

**Management Summary**

* Problemstellung
* Methodik/Ablauf
* Resultat

Inhalt

[ETIC2 zur Verwaltung von Ventiltests 5](#_Toc485233886)

[1 Einleitung (1 Seite) 5](#_Toc485233887)

[2 Grundlagen 6](#_Toc485233888)

[2.1 Situation 6](#_Toc485233889)

[2.2 Testumgebung (Ist-Zustand) 7](#_Toc485233890)

[2.2.1 Ventil Hardware 7](#_Toc485233891)

[2.2.2 Test Hardware (NI PXI) 7](#_Toc485233892)

[2.2.3 Test 8](#_Toc485233893)

[2.2.4 ParameterStructBuild 9](#_Toc485233894)

[2.2.5 TestsUpdateFirmware 9](#_Toc485233895)

[2.2.6 TTIC2 Test Tool 10](#_Toc485233896)

[2.2.7 SoftwareVersionsDatabase 12](#_Toc485233897)

[2.2.8 Firmware Database 12](#_Toc485233898)

[2.3 Ausgangslage (Problemstellung) 12](#_Toc485233899)

[3 Zielsetzung 15](#_Toc485233900)

[3.1 Quantitative Ziele 15](#_Toc485233901)

[3.2 Qualitative Ziele 16](#_Toc485233902)

[3.3 Aufgabenbegrenzung 16](#_Toc485233903)

[4 Methodik (Was? + Theorie + Konzept) 17](#_Toc485233904)

[4.1 Modellierung der SoftwareVersionsDatabase 17](#_Toc485233905)

[4.2 Abspeicherung der Software Informationen 17](#_Toc485233906)

[4.3 Abspeicherung der Testinformationen 17](#_Toc485233907)

[4.4 Abspeicherung der Testresultate 17](#_Toc485233908)

[4.5 Abspeicherung von Firmware Bugs 18](#_Toc485233909)

[4.6 Zugriff auf SoftwareVersionsDatabase 18](#_Toc485233910)

[4.7 Erstellung ETIC2 18](#_Toc485233911)

[4.8 Design View Model 18](#_Toc485233912)

[4.9 Codierung nach MVVM 18](#_Toc485233913)

[4.10 Anbindung SoftwareVersionsDatabase 18](#_Toc485233914)

[4.11 Ausgabe Bericht 18](#_Toc485233915)

[5 Beschreibung der Arbeit (Praxis, Umsetzung) 19](#_Toc485233916)

[5.1 Modellierung der SoftwareVersionsDatabase 19](#_Toc485233917)

[5.2 Abspeicherung der Firmware Informationen 19](#_Toc485233918)

[5.3 Abspeicherung der Testinformationen 20](#_Toc485233919)

[5.4 Abspeicherung der Testresultate 21](#_Toc485233920)

[5.5 Abspeicherung von Firmware Bugs 22](#_Toc485233921)

[5.6 Zugriff auf SoftwareVersionsDatabase 22](#_Toc485233922)

[5.7 Erstellung ETIC2 23](#_Toc485233923)

[5.8 Design View Model 23](#_Toc485233924)

[5.9 Codierung nach MVVM 23](#_Toc485233925)

[5.10 Anbindung SoftwareVersionsDatabase 23](#_Toc485233926)

[5.11 Ausgabe Bericht 23](#_Toc485233927)

[6 Ergebnisse (Tool) 24](#_Toc485233928)

[7 Diskussion (Was hat Funktioniert, was nicht) 25](#_Toc485233929)

[8 Ausblick (offene Punkte, wie geht es weiter) 26](#_Toc485233930)

[8.1 Umsetzung Überarbeitung SoftwareVersionsDatabase 26](#_Toc485233931)

[8.2 Integration Buglist in ETIC2 26](#_Toc485233932)

[Verzeichnisse 27](#_Toc485233933)

[Literaturverzeichnis 27](#_Toc485233934)

[Abkürzungsverzeichnis 27](#_Toc485233935)

[Abbildungsverzeichnis 28](#_Toc485233936)

[Tabellenverzeichnis 28](#_Toc485233937)

[Glossar 29](#_Toc485233938)

[Anhang 30](#_Toc485233939)

[Zeitplan 30](#_Toc485233940)

[Selbständigkeitserklärung 31](#_Toc485233941)

# ETIC2 zur Verwaltung von Ventiltests

# Einleitung (1 Seite)

* Problemstellung
* Vorgehen

# Grundlagen

## Situation

Die Firma VAT stellt Vakuumventile her für die Halbleiter- und Medizinalindustrie, die Forschung und Entwicklung sowie für die Automobilindustrie (VAT Group AG, 2017). VAT ist im Bereich der Herstellung von Vakuumventilen und mit einem Marktanteil von über 40% klarer Weltmarktführer. Die Firma ist bekannt für ihre Regelventile, bei welchem ein Controller die Steuerung dieser Vakuumventile übernimmt (siehe Abb. 1). Dieser ist modular aufgebaut und besteht grob gesagt aus drei Komponenten. Die wichtigste Komponente ist das Masterboard, die mir den zentralen Elementen bestückt ist. Dieses ist unerlässlich, wird aber jeweils an die gewünschte Ventilhardware angepasst. Das Herzstück des Controllers ist der Mikrocontroller, für den VAT eine eigene Firmware entwickelt hat. 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 Feldbus System arbeitet, so wird eine Interface Firmware nötig. Der Nutzen von Feldbus Systemen liegt darin, dass von einem Host aus mehrere Teilnehmer angesprochen werden können. Die dritte Komponente ist die Option Unit, die Zusatzfunktionen nach Wunsch beinhalten. Der Kunde kann mit Hilfe des Controllers seine Sensoren direkt speisen. Eine weitere Option ist, dass bei Spannungsabfall das Ventil an eine vordefinierte Position einnimmt. Verwendet der Kunde kein Feldbus System, so kann er mit Hilfe eines Cluster Systems mehrere Ventile ansprechen. (Marugg, 2010).



Abbildung : Basiskonzept Ventil Controller (Marugg, 2010)

