Automatisierte Erstellung von Word Rapporten

Gennaro Piano

16. April 2014

Zusammenfassung

Die KMU IT Management AG erstellt monatlich für jeden Kunden einen individuellen Bericht. Die Erstellung aller Berichte benötigt eins bis zwei Arbeitstage. Das Ziel dieser Arbeit ist es, einen Prototyp zu entwickeln, der diesen Prozess stark verkürzt. Der Prototyp wird mit Hilfe von Visual Studio und C# entwickelt und wird in zwei Teile geteilt. Der erste Teil beschäftigt sich mit der Beschreibung des Rapports. Die Beschreibung des Rapports wird in einem XML Dokument zusammengefasst. Die Beschreibung beinhaltet Textblöcke, Tabellen und Datenbankabfragen. Das XML Dokument kann über eine Benutzeroberfläche erstellt werden. Dadurch kann die Beschreibung eines Rapports, ohne grosse Fachkenntnisse der XML Sprache, erstellt werden. Der zweite Teil des Prototyps übernimmt die Beschreibung des Rapports in ein Word Dokument. In diesem Teil werden die Datenbankabfragen ausgeführt und Texte sowie Tabellen in das Word Dokument eingefügt. Als Resultat erhält man ein fertiggestelltes Word Dokument. Der zweite Teil der Prototyps kann über die Command Line ausgeführt werden und somit als geplanter Task ausgeführt werden. Dieser Bericht befasst sich mit dem gesamten Projekt. Er soll aufzeigen, wie dieser Prototyp entwickelt und fertiggestellt wurde.

Inhaltsverzeichnis

Glossary

.NET Software Framework von Microsoft, welches zur Entwicklung und Ausführung von Anwendungen dient.

C# Eine von Microsoft entwickelte objektorientierte Programmiersprache.

Command Line Siehe Windows Command.

Document Object Model Document Object Model, kurz DOM, ist eine Klassenbibliothek, welche XML Dokumente schreibt und liest.

Exception Excpetions sind Ausnahmebehandlungen während der Laufzeit eines Programmes. Exceptions können abgefangen und behandelt werden. Die Behandlung verhindert den Absturz der gesamten Applikation.

Firewall Eine Firewall schützt ein internes Netzwerk vor unerlaubtem Zugriff von aussen.

GitHub GitHub ist ein webbasierter Hosting-Dienst für Sofware-Entwicklungsprojekte.

Interface Baustein eines Programmes, welches den Aufbau verschiedener Klassen beschreiben kann.

LINQ LINQ verbessert die Abfragemöglichkeiten der Programmiersprache C#..

LINQ to XML LINQ to XML ist eine Klassenbibliothek innerhalb von .NET, welche mit Hilfe von LINQ ein XML Dokument schreiben oder lesen kann.

Microsoft Office Word Textverarbeitungsprogramm von Microsoft, aktuellste Version ist Microsoft Office Word 2013.

Microsoft SQL Datenbankmanagementsystem von Microsoft.

MySQL Relationales Datenbankmanagementsystem von Oracle.

Open XML Ein offener Standard für Büroanwendungen basierend auf XML-Dateiformate. Dieser Standard wurde von Microsoft entwickelt. Microsoft Office benutzt seit Office 2007 Open XML.

Oracle Oracle ist ein Softwarehersteller, welcher auf Datenbankmanagementsystem spezialisiert ist.

SDK Software Development Kit, kurz SDK ist eine Sammlung von Werkzeugen, welche dazu dienen eine Software zu erstellen..

Unit Test Unit Tests, auf Deutsch Komponententest, testen einzelne Komponenten des Programms. Durch Unit Tests kann das Verhalten einer Komponente analysiert werden.

Visual Studio Visual Studio ist ein Entwicklungsumgebung von Microsoft für verschiedene Programmiersprachen.

Windows 7 Weitverbreitetes Betriebssystem von Microsoft.

Windows Command Kommandozeileninterpreter von Windows, welcher für Konsolenanwendungen benutzt werden kann.

WPF Windows Presentation Foundation, kurz WPF, ist ein grafisches Framework und gehört zur .NET Familie. WPF trennt die Präsentationsicht von der Logik.

XAML E**X**tensible **A**pplication **M**arkup **L**anguage kurz XAML, ist eine Beschreibungssprache für die Entwicklung von Benutzeroberflächen. XAML wurde von Microsoft entwickelt.

XDocument XDocument ist eine Klasse innerhalb der LINQ to XML Klassenbibliothek. Sie repräsentiert ein XML Dokument.

XElement XElement ist eine Klasse innerhalb der LINQ to XML Klassenbibliothek. Sie repräsentiert ein XML Element.

XML E**X**tensible **M**arkup **L**anguage ist eine Sprache zur Darstellung hierarchisch strukturierter Daten in Form von Textdateien.

XML Schema XML Schemas definieren die Struktur von XML Dokumente. Das Schema wird in einem eigenem XML Dokument beschrieben.

XmlDocument XmlDocument ist eine Klasse innerhalb der Klassenbibliothek DOM. Sie repräsentiert ein XML Dokument.

XmlReader XmlReader ist eine Klasse innerhalb von .NET, welche ein XML Dokument sequentiell lesen kann.

XMLSerlializer Diese Klasse serialisiert und deserialisiert Objekte in und aus einem XML Dokument.

XmlWriter XmlWriter ist eine Klasse, welche ein XML Dokument sequentiell schreiben kann.

XPathNavigator XPathNavigator ist eine Klasse, welche in einem XML Dokument navigieren kann.

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

1.1 Ausgangslage

Die Firma KMU IT Management AG erstellt für jeden Kunden monatlich jeweils einen Statusrapport. Durch den Statusrapport hat der Kunde einen Einblick über den momentanen Zustand seiner Informatikstruktur. In einem Statusrapport befinden sich offene sowie geschlossene Tickets, Anzahl an Supportstunden sowie weitere kundenspezifische Informationen. Die Erstellung eines Statusrapports ist jedoch sehr aufwendig und somit dementsprechend teuer. Die benötigten Daten befinden sich in verschiedenen Datenbanken. Um den Rapport zu erstellen, müssen verschiedene Datenbanken abgefragt und die Resultate in einem Word Dokument übernommen werden.

Nach dem Einfügen der Daten muss das Word Dokument formatiert werden und für den Kunden in einem visuell attraktiven Zustand gebracht werden. Dieser Prozess wird einmal im Monat für jeden Kunden durchgeführt. Die Erstellung der Rapporte kostet dem Vorgesetzten der KMU IT Management AG pro Monat ein bis zwei Arbeitstage. Die Firma möchte diesen Prozess massiv verkürzen. Es soll eine Software entwickelt werden, mit welcher der Vorgesetzte Rapporte mit kleinem Zeitaufwand erstellen kann.

1.2 Ziele

Das Ziel der Semesterarbeit ist es einen Prototyp zu entwickeln, um diesen Prozess zu verkürzen. Der Prototyp soll aus zwei Komponenten bestehen. Die erste Komponente dient der Beschreibung des Rapports. In dieser Komponente soll der Benutzer alle Abfragen, Formatierungen sowie Texte des Rapports definieren können. Es sollen Microsoft SQL sowie MySQL Datenbankabfragen definiert werden können. Die Formatierung beinhaltet Absätze, Schriftart, Schriftfarbe, Schriftgrösse, Schriftstil, sowie die Ausrichtung des Dokuments. Für ständig gleiche Formatierungen, wie Kopf- und Fusszeile kann auch eine Word Vorlage benutzt werden.

Optional sollen auch Variablen definiert werden können. Variablen sollen für Rechnungen oder als reiner Text genutzt werden können. Die Definition des Rapports wird durch eine selbstentwickelte Benutzeroberfläche ermöglicht. Die Konfigurationsdatei wird als XML Datei gespeichert und soll auch im Nachhinein manuell angepasst werden können.

Die zweite Komponente dient der Erstellung des Rapports. In dieser Komponente wird die Konfigurationsdatei der ersten Komponente eingelesen und in ein Word Dokument umgewandelt. Diese Komponente wird in der Windows Command oder als geplanter Task aufgerufen. Der geplante Task wird vom Arbeitgeber selber erstellt. Die Performance der Software wird nicht berücksichtigt, da es nur von einer Person genutzt wird. Die meisten Rapporte werden als geplante

Tasks in der Nacht erstellt, somit ist die Laufzeit des Programms nicht relevant.

1.3 Aufgabenstellung

Anforderungen
Die Anforderungen wurden vom Geschäftsführer der KMU IT Management vollständig definiert. Die Anforderungen, welche sich hauptsächlich auf die Benutzeroberfläche und Details der Ausgabe beschränken, werden in der schriftlichen Arbeit dokumentiert.

Konzept
Vor der Umsetzung des Projektes soll ein Grobkonzept erstellt werden, welches folgendes beinhaltet:

- Projektplan
- Grobkonzept der Software
  - Klassendiagramm
  - Entwurf der Benutzeroberfläche

Das Grobkonzept soll in der Projektdokumentation einsehbar sein.

Umsetzung
Als Programmiersprache wurde C# ausgewählt, weil der Auftraggeber .NET vorgegeben hat. Während der Semesterarbeit werden vom Studenten folgende Aufgaben durchgeführt:

1. Recherche über die Microsoft Office Word Komponente in Visual Studio.
2. Recherche über Microsoft SQL und MySQL Datenbankanbindungen in C#
3. Entwurf des Projekts
4. Erstellung der Benutzeroberfläche
5. Erstellung der XML Datei durch die Benutzeroberfläche
6. Datenbank Statements aus einer XML Datei auslesen und durchführen
7. Word Automatisierung
   (a) Texte und Tabellen aus einer XML Datei auslesen und in einem Word Dokument einfügen
   (b) Ausrichtung des Dokuments
   (c) Einlesen einer Vorlage
   (d) Formatierung eines Textes
   (e) Formatierung einer Tabelle

8. Einen gesamten Rapport aus der XML Datei auslesen und erstellen
9. Dokumentation des gesamten Projekts
10. Präsentation des lauffähigen Prototyps

Fazit
In der Projektdokumentation soll ein einseitiges Fazit existieren, welches als Rückblende dienen soll.

1.4 Erwartete Resultate

Anforderungen
Die Anforderungen sind in der Projektdokumentation enthalten.

Konzept
Das Grobkonzept ist in der Projektdokumentation vorhanden.

Umsetzung
Folgende Resultate werden vom Studenten erwartet

1. Die Ergebnisse der Recherche der Word Komponente
2. Die Ergebnisse der Recherche der Datenbankanbindung
3. Eine Benutzeroberfläche für die Erstellung der Konfigurationsdatei
4. Definition des Word Dokuments wird in einer XML Datei gespeichert.
5. Datenbank Statements werden aus einer XML Datei ausgelesen und ausgeführt
6. Word Automatisierung
   (a) Texte und Tabellen werden aus einer XML Datei richtig interpretiert und in einem Word Dokument eingefügt.
   (b) Die Ausrichtung des Dokuments wird aus einer XML Datei ausgelesen und dementsprechend gesetzt
   (c) Der Speicherort einer Vorlage wird aus einer XML Datei ausgelesen und als Word Vorlage benutzt.
   (d) Die Formatierung der Texte wird aus einer XML Datei ausgelesen und in dem Word Dokument übernommen.
   (e) Die Formatierung der Tabelle wird aus einer XML Datei ausgelesen und in dem Word Dokument übernommen.
7. Der gesamte Rapport wird aus einer XML Datei ausgelesen und in einem Word Dokument gespeichert.
8. Es existiert eine Dokumentation des gesamten Projekts

9. Das Projekt wird in der Schule vorgestellt und ein lauffähiger Prototyp demonstriert.

Fazit
In der Projektdokumentation ist ein Fazit vorhanden, welches als Rückblende dient.

