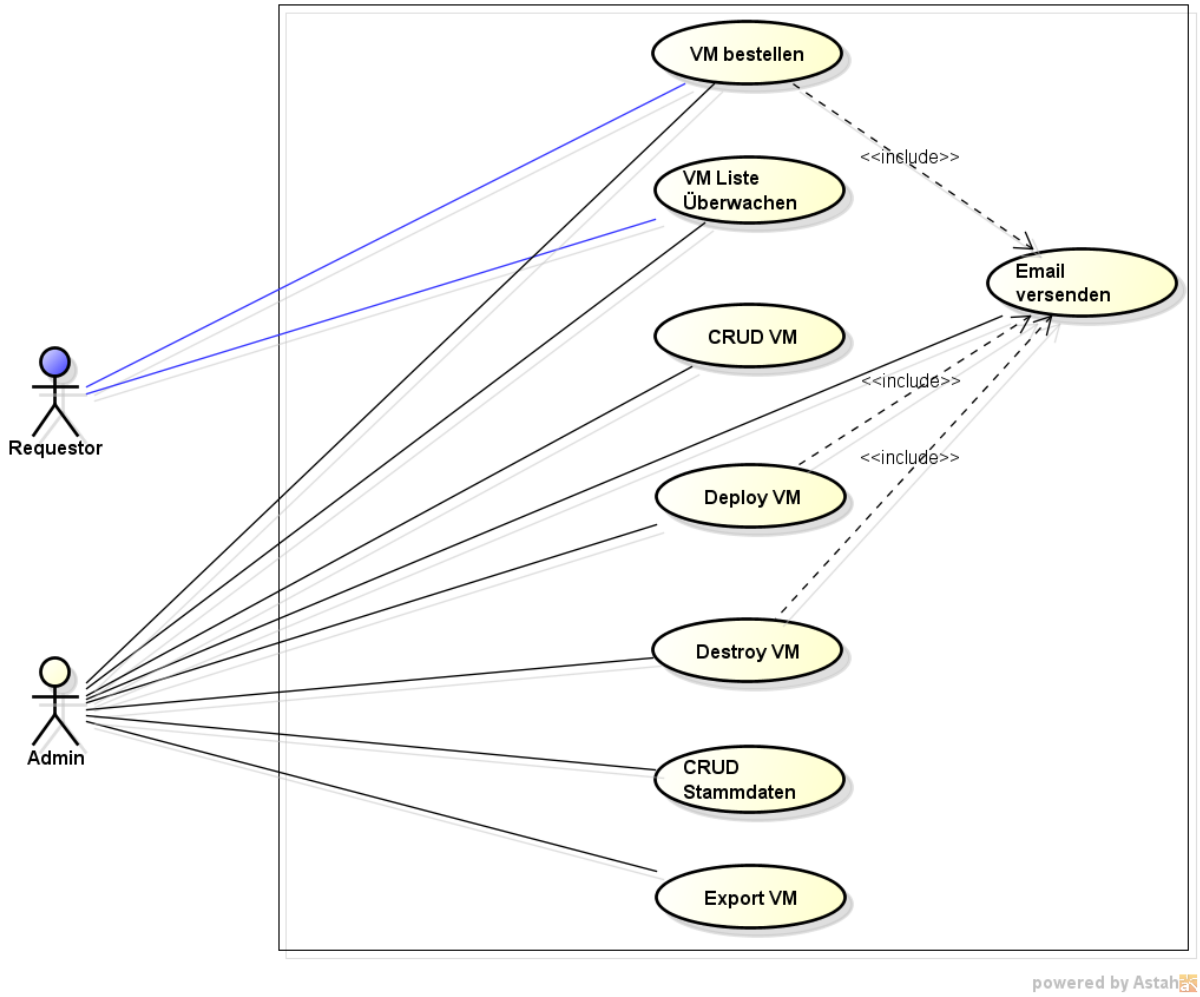

Abbildung 3: Use Case "VLCM"

Beim Requester handelt es sich um den Auftraggeber der virtuellen Maschinen, in der Regel Dozenten, welche für Studentenarbeiten Resourcen benötigen. Der Bestellvorgang (Order Process) verlangt die Erfassung des Requesters/Advisors und der Benutzer der zu erstellenden Maschine. Requestors können ihre bestellten VMs inklusive Details jederzeit anzeigen lassen.

Die Administrator-Rolle ist zuständig vom Entgegennehmen der Bestellungen über das Deployen bis zur Vernichtung (Destroy VM) der Hosts. Die Kommunikation zwischen Administrator und Requestor erfolgt primär über Templates (Word/PDF), welche via E-Mail vom System versendet werden.

Der wichtigste Use-Case für den Administrator stellt „VM Liste überwachen“ dar. Der Requester besitzt den Use-Case ebenfalls, allerdings mit beschränkter Funktionalität. So kann der Admin E-Mails für Anleitungs- und Lösch-Infos versenden; der Requester besitzt dieses Feature nicht.

„Export VM“ ist ein optionales Feature, was den kompletten Export der VM auf einen Fileserver beschreibt. Gedacht für Projekte, welche in einer anderen Infrastruktur weiter entwickelt werden oder auch für die Integration ausserhalb der HSR.

2.3.5 Datengerüst

Die Datenmenge bewegt sich für ein IT-System im kleinen Rahmen. Rows für Stammdaten sind in der Regel weniger als 10 pro Klasse. VMs kommen jedes Jahr im Schnitt etwa 100 hinzu. Für die Performance auf Datenbank-Ebene ist das nicht relevant. Einzig das Rendering des Browser bei mehreren hundert VMs könnte langsam werden. Eine History-Funktion für Gelöschte wirkt diesem Effekt entgegen, da normalerweise nicht mehr wie 80 virtuelle Maschinen aktiv sind. Unterstützt wird das Handling der Daten durch eine Filter-Funktion für VMs (VM Overview).

2.4 Domainanalyse

2.4.1 Domainmodel

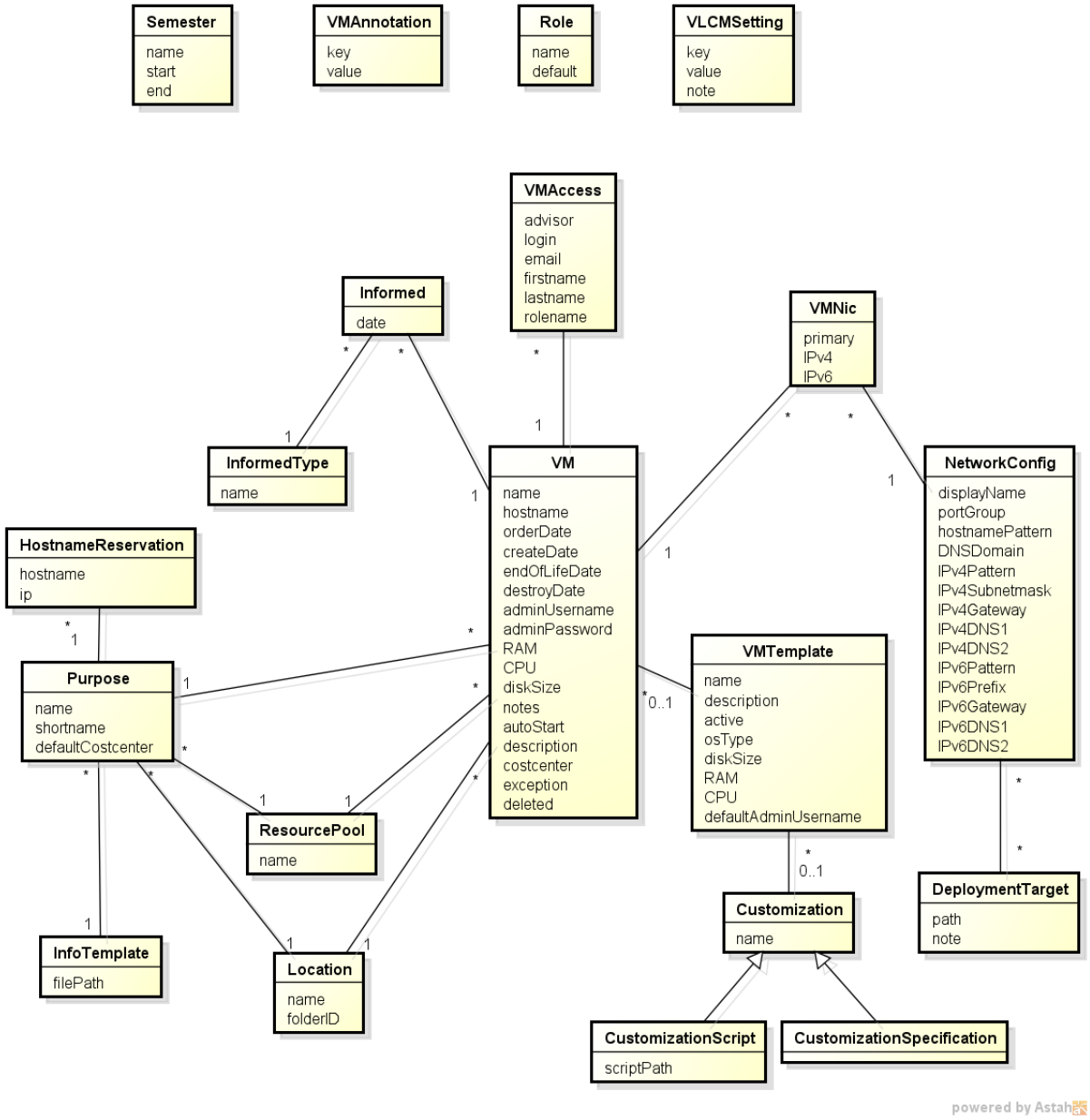

Abbildung 4: Domain Model "VLCM"

Kern des Datenmodells stellt die Klasse **VM** (virtual machine) mit den Berechtigungen der Benutzer (**VMAccess**) dar. Das Image für den Deploy-Prozess wird durch **VMTemplate** vorgegeben, was zusätzliche durch eine **CustomizationSpecification** (Windows) und/oder einem **CustomizationScript**

(Windows + Linux) erweitert werden kann. Netzwerkkarten (**VMNic**) werden einer spezifischen **NetworkConfig** zugeordnet, was in vSphere einer port group entspricht. Die Art der Arbeit, z.B. ob es sich um eine Bachelorarbeit handelt, ist unter **Purpose** festgehalten. Hostnamen können in **HostnameReservation** für einzelne Purposes reserviert und dann nicht anderweitig verwendet werden. Man kann verschiedene File-Templates (**InfoTemplate**) für unterschiedliche Arten von Aufträgen definieren. **Location** legt den Ordner für die Übersicht in der vSphere-Konsole fest. In **Informed** und **InformedType** wird der Versand der E-Mails mit den entsprechenden InfoTemplates festgehalten.

