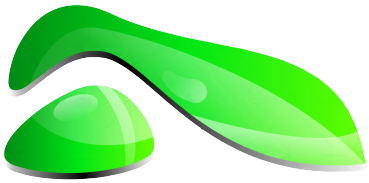
HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

INSTITUT
FÜR
SOFTWARE

User Tracking Framework für WPF Applikationen



Studienarbeit

Studienarbeit HS2012

Studenten: Roger Landolt, Marco Tanner

Betreuer: Michael Gfeller, Silvan Gehrig

Betreuender Dozent: Hans Rudin

Änderungsnachweis

Version	Änderung	Autor	Datum
1.0	Entwurf	Marco Tanner	09.12.2012
1.1	Einfügen der Subdokumente	Marco Tanner	10.12.2012
1.2	Management Summary nachgeführt	Roger Landolt	17.12.2012

Inhaltsverzeichnis

1. Erklärung der Eigenständigkeit

Wir erklären hiermit,

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

Ort, Datum: Rapperswil, den 19. Dezember 2012

Name, Unterschrift

Roger Landolt

Name, Unterschrift

Marco Tanner

2. Abstract

Die vorliegende Studienarbeit hat zum Ziel, ein nützliches Analyse-Werkzeug für Hersteller von WPF-Applikationen zu realisieren.

Es wurde ein Basis-Framework geschaffen, das den Entwicklern Feedback zu Benutzerverhalten und Informationen zu, während dem Betrieb Ihrer Applikation, auftretenden Fehlern liefert. Diese Informationen waren bisher schwierig und kaum automatisiert zu erhalten.

Das Framework wurde bewusst so implementiert, dass es ohne grossen Konfigurationsaufwand in ein beliebiges bestehendes Projekt integriert werden kann, das auf die WPF-Technologie setzt. Bei Bedarf können, mithilfe einer Deklaration via XML-Datei, detaillierte Einstellungsmöglichkeiten genutzt werden um spezifische Auswertungen zu ermöglichen. So können ganz bewusst einzelne Programmereignisse analysiert werden, oder auch alle einer bestimmten Kategorie, wie z.B. aller Klicks auf Schaltflächen.

Die während der Auswertung zusammengetragenen Daten werden über eine verschlüsselte Verbindung auf einen zentralen Server übertragen und auf diesem zur späteren Analyse gespeichert.

Zur Wahrung des Datenschutzes werden dabei keine Daten über den Benutzer selbst erfasst. Um das Nutzungsverhalten dennoch gezielt analysieren zu können, werden gewisse Informationen über den PC sowie dessen Sprach- und Tastatureinstellungen gesammelt.
Des Weiteren wurde entschieden den Softwareherstellern einen eigenen Server zu ermöglichen. Dadurch wird sichergestellt, dass die Firmen die Nutzungsdaten nicht an Drittanbieter weitergeben müssen.

3. Management Summary

3.1. Ausgangslage

Feedback über eine Applikation, welche beim Endkunden im Einsatz ist, ist nur schwierig zu erhalten. Umfragen und Bewertungssysteme sind nicht nicht nur anfällig auf Missbrauch und absichtliche Falschantworten, sondern auch ein Ärgernis vieler Benutzer. Daher ist es eine gute Idee, Feedback möglichst automatisiert zu sammeln. Dazu gehören nicht zuletzt auch Informationen zu Fehlern, die im Betrieb der Applikation auftreten. UTF4WPF verringert diese Schwierigkeiten und bietet eine einfache und automatisierte Art und Weise um an Daten für Optimierungen und Verbesserungen zu kommen.

3.2. Technologie

Da UTF4WPF auf Applikationen mit WPF-GUI ausgelegt ist, die also auf Microsoft .NET basieren, ist es nur logisch, selber ebenfalls auf .NET-Technologie zu setzen. Für die zentrale Datensammlung wird ein SQL Server 2008 (oder höher) verwendet, welcher beinahe auf beliebige Datenmengen skaliert werden kann und, mit der Business Intelligence Komponente, bereits über ein gutes Werkzeug für Auswertungen mitbringt.

3.3. Ergebnis

Vorauszuschicken ist dass sämtliche Anforderungen, welche im Requirementsdokument genannt wurden, abgedeckt werden konnten. Bei der Umsetzung wurde stark darauf geachtet, dass bei der Integration des Frameworks möglichst wenig am Produkt, welches überwacht werden soll, verändert werden muss. Damit können die Abhängigkeiten vom Framework tief gehalten werden, was in kurzer Integrations- und auch Segregationszeit resultiert.
In grösseren Software-Teams sind die GUI-Designer häufig von den Programmierern getrennt. Deshalb wurde darauf geachtet, dass die Konfiguration, welche Events aufgezeichnet werden sollen, auch von einem GUI-Designer ohne Programmierkenntnisse durchgeführt werden kann.
Ebenso wird dem Endkunden, welcher das getrackte Produkt einsetzt, keine Installation von Zusatzkomponenten aufgebürdet und seine Privatsphäre wird gewahrt.

3.4. Ausblick

Die Studienarbeit von Roger Landolt und Marco Tanner beschäftigte sich hauptsächlich mit der Sammlung und Übermittlung der Daten an einen zentralen Server. Die Auswertung der

gesammelten Daten wurde vorhin als Folgearbeit definiert und ist somit noch umzusetzen. Durch die Ablage in einer MSSQL Datenbank ist der Zugriff auf diese Daten jedoch aus einer beliebigen Applikation möglich, dadurch kann eine Auswertung grundsätzlich auch losgelöst von den bereits entwickelten Komponenten durchgeführt werden.
Neben der Entwicklung einer Auswertungskomponente gibt es auch im UTF4WPF Tracker selbst noch einzelne Punkte mit Verbesserungspotential:

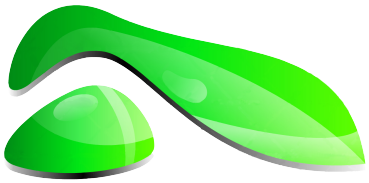
- Frameworknutzern die Möglichkeit bieten eine Tracking-Session ohne Synchronizer zu starten und/oder die Synchronisation zu einem beliebigen Zeitpunkt zu starten/unterbrechen. Dies ist insbesondere bei mobilen Anwendungen wünschenswert. Diese verfügen oft über eine Internetverbindung, welche aber nicht durch Komponenten genutzt werden soll, die die Kommunikation auch zu einem späteren Zeitpunkt durchführen könnten.
- In der *App.Config* könnten weitere Einstellungen entfernt bzw. in den Code verlegt werden. Dies sind generelle Einstellungen welche der Benutzer nicht einrichten, sondern aktuell lediglich kopieren muss.
- Optimierung in der Datenübermittlung aufgrund gesammelter Erfahrungen: Bei einem Einsatz in einer echten Anwendung können Erfahrungswerte zum Auftreten der Daten (Sessions, Occurrences etc) gesammelt werden. Allenfalls lässt sich daraus eine Optimierung für die Datenübertragung ermitteln und so die Netzwerkbelastung weiter senken.

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

INSTITUT
FÜR
SOFTWARE

User Tracking Framework für WPF Applikationen



Analyse

Studienarbeit HS2012

Studenten: Roger Landolt, Marco Tanner

Betreuer: Michael Gfeller, Silvan Gehrig

Betreuender Dozent: Hans Rudin

4. Konkurrenzanalyse

Damit wir uns ein Bild über allfällige, bereits realisierte Lösungen zur Aufgabenstellung machen konnten, wurden verschiedene Tools betrachtet und auf Ihre Funktionen untersucht. Die Programme wurden teilweise via Web-Suche, teilweise auf einen Hinweis durch unsere Betreuer gefunden. Anhand der Resultate der Studie soll eine Feature-Liste für das geplante UTF4WPF erstellt werden.

4.1. Snoop

Abbildung 4.1.: Snoop Logo

Übersicht

Name:	SNOOP - The WPF Spy Utility
Lizenz:	Microsoft Public License (Ms-PL)
Website:	http://snoopwpf.codeplex.com
Version:	2.7.1
Typ:	Standalone Applikation

Snoop ist eine open-source Applikation die “spying” in WPF-Applikationen ermöglicht. Es ist in C# geschrieben und wird als installierbares MSI-Paket ausgeliefert. Es kann den Visual Tree einer WPF Applikation auflisten und Eigenschaften von Controls zur Laufzeit abändern. Zudem wird eine Vergrösserungs- und Zoomfunktion mitgeliefert die ein Vergrössern eines Fensterausschnitts ermöglicht. Es gibt keinerlei Persistenz, gelesene Informationen können nicht abgespeichert werden. Beim Start muss Snoop auf eine Applikation angesetzt werden, danach können auch GUI-Events wie Klicks oder Tastendrücke live mitverfolgt werden.

4.2. StatWin

Abbildung 4.2.: StatWin Logo

Übersicht

Name:	StatWin - Computer Monitoring Software
Lizenz:	Kostet zwischen $59 und $79 pro Client, $399 für den Server
Website:	http://www.statwin.com
Version:	8.7
Typ:	Client-Server Applikation Applikation

StatWin setzt eher auf Seite der Benutzer-Überwachung und nicht der Beobachtung an. So sind einige Stichworte der Website auch Information Security und Time Tracking Software.
Neben der Überwachung von Programminstallationen, Internet und Druckern erlaubt StatWin aber auch das Abhören von Tastatur und Maus. So können sogar Screenshots bei jedem Mausklick erstellt und an einen zentralen Server (Version Enterprise) übermittelt werden.
Für die Beobachtung innerhalb einer Applikation ist es damit aber nur bedingt geeignet und es lässt sich nicht in ein eigenes Produkt integrieren sondern muss separat betrieben werden.

4.3. DeskMetrics

Abbildung 4.3.: DeskMetrics Logo

Übersicht

Name:	DeskMetrics - Powerful analytics for desktop applications.
Lizenz:	Momentan noch Closed Beta
Website:	http://deskmetrics.com/
Version:	2 (Closed Beta)
Typ:	Unklar

DeskMetrics befindet sich momentan in einer Closed Beta-Phase und verrät auf seiner Homepage nur sehr wenig. Aus den Blog Einträge ist aber ersichtlich, dass es gedacht ist um die Verteilung und Nutzung einer Applikation weltweit zu überwachen.
Ob dies einzelne Button-Klicks oder eher die Anzahl Programmnutzungen betrifft ist nicht eindeutig ersichtlich. Zumindest der Blog-Eintrag *Why You Should Use An Application Analytics (26. Mai 2012)* deutet auf eine reine Analyse der Programmstarts und Installlationen hin. Wir haben uns für die Closed Beta eingeschrieben und warten ab ob wir eine Einladung erhalten. Da die uns bekannten Informationen nicht für einen Konkurrenzvergleich ausreichen, wird DeskMetrics in der Übersichtstabelle nicht aufgeführt.

4.4. JAMon

Abbildung 4.4.: JAMon Logo

Übersicht

Name:	JAMon - Java Application Monitor
Lizenz:	BSD Lizenz
Website:	http://jamonapi.sourceforge.net/
Version:	2.73
Typ:	API / Library

Die JAMon API ist nach eigenen Angaben eine freie, simple, hochperformante und Threadsichere Java API die Entwicklern erlaubt, die Leistung und Skalierbarkeit ihrer Applikationen zu überwachen. Sie ist darauf ausgerichtet, Java-Applikationen zu überwachen und insbesondere Flaschenhälse in der Leistung zu finden. Es wird direkt im Code angegeben, welche Teile der Applikation überwacht sollen werden. Theoretisch könnten so auch die Anzahl “Hits” für GUI-Elemente über eine Zeit gemessen werden. Dies dürfte sich jedoch als ziemlich aufwändig erweisen, da die API vorallem auf die typischen Bottlenecks wie zum Beispiel I/O ausgerichtet ist. Auch lassen sich keine Auswertungen über längere Zeit durchführen, die gesammelten Daten werden in einer in-memory Datenbank nachgeführt.

4.5. ClickStream

Übersicht

Name:	ClickStream
Lizenz:	Unklar
Website:	http://www.opensymphony.com/clickstream/
Version:	Unklar
Typ:	API / Library

ClickStream ist eine Library die detailiertes User-Tracking erlaubt um die “Wege” von Benutzern auf Webseiten zu erfassen. Es kommt der Zielsetzung unseres Projektes ziemliche nahe, kann aber keine GUI-Applikationen tracken.
ClickStream wird seit 2011 nicht mehr weitergeführt und wird deshalb in der Übersichtstabelle wie auch DeskMetrics nicht aufgeführt.

4.6. Trackerbird

Abbildung 4.5.: TrackerBird Logo

Übersicht

Name:	Trackerbird
Lizenz:	Kommerziell
Website:	http://www.trackerbird.com
Version:	Unklar
Typ:	API / Library

Trackerbird ist eine Library die sehr viele Informationen über .NET-Applikationen sammeln kann. Es wird ähnlich wie unser geplantes Framwork in eine bestehende Applikation eingebaut und sendet dann die Tracking-Daten an einen zentralen Server. Im Gegensatz zu unserem Projekt jedoch, kann der Server nicht selber bestimmt werden sondern wird durch Trackerbird verwaltet. Deshalb ist das Bezahlmodell für den Dienst auch auf monatlicher Basis eingerichtet. Die günstigste Version, die Feature-Tracking bietet (nur 12 einzelne Features) schlägt mit 49$ pro Monat zu Buche. Für 25 Features sind es dann schon 349$ pro Monat. Für eine unlimitierte Anzahl an getrackten Features muss man direkt mit dem Hersteller in Kontakt treten. (Für die komplette Preisliste siehe auch die Trackerbird Preisliste) Neben Feature- und Exceptiontracking bietet Trackerbird auch Download- Installation- und Deinstallationsstatistiken und weitere Funktionen wie z.B. geographische Verteilung der Kundschaft. Es wird mit einer kurzen Einbindungszeit von lediglich 30 Minuten geworben. Support für C/C++, Java und Python sind in Planung.

4.7. Übersichtstabelle

In der folgenden Tabelle sind die einzelnen, erkannten Features in einem Vergleich mit dem geplanten Framework UTF4PWF aufgelistet.

Tabelle 4.1.: Featuretabelle

	Snoop	StatWin	JAMon	Trackerbird	**UTF4WPF**
Standalone	Ja	Ja	Nein	Nein	Nein
Zielplattform	.NET	Windows	Java	.NET	.NET
Open Source	Ja	Nein	Ja	Nein	Nein
Persistenz	Nein	Ja	Nein	Limitiert	Ja
GUI Event Tracing	Ja	Nein	Ja	Ja	Ja
Mouse- & Keylogger	Nein	Ja	Ja	Nein	Ja
Informationen für End-User	Ja	Nein	Nein	Nein	Nein
Anpassbare Überwachung	Nein	Nein	Ja	Ja	Ja
Integration im Sourcecode	Nein	Nein	Ja	Ja	Ja
Tracing von mehr als 1 Applikation	Nein	Ja	Ja	Ja	Ja
Tracing von mehreren Benutzern	Nein	Ja	Nein	Ja	Ja

4.8. Schlussfolgerung

Die Studie hat ergeben, dass keines der bestehenden Produkte die Anforderungen der Aufgabenstellung vollständig abdeckt, dafür hat sie aber nützliche Anhaltspunkte geliefert, was an Zusatzfunktionen in UTF4WPF einfliessen könnte. Einige Funktionen der oben genannten Produkte erscheinen daher als Requirements im nächsten Kapitel.

5. Studie zur Datenhaltung

Für die Art und Weise, wie die gesammelten Daten erfasst werden gibt es verschiedene Möglichkeiten bezüglich Detailgrad und Umfang. Im Folgenden sind verschiedene denkbare Alternativen aufgelistet, am Ende des Kapitels wird aus den vorgestellten Varianten ein konkreter Vorschlag vorgestellt.

5.1. Zeit

Es gibt verschiedene denkbare Möglichkeiten, wie die Zeit der erfassten Events gespeichert werden kann:

- **Lokal/Absolut:** Die Events werden mit dem aktuellen Zeitstempel des Clients, evtl. mit zusätzlichem Millisekunden-Offset für bessere Genauigkeit abgespeichert. Damit Vergleiche gut möglich sind, wird eine einheitliche Zeitzone unabhängig vom Standort des Clients verwendet.
- **Relativ:** Die Events werden jeweils mit der Zeit relativ zum Applikationsstart gespeichert wobei jeder Start der Applikation die Zeit 0 darstellt.
- **Logisch:** Es wird keine konkrete Zeit gespeichert, die Events werden chronologisch nummeriert.

Die Option, die Events mit der aktuellen Serverzeit zu versehen, fällt weg, da die Erfassung der Events auch ohne Serververbindung machbar sein muss.

5.2. De-personalisierung

Die gesammelten Daten dürfen keine Rückschlüsse auf den Namen oder sonstige Personalien des Benutzers erlauben. Für die Analyse kann es aber dennoch hilfreich sein, zumindest eine gewisse Wiedererkennung des Benutzers zu ermöglichen. Folgende Varianten wären in diesem Zusammenhang denkbar:

- **Maschine:** Jedes Client-Gerät erhält eine eindeutige ID, die jeweils mit den Log-Daten an den Server geschickt wird. Die ID ist unabhängig vom Computernamen oder IP-Adresse, so dass keine Rückverfolgung möglich ist.
- **Benutzer:** Wie beim Maschinen-Prinzip, jedoch wird für jeden Benutzer pro PC eine eigene ID generiert. Auch hier wäre die ID unabhängig vom effektiven Benutzernamen.
- **Kein:** Es wird keine Wiedererkennung von Benutzern oder Geräten versucht. Die einzelnen Datensätze (jeweils eine Benutzer-Session) werden komplett unabhängig voneinander gespeichert.

Dass die Erkennung der getrackten Applikation und deren Version jederzeit möglich sein muss, versteht sich von selbst.

5.3. Anonyme Benutzer und/oder Gerätedaten

Zu jeder Benutzersession wird ein Set von nicht Rückverfolgbaren, Benutzer- und Gerätedaten erfasst. Dazu gehören beispielsweise Informationen wie die Grösse des Arbeitsspeichers, das verwendete Betriebssystem sowie das gewählte Tastaturlayout.

5.4. Konkreter Vorschlag

Aus den obigen Optionen wird vom Projektteam folgender konkreter Vorschlag zur Datenhaltung gemacht:
Die Events werden mit einem Zeitstempel in GMT-Zeit +Offset in Millisekunden aufgezeichnet. Die Genaugkeit auf Tausendstel-Sekunde ist notwendig damit kurz aufeinanderfolgende Events (z.B. MouseDown und MouseUp) nicht als “gleichzeitig” erkannt werden. Die Startzeit pro Session wird ebenfalls erfasst, somit kann bei Bedarf die Option mit relativen Zeitstempeln implizit auch realisiert werden.
Für die zusätzlichen Informationen wird vorgeschlagen, einige wenige Parameter mitaufzuzeichnen:

- Aktuelles Betriebsssystem
- Systemsprache und Tastaturlayout (für I18n-Optimierungsanalysen)
- CPU Typ und Grösse des Arbeitsspeichers
- Informationen zu angeschlossenen “Pointing Devices” (Mäuse, Touchscreens, etc.)

5.4.1. Lokale Datenhaltung

Neben der zentralen Speicherung der Daten auf dem Server gibt es diverse Gründe weshalb die Daten zuerst lokal persistent abgelegt werden:

- Bei fehlender Server-/Internet-Verbindung soll trotzdem eine Sammlung von Benutzerdaten möglich sein.
- Der Overhead durch Aufbau einer Verbindung zwischen Client und Server soll gering gehalten werden.

Maximale lokale Aufbewahrungsdauer

Alte Aufzeichnungen können einen negativen Einfluss auf die Auswertungen haben bzw. diese nachträglich verfälschen. Deshalb werden alle Sessions, die älter als 7 Tage sind und noch nicht an den Server übertragen wurden, gelöscht.

Vorschlag

Um die Performance durch ein ständiges Prüfen dieser Meldungen zu verhindern ist die Limitierung einmalig beim Programmstart und allenfalls Programmende zu prüfen und entsprechend zu behandeln.

5.4.2. Grösse einer Datenlieferung

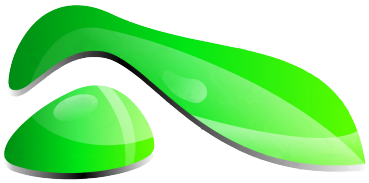
Wie bereits geschrieben soll der Overhead durch den Verbindungsaufbau zwischen Client und Server möglichst klein gehalten werden. Auch die Ressourcen sollen effizient genutzt werden, eine ständige Verbindung zum Server soll vermieden werden.
Umd dieses Ziel zu erreichen werden jeweils 100 GUIOccurrences als ein Paket gesendet.

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

INSTITUT
FÜR
SOFTWARE

User Tracking Framework für WPF Applikationen



Requirements

Studienarbeit HS2012

Studenten: Roger Landolt, Marco Tanner

Betreuer: Michael Gfeller, Silvan Gehrig

Betreuender Dozent: Hans Rudin

6. Anforderungsspezifikation

Aus der vorangegangenen Konkurrenzanalyse sowie der Aufgabenstellung wurde folgendes Pflichtenheft erarbeitet:

6.1. Zielbestimmung

Aus der Aufgabenstellung:

> User-Feedback über die Art und Häufigkeit der Nutzung von Programm-Features oder Navigation ist schwierig zu erhalten. Dieses Feedback wird allerdings auf Seite der Hersteller dringend benötigt, um die Software weiter zu entwickeln und das Benutzererlebnis zu verbessern. Hier setzt diese Studienarbeit an.
>
> Die Studienarbeit *User Tracking Framework für WPF-Applikationen* wurde initiiert um ein Basis-Framework zu schaffen, welches die Messung der Art und Häufigkeit der Nutzung von Programm-Features erlaubt. So werden mit Erlaubnis des Users 'unpersonalisierte' Daten gesammelt und diese auf einen zentralen Server in der Cloud übertragen. Bei fehlender Verbindung zum Server können die Daten lokal zwischengespeichert und zu einem späteren Zeitpunkt in die Cloud übertragen werden. Auf Seite des Herstellers lassen sich somit Muster in der Benutzung der Anwendung, also beispielsweise Optimierungspotential in der Usability erkennen. Ein weiteres Merkmal des Frameworks stellt die Behandlung (Logging) von Ausnahmen und Programm-Fehlverhalten dar. Im Rahmen des Versendens von Fehlerinformationen soll es möglich sein, eine erneute (separate) Erlaubnis vom Benutzer zum Datenversand zu erhalten.
>
> Das Auswerten der Daten auf Herstellerseite der Daten ist nicht Teil der Arbeit *User Tracking Framework für WPF Applikationen*. Die Studienarbeit konzentriert sich auf das Sammeln und sichere Übertragen der Daten.

6.2. Gliederung in Teilprodukte

Die Software gliedert sich in drei Teilprodukte:

1. Clientframework: Für die Clients wird ein Framework erstellt, welches in beliebige WPF-Anwendungen integriert und für diese konfiguriert werden kann.
2. Serverkomponente: Kommuniziert mit den Clients und speichert Meldungen zentral in einer Datenbank ab.
3. Auswertungskomponente: Dient zur Analyse der gesammelten Daten. Diese Komponente ist **nicht** Teil der Studienarbeit.

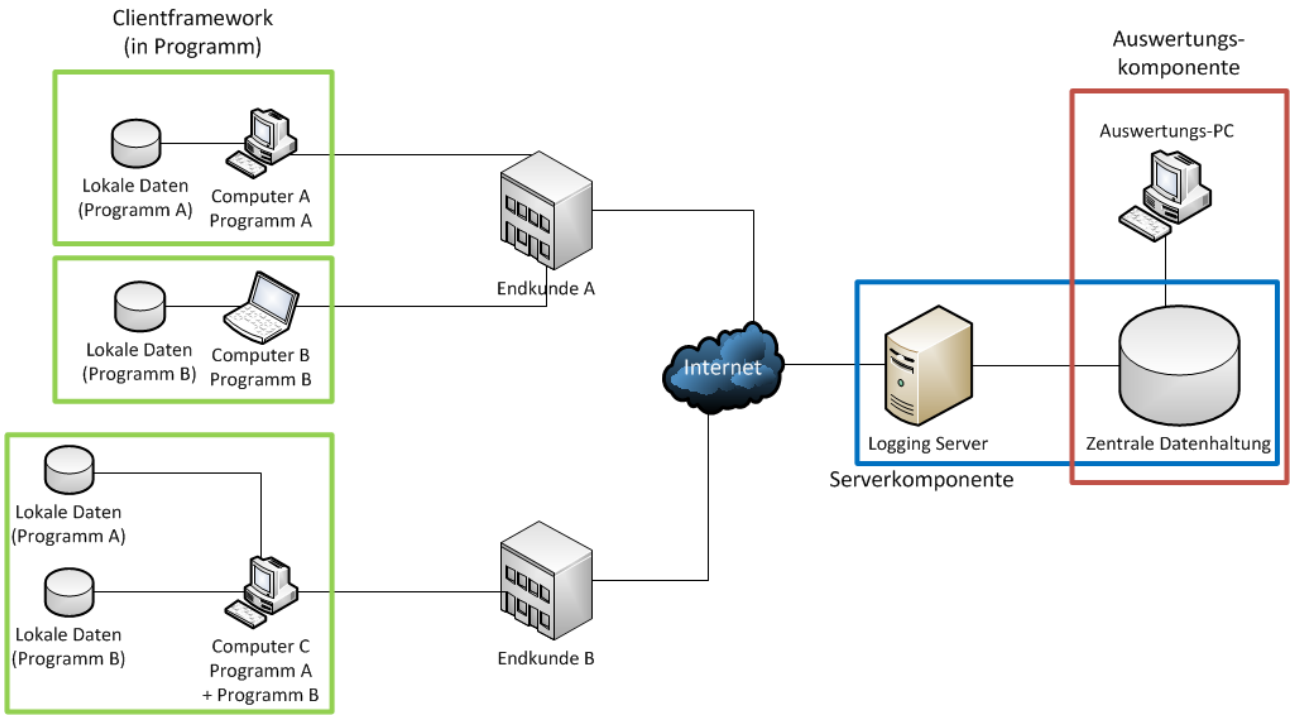


Abbildung 6.1.: Gliederung in Teilprodukte

6.3. Muss-Kriterien

M01	**Sammeln von Events**
Sammeln von Events welche die Frameworknutzer auswählen. Es sollen mehrere Auswahlmöglichkeiten zur Verfügung stehen: M1.1 Alle Events eines Control-Typs M1.2 Bestimmtes Event eines Control-Typs M1.3 Alle Events eines bestimmten Controls M1.4 Ein bestimmtes Event	
M02	**Sammeln von Ausnahmen und Fehlern**
Gemäss Rücksprache mit den Auftraggebern sind damit die unbehandelten Exceptions gemeint. Beim Fehlerfall soll der Benutzer in einem ähnlichen Stil gefragt werden ob er den Fehler dem Hersteller zusenden möchte.	
M03	**Die Datensammlung ist auch bei fehlender Verbindung möglich**
Damit dies möglich ist, wird lokal eine Art Zwischenspeicher benötigt. Resultiert in lokaler Speicherung für spätere Übertragung.	
M04	**Zentrale Sammlung der Daten**
Bei bestehender Netzwerkverbindung sollen die gesammelten Daten regelmässig an eine zentrale Stelle übermittelt werden und dort in einer MSSQL-Datenbank gespeichert werden. Dies bietet die Grundlage für die spätere Auswertung über mehrere Nutzer	
M05	**Gesicherte Handhabung der Daten**
Kritische Daten müssen so gesichert werden, dass diese nicht unberechtigt verändert oder gelesen werden können, für die Verbindung zwischen Client und Server ist dies ein MUSS. Mehr zur Client/Server-Verbindung ist in M06 deklariert.	
M06	**Client/Server-Verbindung auch bei strikten Firewall regeln möglich**
Um dies umzusetzen soll auf eine bewährte Lösung gesetzt werden. In Kombination mit M05 soll dazu eine Verbindung über HTTPS in Betracht gezogen werden. Der HTTPS-Port 443 ist für ausgehende Verbindungen in beinahe allen Netzwerken erlaubt.	
M07	**Integration in bestehendes Produkt**
Zu Präsentations und Testzwecken ist das Framework in ein bestehendes Produkt zu integrieren. Ebenfalls soll ein Integrationshandbuch erstellt werden um Frameworknutzern die Einarbeitung zu erleichtern.	

Tabelle 6.1.: Muss-Kriterien

6.4. Soll-Kriterien

S01	**Server kann mit heterogener Produktelandschaft umgehen**
Die Serverkomponente soll mit mehreren Produkten und Versionen umgehen können. Durch diese Eigenschaft kann die Wiederverwendbarkeit des Servers sichergestellt werden und auch kleine Firmen können sich die breite Verwendung des Frameworks leisten.	
S02	**Framework in Setup integrierbar**
Die Installation des Frameworks soll zusammen mit dem Produkt möglich sein. Beim Endkunden soll somit keine zusätzliche Installation notwendig werden.	