1.5 Sourcecode

Der Sourcecode des Prototyps wurde aus Platzgründen nicht in die Dokumentation eingefügt, jedoch kann er direkt aus dem GitHub Repository heruntergeladen werden. Das Repository befindet sich unter https://github.com/pianogen/Semesterarbeit.

1.6 Bemerkung

Die Applikation wird als Prototyp abgegeben, dadurch wird bei der Abgabe der Semesterarbeit die Software nicht vollständig getestet sein.

1.7 Änderungen

Während des Projekts wurden folgende Anforderungen geändert:

- Die Ausrichtung des Dokuments muss nicht gesetzt werden. Diese Einstellung kann durch das Einlesen der Vorlage übernommen werden
- Optionale Anforderung: Definition von mathematischen Operanden, welche berechnet und als Variable gesetzt werden können.

2 Requirements

2.1 Einleitung

Zweck
Dieser Teil des Dokuments befasst sich mit den Anforderungen des Projekts. Die Anforderungen werden mit dem Auftraggeber besprochen. Das Projekt wird freigegeben, sobald die Anforderungen vom Auftraggeber bestätigt worden sind und seiner Meinung nach alle seine Wünsche abgedeckt sind.

Systemumfang
Die Applikation wird für den internen Gebrauch entwickelt und wird von aussen nicht zugänglich sein. Somit besteht kein Bedarf, die Applikation vor unerlaubtem Zugriff zu schützen. Jegliche Art unerlaubter Zugriffe wird von der Firewall geblockt.

Auftraggeber
Der Auftraggeber dieses Projekts ist der Vorgesetzte der KMU IT Management AG. Somit wird das Projekt als internes Projekt gehandhabt. Der Vorgesetzte ist kein Entwickler und stellt deswegen keine technische Anforderungen an das Projekt. Der Vorgesetzte wird während des Projekts als Kunde betrachtet.

2.2 Allgemeine Übersicht

Systemumfeld
Die Anwendung benötigt mehrere Schnittstelle zu verschiedenen Datenbankservern.

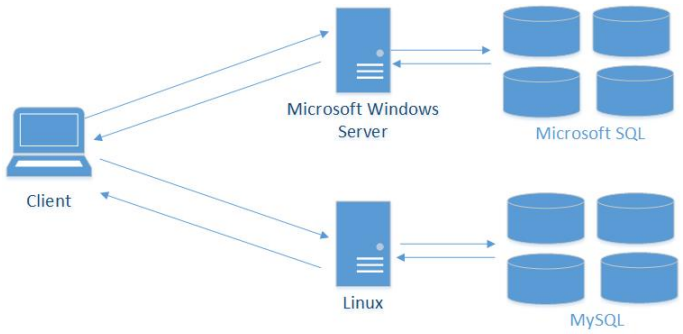


Abbildung 1: Systemumfeld

Architekturbeschreibung
Um die Applikation zu entwickeln, wird das Visual Studio 2012 Professional benutzt. Die Lizenz wird von der Firma bereitgestellt. Die benötigten Datenbankdaten sind schon zugänglich und werden von den internen Datenbankservern zur Verfügung gestellt. Entsprechende Datenbankbenutzer, um auf die Datenbanken zuzugreifen, existieren bereits. Das Visual Studio benötigt Schnittstellen um mit den Datenbankservern und der Applikation Microsoft Office Word 2010 zu kommunizieren. Die Schnittstelle zu Microsoft SQL ist im Visual Studio integriert. Die Schnittstellen zur MySQL Datenbank und Word 2010 werden manuell nach installiert.

Systemfunktionalität
Das Use Case Diagramm soll die Funktionalität des Systems aufzeigen.

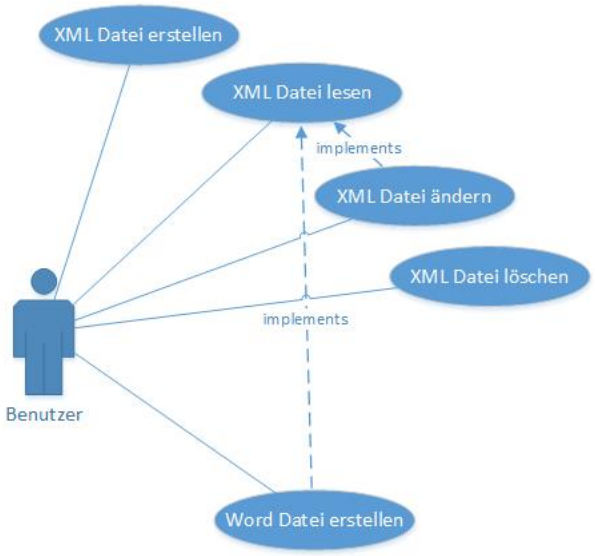


Abbildung 2: Use Case Diagramm

Nutzer und Zielgruppen
Der Nutzer ist gleichzeitig der Auftraggeber dieses Projekts. Der Auftraggeber ist der einzige Nutzer dieser Anwendung.

Annahmen
Das Resultat der Semesterarbeit ist wie bereits erwähnt ein Prototyp der Anwendung. Die Semesterarbeit beinhaltet hauptsächlich die Planung, die Umsetzung der Applikation sowie einige Tests. Weitere Tests und die finale Ausgabe der Anwendung erfolgen nach der Semesterarbeit.

2.3 Anforderungen

Allgemeine Anforderung

- Microsoft SQL Datenbanken müssen abgefragt werden können. Die Verbindungen zu den Datenbanken erfolgen, jeweils nur mit Datenbankbenutzer.
- MySQL Datenbanken müssen abgefragt werden können.
- Formatierter Text muss definiert werden können.
  - Inhalt des Textes
  - Textgrösse
  - Schriftart
  - Schrifstil: Fett, Kursiv, Unterstrichen
- Formatierter Titel muss definiert werden können.
  - Inhalt des Titels
  - Textgrösse
  - Schriftart
  - Schriftstil: Fett, Kursiv, Unterstrichen
- Formatierte Tabelle muss definiert werden können.
  - SQL Abfrage
  - Hintergrundfarbe der ersten Zeile
  - Fettgedruckte oder normal gedruckte erste Zeile
  - Ohne oder mit normalen Tabellenrand
- Variablen für die spätere Nutzung sind optional
- Allgemeine Word Einstellungen müssen definiert werden können.
  - Speicherort des Word Dokuments
  - Speicherort der Vorlage

Anforderungen zur Benutzeroberfläche

- Die Benutzeroberfläche muss zum Design von Windows 7 passen.
- Das Erstellen einer XML Datei muss über die Benutzeroberfläche möglich sein.
- Das Öffnen einer XML Datei muss über die Benutzeroberfläche möglich sein.
- Eine erstellte XML Datei muss über die Benutzeroberfläche gespeichert werden können.
- Der Inhalt einer XML Datei muss über die Benutzeroberfläche ersichtlich sein.
- Folgende Bausteine der XML Datei müssen über die Benutzeroberfläche erstellt werden können:
  - Texte
  - Titel
  - Tabellen
  - Datenbankabfragen
  - Allgemeine Einstellungen der Word Datei
- Folgende Bausteine der XML Datei können über die Benutzeroberfläche erstellt werden können:
  - Variablen
  - Mathematische Operanden

Anforderungen zur Formatierung

- Texte und Titel haben eine Standardgrösse, eine Standardschriftart sowie einen Standardschrifstil vorgegeben. Alle drei Elemente müssen jedoch veränderbar sein. Die Formatierung gilt jeweils für den gesamten Textabsatz.
- Tabellen müssen standardmässig weiss sein und einen normalen Rahmen haben. Der Rahmen darf entfernt werden. Die erste Zeile und dessen Hintergrundfarbe müssen angepasst werden können.

Anforderung zum Rapport

- Der Rapport muss im Microsoft Office Word 2010 Format erstellt werden
- Die Generierung des Rapports muss über die Command Line oder als Task ausgeführt werden können.

3 Recherche

3.1 XML

Es wird eine Komponente benötigt, um die Definition des Dokuments in eine XML Datei zu schreiben. Diese Datei muss später wiederum eingelesen werden können. Visual Studio bietet verschiedene Möglichkeiten an, welche bereits im .NET Framework implementiert sind.

3.1.1 XmlReader

XmlReader liest die Element eines XML Dokuments einzeln und der Reihen nach ein. In anderen Worten sequentiell. Die Daten sind schreibgeschützt und bleiben nicht im Arbeitsspeicher. Somit werden Speicherressourcen geschont und der Lesevorgang ist schnell. Diese Methode bietet sich an wenn eine XML Datei sequentiell gelesen und im Dokument nicht nach bestimmten Daten gesucht werden muss.

3.1.2 XmlWriter

Der XmlWriter funktioniert gleich wie der XmlReader. Der XmlReader liest ein XML Dokument und der XmlWriter schreibt ein XML Dokument.

3.1.3 XPathNavigator

XPathNavigator ermöglicht es in einer XML Datei beliebig zu navigieren. Im Gegensatz zur XmlReader Klasse ist es somit möglich rückwarts zu navigieren. Mit Hilfe eines Suchmuster wird zusätzlich ein Filtern der Datei möglich. [5] Diese Art des Lesens hat den Nachteil, dass die gesamte XML Datei zuerst in den Speicher geladen werden muss. Dadurch muss ein Verlust der Performance in Kauf genommen werden.

3.1.4 Document Object Model

Die ganze XML Datei wird in den Speicher geladen und in Form einer Baumstruktur dargestellt. Um die XML Datei zu speichern und zu lesen wird die Klasse XmlDocument genutzt. Diese Methode kann zusätzlich mit der Klasse XPathNavigator verknüpft werden, um dann die Navigationsvorteile des XPathNavigator zu nutzen. Jedoch muss durch die Verknüpfung, das XML Dokument zwei mal im Speicher abgebildet werden. [5]

3.1.5 XmlSerializer

Diese Klasse erstellt die XML Datei indem sie alle gewünschten Objekte serialisiert. Dadurch wird der jeweilige Zustand des Objekts in die XML Datei geschrieben. Um die Datei zu lesen, wird die XML Datei deserialisiert, dadurch werden die Objekte und deren Zustände dem Programm übermittelt.

3.1.6 LINQ to XML

Diese Methode ermöglicht es XML Daten zu verwalten, abzufragen und zu ändern. Um dies zu ermöglichen, wird die XML Datei in den Speicher geladen. LINQ to XML ist eine eigenständige Klassenbibliothek und ermöglicht somit das Arbeiten mit XML-Dokumenten ohne zwangsläufig LINQ Abfragen zu nutzen.

3.1.7 Entscheid

Das XML Dokument beinhaltet drei Hauptteile. Eines für die Datenbankverbindungen, eines für die Variablen und eines für die Einstellungen, Texte, Titel und Tabellen. Diese Struktur kann durch sequentielles Schreiben des XML Dokuments nicht erstellt werden. Somit fällt die Klassenbibliothek XmlWriter weg.

In dem Hauptteil für die Einstellungen, Texte, Titel und Tabellen wird wiederum das sequentielle Schreiben benötigt, da in diesem Teil des Dokuments die Reihenfolge der XML Elemente wichtig ist. Diese Anforderung kann nicht mit der Klasse XMLSerlializer erfüllt werden, ausser man würde die Text-, Titel- und Tabellenobjekte in einen gemeinsamen Container einfügen.