Semester werden den VMs anhand des createDates zugeordnet. **VLCMSetting** ist eine Key/Value-Klasse um die Webapplikation auf globaler Ebene zu konfigurieren. **ResourcePool** bedeutet in der vSphere-Umgebung eine Gruppe mit zugesicherten Hardware-Komponenten (CPU, RAM, …). Im VLCM wird dies Primär als Deployment-Ziel verwendet. (Beim Deployment der VMs muss ein „Ziel“, wie z.B. Host oder ResourcePool angegeben werden.)

2.5 User Interface

2.5.1 Card Sorting

Mit Hilfe von Karteikarten wurde zusammen mit Oliver Rehmann die Gruppierung und Beschriftung der verschiedenen Klassen vorgenommen. Als Erstes hat das Entwicklungsteam Karten mit den Klassennamen aus dem Domain Model beschriftet. Im zweiten Schritt wurden diese dann Oliver Rehmann unsortiert auf dem Tisch vorgelegt. Die Aufgabe war nun das Gruppieren der Karteikarten und diese Gruppen anschliessend mit einer Überschrift zu versehen. Die erhaltene Struktur wurde dann für die Navigation und die visuelle Abhängigkeit in den CRUD-Masken angewendet. Das Ergebnis des Card Sortings ist in der Liste unterhalb zu sehen. Fett gekennzeichnet sind die definierten Gruppennamen, welche die dazugehörigen Klassen auflistet:

vSphere

- VM Host
- Role
- Data Store (obsolete)
- Resource Pool

Customization

- Network Config
- VM Template
- Customization
- Purpose
- Location
- Semester

VM

- VM Overview
- VM Nic
- VM Access

VLCMSetting *(keinem Gruppennamen oder Klassen zugeordnet)*

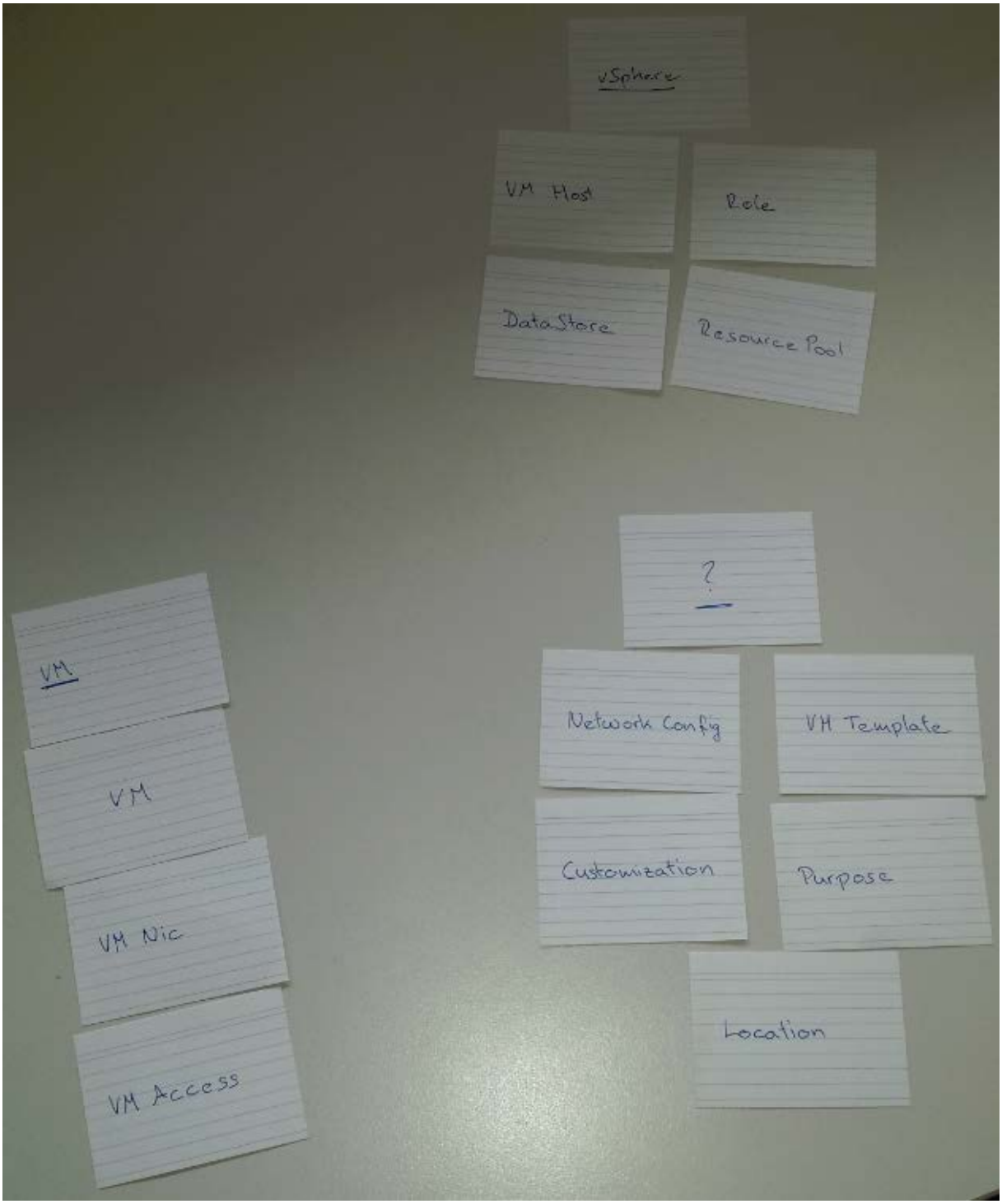

Abbildung 5: Karteikärtchen durch zukünftigen Benutzer strukturiert

2.5.2 Wireframes

Um ein ersten Vorschlag für das Layout und Design der Webapplikation zu präsentieren wurde mittels balsamiq (www.balsamiq.com) mehrere Wireframes ausgearbeitet. Eine Test-Lizenz haben wir von Markus Stolze auf Anfrage erhalten.

2.5.2.1 VM Overview

Den wichtigsten Screen für den Administrator stellt die Übersichts-Seite (Overview) dar. Auf dieser Page muss für den Verantwortlichen auf einen Blick klar sein, wo als nächstes Handlungsbedarf ansteht. Ebenfalls müssen sämtliche VMs abgefragt, deployed und destroyed werden können.

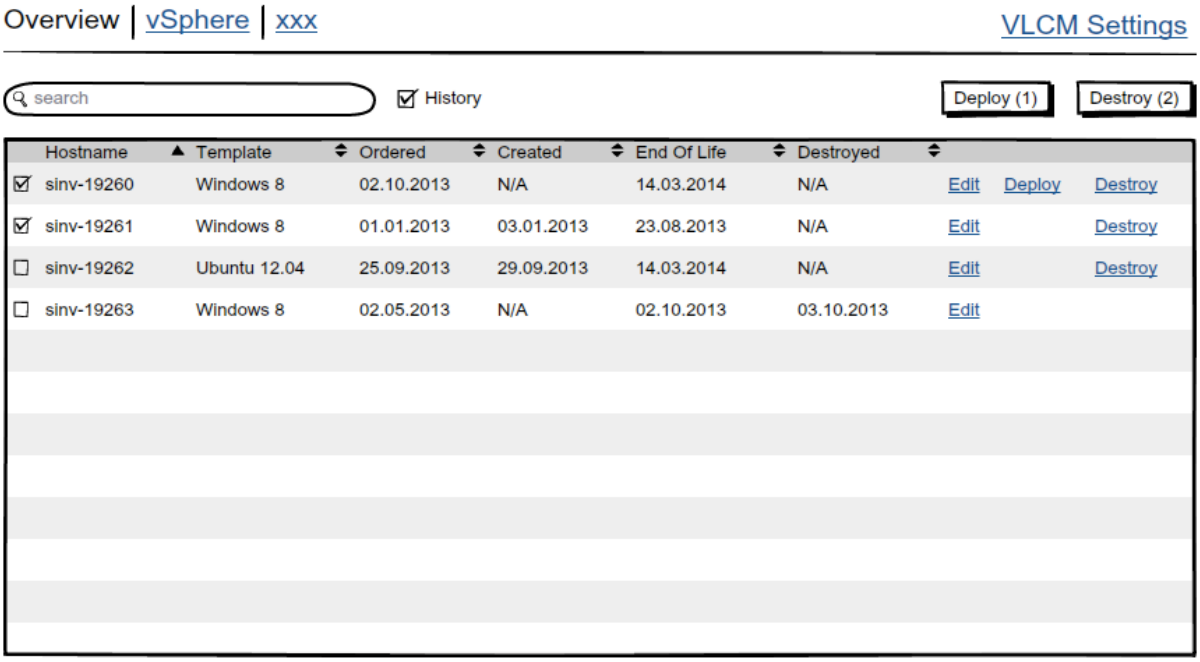

Abbildung 6: Entwurf der Startseite für die Admins

Das allgemeine Layout inklusive Menü wurde basierend auf dieser Vorlage umgesetzt. Während Benutzerbeobachtungen zwischen den einzelnen Entwicklungs-Iterationen haben wir uns entschieden, auch für den E-Mail-Versand Buttons anzulegen. Dies ermöglicht das Versenden von Notifikationen für mehrere virtuelle Maschinen bzw. deren Benutzern. (Batch-Operationen, wie es für „Deploy“ und „Destroy“ zu sehen ist.)

