HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL
FHO Fachhochschule Ostschweiz

SEVITEC
SOFTWARE-LÖSUNGEN NACH MASS

Einbindung von Online-Telefonverzeichnissen in ein bestehendes Dynamic Document Creation System

Studienarbeit

Abteilung Informatik
Hochschule für Technik Rapperswil

Frühjahrssemester 2011

Autor(en):	Ramon Müller
Betreuer:	Prof. Hansjoerg Huser
Projektpartner:	Simon Baer
	Sevitec, Eschlikon

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

Erklärung über die eigenständige Arbeit

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.

Ort, Datum:

Rapperswil, 10.6.2011

Name, Unterschrift:

Müller Ramon

R. Müller

Abstract

Abteilung	**Informatik**
Name[n] der Studierenden	**Ramon Müller**
Studienjahr	**FS2011**
Titel der Studienarbeit	**Einbindung von Online-Telefonverzeichnissen in ein bestehendes Dynamic Document Creation System unter Microsoft Office, mit .NET, C#, Visual Studio 2010**
Examinatorin / Examinator	**Hansjörg Huser**
Themengebiet	**Softwareentwicklung**
Projektpartner	**Sevitec** **Bahnhofstrasse 4** **8360 Eschlikon**
Institut	**Institut für vernetzte Systeme (INS)**

Hintergrund: Sevitec AG ist eine Softwareentwicklungsfirma die ein Office Add-on zur vereinfachten Handhabung von Corporate Identity entwickelt. Mit Hilfe dieses Add-ons können personalisierte Briefköpfe und vordefinierte Textblöcke generiert werden, wodurch ein möglichst einfaches und einheitliches Corporate Identity-Management ermöglicht wird. Die bestehende Software ist in Visual Basic implementiert und seit einiger Zeit in Betrieb, eine Neuentwicklung in C# ist derzeit im Gange.

Aufgabe: Mit diesem Add-on ist es möglich Adressdaten aus diversen Quellen (Outlook, Exchange-Server, lokale Adressverzeichnisse, …) abzufragen und in dem System weiterzuverwenden. Die Aufgabe bestand nun darin, abzuklären ob/was es für Möglichkeiten gibt auch online Verzeichnisse wie online-Telefonbücher oder Verzeichnisse abzufragen. Bei der Evaluation gab es sowohl technische wie auch rechtliche Standpunkte zu beachten. Da die Neuentwicklung in diesem Bereich noch in den Anfangsschuhen stand war ich relativ frei was Interfacedefinitionen oder Architekturelle Entscheide anging.

Resultat: Das Resultat meiner Studienarbeit beinhaltet ein MEF-Plug-In mit einer Serviceschnittstelle, die auf einen RESTful Service von Search.ch zugreift. Dieser Service antwortet auf die Suchanfrage mit einem ATOM-Feed in dem die Treffer der Suchanfrage sowie einige weitere Details zur Suche enthalten sind. Da das Bedürfnis bestand die Abhängigkeiten zum Webservice möglichst gering zu halten, wird die Antwort anschliessend mit Hilfe einer XSL-Transformation in eine interne XML-Struktur umgebaut. Dies ermöglicht es Änderungen am Webservice nur durch anpassen der XSL-Datei - also ohne neu kompilieren des MEF-Plug-Ins - nachzuziehen.

Projektplan

Dokumentinformationen

Zweck
Dieses Dokument dient dem Planen des Projektes.

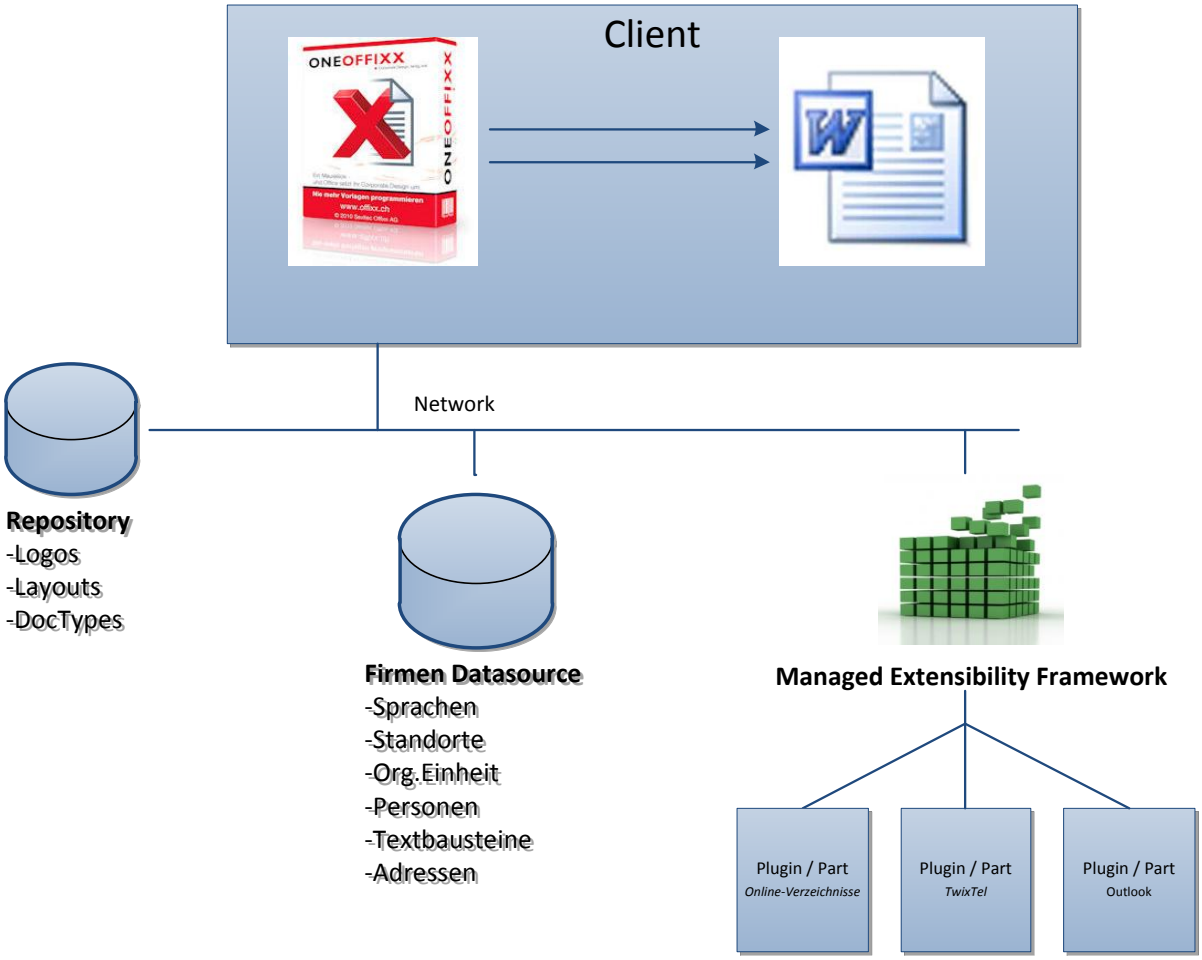
Gültigkeitsbereich
Dieses Dokument ist für den kompletten Verlauf des Projektes gültig.

Übersicht
Im Projektplan wird die Planung des Projekts geschildert. Er gibt Auskunft über Umfang, Zeitplanung sowie Iterationsplanung.

Änderungsgeschichte

Datum	Version	Änderung	Autor
7.3.2011	0.1	Erstellung des Dokuments	Ramon Müller
14.3.2011	1.0	Review & Freigabe	Ramon Müller

Projekt Übersicht



Ausgangslage

Dieses Projekt baut auf einer vorhanden Software von Sevitec[1] ONEOFFIXXX auf. Diese Software ist ein Zusatz zu den Microsoft Office Produkten und ermöglicht ein konsistentes und möglichst Anwenderfreundliches Corporate Identity anzuwenden. Es unterstützt bei der Erstellung von Firmen-Dokumenten indem es das Corporate Identity automatisch auf das Dokument generiert. Das Corporate Identity umfasst verschiedene Features wie zum Beispiel eine Sammlung von häufig verwendeten Adressaten, Textbausteine, diverse Layouts und vieles mehr.

Zweck und Ziel

Ziel dieses Projekts ist es dem bestehenden Produkt ONEOFFIXXX eine ausbaufähige Schnittstelle bereitzustellen wodurch direkt aus ONEOFFIXX heraus online-Adressverzeichnisse angesteuert werden können. Diese Schnittstelle soll eine Anfrage entgegen nehmen, diese Verarbeiten und in einem geeigneten Format wieder an den Aufrufer zurückgeben. Dabei gilt es zu beachten dass doppelte Einträge (z.B. aus verschiedenen Verzeichnissen) herausgefiltert werden sollen. Ausserdem soll es in einem weiteren Schritt möglich sein die Suche auf weitere Länder wie Deutschland, Österreich oder Liechtenstein auszuweiten.

[1] Die Partnerfirma dieser Studienarbeit. (siehe www.sevitec.ch)

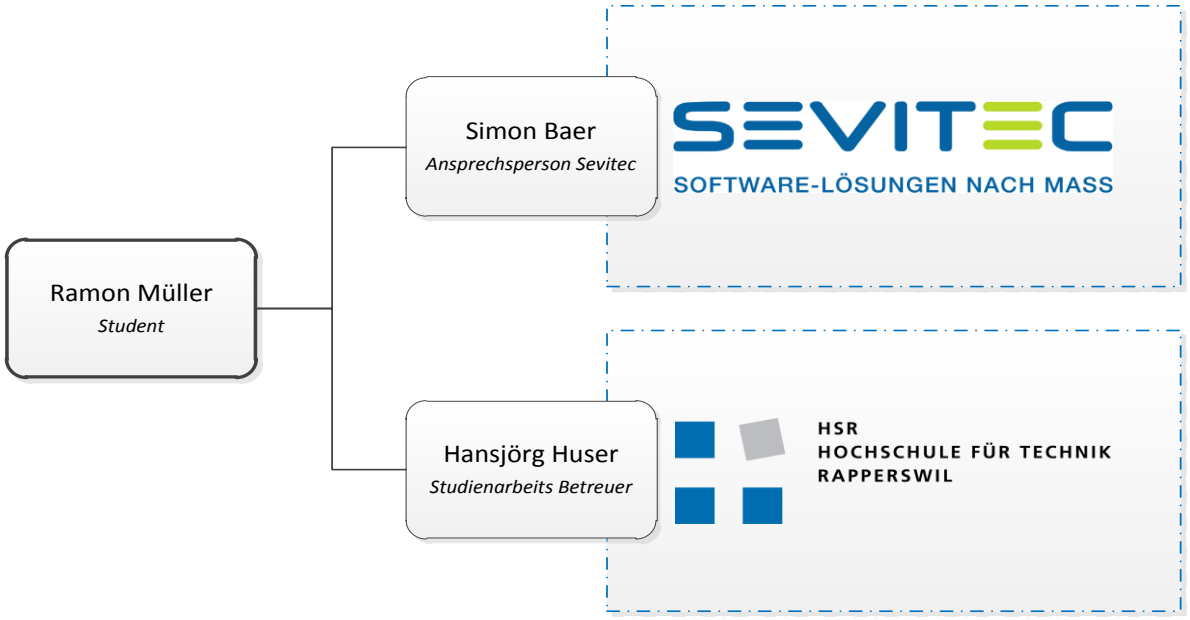
Annahmen und Einschränkungen

Es steht die Entwicklung der Schnittstelle zur Anbindung an die online-Verzeichnisse sowie die Logik zur automatische Erkennung von verfügbaren Services in einem Firmenumfeld im Vordergrund, die Einbindung des Services in die Applikation selbst ist nicht Teil der Arbeit.

Mögliche Erweiterungen

- Erweiterung der bereitgestellten Adress-Such-Services um Outlook-Kontakte, TwixTel sowie Exchange
- Erweiterung der unterstützten online-Verzeichnisse (Deutschland, Österreich, …)
- Einbindung des Services in die Applikation ONEOFFIXX

Projektorganisation



Projektmitglieder & Betreuung

Das Projekt wird von Ramon Müller alleine bestritten. Er wird Betreut von Herrn Hansjörg Huser, der die Studienarbeitsbetreuung von seitens der HSR übernimmt, sowie Herrn Simon Baer der die Betreuung und Anlaufstelle seitens Sevitec abdeckt.