Bleiben noch zwei Methoden für das Schreiben des XML Dokuments. Die Klassenbibliothek Document Object Model und die Klassenbibliothek LINQ to XML. Sie erfüllen beide die Anforderungen, die benötigt werden. Die Entscheidung fällt auf LINQ to XML, da ich diese Klassenbibliothek bereits kenne und somit der kleinste Einlesebedarf nötig ist. Zusätzlich ist nach Meinung von Microsoft XML to LINQ einfacher in der Handhabung.[10]

Für das Lesen der Datei fällt die Klasse XMLSerlializer aus dem gleichen Grund wie beim Schreiben aus. Die Klasse XmlWriter fällt weg, da diese nur für das Schreiben benutzt werden kann. Die XML Datei muss sequentiell eingelesen werden, da die Reihenfolge beim Einlesen extrem wichtig ist. Der XmlReader ist eine einfache und effiziente Methode, um Dokumente mit gleicher Struktur sequentiell einzulesen.[10] Somit wird die XML Datei mit dem XmlReader ausgelesen.

3.2 Datenbankanbindung

Es braucht jeweils eine Schnittstelle für die Microsoft SQL Datenbankanbindung und eine für die MySQL Datenbankanbindung. Für die Microsoft SQL Datenbankanbindung wird die native Schnittstelle, die schon im Visual Studio 2012 implementiert ist, benutzt. Für die MySQL Datenbankanbindung wird die Schnittstelle von Oracle selbst benutzt. Diese Schnittstelle muss installiert und in das Visual Studio implementiert werden.

3.3 Word Schnittstelle

Es wird eine Schnittstelle benötigt, damit das Visual Studio eine Word Datei generieren kann. Nach einer Recherche im Internet wurden zwei mögliche Schnittstellen in die engere Wahl genommen.

3.3.1 Microsoft.Office.Interop.Word

Die Interop Schnittstelle ist die Office Schnittstelle von Microsoft für die Programmiersprache C#.

3.3.2 Open XML SDK 2.0

Open XML ist ein Standard, welcher XML Schemas für Tabellen, Präsentationen und Textdokumente definiert. Word 2010 benutzt Open XML als Standarddateiformat. Das Open XML SDK 2.0 für Microsoft Office bietet stark typisierte Klassen für die Manipulation von Open XML Dokumente an. Die SDK benutzt zusätzlich LINQ um stark typisierten Objektzugriff auf den XML Inhalt im Dokument anzubieten.

3.3.3 Entscheid

Tabelle 1: Vergleich der Word Schnittstellen

	Interop	Open XML
Text und Tabellen mutieren	x	x
Dokument ausrichten	x	x
Vorlage einlesen	x	x
Text formatieren	x	x
Tabelle formatieren	x	x

Tabelle 1 zeigt, dass beide Varianten, die Bedingungen um alle Anforderungen in das Projekt zu implementieren, erfüllen.

Weitere Recherchen im Internet haben ergeben, dass die Open XML SDK der Interop Schnittstelle zu bevorzugen ist. [9] Die Open XML SDK ist performancemässig schneller und benötigt für die gleichen Aufgaben weniger Zeilen. Ein weiterer Punkt für OpenXML ist, dass Microsoft Office Word 2010 auf Open XML basieren. Somit wird für dieses Projekt die Open XML SDK benutzt.

4 Konzept

4.1 Entwurf

4.1.1 XML Datei

Der gewünschte Rapport wird in einer XML Datei beschrieben. Die XML Datei benutzt ISO-8859-1[1] als Encoding, dadurch sind Umlaute in der XML Datei erlaubt.

Die XML Datei wird folgende Struktur annehmen:

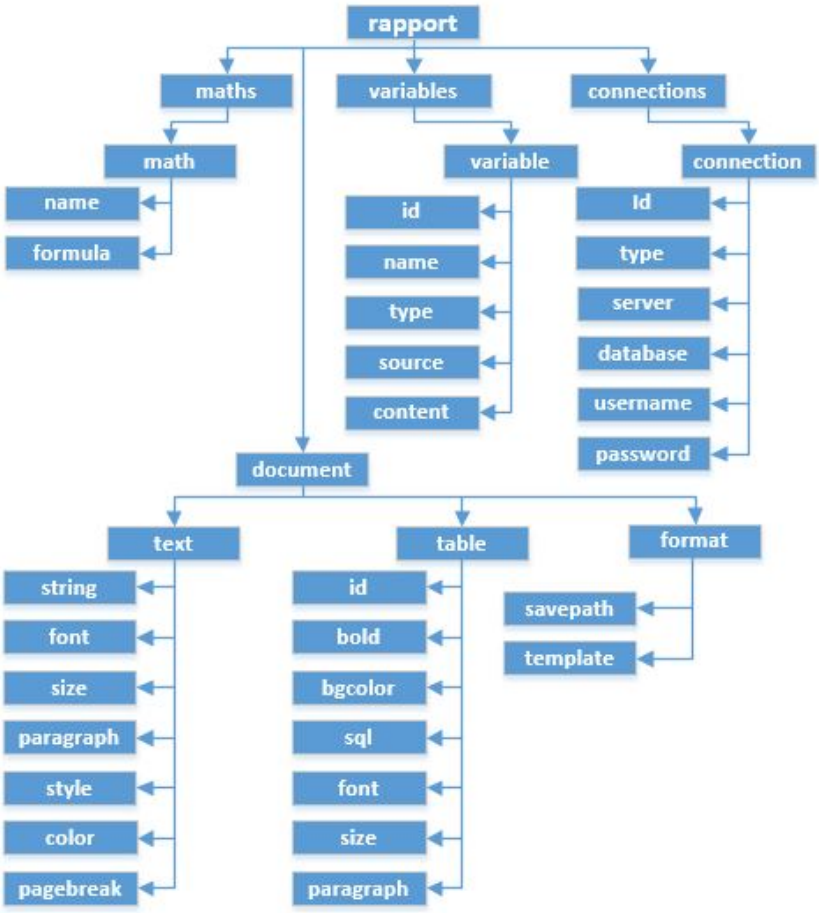


Abbildung 3: Beschreibung des Rapports in XML

[1] Ein Standard für die Zeichencodierung mit acht Bit

Allgemein
Das XML Dokument hat immer die genau gleiche Struktur. Die Struktur des XML Dokuments sieht folgendermassen aus:

```
<rapport>
        <connections>
                Alle Unterknoten dieses Knotens
        </connections>
        <variables>
                Alle Unterknoten dieses Knotens
        </variables>
        <maths >
                Alle Unterknoten dieses Knotens
        </maths>
        <document>
                <format>
                        Alle weiteren Unterelemente von diesem Element
                </format>
                Alle weiteren Unterelemente von diesem Element
        </document>
</rapport>
```

Durch diese Struktur ist die Datei für den Benutzer einfacher zu lesen. Teilweise ist diese Reihenfolge ein Muss. Ein Grund für dieses Muss ist, dass zuerst die Datenbankverbindungen ausgewertet werden müssen um eine Variable oder Tabelle auszuwerten. Durch diese Reihenfolge werden zuerst alle Datenbankverbindungsobjekte instanziiert. Danach alle Variablen ausgewertet, womöglich benötigt man dafür eine Datenbankverbindung. Als nächstes werden die mathematischen Formeln ausgewertet, dafür werden womöglich wiederum Variablen benötigt. Zum Schluss werden die eigentlichen Informationen für das Microsoft Office Word Dokument verarbeitet. Das wären Text, Tabelle und Allgemeine Einstellungen.

Das Element Format ist das einzige Element der zweiten Stufe, welches schon zu Beginn der Programminitialisierung gesetzt wird. Somit erzwingt man, dass vor dem Einfügen eines Textes, das Dokument initialisiert wird.

Connection
Alle relevanten Datenbankverbindungsinformationen befinden sich in diesem Element.

- id
  Der Blattknoten id beinhaltet die Identifikation der Datenbankverbindung. Der Inhalt is eine beliebige Zeichenfolge.

- type
  Dieser Blattknoten definiert den Schnittstellentyp der Datenbankverbindung. Die Schnittstelle kann entweder ein Microsoft SQL Server oder ein MySQL Server sein.

- server
  Dieser Blattknoten beinhaltet die IP Adresse des Datenbankservers und womöglich den Instanznamen des Servers. Die IP Adresse[2] wird im IPv4[3] aufgenommen. Der Inhalt dieses Elements ist entweder eine IP Adresse oder eine IP Adresse und eine Instanz[4].

- database
  Der Blattknoten database definiert den Namen der Datenbank mit der man sich verbinden möchte.

- username
  Dieser Blattknoten beinhaltet den Datenbankbenutzernamen. Der Benutzer muss bezogen auf die Datenbank leseberechtigt sein.

- password
  Dieser Blattknoten beinhaltet das Passwort des Datenbankbenutzers. Das Passwort wird in Klartext gespeichert. Diese Sicherheitslücke wird in Kauf genommen, da nur der Vorgesetzte der Firma Zugriff auf die XML Dateien hat.

[2] Die IP Adresse ist die Netzwerkkennung des Kommunikationsgerät im Netz
[3] Standardprotokoll für IP Adressen
[4] Eine Instanz ist eine eigene Installation eines SQL Servers. Ein Server kann mehrere Instanzen haben

Variable
Dieses Element beinhaltet eine Variable, die in einem Titel oder einem Text eingesetzt werden kann.

- id
  Dieser Blattknoten beinhaltet die Identifikation einer Datenbankverbindung. Die Identifikation muss vorhanden sein, falls die Variable aus einer Datenbank abgefragt wird.

- name
  Der Blattknoten name beinhaltet den Variablennamen. Der Variablenname kann aus Zahlen oder Buchstaben bestehen.

- type
  Dieser Blattknoten beinhaltet den Datentyp der Variable. Der Typ kann entweder numerisch(int) oder alphabetisch(string) sein.

- source
  Der Blattknoten source definiert die Quelle der Variable. Eine Variable kann entweder ein fester Wert oder ein Resultat aus einer Datenbankabfrage sein.

- content
  Dieser Blattknoten beinhaltet den Inhalt der Variable. Der Inhalt dieses Elements kann entweder ein fester Wert oder eine Datenbankabfrage sein.

Math
Dieses Element wird benutzt, falls mit festen Zahlen oder Variablen gerechnet werden will. Das Resultat ist eine neue Variable.

- name
  Dieser Blattknoten definiert den Namen der neuen Variable. Der Name kann aus Zahlen oder Buchstaben bestehen.

- formula
  Dieser Blattknoten beinhaltet die Rechnungsformel für die neue Variable. Der Inhalt dieses Elements können definierte Variablen oder Zahlen sein.

Format
Alle allgemeine Einstellungen für das Dokument werden in diesem Element definiert.

- save
  Dieser Blattknoten beinhaltet den gesamten Speicherpfad des Word Dokuments.
- template
  In diesem Blattknoten befindet sich der Pfad der Vorlage des Word Dokuments.

Text
Dieses Element beinhaltet einen formatierten Text oder einen formatierten Titel. Auf dieser Ebene gibt es keinen Unterschied zwischen einem Titel und einem Text, somit wurden diese zwei Bausteine zusammengeführt.

- string
  Dieser Blattknoten definiert den Inhalt des Textes.
- font
  Der Blattknoten font definiert die Schriftart des Textes. Die Schriftart muss in Microsoft Office Word 2010 vorhanden sein.
- size
  Dieser Blattknoten definiert die Schriftgrösse des Textes. Der Inhalt dieses Knotens muss eine Zahl sein.
- paragraph
  Dieser Blattknoten definiert die Absatzgrösse nach dem Text. Dieser Inhalt muss ebenfalls eine Zahl sein
- style
  Der Blattknoten style definiert den Schriftstil des Textes, es wird zwischen normal, fett, kursiv und unterstrichen unterschieden.
- color
  Der Blattknoten color definiert die Schriftfarbe des Textes. Die Schriftfarbe muss in Microsoft Office Word 2010 vorhanden sein.
- pageBreak
  Dieser Blattknoten definiert, ob der Inhalt auf einer neuen Seite eingefügt werden soll.

