

# **ETIC2 zur Verwaltung von Ventiltests**

Masterarbeit zur Erlangung des  
Master of Advanced Studies ZFH in  
**Informatik**

vorgelegt von

**Andreas Stucki**

geboren am 02.01.1989  
von Glarus Nord, Glarus

eingereicht

**Marc-André Bumann**

**Zürich, 31. August 2017**

# Management Summary

This thesis is concerned with the topic of software quality and testing. The aim of my work is to test the in-house developed firmware for our pressure valves. There is an imbalance of work force of 4 to 1 in the field of software development versus software quality control in the company VAT. It is therefore of paramount importance to develop more tests, which can verify the correct working of the firmware in a fully automated procedure without the need of user inputs. For this purpose, it is not enough to extend the current tests with a few more, but rather a new testing framework is required which can handle a large amount of functional and unit tests and which allows to easily add tests for upcoming functionality in the firmware.

The specific aim of this work is to simplify and improve the evaluation of test results which are generated by the improved TTIC2. The TTIC2 allows a user to specify a suite of tests which are then executed. Moreover, a data provenance concept is required to store test results for later analysis. Due to the amount and size of the tests, the testing framework is required to support distributed execution of tests on different workstations. To achieve these goals, a database is needed to store all the test results and a front-end called ETIC2 is developed for displaying the test information.

This thesis provides solutions for the following 4 main goals:

- The front-end ETIC2 requires a simple and stable user interface which allows for fast evaluation of test results and which provides a summary of all failing tests in a test collection.
- ETIC2 should contain an advanced search functionality to quickly find error messages.
- An export functionality is import for the test results.
- The initial state is only allowed to be defined by previous entries in the SoftwareVersionsDatabase

A step-by-step procedure was used to achieve the above-mentioned goals for the testing framework. In a first step, the existing SoftwareVersionsDatabase, which previously only stored information about the firmware, has been extended such that test results, test information, and hardware requirements for the tests can be saved as well. The test information and hardware requirements first need to be inserted into the individual tests in the database before they can be used by the TTIC2. The obtained test results are now saved in the database which can later be analyzed using the newly developed ETIC2 user interface.

All the goals of this thesis have been successfully achieved. The initial state was defined by the already in use firmware, customer specific settings, and the previous test collection. The design of the SoftwareVersionsDatabase does not allow empty entries but only references to other tables are valid entries. The developed ETIC2 user interface allows a user to choose between three different views of the test results for evaluation: Firstly, test results can be sorted hierarchical according to the used firmware version or secondly, they can be sorted by the different hardware (valves) used in the test. Thirdly, the error messages along with the used firmware as well as hardware can be listed. This view also contains an advanced search functionality over all fields. All these three views of the test data can be easily exported to either PDF or directly printed out.

## Inhalt

<b>1 EINLEITUNG .....</b>	<b>1</b>
<b>2 GRUNDLAGEN .....</b>	<b>2</b>
2.1 SITUATION.....	2
2.1.1 Controller .....	2
2.1.2 Firmware .....	3
2.1.3 Entstehung von Qualitätstest .....	3
2.2 TESTUMGEBUNG .....	4
2.2.1 Ventil Hardware .....	4
2.2.2 Test Hardware (NI PXI) .....	5
2.2.3 Test .....	6
2.2.4 ParameterStructBuild.....	6
2.2.5 TestUpdateFirmware.....	7
2.2.6 TTIC2 (Test Tool).....	8
2.2.7 SoftwareVersionsDatabase.....	9
2.2.8 Firmware Database.....	10
<b>3 ZIELSETZUNG .....</b>	<b>11</b>
3.1 ZIELE ZUM STARTZEITPUNKT .....	11
3.2 ZIELE NACH DER ERSTEN ANALYSE.....	12
3.3 QUANTITATIVE ZIELE .....	13
3.4 QUALITATIVE ZIELE.....	13
3.5 AUFGABENBEGRENZUNG .....	13
<b>4 METHODIK.....</b>	<b>14</b>
4.1 VORGEHENSWEISE .....	14
4.2 HILFSMITTEL.....	16
4.2.1 SoftwareVersionsDatabase .....	16
4.2.2 Test, TestUpdateFirmware, TTIC2.....	18
4.2.3 ETIC2 .....	18
<b>5 AUSWERTUNG DER TESTRESULTATE.....</b>	<b>19</b>
5.1 SOFTWAREVERSIONSDATABASE .....	19
5.1.1 Modellierung Firmware Informationen.....	19
5.1.2 Modellierung Testinformationen .....	21
5.1.3 Modellierung Testresultate.....	23
5.1.4 Modellierung Firmware Bugs .....	25
5.2 TEST .....	26
5.2.1 Abspeicherung Testinformationen .....	26
5.2.2 Testinformationen an TestUpdateFirmware über Umgebungsvariablen.....	28
5.3 TESTUPDATEFIRMWARE.....	29

5.3.1	<i>Anweisung Update Testinformationen</i> .....	29
5.3.2	<i>Abfrage Testinformationen</i> .....	29
5.3.3	<i>Testinformationen in SoftwareVersionsDatabase schreiben</i> .....	29
5.4	TTIC2 .....	30
5.4.1	<i>Auslesung der Testinformationen</i> .....	30
5.4.2	<i>Hinterlegung des Grundzustandes</i> .....	31
5.4.3	<i>Abspeicherung der Testresultate</i> .....	32
5.5	ETIC2 .....	33
5.5.1	<i>Anbindung SoftwareVersionsDatabase</i> .....	33
5.5.2	<i>Codierung nach MVVM</i> .....	33
5.5.3	<i>Ausgabe Bericht</i> .....	38
<b>6</b>	<b>ERGEBNISSE</b> .....	<b>39</b>
6.1	QUANTITATIVE ZIELE .....	39
6.2	QUALITATIVE ZIELE.....	41
<b>7</b>	<b>DISKUSSION</b> .....	<b>45</b>
<b>8</b>	<b>AUSBLICK</b> .....	<b>46</b>
8.1	OFFENE PUNKTE .....	46
8.1.1	<i>Umsetzung Überarbeitung SoftwareVersionsDatabase</i> .....	46
8.1.2	<i>Integration Buglist in ETIC2</i> .....	46
8.1.3	<i>LogFile in ETIC2</i> .....	46
8.2	NÄCHSTE SCHRITTE .....	46
<b>9</b>	<b>VERZEICHNISSE</b> .....	<b>47</b>
	LITERATURVERZEICHNIS .....	47
	ABKÜRZUNGSVERZEICHNIS .....	47
	ABBILDUNGSVERZEICHNIS .....	48
	TABELLENVERZEICHNIS.....	49
	GLOSSAR.....	50
<b>A.</b>	<b>PROJEKTMANAGEMENT</b> .....	<b>51</b>
A.1.	ZEITPLAN .....	51
<b>B.</b>	<b>TESTUMGEBUNG</b> .....	<b>52</b>
B.1.	TTIC2 .....	52
B.2.	ETIC2 .....	53
<b>C.</b>	<b>HILFSMITTEL</b> .....	<b>54</b>
C.1.	CVI .....	54
C.1.1.	<i>Umgebungsvariable</i> .....	54
C.1.1.1.	<i>Gegenüberstellung zu C Library Umgebungsvaible</i> .....	54
C.2.	SQL TOOLKIT .....	55
C.2.1.	<i>Funktionsumfang</i> .....	55

C.2.2	<i>Code Ausschnitt um einen neuen Eintrag zu erstellen.....</i>	56
<b>SELBSTÄNDIGKEITSERKLÄRUNG .....</b>	<b>58</b>	

# 1 Einleitung

«Qualität heisst, weniger Ärger mit den Kunden.»

*Prof. Dr. Hans-Jürgen Quadbeck-Seeger*

Dieses Zitat liefert den Grund zu meiner Anstellung in der Firma VAT sowie der Erstellung dieser Arbeit. Das Hauptziel meiner Tätigkeit ist es die Software Qualität der eigens entwickelten Vakuum Firmwaren sicherzustellen. Das Thema wird in vielen Unternehmen unterschätzt, da es keinen direkten Verdienst nach sich zieht. Das zeigt sich auch in meinem Fall, da die Abteilung vier Software Entwickler zählt hingegen nur mich der diese überprüft. Daraus folgt das oberste Ziel des Anwendertests: Es muss vollkommen automatisiert werden und darf keine Benutzereingaben erfordern. Daraus resultiert nicht nur einen lauffähigen Test sondern eine ganze Testumgebung, welche die Tests immer wieder mit den neuen Funktionen der Firmwaren erweitert.

Das Hauptziel der Arbeit besteht darin, eine Oberfläche das ETIC2 zu erstellt, welche Testresultate anzeigt. Für diese Oberfläche wird der Einsatz einer Datenbank zwingend. Ich entschied mich die bestehende SoftwareVersionsDatabase zu erweitern, da hier schon fundierte Kenntnisse vorhanden waren. Das bis anhin benutzte und selbstprogrammierte Element TTIC2 blieb auch bestehen, wurde aber teilweise erweitert. Dieses Programm wird eingesetzt, um Tests zu einer Kollektion zusammen zu fassen und diese anschliessend auszuführen und zeigt dem Benutzer Testinformationen an und braucht weiter die Testanforderungen bezüglich der Hardware um vor der Ausführung der Testkollektion entscheiden zu können, ob alle Tests ausgeführt werden können. Diese Informationen werden zusätzlich in der SoftwareVersionsDatabase hinterlegt.

Die Arbeit gliedert sich in fünf Hauptteile und endet in einer Diskussion und einem Ausblick. Der erste Teil umfasst die Grundlagen der Arbeit. Es wird erklärt, welche Anwendungen schon bestehen, welche erweitert werden und welche neu dazu kommen. Um das komplexe Thema zu vereinfachen, wurde häufig auf Darstellungen als Hilfsmittel zurückgegriffen. Danach folgen die Kapitel Zielsetzung und Methodik, worin die Ziele und Abgrenzungen der Arbeit nochmals genau beschrieben werden und mit welcher Methode, die Arbeit zu Stande kam. Der fünfte Teil umfasst die Auswertung der Testresultate und ist das Herzstück der Arbeit. Dort wird auch die Entstehung der Oberfläche ETIC2 genauer beschrieben. Zur Verständlichkeit der Arbeit wurde eine Glossar am Ende der Arbeit ein Glossar erstellt.

Das resultierende Produkt dieser Arbeit (ETIC2) wird für den internen VAT Verwendungszweck entwickelt und nicht für den kommerziellen Gebrauch konzipiert.

## 2 Grundlagen

Das Grundlagenkapitel ist in zwei wichtige Teile unterteilt. Im einen Teil wird die Umgebung, die Situation und der aktuelle Stand sozusagen die Grundvoraussetzungen beschrieben. Im darauf folgenden Teil wird die Hauptaufgabe dieser Masterarbeit nämlich die Erweiterung der Testumgebung beschrieben. Dazu gehört; was schon entwickelt wurde, wo es Anpassungen geben wird und wie das neue Produkt das Evaluation Tool Integrierter Controller 2 (ETIC2) angedacht ist.

### 2.1 Situation

Dieses Unterkapitel beschreibt die zentrale Steuereinheit, die für die Testumgebung entwickelt wird. Diese wird benötigt, um ein Ventil steuern zu können. Das Verhalten des Ventils wird durch die Software auf dem Controller gesteuert, welche auf Anweisungen des Benutzers wartet. Diese Verhalten werden durch Tests überprüft und erhöhen so die Software Qualität.

#### 2.1.1 Controller

Die Firma VAT stellt Vakuumventile her für die Halbleiter- und Medizinalindustrie, für deren Forschung und Entwicklung sowie für die Automobilindustrie (VAT Group AG, 2017). VAT ist im Bereich der Herstellung von Vakuumventilen mit einem Marktanteil von über 40% klarer Weltmarktführer. Die Firma ist bekannt für ihre Regelventile, bei denen ein Controller die Steuerung dieser Vakuumentventile übernimmt (Abb. 1). Dieser Controller ist modular aufgebaut und besteht grob gesagt aus drei Komponenten. Die wichtigste Komponente ist das Masterboard, welches mit den zentralen Elementen bestückt ist. Dieses ist unerlässlich und wird jeweils an die gewünschte Ventilhardware angepasst. Das Herzstück des Controllers ist der Mikrocontroller, für den VAT eine eigene Firmware entwickelt. Dazu ist oder sind Motorbausteine nötig, die eine weitere Firmware von externen Lieferanten benötigt. Die zweite Komponente ist das Interface Board. Dieses wird nach Kundenwunsch angefertigt. Falls der Kunde mit einem Feldbusssystem arbeitet, wird eine Interface Firmware nötig. Der Nutzen von Feldbusssystemen ist, dass von einem Host aus mehrere Teilnehmer angesprochen werden können. Die dritte Komponente ist die Option Unit, die Zusatzfunktionen nach Wunsch beinhaltet. Der Kunde kann mit Hilfe des Controllers seine Sensoren direkt speisen. Eine weitere Option ist, dass bei Spannungsabfall das Ventil eine vordefinierte Position einnimmt. Verwendet der Kunde kein Feldbusssystem, so kann er mit Hilfe eines Cluster Systems oder über die RS485 Interface Schnittstelle mehrere Ventile ansprechen. (Marugg, 2010).

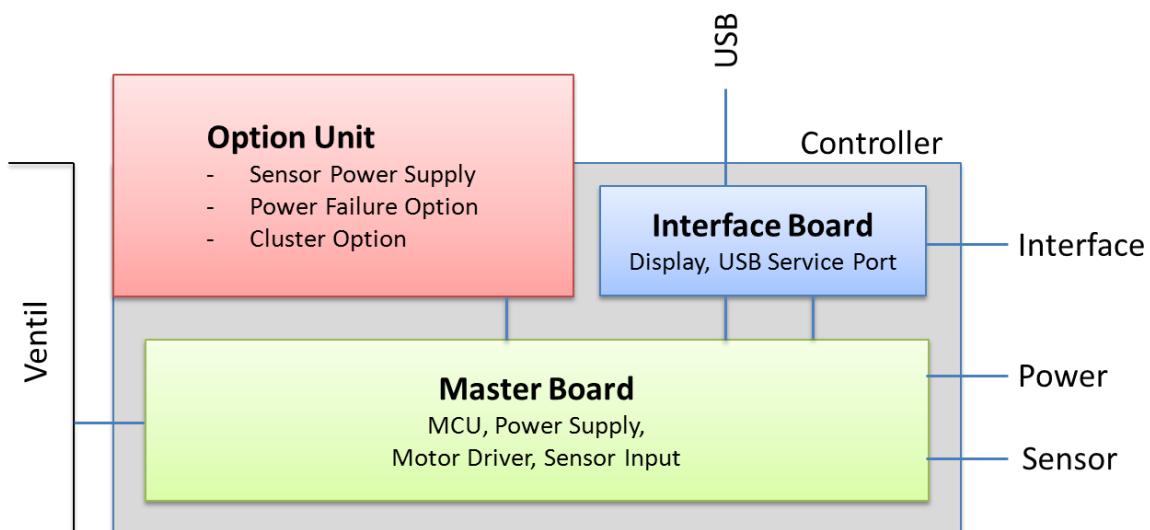
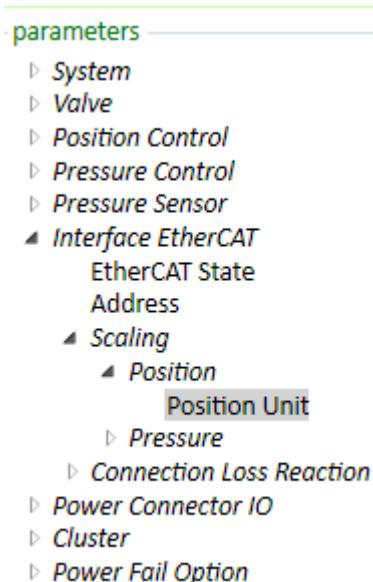


Abbildung 1: Basiskonzept Ventil Controller (Marugg, 2010)

## 2.1.2 Firmware



Wie bereits erwähnt, ist das Masterboard mit einem Mikrocontroller bestückt. Die Gründe für den Einsatz eines Mikrocontrollers liegen bei den niedrigen Anschaffungskosten und den fundierten Programmierkenntnissen der Mitarbeiter mit der Programmiersprache C. Auf dieser läuft die von VAT entwickelte Firmware, welche ohne Betriebssystem auskommt. Die Firmware wartet ständig auf neue Befehle des Benutzers und führt diese nach Ablauf der Zykluszeit immer wieder aus. Um den Anwendern das Leben zu erleichtern, wurde eine Parameterbaumstruktur eingefügt, bei der jeder Parameter eine Funktionalität des Ventils wiederspiegelt. Dabei kann es sich um Parameter handeln, die gesetzt werden oder die reine Statusinformationen abfragen können. Dies passiert entweder über den Service (USB) oder über den Interface Kanal. Die Parameter sind in vier Ebenen unterteilt, wie sie Abbildung 2 zeigt. Das soll den Anwendern bei der Suche nach der gewünschten Einstellung helfen.

Abbildung 2: Parameterbaumstruktur der Software

## 2.1.3 Entstehung von Qualitätstest

Aufgrund der Komplexität dieser Firmware und der damit verbundenen zunehmenden Fehlerquote, wurde ich von der Firma VAT eingestellt und beauftragt die Qualität der Firmware sicherzustellen. In der Vergangenheit wurde diese Aufgabe vom Software Entwickler selbst übernommen. Dabei kam es immer wieder vor, dass die Qualitätsprüfung aus Zeitgründen sehr schmal ausfiel. Die Basis bildeten Excel Listen mit Ventilbefehlen, welche sequentiell abgespielt wurden und Überprüfkommandos beinhalteten. Die Schwachstellen des Systems lagen sowohl in der Wartung, als auch in den begrenzten Funktionsmöglichkeiten, welche durch visuelle Überprüfung des Testers übernommen werden mussten.

Um diese Probleme für die neu integrierte Controller Generation zwei (IC2) zu lösen, musste ein neues System gefunden werden. Das primäre Ziel war es völlig automatische Tests zu entwickeln, welche keinen Einfluss des Testers benötigten. Dadurch war schnell klar, um Berechnungen durchzuführen sowie die Firmware Funktionalität zu prüfen, wird der Einsatz einer Programmiersprache nötig. Zudem soll eine schnelle und einfache Auswertung der Tests möglich sein. Dies erfordert nicht nur den Einsatz eines Tests, sondern einer ganzen Umgebung, welches im nächsten Kapitel näher erläutert wird.

## 2.2 Testumgebung

Bei der Testumgebung werden die einzelnen Elementen der Testumgebung näher beschrieben, welche in der Abbildung 3 ersichtlich sind. Aus Gründen der Leserfreundlichkeit, wird die Beschreibung der Elemente in drei Teile aufgeteilt, und zwar in Ist-Zustand, Stärken und Schwächen sowie in die Problemstellung.

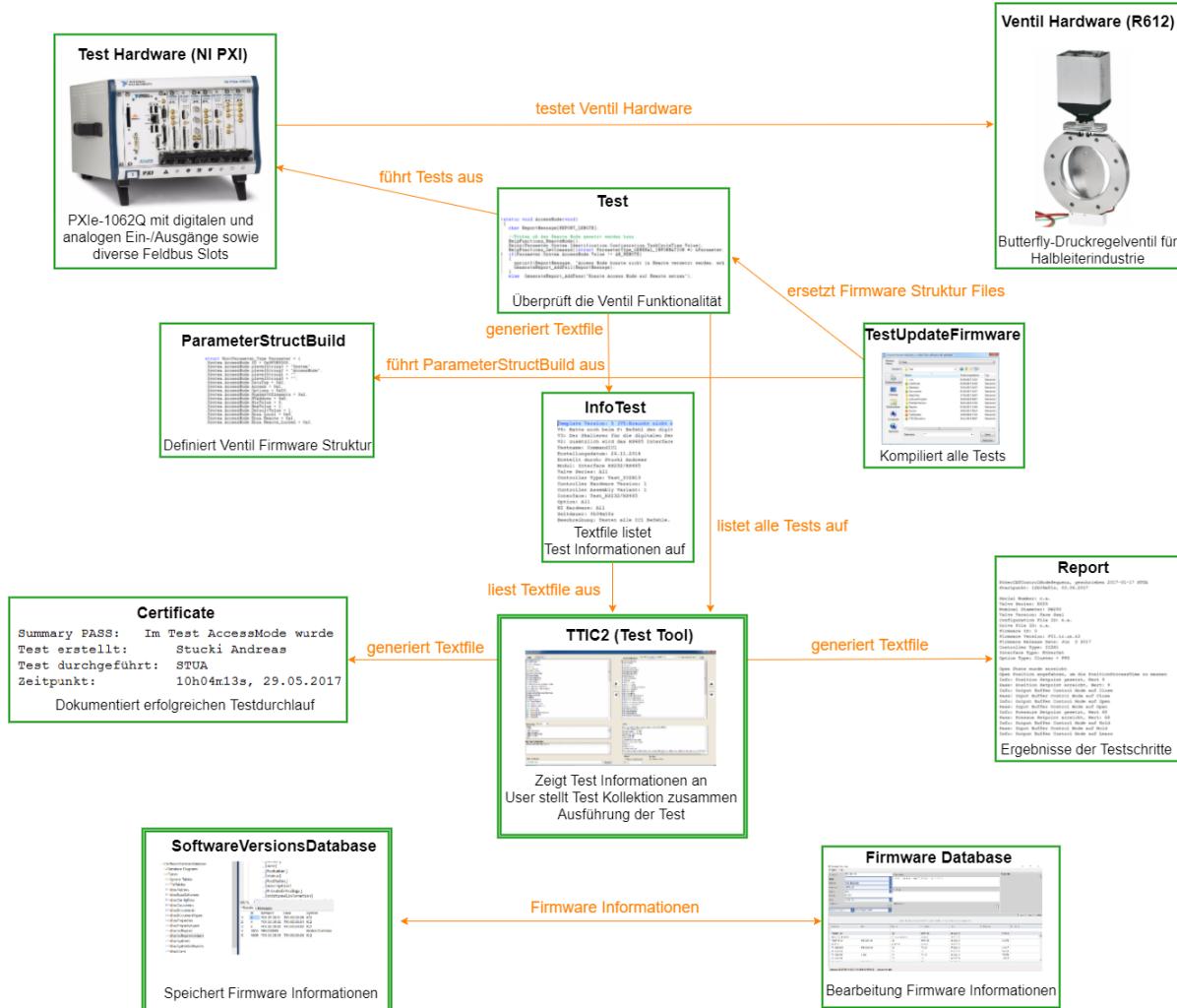


Abbildung 3: Ist-Zustand der Testumgebung

### 2.2.1 Ventil Hardware

#### 2.2.1.1 Ist-Zustand

Mit Hilfe der Testumgebung werden unterschiedliche Ventil Hardwares getestet. Dabei wird der Controller der jeweiligen Ventil Hardware Anforderungen angepasst. Diese Anforderungen beinhalten Unterschiede in der Bestückung des Masterboards zum Beispiel die Anzahl Motoren oder deren Leistung. Das bedeutet, dass es unterschiedliche Controller Ausführungen gibt. Der Grundgedanke am Aufbau des Controllers mit den drei Boards bleibt jedoch immer gleich sowie die Struktur der Ventil Firmware. Um die Hardware Eingänge des Controllers überprüfen zu können, wird eine Test Hardware benötigt, mit welcher analoge und digitale Signale erzeugt und ausgelesen werden können.

### **2.2.1.2 Problemstellung**

Über den Lebenszyklus eines Controllers werden immer mehr verschiedene Ventil Hardwares unterstützt. Dies hat zur Folge, dass unterschiedliche Controller Ausführungen erstellt werden müssen. Um diesem Umstand gerecht zu werden, ist eine flexible Testumgebung zwingend. Es müssen alle Controller Typen mit den Tests qualifiziert werden können.

### **2.2.2 Test Hardware (NI PXI)**

#### **2.2.2.1 Ist-Zustand**

Der Controller hat mehrere Schnittstellen zum Hostsystem des Vakuumkammer Betreibers. Jeder Controller hat einen USB Stecker, welcher verwendet wird, um mit einer Service Applikation das Ventil in Betrieb zu nehmen. Im Betrieb wird das Ventil vom Host über die Interface Schnittstelle angesprochen. Heutzutage spricht der Host seine Komponenten immer häufiger mit Hilfe einer Feldbus Lösung an.

Für den Low-end Markt wird der Controller mit einer Logik Interface ausgerüstet. Dabei steuert der Host das Ventil mit Hilfe von analogen und digitalen Signalen. Auch die Status Informationen des Ventils werden über analoge und digitale Signale an den Host zurückgeliefert.

Um das Ventil wie auch den Controller betreiben zu können, braucht es einen Power Stecker. Dieser Stecker bietet zusätzlich eine Safety Möglichkeit, mit dieser kann das Ventil durch die Hardware geöffnet und geschlossen werden.