Tabelle 6.2.: Soll-Kriterien

6.5. Kann-Kriterien

K01	**Entwicklermitteilungen möglich**
Allenfalls soll für die Frameworknutzer eine Funktionalität zur Verfügung gestellt werden um beliebige Messages abzulegen. Dies ist ähnlich eines Debug.Log gedacht, allerdings mit der bestehenden Funktionalität des zentralen Servers etc. Damit wird es den Entwicklern möglich den Benutzerweg inkl. allfälliger Parameter besser zu verfolgen.	
K02	**Sammlung von Infos zum Gerät**
Allenfalls soll für die Frameworknutzer eine Funktionalität zur Verfügung gestellt werden um Allgemeine Infos zum PC zu sammeln, als Beispiel: Betriebssystem/-Version, Version der .NET Umgebung, CPU-Typ, Grösse des Arbeitsspeichers, Systemsprache und Tastaturlayout (für allfällige I18N-Optimierungsanalysen) Die Daten sind **nicht** dazu gedacht einen Computer eindeutig zu identifizieren, sondern um Analysen im Bezug auf die Hardware-Konfiguration zu ermöglichen	

Tabelle 6.3.: Kann-Kriterien

6.6. Produkteinsatz

6.6.1. Anwendungsbereiche

Das Framework soll grundsätzlich in allen WPF-Applikationen zur Anwendung kommen können. Seien dies neue Applikationen welche noch in einem Alpha- oder Beta-Stadium sind oder aber auch Anwendungen welche über die Jahre gewachsen sind und nun verbessert werden sollen. Die gesammelten Daten können beispielsweise genutzt werden für:

- Usability Analysen
- GUI-Optimierungen

- Fehleranalyse
- Allenfalls zur Optimierung von Schulungen aufgrund der allgemein festgestellten Benutzerfehler.

6.6.2. Zielgruppen

Als Zielgruppe sind Entwicklungsteams gedacht welche die Benutzerinteraktion optimieren und dazu ein Analysetool verwenden möchten.

6.6.3. Betriebsbedingungen

Auf Seite der Entwickler muss ein Server zur Verfügung stehen, welcher die Events entgegennimmt und speichert. Aufgrund allfälliger Verbindungsabbrüche auf der Clientseite werden die Daten lokal zwischengespeichert, gewisse Offline-Zeiten des Servermoduls können deshalb ignoriert werden.

Auf Seiten des Clients kann die Applikation aufgrund der vollständigen Integration normal verwendet werden und es muss kein zusätzliches Tool installiert werden. Die Anforderungen an Hard- und Software von Server sowie Clients sind weiter unten in der technischen Produkteumgebung zu finden.

6.7. Nichtfunktionale Anforderungen

6.7.1. Leistungsanforderungen

Die lokale Komponente besteht aus den Hauptfunktionen des Event-Abfangens und der Kommunikation mit dem Server. Die Leistungen sind natürlich abhängig vom verwendeten System und der Netzwerkanbindung bzw. auch der Internetanbindung.
Weiteren Einfluss hat die überwachte Applikation selbst. Gewisse Applikationen sind sehr Benutzer-intensiv, andere dagegen eher reine Anzeigetools. Eine genaue Datenrate ist daher schwer zu bestimmen, bewegt sich aber im tieferen Bereich. Werden z.B. rein die Klicks auf Buttons aufgezeichnet ist es nicht realistisch das mehrere Ereignisse pro Sekunde eintreffen werden.
Wichtig ist es aber, dass die Aufzeichnung und speziell die Datenübertragung zum Server in einem separaten Thread abgehandelt wird und nicht den Programmverlauf “einfriert”.

6.7.2. Mengenanforderung

Die Anzahl Einträge ist ebenfalls stark von der Applikation abhängig und natürlich von der Anzahl überwachter Nutzer. Als Grundsatz ist jedoch zu sagen, dass zur aussagekräftigen Analyse möglichst viel Daten gesammelt werden sollen.
Folgendes Szenario wird als Überlegungsgrundlage verwendet:

- Es wird eine Applikation in der Beta-Phase überwacht
- Die Überwachung findet einen Monat lang statt (22 Arbeitstage)
- Durchschnittlich wird drei Stunden pro Tag daran gearbeitet, wobei die Benutzer das Programm am Morgen starten und dann geöffnet lassen.
- Applikation wird von 25 Benutzern benutzt
- Es werden nur die Buttons überwacht, wobei ein Benutzer über die Gesamtzeit gesehen ca. jede Minute ein Button anklickt.

Somit haben wir rund 100'000 aufzuzeichnende Events ($1App \cdot 25Benutzer \cdot 22Tage \cdot 3Stunden \cdot 60Klicks = 99'000$) welche auf dem Server zu speichern sind. Dazu kommt ein Gerüst für die 550 Benutzersessions (25 Benutzer an 22 Tagen) sowie allfällige weitere Daten die dazu zu speichern sind. Die zusätzlichen Daten hängen von der Varianten-Entscheidung ab, welche noch offen ist.

Auf einem MSSQL-Server welcher den entsprechenden Speicher zur Verfügung stellt ist diese Datenmenge gemäss den Spezifikationen[1] kein Problem.

Für die lokale Datenhaltung fallen bei dem gegebenen Szenario maximal, also wenn der User erst am Ende der Testphase eine Netzwerkverbindung hat, 3'960 Datensätze an. Auch dies liegt im Bereich des Handhabbarem.

6.7.3. Qualitätsanforderungen

Die Anforderungsspezifikationen vom WPF User Tracking Framework folgen der ISO/IECC 9126[2] Norm.

- Kommunikation auch in komplexen Firewall-Situationen möglich. Der Serverport soll konfigurierbar sein.
- Optimiertes Versenden von Daten, es soll nicht jedes Event einzeln versendet werden um den Anteil des Overheads (für die Verbindung etc.) am Gesamtvolumen möglichst klein ist. Mindestens Anzahl Events pro Versand konfigurierbar, Ausnahmen auf expliziten Befehl oder beim Schliessen des Programmes.

[1] http://msdn.microsoft.com/en-us/library/ms143432.aspx
[2] http://de.wikipedia.org/wiki/ISO/IEC_9126

Funktionalität

Angemessenheit	Ein Framework zu planen ist immer eine Gratwanderung zwischen Over- und Under-Engineering. Je nachdem in was für einem Produkt das Framework umgesetzt werden soll ist der Rahmen grösser oder kleiner zu halten. Der Implementationsaufwand wird möglichst gering gehalten, ohne den möglichen Nutzungsumfang unnötig einzuschränken. Die Mindestkonfiguration soll in maximal 10 Zeilen möglich sein, pro zusätzliches Fenster welches überwacht werden soll maximal 5 zusätzliche Zeilen.
Richtigkeit	Die gesammelten Daten sollen dem vollständig gewünschten Stand entsprechen. So werden Daten auch bei fehlender Verbindung gesammelt.
Interoperabilität	Die Datensammlung und Verbindung zum Server soll ein geschlossenes Framework darstellen. Um mit verschiedenen Systemen zu arbeiten soll das Framework aber durch diverse Einstellungen angepasst werden können. Des weiteren sollen die gespeicherten Daten grundsätzlich auch von anderen Systemen auswertbar sein, sofern die Berechtigungen erteilt werden.
Ordnungmässigkeit	Das Framework verhält sich so wie in der vorliegenden Spezifikation beschrieben. Grundlage dafür bietet die Aufgabenstellung der Studienarbeit.
Sicherheit	Mithilfe des Frameworks werden Daten über das Benutzerverhalten in einer Applikation gesammelt. Je nach Applikation können diese Informationen schützenswert sein. Die Daten sollen deshalb nur über eine gesicherte Verbindung (HTTPS) an den Server übermittelt werden. Die Ablage auf dem zentralen MSSQL-Server soll ebenfalls geschützt werden und nur mit entsprechendem User/Passwort abfragbar sein. Die lokale Zwischenspeicherung findet im Benutzerverzeichnis statt, ob dabei eine spezielle Verschlüsselung notwendig ist, und wie diese umgesetzt werden kann soll geprüft werden.

Tabelle 6.4.: Funktionalität

Zuverlässigkeit

Reife	Das Framework soll alle offiziellen Events auf WPF Events abfangen können und bei den 5 wichtigsten Event-Typen zusätzliche Informationen mitliefern können. (z.B. Ort des Mausklicks) Welches die wichtigsten Event-Typen sind ist Teil der genaueren Analyse.
Fehlertoleranz	Verbindungsfehler sollen die Datensammlung nicht behindern und entsprechend umgangen werden können. Der Server soll Client-Verbindungen welche nicht korrekt aufgebaut oder geschlossen werden entsprechend behandeln können. Ebenso können nicht korrekt gelieferte Meldungen beim Server abgefangen und zurückgemeldet werden können.

Tabelle 6.5.: Zuverlässigkeit

Benutzbarkeit

Verständlichkeit	Der Code wird mit den XML-Summary-Kommentaren versehen. Bei der Programmierung mit IntelliSense werden so entsprechende Hinweise angezeigt. Mehr dazu siehe auch bei der Erlenbarkeit direkt unten.
Erlernbarkeit	Es wird eine Dokumentation zur Integration in ein Produkt erstellt. Diese dient als Leitfaden und deckt die wichtigsten Aspekte ab.
Wiederherstellbarkeit	Ein Programmabsturz führt zum Verlust sämtlicher sich im Speicher befindlichen Daten. Nicht vom Datenverlust betroffen sollen die lokal gehaltenen Records sowie die bereits auf das zentrale System gespeicherten Werte sein. Übertragungen von Records welche nicht komplett ausgeführt werden können sollen nicht verloren gehen.

Tabelle 6.6.: Benutzbarkeit

Effizienz

Zeitverhalten	Wie bereits unter den Leistungsanforderungen genannt, soll die Aufzeichnung und speziell die Datenübertragung zum Server in einem separaten Thread abgehandelt werden und nicht den Programmverlauf “einfrieren”. Ebenso soll sich auch die Sammlung der Daten verhalten. Das Ziel des Zeitverhaltens in Bezug auf die Datensammlung soll eine Abarbeitung der neuen Meldungen im einstelligen Millisekunden-Bereich sein. Das Ziel des Zeitverhaltens in Bezug auf die Netzwerk-Verbindungsperformance des Frameworks ist nicht so hoch einzuschätzen wie die Performance bei der Datensammlung. Dies liegt daran, dass die Daten nicht sofort auf dem Zielsystem angelangen müssen, sondern über eine längere Zeitdauer zu statistischen Zwecken gesammelt werden. Des weiteren ist im Framework die verschlüsselte Übertragung wichtiger als das reine Zeitverhalten. Es ist zudem anzumerken, dass die Grösse einer Sendung, also die Anzahl zu sendender Records, durch die Frameworknutzer festgelegt werden kann um sich so Ihren Endkunden anpassen zu können.
Verbrauchs-verhalten	Die Clients sollen nicht durch die Datensammlung belastet werden, aufgrund dessen ist auch die lokale Datenablage klein zu halten. Dem Frameworknutzer sollen dazu Funktionalitäten zur Verfügung gestellt werden um die Anzahl Records und/oder die maximale Aufbewahrungsdauer einschränken zu können.

Tabelle 6.7.: Effizienz

Änderbarkeit

Analysierbarkeit	Das Framework wird während der Studienarbeit von Roger Landolt und Marco Tanner entworfen, soll aber später weitergeführt werden. Die Analysierbarkeit ist deshalb ein wichtiges Qualitätskriterium. Dazu wurden einige Kriterien aufgestellt. Wir werden die gesetzten Coding Standards (siehe Übertragbarkeit→ Konformität) einhalten und entsprechend mit ReSharper prüfen. Einen weiteren Punkt bieten die XML-Summaries (Methoden/Klassenkommentare) welche die wichtigen Funktionen beschreiben. Zudem werden die Dokumentationen, insbesondere das Architekturdokument, über unsere Arbeit jeweils auf dem neuesten Stand der Entwicklung gehalten um die Entscheidungen nachvollziehbar zu machen und als Nachschlagewerk zu dienen.
Modifizierbarkeit	Bei der Entwicklung wird zudem darauf geachtet Erweiterungen nicht auszuschliessen, so soll zum Beispiel darauf geachtet werden, dass grundsätzlich auch weitere Meldungstypen möglich wären und auch die Datenablage verändert werden könnte.

Tabelle 6.8.: Änderbarkeit

Übertragbarkeit

Anpassbarkeit	Ein Framework muss per se anpassbar sein um flexibel genutzt werden zu können. Den Frameworknutzern sollen dazu diverse Einstellungsmöglichkeiten geboten werden, um das Framework möglichst optimal an seine Applikation anpassen zu können.
Installierbarkeit	Ziel ist es, dass clientseitig keine zusätzliche Installation von Framework-Komponenten notwendig wird. Auf der Serverseite soll ein Setup zur Verfügung gestellt werden, welches die notwendigen Schritte (Service-Installation, Firewall, Verbindung und Einrichtung Datenbank) übernimmt und soweit wie möglich automatisiert. Anmerkung: Heute (10. Oktober) haben wir festgestellt, dass die Setup-Komponente in Visual Studio 2012 nicht mehr vorhanden ist. Als Alternative wird eine Lite-Version von InstallShield angeboten, welche aber die Verbindungen mit SQL-Servern, ausführen von Skripts und das Einrichten von Firewall-Regeln nicht mehr ermöglicht.
Konformität	Der Code richtet sich nach den Coding Standards von Lance Hunt (`http://www.lance-hunt.net`). Diese sind stark an die Empfehlungen von Microsoft angelehnt, bieten jedoch eine bessere Übersicht. Diese Standards sind in der Industrie, aber auch im Hochschulbereich (z.B. an der ETH Zürich) bekannt und werden entsprechend verwendet. Wir werden die Einhaltung der des Coding Standards in regelmässigen Abständen mit ReSharper prüfen.

Tabelle 6.9.: Übertragbarkeit

6.8. Technische Produktumgebung

6.8.1. Software

Anforderungen an den Server:

- Windows 2003 SP2 oder neuer
- MSSQL-Server 2008 oder neuer
- .NET Framework 4.0 oder neuer

Anforderungen an den Client:

- .NET Framework 4.0 oder neuer (Bereits für die WPF-Applikation selbst notwendig)

6.8.2. Hardware

Für die Kommunikation benötigen sowohl die Clients als auch der Server eine Internetanbindung, handelt es sich um eine interne Applikation reicht eine normale

Netwerkverbindung zwischen den beiden Komponenten aus.
Die restlichen Anforderungen an den Client beruhen auf den Anforderungen der Applikation in welche das Framework integriert wurde.
Die Anforderungen an den Server häng von der der Anzahl zu erwartenden Clients ab. Handelt es sich lediglich um eine kleine Gruppe Beta-Tester (<10 Personen) reicht grundsätzlich auch eine normale Workstation mit der gegebenen Software-Konfiguration aus.
Wenn es sich aber um ein Produkt im Einsatz handelt welches bereits mehr als 100 User hat müssen die Anforderungen betreffend Netzwerkanbindung, CPU, Arbeitsspeicher, Festplattenspeicher etc. genauer geplant werden.

6.8.3. Orgware

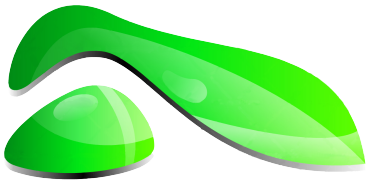
Die Daten werden über HTTPS versendet. Auf Seite des Servers müssen HTTPS-Anfragen deshalb möglich sein, der Client muss ausgehende HTTPS-Verbindungen erlauben. Der verwendete Netzwerkport kann jedoch frei gewählt und mittels Konfigurationsdatei festgelegt werden.

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

INSTITUT
FÜR
SOFTWARE

User Tracking Framework für WPF Applikationen



Software Architektur

Studienarbeit HS2012

Studenten: Roger Landolt, Marco Tanner

Betreuer: Michael Gfeller, Silvan Gehrig

Betreuender Dozent: Hans Rudin

7. Systemaufbau

7.1. Deploymentdiagramm

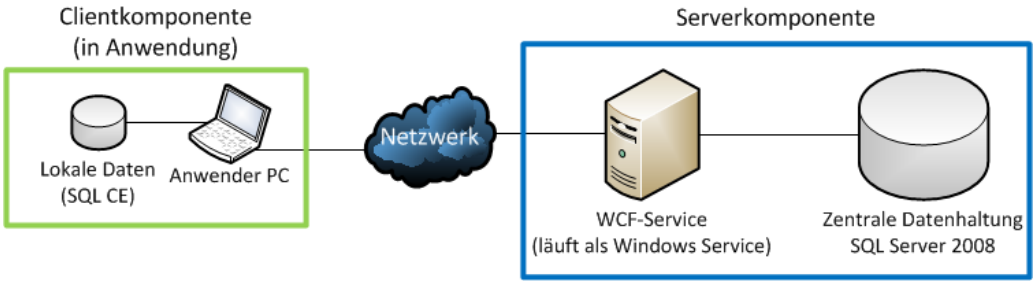


Abbildung 7.1.: Systemaufbau / Deploymentdiagram

Das User Tracking Framework für WPF besteht aus zwei Hauptkomponenten, die Clientkomponente welcher in der Anwendung integriert ist sowie der Serverkomponente welche an zentraler Stelle läuft.
Ein Server kann von einer beliebigen Anzahl Clients genutzt werden. Eine Einschränkung der Anzahl Clients kann nicht klar definiert werden, da dies stark von der Anwendung, dem Nutzungsverhalten und der Tracking-Optionen abhängt. Es ist anzumerken das auf den Clients beliebige Software laufen kann, so auch mehrere verschiedene Programme die UTF4WPF verwenden.

7.2. Layer

Das Projekt teilt sich in die folgenden, logischen Schichten auf:

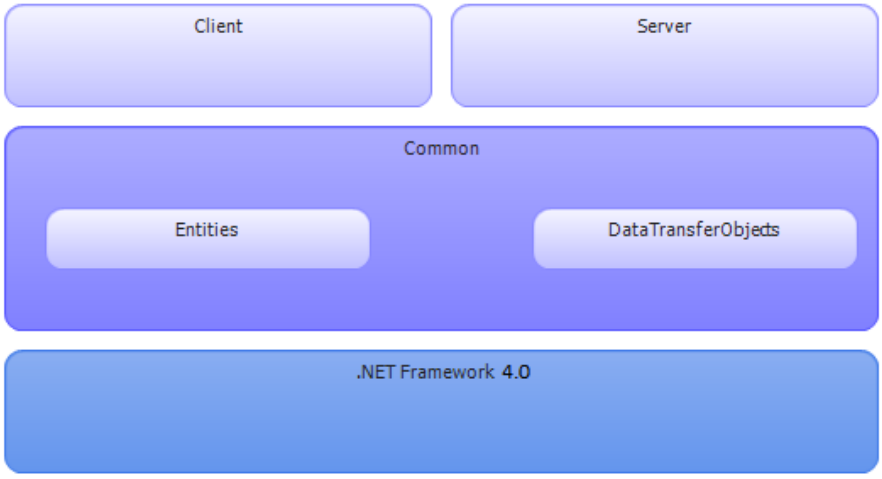


Abbildung 7.2.: Layerdiagramm

Es soll möglichst viel Funktionalität resp. Code sowohl vom Client als auch vom Server genutzt werden können. Deshalb wird versucht, das Common-Layer möglichst umfangreich zu gestalten.
Die obigen Layers werden auf folgende Assemblies abgebildet:

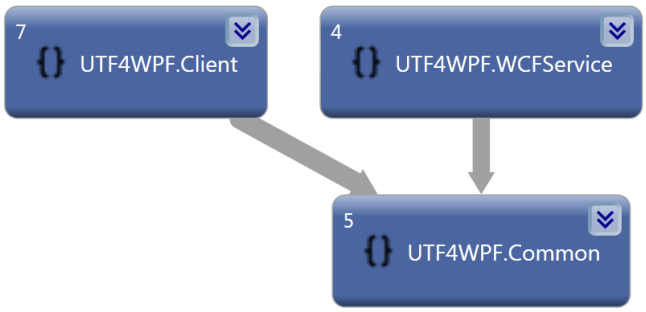


Abbildung 7.3.: Assembly-Namespaces

7.3. Packages

Die oben genannten Assemblies, bzw. deren Packages werden folgend beschrieben. Es ist zu beachten, dass mit einer Inline-Dokumentation gearbeitet wurde. Eine ausführlichere Code-Beschreibung kann somit in dieser nachgeschlagen werden.

7.3.1. Client

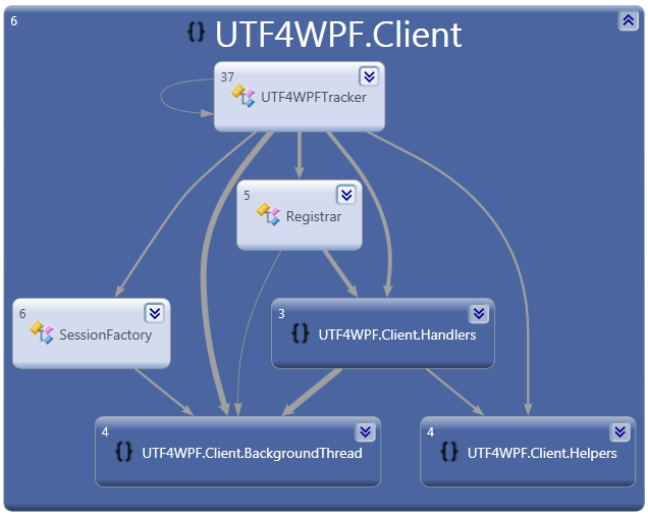


Abbildung 7.4.: Client Namespace

Der Client-Namespace enthält, seinem Namen entsprechend, Client-spezifischen Code. Darin enthalten sind die Tracker-Klasse, die den Benutzern des Frameworks als API dient, sowie die Mechanismen um Eventhandler zu registrieren und auftretende Events aufzuzeichnen.

- **UTF4WPFTracker** Dies ist die Schnittstelle zur Anwendungssoftware. Sie ist zuständig um eine neue Session zu starten, die Konfiguration einzulesen und umzusetzen.
- **Registrar** Ist zuständig für die Registrierung und anschliessende Deregistrierung von Event-Handlern.
- **SessionFactory** In der Session Factory wird die Session initialisiert und anschliessend über ein Work-Item des ThreadPools mit den Umgebungsangaben (OS, CPU, Lokalitätsinfos etc) versehen.
- **Subpackage Handlers** Beinhalten die verschiedenen Handler um zu trackende Events und Exceptions entsprechend abzuhandeln bzw. zu registrieren.
- **Subpackage BackgroundThread** Beinhaltet alle Operationen, welche im Hintergrund ausgeführt werden. Darin enthalten ist, neben dem eigenen Threadpool und der Komponente welche die Events speichert, auch die

Synchronisationsklasse. Diese Komponente ist zuständig für den Abgleich mit dem Server. In dieser werden Lookup-Tables mit dem Server synchronisiert und die anfallenden Sessions und Occurrences an den Server übermittelt.

- **Subpackage Helpers** Bietet einige Helfer-Klasse an, dies sind Extension Methods zur Informationssammlung sowie ein MultiValueDictionary um, pro geöffnetem Window, die registrierten Handler festzuhalten.

7.3.2. Service

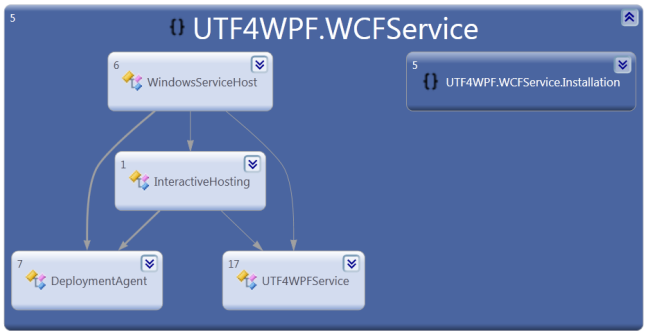


Abbildung 7.5.: Service Namespace

Dies ist die Serverkomponente, welche an zentraler Stelle die Sessions mit deren Eigenschaften und Occurrences von den verschiedenen Clients entgegennimmt.

- **UTF4WPFService** Beinhaltet alle Methoden welche über den WCF Service erreichbar sind.
- **Hostingkomponenten** Der Server kann entweder als Windows Service gehostet werden, oder aber mittels Interactive-Hosting, also in einer Konsolen-Anwendung.
- **DeploymentAgent** Stellt die Initialisierung und Erreichbarkeit der zentralen Datenbank sicher.
- **Subpackage Installation** Darin sind Komponenten implementiert, welche die Installation ermöglichen bzw. vereinfachen. Diese installieren den Windows-Service, ermöglichen die Auswahl einer SQL-Server Datenbank sowie die Einrichtung einer Windows Firewall-Regel.

7.3.3. Common

Was von Client und Server gleichermassen verwendet wird, ist im Common-Namespace implementiert.

- **Subpackage Entities** Beinhaltet den Aufbau der Datenbank in Form des Entity Models, welches auf dem Server und dem Client identisch aufgebaut ist. Die daraus generierten Entity Klassen, eigene Erweiterungen dazu, sowie die SQL-Files sind ebenfalls in diesem Package. Dazu gibt es zwei

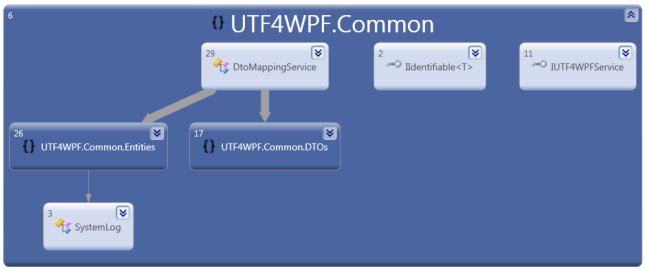


Abbildung 7.6.: Common Namespace

Generator-Files für die Erstellung der Entity Klassen und der entsprechenden Umsetzung für den SQL-Server. Ebenfalls bieten darin DataFacade's den Zugriff auf die Daten an, wobei soviel wie möglich mit Generics vereinheitlicht wurde.

- **Subpackage DTOs** DataTransferObjects sind ein minimierte Versionen der Entity Klassen, diese beinhalten keine Referenzen und Methoden, sondern sind rein für den Transport zwischen Client und Server gedacht.
- **DtoMappingService** Konvertiert DTOs zu Entities und umgekehrt.
- **IUTF4WPFService** Das Service Interface wird ebenfalls hier bereitgestellt und ermöglicht das einfache Entwickeln gegen den Service, ohne das der Server die Informationen über das Internet zur Verfügung stellen muss.
- **SystemLog** Um Informationen auf eine einfache Art in das EventLog von Windows einzufügen, stellt diese Klasse uns eine einheitlichen Zugriff zur Verfügung.

8. Architektonische Ziele und Prinzipien

An dieser Stelle sollen einige grundsätzliche Überlegungen und Wünsche ausformuliert werden. Die hier beschriebenen Punkte sollen weder als vollständige Liste wahrgenommen werden, noch sollen sie als direkte Quelle bei der Implementierung herangezogen werden.

8.1. Grundsätzliche Ziele

8.1.1. Stabilität und Integrität

Bei der Kommunikation über ein Netzwerk ist die Liste potentieller Fehler traditionell sehr lang. So auch beim Umgang mit dem UTF4WPF-Service. Die Client-Applikation soll nicht von der Stabilität der Verbindung abhängen, sondern in jedem Fall normal weiter funktionieren können. Fehler im Bereich des Frameworks sollen deshalb nicht zum Nutzer gelangen, sofern dies nicht aufgrund eines kritischen Ausmasses notwendig ist.

8.1.2. Detaillierte Inline-Dokumentation für Entwickler

Dokumentation veraltet, sobald sie einmal geschrieben ist. Je grösser die Distanz zwischen der Dokumentation und darin beschriebenen Aspekte ist, desto schwieriger wird es, die Dokumentation auf aktuellem Stand zu bringen. Diesem Umstand soll entgegen gewirkt werden.
Wie in den Entscheidungen auf Seite 61 genannt wurde wird auf XML-Annotations im Code gesetzt.

8.1.3. Hosting des Services

Das Hosting des Server-Teils, einer Komponente die mit WCF (Windows Communication Foundation[1]) umgesetzt wird, bietet verschiedene Möglichkeiten an.
Es wurde bestimmt, dass der Service als Standalone Windows-Service laufen soll. Die verschiedenen Möglichkeiten und unsere Entscheidungsgründe sind auf Seite 62 beschrieben.

[1] `http://en.wikipedia.org/wiki/Windows_Communication_Foundation`

8.2. Geplante Integrationszeit

8.2.1. Client

Für die Einbindung in einem Projekt muss eine Referenz auf das UTF4WPF-Assembly im Visual Studio hinzugefügt werden. Danach ist eine einmalige Einbindung im Quelltext notwendig. Dazu gehören weniger als fünf Zeilen Code für die Initialisierung des Trackers und die Konfiguration der zu überwachenden Events in der entsprechenden (XML-)Konfigurationsdatei. Sofern der Mechanismus und die zu sammelnden Events bekannt sind wird für die ganze Arbeit eine halbe Stunde gerechnet.

8.2.2. Server

Die Serverseite muss nur ein einziges Mal eingerichtet werden, es kann derselbe Server für mehrere Projekte verwendet werden. Als Grundlage wird ein bestehender Server mit folgendem Minimalsetup vorausgesetzt:

- Windows Server 2003 (mindestens SP2) oder neuer
- Microsoft SQL Server 2008 oder neuer
- Erreichbar über das Internet bzw. Netzwerk

Die Einrichtung besteht im Download und Ausführen eines Setups. Dies dauert geschätzte 20 Minuten.