Table
Dieses Element beinhaltet eine formatierte Tabelle.

- id
  Dieser Blattknoten beinhaltet die Identifikation der Datenbankverbindung. Die Identifikation muss als Datenbankverbindung aufgenommen worden sein.

- bold
  Der Blattknoten bold definiert, ob die erste Zeile der Tabelle fett geschrieben sein muss. Der Inhalt dieses Knoten darf nur wahr oder falsch sein.

- background
  Dieser Blattknoten definiert die Hintergrundfarbe der Tabelle. Die Hintergrundfarbe muss in Microsoft Office Word 2010 vorhanden sein.

- sql
  Der Blattknoten sql beinhaltet die SQL Abfrage, um die Tabelle mit den gewünschten Daten zu füllen. Die SQL Abfrage muss von der SQL Syntax her korrekt sein.

- font
  Der Blattknoten font definiert die Schriftart der Tabelle. Die Schriftart muss in Microsoft Office Word 2010 vorhanden sein.

- size
  Dieser Blattknoten definiert die Schriftgrösse der Tabelle. Der Inhalt dieses Elements muss eine Zahl sein.

- paragraph
  Dieser Blattknoten definiert die Absatzgrösse nach dem der Tabelle. Dieser Inhalt muss ebenfalls eine Zahl sein

4.1.2 Benutzeroberfläche

Hauptfenster
Die XML Datei wird über eine selbst erstellte Benutzeroberfläche kreiert. Der Prototyp der Benutzeroberfläche besteht aus einem Hauptfenster und mehreren Unterfenster, welche über Interaktionsknöpfe geöffnet werden können. Im Hauptfenster befinden sich die Zugänge zu den Unterfenster und eine Vorschau der XML Datei. Sobald das Programm gestartet wird, enthält die Vorschau der XML Datei bereits alle XML Elemente der ersten Stufe und das XML Element format aus der zweiten Stufe. Im Hauptfenster kann die XML Datei gespeichert werden oder eine schon vorhandene Datei geöffnet werden. Der Knopf Rapport ermöglicht es, das Microsoft Office Word Dokument direkt aus dem Inhalt der Textbox zu erstellen.

Abbildung 4: Entwurf der Benutzeroberfläche

Unterfenster
Die Unterfenster werden benötigt, um die Elemente der zweiten Stufe, sowie all ihre Kindselemente zu definieren. Sobald die Eingaben im Unterfenster bestätigt werden schliesst sich das Fenster und die Vorschau der XML Datei aktualisiert sich mit den eben eingegeben Daten. Das Fenster für Titel und Text wurde auf dem untenstehenden Bild aus Platzgründen zusammengeführt. Die Unterschiede zwischen den Bausteinen Text und Titel sind die Standardwerte und die Option New Page.

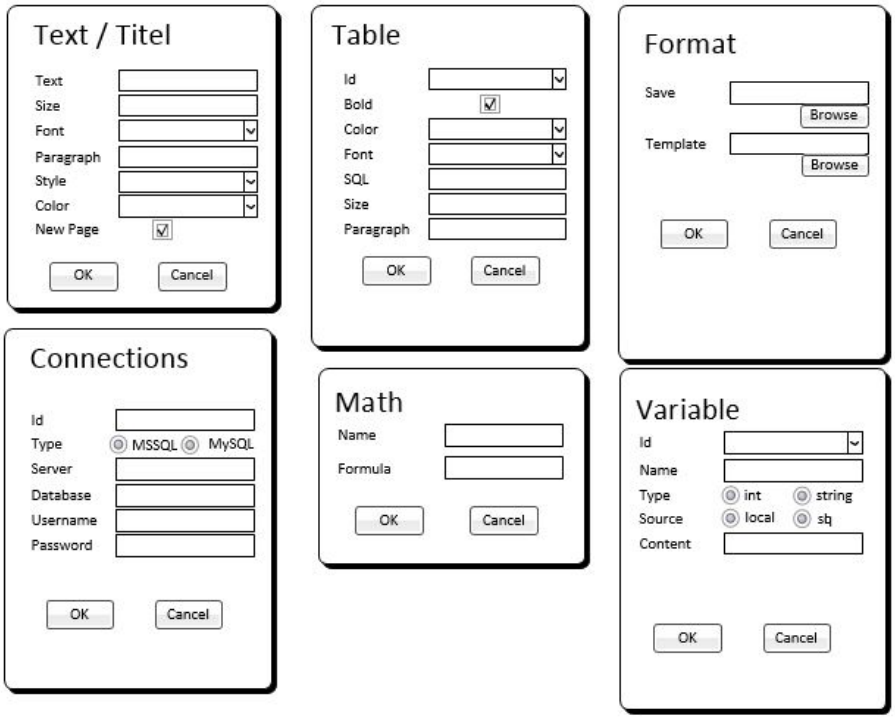


Abbildung 5: Entwurf der Unterfenster

4.2 Softwarearchitektur

Der Prototyp wird auf Basis von Visual Studio 2012 und der Programmiersprache C# entwickelt. Das Programm wird in zwei Teilprojekte eingeteilt. In einem Teilprojekt wird die Erstellung der XML Datei realisiert. Wie im Kapitel 4.1 bereits erwähnt, wird die XML Datei mit Hilfe einer Benutzeroberfläche erstellt. Die Benutzeroberfläche ermöglicht es, eine XML Datei zu erstellen ohne die gesamte XML Theorie zu kennen. Im anderen Teilprojekt wird die erstellte XML Datei verarbeitet und in einem Microsoft Office Word Dokument ausgegeben. Dieses Teilprojekt ist eine Konsolenanwendung. Als Parameter wird die gewünschte XML Datei mitgegeben. Klassen werden teilweise mit Absicht falsch benannt, um Verwechslungen mit bestehenden Klassen zu vermeiden.

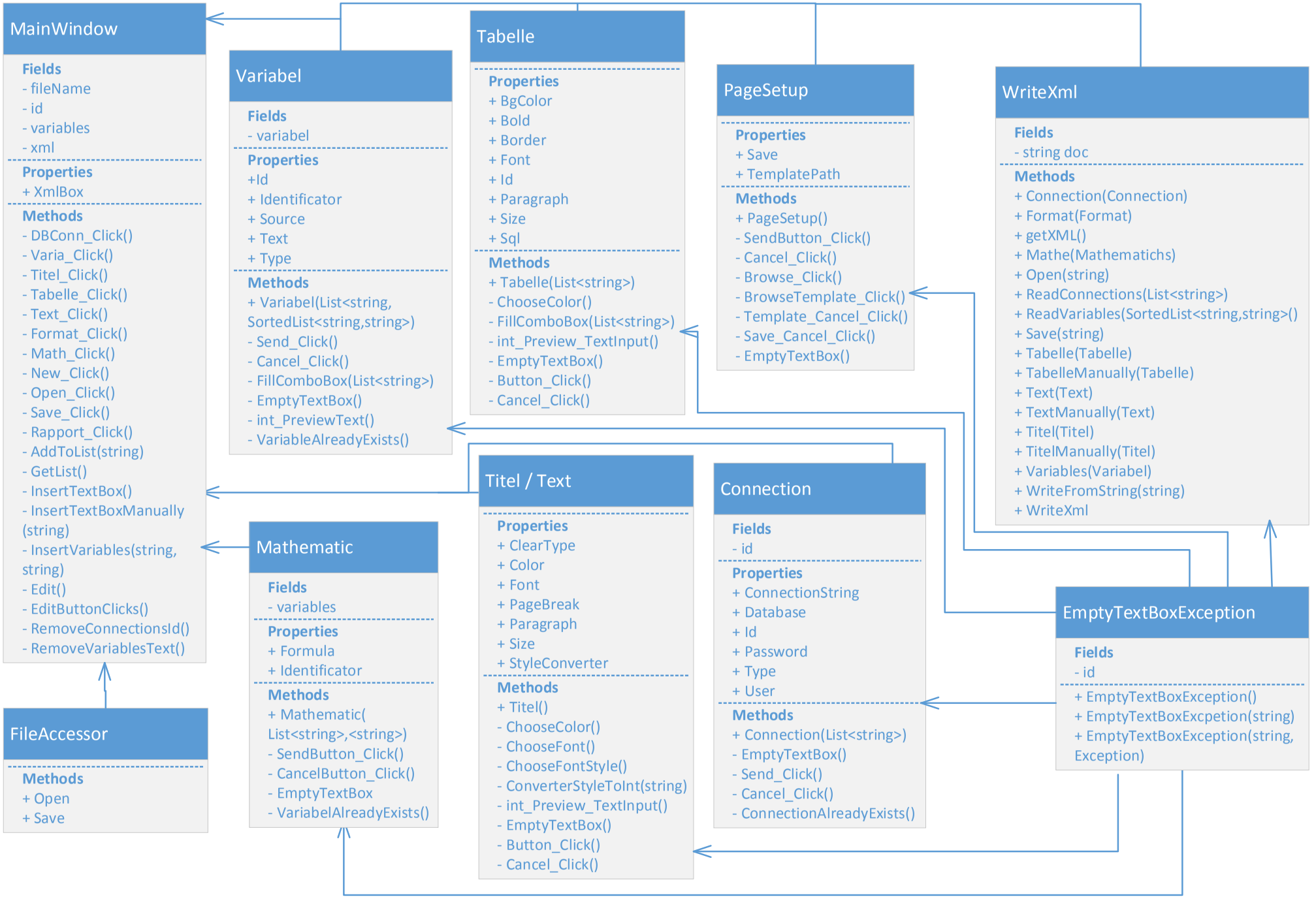
4.2.1 Projekt: XML Generierung

Dieses Projekt wird als WPF Applikation erstellt. Die grafischen Elemente werden mit XAML realisiert, die Logik dahinter mit C#. Auf den nächsten Seiten wird auf die Architektur der Logik dieses Projekts eingegangen, sowie ein Klassendiagramm der Logik aufgezeigt. Auf die grafischen Elemente wird nicht gross eingegangen, da diese mit Hilfe von Visual Studio und XAML realisiert wurden.

Dieses Projekt hat eine Referenz auf das zweite Projekt. Dadurch kann der Microsoft Office Word Rapport direkt über die Benutzeroberfläche erstellt werden. Das Programm bietet auf Wunsch des Arbeitgebers einen manuellen Modus. Im manuellen Modus kann das XML Dokument manuell verändert werden. Wird die Datei in diesem Modus unachtsam geändert, kann dies die XML Struktur verändern und die Ausgabe des Microsoft Office Word Dokuments verfälschen. Die Sql Eingaben werden in diesem Projekt nicht überprüft. Somit können XML Dateien auch im Offline Modus generiert werden.

MainWindow
Durch diesen Baustein wird dieses Projekt gestartet. Sie ist die zentrale Einheit des Programms. Alle anderen Instanzen werden durch diese Klasse aufgerufen. Diese Klasse behandelt alle Knöpfe und die Textbox mit dem Inhalt des XML Rapports im Hauptfenster des Programmes. Die Knöpfe Save und Report werden nicht aktiviert, solange die allgemeinen Einstellungen des Word Dokuments nicht definiert sind. Sobald die Einstellungen definiert sind, werden die Knöpfe aktiviert und der Knopf PageSetup deaktiviert.

FileAccessor
Dieser Baustein behandelt den Zugriff auf das Windows Dateisystem, um Dateien einzulesen und zu speichern. Bei einem Speicher- oder Ladefehler wird eine Exception geworfen.


