Hochschule für Technik
und Wirtschaft Berlin

*University of Applied Sciences*

# BACHELORARBEIT

## Implementierung einer plattformunabhängigen Testreport Generators

von Christoph Franke

Fachbereich 1
Ingenieurwissenschaften - Energie und Information
Computer Engineering

Hochschule für Technik und Wirtschaft Berlin

University of Applied Sciences

swissbit®

Dokumentation

# Implementierung einer plattformunabhängigen Testreport Generators

Christoph Franke

*1. Betreuende Prüfer* Prof. Thomas Baar
Fachbereich 1
Hochschule für Technik und Wirtschaft

*2. Betreuende Prüfer* Dr. -Ing. Marco Beyer
Head of Test Development
Swissbit Germany AG

29. Juni 2017

**Christoph Franke**
*Implementierung einer plattformunabhängigen Testreport Generators*
Dokumentation, 29. Juni 2017
Betreuende Prüfer: Prof. Thomas Baar und Dr. -Ing. Marco Beyer

**Hochschule für Technik und Wirtschaft**
Fachbereich 1
Wilhelminenhofstraße 75A
12459 Berlin

# Sperrvermerk

Die vorliegende Thesis mit dem Titel *„Implementierung eines plattformunabhängigen Testreport Generators“* enthält vertrauliche Informationen der Swissbit Germany AG.

Veröffentlichung, Vervielfältigung, Weitergabe – auch auszugsweise – sowie das Einstellen in eine nicht- bzw. öffentliche Bibliothek sind ohne schriftliche Genehmigung der Swissbit Germany AG nicht gestattet. Ausgenommen von dieser Zustimmungserfordernis ist die Verwendung, die aufgrund einer Prüfungs- oder Studienordnung zur Bewertung der Thesis erfolgt.

Berlin, den 20.07.2017

(Christoph Franke)

# Abkürzungsverzeichnis

**API** Application Programming Interface

**ASCII** American Standard Code for Information

**COM** Component Object Model

**CSS** Cascading Style Sheets

**CFSDMK** Compact-Flash-Card SD-Card Manufacturing-Kit

**DLL** Dynamic Link Library

**EULA** End User License Agreement

**GPL** General Public License

**GUI** Graphical User Interface

**HTML** Hyper Text Markup Language

**IDE** Integrated Development Enviroment

**IO** Input and Output

**JS** Java Script

**LGPL** GNU Lesser General Public License

**LTM** Life Time Monitoring

**PA** Produktionsauftrag

**QML** Qt Meta Language oder Qt Modeling Language

**SQL** Structured Query Language

**SVG** Scalable Vector Graphics

**SGML** Standard Generalized Markup Language

**UML** Unified Modeling Language

**XHTML** Extensible Hypertext Markup Language

**XML** Extensible Markup Language

**WWW** World Wide Web

# Inhaltsverzeichnis

# 1 Einleitung

Diese Bachelorarbeit beschreibt die Entwicklung eines plattformunabhängigen Testreport Generators. Als Marktführer in Europa für industrielle Speichermedien ist es der Swissbit AG wichtig die eigene Produktpalette stetig zu verbessern und zu erweitern. Je nach Produkt und Kunde werden verschiedene Testszenarien durchlaufen. Diese verschiedenen Tests speichern unterschiedlich große Mengen an Informationen persistent in der FlashDB-Datenbank. Mithilfe dieser Daten kann eine genaue Entwicklungs- und Prozessplanung erfolgen, desweitern ist es möglich bei auftretenden Problemen diese zu lokalisieren und nachzuvollziehen. Für eine benutzerfreundliche Darstellung dieser Informationen existiert eine Webseite, welche mithilfe eines Interfaces auf die Datenbank zugreift.

## 1.1 Firmenvorstellung

"Die Swissbit AG, der größte unabhängige Speichermedienhersteller in Europa, gründete sich durch ein Management-Buy-Out von Siemens Halbleiter im Jahre 2001. Mit über 20 Jahren Erfahrung in der Speicherindustrie wurde Swissbit weltweit führend in Technologie-, hohe Qualitäts- und hohe Verlässigkeitsansprüchen für Speicherlösungen in allen hergestellten DRAM und Flash Interfaces."[**SwissbitAbout**] Der Hauptsitz befindet sich in Bronschhofen, Schweiz. Weitere Niederlassungen finden sich in Deutschland, USA, Japan und Taiwan.Die Produktion beschränkt sich ausschließlich auf den Standort Berlin, wobei Forschung und Entwicklung in Deutschland, Schweiz und USA stattfinden. Weltweit beschäftigt die Swissbit AG 250 Mitarbeiter.
Die Produktpalette reicht von DRAM Modulen über (micro-)SD Karten bis zu SSD Festplatten und umfasst 140 verschiedene Produkte. Um auf verschiedenen Gebieten höchsten Anforderungen gerecht zu werden, gibt es strategische Lieferantenpartnerschaften mit führenden Halbleiterherstellern wie zum Beispiel Toshiba oder Hyperstone welche Wafer Support und gemeinse Entwicklungsprogramme beinhalten.

## 1.2 Motivation

Für die Entwicklung eines neuen Produktes oder einer neuen Produktgruppe ist es unter anderem Notwendig unterschiedliche Tests mit den Muster- oder Entwicklungsteilen durchzuführen. Die relevanten Informationen werden bis heute von einem Mitarbeiter in einer Microsoft Word Vorlage eingefügt und damit ein Testreport erstellt. Diese diese Arbeit kann nur von einem Mitarbeiter aus der Testentwicklung durchgeführt werden, da dieser mit den Abläufen der Tests und den daraus resultierenden Informationen vertraut ist. Da dies wichtige humane Ressourcen für einen gewissen Zeitraum bindet, ist es sinnvoll diesen Prozess zu automatisieren. Ein Programm womit jeder Nutzer nur durch die Eingabe eines Parameters einen Testreport erstellen kann.

## 1.3 Aufgabenstellung und Zielsetzung

Aufgabe ist die Entwicklung einer plattformunabhängigen Anwendung mit Graphical User Interface (GUI) welche mittels der Produktionsauftrag (PA) Nummer einen Testreport im Hyper Text Markup Language (HTML) Dateiformat erzeugt. Mittels dem GUI-Toolkit Qt soll ein benutzerfreundliches Programm zu automatisierten Erstellung von Testreports entwickelt werden. Die benötigten Informationen werde aus dem Webinterface der FlashDB-Datenbank und mittels Structured Query Language (SQL) Anfragen zusammengetragen. Die Darstellung des Testreports erfolgt in HTML, einer Auszeichnugssprache zu Darstellung von digitalen Dokumenten, und optional in dem Microsoft Word Dateiformat .docx. Der Testreport kann aus einem oder mehreren Produktionsaufträgen bestehen und ist für jede Produkttype gültig. Ziel ist es ein Softwareprojekt, mittels der Programmiersprache C++ und der Qt-Bibliothek, selbstständig zu entwickeln. Dabei sollen die im Studium und die, während der Tätigkeit als Werkstudent, erworbenen Kenntnisse und Methoden genutzt werden.

## 1.4 Vorgehensweise und Aufbau der Arbeit

# Grundlagen

# 2

In diesem Kapitel werden die für diese Bachelorarbeit wichtigen Grundlagen erläutert. Der Übersichtlichkeitshalber gehe ich nur auf die für mein Projekt relevanten Bereiche ein.

## 2.1 Qt

Dieser Abschnitt behandelt das GUI-Toolkit Qt. Da es sich bei Qt um ein sehr umfangreiches Framework handelt, wird hier nur auf die, für die Entwicklung des Programms, notwendigen Teilbereiche eingegangen. Eine Begründung warum die Entscheidung auf dieses Toolkit gefallen ist, erfolgt im Kapitel 3.4.

### 2.1.1 Definition

"Qt ist eine plattformunabhängige Entwicklungsumgebung für Desktop-PC, eingebettete und mobile Systeme. Unterstützt werden Linux, OS X, Windows, VxWorks, QNX, Android, iOS, Blackbarry, Sailfish OS und andere Plattformen. Qt ist keine Programmiersprache, sondern ein Framework, geschrieben in C++. Der Präprozessor Meta Object Compiler (MOC) erweitert die C++ Programmiersprache mit den neuen Features wie Signale und Slots. Vor dem Kompilierungsschritt analysiert der MOC die Quelldateien, die in erweitertem Qt-C++ geschrieben sind und erzeugt daraus standardkonforme C++ Quelldateien. So kann das Framework selbst, aber auch die Anwendungen/Bibliotheken mit einem standardkonformen C++ Compiler wie Clang, GCC, ICC, MinGW und MSVC kompiliert werden."[Qt17a]

### 2.1.2 Qt IDE - QtCreator

Die Integrated Development Enviroment (IDE) namens „QtCreator“ bietet viele bekannte Hilfen wie Syntax-Hervorhebung, Code-Vervollständigung, Debugger und eine Integration für alle gängigen Versionskontrollsystem wie zum Beispiel, git oder svn.

### 2.1.3 Qt Build System

Qmake ist ein Tool welches hilft den Build-Prozess für verschiedene Plattformen zu vereinfachen. Es automatisiert die Generierung von makefiles sodass wenige Zeilen mit Informationen genügen um jedes Make-File zu kreiieren. Man kann dies für jedes Softwareprojekt nutzen, egal ob es mit Qt geschrieben wurde oder nicht. (vgl.[Qt17e]
Jedes Qt-Projekt besitzt eine .pro Datei, diese Projekt Datei beinhaltet die zu ladenden Module, Konfigurationseinstellungen, Header- und Source-Files. Qmake erstellt aus den Informationen der Projekt Datei ein Make-File. Dieses Build-Tool bietet auch die Möglichkeit, Make-Dateien, die andere Make-Dateien rekursiv einbinden zu erstellen und zusätzlich noch bestimmte Eigenschaften in Abhängigkeit von der Zielplattform an- oder abzuschalten. (vlg. [**S.219**; JB09])
Eine Alternative zum Erstellen von Qt-Projekten bringt das plattformübergreifende make bzw. cmake Build-Management-Tool.