8.2.3. Total

Somit kann ein C#-Programmierer das ganze Framework in rund einer Stunde nutzbar machen. Für einen produktiveinsatz wird empfohlen, sich genaue Gedanken zu machen, welche Events aufgezeichnet werden sollen.

8.3. Wahrung der Benutzerprivatsphäre

Die Privatsphäre der Enduser soll jederzeit gewahrt werden. Es werden daher keine Informationen gespeichert, die einen Rückschluss auf einen Benutzer zulassen, wie etwa eine IP-Adresse. Es wird zudem verlangt, dass die Entwickler, die das Framework einsetzen, die Erlaubnis des Endusers einholen bevor sie den Tracker starten können. Dies zu erzwingen ist nicht via Programmcode möglich, es wird aber bewusst in der Integrationsdokumentation erwähnt, und auch die Beispielkonfiguration im Manual kann nicht “Out of the box” verwendet werden, bevor nicht eine entsprechende Funktion zur Einholung der Erlaubnis implementiert wurde.

9. Domänenmodell

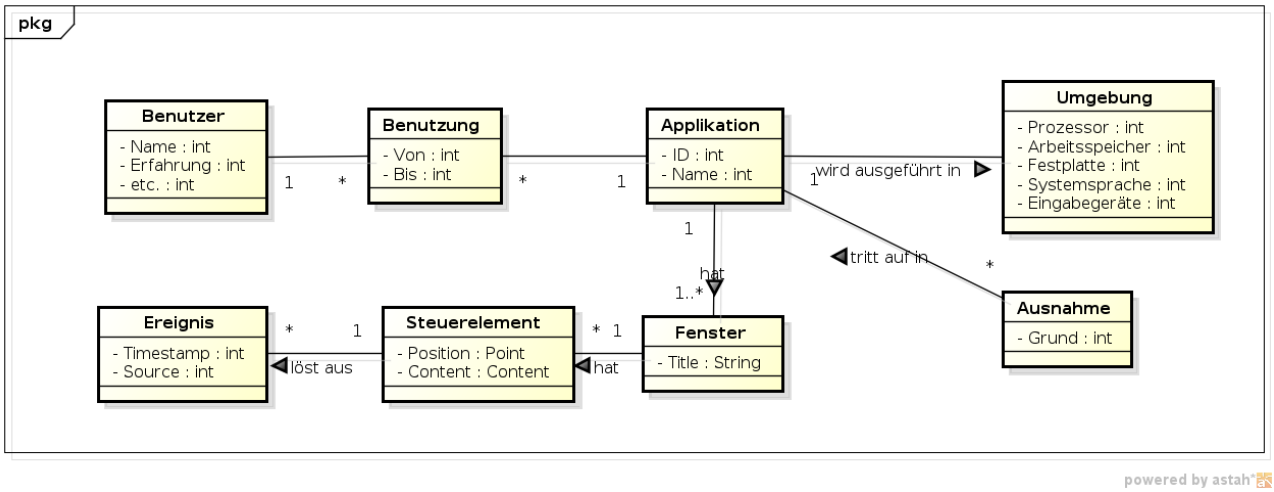


Abbildung 9.1.: Domänenmodell

UTF4WPF zeichnet Ereignisse in der grafischen Oberfläche von Applikationen auf. Dazu werden die einzelnen Steuerelemente wie Buttons, Dropdown-Menüs oder Texteingabefelder überwacht. Jeweils alle Events vom Zeitpunkt des Applkationsstarts bis zu deren Beendung werden als eine zusammengehörende Benutzung gewertet. Während einer Benutzung werden auch auftretende Ausnahmen, also Fehler, mitaufgezeichnet. Für zusätzliche Analysen wird auch der Kontext, in dem die Applikation ausgeführt wird (Umgebung) erfasst.

10. Datenmodell

Erläuterungen zum Diagramm auf Seite 45

Sowohl für den Server, als auch den Client wird die gesamte Tabellenstruktur verwendet, auf dem Client wird diese jedoch mit deutlich weniger Datensätzen befüllt. Trotz dessen bietet dies Vorteile wie die Wiederverwendbarkeit und den geringeren Synchronisationsaufwand.
Dazu werden für die Entitäten CPUModel, OperatingSystem, Locale, Program, GUIComponent und EventType jeweils dynamische Lookup-Tables verwendet. Diese Tabellen sind bei der Initialisierung leer und werden fortlaufend mit den anfallenden Datensätzen befüllt, wobei Duplikate vermieden werden. Die IDs dieser Einträge werden jeweils einmalig mit dem Server abgeglichen und anschliessend wiederverwendet. Nach der ersten Synchronisation hat der Client somit die gleichen IDs wieder der Server und es ist kein zusätzlicher Synchronisationsaufwand für diese Einträge notwendig.

Mapping Domänenmodell	$\Leftrightarrow$	**ER-Diagramm:**
Benutzung	$\leftrightarrow$	Session
Applikation	$\leftrightarrow$	Program
Umgebung	$\leftrightarrow$	Environment
Ausnahme	$\leftrightarrow$	ExceptionalOccurrence
Fenster, Steuerelement	$\leftrightarrow$	GUIComponent
Ereignis	$\leftrightarrow$	Occurrence

Da keine Rückschlüsse auf den Benutzer möglich sein sollen, werden keine Informationen diesbezüglich erfasst, folglich erscheint er auch nicht im ER-Diagramm.

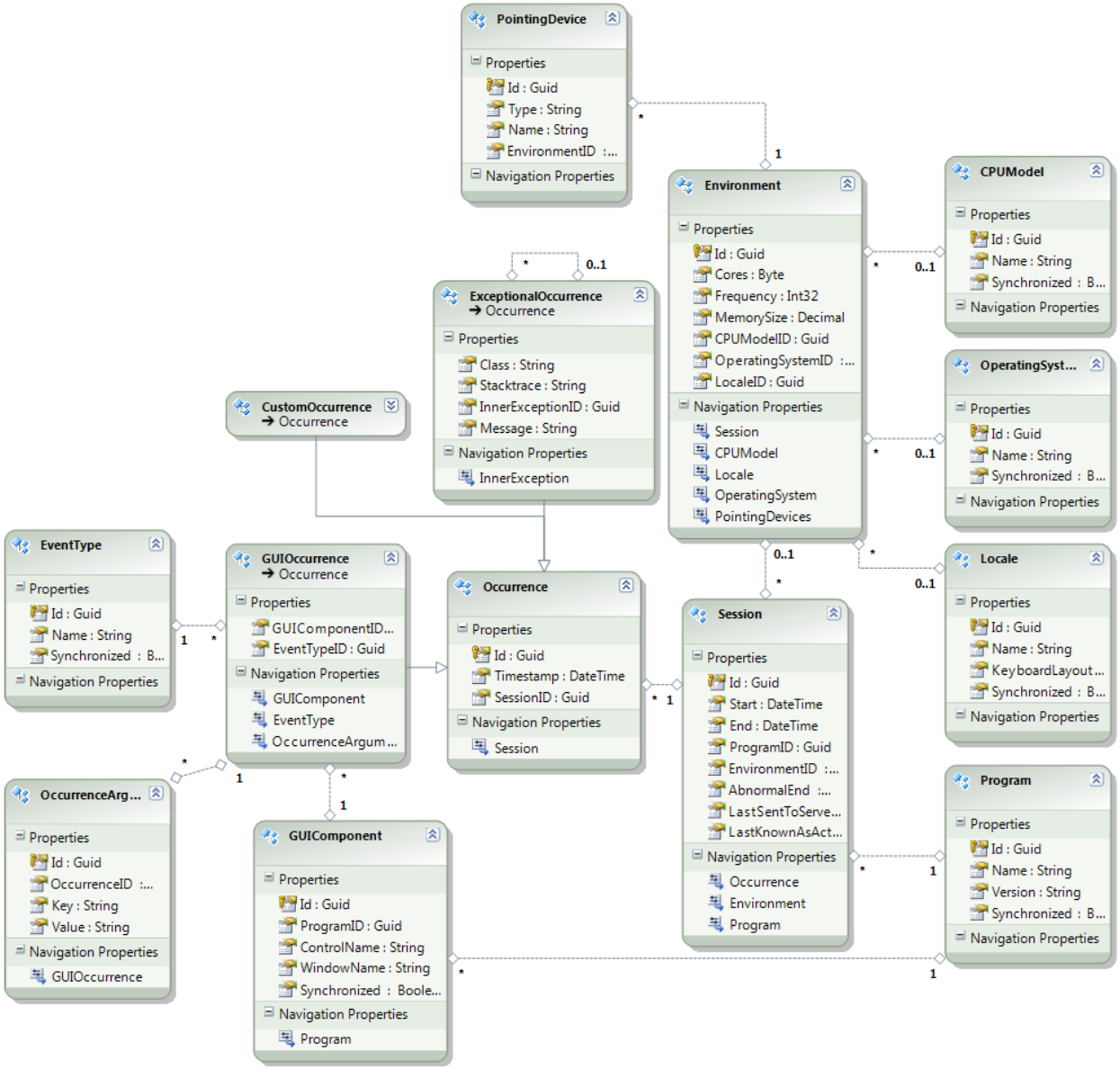


Abbildung 10.1.: ER-Diagramm

10.1. Verschiedene Datenbanksysteme

Die Sammlung der Daten soll auch dann stattfinden, wenn keine Verbindung zum Server besteht. Auf der Serverseite benötigen wir aufgrund der reinen Datenmenge und der Auswertbarkeit einen SQL Server 2008 oder höher. Auf dem Client soll jedoch möglichst wenig installiert werden, sodass ein SQL Server Compact Edition verwendet wird.
Diese Systeme unterscheiden sich minimal voneinander, weshalb die Datenablage lokal und auf dem Server über das gleiche Datenmodell ablaufen kann.
Es handelt sich dabei jedoch um zwei verschiedene Datenbank Provider wodurch ein gewisser Extra-Aufwand bei der Implementierung entsteht:

- Um die Datenbanken anzusprechen ist im Client- und im Server Projekt ein separater Connection String anzugeben
- In der SSDL (Store Schema Definition Language) wird das Schema für das jeweilige Datenbanksystem definiert. Das SSDL ist ein Bestandteil des EDMX-Files welches das Konzeptuelle- bzw. das Speichermodell definiert und aufeinander abbildet. Das EDMX, bzw. dessen SSDL, unterscheidet sich je nach Datenbanksystem minim. Im Projekt UTF4WPF.Common wurde deshalb ein Generator File erstellt (Entities\UTF4WPF_Server.tt) welches das EDMX für die SQL-Server Datenbank automatisch aus dem EDMX-File des Clients (SQL CE) generiert.

11. Abgleich Server und Clients

11.1. Logische Kommunikation Client $\Leftrightarrow$ Server

Die Kommunikation wird periodisch vom Client ausgelöst. Falls noch nicht geschehen, gibt er dem Server die lokalen Informationen zu GUI-Komponenten, Hardwareinformationen und Session-Informationen bekannt. Diese Meta-Informationen werden vom Server eingetragen, falls einer der Einträge auf dem Server bereits existiert, gibt er die ID des Server-Eintrags dem Client bekannt. Dieser updatet dann seinen Eintrag und alle lokalen Referenzen darauf. Dieser Vorgang ist im Kapitel 11.3 genauer beschrieben.
Danach werden die einzelnen Events (Occurrences) an den Server gesandt. Diese werden beim Client zu Tranchen von mehreren Events gebündelt, welche dann als Ganzes verschickt werden, um die Anzahl der Netzwerkrequests zu reduzieren. Eine Tranche kann nur vollständig oder gar nicht übertragen werden. Nach der Übertragung und erfolgreicher Rückmeldung des Servers wird die Tranche beim Client gelöscht.
Wenn eine Benutzersession geendet hat, werden die letzten Events und die Endzeit an den Server übertragen. Sobald die Session komplett an den Server gesandt wurde wird diese beim Client gelöscht.
Im nachfolgenden Diagramm (Seite 48) ist dieser Ablauf bildlich dargestellt.

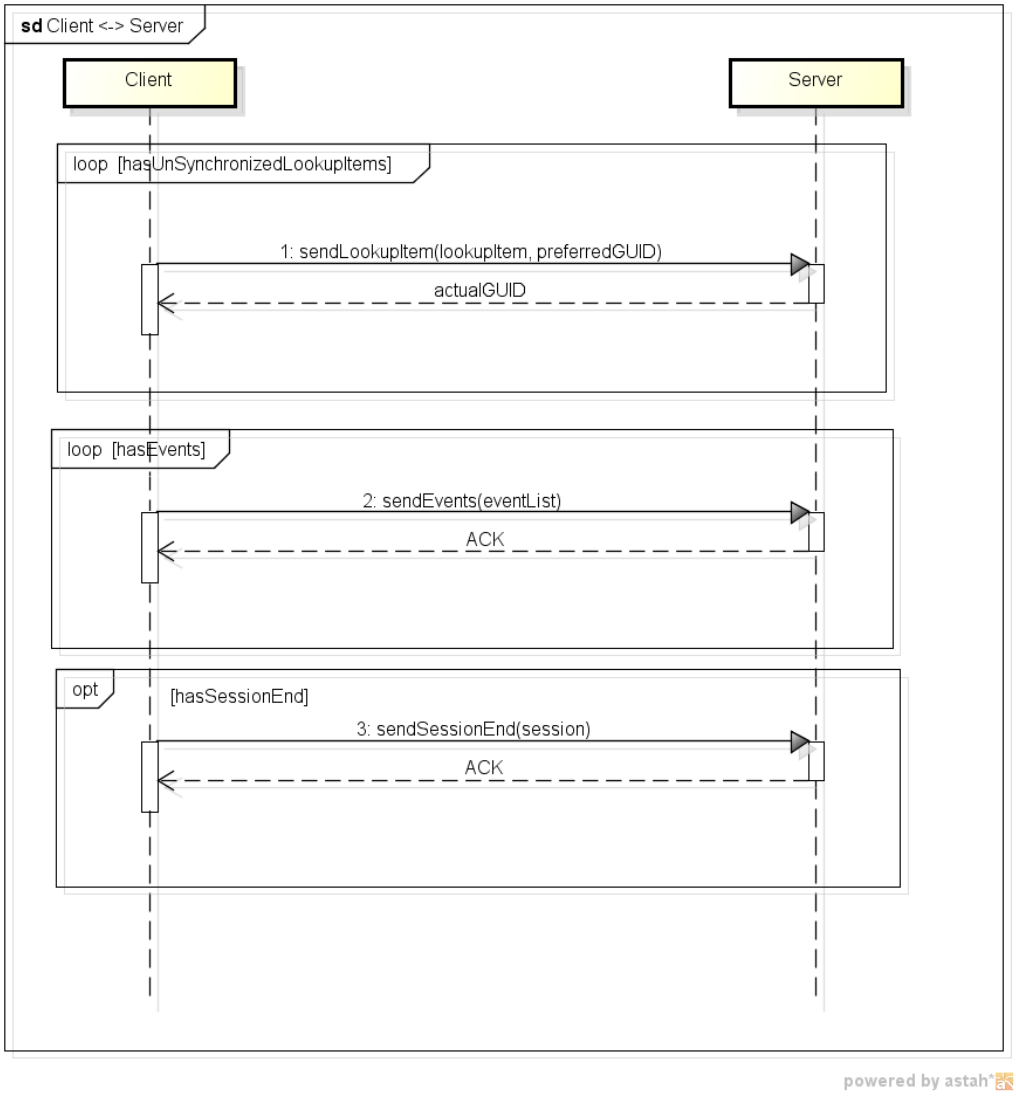


Abbildung 11.1.: Client ⇔ Server Kommunikation

11.2. Fehlersituationen

Bei der Übertragung kann es zu einigen speziellen Konstellationen kommen, diese werden im Folgenden beschrieben.

11.2.1. Server nicht erreichbar

Wenn der Server nicht erreichbar ist, wird die Übertragung periodisch wiederholt. Dabei wird das Warteintervall mit jedem misslungenen Versuch um 20% gesteigert. Das maximale Warteintervall zwischen den Versuchen wurde auf 10 Minuten festgesetzt. Nach einer erfolgreichen Übertragung wird dieser wieder auf eine Minute herabgesetzt.

11.2.2. Server gibt keine Rückmeldung

Der Server hat die Anfrage erhalten, jedoch geht die Rückmeldung des Servers auf dem Weg zum Client verloren. Aus Sicht des Clients ist die Übertragung somit noch offen, er versucht die Übertragung später nochmals. Wie mit doppelt gesendeten Entities umgegangen wird ist im Kapitel 11.2.3 beschrieben.

11.2.3. Handhabung bereits bekannter Guids

Ist die Guid einer empfangenen Entity bereits auf dem Server vorhanden können zwei Situationen eingetreten sein:

1. Doppelte Übertragung, etwa aufgrund eines Verbindungsunterbruchs
2. Zufall einer doppelt vergebenen Guid. Mehr dazu kann im Kapitel 15 nachgelesen werden. Die Chancen dafür sind grundsätzlich sehr, sehr klein.

Je nach Typ der empfangenen Entities wird anders reagiert:

- **Session/Environment:** Auf dem Server wird geprüft, ob die Kombination von Session/Environment IDs des Empfangenen mit jener auf dem Datenbank-Server übereinstimmt. Ist dies der Fall wird dem Client ein erfolgreiches Speichern mitgeteilt. Stimmt die Kombination nicht überein könnte der Client zu einer Guid-Anpassung beauftragt werden, momentan wird dies aber nicht implementiert.
- **Entities in Lookup-Tabellen**. Wenn hier doppelte Guids eintreffen, handelt es sich entweder um den gleichen Eintrag, was dem Client als erfolgreiche Synchronisation mitgeteilt wird, oder aber es handelt sich tatsächlich um eine die gleiche Guid für verschiedene Einträge. In diesem Fall wird der Eintrag mit einer neuen Guid abgelegt und dem Client als Guid-Wechsel mitgeteilt.
- **Occurrences** haben die weit höhere Wahrscheinlichkeit auf doppelte Übermittlung (siehe Fehlsituationen) als die der doppelten Guid's. Es wird somit angenommen, dass die Occurrence doppelt übermittelt wurde.

Der Grund für die Abweichung zwischen den Entity-Typen:

1. Auf den Lookup-Tabellen musste der Guid-Wechsel bereits aufgrund derer Natur, dass alle Clients mit den gleichen Guids arbeiten sollen, umgesetzt werden.
2. Auf den Sessions führt die Umsetzung zu einem Mehraufwand der sich nicht mit der niedrigen Wahrscheinlichkeit in Verbindung bringen lässt.

11.3. Abgleich von Server und Client-Datensätzen

Für den Abgleich von Datensätzen zwischen Client und Server wurde entschieden, vor der Übertragung der eigentlichen Daten eine Synchronisation der Lookup-Tabellen durchzuführen. Es werden somit nur Sessions und Occurrences übermittelt, die auf bereits synchronisierte Lookup-Einträge zeigen. Mehr zu den geprüften Varianten und der Entscheidung ist auf Seite 63 zu finden.

Der Abgleich dieser Lookuptabellen funktioniert, gezeigt am Beispiel der “Locale” Tabelle, folgendermassen:

1. Ein imaginärer Client A verwende als Systemsprache English (US) und als KeyboardLayout English(US). Beim ersten Start des Tracking-Frameworks wird in der lokalen Datenbank ein Eintrag mit der zufällig gewählten GUID 888 angelegt (Die tatsächlichen GUIDs sind viel länger, um Kollisionen zu vermeiden, hier werden der Einfachheit halber kurze GUIDs verwendet).
2. Der Client sendet den neuen, bisher unsynchronisierten Eintrag an den Server. Dieser wiederum legt den Eintrag, welcher dem Server bisher nicht bekannt war, bei sich an und bestätigt dem Client die ID 888. Der Client markiert die Locale bei sich als synchronisiert.
3. Ein zweiter Client B verwendet die selben Einstellungen wie A. Auch bei ihm wird lokal ein Eintrag generiert, allerdings mit der ebenfalls zufällig gewählten ID 999. Bei der Synchronisation erkennt der Server das Duplikat und sendet B die zuvor gewählte ID 888 zurück.
4. Der Client aktualisiert den lokalen Eintrag, sowie alle Einträge die darauf referenzieren, mit der erhaltenen ID 888 und markiert die Locale als synchronisiert.
5. Ein Client C verwende als Systemsprache Deutsch (Schweiz) und als KeyboardLayout Dvorak. Die zufällig gewählte ID sei 555. Beim Synchronisieren wird der Server den Eintrag bei sich als unbekannt erkennen, neu anlegen und C die ID 555 bestätigen.
6. Der Client markiert die Locale als synchronisiert.

11.4. Data Transfer Objects

Über das Netzwerk werden nicht die Entity Objekte gesendet, sondern sogenannte Data Transfer Objects (DTO[1]). DTOs sind ein Design Pattern welches genutzt wird um Daten zwischen Software(-Komponenten) zu übertragen.
Wikipedia[2] deklariert den Unterschied zwischen den Business Objekten und DTOs wie folgt:

[1] http://msdn.microsoft.com/en-us/library/ms978717.aspx
[2] http://en.wikipedia.org/wiki/Data_transfer_object

The difference between data transfer objects and business objects or data access objects is that a DTO does not have any behavior except for storage and retrieval of its own data (accessors and mutators). DTOs are simple objects that should not contain any business logic that would require testing.

12. Wichtige Abläufe auf dem Client

12.1. Tracker-Initialisierung

Der Tracker wird jeweils beim Applikationsstart initialisiert, zumindest empfehlen wir dies dem Frameworknutzer. Beim Start werden einige wichtige Komponenten sowie die Datenbank initialisiert. Ebenfalls registrieren wir uns auf einige Events welche unser Framework aufrufen.

- **Applikationsende** Damit wir eine gestartete Session sauber beenden können, lassen wir uns über das Applikationsende informieren.
- **UnhandledExceptions** Als wichtige Anforderung sollen Exceptions, welche das Programm nicht abhandelt, geloggt werden.
- **Loaded-Event von Windows und UserControls** Anhand dieser Events können wir in der Konfiguration prüfen, ob auf den neuen Windows/UserControls zusätzliche Elemente zu überwachen sind. Ein Beispiel für eine solche fensterspezifische Aktion ist das Tracken eines bestimmten Buttons, wie es im Sequenzdiagramm im Kapitel 12.2 gezeigt wird. Die Registrierung auf neue Events übernimmt eine Instanz der Registrar-Klasse.

Im nachfolgenden Diagramm ist der Start bildlich dargestellt.

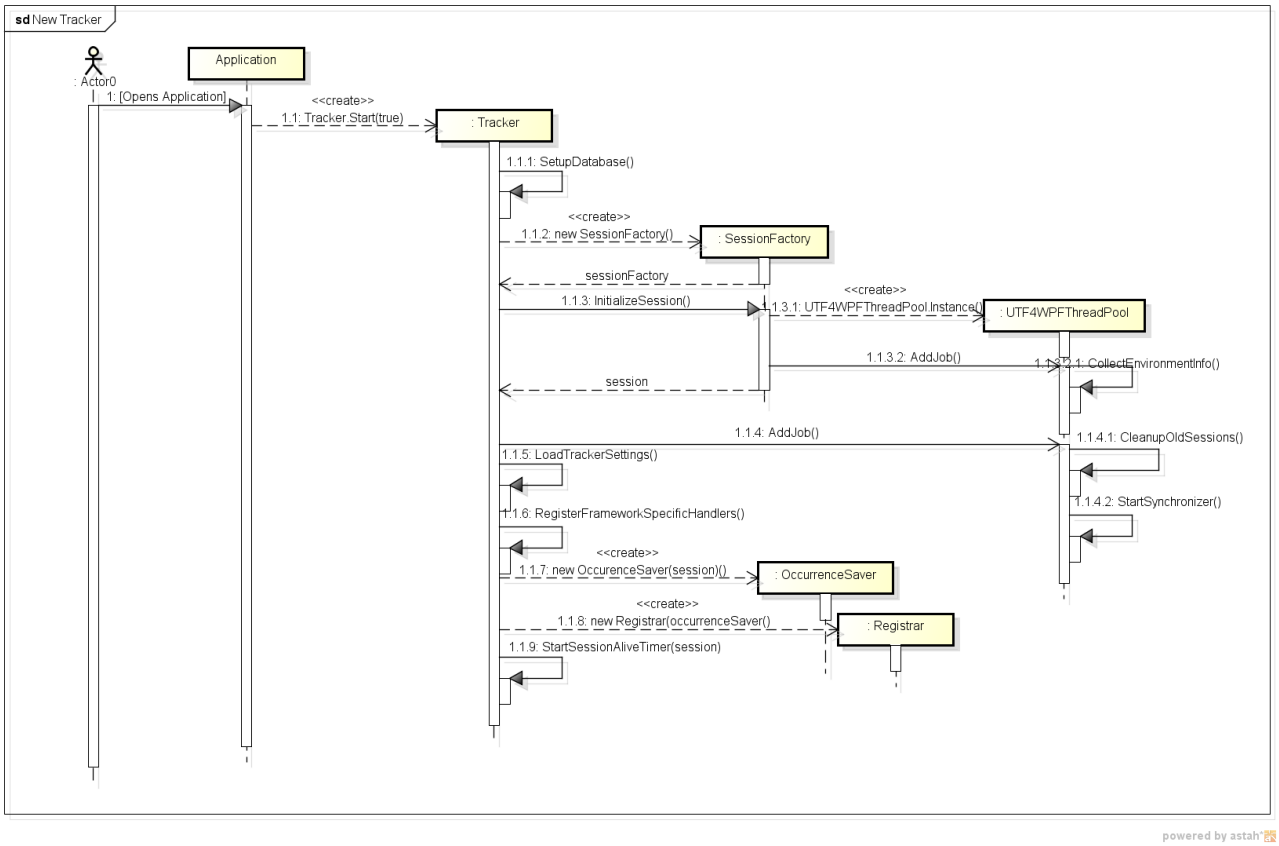


Abbildung 12.1.: Ablauf Tracker-Initialisierung

12.2. Prüfung Fenster spezifischer Aktionen

Wird innerhalb einer überwachten Applikation ein neues Fenster geladen, so wird der Tracker entsprechend informiert. Dies geschieht über das LoadedEvent des Fensters welches während der Initialisierung des Trackers (vgl. Kapitel 12.1) registriert wurde. Dieses Event[1] wird ausgelöst sobald das Fenster gerendert und zur Interaktion bereit ist.
Der Tracker wird anhand der *filters.xml* prüfen welche Events zur prüfen sind und diese zur Registrierung dem Registrar übergeben. Der Registrar führt intern eine Liste der registrierten Handlers; anhand dieser kann er beim Schliessen eines Fensters die Eventhandler wieder deregistrieren. Im Beispiel wird dabei explizit das Click-Event des ExpButtons zur Überwachung registriert.
Um uns an die Events anhängen zu können wird auf Reflections zurückgegriffen. Damit ist es möglich Informationen aus fremden Assemblies, in unserem Fall die zu überwachende Applikation, abzufragen und auch anzusprechen.
Im nachfolgenden Diagramm ist dieser Ablauf bildlich dargestellt.