Projektplan

Zeitplan

Insgesamt stehen dem Projekt 14 Wochen zur Verfügung. Bei einem erforderlichen Arbeitsaufwand von 30 Stunden pro ETCS Punkt ergeben sich für die Durchführung der Studienarbeit insgesamt 240 Arbeitsstunden.

Iterationsplan

Iteration	Ziel und Arbeiten	Dauer in Projektwochen	Milestone
Inception1	Grober Projektplan sowie Aufnahme und Festlegung der Requirements	2 Wochen	MS1

Elaboration 1	Planung der weiteren Phasen und Abklärungen/Einarbeitung bezüglich Technologie	2 Wochen	
Elaboration 2	Abschluss Analyse, erste Version des Prototyps	2 Wochen	MS2
Construction 1	Fertigstellung des Prototypen sowie eines TestClients	2 Woche	
Construction 2	Ausarbeiten des Grossteils der Funktionen	3 Wochen	MS3
Construction 3	Fehlerbehebung und Fertigstellung des Services	2 Wochen	MS4
Transition 1	Bereinigung der Dokumentation, Abschliessen des Projekts sowie „Plakaterstellung“	1 Woche	MS5

Meilensteine / Abgaben

Meilenstein	Iteration	Zeitpunkt	Prüfbares Produkt
MS1	Ende Inception	Ende Woche 2	Dokument: Anforderungsspezifikation
MS2	Ende Elaboration	Ende Woche 6	Lauffähiger Prototyp / Durchstich
MS3	Ende Construction 2	Ende Woche 11	Erster stabiler Release des Plugin / Part
MS4	Ende Construction 3	Ende Woche 13	Abgeschlossene Arbeiten / final Release
MS5	Ende Transition	Ende Woche 14	Bereinigte Dokumentation, fertiges „Plakat“

Besprechungen

Wöchentlich jeweils Mittwoch 10:30h im Raum 6.010 mit Herrn Huser.

Termine mit Herrn Baer finden ungefähr alle 2 Wochen im Firmensitz von Sevitec, Eschlikon statt. Genaue Termine werden vor zu ausgehandelt.

Risiko Management

Risiko-ID	Risikotitel	Risikobeschreibung	max. Schaden [h]	Eintrittswahrscheinlichkeit	gewichteter Schaden [h]	Massnahmen zur Vermeidung/Verminderung	Vorgehen bei Eintreffen
1	Fehleinschätzung des Aufwandes	Der Zeitplan kann aufgrund einer Fehleinschätzung nicht eingehalten werden	16	25.00 %	4.00	Kritische Arbeitspakete frühzeitig angehen, Technologiefragen früh klären	Funktionsumfang kürzen
2	Technologierisiko "Managed Extensibility Framework" (MEF)	Die Technologie"Managed Extensibility Framework" ist dem Student bisher unbekannt	8	20.00 %	1.60	Frühe Einarbeitung in die Technologie, frühzeitige Hilfestellungen suchen	Zeit investieren um sich in die Technologie einzulesen, gegebenfalls Hilfestellung suchen
3	Technologierisiko Webservice der online-Adressverzeichnisse	Die Technologie sowie der Umgang mit den WebServices ist unbekannt	16	10.00 %	1.60	Frühe Einarbeitung in die Technologie, frühzeitige Hilfestellungen suchen	Zeit investieren um sich in die Technologie einzulesen, gegebenfalls Hilfestellung suchen
4	Technologierisiko "Poco"	Die Technologie "Poco" ist dem Student bisher unbekannt	8	10.00 %	0.80	Frühe Einarbeitung in die Technologie, frühzeitige Hilfestellungen suchen	Zeit investieren um sich in die Technologie einzulesen, gegebenfalls Hilfestellung suchen
5	Krankheitsfall Student / Betreuer	Es gibt einen Krankheitsfall eines Betreuers oder des Studenten selbst	16	10.00 %	1.60	keine	Funktionsumfang kürzen, Zeitplan überarbeiten
6	Datenverlust	Erarbeitete Projektdaten gehen verloren	6	5.00 %	0.30	Regelmässige Backups erstellen	Daten durch Backup wiederherstellen
7	Ausfall Notebook	Das Arbeitsnotebook fällt aus	5	5.00 %	0.25	Regelmässige check-Ins um Datenverlust gering zu halten	Arbeitsplatzrechner der HSR verwenden
8	Ausfall SVN-Server	SVN-Server fällt aus	5	5.00 %	0.25	Kopien auf Arbeitsrechner aktuell halten	auf alternatives SVN-System zurückgreifen (code.google.com, TFS, ...)
Summe			**80**	**90.00 %**	**10.40**		

Arbeitspakete

MEF Host

Dieses Arbeitspaket umfasst den MEF Host aufsetzen, Metadaten der Parts festlegen sowie die Infrastruktur für die Plugins / Parts entsprechend zur Verfügung zu stellen.

MEF Plugin / Part

Dieses Arbeitspaket enthält den eigentlichen Service, der zur Verfügung gestellt wird (im Folgenden als MEF Part bezeichnet). Er enthält die Logik um die Daten eines Adressempfängers aus einem online-Telefonbuch abzufragen und an die aufrufende Applikation zurückzugeben.

Schnittstellendefinition

In diesem Arbeitspaket geht es darum geeignete Schnittstellen für die verschiedenen Imports / Exports zu definieren damit die verschiedenen Parts im Zusammenspiel funktionieren, gleichzeitig aber diese Schnittstellen möglichst ausbaufähig bleiben.

TestClient

In diesem Arbeitspaket geht es darum einen Client zu bauen der das Verhalten von ONEOFFIXX simuliert und Systemtests für das MEF Plugin zulässt.

Infrastruktur

Räumlichkeiten

Es steht ein Studienarbeitsplatz in den Räumlichkeiten der HSR zur Verfügung. Dieser befindet sich im Gebäude 1, Raum 1.206.

Hardware

Workstation (SA-Raum)

- Windows 7 SP1 64-bit
- 4GB Ram
- 250 GB Festplatte

Notebook (Student)

- Windows 7 SP1 64-bit
- 8GB Ram
- 120 GB SSD
- 450 GB Festplatte

Software

- Windows 7
- Microsoft Office 2010
- Microsoft Visio
- Borland Architect
- Microsoft Visual Studio 2010 Ultimate

Backup

SVN Server
Das SVN Repository wird jeweils über Nacht von der HSR gesichert.

Notebook
Die Daten werden in regelmässigen Abständen (Grössenordnung eine Woche) gesichert.

Qualitätsmassnahmen

Wöchentliche Sitzungen
Siehe Kapitel **Besprechungen**

Coderichtlinien
Es werden die .NET spezifischen Coding Guidelines angewendet. Einzusehen sind diese unter folgender Adresse: http://blogs.msdn.com/b/brada/archive/2005/01/26/361363.aspx. Dies entspricht den im Modul MsTech gelernten Coding Guidelines.

Reviews

Dokumentreviews
Dokumentreviews finden an den wöchentlichen Sitzungen mit Herrn Huser, sowie den Sitzungen mit Herrn Baer statt. Dabei geht es vor allem um den Projekt- und Zeitplan sowie die Analyse der verschiedenen online-Telefonbücher.

Codereviews
Die Codereviews werden jeweils beim Erreichen eines Milestones durchgeführt. Die betrifft vor allem die Meilensteine 2-4 deren Produkt jeweils ein Software-Release beinhaltet. Die Codereviews werden durch den Studenten selbst durchgeführt und es wird anschliessend ein Protokoll erstellt.

Testing

Automatische Tests
Es werden automatische Tests ausgearbeitet die möglichst Flächendeckend die Funktionalitäten des Plugins abdecken. Insgesamt wird eine Testabdeckung von 90% angestrebt. Geeignete Metrik-Software für die Überprüfung muss noch gesucht werden.

Systemtests
Ausserdem werden Ende des Meilensteins 3 Systemtests mit Hilfe des Testclients durchgeführt. Diese werden Protokolliert und allfällige Bugs weiterverarbeitet.

Bug Tracking
Für das Verwalten von Bugreports steht das BugTracking Tool Mantis zur Verfügung. Es ist eine Webapplikation die vom Student selber auf einem Webserver betrieben wird.

Online Telefonbuch-Provider Analyse

Dokumentinformationen

Zweck

Dieses Dokument beschreibt die Analyse der verschiedenen online Telefonbuch-Provider, die für dieses Projekt untersucht wurden und dient als Grundlage für den weiteren Verlauf des Projektes.

Gültigkeitsbereich

Dieses Dokument ist für den kompletten Verlauf des Projektes gültig. Alle Daten beziehen sich auf den Stand März / April 2011 und können sich in Zukunft ändern.

Übersicht

Es werden diverse online Telefonbuch-Provider auf Tauglichkeit zur online Adresssuche in ONEOFFIXX geprüft. Diese Analyse umfasst 6 der bekanntesten Provider für online Adressdaten in der Umgebung Schweiz, Deutschland, Österreich und Liechtenstein.

Änderungsgeschichte

Datum	Version	Änderung	Autor
21.3.2011	0.1	Erstellung des Dokuments	Ramon Müller
4.4.2011	1.0	Review & Freigabe	Ramon Müller

Rating

Es wird ein Rating-System verwendet, um die Tauglichkeit der verschiedenen Provider für dieses Projekt zu untersuchen.
Insgesamt reicht die Skala von 1 bis 10 Punkten, wobei 10 die Bestnote darstellt.
Die Punkte kommen wie folgt zustande:

Kostenlos	3 Punkte
Rechtliches	2 Punkte
Unterstützte Länder	2 Punkte
Lizenzierung	2 Punkte
API	1 Punkte

Kriterium „Kostenlos“

Dieser Punkt bewertet einen Telefonbuchprovider anhand der nötigen Vergütung für die Nutzung der Daten. Da dies ein wichtiger Punkt ist wird er mit 3 Punkten gewichtet. Hier gibt es noch eine Unterteilung je nach Umfang dieser Kosten, eine Bewertung von 1-3 Punkten ist hier möglich.

Kriterium „Rechtliches“

Ist aus rechtlicher Sicht eine Nutzung der Daten für die Zwecke von ONEOFFIXX möglich? Da dies praktisch ein K.O.-Kriterium darstellt wird es mit 2 Punkten bewertet.

Kriterium „Unterstützte Länder“

Bei diesem Bewertungskriterium geht es um eine Bewertung der verfügbaren Daten verschiedener Länder. Es gibt einen Punkt für jedes unterstützte Land, wobei hier nur Länder angerechnet werden die auch eine sinnvolle Anwendung in ONEOFFIXX erlauben. Die Schweiz und Liechtenstein werden hierbei zusammen als 1 Punkt gerechnet. Es sind maximal 2 Punkte möglich.

Kriterium „Lizenzierung“

Hierbei wird bewertet, wie die Lizenzierung der Nutzung des Services abgewickelt wird. Ist eine Lizenzierung für jeden Client nötig oder gibt es nur eine Lizenzierung für den Zugriff zum Service? Maximal 2 Punkte.

Kriterium „API“

Bei diesem Punkt geht es darum zu bewerten ob eine sinnvolle programmierschnittstelle existiert und wie leicht eine Anbindung an diese Möglich ist. Da im schlimmsten Fall auch eine HTML Seite geparst werden könnte wird dieses Kriterium mit lediglich 1 Punkt gewertet.

Search.ch

Rating		**Total 8 Punkte**
Rechtliches	Eine Nutzung der Daten ist nicht ausgeschlossen	(2 Punkte)
Unterstützte Länder	CH	(1 Punkt)
Lizenzierung	Registration für den Service erforderlich	(1 Punkt)
Kostenpflichtig	Nein	(3 Punkte)
Typ	REST Webservice	(1 Punkt)

Zusätzliches

Es ist vorgängig eine Registration notwendig, die jedoch nur wenige Augenblicke dauert. Darin ist unter anderem der Verwendungszweck anzugeben. Durch die Registrierung erhält man einen Schlüssel, der dann bei jedem Aufruf mitzugeben ist. In den AGB ist spezifiziert dass die Login-Daten (und somit der Key) nicht durch andere Personen genutzt werden darf, deshalb müsste man vermutlich für jeden Kunden einen eigenen Key registrieren lassen. Dazu sollten aber noch weitere Abklärungen unter Einbezug von search.ch getroffen werden.