### 2.1.4 Signale und Slots

Der Signale und Slots Mechanismus ist das Hauptmerkmal von Qt. Es ist auch der Teil, der das Qt Framework am meisten von den anderen Frameworks unterscheidet. [Qt17j]
Jedes Element oder Objekt ist in der Lage miteinander zu kommunizieren. Dies funktioniert nach dem Äktion = Reaktion"Prinzip. Es wird ein Button geklickt, also ein GUI Element. Dieses sendet ein Signal aus und wird von einem Slot empfangen, der eine Funktion startet die zum Beispiel ein Menü öffnet. Es ist auch möglich ein Signal mehreren Empfängern und mehrere Signale einem Empfänger zu zuweisen (siehe Abb. 2.1).

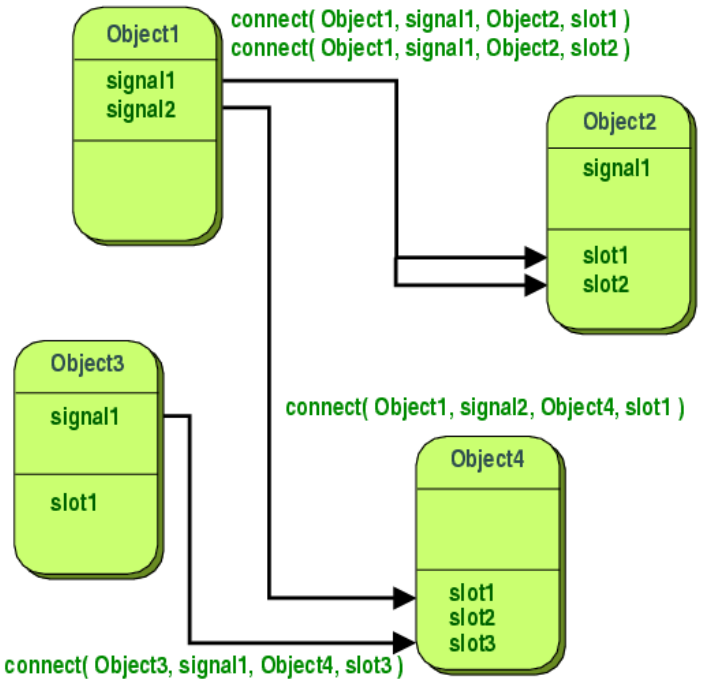



**Abb. 2.1.:** Signal-Slot-Konzept [Qt17b, Web]

Um diesen Mechanismus nutzen zu können, muss man in der Header Datei einer Klasse das Q_Object Macro hinzufügen. Dies geschieht üblich vor dem Konstruktor. Funktionen die im Slot Block stehen werden in der Header Datei in einen 'private slot:' bzw. 'public slot:' Block geschrieben. Signale werden in einen 'signal' Block geschrieben.

```
class Testklasse {
    Q_Object
public:
    Testklasse()

public slots:
    void Funktion()

signals:
    void abgeschicktesSignal();
 }
```

Eine Verbindung ensteht mit folgender Syntax:

*connect(SENDER, SIGNAL(abgeschicktesSignal()), EMPFÄNGER, SLOT(Funktion()));*

Ein großer Vorteil von Qt ist die automatische Trennung zwischen Sender und Empfänger wenn ein Objekt zerstört wird. Es ist aber auch möglich mit disconnect() eine Objekt Kommunikation manuell zu trennen.

### 2.1.5 Qt's Module und Klassen

Qt Module sind Klassenbibliotheken, die je nach Bedarf in der Projektdatei eingefügt werden.
Standartmäßig werden die Module Qt Core und Qt Gui für jedes Projekt zu Verfügung gestellt.
Die nachfolgenden Aussagen beziehen sich auf die softwareeigene Dokumentation von Qt.

#### QObject Klasse

"Die QObject Klasse ist die Basis Klasse für alle Qt Objekte."QObject ist das Herz von dem Qt Objekt Model. Die zentrale Eigenschaft in diesem Model, ist der sehr mächtige Mechanismus für die nahtlose Objektkommunikation genannt Signale und Slots."[Qt17f]
Diese Klasse bietet eine Vielzahl von Funktionen die die Organisationstruktur deutlich macht z.B parent() oder children(). Erzeugte Objekte gliedern sich in eine Baumstruktur. Erstellt man ein QOjekt mithilfe eines anderen Objektes, so wird dieses automatisch zu Liste der Kinder des Objekts hinzugefügt. Eine Löschung des Elternobjekts zieht eine automatische Löschung der Kinderobjekte nach sich.

#### Qt WebEngine Modul

Das Qt WebEngine Modul bietet eine Webbrowser Engine die es einfach macht, Inhalte aus dem World Wide Web (WWW) in Ihre Qt Applikation einzubetten. Qt WebEngine bietet C++ Klassen and Qt Meta Language oder Qt Modeling Language (QML) Typen für das Rendern von HTML, Extensible Hypertext Markup Language (XHTML), und Scalable Vector Graphics (SVG)

Dokumenten gestaltet mithilfe von Cascading Style Sheets (CSS) und der Skriptsprache Java Script (JS). HTML Dokumente können vom Benutzer, über die Nutzung des „contenteditable“ (inhaltseditierbar) Attribute auf HTML Elemente, vollkommen editiert werden. Hierfür integriert die QtWebEngine Chromiums schnelle Web Möglichkeiten in Qt. Die Integration mit Qt fokusiert sich auf ein Application Programming Interface (API) das einfach zu benutzen und dennoch erweiterbar ist. (vgl. [Qt17h])

### QXmlStreamWriter Klasse

"Diese Klasse bietet eine XML Dokumenten Schreiber mit einer simplen streaming API."[Qt17i] Es ermöglicht einem mit einfachen und spezialisierten Funktionen ein XML Dokument zu erstellen. Dies kann zur halbautomatische Generierung von HTML Syntax genutzt werden. Generell beginnt jedes Dokument mit der writeStartDocument() Funktion und endet mit der writeEndDocument() Funktion. Diese und weitere Funktionen setzen die vom Benutzer eingegebenen Informationen in so genannte Tags. Tags können als Steuerelemente betrachtet werden, diese Elemente dienen zur Strukturierung von Texten oder Bildern.

### QFile Klasse

Die QFile Klasse ist ein Input and Output (IO) Gerät. Dieses hat den Zweck Text- oder Binärdateien zu lesen bzw zu schreiben. Der Dateinamen wird direkt im Konstruktor mit angegeben. Um Textdateien verarbeiten zu können, muss man QTextStream Klasse verwenden. Diese ermöglicht es den Datenstrom von der QFile Klasse zeilenweise, komplett oder bis zu einer bestimmten Länge zu lesen. (vgl. [Qt17d])

### Qt Sql

Wie alle Module, muss auch dieses in der Projekt Datei bekannt gemacht werden. Dies geschieht in dem man *QT += sql* hinzufügt. Die SQL API ist in drei verschiedene Schichten aufgeteilt:

- Driver layer
- SQL API layer
- User interface layer

(vlg. [Qt17g])

### 2.1.6 Lizensierung

Softwarelizenzen nehmen eine wichtige Rolle in der Entwicklung von Programmen ein. Sie dienen dazu die Nutzung und Verbreitung einer Software zu regeln und die Rechte des Urhebers zu schützen. Lizenzen werden in zwei Kategorien unterteilt. Freie Software, auch Open Source Software genannt, und nicht freie Software, so genannte Endbenutzer-Lizenzvertrag Software End User License Agreement (EULA).
Qt bietet eine duale Linzenzierung. Die kommerzielle Lizenzierung ermöglicht Entwicklung und Vertrieb einer Software unter eigenen Bedingungen, wobei dann eine Gebühr an Qt fällig wird. Die Open Source Linzenzierung ist gegeben unter General Public License (GPL) und GNU Lesser General Public License (LGPL). Qt hat sich für LGPL als primäre Open Source Lizenz aus folgenden Gründen entschieden:

- Die Freiheit das Programm für jeden Zweck auszuführen.
- Die Freiheit das Programm zu untersuchen und es auf spezielle Bedürfnisse anzupassen
- Die Freiheit das Programm weiterzuverteilen
- Die Freiheit das Programm zu verbessern und es der Öffentlichkeit zugänglich zu machen.

Somit ist es möglich eine proprietäre Software zu entwickeln, es müssen jedoch einige Lizenzvereinbarungen beachtet werden. (vlg. [Qt17c])

## 2.2 HTML und XML

"Die Struktur von Webseiten wird durch die Auszeichnungssprache HTML dargestellt. Der Name steht für Hypertext Markup Language (Auszeichnungssprache für Hyptertext, also Text mit integrierten Stukturinformationen und Querverweisen). HTML-Code sieht im Wesentlichen genauso aus wie [...] Extensible Markup Language (XML) - die gemeinsame Wurzel von XML und HTML ist die Auszeichnungssprache Standard Generalized Markup Language (SGML)."[**S.840**; Ker08]

### 2.2.1 HTML

Die Grundstuktur von HTML Dokumenten ist immer gleich. Durch den öffnenden Tag <*html*> und dem schließenden Tag </*html*> wird das Dokument als HTML gekennzeichnet. Nach dem öffnenden Tag kommt der Kopf, der so genannten Head <*head*>, der Informationen wie z.B. den Titel enthält und dem Dokumentenkörper, Body, in dem sich der Inhalt befindet.
Eine Textformatierung findet in HTML nicht statt. Der Inhalt der Dokumentes wird im Browser schlicht als Fließtext angezeigt. Möchte man einen Zeilenumbruch erzwingen oder eine Textpassage dick hervorheben, muss dies ebenfalls mit Tags realisiert werden. Eine weiter Fehlerquelle sind Sonderzeichen. Generell benutzt HTML die American Standard Code for Information (ASCII) Kodierung. Diese enthält die 128 weltweit gleichen Zeichen. Möchte man den für sein Land spezifischen Zeichensatz verwenden kann man dies im Head mit dem Meta-Tag<*meta*> deklarieren, für die USA und Westeuropa ist dies *iso-latin-1*, oder man benutzt Entity-Referenzen. Diese Entity-Refenzen sind eine Zeichenfolge die es ermöglichen bestimmt Sonderzeichen darzustellen. Möchte man zum Beispiel dem Umlaut ä darstellen so ist Entity-Referenz hierfür *ä* .
Desweiteren gibt es die Möglichkeit Tags mit Eigenschaften zu versehen um Darstellung schöner zu gestalten. Eigenschaften könne Textausrichtung, Identifikationsnamen, Farben und Größenangaben wie Zoomfaktor bei Bildern sein. Die aktuelle HTML Version ist HTML5.1 .

