Zürcher Hochschule
für Angewandte Wissenschaften

zh aw School of Engineering

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

- Problemstellung
- Methodik/Ablauf
- Resultat

Inhalt

ETIC2 zur Verwaltung von Ventiltests

1 Einleitung (1 Seite)

- Problemstellung
- Vorgehen

2 Grundlagen

Das Grundlagenkapitel ist in zwei wichtige Elemente unterteilt. Einerseits wird die Umgebung, die Situation und der aktuelle Stand sozusagen die Grundvoraussetzungen beschrieben. Darauf folgt andererseits die Hauptaufgabe dieser Masterarbeit die Erweiterung der Testumgebung, dabei wird beschrieben was schon entwickelt wurde, wo es Anpassungen geben wird und wie das neue Produkt (ETIC2) angedacht ist.

2.1 Situation

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

2.1.1 Controller

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 mit einem Marktanteil von über 40% klarer Weltmarktführer. Die Firma ist bekannt für ihre Regelventile, bei denen ein Controller die Steuerung dieser Vakuumventile übernimmt (siehe Abb. 1). Dieser Controller ist modular aufgebaut und besteht grob gesagt aus drei Komponenten. Die wichtigste Komponente ist das Masterboard, die 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 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, 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 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 Feldbus System, so kann er mit Hilfe eines Cluster Systems mehrere Ventile ansprechen. (Marugg, 2010).

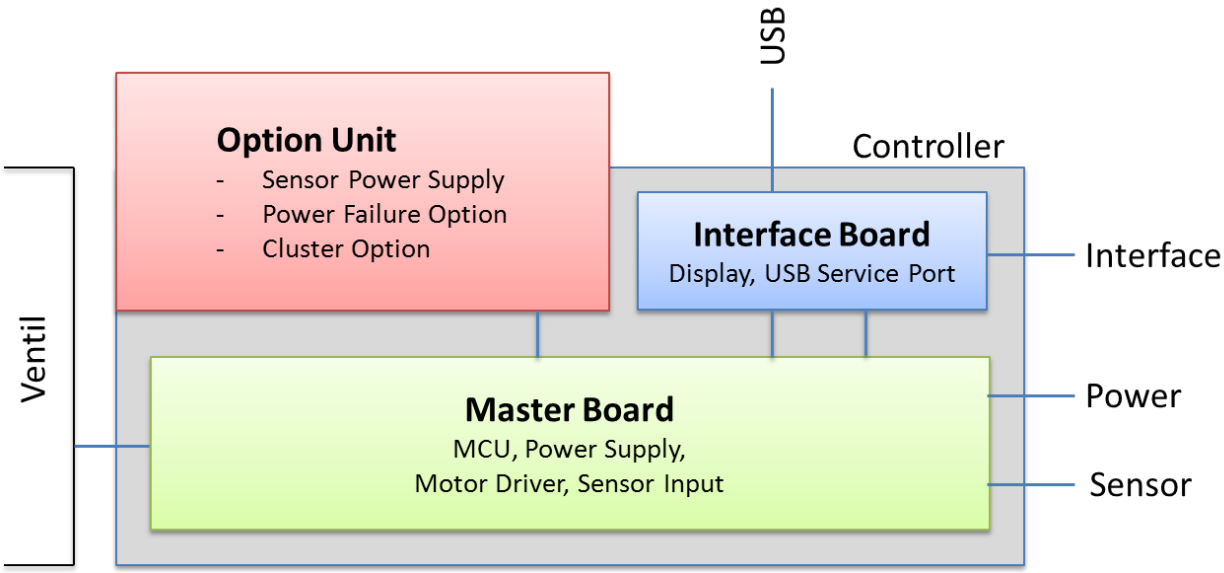


Abbildung 1: Basiskonzept Ventil Controller (Marugg, 2010)

