AdminTool Designstudie

ABSCHLUSSBERICHT

Software Projekt an der Universität Zürich, 2011/2012

Studenten: André Meyer, Andreas Albrecht, Claudio Anliker

Betreuer: Michael Böhlen

Universität Zürich[UZH]

MIT INNOVATION AG
Ein Unternehmen der MIT-GROUP

INHALTSVERZEICHNIS

PROJEKT-SETTING

Arbeitgeber:

MIT Innovation AG (ein Unternehmen der MIT-GROUP), www.mit-group.ch

Projektleiter seitens der MIT Innovation AG:

Linard	Moll	LM	Senior Consultant (MSc Software Systems)

Projekt-Mitarbeiter:

André	Meyer	AM	BSc Student Software Systems	s09-736-398
Andreas	Albrecht	AAL	BSc Student Software Systems	s09-722-067
Claudio	Anliker	CA	BSc Student Software Systems	s09-713-207

Dauer:

28.11.2011 - 23.03.2012 - 6.5 Arbeitswochen

AUFGABENSTELLUNG

ZIELE & ZIELDEFINITION

Das Ziel des Praktikums ist die Erstellung von Designvorgaben anhand eines neuen Oberflächen-Designprototypen der bestehenden und produktiv genutzten Administrationslösung der MIT-GROUP. Das Projekt findet im Rahmen eines Software-Projektes (Praktikum) der Universität Zürich statt.

Dazu müssen:

1. Die aktuelle Applikation (näher) bzw. die Arbeitsweise der Stakeholder (grob) analysiert und Optimierungsmöglichkeiten gefunden werden.
2. Die Möglichkeiten des .NET 4 Grafikframeworks Window Presentation Foundation (WPF4) analysiert werden.
3. Rapidprototyping einer neuen Oberfläche anhand der vorgängig erstellten Anforderungsanalyse mit Iterationen und direktem Stakeholderfeedback. Dazu werden vorgängig Testdaten zur Verfügung gestellt.

Für den Design-Entwurf steht die Verwendung von gängigen Design Patterns im Vordergrund. Eine von uns definierte Designvorschrift soll die Erkenntnisse für die weitere Entwicklung der Applikation sichern.

MEILENSTEINE

Domäne/ Unterfunktion	Position	Funktion/Aufgabe	Soll/Must
Einarbeitung	E 1.1	Bestehende Applikation	Must
	E 1.2	Stakeholderanalyse, - Erstkontakt	Must
	E 1.3	Installation der Arbeitsumgebung, Zugang Code-Repository	Must
	E 1.4	Einarbeitung in WPF4, Frameworks, Patterns	Must
Iterationen	I 1.1	Kleinere Technikprototypen entwerfen, Patterns sammeln, umsetzen. Validierung im Team	Must
	I 1.2	Designprototyp entwerfen, Usability Analyse, Validierung im Team	Must
	I 1.3-1.5	Designstudie mit Stakeholder durchführen, Analyse, Verbesserung	Must
	I 1.6	Finaler Prototyp steht, Design-Guidelines finalisieren, Dokumentation	Soll
Abschluss	A 1.1	Aufräumarbeiten (Design-Guideline abschliessen, Cleanup Code-Repository)	Must
	A 1.2	Präsentation der Ergebnisse vor MIT Innovation AG	Must
	A 1.3	Präsentation der Ergebnisse in Form eines Berichtes (UZH)	Soll

EINFÜHRUNG ADMINTOOL

Das AdminTool der MIT-GROUP ist eine Applikation, welche von der MIT-GROUP über mehrere Jahre hinweg entwickelt wurde. Es wird intern für die Verwaltung der Kunden sowie deren Produkte und Aufträge verwendet. Gleichzeitig dient das Tool auch der internen Zuweisung von Aufträgen an die Mitarbeiter (Supporter). Entsprechend ist ein Ziel der Applikation, dass alle Aufträge zentral erfasst und gesteuert werden können. Zugleich sind alle Informationen zu einem Auftrag schnell erreichbar und klar ersichtlich: Von wem der Auftrag kommt, wer ihn bis wann bearbeitet und welche Aktionen in Vergangenheit im Zusammenhang mit diesem Auftrag getätigt wurden. Weitere Informationen über einen Auftrag, wie beispielsweise die Zuordnung von Produkten zu den Kunden und Benutzern, sind ebenfalls über das AdminTool zugänglich.
Das AdminTool wird zurzeit nur intern verwendet. Allerdings gibt es eine Schnittstelle zu einem Webinterface, welches den Kunden erlaubt, neue Aufträge zu erstellen sowie den Status ihrer bestehenden Aufträge zu verfolgen.
Beim Software Projekt geht es vor allem um die Desktop-Applikation. Deshalb liegt der Fokus dieses Berichts auf dem AdminTool. Das Webinterface wird in diesem Bericht nicht näher behandelt.
Neben den Kunden und dem Service Desk (Supporter) verwendet auch die Betriebsleitung sowie die Administration das AdminTool. Die Betriebsleitung nutzt die Applikation, um die Auslastung der Supportmitarbeiter zu überwachen und die Aufträge entsprechend der individuellen Auslastung zuzuweisen. Die Administration kann anhand der vorhandenen Daten verschiedenste Reports erstellen, um die Abrechnung für den Kunden zu generieren und zu prüfen.

ABLAUF DES PRAKTIKUMS

EINARBEITUNG

Unser Praktikum bei der MIT-GROUP in Schwyz begann am 28. November 2011 mit unserem ersten Arbeitstag. Unser Betreuer, Herr Linard Moll, zeigte uns zu Beginn den Umfang unseres Projektes und erklärte unseren Auftrag. Zeitgleich haben wir unsere Arbeitsplätze eingerichtet, um so bald wie möglich mit der Arbeit beginnen zu können. Dazu gehören neben der Konfiguration der Entwicklungsumgebung (in unserem Fall Visual Studio 2010) auch die Installation diverser Add-Ons, ein gemeinsames Repository inkl. SVN zur Versionskontrolle, OneNote-Notizbücher zur Dokumentation und zum Informationsaustausch und die Konfiguration eines E-Mail-Kontos in der Firma.

Die ersten Arbeitswochen waren wir damit beschäftigt, die Grundlagen der Programmierung mit der Windows Presentation Foundation (WPF) zu erlernen, mit welcher der AdminTool-Prototyp später implementiert werden sollte. In dieser Zeit programmierten wir einen RSS-Feed-Reader, um die erlernten Grundlagen zeitgleich praktisch anzuwenden. Die Lernphase zog sich über das ganze Projekt hin, da wir grundsätzlich jedes Feature zum ersten Mal entwickelten und ständig neuen Herausforderungen gegenüber standen. In den kommenden Wochen wurden unsere Programmierfähigkeiten in zwei weiteren technischen Kurztasks überprüft.

RSS-FEED-READER

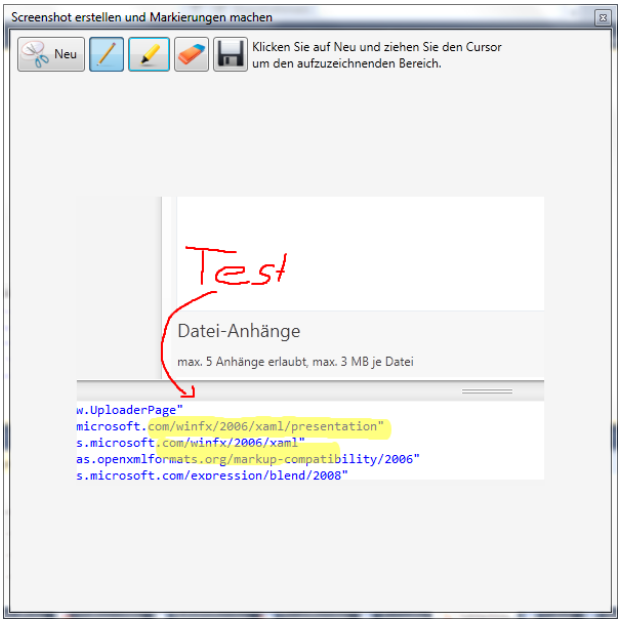
Der RssFeedReader war die erste Applikation, abgesehen von den zahlreichen Tutorials, die wir zusammen programmiert haben. Es handelte sich um einen Wegwerf-Prototypen, in dem wir die bis dahin erlernten WPF-Grundlagen und Designprinzipien umsetzten. Die Applikation wurde mit dem Model-View-ViewModel-Pattern (MVVM) implementiert und enthält verschiedene WPF-Komponenten wie Commands, VisibilityConverter, DataTemplates und selbst definierte UserControls. Der RssFeedReader half uns, das Konzept des Bindings von Properties aus dem C#-Code mit der in XAML definierten GUI zu verstehen. Auf diesem Weg wird die GUI mit der Logik der Applikation verknüpft. Der RssFeedReader selbst war in seiner Funktion nicht sehr umfangreich. Er erlaubte es, manuelle Feeds zu erfassen und zu entfernen, die gelieferten Beiträge anzuzeigen und bei Bedarf durch eine integrierte Suche zu filtern. Eine Statusbar am unteren Ende zeigt bei vielen Feeds und entsprechenden Verzögerungen den Ladefortschritt an.