Sämtliche dieser Funktionalitäten müssen mit Hilfe der Test Hardware emuliert und geprüft werden können. Dabei bietet National Instruments (NI) eine flexible Kartenbestückungslösung an, um über eine grafische Programmiersprache (LabVIEW) oder einer textbasierten Programmiersprache (CVI) das PXI System ansprechen zu können. Dieses System ermöglicht eine kurze Entwicklungszeit der Prüfsoftware für die Controller Hardware, da es sehr viele Hilfsfunktionalitäten zur Verfügung stellt. Dabei gibt es zwei Einschränkungen bezüglich der Speisung und der Unterstützung der CCLink Interface Anbindung. Die Speisung wird mit Hilfe eines separaten Speisegeräts per Software (LabVIEW) gesteuert. Die Kommunikation über CCLink mit dem Controller wird über ein Drittanbieterprodukt erreicht.

#### **2.2.2.2 Stärken**

- Schnelle Inbetriebnahme der Hardware über die NI Programmierumgebung
- Fast alle Hardware Schnittstellen können über die NI Hardware getestet werden
- Das System kann solange erweitert werden, bis alle acht Steckplätze besetzt sind
- Engineering Support durch NI

#### **2.2.2.3 Schwächen**

- Jährliche Lizenzkosten für die Programmierumgebung
- Hohe Anschaffungskosten

#### **2.2.2.4 Problemstellung**

Die Test Hardware wurde zu einem früheren Zeitpunkt evaluiert und angeschafft. Da damals noch nicht alle Steckplätze mit Karten ausgestattet waren, kann es heute noch zu Erweiterungen kommen. Der Grund für die Steckplätze ohne Karten liegt in der Vergrösserung des Portfolios an Ventilen, welche mit dem IC2 Controller angesprochen werden. Diese einzelnen Controller unterscheiden sich in Bezug auf den Power Stecker. Somit steigt der Bedarf an analogen- und digitalen Karten an.

## 2.2.3 Test

### 2.2.3.1 Ist-Zustand

Bei den geschriebenen Tests handelt es sich um automatisierte Anwendertests, welche mit der CVI Programmierumgebung erstellt wurden. Hierin werden die Ventil Funktionalitäten im laufenden Betrieb geprüft. Diese Tests werden eingesetzt, um eine Ventil Firmware zu qualifizieren. Dabei werden diese meistens übers Wochenende oder über Nacht ausgeführt, so muss der Anwender nicht mehr aktiv in die Ausführung eingreifen. Die Tests werden bei der zugehörigen Ventil Firmware in der SVN hinterlegt, um bei allfälligen Fehlern im Feld eine Analyse durchführen zu können. In den Tests werden die Anforderungen bezüglich der Ventil Hardware definiert. Wichtig ist dabei, dass die Tests möglichst universell eingesetzt werden können.

Im Test wird mit Parameternamen gearbeitet, um den Code einfacher lesen zu können. Die Identifikationsnummer kann sich so ändern, ohne dass eine Anpassung des Tests notwendig wird. Diese Informationen werden durch die erzeugten Files des ParameterStructBuild Programms bezogen. Die Testfunktionalitäten werden in Files definiert, welche in allen Tests verwendet werden. Im Hauptfile wird der Ablauf eines Tests definiert.

### 2.2.3.2 Stärken

- Vollautomatische Tests
- Testfunktionen nur in einem File, der Rest sind Library Files
- In einer Struktur sind alle Ventil Parameter sowie seine Enum Werte hinterlegt

### 2.2.3.3 Schwächen

- Testinformationen werden in einem Textfile hinterlegt
- In den Testinformationen können nicht mehrere Werte für ein Attribut definiert werden

### 2.2.3.4 Problemstellung

Im Test werden einerseits Informationen zur Kennzeichnung des Tests hinterlegt. Zu diesen gehören eine Beschreibung, die aktuelle Version, der Autor und das Erstellungsdatum. Anderseits wird definiert für welche Ventil Hardware der Test ausführbar ist. Diese Informationen werden im gleichen Ordner wie die Tests in einem Textfile hinterlegt. Dieses Textfile wird erzeugt, wenn das Test Tool Integrierter Controller 2 (TTIC2) Programm kein Textfile im Testordner finden kann. So kann der Fall auftreten, dass die Informationen im Test im Vergleich zum Textfile nicht übereinstimmen.

## 2.2.4 ParameterStructBuild

### 2.2.4.1 Ist-Zustand

Dieses Programm liest die aktuellen Parameter der Firmware aus und speichert diese in Strukturform ab. Daraus resultiert ein Header File, in welchem die Struktur definiert wird. Als zweites wird ein Source File generiert, welches den Inhalt der Struktur füllt. Diese mehrdimensionale Struktur spiegelt den Parameterbaum wieder. Wichtige Elemente bilden dabei die Identifikationsnummer, welche aus einer achtstelligen Hex-Zahl besteht. Dabei werden jeweils zwei Stellen für eine Ebene verwendet. Anschliessend werden die einzelnen Ebenen in Textform definiert. Der Datentyp, der Zugriff und der Wertebereich des Parameters werden ausgelesen. Dazu enthalten einige Parameter Enum Werte, welche den Parameterwert in Textform beschreibt. Diese werden auch in dieser Strukturform hinterlegt. Sämtliche Informationen werden später in den Tests genutzt, sodass der Test Schreiber sieht, welche Parameter von der Firmware unterstützt werden. Er muss keine Kenntnisse über die Identifikationsnummer haben. Die Enum Werte erlauben zudem eine Auflistung aller zulässigen Parameterwerte und eine einfachere Lesbarkeit.

#### **2.2.4.2 Stärken**

- Automatische Erstellung einer Struktur mit allen Parameter Informationen
- Kann für alle IC2 Generation Controller verwendet werden

#### **2.2.4.3 Schwächen**

- Es braucht eine aktive Verbindung über USB mit dem Controller
- Weitere Applikationen nötig um die zyklischen und azyklischen Parameter der Feldbusssysteme auslesen zu können

#### **2.2.4.4 Problemstellung**

Da das Programm sehr weit ausgereift ist, gibt es prinzipiell kein Problem. Man könnte kleinere Anpassungen erstellen, wie zum Beispiel die Feldbus Objekte zu integrieren. Dies wird zum entsprechenden Zeitpunkt realisiert, sobald das Interface fertig entwickelt ist.

### **2.2.5 TestUpdateFirmware**

#### **2.2.5.1 Ist-Zustand**

Die TestUpdateFirmware ruft zuerst das ParameterStructBuild auf, um die Parameterstruktur Files zu erstellen. Anschliessend werden die alten Parameterstruktur Files mit den neuen ersetzt. Diese Files sind in allen Tests sichtbar, da alle Library Files wie auch die Parameterstruktur Files in einem gemeinsamen Ordner liegen. Der Benutzer kann jetzt angeben, in welchem Testordner alle Tests mit den neuen Parameter Informationen neu kompiliert werden sollen. Wenn alle Tests kompiliert sind, erscheint ein Fenster mit der Auflistung der Tests, die mit Fehlern kompiliert wurden. Weiter gibt der Benutzer auch den Ordner an, wo das TTIC2 Programm hinterlegt ist, um auch hier die neusten Parameterstrukturen im Programm zu verwenden.

#### **2.2.5.2 Stärken**

- Die Parameter Files werden automatisch ersetzt
- Es werden automatisch alle Tests mit den neusten Parameter Informationen kompiliert
- Auflistung über Tests, welche mit Fehlern kompiliert wurden

#### **2.2.5.3 Schwächen**

- Die Tests werden auf der Konsole kompiliert → Arbeit am PC wird eingeschränkt
- Mit der wachsenden Anzahl von Tests braucht der Vorgang seine Zeit

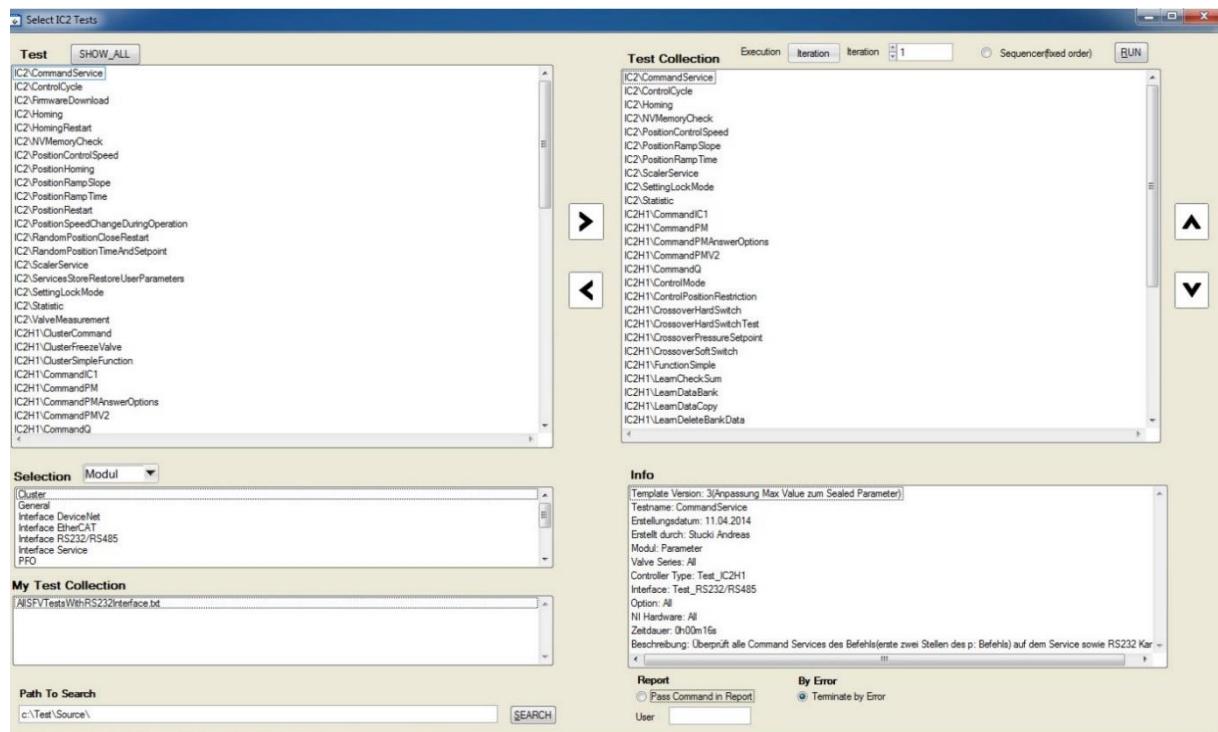
#### **2.2.5.4 Problemstellung**

Aktuell werden immer alle Tests mit den entsprechenden Parameter Informationen neu kompiliert. Besitzt das Ventil nicht den gleichen Controller Typ, so kann der Test nicht ausgeführt werden. Der Grund liegt darin, dass die Firmware nur Parameter besitzt, welche durch die Controller Hardware ausgeführt werden. Somit sind die Controller spezifischen Parameter, welche im Test gebraucht werden, bei der Auslesung der Firmware nicht mehr definiert. Aus diesem Grund, müssen diese Parameter im Test ausgeblendet werden, um einen Fehler bei der Kompilation des Tests zu vermeiden.

## 2.2.6 TTIC2 (Test Tool)

### 2.2.6.1 Ist-Zustand

Wird das TTIC2 gestartet, wird der Benutzer als erstes aufgefordert den Pfad auszuwählen, aus welchem die Tests geladen werden sollen. Dabei werden nur Tests in der linken Spalte aufgelistet, welche mit dem aktuell angeschlossenen Controller ausgeführt werden können (Abbildung 4). Nun können die gewünschten Tests ausgewählt werden und in die rechte Spalte verschoben werden, um die Tests später auszuführen. Unter dem Punkt Selection gibt es verschiedene Filtermöglichkeiten. Es können einzelne oder mehrere Filter gesetzt werden. Wird ein Filter gesetzt, so wird in der linken Spalte die Auflistung der verfügbaren Tests angepasst. Weiter kann unter My Test Collection eine vorher abgespeicherte Test Kollektion geladen oder eine neue erstellt werden. Dabei werden alle Tests in der rechten Spalte zu einer Kollektion zusammengefasst. Zusätzlich können spezifische Testeinstellungen wie beispielsweise «soll der Test bei einem Fehler abgebrochen werden» und «wie viele Informationen soll der Report liefern» vor der Ausführung definiert werden. Mit dem Drücken des Run Knopfes werden die Tests gestartet.



**Abbildung 4: Ansicht der TTIC2 Oberfläche für die Auswahl der Testkollektion**

Anschliessend wird geprüft, ob alle Tests mit der angeschlossenen Hardware ausgeführt werden können. Wenn dies nicht der Fall ist, kann der Benutzer entscheiden, ob die Tests trotzdem gestartet werden sollen und der Benutzer während des Ablaufs der Tests die Ventil Hardware wechselt.

Nach der Überprüfung wird ein Reportfenster geöffnet. Dieses zeigt bei der Ausführung die aktuellen, die bereits ausgeführten und die anstehenden Tests. Eine Ansicht des Reportfensters ist im Anhang zu finden (B.1. TTIC2). Obendrein wird jedes Testergebnis notiert. Wird ein Fehler detektiert, wird dieser rot hervorgehoben. Der Anwender hat immer die Möglichkeit die Test Kollektion abzubrechen.

Ist die Test Kollektion ausgeführt, kann der User den Report am gewünschten Ort abspeichern. Wird diese Option nicht gezogen, wird der Report gelöscht. Für jeden erfolgreichen Test wird ein Zertifizierungsfile erstellt. Bei allen fehlerhaften Tests wird ein Diagnistikfile erstellt.

### **2.2.6.2 Stärken**

- Alle Funktionalitäten sind auf der Oberfläche ersichtlich (keine Verschachtelungen)
- Die verfügbaren Tests können nach Hardware Eigenschaften gefiltert werden
- Wird ein einzelner Test angewählt, so wird eine Beschreibung des Tests sowie die Hardware Anforderungen angezeigt
- Es können Testkollektionen abgespeichert werden
- Der fortlaufende Report wird auf der Oberfläche angezeigt und im Hintergrund in einem Textfile hinterlegt
- Die automatische Generierung von Zertifizierungsfiles

### **2.2.6.3 Schwächen**

- **Das Programm wird auf mehreren Rechnern ausgeführt**
  - Erschwerte Auswertung der Tests
  - Die abgespeicherten Testkollektionen sind nur auf dem jeweiligen Rechner sichtbar
- **Keinen Verlauf der Testergebnisse der verschiedenen Ventil Firmwaren ersichtlich**
  - Letzter Reportfile wird im SVN abgelegt
  - Keine schnelle Suche, ob der Testfehler schon einmal aufgetreten ist
  - Fehlermeldung nur im Reportfile ersichtlich
- Keine Sicherstellung des Grundzustandes

### **2.2.6.4 Problemstellung**

Das Problem ist, dass aktuell nach der Ausführung der Testkollektion, das entstandene Report File manuell und zeitaufwändig nach fehlerhaften Testdurchläufen durchsucht werden muss. Das Reportfile enthält hinzukommend alle Testschritte und erreicht dadurch eine sehr grosse Datenmenge. Deshalb wird nur der letzte Report bei einer Ventil Firmware Freigabe in der SVN abgelegt. Dies erschwert die Auswertung der Tests enorm. Auch das Ablegen der Zertifizierungsfiles erhöht die Datenmenge weiter und kommt deshalb nochmals erschwerend hinzu.

Beim Start des TTIC2 Programms liest die Oberfläche die Testinformationen über ein Textfile aus. Wie schon unter dem Punkt 2.2.3 Test erwähnt, können diese Informationen nicht immer aktuell sein.

## **2.2.7 SoftwareVersionsDatabase**

### **2.2.7.1 Ist-Zustand**

Die SoftwareVersionsDatabase ist eine SQL Datenbank, welche aktuell genutzt wird um die Firmware Informationen der IC2 Controller Generation abzuspeichern. Der Grund liegt darin, dass die VAT SQL Server im Einsatz hat.

Dabei werden verschiedene Typen von Firmwares hinterlegt, wie z.B. Ventil, Motion Controller sowie auch Interface. Dabei wird auch definiert, welche Ventil Firmware mit welchem Motion- und Interface Firmwares lauffähig ist.

### **2.2.7.2 Stärken**

- Firmware Informationen sind für viele Benutzer ersichtlich
- Firmware Informationen können editiert werden

### **2.2.7.3 Schwächen**

- In der Modellierung wurden die Regeln der Normalform nur teilweise umgesetzt
- Ungeeignete Namensgebung für Zwischentabellen

#### **2.2.7.4 Problemstellung**

Die Datenbank enthält aktuell Firmware Informationen. Die Modellierung beachtet nicht alle Regeln zur Normalform und es werden Funktionen der Modellierung durch die Firmware Database Applikation übernommen. Das Modell soll angepasst sowie die aktuellen Daten migriert werden. Weiter wird eine Erweiterung nötig, um die Testresultate, Testinformationen und Firmware Bugs darin ablegen zu können.

### **2.2.8 Firmware Database**

#### **2.2.8.1 Ist-Zustand**

Die Firmware Database Applikation ist eine in C# realisierte Oberfläche, mit der die Firmware Informationen gelesen und bearbeitet werden. Die Oberflächenansicht besteht im Grunde aus zwei Bereichen. Im oberen Bereich kann ein einzelner Eintrag bearbeitet werden. Im unteren Bereich werden alle Firmware Einträge aufgelistet.

#### **2.2.8.2 Stärken**

- Viele Such- und Filtermöglichkeiten der einzelnen Firmware Einträge
- Viele Firmware Einträge besitzen vordefinierte Auswahlfelder

#### **2.2.8.3 Schwächen**

- Noch keine Benutzerverwaltung (Diese wird gebraucht, um zum Beispiel die gefunden Fehler in der Firmware beschreiben zu können, welche der Verkäufer nicht sehen sollte)
- Keine Refresh Möglichkeit (Applikation muss neu gestartet werden)

#### **2.2.8.4 Problemstellung**

Die Applikation übernimmt aktuell Funktionen der Datenbankmodellierung. Weiter werden auch die Datenbanktypen in der Applikation verwendet. Werden Firmware Einträge geändert, so sind diese nur in dieser Applikation ersichtlich. Jedoch wird diese Applikation von mehreren Usern genutzt, welche die Applikation meistens geöffnet lassen und somit keinen Zugriff auf die aktuellsten Daten haben.

### 3 Zielsetzung

Zuerst werden die Ursprungsziele der Masterarbeit aufgezeigt. Im zweiten Teil werden die Ziele nach der ersten Analyse revidiert, da bei der Modellierung der SoftwareVersionsDatabase auffiel, dass zusätzlich die Testinformationen abgespeichert werden sollen. Der Grund wurde im Kapitel 2.2.2.4 näher beschrieben.

#### 3.1 Ziele zum Startzeitpunkt



Abbildung 5: Ursprüngliches Konzept Masterarbeit

Die Zielsetzung nach Stand der Disposition besteht darin die derzeitige SoftwareVersionsDatabase Datenbank so zu erweitern, dass die Testresultate vom TTIC2 abgespeichert werden. So soll auch der Grundzustand, welcher vor der Ausführung der Testkollektion definiert wird, hinterlegt werden.

Die TTIC2 Oberfläche soll so angepasst werden, dass der Benutzer vor dem Start einer Testkollektion ein Grundzustand definieren muss. Dieser kann nur aus bereits hinterlegten Informationen in der SoftwareVersionsDatabase erstellt werden. Diese Einträge werden in der Firmware Oberfläche erstellt.

Das Hauptaugenmerk der Arbeit liegt in der Erstellung des ETIC2. Die Applikation gibt die generierten Testresultate vom TTIC2 wieder. Die Oberfläche ist in drei Ebenen gegliedert. Die oberste Ebene unterteilt die Einträge nach den unterschiedlichen Grundzuständen. Unter dieser Ebene folgt die Testkollektion. Hier wird die Anzahl fehlerhaften Tests angezeigt. In der untersten Ebene werden alle Tests mit ihren allfälligen Fehlermeldungen zu der jeweiligen Testkollektion aufgelistet. Über alle Ebenen ist eine Suche nach bestimmten Suchwörtern möglich.

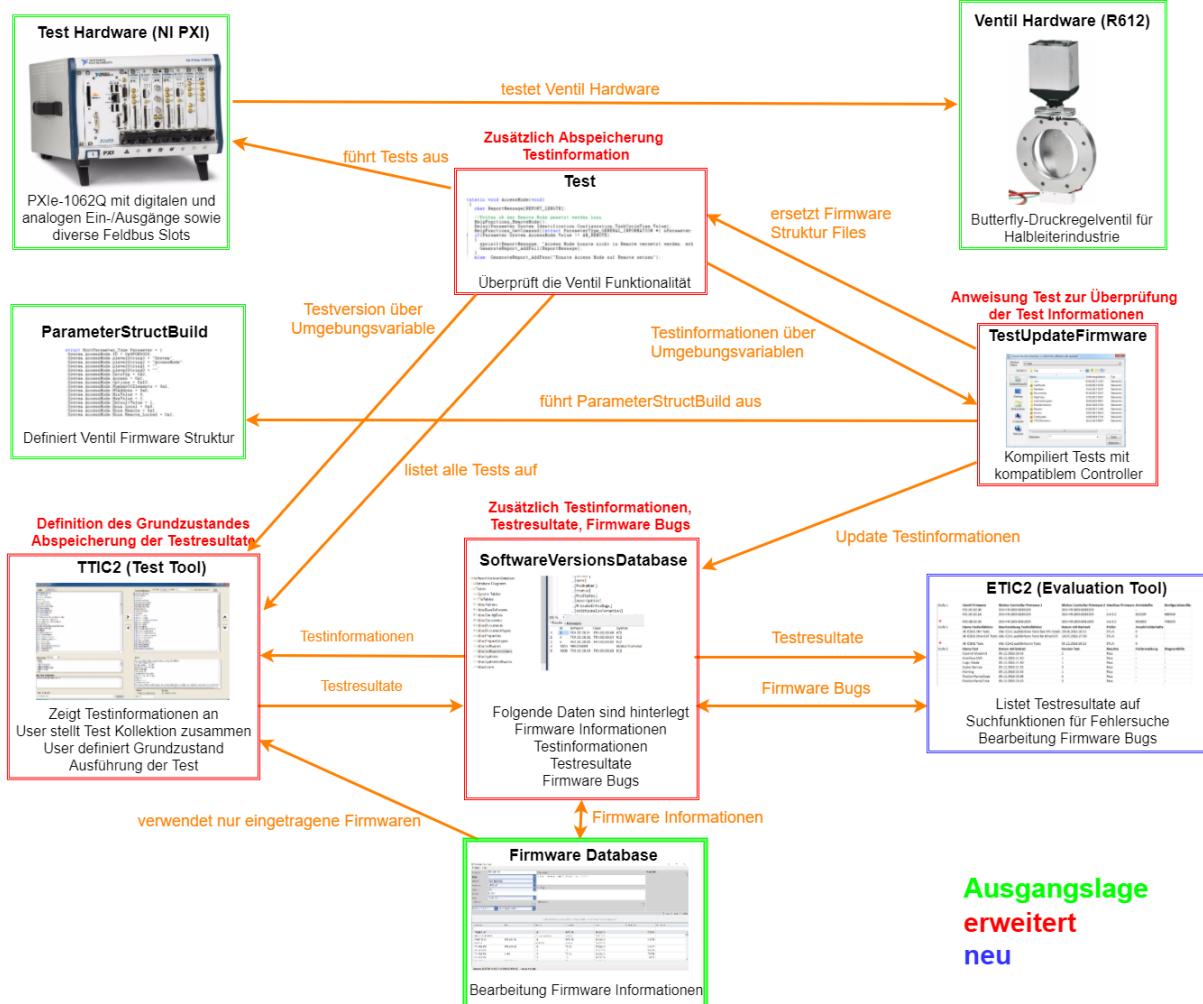
Durch die Angabe eines Grundzustandes, welche aus einzelnen Elementen besteht, die in der SoftwareVersionsDatabase hinterlegt sind, werden alle Testresultate in einem Report aufgelistet. Speziell dabei ist, dass nicht alle Elemente des Grundzustandes definiert werden müssen. Somit kann zum Beispiel geschaut werden, ob eine Firmware Version mit mehreren Motoren Firmwares geprüft wurde und somit mehrere Versionen kompatibel sind.

In der ersten Phase der Arbeit dem Erweitern der SoftwareVersionsDatabase fiel auf, dass es Sinn macht nicht nur die Testergebnisse abzuspeichern sondern auch die Testinformationen. Der Grundgedanke dahinter ist, dass die Testinformationen somit immer aktuell sind. Ausserdem wurde eine Erweiterung der Datenbank mit den Firmware Bugs Informationen als sinnvoll erachtet. Da die Firmware Bugs aus den Testresultaten ersichtlich werden.

Das Resultat nach der ersten Analyse machte klar, dass das primäre Ziel eine lauffähige Testumgebung als Resultat der Arbeit sein muss und nicht eine abgegrenzte Arbeit in welchem die Testresultate angezeigt werden.

## 3.2 Ziele nach der ersten Analyse

Aus der Abbildung 6 geht hervor, welche Elemente in der Testumgebung gleichbleiben und welche angepasst oder neu erstellt werden. Der nächste Abschnitt erklärt die Ziele der einzelnen Anpassungen.