2.5.2.2 VM Detailansicht

Die Übersichtsseite bringt uns zur nächsten wichtigen Page – Detailansicht einer VM. In der linken Spalte sind in der Groupbox „VM“ die Daten der virtuellen Maschine selbst zu sehen. Rechts davon referenzierte Klassen aus dem Domain Model (siehe Kapitel 2.4.1), welche logisch gesehen eine Einheit sind. Um zu entscheiden, was zusammengehört, wurde jeweils auf die Ergebnisse des Cardsortings aus Kapitel 2.5.1 zurückgegriffen.

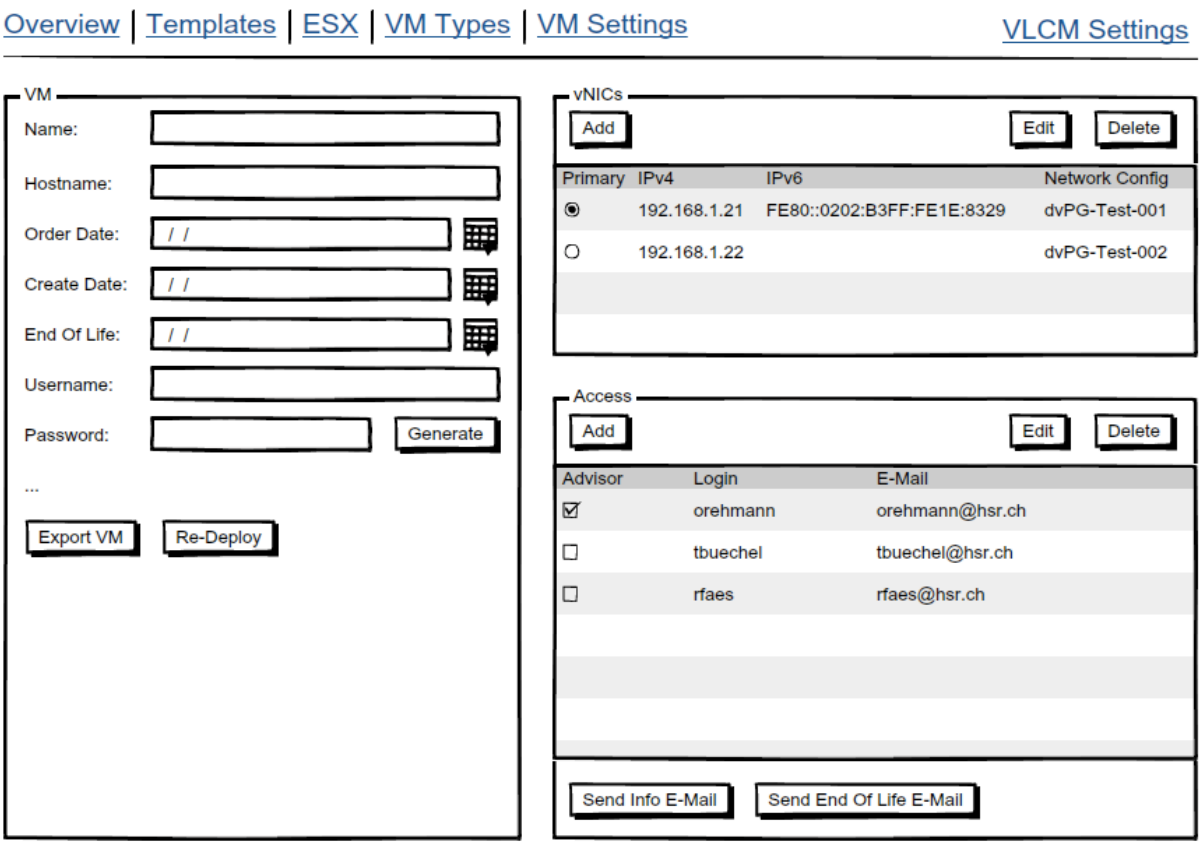

Abbildung 7: VMs - zentrale Daten für VLCM

Wichtig bei diesem Screen ist, dass die referenzierten Klassen auf der rechten Seite bearbeitet werden können, ohne dass die Seite verlassen wird. So öffnet ein Klick auf „Add“ oder „Edit“ folgendes modulares Fenster:

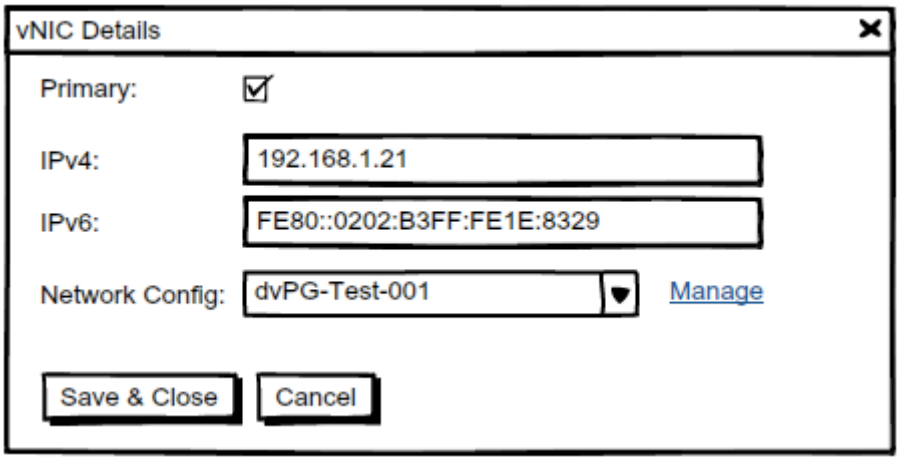

Abbildung 8: Fenster zum Erstellen und Bearbeiten von Interfaces

Damit wird erreicht, dass die Navigation durch das Objekt (VM mit ihren NICs), sich analog zum kognitiven Kontext des Benutzers verhält: Beim Bearbeiten eines referenzierten Objekt-Typs (Klasse) ist das Hauptobjekt immer noch im Hintergrund zu sehen und wird nicht durch eine Weiterleitung auf eine andere Webseite geschlossen.

Diese Strategie wurde auch bei anderen Klassen umgesetzt:

- VM & VMAccess
- VMTemplate & Customization
- NetworkConfig & DeploymentTarget
- Purpose & HostnameReservation

3 Projektplan

3.1 Projektorganisation

3.1.1 Organisationsstruktur

3.1.1.1 Tobias Büchel

Kenntnisse in: C#, SQL
Verantwortlichkeit: Datenbank, UI

3.1.1.2 Raphael Faes

Kenntnisse in: C#, SQL, JavaScript, jQuery
Verantwortlichkeit: EF, MVC

3.1.2 Externe Schnittstellen

3.1.2.1 HSR

Prof. Hansjörg Huser
Verantwortlichkeit: Betreuung, Beratung

Oliver Rehmann
Verantwortlichkeit: Industrypartner

3.2 Management Abläufe

3.2.1 Zeitliche Planung

Ganz allgemein wird das Projekt in zwei Teile aufgegliedert:

- Web-GUI und Business-Logik
- vSphere-Schnittstelle

Die Priorisierung liegt dabei auf der ersten Komponente, was in den Meilensteine **III** und **IV** zu sehen ist. Bei eventuellen Verzögerungen ist so die Fertigstellung des Web-Interfaces nicht gefährdet.

#	Datum	Phase	Goals
I	25.09.2013	End of Inception	Projektplan first darft inkl. Zeitplanung Infrastruktur set-up
II	16.10.2013	End of Elaboration	Prototyp
III	20.11.2013	Construction	GUI & Business inkl. Mail-Versand
IV	04.12.2013	End of Construction	vSphere Schnittstelle inkl. Web-Anbindung
V	13.12.2013	Transition	Systemtests abgeschlossen
VI	18.12.2013	End of Transition	Übergabe / Abschlussmeeting

3.2.2 Besprechungen

Wochentag: Mittwoch

Zeitpunkt: 15:00 – 16:00

Ort: 6.010

Teilnehmer: Prof. Hansjörg Huser,
Oliver Rehman,
Raphael Faes,
Tobias Büchel

3.3 Arbeitspakete

Arbeitspakete werden in Redmine geführt, siehe dazu Kapitel 2.2.3.

3.4 Qualitätsmassnahmen

3.4.1 Coding Guidelines

Betreffend Microsoft-Technologien werden „Coding Techniques and Programming Practices“ verwendet: http://msdn.microsoft.com/en-us/library/aa260844(v=vs.60).aspx

3.4.2 Pair Programming

Kritische oder komplexe Komponenten werden zu zweit an einer Workstation programmiert. Dabei übernimmt eine Person den Lead und tippt den Code, während der Andere den Code in Echtzeit Reviewed und dem Leader als Unterstützung dient.