### 2.2.2 XML

Bei dieser erweiterbaren Auszeichnungssprache handelt es sich nicht um ein bestimmtes Dokumentformat sonder im eine Metasprache die beliebige Auszeichnungssprachen definiert. Es kann sich dabei um Textdokumente, Vektorgrafiken, multimediale Präsentation, Datenbanken oder andere Arten von strukturierten Daten handeln. Der größte Vorteil von XML ist, dass es sich um eine universelle Sprache handelt. Im Gegensatz zu LaTex, was zum Erstellen wissenschaftlichen Arbeiten dient, ist es bei XML unerheblich um was für ein Dokument es sich handelt. Ebenso wie bei HTML strukturiert diese Auszeichnungsspache nur die Daten, eine Formatierung kann durch eine zusätzliche Formatierungssprache, wie zum Beispiel CSS, vorgenommen werden. (vgl. [**S.769/S.770**; Ker08]) Der Aufbau dieses Dokumentes ähnelt dem HTML sehr.
"Jedes XML-Dokument besteht aus einer Hierarchie ineinander verschachtelter Steuerungsanweisungen, die als Element oder Tags bezeichnet werden, und kann zusätzlich einfachen Text enthalten."[**S.771**; Ker08]
Der erste Tag in einem Dokument definiert den Dokumenttyp. Dies geschieht hier mit folgender Syntax: *<?xml version="1.0" encoding=ütf-8"?>*
Nun da der Dokumentyp als XML definiert wurde, kann man seinen Informationen in Tags stukturieren die selbst benannt werden z.B. *<automarke>Audi</automarke>* . Somit haben wir die Informationen *Audi* dem Tag *automarke* zugewiesen. Ein anderes Programm kann nun beim Lesen des Dokumentes erkennen, dass es sich bei Audi um eine Automarke handelt.

## 2.3 GUI Gestaltung

Die GUI, also die grafische Benutzeroberfläche ist die Schnittstelle zwischen dem Computerprogramm und dem Benutzer. Sie dient der Steuerung des Programms durch grafische Bedienelemente. Für die Gestaltung einer Oberfläche gibt es eine internationale Standard Richtlinie, die Norm **EN ISO 9241**.
"Die Normenreihe beschreibt Anforderungen an die Arbeitsumgebung, Hardware und Software. Ziel der Richtlinie ist es, gesundheitliche Schäden beim Arbeiten am Bildschirm zu vermeiden und dem Benutzer die Ausführung seiner Aufgaben zu erleichtern."[Wk9]
Die sieben wichtigsten Grundlagen zur Dialoggestaltung finden sich in der Norm EN ISO 9241-110 und lauten:

1. **Aufgabenangemessenheit**
   Die Gestaltung sollte möglichst einfach gehalten werden, damit der Benutzer nicht unnötig verwirrt wird. Die Bedienung sollte als Ziel eine Vereinfachung des Arbeitsaufwandes als Ziel haben.

2. **Seblstbeschreibungsfähigkeit**
   Die Oberfläche sollte dem User ausreichend Sinn und Nutzung des Programmes erklären.

3. **Erwartungskonformität**
   Durch Konsistenz sollte dem Benutzer eine gewohnte Bedienbarkeit gegeben werden.

4. **Lernförderlichkeit**
   Mithilfe weniger Erläuterungen sollte dem Benutzer eine schnellstmögliche Bedienung gewährleistet werden.

5. **Steuerbarkeit**
   Die Steuerung durch den User sollte möglich sein.

6. **Fehlertoleranz**
   Das Programm sollte den Benutzer durch Korrektur oder Hinweise möglicher Fehler bei der Eingabe unterstützen.

7. **Individualisierbarkeit**
   Eine Anpassung auf spezifische Anforderungen des Benutzer sollte möglich sein.

Mithilfe dieser Regeln ist es dem Benutzer möglich schnell neue Programme zu bedienen.
Die Bedien- bzw. Steuerungselemente einer GUI werden **Widgets** gennant. Dieses Wort wurde aus den englischen Wörtern *Window*, Fenster, und *Gadget*, Gerät, zusammengesetzt.

Man unterscheidet zwei Arten von Widgets.

**Einfache Widgets:**

- Fenster und Menüs
- Buttons
- Checkboxen
- Combo- und Listboxen
- Radiobuttons
- Textfelder

**Komplexe Widgets:**

- Tabellen
- DateTime-Picker
- Öffnen-, Speichern-Dialoge
- Symbolleisten
- TreeViews
- Selbst erstellte Widgets

(vlg. [Fuc10])

# 3 Analyse

Dieses Kapitel beschäftigt sich mit der Untersuchung des Projektes. Es müssen alle Prozesse, Anforderungen und Informationen, die für die Software von Bedeutung sind, möglichst genau bestimmt und verstanden werden. Es wird zunächst aktuelle Zustand betrachtet, dann werden die Probleme und Schwächen aufgelistet. Anschließend erfolgt eine Anforderungsdefinition und zum Schluss werden mögliche Lösungen skizziert.

## 3.1 Analyse des Ist-Zustandes

Zu Beginn dieser Bachelorarbeit wurden die Testreporte von einem Testentwickler erstellt. Die relevanten Informationen wurden über das WebInterface der FlashDB-Datenbank und über vom Tester angelgte Logfiles ermittelt und dann in eine Microsoft Word Vorlage eingefügt.

### 3.1.1 Datenzustand

Jede Woche werden tausende Speichermedien unter unteschiedlichen Testbedingungen und Anforderungen getestet. Dies versucht eine große Menge an Daten von Testergebnissen. Diese werden auf dem eigenen Server *FlashDB-Server* gespeichert und mittels firmeneigener Software in eine von zwei Datenbanken implementiert. Die *Products* Datenbank beinhaltet Produktspezifikationen sowie zu benutzende Testbedingung, die vom Tester vor Beginn eines Tests ausgelesen werden. In die *Results* Datenbank werden die Testergebnisse vom Tester geschrieben. Durch die Implementierung eines VBA Interfaces durch einen Mitarbeiter der Testentwicklung, kann man viele Informationen beider Datenbanken im Browser unter *http://flashdb* abfragen (Abb.3.1) . Alle Daten werden persistent gespeichert, weshalb die Speicherbelegung stetig anwächst. Zum Zeitpunkt dieser Arbeit belegen alle Daten einen Speicherplatz von 60 GB. Diese Daten werden für verschiedene Zwecke erhoben. Sie dienen als Hilfe für die Geschäftsführung, die wöchentlich eine

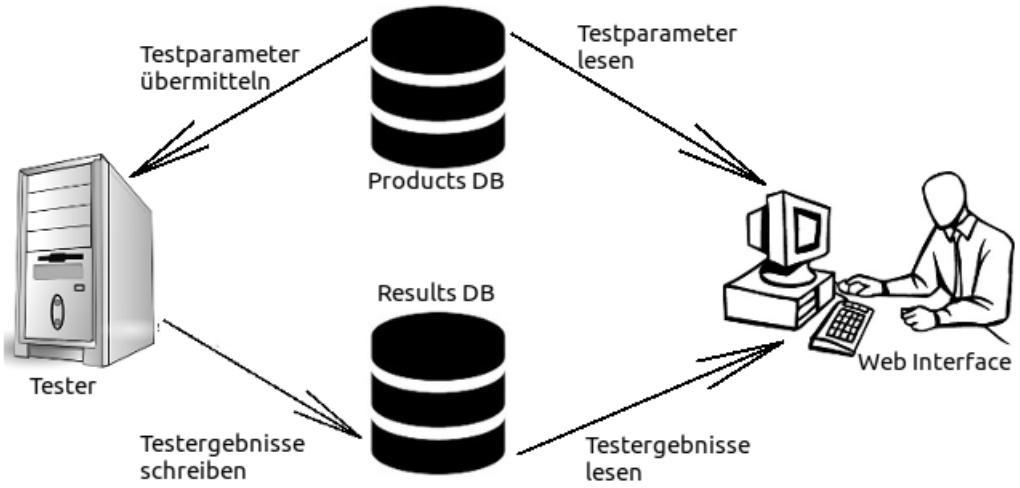



**Abb. 3.1.:** Datenhandling vom Tester bis zum Nutzer

Statistik erhält. Des weiteren helfen sie bei der Fehler- und Fehlerquellensuche und als Beleg für Kunden über die durchgeführten Testschritte.