**Abbildung 6: Konzept Masterarbeit Testumgebung komplett**

Die Testinformationen bezüglich des Tests sowie die Anforderungen an die Ventil Hardware sollen neu in der SoftwareVersionsDatabase hinterlegt werden.

Die TestUpdateFirmware Applikation soll nur noch die Tests kompilieren, die mit dem Controller der Ventil Hardware lauffähig sind.

Die Testinformationen bezüglich des Tests und der Ventil Hardware soll neu im TTIC2 aus der SoftwareVersionsDatabase ausgelesen werden. Neu soll der Grundzustand vor der Ausführung der Testkollektion definiert werden und zwar nur aus eingetragenen Werten in der SoftwareVersionsDatabase. Sind diese Werte noch nicht auf der Ventil Hardware eingestellt werden diese aktualisiert, bevor die Tests ausgeführt werden. Nach der Ausführung der Test Kollektion werden die Resultate in der SoftwareVersionsDatabase hinterlegt.

Die bereits existierende SoftwareVersionsDatabase soll so erweitert werden, dass zusätzlich noch Werte bezüglich Testinformationen, Testresultate sowie Firmware Bugs hinterlegt werden können.

Die neue ETIC2 Oberfläche soll eine schnelle Auswertung der Testresultate ermöglichen. Zusätzlich soll ein Report zu einem bestimmten Grundzustand mit allen Testresultaten per Knopfdruck erzeugt werden können.

### **3.3 Quantitative Ziele**

- Jede einzelne ausgeführte Testkollektion muss im ETIC2 zu einem Grundzustand zugeordnet werden.
- Der Grundzustand kann nur mit bereits vorhandenen Einträgen in der SoftwareVersionsDatabase definiert werden.
- Die SoftwareVersionDatabase muss Schreibanfragen von vier Benutzern bearbeiten können.

### **3.4 Qualitative Ziele**

- Das ETIC2 soll sich durch seinen einfachen und stabilen Aufbau, verbunden mit der raschen Auswertung, ob ein Fehler in der ausgeführten Testkollektion aufgetreten ist, auszeichnen.
- Eine ausgeprägte Suchfunktion soll ein Bestandteil des ETIC2 sein, welche eine schnelle Suche nach Fehlermeldungen erlaubt.
- Mit dem ETIC2 soll das Resultat der ausgeführten Testkollektion unmittelbar und einfach ersichtlich sein.
- Auf Knopfdruck sollen die Testresultate exportiert werden können.
- (optional) Das TTIC2 soll für die Anzeige der Testinformationen immer auf die aktuellsten im Test definierten Beschreibungen zugreifen. Auch die Anforderungen an die Ventil Hardware, welche in den Tests definiert werden, sollen auf dem aktuellsten Stand im TTIC2 sein. (Bei einem Release Prozess)
- (optional) Das TTIC2 muss vor Ausführung der Test Kollektion automatisch die definierten Firmwares und Parameterfiles (Drive- und Configuration File) auf die Ventil Hardware updaten.

### **3.5 Aufgabenbegrenzung**

- Die Ventil Hardware ist als Verständnishilfe im Konzept zu finden und nicht als Teil der Arbeit
- Die Test Hardware wird verwendet um die Ventil Hardware anzusprechen. Sie ist aber nicht Teil dieser Arbeit.
- Die Hauptaufgabe des ParameterStructBuild Programms liegt in der Auslesung der Parameter Informationen der Firmware, diese entspricht den Anforderungen und wird in dieser Arbeit nicht angepasst.
- Anpassungen an den einzelnen Tests gehören nicht zur Arbeit
  - Ausnahme: Testinformationen in SoftwareVersionsDatabase zu hinterlegen
- Die Weiterentwicklung der TTIC2 Applikation ist nicht Teil der Masterarbeit. Ausnahmen sind:
  - Auslesung der Testinformationen aus der SoftwareVersionsDatabase
  - Definition des Grundzustandes vor der Ausführung der Test Kollektion
  - Abspeicherung der Testresultate in der SoftwareVersionsDatabase
- Die SoftwareVersionDatabase wird erweitert aber die bestehenden Attribute und Inhalte werden nicht bearbeitet.
- Die Firmware Database ist bereits im Einsatz um die Firmware Informationen einzusehen, das ist aktuell ausreichend.
- ETIC2 wird für den internen VAT Verwendungszweck entwickelt und nicht für den kommerziellen Gebrauch konzipiert.

## 4 Methodik

Das Kapitel Methodik umfasst zwei Teile, einerseits die Vorgehensweise in der Arbeit. Diese inkludiert das angewendete Konzept der Software Entwicklung sowie den chronologischen Ablauf der Arbeitsschritte. Andererseits werden die verwendeten Hilfsmittel erläutert.

### 4.1 Vorgehensweise

In der Software Entwicklung gibt es hauptsächlich zwei Vorgehensweisen nämlich eine agile Methode oder das Wasserfall-Modell. Die agile Variante zeichnet sich durch ihre variable Änderung der Anforderungen aus, wohingegen sich die Wasserfallmethode durch ihre festen Anforderungen und klaren Phasenabgrenzungen auszeichnet. (Pavlov, 2014)

Mit der Erstellung von automatischen Ventiltests sowie der dazu benötigten Ablaufsteuerung durch das TTIC2 befasse ich mich bereits seit drei Jahren. In der Anfangszeit wurde die Ablaufsteuerung parallel mit immer neuen Tests entwickelt. Im letzten Jahr wurde der Hauptfokus auf die Erweiterung der Testabdeckung gelegt. In dieser Zeitspanne wurde das TTIC2 bereits produktiv eingesetzt und die Schwachstellen wurden von meiner Seite evaluiert, da es keine weiteren beteiligten Personen gibt. Daraus resultierten genaue Anforderungen an eine Erweiterung des Systems. Daher kann davon ausgegangen werden, dass sich die Anforderungen während der Arbeit nicht verändern. Weiter sind auch die verwendeten Technologien und Hilfsmittel vor Beginn der Arbeit definiert worden (Kapitel 4.2) und bleiben unverändert. Die Arbeit wird nur durch mich ausgeführt, somit muss das Knowhow nicht ausgetauscht werden.

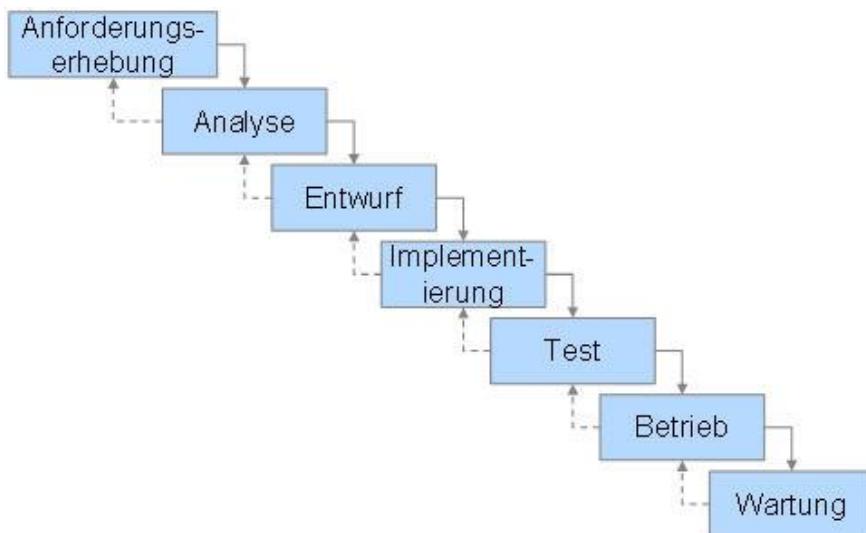


Abbildung 7: Wasserfallmodell (QMETHODS, 2017)

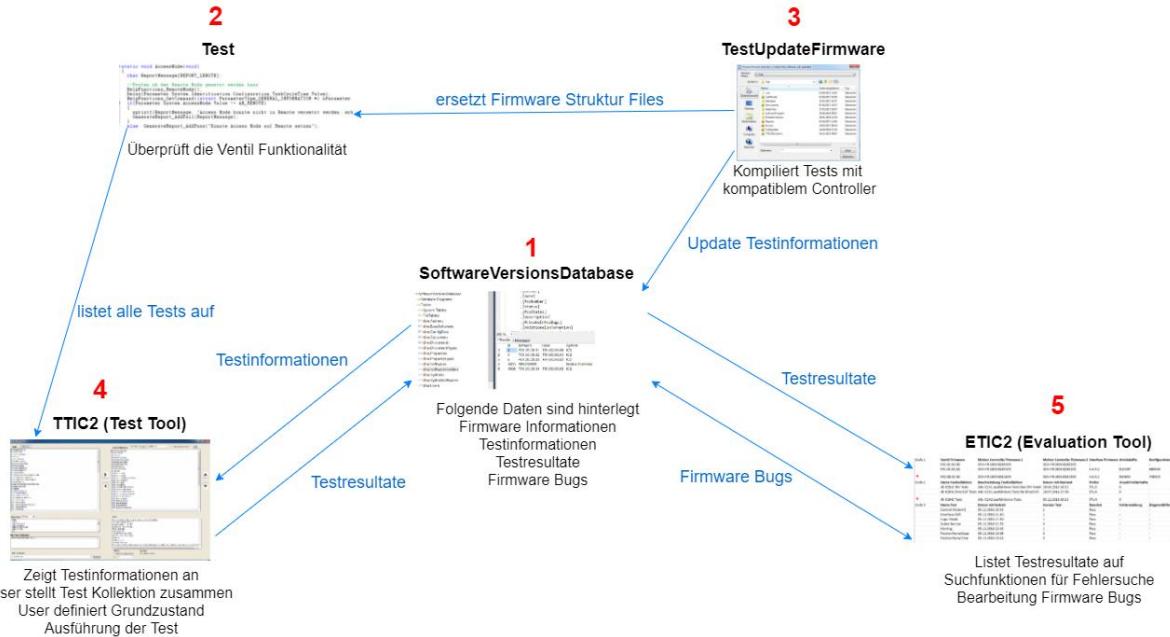
Aus diesen Gründen wird die Arbeit prinzipiell mit dem Wasserfall Modell gelöst. Die Abbildung 7 zeigt die einzelnen Phasen des Wasserfallmodells. Um das methodische Vorgehen in der Arbeit darzustellen, wird in der Tabelle 1 der Ort der einzelnen Phasen des Wasserfallmodells aufgezeigt.

Wasserfall Phase	Kapitel	Name des Kapitels	Inhalt
Anforderungserhebung	2.2	Testumgebung	Auflistung der Schwachstellen
Analyse	3.2	Ziele nach der ersten Analyse	Ziele aus den Anforderungen erstellt
Entwurf	5	Auswertung der Testresultate (erstes Unterkapitel)	Grobes Konzept
Implementierung	5	Auswertung der Testresultate (zweites Unterkapitel)	Beschreibung der Umsetzung
Test	6	Ergebnisse	Erreichung der Ziele

Tabelle 1: Phasen des Wasserfallmodells in der Dokumentation

Die Phase des Betriebs wird erst nach Beendigung dieser Arbeit in Kraft treten.

Der nächste Abschnitt beschreibt den chronologischen Ablauf der Arbeit.



**Abbildung 8: Vorgehensweise Testumgebung**

Aus der Abbildung 8 ist die Reihenfolge der einzelnen Testumgebungselemente zu sehen, welche in der Arbeit behandelt werden. Als erstes wird die bereits existierende SoftwareVersionsDatabase erweitert, um Testinformationen, Testresultate sowie Firmware Bugs Informationen abzuspeichern. Als nächstes werden die Tests so angepasst, dass die Testinformationen in der Datenbank abgelegt werden. Das TestUpdateFirmware Programm wird so angepasst, dass nur noch Tests kompiliert werden, bei denen der angeschlossene Controller Typ übereinstimmt. Anschliessend wird das TTIC2 so erweitert, dass es erstens die Testinformationen und Hardware Anforderungen der Tests von der Datenbank bezieht und zweitens die Testresultate hinterlegt. Zudem soll das TTIC2 dem Benutzer die Möglichkeit bieten, den Grundzustand zu definieren. Als letztes werden die Testresultate im ETIC2 in einfacher Form wiedergegeben.

Zwischen den Phasen Entwurf bis hin zum Test wird das Wasserfallmodell nicht streng befolgt. Es wird jeweils ein Testumgebungselement genommen und es durchläuft den Entwurf, die Implementierungsphase sowie die Testphase. Erst nach erfolgreichem Testdurchlauf wird mit dem nächsten Element der Testumgebung fortgefahrene. Tritt bei einem späteren Testumgebungselement ein Problem auf, so kann das Konzept sowie die Implementierung des zuvor entwickelten Testumgebungselements nochmals überarbeitet werden.

## 4.2 Hilfsmittel

Hier werden die Hilfsmittel der einzelnen Testumgebungselemente näher beschrieben und ihre Auswahl begründet.

### 4.2.1 SoftwareVersionsDatabase

Die Firma VAT bietet einerseits das Microsoft Access Programm an, um Daten oder auch den Zugang zu MS-SQL Servern abzuspeichern. Im Folgenden wird eine Nutzwertanalyse durchgeführt, um die geeignete Datenbanklösung zu bestimmen (Tabelle 2).

Kriterium	Gewichtung	Datenbanklösung			
		Microsoft Access		MS-SQL Server	
		Punkte	Gewichtet	Punkte	Gewichtet
Preis	5	10	50	5	25
Bereits im Einsatz	15	0	0	10	150
Bedienoberfläche	10	10	100	5	50
Fehleranfälligkeit	10	5	50	8	80
Mehrbenutzerverwaltung	5	3	15	6	30
Mehrbenutzersynchronisation	5	3	15	5	25
<b>Total</b>	100		<b>230</b>		<b>360</b>

**Tabelle 2: Nutzwertanalyse Datenbanklösung**

Der Ausschlag ist auf die Lösung MS-SQL Server gefallen, da bereits eine SoftwareVersionsDatabase im Einsatz ist, welche in einer früheren eigenen Projektarbeit erstellt wurde. Die MS Access Lösung ist nur für wenige Benutzer geeignet. Dies fällt im Bereich der Benutzerverwaltung und Synchronisation negativ ins Gewicht. Sie sind aber für die Firmware Database Applikation nötig, deshalb fällt diese Lösung sicherlich weg.

Für die Modellierung der Datenbanktabellen wird mit MySQL Workbench gearbeitet, da hier bereits Kenntnisse aus dem Studium vorhanden sind. Das Programm erlaubt es grafisch die Struktur der Datenbank wiederzugeben. Ist diese Struktur vollständig erstellt, so werden Skripts geschrieben, welche im SQL Server Management Studio ausgeführt werden, um die einzelnen Tabellen zu erstellen. Weiter bietet dieses Programm die Möglichkeit, die einzelnen Tabellen Werte auszulesen und zu manipulieren.

#### 4.2.1.1 Zugriff auf SoftwareVersionsDatabase

Der Zugriff auf die SoftwareVersionsDatabase ist nötig, um Anforderungen bezüglich Testinformationen in der Datenbank abzulegen und um sie später im TTIC2 verwenden zu können. Wie auch um die Testresultate abzuspeichern, welche das TTIC2 produziert, um diese später im ETIC2 anzuzeigen.

Das CVI bietet ein SQL Toolkit an, welches einen schnellen und einfachen Zugang zu SQL Servern erlaubt. Im Gegensatz bietet die .NET Umgebung mit dem Entity Framework die Möglichkeit den SQL Server abzufragen und erstellt ein passendes Modell der Datenbankstruktur.

### Fall 1: Testinformationen in SoftwareVersionsDatabase hinterlegen und auslesen

- Lösungsvorschlag 1: Programmierumgebung CVI mit SQL Toolkit
- Lösungsvorschlag 2: C Programm in Visual Studio mit Entity Framework
- Lösungsvorschlag 3: Programmierumgebung CVI mit DLL Programm für Zugang zur Datenbank (DLL in Visual Studio mit Entity Framework)

Auch in diesem Fall wird eine Nutzwertanalyse durchgeführt, um eine Entscheidung zu treffen:

Kriterium	Gewichtung	Testinformationen in SoftwareVersionsDatabase					
		Lösung 1		Lösung 2		Lösung 3	
		Punkte	Gewichtet	Punkte	Gewichtet	Punkte	Gewichtet
Preis	10	3	30	8	80	5	50
Unterstützung NI Hardware	40	10	400	0	0	10	400
Direkter Zugriff	25	8	200	8	200	0	0
Entwicklungszeit	10	3	30	6	60	5	50
Wartung	20	8	160	7	140	2	40
<b>Total</b>	100		<b>820</b>		<b>480</b>		<b>540</b>

**Tabelle 3: Nutzwertanalyse Zugriff auf SoftwareVersionsDatabase**

Nach Auswertung der Nutzwertanalyse (Tabelle 3) zeigt sich ein anderer Lösungsansatz als der ursprüngliche Lösungsvorschlag 3 aus der Disposition. Da dieser keinen direkten Zugriff auf die Datenbank ermöglicht, was in einem höheren Wartungsaufwand resultiert. Die Gewichtung kam zustande, da aus Sicht des Software Tester der Hauptfokus auf der Testabdeckung der Hardware des Ventils liegt und nicht auf den Anschaffungskosten.

### Fall 2: Testresultate in SoftwareVersionsDatabase hinterlegen

Gleiche Ausgangslage wie in Fall 1, was in der gleichen Lösung resultiert.

### Fall 3: Testresultate in SoftwareVersionsDatabase auslesen

Die Testresultate werden in der ETIC2 Oberfläche ausgelesen. Im Kapitel 4.2.3 fällt die Entscheidung für die Erstellung des ETIC2 die C# Programmiersprache zu verwenden. Daraus resultieren zwei Alternativen, um die Testresultate auszulesen; einerseits mit Hilfe des Entity Framework andererseits mit Hilfe einer DLL, welche in CVI mit Hilfe des SQL Toolkits erstellt wird.

- Lösungsvorschlag 1: C# mit Entity Framework
- Lösungsvorschlag 2: C# mit DLL Programm für Zugang zur Datenbank (DLL in CVI mit SQL Toolkit)

Nimmt man auch hier die gleichen Beurteilungskriterien wie in der Nutzeranalyse (Tabelle 2) aus Fall 1. Dabei fällt allerdings das Kriterium Unterstützung NI Hardware weg, da diese nicht angesprochen werden muss. So schliesst der Lösungsvorschlag 1 in allen Kriterien (Preis, Direkter Zugriff, Entwicklungszeit, Wartung) am besten ab und wird nun für die Implementation verwendet.

### **4.2.2 Test, TestUpdateFirmware, TTIC2**

Wie schon im Kapitel 2.2.3.1 erwähnt werden die Tests in der CVI Programmierumgebung erstellt. Die Gründe liegen einerseits in den persönlichen Programmiererfahrungen, im textbasierten Umfeld sowie der besseren Übersicht in einem grösseren Projekt.

Aus denselben Gründen wurde für das TTIC2 auch CVI als Programmierumgebung ausgewählt. Da das TestUpdateFirmware Programm auch mit den einzelnen Tests zu tun hat, fiel auch hier der Entscheid mit CVI zu entwickeln.

### **4.2.3 ETIC2**

Bei der Wahl der Programmiersprache für die ETIC2 Oberfläche, welche eine klassische Desktopanwendung darstellt, weil die angezeigten Informationen firmenintern bleiben und somit keinen Webzugang benötigt, fiel die Auswahl auf Java oder C#. Dies nicht nur aus persönlichen Interesse an den Sprachen, sondern auch da beide im Studium näher vorgestellt wurden.

Weitere Gründe für die beiden Programmiersprachen sind (Gebhart, 2013):

- Grosse Auswahl an Hilfsfunktionen (Anteil bei 70% einer Applikation)
  - Reduziert den notwendigen Programmieraufwand auf weniger als 40%
  - Minimiert das Risiko für fehlerhafte Implementierung
  - Robuster, da nicht direkt auf Speicherzellen zugegriffen werden kann
  - Komplette Aufrufhierarchie bei Absturz

Die Entscheidung fiel letztendlich auf C#, was mit folgenden Argumenten begründet wird:

- Firmware Database ist bereits mit C# implementiert worden
- DevExpress wurde schon erworben (grosse Anzahl an Bedienelementen und Bibliotheken)
- Einfacher zu handhabende Programmierumgebung (Visual Studio)

## 5 Auswertung der Testresultate

Das folgende Kapitel ist das Herz der Arbeit. Es dokumentiert die Umsetzung der Arbeit. Die Kapitel sind chronologisch zu den Testumgebungselementen geordnet (Abbildung 8). Wegen der Wasserfallmethode wurde jedes Kapitel in Entwurf und Implementierung unterteilt.

### 5.1 SoftwareVersionsDatabase

#### 5.1.1 Modellierung Firmware Informationen

Hier wird die Modellierung der Firmware Informationen beschrieben, welche in einem früheren Projekt erstellt wurden, um Informationen bezüglich einzelner Firmwares abzuspeichern.

##### 5.1.1.1 Entwurf

Es werden verschiedene Typen von Firmwares in der gleichen Tabelle abgelegt. In einer weiteren Tabelle wird definiert, welche Firmwares miteinander kompatibel sind.

Wichtigste Informationen bezüglich einer Firmware, die hinterlegt werden, sind:

- Name
- Basis
- System
- Customer
- Autor
- Erstellungsdatum
- PSS Nummer (interne Produkt Nummer)
- Beschreibung
- Kompatible Firmwaren

In der Datenbank werden die Ventilfirmwaren sowie die Motion Controller Firmwaren wie auch Feldbus Firmwaren hinterlegt. Diese Unterscheidung wird im Feld System erkennbar.

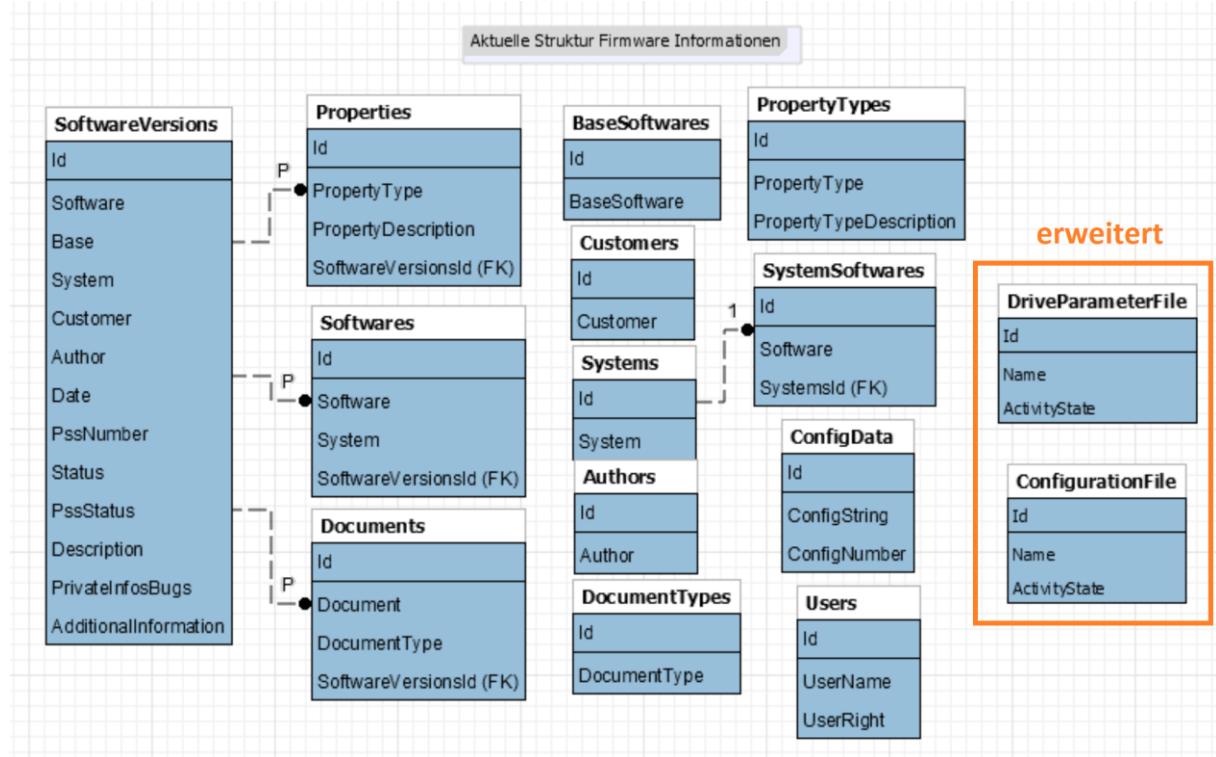
Die PSS Nummer ermöglicht den Zugang zum internen ERP System.

Unter kompatible Firmwaren werden die Motion Controller Firmwaren und Feldbus Firmwaren notiert, welche mit der Ventilsoftware lauffähig sind. Das heisst es gibt nur Einträge, wenn es sich beim aktuellen Firmware Eintrag um eine Ventilfirmware handelt.

### 5.1.1.2 Implementierung

#### Aktuelle Version

In der Abbildung 9 ist die aktuelle Version des SoftwareVersionsDatabase Modells, welches heute im Einsatz ist, ersichtlich.



