

**Management Summary**

* Problemstellung
* Methodik/Ablauf
* Resultat

Inhalt

[ETIC2 zur Verwaltung von Ventiltests 5](#_Toc484947731)

[1 Einleitung (1 Seite) 5](#_Toc484947732)

[2 Grundlagen 6](#_Toc484947733)

[2.1 Situation 6](#_Toc484947734)

[2.2 Testumgebung (Ist-Zustand) 7](#_Toc484947735)

[2.2.1 Ventil Hardware 7](#_Toc484947736)

[2.2.2 Test Hardware (NI PXI) 7](#_Toc484947737)

[2.2.3 Test 7](#_Toc484947738)

[2.2.4 ParameterStructBuild 7](#_Toc484947739)

[2.2.5 TestUpdateFirmware 7](#_Toc484947740)

[2.2.6 TTIC2 Test Tool 7](#_Toc484947741)

[2.2.7 SoftwareVersionsDatabase 10](#_Toc484947742)

[2.2.8 FirmwareDatabase 10](#_Toc484947743)

[2.3 Ausgangslage (Problemstellung) 10](#_Toc484947744)

[3 Zielsetzung 11](#_Toc484947745)

[3.1 Quantitative Ziele 11](#_Toc484947746)

[3.2 Qualitative Ziele 11](#_Toc484947747)

[3.3 Aufgabenbegrenzung 12](#_Toc484947748)

[4 Methodik (Was? + Theorie + Konzept) 13](#_Toc484947749)

[4.1 Modellierung der SoftwareVersionsDatabase 13](#_Toc484947750)

[4.2 Aktuelle Tabellenstruktur 13](#_Toc484947751)

[4.3 Abspeicherung der Testinformationen 13](#_Toc484947752)

[4.4 Abspeicherung der Testresultate 13](#_Toc484947753)

[4.5 Abspeicherung von Firmware Bugs 14](#_Toc484947754)

[4.6 Zugriff auf SoftwareVersionsDatabase 14](#_Toc484947755)

[4.7 Erstellung ETIC2 14](#_Toc484947756)

[4.8 Design View Model 14](#_Toc484947757)

[4.9 Codierung nach MVVM 14](#_Toc484947758)

[4.10 Anbindung SoftwareVersionsDatabase 14](#_Toc484947759)

[4.11 Ausgabe Bericht 14](#_Toc484947760)

[5 Beschreibung der Arbeit (Praxis, Umsetzung) 15](#_Toc484947761)

[5.1 Modellierung der SoftwareVersionsDatabase 15](#_Toc484947762)

[5.2 Aktuelle Tabellenstruktur 15](#_Toc484947763)

[5.3 Abspeicherung der Testinformationen 16](#_Toc484947764)

[5.4 Abspeicherung der Testresultate 16](#_Toc484947765)

[5.5 Abspeicherung von Firmware Bugs 18](#_Toc484947766)

[5.6 Zugriff auf SoftwareVersionsDatabase 18](#_Toc484947767)

[5.7 Erstellung ETIC2 18](#_Toc484947768)

[5.8 Design View Model 18](#_Toc484947769)

[5.9 Codierung nach MVVM 18](#_Toc484947770)

[5.10 Anbindung SoftwareVersionsDatabase 18](#_Toc484947771)

[5.11 Ausgabe Bericht 18](#_Toc484947772)

[6 Ergebnisse (Tool) 19](#_Toc484947773)

[7 Diskussion (Was hat Funktioniert, was nicht) 20](#_Toc484947774)

[8 Ausblick (offene Punkte, wie geht es weiter) 21](#_Toc484947775)

[8.1 Umsetzung Überarbeitung SoftwareVersionsDatabase 21](#_Toc484947776)

[8.2 Integration Buglist in ETIC2 21](#_Toc484947777)

[Verzeichnisse 22](#_Toc484947778)

[Literaturverzeichnis 22](#_Toc484947779)

[Abbildungsverzeichnis 23](#_Toc484947780)

[Tabellenverzeichnis 23](#_Toc484947781)