### 3.1.2 Datenauswertung

Anhand der PA Nummer ist es möglich verwendete Prüflinge, durchgeführte Tests, Testzeiten und Testergebnisse zu ermitteln. Diese werden immer in ein gleich strukturiertes Word Dokument implementiert. Dies gliedert sich wie folgt:

1. Verwendete Prüflinge
2. Durchgeführte Tests
3. Ermittelte Testzeiten
4. Testergebnisse
5. Auswertung

Es erfordert Zeit sich durch einen PA durchzuklicken um alles nötige zu beschaffen, da nicht alle Informationen sofort ersichtlich sind. Die erste Ansicht nach Eingabe der PA Nummer ist in Abbildung 3.2 zu sehen. Bei

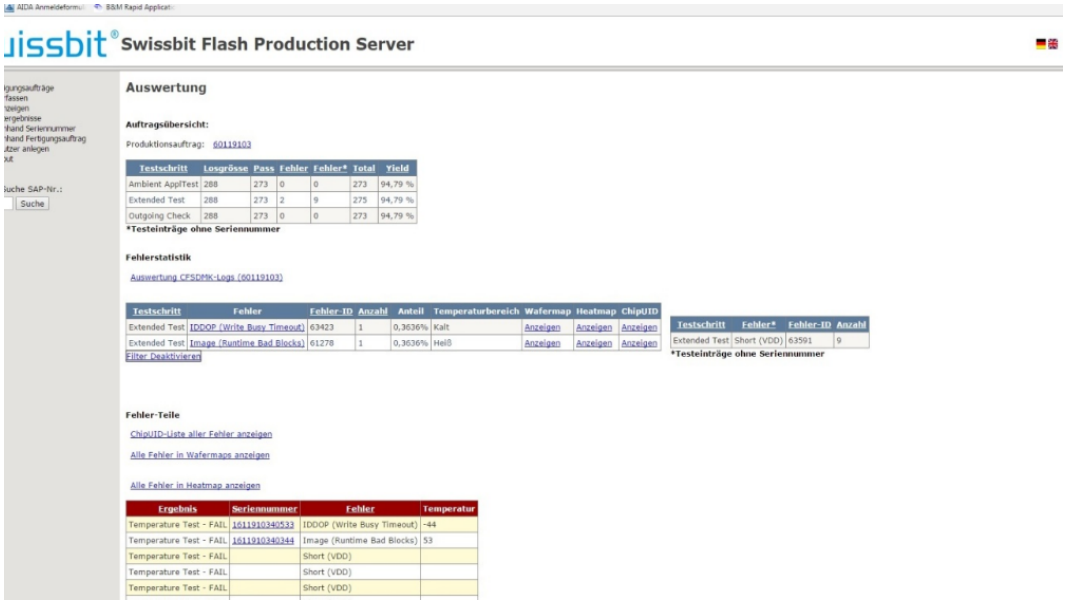



**Abb. 3.2.:** Web Ansicht FlashDB-Datenbank

dem Test *Preformat/Firmware einspielen auf Compact-Flash-Card SD-Card Manufacturing-Kit (CFSDMK)* kann die Testzeit, Firmware, Testersoftware, Testerfirmware und mögliche Fehler nur durch die Auswertung von Logfiles, welche auf dem Server *sbdesql01* abgelegt werden, ermittelt werden. Dies erfordert spezielles Wissen über den Test und die im Logfile hinterlegten Informationen.
Des weiteren müssen Tabellen mit Fehlerstatistiken übernommen werden.

### 3.1.3 Tests

Jeder Kunde der Swissbit AG stellt eigene Anforderungen an unsere Produkte. Diese Anforderungen sind Abhängig von dem vom Kunden vorhergesehenen Einsatzort und Einsatzzweck des Speichermediums. Hierfür hat die Testentwicklung unterschiedliche Testszenarien entwickelt und in der Products-Datenbank hinterlegt. Jedes Speichermedium durchläuft nach der Fertigung Tests. Es werden sieben Hauptkategorieren unterschieden.

- Preformat
- Ambient ApplTest
- Extended Temperature Test

- Final Test
- Outgoing Check
- QS Verify
- Security Configuration
- Image Configuration

Für jeden Speichermedientyp existiert ein Testgerät auf dem die Produkte von einem Mitarbeiter in einen Sockel aufgesteckt werden. Dieser gibt dann in der firmeneigenen Testsoftware die PA Nummer ein und startet den Test (Abb. 3.3). Die Swissbit AG bietet dem Kunden die Testverfahren drei Temperaturbereiche.
*Commercial* von 0°C bis +70°C.
*Extended* von -25°C bis +85°C.
*Industrial* von -40°C bis 85°C.
In diesen Temperaturspannen durchlaufen die Produkte mehrmalige Schreib- und Lesezyklen. Sämtliche Informationen wie zum Beispiel über die erreichte Temperatur, geschrieben und gelesene Blöcke werden von der Software erfasst und gespeichert.

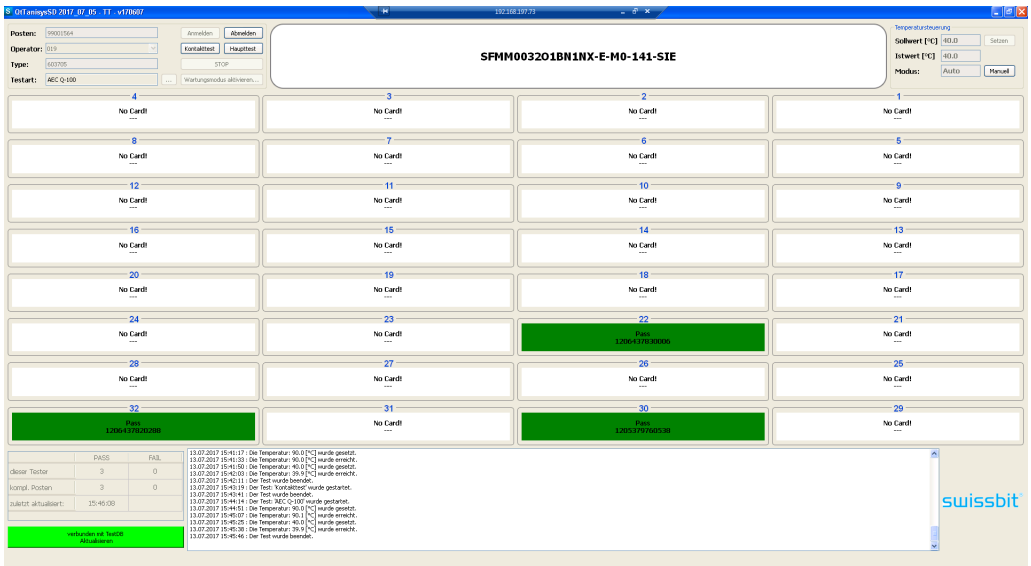



**Abb. 3.3.:** Testersoftware

### 3.1.4 Datenbank

Die Swissbit AG nutzt eine Microsoft SQL Server. Es handelt sich hierbei relationelles Datenbankmanagementsystem. Um eine Verbindung zwischen Tester und Datenbank zu ermöglichen wurde mithilfe von Qt eine Bibliothek *FlashDBLib* entwickelt. Diese bietet verschiedene Funktionen die SQL Queries nutzt um Daten in die Datenbank zu schreiben oder aus der Datenbank zu lesen. Die stetige Weiterentwicklung erfordert sich mit entwickelnde Dokumentation. Realisiert wird dies durch die Dokumentationsoftware *Doxygen*. Die Datenbank befindet sich auf einen lokalen Server und wird wöchentlich auf einen zweiten Server gespiegelt. Des weiteren werden die Datensätze permament auf einen zweiten Server geschrieben. Somit wird eine Ausfallsicherheit gewährleistet. Jede Tabelle in einer Datenbank besitzt einen Primär- und einen Fremdschlüssel die Abhängkeiten der verschiedenen Datensätze erstellen (Abb. 3.4). Die *Products* Datenbank beinhaltet die Produktspezifika-

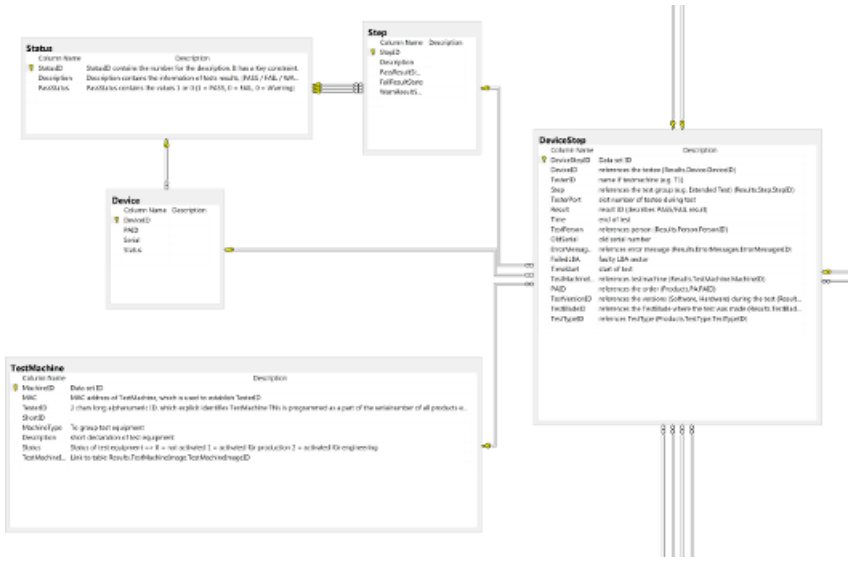



**Abb. 3.4.:** Ausschnitt der Abhängigkeiten ResultsDB

tionen und belegt zum Zeitpunkt dieser Arbeit 185 MB Speicherplatz. Die *Results* Datenbank beinhaltet die Testergebnisse und belegt zum Zeitpunkt dieser Arbeit 16 GB.

## 3.2 Schwachstellenanalyse

Der im vorherigen Kapitel beschriebene Ablaufprozess eines Testreports ist eine seit Jahren funktionierende Methode. Trotz der Routine vieler Mitarbeiter sind Fehler nicht immer zu vermeiden. Schaut man sich den Ablauf genauer an, ist schnell zu erkennen das der Mensch die Schwachstelle ist. Die Analyse ergab folgende Hauptfehlerquellen.