## Testumgebung (Ist-Zustand)

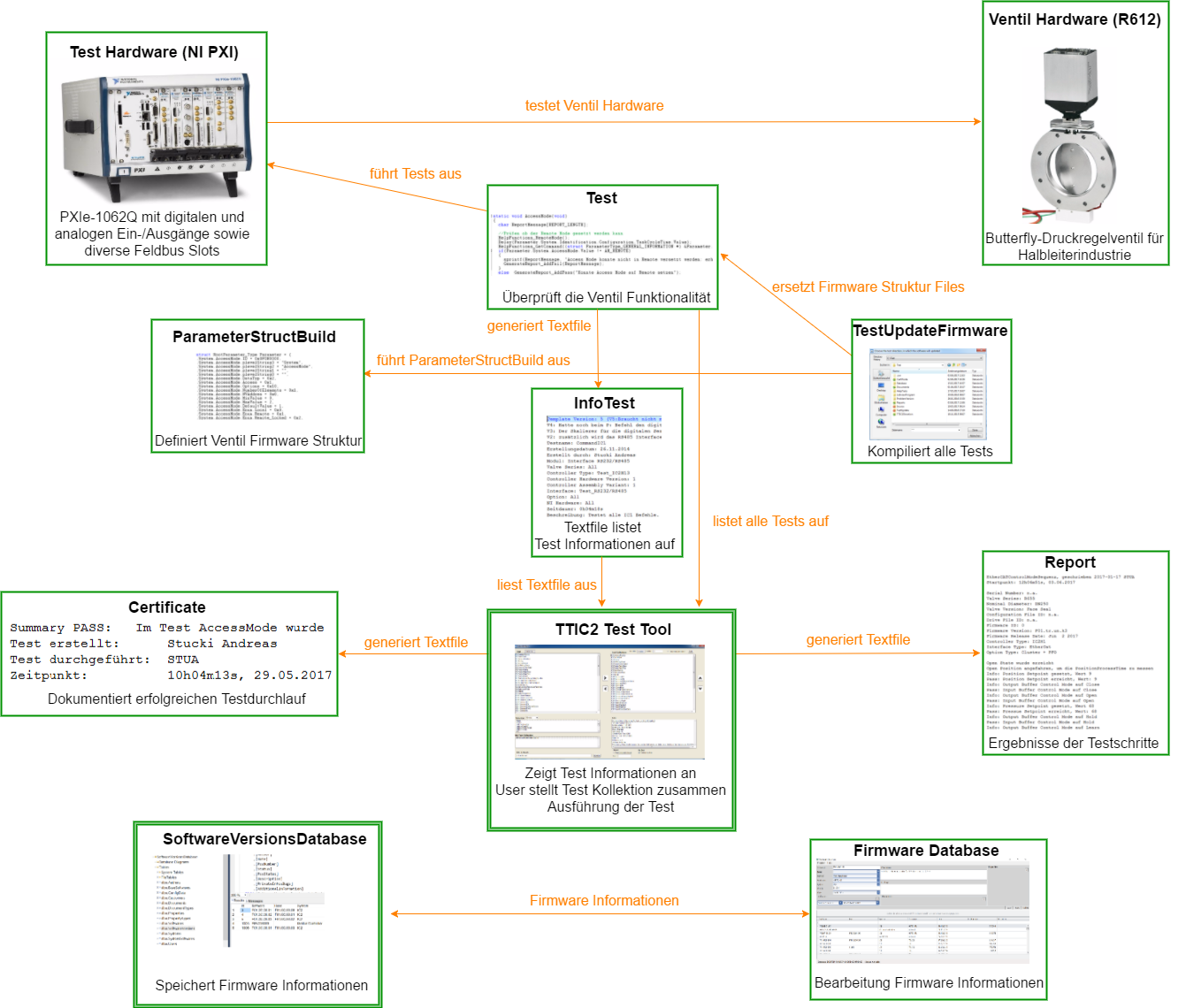
In diesem Unterkapitel wird die gesamte Testumgebung kurz erklärt. Das primäre Ziel liegt in der automatischen und wieder Verwendbarkeit von einzelnen Tests. Weiter soll der Wartungsaufwand klein gehalten werden um verschiedene Ventilhardware mit der gleichen Testumgebung qualifizieren zu können. Ein weiteres zentrales Element bildet die Dokumentation der Testschritte und deren Ergebnisse.

Abbildung : Ist-Zustand der Testumgebung

### Ventil Hardware

Mit Hilfe der Testumgebung werden unterschiedliche Ventil Hardware getestet. Der Controller wird der jeweiligen Ventil Hardware Anforderungen angepasst. Das bedeutet, dass es unterschiedliche Controller Ausführungen gibt. Der Grundgedanke am Aufbau des Controllers bleibt jedoch gleich wie auch die Struktur der Ventil Firmware. Der Aufbau und die Funktionsweise des Ventils liefern unterschiedliche Anforderungen an die Bestückung des Motors in Bezug auf deren Anzahl und Leistung.

### Test Hardware (NI PXI)

Der Controller hat mehrere Schnittstellen zum Hostsystem des Vakuumkammer Betreibers. Einerseits hat jeder Controller eine USB Stecker, welches 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. Immer häufiger spricht der Host seine Komponenten mit Hilfe einer Feldbus Lösung an. Somit können mehrere Teilnehmer im System miteinander kommunizieren. Historisch bedingt gibt es auch die Lösungen dies mit einer internen entwickelten Cluster Option des Ventils oder über die RS485 Interface Schnittstelle.

Für den Low-end Markt wird der Controller mit einer Logik Interface ausgerüstet. Der Host steuert das Ventil dabei mit Hilfe von Analogen und Digitalen Signalen. Dabei werden auch die Status Informationen des Ventils mit Hilfe von Analogen und Digitalen Signalen dem Host zurück geliefert.

Das Ventil wie auch der Controller betreiben zu können, braucht es einen Power Stecker. Dieser Stecker bietet zusätzlich eine Safety Möglichkeit sowie kann das Ventil hardwaremässig geöffnet und geschlossen werden.

All diese Funktionalitäten muss mit Hilfe der Test Hardware emuliert und geprüft werden können. Dabei bietet National Instruments eine flexible Kartenbestückungslösung 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 sie sehr viele Hilfsfunktionalitäten zur Verfügung stellen. Dabei gibt es zwei Einschränkungen bezüglich der Speisung und der Unterstützung der CCLink Interface Anbindung. Die Speisung wird mit Hilfe einem separaten Speisegerät per Software (LabVIEW) gesteuert. Die Kommunikation über CCLink mit dem Controller kann über einen Umweg eines Drittanbieters Produktes erreicht werden.

#### Stärken

* Schnelle Inbetriebnahme der Hardware über die NI Programmierumgebung
* Fast alle Hardware Schnittstellen können über die NI Hardware getestet werden
* Das System kann einfach erweitert werden, bis alle 8 Steckplätze besetzt sind
* Engineering Support durch NI

#### Schwächen

* Jährliche Lizenzkosten für die Programmierumgebung
* Relativ hohe Anschaffungskosten

### Test

Aus persönlichen Programmiererfahrung im textbasierten Umfeld wurde die CVI Programmierumgebung ausgewählt um die Test Hardware ansprechen zu können. Bei den geschriebenen Tests handelt es sich um automatisierte Anwendertests. Dabei 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, was zur Folge hat, das der Anwender nicht mehr aktiv in die Ausführung eingreifen muss. Die Tests werden bei der zugehörigen Ventil Firmware im Subversion hinterlegt, umso bei allfälligen Fehler 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.

Die Ventil Firmware basiert auf einem Parameterbaum Protokoll, welches über vier Hierarchiestufen geht. Dabei kann grob gesagt werden, dass jeder Parameter eine Einstellung des Systems meldet, ändert oder eine Ventil Funktion auslösen kann. Diese Parameter werden durch einen Befehl über die Service (USB) oder dem Interface Kanal verändert und abfragen. Es gibt Hilfsbefehle, welche es erlauben, alle Parameter Identifikationsnummer sowie den Namen, den Datentyp, der erlaubte Wertebereich sowie seinen Zugriff auszulesen. Im Test werden mit den Parameternamen gearbeitet, um dem Code einfacher lesen zu können, sowie kann die Identifikationsnummer sich ändern, ohne eine Anpassung des Tests notwendig wird.

#### Stärken

* Voll automatische Tests
* Testfunktionen nur in einem File, der Rest sind Libary Files
* In einer Struktur sind alle Ventil Parameter hinterlegt sowie seine Enum Werte

#### Schwächen

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

### ParameterStructBuild

Dieses Programm liest die aktuellen Parameter der Software aus und speichert diese in Struktur Form ab. Diese Mehrdimensionale Struktur spiegelt den Parameterbaum nach. Wichtige Elemente bilden dabei die Identifikationsnummer, welche aus einer achtstelligen Nummer besteht. Dabei werden jeweils zwei Stellen für eine Ebene verwendet. Weiter werden die einzelnen Ebenen in Textform definiert. Sowie den Datentyp, den Zugriff wie auch den Wertebereich des Parameters wird ausgelesen. Dazu enthalten einige Parameter Enum Werte, welche den Parameterwert in Textform beschreibt. Diese wird auch in dieser Struktur Form hinterlegt. Die CVI Programmierumgebung basiert auf der Programmiersprache C. Die Definition der Struktur wird in einem Header File abgespeichert. Die dazugehörigen Werten werden in einem Source File abgespeichert. Diese Informationen werden später in den Tests genutzt, sodass der Test Schreiber sieht, welche Parameter von der Firmware unterstützt wird. Weiter muss er keine Kenntnisse über die Identifikationsnummer haben. Zudem erlauben die Enum Werte eine Auflistung aller zulässigen Parameterwerte und eine einfachere Lesbarkeit.