[Glossar 24](#_Toc484947782)

[Anhang 25](#_Toc484947783)

[Zeitplan 25](#_Toc484947784)

[Selbständigkeitserklärung 26](#_Toc484947785)

# 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 Software 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)

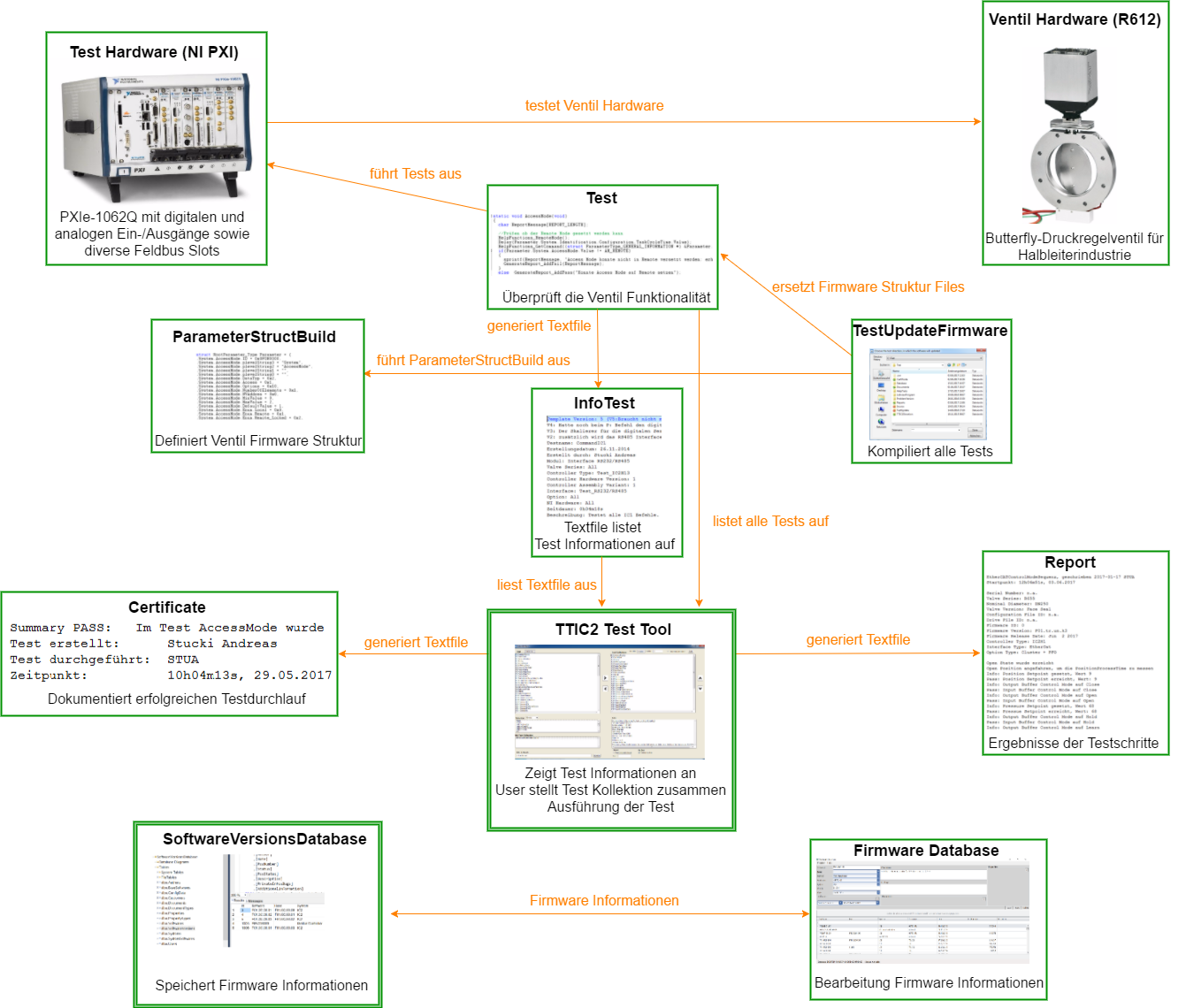
In diesem Unterkapitel wird die gesamte Testumgebung kurz erklärt. Das primäre Ziel liegt in der automatischen und wieder Verwendbarkeit von einzelnen Tests. Weiter soll der Wartungsaufwand klein gehalten werden um verschiedene Ventilhardware mit der gleichen Testumgebung qualifizieren zu können. Ein weiteres zentrales Element bildet die Dokumentation der Testschritte und 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 angesprochen werden. Historisch bedingt gibt es auch die Lösungen dies mit der Cluster Option des Ventils oder über die RS485 Interface Schnittstelle, mehrere Ventile ansprechen zu können.