MainWindow
Fields
- fileName
- id
- variables
- xml
Properties
+ XmlBox
Methods
- DBConn_Click()
- Varia_Click()
- Titel_Click()
- Tabelle_Click()
- Text_Click()
- Format_Click()
- Math_Click()
- New_Click()
- Open_Click()
- Save_Click()
- Rapport_Click()
- AddToList(string)
- GetList()
- InsertTextBox()
- InsertTextBoxManually (string)
- InsertVariables(string, string)
- Edit()
- EditButtonClicks()
- RemoveConnectionsId()
- RemoveVariablesText()
Variabel
Fields
- variabel
Properties
+Id
+ Identificator
+ Source
+ Text
+ Type
Methods
+ Variabel(List<string, SortedList<string,string>)
- Send_Click()
- Cancel_Click()
- FillComboBox(List<string>)
- EmptyTextBox()
- int_PreviewText()
- VariableAlreadyExists()
Tabelle
Properties
+ BgColor
+ Bold
+ Border
+ Font
+ Id
+ Paragraph
+ Size
+ Sql
Methods
+ Tabelle(List<string>)
- ChooseColor()
- FillComboBox(List<string>)
- int_Preview_TextInput()
- EmptyTextBox()
- Button_Click()
- Cancel_Click()
PageSetup
Properties
+ Save
+ TemplatePath
Methods
+ PageSetup()
- SendButton_Click()
- Cancel_Click()
- Browse_Click()
- BrowseTemplate_Click()
- Template_Cancel_Click()
- Save_Cancel_Click()
- EmptyTextBox()
WriteXml
Fields
- string doc
Methods
+ Connection(Connection)
+ Format(Format)
+ getXML()
+ Mathe(Mathematichs)
+ Open(string)
+ ReadConnections(List<string>)
+ ReadVariables(SortedList<string,string>()
+ Save(string)
+ Tabelle(Tabelle)
+ TabelleManually(Tabelle)
+ Text(Text)
+ TextManually(Text)
+ Titel(Titel)
+ TitelManually(Titel)
+ Variables(Variabel)
+ WriteFromString(string)
+ WriteXml
Titel / Text
Properties
+ ClearType
+ Color
+ Font
+ PageBreak
+ Paragraph
+ Size
+ StyleConverter
Methods
+ Titel()
- ChooseColor()
- ChooseFont()
- ChooseFontStyle()
- ConverterStyleToInt(string)
- int_Preview_TextInput()
- EmptyTextBox()
- Button_Click()
- Cancel_Click()
Connection
Fields
- id
Properties
+ ConnectionString
+ Database
+ Id
+ Password
+ Type
+ User
Methods
+ Connection(List<string>)
- EmptyTextBox()
- Send_Click()
- Cancel_Click()
- ConnectionAlreadyExists()
Mathematic
Fields
- variables
Properties
+ Formula
+ Identificator
Methods
+ Mathematic(List<string>,<string>)
- SendButton_Click()
- CancelButton_Click()
- EmptyTextBox
- VariabelAlreadyExists()
EmptyTextBoxException
Fields
- id
+ EmptyTextBoxException()
+ EmptyTextBoxExcpetion(string)
+ EmptyTextBoxException(string, Exception)
FileAccessor
Methods
+ Open
+ Save

WriteXml
Dieser Baustein erstellt und bearbeitet das XML Objekt. Alle XML Aufgaben in dieser Klasse werden mit den Klassen XDocument und XElement durchgeführt. Sobald das Programm gestartet wird, wird das XML Objekt mit der ersten Stufe der XML Struktur befüllt. Eine der folgenden Methoden wird aufgerufen, sobald ein Element der zweiten Stufe, durch die Benutzeroberfläche erstellt wird:

- Connection
- Format
- Mathe
- Tabelle
- Text
- Titel
- Variable

Diese Methoden erstellen mit Hilfe der Klasse XElement ein Element der zweiten Stufe. Der Ablauf ist jeweils identisch. Es wird ein neues Unterelement in das entsprechende Elternelement erstellt. Dieses XML Element hat als Unterelement alle eingegebenen Parameter, welche der Benutzer eingegeben hat. Die Elemente werden jeweils am unteren Ende des Elternknoten gesetzt.

Im manuellen Modus ist es möglich Texte, Tabellen oder Titel an einer beliebigen Stelle einzufügen. Für diese Art des Einfügens sind folgende Methoden zuständig:

- TextManually
- TitelManually
- TabelleManually

Diese Methoden unterscheiden sich nur geringfügig von den oberen Methoden. Anstatt das Element am unteren Ende des Knotens zu setzen, werden Sie an der gewünschten Stelle eingefügt. Der Benutzer gibt die gewünschte Stelle mit dem Cursor in der TextBox an.

Die Methode **WriteFromString** wird nur im manuellen Modus aufgerufen. Diese Methode speichert den gesamten Inhalt der Textbox direkt in das XML Objekt. Falls sich unzulässige Daten im Inhalt befinden, wird eine Exception geworfen. Die Reihenfolge und die Stelle worin sich die Elemente befinden, wird nicht überprüft. Dies könnte man mit einem XML Schema abfangen.

Die Methode **ReadConnection** und **ReadVariables** werden beim Öffnen einer vorhandenen XML Datei aufgerufen. Sie lesen alle vorhandenen Datenbankverbindungen und Variablen der Datei ein und schreiben Sie in die entsprechende Objekte.

Die Methode **getXml** wandelt das gesamte XML Objekt in einen Text um. Diese Methode wird benötigt, um das XML Objekt in der Textbox anzuzeigen.

Die Methode **Save** speichert das XML Objekt als XML Datei in das Windows Dateisystem ab. Die Methode **Open** ladet eine XML Datei aus dem Windows Dateisystem in das XML Objekt.

Alle Unterfenster
Zu dieser Gruppe gehören folgende Klassen:

- Mathematics
- Titel
- PageSetup
- Text
- Tabelle
- Connection
- Variabel

Auf diese Klassen wird nicht einzeln eingegangen, da die Logik relativ ähnlich ist. Die Parameter, die der Benutzer eingibt, werden als Properties zwischengespeichert. Die weiteren Methoden dieser Klassen werden jeweils für folgendes benötigt:

- Um Fenster zu schliessen und Parameter zu speichern.
- Um Fenster zu schliessen und Parameter nicht zu speichern.
- Um die Gültigkeit und Vollständigkeit der eingegebenen Parameter zu überprüfen.
- Um die Auswahlboxen mit vordefinierten Werten zu füllen.

EmptyTextBoxException
Das ist eine selbstimplementierte Exception. Diese Exception wird aufgerufen, falls die Gültigkeit oder die Vollständigkeit der eingegebenen Parameter nicht gewährleistet ist.

4.2.2 Projekt: Word Generierung

Dieses Projekt ist ein reines C# Projekt. Die Hauptaufgabe dieses Prototyps ist es, die XML Datei sequentiell abzuarbeiten. Währenddessen wird das Microsoft Office Word Dokument erstellt. Die Abarbeitung läuft völlig automatisch ab, die einzige Benutzereingabe die benötigt wird, ist der Pfad der XML Datei.

Program
Dieser Baustein ruft das gesamte Programm auf. Es überprüft, ob die eingegebene Datei existiert. Falls die Datei vorhanden ist, wird ein Objekt der Klasse ImportInWord instanziert und mit dem Pfad der XML Datei dem Baustein ReadXml weitergegeben. Falls die Datei nicht vorhanden ist wird das Programm gestoppt.

ReadXml
Dieser Baustein verarbeitet die gesamte XML Datei und ist somit der Mittelpunkt der Applikation. Die XML Datei wird mit Hilfe der XmlReader Klasse verarbeitet. Als erstes werden alle Datenbankverbindungen ausgelesen und gespeichert. Alle Informationen der Datenbankverbindungen werden in einer Liste gespeichert.

Nach der Verarbeitung der Datenbankverbindungen werden alle Variablen verarbeitet. Die Variablen werden ebenfalls in einer Liste gespeichert. Als drittes werden alle mathematischen Operatoren verarbeitet, diese dürfen erst nach den Variablen verarbeitet werden, da ein Operator womöglich eine Variable ist. Der Name und das Resultat der mathematischen Operation wird ebenfalls in der Liste der Variablen aufgenommen.

Nach der Verarbeitung der mathematischen Formel beginnt die Verarbeitung des letzten Teils und somit die Erstellung des Microsoft Office Word Dokuments. Im letzten Teil wird zuerst die Formatierung verarbeitet und direkt im Microsoft Office Word Dokument übernommen. Nach der Formatierung werden die Tabellen und Texte verarbeitet und direkt im Microsoft Office Word Dokument übernommen. Bei einem Laufzeitfehler wird das Programm mit einer passenden Fehlermeldung gestoppt. Falls das Microsoft Office Word Dokument vor dem Laufzeitfehler erstellt wurde, wird es gelöscht.

PageSetup
In diesem Baustein werden die allgemeinen Einstellungen zwischengespeichert.

Text
In diesem Baustein werden die Parameter eines Textes zwischengespeichert.

Connection
In diesem Baustein werden die Parameter einer Datenbankverbindung zwischengespeichert.