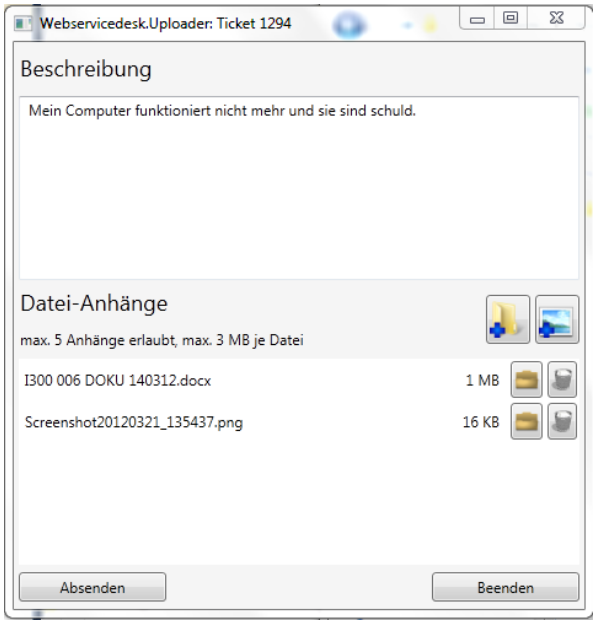
KURZTASK 1

Im Rahmen des Kurztask 1 implementierten wir eine kleine Applikation, die Daten aus einer XML-Datei importiert und in Tabellenform darstellt. Ein Hauptziel dieses Tasks war es, grosse Datenmengen ohne Performanceeinbussen zu laden und zur Laufzeit noch während dem Ladevorgang dynamisch zu filtern (Das Ziel dabei ist, dass die Applikation stets ansprechbar ist und auf Eingaben reagiert). Aus diesem Grund beschäftigten wir uns im Rahmen dieses Tasks mit Multithreading in WPF und lagerten den Datenimport aus dem Hauptthread in einen Background-Thread aus, damit die Performanz der Interaktion zwischen der GUI und dem Benutzer nicht beeinträchtigt wird. Des Weitern beschäftigten wir uns mit der Serialisierung und Deserialisierung, weil die Daten aus dem XML-File in C#-Objekte konvertiert werden müssen. Neben den bisherigen Komponenten wurden zusätzlich eigene EventArgs und DependencyProperties verwendet.

KURZTASK 2

Der Kurztask 2 bestand aus einem kleinen Tool, dass es erlaubt, über einen Dialog dem Webservicedesk eine Support-Anfrage zu schicken. Neben einer Textnachricht ist es auch möglich, einige Daten anzuhängen oder direkt per Klick auf einen Button und via Maus-Drag einen Screenshot zu erfassen. Nachdem der Screenshot angefertigt wurde, kann er mit einem gelben Marker oder einem roten Stift verändert werden, um beispielsweise einen Fehler zu markieren. In dieser Applikation wurden zum ersten Mal DelegateCommands verwendet.

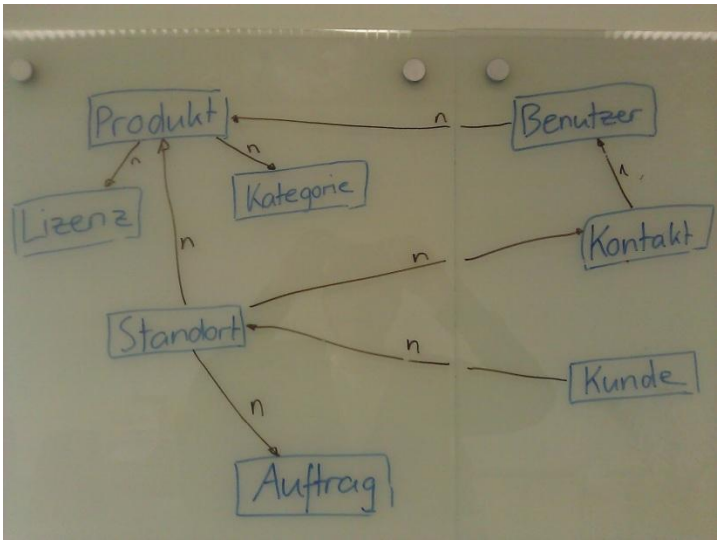




DESIGNSTUDIE

Ein Hauptziel des Software Projekts war die Neugestaltung und Optimierung der grafischen Oberfläche des AdminTools. Nach der erfolgreichen Einarbeitung in die zu verwendenden Technologien konzentrierten wir uns auf das bestehende AdminTool. Um die Verwendung und die Usability beurteilen zu können, war es wichtig, das bisherige Tool kennenzulernen. Zu diesem Zweck wurde eine Testumgebung mit einer AdminTool Instanz aufgesetzt, welche uns zur Verfügung stand.
Die Applikation wurde von uns systematisch erforscht und die einzelnen Geschäftsabläufe wurden durchgespielt, um ein Gefühl für die Applikation und den Workflow zu bekommen. Danach wurden auf Grund der ersten Erfahrungen die verschiedenen Entitäten der Applikation und ihre Beziehung zueinander skizziert (siehe Abbildung). Dabei wurde auch die Funktionalität der Applikation dokumentiert (Inventar, IST Analyse). Bei der Benutzung des Tools zeigten sich uns erste unkomfortable Navigationsvorgänge und teils eine ungewöhnliche Usability. Unter Berücksichtigung der ersten Erkenntnisse wurde dann erste Änderungen diskutiert und ein erster GUI Vorschlag ausgearbeitet (ohne Funktionalität, Designprototyp). Zudem wurden alle Änderungsvorschläge und Ideen mit dem Betreuer diskutiert.
Nachdem die Einarbeitung in das AdminTool abgeschlossen war, konzentrierten wir uns auf die Stakeholderanalyse. Dazu haben wir uns in einem ersten Schritt individuell theoretisches Wissen über Interviewtechniken und Requirements-Engineering angeeignet und dieses anschliessend zusammengetragen. In einem nächsten Schritt wurde daraus ein systematischer Ablauf für die Befragung der verschiedenen Stakeholdergruppen ausgearbeitet. Dazu gehört einerseits ein Fragebogen, andererseits die

verschiedenen Bereiche, die wir mit der Analyse abdecken wollten (SOLL Analyse, Designstudie). Die Interviews wurden danach jeweils mit mehreren Personen pro Stakeholdergruppe durchgeführt. Dabei wurden einerseits spezifische Fragen gemäss unserem geplanten Interview bzw. dem Fragenkatalog gestellt. Andrerseits konnten sich die Personen auch frei zur Applikation äussern und uns ihr Verhalten bzw. ihren Workflow demonstrieren. Alle Informationen wurden entsprechend protokolliert und danach für die weitere Verwendung zusammengefasst.



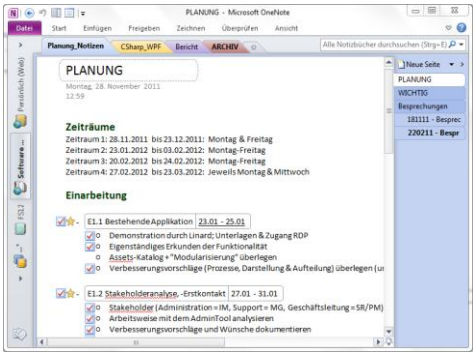
Eine detailliertere Beschreibung zum Vorgehen allgemein sowie spezifische Details zur Designstudie und den Erkenntnissen befinden sich in Kapitel „Designstudie“.

IMPLEMENTIERUNG

Nach dem Erlernen der WPF-Grundprinzipien und ihrer Anwendung wurde der Fokus auf das Framework Prism gelenkt, dessen Grundidee und Funktionsweise im Kapitel genauer erläutert wird. Danach wurde mit der Implementierung des AdminTools begonnen. Da es sich um einen evolutionären Prototyp handelte, wurde schon zu Beginn grossen Wert auf eine saubere Programmierung und ein stabiles Architektur-Design gelegt, um den Aufwand bei Änderungen und Erweiterungen möglichst gering und die Architektur übersichtlich zu halten. Aus diesem Grund wurde die Implementierung oft überarbeitet, Code refakturiert und verbessert. Der letzte grosse Teil war dann das Anbinden des Prototyps an Versuchsdaten durch das Entity Framework. Dieser Teil zeigte uns, wie gut das AdminTool im Umgang mit den realen Daten funktioniert und erlaubte es, einige Formulare vollständig zu implementieren und zu testen.

DOKUMENTATION

Während unserem Software Projekt wurden alle relevanten Änderungen, Erkenntnisse und Fortschritte mit Microsoft OneNote dokumentiert. Diese Software erlaubt es, ein Online-Notizbuch zu führen und zu teilen, so dass wir zu dritt an den gleichen Orten Notizen hinterlegen konnten. Verschiedene, nach Themen getrennte Notizbücher erlaubten es, uns über die aktuellen Implementierungstasks, unsere technischen Kenntnisse und den momentanen Planungsstand auszutauschen. Durch die regelmässige Erfassung von wichtigen Informationen in diesem Tool wurde es am Schluss auch einfacher, die Entwicklerdokumentation und den Abschlussbericht zu schreiben.