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. Weiter bietet dieser Stecker 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 schnelle Ansprechung der Controller Hardware. 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 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 Software 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.

### TestUpdateFirmware

### TTIC2 Test Tool

Das TTIC2 sowie die Ventiltests sind in CVI geschrieben. Der Grund liegt in der einfachen ansprechen der Ventilhardware über die National Instruments Komponenten. 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

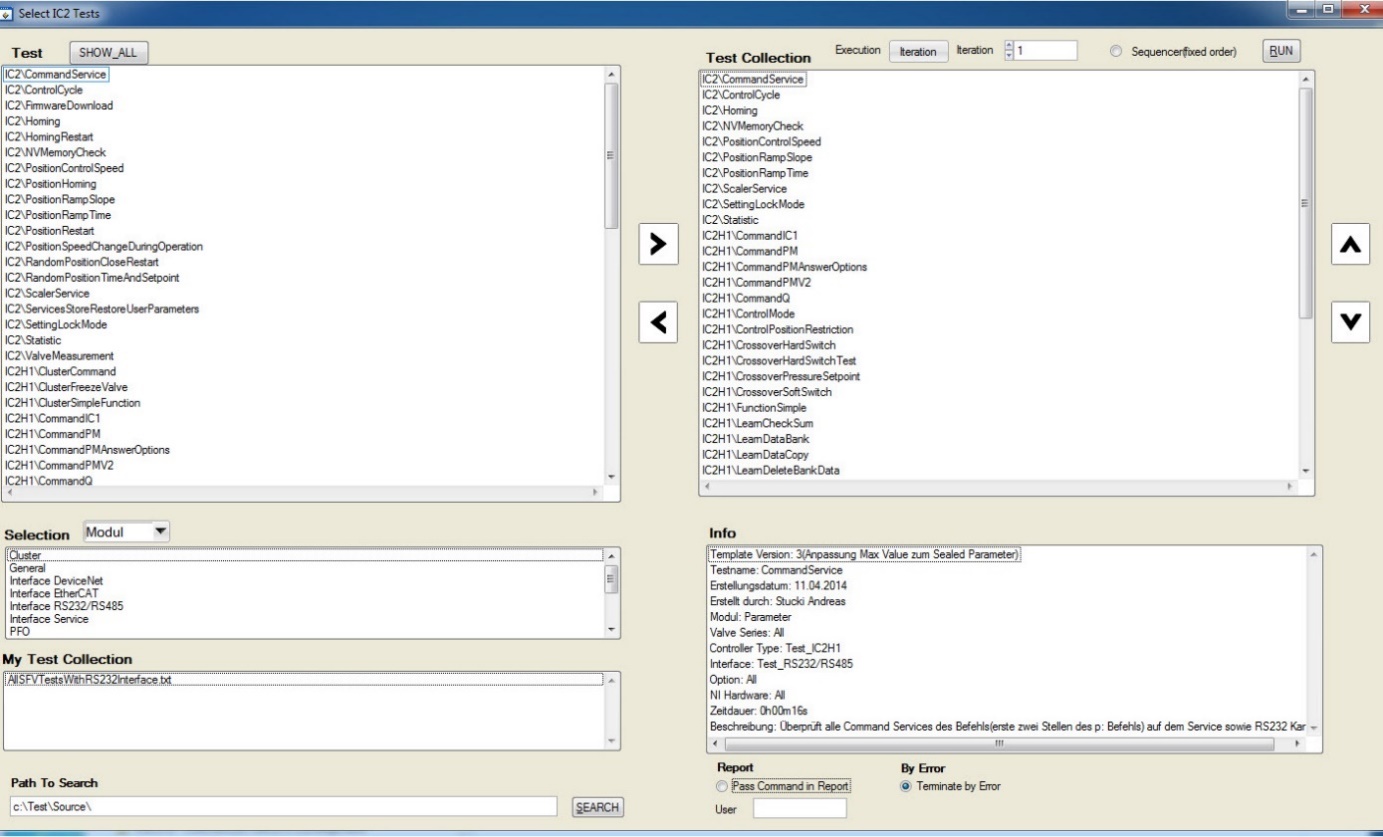


Abbildung : Ansicht der TTIC2 Oberfläche für die Auswahl der Testkollektion

**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

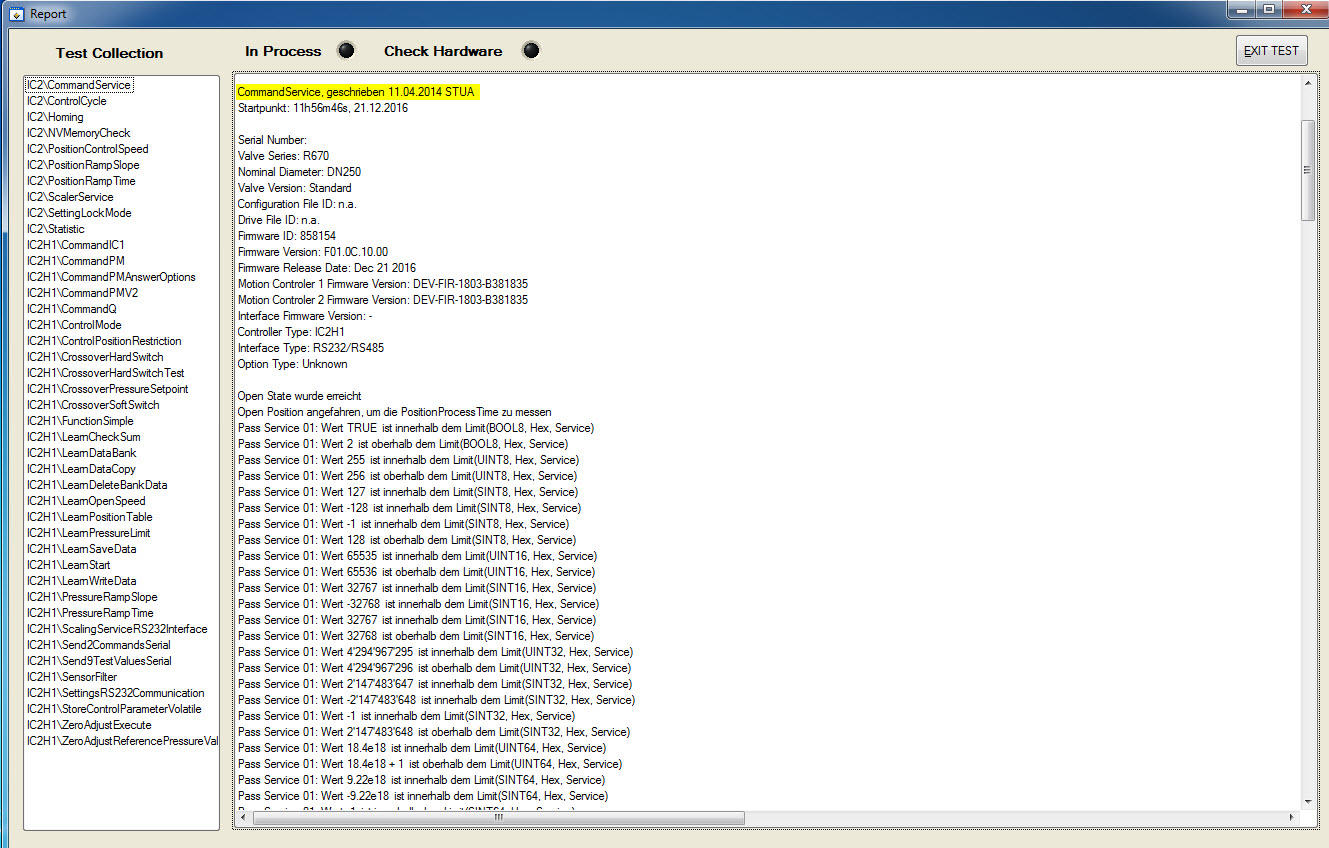


Abbildung : Report Ansicht währendem die Tests ausgeführt werden

**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

### FirmwareDatabase

## Ausgangslage (Problemstellung)

Ich wurde von der Firma VAT eingestellt und beauftragt die Qualität des Controllers, mit Hilfe einer Testumgebung zu überprüfen. 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.