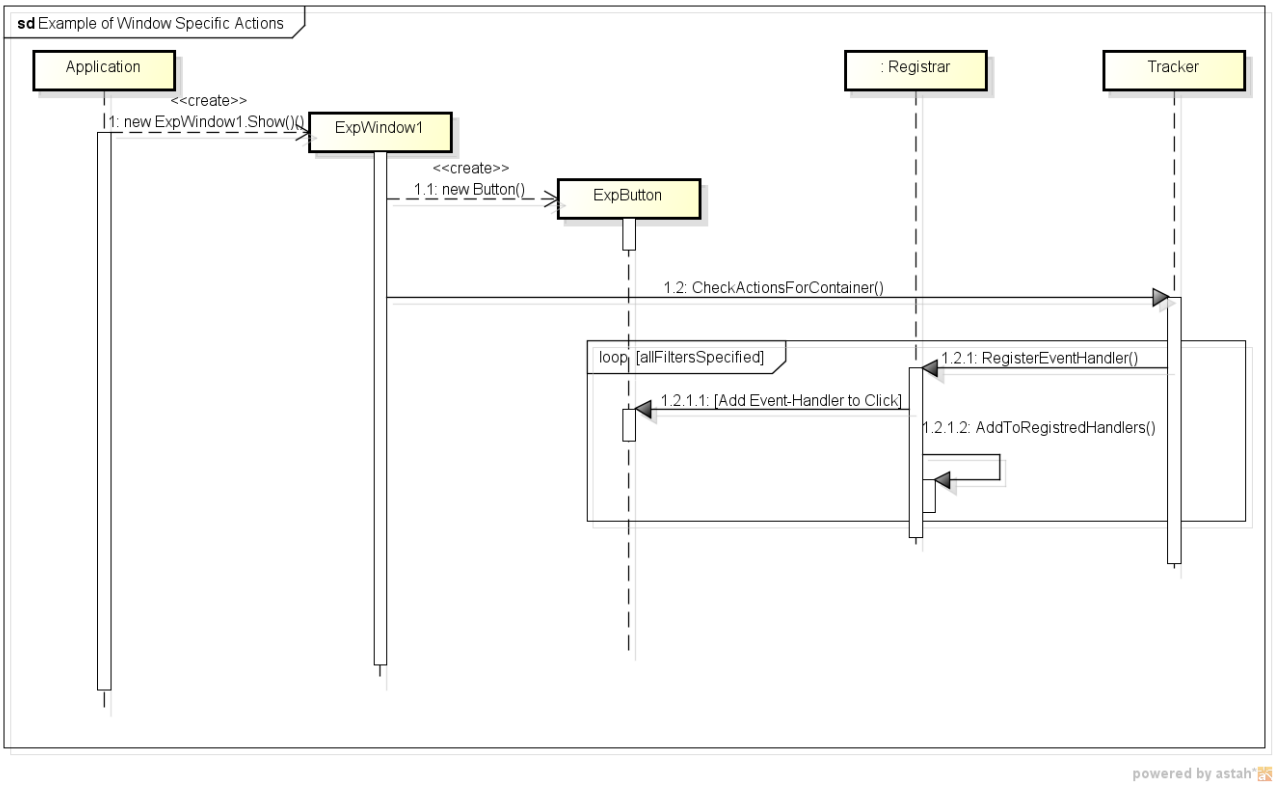


Abbildung 12.2.: Ablauf zur Prüfung von Fenster spezifischen Aktionen

[1] http://msdn.microsoft.com/en-us/library/system.windows.frameworkelement.loaded.aspx

12.3. Abfangen eines Events

Sobald ein Event geworfen wird, z.B. ein Klick auf einen Button, so werden die registrierten Event-Handler darüber informiert. Dies kann in unserem Falle neben der normalen Eventbehandlung auch der CommonEventHandler des Trackers sein. Ist dies der Fall, so werden die gewünschten Informationen gesammelt (Zeitpunkt, Auslöser, Fenster, Eventname wie z.B. Click etc.). Diese Informationen werden in einen UTF4WPFThreadPool zur Speicherung übergeben. Die Speicherung selbst übernimmt der OccurrenceSaver.

Im nachfolgenden Diagramm ist dieser Ablauf bildlich dargestellt.

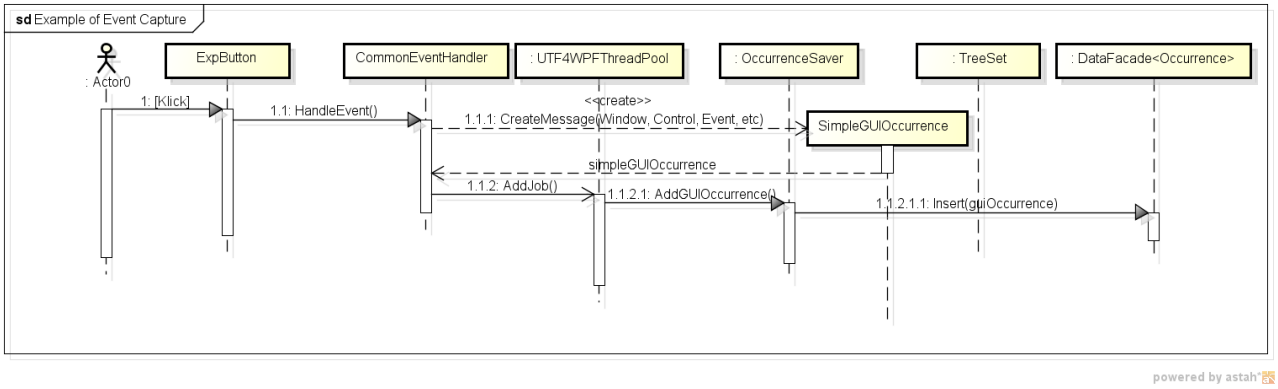


Abbildung 12.3.: Ablauf des Abfangens eines Events

13. Datenverwaltung

Erläuterungen zum Diagramm auf Seite 58

Das Diagramm beschreibt die Art der Datenhaltung wie sie auf dem Client und auch auf dem Server durchgeführt wird. Die Klassen sind in ein eigenes Assembly (“UTF4WPF.Common.dll”) ausgelagert, damit Server und Client einheitlich auf den Daten operieren.

Für **lokale** Aktionen, sei es Client- oder Serverseitig wird mit dem Package “Entities” gearbeitet. Für jede Tabelle im Datenbankschema existiert eine zugehörige Klasse welche die Daten um Funktionalität erweitert. Dazu existiert pro SubKlasse von “EntityObject” eine Instanz von “DataFacade”, welche alle benötigten Datenbankoperationen zur Verfügung stellt. Sowohl Client als auch Server können alle Datenoperationen via DataFacade ausführen, ohne sich um die Datenbankverbindung kümmern zu müssen. DataFacade verwendet dabei Generics, mehr dazu wird auf Seite 64 beschrieben.

Für die Übertragung über eine Netzwerkverbindung werden DataTransferObjects verwendet, siehe dazu auch Kapitel 11.4 auf Seite 50. Rein *bildlich* gesprochen funktioniert der Ablauf vor und nach der Übertragung wie folgt:

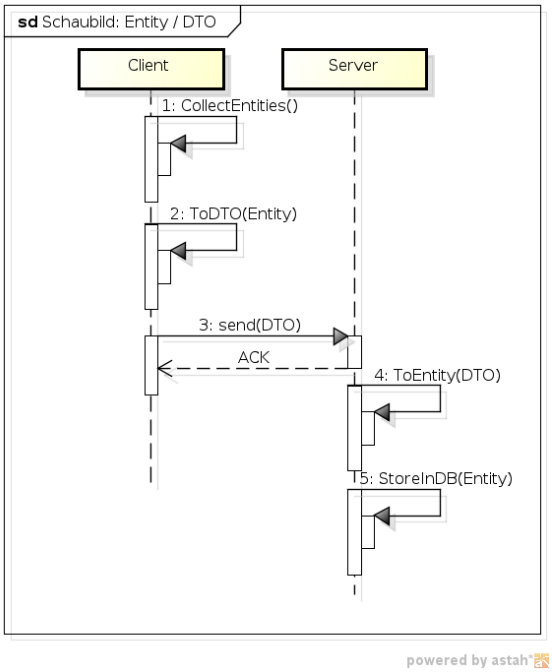


Abbildung 13.1.: Umwandlung Entity $\leftrightarrow$ DTO

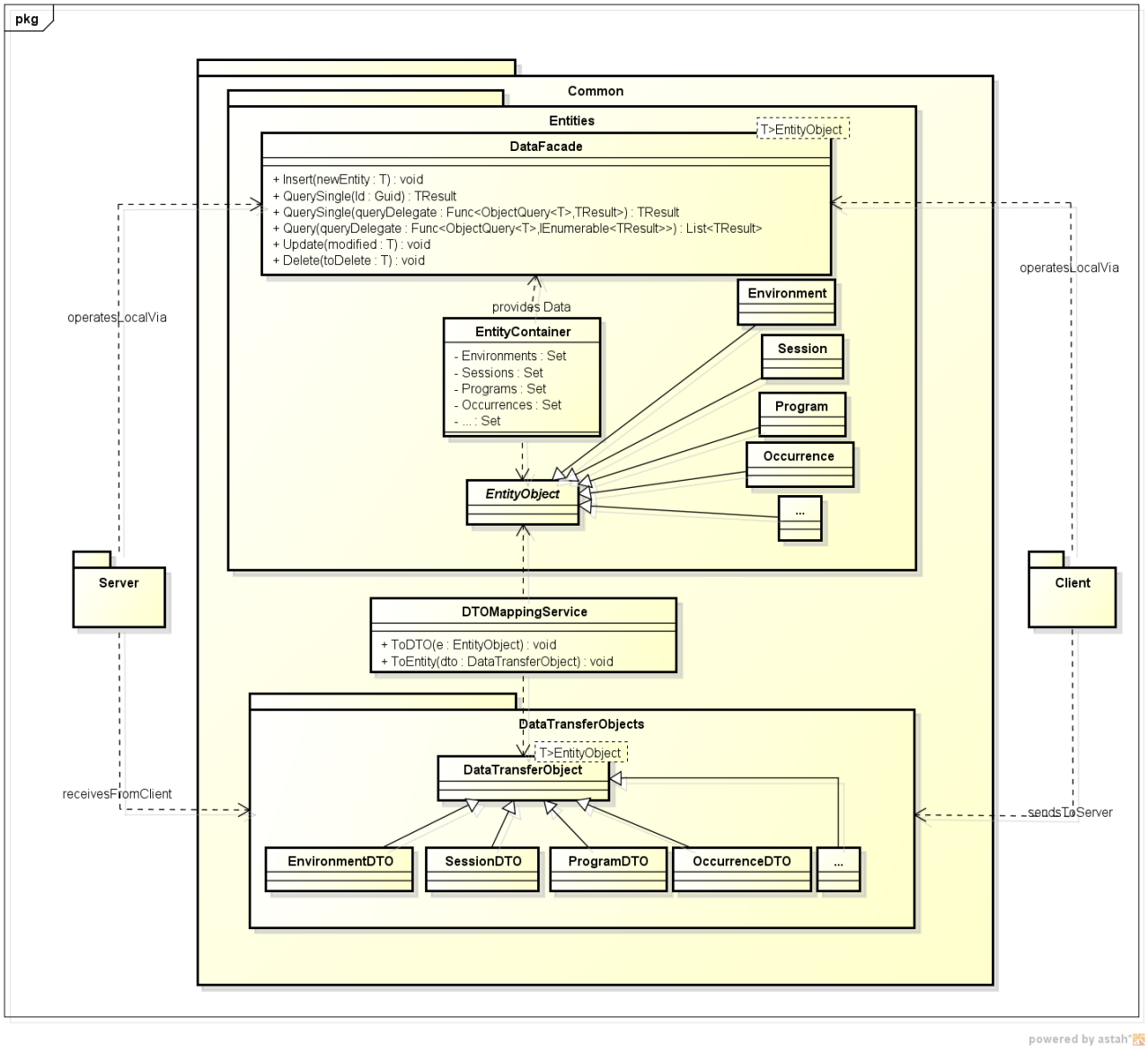


Abbildung 13.2.: Datenverwaltung

13.1. Datenmenge

Die anfallende Datenmenge muss den Anforderungen[1] gewachsen sein. Im Folgenden wird versucht, die Datenmenge für das definierte Beispielszenario zu schätzen. Für die Grösse der einzelnen Datensätz werden die "Breiten" der Datenbank-Tabellenspalten addiert.
Es ist von 100'000 zu speichernden Events die Rede, angenommen diese teilen sich auf folgendermassen in Unter-Typen auf,

95%	GUI	à	$(6\,ID's + 1\,Datetime + 5\,EventArgs \simeq)$	250 Bytes
<5%	Exceptional	à	$(3\,ID's + 1\,Datetime + 2\,Strings + 1\,Stacktrace \simeq)$	4100 Bytes
<1%	Custom	à	$(2\,ID's + 1\,Datetime + 1\,Message \simeq)$	100 Bytes

und ergibt somit die folgenden Datenmenge:

$$95'000 \cdot 250\,Bytes + 5000 \cdot 4100\,Bytes + 1000 * 100\,Bytes \simeq 42 \cdot 10^6\,Bytes \widehat{=} 42\,MB$$

Hinzu kommen die Zusatzinformationen von 550 Sessions und 25 Arbeitsumgebungen der Benutzer die sich wie folgt zusammensetzen könnten:

550	Sessions	à ca.	60	Bytes
550	Environments	à ca.	70	Bytes
70	GUIControls	à ca.	70	Bytes
25	PointingDevices	à ca.	60	Bytes
10	CPUModels	à ca.	40	Bytes
5	EventTypes	à ca.	30	Bytes
4	OperatingSystems	à ca.	40	Bytes
4	Locales	à ca.	50	Bytes
1	Program	à ca.	50	Bytes
	Total	ca.	80'000	Bytes

Dies ergibt in Summe also zusätzliche 80 Kilobytes, die im Vergleich zu den Occurrences vernachlässigbar klein sind. Die Gesamtdatenmenge für das Szenario beträgt also **weniger als 50 Megabytes**, dies kann selbst auf einem minimal ausgestatteten Server problemlos gespeichert werden. Im Rückschluss bedeutet dies pro Client ca. 2 Megabytes an Netzwerktraffic (nur Nutzdaten) im Zeitraum eines Monats. Auch dies dürfte bei heutigen Internetanbindungen kein Problem darstellen.

[1] Siehe Anforderungs-Dokument, Kapitel 6.7.2:"Mengenanforderung"

14. Threading

Das Framework wird in den Prozess der zu überwachenden Applikation integriert und teilt sich in mehrere Threads auf:

Tracker

Der Tracker ist dafür zuständig, die eigentlichen Events und Meldungen zu sammeln und in den lokalen Puffer (lokale Datenbankdatei) zwischenzuspeichern. Der Tracker ist in den GUI-Thread der überwachten Host-Applikation integriert. Er ist während der gesamten Laufzeit der Host-Applikation aktiv, bzw. vom Start bis zum Stop Aufruf. Damit die Reaktionsgeschwindigkeit des GUI nicht beeinträchtigt wird, werden die GUI-Occurrence, zur Speicherung in die Datenbank, dem vom UTF4WPFThreadPool, welcher auf Seite 60 beschrieben ist.

Synchronizer

Der Synchronizer ist eine Singleton-Klasse, die sich mithilfe eines Timers selbst regelmässig aufruft. Je nach Bedarf wird beim Ablauf des Timers einer oder mehrere der folgenden Aufgaben ausgeführt:

- Erzeuge eine neue Session auf dem Server
- Sende vorhandene Occurrences an den Server
- Beende/Update eine Session auf dem Server

Sollte eine dieser Aufgaben fehlschlagen, zum Beispiel weil keine Verbindung zum Server besteht, so wird sie beim nächsten Ablauf des Timers erneut aufgerufen. Solange bis die Übertragung funktioniert. Die Aufgaben werden ebenfalls im Threadpool des .NET-Frameworks ausgeführt, allerdings strikte sequentiell.

UTF4WPFThreadPool

Der Standard ThreadPool von .Net stellt keine Option zur Verfügung um beim Schliessen auf die Beendigung der offenen Tasks zu warten. Deshalb wurde dieser Wrapper erstellt welche eine Close-Methode mit einem Timout zur Verfügung stellt. Zudem kann eine maximale Anzahl Threads definiert werden, da die 1024 Threads des .NET-Threadpools (abhängig von der Hardware) für die Datenspeicherung eher die Performance beeinträchtigen als fördern.

15. Entscheidungen

Verwendung von SQL Server Compact

Für die Zwischenspeicherung der gesammelten Daten beim Client wird SQL Server Compact verwendet. Dies hat den Vorteil dass die Funktionalitäten einer Datenbank genutzt werden können, ohne die Installation eines zusätzlichen Datenbankdienstes zu erfordern. Zudem gleicht die Struktur des SQL Server Compacts stark jener des MSSQL Servers. Somit treten weniger Probleme auf, als es beispielsweise bei der Verwendung von SQLite gewesen wäre.

GUID als Identifier

Es wurde entschieden, für alle Datenbank-Records eine GUID[1] als Identifier in der Datenbank zu generieren, so kann die Vergabe der Session-ID direkt beim Client geschehen und dieser muss keine ID beim Server anfordern. Die Sessions können immer noch chronologisch mithilfe des Startzeit-Attributes geordnet werden. Auch die Identifizierung der Applikationen wird mit Hilfe der Projekt-GUID gelöst. (Siehe auch Datenmodell auf Seite 45).
Zur Eindeutigkeit einer Guid bzw. der Wahrscheinlichkeit einer doppelten Guid gibt es diverse Diskussionen, beispielsweise bei Stackoverflow[2]. Kurz gesagt sind die Chancen so enorm klein, dass diese ignoriert werden können.

Erfassung von Input-Device Informationen

Es wird, als zusätzliche Information zum Environment, die Eigenschaften der angehängten Pointing Devices[3] und zum ausgewählten Tastaturlayout erhoben. Dies hilft beispielsweise bei einer Usability-Analyse, ob GUI-Elemente für die Bedienung via Touch-Screen zu klein sind.

Inline-Dokumentation

Dokumentation veraltet, sobald sie einmal geschrieben ist. Je grösser die Distanz zwischen der Dokumentation und darin beschriebenen Aspekte ist, desto schwieriger wird es, die Dokumentation auf aktuellem Stand zu halten. Diesem Umstand soll entgegen gewirkt werden.

[1] http://de.wikipedia.org/wiki/GUID
[2] http://stackoverflow.com/questions/39771/is-a-guid-unique-100-of-the-time
[3] http://msdn.microsoft.com/en-us/library/aa394356.aspx

Was die Kommentare zum Quelltext von UTF4WPF betrifft, kann die Lücke geschlossen werden, indem man diese in Form von XML-Annotations direkt im Quelltext unterbringt. Das Resultat ist, dass die Dokumentation des Quelltextes, d.h. der darin beschriebenen Klassen und Methoden im Quelltext selbst vorhanden ist und nicht separat geschrieben werden muss. So wird nicht nur die Instandhaltung der Dokumentation gefördert, sondern in gewissem Umfang auch deren strukturelle Korrektheit validiert[4]. Weiterhin wird die Dokumentation im IntelliSense in Echtzeit für die Entwickler sowie auch die Frameworknutzer bereitgestellt, was die Arbeit im Team aber auch die Verwendung des Frameworks vereinfachen kann und wird.
Aus dem Quelltext wird direkt eine vollständige, strukturierte Entwickler-Dokumentation generiert. Dies wird mittels Doxygen erreicht, welches aus den XML-Kommentaren im Repository eine vollständig navigierbare Website generiert.

Hosting des Services

Für ein WCF-Hosting stehen gemäss Microsoft-Dokumentation[5] drei Hosting-Arten zur Verfügung:

1. Self-Hosting in a Managed Application: Flexibel und bei der Entwicklung nützlich. Für Enterprise-Solutions aber nicht geeignet.
2. Internet Information Services (IIS): IIS ist zwar inzwischen auf den meisten Windows-Versionen vorhanden, es ist jedoch zu beachten, das unter IIS 5.1 und 6 **keine** WCF-Services über HTTPS möglich sind. Als Vorteile sind dagegen zu vermerken, dass es eine Prozessüberwachung ermöglicht.
3. Managed Windows Services: Hosting ausserhalb von IIS und von allen Windows-Versionen unterstützt. Zudem ermöglicht dies auch Services über HTTPS.
4. Windows Process Activation Service (WAS): Ist eine neue Hosting Art welche jedoch die sogenannten WAS-Komponenten voraussetzt, die erst ab Windows Server 2008 und Windows Vista zur Verfügung stehen.

Das Framework ist für eine möglichst breite Verwendung gedacht und soll deshalb auch kleinen Entwicklungsteams mit wenig Ressourcen die Nutzung ermöglichen. Die Voraussetzung an die Infrastruktur sollen deshalb so niedrig wie möglich gehalten werden. In einer Diskussion mit Silvan Gehrig haben wir uns entschieden den Service als Managed Windows Service zu hosten. Durch diesen Schritt bleiben wir unabhängig von der Einrichtung eines IIS-Servers, können alle Windows-Versionen unterstützen und dem Kunden eine Installationsroutine anbieten, die das einfache Einrichten eines Servers erlaubt. Ein Umstellung auf ein anderes Hosting, z.B. über IIS, ist auch zu einem späteren Zeitpunkt noch möglich. Diese Entscheidung kann somit zu einem späteren Zeitpunkt den tatsächlichen Wünschen angepasst werden.

[4] Visual Studio, bzw. der C# Compiler prüfen die Bezüge innerhalb der XML Dokumentation auf ihre formale Korrektheit. So werden beispielsweise bei der Dokumentation einer Methode fehlerhafte Parameternamen als solches erkannt und gemeldet.

[5] Microsoft Dokumentation zum WCF-Hosting `http://msdn.microsoft.com/en-us/library/ms730158.aspx`

Filterkonfiguration mit XML

Die Konfiguration, welche Events auf welchen GUI-Elemente aufgezeichnet werden sollen, kann der Entwickler in einer XML-Konfigurationsdatei definieren. Dazu wird eine eigens kreierte DTD zur Validierung verwendet. Nachfolgend ist eine beispielhafte Konfiguration aufgeführt. Dies hat zum Vorteil, das die Konfiguration zum einen nicht direkt im Code geschehen mus, desweiteren ist die XML-Struktur sehr viel einfacher verständlich und kann allenfalls auch von einem (C# -)Laien verstanden werden.

```
1  <?xml version="1.0" encoding="utf-8" standalone="no" ?>
2  <!DOCTYPE filter SYSTEM "filters.dtd">
3  <!-- For the XML and the DTD go to the Properties and set Build Action to "Embedded
       Ressource" -->
4  <filter>
5    <container-criteria name="TrackingTestEnvironment.MainWindow">
6      <event name="SizeChanged" />
7    </container-criteria>
8    <container-criteria name="TrackingTestEnvironment.SecondaryWindow">
9      <control-criteria name="button3">
10       <event name="Click" />
11     </control-criteria>
12     <control-criteria name="textBox1">
13       <event name="TextChanged" />
14     </control-criteria>
15   </container-criteria>
16 </filter>
```

filters.xml

Konfiguration der Erfassung von EventArgs

Auch die Erfassung der EventArgs wird mittels XML-Tags gesteuert. Dabei kann angegeben werden, welche EventArgs (identifiziert durch ihren Namen) gespeichert werden sollen. Diese Konfiguration kann aber nur einmal für alle Events gemacht werden, und nicht pro Filter.

Client/Server Abgleich von Datensätzen in Lookup-Tabellen

Ein Teil der vorhandenen Tabellen sind sogenannte Lookup-Tabellen, ein Eintrag wird darin nur dann gemacht wenn dieser noch nicht vorhanden ist. Als Lookup Kriterium kann dabei ein Name oder eine Kombination von Attributen dienen. Beispiele für solche Lookup-Tabellen sind das CPU-Modell oder auch die GUI-Komponenten.
Natürlich kommt es vor das gewisse Einträge, z.B. CPU-Modelle, bei mehreren Clients vorkommen, beim Abgleich auf den Server soll deshalb darauf geachtet werden, dass der Eintrag auf dem Server trotzdem nur einmalig erstellt wird. Für dies gibt es nun zwei Varianten:

1. **Synchronisation (Abfragen und ersetzen der GUIDs)**
   Sobald der Client eine Verbindung zum Server hat soll er mit dem Server alle bisher unabgeglichenen Einträge synchronisieren. Dazu sendet der Client dem Server einen Eintrag zu und erhält als Antwort die GUID unter welcher

der Eintrag auf dem Server bekannt ist. Nun erstellt der Client den Eintrag mit der neuen GUID in der lokalen Datenbank, ersetzt alle Vorkomnisse der bisherig bekannten GUID mit jener die vom Server ankam und löscht den Eintrag mit der bisher genutzten GUID. Folgerungen:

- Server muss den Eintrag höchstens einmal pro Client abgleichen und die Guid zurücksenden.
- Der Client muss die Einträge, welche bis zur ersten Abfrage die rein lokale ID verwendet haben, aktualisieren.
- Auf den Lookup-Tables ist jeweils in einer Spalte festzuhalten, ob der Eintrag bereits mit dem Server abgeglichen wurde. Allenfalls macht diese Spalte in der Server-DB wenig bis keinen Sinn, jedoch rechtfertigt dies alleine nicht ein separates Datenmodell für Client/Server.
- Bevor Daten aus den Haupttabellen (Sessions/Occurrences) gesendet werden können müssen zuvor alle nicht synchronisierten Einträge aus Lookup-Tabellen abgeglichen werden. Ansonsten besteht die Gefahr, dass ein Fremdschlüssel auf einen Eintrag zeigt der auf dem Server (noch) nicht vorhanden ist.

2. **Reines senden der Werte**
   Die Lookup-Tabellen selbst werden nicht mit dem Server abgeglichen. Stattdessen sendet der Client zu jenen Einträgen welche auf eine Lookup-Tabelle zeigen nicht den Fremdschlüssel (die Guid) sondern den tatsächlichen Eintrag zu.
   Folgerungen:
   - Der Client kann jeweils direkt senden und muss sich nicht darum kümmern ob dem Server alle Einträge aus den Lookup-Tabellen bekannt sind.
   - In den Lookup-Tabellen ist keine zusätzliche Spalte notwendig um den Synchronisationsstand zu deklarieren.
   - Der Server hat für die Abfrage einen Zusatzaufwand, da er immer die Einträge in den Lookup-Tabellen prüfen/hinzufügen muss. So wären beispielsweise für jede GUIOccurence zwei zusätzliche Abfragen notwendig um die ID's der GUI-Komponente sowie des Event-Typs abzufragen.

Entscheidung: Wir haben uns für Variante 1 entschieden, da dies die Auslastung des Servers stark minimieren kann. Auch der Client muss den Abgleich nur einmalig pro Eintrag in den Lookup-Tables durchführen. Dies jeweils bei der ersten Serververbindung nach dem der Eintrag angelegt wurde.

DataFacade verwendet Generics anstelle Vererbung

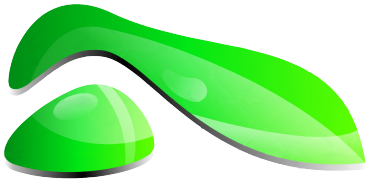
Die Klasse DataFacade benötigt ein Typ-Argument (Generic). Das Argument kann soweit eingeschränkt werden, dass es entweder EntityObject oder eine von EntityObject abgeleitete Klasse sein muss. Dies wurde so implementiert, dass der Eingabe- und Ausgabetyp des DataFacade effektiv derselbe sein kann, ohne auf Polymorphie zurückzugreifen, so fallen an vielen Stellen im Code Typecasts weg und die Typsicherheit kann damit gewährleistet werden.

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

INSTITUT
FÜR
SOFTWARE

User Tracking Framework für WPF Applikationen



Integration Manual

Studienarbeit HS2012

Studenten: Roger Landolt, Marco Tanner

Betreuer: Michael Gfeller, Silvan Gehrig

Betreuender Dozent: Hans Rudin

16. Installation

16.1. Grundlagen

UTF4WPF ist für die Integration in eine bestehende WFP-Applikation gedacht. Es sammelt Daten zu GUI-Events und kann diese an einen zentralen Server übermitteln. Wenn keine Internetverbindung besteht, kann der Client gesammelte Daten lokal zwischenspeichern und diese dann bei bestehender Verbindung nachsenden. Damit die Datenmenge auf dem Client nicht zu gross wird, werden maximal die Daten der letzten 7 Tage zwischengespeichert.

16.2. Prerequisites

Um UTF4WPF einsetzen zu können müssen folgende Voraussetzungen erfüllt werden:

Server

Es muss ein über das Netzwerk (bzw. Internet) erreichbarer Server eingerichtet werden, um die Daten der Clients zu sammeln. Der Netzwerkport kann grundsätzlich frei gewählt werden, es wird aber empfohlen den Standard-Port 443 (HTTPS) zu verwenden. Auf dem Server muss zudem das .NET Framework 4 (oder neuer) sowie ein Microsoft SQL-Server installiert sein.

Client

Die zu integrierende Applikation muss in Form von Source-Code vorliegen, damit die Integration durchgeführt werden kann.