3.4.2.1 Unit Tests

Eine Codeabdeckung von 100% ist nicht erreichbar. Für **Low-Level Funktionalität** streben wir ein Coverage von **85%** an. Code in **höheren Layern** (ohne UI) sollte eine Abdeckung von **75%** erreichen.

3.4.2.2 Systemtests

Vor Abschluss eines Meilensteins, die in der Regel mittwochs sind, wird ein Systemtest durchgeführt und dokumentiert.

Getestet wird dabei gegen geeignete Szenarien und insbesondere gegen die Use Cases.

Handlungsbedarf aus Systemtests werden vom Verantwortlichen selbständig noch am selben Tag koordiniert.

3.5 Risikomanagement

3.5.1 Analyse

#	Titel	Beschreibung	Max. Schaden [h]	Eintr. Wahrsch.	Gew. Schaden [h]
R1	Vision und Ablauf verstehen	Die Einarbeitung in dieses Thema benötigt mehr Zeit als geplant	15	5%	0.75
R2	Projektabgrenzung	Es wird versucht zu viele Feautures zu implementieren, was den Zeitrahmen sprengen könnte	40	10%	4
R3	Faking und Mocking	Das Faking und Mocking der einzelnen Komponenten benötigt mehr Zeit als geplant	10	10%	1
R4	Deployen von VM	Das Deployen der VM benötigt mit dem Assembly von vmWare mehr Zeit als geplant	10	10%	1
R5	Einarbeitung Javascript	Die Einarbeitung in dieses Thema benötigt mehr Zeit als geplant	5	5%	0.25
R6	Einarbeitung MVC	Die Einarbeitung in dieses Thema benötigt mehr Zeit als geplant	5	5%	0.25

3.5.2 Auswertung

Risiken R1-R6 aus der Analyse konnten alle erfolgreich vermieden werden. Am Ende der Projektphase wurde uns allerdings bewusst, dass wir für die Integration zuwenig Zeit geplant haben.

#	Titel	Beschreibung	Verursachter Schaden [h]
R7	Integration	Die Überführung der Software in die produktive Umgebung kann zu Problemen führen ➢ Anpassungen der SW nötig	30-40

Aufgrund unterschiedlicher Konfigurationsaspekte (z.B. Berechtigungen) zwischen unserer Entwicklungsumgebung und der produktiven Umgebung, wurde viel Zeit für die Vorbereitung und Durchführung der Integration gebraucht. Gewisse Probleme konnten in der vorgegebenen Zeit nicht gelöst werden und die Integration konnte nicht wie geplant abgeschlossen werden.

- Rückblickend sollte die Integration in so kurzen Projekten wie der Studienarbeit (SA) ausserhalb der vorgegebenen Zeit geplant werden, da dieser Teil schwer einzuschätzen ist, vor allem wenn es Schnittstellen an bestehende Systeme beinhaltet. (AD, vSphere)

3.6 Zeitauswertung

3.6.1 Stundenübersicht

3.6.1.1 Vergleich im Team

In der folgenden Grafik sind die Projektmitarbeiter mit ihren Stunden gegenübergestellt. Die vertikale Achse stellt die eingetragene Stunden für das Projekt dar – die horizontale Achse ist nach dem Muster <Jahr>-<Kalenderwoche> beschriftet.

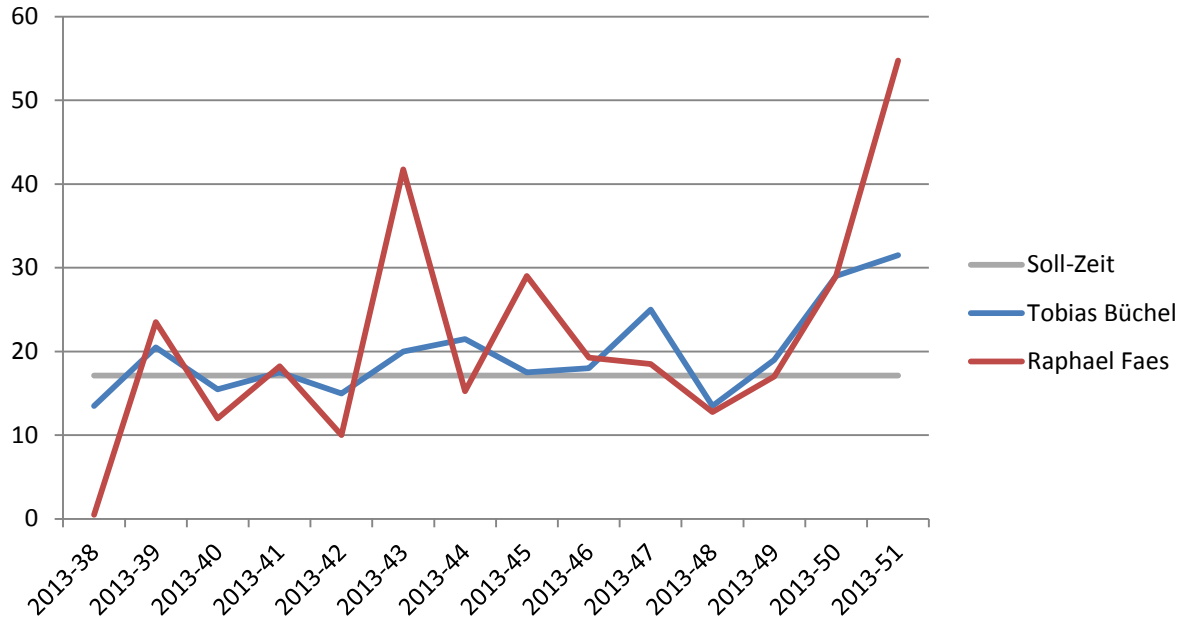

Abbildung 9: Zeitvergleich im Projektteam

Person	Stunden	Prozent
Tobias Büchel	277	47.9
Raphael Faes	301.5	52.1
Total	**578.5**	**100**

3.6.1.2 Totalaufwand

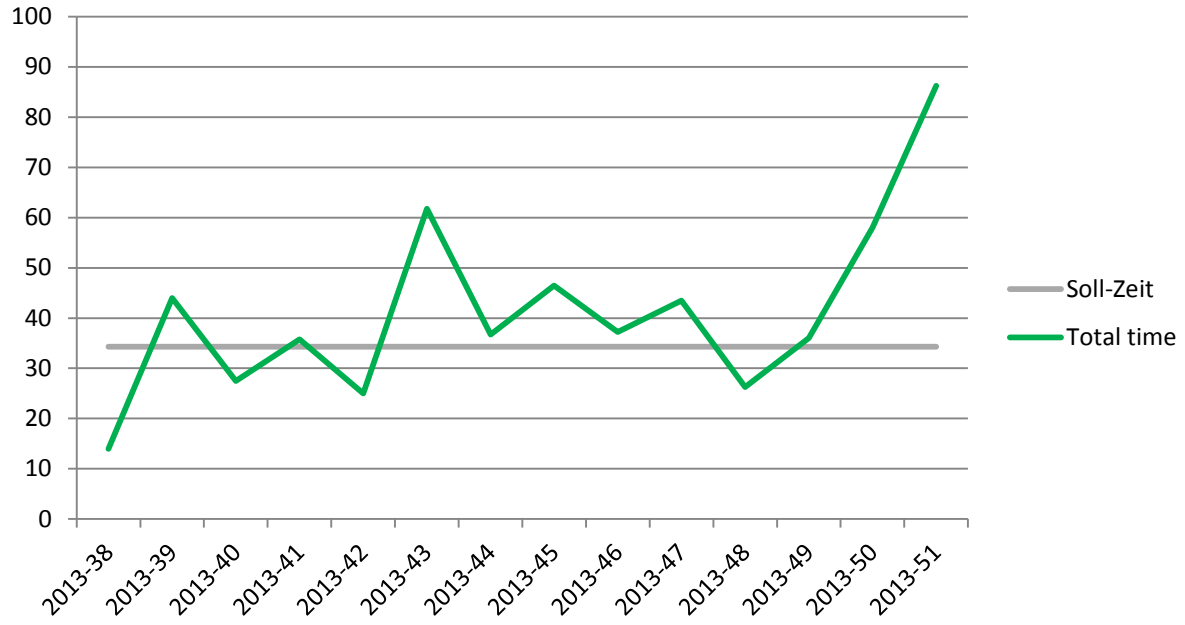

Abbildung 10: Totalaufwand

Zwei Peaks in Abbildung 10: Totalaufwand sind besonders auffällig:

- KW 43: Der erhöhte Aufwand ist dem Tag-Converter für das Parsen der Dokumente zuzuschreiben. Diese Aufgabe stellte mehr Aufwand als erwartet dar und wurde von Raphael Faes am Wochenende mit zusätzlicher Einsatzbereitschaft realisisert. Das dynamische Laden der Tags mittels Reflections, war **eine Herausforderung, die wir uns selber gestellt haben** und für uns unbedingt realisieren wollten.
- KW 51: Wie zu den Risiken in Kapitel 3.5.2 aufgezeigt wurde, stand für die Integrationsphase zu wenig Zeit zur Verfügung. Der Versuch, es dennoch in der letzten Woche zu integrieren kostete uns viel Zeit – eine Zusatzschicht musste eingelegt werden damit wir die Dokumentation noch fertig stellen konnten.

4 Architektur

Dieser Abschnitt beschreibt die Architektur Top-Down, d.h. beginnend von der allgemeinsten Ansicht wird Kapitel für Kapitel die Dokumentation technisch spezifischer und der Scope kleiner. Als Navigation durch die Komponenten dient das „Big Picture“ im nächsten Kapitel als Ausgangspunkt.