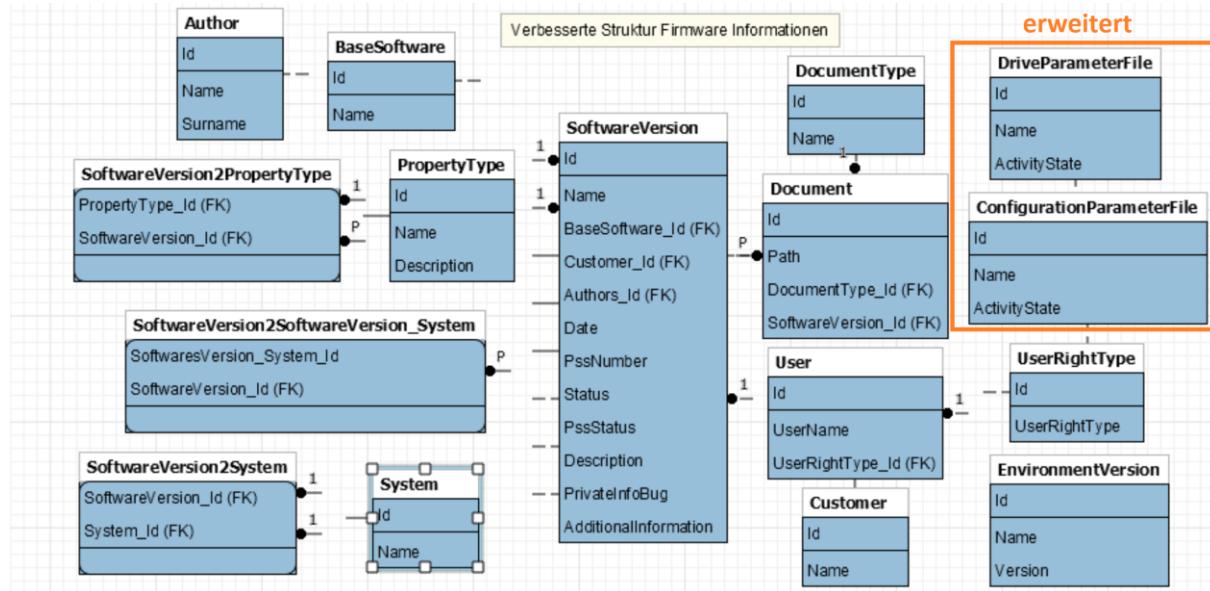
**Abbildung 9 Aktuelle SoftwareVersionsDatabase Struktur**

Im Modell wurden noch die DriveParameterFiles und ConfigurationFiles eingezeichnet, da diese näher an den Informationen der Firmware liegen als bei den Testresultaten. Diese beiden Tabellen werden für den Grundzustand des Ventils vor der Ausführung eines Tests gebraucht.

Bei genauerem Betrachten der Attribute der einzelnen Tabellen fällt auf, dass diese mehrfach vorkommen. Wie zum Beispiel beim Eintrag des Customer zu sehen ist. Es gibt einen Eintrag Customer in Customers, welche die einzelnen Customers beinhaltet wie auch in der Haupttabelle SoftwareVersions. Das widerspricht der Normalisierungsregel. Die Idee der Customers Tabelle liegt darin, dass nur Einträge dieser Tabelle in der Haupttabelle eingetragen werden können. So sollen zum Beispiel Schreibfehlern oder eine unterschiedliche Reihenfolge des Namens und Vornamens vorgebeugt werden. Wird ein Fehler eruiert, werden bei der Korrektur alle Einträge in der Haupttabelle mitgeändert. Das Konzept wurde hier von der Applikation übernommen, da nur Einträge für den User angezeigt werden, welche in der Customers Tabelle hinterlegt sind.

## Verbesserte Version

Infolgedessen wurde das SoftwareVersionsDatabase Modell überarbeitet, diese Überarbeitung ist in der Abbildung 10 ersichtlich, jedoch wird aktuell noch nicht damit gearbeitet.



**Abbildung 10:** Überarbeitete SoftwareVersionsDatabase Struktur

In diesem Modell stellt die Datenbank sicher, dass der Customer zuerst in der Customers Tabelle eingetragen werden muss und nur seine Referenz in der Haupttabelle eingetragen wird.

Das Modell unterscheidet sich markant in Anbetracht der Namensgebung der einzelnen Attribute. Der Tabellenname kommt jetzt nicht mehr in den einzelnen Attributen zum Tragen. Zudem ist die Authors Tabelle in Bezug auf den Namen aufgesplittet worden, was bei einer späteren Auswertung nützlich sein kann um beispielsweise nach dem Nachnamen sortieren zu können.

Um dieses Modell auszuführen, muss die Oberfläche des Firmware Verwaltungstools angepasst werden. Dies kann aus Zeitgründen nicht weiterverfolgt werden. Somit wird in dieser Arbeit noch mit dem ursprünglichen Modell weitergearbeitet. Im Kapitel 8.1.1 ist beschrieben, wie der Umbau auf das neue Modell realisiert werden soll.

### 5.1.2 Modellierung Testinformationen

Nachfolgend wird die Erweiterung der SoftwareVersionsDatabase beschrieben. Damit die Testinformationen hinterlegt werden können.

#### 5.1.2.1 Entwurf

Bei den Testinformationen handelt es sich einerseits um Informationen, die den Test näher definieren. Diese sind:

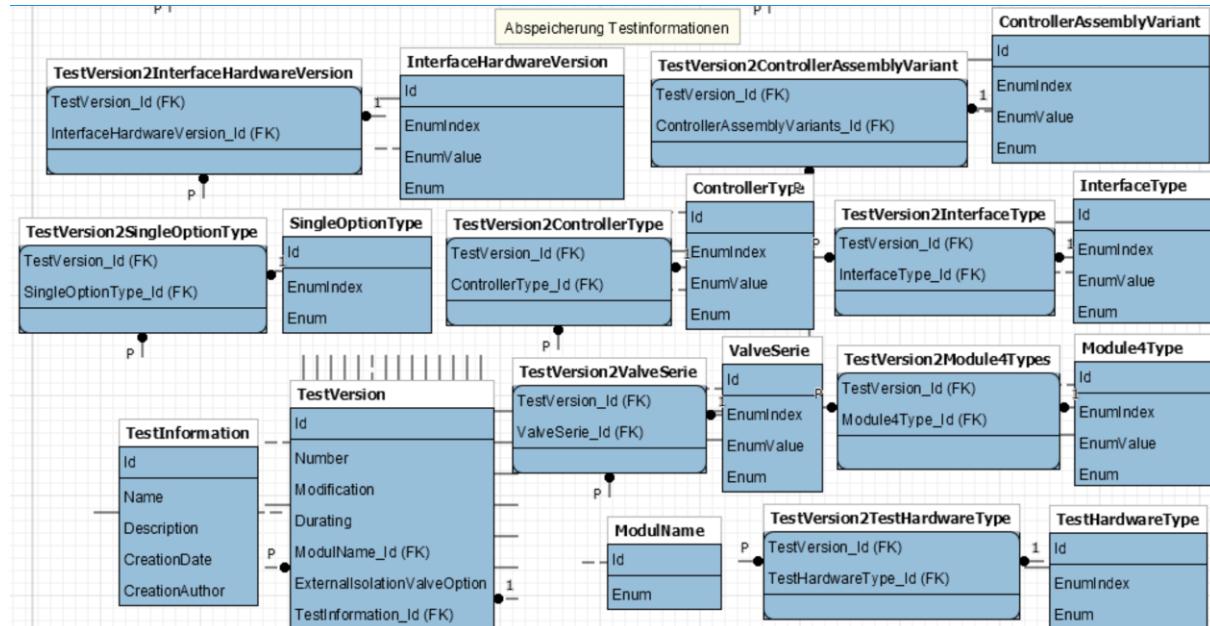
- Name
- Beschreibung
- Erstellungsdatum
- Autor

Andererseits werden Anforderungen des Tests an die Ventil Hardware sowie an die Test Hardware definiert. Die wichtigsten Einträge sind dabei:

- Modul
- Ventilreihe
- Controller
- Interface
- Option
- Test Hardware

### 5.1.2.2 Implementierung

Die Umsetzung der Modellierung bezüglich Testinformationen ist in Abbildung 11 ersichtlich.



**Abbildung 11: Erweiterung SoftwareVersionsDatabase zur Speicherung der Testinformationen**

Die Haupttabelle ist die TestInformation. In dieser gibt es genau einen Eintrag für jeden Test. Hier sind zudem die generellen Daten des Tests hinterlegt. Erweitert wird das Modell mit den einzelnen Testversionen. So können später nur durch Angabe der Test Version alle Anforderungen bezüglich der Ventil Hardware aus der Datenbank herausgelesen werden. Denn alle Ventil Anforderungen werden mit der Test Version verheiratet.

Ebenfalls gibt es zu jeder Ventil Hardware Anforderung eine Hilfstabelle, welche die einzelnen Werte definiert. Existiert ein Parameter in der Software mit dieser Information, so wird ein EnumValue Eintrag in der Tabelle eingefügt. Zu diesem Wert wird immer eine Beschreibung dieses Wertes in Textform hinterlegt, dies ist im Feld Enum hinterlegt. Ausser bei der Modul Information kann es sein, dass mehrere Einträge einer Hardware Anforderung definiert werden sollen. Zum Beispiel kann ein Test für mehrere Controller Typen ausführbar sein aber nicht für alle. Das soll mit einem Beispiel verdeutlicht werden:

Der Controller Typ 1 mit Index 1 und Enum ‘IC2H1’ ist sehr ähnlich aufgebaut wie der Controller Typ 3 mit Index 3 und Enum ‘IC2H3’. Mit dem Bitwert  $2^1 + 2^3 = 10$  können mit Hilfe eines Wertes mehrere Einträge definiert werden.

In der ersten Version der Modellierung wurde ein Wert für die Definition des OptionTypes verwendet. Der Nachteil ist aber das später in der TTIC2 Oberfläche nicht nach einer unterstützten Option gefiltert werden kann, dies wurde mit der SingleOptionType Modellierung gelöst.

### **5.1.3 Modellierung Testresultate**

Der nächste Abschnitt beschreibt die Erweiterung der SoftwareVersionsDatabase mit den Testresultaten. Sie werden vom ETIC2 gebraucht, um die Resultate zu den einzelnen Ventilfirmwaren aufzulisten.

#### **5.1.3.1 Entwurf**

Der Grund der Zusammenlegung von Firmware Informationen und Testresultaten liegt darin, dass nur eingetragene Firmwares zum Testfall zugelassen sind. Unabhängig ob es sich hierbei um eine Ventil-, Motion Controller- oder Feldbus Firmware handelt.

Um die Spezifikation des ETIC2 zu erfüllen, sind folgende zusätzliche Informationen nötig:

- Antriebsfile
- Konfigurationsfile
- Test Kollektion
- Anzahl fehlerhafte Tests
- Ventil Informationen
- Testversion
- Testresultat
- Fehlermeldungen eines Tests

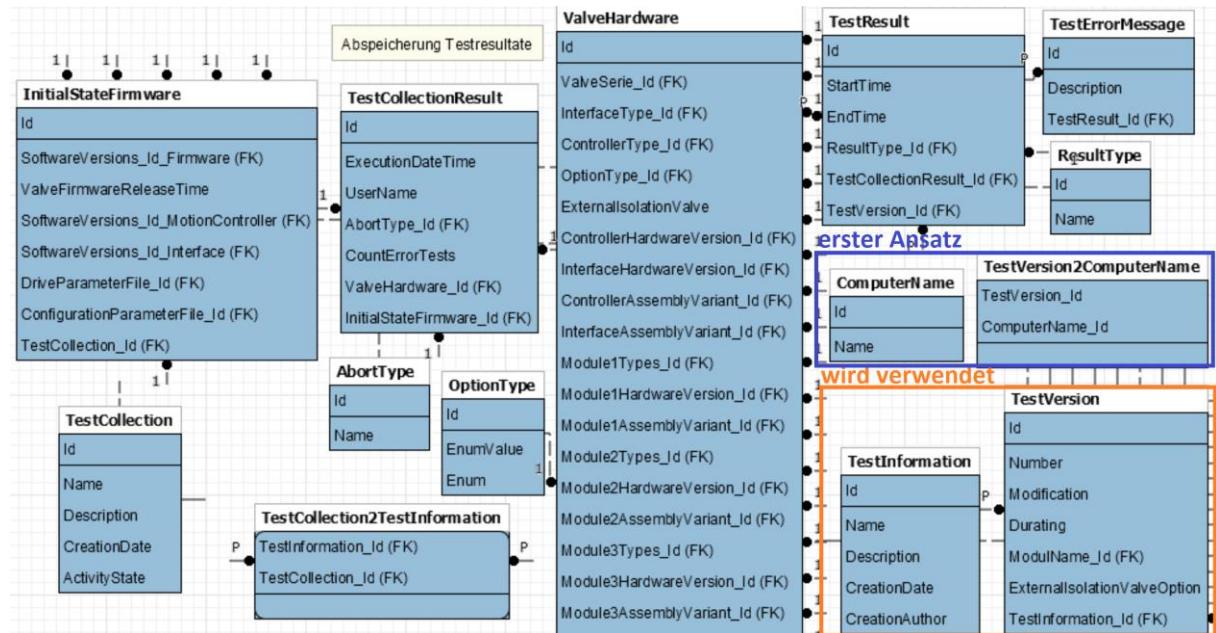
Wichtig ist zudem die Information, welche Tests eine Kollektion beinhaltet. Die Information der Test Version spielt hierbei keine Rolle. Eine Rolle spielt aber die verwendete Ventil Hardware, um die Test Kollektion auszuführen.

Die Test Version ist wichtig für die Information bezüglich Ventilhardware. Diese wird benötigt, um den Test ausführen zu können. Der Test wird versioniert, um Änderungen bezüglich Hardware berücksichtigen zu können, da das TTIC2 jederzeit auch die älteren Test Kollektionen noch ausführen können muss. Die meisten Ventilfirmwaren sind kundenspezifische Entwicklungen, deshalb wird meistens nicht von der aktuellsten Version weitergearbeitet, sondern eine ältere Version erweitert. Diese Anforderung muss auch für die Tests gelten.

Um eine Auswertung bezüglich ähnlichen Testfehlern machen zu können, sind die ersten Fehlermeldungen in der Datenbank hinterlegt.

### 5.1.3.2 Implementierung

In der Abbildung 12 ist die Erweiterung der SoftwareVersionsDatabase zu sehen, um die Testresultate abzuspeichern.



**Abbildung 12: Erweiterung SoftwareVersionsDatabase zur Speicherung der Testresultate**

Dabei übernimmt die InitialStateFirmware Tabelle die Funktion einen Grundzustand vor dem Start der Test Kollektion herzustellen. Die Modellierung erzwingt, dass die Firmwares wie auch die DriveParameter- und ConfigurationFile schon in der Datenbank hinterlegt sein müssen, bevor der Grundzustand definiert werden kann.

Aus softwareentwicklungstechnischen Gründen wird vor der Freigabe einer Firmware mit einer sogenannten trunk Version gearbeitet. Diese bildet immer zum entsprechenden Zeitpunkt die aktuellste Firmware ab. Um diesem Umstand Rechnung zu tragen und die einzelnen trunk Versionen unterscheiden zu können, wurde ein zusätzliches Attribut die ValveFirmwareReleaseTime eingefügt.

In der nächsten tieferen Ebene wird die Test Kollektion sowie die Ventil Hardware, auf welcher die Test Kollektion ausgeführt wurde, hinterlegt. Was der Anwender hier interessiert ist die Anzahl fehlerhafter Tests. Weiter ist auch ersichtlich, ob der User den Testablauf abgebrochen hat. Die Test Kollektion besitzt zusätzlich noch einen ActivityState. Dieser wird eingefügt, um zu verhindern das Test Kollektionen, welche schon einen Eintrag im TestCollectionResult haben, gelöscht werden können. So wird zwar die Test Kollektion nicht mehr angezeigt aber ihr ActivityState wird auf False gesetzt.

Die Modellierung lässt bewusst die Möglichkeit offen, ob die erste Ebene den Grundzustand der Firmware abbildet oder die benutzte Ventil Hardware. Wird in der ersten Ebene die Ventil Hardware beschrieben, so können in der zweiten Ebene die Firmware Informationen aufgelistet werden.

In der untersten Ebene sind alle Tests der Test Kollektion aufgelistet. Hier findet man bei einem Fehler auch die dazugehörigen Fehlermeldungen. Diese ist sehr wichtig, um einen Bug in der Firmware erkennen zu können.

Speziell am Modell ist die TestVersion2ComputerName Tabelle. Diese wird bei jedem Start des TTIC2 Programms neu mit allen Versionen der Tests gefüllt, welche vom TTIC2 geladen werden. Die Anforderung, dass mehrere Applikationen auf unterschiedlichen Computern gleichzeitig gestartet werden können, führt zum Problem, dass gleichzeitig in die gleiche Tabelle geschrieben werden kann. Um diesen Umstand zu vermeiden wurde für den ersten Ansatz ein zusätzliches Feld mit dem Computernamen eingeführt, um so die einzelnen Applikationen unterscheiden zu können. Doch dieses Konzept scheiterte in der Umsetzung, welches im Kapitel 5.3.1 näher beschrieben wird. Das TTIC2 löst diesen Umstand neu so, dass jeder Test mit Hilfe einer Umgebungsvariablen seine aktuelle Version

liefert. Diese wird beim Aufstarten der Applikation in einer Liste hinterlegt. Durch diesen Lösungsweg wird die TestVersion2ComputerName und ComputerNames Tabellen nicht mehr gebraucht.

Die aktuelle Testversion wird benötigt, um die entsprechenden Hardware Anforderungen der Tests auslesen zu können und zu entscheiden, ob der Test mit der angeschlossenen Hardware ausführbar ist. Über die Lebenszeit der Tests können sich die Anforderungen verändern, da neue Testfunktionen hinzukommen können. Weiter ist sie auch ein wichtiger Bestandteil für die Auswertung.

In der Testphase wurde bewusst, dass der erste Ansatz mit dem ExecutionDate in der TestCollection-Result zum Problem führen kann, wenn die gleiche Test Kollektion am gleichen Tag ausgeführt wird. Deshalb wird jetzt die CreationDateTime verwendet, um auch in diesem Fall das Resultat der Test Kollektion auseinander halten zu können.

### 5.1.4 Modellierung Firmware Bugs

Der nächste Abschnitt zeigt die Modellierung der Firmware Bugs auf. Diese ist auch in der Software-VersionsDatabase zu finden, da diese aus den Testresultaten resultieren, welche auch im ETIC2 verwaltet werden.

#### 5.1.4.1 Entwurf

Die Umsetzung wird aus Zeitgründen nicht als realistisch für die Arbeit betrachtet. Was aber umgesetzt wird, sind die nötigen Datenbankfelder. Darunter fallen folgende Kriterien:

- Fehlerart
- Status des Fehlers
- Ventil Hardware
- Fehlerbeschreibung
- Wichtigkeit der Fehlerbehebung
- Datum des Fehlerfundes
- Datum der Fehlerbereinigung

#### 5.1.4.2 Implementierung

Die Umsetzung der Firmware Bugs Modellierung ist aus der Abbildung 13 zu entnehmen.

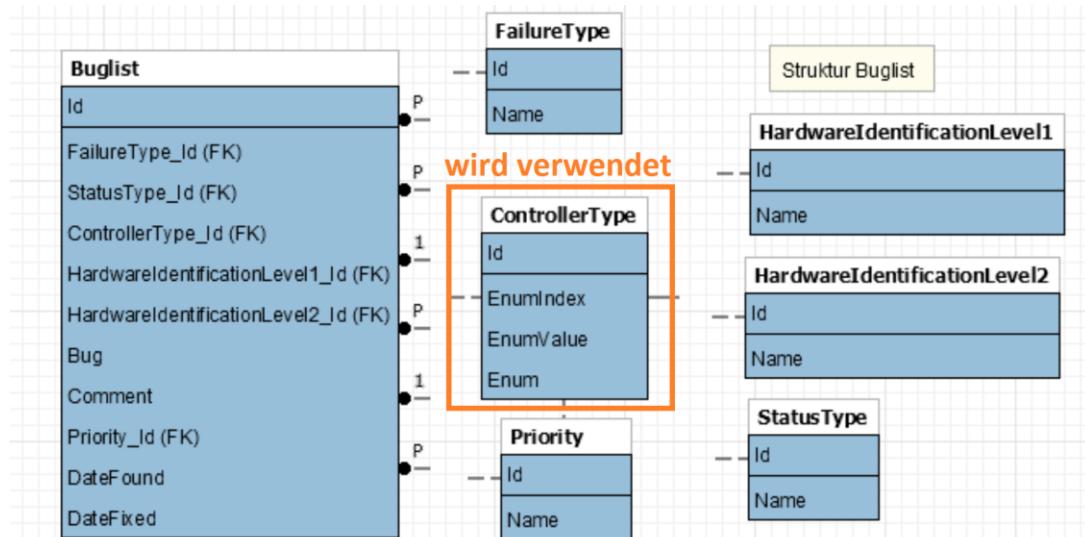


Abbildung 13: Erweiterung SoftwareVersionsDatabase zur Speicherung der Firmware Bugs

Über den FailureType Eintrag kann herausgelesen werden, ob der Fehler reproduzierbar ist oder nur sporadisch auftritt. Der StatusType zeigt an, ob der Fehler schon behoben, noch offen ist oder das Fehlverhalten nicht behoben werden kann oder soll. Mit den ControllerType und HardwareIdentificationLevel1 und 2 Felder wird die verwendete Ventil Hardware gekennzeichnet bei welcher der Fehler aufgetreten ist. Neben dem Controller wird in dem Hardware Identifikationslevel 1 zwischen dem verwendeten Interface, der Option oder der Ventilreihe unterschieden. In der zweiten Ebene wird diese Eigenschaft konkretisiert. Im Falle der Ventilreihe werden in der zweiten Ebene Einträge wie FaceSerial, SFV oder R655 eingetragen. Dabei werden die Hardware Identifikationslevels nur benutzt, wenn nachweislich der Fehler nur in dieser Konstellation auftritt und sich nicht um ein generelles Problem handelt. Zusätzlich wird unter Bug das Fehlverhalten beschrieben. Ebenfalls werden im Comment ergänzende Informationen unter welchen Umständen der Fehler ausgelöst wird notiert. Zudem wird kategorisiert, wie schlimm die Folgen des Fehlers in der Praxis sind. Somit kann anschliessend bestimmt werden, welche Fehler zuerst in der Software behoben werden sollen. Dazu werden das Datum des ersten Auftretens des Fehlers sowie das Beheben des Fehlers abgespeichert.

## 5.2 Test

### 5.2.1 Abspeicherung Testinformationen

Nach Komplettierung der SoftwareVersionsDatabase folgt als nächster Schritt die Abspeicherung der Testinformationen in der erstellten Datenbank. Diese erfolgt wie schon im Kapitel 4.2.1.1 erwähnt, mit Hilfe des SQL Toolkits, Details zum Funktionsumfang sowie einem Codebeispiel sind im Anhang unter C.2. SQL Toolkit zu entnehmen.

#### 5.2.1.1 Entwurf

In den Testinformationen sind einerseits die spezifischen Informationen zum jeweiligen Test hinterlegt. Diese werden im TTIC2 gebraucht, um dem Anwender den Zweck des Tests zu erklären. Andererseits sind dort die Anforderung an die Ventil Hardware hinterlegt. Die detaillierten Elemente der Testinformationen sind im Kapitel 5.1.1 ersichtlich.

Die Abbildung 14 zeigt unter anderem die Ausgangslage, wie die Testinformationen früher in einem Textfile abgespeichert, später im TTIC2 wieder ausgelesen und in der Oberfläche angezeigt wurden.