16.3. Installation SSL-Zertifikat

Die Datenübertragung zum Server wird über eine mit SSL gesicherte Verbindung übertragen. Dazu ist ein SSL-Zertifikat notwendig. Für Testzwecke wird ein selbst signiertes Test-Zertifikat mitgeliefert, die nachfolgenden Schritte können aber mit jedem Zertifikat durchgeführt werden. Im Kapitel 17.8 dieses Dokuments findet sich eine Anleitung, wie Sie selbst ein inoffizielles, selber signiertes Zertifikat generieren können.
Achtung: Es wird strengstens empfohlen, mit einem eigenen Zertifikat zu arbeiten um die Datensicherheit der Übertragung zu gewährleisten

1. Starten Sie den Vorgang, indem Sie auf die pfx-Datei rechtsklicken und $Install\ PFX$ wählen.

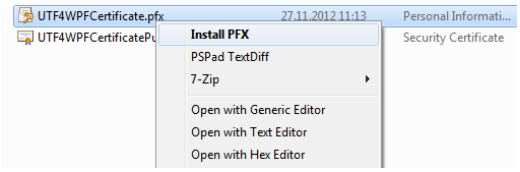


Abbildung 16.1.: Installation Zertifikat Schritt 1

2. Wählen Sie $Next$.

Abbildung 16.2.: Installation Zertifikat Schritt 2

3. Der Pfad wird durch die Auswahl der Datei zuvor automatisch korrekt eingetragen. Wählen Sie $Next$.

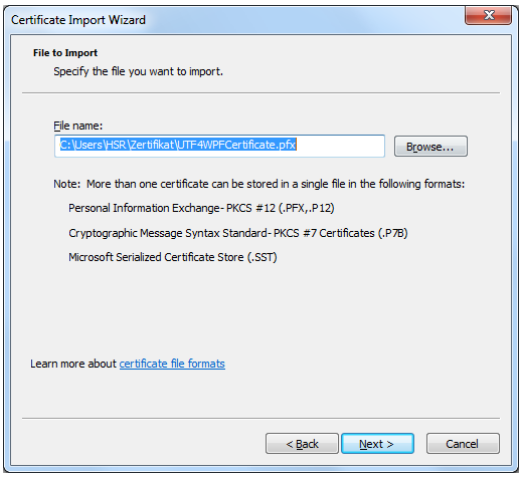


Abbildung 16.3.: Installation Zertifikat Schritt 3

4. Das Passwort für das Standardzertifikat lautet *utf4wpf*. Wählen Sie $Next$.

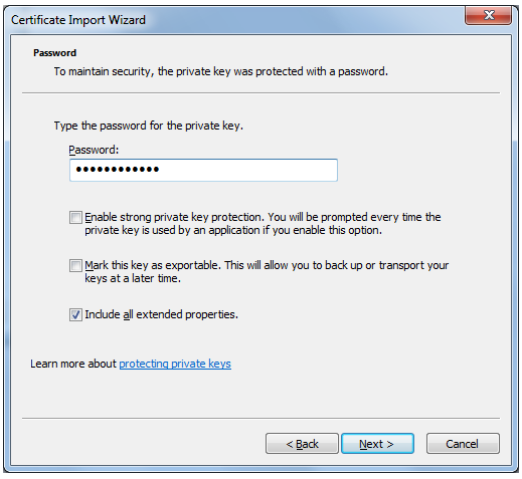


Abbildung 16.4.: Installation Zertifikat Schritt 4

5. Geben Sie den Certificate Store $Trusted\ People\backslash Local\ Computer$ an und wählen Sie $Next$.

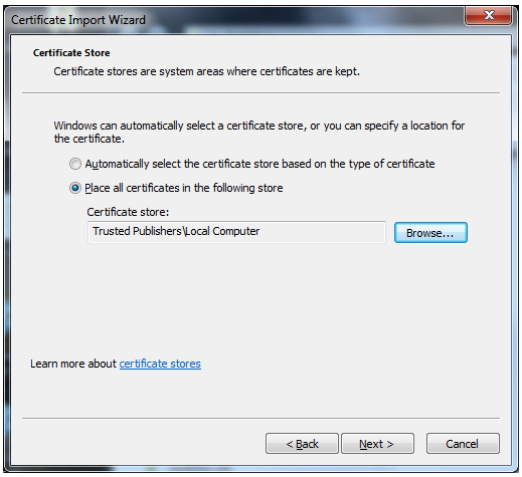


Abbildung 16.5.: Installation Zertifikat Schritt 5

6. Wählen Sie $Finish$.

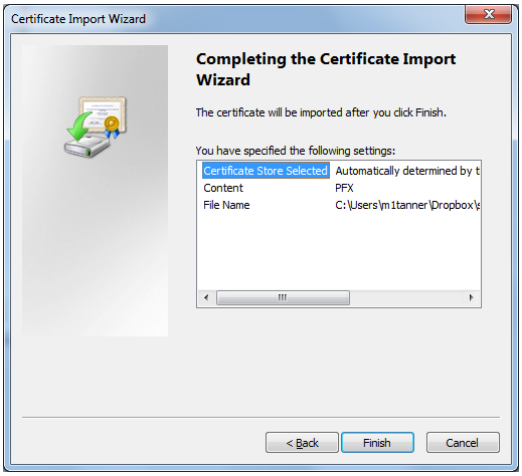


Abbildung 16.6.: Installation Zertifikat Schritt 6

7. Wählen Sie OK.

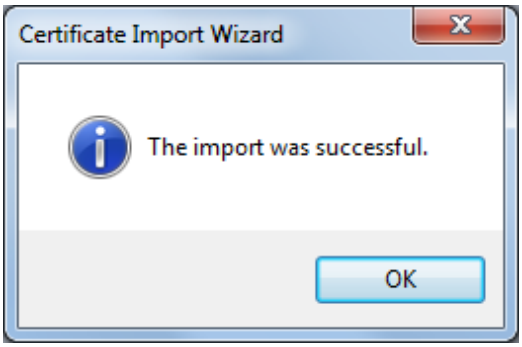


Abbildung 16.7.: Installation Zertifikat Schritt 7

8. Das Zertifikat muss nun noch an den Port 443 gebunden werden. Führen Sie dazu den folgenden Befehl auf der Kommandozeile ($cmd.exe$) aus, fügen Sie als $certhash$ den Thumbprint Ihres Zertifikats ein.
   `netsh http add sslcert ipport=0.0.0.0:443 certhash=YOUR-THUMBPRINT-HERE clientcertnegotiation=enable appid={6b3655c7-8a98-4822-bcf4-8cdc2c60e91e} certstore=trustedpublisher`

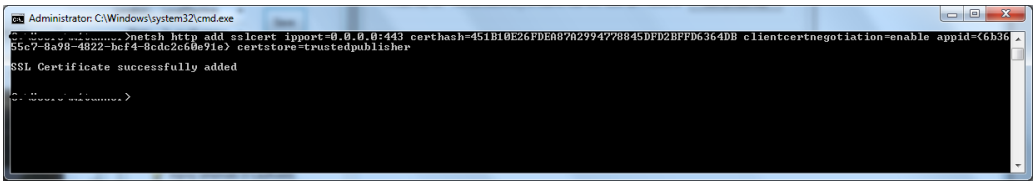


Abbildung 16.8.: Installation Zertifikat Schritt 8

16.4. Installation Server

1. Starten Sie die Datei $setup.exe$ mit Administratorenrechten

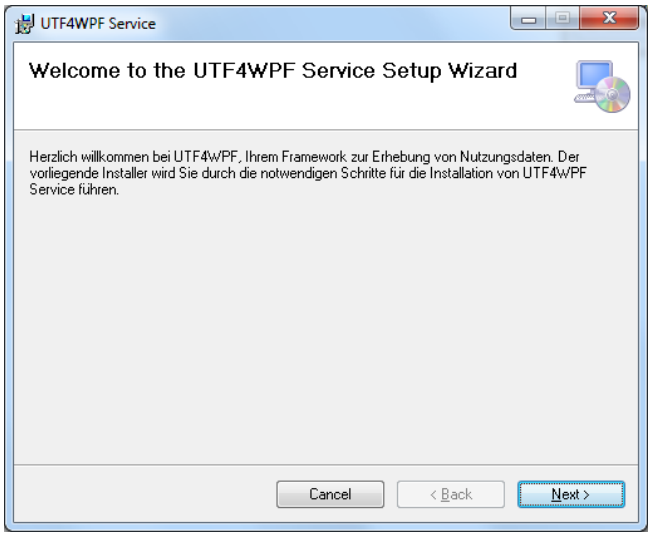


Abbildung 16.9.: Server Setup Schritt 1

2. Geben Sie den gewünschten Installationspfad an. Wählen Sie $Next$.

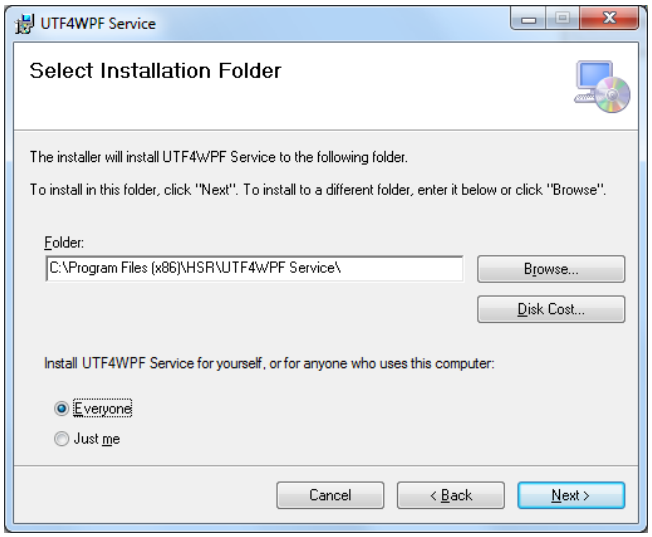


Abbildung 16.10.: Server Setup Schritt 2

3. Wählen Sie $Next$.

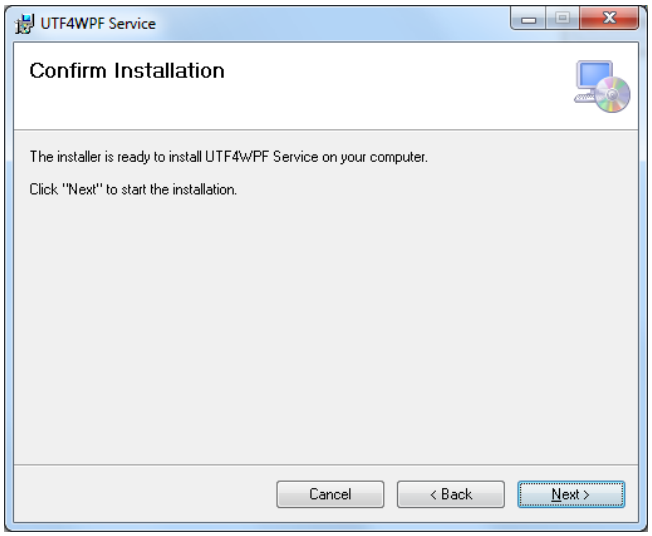


Abbildung 16.11.: Server Setup Schritt 3

4. Warten Sie, während die Installation durchgeführt wird.

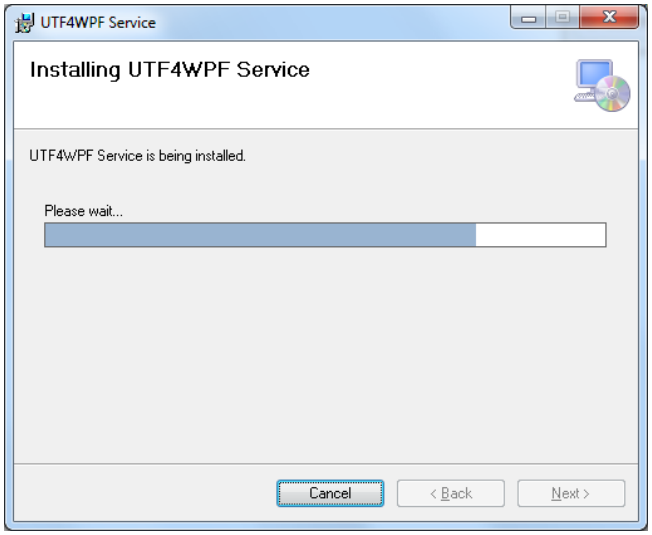


Abbildung 16.12.: Server Setup Schritt 4

5. Wählen Sie die SQL-Server Instanz, auf dem die UTF4WPF-Datenbank eingerichtet werden soll. Geben Sie wenn nötig einen gültigen Benutzernamen und Passwort ein. Wählen Sie OK.

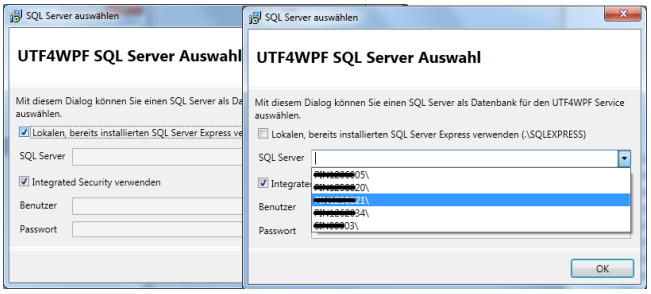


Abbildung 16.13.: Server Setup Schritt 5

6. Wählen Sie $Close$.

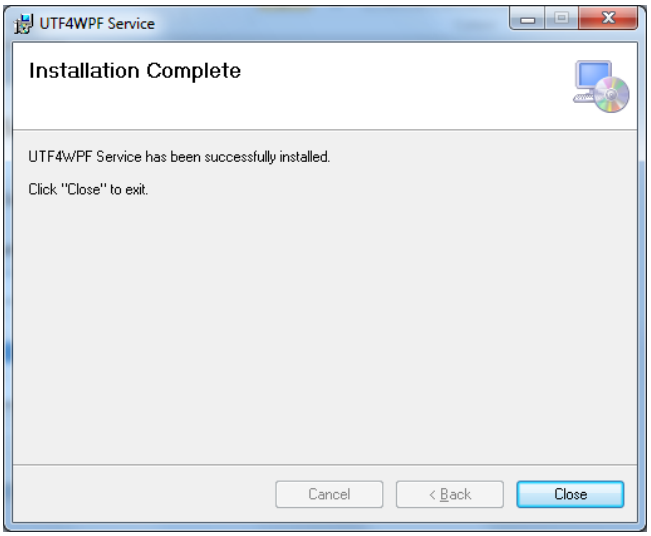


Abbildung 16.14.: Server Setup Schritt 6

7. Die Datenstruktur wird nun automatisch angelegt und der Dienst wird auf Port 443 registriert und gestartet.

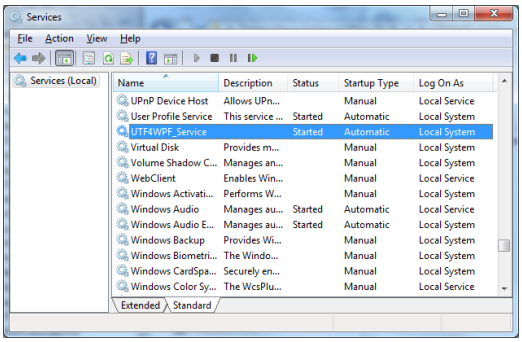


Abbildung 16.15.: Kontrolle der Installation 1

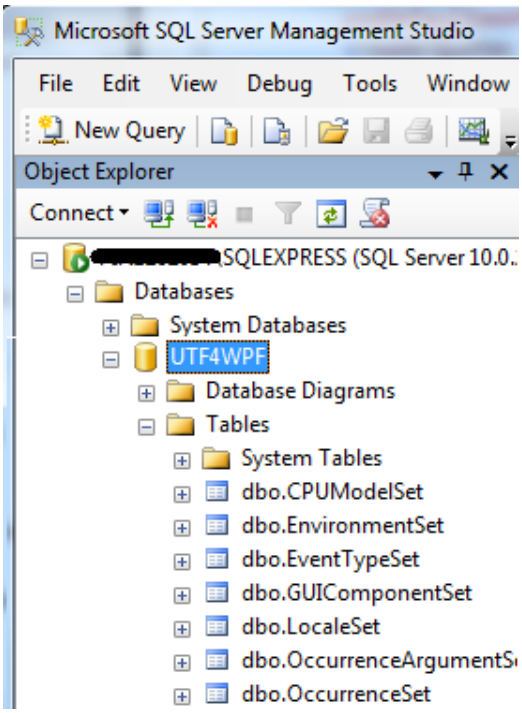


Abbildung 16.16.: Kontrolle der Installation 2

17. Integration in ein bestehendes Projekt

1. Referenz in Projekt
   Für die Integration muss eine Referenz auf das UTF4WPF.Client-Assembly im Projekt hinzugefügt werden. Dazu wird die UTF4WPF.Client.dll in einem belieben Ort gespeichert. Danach kann im Visual Studio via
   $Rechtsklick\, auf\, das\, Projekt \rightarrow Referenz\, hinzufügen\ldots \rightarrow Durchsuchen \rightarrow$
   $UTF4WPF.dll\ auswählen \rightarrow OK$ die DLL eingebunden werden.
2. App.config
   Die Datei App.config muss angelegt bzw. angepasst werden. Es muss ein Connection String für die lokale Client-Datenbank definiert werden. Dies muss **zwingend** ein lokaler Pfad sein. Es werden keine Netzwerkshares oder Ähnliches unterstützt. Wir empfehlen folgenden Pfad:
   $\%LocalAppData\% \setminus < HERSTELLER > \setminus < PRDOUKT > .sdf$.
   Achtung: Zurzeit gibt es keine dynamische Unterstützung für Umgebunsvariabeln ausser $\%LocalAppData\%$**.**

```
<configuration>
        <connectionStrings>
         <add
           name="UTF4WPFEntities"
           connectionString="Data Source=%LocalAppData%\UTF4WPF\client.sdf"
           providerName="System.Data.EntityCLient"
         />
        </connectionStrings>
</configuration>
```

Ein komplettes Beispiel der App Config ist auf Seite 85 zu finden.

3. Instanzierung des Trackers
   Als letztes muss eine Instanz der Klasse Tracker gestartet werden. Dazu wird empfehlen wir die Datei App.xaml.cs folgendermassen zu erweitern. Die Methode Start erwartet einen bool als Parameter, welcher anzeigt ob der Benutzer die Erlaubnis zur Sammlung seiner Tracking-Daten gegeben hat. Aus Datenschutzgründen muss diese Erlaubnis zuerst eingeholt werden. Die Art und Weise, wie die Erlaubnis eingeholt wird bleibt dem Entwickler überlassen.
   Wir empfehlen zudem, wie unten aufgezeigt, ein try-catch für die UTF4WPFException durchzuführen. Alle Exceptions, welche beim Start auftreten, werden einerseits im Windows Event Log (unter den Applikations- und Service-Logs) eingetragen und mit einer UTF4WPFException an den Aufrufer weitergegeben. Dabei enthält die InnerException die ursprüngliche Exception.

```
1  using UTF4WPF.Client;
2
3  namespace ProjectX
4  {
5
6    public partial class App : Application
7    {
8      protected override void OnStartup(StartupEventArgs e)
9      {
10       base.OnStartup(e);
11
12       bool allow = /*AskForUserPermission(); Implement such a method */
13       try
14       {
15         UTF4WPFTracker.Start(allow);
16       }
17       catch(UTF4WPFException utf4wpfException)
18       {
19         // What to do if we could not start, InnerException gives you the original
               Exception.
20       }
21     }
22   }
23 }
```

17.1. Einbinden des Zertifikats im zu trackenden Projekt

1. **Auf dem Server:** Öffnen Sie den lokalen Zertifikatmanager indem Sie im $Ausführen\ldots$-Dialog $certmgr.msc$ eingeben, und zu Ihrem Zertifikat navigieren. Öffnen Sie das Zertifikat mit einem Doppelklick.
2. Wechseln Sie zum Reiter $Details$ und Wählen Sie $Copy\ to\ file\ldots$

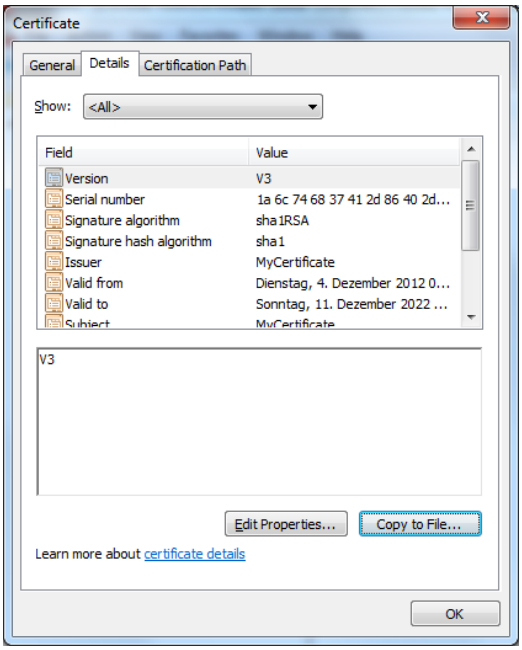


Abbildung 17.1.: Einbindung Zertifikat Schritt 1

3. Wählen Sie $Next$.

Abbildung 17.2.: Einbindung Zertifikat Schritt 2

4. Wählen Sie $Next$.

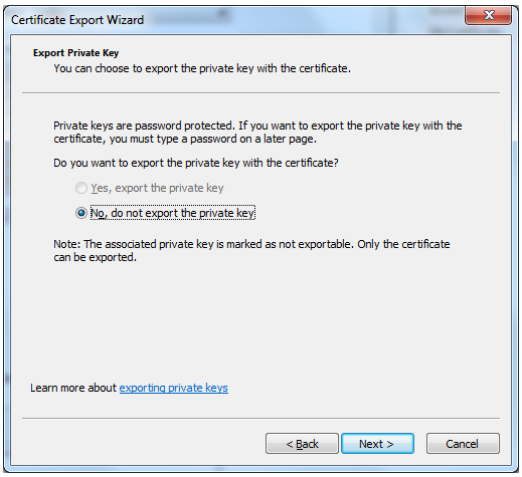


Abbildung 17.3.: Einbindung Zertifikat Schritt 3

5. Wählen Sie $Base-64\ encoded\ X.509$. Wählen Sie $Next$.

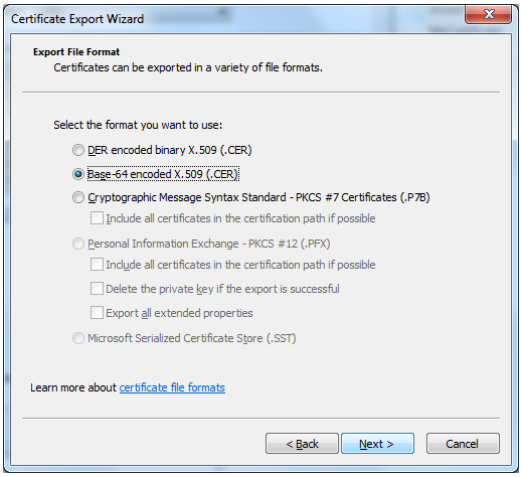


Abbildung 17.4.: Einbindung Zertifikat Schritt 4

6. Wählen Sie einen Dateinamen. Wählen Sie $Next$.

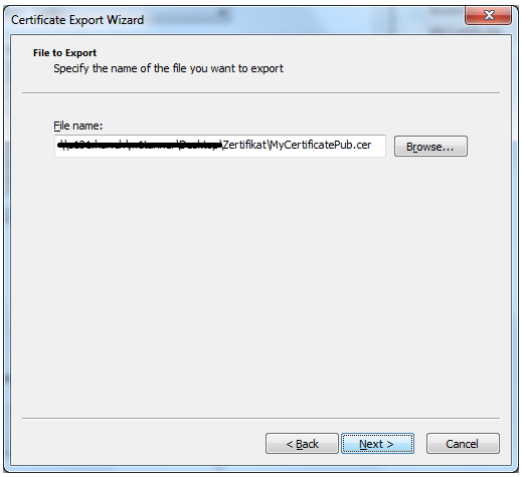


Abbildung 17.5.: Einbindung Zertifikat Schritt 5

7. Wählen Sie $Finish$.

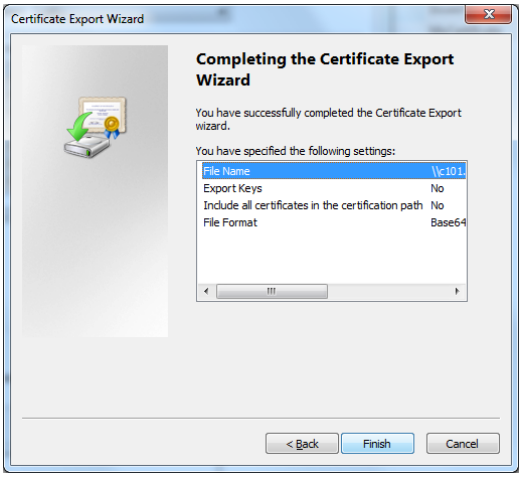


Abbildung 17.6.: Einbindung Zertifikat Schritt 6

8. Öffnen Sie die soeben gespeicherte Datei mit einem Texteditor und kopieren Sie den Inhalt wie auf dem Screenshot gezeigt.

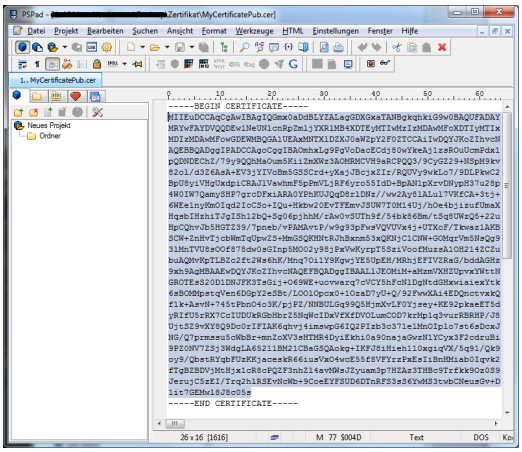


Abbildung 17.7.: Einbindung Zertifikat Schritt 7

9. Öffnen Sie die Datei *App.config* **Ihres** Projekts, in welchem Sie UTF4WPF nutzen möchten, und tragen Sie den kopierten Wert wie folgt in die XML-Struktur ein:

```
<system.ServiceModel>
  <client>
    <endpoint>
      <identity>
        <certificate encodedValue="MAeikjeasdjhiK3hlajkhsd3..." />
      </identity>
    </endpoint>
  </client>
</system.ServiceModel>
```

10. Tragen Sie zusätzlich folgenden Abschnitt ein:

```
<appSettings>
  <!-- Verwenden Sie als Value Ihren distinguished name: -->
  <add key="AllowedCertificateDistinguishedName" value="CN=MyCertificate"/>
  <!-- Verwenden Sie als Value Ihren Zertifikat-Thumbprint: -->
  <add key="AwaitedCertificateThumbprint" value="563BEDF3A..."/>
```

17.2. Filterkonfiguration

Sämtliche Filteroptionen werden in der Datei filters.xml festgelegt. Diese Datei muss in der aufrufenden Assembly als Embedded Ressource erstellt werden. Die zugehörige DTD kann auf Seite 86 zu Referenzzwecken eingesehen werden.
Es gibt drei verschiedene Sorten von Kriterium-Tags, sowie ein Event-Tag, die eine Auswahl bezüglich der zu erfassenden Events ermöglichen.

17.2.1. container-criteria

Mit diesem Kriterium kann man die Events eines bestimmten Containers (ein Window oder UserControl) abfangen. Der Container wird via name-Attribute spezifiziert.

Beispiel:

```
1 <container-criteria name="NAMESPACE.MainWindow">
2   <!-- events to capture here -->
3 </container-criteria>
```

17.2.2. controltype-criteria

Mit diesem Kriterium können alle GUI-Elemente eines bestimmten **Typs** (z.B. alle Dropdown-Menüs) erfasst werden. Die Auswahl der Elemente wird mithilfe deren Klasse bestimmt, die via type-Attribut definiert wird. Es können auch von den Standard-Controls abgeleitete Klassen angegeben werden. Jede Auswahl beinhaltet automatisch deren Subklassen. So werden im nachfolgenden Beispiel auch alle Instanzen von MyDerivedButton überwacht.
Eine Liste häufig verwendeter GUI-Komponenten findet sich auf Seite 84, die komplette Liste ist in der MSDN-Referenz[1] enthalten.
Dieses Tag kann direkt im *<filter />* oder innerhalb eines *container-criteria*'s verwendet werden. Je nachdem werden somit alle entsprechenden Controls, oder nur die auf dem entsprechenden Container (Window oder UserControl) abgefangen.

Beispiel:

```
1 <controltype-criteria type="System.Windows.Controls.Button">
2   <!-- events to capture here -->
3 </controltype-criteria>
```

[1] http://msdn.microsoft.com/en-us/library/ms590941.aspx

17.2.3. control-criteria

Mit diesem Kriterium kann ein **einziges** GUI-Element zur Überwachung ausgewählt werden. Dieses Tag kann nur innerhalb eines *container-criteria*'s vorkommen und das Control wird anhand des Name Attributs im XAML gesucht. Wenn kein Element mit dem angegebenen Namen gefunden werden kann, wird beim Start des Trackers eine entsprechende Fehlermeldung in der Datenbank vermerkt.

Beispiel:

```
<control-criteria name="CountrySelectBox">
  <!-- events to capture here -->
</control-criteria>
```

17.2.4. event

Dieses Tag, in Kombination mit den obigen Kriterien, ermöglicht die Auswahl, welche Events erfasst werden. Der Event-Typ wird via name-Attribut definiert. Ist ein Event auf einem GUI-Element nicht definiert, wird beim Start des Tracker eine entsprechende Fehlermeldung in der Datenbank vermerkt. Die komplette Liste der Events (und auf welchen Controls welche Events definiert sind) kann in der MSDN-Referenz[2] eingesehen werden. Es können beliebig viele Event-Tags pro Kriteriums-Tag definiert werden, dies ist auf *container-criteria*s, *controltype-criteria*s und *control-criteria*s möglich.

Beispiel:

```
<event name="Click" />
<event name="KeyPressed" />
```

17.2.5. event-all

Es ist möglich, **alle** Events eines GUI-Elements abzufangen. Dies kann mit dem *event-all*-Tag erreicht werden. Das Tag kann an allen Stellen platziert werden, wo auch event-Tags möglich sind, aber es kann pro Kriterium nur entweder mit event-Tags, oder mit dem event-all-Tag gearbeitet werden, beide im selben Kriterium zu platzieren ist nicht möglich.

[2] http://msdn.microsoft.com/en-us/library/ms754204.aspx

Achtung:

Das event-all-Tag kann unter Umständen sehr viele Events betreffen. Dies beeinträchtigt die Performance der Applikation und kann die Client- und die Serverdatenbank aufblähen. Es sollte nur in Ausnahmefällen verwendet werden.