2.1.2 Firmware

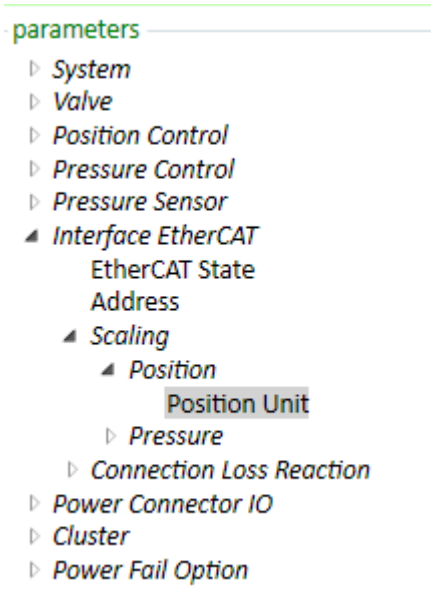


Abbildung 2: Parameterbaumstruktur der Software

Wie bereits im vorherigen Kapitel erwähnt, ist das Masterboard mit einem Mikrocontroller bestückt. Die Gründe für den Einsatz des Mikrocontrollers liegen bei den niedrigen Anschaffungskosten und den fundierten Programmierkenntnissen der Mitarbeitenden 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 welcher grob gesagt jeder Parameter eine Funktionalität des Ventils wiederspiegelt. Dabei kann es sich um Parameter handeln, die gesetzt werden oder die reine Status Informationen 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 Anwender bei der Suche nach der gewünschten Einstellung helfen.

2.1.3 Test

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 Zeit für die Überprüfung aus Zeitgründen sehr schmal ausfiel. Die Basis bildete Excel Listen mit Ventilbefehlen, welche sequentiell abgespielt werden und Überprüfkommandos beinhalteten. Die Schwachstellen dieses System lagen in der Wartung sowie in den begrenzten Funktionsmöglichkeiten, welche durch visuelle Überprüfung des Testers übernommen werden musste.

Um diese Probleme für die neue integrierte Controller Generation zwei (IC2) zu lösen, musste ein neues System gefunden werde. Das primäre Ziel war es völlig automatische Tests zu entwickeln, welche keinen Einfluss des Testers benötigten. Dadurch war schnell klar, um Berechnungen durchzuführen und um 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 Unterkapitel näher beschrieben wird.

2.2 Testumgebung (Ist-Zustand, Problemstellung)

Im Unterkapitel 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 Problemstellung.