ABSCHLUSS

Der Abschluss des Praktikums bestand in der technischen Dokumentation, deren primären Ziele es sind, künftigen Entwicklern einen guten Überblick über die entstandene Software zu geben und einen einfachen Einstieg in die folgende Weiterentwicklung zu ermöglichen. Dabei werden jedoch die Grundlagenkenntnisse von WPF, sowie eine selbstständige Einarbeitung in die Frameworks Prism und Entity Framework vorausgesetzt.

DESIGNSTUDIE

Basierend auf den in Kapitel xx definierten Zielen, ist ein Hauptziel die Optimierung und Neugestaltung des Oberflächendesigns des AdminTools. Deshalb mussten wir zunächst den IST-Zustand erfassen und dokumentieren. Dabei tauchten bereits erste Erkenntnisse zur Bedienung, Ideen zu möglichen Optimierungen und Ideen auf, welche laufend notiert worden sind. Basierend auf diesen Erkenntnissen, haben wir dann einen ersten kleineren Designprototypen in Microsoft PowerPoint erstellt. Dieser Prototyp diente dann bei der anschliessenden Designstudie als Hilfestellung, damit die Fragen detaillierter und gezielter gestellt werden konnten. Die Bedürfnisse und Vorschläge der entsprechenden User-Anspruchsgruppen haben wir dann systematisch erfasst, verglichen und dokumentiert. Anschliessend konnte ein entsprechender SOLL-Zustand vereinbart werden. Diesen Zustand versuchten wir während des Projektes bestmöglich zu implementieren. Im Kapitel Zielerreichung wird dargelegt, wie gut uns das gelungen ist.

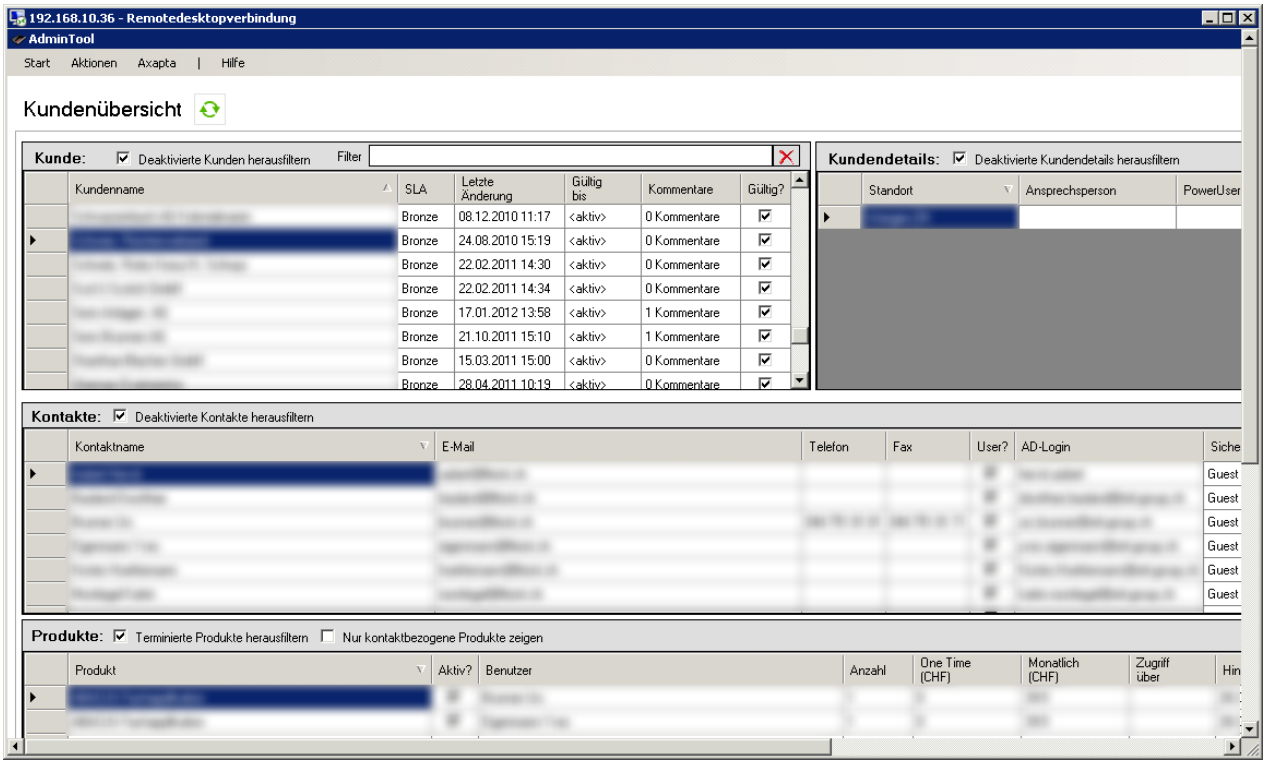
IST-ANALYSE

Um die Designstudie bzw. das Endergebnis möglichst optimal zu gestalten, haben wir zuerst eine umfassende Analyse des IST-Zustandes des AdminTools durchgeführt. Untersucht und dokumentiert wurden dabei die einzelnen Funktionen und Formulare und deren Kategorisierung (Eingliederung). Des Weiteren haben wir Abläufe, Einstellungsmöglichkeiten, Verbindungen und Verknüpfungen zwischen den einzelnen Funktionen festgestellt und ebenfalls dokumentiert, mit der Absicht, alle bestehenden Funktionen auch in der neuen Version des AdminTools beizubehalten und die Bedienung oder Anordnung gegebenenfalls zu optimieren.

Wie bereits erwähnt, war ein zentrales Ziel, die bestehenden Funktionen auch weiterhin anzubieten, deren Nutzung aber zu erleichtern und intuitiver zu gestalten. Während der Analyse sind folgende Verbesserungsmöglichkeiten aufgefallen (priorisiert):

Kategorie	Titel	Beschreibung	Prio
Allgemein	Anzeige auf kleinen Bildschirmen	Loggt sich ein User mit einer Auflösung ein, die kleiner als 1440x900 ist, erscheint eine Meldung, dass das AdminTool nicht auf diese Bildschirmgrösse optimiert dargestellt werden kann. Wenn möglich, sollte die Anzeige auch für kleine Bildschirme optimiert werden	3
	Weg-zum-Ziel	Teilweise ist es etwas umständlich bzw. unklar, eine Funktion bzw. einen Task zu bedienen, da der Ablauf kompliziert ist. Das Ziel ist, dass die wichtigsten Tasks in wenigen Klicks verfügbar sind. Ausserdem kann eventuell auch die Kategorisierung und Formulierung der einzelnen Elemente optimiert werden.	1
	Breadcrumbs	Mittels Breadcrumbs (Navigations-Pfad) kann die Übersichtlichkeit weiter verbessert werden. Der User weiss dann laufend, in welcher Ansicht bzw. Struktur er sich befindet.	2
	Rollenmodell	Einzelne Elemente und Funktionen sollen nicht allen Usern zugänglich gemacht werden - es gibt also Anzeigeunterschiede zwischen verschiedenen Rollen.	2
	Performance	Falls möglich, soll die Performance weiter gesteigert werden.	3
	Meldungen	Fehlermeldungen und Nachrichten sollen einheitlich dargestellt und durch ein übergeordnetes Management von Meldungen behandelt werden.	2
	Einheitliches Erscheinungsbild	Teilweise werden Formulare in demselben Fenster und manchmal in PopUps geöffnet. Ziel soll die Vereinheitlichung der Formulare und deren Anzeige sein.	2
Suche	Suchresultate	Die Suchresultate werden in Menü-Dropdowns und dann in weiteren Subdropdowns dargestellt, was unter Umständen schnell unübersichtlich werden kann. Eine eigene Anzeige für die Suchresultate ist geplant, da die Suche wahrscheinlich oft verwendet wird.	2
Kunden	Kundenübersicht	Die Kundenübersicht bietet sehr viele Informationen, ist aber leider (insbesondere auf kleinen Bildschirmen) nicht mehr übersichtlich und wirkt überfüllt. Weniger wichtige Informationen sollten ausgeblendet werden. Eventuell kann die Bedienung optimiert werden, indem man die Grösse der einzelnen Container flexibler gestaltet und die Container zuklappen/ausblenden kann.	1