Beispiel:

```
<event-all />
```

17.3. EventArgs

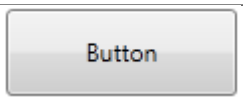
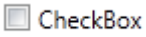
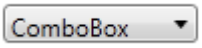
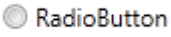
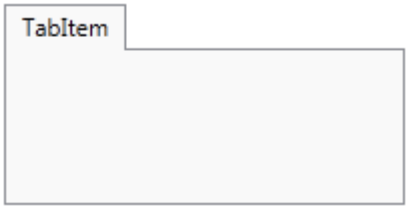
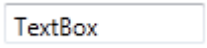
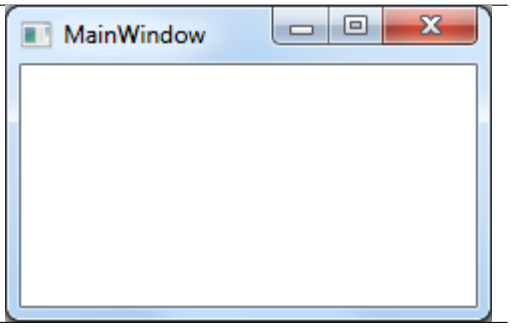
Jedes .NET-Event wir zusammen mit einer Instanz der Klasse EventArgs ausgelöst. Die Eigenschaften dieser EventArgs können bei Bedarf pro Event mitaufgezeichnet werden, in dem dessen Name angegeben wird. Diese Konfiguration gilt für ALLE aufgezeichneten Events. Daher wird das Tag direkt im root-Element platziert. Desweiteren ist sie optional, wenn kein *eventargs*-Tag vorhanden ist werden keine EventArgs aufgezeichnet.
Die komplette Liste aller EventArgs kann in der MSDN-Referenz[3] eingesehen werden.

Beispiel:

```
<eventargs>
  <arg name="Handled" />
  <arg name="Source" />
</eventargs>
```

[3] http://msdn.microsoft.com/en-us/library/system.eventargs.aspx

17.4. Liste oft verwendeter GUI-Komponenten

System.Windows.Controls.	Interessante Events	Beispiel
Button	Click, TouchDown	
CheckBox	Checked, Unchecked	
ComboBox	SelectionChanged, DropDownOpened, DropdownClosed	
RadioButton	Checked, Unchecked	
TabControl	SelectionChanged	
TextBox	TextInput, TextChanged	
System.Windows.	**Interessante Events**	**Beispiel**
Window	Activated, Closed, Drop, SizeChanged	

17.5. Beispiel App.Config

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <appSettings>
    <add key="AllowedCertificateDistinguishedName" value="[DISTINGUISHED_NAME_OF_CERTIFICATE]"/>
    <add key="AwaitedCertificateThumbprint" value="[THUMBPRINT_OF_CERTIFICATE]"/>
  </appSettings>

  <connectionStrings>
    <add name="UTF4WPFEntities_Client" providerName="System.Data.EntityClient"
     connectionString="Data Source=%LocalAppData%\[HERSTELLER]\[PRODUKT].sdf" />
  </connectionStrings>

  <system.serviceModel>
    <bindings>
      <basicHttpBinding>
        <binding name="BasicHttpBinding_IUTF4WPFService" closeTimeout="00:01:00"
         openTimeout="00:01:00" receiveTimeout="02:00:00" sendTimeout="02:00:00"
         bypassProxyOnLocal="false" hostNameComparisonMode="StrongWildcard"
         maxBufferPoolSize="524288" maxReceivedMessageSize="65536"
         messageEncoding="Text" textEncoding="utf-8" useDefaultWebProxy="true" allowCookies="false">
         <readerQuotas maxDepth="32" maxBytesPerRead="4096" maxNameTableCharCount="16384"
           maxStringContentLength="8192" maxArrayLength="16384" />
         <security mode="Transport">
           <transport clientCredentialType="None"/>
         </security>
        </binding>
      </basicHttpBinding>
    </bindings>

    <behaviors>
      <endpointBehaviors>
        <behavior name="ClientCertificateBehavior">
          <clientCredentials>
            <serviceCertificate>
              <authentication certificateValidationMode="PeerOrChainTrust" />
            </serviceCertificate>
          </clientCredentials>
        </behavior>
      </endpointBehaviors>
    </behaviors>

    <client>
      <endpoint address="https://[SERVERDOMAIN_OR_IP]:443/UTF4WPFService/"
          binding="basicHttpBinding" bindingConfiguration="BasicHttpBinding_IUTF4WPFService"
          contract="UTF4WPF.Common.IUTF4WPFService" name="UTF4WPFService" behaviorConfiguration="
              ClientCertificateBehavior">
        <identity>
          <certificate encodedValue="[BASE64_PUBLIC_KEY]" />
        </identity>
      </endpoint>
    </client>
  </system.serviceModel>
</configuration>
```

App.Config

17.6. Beispiel Filterkonfiguration

```
<?xml version="1.0" encoding="utf-8" standalone="no" ?>
<!DOCTYPE filter SYSTEM "filters.dtd">

<filter>

  <!-- Erfasse, wann das MainWindow in seiner Groesse veraendert wurde -->
  <container-criteria name="TrackingTestEnvironment.MainWindow">
    <event name="SizeChanged" />
  </container-criteria>

  <!--
       Erfasse im SecondaryWindow alle Klicks auf Button3, alle textuellen Änderungen in TextBox1
       sowie alle Doppeklicks im gesamten Fenster
  -->
  <container-criteria name="TrackingTestEnvironment.SecondaryWindow">
    <control-criteria name="button3">
      <event name="Click" />
    </control-criteria>
    <control-criteria name="textBox1">
      <event name="TextChanged" />
    </control-criteria>

    <event name="DoubleClick" />
  </container-criteria>
</filter>
```

example.xml

17.7. Konfigurations-DTD

Die Konfiguration wird gegen folgende DTD validiert:

```
<?xml version="1.0" encoding="utf-8" ?>
<!-- For the XML and the DTD go to the Properties and set Build Action to "Embedded Ressource" -->

<!ELEMENT filter ((controltype-criteria|container-criteria)*,eventargs?)>

<!ELEMENT eventargs (arg*)>

<!ELEMENT arg EMPTY>
<!ATTLIST arg name CDATA #REQUIRED>

<!ELEMENT container-criteria ((event+ | event-all)?, (controltype-criteria|control-criteria)*)>
<!ATTLIST container-criteria name CDATA #REQUIRED>

<!ELEMENT controltype-criteria (event+ | event-all)>
<!ATTLIST controltype-criteria type CDATA #REQUIRED>

<!ELEMENT control-criteria (event+ | event-all)>
<!ATTLIST control-criteria name CDATA #REQUIRED>

<!ELEMENT event EMPTY>
<!ATTLIST event name CDATA #REQUIRED>

<!ELEMENT event-all EMPTY>
```

filters.dtd

17.8. Eigenes SSL-Zertifikat erstellen

Die folgenden Schritte erklären, wie man ein selbst signiertes Zertifikat generiert und installiert.

1. Laden Sie sich das Tool self-cert von pluralsight herunter.[4] [5]
2. Starten Sie $SelfCert.exe$
3. Wählen Sie einen X.500 distinguished name aus. Wählen Sie als Store $TrustedPublisher$ aus.

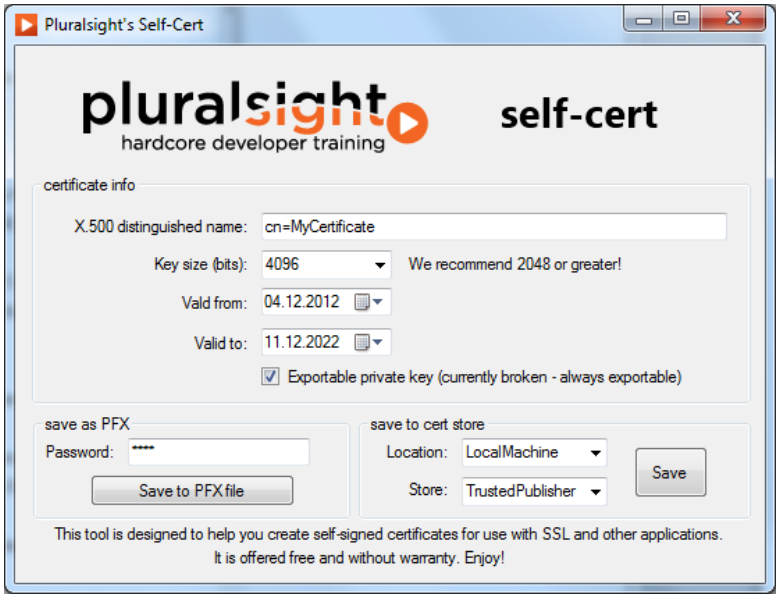


Abbildung 17.8.: Eigenes Zertifikat Schritt 1

4. Wählen Sie $Save$.
5. Warten Sie, während der Key generiert wird.

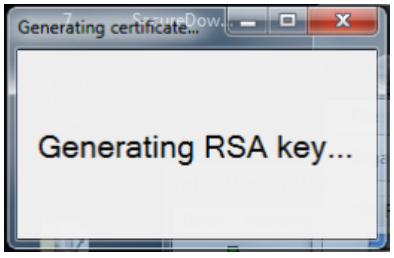


Abbildung 17.9.: Eigenes Zertifikat Schritt 2

[4] https://s3.amazonaws.com/pluralsight-free/keith-brown/samples/SelfCert.zip
[5] http://pluralsight.com

6. Die Zertifikatinformationen werden Ihnen angezeigt. Notieren Sie sich den angezeigten Thumbprint für die weitere Konfiguration.

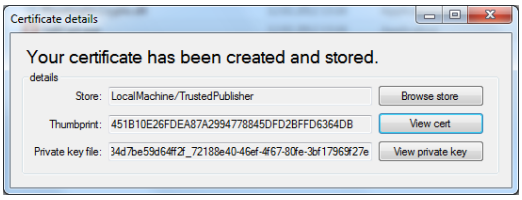


Abbildung 17.10.: Eigenes Zertifikat Schritt 3

17.8.1. Einbinden des Zertifikats im Server

7. Damit der Server das korrekte Zertifikat verwendet muss die Datei $UTF4WPF.WCFService.exe.config$ angepasst werden. Die Datei befindet sich im Installationsverzeichnis des Serverdienstes. Öffnen Sie die Datei mit einem Text-Editor und lokalisieren Sie folgenden Abschnitt:

```
1 <serviceCertificate findValue="UTF4WPFCertificate"
2   storeLocation="LocalMachine" storeName="TrustedPublisher"
3   x509FindType="FindBySubjectName"
4 />
```

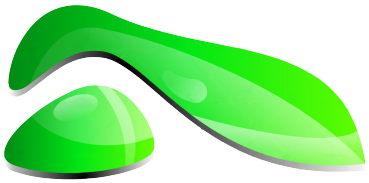
Tragen Sie den von Ihnen gewählten X.500 distinguished name als $findValue$ ein.

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

INSTITUT
FÜR
SOFTWARE

User Tracking Framework für WPF Applikationen



Testspezifikation

Studienarbeit HS2012

Studenten: Roger Landolt, Marco Tanner

Betreuer: Michael Gfeller, Silvan Gehrig

Betreuender Dozent: Hans Rudin

18. Allgemeines

Die Tracking Komponente ist stark mit dem System verknüpft und hilft uns Daten vom System bzw. der laufenden Applikation auszulesen.
Das UTF4WPF.Client Assembly soll die Komplexität so gut wie möglich vor dem Frameworknutzer verbergen. Deshalb wurde die Sichtbarkeit von aussen (public visibility) lediglich auf eine Klassse und deren vier Public-Funktionen ansprechbar. Die restlichen Klassen/Methoden sind sind mit einer private oder internal (nur im Assembly) Sichtbarkeit versehen.
Eine Integration der Tests direkt in das Assembly würde zudem zusätzliche Abhängigkeiten und eine Vermischung von produktivem und Test-Code mit sich bringen.
Wir haben uns deshalb entschieden, dass das UTF4WPF.Client Assembly mithilfe manuell durchgeführter Testfälle überprüft werden soll. Im Folgenden werden dazu die Testfälle mit deren Vorgehensweise und dem erwarteten Resultat aufgelistet.

18.1. Vorbereitung

Es wird mit dem Projekt “TrackingTestEnvironment” gearbeitet. Die Angaben in Klammern hinter den Testfällen bezeichnen die zugehörige Anforderung aus dem Anforderungsdokument. Vor dem Testlauf müssen folgende Vorbereitungsarbeiten durchgeführt werden:

1. `%localappdata%\UTF4WPF\client.sdf` löschen. Damit wird die Datenbank in der neuesten Schemastruktur und ohne bestehende Einträge erstellt.
2. Die Konfigurationsdatei `filters.xml` des TrackingTestEnvironment Projektes ersetzen durch die Datei im Anhang A.
3. In der `App.config` des TrackingTestEnvironment Projektes prüfen, dass beim endpoint-Tag die Adresse `https://localhost:443/UTF4WPFService/` angegeben wurde.
4. In der `App.config` des UTF4WPF.WCFService Projektes prüfen, dass beim baseAddresses-Tag eine baseAddress `https://localhost:443/UTF4WPFService/` hinzugefügt wurde.
5. Bei Testfällen wo nichts anderes angegeben wurde sollen die Projekte TrackingTestEnvironment sowie UTF4WPF.WCFService als Startup-Projekte angegeben werden.

18.2. Hinweis zu Timestamps

Alle Timestamps werden gemäss UTC-Zeit abgelegt. Gegenüber Schweizer Zeit ist dies somit minus eine Stunde. (z.B. 10:05 eingetragen anstelle von 11:05 schweizerischer Zeit)

19. Manuelle Testfälle

19.1. Zentrale Sammlung der Daten (M4)

Vorgehensweise:

1. Die Testapplikation starten
2. Die verbleibenden Fenster der Testapplikation schliessen

Erwartetes Resultat:

1. In der Tabelle SessionSet der Server-Datenbank soll ein neuer Eintrag mit dem aktuellen Zeitstempel als “Start”-Wert entstanden sein.
2. In der Tabelle EnvironmentSet soll ein Eintrag entstanden sein, dessen “Id” mit dem “EnvironmentID”-Wert der neuen Session übereinstimmt.
3. In der Tabelle ProgramSet soll ein Eintrag vorhanden sein, dessen “Id” mit dem “ProgramID”-Wert der neuen Session übereinstimmt und dessen Name “TrackingTestEnvironment.vshost.exe” lautet.

Testlauf am 10. Dezember 2012

Die oben genannten Kriterien wurden erfolgreich erfüllt.

Testlauf am 15. Dezember 2012

Die oben genannten Kriterien wurden erfolgreich erfüllt.

19.2. Sammeln von Events (M1)

Vorgehensweise:

1. Die Testapplikation starten
2. Auf den Button “Neues Fenster öffnen” klicken
3. Die CheckBox “CheckBox” zwei mal anklicken (“Check” und wieder “Uncheck”)
4. Auf den Button “Button” klicken
5. In das Textfeld “TextFeld” den Text “hallo” eintippen
6. Die verbleibenden Fenster der Testapplikation schliessen

Erwartetes Resultat:

In der Server-Datenbank müssen folgende Kriterien bestätigt werden:

- Es gibt eine neue Session (Tabelle SessionSet) mit korrekter “Start”- und “End”-Zeit, identisch mit der Zeit, während der die Testapplikation aktiv war
- Innerhalb der GUIOccurrences, bei denen die “SessionID” mit der “ID” der neuen Session übereinstimmt existieren unter Anderem:
  - Ein Event vom Typ “Checked” auf dem Control “CheckBox”
  - Ein Event vom Typ “Unchecked” auf dem Control “CheckBox”
  - Ein Event vom Typ “Click” auf dem Control “OpenOtherWindow”
  - Zwei Events vom Typ “Click” auf dem Control “button3” (einmal durch Controltype-criteria zu M1.2 und einmal zu control-criteria zu M1.3 mit event-all)
  - Fünf Events vom Typ “KeyDown” auf dem Control “textBox1”

  Dies ist am einfachsten mit der Query im Anhang B zu prüfen.

Testlauf am 10. Dezember 2012

Die oben genannten Kriterien wurden erfolgreich erfüllt.

Testlauf am 15. Dezember 2012

Die oben genannten Kriterien wurden erfolgreich erfüllt.

19.3. Sammeln von Ausnahmen und Fehlern (M2)

Vorgehensweise:

1. Die Testapplikation starten
2. Auf den Button “Neues Fenster öffnen” klicken
3. Auf den Button “Exception Thrower” klicken
4. Das Fenster mit der Stacktrace schliessen
5. Die verbleibenden Fenster der Testapplikation schliesen

Erwartetes Resultat:

In der Tabelle ExceptionalOccurrenceSet der lokalen Datenbank sollen drei neue Einträge entstanden sein. Dies aufgrund einer Verschachtelung von drei Exceptions welche absichtlich geworfen wurden. Zwei der Meldungen haben eine InnerExceptionID angegeben, wovon eine (die TestException) zudem den zuvor angezeigten Stacktrace beinhaltet.

Testlauf am 10. Dezember 2012

Die Einträge wurden wie erwartet vorgefunden.

Testlauf am 15. Dezember 2012

Die Einträge wurden wie erwartet vorgefunden.

19.4. Die Datensammlung ist auch bei fehlender Netzverbindung möglich (M3)

Vorgehensweise:

1. Die Client-Datenbank-Datei löschen (Siehe Vorbereitung)
2. Lediglich das Projekt TrackingTestEnvironment als Startup-Projekt festlegen
3. Die Testapplikation starten
4. Auf den Button “Neues Fenster öffnen” klicken
5. Die verbleibenden Fenster der Testapplikation schliessen
6. Auch das Projekt UTF4WPF.WCFService wieder zu den Startup-Projekten hinzufügen
7. Die Test-Applikation starten
8. Die verbleibenden Fenster der Testapplikation schliessen

Erwartetes Resultat:

In der Tabelle SessionSet der Server-Datenbank müssen zwei neue Einträge mit dem jeweils korrekten Zeitstempel angelegt worden sein. Auch die erste Session, welche während des Netzunterbruchs stattfand, muss auf dem Server sichtbar sein.

Testlauf am 10. Dezember 2012

Beim zweiten Start wurden beide Sessions erfolgreich an den Server übermittelt.

Testlauf am 15. Dezember 2012

Beim zweiten Start wurden beide Sessions erfolgreich an den Server übermittelt.

19.5. Entwicklermeldungen(K1)

Vorgehensweise:

1. In der Datei “App.xaml.cs” in der Methode OnStartup die folgende Zeile anfügen:
   UTF4WPFTracker.LogMessage("TEST MESSAGE");
2. Die Testapplikation kompilieren
3. Die Test-Applikation starten
4. Die verbleibenden Fenster der Testapplikation schliessen
5. Die eingefügte Zeile im Quellcode wieder entfernen

Erwartetes Resultat:

In der Tabelle CustomOccurrence (Tabelle OccurrenceSet_CustomOccurrenceSet) soll ein neuer Eintrag mit der Message “TEST MESSAGE” und dem korrekten Zeitstempel angelegt worden sein.

Testlauf am 10. Dezember 2012

Fehlermeldung beim Aufruf von *UTF4WPFTracker.LogMessage("TEST MESSAGE");* => Speicherfehler wegen fehlender ID der Session.

Testlauf am 15. Dezember 2012

Die Meldung wird problemlos eingetragen

19.6. Sammlung von Infos zum Client-Gerät (K2)

Vorgehensweise:

1. Die Test-Applikation starten
2. Die verbleibenden Fenster der Testapplikation schliessen

Erwartetes Resultat:

Die SQL-Abfrage im Anhang C auf der Server-Datenbank ausführen. Bei der neuesten Session Kontrollieren, ob die ausgewiesenen Informationen zu der Hardware etc. mit den effektiven Werten des Clients übereinstimmen.

Testlauf am 10. Dezember 2012

Die Einträge stimmen mit dem Testsystem überein.

Testlauf am 15. Dezember 2012

Die Einträge stimmen mit dem Testsystem überein.

20. Anhang Testspezifikation

20.1. Konfiguration

```
<?xml version="1.0" encoding="utf-8" standalone="no" ?>
<!DOCTYPE filter SYSTEM "filters.dtd">
<!-- For the XML and the DTD go to the Properties
     and set Build Action to "Embedded Ressource" -->
<filter>
  <!-- to satisfy requirement M1.1 -->
  <controltype-criteria type="System.Windows.Controls.CheckBox">
    <event-all />
  </controltype-criteria>

  <!-- to satisfy requirement M1.2 -->
  <controltype-criteria type="System.Windows.Controls.Button">
    <event name="Click" />
  </controltype-criteria>

  <container-criteria name="TrackingTestEnvironment.SecondaryWindow">
    <!-- to satisfy requirement M1.3-->
    <control-criteria name="button3">
      <event-all />
    </control-criteria>
    <!-- to satisfy requirement M1.4 -->
    <control-criteria name="textBox1">
      <event-all />
    </control-criteria>
  </container-criteria>
</filter>
```

testfilters.xml

20.2. Event-Query

```
1  SELECT      TOP (100) PERCENT dbo.EventTypeSet.Name AS Event, dbo.GUIComponentSet.
       ControlName AS Control, dbo.OccurrenceSet.Timestamp AS At,
2                            dbo.SessionSet.Id AS Session
3  FROM           dbo.EventTypeSet INNER JOIN
4                            dbo.OccurrenceSet_GUIOccurrence ON dbo.EventTypeSet.Id = dbo.
                                 OccurrenceSet_GUIOccurrence.EventTypeID INNER JOIN
5                            dbo.OccurrenceSet ON dbo.OccurrenceSet_GUIOccurrence.Id = dbo.
                                 OccurrenceSet.Id INNER JOIN
6                            dbo.GUIComponentSet ON dbo.OccurrenceSet_GUIOccurrence.
                                 GUIComponentID = dbo.GUIComponentSet.Id INNER JOIN
7                            dbo.ProgramSet ON dbo.GUIComponentSet.ProgramID = dbo.ProgramSet.
                                 Id INNER JOIN
8                            dbo.SessionSet ON dbo.OccurrenceSet.SessionID = dbo.SessionSet.Id
9  WHERE       (dbo.ProgramSet.Name = N'TrackingTestEnvironment.vshost.exe')
10 ORDER BY At DESC
```

eventquery.sql

20.3. Environment-Query