Der Inhaltliche Umfang beschränkt sich auf Aspekte, welche von der allgemeinen Vorstellung der einzelnen Technologien abweichen. Das beudetet zum Beispiel, dass das Prinzip einer MVC-Applikation nicht erklärt wird – beschrieben werden **Designentscheide basierend auf** diesen **Technologien** wie unser „Generic Controller“ (siehe Kapitel 4.4.2.2), welcher auf MVC beruht.

4.1 Big Picture

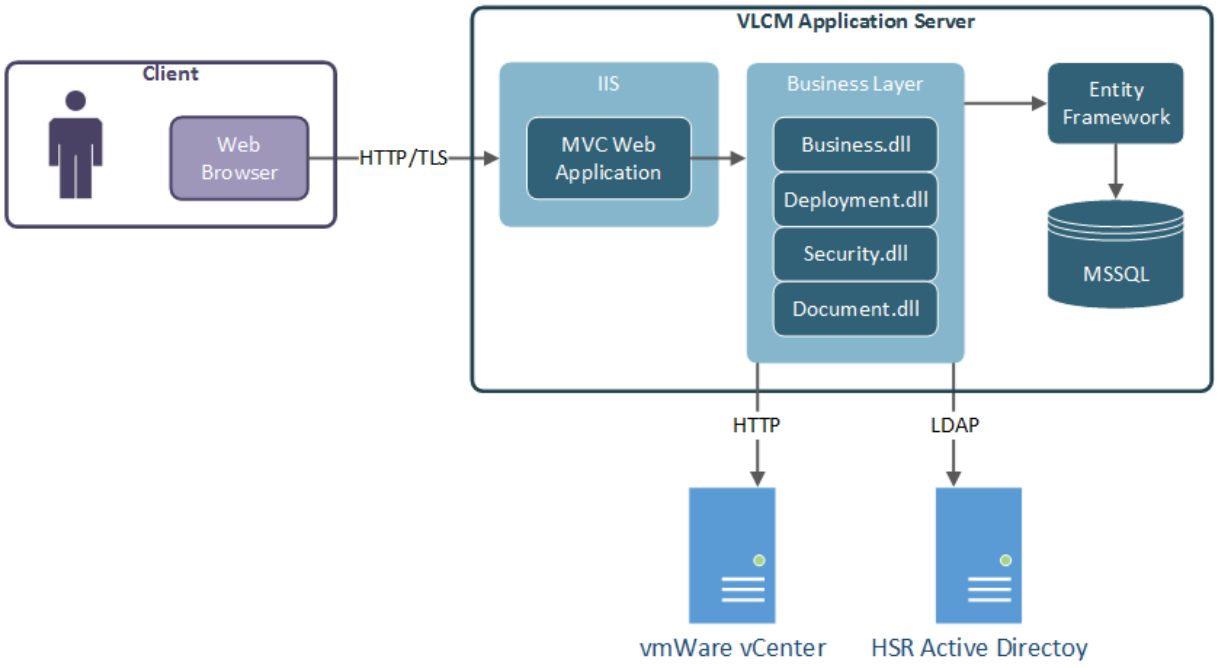

Abbildung 11: Big Picture

Schnittstelle	Protokoll	Aufgabe
Client	HTTP/TLS	UI für Kunden und Administrator
vmWare vCenter	HTTP	Deployment der virtuellen Maschinen
HSR Active Directory	LDAP	Authentifizierung der Website-Benutzer

In der folgenden Tabelle sind Referenzen auf detailierte Dokumentationen der Komponenten zu finden:

Komponente	Weiterführende Kapitel
MCV Web Application	4.4.2 Web.dll (Seite 22)
Business Layer	4.4.1 Assemblies (Seite 21)
MSSQL	4.3 Datenbank (Seite 20)
vmWare vCenter	4.5 vSphere-Schnittstelle (Seite 26)
HSR Active Directory	4.6 Active Directory-Schnittstelle (Seite 26)

4.2 Entwicklungs-Infrastruktur

Die Hosts und deren Beziehungen in der **Entwicklungsumgebung** sind unterhalb dargestellt:

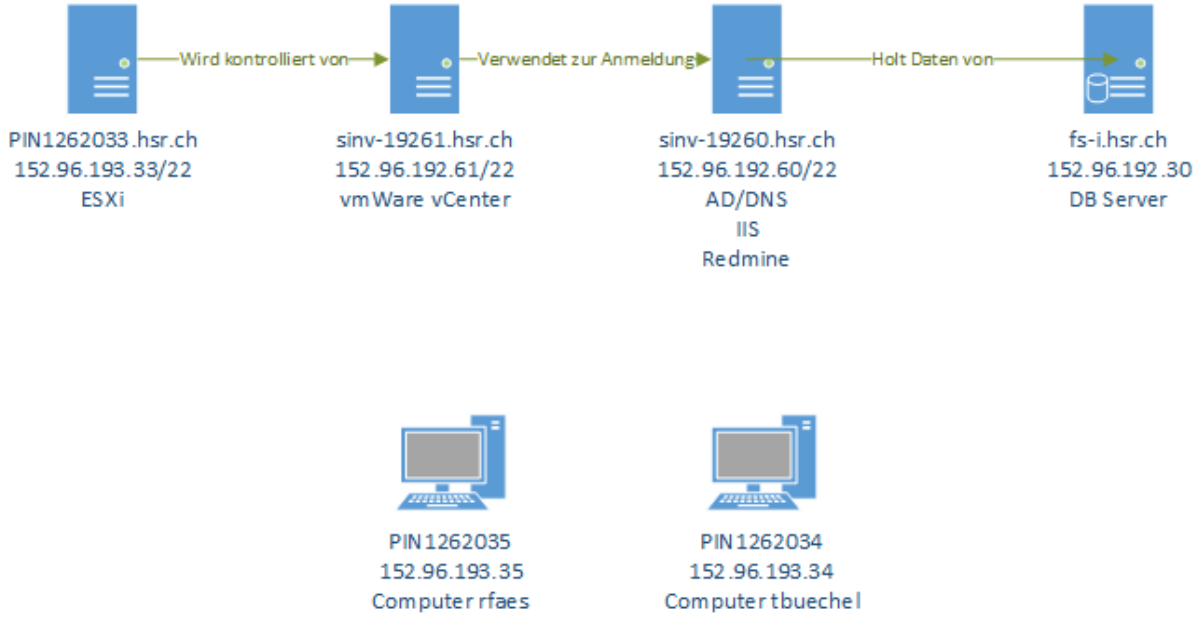

Abbildung 12: Infrastruktur während des Projektes

4.3 Datenbank

4.3.1 Designentscheid „Tables Only“

Wir haben uns gegen die Verwendung von Views und Stored Procedures entschieden, da neben VLCM voraussichtlich keine weiteren Umsysteme Zugriff auf die Daten benötigen. Dies ermöglicht, die gesamte Business-Logik zentral im C#-Code in den Assemblies abzulegen und die Datenbank als reine Datenschnittstelle zu designen (-> **separation of concerns**).

4.3.2 Database model

Das Datenbankmodell ist bis auf eine Ausnahme direkt von den Klassen des Domain Models (siehe Kapitel 2.4.1) abgeleitet: Zwischen **NetworkConfig** und **DeploymentTarget** besteht eine n:m-Beziehung was eine Zwischentabelle verlangt: **DeploymentMapping**.

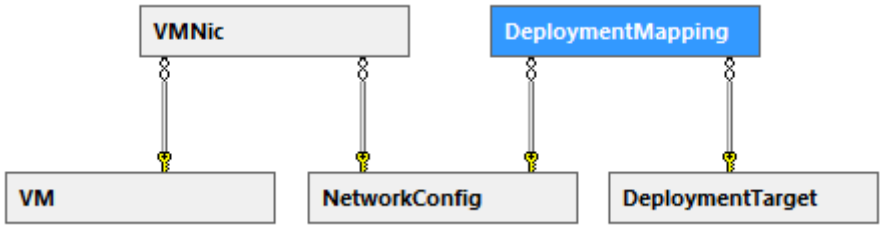

Abbildung 13: Zwischentabelle DeploymentMapping

Auf das UI hat dies keinen Einfluss, sprich erfordert keine zusätzliche View.

4.4 Datenmodell

4.4.1 Assemblies

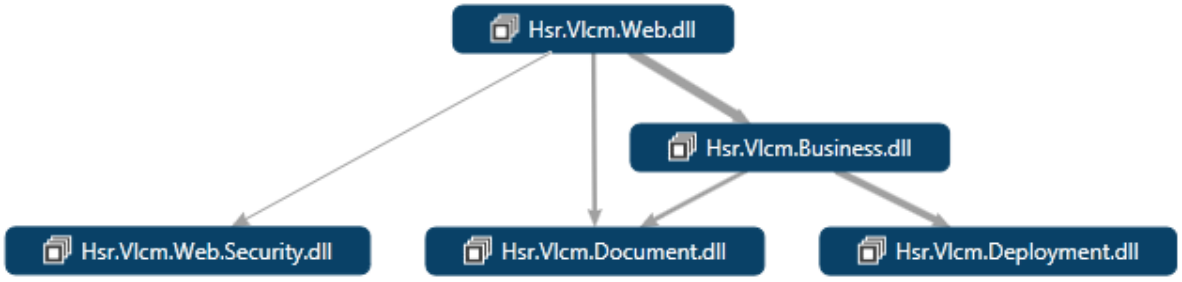

Abbildung 14: Abhängigkeit der Assemblies untereinander

Speziell am Deployment-Assembly ist, dass es zur dynamisch Laufzeit geladen wird und via app.conf-Konfiguration ausgetauscht werden kann.