#### Stärken

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

#### Schwächen

* Es braucht eine aktive Verbindung über USB mit dem Controller

### TestsUpdateFirmware

Das TestUpdateFirmware ruft zuerst das ParameterStructBuild auf um die Parameter Struktur File erstellen zu lassen. Anschliessend wird das alte Parameter Struktur File mit dem neuen ersetzt. Dieses File ist für alle Tests sichtbar, da alle Linary Files wie auch die Parameter Struktur Files in einem gemeinsamen Ordner liegen. Der Benutzer kann jetzt angeben, in welchem Testordner alle Tests neu kompiliert werden soll mit den neuen Parameter Informationen. Weiter gibt der Benutzer auch den Ordner an, an welchem das TTIC2 Programm hinterlegt ist, um auch hier die neusten Parameter Strukturen im Programm zu verwenden.

#### Stärken

* Die Parameter Files werden automatisch ersetzt
* Es werden automatisch alle Tests mir den neuesten Parameter Informationen kompiliert
* Auflistung über Tests, welche nicht ohne Fehler kompiliert werden konnten

#### Schwächen

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

### TTIC2 Test Tool

In den nächsten drei Abschnitten werden die Spezifikationen des TTIC2 aufgezeigt.

**Vor der Ausführung**

* Mit Angabe des Pfades werden alle Tests aufgelistet, welche mit der aktuellen Controller Generation lauffähig sind
* Aus dieser Auflistung können die gewünschten Tests ausgewählt werden
* Es gibt verschiedene Filtermöglichkeiten, welche die Testauswahl anpasst (Mehrfachfilter möglich)
* Die aktuelle Testkollektion kann abgespeichert werden oder eine zuvor gespeicherte kann geladen werden
* Weiter können spezifische Testeinstellungen (soll der Test bei einem Fehler abgebrochen werden, wie viele Informationen soll der Report liefern) vor der Ausführung definiert werden

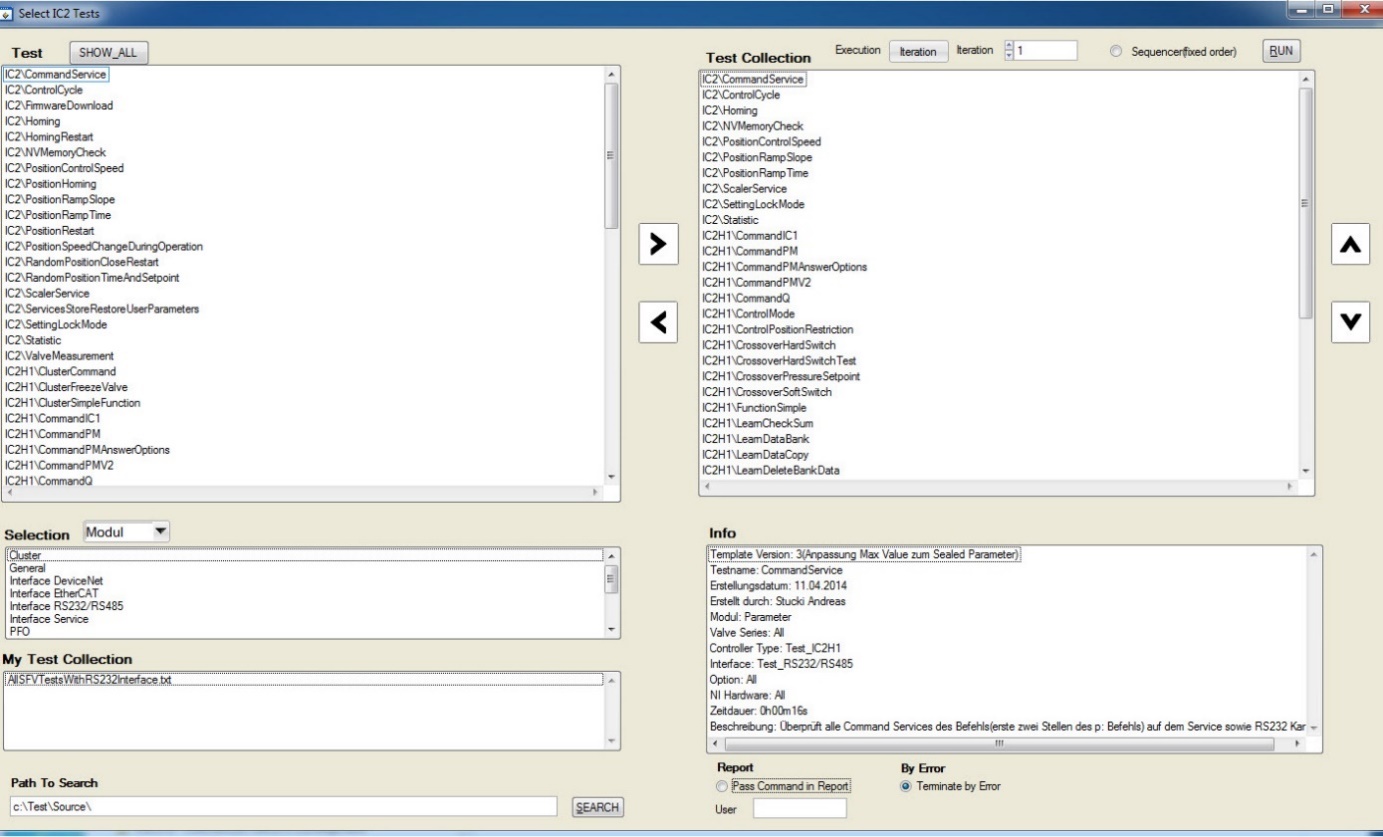


Abbildung : Ansicht der TTIC2 Oberfläche für die Auswahl der Testkollektion

**Während der Ausführung**

* Während die Testkollektion abläuft zeigt das Reportfenster, den aktuellen Test sowie welche die bereits ausgeführt worden sind sowie noch werden
* Weiter wird jedes Testergebnis notiert
* Wird ein Fehler detektiert, so wird diese Fehlermeldung Rot hervorgehoben
* Der User kann den Ablauf der Testkollektion abbrechen