```
1  SELECT      TOP (1) dbo.CPUModelSet.Name AS CPUModel, dbo.SessionSet.Start, dbo.
       EnvironmentSet.MemorySize, dbo.EnvironmentSet.Frequency, dbo.EnvironmentSet.Cores,
2                            dbo.OperatingSystemSet.Name AS OS, dbo.LocaleSet.KeyboardLayout,
                                 dbo.LocaleSet.Name AS Locale, dbo.PointingDeviceSet.Name AS
                                 PointingDevice,
3                            dbo.PointingDeviceSet.Type
4  FROM           dbo.CPUModelSet INNER JOIN
5                            dbo.EnvironmentSet ON dbo.CPUModelSet.Id = dbo.EnvironmentSet.
                                 CPUModelID INNER JOIN
6                            dbo.LocaleSet ON dbo.EnvironmentSet.LocaleID = dbo.LocaleSet.Id
                                 INNER JOIN
7                            dbo.OperatingSystemSet ON dbo.EnvironmentSet.OperatingSystemID =
                                 dbo.OperatingSystemSet.Id INNER JOIN
8                            dbo.PointingDeviceSet ON dbo.EnvironmentSet.Id = dbo.
                                 PointingDeviceSet.EnvironmentID INNER JOIN
9                            dbo.SessionSet ON dbo.EnvironmentSet.Id = dbo.SessionSet.
                                 EnvironmentID
10 ORDER BY dbo.SessionSet.Start DESC
```

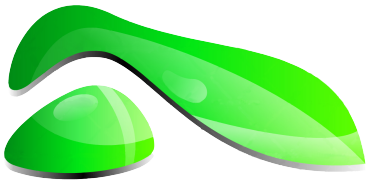
environment.sql

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

INSTITUT
FÜR
SOFTWARE

User Tracking Framework für WPF Applikationen



Projektplan

Studienarbeit HS2012

Studenten: Roger Landolt, Marco Tanner

Betreuer: Michael Gfeller, Silvan Gehrig

Betreuender Dozent: Hans Rudin

21. Einführung

21.1. Zweck

In diesem Dokument werden die Details zur Studienarbeit “User Tracking Framework für WPF Applikationen” dargelegt sowie die Projektmeilensteine definiert.

21.2. Aufgabenstellung

Die Semesterarbeit “User Tracking Framework für WPF Applikationen” wurde initiiert um ein Basis-Framework zu schaffen, welches die Messung der Art und Häufigkeit der Nutzung von Programm-Features erlaubt. So werden, mit Erlaubnis des Users, “unpersonalisierte” Daten gesammelt und dies auf einen zentralen Server in der Cloud übertragen. Bei fehlender Verbindung zum Server können die Daten lokal zwischen gespeichert und zu einem späteren Zeitpunkt in die Cloud übertragen werden. Auf Seite des Herstellers lassen sich somit Muster in der Benutzung der Anwendung, also beispielsweise Optimierungspotential in der Usability, erkennen. Ein weiteres Merkmal des Frameworks stellt die Behandlung (Logging) von Ausnahmen und Programm-Fehlverhalten dar. Im Rahmen des Versendens von Fehlerinformationen soll es möglich sein, eine erneute (separate) Erlaubnis vom Benutzer zum Datenversand zu erhalten.

Die komplette Aufgabenstellung findet sich im Anhang (Seite 109).

21.3. Abgabe

Der späteste Abgabetermin ist der 21.12.2012 um 17:00 Uhr.

22. Projektorganisation

22.1. Team

Das Projekt wird von Roger Landolt und Marco Tanner bearbeitet. Die Projektleitung wird im Team abgewickelt. Da Roger verhältnismassig viele Erfahrungen mit C# gemacht hat wird er sich schwerpunktmässig auf die Details der Entwicklung konzentrieren während Marco Tanner für die Struktur und Organisation der Dokumentation sowie die Feinheiten im LaTeX-Textsatz zuständig sein wird.

22.2. Externe Kontakte

Der Auftraggeber des Projekts ist das HSR Institut für Software. Unsere Kontakte und Projektbetreuer sind Michael Gfeller und Silvan Gehrig, beide vom IFS.
Hans Rudin, ebenfalls vom IFS, wird während der Projektdauer als betreuender Dozent fungieren.

22.3. Besprechungen

Als wöchentlicher Besprechungstermin wurde der Mittwoch von 13:10-14:50 Uhr festgelegt. Der Termin kann je nach Projektstatus nach Rücksprache in der Woche vor dem Termin mit den Betreuern geändert (oder ganz abgesagt) werden.

22.4. Arbeitsumgebung

22.4.1. Tools

Folgende Werkzeuge finden Einsatz in diesem Projekt:

- **MS Visual Studio 2010**
  Wird als IDE verwendet. Als zusätzliches Werkzeug wird ReSharper verwendet, um die Einhaltung der Coding Guidelines zu prüfen.
- **ReSharper für Visual Studio**
  Bietet bessere Korrekturmöglichkeiten für Coding-Guidelines sowie erweiterte Refactoring-Möglichkeiten
- **MS Team Foundation Server**
  Wird verwendet für Source Control und Versionierung.

- **Dropbox**
  Für die Ablage der Dokumentation
- **LaTeX**
  Wird für das Setzen der Dokumentation verwendet.
- **astah***
  Für das Erstellen von UML-Diagrammen.

22.4.2. Coding Style

Es wird nach den Vorgaben der “C# Coding Standards for .NET” von Lance Hunt[1] programmiert, die Konventionen wurden entsprechend mithilfe vom ReSharper umgesetzt:

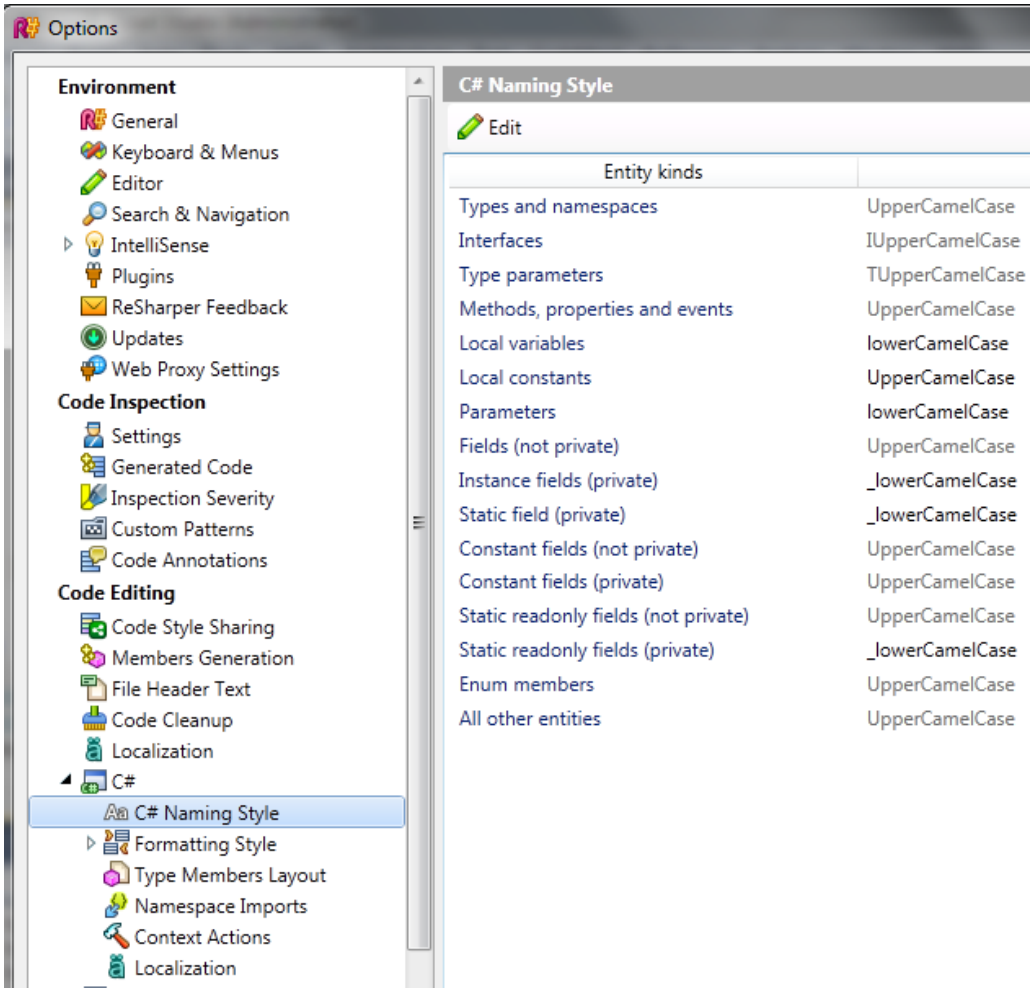


Abbildung 22.1.: Namenskonventionen ReSharper

[1] http://www.lance-hunt.net

23. Meilensteine

Die folgenden Meilensteine sollen den Überblick über den Projektverlauf veranschaulichen und sicherstellen das der korrekte Ablauf gewährleistet bleibt, die Meilensteine und der Zeitplan (Seite 112) sind dabei im Wesentlichen an RUP angelehnt.

23.1. Anforderungen & Analyse

Termin: 5. Oktober 2012
Zu diesem Zeitpunkt ist der Projektplan sowie die Analyse der Aufgabenstellung grundlegend erfasst. Eventuelle Änderungen können während des weiteren Projektverlaufs eingebracht werden. Zudem wurden bereits Konkurrenzprodukte auf ihre Features hin untersucht und ein Requirements-Dokument für das Projekt erstellt.

23.2. Abschluss Designphase

Termin: 26. Oktober 2012
Bei diesem Meilenstein soll das Softwarearchitekturdokument inklusive Domain-Modell, verwendeter Technologien, Deployment-Diagramm und Package-Struktur auf einem Stand sein, damit in die Implementierungsphase übergegangen werden kann.

23.3. Erster (lokaler) Prototyp

Termin: 9. November 2012
Der erste Prototyp beinhaltet die User- und Fehlertracking Funktionalität und kann die gesammelten Daten lokal abspeichern.

23.4. Erster Prototyp Server-Synchronisation

Termin: 21. November 2012
Eine grundlegende Synchronisation der Client-Daten mit dem Server über eine ungesicherte Verbindung funktioniert und kann den Betreuern vorgeführt werden.

23.5. Code-Freeze

Termin: 10. Dezember 2012
Der Programmcode implementiert alle geplanten Features. Es werden nur noch Fehler behoben und/oder Refactorings durchgeführt, aber keine Funktionalität mehr hinzugefügt.

23.6. Abgabe

Termin: 21. Dezember 2012 17:00
Das Programm und die Dokumentation sind bereinigt und abgabefertig. Das A0-Poster wurde bereits zuvor an die Betreuer gesandt. Die Abgabemedien wurden gemäss den Vorgaben der Hochschule gedruckt bzw. gebrannt und den jeweiligen Empfängern zugestellt.

24. Persönliche Berichte

24.1. Roger Landolt

Als wir im Sommer die Studienarbeit “ScrumTable: Redmine Data Driver” zugeteilt bekamen war ich sehr froh darüber. Die Arbeit versprach viel neues zu bringen und mit bereits bekanntem zu verbinden. So kannte ich z.B. C# und auch Redmine bereits aus dem Software-Engineering 2 Projekt, Ruby hingegen war für mich komplett Neuland.

In der ersten Semesterwoche hiess es dann: Abbruch! Die Studienarbeit sollte so nicht durchgeführt werden. Zwar fand sich, zusammen mit unseren Betreuern, bald eine neue Arbeit im Bereich .NET, bereits getätigte Überlegungen und der gesammelte Elan verpufften jedoch kurzzeitig. Nachdem wir dann, mit zirka einer Woche Verspätung, die offizielle Aufgabenstellung erhalten haben, kam die Lust aber bald zurück und die Ideen begannen sich aufs Neue auszuprägen.

Die C#-Vorkenntnisse (bzw. von .NET im Allgemeinen) von Marco und mir unterschieden sich stark. So drängte sich schon bald eine Aufteilung auf, bei welcher Marcos Haupttätigkeiten in den Bereich LaTeX fielen, die meinigen jedoch im C# Bereich lagen.

Wir haben deshalb versucht, möglichst oft über den “Gartenzaun” auf die andere Seite zu schauen und dabei soviel mitzunehmen wie möglich. So konnte Marco beispielsweise das Datenmodel im Entity Framework umsetzen und ich habe sicherlich Grundkenntnisse in der Dokumentation mit LaTeX erhalten.

Im Rückblick gibt es an gewissen Stellen natürlich noch Verbesserungspotential bzw. einzelne Punkte welche wir, gegenüber dem Erstentwurf, nachgebessert haben. Ich bin jedoch sehr zufrieden mit dem erzielten Ergebnis und der Tatsache dass wir die Kriterien erfüllen konnten.

Ich konnte in diversen Bereichen neues mitnehmen, sei es bei der HTTPS-Verbindung mit Zertifikaten, dem .NET-Eventhandling oder auch der Synchronisation von Datenbeständen. Ich kann zudem einige Erfahrung im Bereich Projektmanagement und Zeitplanung mit in die Bachelorarbeit mitnehmen.

An dieser Stelle möchte ich mich noch bei unseren Betreuern für die Unterstützung bedanken.

24.2. Marco Tanner

Ursprünglich war als Thema für diese Studienarbeit das Thema “Redmine Treiber für ScrumTable” angedacht. Das Thema hatt mich insbesondere deshalb interessiert, weil die Arbeit einer “Hintergrundapplikation”, also ohne direkte interaktion mit dem Endbenutzer, aber auch der Einsatz einer für mich neuen und modernen Sprache, Ruby, geplant war.

Ich war am Anfang deshalb recht skeptisch als es hiess, die SA werde nicht wie geplant durchgeführt und uns stattdessen als alternatives Projekt angeboten wurde. Meine C#-Kenntnisse waren zu Beginn der Arbeit so gut wie nicht vorhanden, und ich hätte mir ein Projekt erhofft, dass weniger stark von Microsoft-Technologien abhängt.

Im Nachinhein bin ich aber sowohl mit dem Thema als auch mit dem Endergebnis der Arbeit sehr zufrieden. Ich konnte viel von Roger's umfassenden Kentnissen im .NET-Bereich lernen und habe Einblicke in C# erhalten die mir ansonsten verwehrt geblieben wären.

Da ich technisch gesehen relativ stark von Roger abhängig war habe ich mich besonders stark in der Dokumentation des Projekts eingesetzt. Es war für mich eine neue Erfahrung, nicht von Anfang an alles Aspekte eines Projektes und deren Umsetzung zu verstehen sondern erst mit Erklärungen gewisse Details zu verstehen.

Umso mehr hat es mich daher gefreut, meine LaTeX-Kenntnisse in das Projekt einzubringen. Für mich ist die Arbeit mit dem Textsatzssytem eine angenehme Art zu dokumentieren, da sowohl meine Freude an freier Software (Vim, LaTeX, GNU/Linux allgemein,...) als auch der Programmierer in mir auf seine Rechnung kommt.

Ich konnte meine Kenntnisse sowohl diesbezüglich als auch in Sachen Projektmanagement und Zeitplanung während der SA erweitern und bin nun bestens gerüstet für die kommende Bachelorarbeit im nächsten Semester.

Teil VII.

Appendix

Inhaltsverzeichnis

Aufgabenstellung Studienarbeit
für Herrn Roger Landolt und Herrn Marco Tanner

User Tracking Framework für WPF Applikationen

1. Betreuer

Diese Studienarbeit wird für das Institut für Software durchgeführt.

Auftraggeber (Entwickler des ScrumTable):

- Silvan Gehrig, Institut für Software sgehrig@hsr.ch
- Michael Gfeller, Institut für Software, mgfeller@hsr.ch

Betreuer:

- Prof. Hans Rudin, Institut für Software hrudin@hsr.ch
- Silvan Gehrig, Institut für Software sgehrig@hsr.ch
- Michael Gfeller, Institut für Software, mgfeller@hsr.ch

2. Ziele und Aufgabenstellung

User-Feedback über die Art und Häufigkeit der Nutzung von Programm-Features oder Navigation ist schwierig zu erhalten. Dieses Feedback wird allerdings auf Seite der Hersteller dringend benötigt, um die Software weiter zu entwickeln und das Benutzererlebnis zu verbessern. Hier setzt diese Semesterarbeit an.

Die Semesterarbeit *User Tracking Framework für WPF Applikationen* wurde initiiert um ein Basis-Framework zu schaffen, welches die Messung der Art und Häufigkeit der Nutzung von Programm-Features erlaubt. So werden mit Erlaubnis des Users „unpersonalisierte" Daten gesammelt und diese auf einen zentralen Server in der Cloud übertragen. Bei fehlender Verbindung zum Server können die Daten lokal zwischengespeichert und zu einem späteren Zeitpunkt in die Cloud übertragen werden. Auf Seite des Herstellers lassen sich somit Muster in der Benutzung der Anwendung, also beispielsweise Optimierungspotential in der Usability, erkennen. Ein weiteres Merkmal des Frameworks stellt die Behandlung (Logging) von Ausnahmen und Programm-Fehlverhalten dar. Im Rahmen des Versendens von Fehlerinformationen soll es möglich sein, eine erneute (separate) Erlaubnis vom Benutzer zum Datenversand zu erhalten.

Das Auswerten der Daten auf Herstellerseite der Daten ist nicht Teil der Arbeit *User Tracking Framework für WPF Applikationen*. Die Semesterarbeit konzentriert sich auf das Sammeln und sichere Übertragen der Daten.

Konzeptionell soll die Arbeit eine Übersicht existierender Lösungen im Bereich User Tracking liefern. So können wichtigste Features aus bestehenden Konzepten und Lösungen in das Pflichtenheft des zu realisierenden Frameworks einfliessen.

Herausforderungen der Arbeit *User Tracking Framework für WPF Applikationen*:

- Kurze Studie bestehender User Tracking Lösungen für die Erarbeitung eines Pflichtenhefts
- Ausarbeiten eines Pflichtenhefts entsprechend der Aufgabenstellung
- Aufbauen eines Frameworks auf Basis in WPF / .NET 4.0
- Erarbeiten eines Konzeptes für die sichere Client- / Server-Kommunikation
- Design einer flexiblen Datenhaltung für die Persistierung/Selektion der Daten auf dem Server
- Realisation des Frameworks auf Client-Seite mittels C# .NET 4.0
- Einbinden des entwickelten Frameworks in ein bestehendes Projekt (z.B. ScrumTable)

Optimierungskriterien / Nichtfunktionale Anforderungen

- Einfaches Einbinden in bestehenden Code
- Kommunikation auch in komplexen Firewall-Situationen möglich

3. Zur Durchführung

Mit dem/den Betreuer(n) finden wöchentliche Besprechungen statt. Zusätzliche Besprechungen sind nach Bedarf durch die Studierenden zu veranlassen. Ausser dem Kick-off Meeting sind alle Besprechungen von den Studierenden mit einer Traktandenliste vorzubereiten, die Besprechung ist durch die Studierenden zu leiten und die Ergebnisse sind in einem Protokoll festzuhalten, das den Betreuern per E-Mail zugestellt wird.
Für die Durchführung der Arbeit ist ein Projektplan zu erstellen. Dabei ist auf einen kontinuierlichen und sichtbaren Arbeitsfortschritt zu achten. An Meilensteinen gemäss Projektplan sind einzelne Arbeitsresultate in vorläufigen Versionen abzugeben. Über die abgegebenen Arbeitsresultate erhalten die Studierenden ein vorläufiges Feedback. Eine definitive Beurteilung erfolgt auf Grund der am Abgabetermin abgelieferten Dokumentation.

4. Dokumentation

Über diese Arbeit ist eine Dokumentation gemäss den Richtlinien der Abteilung Informatik zu verfassen (siehe https://www.hsr.ch/Allgemeine-Infos-Diplom-Bach.4418.0.html). Die zu erstellenden Dokumente sind im Projektplan festzuhalten. Alle Dokumente sind nachzuführen, d.h. sie sollten den Stand der Arbeit bei der Abgabe in konsistenter Form dokumentieren. Alle Resultate sind vollständig auf CD/DVD in 3 Exemplaren abzugeben. Der Bericht ist ausgedruckt in doppelter Ausführung abzugeben.

5. Termine

Siehe auch Terminplan auf https://www.hsr.ch/Termine-Diplom-Bachelor-und.5142.0.html?&L=0

Montag, den 17. 09. 2012	Beginn der Studienarbeit, Ausgabe der Aufgabenstellung durch die Betreuer
17. 12.2012	Die Studierenden senden folgende Dokumente der Arbeit per Email zur Prüfung an ihre Betreuer: - Kurzfassung - A0-Poster Vorlagen sowie eine ausführliche Anleitung betreffend Dokumentation stehen unter den allgemeinen Infos Diplom-, Bachelor- und Studienarbeiten zur Verfügung.
21. 12. 2012	Die Studierenden senden die vom Betreuer abgenommene und freigegebene Kurzfassung als Word-Dokument an das Studiengangsekretariat (cfurrer(at)hsr.ch).
21. 12. 2012	Abgabe des Berichtes an den Betreuer bis 17.00 Uhr.

Allfällige weitere Termine sind am Sekretariat der Abteilung Informatik zu erfragen und sollten entsprechend in einem Sitzungsprotokoll dokumentiert werden.

6. Beurteilung

Eine erfolgreiche SA zählt 8 ECTS-Punkte pro Studierenden. Für 1 ECTS Punkt ist eine Arbeitsleistung von ca. 25 bis 30 Stunden budgetiert. Entsprechend sollten ca. 240h Arbeit für die Studienarbeit aufgewendet werden. Dies entspricht ungefähr 17h pro Woche (auf 14 Wochen) und damit ca. 2 Tage Arbeit pro Woche.

Für die Beurteilung ist der HSR-Betreuer verantwortlich.

Rapperswil, den 22. September 2012

Prof. Hans Rudin
Institut für Software
Hochschule für Technik Rapperswil

Zeitplan - User Tracking Framework für WPF

				Inception									Elaboration								
	Iterationen																				
	Meilenstein								MS 1										MS 2		
				SW01			SW02			SW03			SW04			SW05			SW06		
Nr	Task	Soll	Ist	Soll	mt	rl	Soll	mt	rl	Soll	mt	rl	Soll	mt	rl	Soll	mt	rl	Soll	mt	rl
1x	Projektplanung	28.0	28.5	5.0	2.5	2.5	13.0	7.0	6.0	6.0	5.0	2.0	0.0	0.0	0.5	0.0	0.5	0.0	2.0	1.0	1.0
11	Projekt- & Zeitplan	12.0	12.5				8.0	4.0	4.0	1.0	1.0	1.0			0.5				2.0	1.0	1.0
12	Risiko Analyse & Management	4.0	4.5				3.0	1.0	2.0	1.0		1.0									
13	Auftritt & Dokumentvorlagen und -struktur	12.0	11.5	5.0	2.5	2.5	2.0	2.0		4.0	4.0						0.5				
2x	Requirement Analysis	42.0	40.5	0.0	0.0	0.0	13.0	6.0	8.0	19.0	8.0	9.0	9.0	5.0	4.0	0.0	0.0	0.5	0.0	0.0	0.0
21	Konkurrenzanalyse	12.0	12.0				5.0	3.0	2.0	5.0	4.0	1.0	2.0	2.0							
22	Konzepte Datenablage	10.0	10.0				3.0	2.0	2.0	4.0	2.0	1.0	3.0	3.0							
23	Pflichtenheft	20.0	18.5				5.0	1.0	4.0	10.0	2.0	7.0	4.0		4.0			0.5			
4x	Domain Analysis	23.0	21.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.5	1.0	13.0	5.0	7.0	1.0	0.0	1.0	0.0	0.0	0.0
41	Domain Model	8.0	4.0										2.0	2.0							
42	Sequenzdiagramm	5.0	7.0										3.0		3.0	1.0		1.0			
43	Kommunikationsdiagramm	5.0	6.5								1.5	1.0	4.0	1.0	2.0						
44	Technologieabklärungen (DB, ORM)	5.0	4.0										4.0	2.0	2.0						
6x	Design / Architektur	50.0	38.5	0.0	0.0	0.0	0.0	0.0	0.0	2.0	0.0	2.0	2.0	1.0	1.0	2.0	4.0	0.0	6.0	1.0	3.5
61	Design Model	15.0	5.0										1.0		1.0				3.0	1.0	1.0
62	Logische Architektur	30.0	30.5										1.0	1.0		2.0	4.0		2.0		1.5
63	Deployment (Diagramm)	5.0	3.0							2.0		2.0							1.0		1.0
8x	Implementation 1	106.0	106.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	16.0	7.5	11.5	14.0	2.0	11.0
81	Framework-Schnittstelle	16.0	19.5												1.0	2.0		1.0	5.0		5.0
82	User Tracking	33.0	34.0													4.0	3.0	2.5	5.0	1.0	4.5
83	Error Tracking	9.0	8.5																		
84	Datenbankrealisierung&Anbindung	40.0	40.0													10.0	4.5	8.0	4.0	1.0	1.5
85	Einbauen in Beispielprojekt	8.0	4.5																		
9x	Implementation 2	104.0	87.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
91	Serverdienst	16.0	14.0																		
92	Client / Server Kommunikation	28.0	19.5																		
93	Datenbankabgleich, Datenvereinheitlichung	28.0	25.0																		
94	Integrationsdokumentation	12.0	10.5																		
95	Refactoring	10.0	9.0																		
96	Setup	10.0	9.5																		
10x	Testing	31.0	31.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
101	Integration Tests	12.0	15.0																		
102	Unit Tests	19.0	16.0																		
11x	Abschluss	40.0	51.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
111	A0-Poster	10.0	9.0																		
112	Bereinigung Dokumente, Code, ...	20.0	31.0																		
113	Abgabemedien vorbereiten	10.0	11.5																		
12x	Sonstiges	54.0	43.0	6.5	2.5	4.0	2.0	2.0	2.0	2.0	2.0	2.0	8.0	5.0	4.0	2.0	2.0	2.5	2.0	0.5	1.0
121	Team Foundation Server	2.0	2.0	1.5	0.5	1.0												0.5			
122	(Persönliche) Entwicklungsumgebungen	16.0	12.0	3.0	1.0	2.0							6.0	3.0	2.0						0.5
129	Sitzungen/Meetings	36.0	29.0	2.0	1.0	1.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	0.5	0.5
	Total	478.0	448.5	11.5	5.0	6.5	28.0	15.0	16.0	29.0	16.5	16.0	32.0	16.0	17.5	21.0	14.0	15.5	24.0	4.5	16.5

Zeitplan - User Tracking Framework für WPF

				Construction 2									Transition					
	Iterationen																	
	Meilenstein						MS 4			MS 5						MS 6		
				SW10			SW11			SW12			SW13			SW14		
Nr	Task	Soll	Ist	Soll	mt	rl	Soll	mt	rl	Soll	mt	rl	Soll	mt	rl	Soll	mt	rl
1x	Projektplanung	28.0	28.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.5	0.0	0.0	0.0
11	Projekt- & Zeitplan	12.0	12.5															
12	Risiko Analyse & Management	4.0	4.5												0.5			
13	Auftritt & Dokumentvorlagen und -struktur	12.0	11.5															
2x	Requirement Analysis	42.0	40.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
21	Konkurrenzanalyse	12.0	12.0															
22	Konzepte Datenablage	10.0	10.0															
23	Pflichtenheft	20.0	18.5															
4x	Domain Analysis	23.0	21.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	4.0	3.5	0.0	2.0	0.0	1.5
41	Domain Model	8.0	4.0										2.0	1.5				
42	Sequenzdiagramm	5.0	7.0										1.0	1.0		2.0		1.5
43	Kommunikationsdiagramm	5.0	6.5										1.0	1.0				
44	Technologieabklärungen (DB, ORM)	5.0	4.0															
6x	Design / Architektur	50.0	38.5	3.0	0.0	2.0	5.0	9.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	4.0	2.0	3.0
61	Design Model	15.0	5.0														1.0	
62	Logische Architektur	30.0	30.5	3.0		2.0	5.0	9.0								4.0	1.0	3.0
63	Deployment (Diagramm)	5.0	3.0															
8x	Implementation 1	106.0	106.5	0.0	2.0	0.0	8.0	1.0	8.0	0.0	0.0	4.0	0.0	0.0	1.5	2.0	0.0	1.5
81	Framework-Schnittstelle	16.0	19.5				3.0	1.0	3.0			2.0			1.0	2.0		0.5
82	User Tracking	33.0	34.0		2.0		5.0		4.0			1.0						
83	Error Tracking	9.0	8.5						1.0			0.5						
84	Datenbankrealisierung&Anbindung	40.0	40.0												0.5			1.0
85	Einbauen in Beispielprojekt	8.0	4.5									0.5						
9x	Implementation 2	104.0	87.5	26.0	5.0	16.5	28.0	4.0	19.5	12.0	0.0	10.5	6.0	2.0	3.0	3.0	1.0	3.0
91	Serverdienst	16.0	14.0	2.0		1.0	6.0	2.0	3.0	2.0		2.0						
92	Client / Server Kommunikation	28.0	19.5	8.0		5.5	8.0	1.0	6.0									
93	Datenbankabgleich, Datenvereinheitlichung	28.0	25.0	16.0	5.0	9.0	4.0		3.5	2.0		2.0						1.0
94	Integrationsdokumentation	12.0	10.5					1.0					2.0	2.0		2.0	1.0	1.0
95	Refactoring	10.0	9.0				8.0		6.0				2.0		2.0	1.0		1.0
96	Setup	10.0	9.5			1.0	2.0		1.0	8.0		6.5	2.0		1.0			
10x	Testing	31.0	31.0	8.0	8.5	0.0	0.0	0.0	0.0	4.0	0.0	2.5	18.0	2.0	15.0	4.0	0.0	3.0
101	Integration Tests	12.0	15.0	8.0	8.5					2.0		1.5	2.0	2.0		4.0		3.0
102	Unit Tests	19.0	16.0							2.0		1.0	16.0		15.0			
11x	Abschluss	40.0	51.5	0.0	0.0	0.0	0.0	2.0	0.0	16.0	12.0	2.0	22.0	13.0	9.0	12.0	7.0	6.5
111	A0-Poster	10.0	9.0							8.0	6.0		2.0		2.0		1.0	
112	Bereinigung Dokumente, Code, ...	20.0	31.0					2.0		8.0	6.0	2.0	20.0	13.0	6.0	2.0		2.0
113	Abgabemedien vorbereiten	10.0	11.5												1.0	10.0	6.0	4.5
12x	Sonstiges	54.0	43.0	3.0	1.0	1.5	2.0	1.0	1.0	2.0	1.0	1.0	2.0	1.0	1.0	0.0	0.0	0.0
121	Team Foundation Server	2.0	2.0															
122	(Persönliche) Entwicklungsumgebungen	16.0	12.0	1.0		0.5												
129	Sitzungen/Meetings	36.0	29.0	2.0	1.0	1.0	2.0	1.0	1.0	2.0	1.0	1.0	2.0	1.0	1.0			
	Total	478.0	448.5	40.0	16.5	20.0	43.0	17.0	28.5	34.0	13.0	20.0	52.0	21.5	30.0	27.0	10.0	18.5

Risikomanagement

Projekt: **User Tracking Framework für WPF**
Erstellt am: 20.09.2012
Zuletzt aktualisiert: 27.11.2012
Autor: Roger Landolt, Marco Tanner