- **Fehler beim Anlegen des Produktionsauftrages**
  Beim Erstellen eines Produktionsauftrag kann durch nicht korrekte Eingabe der Auftragsdaten ein falscher Produktionsauftrag entstehen Dies hat zur Folge, dass Testdaten z.B. unter einer falschen PA-Nummer abgelegt werden und dann eventuell nicht mehr auffindbar sind.

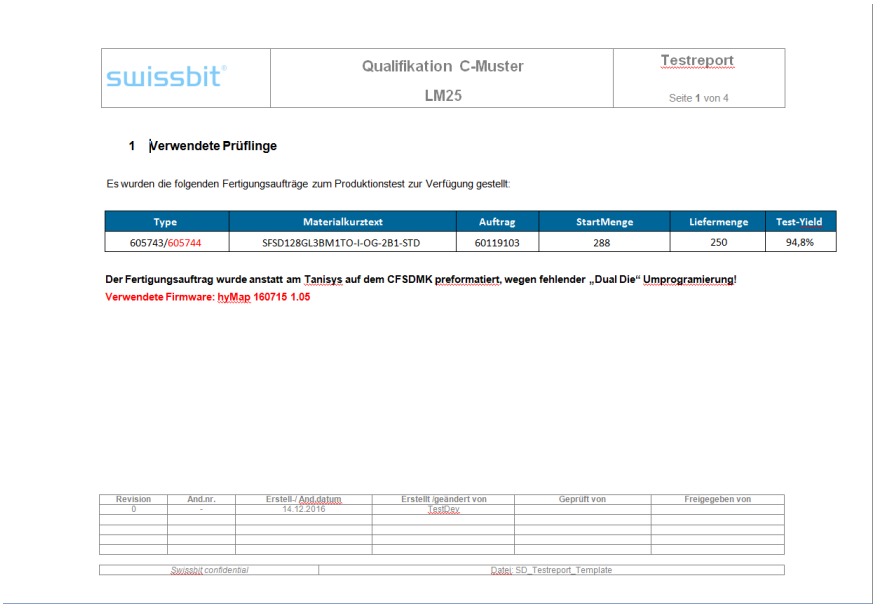
- **Vertauschung von Speichermedien**
  Produktionsmitarbeiter kann durch Unachtsamkeit Speichermedien mit unterschiedlichen Produktionsaufträgen vertauschen. Dadurch würden die Testergebnisse verfälscht, da die Anforderungen und Spezifikation sich unterscheiden.

- **Fehler beim Erstellen des Testreports**
  Die Berechnung der Testzeit aus den Logfiles des CFSDMK Tester kann falsch vorgenommen werden. Während der Eingabe der Informationen kann durch den Mitarbeiter ein Fehler gemacht werden. Zum Beispiel das Vertauschen von Zahlen oder eine Eingabe an einer falschen Position.

- **Erstellung nur durch TDEV Mitarbeiter**
  Nur ein Mitarbeiter der Testentwicklung hat das nötige Wissen um einen Testreport anzulegen. Dies schränkt die Anzahl der Personen, die dazu fähig sind, ein.

## 3.3 Soll Konzept

Die Anwendung soll in der Lage sein, durch die Eingabe eines Parameters einen Testreport automatisch zu erstellen. Die Darstellung des Testreports soll sich nicht sehr stark von der alten Darstellung unterscheiden. Das bedeutet, dass es auf den ersten Blick ersichtlich sein sollte das es sich um einen Testreport handelt (Abb 3.5). Die Bedienung des Progamms soll über eine GUI möglich sein. Der Testreport soll durch eigene Kommentare ergänzt werden können.


swissbit

Qualifikation C-Muster

LM25

Testreport

Seite 1 von 4

1 Verwendete Prüflinge

Es wurden die folgenden Fertigungsaufträge zum Produktionstest zur Verfügung gestellt:

| Type | Materialkurztext | Auftrag | StartMenge | Liefermenge | Test-Yield |
|---|---|---|---|---|---|
| 605743/605744 | SFSD128GL3BM1TO-I-OG-2B1-STD | 60119103 | 288 | 250 | 94,8% |

Der Fertigungsauftrag wurde anstatt am Tanisys auf dem CFSDMK preformatiert, wegen fehlender „Dual Die" Umprogramierung!

Verwendete Firmware: hyMap 160715 1.05

| Revision | And.nr. | Erstell./ And.datum | Erstellt /geändert von | Geprüft von | Freigegeben von |
|---|---|---|---|---|---|
| 0 | - | 14.12.2016 | TestDex | | |

Swissbit confidential | Datei: SD_Testreport_Template

**Abb. 3.5.:** Muster Testreport

Die für den Testreport notwendigen Informationen sollen über das Webinterface der FlashDB und gegebenenfalls direkt von der SQL Datenbank beschafft werden. In der Browserdarstellung sind viele Informationen zu einem Produktionsauftrag zusammengefasst, daher ist es sinnvoll diesen Vorteil zu nutzen und die Daten über das Web Frontend abzugreifen. Das Beschaffen der Testreport Daten ist auch direkt mittel SQL Querys (Anfragen) möglich. Da dies durch die Abhängigkeiten der Datensätze zu sehr großen und komplexen Anfrageblöcken führen kann und die Ressourcen des Server durch diese stark beansprucht werden könnten, viel die Entscheidung auf das Web Frontend.

## 3.4 Lösungsansatz

Dieses Kapitel befasst sich mit einem ersten Ansatz zur Lösung der im Kapitel 3.3 gezeigten Vorgaben. Der Gedankenweg und Lösungsansatz soll verstanden und nachvollzogen werden können. Ziel ist eine effektive und sinnvole Lösung der gestellten Aufgabe zu erarbeiten.

### 3.4.1 Datenbeschaffung

Durch Reflexion des Studiums und der, während der Werkstudententätigkeit, gesammelten Erfahrungen wurden zwei Ideen entwickelt. Eine Methode diese zu erhalten, ist das Suchen nach bestimmten Stichworten. Dies hat jedoch den Nachteil, dass man davon ausgeht das bei jedem Produkt die Informationen immer unter dem gleichen Namen steht. So kann zum Beispiel nach einem Testschritt gesucht werden, um weitere Informationen wie zum Beispiel die Anzahl der Pass-Teile zu ermitteln. Dieser Ablauf ist in Abbildung 3.6 zu sehen.

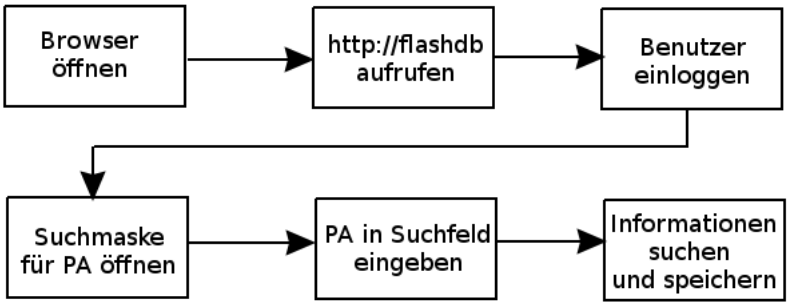



**Abb. 3.6.:** Ablauf Informationsbeschaffung

Verschiedene Produkte durchlaufen verschiedene Testschritte. Wird ein Testschritt gesucht der nicht durchgeführt wurde, kommt es zu fehlerhaften Informationen. Dieser Fall muss abgefangen werden.
Eine zweite Möglichkeit ist die Suche nach so genannten Keywords im Quelltext der Webseite. Dies hat den Vorteil gegenüber der vorherigen Methode, dass nach bestimmten Tags, Tag-ID's oder Tag-Namen Kombinationen gesucht werden kann. Somit kann bei mehrfachen Vorkommen eines Wortes, eine bessere Unterscheidung zwischen diesen getroffen werden. Des weiteren können mit dieser Vorgehensweise nicht nur Informationen aus Tabellen geholt

werden, sondern es kann der ganze Quelltext der Tabelle kopiert werden. Dies hat den Vorteil das die Struktur gleich ist, Fehler bei der Übertragung der Werte vermieden werden und die Implementierung simpler ist. Die so erhalteten Information werden als String Datentypen gespeichert.
Noch dazu werden für Speichermedien, die den *Preformat* Testschritt auf dem CFSDMK Tester durchlaufen, Logfiles auf dem *sbdesql01* Server gespeichert. Aus diesem Grund macht es Sinn, dass die Software anhand der PA-Nummer den Server nach den zugehörigen Logfiles durchsucht. Die Fundstücke sollten temporär und lokal gespeichert werden um Ressourcen des Server zu sparen und den Datenverkehr im Netzwerk zu reduzieren.

### 3.4.2 Verarbeitung

Die gesammelten Informationen sollten auf Sinnhaftigkeit überprüft werden. Dies kann realisiert werden durch Funktionen, die bei Mengeninformationen überprüfen ob es sich um Zahlen handelt, Fehlerbeschreibungen untersuchen ob Buchstaben enthalten sind und bei Temperatur- oder Zeitangaben überprüfen ob die angegeben Einheiten mit den Werten zusammenpassen. Eine weitere Verarbeitungsfunktion ist die Verkettung oder Zerstückelung von Informationsstrings, da in manchen Strings mehrere Informationen enthalten sind. Dies ist im Testreport als Tabelle dargestellt, deswegen ist eine Zerstückelung beziehungsweise Aufteilung der Informationen in einzelne Strings sinnvoll (Abb. 3.7).