# Zielsetzung

Abbildung : Konzept Masterarbeit

Die Testergebnisse werden in der SoftwareVersionsDatabase (SQL, Structured Query Language) hinterlegt. Diese Datenbank besteht bereits und hat die einzelnen Firmware Inhalte abgespeichert.

Nun soll sie mit folgenden Elementen erweitert werden:

* Name der Testkollektion
* Datum der Ausführung
* Anzahl fehlerhafte Tests
* Resultat der einzelnen Tests
* Bei fehlerhaften Testdurchlauf sollen die ersten kurzen Fehlermeldungen abgespeichert werden
* Zudem soll ein Diagnostikfile abgelegt werden

Es soll weiter mit folgenden Informationen der Grundzustand des Testablaufs festgelegt werden:

* Ventil Firmware
* Motion Controller Firmware
* Interface Firmware
* Antriebsfile
* Konfigurationsfile

Das Programm ETIC2 (E für Evaluation) soll erstellt werden, welches eine Übersicht über die Ergebnisse der einzelnen Testkollektionen gibt. Das ETIC2 listet die einzelnen Testkollektionen unter dem vorherig definierten Grundzustand auf. Das ETIC2 soll zudem noch eine Report Funktion beinhalten, die zum definiertem Grundzustand alle Ergebnisse rapportiert.

## 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

### Aktuelle Tabellenstruktur

Die SoftwareVersionsDatabase Datenbank wurde in einem früheren Projekt erstellt um die Informationen einzelner Software abzuspeichern. Es werden verschiedene Typen von Softwaren in der gleichen Tabelle abgelegt. In einer weiteren Tabelle wird definiert, welche Softwaren miteinander kompatibel sind.

Wichtigste Informationen bezüglich einer Software die hinterlegt werden sind:

* Name
* Basis
* System
* Customer
* Autor
* Erstellungsdatum
* PSS Nummer (interne Produkt Nummer)
* Beschreibung
* Kompatible Softwaren

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

Die PSS Nummer ermöglicht den Zugang zum interne ERP System.

Unter kompatible Softwaren werden die Motion Controller Firmwaren und Feldbus Softwaren notiert, welche mit der Ventilsoftware lauffähig sind. D.h. es gibt nur Einträge, wenn es sich beim aktuellen Software 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 Softwaren zum Testfall zugelassen sind. Unabhängig ob es sich hierbei um eine Ventil- Motion Controller Firmware oder Feldbus Software 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 Test Resultate 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