Gewichteter Schaden: 20.75

Infrastruktur Risiken

Implementationsrisiken

Nr	Titel	Beschreibung	Schaden [h]	Eintrittswahrscheinlichkeit	Gewichtet	Präventionsmassnahmen	Verhalten beim Eintreten	Betroffene Arbeitspakete	Spätesteste Iteration	Risiko beseitigt
R01	Ausfall Internetprovider	Die Internetverbindung eines Projektmitglieds fällt aus	5	5.0%	0.25	Keine	Andere Internetverbindung verwenden/ Arbeitsplatz zur HSR verschieben, Arbeiten verschieben	Alle	x	0
R02	Ausfall HSR Netz	Das Netzwerk der HSR fällt aus und der TFS-Server ist nicht mehr erreichbar	10	5.0%	0.50	Immer vor Arbeitsschritt aktuelle Version von TFS beziehen und eigenen Stand auf TFS laden. Regelmässiges Backup der Sourcen.	Abklären wie lange der Ausfall dauert. Im Extremfall Ausweich TFS erstellen, dies ist aber kaum notwendig.	Alle	x	0
R03	Ausfall Arbeitsplatz	Der Arbeitsplatz eines Projektmitglieds fällt aus	10	5.0%	0.50	Einrichten eines Ersatz Arbeitsplatzes	Arbeiten im Studienzimmer, Ersatznotebook organisieren.	Alle	x	0
R04	Datenverlust	Es gehen Daten verloren oder werden unabsichtlich gelöscht	10	10.0%	1.00	TFS Source-Verwaltung, regelmässige Backups der Dokumente	Wiederherstellen des letzten Standes und Neuerarbeitung der Daten	Alle	x	0
R05	Ausfall TFS	Der Team-Foundation Server fällt aus	25	10.0%	2.50	Immer vor Arbeitsschritt aktuelle Version von TFS beziehen und eigenen Stand auf TFS laden. Regelmässiges Backup der Sourcen.	Erstellen einer provisorischen TFS-Instanz, Reinitialisierung mit den Daten.	Alle	x	0
R06	Event-Tracking zu kompliziert	Das Abfangen der Events eines Projektes erweist sich als sehr aufwändig oder nicht möglich.	50	10.0%	5.00	Informieren über Tools mit ähnlicher Funktionalität, möglichst frühe Erkundigung über diesen kritischen Punkt.	Nur auf bestimmte Events konzentrieren, Abklärung mit dem Projektleiter	Implementation	Construction 1	1
R07	Performance eingeschränkt	Die Performance des User-Trackings vermindert die Usability des Hauptprogrammes	40	10.0%	4.00	Identifizieren und einkalkulieren bekannter Bottlenecks im Vorraus, Asynchroner Aufbau	Verringern der verfolgten und gesammelten Events.	Testing	Construction 2	1
R08	Security lässt sich nicht implementieren	Die Security lässt sich nicht wie gewünscht implementieren.	50	5.0%	2.50	Prüfen von Applikationen mit ähnlichen Anforderungen		Implementation	Construction 1	1
R09	Keine Einfache Integration in bestehendes Projekt möglich	Der Aufwand um die erstellte Library in ein Projekt einzubinden wird sehr gross.	30	15.0%	4.50	Bereits zu Beginn der Construction in Beispielprojekt einbauen. Verfolgen des Implementationsaufwandes.	Konfigurationsmanager zur Verfügung stellen der Minimalkonfiguration ermöglicht.	Implementation	Construction 1	1
		Total Schadenspotential	215	**Gewichtet**	20.75					

D. Zeitauswertung

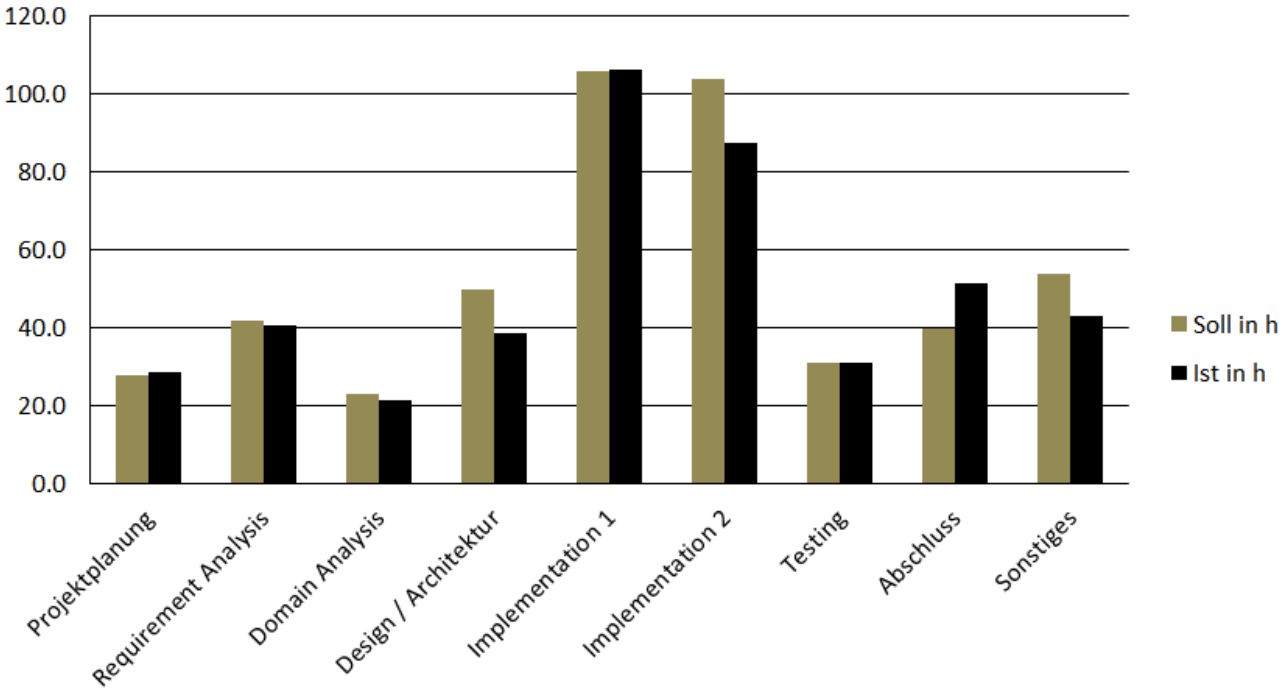
Dieses Dokument zeigt die aufgewendeten Stunden von Roger Landolt und Marco Tanner in aufbereiteter Form. Die Stunden wurden in einem separaten Excelsheet erfasst.

D.1. Anmerkung Sollstunden

Die Studienarbeit wird mit 8 ECTS-Punkten bewertet. Ein Punkt entspricht dabei einem Arbeitsaufwand von 25-30 Stunden. Mit zwei Mitgliedern liegen wir somit bei 400 bis 480 Stunden.

D.2. Auswertungen

D.2.1. Soll/Ist Vergleich pro Kategorie

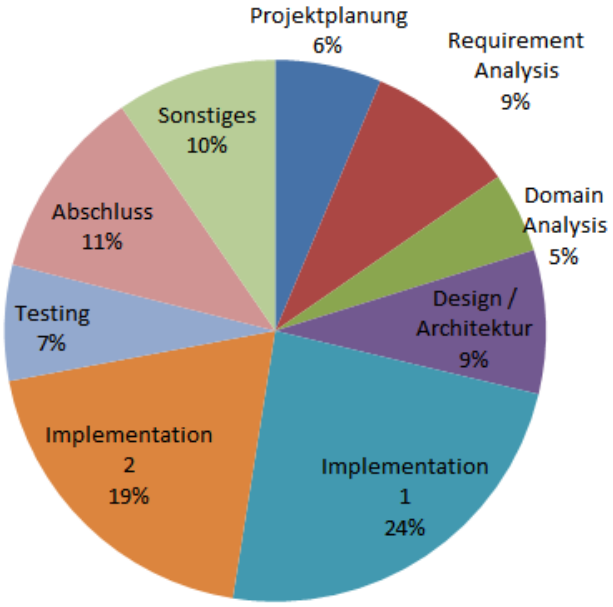


Die Obergrenze von 480 Stunden haben wir zu Beginn des Projektes auf die verschiedenen Kategorien und Arbeitspakete aufgeteilt.

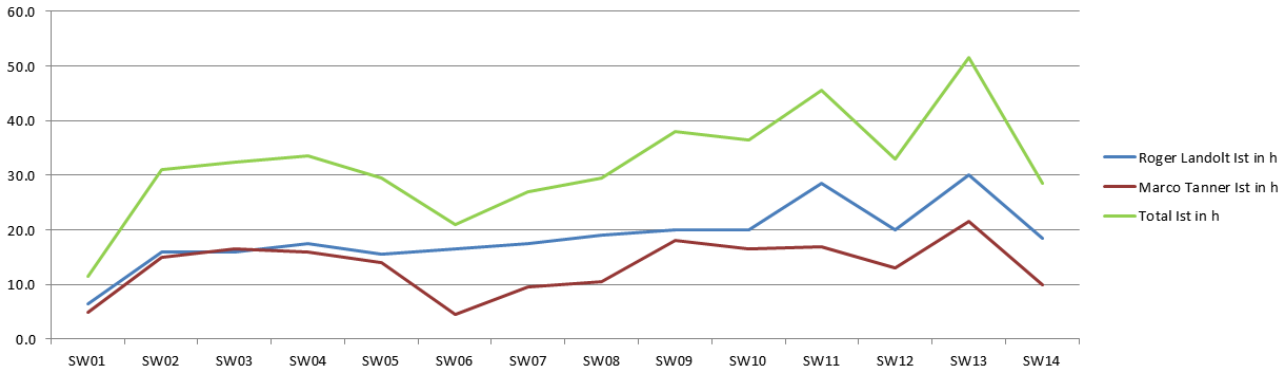
Wir haben diese Sollstunden nach dem Projektbeginn nicht manipuliert. Durch diese Entscheidung können wir daraus bessere Erfahrungen in der Zeit-Schätzung sammeln und somit auch vom grösseren Lerneffekt profitieren.
Im grossen und ganzen sind wir mit den Soll/Ist Zeiten sehr zufrieden. In den Kategorien Design/Architektur sowie Implementation 2 benötigten wir weniger Zeit als eingeschätzt. In der Implementation 2 entstand die Abweichung hauptsächlich durch den überschätzen Aufwand zur Datensynchronisation. Eine Reserve in dieser Kategorie war aber durchaus sinnvoll, da es eine kritische Komponente umfasst.
Unterschätzt haben wir den Aufwand des Abschlusses, wo wir mit der Bereinigung und Zusammenführung der Dokumente mehr Zeit als gedacht aufwenden mussten.

D.2.2. Anteil pro Kategorie

Der Hauptteil der Arbeit, genau genommen 43%, wurde für die Implementation aufgewendet.
Die Analyse (Requirements- und Domainanalyse) nahm zusammen mit der Design- und Architekturphase rund 23% in Anspruch.
Die weiteren Angaben können in folgendem Diagramm abgelesen werden.

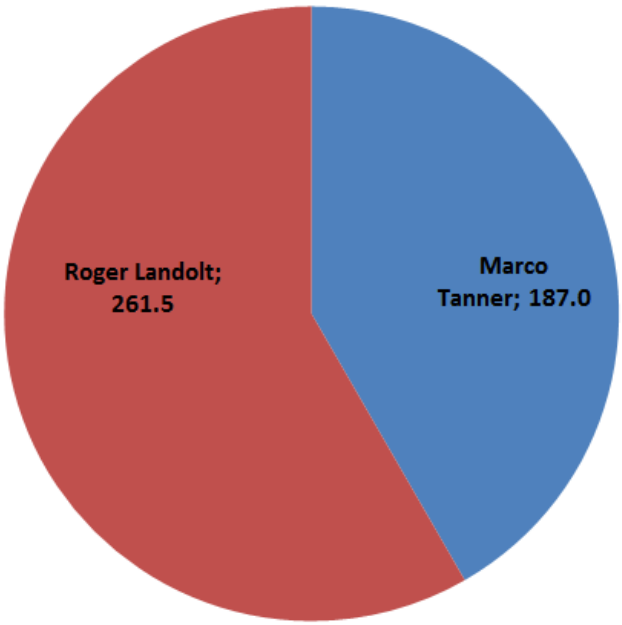


D.2.3. Aufwand pro Woche



An dieser Stelle ist anzumerken, dass in der ersten Schulwoche unsere ursprüngliche Studienarbeit “ScrumTable: Redmine Data Driver” zurückgezogen wurde. Aufgrund dessen konnten wir erst in der zweiten Woche, nach Erhalt der Aufgabenstellung, richtig beginnen.
Der Zacken in den Wochen 12 und 13 ist durch andere Module, bzw. Aufwand zu Gunsten dieser, entstanden. Zudem wollten wir in der letzten Woche nicht unter zu starken Zeitdruck geraten.
Neben einem WK-Aufgebot von Roger Landolt in der Woche 6 gab es keine Abwesenheiten.

D.2.4. Aufwand pro Projektmitglied



Der Mehraufwand von Roger Landolt liegt vor Allem an den fortgeschrittenen .Net-Kentnissen gegenüber Marco Tanner.

E. Projektglossar

- **Client**
  Der Client wird zusammen mit der Host-Applikation beim Enduser installiert und bildet den Hauptteil des Projekts.
- **Enduser/Endbenutzer**
  Der Endbenutzer der Host-Applikation. Der Enduser sollte keinen direkten Kontakt mit dem Framework haben ausser die Frage um seine Erlaubnis für das Sammeln seiner Benutzungsdaten.
- **Entwickler**
  Der/die Benutzer des Frameworks. Der Benuzter integriert das Framework in die Host-Applikation.
- **Event/Occurrence**
  Ein Event das aufgezeichnet wurde. Dies kann ein Ereignis auf einem GUI-Control sein, eine Exception, ein Flusssteuerungsevent oder eine entwicklerspezifische Meldung.
- **GUI**
  Graphical User Interface: Eine grafische Oberfläche eines Programms.
- **Host-Applikation**
  Ein (möglicherweise schon bestehendes) Software-Programm welches auf dem .NET-Framework aufbaut und eine WPF-GUI mitbringt. In ein solches Projekt soll das geplante User Tracking Framework integriert werden.
- **Lookup-Tabellen**
  Dies sind Tabellen welche als dynamische Enums angesehen werden können. Sie beinhalten alle bisher vorgekommenen Werte und werden laufend erweitert. Einträge darin werden lokal erstellt und bei vorhandener Verbindung werden die benötigten Einträge mit dem Server abgeglichen. Falls der vom Client erfasste Eintrag auf dem Server bereits vorkommt wird der Eintrag auf dem Client und alle Referenzen darauf angepasst.
- **Server**
  Die Serverkomponente des entwickelten Frameworks UTF4WPF.
- **SSDL**
  Die *Store schema definition language* ist eine XML-basierte Sprache um das Storage-Model einer Entity Framework Anwendung zu beschreiben.
- **Tracing/Tracking/Überwachung**
  Die Hauptaufgabe des Clientteils. Bezeichnet das Aufzeichnen und Festhalten von Events.

- **UTF4WPF**
  User Tracking Framework 4 WPF - Applications: Der Name des entwickelten Frameworks / der Titel der Studienarbeit.
- **WPF**
  Windows Presentation Foundation: Das ins .NET-Framework integrierte GUI-Framework.

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL
FHO Fachhochschule Ostschweiz

INSTITUT
FÜR
SOFTWARE

Sitzungsprotokoll 19.09.2012

Ort: HSR, 1.275
Zeit: 10:10
Anwesend: H. Rudin, M. Gfeller, S. Gehrig, R. Landolt, M. Tanner

Wichtige Beschlüsse:

- Die SA "Redmine-Treiber für Scrum-Table" wird **nicht** durchgeführt
- Stattdessen wird das neue SA-Thema auf "WPF-Usertracking" festgelegt
- Die Aufgabenstellung wird bis am Freitag, 21.09. abgegeben.
- Die Sitzungen werden zukünftig jeweils wöchentlich am Mittwoch von 13:10-14:50 geplant. Allfällige Terminänderungen werden per Mail kommuniziert.
- Die Zeit bis am Freitag wird dazu genutzt, die Entwicklungsumgebung einzurichten sowie mit dem Risiko-Management zu beginnen.

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL
FHO Fachhochschule Ostschweiz

INSTITUT
FÜR
SOFTWARE

Projektsitzung vom 26.09.2012

Ort: HSR, 1.275
Zeit: 13:10
Eingeladen: H. Rudin, M. Gfeller, S. Gehrig, R. Landolt, M. Tanner
Abwesend: Niemand

Traktanden

1. Aufgabenstellung besprechen
2. Verlangter Dokumentationsumfang
3. Firewall versus TFS
4. Was ist mit Error-Tracking gemeint?
5. Vorlage Projektplan-Entwurf

Protokoll

Das Protokoll vom 19.09.2012 wurde genehmigt.

TFS

Der TFS wird durch das Institut wöchentlich gesichert. Zugriff von ausserhalb des HSR-Netzwerks ist nicht möglich.

Dokumentation

In der Sitzung besprochene Dokumente sollen jeweils, wenn möglich am Dienstagabend, an die Betreuer gesandt werden.
Systemsequenzdiagramme und Operation Contracts werden weggelassen, da es ohnehin keine Use-Cases gibt.

Requirements
Allgemeine Ideen und die “Vision” werden im Requirementsdokument festgehalten.

Nichtfunktionale Anforderungen sollen so formuliert sein, dass sie mess- oder testbar sind. Es werden verschiedene Vorschläge durch die Studenten ausgearbeitet, in welchem Umfang die messbaren Daten erfasst und gespeichert werden, mit besonderem Hinblick auf die Wiedererkennbarkeit einzelner User und Applikationen. Die einzelnene Vorschläge werden durch die Betreuer begutachtet und sie entscheiden sich dann für die zu implementierende Variante.

Error-Tracking
Unter Error-Tracking ist zu verstehen, das auch (unhandled) Exceptions mitgeloggt werden und mit an den Server geschickt werden. Dies beinhaltet auch Programmabstürze. Hat der Benutzer zuvor keine Erlaubnis für das Versenden seiner (User-Tracking-)Daten gegeben so werden die Usertracking Daten zusammen mit dem Error-Report versandt. Das bedeutet, dass die User Tracking-Komponente immer mitläuft, auch ohne Versand-Erlaubnis.

Varia
R. Landolt ist am 30. und 31. Oktober im WK und kann deshalb nicht an der Sitzung teilnehmen.

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL
FHO Fachhochschule Ostschweiz

INSTITUT
FÜR
SOFTWARE

Projektsitzung vom 03.10.2012

Ort: HSR, 1.275
Zeit: 13:10
Eingeladen: H. Rudin, M. Gfeller, S. Gehrig, R. Landolt, M. Tanner
Abwesend: Niemand

Dokumente: Projektplan, Anforderungs-Dokument

Traktanden

1. Anmerkungen zum Projektplan
2. Diskussion der Requirements
3. Datenhaltung

Protokoll

Das Protokoll vom 26.09.2012 wurde genehmigt.

Allgemein

Um zukünftige Meetings effizienter zu gestalten soll in Zukunft der Beamer miteinbezogen werden um besprochene Dokumente für alle anzuzeigen. Zusätzlich soll jeweils als zweites Traktandum ein kurzer Rückblick auf die Woche (Stand des Projekts, Aktivitäten) gegeben werden.

Erlaubnis des Endbenutzers

Ohne Erlaubnis des Benutzers werden nur Exceptions geloggt, keine Events. Denkbar wäre, das die Events der aktuellen Session jeweils aufgezeichnet werden und im Fehlerfall als erweiterte Fehlerinformationen mitgeschickt werden, im regulären Fall werden Sie beim Beenden der Applikation gelöscht.

Anforderungsdokument

Die Konkurrenzanalyse ist zu mager. Es soll zusätzlich nach weiteren User-

Tracking Produkten in anderen Umgebungen (Java, iOS, . . .) gesucht werden.
Weitere Kritik:

- Eigene Applikation aus Featuretabelle entfernen und separat aufführen und detaillierter beschreiben oder besser hervorheben
- Mengen- und/oder Zeitmässig einstellbar, wieviele Daten maximal lokal zwischengespeichert werden
- Anforderungen nummerieren oder identifizierbar machen
- Kapitel 2.3 und 2.1.1 zusammenführen
- Qualitätsanforderungen detaillierter angeben und Text anpassen. Neu unter Nichtfunktionale Anforderungen einordnen. Allenfalls Qualitätsanforderungen in ISO-Modell einpassen
- Security-Anforderungen definieren
- Kapitel 2.4 entfernen
- Kapitel 2.1.1 und 2.7 das Wort “Projekt” ersetzen durch “Produkt”
- Kapitel 2.10 an den Anfang verschieben
- Allfällige Serveranforderungen definieren
- Genauere Anforderungen zu den Themen
  - Installation/Einbindung des Frameworks
  - Erfragen der Benutzererlaubnis
  - Kommunikation (durch Firewalls)
  - Eigene Dialoge

  definieren.

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL
FHO Fachhochschule Ostschweiz

INSTITUT
FÜR
SOFTWARE

Projektsitzung vom 10.10.2012

Ort: HSR, 1.275
Zeit: 13:10
Eingeladen: H. Rudin, M. Gfeller, S. Gehrig, R. Landolt, M. Tanner
Abwesend: S. Gehrig

Traktanden

1. Protokoll besprechen
2. Projektstatus-Update
3. Requirements
4. Architektur
5. TFS
6. Visual Studio
7. weiteres Vorgehen

Protokoll

Das Protokoll vom 03.10.2012 wurde genehmigt.

Requirements-Dokument

- Das Dokument soll aufgeteilt in 2 Teile geteilt werden, eine Analyse/ Studie und eine Anforderungsspezifikation/Pflichtenheft.
- In die Anforderungen soll der konkrete Vorschlag zur Datenhaltung eingearbeitet werden. Dieser wird dann Markus Stolze zur Kontrolle vorgelegt.
- Kleinere Typos und Fehler

Architektur-Dokument

- Eine Einleitung hinzufügen, die die generelle Architektur in Worte fasst.
- Das Glossar soll in ein eigenes Dokument ausgelagert werden.
- Der Abschnitt zu den Tools soll in den Projektplan verschoben werden.
- Das Domain-Modell soll in die Analyse/Studie verschoben werden.
- Der Begriff Puffer / Lokale Datenbank muss im Glossar erklärt werden.

Varia

- Die Arbeitszeiten sollen im Plan nachgeführt und bei Meetings gezeigt werden.
- Wegen fehlender SQLite-Unterstützung wird anstelle von Visual Studio 2012 mit Visual Studio 2010 gearbeitet.

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL
FHO Fachhochschule Ostschweiz

INSTITUT
FÜR
SOFTWARE

Projektsitzung vom 17.10.2012

Ort: HSR, 1.275
Zeit: 13:10
Eingeladen: H. Rudin, M. Gfeller, S. Gehrig, R. Landolt, M. Tanner
Abwesend: Niemand

Traktanden

1. Protokoll der letzten Sitzung besprechen
2. Projektstatus-Update
3. Einblick zur Umsetzung des Datenmodells
4. Kurze Demo des Event-Capture Testprototyps
5. Besprechung der Rückmeldung von M. Stolze

Protokoll

Das Protokoll vom 10.10.2012 wurde genehmigt.

Custom Controls:
Sicher alle built-in Controls unterstützen, stark customized Controls via manuellem Hook abfangbar machen.
Serverseite:
Als Basis für den Serverpart wird IIS (ab Version 7.0) verwendet.
ERD:
Die erfassten Daten zu den GUI-Controls müssen detailierter modelliert werden. Insbesondere die Wiedererkennbarkeit von Controls, damit alle Ocurrences sicher einem Control zugeordnet werden können. Allenfalls variable Zusatzdaten mit Key/Value-Paaren abspeichern. Bei den Exceptions müssen die Inner Exceptions mitaufgezeichnet werden.
Sequence Diagram:
Benötigt noch Erkärungen in Textform. (Für das Erzeugen neuer Objekte gibt es in astah* “Create messages”)

Varia:
Im Kopf der bisherigen Dokumente soll H. Rudin zukünftig als betreuender Dozent anstelle des Experten genannt werden.

Im Projektstatus update während der Sitzung soll genauer bescheid gegeben werden woran konkret gearbeitet wurde und wieviel Zeit pro Aktivität aufgewendet wurde.

Rückmeldung von M. Stolze

Die Rückmeldung von Professor Stolze ist heute angekommen:

> Dies scheint die Anforderungen für das geplante Framework sinnvoll zu umschreiben.
>
> Eine Unterscheidung in der Nutzung welche ich vermisse ist
> - 1) Entwickler welche ein *bestehendes* System mit dem Framework ausrüsten wollen.
> - 2) Entwickler welche das Framework von Anfang an nutzen.
>
> Bei 1) werden sich Fragen ergeben: sind alle relevanten Controls typisiert und benannt?, Wenn Nein was dann, ... Etc. . Das heisst es wäre sinnvoll nicht nur ein bestehendes System (ScrumTable) zu analysieren, sondern mindestens noch ein zweites System (besser mehr)
>
> In die gleiche Richtung geht die Frage: ”wie viel Aufwand ist es ein bestehendes System zu instrumentieren, wie viel Aufwand ist es die Instrumentierung wieder zu entfernen. Auch auf diese Fragen möchte ich Antworten sehen.
>
> Viele Grüsse :mstolze

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL
FHO Fachhochschule Ostschweiz

INSTITUT
FÜR
SOFTWARE

Projektsitzung vom 24.10.2012

Ort: HSR, 1.275
Zeit: 13:10
Eingeladen: H. Rudin, M. Gfeller, S. Gehrig, R. Landolt, M. Tanner
Abwesend: S.Gehrig

Traktanden

1. Protokoll der letzten Sitzung besprechen
2. Projektstatus-Update
3. Lösung der Properties-Frage
4. *getOrCreate()* - Problematik
5. Sitzung vom 31. Oktober: Ausfall?

Protokoll

Das Protokoll vom 17.10.2012 wurde genehmigt.

Properties:
Die Lösung wurde vorgestellt und für OK befunden.
Zusatzbemerkung: Bei den EventArgs darf nur eine Hierarchiestufe der Properties gesichert werden, da ansonsten die Datenmenge explodieren könnte.

***getOrCreate()*:**
Kann allenfalls auch nur get() benannt werden. Ansonsten wenn sinnvoll, ist es kein Problem, solche Methoden einzubauen.

Ausfall:
Die Sitzung vom 31. Oktober fällt aus. Die nächste Sitzung findet somit am 7. November statt.

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL
FHO Fachhochschule Ostschweiz

INSTITUT
FÜR
SOFTWARE

Projektsitzung vom 07.11.2012

Ort: HSR, 1.275
Zeit: 13:10
Eingeladen: H. Rudin, M. Gfeller, S. Gehrig, R. Landolt, M. Tanner
Abwesend: S. Gehrig

Traktanden

1. Protokoll der letzten Sitzung besprechen
2. Projektstatus-Update
3. Interne Fehlermeldungen: CustomOccurrence
4. “x:Name”-Attribut auf CustomControls
5. Demo: Einbindung in bestehendes Projekt “TimeKeeper” (dtd, xml...)

Protokoll

Das Protokoll vom 24.10.2012 wurde genehmigt.

Interne Fehlermeldungen:
Interne Fehler werden als CustomOccurrence in die Datenbank gespeichert. Fehler die auftreten bevor eine Datenbankverbindung zustande gekommen ist werden ins Windows-Eventlog geschrieben.

Demo/Konfiguration:
Die Konfiguration via XML-Datei ist in Ordnung. Es soll ein zusätzliches <event-all>-Tag als “*-Operator” für Events definiert werden.

Architektur:
Die ausführliche Kritik am Architektur wurde dankend zur Kenntnis genommen. Nebst denen im Antwortmail von H. Rudin enthaltenen Kritikpunkten muss angemerkt werden dass das Datenmodell z.Z. in einer Visual-Studio spezifischen Notation gehalten und kein klassisches ER-Diagramm ist.

x:Name
Anstelle des Traktandums 4 wurde ein Problem mit doppelt auftretenden Events von CustomControls diskutiert. Dies wurde auf Fehler im Beispielprojekt zurückgeführt und nicht als allgemeines Problem mit WPF erkannt.

Nachträglich Anmerkung Datenmodell:
Bei der Besprechung wurde angemerkt, dass die Verbindung ExceptionalOccurrence zu ExceptionalOccurrence nicht 0..1 zu 0..1 sein soll sondern 1 zu 0..1.
Nachträglich wurde von Roger Landolt und Marco Tanner festgestellt dass die bisherige Angabe im Sinne des Visual Studio-Diagrammes korrekt ist. Den eine Exception kann keine oder eine InnerException haben, muss aber nicht immer eine Parent-Exception haben.

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL
FHO Fachhochschule Ostschweiz

INSTITUT
FÜR
SOFTWARE

Projektsitzung vom 28.11.2012

Ort: HSR, 1.275
Zeit: 13:10
Eingeladen: H. Rudin, M. Gfeller, S. Gehrig, R. Landolt, M. Tanner
Abwesend: S. Gehrig

Traktanden

1. Statusupdate
2. Datenverwaltung
3. SSL - Zertifikate
4. OccurrenceArgs konfigurierbar

Protokoll

Das Protokoll vom 21.11.2012 wurde genehmigt.

Datenverwaltung:
Inhalt des Diagramms ist in Ordnung. Für zusätzliche Klarheit braucht es noch ein Sequenzdiagramm Client $\leftrightarrow$ Server und einen einleitenden Erkärungstext (insbesondere für die abgeleiteten CRUD-Manager).

SSL - Zertifikate:
Die Umsetzung / Begründung der Art der Umsetzung im SAD nachführen.
Auf Client-Authentifizierung wird verzichtet.

OccurrenceArgs konfigurierbar:
Die Änderung an der Erfassunsmethode der OccurrenceArgs (namentlich dass sie jetzt doch einstellbar sind) ist in Ordnung.

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL
FHO Fachhochschule Ostschweiz

INSTITUT
FÜR
SOFTWARE

Projektsitzung vom 05.12.2012

Ort: HSR, 1.275
Zeit: 13:10
Eingeladen: H. Rudin, M. Gfeller, S. Gehrig, R. Landolt, M. Tanner
Abwesend: S. Gehrig

Traktanden

1. Statusupdate
2. Entwurf Poster

Protokoll

Das Protokoll vom 28.11.2012 wurde genehmigt.

Statusupdate:
Sämtliche Dokumentation soll für eine letzte Feedback-Runde nocheinmal an die Betreuer gesandt werden.

Poster:
Allgemein sollen eher Aufzählungen mit Bullets anstelle von Fliesstext verwendet werden.
Realisierung → Resultat
Idee → Ausgangslage
Der Titel wäre besser komplett in Englisch, inkl. der Abkürzung “UTF4WPF”.

Die Grafik soll um einen Übersichtsteil à la Deployment-Diagramm erweitert werden.

Mehr als 1 Tracker-Instanz:
(Das Thema ist erst am morgen vor der Sitzung aufgetaucht und fehlt deshalb in der Traktandenliste)
Es muss möglich sein, die Client-Applikation mehr als einmal zu starten,

ohne dass sich die jeweiligen Instanzen des UTF4WPF-Tracker in die Quere kommen, dies insbesondere im Hinblick auf Standby- resp. Hibernate-Modi des Betiebssystems.

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL
FHO Fachhochschule Ostschweiz

INSTITUT
FÜR
SOFTWARE

Projektsitzung vom 12.12.2012

Ort: HSR, 1.275
Zeit: 13:10
Eingeladen: H. Rudin, M. Gfeller, S. Gehrig, R. Landolt, M. Tanner
Abwesend: S. Gehrig

Traktanden

1. Statusupdate
2. Code-Freeze
3. Abgabe: Kriterien, Formelles
4. Unit-Testing

Protokoll

Das Protokoll vom 05.12.2012 wurde genehmigt.

Abgabe:
Die Abgabematerialien müssen folgende Kriterien erfüllen bzw. Inhalte mitbringen:

- Management-Summary: 2-4 Seiten am Anfang des Dokuments
- Persönliche Berichte: Am Schluss des Dokuments
- Eine gut geordnete Dateistruktur mit allen Subdokumenten, Grafiken, LaTeX-Sourcen, etc. . .
- 2 CD's mit "Allen" Dateien
- 1 CD nur mit dem Hauptdokument für eprints.hsr.ch

Unit-Testing:
Damit auch die $private$-Parts der Clients getestet werden kann, ist es möglich das Testing-Projekt als $friend$ zum Client zu markieren. Allenfalls könnte man auch einen Test mit einem "Klick-Automaten"-Programm getestet werden. Im Testing-Kapitel sollen die Ergebnisse der Unit-Tests der abgegebenen Version des Projekts erwähnt werden.

Feedback zum "Meta"-Dokument

- Aufbau: Zuerst "Sachliches", Verwaltungstechnisches/Projektmgmt am Schluss
- Datenhaltung umbenennen: Ist eine Studie
- Da die Aufgabenstellugn im Abstract erwähnt wird, kann sie im Projektplan weggelassen werden
- Meilensteine: Keine bekannte Vorgehensweise/Prozess, bessere Beschreibungen und Erläuterungen notwendig
- Begleitdokumente zum Projektplan besser umschreiben
- Konkurrenzanalyse: Einführungstext, Erkenntnisse, Schlussfolgerung einfügen
- Pflichtenheft $\rightarrow$ Anforderungsspezifikation
- Domain-Modell: Die assoz. Klasse Benutzung ist falsch, die Beziehung zwischen Applikation und Umgebung läuft ind die falsche Richtung
- Die Auflösung/Qualität der Bilder muss genügend hoch sein
- Formatierung: Teilweise sind Absätze eingerückt
- Überarbeitung in Rechtschreibung, insbesondere Interpunktion

Varia:
Dies ist nach Absprache die letzte Projektsitzung. Die Sitzung am 19.12.2012 fällt aus.

Abbildungsverzeichnis

Tabellenverzeichnis