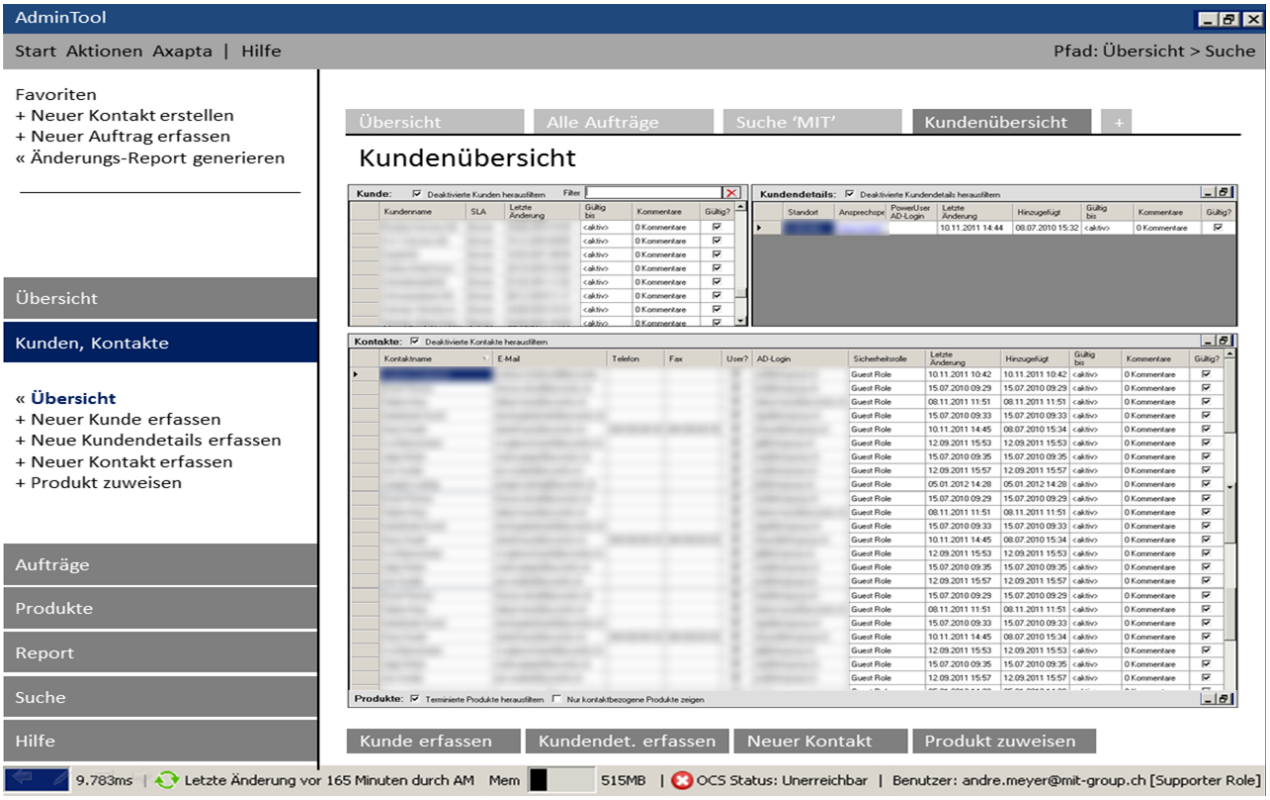
Der folgende Screenshot veranschaulicht die Ausgangslage des AdminTools am Beispiel der Kundenübersicht:



DESIGNPROTOTYP

Basierend auf den ersten Erkenntnissen und Ideen, haben wir mit Hilfe von Microsoft PowerPoint einen Designprototypen entwickelt. Das Ziel bestand darin, Ideen zu besprechen und ein gemeinsames Verständnis zu entwickeln, um sich auf das Interview vorbereiten zu können.

Der folgende Screenshot zeigt den Designprototypen einer möglichen Umsetzung für das zukünftige AdminTool ebenfalls am Beispiel der Kundenübersicht:



SOLL-ANALYSE (DESIGNSTUDIE)

Mit dem Wunsch nach einem ergonomischen und möglichst optimalen Re-Design des AdminTools, haben wir eine systematische SOLL-Analyse durchgeführt. Die eigenen Erkenntnisse sollten mit Wünschen und Ideen der späteren Benutzer ergänzt werden. Aus diesem Grund haben wir ein Interview mit je zwei Leuten aus dem Servicedesk (Supporter), der Administration und der Betriebsleitung (Geschäftsleitung) in dieser Reihenfolge durchgeführt. Das Gespräch mit den Interviewten begann mit einem freien Gespräch, in welchem die Befragten von sich aus erzählten, was ihnen gefällt und was ihrer Meinung nach an Überarbeitung bedarf. Während diesem Gespräch wurden die Antworten in die Leitpunkte und Kernfragen kategorisiert und bei Bedarf anschliessend nachgefragt um die Antworten zu verfeinern. Die Vorschläge, die sich aus den vergangenen Interviews anhäuften, wurden in den folgenden Interviews mit den anderen Akteuren ebenfalls angesprochen und diskutiert. Mehrheitlich waren die Meinungen und Wünsche deckungsgleich.

Damit das Gespräch mit den Benutzern etwas strukturiert und systematisch ablaufen konnte, bzw. damit keine wichtigen Fragen vergessen gingen, wurden vorher einige Leitfragen definiert und mögliche Antworten überlegt, nach denen bei Bedarf gezielt nachgefragt werden konnte. Diese Fragen werden im folgenden Abschnitt schematisch aufgelistet:

1. Wofür brauchen Sie das AdminTool?
   a. Ziel: Hauptaufgaben, Nutzen finden
   b. Welche Dialoge/Prozesse werden verwendet? (Gruppierung)
   c. Welche Masken/Dialoge werden wie gebraucht? (Gruppierung)
   d. Welche Masken können wie erweitert werden? Fehlt etwas?
      i. Erfassen/Neu
      ii. Ändern/Editieren
      iii. Abfragen
   e. Welche Felder/Informationen/Filter müssen auf den ersten Blick verfügbar sein? (Gruppierung)
2. Wie werden Funktionen verwendet (Ablauf)? Gibt es komplizierte oder unlogische Abläufe (Prozesse)?
   f. Wenn ja: Welche?
3. Wie wird gesucht? Wie finden Sie Dinge am Schnellsten?
   g. Nach was wird gesucht? (Inhalte)
   h. Filtern?
   i. Weitere Infos?
4. Was ist die wichtigste Erwartung an das neue AdminTool?

Hinweis: Unter "Gruppierung" wird das Einordnen in die einzelnen Haupt-Kategorien verstanden. Darunter gehören unter anderem Übersicht, Aufträge und Aktivitäten, Kundenübersicht, Reports, Produkte und Warenfluss.

Die Antworten und die daraus abgeleiteten Erkenntnisse der Gespräche bzw. Interviews sind im Folgenden tabellarisch zusammengefasst und wurden mit den Ideen aus der IST-Analyse ergänzt:

Kategorie	Interview Frage	Titel	Beschreibung	Prio
Allgemein	IST-Analyse, 1,4	Anzeige auf kleinen Bildschirmen	Loggt sich ein User mit einer Auflösung ein, die kleiner als 1440x900 ist, erscheint eine Meldung, dass das AdminTool nicht auf diese Bildschirmgrösse optimiert dargestellt werden kann. Wenn möglich, sollte die Anzeige auch für kleine Bildschirme optimiert werden (Wunsch: sogar Smartphones/Tablets, wenn möglich?)	3
	IST-Analyse, 1,2	Tab-Ansicht	Möglichkeit, mehrere Tabs offen zu haben, mit dem Ziel, dass eine Unterbrechung der aktuellen Arbeit möglich ist (z.B. bei Telefon-Unterbruch), um etwas nachzuschauen. Dabei gibt es die Einschränkung, dass jeweils nur ein Formular-Tab geöffnet sein darf.	1
	IST-Analyse	Weg-zum-Ziel	Teilweise ist es etwas umständlich bzw. unklar, eine Funktion bzw. einen Task zu bedienen, da der Ablauf kompliziert ist. Das Ziel ist, dass die wichtigsten Tasks in wenigen Klicks verfügbar sind. Ausserdem kann eventuell auch die Kategorisierung und	1