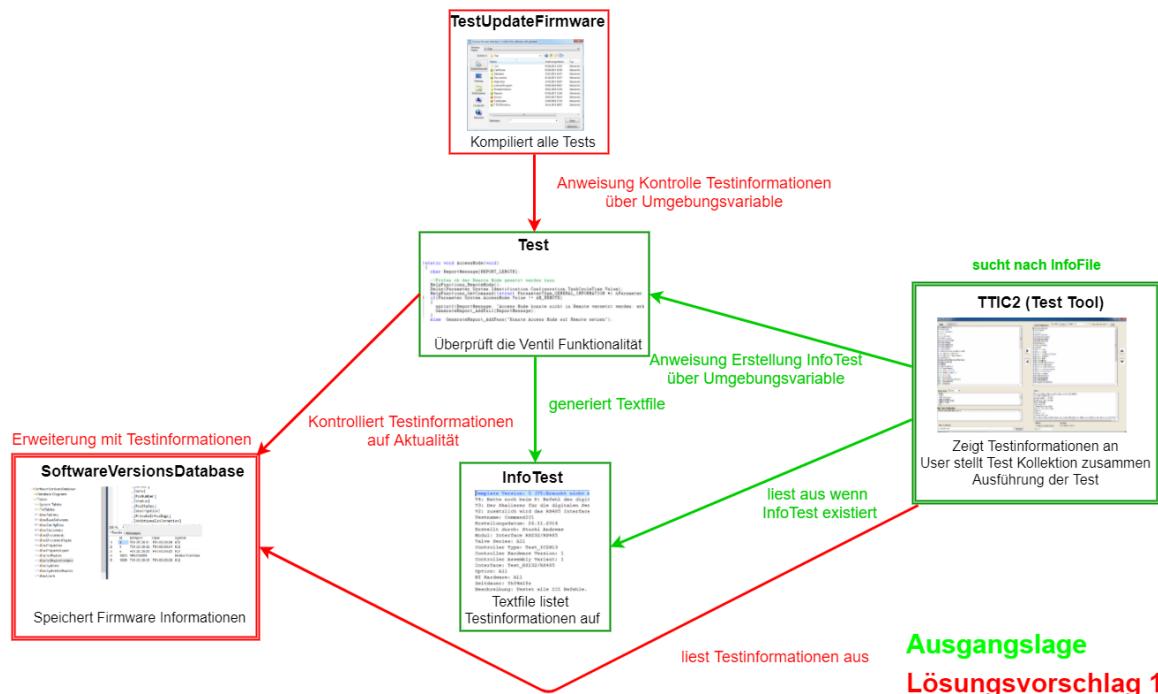


Abbildung 14: Erster Ansatz Abspeicherung Testinformationen

Der Benutzer wählt beim Aufstarten des TTIC2 den Pfad aus, aus welchem die einzelnen Tests aufgelistet werden. Der Grund einer Auswahl liegt darin, dass nicht immer die aktuellste Ventil Firmware getestet werden soll sondern meist verschiedene Firmware Entwicklungen angepasst an den jeweiligen Kunden.

Um die Testinformationen auszulesen, sucht das TTIC2 im Testordner nach dem InfoTest Textfile. Existiert dieses nicht, wird der Test aufgerufen und lässt ihn dieses Textfile erzeugen. Diese Funktion wird über eine Umgebungsvariable initiiert, welche vor Ausführung der Testapplikation gesetzt wird.

Der naheliegende Ansatz für die Abspeicherung der Testinformationen mit Integration der Software-VersionsDatabase ist, dass nicht mehr die InfoTest Textfile erzeugt werden, sondern das TTIC2 die Testinformationen aus der Datenbank ausliest. Um die aktuellen Testinformationen in der Datenbank zu haben, ist es unumgänglich, dass diese auf Aktualität überprüft werden. Da stellt sich die Frage, wann ist das genau notwendig? Die Antwort fällt nicht schwer, nämlich bei jedem neuen Firmware Release. Das TestUpdateFirmware Programm wird schon verwendet, um bei einem neuen Firmware Release die Firmware Struktur upzudaten. Jetzt wird dieses zusätzlich noch damit beauftragt die Testinformationen in allen Tests auf Aktualität zu prüfen (Lösungsvorschlag 1 Abbildung 14).

### 5.2.1.2 Implementierung

Die erste Aufgabe, die Verbindung zur SoftwareVersionsDatabase, erwies sich schwieriger als angenommen. Das Beispiel von Seite des SQL Toolkits dies mit Hilfe eines Data Source Name (DSN) Eintrages zu tun, erwies sich nicht als praktikabel. Da dafür Admin Rechte nötig sind, um den Eintrag neu auf einem Rechner erstellen zu können. Als Alternative kann die Verbindung mit dem SQL Server auch unter Angabe eines Connection Strings aufgebaut werden (Community, 2016).

Im nächsten Schritt werden Funktionen erstellt, um die Testinformationen in die Datenbank zu schreiben, welche im Kapitel 5.1.2.2 in der Modellierung beschrieben sind. Um dem Leser ein besseres Verständnis der Implementierung zu geben, findet sich im Anhang (C.2.2 Code Ausschnitt um einen neuen Eintrag zu erstellen) ein Beispiel, um einen Eintrag in der Datenbank mithilfe des SQL Toolkits zu erstellen.

Im nächsten Abschnitt wird das Prinzip erklärt, wie die Testinformationen und die Hardware Anforderungen des Tests in der Datenbank abgespeichert werden. Zuerst werden die Hardware Anforderungstypen in der Datenbank hinterlegt, welche in einem Strukturfle definiert werden.

Für jeden einzelnen Test wird ein Eintrag erstellt, wenn dieser noch nicht eingetragen ist. Diese besitzen unterschiedliche Versionen, welche in der Versionstabelle eine Referenz auf den jeweiligen Test bekommen. Dabei wird die Hardware Anforderungen an den Test mit einer Verbindungstabelle zwischen der Referenz zur Test Version und der Referenz zum Hardware Anforderungstyp erreicht.

Die Grundidee dieser Umsetzung liegt darin die Testinformationen von den einzelnen Firmwaren zu trennen. So können Rückschlüsse bei jedem Firmware Release gezogen werden, welche Ventil Hardware der Test voraussetzt. Dabei ist es nicht zwingend, dass sich die Testinformationen zwischen den einzelnen Freigaben von Firmwaren verändern.

Als nächstes wurde die Anweisung zur Testinformationsüberprüfung über das TestUpdateFirmware Programm implementiert (siehe Kapitel 5.3.1).

## 5.2.2 Testinformationen an TestUpdateFirmware über Umgebungsvariablen

### 5.2.2.1 Entwurf

Wie im Unterkapitel 5.3.1 erwähnt, wird vermutet, dass das ständige Schliessen und Öffnen der Datenbankverbindung zu unerwarteten Abstürzen führt. Also muss der Lösungsansatz vorsehen, dass die Verbindung nur einmal auf und später abgebaut wird. Um dies zu erreichen, kann der Test selber nicht mehr die Funktion übernehmen, die Daten in der SoftwareVersionsDatabase zu hinterlegen.

Daraus folgt der Lösungsansatz, dass die TestUpdateFirmware die Verbindung zur SoftwareVersionsDatabase herstellt. Die einzelnen Tests liefern über Umgebungsvariablen die Testinformationen. So kann das TestUpdateFirmware Programm die Aktualität der Testinformationen garantieren (Abbildung 15, Lösungsvorschlag 2)

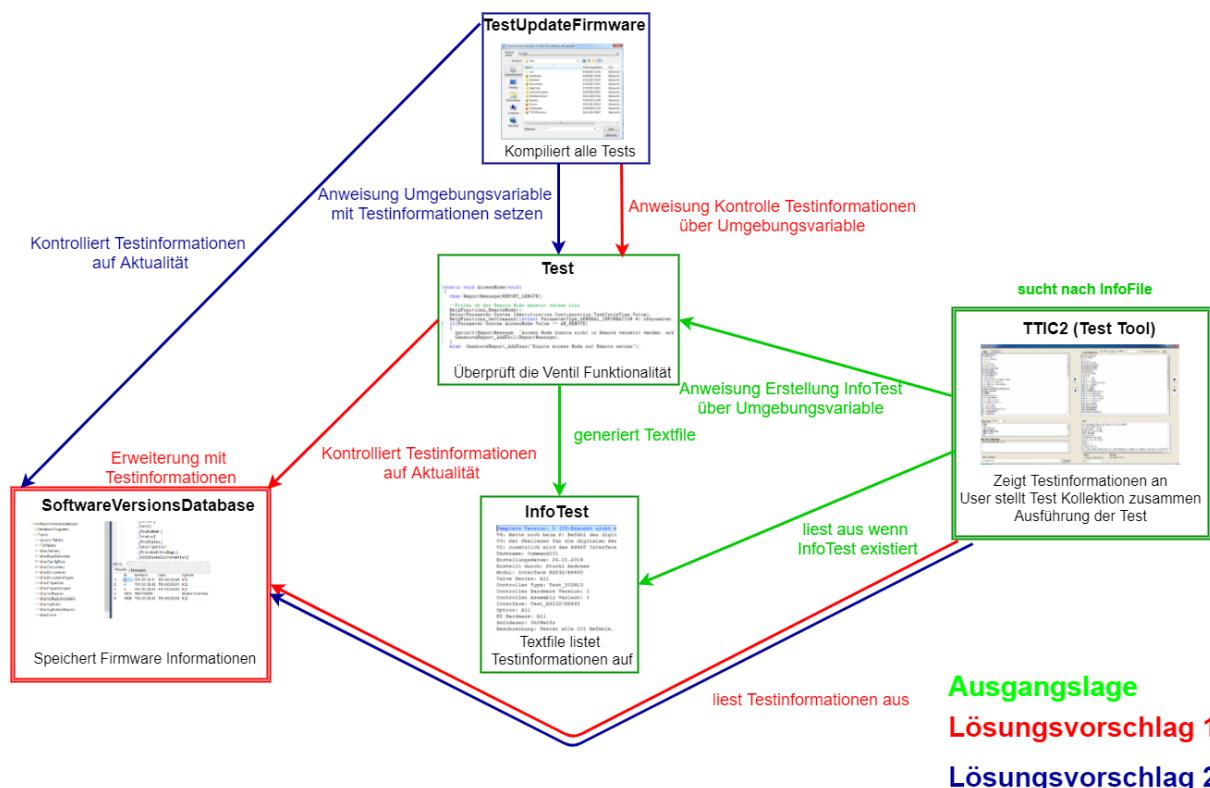


Abbildung 15: Verbessertes Konzept zur Abspeicherung der Testinformationen

### 5.2.2.2 Implementierung

Der Test fragt in der main Funktion die TestOptions Variable ab, welche vom TestUpdateFirmware Programm gesetzt wird (Kapitel 5.3.1), um nur die Testinformationen in die Umgebungsvariablen zu schreiben. Es werden mehrere einzelne Umgebungsvariablen angelegt, um einerseits die Testbeschreibung zu hinterlegen, andererseits werden alle unterstützten Hardware Typen des Tests definiert. Dabei wird auf Umgebungsvariablen von CVI zurückgegriffen, da diese ständig überschrieben werden. Das TestUpdateFirmware aktualisiert anschliessend die Testinformationen in der SoftwareVersionsDatabase (Kapitel 5.3.3), welche vom TTIC2 zur Anzeige in der Oberfläche verwendet wird.

## 5.3 TestUpdateFirmware

### 5.3.1 Anweisung Update Testinformationen

#### 5.3.1.1 Entwurf

Wie im vorherigen Kapitel erwähnt und in Abbildung 14 Lösungsvorschlag 1 ersichtlich, startet das TestUpdateFirmware Programm die einzelnen Tests, sodass die Testinformationen in der Datenbank aktualisiert werden können. Wichtig dabei ist es, dass der Test selbst nicht ausgeführt wird, dies wird mit Hilfe einer Umgebungsvariablen erreicht.

#### 5.3.1.2 Implementierung

Die TestUpdateFirmware setzt eine Umgebungsvariable, welche später im Test dazu genutzt wird, um nur die Testinformationen auf Gültigkeit zu überprüfen. Bei der UmgebungsvARIABLEN handelt es sich um eine von der C Library, damit sie nicht initialisiert werden muss. Genauere Details sind im Anhang (C.1.1.1. Gegenüberstellung zu C Library Umgebungsvariable) zu finden.

In der Testphase dieses Ansatzes tritt nach einer zufälligen Anzahl von erfolgreich ausgeführten Tests ein unerwarteter Absturz auf. Aus dem Absturzprogramm des CVI gehen keine genaueren Informationen hervor. Dies kann auch im Debug Modus nicht reproduziert werden. Meine Vermutung ist, dass beim ständigen Auf- und Abbau der Datenbankverbindung eine Ressource nicht ordnungsgemäß geschlossen werden kann.

Deshalb wird ein neuer Ansatz gewählt, welcher im Kapitel 0 beschrieben wird, da dieser auch Einfluss auf die Testentwicklung hat. Dadurch werden zuerst die Tests und anschliessend das TestUpdateFirmware Programm dementsprechend angepasst.

### 5.3.2 Abfrage Testinformationen

#### 5.3.2.1 Entwurf

Nach dem gescheiterten Lösungsversuch 1 (Abbildung 15) wird nun nach der Ausführung des Tests die UmgebungsvARIABLEN für die Testinformationen ausgelesen und zwischengespeichert. Diese wird für den nächsten Arbeitsschritt benötigt um die Testinformationen auf Aktualität zu überprüfen.

#### 5.3.2.2 Implementierung

Bei den UmgebungsvARIABLEN, welche die Testinformationen des Tests beinhalten, handelt es sich um die von CVI (Kapitel 5.2.2.2). CVI bietet Callback Funktionen an, welche bei Veränderungen der UmgebungsvARIABLEN ausgelöst werden. So wird nach Ausführung des Tests, welche die UmgebungsvARIABLEN mit den Testinformationen beschreibt, der Wert der UmgebungsvARIABLEN ausgelesen und zwischengespeichert. Dieser wird im nächsten Arbeitsschritt mit dem Wert in der SoftwareVersionsDatabase auf Aktualität überprüft.

### 5.3.3 Testinformationen in SoftwareVersionsDatabase schreiben

#### 5.3.3.1 Entwurf

Zuerst werden die Hardware Anforderungstypen auf Aktualität geprüft. Anschliessend werden die ausgelesenen Testinformationen und die UmgebungsvARIABLEN Werte auf Aktualität in der SoftwareVersionsDatabase geprüft. Diese Funktionalität wurde bereits im Kapitel 5.2.1.2 beschrieben. Nun wurde sie von den einzelnen Tests in das TestUpdateFirmware Programm übernommen. Deswegen wird auf das Implementationskapitel verzichtet.

## 5.4 TTIC2

In

Abbildung 16 sind die beteiligten Testumgebungselemente, welche für die Erweiterung des TTIC2 gebraucht werden ersichtlich.



Abbildung 16: TTIC2 Zugriff auf SoftwareVersionsDatabase

Die SoftwareVersionsDatabase liefert auf der einen Seite die Hardware Anforderungen. Andererseits werden die Testresultate abgespeichert. In den nächsten Unterkapiteln werden die einzelnen Arbeitsschritte genauer erläutert.

### 5.4.1 Auslesung der Testinformationen

#### 5.4.1.1 Entwurf

Die wichtigste Frage im Entwurf zur Auslesung der Testinformationen befasst sich damit, wann die Daten ausgelesen werden sollen. Dabei gibt es mehrere Ansätze. Einer ist die Daten können beim Aufstarten des TTIC2 ausgelesen und anschliessend zwischengespeichert werden. Dies hat zum Vorteil, dass anschliessend die Daten immer schnell zur Verfügung stehen. Der zweite Ansatz ist, dass die Daten nur bei Anforderung ausgelesen werden. Hier liegt der Vorteil darin, dass die Daten nur ausgelesen werden, wenn sie auch wirklich benötigt werden und somit braucht das Programm weniger lange beim Aufstarten. Da die Applikation nicht zeitkritisch ist, werden die Daten nur bei Anforderung ausgelesen.

#### 5.4.1.2 Implementierung

Die erste Frage stellt sich, welche Informationen werden minimal beim Aufstarten des TTIC2 benötigt, um anschliessend die Testinformationen auf Aufforderung zu bearbeiten. Unabhängig von den Testinformationen muss zuerst eine Liste mit allen verfügbaren Tests erstellt werden. Dies erreicht man mit der Suche im Test Pfad nach exe Files. Der Pfad wird durch den Benutzer beim Aufstarten des TTIC2 ausgewählt.

Die Hardware Anforderungen werden mit der Testversion verlinkt. Hat man eine Referenz zu der Testversion kann man nicht nur die verlinkten Hardware Anforderungen auslesen sondern auch die generellen Testversions Informationen. Die generellen Testinformationen können mit Hilfe des Testnamens ausgelesen werden.

## Fazit

Zusätzlich wird die Referenz zur Testversion gebraucht. Die Testversion ist nur Inhalt des einzelnen Tests. Da die vorherigen Testversionen auch noch unterstützt werden müssen, kann nicht die aktuellste Version in der Datenbank zum entsprechenden Test hinterlegt werden. Somit muss unweigerlich der einzelne Test aufgerufen werden und seine Testversion abgefragt werden. Dies wird nachdem der Benutzer den Pfad angegeben hat durchgeführt und jeder Test wird einzeln abgefragt. In einem Ladebalken wird der Fortschritt grafisch dargestellt (Abbildung 17).

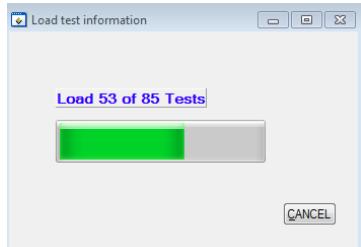


Abbildung 17: Fortschrittbalken Auslesung Testversionen

### 5.4.2 Hinterlegung des Grundzustandes

#### 5.4.2.1 Entwurf

Um zu einem späteren Zeitpunkt ein Fehlerfall wieder reproduzieren zu können, ist es unumgänglich die gleiche Ausgangslage wie vor der Ausführung des Tests herzustellen. Dazu zählen die einzelnen Firmwares, welche das Ventil benötigt. Zusätzlich können kundenspezifische Ventileinstellungen in einem File definiert werden. Diese können sich über die Laufzeit verändern und haben zusätzlich einen Index, welcher hochgezählt wird. Außerdem kann auch die Reihenfolge der Tests einen Einfluss auf das Testergebnis haben, welche in der Testkollektion definiert wird.

#### 5.4.2.2 Implementierung

Nachdem der Benutzer den Pfad der Tests ausgewählt hat, erscheint die Standardansicht des TTIC2, in der die Testkollektion zusammen geführt wird. Neu in dieser Masterarbeit realisiert, kann im oberen Bereich der Ansicht der Grundzustand vom Benutzer definiert werden (Abbildung 18).

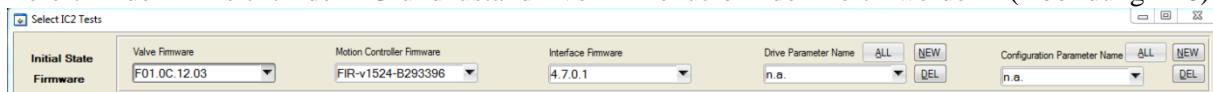


Abbildung 18: Definition Grundzustand TTIC2

Beim Aufstarten des TTIC2 wird die aktuell angeschlossene Ventil Hardware ausgelesen und ihre Werte werden in die Auswahlfelder gesetzt. Die Auswahlfelder werden mit den Werten, welche in der SoftwareVersionsDatabase hinterlegt sind, gefüllt. Ist ein Eintrag nicht in der Datenbank zu finden, so wird der Eintrag undef gesetzt. Bei der Valve Firmware kann eine trunk Version manuell zum Laden ausgewählt werden. Ist kein Feldbus Interface angeschlossen (EtherCAT und DeviceNet) wird auch hier der undef Eintrag gesetzt. Wenn keine kundenspezifische Ventileinstellungen ausgewählt werden (Drive- und Configuration File), werden alle Ventileinstellungen auf Standard zurückgesetzt.

Da die Drive- und Configuration Files nicht in der Firmware Database bearbeitet werden können, liefert die TTIC2 Oberfläche die Möglichkeit den Filennamen zu erstellen oder zu entfernen. Ist das File, welches gelöscht werden möchte, schon in einem älteren Testresultat verwendet worden, so kann dieser Eintrag nicht gelöscht werden, sondern der Activity State wird auf False gesetzt. Diese Files führen einen Index, wird dieser erhöht, so wird der Activity State des letzten aktiven Files auf False gesetzt. Mit dem All-Knopf werden alle inaktiven Files aufgelistet.

Ist der Grundzustand und die Testzusammenstellung vom Benutzer ausgewählt worden, startet der Ablauf der Testkollektion mit dem Drücken des Run-Knopfes. Jetzt wird geprüft, ob der definierte Grundzustand sich bereits auf der Ventil Hardware befindet. Falls nicht wird dieser automatisch hergestellt. Ausnahmen sind; wenn bei der Valve Firmware eine trunk Version ausgewählt ist oder bei den anderen Firmwaren der Eintrag undef ausgewählt ist, außer wenn es sich nicht um ein Feldbus Interface handelt. Bei diesen Ausnahmen wird der Benutzer aufgefordert die Firmware manuell auszuwählen.

### 5.4.3 Abspeicherung der Testresultate

#### 5.4.3.1 Entwurf

Zur Auswertung ist einerseits die Ausgangslage von besonderer Bedeutung, um später ein Fehlerfall nachbilden zu können. Weiter ist interessant, welche Tests sich in der Kollektion befinden. Die einzelne Auswertungen der Tests mit den dazugehörigen Fehlermeldungen in einem negativen Testfall dürfen dabei nicht fehlen.

#### 5.4.3.2 Implementierung

Startet der Ablauf der Testkollektion, werden zuerst alle Parameter zurückgesetzt. Sind in der Ausgangslage Drive- oder Configuration Files definiert, so werden diese Parameter jetzt gesetzt. Anschliessend wird dieser Zustand abgespeichert, um diesen nach jedem ausgeführten Test wieder neu zu setzen. Im nächsten Schritt wird der Grundzustand und die Test Kollektion in der SoftwareVersions-Database hinterlegt. Weiter wird auch die verwendete Ventil Hardware abgespeichert. Startet ein einzelner Test so wird dieser in der ResultTestCollection Tabelle eingetragen. Nach Ablauf des Tests werden sein Resultat und die Endzeit neu gesetzt. Dabei fällt auf, dass die von CVI zur Verfügung gestellte Funktion GetSystemTime für die Stunde nach Mitternacht den Wert 24 liefert. Hingegen wird dieser Wert mit 0 in der Datenbank hinterlegt und auch wieder ausgelesen. Dies musste in der Implementation berücksichtigt werden. Je nach Einstellung der «soll der Test bei einem Fehler abgebrochen werden» wird der Testablauf bei einem Fehler abgebrochen oder fortgeführt. Nach Ablauf der Testkollektion wird der Eintrag mit der Anzahl fehlerhaften Tests sowie die Endzeit der Testkollektion neu gesetzt. Dazu wird bei einem Abbruch deren Grund hinterlegt.

## 5.5 ETIC2

In diesem Kapitel wird erklärt, wie die Testresultate im ETIC2 angezeigt werden, um eine schnelle und einfache Auswertung vornehmen zu können. Weiter soll eine einfache Suche nach Fehlermeldungen inkludiert sein, um eine Aussage zu treffen, ob das Fehlerbild schon in der Vergangenheit aufgetreten ist. In der Disposition wurde zudem die Aufgabe die Applikation mit Unit Tests zu verifizieren definiert. Da das ETIC2 nur Daten von der Datenbank über das Entity Framework wiedergibt, wird aber bewusst darauf verzichtet.

### 5.5.1 Anbindung SoftwareVersionsDatabase

#### 5.5.1.1 Entwurf

Das ETIC2 braucht Zugriff auf die Testresultate der SoftwareVersionsDatabase, um diese später anzuzeigen. Hierbei handelt es sich um eine bereits existierende Datenbank.

#### 5.5.1.2 Implementierung

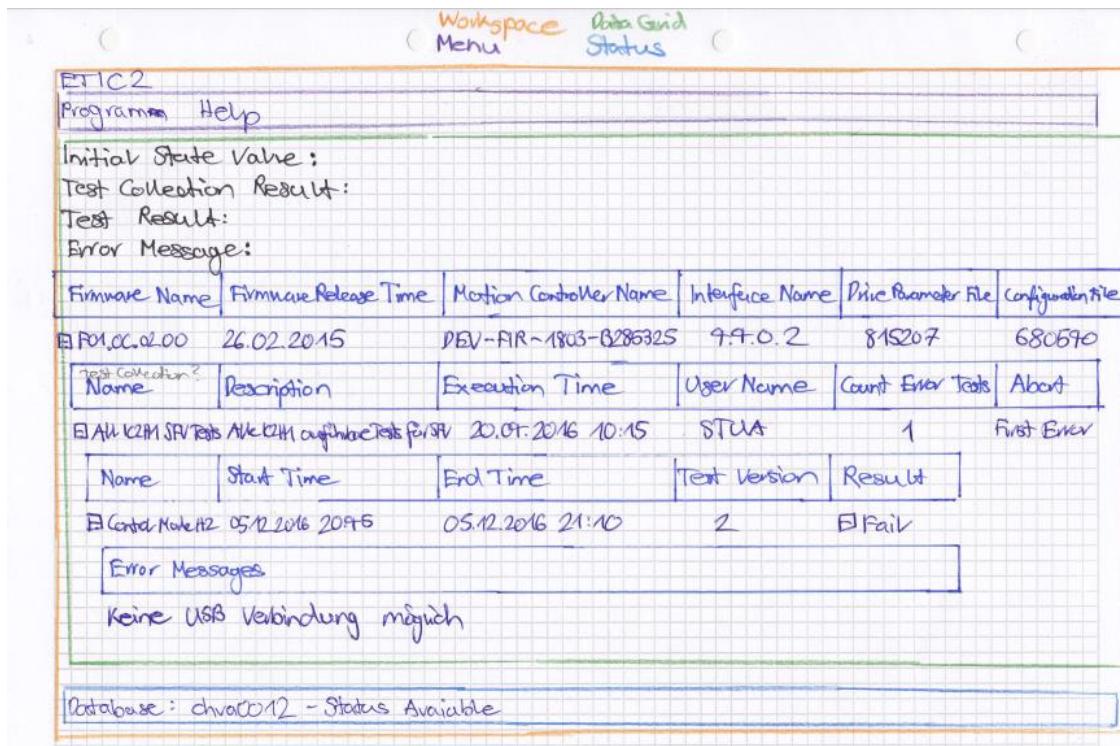
Das .NET Framework, zu welchem auch das Visual Studio gehört, bietet gratis das Entity Framework für relationale Datenbanken an. Dieses bietet verschiedene Modellierungsansätze an, je nachdem ob die Datenbank schon besteht oder ob der Code oder das Model zuerst kommt. Da die Datenbank schon besteht und die manuelle Erstellung von Entity-Klassen sehr arbeitsintensiv ist, wird der Ansatz Model First mit bestehender Datenbank gewählt. Mit Hilfe des Assistenten vom Entity Framework wird nun aus der bestehenden Datenbankstruktur das passende Modell erstellt, diese werden in entsprechende Klassen umgewandelt und im Projekt inkludiert. Im erzeugten Code wird einzig der Datenbankkontext angepasst, so dass ein Verbindungsstring übergeben werden kann.