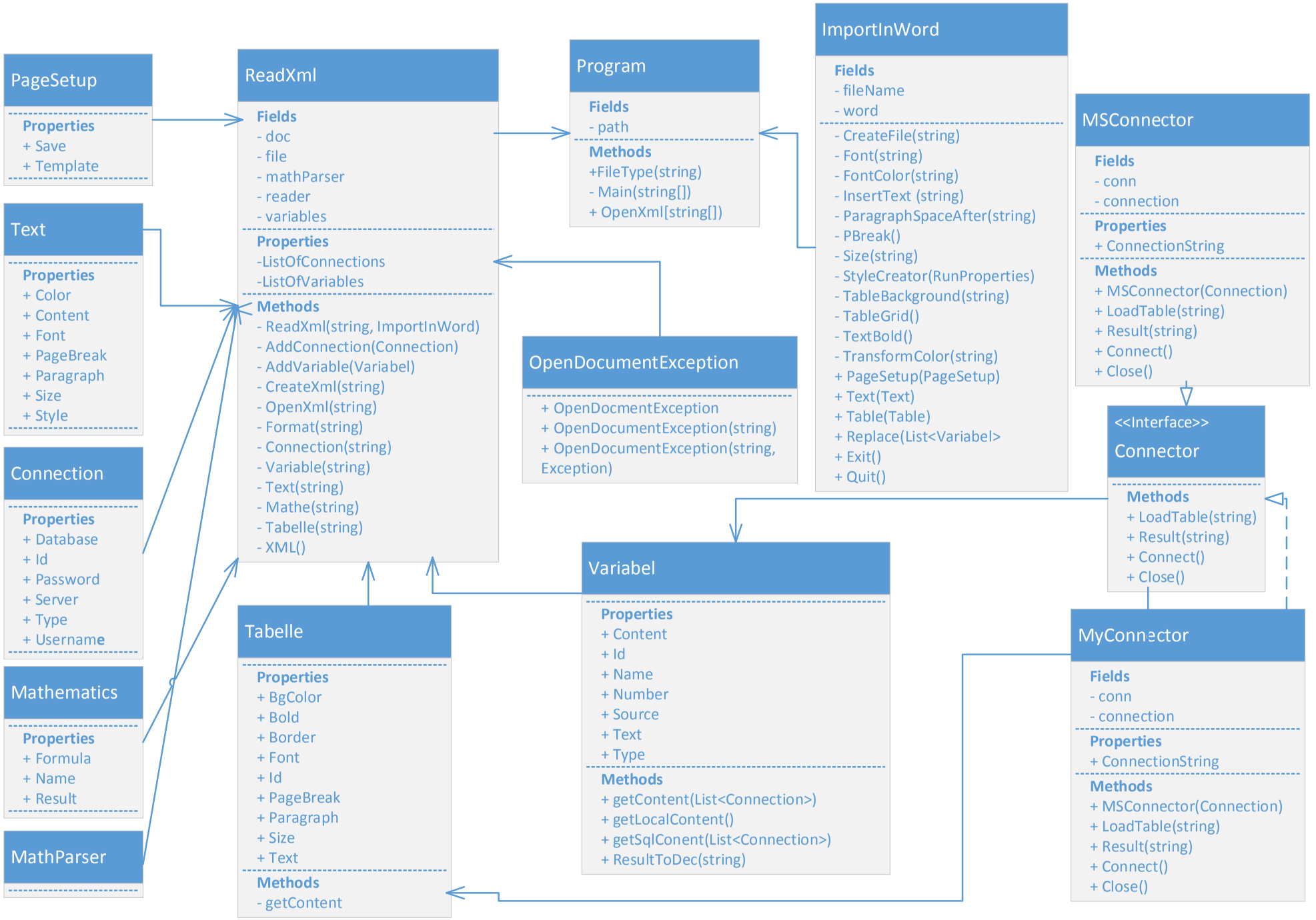

PageSetup
Properties
+ Save
+ Template
ReadXml
Fields
- doc
- file
- mathParser
- reader
- variables
Properties
-ListOfConnections
-ListOfVariables
Methods
- ReadXml(string, ImportInWord)
- AddConnection(Connection)
- AddVariable(Variabel)
- CreateXml(string)
- OpenXml(string)
- Format(string)
- Connection(string)
- Variable(string)
- Text(string)
- Mathe(string)
- Tabelle(string)
- XML()
Program
Fields
- path
Methods
+FileType(string)
- Main(string[])
+ OpenXml[string[])
ImportInWord
Fields
- fileName
- word
- CreateFile(string)
- Font(string)
- FontColor(string)
- InsertText (string)
- ParagraphSpaceAfter(string)
- PBreak()
- Size(string)
- StyleCreator(RunProperties)
- TableBackground(string)
- TableGrid()
- TextBold()
- TransformColor(string)
+ PageSetup(PageSetup)
+ Text(Text)
+ Table(Table)
+ Replace(List<Variabel>
+ Exit()
+ Quit()
MSConnector
Fields
- conn
- connection
Properties
+ ConnectionString
Methods
+ MSConnector(Connection)
+ LoadTable(string)
+ Result(string)
+ Connect()
+ Close()
Text
Properties
+ Color
+ Content
+ Font
+ PageBreak
+ Paragraph
+ Size
+ Style
OpenDocumentException
+ OpenDocmentException
+ OpenDocumentException(string)
+ OpenDocumentException(string, Exception)
<<Interface>>
Connector
Methods
+ LoadTable(string)
+ Result(string)
+ Connect()
+ Close()
Connection
Properties
+ Database
+ Id
+ Password
+ Server
+ Type
+ Username
Variabel
Properties
+ Content
+ Id
+ Name
+ Number
+ Source
+ Text
+ Type
Methods
+ getContent(List<Connection>)
+ getLocalContent()
+ getSqlConent(List<Connection>)
+ ResultToDec(string)
MyConnector
Fields
- conn
- connection
Properties
+ ConnectionString
Methods
+ MSConnector(Connection)
+ LoadTable(string)
+ Result(string)
+ Connect()
+ Close()
Mathematics
Properties
+ Formula
+ Name
+ Result
Tabelle
Properties
+ BgColor
+ Bold
+ Border
+ Font
+ Id
+ PageBreak
+ Paragraph
+ Size
+ Text
Methods
- getContent
MathParser

Tabelle
In diesem Baustein werden die Parameter einer Tabelle zwischengespeichert. Das Resultat der Abfrage wird ebenfalls zwischengespeichert.

Variable
In diesem Baustein werden die Parameter einer Variable zwischengespeichert. Der ausgewertete Wert der Variable wird ebenfalls gespeichert.

Mathematics
Dieser Baustein dient zur Zwischenspeicherung einer mathematischen Formel. Der Name und das Resultat werden ebenfalls zwischengespeichert.

MathParser
Dieser Baustein wird aus dritter Hand übernommen. Er berechnet das Resultat der mathematischen Formel. Dieser Baustein war keine Anforderung des Projekts und wurde in Absprache mit meinem Arbeitgeber und dem Dozenten aus dem Internet eins zu eins übernommen[8]. Der Baustein wird so angepasst, das alle Zahlvariablen und ihre Werte erkannt werden. Dadurch lässt sich mit den gesetzten Variablen rechnen.

Connector, MSConnector und MyConnector
Die beiden Bausteine MSConnector und MyConnector implementieren beide das Interface Connector. Die zwei Bausteine machen genau dasselbe. Sie verbinden sich mit einer SQL Datenbank und fragen bestimmte Werte oder Wertbereiche ab. Der Unterschied der beiden Bausteine ist, dass die eine Klasse eine Microsoft SQL Datenbank abfragen kann und die andere eine MySQL Datenbank.

OpenDocumentException
Dieser Baustein ist eine selbstgeschrieben Exception. Sie wird nur geworfen, falls beim Auftreten des Laufzeitfehlers ein Teil des Word Dokuments bereits erstellt wurde. Diese Exception wird benötigt, um das nicht fertiggestellte und somit fehlerhafte Microsoft Office Word Dokument zu löschen.

ImportInWord
Dieser Baustein ist für die Erstellung und Formatierung des Microsoft Office Word Dokuments zuständig. Das Microsoft Office Word Dokument wird mit Hilfe der Open XML SDK erstellt. Stark wiederholende Mechanismen werden in separate Methoden aufgenommen. Sobald die XML Datei fertig verarbeitet ist, wird das gesamte Microsoft Office Word Dokument nach Variablen durchsucht und die wirklichen Werte eingefügt. Zum Schluss werden alle Änderungen gespeichert und die Schnittstelle zu Microsoft Office Word geschlossen.

Die Methode **CreateFile** kopiert die Vorlage, speichert die Kopie mit dem gewünschten Namen ab und ändert das Format von einer Vorlage zu einem Dokument. Falls dieser Prozess fehlschlägt, wird eine Excpetion abgefangen und das Programm gestoppt.

Die Methode **Font** definiert die Schriftart des Textes. Die Methode **FontColor** definiert die Schriftfarbe des Textes. Die Methode **InsertText** fügt den Text ein.

Die Methode **ParagraphSpaceAfter** definiert den Abstand nach dem Paragraph. Die Methode **PBreak** setzt den Inhalt auf einer neuen Seite. Die Methode **StyleCreator** definiert den Schriftstil des Textblocks.

Die Methode **TableBackground** setzt die Hintergrundfarbe der ersten Zeile einer Tabelle. Die Methode **TableGrid** definiert den Rahmen einer Tabelle.

Die Methode **TextBold** setzt den Text der ersten Zeile einer Tabelle fett. Die Methode **TransformColor** wandelt den Namen der Farbe in einen hexadezimalen Code um.

Die Methode **Exit** speichert das Dokument ab. Die Methode **Quit** löscht das erstellte Dokument.

5 Umsetzung

5.1 Installation

Für die Realisierung dieses Projekts wurden ein paar Programme installiert. Auf dem Arbeitsnotebook wurde Visual Studio 2012 Professional Edition installiert. Diese kostenpflichtige Applikation wurde vom Arbeitgeber bereits gekauft. Das Microsoft Office Word Dokument wird, wie bereits erwähnt, mit der Open XML 2.0 SDK erstellt. Somit wurde das Open XML 2.0 SDK heruntergeladen und installiert. Dasselbe gilt für die MySQL Anbindung. Der MySQL Connector wurde heruntergeladen und installiert.

Für Testzwecke sowie für die Demonstration des Prototyps wurde auf dem Arbeitsnotebook ein Microsoft SQL Server und ein MySQL Server installiert.

5.2 Implementierung

5.2.1 XML Generierung

Als erstes wurde das Grundgerüst der grafischen Benutzeroberfläche erstellt. Das Hauptfenster mit all seinen Elementen sowie die Unterfenster mit all ihren Elementen. Das Layout der einzelnen Elemente wurde in einer separaten Datei implementiert.

Nachdem die Benutzeroberfläche erstellt wurde, wurde der Speicher- und Ladevorgang der XML Datei über das Windows Dateisystem implementiert. Danach wurde die Zwischenspeicherung der Parameter, sowie deren Verknüpfungen implementiert. Zum Schluss wurde die Behandlung des XML Objekts implementiert.

5.2.2 Word Dokument Generierung

Als erstes wurde die Startkomponente des Prototyps erstellt. Diese Komponente überprüft, ob die angegebene Datei eine XML Datei ist. Als zweites wurde die Microsoft Office Word Komponente realisiert, diese Komponente sollte möglichst früh fertig sein, da die Open XML SDK die unbekannteste Komponente ist, und somit am meisten Probleme verursachen kann.

Nach der Fertigstellung der Word Komponente wurde die Verarbeitung der XML Datei entwickelt. Zum Schluss wurden die Bausteine zur Zwischenspeicherung der Parameter und die Datenbankanbindung entwickelt.

Die Exception Behandlung ist nicht optimal gelöst. Es gibt zwei verschiedene Situationen, welche behandelt werden müssen. Wird eine Exception vor der Generierung des Dokuments abgefangen, wird der Benutzer darüber informiert und das Programm automatisch beendet. Wird eine Exception während der Generierung des Dokuments abgefangen, wird der Benutzer darüber informiert, das fehlerhafte Word Dokument gelöscht und das Programm automatisch beendet. Durch diese Abhandlung werden keine fehlerhaften Dokumente gespeichert, jedoch sind die Fehlermeldungen nicht immer eindeutig. Die Exception Behandlung wird nach der Semesterarbeit überarbeitet.

Während dem Einlesen der XML Datei wird nicht überprüft, ob ein Element einen leeren Inhalt hat. Diese Überprüfung wurde weggelassen, da im ersten Teil des Projekts ein Element mit leeren Parameter nicht gespeichert werden kann. Bei einer manuellen Bearbeitung des XML Dokuments kann jedoch ein Element mit leerem Inhalt erstellt werden. Diese Überprüfung wird nach der Semesterarbeit implementiert, falls der Kunde dies für nötig hält.

6 Testkonzept

6.1 Teststrategie

Aus Zeitgründen wird nur das Projekt 'Word Generierung' getestet. Das Testen des Projekts 'XML Generierung' fällt während der Semesterarbeit weg, weil das Testen einer WPF Applikation sehr zeitaufwändig sein kann. Die Benutzeroberfläche wurde während der Realisierung visuell überprüft. Die Logik dahinter wurde während der Entwicklung getestet. Ein komplettes Testkonzept ist für dieses Projekt jedoch nicht vorhanden. Die Tests des Projekts 'Word Generierung' werden in drei verschiedene Arten unterteilt:

- Unit Test
- Integrationstest
- Systemtest

6.2 Testwerkzeug

Als Testwerkzeug dienen die integrierten Testkomponenten von Visual Studio. Die automatische Erstellung von Tests wurde in Visual Studio 2012 von Microsoft deaktiviert. Diese Funktion wird aktiviert um eine gute Vorlage für die bevorstehenden Tests zu haben.