			Formulierung der einzelnen Elemente optimiert werden.	
	IST-Analyse	Breadcrumbs	Mittels Breadcrumbs (Navigations-Pfad) kann die Übersichtlichkeit weiter verbessert werden. Der User weiss dann laufend, in welcher Ansicht bzw. Struktur er sich befindet.	2
	IST-Analyse	Rollenmodell	Einzelne Elemente und Funktionen sollen nicht allen Usern zugänglich gemacht werden - es gibt also Anzeigeunterschiede zwischen verschiedenen Rollen.	2
	IST-Analyse	Performance	Falls möglich, soll die Performance weiter gesteigert werden.	3
	IST-Analyse	Einheitliches Erscheinungsbild	Teilweise werden Formulare in demselben Fenster und manchmal in PopUps geöffnet. Ziel soll die Vereinheitlichung der Formulare und deren Anzeige sein.	2
	1	Benutzerführung, Verwendung Tool	Menü wird kaum oder gar nicht verwendet --> nur wo es nicht anders geht. Das Kontextmenü auf DataGrids wird meist verwendet, da intuitiver Doppelklick nicht überall funktioniert oder Doppelklick auf gut Glück Zeile markieren & Button unten anklicken wird auch verwendet. Fazit: Jeder verwendet das Tool auf eine etwas andere Art, hat sich dran gewöhnt. Hier muss nicht zwangsläufig etwas geändert werden.	3
	IST-Analyse, 1	Meldungen	Servicedesk (SD): Viele Checkfragen nach Erfassung/Editierung sind mühsam --> werden ungelesen weggeklickt. Gewisse (teilw.) wichtige Meldungen gehen unter Fazit: Das Notification-Management soll vereinheitlicht und optimiert werden. Informationen über die aktuelle Aktion, Warnungen, Fehler, Fehleingaben den Status oder während Wartezeiten sind sinnvoll!	1
	1	Einstellungen	Speichern von Ansichten, Filteroptionen, Spaltensortierungen in DataGrids wünschenswert	3
Reports	1	Änderungen?	Kaum verändern, funktioniert gut, wie bisher. Allenfalls werden neue Reports hinzugefügt.	-
	1	Filter, Felder	Benötigte Felder: Datum, Beschreibung, Kundenname, Kontakt, Adresse, Gelöst? wichtig: Beschreibung nach vorne, Rest kann entfernt werden	2
Suche	1,3	Globale Suche	Globale Suche gehört zu den wichtigsten Features, muss zwingend beibehalten werden - kann eventuell einfacher zugänglich bzw. prominenter platziert werden	1
	IST-Analyse	Suchresultate	Die Suchresultate werden in Menü-Dropdowns und dann weiteren Subdropdowns dargestellt, was unter Umständen schnell unübersichtlich werden kann. Eine eigene Anzeige für die Suchresultate ist geplant - da die Suche wahrscheinlich oft verwendet wird.	2
	1	Filter	Filter-Optionen (nach Gruppierung/Kategorisierung) für Suche sehr wünschenswert	2
Kundenverwaltung	IST-Analyse, 1	Kundenübersicht	Die Kundenübersicht bietet sehr viele Informationen, ist aber leider (insbesondere auf kleinen Bildschirmen) nicht mehr übersichtlich und wirkt überfüllt. Weniger wichtige Informationen sollten ausgeblendet werden. Eventuell kann die Bedienung optimiert werden, indem man die Grösse der einzelnen Container flexibler gestaltet und die Container zuklappen/ausblenden kann. Alternativ könnte man sich auch eine neue Auswahlidee vorstellen, in der nur die wichtigste Spalte angezeigt wird und man dann (ähnlich der ITunes-Musik-Auswahl) Spalte für Spalte durchnavigiert, bis man schlussendlich zu der entsprechenden Information kommt. Die Details der aktuellen Selektion könnten dann beispielsweise unterhalb dieses Controls angezeigt werden.	1

			Administration braucht diese Übersicht selten, da etwas zu unübersichtlich --> exportieren Excel Reports mit Details	
	1,3	Filter	Featurewunsch: Wortfilter für alle Tabellen (Kunden, Kundendetails, Kontakte, Produkte) gewünscht Andere Filter zwingend beibehalten	2
	1	Felder	Kunde: Firmenname Kundendetails: Standort, Ansprechperson Kontakte: Name, E-Mail, Telefonnummer, Gültig bis, Kommentare Produkte: Produkt, aktiv, Benutzer, Anzahl, hinzugefügt, letzte Änderung, Gültig seit, Kommentare	2
Produkte	1	Änderungen?	Kaum verändern, funktioniert gut, wie bisher. Ansicht wird eher selten verwendet	-
	1	Felder	Produktbezeichnung, Kurzbezeichnung, einmalig, wiederkehrend	2
Übersicht	1	Inhalte	Alle: Neue/Offene Tickets-Anzeige wichtig SD: Übersicht Aufträge-Anzeige kann entfernt werden, da meist nur "Alle Aufträge anzeigen" verwendet wird; Grafiken werden selten verwendet, da sie an die Überlastung erinnern Betriebsleiter (BL): Wichtig sind die Grafiken über die Auslastung der Supporter (gebraucht um die Tickets einem SD-Mitarbeiter zuzuordnen)	2

ZIELERREICHUNG

Nachdem wir die Spezifikation erstellt und die Featurewünsche dokumentiert haben, haben wir den Design-Prototypen in Microsoft PowerPoint geringfügig optimiert. Dann starteten wir die Entwicklung des neuen AdminTool-Design-Prototypen. Dabei wurde sehr viel Wert auf die saubere Programmierung und Dokumentierung, das Einhalten der wichtigsten Design Patterns und natürlich auch auf die Optimierung der Oberfläche gelegt. Viele Details dazu sind in diesem Abschlussbericht, in der Dokumentation und natürlich auch im Source-Code zu finden.

Da die Projekt-Dauer recht beschränkt war, lag das Schwergewicht auf dem Grunddesign der Oberfläche, der Navigation, dem Handling von Formularen und der Beispiel-Umsetzung auf einem Modul (einer Komponente/Kategorie) mit der Absicht, dass ein anderer Entwickler damit und mit Hilfe der vorliegenden Dokumentation selbstständig die Umsetzung vornehmen kann. Aus den genannten Gründen und weil es um einen Oberflächen-Prototyp geht, sind die meisten Module und entsprechenden Formulare relativ leer. Die Datenbankanbindung und die Datenabfragen wurden nur minimal ausprogrammiert und Anbindungen an weitere Dienste (Lync, Axapta, etc.) wurden vollständig weggelassen.

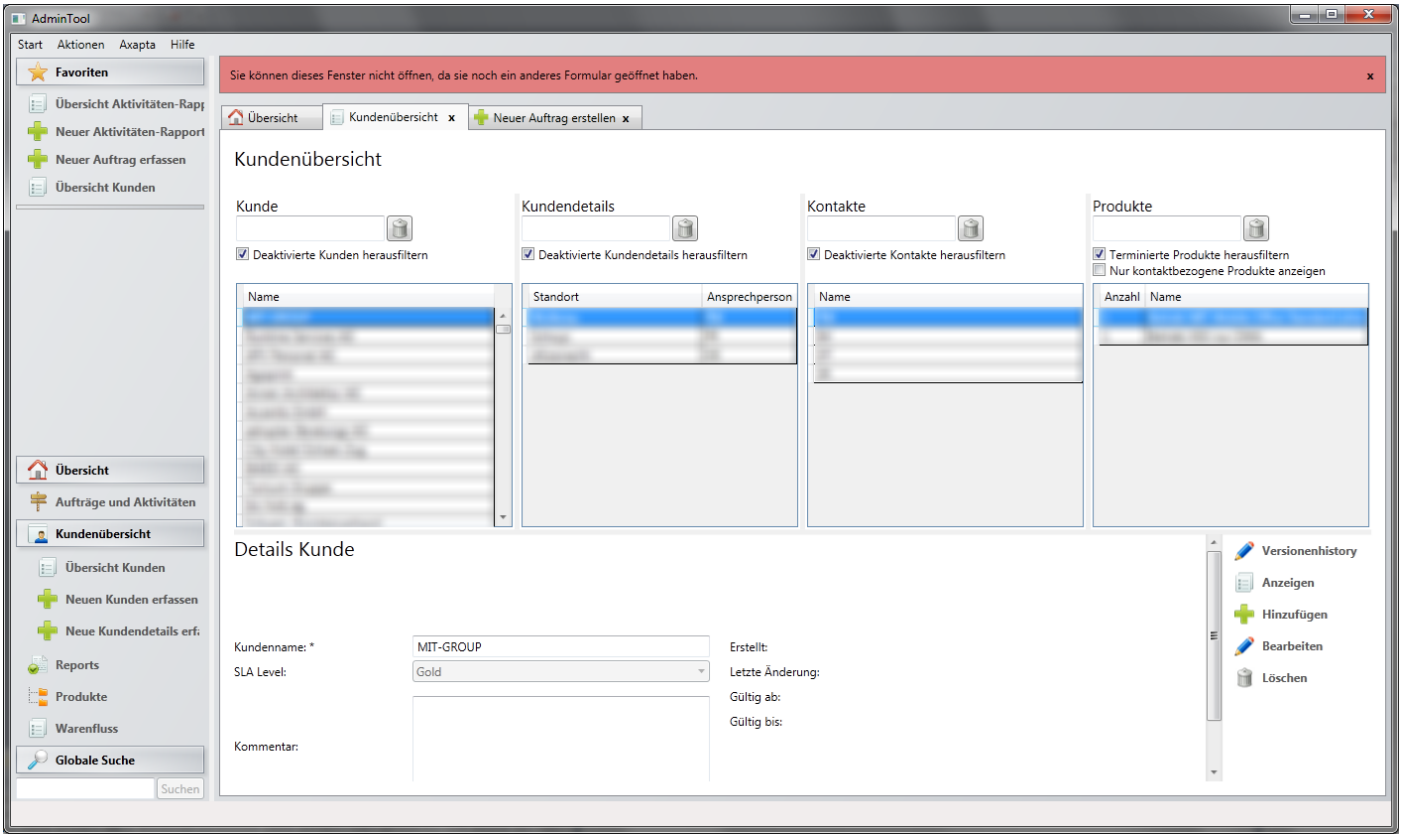
Die persönliche Einschätzung der erreichten Ziele im Bezug auf den Design-Prototypen sind in der folgenden Tabelle zusammengefasst:

Kategorie	Titel	Beschreibung	Zielerreichung (Beurteilung)
Allgemein	Anzeige auf kleinen Bildschirmen	Loggt sich ein User mit einer Auflösung ein, die kleiner als 1440x900 ist, erscheint eine Meldung, dass das AdminTool nicht auf diese Bildschirmgrösse optimiert dargestellt werden kann. Wenn möglich, sollte die Anzeige auch für kleine Bildschirme optimiert werden (Wunsch: sogar Smartphones/Tablets, wenn möglich?)	Der Prototyp wurde so optimiert, dass die Applikation auch auf Bildschirmen mit kleinerer Auflösung mit wenigen Einschränkungen verwendet werden kann. Ansichten mit sehr viel Content (Kundenübersicht) wurden stark vereinfacht und benötigen weniger Platz. Aktuell gibt es keine Umsetzung für Smartphones oder Tablets. Da die Applikation in WPF programmiert worden ist, lässt sie sich aber problemlos und mit relativ geringem Aufwand in eine Windows 8-App oder Windows Phone 7.5-App portieren.
	Tab-Ansicht	Möglichkeit, mehrere Tabs offen zu haben, mit dem Ziel, dass eine Unterbrechung der aktuellen Arbeit möglich ist (z.B. bei Telefon-Unterbruch), um etwas nachzuschauen. Es soll jeweils nur ein Formular geöffnet sein.	Die Tab-Ansicht wurde entsprechend den Wünschen umgesetzt. Der Übersichts-Tab kann (absichtlich) nicht geschlossen werden. Die anderen Tabs haben eindeutige Bezeichnungen sowie Icons und können geschlossen werden.
	Weg-zum-Ziel	Teilweise ist es etwas umständlich bzw. unklar, eine Funktion bzw. einen Task zu bedienen, da der Ablauf kompliziert ist. Das Ziel ist, dass die wichtigsten Tasks in wenigen Klicks verfügbar sind. Ausserdem kann eventuell auch die Kategorisierung und Formulierung der einzelnen Elemente optimiert werden.	Die gesamte Struktur als auch die Kategorisierung wurde stark angepasst und verändert. Die Menüführung konzentriert sich auf eine Outlook-ähnliche Ansicht in der linken Spalte. Jeder dieser Unterpunkte (z.B. Kundenübersicht, neuer Kunde, etc.) kann auch als Favorit hinzugefügt werden. Dadurch werden alle wichtigen Funktionen leichter und mit weniger Klicks zugänglich.
	Breadcrumbs	Mittels Breadcrumbs (Navigations-Pfad) kann die Übersichtlichkeit weiter verbessert werden. Der User weiss dann laufend, in welcher Ansicht bzw. Struktur er sich befindet.	Da die Hierarchie nicht so tief geht, wie ursprünglich angenommen und die Menüführung in der linken Spalte ausreicht, wurden keine Breadcrumbs eingebaut und die Idee verworfen.
	Rollenmodell	Einzelne Elemente und Funktionen sollen nicht allen Usern zugänglich gemacht werden - es gibt also Anzeigeunterschiede zwischen verschiedenen Rollen.	Die Modularisierung durch Prism erlaubt es, einzelne Module je nach Rolle ein- bzw. auszublenden.
	Performance	Falls möglich, soll die Performance weiter gesteigert werden.	Ob es Performance-Steigerungen gibt, kann zum jetzigen Zeitpunkt nur sehr schwer eingeschätzt werden, da die Datenbank-Einbindung nur rudimentär umgesetzt bzw. vorhanden ist und die Abfrage-Queries kaum optimiert worden sind. Des Weitern ist zu erwarten, dass die Performance auf dem Server abnehmen wird, da gleichzeitig mehrere User auf das AdminTool und die Datenbank zugreifen.
	Einheitliches Erscheinungsbild	Teilweise werden Formulare in demselben Fenster und manchmal in PopUps geöffnet. Ziel soll die Vereinheitlichung der Formulare	Alle Formulare (z.B. Neuer Kontakt, bestehender Kontakt editieren) werden nun in einem neuen Tab

		und deren Anzeige sein.	angezeigt. Es kann jeweils nur ein einzelner Edit/New-Tab geöffnet werden. Kleine Fenster, wie z.B. die Axapta-Projektnummer-Auswahl, werden wie bisher in PopUps angezeigt. Eine wichtige Neuerung und ein grosser Vorteil ist, dass jedes Formular nur noch einmal umgesetzt werden muss. Jedes Formular (bzw. jeder Tab) hat ein Property FormType, welches die Anzeige steuert. Beispielsweise können dann keine Einträge editiert werden (Default-View-Ansicht), es werden keine Inhalte angezeigt (New-Ansicht) oder es kann alles editiert werden (Edit-Ansicht).
	Benutzerführung, Verwendung Tool	Menü wird kaum/nicht verwendet --> nur wo es nicht anders geht Kontextmenü auf DataGrids wird meist verwendet, da intuitiver Doppelklick nicht überall funktioniert oder Doppelklick auf gut Glück Zeile markieren & Button unten anklicken wird auch verwendet. Fazit: Jeder verwendet das Tool auf eine etwas andere Art, hat sich dran gewöhnt. Hier muss nicht zwangsläufig etwas geändert werden.	Die Benutzerführung darunter auch das Menü wurde stark überarbeitet. Siehe "Weg-zum-Ziel". Das Menü im oberen Teil des Fensters wurde übernommen, wurde aber noch nicht implementiert, da es wahrscheinlich auch weggelassen werden kann, sofern sich die User schnell mit der neuen “Outlook-Menü”-Führung anfreunden.
	Meldungen	SD: Viele Checkfragen nach Erfassung/Editierung sind mühsam --> werden ungelesen weggeklickt. Gewisse (teilw.) wichtige Meldungen gehen unter Fazit: Das Notification-Management soll vereinheitlicht und optimiert werden. Informationen über die aktuelle Aktion, Warnungen, Fehler, Fehleingaben den Status oder während Wartezeiten sind sinnvoll!	Ein einheitliches Notification-Panel wurde entwickelt. Die aktuellste Meldung macht sich durch eine Animation bemerkbar. Durch Klick auf die Meldung gelangt man zum entsprechenden Tab.
	Einstellungen	Speichern von Ansichten, Filteroptionen, Spaltensortierungen in DataGrids wünschenswert	Wurde bisher noch nicht umgesetzt.
Reports	Änderungen?	Kaum verändern, funktioniert gut, wie bisher. Allenfalls werden neue Reports hinzugefügt.	Wurde bisher noch nicht umgesetzt.
	Filter, Felder	Benötigte Felder: Datum, Beschreibung, Kundenname, Kontakt, Adresse, Gelöst? wichtig: Beschreibung nach vorne, Rest kann entfernt werden	Wurde bisher noch nicht umgesetzt.
Suche	Globale Suche	Globale Suche gehört zu den wichtigsten Features, muss zwingend beibehalten werden - kann eventuell einfacher zugänglich bzw. prominenter platziert werden	Die Suche ist im neuen Menü direkt eingebaut und kann mit einem Klick verwendet werden.
	Suchresultate	Die Suchresultate werden in Menü-Dropdowns und dann weiteren Subdropdowns dargestellt, was unter Umständen schnell unübersichtlich werden kann. Eine eigene Anzeige für die Suchresultate ist geplant - da die Suche	Die Suchresultate werden jeweils in einem eigenen Tab angezeigt und zeigen (je nach Kategorie/Gruppierung) die wichtigsten Informationen.

		wahrscheinlich oft verwendet wird.	
	Filter	Filter-Optionen (nach Gruppierung/Kategorisierung) für Suche sehr wünschenswert	Die Suchresultate können nach Kategorie/Gruppierung gefiltert werden.
Kundenverwaltung	Kundenübersicht	Die Kundenübersicht bietet sehr viele Informationen, ist aber leider (insbesondere auf kleinen Bildschirmen) nicht mehr übersichtlich und wirkt überfüllt. Weniger wichtige Informationen sollten ausgeblendet werden. Eventuell kann die Bedienung optimiert werden, indem man die Grösse der einzelnen Container flexibler gestaltet und die Container zuklappen/ausblenden kann. Alternativ könnte man sich auch eine neue Auswahlidee vorstellen, in der nur die wichtigste Spalte angezeigt wird und man dann (ähnlich der iTunes-Musik-Auswahl) Spalte für Spalte durchnavigiert, bis man schlussendlich zu der entsprechenden Information kommt. Die Details der aktuellen Selektion könnten dann beispielsweise unterhalb dieses Controls angezeigt werden. Administration braucht diese Übersicht selten, da etwas zu unübersichtlich --> exportieren Excel Reports mit Details	In der neu strukturierten Kundenübersicht wurde die Idee analog zur “iTunes-Musik-Auswahl” umgesetzt. Die neue Ansicht besteht aus den folgenden Spalten (mit jeweiligen Filtern und Wortfiltern): - Kunde: Name - Kundendetails: Standort, Ansprechperson - Kontakte: Name - Produkte: Anzahl, Name Durch Klick auf eines dieser Elemente, werden unterhalb dieses Controls alle Details eingeblendet. Abhängig vom Kontext kann dann die aktuelle Auswahl angesehen, bearbeitet, gelöscht oder neu erstellt werden.
	Filter	Featurewunsch: Wortfilter für alle Tabellen (Kunden, Kundendetails, Kontakte, Produkte) gewünscht Andere Filter zwingend beibehalten	Ist umgesetzt.
	Felder	Kunde: Firmenname Kundendetails: Standort, Ansprechperson Kontakte: Name, E-Mail, Telefonnummer, Gültig bis, Kommentare Produkte: Produkt, aktiv, Benutzer, Anzahl, hinzugefügt, letzte Änderung, Gültig seit, Kommentare	Wird nicht mehr benötigt, da alle Details unterhalb des Controls angezeigt werden.
Produkte	Änderungen?	Kaum verändern, funktioniert gut, wie bisher. Ansicht wird eher selten verwendet	Umsetzung zu Testzwecken begonnen. Nicht definitiv umgesetzt.
	Felder	Produktbezeichnung, Kurzbezeichnung, einmalig, wiederkehrend	Wurde bisher noch nicht umgesetzt.
Übersicht	Inhalte	Alle: Neue/Offene Tickets-Anzeige wichtig SD: Übersicht Aufträge-Anzeige kann entfernt werden, da meist nur "Alle Aufträge anzeigen" verwendet wird; Grafiken werden selten verwendet, da sie an die Überlastung erinnern BL: Wichtig sind die Grafiken über die Auslastung der Supporter (gebraucht um die Tickets einem SD-Mitarbeiter zuzuordnen)	Wurde bisher noch nicht umgesetzt. Der Übersichts-Tab wird bei Applikations-Start angezeigt und kann nicht geschlossen werden.