Rechtliches

Auszug aus den Nutzungsregeln von search.ch[2]:
„Der Kunde verzichtet ausdrücklich darauf, sich mehr als einmal anzumelden oder unter seiner Adresse anderen Personen den Zugang zu den einzelnen Diensten zu ermöglichen.“
„4.4 Es ist unzulässig, search.ch als Meta–Suchmaschine einzusetzen.“

Fazit

Search.ch bietet einen professionellen Webservice an, der noch dazu kostenlos ist. Leider sind die Telefonbuchdaten auf die Schweiz und Liechtenstein begrenzt und es ist eine vorgängige Registrierung notwendig. Trotzdem bietet dieser Service alle Funktionen die für unser Anwendungszweck benötigt werden.

[2] Quelle: [LINK] Nutzungsbedingungen von search.ch

Directories.ch

Rating		**Total 6 Punkte**
Rechtliches	Nutzung der Daten wird nicht ausgeschlossen	(2 Punkte)
Unterstützte Länder	CH / DE / FR / AT	(2 Punkte)
Lizenzierung	Registration erforderlich	(1 Punkt)
Kostenpflichtig	Ja	(0 Punkte)
Preise	CH: 2 Rp / Datensatz DE/FR/AT: 10 Rp / Datensatz CHF 250.- Mindestbetrag / Monat	
Typ	SOAP Webservice	(1 Punkte)

Fazit

Directories.ch bietet eine vollumfänglich professionelle Lösung für die Telefondatenabfrage aus Deutschland, Frankreich, Österreich sowie der Schweiz. Leider ist dieser Dienst kostenpflichtig und relativ teuer. Durch die Statistik-Funktion die dem „Master-User“ zur Verfügung steht sind zwar die verursachten Kosten auf den einzelnen Kunden zurückzuführen und somit weiterzuverrechnen, allerdings dürfte das einiges an Mehraufwand mit sich bringen.

Dastelefonbuch.de

Rating		**Total 1 Punkt**
Rechtliches	Nutzung der Daten verboten	(0 Punkte)
Unterstützte Länder	DE	(1 Punkt)
Lizenzierung	Arbeitsplatz-Lizenz pro Rechner nötig	(0 Punkte)
Kostenpflichtig	ja	(0 Punkte)
Preise	~15€ / Einzelplatzlizenz ~55€ für Netzwerkversion (Bis 30 Personen)	
Typ	Application API	(0 Punkte)

Rechtliches

Auszug aus den Nutzungsbedingungen von DasTelefonbuch.de[3]:
„So ist insbesondere die vollständige, teilweise oder auszugsweise Verwendung des Verzeichnisses DasTelefonbuch im Internet für gewerbliche Adressenverwertung oder als Unterlage bzw. Hilfsmittel für die Zusammenstellung oder Ergänzung von Teilnehmer-, Adress- oder anderen Verzeichnissen sowie das Auslesen der Daten im Internet zu den vorgenannten Zwecken sowie zu Zwecken sonstiger kommerzieller Verwendung nicht gestattet und wird von den Anbietern nach geltendem Recht unter Ausschöpfung des Rechtsweges verfolgt.“

Fazit

DasTelefonbuch.de bietet eine umfangreiche Telefondatenbank von Deutschland, sowie eine professionelle API um auf diese Daten zuzugreifen an. Leider handelt es sich hierbei um eine reine Application API, sodass ein Produkt von DasTelefonbuch entweder auf jedem Rechner installiert sein muss der die API verwenden will, oder aber nur eine einzelne Installation auf dem Server von dem aus die Suche gestartet wird – wobei man bei dieser Lösung nochmal die Lizenzierung genau betrachten müsste.

[3] Quelle: [LINK] Nutzungsbedingungen von dastelefonbuch.de

Local.ch

Rating		Total 1 Punkt
Rechtliches	Nutzung der Daten verboten	(0 Punkte)
Unterstützte Länder	CH	(1 Punkt)
Webservice	Keiner. (Zurzeit nur Städte-Suche)	(0 Punkte)

Zusätzliches

Laut unzuverlässigen, aber sehr aktuellen Informationen im Netz wird es in naher Zukunft auch von local.ch eine API für Telefonbucheinträge geben. Zitat: „Wir hatten bis vor kurzem eine API bei local.ch (Telefonbuch und Gelbe Seiten Schweiz), bis wir gesetzlich nicht mehr durften. Seit letzter Woche dürfen wir wieder eine API anbieten und werden das auch bald machen. […]“[4]

Rechtliches

Auszug aus den AGB von local.ch[5]:

„Die Weiterverwendung und die Weitergabe von Daten (Elektronisch, auf Papier, etc.) ohne vorherige schriftliche Zustimmung durch local.ch AG ist strikte untersagt.“

Fazit

Zurzeit bietet local.ch keinen Webservice für die Telefonbuchdaten an. Da die AGB die Weiterverwendung der Daten verbietet, ist das parsen der HTML Seite aus meiner Sicht leider auch keine Option. Local.ch müsste man vielleicht nach der vermeintlichen Veröffentlichung des Webservices nochmal neu Evaluieren, jedoch gibt es zurzeit keine Anzeichen dass dies noch während der Projektlaufzeit passieren wird.

Tel.ch / Telefonbuch.ch

Rating (1-10, wobei 10 das Beste)		Total 1 Punkt
Rechtliches	Nutzung der Daten verboten	(0 Punkte)
Unterstützte Länder	CH	(1 Punkt)
Webservice	Keiner	(0 Punkte)

Rechtliches

Auszug aus den ANB von tel.ch[6]:

„Jede zweckfremde Nutzung oder Verwertung der Daten ist unzulässig. Insbesondere ist die vollständige, teilweise oder auszugsweise Verwendung des Verzeichnisses für gewerbliche Adressenverwertung oder als Hilfsmittel für die Erstellung oder Ergänzung von Teilnehmer-, telefon-, Adress- und ähnlichen Verzeichnissen oder die Verwendung der Daten für telefonische Auskunftsdienstleistungen und das Auslesen der Daten mit irgendwelchen Verfahren oder das automatisierte Abrufen zu diesen oder anderen Zwecken kommerzieller Verwertung nicht erlaubt.“

Fazit

Tel.ch bietet keinen Webservice an um auf die Telefonbuchdaten zuzugreifen und verbietet über die ANB ausserdem ziemlich eindeutig die Verwendung der Daten für unsere Zwecke.

---

[4] Quelle: [LINK] Kommentar weiter unten von „Moritz Adler“
[5] Quelle: [LINK] AGB von local.ch
[6] Quelle: [LINK] ANB von tel.ch

Twixtel.ch

Rating (1-10, wobei 10 das Beste)		Total 1 Punkt
Rechtliches	Nutzung der Daten verboten	(0 Punkte)
Unterstützte Länder	CH	(1 Punkt)
Webservice	keiner	(0 Punkte)

Rechtliches

Auszug aus den Nutzungsbestimmung von twixtel.ch[7]:
„Das Vervielfältigen, Vertreiben, Übertragen, Bereithalten zum Abruf, Übermitteln, Verändern, Verknüpfen und jedes sonstige Verwenden dieser Inhalte (oder Teile dieser Inhalte) für öffentliche oderkommerzielle Zwecke ist nur mit vorgängiger schriftlicher Zustimmung von Twix gestattet.“

Fazit

Twixtel.ch bietet keinen Webservice an um auf die Telefonbuchdaten zuzugreifen und verbietet über die Nutzungsbestimmungen indirekt die Nutzung dieser Daten für Kommerzielle Zwecke, sodass ein Auslesen/Parsen der HTML Seite wohl auch vom rechtlichen Standpunkt kritisch ist.

Empfehlung

Aufgrund der Bewertung der diversen Telefonbuch Provider ergibt sich ein relativ klares Bild. Die allgemeine Situation der kostenlosen Webservices ist in der Schweiz noch sehr begrenzt und die Daten werden kaum freiwillig zur Verfügung gestellt. Der Spitzenreiter Search.ch bietet eine professionelle Webservice Schnittstelle und ist noch dazu kostenlos. Die Nachteile von Search.ch sind einerseits die notwendige Registrierung, andererseits die Begrenzung der Daten auf die Schweiz. Trotzdem bin ich der Meinung dass Search.ch die beste Wahl bleibt, um die Anforderungen von ONEOFFIXX zu decken.

[7] Quelle: [LINK] Nutzungsbestimmungen von twixtel.ch

Anforderungsspezifikation

Dokumentinformationen

Zweck
Dieses Dokument beschreibt die Anforderungen an die Applikation.

Gültigkeitsbereich
Dieses Dokument ist für den kompletten Verlauf des Projektes gültig.

Übersicht
Dieses Dokument gibt Auskunft aber die nötigen Anforderungen des Projekts.

Änderungsgeschichte

Datum	Version	Änderung	Autor
3.4.2011	0.1	Erstellung des Dokuments	Ramon Müller
11.4.2011	0.2	Anforderungen beschrieben	Ramon Müller
13.4.2011	0.3	User Stories hinzugefügt	Ramon Müller
13.4.2011	1.0	Review & Freigabe	Ramon Müller

Allgemeine Beschreibung

Produkt Funktion

Das Produkt ermöglicht direkt aus dem Office heraus Zugriff auf Adressbestände im Internet (z.B. online-Telefonbücher) oder aus dem Firmennetz (Spezifischer Server mit Zugriff auf TwixTel, Sharepoint oder interne Kundendatenbanken).

Benutzer Charakteristik

Das Endprodukt ONEOFFIXX wird vor allem von Anwendern im Businessumfeld genutzt. Die Anwender verfügen über das Know-how Office und die darauf aufbauende Software ONEOFFIXX zu bedienen. Diese Zielgruppe ist vor allem für die Generierung der Fehlermeldungen und allfällige andere Meldungen des Services zu beachten.

Der Service für die Beschaffung der Adressdaten wird von Softwareentwicklern verwendet um ihn im Produkt ONEOFFIXX einzubinden. Sie verfügen über das Know-How einen solchen Service anzusteuern sowie APIs zu lesen und zu verstehen.

Einschränkungen

- Die Einbindung in das Produkt ONEOFFIXX ist nicht Teil dieses Projekts
- Es wird sich in erster Linie auf die Adresssuche in der Schweiz konzentriert, weitere Länder können später noch zusätzlich unterstützt werden.

Abhängigkeiten

Das Produkt basiert auf .NET 4 und ist somit nur auf Windowssystemen verfügbar.

Ausserdem wird das „Managed Extensibility Framework“ eingesetzt wodurch eine Abhängigkeit zustande kommt.

Spezifische Anforderungen

Funktionale Anforderungen

Adressbestand Suche

Es muss möglich sein eine Adresse anhand von Vorname, Nachname sowie Telefonnummer zu suchen. Es soll stets eine Adresse zurückgegeben werden, vorausgesetzt diese Adresse existiert in der Datenbank des Telefonbuch Providers.

Einschränkungen der Suche

Eine Suche soll eingeschränkt werden können aufgrund von Postleitzahl, Ort sowie Geschäfts-/ oder Privatsuche.

Schnittstellen

Die Schnittstellen sollen so gewählt werden, dass sie möglichst leicht erweiterbar sind um in Zukunft weitere ausgelesene Felder oder spezifischere Suchanfragen mit möglichst wenig Aufwand zu unterstützen.

Automatische Erkennung
Die Adresssuche in online Telefonbüchern soll direkt nach der Installation auf einem spezifischen Firmeninternen Server in der Applikation ONEOFFIXX verfügbar sein. Dies bedeutet dass der Service gewisse Metadaten über sich selbst verfügbar machen muss sobald er gestartet worden ist, damit er anschliessend automatisch in der Applikation ONEOFFIXX eingebunden werden kann.