Assembly	Beschreibung
Hsr.Vlcm.**Web.dll**	GUI, MVC, webpage resources (images, html, css, …)
Hsr.Vlcm.**Business.dll**	Business-Logik Entity Framework **inkl. DAL** (Data Access Layer) E-Mail handling
Hsr.Vlcm.**Document.dll**	document parser property tag converter
Hsr.Vlcm.**Deployment.dll**	VM deployment (und destroy)
Hsr.Vlcm.**Security.dll**	Authentifizierung via AD/LDAP

4.4.1.1 Designentscheid „No DAL“

Wir haben uns gegen einen DAL entschieden. Wir integrieren das Entity Framework in den Business Layer. Der Grund dafür ist, dass ansonsten der DAL nur ein Durchlauferhitzer wäre. Ein weiterer Grund ist, dass sonst die oberen Layer eine Reference auf diesen Layer haben müssten.

4.4.1.2 Designentscheid „Database First“

Zu Beginn des Projektes habe wir uns entschieden, dass wir nach der Ausarbeitung des Domain Models dieses auf die Datenbank übertragen und anschliessend daraus C#-Klassen generieren. Ausschlaggebend für diese Entscheidung war:

- VLCM ist sehr datenlastig.
  - Analyse mit Auftraggeber erfolgt mit Hilfe vom Altsystem, welches mit CSV-Dateien arbeitet. Dies führt zu einem einfacher zu kontrollierendem Mapping und zu mehr Qualität der Datenhaltung.
- Späte Änderungen im Datenmodell sind einfacher umzusetzen.
- Bessere Kontrolle über die Konsistenz der Daten.
- Wenig Erfahrung der Projektmitarbeiter mit „Code First“-Ansatz.

4.4.2 Web.dll

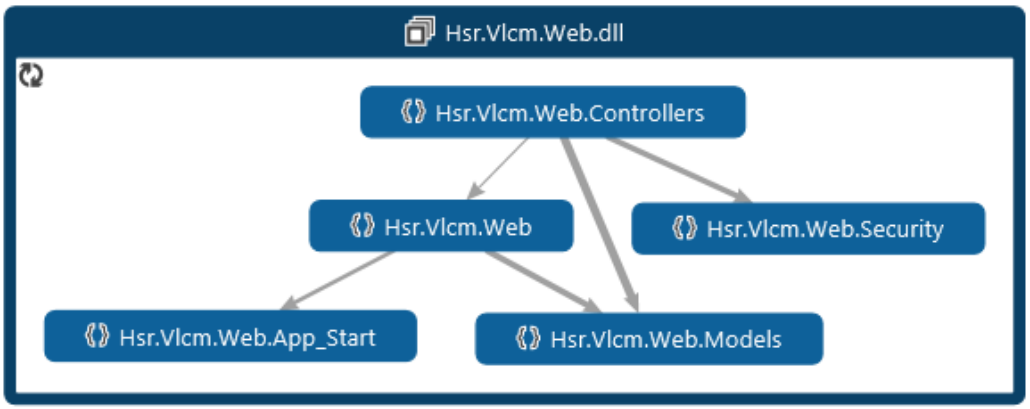

Abbildung 15: Namespaces innerhalb von Web.dll

Namespace	Beschreibung
Hsr.Vlcm.**Web**	Application Start
Hsr.Vlcm.**Web.Controllers**	MVC-Controller
Hsr.Vlcm.**Web.Security**	VlcmAuthorizeAttribute (Berechtigungens-Attribut für Controller)
Hsr.Vlcm.**Web.App_Start**	MVC Initial-Setup (routing, javascript bundles)
Hsr.Vlcm.**Web.Models**	ViewModels

Ausserhalb der Namespaces befinden sich noch die Webresourcen wie z.B. Bilder, CSHTML-Pages und Javascript-Bundles.

4.4.2.1 Designentscheid „ViewModel Wrapper“

Da wir mittels Entity Framework unsere Modell-Klassen generieren lassen, sind diese für die Views nicht geeignet. Um den Views weitere Daten zur Verfügung zu stellen, haben wir uns für eigene Klassen im UI entschieden – das ViewModel. Zu diesem Zweck wurde das *IViewModel* eingeführt, das das Model aus dem EF wrapped (siehe dazu Abbildung 16: ControllerTemplate mit Model-Wrapper IViewModel). Implementationen eines ViewModels können dann spezifische Methoden, Properties und Members definieren (z.B. Combobox-Items).

4.4.2.2 Designentscheid „Generic Controller“

Wie im Domain Model ersichtlich ist, besitzt VLCM einige Stammdaten. Für die meisten werden CRUD-Views benötigt, die sich sehr ähnlich verhalten, was für CRUD-Operation von Stammdaten nicht überraschend ist. Um Redundanzen im Code zu vermeiden bzw. das DRY Prinzip möglichst nicht zu verletzen, wurde eine **abstrakte generische Controller-Klasse** implementiert. Das ControllerTemplate implementiert bereits Insert, Update, Delete und Read Methoden für das ViewModel-Interface (*IViewModel*) über das Business-Interface (*IManager*).

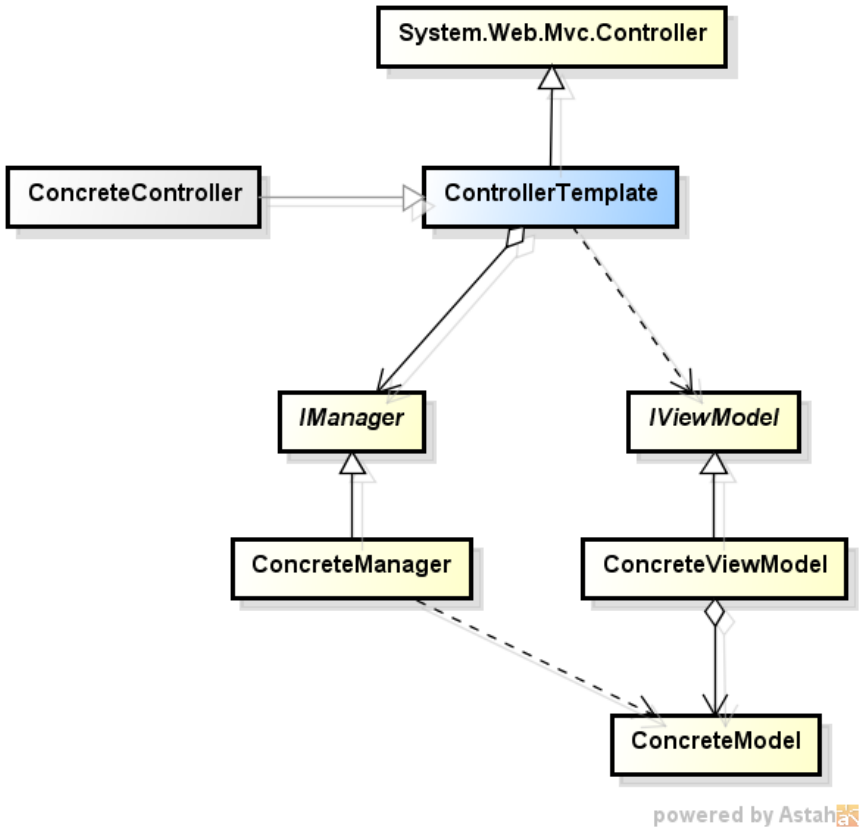

Abbildung 16: ControllerTemplate mit Model-Wrapper IViewModel

Das Pattern ermöglicht die Definition eines Standard-Controllers (ConcreteController) mit einer Zeile Code ohne Flexibilität einzubüssen, da die CRUD-Methoden bei Bedarf einfach überschrieben werden können. Als Beispiel ist hier der Code des **kompletten RoleController**s:

```
public class RoleController : ControllerTemplate<Role, RoleModel, RoleManager>{}
```

Möchte man z.B. die Edit-Methode für einen spezifischen Controller anpassen:

```
public class RoleController : ControllerTemplate<Role, RoleModel, RoleManager>
{
	public override ActionResult Edit(RoleModel model)
	{
		// some additional code...
		_manager.Edit(model.Item);
		return View(model);
	}
}
```

4.4.3 Business.dll

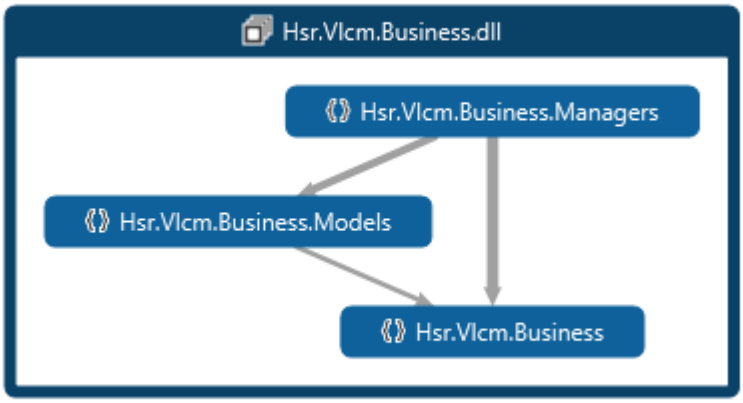

Abbildung 17: Namespace in Business.dll

Namespace	Beschreibung
Hsr.Vlcm.**Business**	Generiertes Model (Klassen) vom Entity Framework MetaData-Erweiterung der Klassen (Validierungs-Attribute)
Hsr.Vlcm.**Business.Managers**	Business-Logik Für jedes ViewModel gibt es einen entsprechenden Manager (z.B. RoleManger, VlcmSettingManager, OrderManager, …)
Hsr.Vlcm.**Business.Models**	Model-Klassen die nicht persistiert werden und somit nicht vom Entity Framework generiert werden: **Order** ➢ Wird vom VmManager in-memory in eine VM transformiert. **DeployModel** ➢ Wird nur temporär für die Ansicht vor dem Deploy und Destroy von VMs benötigt.