Ein Screenshot der endgültigen Umsetzung am Beispiel der Kundenübersicht ist im Folgenden angefügt.



BESCHREIBUNG DER VERWENDETEN METHODEN UND HILFSMITTEL

WINDOWS PRESENTATION FOUNDATION (WPF)

ERSTE SCHRITTE

Die ersten beiden Arbeitswochen waren wir damit beschäftigt, die Grundlagen der Programmierung mit der Windows Presentation Foundation (WPF) zu erlernen, mit welcher das AdminTool später implementiert wurde.

WPF ist ein Framework von Microsoft. Es ist Teil des bekannten .NET-Frameworks und seit .NET 3.0 enthalten. Momentan steht WPF in der Version 4 (seit April 2010) zur Verfügung. WPF stellt sehr umfangreiche Möglichkeiten bereit, um Applikationen für Microsoft Windows zu entwickeln. Es ist hierbei möglich, die Logik einer Applikation von ihrer Darstellung zu trennen. Während die Logik grundsätzlich in C# entwickelt wird, wird das Design der Applikation in XAML (eXtended Application Markup Language) deklarativ definiert. Alles, was im .xaml-File steht, könnte auf Kosten der Übersicht und der Trennung von Logik und Design grundsätzlich auch mit C# implementiert werden. Wie der Name schon erraten lässt, baut XAML auf XML auf und nutzt dessen Syntax, um die graphische Benutzeroberfläche zu definieren. Bei der Kompilierung wird der C#-Code automatisch mit dem XAML verknüpft.

Neben der sehr umfangreichen Funktionalität von WPF gibt es zusätzliche die Möglichkeit, dieses durch andere Frameworks und Klassenbibliotheken zu erweitern. Im Laufe unseres Projekts sind vor allem Prism sowie das EntityFramework als wichtige Frameworks dazugekommen.

FRAMEWORK: PRISM

Dieses Framework erlaubt es, einfach umfangreiche und flexible WPF-Desktop-Applikationen zu entwickeln, die mit geringem Aufwand unterhalten werden können. Dazu nutzt es Design Patterns, die wichtige Architekturprinzipien wie Separation of Concerns und Loose Coupling umsetzen. Prism erlaubt es, eine Applikation aus lose gekoppelten Einzelmodulen zu erstellen, die einfach in das bestehende Projekt integriert oder aus diesem ausgeschlossen werden können. Dieser Applikationstyp wird im Englischen als composite application bezeichnet.

Bezogen auf unser Projekt erlaubte es Prism, das AdminTool in einzelne Module zu gliedern, die verschiedenen Aufgabenbereichen entsprechen und nur geringfügig voneinander abhängig sind. So beschäftigt sich beispielsweise das CustomerModule mit der Kundenverwaltung und das TicketModule mit der Auftragsverwaltung. Einige Module wie das SearchModule (globale Suche) oder das FavouriteModule (Auswahl von bestimmten Aufgaben in einer Favoriten-Ansicht) wurden aufgabenunabhängig designed. Die einzelnen Module werden durch ein Kernmodul geladen, welches sich Shell nennt. Da das AdminTool von Supportern des ServiceDesks, der Geschäftsleitung, der Administration und später auch von einzelnen Kunden verwendet wird, erlaubt es diese Aufteilung, bestimmte Funktionalität rechteabhängig für bestimmte Benutzergruppen freizugeben oder zu sperren. Zu diesem Zweck werden lediglich bestimmte Module beim Applikationsstart geladen und andere nicht.

FRAMEWORK: ENTITYFRAMEWORK

Hierbei handelt es sich um ein Object/Relational Mapping (ORM) Framework, dass es Entwicklern erlaubt, mit Daten aus relationalen Datenbanken objektorientiert zu arbeiten, indem die Tupel einer Relation in C#-Objekte konvertiert werden. Das Entity Framework stellt eine Möglichkeit zur Verfügung, Datenbankabfragen durch sogenannte LINQ-Queries durchzuführen. LINQ (Language Integrated Query) ist eine Komponente des .NET-Frameworks zur Abfrage von beliebigen Datenquellen. Dank dem Entity Framework war es uns im Rahmen unseres Projektes möglich, unseren Prototyp des AdminTools mit Beispieldaten zu testen und um eine Datenbankschnittstelle zu erweitern. Das automatische ORM hat uns hier viel Programmierarbeit abgenommen. Da wir das Datenbankschema inklusive Beispieldaten für unsere Zwecke erhalten haben, ist diese Schnittstelle für die weitere Entwicklung des AdminTools verwendbar.

DESIGNPATTERNS

MODEL VIEW VIEWMODEL (MVVM)

Das MVVM Pattern ist ein Design-Pattern von Microsoft, dass mit WPF und Silverlight (.NET 3.0) bekannt wurde und dem klassischen Model View Controller (MVC) Pattern gleicht. MVVM wurde entwickelt, um die Benutzeroberfläche und die Logik der Applikation voneinander zu trennen. Dadurch können diese Komponenten unabhängig voneinander entwickelt werden.

Model: Stellt der Applikation Daten zur Verfügung, die in der View dargestellt werden sollen. Die View kennt das Model jedoch nicht und hat keinen direkten Zugriff auf diese Daten.
ViewModel: Kennt das Model und bezieht über dieses Daten. Es bereitet diese entsprechend auf und bietet sie der View an.
View: Definiert die graphische Benutzeroberfläche. Sie bekommt Daten zur Darstellung vom ViewModel. Die View hat keinen direkten Kontakt zum Model und kennt lediglich das ViewModel.

In unserem Projekt wurden die einzelnen Module des AdminTool-Prototyps mit diesem Pattern entwickelt. In der Regel besteht das Model aus einer Datenbankanbindung, aus welcher der Applikation mit Hilfe des EntityFrameworks Daten über Kunden, Standorte, Produkte und Aufträge zur Verfügung gestellt werden. Im Normalfall definiert das ViewModel Listen und Collections, in welchen diese Daten organisiert und der View zur Verfügung gestellt werden. Die View wiederum definiert Tabellen und Grids, welche an die Listen des ViewModels gebunden werden und die erhaltenen Daten darstellen. Interaktionen mit der GUI ändern somit auch keine Daten direkt. Werden beispielsweise durch einen Klick neue Daten angefordert, interagiert die View mit dem ViewModel, welches die neuen Daten aus dem Model extrahiert und der View erneut zur Verfügung stellt.

SINGLETON

Das Singleton Design-Pattern gehört zur Gruppe der Creational Patterns. Es garantiert, dass von einer Klasse lediglich eine (in der Regel global verfügbare) Instanz erstellt wird. Das Pattern ist bekannt und im Internet sehr gut dokumentiert. Es wurde im Rahmen des AdminTools nur einmal verwendet und wird aus Platzgründen nicht genauer erklärt.

Für das AdminTool ist es wichtig, dass jedes Modul nur einen Datenkontext hat. Andernfalls führt dies bei Manipulationen an der Datenbank zu Inkonsistenzen. Aus diesem Grund wurde bei der Klasse, welche die Anbindung an die Datenbank im Model ermöglicht, das Singleton-Pattern verwendet.