Fehlerfälle
Der Service soll für diverse Fehlerfälle sinnvolle Meldungen zurückgeben. Dies betrifft:

- Keine Treffer für die gewählten Suchparameter
- Kein Internetzugriff
- Fehler beim parsen des Atom-Feeds
- Abbruch des Suchvorgangs

Nichtfunktionale Anforderungen

Bedienbarkeit
Der bereitgestellte Service soll möglichst intuitiv bedienbar sein und mithilfe der API in unter einer Stunde soweit verstanden werden können, dass er erfolgreich angewendet werden kann.

Zuverlässigkeit
Der zu erarbeitende Service soll sehr zuverlässig verfügbar sein. Sobald der Service gestartet wurde soll er in der Applikation verfügbar sein. Ausserdem kann es zu einer Replikation kommen, wodurch der Service auch lokal auf einem Notebook läuft, wenn dieses zum Beispiel vom Firmennetz getrennt wird. Dies soll die Applikation bemerken und möglichst reibungslos auf den lokalen Service umschalten.

Erweiterbarkeit
Das Service-Framework (MEF) soll so aufgebaut werden, dass Zusätzliche Services / Plugins einfach hinzugefügt, eingebunden und verwendet werden können.

Identifikationsschlüssel / Lizenzierung¨
Um den Webservice von search.ch verwenden zu können ist ein Identifikationsschlüssel erforderlich. Dieser muss vorgängig bei search.ch registriert werden.

User Stories

Business Anwender: Adresssuche
Ich als [Business Anwender] möchte über ein Menü eine Adresssuche starten. Ich gebe meine Suchparameter ein und drücke auf „Suchen“ und erwarte eine Liste mit allen gefundenen Treffern worin ich anschliessend die richtige auswähle, um mit ihr in ONEOFFIXX weiterzuarbeiten.

b) Falls bei der Anfrage ein Fehler auftritt möchte ich eine möglichst detaillierte Fehlermeldung, die ich verstehen kann.

Software Engineer: Benutzen des Services

Ich als [Software Engineer] möchte mit Hilfe der Dokumentation innert einer Stunde in der Lage sein den Service soweit zu verstehen dass ich ihn in ein bestehendes Programm einbinden kann.

Domainanalyse

Dokumentinformationen

Zweck
Dieses Dokument beschreibt die Analyse der Prozesse sowie deren Lösungsansätze in diesem Projekt.

Gültigkeitsbereich
Dieses Dokument ist für den kompletten Verlauf des Projektes gültig.

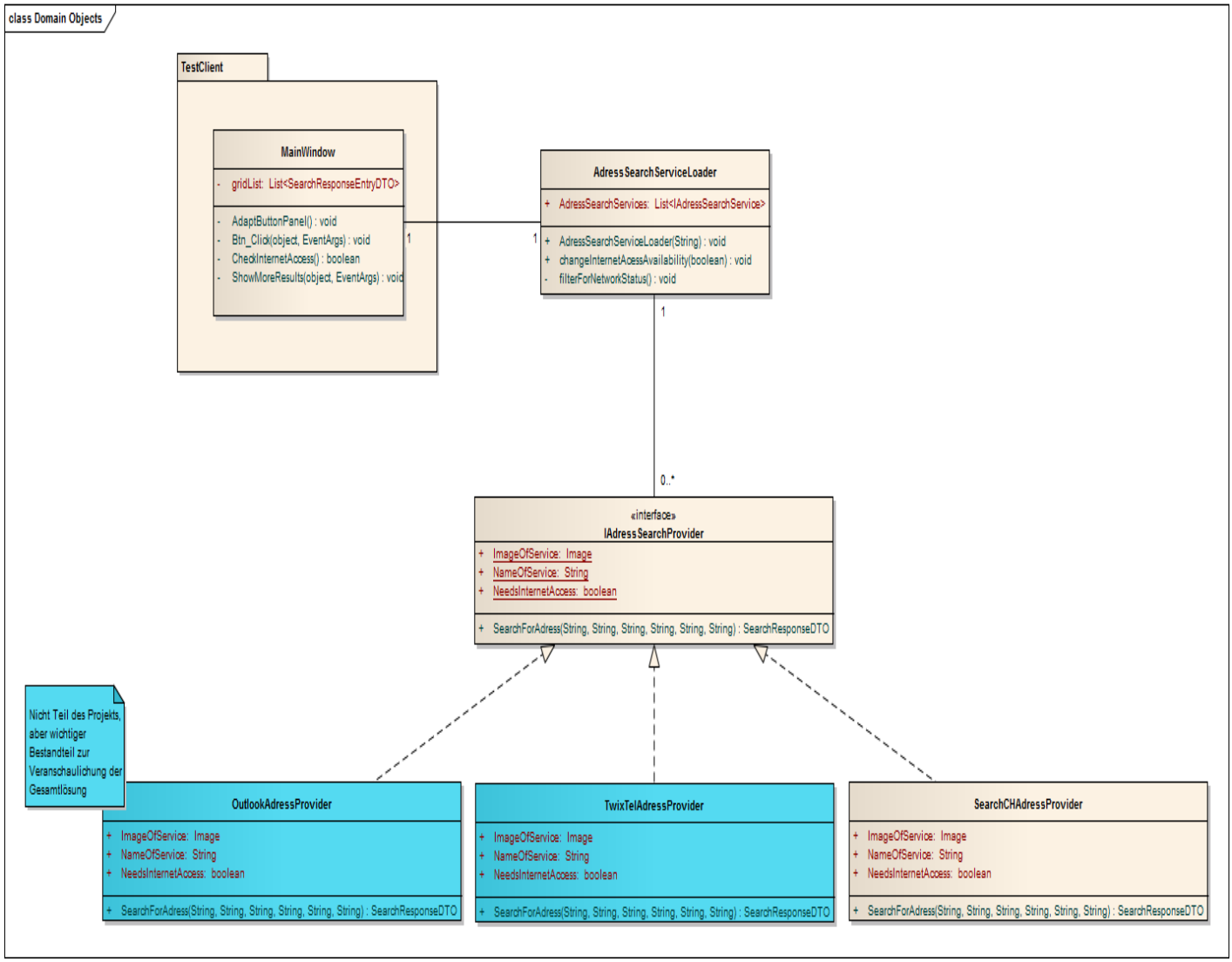
Übersicht
Dieses Dokument zeigt die wichtigsten Prozesse des Projektes auf und zeigt wie diese umgesetzt werden.

Änderungsgeschichte

Datum	Version	Änderung	Autor
2.4.2011	0.1	Erstellung des Dokuments	Ramon Müller
10.5.2011	0.2	Kapitel Domain Model & System Sequenz Diagramme	Ramon Müller
7.6.2011	1.0	Review & Freigabe	Ramon Müller

Domain Model

Das Domain Model beschreibt das Zusammenspiel der wichtigsten Komponenten innerhalb des Projekts.



MainWindow

Die MainWindow Klasse ist für die Aufgaben im User Interface sowie die Verarbeitung und Umsetzung von Adressabfragen auf die verschiedenen AdressSearchProvider verantwortlich.

AdressSearchServiceLoader

Die AdressSearchServiceLoader Klasse des Test Clients verwaltet die verschiedenen AdressSearchProvider die dem Benutzer zur Verfügung gestellt werden. Er kümmert sich um das Auffinden, sowie den Import der verschiedenen Provider. Er ist ausserdem dafür verantwortlich nur jene AdressSearchProvider in die Auswahl aufzunehmen die zu diesem Zeitpunkt verwendbar sind, abhängig zum Beispiel vom Status der Internetverbindung.

IAdressSearchProvider

Das Interface IAdressSearchProvider stellt die Schnittstelle zu den verschiedenen AdressSearchProvider dar. Anhand dieses Interfaces weiss das Managed Extensibility Framework

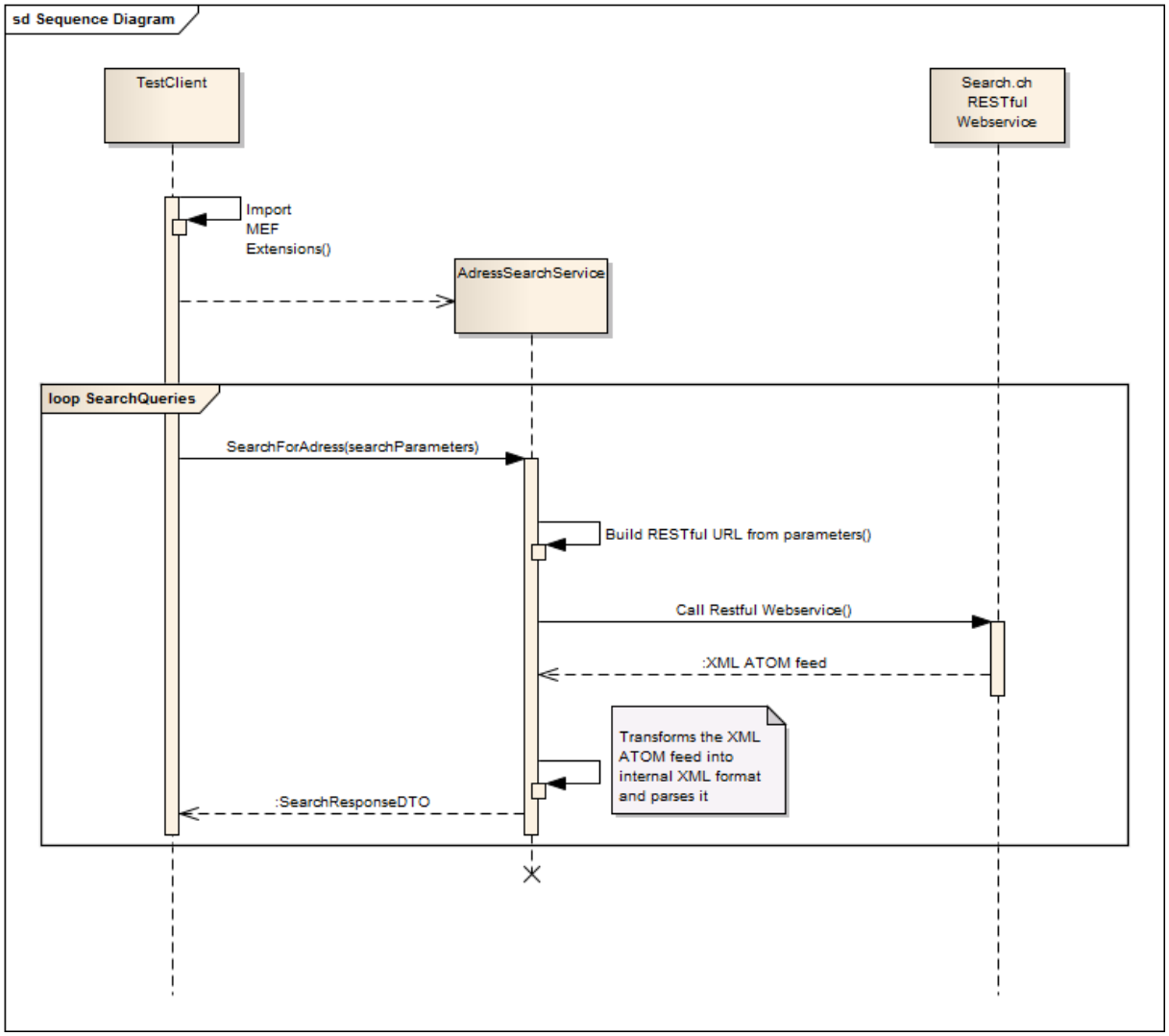
welche Klassen wohin verdrahtet werden müssen. Der AdressSearchServiceLoader importiert alle Klassen die dieses Interface Exportieren.

SearchCHAdressProvider

Dies ist die Implementation eines AdressProviders. Er ist dafür zuständig die Suche schlussendlich durchzuführen. Bei diesem Beispiel wird er einen RESTful Service von Search.ch mit den Suchparametern aufrufen und die Antwort anschliessend in geeigneter Form zurücksenden.