4.4.4 Document.dll

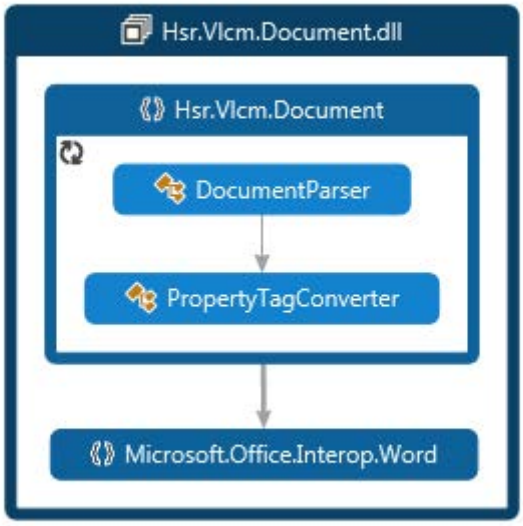

Abbildung 18: Namespaces und Klassen von Document.dll

Namespace	Beschreibung
Hsr.Vlcm.**Document**	Parsing-Logik für Dokumente, welche via E-Mail verschickt werden
Microsoft.Office.Interop.Word	Externe Interoperability-Bibliothek für die Manipulation von Word-Dokumenten

4.4.5 Deployment.dll

Das Assembly ist Zuständig für den **Deploy- und Destroy**-Prozess der virtuellen Maschinen.

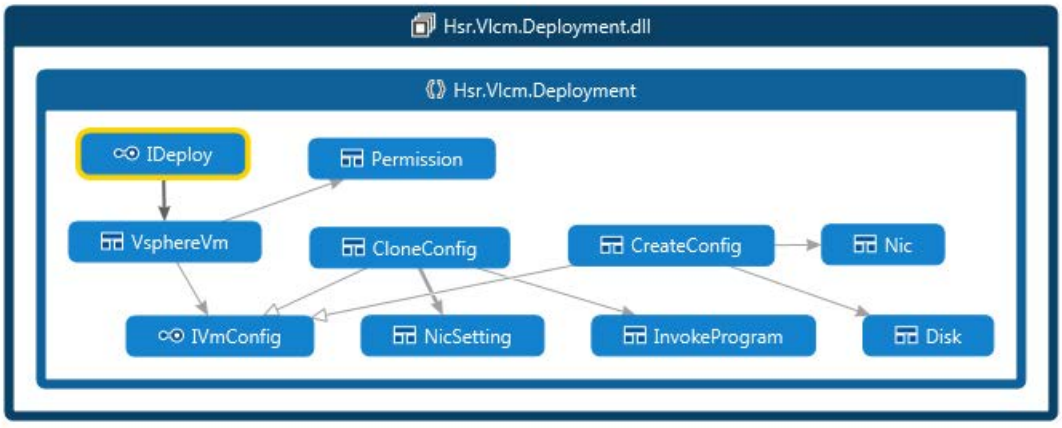

Abbildung 19: Klassen von Deployment.dll

Klasse/Interface	Beschreibung
IDeploy	Schnittstelle für Business-Logik in Business.dll
VsphereVm	Daten-Klasse (Parameter) für Deploy/Destroy-Methode
IVmConfig	Schnittstelle für VM-Config-Details der virtuellen Maschine
CreateConfig	Konkrete Konfiguration für „leere“ VM (kein Template/OS)
CloneConfig	Konkrete Konfiguration für VMs mit Template
InvokeProgram	Script, das nach dem Deploy auf der virtuellen Maschine ausgeführt wird. Dient zum Zweck der Konfiguration, das über vSphere-Möglichkeiten hinausgeht. Primär für Linux-Systeme.
Permission	Zugewiesene Benutzer (Access)
NicSetting/Nic/Disk	Network-Interface- und Disk-Konfiguration für VM

4.5 vSphere-Schnittstelle

4.5.1 Designentscheid „Laden der Library zur Laufzeit “

Damit der Deploymentprozess einer VM auch auf neueren Version von vSphere oder auch über andere Libraries geschehen kann, wird die Deploymentlibrary zur Laufzeit geladen.

Dazu muss die Library das IDeploy Interface implementieren.

4.6 Active Directory-Schnittstelle

4.6.1 Designentscheid „Verwendung von ActiveDirectoryMembershipProvider“

Zur Authentifizierung wird der ActiveDirectoryMembershipProvider von Microsoft verwendet. Dieser benötigt für die Überprüfung der Loginformationen lediglich den Connectionstring zum LDAP und attributeMapUsername.

Für die Verwendung des AD FS hat sich unser Industriepartner enstschieden.

4.6.2 Designentscheid „Verwendung von ActiveDirectoryRoleProvider“

Zur Überprüfung welche Rollen eine Person besitzt wurde die Abstrakte Klasse „RoleProvider“ implementiert. Diese überprüft die Gruppenzugehörigkeit mithilfe von LDAP Queries. Die Verwendung der Klasse „ System.DirectoryServices.AccountManagement.UserPrincipal“ hat nicht funktioniert. Das Active Directory warf eine Exception beim Versuch die „Security Groups“ zu holen. Die Funktion „Groups“ konnte aufgerufen werden, doch diese lieferte nicht immer alle Gruppen einer Person.

Die Gruppenzugehörigkeit einer Person kann über das Property „memberof“ geladen werden. Dieser Aufruf liefert jedoch nicht im jeden Fall die Gruppen eines Benutzers.

Daher wurde die Überprüfung der Gruppenzugehörigkeit aus Sicht der Gruppen, welche die Webapplikation verwenden können, gemacht. Es wird überprüft, ob die Person direkt in dieser Gruppe ist oder in einer Untergruppe. Um Zyklen zu verhindern werden die bereits besuchten Gruppen temporär gespeichert und gegebenenfalls ignoriert.

4.6.3 Designentscheid „Vorschläge während der Eingabe“

Für Bestellung der Virtuellen Maschinen müssen Personen konfiguriert werden, welche Zugriff benötigen. Um die User Experience des Bestellers zu verbessern schlägt das System Benutzernamen vor sobald mindestens zwei Buchstaben eingegeben wurden. Da der Zugriff auf das Active Directory teuer ist, haben wir uns dafür entschieden die Vorschläge erst ab zwei Buchstaben anzuzeigen, da ansonsten zu viele Daten aus dem Active Directory geladen werden.

4.7 MVC

4.7.1 Designentscheid „Validierung mit jquery.validate.unobtrusive“

Auf dem Client und dem Server sollen die eingegebenen Daten des Benutzers validiert werden. Damit diese Validierung nicht zweimal implementiert werden muss, kann das jquery.validate.unobtrusive auf Client Seite und das Modelstate.IsValid auf der Serverseite verwendet werden.

Damit auf Client und Serverseite validiert werden kann, müssen auf den Properties der Models Data Annotations gesetzt werden. Ob diese Annotations eingehalten wurden, wird mit dem Modelstate.IsValid überprüft.

Für die Clientseite werden aus den Annotations HTML5 Data Attribute erzeugt. Diese verarbeitet das jquery.validate.unotrusive und fügt die entsprechenden Regeln der jquery validation hinzu.

Dank diesem Designentscheid, kann an einem Ort für beide Seiten der Applikation die Validierung durchgeführt werden.

Das jquery.validate.unotrusive führt die Umwandlung der HTML5 Data Attribute in die jquery.validation Regeln um sobald das Document ready ist. Dies hat zur Folge, dass nachträglich eingefügte Eingabefelder nicht mehr Clientseitig validiert werden.
Damit die Umwandlung nochmals geschieht, muss als erstes der Form die Validations- und die UnobtrusiveValidations-Daten entfernt werden. Anschliessend kann der Unobtrusive Parser nochmals gestartet werden.

```
$("#form").removeData("validator").removeData("unobtrusiveValidation");
$.validator.unobtrusive.parse("#form");
```

4.7.2 Designentscheid „ConvertObjectToJson anstelle von Ajax“

Das von der Web Applikation verwendete Jquery Datatables , welches zur Anzeige und Bearbeitung der Daten dient, kann die Anzeigedaten entweder direkt per Ajax oder per Javascript erhalten. Da das anzuzeigende Model die entsprechenden Daten enthält, welche für die Anzeige der Datentabelle benötigt wird, verwenden wir einen von uns geschriebenen ConvertObjectToJson Funktion, welche die entsprechenden Daten umwandelt und direkt als Json der Tabelle anfügt.

So wird nicht nur die Anzahl der Anfragen an den Server minimiert, sondern auch das Verhalten hat sich Sicht des Benutzers verbessert. Wenn der Benutzer Daten dieses Model verändert und die geänderten Daten abspeichern will, und Serverseitig gibt es einen Validierungsfehler, gehen so die Eingegebenen Daten in der Tabelle nicht verloren. Denn diese werden an die Action des Controllers gesendet. Dort wird überprüft, ob die Daten gültig sind. Falls nicht, wird die gleiche View mit den veränderten Daten wieder erstellt. Da nun aus den bearbeiteten Daten das Json erstellt wird, sind die geänderten Daten nicht verloren, sondern in der Tabelle als hidden inputs hinterlegt.