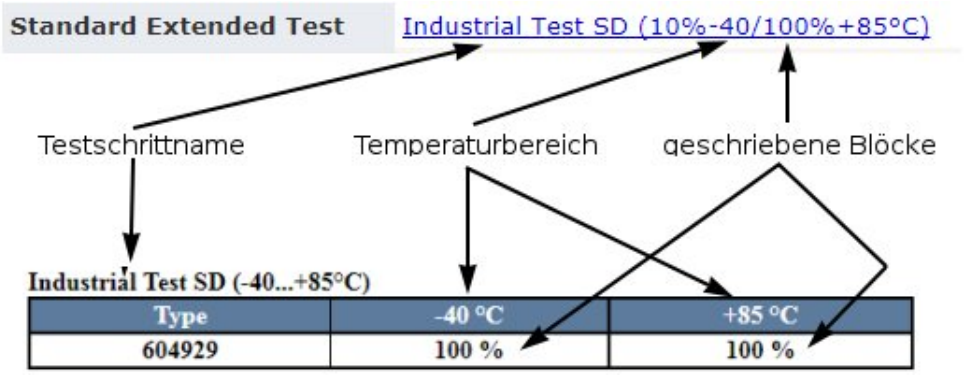



**Abb. 3.7.:** Temperaturtestschritt

Die Inhalte der Logfiles werden für die Auswertung des Preformat Testschrittes auf dem CFSDMK Tester benötigt, unter anderem werden mithilfe dieser die Testzeiten ermittelt als auch die verwendete Testersoftware und auf dem Speichermedium eingespielte Firmware. Eine Funktion ist notwendig, die

erkennt bei welchem Inhalt es sich um Firmware handelt und an welcher Stelle ein Test beginnt und an welcher Stelle endet. Da jede Meldung im Logfile einen Zeitstempel hat, ist die Testzeitberechnung relativ einfach zu lösen. Es ist dabei zu beachten, dass bei Erkennung eines Fehlers, dieser gespeichert wird und die Testzeitberechnung beendet wird.

### 3.4.3 Darstellung

Die Darstellung des Testreports soll übersichtlich und leicht verständlich sein. Als Vorlage dienen die existierenden Testreporte. Aufgrund einer möglichen Einbindung in das Web Frontend ist die Ausgabe in dem HTML Dateiformat sinnvoll. Eine Auflistung in fünf Hauptkategorien ist vorgesehen.

1. **Verwendete Prüflinge**
   In einer Tabelle werden die Prüflinge aufgelistet, diese beinhaltet Type, Materialkurztext, Auftragsnummer, Startmenge, Liefermenge, Test-Yield. Somit hat der Leser einen Überblick.

2. **Durchgeführte Tests**
   Hier werden die notwendigen Informationen der durchgeführten Test aufgelistet. Bei mehreren PA's werden diese, getrennt durch ihre Auftragsnummer, nacheinander aufgelistet.

3. **Ermittelte Testzeiten**
   In Tabellenform werden die Testzeiten für die jeweiligen Testschritte dargestellt.

4. **Testergebnisse**
   Fehler bei Testschritten werden in Tabellen aufgeführt.

5. **Auswertung**
   Dem Ersteller wird hier die Möglichkeit gegeben eigene Kommentare einzufügen um Besonderheiten zu erwähnen.

### 3.4.4 Handhabung

Die Ergonomie der Software soll so gestaltet werden, dass dem Benutzer der Gebrauch und Zweck auf Anhieb verständlich ist. Allgemein werden die in Kapitel 2.3 aufgelisteten Grundlagen der Mensch-Computer-Interaktion berücksichtigt. Es ist vorgesehen ein Hauptfenster zu gestalten und weitere Dialogfenster, dessen Aufbau sich nach dem der Microsoft Word Dokument Muster Vorlage richtet. Diese Dialogfenster sollem dem Benutzer eine gewohnte Darstellung der Informationen bieten und die Möglichkeit, Werte zu verändern gegebenenfalls zu korrigieren. Zu vermeiden ist eine verkomplizierung der Handhabung durch unnötige Einstellmöglichkeiten. Aus diesen Gründen ist es zweckmässig die Auswahlmöglichkeiten für den Nutzer zu beschränken und nur dann anzubieten wenn diese notwendig sind um eine zweckmässige Fortführung des Programms zu gewährleisten.
Da es vorkommen kann, dass mehrere Produktionsaufträge zu einem Report zusammengefasst werden, ist eine Textfeld mit der Möglichkeit mehrere PA-Nummern einzugeben praktisch. Dialogfenster, die zur Überprüfung der Werte dienen, sollten eine Möglichkeit haben, die Ausgabe ohne weitere Überprüfung zu ermöglichen.

# 4 Realisierung

## 4.1 Randbedingungen

Allgemein kann gesagt werden, dass für die Realisierung dieses Projektes sehr viel Freiheiten gewährt wurden. Es wurden lediglich ein paar Bedingungen vorgegeben. Wichtig ist das Ergebnis.

### 4.1.1 Musskriterien

Der Testreport-Generator muss plattformunabhängig sein. Die in der Swissbit AG hauptsächlich benutzten Betriebssysteme sind Mircrosoft Windows 7, Microsoft Windows 8 und Linux. Für alle soll die Software kompilierbar und ausführbar sein. Die Programmiersprache muss C++ sein und es soll mit dem GUI-Toolkit Qt entwickelt werden. Die Entwicklung muss selbständigt geschehen. Das Einholen der Information muss über das Web Interface der FlashDB Datenbank realisiert werden. Die Software muss in der Lage sein, mehrere Produktionsaufträge verarbeiten zu können. Das Erstellen des Testreports muss erheblich weniger Aufwand benötigen als die bisherige Lösung. Außerdem muss die Informationsbeschaffung automatisiert von statten gehen. Es ist notwendig die Inhalte auf ihre Richtigkeit zu überprüfen. Das Programm soll in einer ausführbaren Form vorliegen, das heißt es soll selbständig funktionsfähig sein.

### 4.1.2 Wunschkriterien

Optional soll die Ausgabe des Testreports noch im Microsoft Word Dateiformat erfolgen. Um Benutzerfreundlichkeit zu gewähren, soll ein Installationsprogramm die Software mit den zugehörigen Bibliotheken benutzerfreundlich in das System integrieren. Für eine bessere Nachvollziehbarkeit sollten die Arbeitsschritte der Software erkennbar sein.

## 4.2 Entwurf der Qt-Anwendung

In diesem Abschnitt wird der Entwurf des Softwareprojektes vorgestellt. Die Entwurfsplanung orientiert sich an den zwei Qt-Softwareprojekten die bereits verwirklicht wurden. Eine Vorgehensweise nach dem *Scrum* Softwareentwicklungsprinzip wurde nicht angewandt, da das Entwicklerteam nur aus einer Person besteht und diese Vorgehensweise erst ab einer Teamgröße von mehreren Personen überzeugend ist.

### 4.2.1 Funktionieller Entwurf der Qt-Anwendung

Der Testreport Generator fordert zunächst auf, PA-Nummer(n) einzugeben. Da nicht bei jedem Arbeitsplatz die Netzlaufwerkverbindung zu dem Server *sbdesql01* mit dem gleichen Laufwerksbuchstaben besteht, öffnet sich ein Dialog Fenster. Dieses fordet auf, den Ordner mit den Logfiles des CFSDMK auszuwählen. Die Suche nach den benötigten Dateien beginnt in einem neuem Thread, dies ist sinnvoll um ein Abbrechen und Beenden der Anwendung auf während der Suche zu ermöglichen .
Nach Beendigung der Suche öffnet sich in der Anwendung ein Browser um eine Verbindung zum Web Frontend der FlashDB-Datenbank herzustellen. Benutzername und Passwort sind im Quelltext der Anwendung hinterlegt und können nur dort verändert werden. Der Benutzer hat nur das Recht lesend auf das WebInterface der flashdb-Datenbank zuzugreifen. Es folgt eine automatisierte Anfrage. Webseiten, die benötigte Informationen enthalten, werden lokal und temporär als HTML-Datei (Quelltext) und als Plaintext-Datei (Fließtext) gespeichert. Dies wird aus folgenden zwei Gründen gemacht:

- **Weiternavigation**
  Da nicht jeder Link auf einer Webseite eine Tag-ID besitzt, wodurch er eindeutig identifizierbar und mithilfe spezieller Funktionen erreichbar ist. Werden Quelltextdateien gespeichert, die dann durch selbst entwickelter Algorithmen nach weiterzuverfolgenden Links durchsucht werden.

- **Informationssammlung**
  Die Qt-Anwendung durchsucht die Fließtextdateien nach benötigten Informationen und übergibt sie bei erfolgreicher Suche dem Programm.

Falls für den Preformat Testschritt keine Firmware gefunden werden kann, wird ein SQL-Verbindung zur ProductsDB-Datenbank hergestellt um nach dem Firmwarenamen und der Firmwareversion zu suchen. Die Loginparameter für die SQL-Verbindung sind ebenfalls im Quelltext hinterlegt, ebenso wie bei der Browserdarstellung ist nur einer lesender Zugriff gestattet. Das Erscheinen eines neuen Fenster, mit der Ansicht über den verwendeten Prüfling, zeigt an das die Informationsbeschaffung beendet ist. Die Anwendung bietet nun drei Möglichkeiten fortzufahren. Der "**BACK**"-Button, bringt den Nutzer zum vorherigen Dialog. "**NEXT**"-Button navigiert zum nächsten Dialogfenster. Eine Besonderheit hierbei ist, dass nur Dialogfenster angezeigt werden die auch Werte enthalten. Unnötiges weiterklicken wird damit unterbunden. "**FINISH**"-Button überspringt alle folgenden Dialogfenster. In dem Dialogfenster *Testreport Generator - Site6* das die Testzeiten der Testschritte präsentiert, blendet nicht durchgeführte Testschritte aus. Am Ende wird ein Testreport unter der Produktionsauftragsnummer im HTML-Dateiformat abgespeichert. Sollten noch nicht abgearbeite PA's existieren, wiederholt sich der Vorgang bis alle abgearbeitet wurden. Nach dem letzten Testreport, wird eine Funktion aufgerufen die alle Testreporte zu einem zusammenfasst.
Um eine negative Beeinflussung in Form von Ressourcenbelegung oder Datenmanipulation zu vermeiden, wird das Softwareprojekt mit der Sicherungskopie des Web Frontend *flashdb-dev* und der *development*-Datenbank arbeiten.
In Abbildung 4.1 ist der funktionelle Ablauf der Qt-Anwendung skizziert. Je dicker ein Pfeil, desto mehr Informationen enthält der Datenstrom.