System Sequenzdiagramm

Das Sequenzdiagramm zeigt einen vereinfachten Ablauf einer Adressabfrage auf einen AdressSearchProvider. In diesem Beispiel ist der verwendete AdressSearchProvider der SearchCHAdressProvider, der anschliessend auf die externe Ressource von search.ch zugreift.

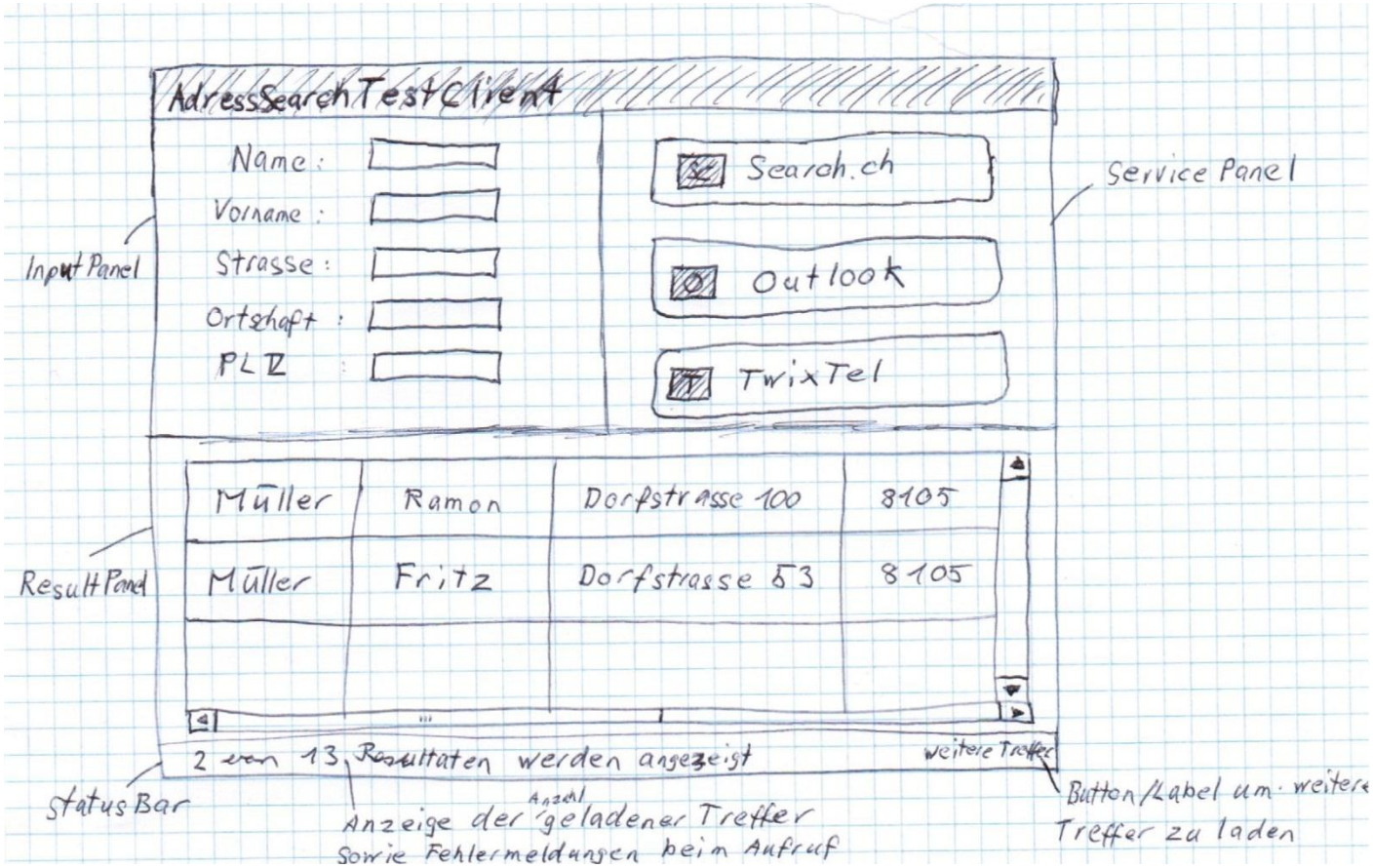


Paperprototyp für den Test Client

Der Paperprototype wurde mit Bedacht auf eine dynamische Generierung der AdressSearchProvider-Buttons ausgerichtet. Das ServicePanel wird dynamisch mit den verschiedenen Provider-Buttons aufgebaut. Die Buttons enthalten ein Icon sowie den Text, die beide von den AdressSearchProvidern stammen. So kann also der AdressSearchProvider zu einem Teil selber bestimmen wie er dargestellt wird.

Die Statusbar soll dem Benutzer jederzeit Informationen zur Suchabfrage liefern oder über aufgetretene Fehler informieren.

Durch einen Klick auf das „Weitere Treffer…“-Label werden zusätzlich Treffer geladen und im ResultPanel angezeigt.



Software Architecture Document

Dokumentinformationen

Zweck
Dieses Dokument beschreibt die technische Umsetzung und Überlegungen die während des Projekts gemacht wurden.

Gültigkeitsbereich
Dieses Dokument ist für den kompletten Verlauf des Projektes gültig.

Übersicht
Dieses Dokument zeigt die wichtigsten Prozesse des Projektes auf und zeigt auf wie diese umgesetzt wurden.

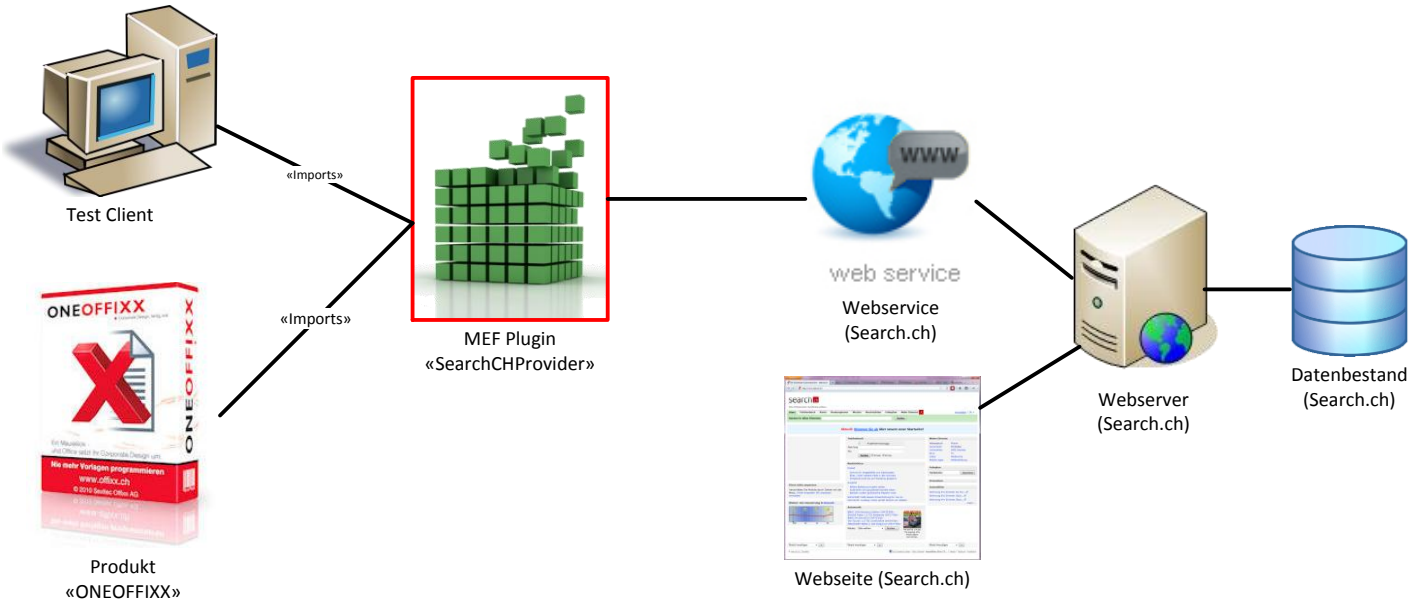
Änderungsgeschichte

Datum	Version	Änderung	Autor
2.4.2011	0.1	Erstellung des Dokuments	Ramon Müller
6.6.2011	0.2	Kapitel Architektonische Ziele	Ramon Müller
8.6.2011	0.3	Logische Architektur und Codeauswertungen	Ramon Müller
9.6.2011	0.4	Architektonische Übersicht	Ramon Müller
9.6.2011	1.0	Review & Freigabe	Ramon Müller

Referenzen

Nr.	Dokument	Autor
1	http://mef.codeplex.com/	CodePlex
2	http://www.oneoffixx.ch	Sevitec AG
3	http://www.w3schools.com/xsl/	W3schools
4	http://msdn.microsoft.com/en-us/library/bb387098.aspx	msdn
5	http://tel.search.ch/api/help	Search.ch

Architektonische Übersicht



Das Projekt beruht auf dem „Managed Extensibility Framework“ (MEF) [1]. Der Test Client und später auch das Produkt ONEOFFIXX[2] von Sevitec importieren die einzelnen AdressProvider als MEF-Plug-Ins. Diese AdressProvider greifen dann auf ihre vordefinierten Ressourcen zu, in diesem Fall auf einen Webservice von Search.ch

Architektonische Ziele

Hohe Erweiterbarkeit

Eine hohe Erweiterbarkeit wird durch die Nutzung des Managed Extensibility Framework erreicht. Die einzelnen AdressSearchProvider werden lediglich über ein Interface Importiert. Ein neuer AdressSearchProvider muss also lediglich ein Interface Exportieren um in die Sammlung aufgenommen zu werden. Die Kommunikation findet dann über dieses Interface statt.

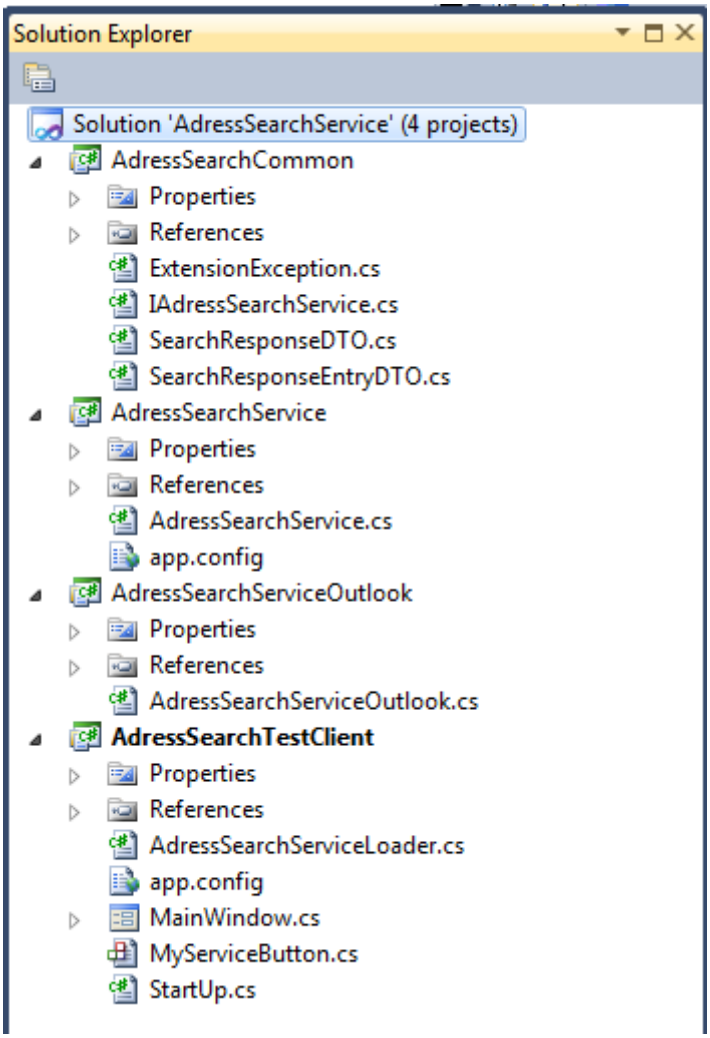
Möglichst unabhängig von den externen Ressourcen

Die Unabhängigkeit von den externen Ressourcen war in diesem Fall besonders wichtig. Falls sich die Struktur der Antwort des Webservices von Search.ch ändert, müsste ansonsten jedes Mal eine neue Version des AdressSearchProviders ausgeliefert werden. Ich habe mich deshalb dazu entschlossen eine weitere Schicht einzubauen und die Antwort des Webservices erst mit einer XSL-Transformation[3] umzuformen, bevor diese schliesslich geparst wird. Dadurch erlangt man eine zusätzliche Abstraktionsschicht welche den AdressSeachProvider weiter von der externen Ressource entkoppelt. Bei einer Änderung am Webservice von Search.ch muss lediglich die Transformationsdatei ausgetauscht werden, was sogar zur Laufzeit geschehen kann.