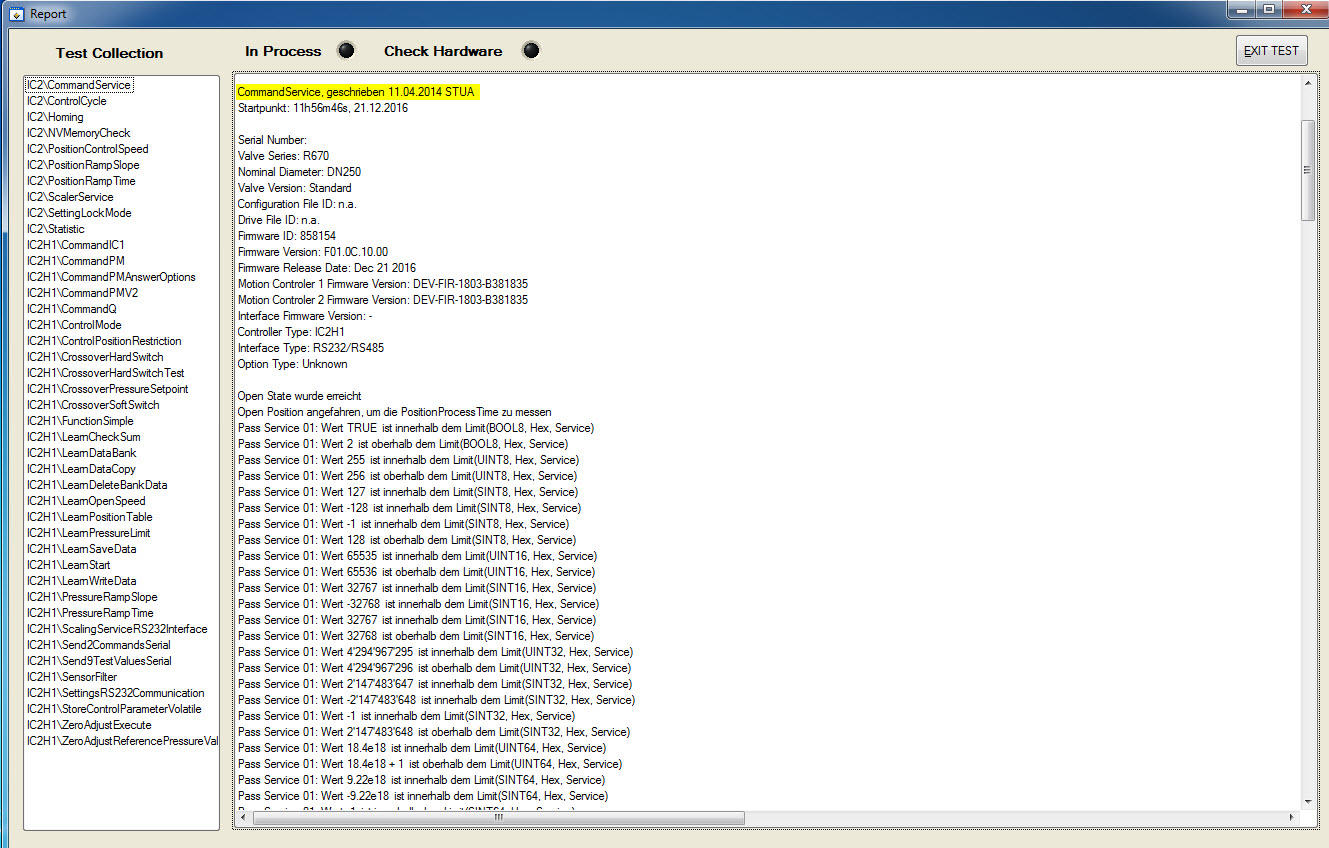


Abbildung : Report Ansicht währendem die Tests ausgeführt werden

**Nach der Ausführung**

* Der User kann den Report an einem gewünschten Ort abspeichern, wenn er dies nicht möchte wird der Report gelöscht
* Für jeden erfolgreichen Test wird ein Zertifizierungsfile erstellt.
* Bei allen fehlgeschlagenen Tests wird ein Diagnostik File erstellt.

#### 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

#### 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
* Auslesen der Testergebnisse geschieht im Reportfile

### 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 Firmwaren hinterleg, wie z.B. die Ventil, Motion Controller sowie auch Interface. Dabei wird auch definiert, welche Ventil Firmware mit welchem Motion- und Interface Firmwaren lauffähig ist.

#### Stärken

* Firmware Informationen sind für viele Benutzer ersichtlich und bearbeitbar.

#### Schwächen

* In der Modellierung wurden die Regeln der Normalform nur teilweise umgesetzt

### Firmware Database

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

#### Stärken

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

#### Schwächen

* Noch keine Benutzerverwaltung (Wird gebraucht um gewisse Informationen zu verstecken)

## Ausgangslage (Problemstellung)

Ich wurde von der Firma VAT eingestellt und beauftragt die Qualität des Controllers, mit Hilfe einer Testumgebung zu überprüfen. Im Kapitel 2.2 sind die einzelnen Elemente beschrieben. In diesem Abschnitt wird erklärt, welche Elemente und aus welchen Gründen eine Anpassung notwendig ist.

**Ventil Hardware:**

An der zu testenden Ventil Hardware wird sich nichts ändern, ausser das dir Produktepalette, welche qualifiziert werden soll, immer grösser wird. Was eine flexible Testumgebung voraus setzt.

**Test Hardware (NI PXI):**

Die Test Hardware wurde zu einem früheren Zeitpunkt evaluiert und angeschafft. Da die Anschaffungskosten erheblich sind, wird hier auch keine Änderung vorgenommen. Was aber vorkommen wird, ist eine Erweiterung der Karten. Der Grund liegt in der Vergrössrung 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.

**Test**

Im Test werden einerseits Informationen zur Kennzeichnung des Tests hinterlegt. Zu diesen gehören eine Beschreibung, die aktuelle Version, den Autor oder das Erstellungsdatum. Zusätzlich wird definiert für welche Ventil Hardware der Test ausführbar ist. Diese Informationen werden im gleichen Ordner, in welchem sich der Test befindet, in einem Textfile hinterlegt. Dieses File wird erzeugt, wenn das TTIC2 Programm kein Textfile im Testordner finden kann. So kann der Fall auftreten, dass die Informationen im Test im Vergleich zum Textfile nicht übereinstimmen. Aus diesem Grund werden diese Informationen neu in der SoftwareVersionsDatabase hinterlegt.

**ParameterStructBuild**

Das Programm ist schon so weit ausgereift, dass alle Parameter Informationen ausgelesen werden können. Kleinere Anpassungen, wie z.B. die Feldbus Objekte zu integrieren, werden zum entsprechenden Zeitpunkt realisiert, wenn das Interface fertig entwickelt ist. Dies wird nicht in dieser Masterarbeit behandelt.

**TestUpdateFirmware**

Aktuell werden immer alle Tests mit den entsprechenden Parameter Informationen neu kompiliert. Besitzt das Ventil nicht den gleichen Controller Typ, so kann der Test sowieso nicht ausgeführt werden. Somit wird das Programm so angepasst, dass nur entsprechende Tests kompiliert werden, welche den gleichen Controller Type besitzen wie die Ventil Hardware.

**TTIC2 Test Tool**

Mit dem Programm TTIC2 entwickelte ich eine Testoberfläche, welche einzelne Tests in einer Kollektion zusammenfasst und nacheinander ausführt.

Das Problem ist, dass aktuell nach der Ausführung der Testkollektion, das entstandene Report File manuell nach fehlerhaften Testdurchläufen durchsucht werden muss. Dazu enthält das Reportfile alle Testschritte und erreicht eine sehr grosse Datenmenge. Aus diesem Grund wird nur der letzte Report vor einer Ventil Firmware Freigabe im SVN abgelegt. Dies erschwert die Auswertung der Tests enorm. Zusätzlich werden auch Zertifizierungsfiles abgelegt, was durch die weitere Datenmenge nochmals erschwerend hinzukommt. Das soll neu mit dem ETIC2 Programm gelöst werden.

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

**SoftwareVerionsDatabase**

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

**Firmware Database**

Listet alle Firmware auf mit den dazugehörigen Informationen. Die Applikation übernimmt aktuell Funktionen der Datenbankmodellierung. Weiter werden auch die Datenbanktypen in der Applikation verwendet. Dies soll mit der Anpassung der SoftwareVersionsDatabase Modellierung verbessert werden sowie eine Converter Klasse eingefügt werden um eine saubere Trennung zwischen der Applikation und der Datenbank zu erhalten.

# Zielsetzung

Im ersten Unterkapitel werden die Ursprungsziele der Masterarbeit aufgezeigt. Im zweiten Teil werden die Ziele nach der ersten Analyse revidiert und auf die gesamte Testumgebung erweitert.

## Ziele zum Startzeitpunkt

Abbildung : Ursprüngliches Konzept Masterarbeit

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

Weiter soll die TTIC2 Oberfläche 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 Database Oberfläche erstellt.