### 5.5.2 Codierung nach MVVM

#### 5.5.2.1 Entwurf

Durch die Auswahl der Programmiersprache C# für das Auswertungstool ETIC2 (Kapitel 4.2.3) und den bereits erworbenen Erfahrungen aus der Vergangenheit mit der Firmware Database sowie der erworbenem DevExpress Bibliothek für WPF Elemente wird das ETIC2 mit dem MVVM Entwurfsmuster gelöst.

In der Entwurfsphase lag der Fokus in der Gestaltung der Ansicht (Abbildung 19). Dies wurde auch schon beim Start der gesamten Arbeit überlegt, da die Modellierung der Datenbank sehr entscheidend ist und auf die Applikation abgestimmt ist. Das Ziel lag im einfachen Aufbau und der schnellen Auswertung der Testresultate. Um dies zu erreichen teilte ich die Daten in unterschiedliche Ebenen ein. Dabei stellte sich die Frage, welche Daten stehen im Vordergrund. Diese konnte einerseits mit der Firmware beantwortet werden, da für einen Testreport einer Firmware die Auflistung aller ausgeführten Tests zu dieser Firmware interessieren. Wie bekannt führt das TTIC2 nicht einzelne Tests aus sondern eine Test Kollektion. Da nicht nur die Ventil Firmware entscheidend ist für das Testergebnis sondern auch die Motion Controller sowie die Interface Firmwares, werden diese zusammen in der gleichen Ebene aufgelistet. Zum Ausgangszustand sind neben den Firmwares auch die kundenspezifischen Einstellungen sowie die ausgewählte Test Kollektion entscheidend.



**Abbildung 19: ETIC2 Konzept Ansicht Testresultate**

Für die Einführung und Garantie über Einhaltung der Anforderungen an die Ventilmechanik ist es wichtig die einzelnen Tests auf den verschiedenen Ventilreihen auszuführen. Um dies einfach und schnell zu kontrollieren sowie einen Testreport erstellen zu können, soll es eine weitere Ansicht geben, in welcher diese Informationen in der obersten Ebene angezeigt werden.

In den unteren Ebenen sollen Informationen zur Anzahl fehlerhaften Tests folgen sowie der Auflistung der einzelnen Tests. Wird in einem Test ein Fehlerfall festgestellt, so sollen die letzten 3 Fehlermeldungen aufgelistet werden. Diese sollen nach ähnlichen in der Vergangenheit aufgetretenen Fehlern durchsucht werden.

### 5.5.2.2 Implementierung

In der Umsetzungsphase würde als erstes die Umsetzung der gewünschten Ansicht geprüft. Dabei wurde als erstes die DevExpress Bibliothek nach nützlichen Datenansichten durchsucht. Dabei stellte sich schnell heraus, dass das Master-Detail via Entity Framework Element sehr nahe an das Konzept kommt. (Abbildung 20).

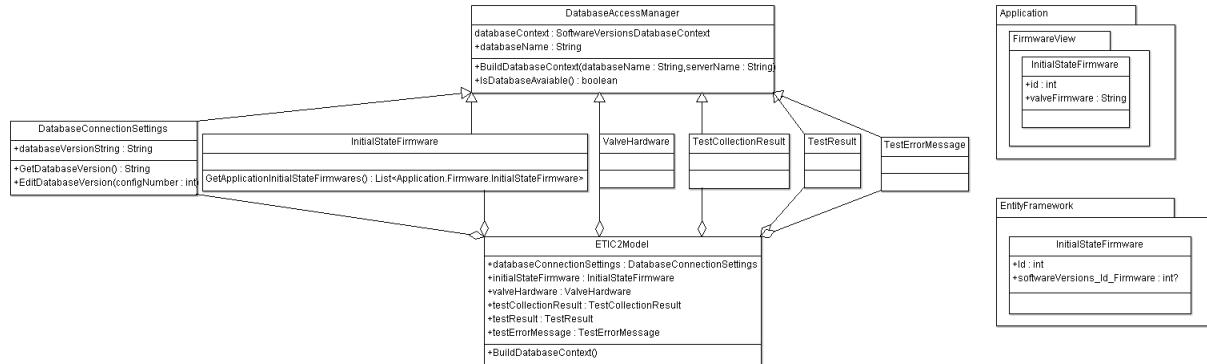
Contact Name	Country	City	Address	Postal Code	Customer ID
Maria Anders	Germany	Berlin	Obere Str. 57	12209	ALFKI
Order Date	Shipped Date	Ship Postal Code	Freight	Ship Country	
25.08.1997	02.09.1997	12209	29.46	Germany	
Product Name	Unit Price	Quantity	Total Price	Discount	
Rössle Sauerkraut		45.6	684.0		0.25
Chartreuse verte		18	378		0.25
Spegesild		12	24		0.25
Anzahl=3			Sum=38	Sum=1086	
03.10.1997	13.10.1997	12209	61.02	Germany	
13.10.1997	21.10.1997	12209	23.94	Germany	
15.01.1998	21.01.1998	12209	69.53	Germany	
16.03.1998	24.03.1998	12209	40.42	Germany	
09.04.1998	13.04.1998	12209	1.21	Germany	
Anzahl=1					

**Abbildung 20: DevExpress Master-Detail via EntityFramework Element**

## Model

Im Model werden die Testresultate von der SoftwareVersionsDatabase ausgelesen. Es soll eine Trennung der Datenbankklassen von der Applikation erreicht werden.

Der DatabaseAccessManager stellt eine Verbindung durch Übergabe des Verbindungsnamen mit dem vorher erstellten Datenbankkontext vom Entity Framework her (Abbildung 21).



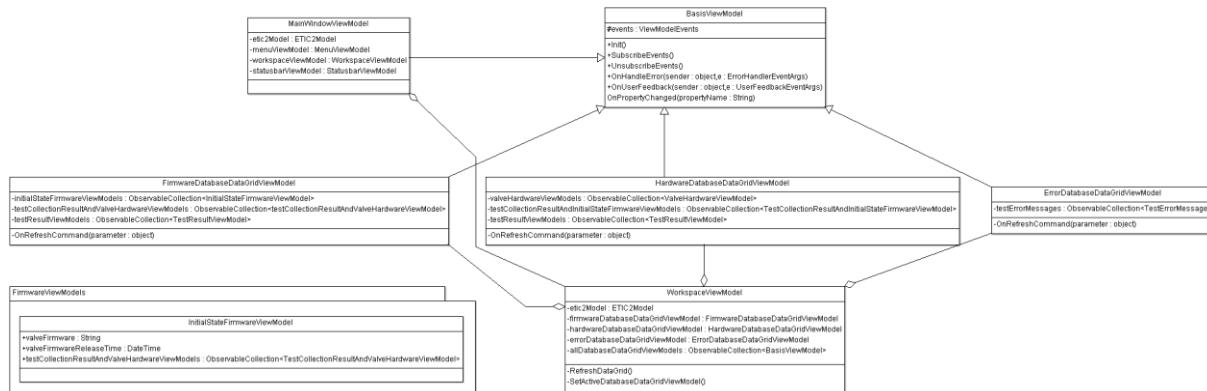
**Abbildung 21: Model ETIC2 in UML**

Die DatabaseConnectionSettings Klasse wird erstellt, um der Datenbankversion eine Nummer zu geben, welche später für die Überprüfung, ob die Applikationsversion mit der Datenbankversion dient. Weiter wurden Klassen (InitialStateFirmware, ValveHardware, TestCollectionResult, TestResult, TestErrorMessage) erstellt, um die Daten aus der Datenbank abzuholen und diese in verwendbare Daten für die Applikation umzuwandeln. Das heisst, dass die Referenzen ersetzt werden mit den realen Datenwerten. Dazu wurden die Klassen für die Applikation in einem eigenen Namensraum definiert.

Das ETIC2 Model beinhaltet alle Datenbankobjekte, welche vom DatabaseAccessManager ableiten. Diese werden später im ViewModel angesprochen, um die Daten auszulesen. Weiter besitzt das ETIC2 Model auch eine Funktion, um die Datenbankverbindung mit allen Datenbankklassen herzustellen. Da diese in der Applikation geändert werden können.

## ViewModel

Um das Master-Detail Element umsetzen zu können, müssen die Daten der einzelnen Werte zusammengefasst werden. Die unteren müssen in der oberen Ebene in einer Liste eingetragen werden. Das wird in der InitialStateFirmwareViewModel Klasse gezeigt, welche die oberste Ebene der Firmware Ansicht darstellt (Abbildung 22).



**Abbildung 22 ViewModel ETIC2 in UML**

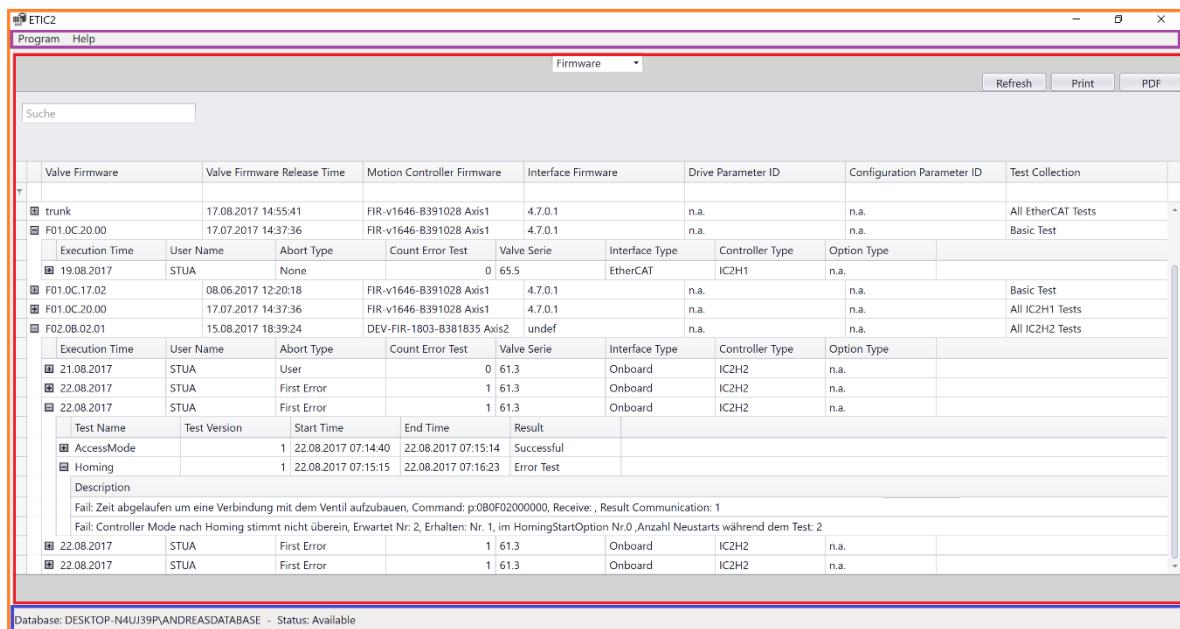
In der unteren Ebene werden die Informationen der Test Kollektion sowie der verwendeten Test Hardware aufgelistet. Die gesamten Daten sind in der FirmwareDatabaseDataGridViewModel hinterlegt. Es gibt noch zwei weitere, bei einer wird die Hardware Auflistung nach dem gleichen System entwickelt. Bei der anderen gibt es noch eine Ansicht der Fehler, welche nicht in Ebenen aufgeteilt ist.

Der Verwendungszweck wird anschliessend im Unterpunkt View erklärt. In Zukunft werden die Ansichten noch erweitert mit der Buglist, um die Fehler im ETIC2 bearbeiten zu können (Kapitel 8.1.2). Diese einzelnen Ansichten werden im WorkspaceViewModel verwendet, wobei nur eine aktiv sein kann. Das WorkspaceViewModel wird im MainWindowViewModel erzeugt. Welches noch weitere ViewModels für das Menü sowie der StatusBar besitzt.

Alle ViewModel Klassen leiten von der BasisViewModel Klasse ab. Diese Klasse leitet ihrerseits von INotifyPropertyChanged ab, welche benötigt wird um Änderungen einer Geschäftsobjekteigenschaft an ein gebundenes Steuerelement mitzuteilen. Weiter besitzt das BasisViewModel virtuelle Methoden (Init, SubscribeEvents, UnsubscribeEvents), um diese in den ViewModels zu überschreiben. Dazu besitzt die Klasse Events, um ein Dialog zu öffnen oder den Benutzer über Auffälligkeiten oder Fehler zu informieren.

## View

Die Ansicht des TTIC2 ist in vier Module unterteilt (Abbildung 23).



## Workspace, Menu, DataGridView, Status

Abbildung 23: View ETIC2

Das Workspace umfasst die ganze Ansicht. Darin befindet sich im oberen Bereich das Menü. In welchem die Datenbankeinstellungen geändert werden können. Zusätzlich können weitere Informationen über die Applikation abgefragt sowie die Applikation beenden werden.

Unterhalb des Menüs kann der Benutzer mit Hilfe eines Auswahlfeldes bestimmen, welche Ansicht des DataGridViews er gerne haben möchte. Dabei kann er zwischen Firmware, Hardware und Error selektieren. Wählt er zum Beispiel Firmware aus, so wird in der ersten Ebene der Grundzustand der Firmwares aufgelistet. Die Ansicht umfasst vier Ebenen bis in der untersten die einzelnen Fehlermeldungen aufgelistet werden. In der Implementierungsphase fiel auf, dass das Master-Detail Element von DevExpress Einschränkungen bezüglich des Suchpanels hat. Es kann nämlich nur über die erste Ebene gesucht und gefiltert werden, was für den Anwendungsfall nicht akzeptabel ist. Die Recherche bei DevExpress hat ergeben, dass mit einer kleinen Anpassung auch die zweite Ebene durchsucht werden kann. Was aber bei dieser Alternative immer noch eine Einschränkung ist, dass nur noch zwei Ebenen mit dem Master-Detail Element möglich sind. Um die Suche auch über vier Ebenen zu ermöglichen, müsste man zwangsläufig ein eigener Filter implementieren und auch die entsprechenden Ebenen müssten aufgeklappt werden. Dieser Ansatz wurde aus Zeitgründen nicht weiter verfolgt.

## Fazit

Das Master-Detail bietet nur eine Suche in der ersten Ebene an, was schlussendlich zur Lösung führte. Die Fehlermeldungen müssen zwangsweise in der ersten Ebene aufgelistet werden. Die Suche nach den Fehlermeldungen wird gebraucht, wenn bei der Auswertung ausfällt das ein Fehler aufgetreten ist. Anschliessend möchte man herausfinden, ob dieser schon einmal aufgetreten ist. Wenn dieser öfters auftrat, möchte man eine Systematik des Fehlers hinsichtlich der verwendeten Hardware oder auch der Software herausfinden. Dies liefert also neben der Anforderung die Fehlermeldungen in die erste Ebene zu nehmen auch noch, dass zusätzlich Hardware und Firmware Informationen in dieser aufgelistet werden. Dies wurde schliesslich auch unter dieser Ansicht umgesetzt, in der Tabelle 4 sind nochmals alle Lösungsvorschläge und ihre Einschränkungen kurz aufgelistet.

Lösungsvorschlag	Einschränkung
Detail-View über vier Ebenen	Suchfeld nur über erste Ebene
Detail-View mit Workaround Ansatz DevExpress (DevExpress, 2016)	Detail-View nur über zwei Ebenen
Detail-View über vier Ebenen mit eigenem Filter	Nicht innerhalb geeigneter Zeit implementierbar
Detail-View mit einer Ebene	Keine Einteilung der Daten mehr möglich

**Tabelle 4: Lösungsvorschläge Suche nach Fehlermeldung in ETIC2**

Die letzte Ansicht der Hardware ist das Konzept mit den vier Ebenen so wie bei der Firmware Ansicht. Der einzige Unterschied liegt darin, dass bei der Hardware Ansicht die erste Ebene von der Hardware eingenommen wird und die Informationen bezüglich der verwendeten Firmwares wird in der zweiten Ebene aufgelistet. Alle drei Ebenen können mit einem Refresh Knopf upgedatet werden. Dabei werden alle Daten neu aus der Datenbank ausgelesen und in die Ansicht eingefügt.

Unterhalb dieser DataGrid Ansicht befindet sich die Datenbank Statuszeile. Diese liefert den aktuellen Datenbanknamen der Verbindung sowie ihren Status.

## **5.5.3 Ausgabe Bericht**

### **5.5.3.1 Entwurf**

Für die interne Analyse sowie die Auszeichnung gegenüber dem Kunden in Bezug auf die Software Qualität ist eines der wichtigsten Ziele des ETIC2, auf Knopfdruck einen Report der Testresultate erstellen zu können. Das Drucken der Testansicht wird als weitere eine nützliche Funktion erachtet.

### **5.5.3.2 Implementierung**

Die DevExpress Bibliothek liefert viele Möglichkeiten wie das DataGrid exportiert werden kann. Zu den möglichen Formaten gehören PDF, SLS, XLSX, RTF und CSV. Um die Daten an einen Kunden zu schicken, bietet sich das PDF Format an, da dieses Format auf allen Betriebssystemen gleich aussieht. Die Standardansicht des PDF ist im Hochformat, aber dies ist bei einer breiten Tabelle nicht von Vorteil. Aus diesem Grund werden die Testresultate des ETIC2 im Querformat erstellt. Das wurde auch mit dem CSV Format ausprobiert, da es aber nur die erste Ebene auflistet, wird es nicht als brauchbar eingestuft.

Hinsichtlich der Möglichkeit die Ansicht zu drucken, bietet DevExpress zwei Standardfunktionen an. Einerseits kann eine Druckvorschau erstellt werden oder andererseits gleich direkt gedruckt werden. Die Vorschau macht nicht viel Sinn, da auch hier nur das Querformat in Frage kommt. Der zu verwendende Drucker kann auch in der direkten Form noch ausgewählt werden, dadurch wurde die direkte Druckfunktion implementiert. Alle drei unterschiedlichen Ansichten der Testresultate können über die Knöpfe PDF exportiert bzw. über Print gedruckt werden. Diese Funktionen sind im Codebehind programmiert worden, da es ausschliesslich mit der View zu tun hat. Zusätzlich im Codebehind wurde gelöst, dass die Einstellungen der Ansichten beim Schliessen der Applikation abgespeichert und beim nächsten Aufstarten wieder übernommen werden.

## 6 Ergebnisse

Hier wird beschrieben, ob und wie die in Kapitel 3.3 und 3.4 definierten Ziele erfüllt wurden.

### 6.1 Quantitative Ziele

- Jede einzelne ausgeführte Testkollektion muss im ETIC2 zu einem Grundzustand zugeordnet werden.

Dieses Ziel wurde erreicht, die Modellierung des Grundzustandes (InitialStateFirmware Tabelle) erzwingt in den Testresultaten immer einen Wert für die Testkollektion (Abbildung 24). Die Firmware Ansicht wird in der obersten Ebene und die Hardware Ansicht in der zweitobersten Ebene der Testkollektion aufgelistet.

CHVL0281\SOFTWARE DATABASES.SoftwareVersionsDatabase - dboInitialStateFirmware		
Column Name	Data Type	Allow Nulls
Id	int	<input type="checkbox"/>
SoftwareVersions_Id_Firmware	int	<input type="checkbox"/>
ValveFirmwareReleaseTime	datetime	<input type="checkbox"/>
SoftwareVersions_Id_MotionController	int	<input type="checkbox"/>
SoftwareVersions_Id_Interface	int	<input type="checkbox"/>
DriveParameterFile_ID	int	<input type="checkbox"/>
ConfigurationParameterFile_ID	int	<input type="checkbox"/>
TestCollection_Id	int	<input type="checkbox"/>

Abbildung 24: Modellierung InitialStateFirmware der SoftwareVersionsDatabase

- Der Grundzustand kann nur mit bereits vorhandenen Einträgen in der SoftwareVersionsDatabase definiert werden.

Dieses Ziel wurde mit der Modellierung des Grundzustande erreicht. Aus der Abbildung 24 ist ersichtlich, dass nur Referenzen der SoftwareVersions Tabelle für die Firmwaren eingetragen werden können. Für die trunk Version sowie wenn es keine Interface Firmware hat (kein Feldbus Interface) wurde zusätzlich ein undef Eintrag in der SoftwareVersions Tabelle erstellt.

- Die SoftwareVersionDatabase muss Schreibanfragen von vier Benutzern bearbeiten können.

Dieses Ziel entstand aus der Anforderung, dass vier TTIC2 auf unterschiedlichen Computern ausgeführt werden müssen. Das Ventilportfolio und die Anzahl der Tests vergrössern sich laufend, dieser Umstand soll mit Hilfe von mehreren gleichzeitig laufenden Testventilen gelöst werden.

Zuerst stellt sich die Frage, welche Programme auf die SoftwareVersionsDatabase schreiben. Die Antwort lautet einerseits; das TestUpdateFirmware Programm, welches bei einer Freigabe der Firmware, die Testinformationen sowie die Hardware Anforderungen der Tests aktualisiert. Dies wird nur auf einem Rechner ausgeführt, somit stellt es kein Problem dar. Andererseits schreibt das TTIC2 seine Testresultate in die SoftwareVersionsDatabase. Das TTIC2 wird aber auf mehreren Rechnern ausgeführt, deshalb taucht die Frage nach einem lost update Problem auf.

Dieses Problem konnte in der Realität noch nicht überprüft werden. Da sich die umgesetzte Arbeit noch in der Testphase befindet. Es wird erst auf lokalen Datenbanken ausgeführt und erst zu einem späteren Zeitpunkt produktiv auf einer Remote Datenbank eingesetzt.

Nun wird die Implementierung der Abspeicherung der Testresultate genauer erläutert, um das Risiko des Problems einschätzen zu können. Bei der Ausführung der Testkollektion erstellt das TTIC2 einen neuen InitialStateFirmware Eintrag, falls dieser noch nicht existiert. Anschliessend speichert es diese Referenz (ID des Eintrages) intern. Anschliessend wird ein neuer TestCollectionResult Eintrag erstellt. Die Referenz wird erneut zwischengespeichert, um bei Beendigung der Testkollektion die Attribute CountErrorTests und AbortTypeId im Bedarfsfall anzupassen. Wenn ein Test gestartet wird, so wird ein neuer Eintrag in der TestResult Tabelle erstellt. Mit der zwischengespeicherten Referenz wird die Endzeit und im Bedarfsfall das Resultat angepasst.

### Fazit der Analyse

Es unmöglich ist, dass mehrere TTIC2 Programme auf den gleichen Eintrag einer Tabelle schreiben. Es ist aber möglich, dass ein neuer InitialStateFirmware oder ValveHardware Eintrag in einem TTIC2 Programm erstellt wird. Dieser ist zum Zeitpunkt der Erstellung noch nicht in der SoftwareVersions-Database hinterlegt und dadurch besteht die Möglichkeit, dass das zweite Programm den gleichen Eintrag nochmals erstellt.

Die Folgen sind, dass bei einem zusätzlichen ValveHardware Eintrag die Tabelle um einen unnötigen Eintrag erweitert wird. Und wird ein InitialStateFirmware Eintrag doppelt angelegt, so kommt hinzu, dass der gleiche Grundzustand im ETIC2 mehrfach aufgelistet wird.

Dieses Risiko wird aktuell bewusst eingegangen. Falls sich dies in der Praxis als ein erstzunehmendes Problem herausstellt, wird über einen Einsatz eines Windows Services nachgedacht. Dieser lokализiert über ein Zeitstempelverfahren, ob die zuvor ausgelesenen Daten wirklich aktuell waren. Ist dies nicht der Fall, wird der neu zu erstellende Eintrag verworfen.

## 6.2 Qualitative Ziele

- Das ETIC2 soll sich durch seinen einfachen und stabilen Aufbau, verbunden mit der raschen Auswertung, ob ein Fehler in der ausgeführten Testkollektion aufgetreten ist, auszeichnen.