AdressProvider soll Robust sein

Da die AdressProvider zu Zeit synchron aufgerufen werden ist es sehr wichtig dass diese nicht einfach abstürzen oder sich aufhängen, denn dadurch würde die ganze Applikation lahm gelegt. Deshalb wurde darauf geschaut ein gutes Error Handling umzusetzen sowie geeignete Timeouts einzubauen.

Logische Architektur



AdressSearchCommon

Dieses Projekt wird sowohl von den verschiedenen AdressSearchServices sowie vom Test Client referenziert. Es dient als gemeinsame Basis für das gemeinsame Interface das für den Import/Export der MEF Plug-Ins verwendet wird, sowie der beiden Daten-Klassen SearchResponseDTO und SearchResponseEntryDTO die als Antwort auf eine Suchanfrage von den AdressSearchServices zurückgeschickt wird.

AdressSearchService

Dieses Projekt enthält eine Implementation des AdressSearchService. Es handelt sich hierbei um die Implementation für die Adresssuche auf Search.ch. Es enthält lediglich die Implementierende Service-Klasse, die das Interface IAdressSeachService Exportiert.

AdressSearchServiceOutlook

Hierbei handelt es sich um eine Dummy Implementation eines AdressSearchServices. Er wird als zweiter Service im Test Client angezeigt, um die Unterschiede zwischen einem lokalen und einem Online-Service aufzuzeigen. Die Implementation dieses Dummy Outlook Service gibt lediglich einen einzigen Datensatz zurück und ist nur für den Test Client von bedeutung.

AdressSearchTestClient

Bei diesem Projekt handelt es sich um den Test Client, der die Anwendung von MEF und den verschiedenen Services veranschaulichen soll.

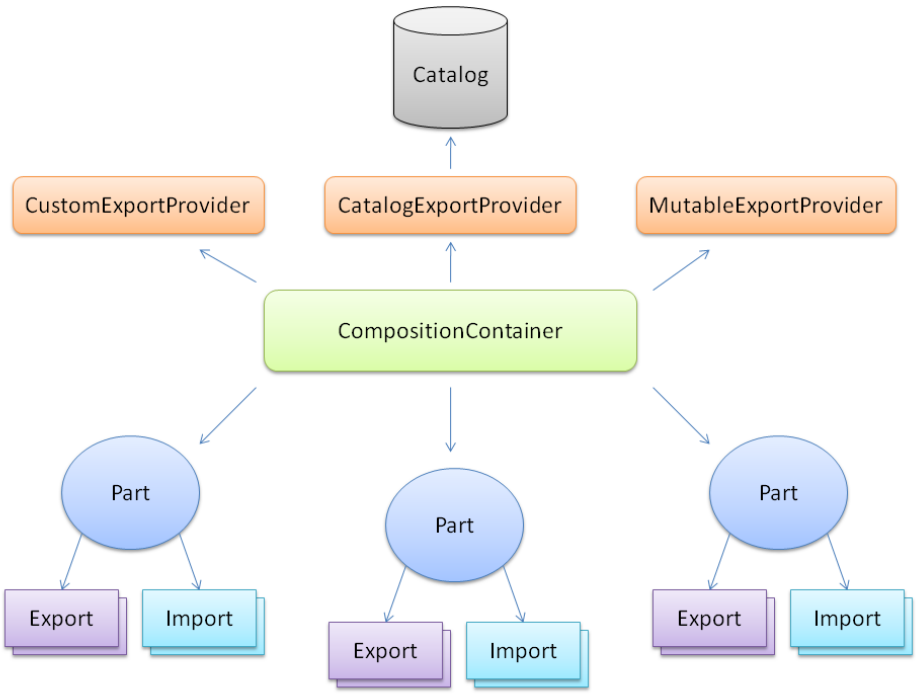
Er enthält die MainWindow Klasse, die ein Windows-Forms GUI für die Darstellung enthält und die Service-Aufrufe realisiert. In der Regel würde man diese Funktionalität in ein ViewModel ausgliedern, da die Logik allerdings kaum eine eigene Klasse rechtfertigt habe ich mich dazu entschlossen die Aufrufe direkt aus der View heraus zu starten.

Die MyServiceButton Klasse, ist eine Eigenimplementation des Windows-Form-Buttons. Diese Eigenimplementation ist nötig, da sich jeder Button merken muss zu welchem Service er gehört. Dies wird über ein Attribut ServiceID realisiert und spezifiziert die ID des Services den dieser in der Auflistung des Loaders einnimmt.

Der AdressSearchServiceLoader ist die Klasse die alles rund um die Services handhabt. Er importiert die verschiedenen Services über das IAdressSearchService Interface, die entsprechenden Assemblys findet er in einem vordefinierten Verzeichnis, das ihm bei der Instanzierung übergeben wird. Sobald er über den Wechsel des Internet Status unterrichtet wird beginnt er damit die diversen Services zu filtern, damit nur die Services angezeigt werden die für den aktuellen Internet Status auch verfügbar sind.

Managed Extensibility Framework

Das MEF ist ein Framework womit man mit möglichst wenig Aufwand ein vollständiges Plug-In-Modell unterstützen kann. Das Framework ist relativ neu und doch schon sehr mächtig: es unterstützt zum Beispiel Abhängigkeiten verschiedener Plug-Ins untereinander oder die Mehrfachverwendung von Plug-Ins in verschiedenen Applikationen. In diesem Projekt wurde aber nur das Grundkonzept, also das Plug-In-Modell eingesetzt. Es dient enorm der Erweiterbarkeit der Applikation wenn diese Modular aufgebaut ist.



Das MEF besteht grundsätzlich aus drei wichtigen Objekten:

Parts

Die Parts sind die eigentlichen Plug-Ins. Sie Exportieren stets ein Interface, dass die Gegenseite (Zum Beispiel der Client) dann importiert. In ihnen findet die eigentliche Logik ihren Platz.

Catalogs

Der Katalog ist dafür zuständig die Plug-Ins zu finden. Es gibt verschiedene Kataloge die die Parts aus verschiedenen Quellen finden sollen. Der AssemblyCatalog sucht zum Beispiel Parts in einem angegebenen Assembly, während der DirectoryCatalog ein Verzeichnis nach Parts durchsucht.

Container

Der Container ist dafür zuständig die verschiedenen Parts richtig zu verdrahten, er kümmert sich also darum dass die Exports mit den richtigen Imports verknüpft und korrekt Instanziert werden.

XSL Transformation

Um möglichst unabhängig von Änderungen des Webservices von Search.ch zu sein, habe ich entschieden das Resultat - das der Webservice in Form eines XML ATOM Feeds sendet - zuerst in ein handlicheres internes Format umzuwandeln, bevor es geparst wird. Dies geschieht mithilfe von XSL Transformation.

```
            <role><xsl:value-of select="openSearch:Query/@role"/></role>
            <startpage><xsl:value-of select="openSearch:Query/@startPage"/></star
        </query>
        <results>
            <xsl:for-each select="f:entry">
                <result>
                    <type><xsl:value-of select="tel:type"/></type>
                    <name><xsl:value-of select="tel:name"/></name>
                    <firstname><xsl:value-of select="tel:firstname"/></firstname>
                    <street><xsl:value-of select="tel:street"/></street>
                    <streetno><xsl:value-of select="tel:streetno"/></streetno>
                    <zip><xsl:value-of select="tel:zip"/></zip>
                    <city><xsl:value-of select="tel:city"/></city>
                    <canton><xsl:value-of select="tel:canton"/></canton>
                    <phone><xsl:value-of select="tel:phone"/></phone>
                </result>
            </xsl:for-each>
        </results>
    </search>
</xsl:template>
l:transform>
```

Das ursprüngliche XML wird also zuerst mithilfe der Transformationsdatei (ausschnitt oben) in eine interne Struktur transformiert.

Linq to XML

Linq to XML [4] wird verwendet, um die transformierte XML-Antwort des Webservices zu parsen. Es werden dabei zuerst die Informationen zur Suchabfrage selbst aus dem internen XML ausgelesen und in die Daten-Klasse „SearchResponseDTO“ abgefüllt. Es werden dabei folgende Informationen ausgelesen und abgespeichert:

Informationen zur Suchanfrage in der XML-Antwort	
TotalResults	Die gesamte Anzahl Resultate zum eingegebenen Suchbegriff
StartIndex	Der Index des ersten Resultats das zurückgegeben wurde. Wird für das Paging verwendet.
ItemsPerPage	Anzahl Resultate die in dieser Antwort enthalten sind
SearchTerm	Der Suchbegriff nach dem gesucht wurde, wobei hier die Einschränkungen bezüglich Ort, Strasse, PLZ oder Kanton nicht aufgelistet werden!
Role	Die Rolle der Anfrage, in der Regel „Request“ als Antwort auf eine Anfrage

Anschliessend werden die Resultate an sich geparst. Für jeden Eintrag wird dazu ein SearchResponseEntryDTO erstellt und entsprechend abgefüllt. Zum Schluss wird dieses Objekt in die Collection der SearchResponseDTO abgelegt. Die einzelnen Daten die zu jeder Person ausgelesen werden sind folgende:

Daten der Suchresultate in der XML-Antwort	
Type	Der Typ dieses Eintrags, entweder Person oder Organisation
Name	Der Name der Person oder Organisation
Firstname	Der Vorname der Person
Street	Die Strasse der Person oder Organisation
Streetno	Die Hausnummer der Person oder Organisation
Zip	Die Postleitzahl der Person oder Organisation
City	Der Ort der Person oder Organisation
Canton	Der Kanton in der die Person oder die Organisation ihren Sitz hat
Phone	Die Telefonnummer der Person oder Organisation

Es wäre weiterhin möglich noch einige zusätzliche Daten auszulesen, wie zum Beispiel Mädchenname der Person, Beruf der Person, Email, Fax oder Webseite. Allerdings sind diese Daten optionale Parameter die nicht immer mit angegeben werden. Sollten in Zukunft weitere Datenfelder interessant werden, ist eine Unterstützung dieser aber kein grosser Aufwand mehr. Weitere Informationen zur Bedienung des Webservices finden sich unter Referenzen [5].