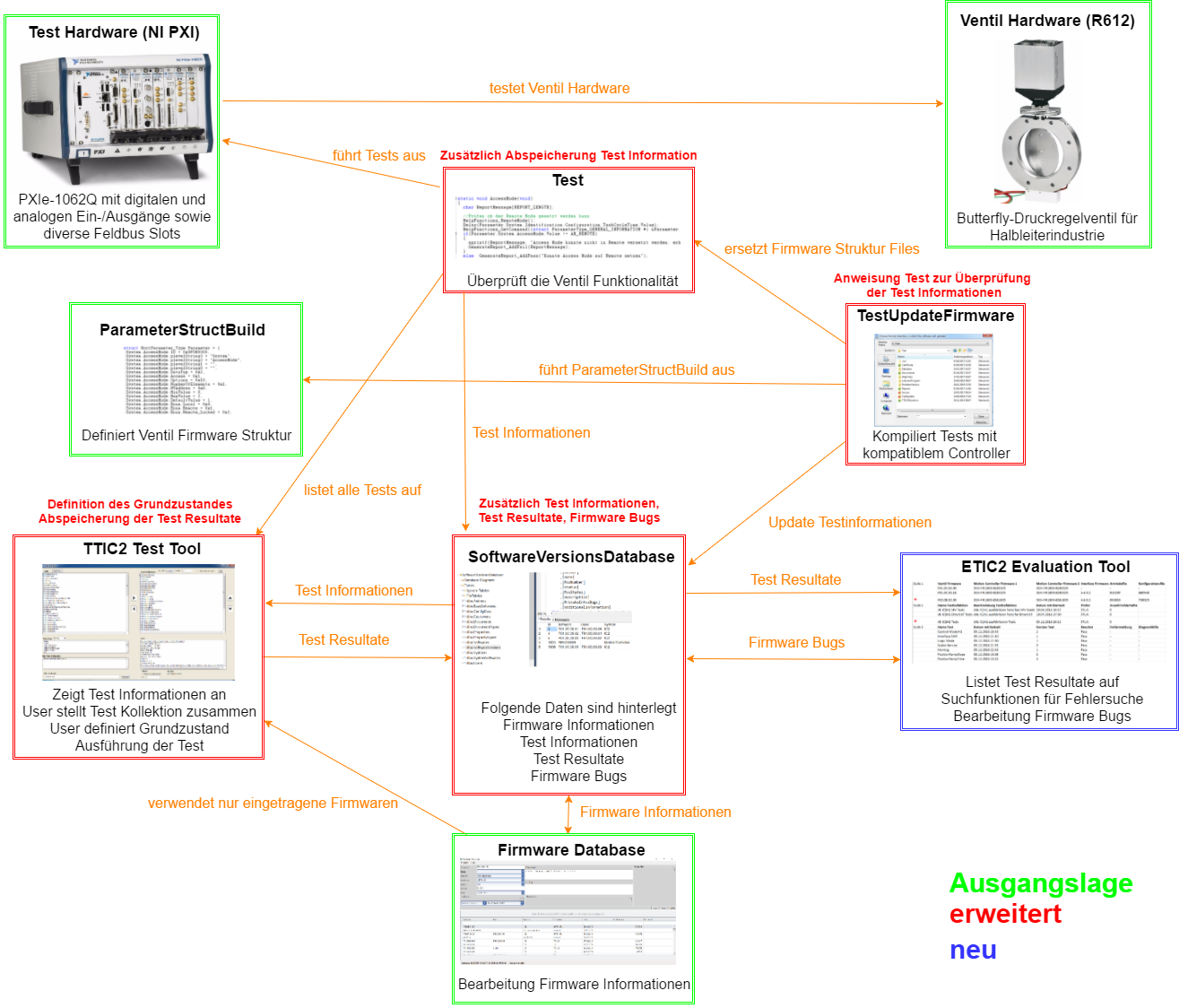
Das Hauptaugenmerk der Arbeit liegt in der Erstellung des Evaluation Tool Integrierter Controller 2 (ETIC2). Die Applikation wiedergibt die generierten Testresultate vom TTIC2. Die Oberfläche ist in drei Ebenen gegliedert. Die höchste Ebene unterteilt die Einträge nach den unterschiedlichen Grundzuständen. 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.

Unter Angabe eines Grundzustandes, welche aus einzelnen Elementen besteht welche in der SoftwareVersionsDatabase hinterlegt sind, werden alle Testresultate in einem Report aufgelistet. Speziell dabei ist, dass nicht alle Elemente des Grundzustandes definiert werden müssen.

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. Dies vor allem aus dem Hintergedanken, dass die Testinformationen immer aktuell sind und nicht zum Zeitpunkt, als das Info Textfile erstellt wurde. Als weiteres wurde auch eine Erweiterung der Datenbank mit den Firmware Bugs Informationen als sinnvoll erachtet.

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

## Ziele nach der ersten Analyse

Abbildung :

## Quantitative Ziele

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

## 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.
* Unter Angabe des Namens des Grundzustandes muss auf Knopfdruck im ETIC2 eine Auswertung aller ausgeführten Testkollektionen mit den dazugehörigen Resultaten aufgelistet werden.

## Aufgabenbegrenzung

* Anpassungen an den einzelnen Tests gehört nicht zur Arbeit
* Die Weiterentwicklung der TTIC2 Applikation ist nicht Teil der Masterarbeit.
  + Ausgenommen ist die Anbindung an die SoftwareVersionDatabase
* Die SoftwareVersionDatabase wird erweitert aber die bestehenden Attribute und Inhalte werden nicht angefasst.
* ETIC2 wird für den internen VAT Verwendungszweck entwickelt und nicht für den kommerziellen Gebrauch konzipiert.

# Methodik (Was? + Theorie + Konzept)

Bei der Modellierung wird das Tool MySQL Workbench eingesetzt. So kann grafisch die Struktur der Datenbank wiedergegeben werden. Weiter können die Attribute der Tabellen definiert werden, wie z.B. der Datentyp oder ob Null Werte zugelassen sind. Weiter sind auch die Beziehungen ersichtlich unter den Tabellen ersichtlich. Im nächsten Schritt werden die SQL Scripts geschrieben um diese anschliessend im SQL Server Management Studio ausführen zu können. Der Grund der Auswahl liegt darin, dass das Unternehmen mit SQL Servern arbeitet.

## Modellierung der SoftwareVersionsDatabase

### Abspeicherung der Software Informationen

Die SoftwareVersionsDatabase Datenbank wurde in einem früheren Projekt erstellt um die Informationen einzelner Firmwaren abzuspeichern. Es werden verschiedene Typen von Firmwaren in der gleichen Tabelle abgelegt. In einer weiteren Tabelle wird definiert, welche Firmwaren 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 Ventilfirmware sowie die Motion Controller Firmwaren wie auch Feldbus Firmwaren hinterlegt. Diese Unterscheidung wird im Feld System erkennbar.

Die PSS Nummer ermöglicht den Zugang zum interne ERP System.

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

### Abspeicherung der Testinformationen

### Abspeicherung der Testresultate

Das ETIC2 zeigt die Testresultate zu den einzelnen Ventilfirmwaren. Hierbei sind die Ventilfirmware Informationen von Interesse bei der Auswertung. Daher sind die Testresultate in der selben Datenbank zu finden. Weiter wird erreicht, dass nur eingetragene Firmwaren zum Testfall zugelassen sind. Unabhängig ob es sich hierbei um eine Ventil- Motion Controller Firmware oder Feldbus Firmware handelt.

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

* Antriebsfile
* Konfigurationsfile
* Test Collection
* Anzahl fehlerhafte Tests
* Test Version
* Test Resultat
* Fehlermeldungen eines Tests

Wichtig ist zudem die Information, welche Tests eine Collection beinhaltet. Die Information der Test Version spielt hierbei keine Rolle.

Die Test Version ist wichtig für die Information bezüglich Ventilhardware, welche benötigt wird 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 Collection noch ausführen können muss. Da die meisten Ventilfirmwaren kundenspezifische Entwicklungen sind, 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.

### Abspeicherung von Firmware Bugs

## Zugriff auf SoftwareVersionsDatabase

Abbildung : TTIC2 Zugriff auf SoftwareVersionsDatabase

Das TTIC2 verwendet die SoftwareVersionsDatabase um die Testresultate zu hinterlegen. Weiter definiert der User den Grundzustand, mit welcher er die Tests ausgeführt haben möchte. Einerseits muss dieser Grundzustand nach Auswahl hinterlegt werden, wie aber auch dem User alle Möglichkeiten aufzeigen, die er hat um einen Grundzustand zu definieren.