### Aktuelle Tabellenstruktur

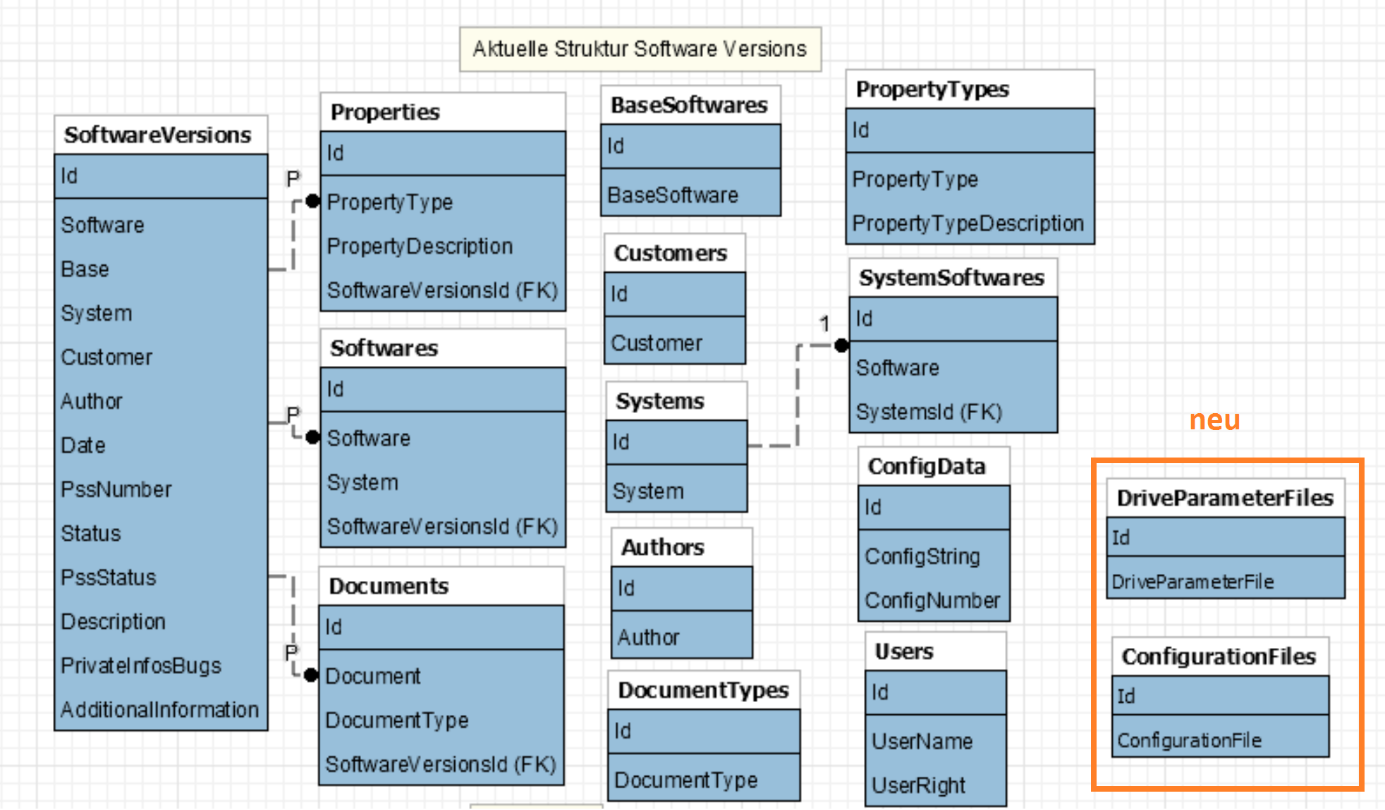
In der Abbildung 6 ist das aktuelle SoftwareVersionsDatabase Modell ersichtlich.

Abbildung Aktuelle SoftwareVersionsDatabase Struktur

Im Modell wurde noch die DriveParameterFiles und ConfigurationFiles eingezeichnet, da diese näher an den Informationen der Software 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.