Internet Status Überwachen

Im Test Client wird stets der Status der Internet Verbindung überwacht, um die Services sofort dem aktuellen Status anzupassen. Dazu sind zwei Mechanismen nötig:

1. Einmaliges auslesen beim Clientstart

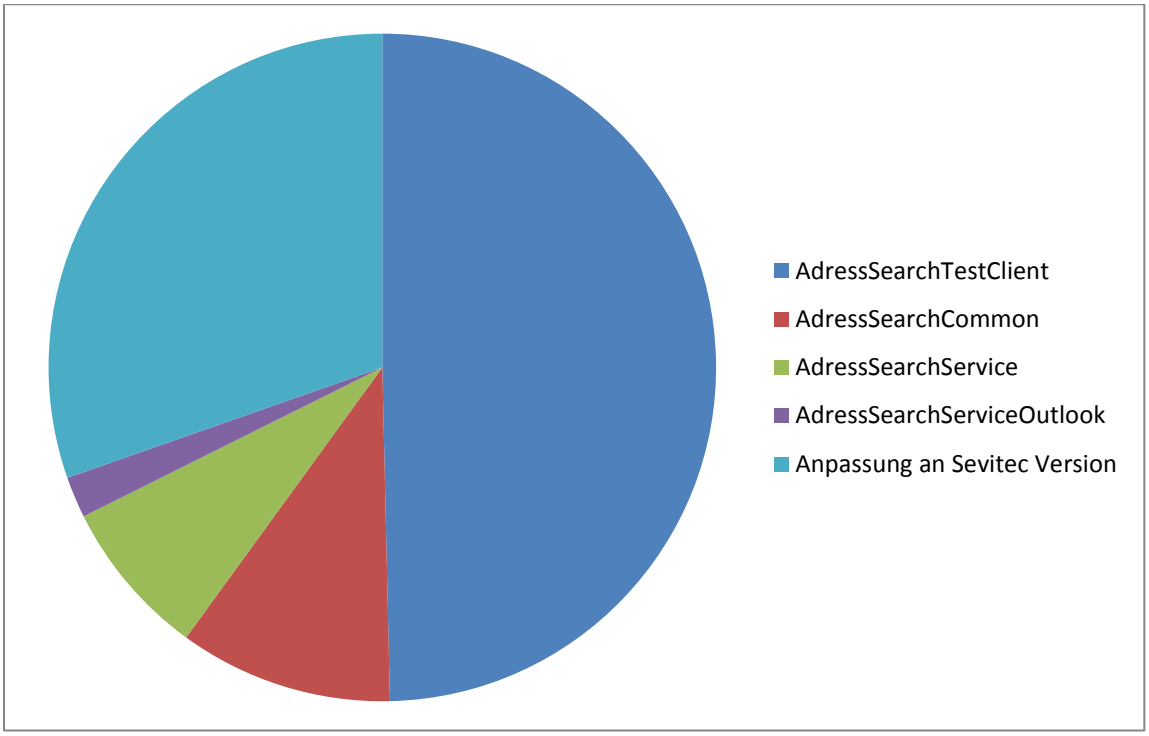
Beim Starten wird automatisch eine Methode checkInternetAccess aufgerufen. In dieser Methode wird ein Ping Request auf die Webseite tel.search.ch abgeschickt. Das Timeout beträgt 2 Sekunden, wenn bis dahin keine Antwort gekommen ist gilt die Verbindung als „Down“, ansonsten als „Up“. Dies wird anschliessend dem Loader gemeldet, der sich dann um die Filterung der Services kümmert.

2. *Registrieren von Events bei Änderungen am Internet Status*

Das Assembly *System.Net.NetworkInformation* enthält für diese Anforderung die Klasse NetworkChange. Bei dieser Klasse kann man sich für die Events *NetworkAvailabilityChanged* und *NetworkAddressChanged* registrieren. Mithilfe des ersten Events wird nun also der Status des Netzwerks überwacht. Sobald sich dieser ändert wird der neue Status dem Loader gemeldet und die Services werden entsprechend gefiltert.

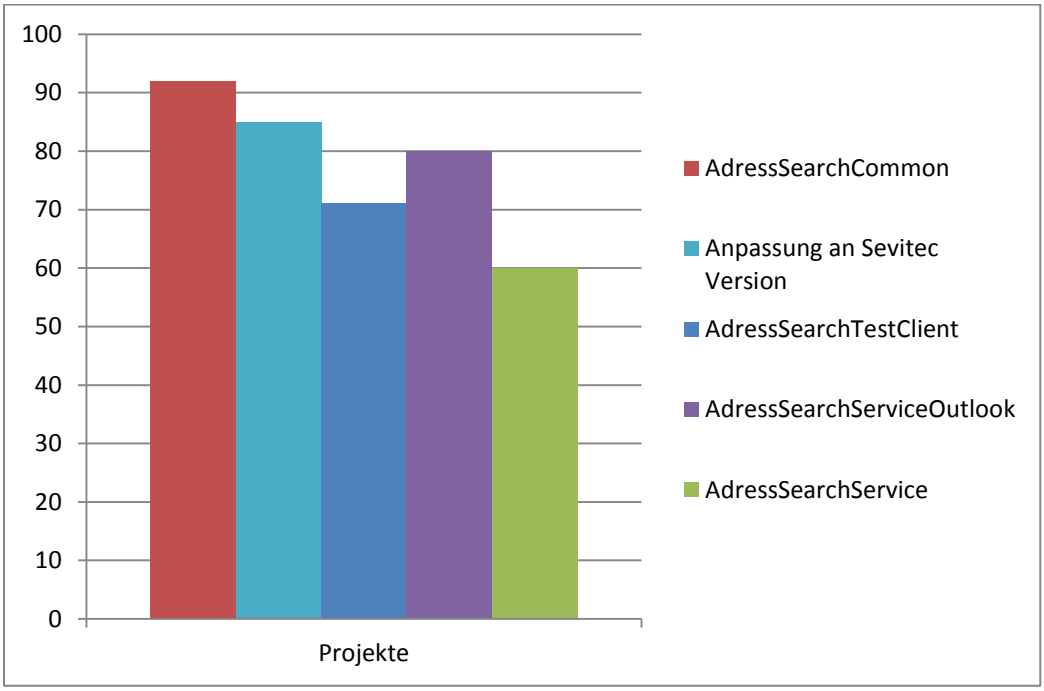
Codeauswertungen

Lines of Code



Es ist kaum verwunderlich dass der TestClient die meisten Codezeilen beinhaltet, denn dort liegt auch die meiste Logik begraben. Die Angepasste Version für Sevitec ist der zweitgrösste Brocken, da bei dieser Version etwas mehr Overhead aufgrund eines allgemeineren Interfaces nötig war.

Maintainability Index



Der Maintainability Index ist allgemein relativ hoch. Einzig beim AdressSearch Service ist er auf 60 Punkte runter. Dies ist mit der Methode SearchForAdress zu erklären, die leider durch das Parsen der XML-Antwort mit LinqToXML bei dieser Metrik nicht gut abschneidet.

Testprotokolle

Dokumentinformationen

Zweck
Dieses Dokument hält die durchgeführten Testfälle des Projekts fest.

Gültigkeitsbereich
Dieses Dokument ist für den kompletten Verlauf des Projektes gültig.

Übersicht
Dieses Dokument zeigt die durchgeführten Tests und ihre Resultate auf.

Änderungsgeschichte

Datum	Version	Änderung	Autor
9.5.2011	0.1	Erstellung des Dokuments	Ramon Müller
13.5.2011	0.2	Testfälle zur ersten Version des Search.ch Services	Ramon Müller
20.5.2011	0.2	Testfälle zum TestClient	Ramon Müller
30.5.2011	0.3	Testfälle zum SearchCHProvider	Ramon Müller
5.6.2011	0.4	Testfälle zum finalen TestClient	Ramon Müller
7.6.2011	0.5	Review & Ausbesserungen	Ramon Müller
9.6.2011	1.0	Freigabe	Ramon Müller

Tests vom 13.5.2011 - erste Version der Search.ch-Servicemethode

Systemtests

Überprüfen der benutzten RESTful URL (mit nur einem Parameter)	
Beschreibung	Die RESTful URL wird aus den verschiedenen Suchparametern zusammengestellt. Es soll überprüft werden ob das zusammenbauen der verschiedenen Parameter korrekt funktioniert. Dazu wird der Service mit den Parametern aufgerufen und anschliessend die vollständig zusammengebaute URL per Debugging ausgelesen. Diese wird dann in den Browser kopiert und die Antwort des Webservers analysiert.
Erwartetes Resultat	Der Webservice antwortet mit einem korrekten XML ATOM Feed der die Resultate der Suchanfrage enthält
Tatsächliches Resultat	Es wurde ein korrekter XML ATOM Feed zurückgegeben.
Fazit	**Okay**

Überprüfen der benutzten RESTful URL (mit mehreren Parametern)	
Beschreibung	Die RESTful URL wird aus den verschiedenen Suchparametern zusammengestellt. Es soll überprüft werden ob das zusammenbauen der verschiedenen Parameter korrekt funktioniert. Dazu wird der Service mit den Parametern aufgerufen und anschliessend die vollständig zusammengebaute URL per Debugging ausgelesen. Diese wird dann in den Browser kopiert und die Antwort des Webservers analysiert.
Erwartetes Resultat	Der Webservice antwortet mit einem korrekten XML ATOM Feed der die Resultate der Suchanfrage enthält
Tatsächliches Resultat	Es wurde ein korrekter XML ATOM Feed zurückgegeben.
Fazit	**Okay**

Überprüfen der Transformation des XML ATOM Feeds	
Beschreibung	Der ATOM Feed wird mit einem Transform-File in eine interne Struktur transformiert, die anschliessend einfach ausgelesen werden kann.
Erwartetes Resultat	Es treten keine Exceptions bei der Transformation des Feeds auf. Die zurückgegebene Struktur stimmt mit der gewünschten internen XML Struktur überein und es wurden alle Daten korrekt und unverfälscht übernommen.
Tatsächliches Resultat	Der XML ATOM Feed wurde korrekt in das interne Format transformiert
Fazit	**Okay**

Überprüfen des Parsen der internen XML Struktur	
Beschreibung	Das transformierte XML wird geparst und in eine geeignete Datenhaltungsklasse abgefüllt. Es soll überprüft werden ob dieses Parsen korrekt funktioniert und die Daten anschliessend alle in der neuen Datenhaltungsklasse (SearchResponseDTO) enthalten sind.
Erwartetes Resultat	Das parsen führt am Schluss zu einem korrekt abgefüllten SearchResponseDTO das alle Daten enthält.
Tatsächliches Resultat	Das parsen ist geglückt und das SearchResponseDTO Objekt korrekt abgefüllt.
Fazit	**Okay**

Tests vom 20.5.2011 – TestClient

Systemtests

Aufstarten des TestClients	
Beschreibung	Aufstarten des TestClients
Erwartetes Resultat	Der Client startet auf und der Benutzer findet sich auf dem MainView Screen. Es treten während dem Aufstarten keine Fehler auf.
Tatsächliches Resultat	Der Client startet erfolgreich
Fazit	**Okay**

Überprüfen der angezeigten Services	
Beschreibung	Es werden dem Benutzer die richtigen Services zur Auswahl gestellt, nämlich diese die im entsprechenden Directory enthalten sind. Zur Überprüfung wird einer der Services anschliessend aus dem Directory gelöscht und der Client neu gestartet.
Erwartetes Resultat	Es werden zunächst beide Services angezeigt (Search.ch und Outlook) nach dem der Outlook Service aus dem Directory gelöscht wurde wird dieser aus dem GUI nach einem Neustart ebenfalls verschwinden.
Tatsächliches Resultat	Es werden zunächst beide Services angezeigt, nach dem löschen fällt der gelöschte Service auch aus dem UI weg.
Fazit	**Okay**

Absetzen einer Suchanfrage an den Outlook-Service	
Beschreibung	Es werden die Input-Felder ausgefüllt und die Suchanfrage an den Outlook Service (Dummy) gesendet.
Erwartetes Resultat	Es wird ein einzelner Datensatz mit den Daten von „Ramon Müller“ zurückgeliefert und in der TextBox dargestellt.
Tatsächliches Resultat	Der Datensatz wird korrekt dargestellt
Fazit	**Okay**

Absetzen einer Suchanfrage an den Search.ch-Service	
Beschreibung	Es werden die Input-Felder ausgefüllt und die Suchanfrage an den Search.ch-Service gesendet. Ausserdem wird dann dieselbe Anfrage auf der Webseite von Search.ch durchgeführt und anschliessend verglichen.
Erwartetes Resultat	Die Resultate sollen sich mit den Resultaten die von der Webseite zurückkommen, decken.
Tatsächliches Resultat	Die Resultate decken sich.
Fazit	**Okay**

Tests vom 30.5.2011 – SearchCHProvider (Angepasste Sevitec Version)

Automatische Tests

LoadSearchProvider	
Beschreibung	Es wird versucht den SearchCHProvider zu laden, dazu muss er im entsprechenden Verzeichnis bereitstehen.
Erwartetes Resultat	Das Laden des Providers verläuft ohne Exceptions und der Provider ist anschliessend in der Providerliste des Loaders hinterlegt
Tatsächliches Resultat	Das Laden verlief ohne Probleme und der Provider steht im Loader zur Verfügung.
Fazit	**Okay**