Als zweites liefert die SoftwareVersionsDatabase die Hardware Anforderungen um die Tests ausführen zu können. Das TTIC2 prüft bevor die Tests ausgeführt werden, ob die angeschlossene Ventilhardware alle Tests ausführen kann und informiert den User darüber.

## Erstellung ETIC2

### Design View Model

### Codierung nach MVVM

### Anbindung SoftwareVersionsDatabase

### Ausgabe Bericht

# Beschreibung der Arbeit (Praxis, Umsetzung)

## Modellierung der SoftwareVersionsDatabase

### Abspeicherung der Firmware Informationen

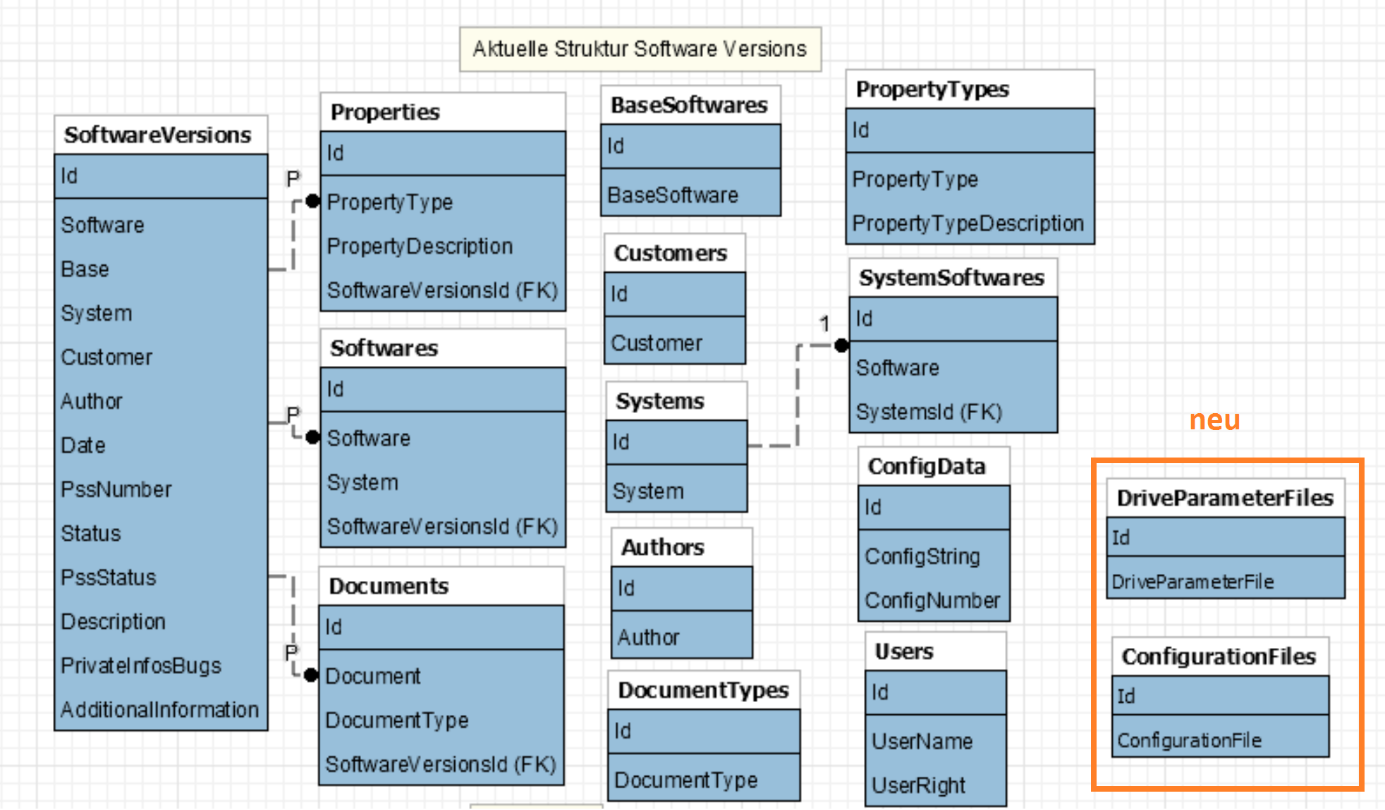
In der Abbildung 6 ist das aktuelle SoftwareVersionsDatabase Modell ersichtlich.

Abbildung Aktuelle SoftwareVersionsDatabase Struktur

Im Modell wurde 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 Testes gebraucht.

Was bei genaueren Betrachten der Attribute der einzelnen Tabellen auffällt ist, dass diese Mehrfach vorkommen. Wie z.B. 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. Dies wiederspricht der Normalisierungsregel. Die Idee der Customers Tabelle liegt darin, dass nur Einträge dieser Tabelle in der Haupttabelle eingetragen werden können. Um z.B. Schreibfehler, unterschiedliche Reihenfolge des Namens und Vornamen vorzubeugen. Wird ein Fehler bemerkt, werden alle Einträge bei Korrektur mitgeändert in der Haupttabelle. Das Konzept wurde hier von der Applikation übernommen, da hier nur Einträge für den User angezeigt werden, welche in der Customers Tabelle hinterlegt sind.

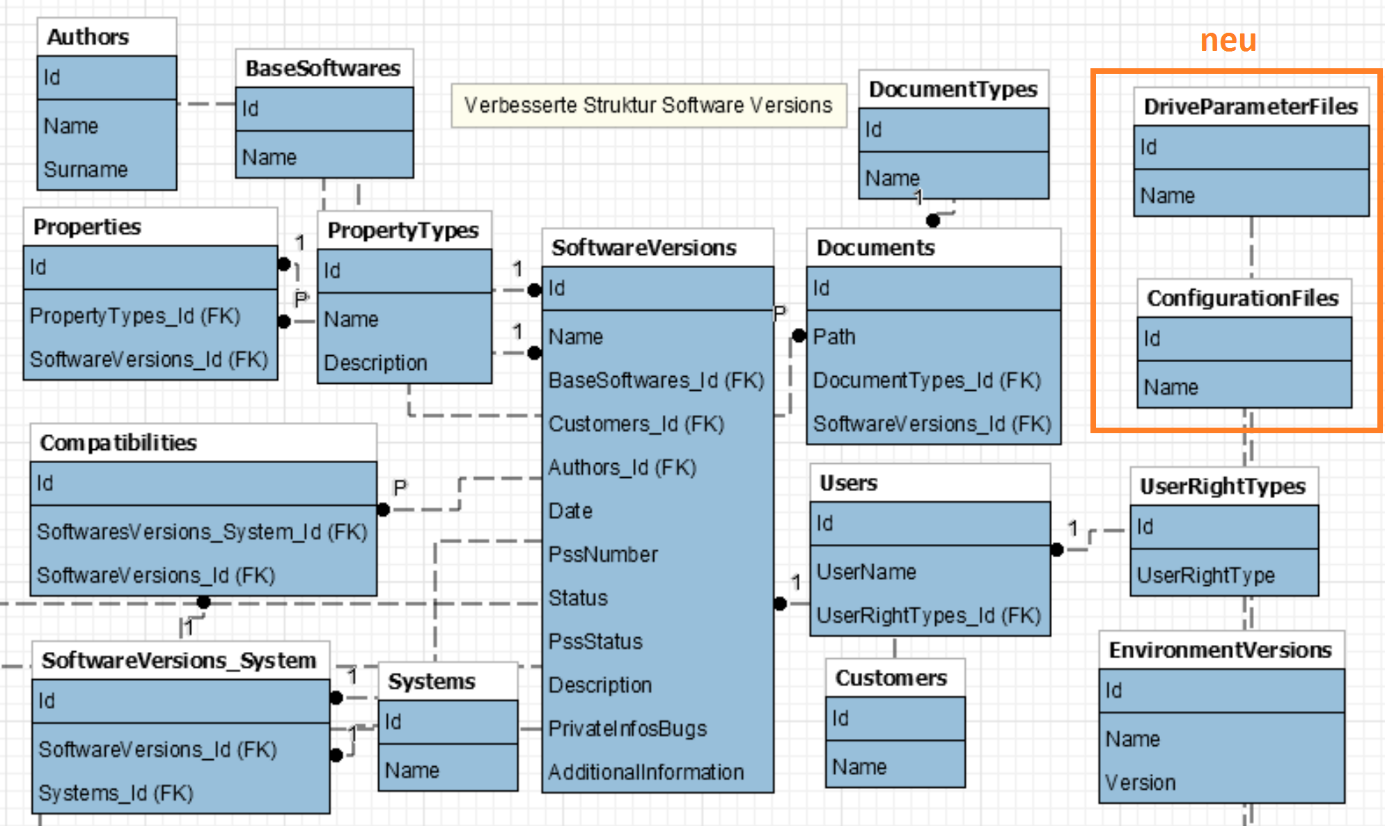
Daher wurde das SoftwareVersionsDatabase Modell überarbeitet, welche in der unteren Abbildung 7 ersichtlich ist.