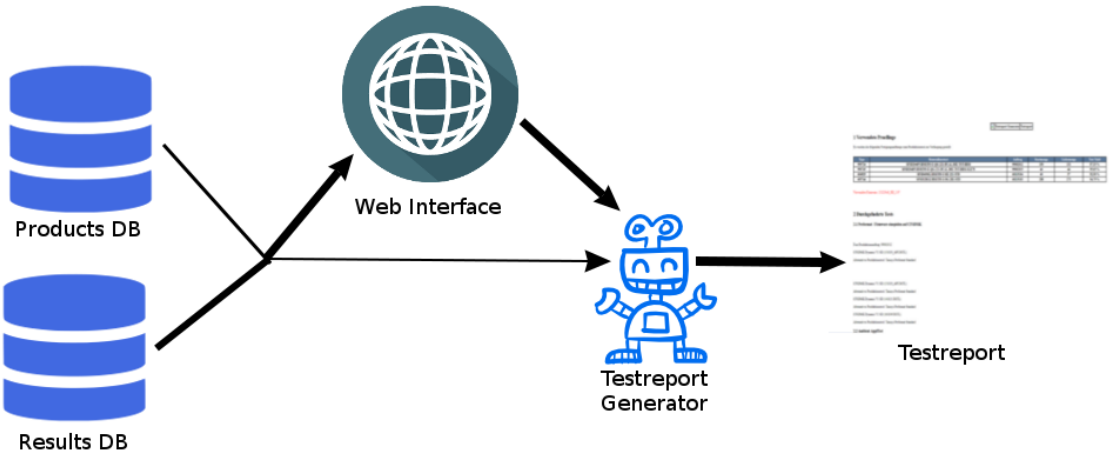



**Abb. 4.1.:** Funktioneller Entwurf

Des weiteren sollte jede Eingabe und Übergabe von Werten auf Ihre Richtigkeit überprüft werden, dies geschieht mit eigens dafür entwickelten Algorithmen. Bei Produktionsauftragsnummern wird kontrolliert ob es sich um eine reine Zahlenkombinatione handelt und ob diese acht Stellen hat. Bei der Datenübergabe ist es wichtig diese ebenfalls nach bestimmtem Kriterien zu untersuchen. Jede Information wird als *QString* übergeben, da Qt viele Funktionen anbietet diese zu untersuchen. Eine Außnahme ist die Testzeitdauerberechnung, nach der Berechnung wird der Datentyp Integer zu QString umgewandelt. Die Darstellungsform des Life Time Monitoring (LTM) wird wegen besserer Eingängkeit beibehalten (Abb 4.2).

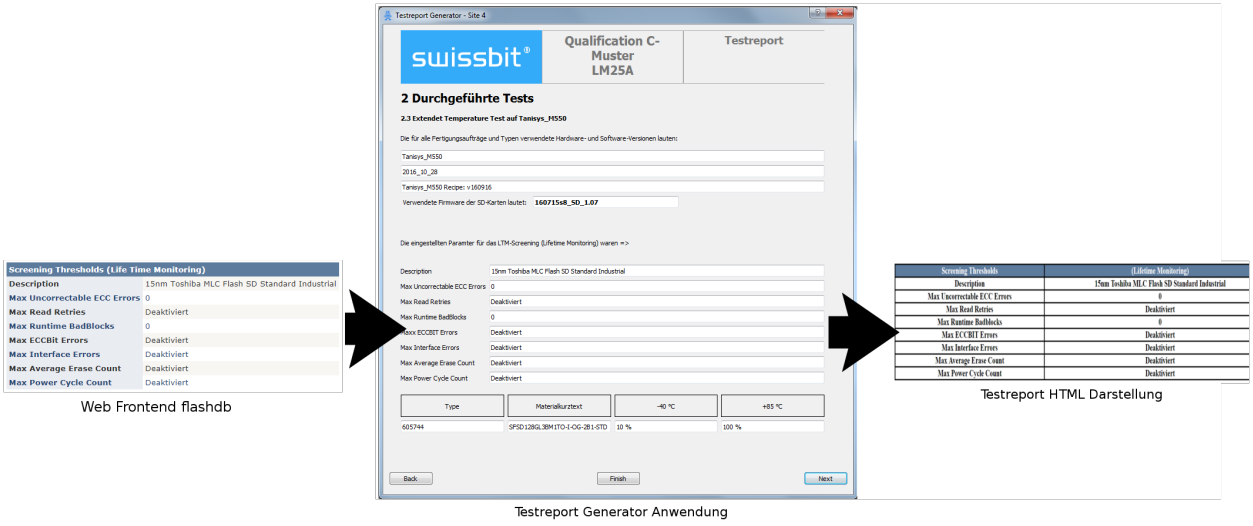



**Abb. 4.2.:** Funktioneller Entwurf

## 4.2.2 Klassenentwurf

Das Unified Modeling Language (UML) Klassendiagramm ist kompakt gehalten und zeigt zur besseren Übersicht nur die wichtigsten Funktionen und Klassen (Abb. 4.3).

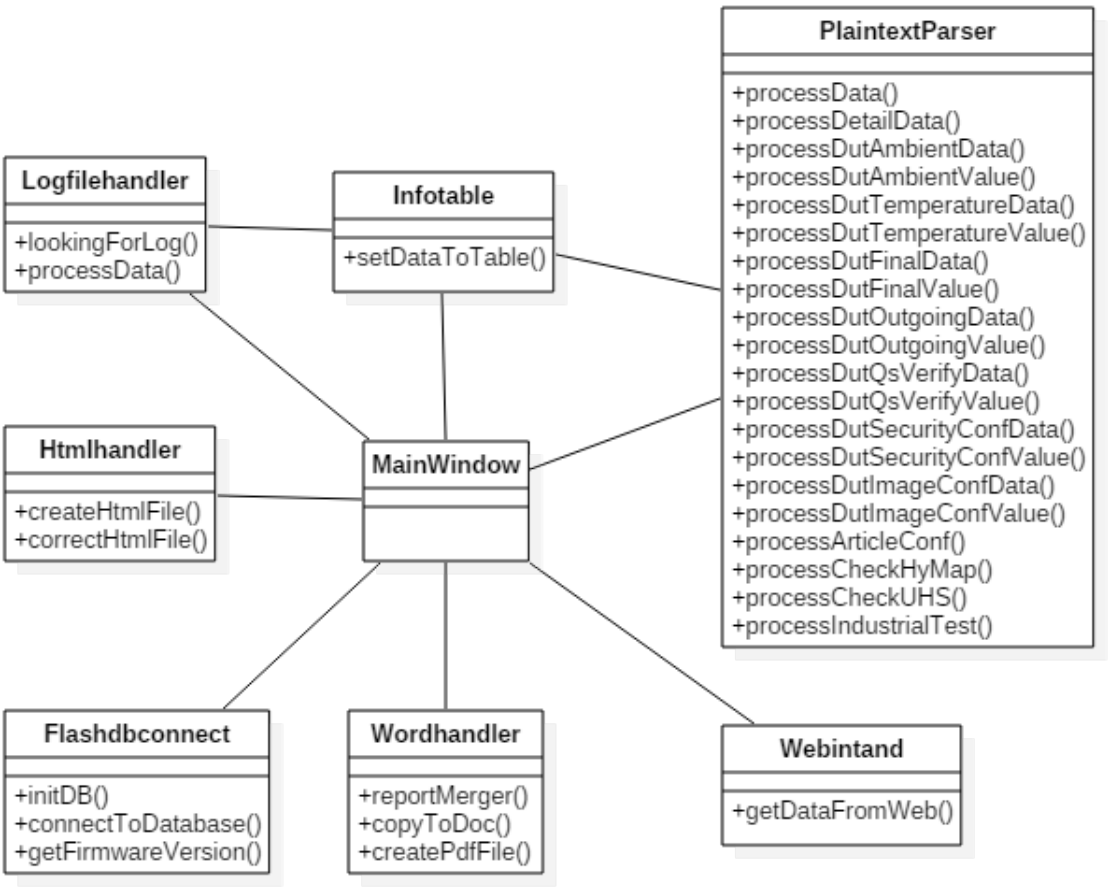



**Abb. 4.3.:** Klassendiagramm Testreport-Generator

Die *MainWindow* Klasse ist die Verwaltungsklasse. Sie kümmert sich um die ordnungsgemäße Abfolge und den Transport der Daten zwischen den Klassen. Infotable ist der Zwischenhändler der Info-Getter, dies sind Funktionen die Informationen einholen.