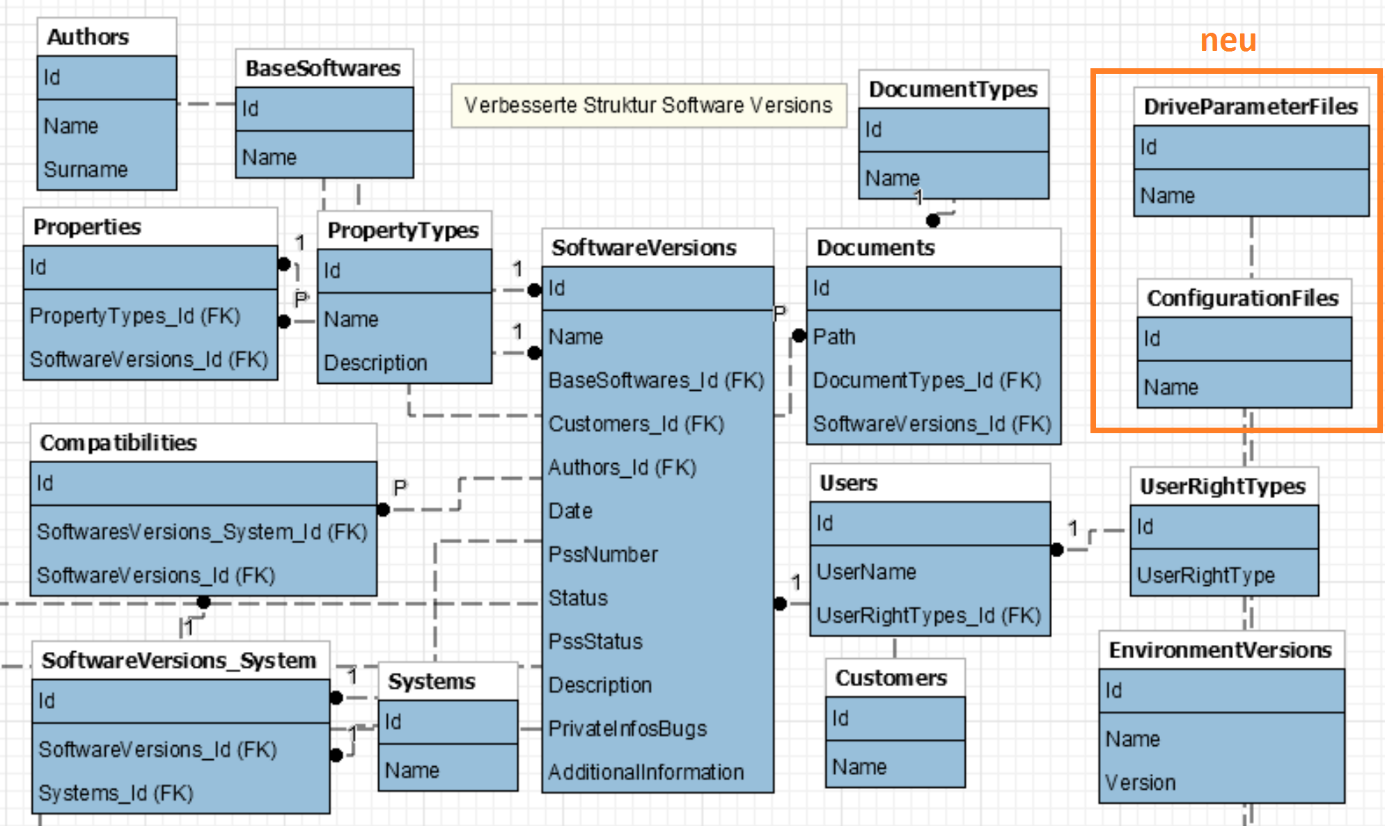
Daher wurde das SoftwareVersionsDatabase Modell überarbeitet, welche in der unteren Abbildung 7 ersichtlich ist.

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 Software 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