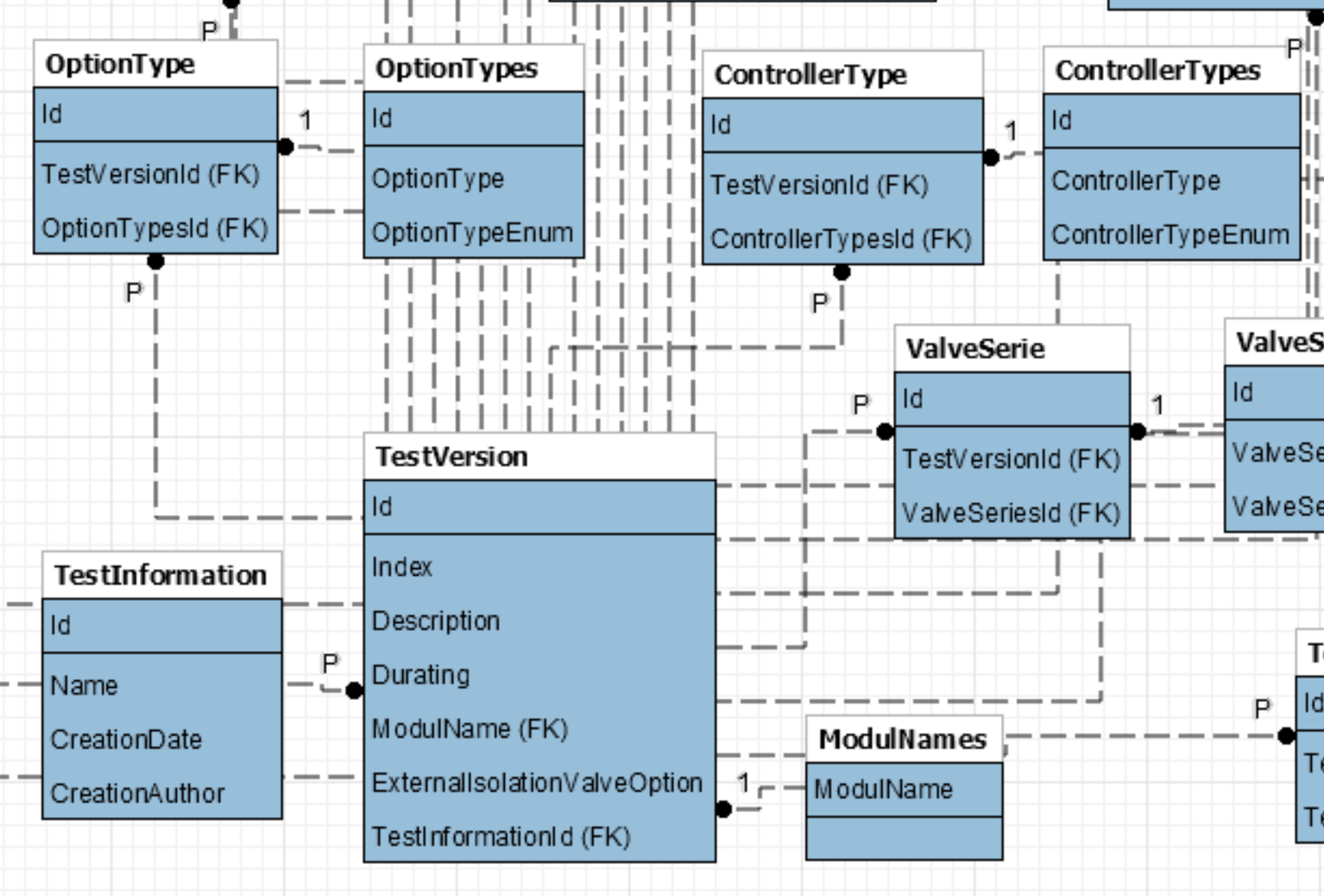
Abbildung : Überarbeitete SoftwareVersionsDatabase Struktur

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.

Was hier weiter auffällt ist, dass die Einzelnen Attribute nicht mehr den Tabellennamen auch noch beinhalten. Weiter wurde hier bei der Authors Tabelle der Name und Vorname aufgesplittet dies kann bei der späteren Auswertung nützlich sein.

Um dieses Modell nun auszuführen, muss die Oberfläche des Firmware Verwaltungstool angepasst werden. Dies wurde aus Zeitgründen nicht weiterverfolgt. Somit wird in dieser Arbeit noch mit dem ursprünglichen Modell weitergearbeitet. Unter Kapitel 8.1 ist beschrieben, wie später der Umbau auf das neuere Modell realisiert werden soll.

### Abspeicherung der Testinformationen

Abbildung : Erweiterung SoftwareVersionsDatabase zur Speicherung der Testinformationen

### Abspeicherung der Testresultate

In der Abbildung 8 ist die Erweiterung der SoftwareVersionsDatabase zu sehen, um die Testresultate abzuspeichern.