6.3 Unit Tests

Diese Tests testen einzelne Methoden. Aus Zeitgründen wurden nicht alle Klassen komplett getestet. Die Klassen ReadXml und ImportInWord werden aus Zeitgründen nicht während des Zeitrahmens der Semesterarbeit getestet. Die Klasse MathParser wurde ebenfalls nicht getestet, da der Code dieser Klasse aus dem Internet übernommen wurde. Auf den folgenden Tabellen wurden jeweils die verschiedenen Tests aufgelistet und das Resultat dokumentiert. In der ersten Spalte befindet sich, welche Methode getestet wird. In der zweiten Spalten steht was getestet wird. Die dritte Spalte zeigt in der ersten Zeile das erwartete Resultat an. Die zweite Zeile zeigt, ob der Test erfolgreich abgeschlossen wurde.

Tabelle 2: Testklasse Connection

Methode	**Was**	**Resultat**
Id	Schreib- und Lesezugriff der Property	OK Test erfolgreich
Server	Schreib- und Lesezugriff der Property	OK Test erfolgreich
Type	Schreib- und Lesezugriff der Property	OK Test erfolgreich
Database	Schreib- und Lesezugriff der Property	OK Test erfolgreich
Username	Schreib- und Lesezugriff der Property	OK Test erfolgreich
Password	Schreib- und Lesezugriff der Property	OK Test erfolgreich

Tabelle 3: Testklasse Mathematics

Methode	**Was**	**Resultat**
Formula	Schreib- und Lesezugriff der Property	OK Test erfolgreich
Name	Schreib- und Lesezugriff der Property	OK Test erfolgreich
Resultat	Schreib- und Lesezugriff der Property	OK Test erfolgreich

Tabelle 4: Testklasse PageSetup

Methode	**Was**	**Resultat**
Save	Schreib- und Lesezugriff der Property	OK Test erfolgreich
Template	Schreib- und Lesezugriff der Property	OK Test erfolgreich

Tabelle 5: Testklasse Tabelle

Methode	**Was**	**Resultat**
BgColor	Schreib- und Lesezugriff der Property	OK Test erfolgreich
Bold	Schreib- und Lesezugriff der Property	OK Test erfolgreich
Border	Schreib- und Lesezugriff der Property	OK Test erfolgreich
Content	Schreib- und Lesezugriff der Property	OK Test erfolgreich
Id	Schreib- und Lesezugriff der Property	OK Test erfolgreich
Text	Schreib- und Lesezugriff der Property	OK Test erfolgreich
getContent	Datenbank wird abgefragt und Resultat in eine DataTable gespeichert	Resultat wird gespeichert Test erfolgreich

Tabelle 6: Testklasse Program

Methode	**Was**	**Resultat**
OpenXMLTest	Gültiger Pfad wird eingeben	Variable wird nicht leer sein Test erfolgreich
FileTypeTest	Xml Datei wird versucht einzulesen	Ok Test erfolgreich
FailedFileTypeTest	Word Datei wird versucht einzulesen	Es wird false zurückgegeben Test erfolgreich

Tabelle 7: Testklasse Variabel

Methode	**Was**	**Resultat**
ResultToDec	Wandelt Text in Zahl um	OK Test erfolgreich
ResultToDecFail	Wandelt Text in Zahl um	Fehlgeschlagen Test erfolgreich
Content	Schreib- und Lesezugriff der Property	OK Test erfolgreich
Id	Schreib- und Lesezugriff der Property	OK Test erfolgreich
Name	Schreib- und Lesezugriff der Property	OK Test erfolgreich
Number	Schreib- und Lesezugriff der Property	OK Test erfolgreich
Source	Schreib- und Lesezugriff der Property	OK Test erfolgreich
Text	Schreib- und Lesezugriff der Property	OK Test erfolgreich
Type	Schreib- und Lesezugriff der Property	OK Test erfolgreich
getContentLocal	Lokale Textvariable wird übeprüft	Wert wird in Content geschrieben Test erfolgreich
getSqlContent	Einzelner Wert wird aus Datenbank gelesen	Wert wird in Content geschrieben Test erfolgreich
getIntLocal	Lokale numerische Variable wird überprüft	Wert wird Number geschrieben Test erfolgreich
getIntSql	Einzelner numerischer Wert wird aus Datenbank gelesen	Wert wird in Number geschrieben Test erfolgreich

Tabelle 8: Testklasse MSConnector und MyConnector

Methode	**Was**	**Resultat**
ConnectionString	Schreib- und Lesezugriff der Property	OK Test erfolgreich
Result	Einzelner Wert wird aus Datenbank abgefragt	Ok Test erfolgreich
loadTable	Wertebereich wird aus Datenbank abgefragt	OK Test erfolgreich
Connect	Verbindung zur Datenbank	OK Test erfolgreich
FailedConnection	Verbindung mit ungültigem Server	Exception wird geworfen Test erfolgreich
FailedDatabaseConnection	Verbindung mit ungültiger Datenbank	Exception wird geworfen Test erfolgreich
FailedUsernameorPW	Verbindung mit ungültigem Benutzername	Exception wird geworfen Test erfolgreich
FailedQueryResult	Ungültige Abfrage eines Wert	Exception wird geworfen Test erfolgreich
FailedLoadTable	Ungültige Abfrage eines Wertbereichs	Exception wird geworfen Test erfolgreich

6.4 Integrationstest

Die Integrationstests koppeln die Methoden der Klassen ReadXml, ImportInWord und der Word Objektklassen zusammen und überprüfen, ob das Zusammenspiel der Komponenten funktioniert. Um diese Tests durchzuführen, erstellt man für jeden Test ein individuelles XML Objekt. Das XML Objekt beinhaltet jeweils die Komponente die man testen möchte. Dadurch entstehen sieben verschiedene XML Objekte. Der Inhalt der XML Objekte sieht folgendermassen aus:

1. XML Objekt mit einer Datenbankverbindung
2. XML Objekt mit einer lokalen Variable und den allgemeinen Word Einstellungen
3. XML Objekt mit einer mathematischen Formel und den allgemeinen Word Einstellungen
4. XML Objekt mit den allgemeinen Word Einstellungen
5. XML Objekt mit den allgemeinen Word Einstellungen und einem Text
6. XML Objekt mit den allgemeinen Word Einstellungen, einer Tabelle und einer Datenbankverbindung
7. XML Objekt mit unterschiedlich gesetzten Objekten

Abgesehen von den Tests der Datenbankverbindungen brauchen alle anderen Tests die Parameter für die allgemeinen Word Einstellungen. Die Einstellungen werden benötigt, weil diese Einstellungen das Microsoft Office Dokument initialisieren.

6.4.1 Resultat

XML Objekt mit einer gesetzten Verbindung
Dieser Test überprüft, ob eine Datenbankverbindung aus dem XML Dokument ausgelesen und in ein 'Connection' Objekt zwischengespeichert wird. Der Test verläuft positiv, falls sich die ID der Datenbankverbindung in der Liste der gesetzten Datenbankverbindung befindet. Dieser Test verlief positiv.

XML Objekt mit einer gesetzten Variable
Dieser Test überprüft, ob eine Variable aus dem XML Dokument ausgelesen wird und in ein 'Variabel' Objekt zwischengespeichert wird. Um den Test einfach zu halten ist die getestete Variabel eine lokale Textvariable. Dieser Test verläuft positiv, falls sich der Variablenname in der Liste der gesetzten Variablen befindet. Dieser Test verlief positiv.

XML Objekt mit einer gesetzten mathematischen Formel
Dieser Test überprüft, ob eine mathematische Variable aus dem XML Objekt

ausgelesen, berechnet und in ein 'Variabel' Objekt zwischengespeichert wird. Der Test verläuft positiv, falls sich das erwartete Resultat der mathematischen Formel in der Liste der gesetzten Variablen befindet. Dieser Test verlief positiv.

XML Objekt mit gesetzten Einstellungen des Word Dokuments
Dieser Test überprüft, ob die allgemeinen Einstellungen aus dem XML Objekt ausgelesen und auf das Microsoft Office Word Dokument übernommen werden. Dieser Test verläuft positiv, falls das gewünschte Microsoft Office Word Dokument existiert und die Einstellungen übernommen wurden. Um die Einstellungen des Microsoft Office Word Dokuments zu prüfen, muss das eben erstellte Word Dokument geöffnet und überprüft werden. Dieser Test verlief positiv.

XML Objekt mit gesetztem Text
Dieser Test überprüft, ob ein Text aus dem XML Objekt ausgelesen und in das Microsoft Office Word Dokument eingefügt wird. Dieser Test verläuft positiv, falls das gewünschte Microsoft Office Word Dokument erstellt wurde und sich der Text im Dokument befindet. Um den Textblock zu überprüfen, muss das eben erstellte Dokument geöffnet werden. Dieser Test verlief positiv.

XML Objekt mit gesetzter Tabelle
Dieser Test überprüft, ob eine Tabelle aus dem XML Objekt ausgelesen und in das Microsoft Office Word Dokument eingefügt wird. Dieser Test verläuft positiv, falls das gewünschte Microsoft Office Word Dokument erstellt wurde und sich die Tabelle im Dokument befindet. Um die Tabelle zu überprüfen, muss das eben erstellte Dokument geöffnet werden. Dieser Test verlief positiv.

XML Dokument mit unterschiedlich gesetzten Objekten
Dieser Test überprüft, ob das gesamte Zusammenspiel der verschiedenen Komponenten funktioniert. Folgende Komponenten werden getestet:

- Zwei verschiedene Datenbankverbindungen
- Vier verschiedene Variablen:
  - Eine lokale Textvariable
  - Eine lokale Zahl
  - Eine datenbankspezifische Textvariable
  - Eine datenbankspezifische Zahl
- Zwei verschiedene mathematische Operanden:
  - Eine Formel mit lokalen Werten
  - Eine Formel mit datenbankspezifischen Werten
- Verschiedene Texte
- Zwei Tabellen

Dieser Test verläuft positiv, falls ein Microsoft Office Word Dokument erstellt wird und sich alle definierten Komponenten im Microsoft Office Word Dokument befinden. Dieser Test verlief positiv.

6.5 Systemtest

Der Systemtest wird genutzt um zu überprüfen, ob alle Anforderungen erfüllt wurden. Bei dem Systemtest wird die Applikation aus den Augen des Auftraggebers betrachtet und dadurch entschieden, ob die Anforderung erfüllt wurde oder nicht.

Microsoft SQL Datenbanken müssen abgefragt werden können
Diese Anforderung wurde erfüllt. Über eine Microsoft SQL Datenbank können einzelne Werte oder gesamte Wertbereiche abgefragt werden.

MySQL Datenbank müssen abgefragt werden können
Diese Anforderung wurde erfüllt. Über eine MySQL können einzelne Werte oder gesamte Wertbereiche abgefragt werden.

Ein formatierter Text muss definiert werden können
Diese Anforderung wurde erfüllt. Der formatierte Text wird korrekt in das Microsoft Office Word Dokument eingefügt.

Ein formatierter Titel muss definiert werden können
Diese Anforderung wurde erfüllt. Der formatierte Titel wird korrekt in das Microsoft Office Word Dokument eingefügt.

Eine formatierte Tabelle muss definiert werden können
Diese Anforderung wurde erfüllt. Die formatierte Tabelle wird korrekt in das Microsoft Office Word Dokument eingefügt.

Variablen für die spätere Nutzung sind optional
Diese optionale Anforderung wurde erfüllt. Bei der Generierung der XML Datei können Variablen definiert werden. Die Variablen im Text werden während der Erstellung des Microsoft Office Word Dokuments mit ihren Werten ersetzt. Eine Variable wird in einem Text zwischen zwei %-Zeichen gesetzt.

Die allgemeine Word Einstellungen müssen definiert werden können
Diese Anforderung wurde erfüllt. Bei der Generierung der XML Datei kann eine Vorlage angegeben werden, welche die allgemeinen Einstellungen des Word Dokuments beinhaltet.