Aus der Abbildung 25 ist der Aufbau des ETIC2 gekennzeichnet. In orange sind die Benutzersteuerungselemente zu sehen. Dabei fällt auf, dass der Benutzer einerseits die Ansicht wechseln, diese neu laden, exportieren oder direkt ausdrucken kann. In der Menü Leiste sind Funktionen verschachtelt, die nicht oft gebraucht werden und somit auch nicht prominent in der Abbildung auffallen sollen. Die Daten sind strukturiert und in Ebenen aufgeteilt, was dem Benutzer hilft sich schneller ein Überblick zu verschaffen. Er hat immer die Möglichkeit die oberste Ebene zu durchsuchen. Eine schnelle Auswertung, ob ein Fehler aufgetreten ist, wird mit dem Feld Count Error Test in der Ansicht Firmware und Hardware erreicht.

The screenshot shows the ETIC2 software interface. At the top, there is a menu bar with 'Program' and 'Help' highlighted in orange. Below the menu is a search bar with the placeholder 'Suche'. To the right of the search bar is a dropdown menu set to 'Firmware'. Further right are buttons for 'Refresh', 'Print', and 'PDF'. The main area contains two tables. The first table has columns: Valve Firmware, Valve Firmware Release Time, Motion Controller Firmware, Interface Firmware, Drive Parameter ID, Configuration Parameter ID, and Test Collection. It lists several entries with details like F01.0C.12.03, 30.06.2017 15:17:11, FIR-v1646-B391028 Axis1, 4.7.0.1, n.a., n.a., Basic Test. The second table has columns: Execution Time, User Name, Abort Type, Count Error Test, Valve Serie, Interface Type, Controller Type, and Option Type. It lists entries with details like 21.08.2017, STUA, User, 0, 61.3, Onboard, IC2H2, n.a. A red box highlights the 'Count Error Test' column in this table. At the bottom left of the interface, it says 'Database: DESKTOP-N4UJ39P\ANDREASDATABASE - Status: Available'.

**Benutzersteuerung, einfache Suche, schnelle Auswertung Fehler, Daten strukturiert**

**Abbildung 25: Aufbau ETIC2**

- Eine ausgeprägte Suchfunktion soll ein Bestandteil des ETIC2 sein, welche eine schnelle Suche nach Fehlermeldungen erlaubt.

Nachdem in der Firmware oder Hardware Ansicht ein Fehler bei der Auswertung gefunden wird, kann zur genaueren Untersuchung über bereits gleiche Auffälligkeiten in der Error-Ansicht nach ähnlichen Fehlermeldungen gesucht werden (Abbildung 26). Zusätzlich können Rückschlüsse gezogen werden, bei welcher Firmware oder Hardware der Fehler aufgetreten ist. Diese Informationen können bei der Suche nach dem Bug in der Firmware sehr hilfreich sein.

The screenshot shows the ETIC2 software interface with the title bar "ETIC2". Below it is a menu bar with "Program" and "Help". The main window has a toolbar with "Error" (selected), "Refresh", "Print", and "PDF". A search bar labeled "Target Position" is present. The main content area displays a table titled "Error Description" with the following data:

Error Description	Test Name	Test...	Valve Firmware	Valve Firm...	Valve Serie	Interface Type	Controller Type	Option...
Fai: Target Position konnte nicht erreicht werden, Actual Position: 34.200001, Target Position: 100.000000	Homing	1	F02.08.02.01	15.08.2017...	61.3	Onboard	IC2H2	n.a.
Fai: Target Position konnte nicht erreicht werden, Actual Position: 42.099998, Target Position: 100.000000	Homing	1	F02.08.02.01	15.08.2017...	61.3	Onboard	IC2H2	n.a.

**Abbildung 26: Error Ansicht des ETIC2**

- Mit dem ETIC2 soll das Resultat der ausgeführten Testkollektion unmittelbar und einfach ersichtlich sein.

Das ETIC2 kann parallel nebst dem laufenden TTIC2 betrieben werden. Das TTIC2 schreibt nach Beendigung eines Testes das Resultat in den zuvor erstellten TestCollectionResult Eintrag. Der Verlauf kann über den Refresh Knopf des ETIC2 verfolgt werden. Aber um den TestCollectionResult Eintrag einsehen zu können, muss der Benutzer den Grundzustand kennen. Wird dabei eine neue Firmware getestet, so kann nach der Firmware Release Spalte sortiert werden (Abbildung 27). In der unteren Ebene werden die Test Kollektionen nach dem Ausführungszeitpunkt aufgelistet.

The screenshot shows the ETIC2 software interface with the title bar "Firmware". Below it is a menu bar with "Program" and "Help". The main window has a toolbar with "Firmware" (selected), "Refresh", "Print", and "PDF". A search bar labeled "Suche" is present. The main content area displays a table with the following columns: Valve Firmware, Valve Firmware Release Time (highlighted with a red box), Motion Controller Firmware, Interface Firmware, Drive Parameter ID, Configuration Parameter ID, and Test Collection. The data is grouped under "trunk" and "17.08.2017". The "Valve Firmware Release Time" column is sorted. The table also includes a section for "Test Name", "Test Version", "Start Time", "End Time", and "Result".

**Nach Firmware Release Zeitpunkt sortiert, Standard nach Zeitpunkt der Test Kollektion sortiert**

**Abbildung 27: Auswertung der letzten ausgeführten Testkollektion**

- Auf Knopfdruck sollen die Testresultate exportiert werden können.

Alle Ansichten der Testresultate (Firmware, Hardware, Error) können über den PDF Knopf exportiert werden. Die Ansicht wird jeweils im Querformat im PDF Dokument abgespeichert (Abbildung 28).

Valve Firmware	Valve Firmware Relea...	Motion Controller Fir...	Interface Firmware	Drive Parameter ID	Configuration Param...	Test Collection	
trunk	17.08.2017 14:55:41	FIR-v1646-B391028...	4.7.0.1	n.a.	n.a.	All EtherCAT Tests	
F02.0B.02.01	15.08.2017 18:39:24	DEV-FIR-1803-B3818...	undef	n.a.	n.a.	All IC2H2 Tests	
Execution Time	User Name	Abort Type	Count Error Test	Valve Serie	Interface Type	Controller Type	Option Type
21.08.2017	STUA	User		0 61.3	Onboard	IC2H2	n.a.
22.08.2017	STUA	First Error		1 61.3	Onboard	IC2H2	n.a.
22.08.2017	STUA	First Error		1 61.3	Onboard	IC2H2	n.a.
22.08.2017	STUA	First Error		1 61.3	Onboard	IC2H2	n.a.
22.08.2017	STUA	First Error		1 61.3	Onboard	IC2H2	n.a.
Test Name	Test Version		Start Time		End Time	Result	
AccessMode			1 22.08.2017 08:15:42		22.08.2017 08:16:17	Successful	
Homing			1 22.08.2017 08:16:17		22.08.2017 08:17:29	Error Test	
trunk	14.08.2017 14:59:23	FIR-v1646-B391028...	4.7.0.1	n.a.	n.a.	All IC2H1 Tests	
F01.0C.20.00	17.07.2017 14:37:36	FIR-v1646-B391028...	4.7.0.1	n.a.	n.a.	All IC2H1 Tests	
Execution Time	User Name	Abort Type	Count Error Test	Valve Serie	Interface Type	Controller Type	Option Type
19.08.2017	STUA	User		0 65.5	EtherCAT	IC2H1	n.a.
Test Name	Test Version		Start Time		End Time	Result	
AccessMode			1 19.08.2017 14:14:01		19.08.2017 14:15:36	Successful	
ControlPositionRestriction			2 19.08.2017 14:15:36		19.08.2017 14:18:47	Successful	
FunctionSimple			3 19.08.2017 14:18:47		19.08.2017 15:00:05	Successful	
FunctionSimpleShort			1 19.08.2017 15:00:05		19.08.2017 15:31:27	Successful	
Homing			1 19.08.2017 15:31:27		19.08.2017 16:11:35	Successful	
LearnCheckSum			2 19.08.2017 16:11:35		19.08.2017 16:26:34	Successful	
LearnDataBase			3 19.08.2017 16:26:34		19.08.2017 16:27:59	Successful	
LearnDataCopy			3 19.08.2017 16:27:59		19.08.2017 16:31:45	Successful	
LearnDeleteBankData			1 19.08.2017 16:31:46		19.08.2017 16:53:24	Successful	
LearnOpenSpeed			2 19.08.2017 16:53:24		19.08.2017 17:32:23	Successful	
LearnPositionTable			1 19.08.2017 17:32:23		19.08.2017 17:35:10	Successful	

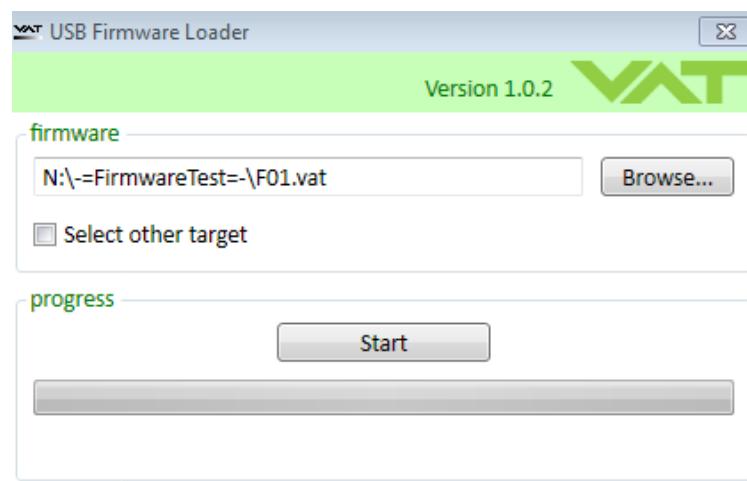
**Abbildung 28: Ansicht Testresultate im PDF**

- (optional) Das TTIC2 soll für die Anzeige der Testinformationen immer auf die aktuellsten im Test definierten Beschreibungen zugreifen. Auch die Anforderungen an die Ventil Hardware, welche in den Tests definiert werden, sollen auf dem aktuellsten Stand im TTIC2 sein. (Bei einem Release Prozess)

Die TestUpdateFirmware, welche früher vor einem Release Prozess aufgerufen wurde, um die Tests mit den aktuellen Parameterwerten von der Firmware zu aktualisieren, wurde erweitert. Diese ruft jeden einzelnen Test auf, dieser liefert seine Testinformationen und Hardware Anforderungen über Umgebungsvariablen zurück. Sie werden anschliessend auf Aktualität geprüft. So kann sichergestellt werden, dass das TTIC2 aktuelle Testinformationen bei einem Release aus der SoftwareVersionsDatabase ausliest.

- (optional) Das TTIC2 muss vor Ausführung der Test Kollektion automatisch die definierten Firmwaren und Parameterfiles (Drive- und Configuration File) auf die Ventil Hardware updaten.

Ist die Testkollektion im TTIC2 definiert und die Ausführung gestartet, wird nachdem die Hardware überprüft wurde kontrolliert ob der Grundzustand sich bereits auf der Ventil Hardware befindet. Wenn nicht wird automatisch ein Firmware Loader Programm aufgerufen, welches die angegebene Firmware im internen VAT Firmware Ordner auswählt und updatet (Abbildung 29). Wenn es sich bei der Ventil Firmware um eine trunk Version handelt, muss der Benutzer diese Firmware manuell im Ordner auswählen, bevor sie upgedatet werden kann.



**Abbildung 29: Firmware Loader**

## 7 Diskussion

Erfreulicherweise wurden alle im Vorhinein gesteckten Ziele erreicht. Das primäre Ziel, eine einfache und schnelle Auswertung der ausgeführten Tests vornehmen zu können, wurde über einen kleinen Umweg mit einer zusätzlichen Error Ansicht erreicht.

In der Startphase der Arbeit wurde schnell klar, dass sich die Arbeit nicht nur um das Erstellen der ETIC2 Applikation dreht. Die Entscheidung das Auswertungstool zu erstellen und diese mit Pseudodaten aus der Datenbank zu füllen, machte wenig Sinn. Um die Anforderung der Arbeit erreichen zu können, mussten Resultate erfasst werden. Dies war kompliziert, weil sie auf mehreren Computern benutzt und ausgeführt werden. Es führte deshalb kein Weg an der Erweiterung der SoftwareVersionsDatabase mit den Testresultaten vorbei. Wenn die Datenbank schon erweitert wird, können gleich auch die Testinformation sowie die Hardware Anforderungen miteinbezogen werden. Diese müssen, auch verteilt auf allen Computern, die gleichen Informationen dem Benutzer liefern. Schliesslich wurde die SoftwareVersionsDatabase noch mit den Feldern für die Erstellung einer Bugliste vorgesehen, welche momentan noch in einem Excel Dokument geführt wird. All dies führte dazu, dass der Hauptfokus der Arbeit sich verlagerte zu der Erstellung einer funktionalen Testumgebung. Dabei war der Arbeitsaufwand grösser bei der Abstimmung der einzelnen Testumgebungselemente als die Erstellung des Auswertungstools.

Das Zusammenspiel aller Testumgebungselemente wurde bereits erfolgreich ausgeführt. Zwar wurden diese auf mehreren Computern mit einer lokalen Datenbank ausgeführt. Das liegt daran, dass die SoftwareVersionsDatabase seit längerem schon produktiv eingesetzt wird. Erfreulicherweise kann diese nach den ersten erfolgreichen Testphasen erweitert werden. So sollen die ersten Kinderkrankheiten vorher eliminiert werden.

Diese Arbeit sowie die früheren Arbeiten des TTIC2 zeigen, dass es sich lohnt bei der Erstellung einer Testumgebung in der ersten Startphase gut zu überlegen, mit welchen Hilfsmitteln die Anforderungen erreicht werden können. Meistens wird mit dem Tool gestartet, in dem die meisten Kenntnisse vorhanden sind und man sich am wohlsten fühlt. Mit der Zeit steigt die Komplexität stetig an und es ist hinsichtlich der Testanforderungen nicht mehr möglich, die verschiedenen Kommunikationsschnittstellen abzudecken speziell bei Kommunikationen über ein Feldbusssystem. Da diese von CVI nicht unterstützt wurden, musste hier mit LabVIEW gearbeitet werden. Zusätzlich wurde das Bedürfnis immer grösser die Tests in RealTime Umgebungen laufen zu lassen, um den Praxisfall bei der EtherCAT Kommunikation abdecken zu können.

Persönlich stellte ich mir die Lösung der Suche nach Fehlermeldungen einfacher vor. Da es sich bei der verwendeten Ansicht um ein von DevExpress entwickeltes Element handelt, welches Informationen in mehreren Ebenen unterteilt. Ich sehe ein grosses Bedürfnis des Anwenders darin Informationen über alle Ebenen zu suchen. Im Nachhinein sehe ich aber auch grosse Vorteile in der aktuellen Lösung mit einer zusätzlichen Ansicht, in der die Fehlermeldungen in der obersten Ebene angezeigt werden. So kann schnell verglichen werden, ob es Auffälligkeiten in Bezug auf Hardware oder Software gibt, wenn der Fehlerfall in der Vergangenheit schon aufgetreten ist. Da in diesem Fall alle Informationen in der gleichen Ebene zu finden sind und diese nicht verschachtelt aufgelistet werden.

## 8 Ausblick

Zuerst werden die offenen Punkte analysiert im zweiten Teil folgt der Ausblick mit den nächsten möglichen Schritten.

### 8.1 Offene Punkte

#### 8.1.1 Umsetzung Überarbeitung SoftwareVersionsDatabase

Wie im Kapitel 5.1.1 **Fehler! Verweisquelle konnte nicht gefunden werden.** beschrieben, besitzt das aktuelle SoftwareVersionDatabase Modell noch Verbesserungspotential. Die Umsetzung braucht die Anpassung der Software Verwaltungstool Oberfläche. Dies wurde in WPF und danach im MVVM Pattern Konzept erstellt. Nun darf die Anpassung keinen Einfluss auf das ModelView wie auch auf die View haben. Daher liegt der Ansatz nahe das MVVM Entwurfsmodell auf die neue Modellierung der SoftwareVersionsDatabase anzupassen. Dazu wird eine Wrapper Klasse erstellt, in der die Umwandlung der alten auf die neue Struktur erfolgt.

#### 8.1.2 Integration Buglist in ETIC2

Die Datenbankfelder für die Verwaltung der Firmware Fehler wurde bereits in dieser Arbeit erstellt. Zusätzlich soll im ETIC2 die Option bestehen diese Firmware Fehler anzeigen zu können. Auch hier ist es wichtig eine schnelle Möglichkeit zu haben, nach Fehlern zu suchen. Weiter soll dem User die Möglichkeit gegeben werden, die Firmware Fehler auch bearbeiten zu können.

Die Ansicht besteht aus einem ausgewählten Eintrag, wo alle Informationen im oberen Bereich aufgelistet werden. Der grösste Teil nimmt die Ansicht aller in der Datenbank abgespeicherten Einträge ein, siehe Anhang B.2. ETIC2 Diese soll über das bereits bestehende Auswahlfeld des ETIC2 ausgewählt werden können.

#### 8.1.3 Logfile in ETIC2

Aktuell werden alle auftretenden Ereignisse dem User über einen Dialog mitgeteilt. Diese ist nur zum aktuellen Zeitpunkt den Benutzern ersichtlich. Diese sollen zusätzlich in einem Logfile hinterlegt werden, um diese später Auswerten und mögliche Bugs in der Implementierung ausmerzen zu können.

### 8.2 Nächste Schritte

Als nächstes wird wie schon in der Diskussion erwähnt weiter die aktuelle Testumgebung auf lokalen Datenbanken betrieben. Wird hier in den nächsten Release Prozessen keine Auffälligkeiten gefunden, so wird die SoftwareVersionsTabelle auf dem SQL Server erweitert, um die Testumgebung in Zukunft produktiv einsetzen zu können. Solange die Kenntnisse des ETIC2 noch frisch sind, ist die Einbindung der Verwaltung von Firmware Bugs angedacht. Zusätzlich soll sicherlich auch ein Logfile erstellt werden, um die Exceptiones zu hinterlegen. Die Firmware Database mit der aktuellen Modellierung der Firmware Informationen stellt aktuell keine Probleme im Betrieb dar. Dadurch wird die Bereinigung nicht die höchste Priorität erhalten. Was aber in naher Zukunft zwingend implementiert werden muss, ist eine Refresh Möglichkeit der Firmware Informationen im laufendem Betrieb, da die Applikation von einigen Benutzern über den Arbeitstag hinweg nicht mehr geschlossen wird.

## 9 Verzeichnisse

### Literaturverzeichnis

- Community. (2016). Abgerufen am 31. August 2017 von SQL Server 2012 connection strings:  
<https://www.connectionstrings.com/sql-server-2012/>
- DevExpress. (27. April 2016). *Search panel for detail views work around*. Abgerufen am 31. August 2017 von <https://www.devexpress.com/Support/Center/Question/Details/T372420/search-panel-for-detail-views-work-around>
- Gebhart, G. (2013). *Praktisches & Grundsätzliches zur Informatik*. Abgerufen am 31. August 2017 von <http://greiterweb.de/spw/Wahl-einer-Programmiersprache.htm>
- Jorgensen, A. (2012). Microsoft SQL Server 2012 Bible. John Wiley & Sons.
- Marugg, L. (03. 04 2010). PG\_Info\_Hardware. VAT Interne Präsentation. Haag.
- National Instruments. (01 2002). Abgerufen am 31. August 2017 von LabWindows/CVI SQL Toolkit Reference Manual: <http://www.ni.com/pdf/manuals/370502a.pdf>
- National Instruments. (08 2012). Abgerufen am 31. August 2017 von Network Variables: <http://zone.ni.com/reference/en-XX/help/370051V-01/cvi/libref/cvinetworkvariables/>
- Pavlov, P. (März 2014). *Scrum vs Wasserfall - Ein Vergleich und Erfahrungen aus der Praxis*. Abgerufen am 31. August 2017 von <https://de.slideshare.net/axxessio/scrum-vs-wasserfall>
- QMETHODS. (12. Dezember 2005). *Glossar Qualitätssicherung & Testmanagement*. Abgerufen am 31. August 2017 von [https://www.qmethods.de/glossar\\_qualitaetssicherung-test.html](https://www.qmethods.de/glossar_qualitaetssicherung-test.html)
- The Open Group. (2016). Abgerufen am 31. August 2017 von putenv: <http://pubs.opengroup.org/onlinepubs/9699919799/functions/putenv.html>
- The Open Group. (2016). Abgerufen am 31. August 2017 von getenv: <http://pubs.opengroup.org/onlinepubs/9699919799/functions/getenv.html>
- VAT Group AG. (2017). Abgerufen am 31. August 2017 von <http://www.vatvalve.com/de/business/industry>

### Abkürzungsverzeichnis

NI	National Instruments
WPF	Windows Presentation Foundation
DLL	Dynamic Link Library

## Abbildungsverzeichnis

Abbildung 1: Basiskonzept Ventil Controller (Marugg, 2010).....	2
Abbildung 2: Parameterbaumstruktur der Software.....	3
Abbildung 3: Ist-Zustand der Testumgebung.....	4
Abbildung 4: Ansicht der TTIC2 Oberfläche für die Auswahl der Testkollektion.....	8
Abbildung 5: Ursprüngliches Konzept Masterarbeit.....	11
Abbildung 6: Konzept Masterarbeit Testumgebung komplett .....	12
Abbildung 7: Wasserfallmodell (QMETHODS, 2017).....	14
Abbildung 8: Vorgehensweise Testumgebung.....	15
Abbildung 9 Aktuelle SoftwareVersionsDatabase Struktur.....	20
Abbildung 10: Überarbeitete SoftwareVersionsDatabase Struktur.....	21
Abbildung 11: Erweiterung SoftwareVersionsDatabase zur Speicherung der Testinformationen .....	22
Abbildung 12: Erweiterung SoftwareVersionsDatabase zur Speicherung der Testresulat .....	24
Abbildung 13: Erweiterung SoftwareVersionsDatabase zur Speicherung der Firmware Bugs .....	25
Abbildung 14: Erster Ansatz Abspeicherung Testinformationen.....	26
Abbildung 15: Verbessertes Konzept zur Abspeicherung der Testinformationen .....	28
Abbildung 16: TTIC2 Zugriff auf SoftwareVersionsDatabase .....	30
Abbildung 17: Fortschrittbalken Auslesung Testversionen .....	31
Abbildung 18: Definition Grundzustand TTIC2 .....	31
Abbildung 19: ETIC2 Konzept Ansicht Testresultate.....	34
Abbildung 20: DevExpress Master-Detail via EntityFramework Element .....	34
Abbildung 21: Model ETIC2 in UML .....	35
Abbildung 22 ViewModel ETIC2 in UML .....	35
Abbildung 23: View ETIC2 .....	36
Abbildung 24: Modellierung InitialStateFirmware der SoftwareVersionsDatabase .....	39
Abbildung 25: Aufbau ETIC2.....	41
Abbildung 26: Error Ansicht des ETIC2.....	42
Abbildung 27: Auswertung der letzten ausgeführten Testkollektion .....	42
Abbildung 28: Ansicht Testresultate im PDF .....	43
Abbildung 29: Firmware Loader .....	44
Abbildung 30: Report Ansicht währendem die Tests ausgeführt werden .....	52
Abbildung 31: Konzept Ansicht Buglist .....	53
Abbildung 32: Funktionsumfang des SQL Toolkits (National Instruments, 2002) .....	55
Abbildung 33: Code Ausschnitt um einen neuen Eintrag zu erstellen .....	56

## **Tabellenverzeichnis**

Tabelle 1: Phasen des Wasserfallmodells in der Dokumentation.....	14
Tabelle 2: Nutzwertanalyse Datenbanklösung .....	16
Tabelle 3: Nutzwertanalyse Zugriff auf SoftwareVersionsDatabase .....	17
Tabelle 4: Lösungsvorschläge Suche nach Fehlermeldung in ETIC2.....	37
Tabelle 5: Vergleich Umgebungsvariable .....	54

## Glossar

Antriebsfile	Enthält alle Ventilhardware spezifischen Abweichungen gegenüber den Standard Einstellungen, welche in der Firmware hinterlegt sind.
CVI	Ist eine ereignisorientierte Programmiersprache, welche auf C basiert und von National Instruments entwickelt wurde..
DevExpress	Käuflich erworbene Bibliothek für die Verwendung von WPF Elementen
Diagnostik File	Enthält alle Ventilparameter mit ihren aktuellen Werten. Zur genaueren Auswertung eines Fehlers.
Enum	Ist ein Aufzählungstyp mit einer endlichen Wertemenge. Die zulässigen Werte werden mit einem eindeutigen Namen definiert.
ERP	Enterprise-Resource-Planing: Unternehmerische Software mit deren Hilfe Ressourcen wie Kapital, Personal rechtzeitig und bedarfsgerecht geplant und gesteuert werden kann.
ETIC2	Evaluation Tool Integrierter Controller 2: Auswertungsoberfläche für die Testkollektionen. Integrierter Controller werden Controller genannt, welche direkt mit der Ventil Hardware verbunden sind.
Grundzustand	Der Grundzustand setzt sich aus den Angaben der Ventil Firmware, der Motion Controller Firmware sowie optional der Interface Firmware, des Antriebsfiles sowie Konfigurationsfiles zusammen. Jeder Grundzustand erhält einen eindeutigen Namen.
IC	Integrierter Controller: Der Controller befindet sich direkt beim Vakuumventil.
IC2	Integrierter Controller der zweiten Generation. Neueste Generation der IC Generation, welche mit den Tests qualifiziert wird.
Konfigurationsfile	Enthält alle Abweichungen der Firmware gegenüber den Standard Ventil Firmware Einstellungen, welche in der Firmware hinterlegt sind.
MVVM	Mode View ViewModel: Ist ein Entwurfsmuster, welches eingesetzt wird um Darstellung und Logik der Benutzerschnittstelle zu trennen.
MySQL Workbench	Wird als Datenbank-Modellierungswerkzeug eingesetzt. Besitzt eine grafische Oberfläche für die Modellierung der Datenbanktabellen.
.NET	Sammelbegriff für mehrere von Microsoft entwickelte Software-Plattformen.
PXI	Ist ein Produkt von NI, wobei es sich um eine PC-basierte Plattform handelt, welches für Automatisierungs- und Messsysteme eingesetzt wird.
SVN	Apache Subversion: Freie Software zur Versionsverwaltung.
trunk	Aktuellste Firmware Stand, welcher noch nicht freigegeben wurde.
TTIC2	Test Tool Integrierter Controller 2: Testoberfläche für alle integrierten Ventilcontroller der 2. Generation
Zykluszeit	Da die eigens entwickelte VAT Firmware kein Betriebssystem verfügt, wird nach Ablauf der Zykluszeit alle äusseren Einflüsse wie z.B. Sensorwerte ausgelesen und auf diese entsprechend reagiert.