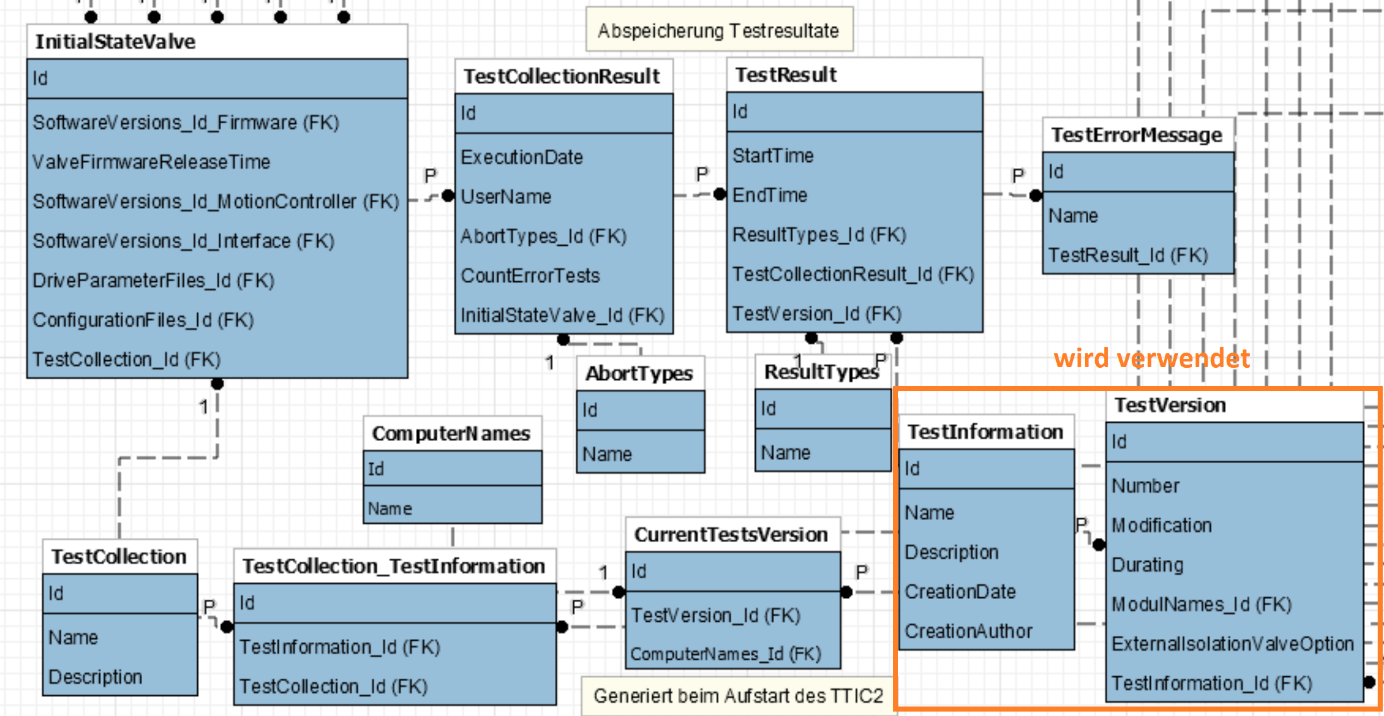


Abbildung : Erweiterung SoftwareVersionsDatabase zur Speicherung der Testresulate

Dabei übernimmt die InitialStateValve Tabelle die Funktion einen Grundzustand vor dem Start der Test Collection herzustellen. In der ETIC2 Applikation werden diese Information in der ersten Ebene angezeigt. Die Modellierung erzwingt, dass die Firmwaren wie auch die DriveParameter- und ConfigurationFile schon in der Datenbank hinterlegt sein müssen, bevor der Grundzustand definiert werden kann.

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

In der nächst tieferen Ebene wird die Test Collection, die ausgeführt wurde hinterlegt. Was der Anwender hier interessiert ist die Anzahl fehlerhaften Tests. Weiter ist auch hier ersichtlich ob der User den Testablauf unterbrochen hat.

In der untersten Ebene sind alle Tests der Test Collection 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 CurrentTestsVersion Tabelle. Diese wird bei jedem Start des TTIC2 Programms neu gefüllt. Dies ist wichtig um die entsprechenden Hardware Anforderungen der Tests auslesen zu können und zu entscheiden ob der Test mit der angeschlossenen Hardware ausführbar ist. Über den Zeitlauf des Tests können sich die Anforderungen verändern, da neue Testfunktionen hinzukommen können.

Im Modell sind auch die Testinformationen aufgezeichnet. Da es bei der Auswertung wichtig ist, welche Version des Tests ausgeführt wurde. Ein möglicher Grund einer Test Anpassung kann in der veränderten Firmware Spezifikation liegen, welche sich über die Laufzeit ändern können. Die Testinformationen bezüglich Hardware Anforderungen wie auch Beschreibung des Tests werden in einem separaten Programm erzeugt. Dies ist nicht Teil dieser Arbeit.

### Abspeicherung von Firmware Bugs

Abbildung : Erweiterung SoftwareVersionsDatabase zur Speicherung der Firmware Bugs

## Zugriff auf SoftwareVersionsDatabase

Das TTIC2 ist wie schon in Kapitel 2.3 beschrieben in CVI realisiert. In der Disposition war angedacht die Anbindung mit Hilfe des Entity Framework zu realisieren. Das hätte bedeutet, dass in C# dll Funktionalitäten geschrieben werden müsste und diese später im TTIC2 integriert würden. Dieser Ansatz bietet mehrere Nachteile. Erstens müsste bei jeder Änderung die dll Datei ersetzt werden. Eine Schnittstelle zwischen unterschiedlichen Programmen bedeutet immer, dass die Übergabeparameter gut überlegt werden sollten.

Das CVI bietet für die Anbindung an einen SQL Server Funktionalitäten in ihrem SQL Toolkit. Daher wird der Ansatz gewählt, dass direkt aus CVI die SoftwareVersionsDatabase angesprochen wird.

## Erstellung ETIC2

### Design View Model

### Codierung nach MVVM

### Anbindung SoftwareVersionsDatabase

### Ausgabe Bericht

# Ergebnisse (Tool)

# Diskussion (Was hat Funktioniert, was nicht)

# Ausblick (offene Punkte, wie geht es weiter)

## Umsetzung Überarbeitung SoftwareVersionsDatabase

Wie unter Kapitel 5.1.1 beschrieben, besitzt das aktuelle SoftwareVersionDatabase Modell noch verbesserungspotential. Die Umsetzung braucht die Anpassung der Software Verwaltungstool Oberfläche. Dies wurde in WPF und nach dem MVVM Pattern Konzept erstellt. Nun die Anpassung darf keinen Einfluss auf das ModelView wie auch der View haben. Daher liegt der Ansatz nahe das Modell auf dem neueren Modell der Datenbank anzupassen. Dazu wird eine Wrapper Klasse erstellt, in welcher die Umwandlung der alten auf die neue Struktur erfolgt.

## Integration Buglist in ETIC2

Die Datenbankfelder für die Verwaltung der Firmware Fehlern wurde bereits in dieser Arbeit erstellt. Weiter soll jetzt im ETIC2 die Option bestehen diese Firmware Fehler anzeigen zu können. Auch in diesem Projekt ist es wichtig eine schnelle Möglichkeit zu haben um 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, welche alle Informationen im oberen Bereich aufgelistet wird. Der grösste Teil nimmt die Ansicht aller in der Datenbank abgespeicherten Einträge ein.

# Verzeichnisse

## Literaturverzeichnis

Leo, M. (03. 04 2010). PG\_Info\_Hardware. VAT Interne Präsentation. Haag.

VAT Group AG. (2017). Abgerufen am 26. Februar 2017 von http://www.vatvalve.com/de/business/industry

## Abkürzungsverzeichnis

|  |  |
| --- | --- |
| NI | National Instruments |
|  |  |
|  |  |

## Abbildungsverzeichnis

[Abbildung 1: Basiskonzept Ventil Controller (Marugg, 2010) 6](#_Toc485145474)

[Abbildung 2: Ist-Zustand der Testumgebung 7](#_Toc485145475)

[Abbildung 3: Ansicht der TTIC2 Oberfläche für die Auswahl der Testkollektion 10](#_Toc485145476)

[Abbildung 4: Report Ansicht währendem die Tests ausgeführt werden 11](#_Toc485145477)

[Abbildung 5: Konzept Masterarbeit 13](#_Toc485145478)

[Abbildung 6: TTIC2 Zugriff auf SoftwareVersionsDatabase 16](#_Toc485145479)

[Abbildung 7 Aktuelle SoftwareVersionsDatabase Struktur 17](#_Toc485145480)

[Abbildung 8: Überarbeitete SoftwareVersionsDatabase Struktur 18](#_Toc485145481)

[Abbildung 9: Erweiterung SoftwareVersionsDatabase zur Speicherung der Testinformationen 19](#_Toc485145482)

[Abbildung 10: Erweiterung SoftwareVersionsDatabase zur Speicherung der Testresulate 19](#_Toc485145483)

[Abbildung 11: Erweiterung SoftwareVersionsDatabase zur Speicherung der Firmware Bugs 20](#_Toc485145484)

## Tabellenverzeichnis

## 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. |
| Konfigurations-  file | Enthält alle Abweichungen der Firmware gegenüber den Standard Ventil Firmware Einstellungen, welche in der Firmware hinterlegt sind. |
| MVVM | Mode View ViewModel |
| SVN | Apache Subversion: Freie Software zur Versionsverwaltung. |
| TTIC2 | Test Tool Integrierter Controller 2: Testoberfläche für alle integrierten Ventilcontroller der 2. Generation |
| WPF | Windows Presentation Foundation |

# Anhang

## Zeitplan

# 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: ………............