### 4.2.3 Benutzeroberflächenentwurf

Das GUI Design richtet sich nach den Vorgaben der **EN ISO 9241** Norm. Qt bietet ein Tool, namens Qt-Designer, zur GUI Gestaltung. Es ermöglicht ein Zusammenstellen verschiedener Widgets mittels *Drag and Drop*.
Das Hauptfenster der Anwendung ist schlicht gestaltet. Das linke Textfeld ist das Eingabefeld indem die PA Nummern eingegeben werden (Abb. 4.4). Das rechte Textfeld ist ein Textbrowser dessen Zweck eine Statusanzeige ist. Die sich im unteren Abschnitt befindliche Fortschrittsanzeige, dient einmal der

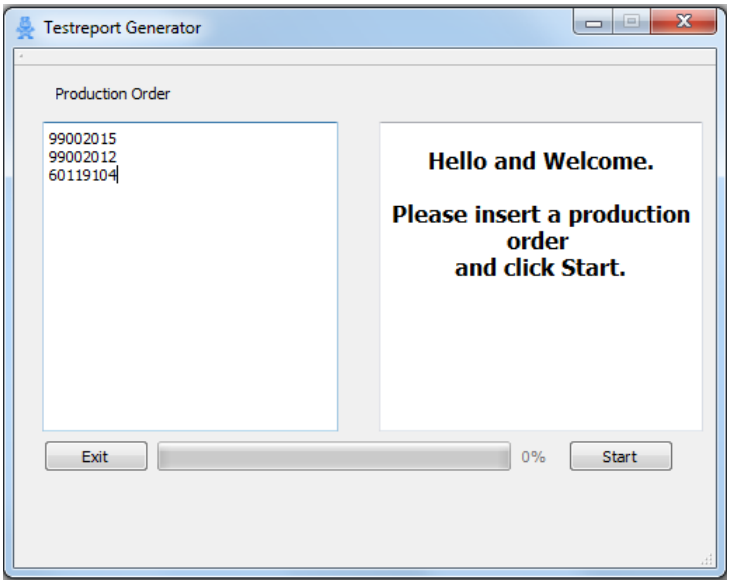



**Abb. 4.4.:** Hauptfenster Testreport Generator

Anzeige des Fortschritts und um feststellen zu können ob sich das Programm aufgehängt hat. Wenn sich keine Animation der Fortschrittsanzeige zu erkennen ist, hat sich die Anwendung aufgehängt. Der *Exit*-Button beendet das Programm, auch während des Suchvorgangs. Der *Start*-Button startet die Generierung des Testreports.
Die Kontrollfenster sind nach der Testreport Vorlage gestaltet. Dies führt dazu, dass die gewohnte Informationsdarstellung gegeben ist und dadurch alles auf einen Blick ersichtlich ist (Abb. 4.5).

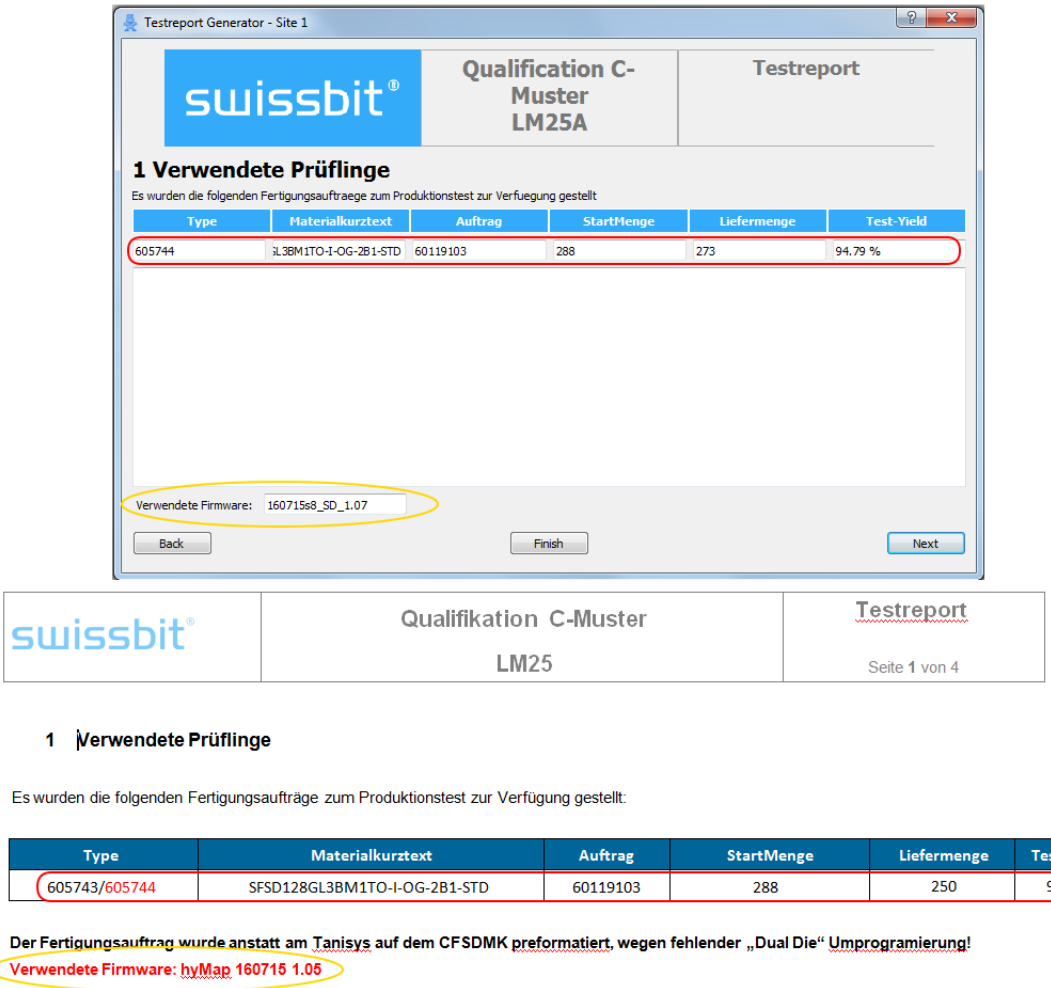



**Abb. 4.5.:** Informationsdarstellung - Anwendung/Word Vorlage

Das Texteingabefeld in Abbildung 4.5 dient als Kommentarmöglichkeit. Diese Kommentare werden unter der entsprechenden Hauptkategorie angehangen. Ferner ist jede Information editierbar. Der Kopf des Testreport Generators besteht aus drei nebeneinander angeordneten Labels. Das erste Label ist der Platzhalter für das Firmenlogo. In der Mitte steht die Bezeichnung der Entwicklungsteile und im dritten Label der Titel.

## 4.3 Implementierung

### 4.3.1 Entwicklungsumgebung und andere Komponenten

### 4.3.2 Projekte und Dateien

### 4.3.3 Webinterface

### 4.3.4 FlashDB

### 4.3.5 Datenverarbeitung

### 4.3.6 GUI Design

# Tests

5

# 6 Zusammenfassung und Ausblick

# Literatur

[Fuc10] Matthias Fuchs. *Vergleich von Cross-Platform GUI-Toolkits*. VDM Verlag, 11. Apr. 2010. 120 S. (zitiert auf Seite 14).

[JB09] Mark Summerfield Jasmin Blachette. *C++ GUI Programmierung mit Qt4*. 2-te Auflage. ISBN 978-3-8273-2729-1. Addison Wesley Verlag, 2009 (zitiert auf Seite 6).

[Ker08] Sascha Kersken. *IT-Handbuch für Fachinformatiker*. Galileo Press, 2008 (zitiert auf den Seiten 11, 12).

## Webseiten

[Qt17a] Qt. *Qt - About Qt*. EN. Qt. Juli 2017. URL: http://wiki.qt.io/About_Qt (zitiert auf Seite 5).

[Qt17b] Qt. *Qt - Bild Signale und Slots*. Qt. Juli 2017. URL: http://doc.qt.io/qt-4.8/images/abstract-connections.png (zitiert auf Seite 7).

[Qt17c] Qt. *Qt - Lizenz*. En. Qt. Juli 2017. URL: https://www.qt.io/qt-licensing-terms/ (zitiert auf Seite 10).

[Qt17d] Qt. *Qt - QFile*. EN. Qt. Juli 2017. URL: http://doc.qt.io/qt-5/qfile.html (zitiert auf Seite 9).

[Qt17e] Qt. *Qt - qmake*. EN. Qt. Juli 2017. URL: http://doc.qt.io/qt-4.8/qmake-manual.html (zitiert auf Seite 6).

[Qt17f] Qt. *Qt - QObject*. EN. Qt. Juli 2017. URL: http://doc.qt.io/qt-5/qobject.html#details (zitiert auf Seite 8).

[Qt17g] Qt. *Qt - Qt Sql*. En. Qt. Juli 2017. URL: http://doc.qt.io/qt-5/qtsql-index.html (zitiert auf Seite 10).

[Qt17h] Qt. *Qt - QtWebEngine*. EN. Qt. Juli 2017. URL: http://doc.qt.io/qt-5/webengine-overview.htmlhttps://wiki.qt.io/QtWebEngine (zitiert auf Seite 9).

[Qt17i] Qt. *Qt - QXmlStreamWriter Klasse*. EN. Juli 2017. URL: http://doc.qt.io/qt-4.8/qxmlstreamwriter.html (zitiert auf Seite 9).

[Qt17j] Qt. *Qt - Signals and Slots*. EN. Qt. Juli 2017. URL: http://doc.qt.io/qt-4.8/signalsandslots.html (zitiert auf Seite 6).

[Wk9] *EN ISO 9241*. Juli 2017. URL: https://de.wikipedia.org/wiki/EN_ISO_9241#EN_ISO_9241-11_Anforderungen_an_die_Gebrauchstauglichkeit (zitiert auf Seite 12).

# Abbildungsverzeichnis

# Anhang

A

# Eidesstattliche Erklärung

Christoph Franke
Silvio-Meier-Straße 1
10247 Berlin

Ich versichere hiermit, dass ich die vorliegende Masterthesis selbstständig und ohne fremde Hilfe angefertigt und keine andere als die angegebene Literatur benutzt habe. Alle von anderen Autoren wörtlich übernommenen Stellen wie auch die sich an die Gedankengänge anderer Autoren eng anlehnenden Ausführungen meiner Arbeit sind besonders gekennzeichnet.

Diese Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht.

Berlin, den 01.08.2017

_____
(Christoph Franke)