## A. Projektmanagement

### A.1. Zeitplan

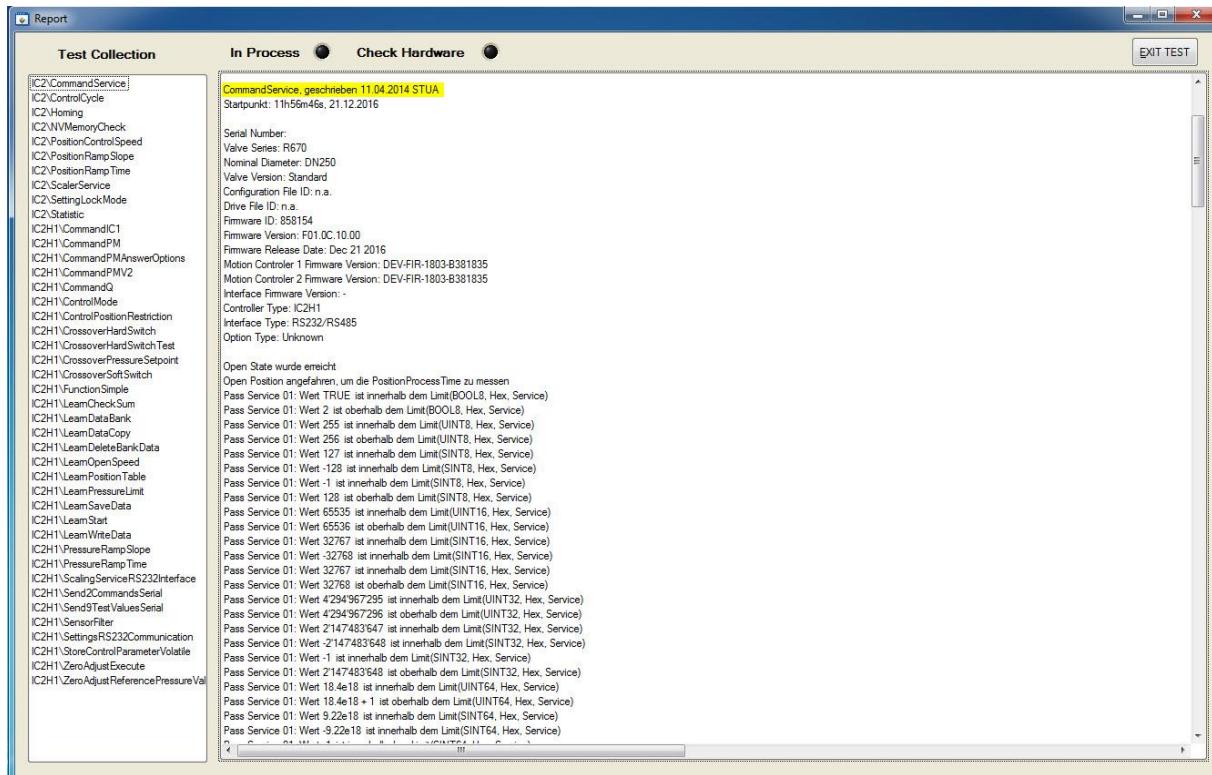
Projektplan	Bemerkungen	Start	Ende	Aufwand [h] (Plan)	Aufwand [h] (effektiv)
Einreichen der Disposition		28.02.2017	28.02.2017		
Start Masterarbeit		01.05.2017	01.05.2017		
<b>SoftwareVersionsDatabase</b>		02.05.2017	18.05.2017	15	31
➤ Modellierung im MySQL Workbench		02.05.2017	17.05.2017	13	21
➤ Scripts erstellen		17.05.2017	18.05.2017	2	10
Firmware Database (Anpassung an neue Datenbankmodellierung)	Abbruch	08.05.2017	08.05.201		
<b>Test</b>		17.05.2017	21.06.2017	*	41.5
➤ Testinformationen in SoftwareVersionsDatabase schreiben	Erster Ansatz	17.05.2017	31.05.2017	*	27.5
➤ Testinformationen an TestUpdateFirmware über Umgebungsvariablen	Aktueller Ansatz	15.06.2017	21.06.2017	*	14
<b>TestUpdateFirmware</b>		18.05.2017	20.06.2017	*	22
➤ Anweisung Update Testinformationen	Erster Ansatz	18.05.2017	18.05.2017	*	3
➤ Abfrage Testinformationen	Aktueller Ansatz	16.06.2017	17.06.2017	*	9
➤ Testinformationen in SoftwareVersionsDatabase schreiben	Aktueller Ansatz	17.06.2017	20.06.2017	*	10
<b>TTIC2</b>		23.06.2017	29.07.2017	30	114
➤ Auslesung der Testinformationen		23.06.2017	30.06.2017	*	26
➤ Hinterlegung des Grundzustandes		01.07.2017	14.07.2017	*	47
➤ Abspeicherung der Testresultate		15.07.2017	29.07.2017	30	41
<b>ETIC2</b>		28.07.2017	28.08.2017	155	117.5
➤ Codierung nach MVVM		28.07.2017	28.08.2017	110	108
➤ Anbindung SoftwareVersionsDatabase		06.08.2017	06.08.2017	5	3
➤ Ausgabe Bericht		21.08.2017	22.08.2017	20	6
➤ Reserve		31.07.2017	13.08.2017	20	-
Schriftliche Arbeit		01.05.2017	30.07.2017	80	53.5
Fertigstellen der schriftlichen Arbeit		14.08.2017	31.08.2017	40	76
Abgabe der Masterarbeit		31.08.2017			
<b>Total Projektarbeit</b>		28.02.2017	31.08.2017	<b>320</b>	<b>461.5</b>

\* Wurde nicht geplant, nach erster Analyse als nötig betrachtet

## B. Testumgebung

### B.1. TTIC2

In der unteren Abbildung wird das Reportfenster gezeigt, welche im Kapitel 2.2.6.1 beschrieben wird.

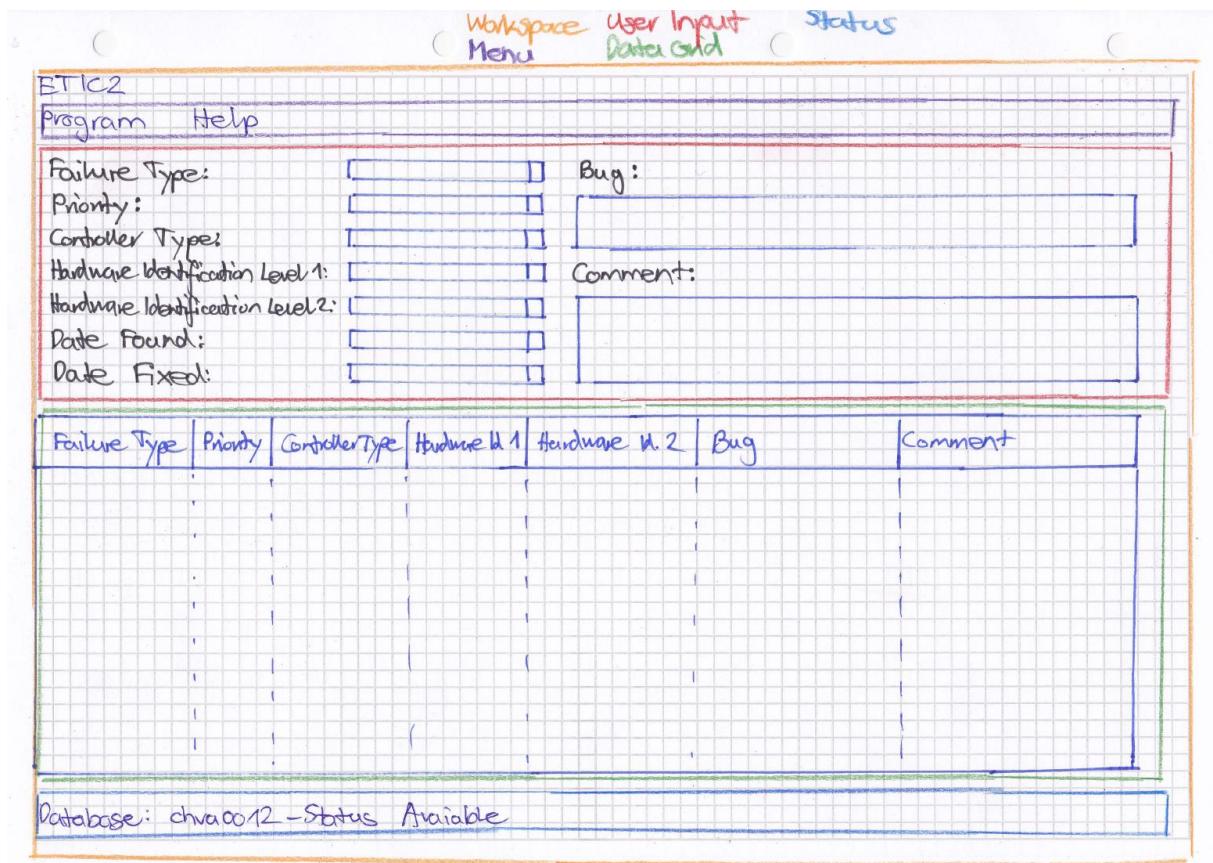


**Abbildung 30: Report Ansicht währendem die Tests ausgeführt werden**

Auf der linken Seite werden alle Test der Testkollektion aufgelistet. Im Hauptfenster werden die einzelnen Testschritte aufgelistet. Mit dem Knopf Exit Test kann der Benutzer die Ausführung der Testkollektion abbrechen, welcher sich rechts oben befindet.

## B.2. ETIC2

Die Abbildung 31 zeigt die Gestaltung der Buglist Ansicht.



**Abbildung 31: Konzept Ansicht Buglist**

Die Ansicht ist in fünf Module (Workspace, Menu, User Input, DataGridView, Status) unterteilt. Dabei sind die Module Workspace, Menu und Status bereits im bestehenden ETIC2 erstellt worden. Das UserInput zeigt den ausgewählten Eintrag aus dem DataGridView. Hier werden alle Einträge der Buglist angezeigt um diese bearbeiten zu können. Im DataGridView Modul werden alle Einträge der Buglist angezeigt welche über ein Suchfeld sortiert werden können.

## C. Hilfsmittel

### C.1. CVI

#### C.1.1. Umgebungsvariable

Im nächsten Abschnitt soll die Umgebungsvariable, welche von NI zur Verfügung gestellt wird, näher vorgestellt werden. Die Umgebungsvariable wird gebraucht um Daten zwischen mehreren Applikationen auf dem gleichen wie auch auf unterschiedlichen Rechnern zu teilen. Wenn sich die Applikationen auf unterschiedlichen Rechnern befindet, werden die Daten über TCP miteinander ausgetauscht. Die Umgebungsvariablen können direkt oder indirekt erzeugt werden. Direkt können die Variablen über das Distributed System Manager Programm erzeugt werden wie auch programmatisch in der CVI Programmierumgebung. Die Variablen können einzeln wie auch mehrere in einem Cluster definiert werden. Es kann definiert werden ob auf die Variable geschrieben oder ob sie nur gelesen werden kann oder beides wie auch deren Datentyp. Es kann auch bestimmt werden, wann die Umgebungsvariable beschrieben werden soll, immer sofort oder erst ab einer bestimmten Datenmenge. Das Distributed System Manager Programm kann bei der indirekten Variante gebraucht werden, um die aktuellen Werte auszulesen oder auch einen Wert zu manipulieren, was beim Debuggen einige Zeit ersparen kann. (Community, 2016)

#### C.1.1.1. Gegenüberstellung zu C Library Umgebungsvariable

Die C Library bietet die Möglichkeit eine Umgebungsvariable vom Typ String zu benutzen. Unter Angabe des Namens der Umgebungsvariable und dessen Wert wird dieser unter dem System abgelegt (The Open Group, 2016). Dieser kann unter Angabe des Namens der Umgebungsvariable ausgelesen werden. (The Open Group, 2016) Dabei kann die erzeugte Umgebungsvariable nicht mehr überschrieben werden.

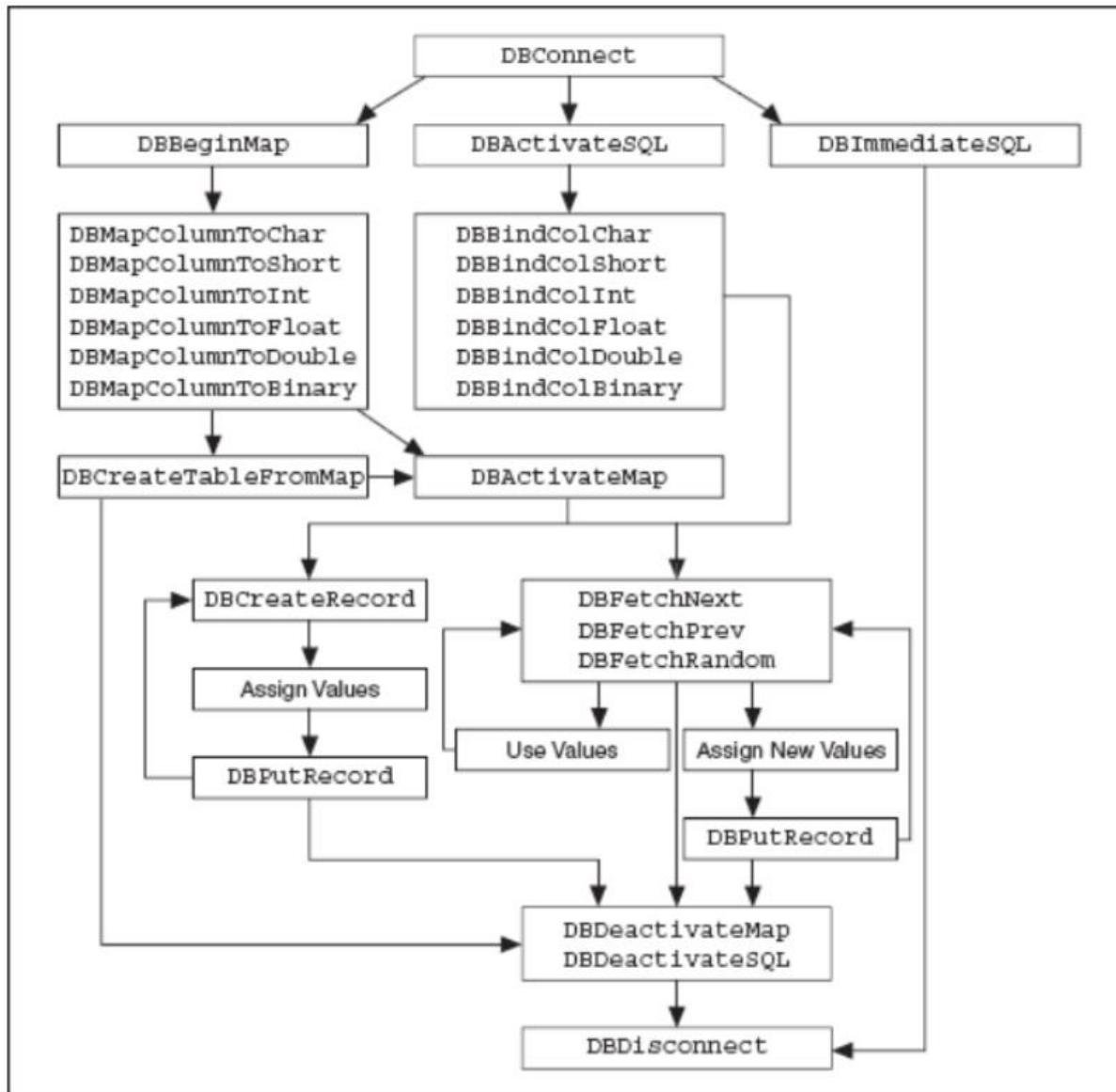
Kriterium	Umgebungsvariable	
	CVI	C Library
Lese Zugriff	immer	immer
Schreib Zugriff	überschreibbar	einmalig im Betrieb
Datentyp	gebräuchlichsten	String
Cluster zusammenfassen	ja	nein
Austausch über unterschiedlichen Rechnern	ja	nein
Initialisierung notwendig	ja	direkt mit Wert versehbar

Tabelle 5: Vergleich Umgebungsvariable

## C.2. SQL Toolkit

### C.2.1. Funktionsumfang

Das Ziel des nächsten Abschnittes ist es den Funktionsumfang des SQL Toolkits vorzustellen. Die Abbildung 32 zeigt dabei die einzelnen Funktionen auf.



**Abbildung 32: Funktionsumfang des SQL Toolkits (National Instruments, 2002)**

Als erster Schritt wird über die **DBConnect** Funktion eine Verbindung zur Datenbank aufgebaut. Diese wird anschliessend gebraucht, um entweder über Automatic SQL, Explicit SQL oder Immediate SQL Anweisungen auszuführen um Daten auszulesen, zu verändern oder zu erstellen, welche in der Datenbank hinterlegt sind.

## Auflistung Vor- und Nachteile:

Die Automatic SQL Variante ermöglicht einfache Select Abfragen wie auch Create Table Befehle. Dabei wird die Anweisung von der SQL Toolkit übernommen.

Bei der Explicit SQL Variante muss die SQL Anweisung vom Programmierer geschrieben werden. Aus diesem Grund können komplexere Select wie auch andere Arten von Anweisungen ausgeführt werden.,

Mit der Immediate SQL Möglichkeit entfallen die DBActiveSQL und DBDeactiveSQL Funktionsaufrufe. Diese wird eingesetzt, wenn nach der Anweisung, keine Daten mehr vorhanden sein müssen.

Bei den ersten beiden Varianten kann die Tabelleninformation in unterschiedliche Datentypen ausgelernt werden und liegt dem Programm gleich zur Weiterverarbeitung bereit. Es ist nötig vor der Ausführung einer SQL Anweisung zuerst System Ressourcen anzulegen, welche nach der Ausführung wieder freigegeben werden. Hierbei handelt es sich um temporäre Files. (National Instruments, 2002)

### C.2.2 Code Ausschnitt um einen neuen Eintrag zu erstellen

```
/*
 **** Procedure ****
 // Description : Definiert den Controller Typ des Tests
 // Wenn der Type noch nicht definiert ist, wird einer erstellt oder angepasst
 // Input : TestVersionId;
 //          Id der Test Version
 // Output *Pointer : -
 //          ControllerTypeTest; Anforderung zum Controller
 // return : Ob ein Fehler aufgetreten ist
 ****
BOOL8 Database_WriteControllerTypeTestInDatabase(SINT32 TestVersionId, struct ControllerType ControllerTypeTest)
{
    SINT32 ControllerId, TestVersionIdDatabase, ControllerTypeIdDatabase;
    long TestVersionIdStat, ControllerTypeIdStat;

    ConnectDatabase();

    //Schauen ob es den Controller Type Eintrag schon gibt, wenn nicht erstellen
    ControllerId = WriteControllerTypesTable(ControllerTypeTest);

    //Controller Type mit TestVersion verknüpfen (wenn nicht vorhanden)
    hstmt = DBActivateSQL (hdbc, "SELECT * FROM ControllerType");
    if (hstmt > 0)
    {
        resCode = DBBindColInt(hstmt, 2, &TestVersionIdDatabase, &TestVersionIdStat);
        resCode = DBBindColInt(hstmt, 3, &ControllerTypeIdDatabase, &ControllerTypeIdStat);
        while ((resCode = DBFetchNext (hstmt)) == DB_SUCCESS)
        {
            if (TestVersionIdDatabase == TestVersionId && ControllerTypeIdDatabase == ControllerId)
            {
                //Eintrag schon vorhanden -> Abbrechen
                resCode = DBDeactivateSQL (hstmt);
                DisconnectDatabase();
                return ErrorState;
            }
        }
        //keinen Verlinkung -> Eintrag erstellen
        resCode = DBDeactivateSQL (hstmt);
        hmap = DBBeginMap (hdbc);
        resCode = DBMapColumnToInt (hmap, "TestVersion_Id", &TestVersionIdDatabase, &TestVersionIdStat);
        resCode = DBMapColumnToInt (hmap, "ControllerTypes_Id", &ControllerTypeIdDatabase, &ControllerTypeIdStat);
        hstmt = DBActivateMap (hmap, "ControllerType");
        resCode = DBCreateRecord (hstmt);
        TestVersionIdDatabase = TestVersionId;
        ControllerTypeIdDatabase = ControllerId;
        resCode = DBPutRecord (hstmt);
        resCode = DBDeactivateMap (hmap);
    }

    DisconnectDatabase();
    return ErrorState;
}
```

Abbildung 33: Code Ausschnitt um einen neuen Eintrag zu erstellen

Im Code-Ausschnitt (Abbildung 33) wurde auf die Fehlerüberprüfung verzichtet, um eine bessere Übersicht dem Leser zu geben. Der Verbindungsauflauf sowie Abbau zur Datenbank wird nur als Funktionsaufruf dargestellt. Die Aufgabe der Funktion Database\_WriteControllerTypeTestInDatabase besteht darin zu prüfen, ob der Controller Type schon mit der Test Version verlinkt wurde. Die Verlinkungstabelle ist nötig, um mehrere Controller Typen einer Test Version zuweisen zu können. Die unterschiedlichen Controller Typen sind in einer eigenen Tabelle (ControllerTypes) definiert. Bei der Verlinkung wird dabei eine Referenz des Controller Typs angegeben.

Nach Aufbau der Verbindung wird mit dem Controller Type Eingangsstruktur, der Controller Type definiert, wenn dieser noch nicht existiert. Die Funktion liefert die ID des Controller Types zurück. Nun wird in der Verlinkungstabelle geprüft, ob die Verbindung schon vorhanden ist. Dazu wird die DBActivateSQL Funktion verwendet, um einen Select Befehl auszuführen. Mit den DBBindCol Funktionen können Tabellenwerte im entsprechenden Datenformat direkt ausgelesen werden. Mit der while Befehl DBFetchNext werden die einzelnen Tupel der ControllerType Tabelle ausgelesen. Mit der if Anweisung wird kontrolliert, ob die Verlinkung schon eingetragen ist, wenn ja wird die while Bedingung abgebrochen und die Funktion verlassen. Ist hingegen kein Eintrag gefunden worden, werden die Expliziten Ressourcen wieder geschlossen und ein neues Mapping mit DBBeginMap Funktion erstellt. Analog mit DBMapColumnTo können Attribute der Tabelle in der Automatic Variante direkt gebunden werden. Mit der DBActiveMap Funktion wird die zu editierende Tabelle angegeben. Die DBCreateRecord erzeugt einen neuen Eintrag, in welchem anschliessend die gebundenen Werte gesetzt werden, bevor mit dem DBPutRecord Anweisung das Tupel in der Datenbank abgespeichert wird. Am Ende werden mit dem DBDeactivateMap Anweisung die erzeugten Automatic Ressourcen geschlossen.

## **Selbständigkeitserklärung**

Mit der Abgabe dieser Abschlussarbeit versichert der/die Studierende, dass er/sie die Arbeit selbständig und ohne fremde Hilfe verfasst hat (Bei Teamarbeiten gelten die Leistungen der übrigen Teammitglieder nicht als fremde Hilfe):

Der/die unterzeichnende Studierende erklärt, dass alle zitierten Quellen (auch Internetseiten) im Text oder Anhang korrekt nachgewiesen sind, d.h. dass die Abschlussarbeit keine Plagiate enthält, also keine Teile, die teilweise oder vollständig aus einem fremdem Text oder einer fremden Arbeit unter Vorgabe der eigenen Urheberschaft bzw. ohne Quellenangabe übernommen worden sind.

Ort, Datum: .....

Unterschrift Studierende/r: .....