Die Benutzeroberfläche muss zum Design von Windows 7 passen
Diese Anforderung wurde erfüllt. Die Benutzeroberfläche passt sich standardmässig der Windows Version an. Dies hat den Vorteil, dass die Benutzeroberfläche unter Windows 8, das Windows 8 Design übernimmt.

Das Erstellen einer XML Datei muss über die Benutzeroberfläche möglich sein
Diese Anforderung wurde erfüllt. Die Benutzeroberfläche ermöglicht es, die Beschreibung des Word Dokuments in einer XML Datei zu erstellen.

Das Öffnen einer XML Datei muss über die Benutzeroberfläche möglich sein
Diese Anforderung wurde erfüllt. Die Benutzeroberfläche ermöglicht es, ein zuvor erstelltes XML Dokument zu öffnen und zu bearbeiten.

Eine erstelle XML Datei muss über die Benutzeroberfläche gespeichert werden können
Diese Anforderung wurde erfüllt. Die Benutzeroberfläche ermöglicht es, ein XML Dokument an einem gewünschten Ort im Windows Dateisystem zu speichern.

Der Inhalt einer XML Datei muss in der Benutzeroberfläche ersichtlich sein
Diese Anforderung wurde erfüllt. Das aktuelle XML Dokument ist jederzeit über die Benutzeroberfläche ersichtlich.

Bausteine
Diese Anforderung wurde erfüllt. Es können alle gewünschten Bausteine über das XML Dokument definiert werden.

Formatierungsmöglichkeiten von Text und Titel
Diese Anforderung wurde erfüllt. Texte und Titel haben eine Standardgrösse, sowie eine Standardschriftart und einen Standardschriftstil vorgegeben. Alle drei Eigenschaften lassen sich beliebig ändern. Zusätzlich kann die Absatzgrösse nach dem Text oder dem Titel definiert werden. Beim Definieren eines neuen Titels kann dieser auf einer neuen Seite erstellt werden.

Formatierungseigenschaften einer Tabelle
Diese Anforderung wurde erfüllt. Die Hintergrundfarbe einer Tabelle ist standardmässig weiss. Die Tabelle hat einen normalen Rahmen. Der Rahmen kann entfernt werden. Die erste Zeile und dessen Hintergrundfarbe kann angepasst werden. Die erste Zeile kann entweder fett geschrieben sein oder normal.

Word 2010 Format
Diese Anforderung wurde erfüllt. Der Rapport wird im Microsoft Office Word 2010 Format erstellt.

Generierung des Rapports über Command Line oder als geplanter Task
Diese Anforderung wurde erfüllt. Der Rapport kann direkt aus der Command Line oder über die Aufgabenplanung von Windows 7 erstellt werden.

7 Fazit

7.1 Review

Das Projekt wurde vollständig realisiert und der Prototyp kann in einer ersten Testphase produktiv genutzt werden. Das Projekt war relativ schwierig einzuschätzen, da ich noch nie ein Programm in dieser Grösse realisiert habe und Softwareentwicklung nicht meinem Schwerpunkt entspricht. Die Grundkenntnisse von Visual Studio und C# habe ich letztes Jahr in einem Modul erlangt. Die tiefere Kenntnisse habe ich mir während der Semesterarbeit zu eigen gemacht.

Die Gestaltung der Benutzeroberfläche ist dank den vorhandenen Tools in Visual Studio relativ simpel. Trotzdem musste ich immer wieder achten nicht zu viel Zeit in kleine Details zu verlieren. Die Logik der Benutzeroberfläche habe zeitlich komplett unterschätzt. Vorallem die Generierung des XML Objekts hat mehr Zeit in Anspruch genommen als gedacht. Ich war sehr erstaunt, was es da alles für Möglichkeiten gibt. Beim Testen der Benutzeroberfläche fiel mir auf, dass es nützlich wäre, eine geöffnetes Dokument unter einem anderen Namen zu speichern. Diese Option wird nach der Semesterarbeit implementiert.

Bei der Generierung des Microsoft Office Word Dokuments hatte ich erstaunlicherweise keine grossen Überraschungen mit der Open XML Komponente. Dies liegt wahrscheinlich daran, dass diese Klassenbibliothek sehr gut auf der Microsoft Seite dokumentiert ist. Der Absatzabstand nach der Tabelle ist nicht implementiert worden, da dieser nicht funktioniert hat. Diese Option war jedoch keine Anforderung des Vorgesetzten. Bei diesem Teilprojekt war die grösste Schwierigkeit das Einlesen der XML Datei. Für diesen Schritt habe ich mich erstmals mit allen möglichen Methoden auseinandergesetzt und mich danach für eine andere Methode als bei der Generierung der XML Datei entschieden. Die Schwierigkeit dieser Aufgabe war, die grosse XML Datei einzulesen und das ganze mit einem strukturierten Programmablauf zu verarbeiten. Meine Entscheidung das XML Dokument in vier Hauptteile zu unterteilen hat mir dabei sehr geholfen. Die Anbindung an die Datenbanken war relativ einfach, da mir die Anbindung an Datenbanken über C# in der Vorlesung des letztjährigen Moduls beigebracht wurde.

Das Testkonzept hat sich komplizierter herausgestellt als gedacht, da viele Methoden weitere Methoden aufrufen. Meiner Ansicht nach wird das Testen von Komponenten dadurch kompliziert.

Als Abschluss habe ich einige generierte Dateien mit dem Open XML SDK Tool validiert. Die Validierung zeigt an, dass Eigenschaften der Tabellenränder nicht stimmen. Sie werden aber trotzdem übernommen. Dieses Problem wird nach Abschluss der Semesterarbeit genauer betrachtet.

Insgesamt betrachtet bin ich sehr zufrieden mit meiner Arbeit. Es wurde alles

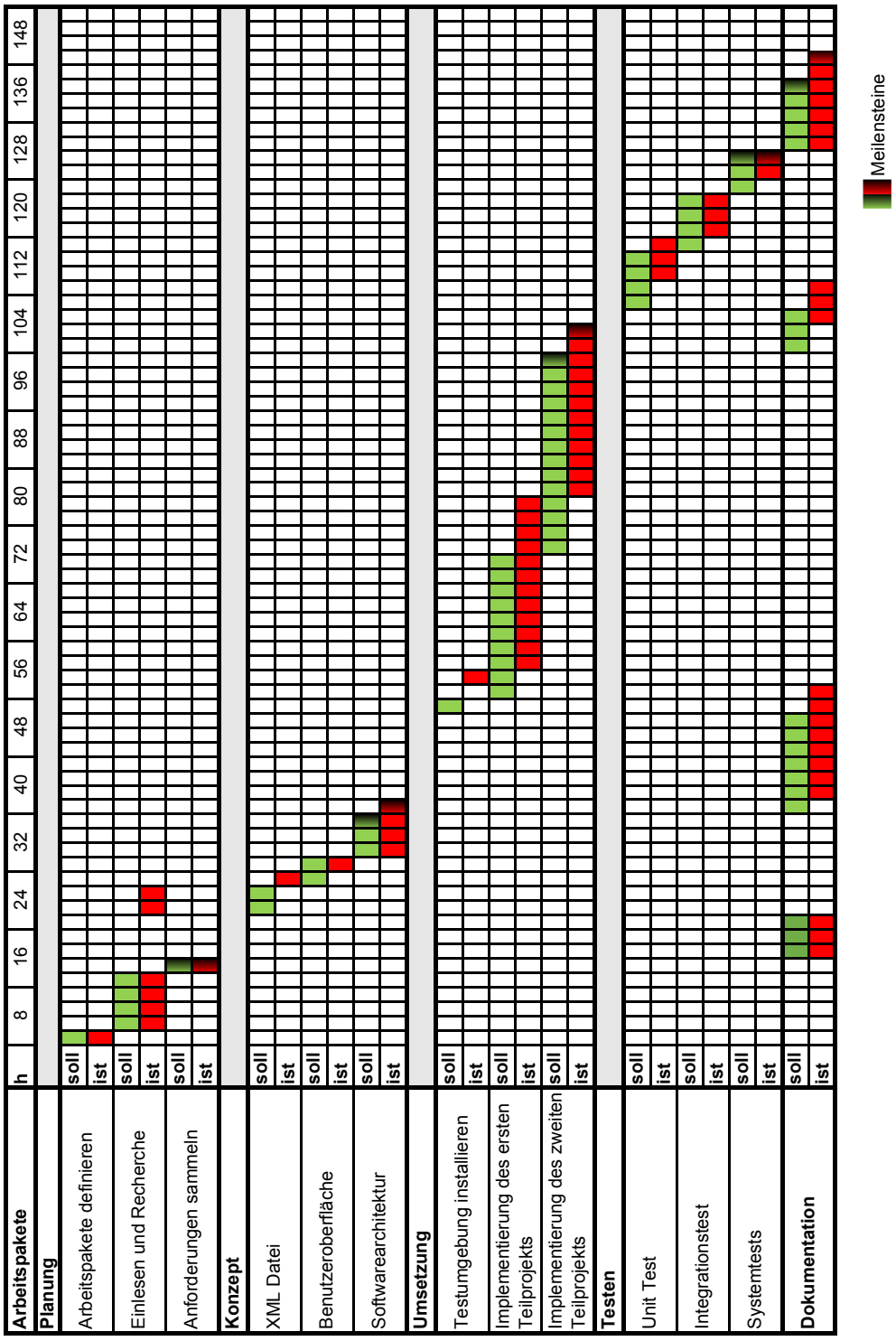
nötige implementiert und die Generierung des Word Dokuments funktioniert einwandfrei. Sehr wahrscheinlich kann man verschiedene Teilaufgaben einfacher implementieren, jedoch hatte ich keine Zeit für weitere Recherchen.

7.2 Ausblick

Es stehen noch folgende offene Arbeiten aus:

- XML Schema implementieren, um die Struktur des XML Dokuments zu gewährleisten
- Gesamtes Testkonzept beider Teilprojekte
- Abnahmegespräch mit Vorgesetzten
- Benutzerdefinierte Kopf- und Fusszeile, sowie Ausrichtung des Dokuments
- Exceptionbehandlung verbessern
- Validierung des Open XML SDK Tool genauer betrachten
- Save As Button in der Benutzeroberfläche einführen
- Absatzabstand nach einer Tabelle einfügen

8 Projektplan



Literatur

[1] Klaus Pohl / Chris Rupp (Juni 2011): *Basiswissen Requirements Engineering*

[2] Mahbouba Gharbi / Arne Koschel / Andreas Rausch / Gernot Starke (Dezember 2012): *Basiswissen für Softwarearchitekten*

[3] Andreas Spillner / Tilo Linz (September 2012): *Basiswissen Softwaretest*

[4] DocumentFormat.OpenXml.WordProcessing-Namespace
http://msdn.microsoft.com/de-de/library/office/
documentformat.openxml.wordprocessing(v=office.15).aspx

[5] Galileo OpenBook
http://openbook.galileocomputing.de/visual_csharp_2010/visual_csharp_2010_16_005.htm

[6] System.Xml.Linq Namespace
http://msdn.microsoft.com/en-us/library/system.xml.linq(v=vs.110).aspx

[7] XmlReader Class
http://msdn.microsoft.com/en-us/library/system.xml.xmlreader.aspx

[8] MathParser Class
http://www.codeproject.com/Tips/381509/Math-Parser-NET-Csharp

[9] Vorteile von Open XML gegenüber Microsoft.Office.Interop
http://sharepoint2010master.blogspot.co.uk/2011/12/advantages-of-
openxml-over-interop-for.html

[10] LINQ to XML http://msdn.microsoft.com/en-us/library/bb387044.aspx