### Abspeicherung der Testresultate

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

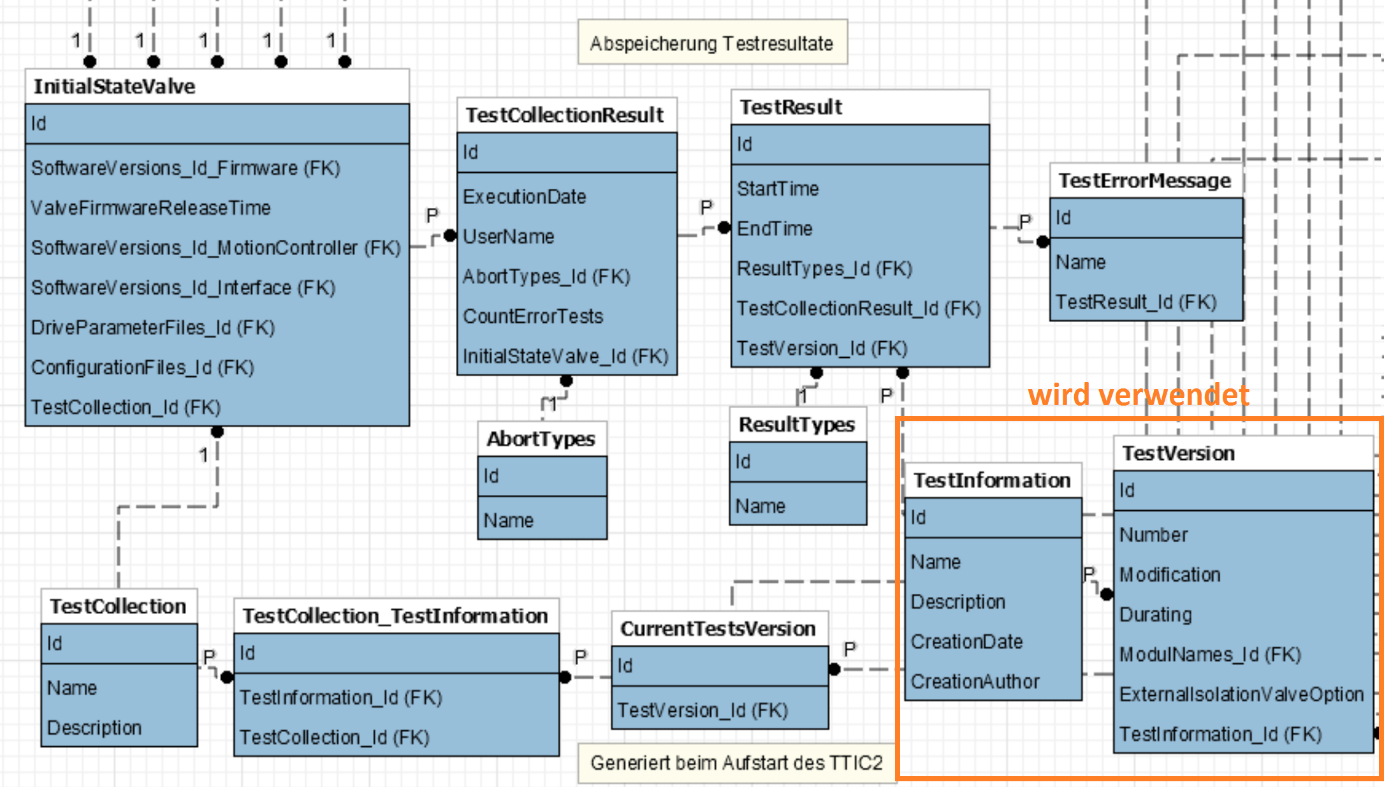


Abbildung : Erweiterung SoftwareVersionsDatabase zur Speicherung der Testresulate

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 Softwaren 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 Software mit einer sogenannten trunk Version gearbeitet. Diese bildet immer zum entsprechenden Zeitpunkt die aktuellste Software 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 Software 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 Software 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

## 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 Software Fehlern wurde bereits in dieser Arbeit erstellt. Weiter soll jetzt im ETIC2 die Option bestehen diese Software 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 Software 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](#_Toc483470341)

[Abbildung 2: Ansicht der TTIC2 Oberfläche für die Auswahl der Testkollektion 7](#_Toc483470342)

[Abbildung 3: Report Ansicht währendem die Tests ausgeführt werden 8](#_Toc483470343)

[Abbildung 4: Konzept Masterarbeit 10](#_Toc483470344)

[Abbildung 5: TTIC2 Zugriff auf SoftwareVersionsDatabase 13](#_Toc483470345)

[Abbildung 6 Aktuelle SoftwareVersionsDatabase Struktur 14](#_Toc483470346)

[Abbildung 7: Überarbeitete SoftwareVersionsDatabase Struktur 15](#_Toc483470347)

## 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 |
| 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 Software 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: ………............