OBSERVER (PUBLISH/SUBSCRIBE)

Das Observer-Pattern gehört zu den Behavioural Patterns und erlaubt es einem Objekt, Informationen und Neuigkeiten einer bestimmten Art bei einem anderen Objekt zu abonnieren. Dieser abonnierende Teil wird auch Subscriber genannt. Der Publisher hingegen schickt eine Nachricht an alle Subscriber, die diese Nachricht abonniert haben. Dieses Pattern wurde im AdminTool an verschiedenen Orten verwendet. Das Framework Prism stellt den generischen CompositePresentationEvent<T> zur Verfügung, über welchen mit Publish/Subscribe Events abonniert und getriggert werden können. Auf diesem Weg funktioniert beispielsweise das Hinzufügen und Entfernen von Favoriten aus der Favoritenliste. Das AdminTool verfügt auch über ein Notification Panel, in welchem Neuigkeiten und Mitteilungen angezeigt werden, die für den Benutzer des AdminTools wichtig sind. Die Veröffentlichung dieser Mitteilungen wird auch mit diesem Pattern gelöst.

ENTWICKLUNGSUMGEBUNG (VISUAL STUDIO 2010)

Visual Studio 2010 ist die ideale, komfortable Entwicklungs-Plattform für kleinere bis ganz grosse Software-Projekte in Visual Basic, C, C++, CSharp, F# und neuerdings auch HTML, CSS und JavaScript. Entwickler, Tester, Projektleiter sowie Designer & Prototyper können diese Plattform mit all ihren Add-Ons und Erweiterungen nutzen. Nebenbei ist es interessant zu erwähnen, dass Visual Studio ebenfalls mit WPF programmiert worden ist. Für die Versionen-Kontrolle des Source-Codes gibt es die Möglichkeit SVN, GIT oder den Microsoft Team Foundation Server zu verwenden. Da wir bereits alle Erfahrungen mit SVN haben, haben wir bei diesem Projekt SVN verwendet (siehe nächstes Kapitel für weitere Details).
Für die Umsetzung des AdminToolPrototype bot uns Visual Studio gute Möglichkeiten zur gemeinsamen Entwicklung am selben Source-Code (SVN), Coding-Unterstützung (Intellisense, ReSharper) und Debbugging-Funktionalität. Speziell am VS10-Debugger ist, dass es sich um einen sogenannten Rückblick-Debugger handelt, der Laufzeitinformationen (z.B. Variableninhalte) von ausgeführtem Code, der vor einem Breakpoint (Haltepunkt) liegt, anzeigen kann.

VERSIONENVERWALTUNG

Wie vorgängig erwähnt, haben wir unseren Code dem MIT-Code-Repository hinzugefügt. Dabei kam SVN zum Einsatz. SVN ist ein Versionsverwaltungs-Tool. Für die Versionenverwaltung im Explorer (dazu gehören Code, Dokumentation und weitere Files) haben wir Tortoise SVN verwendet. Als Erweiterung für die Versionenkontrolle des Source-Codes in Visual Studio verwendeten wir AnkhSVN. Dies war eine grosse Unterstützung, da man laufend mitverfolgen konnte, welches Team-Mitglied woran arbeitet(e). Im Falle von Fehlern gab es ein Backup, bzw. man konnte auf eine ältere Version des Codes zurückgreifen.

SCHLUSSREFLEXION

MEILENSTEINE: IST-SOLL-VERGLEICH

Domäne/ Unterfunktion	Position	Funktion/Aufgabe	Soll/Must	Zielerreichung
Einarbeitung	E 1.1	Bestehende Applikation	Must	erreicht
	E 1.2	Stakeholderanalyse, - Erstkontakt	Must	erreicht
	E 1.3	Installation der Arbeitsumgebung, Zugang Code-Repository	Must	erreicht
	E 1.4	Einarbeitung in WP4, Frameworks, Patterns	Must	erreicht
Iterationen	I 1.1	Kleinere Technikprototypen entwerfen, Patterns sammeln, umsetzen. Validierung im Team	Must	erreicht
	I 1.2	Designprototyp entwerfen, Usability Analyse, Validierung im Team	Must	erreicht
	I 1.3-1.5	Designstudie mit Stakeholder durchführen, Analyse, Verbesserung	Must	erreicht
	I 1.6	Finaler Prototyp steht, Design-Guidelines finalisieren, Dokumentation	Soll	teilweise erreicht *
Abschluss	A 1.1	Aufräumarbeiten (Design-Guideline abschliessen, Cleanup Code-Repository)	Must	erreicht
	A 1.2	Präsentation der Ergebnisse vor MIT Innovation AG	Must	pendent
	A 1.3	Präsentation der Ergebnisse in Form eines Berichtes (UZH)	Soll	erreicht

* Eine nähere Erläuterung betreffend dieser Meilensteine befindet sich im folgenden Abschnitt.

Die geplanten must Meilensteine wurden alle entsprechend den Vorgaben erreicht und mit dem Betreuer besprochen.
Die Iteration I 1.6 ist ein sehr allgemein gehaltener Meilenstein, welcher nicht zuletzt auch als Puffer gedacht war, um sicherlich nicht zu wenig Arbeit zu haben. Da die Projektdauer mit etwas weniger als 7 Wochen sehr begrenzt war, lag der Schwerpunkt weniger auf einer vollständigen Implementierung aller Formulare, Eingabemasken und Dialogen. Viel wichtiger war eine klare Struktur und eine stabile Umgebung, welche als Basis für die Weiterentwicklung dient. Deshalb konzentrierte sich unsere Arbeit vor allem auf ein spezifisches Modul (Customer) sowie die Module, welche zentrale Aufgaben wie die der Navigation, der Suche und Benachrichtigung für andere Module erfüllen (d.h. anderen Modulen Dienste anbieten).
Eine detaillierte Beschreibung zur Zielerreichung befindet sich zudem in Kapitel Designstudie unter Zielerreichung.

LESSONS LEARNED

Im Rahmen des Software-Projektes konnten wir sowohl fachlich wie auch arbeitstechnisch sehr viel profitieren. Dazu gehört, dass wir das Software Engineering nicht nur theoretisch, sondern auch praktisch anwenden konnten und sehr viele Technologien und Techniken systematisch erlernt und direkt umgesetzt haben. Dabei haben wir das erste Mal konkret gesehen, welche Vorteile die Anwendung von Design-Patterns und Architekturrichtlinien in der Entwicklung bringen. Ausserdem haben wir gelernt, wie man sich in einer Gruppe effizienter in ein neues Teilgebiet einarbeitet. Individuelles Lernen, Besprechen im Team und die Anwendung in kleinen Prototypen und Kontrolle mittels Verständnisfragen waren wichtige Komponenten der ganzen Lernphase. Diese Art des Lernens hat uns allen sehr gefallen.

Wir hatten die Möglichkeit, unsere Arbeit sehr frei einzuteilen und zu planen. Dabei mussten wir uns an die vorgegebenen Meilensteine halten, welche zu Beginn des Produktes definiert worden sind. Während dem ganzen Projekt hat sich gezeigt, dass eine Koordination der verschiedenen Tasks und Arbeiten durch eine Person dem gemeinsamen Verständnis und Überblick sehr hilft. Ein Mitglied unserer Gruppe hat diese Arbeit ausgeführt, was sehr wichtig war, weil wir das Praktikum teils in Vollzeit (ganze Woche) sowie auch in Teilzeit (2 Tage die Woche) absolviert haben. Zudem gab es Pausen von mehreren Wochen (Prüfungen) zwischen den einzelnen Phasen. Eine Schwierigkeit war deshalb, dass man nach den Pausen wieder gut in die Projektumgebung fand und an der aktuellen Aufgabe weiterarbeiten konnte. Aus diesen Gründen war die Dokumentation betreffend unserer Projektplanung und Entscheidung sowie auch des umgesetzten Prototyps sehr wichtig, um stets den

Überblick über die gesamte Projektdauer zu behalten, ohne den Fokus zu verlieren. Um diese Ziele zu erreichen, haben wir unter Anderem verschiedene Collaboration-Tools (Microsoft OneNote, TeamViewer, Google Docs, SVN, usw.) verwendet. In der Entwicklungsphase haben wir teils in direkter Zusammenarbeit Konzepte entwickelt, welche dann mittels Pair-Programming oder individuell umgesetzt worden sind. Dazu gehört auch die parallele Arbeit am gleichen Source-Code und dessen Folgen bezüglich der Abstimmung und Koordination aufeinander. Zusätzlich folgen daraus auch die Einhaltung von festgelegten Richtlinien (Interfaces, Code-Dokumentation, etc.) und gemeinsam getroffenen Architekturentscheidungen.

Da sich unsere Arbeitsplätze in Schwyz befanden, mussten wir viel pendeln. Weil die Zeit während dem Semester grundsätzlich knapp war, stellte sich dies als zusätzliche Belastung heraus. Die Atmosphäre im Team war wie erwartet freundschaftlich und angenehm. Gesamthaft haben wir das Software-Praktikum als sehr positive Erfahrung wahrgenommen, die wir nicht missen wollen. Wir möchten uns ganz herzlich bei unserem Arbeitgeber, der MIT Innovation AG, unserem Projektleiter Linard Moll, der Geschäftsleitung und allen anderen Mitarbeitern für die Chance, die tolle Unterstützung und die zahlreichen spannenden und aufschlussreichen Gespräche bedanken. Dasselbe gilt auch für Professor Böhlen, welcher uns betreut und beraten hat.