So können wir eine Stateless Webapplikation garantieren, denn der Browser hält alle Informationen, welche für das Editieren der Daten benötigt werden.

4.7.3 Designentscheid „Kein Lazyloading bei grossen Datenmengen“

Lazyloading wird dazu verwendet um Daten nachladen zu können wenn diese gebraucht werden. Dies funktioniert für kleine Mengen sehr gut. Doch sobald die Datenmenge zunimmt leidet die Performance unter dem ständigen nachladen der Daten stark.

Die Daten können daher per „eager loading“ geladen werden. Das bedeutet, dass die Daten bereits im vorraus aus der Datenbank geladen werden und nicht erst wenn diese gebraucht werden. Dies verringert die Anzahl der Datenbankzugriffe erheblich. Daher wurde die Möglichkeit implementiert, die Daten welche angezeigt werden sollen, direkt per „eager loading“ zu laden.

In der jquery Datatables muss dazu nur der Pfad zum entsprechenden Property eingegeben werden.

```
"aoColumns": [
                    {
                        "sName": "Vm.Purpose.ShortName",
                        "mData": "Vm_Purpose_ShortName",
                    },

…
```

Mit Hilfe dieser Notation weiss der Controller, dass ein VM Objekt angezeigt wird. Für jedes Objekt wird anschliessend das Purpose Properties per „eager loading“ geladen. Angezeigt wird schlussendlich der ShortName dieses Properties.

Auf den Navigation Properties muss für diesen Zweck das „IncludeNavigationPropertyAttribute“ vorhanden sein. Dieses beinhaltet die Information, welches Include für dieses Property geladen werden muss.

Sobald alle Properties, welche geladen werden müssen vorhanden sind, wird die Funktion GetItems aufgerufen. Diese deaktiviert das LazyLoading und ruft die von uns geschriebene Funtion „Includes“ auf. Diese liest den Inhalt der IncludeNavigationPropertyAttribute aus und lädt das Navigation Property.

```
public IEnumerable<Vm> GetItems(IEnumerable<string> includes)
{
   _entities.Configuration.LazyLoadingEnabled = false;
   return entities.Vms.Includes(includes);
}
```

5 Fazit

5.1 Tobias Büchel

Die Studienarbeit war eine gute Chance, mich intensiver in die .NET-Welt von Microsoft einzuarbeiten. Raphael Faes war mir mit seinem Wissen in technischen Fragen immer eine grosse Unterstützung und ich konnte so viel über Technologien wie MVC und Entity Framework lernen. Es wurde aber schnell klar, dass für reine Einarbeitungstätigkeiten der Umfang der Arbeit zu gross war. Die Devise hiess „Learning-By-Doing“.

Die Analyse des Problems wurde durch Vorbereitung des Auftraggebers sehr gut unterstützt, so erhielten wir z.B. ein Flussdiagramm mit dem aktuellen Deploy- und Destroy-Prozess und welche Stellen zu optimieren waren. Vorteilhaft war ausserdem, dass Oliver Rehmann an der HSR arbeitet und so ein enger Kontakt zu unserem Kunden möglich war. So war Herr Rehmann mehrmals die Woche an unserem Arbeitsplatz zu finden, was uns regelmässiges Feedback ermöglichte.

Interessante Aspekte hatte die Arbeit viele. Am besten gefiel mir die Tatsache, dass wir ein sauberes abgeschlossenes System, vom User-Interface bis zur Business-Logik implementieren konnten und nicht nur ein Teilsystem entwickelt haben.Wir hatten viel Freiheiten und durften uns im User-Interface auch kreativ entfallten – so habe ich z.B. erfolgreich das Card-Sorting aus den User-Interface-Modulen ausprobieren können, was sonst nur in Theorie bekannt war.

Meiner Meinung nach hatte es zwei Gründe, warum wir am Schluss zu wenig Zeit hatten:

Erstens wurde der Umfang des Domain-Models unterschätzt. Mit 19 Tabellen ist die Webapplikation nicht mit einer „besseren CD-Verwaltungssoftware“ zu vergleichen. Die Objekte haben ausserdem viele logische Abhängigkeiten untereinander, welche zum Teil erst in der Mitte der Construction-Phase im Gespräch mit dem Auftraggeber aufgetaucht sind. Das führte zum Beispiel dazu, dass wir in der Projektwoche 8 noch Änderungen am Domain Model durchführen mussten. Ich sehe dass nicht als Fehler den wir begangen haben oder als Vorfurf an den Auftraggeber, sondern vielmehr als Veranschaulichung, wie komplex das Thema im Detail wurde. Die vielen Gespräche, die wir mit Oliver Rehmann hatten, waren einfach nötig, um auf ein stabiles Datenmodell zu kommen.

Zweitens haben Unterschiede zwischen Entwicklungs- und Produktivumgebung zu mehreren Problemen in der Integrationsphase, welche bereits nach hinten verschoben werden musste, geführt. So konnte das System leider nicht wie gewünscht mit allen Features eingeführt werden, was aus unserer Sicht ein bisschen enttäuschend ist. Dennoch möchten wir das nach der offiziellen Abgabe der SA mit Herr Rehmann nachholen, damit unsere Software wie gewünscht produktiv in Einsatz kommt.

An dieser Stelle ein herzliches Dankeschön an die Betreuer und Projektmitarbeiter für die wertvolle Zeit und die vielen Erfahrungen die ich machen durfte.

5.2 Raphael Faes

Mein persönliches Fazit dieser Arbeit ist grundsätzlich positiv. Ich durfte mich intensiv mit der Erstellung einer modernen Web Applikation beschäftigen und konnte mein Wissen von der Datenbank bis zum UI in vollen Zügen ausreizen. Ich war positiv überrascht, als ich feststellen durfte, wie viel Wissen ich in meiner Studienzeit aneignen konnte. Sei es für die Verwendung von Patterns für das Lösen von komplexen Problem oder ob es für das Erkennen der Problemdomäne war.

Mit Wehmut muss ich eingestehen, dass nicht alles so gelaufen ist, wie ich mir das gewünscht hätte. Der Umfang der Arbeit wurde von mir zu Beginn der Semesterarbeit stark unterschätzt. So wurden zum Beispiel aus ein paar wenigen Objekten in der Datenbank 20 Objekte von welchen, alle direkt oder indirekt miteinander für den Deplyomentprozess benötigt werden.
So haben wir mehr Zeit aufwenden müssen um unsere Fehleinschätzung ausgleichen zu können, doch selbst mit diesem Mehraufwand haben wir es nicht geschafft alle Features zu implementieren.

Wir haben von der Abteilung I der HSR eine Testumgebung erhalten, welche die wichtigsten Komponenten für das Deployen von VMs auf eine vSphere-Umgebung ermöglicht. Unsere Tests und die Web Applikation an sich haben wunderbar mit dieser Umgebung funktioniert. Leider verhielt sich die Testumgebung nicht gleich wie die aktive Umgebung.
So funktionierte das Auslesen der Gruppen, in welcher der angemeldete Benutzer ist, wunderbar in der Testumgebung, in der Produktivumgebung funktionierte dies aus unerfindlichen Gründen nicht mit jedem Benutzer. Nachdem mehrere Lösungsansätze in der Produktivumgebung getestet wurden, konnte eine Lösung gefunden werden.
Dies hat die Tatsache, dass wir im Zeitplan hinterher hinken noch verschlimmert.

Zu guter Letzt hat dann auch noch der Benutzer Probleme verursacht, welcher für das Deployen der VMs verwendet wird. Wir können uns bis zum Zeitpunkt der Abgabe der Semesterarbeit nicht mit diesem Benutzer an das vSphere anmelden. Ob dies an der vSphere oder an der bereits Kopfschmerzen bereitenden Active Directory liegt, können wir auch nach Untersuchungen nicht beurteilen. Wollten wir dieses Problem genauer ansehen, bräuchten wir mehr Zeit, welche uns leider nicht zur Verfügung steht.

Ich möchte mich besonders für die tolle Unterstützung durch unseren Industriepartner bedanken. Er hat uns durch die ganze Arbeit tatkräftig unterstützt. Nicht nur die Testumgebung hat er für uns aufgebaut sondern war auch immer für uns da falls wir Fragen bezüglich der vSphere Umgebung hatten.

6 Eigenständigkeitserklärung

6.1 Tobias Büchel

Ich erkläre hiermit,

- dass ich die vorliegende Arbeit selber und ohne fremde Hilfe durchgeführt habe, ausser derjenigen, welche explizit in der Aufgabenstellung erwähnt ist oder mit dem Betreuer schriftlich vereinbart wurde,
- dass ich sämtliche verwendeten Quellen erwähnt und gemäss gängigen wissenschaftlichen Zitierregeln korrekt angegeben habe.
- das ich keine durch Copyright geschützten Materialien (z.B. Bilder) in dieser Arbeit in unerlaubter Weise genutzt habe.

Ort, Datum:

Name, Unterschrift:

6.2 Raphael Faes

Ich erkläre hiermit,

- dass ich die vorliegende Arbeit selber und ohne fremde Hilfe durchgeführt habe, ausser derjenigen, welche explizit in der Aufgabenstellung erwähnt ist oder mit dem Betreuer schriftlich vereinbart wurde,
- dass ich sämtliche verwendeten Quellen erwähnt und gemäss gängigen wissenschaftlichen Zitierregeln korrekt angegeben habe.
- das ich keine durch Copyright geschützten Materialien (z.B. Bilder) in dieser Arbeit in unerlaubter Weise genutzt habe.

Ort, Datum:

Name, Unterschrift:

7 Anhang

7.1 Table of Figures