InitializeSearchCHProvider	
Beschreibung	Es wird die Initialize-Methode auf dem Provider aufgerufen.
Erwartetes Resultat	Es soll keine Exception geworfen werden und der Provider soll korrekt Initialisiert werden.
Tatsächliches Resultat	Der Provider wurde initialisiert und es wurde keine Exception geworfen
Fazit	**Okay**

GetProviderName	
Beschreibung	Es wird versucht den Namen des Providers auszulesen.
Erwartetes Resultat	Der Aufruf soll den vordefinierten Namen zurückgeben.
Tatsächliches Resultat	Der Aufruf gab den vordefinierten Namen zurück.
Fazit	**Okay**

CheckPreferencesCount	
Beschreibung	Es wird überprüft ob die korrekte Anzahl Preferences initialisiert wurde
Erwartetes Resultat	Es sollten genau 2 Preferences initialisiert worden sein.
Tatsächliches Resultat	Es wurden genau 2 Preferences initialisiert
Fazit	**Okay**

CheckPreferencesSave_noException	
Beschreibung	Es wird überprüft ob beim Speichern der Preferences eine Exception geworfen wird.
Erwartetes Resultat	Es sollte beim Versuch die Preferences zu speichern keine Exception geworfen werden.
Tatsächliches Resultat	Es wurde keine Exception geworfen
Fazit	**Okay**

Search_on_provider	
Beschreibung	Es wird eine Suche auf dem Provider abgesetzt und das Resultat überprüft
Erwartetes Resultat	Das Resultat soll mit den erwarteten Werten übereinstimmen
Tatsächliches Resultat	Das Resultat hat mit den erwarteten Werten übereingestimmt.
Fazit	**Okay**

Search_on_provider_with_no_matches	
Beschreibung	Es soll eine Suche auf dem Provider abgesetzt werden, wobei die gegebenen Suchparameter keine Resultate liefern sollen.
Erwartetes Resultat	Es soll vom Provider eine leere Liste mit IContact's zurückgegeben werden.
Tatsächliches Resultat	Es wurde eine leere Liste zurückgegeben.
Fazit	**Okay**

Systemtests

Einfügen in TestClient von Sevitec	
Beschreibung	Der SearchCHProvider soll in den TestClient von Sevitec eingebaut werden, dabei wird das Assembly im entsprechenden Verzeichnis hinterlegt und sollte dann automatisch vom TestClient erkannt und importiert werden.
Erwartetes Resultat	Der TestClient von Sevitec importiert den SearchCHProvider erfolgreich und bietet dem Benutzer im GUI zugriff darauf.
Tatsächliches Resultat	Der TestClient hat den SearchCHProvider erfolgreich importiert und im GUI das Dropdown um die Funktionalität der SearchCHProviders erweitert.
Fazit	**Okay**

Suchanfrage an SearchCHProvider stellen	
Beschreibung	Es wird aus dem TestClient von Sevitec eine Suchanfrage an den SearchCHProvider gestellt.
Erwartetes Resultat	Der TestClient soll die korrekten Treffer gemäss den Suchparametern anzeigen.
Tatsächliches Resultat	Es wurden die korrekten Treffer aufgelistet
Fazit	**Okay**

Preferences speichern	
Beschreibung	Es sollen die Preferences verändert und anschliessend gespeichert werden.
Erwartetes Resultat	Beim erneuten Anzeigen der Preferences sollen die neuen Werte zu sehen sein.
Tatsächliches Resultat	Die neuen Preferences wurden übernommen
Fazit	**Okay**

Laden des SearchCHProviders ohne Internet Verbindung	
Beschreibung	Es wird versucht den SearchCHProvider ohne Internet Verbindung zu laden.
Erwartetes Resultat	Es sollte eine AdressProviderException geworfen werden die dem Benutzer erklärt dass dieser Provider nicht ohne eine Internet Verbindung benutzt werden kann.
Tatsächliches Resultat	Der TestClient stürzt ab, keine Meldung wird angezeigt!
Fazit	**Fehlerhaft**
Weiteres Vorgehen	**Eine Limitierung des Test Clients von Sevitec. Exceptions werden nicht sauber abgefangen und entsprechend reagiert.**

Preferences verändern über Neustart des Clients	
Beschreibung	Es sollen die Preferences verändert und anschliessend gespeichert werden. Anschliessend soll ein Neustart durchgeführt werden und überprüft werden ob die Änderungen übernommen wurden.
Erwartetes Resultat	Beim erneuten Anzeigen der Preferences nach dem Neustart sollen die neuen Werte zu sehen sein.
Tatsächliches Resultat	Die Preferences wurden zurückgesetzt
Fazit	**Fehlerhaft**
Weiteres Vorgehen	**Man müsste die Properties der einzelnen Provider entweder manuell in das globale Test Client Property übernehmen, oder aber separate Properties aus den Providern heraus explizit als File erstellen. Nach Rücksprache mit Simon ist das vorerst aber nicht relevant.**

Benutzen des SearchCHProviders ohne transform.xsl file	
Beschreibung	Es soll eine Suchanfrage an den SearchCHProvider abgesetzt werden. Das tranforms.xsl file soll jedoch zuvor verschoben / gelöscht werden.
Erwartetes Resultat	Der Client sollte dem Benutzer eine Meldung geben und ihn darauf hinweisen dass die Datei nicht gefunden wurde
Tatsächliches Resultat	Es wird keine Meldung ausgegeben und der Aufruf gibt keine Resultate zurück
Fazit	**Fehlerhaft**
Weiteres Vorgehen	**Dies ist ebenfalls auf das Problem der nicht sauber abgefangenen Exceptions zurückzuführen. Die Exception wird zwar geworfen, doch im Test Client nirgends behandelt.**

Da die Exceptions nicht sauber abgefangen werden habe ich mir die Arbeit erspart auf alle Exceptions zu prüfen, da diese vermutlich alle Fehlschlagen werden. Ich habe dieselben Test später in meinem eigenen TestClient durchgeführt um das werfen der korrekten Exceptions zu prüfen.

Systemtests vom 5.6.2011 – Finaler TestClient

Systemtests

Aufstarten des Test Clients ohne Internet Verbindung	
Beschreibung	Der Test Client soll ohne Internet Verbindung aufgestartet werden.
Erwartetes Resultat	Es sollen nur die lokalen AdressSeachServices angezeigt werden. Jegliche Services die eine Internet Verbindung benötigen sollen gar nicht erst geladen werden.
Tatsächliches Resultat	Es wurden nur diejenigen Services angezeigt die keine Internet Verbindung benötigen
Fazit	**Okay**

Aufstarten des Test Clients mit Internet Verbindung	
Beschreibung	Der Test Client soll aufgestartet werden während eine Internet Verbindung besteht.
Erwartetes Resultat	Es sollen nun alle AdressSearchServices angezeigt werden, also jene die Internet Zugriff benötigen sowie diejenigen die keinen Internet Zugriff benötigen.
Tatsächliches Resultat	Es werden nun alle Services angezeigt.
Fazit	**Okay**

Die Internet Verbindung soll im laufenden Betrieb gekappt werden	
Beschreibung	Der Client soll mit Internet Verbindung aufgestartet und in Betrieb genommen werden. Anschliessend wird die Internet Verbindung getrennt.
Erwartetes Resultat	Der Test Client soll die Veränderung sofort bemerken und die AdressSearchServices herausfiltern die eine Internet Verbindung benötigen. Die zurzeit angezeigten Daten sollen dabei nicht verändert werden.
Tatsächliches Resultat	Die Services wurden herausgefiltert, die angezeigten Daten blieben bestehen.
Fazit	**Okay**

Test Client offline starten und Verbindung im laufenden Betrieb wieder herstellen	
Beschreibung	Der Test Client soll ohne eine Internet Verbindung aufgestartet werden und anschliessend im laufenden Betrieb wieder hergestellt werden.
Erwartetes Resultat	Der Test Client soll reagieren sobald die Verbindung wieder vollständig hergestellt ist und die Internet Verbindung wieder besteht. Er soll dann die Filterung der Services aufheben und wieder alle Services zur Auswahl stellen.
Tatsächliches Resultat	Die Services wurden wieder angezeigt sobald die Verbindung aufgebaut war.
Fazit	**Okay**

Paging der Suchresultate	
Beschreibung	Bei grossen Resultatmengen soll ein sogenanntes Paging möglich sein. Dadurch werden stets eine gewisse Anzahl (für diesen Test 3, sonst konfigurierbar) Resultate abgefragt und durch einen Klick auf „Weitere Resultate“ jeweils die nächsten 3 Resultate geladen werden.
Erwartetes Resultat	Wenn mehr Resultate vorhanden sind als momentan angezeigt werden soll ein Button „Weitere Resultate“ eingeblendet werden. Durch einen Klick darauf sollen die nächsten 3 Resultate angezeigt werden. Dieser Button soll verschwinden sobald alle Resultate angezeigt werden.
Tatsächliches Resultat	Das Paging der Suchresultate funktioniert wie vorgesehen
Fazit	**Okay**

Exceptionhandling	
Beschreibung	Der Test Client soll auf seine Robustheit geprüft werden indem möglichst viele Fehler erzeugt werden. Darunter gehört das • verschieben / löschen der Transform-Datei • Simulation: Unerreichbarer webservice • Korruptes / falsches transform.xsl • Verschieben / löschen der Image-Datei
Erwartetes Resultat	Für jeder dieser Fälle soll der Test Client dem Benutzer eine vernünftige Fehlermeldung anzeigen sowie so gut wie möglich das weiterverwenden des Clients sichern (Keine Crashes)
Tatsächliches Resultat	Der Client reagiert auf all diese Fälle mit einer Fehler Nachricht an den Benutzer und stürzt weder ab noch bleibt er hängen.
Fazit	**Okay**

Persönlicher Erfahrungsbericht

Erfahrungsbericht von Ramon Müller

Allgemein

Das Arbeiten im C# .NET Umfeld hat mir grossen Spass gemacht. Da ich ursprünglich der Java-Front entsprang gab es für mich einiges Umzudenken und neu zu lernen, damit ich mich in der neuen Welt von .NET zu Recht fand. Gerade was Methodennamen anging hat man meine innere Ablehnung des Gross-schreibens bis in die letzten Codestücke erkennen können. Meistens wurden sie zum Glück im Code-review noch rechtzeitig entdeckt, vereinzelt haben sie es aber bis zur Schlusspräsentation geschafft sich zu verstecken.

Projekt

Das Projekt war für mich sehr interessant, da es einige sehr aktuelle Technologien (MEF, XSL-Transformation) behandelt hat die ich bisher nicht gekannt habe. Ausserdem war die eigentliche Programmieraufgabe relativ überschaubar, sodass ich mich auch als C# Neuling nicht überfordert gefühlt habe. Es hat mir die Möglichkeit geboten einen guten Einblick in die Welt von C# und die Arbeitsweise im .NET Umfeld zu erhalten. Der Einzige Wehrmutstropfen war, dass mein Verschiebungsgesuch fürs Militär nicht genehmigt wurde. Dadurch wurde ich für drei Wochen dem Projekt entzogen, was etwas ungünstig war. Zum Glück konnte ich mit Herrn Huser den Abgabetermin etwas verschieben, was mir zum Schluss nochmal etwas Zeit verschaffte.

Zusammenarbeit mit Sevitec

Sevitec war eine sehr angenehme Partnerfirma. Die regelmässigen Sitzungen die ich mit Simon Baer hatte waren sehr informativ und es wurde jedes Mal sehr hilfreicher Input zu Problemlösungen erbracht.

Fazit

Die Studienarbeit war für mich ein Erfolg. Ich habe wieder viel dazu gelernt, vor allem was das Arbeiten im C# / .NET Umfeld und das Führen eines Projekts angeht. In Zukunft sollte ich mich während des Projekts etwas fleissiger um die Dokumentation kümmern, damit ich gegen Ende des Projekts nicht allzu viel Aufarbeiten muss. Auch wenn der militärisch bedingte Unterbruch etwas unglücklich war denke ich dass ich den Auftrag zur Zufriedenheit von Sevitec erledigen konnte und hoffe dass die Arbeit vielleicht sogar produktiv zur Anwendung kommt, oder aber zumindest als gute Grundlage für weitere Analysen / Neuentwicklungen dienen wird.