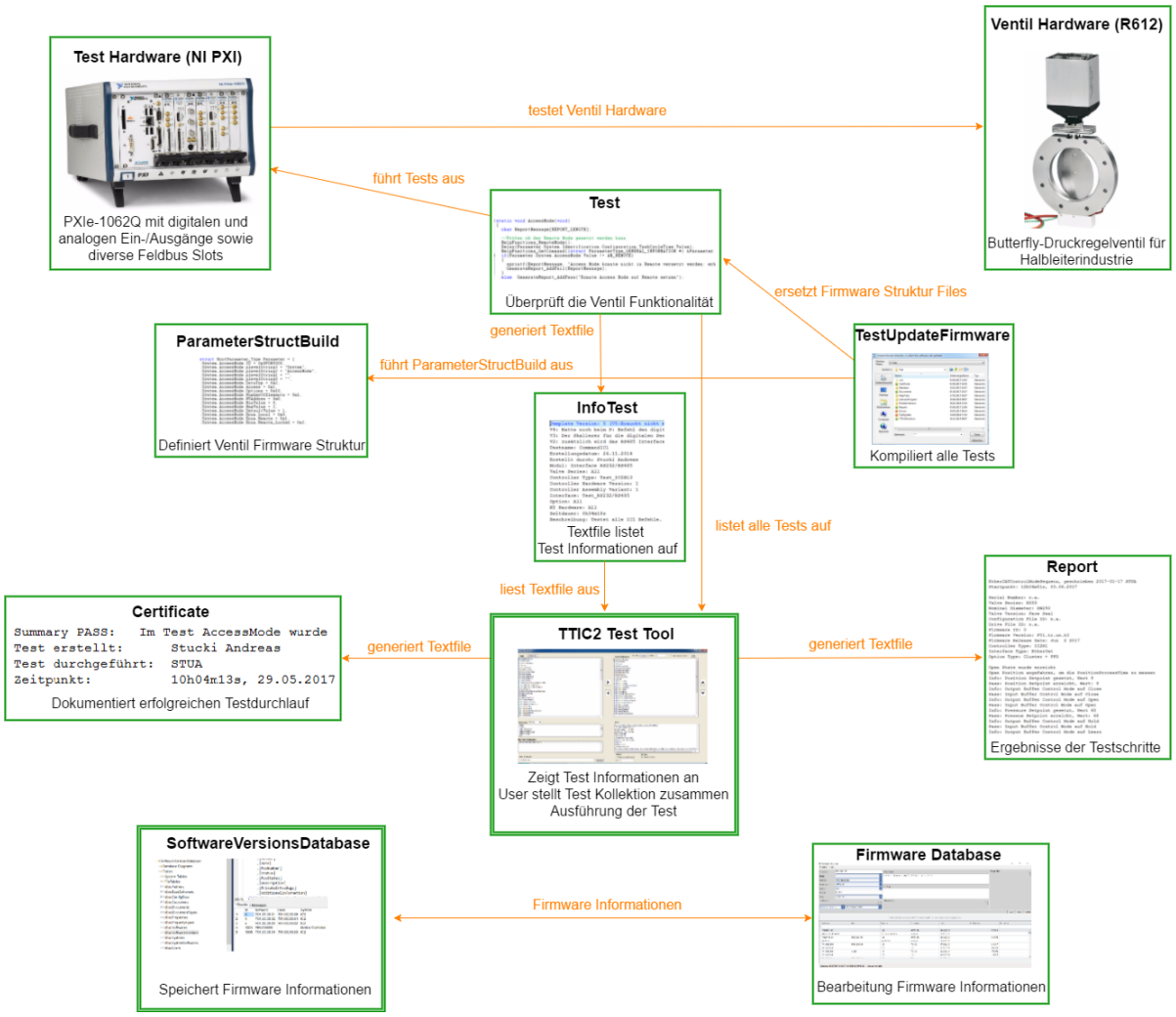


Abbildung 3: Ist-Zustand der Testumgebung

2.2.1 Ventil Hardware

2.2.1.1 Ist-Zustand

Mit Hilfe der Testumgebung werden unterschiedliche Ventil Hardware getestet. Dabei wird der Controller der jeweiligen Ventil Hardware Anforderungen angepasst. Diese beinhaltenen Unterschiede in der Bestückung des Masterboards zum Beispiel an 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 gleich wie auch 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 Ventil Hardware unterstützt. Dies hat zur Folge, dass unterschiedliche Controller Ausführungen erstellt werden müssen. Um diesem Um-

stand 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. Einerseits hat jeder Controller 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. 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, mehrere Ventile mit einer internen entwickelten Cluster Option des Ventils oder über die RS485 Interface Schnittstelle zu tun.

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. Wie 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.

All diese Funktionalitäten müssen mit Hilfe der Test Hardware emuliert und geprüft werden können. Dabei bietet National Instruments (NI) 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 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 einen Umweg eines Drittanbieter Produktes 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 8 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 noch nicht alle Steckplätze mit Karten ausgestattet sind, kann es noch zu Erweiterungen kommen. Der Grund 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

Aus persönlichen Programmiererfahrungen 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. 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, was zur Folge hat, dass der Anwender nicht mehr aktiv in die Ausführung eingreifen muss. Die Tests werden bei der zugehörigen Ventil Firmware in der SVN 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.

Im Test wird mit Parameternamen gearbeitet, um dem 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 hinterlegt sowie seine Enum Werte

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, 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.

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, welche den Inhalt der Struktur füllt. 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. Anschliessend werden die einzelnen Ebenen in Textform definiert. Den Datentyp, der Zugriff sowie den Wertebereich des Parameters wird ausgelesen. Dazu enthalten einige Parameter Enum Werte, welche den Parameterwert in Textform beschreibt. Diese wird auch in dieser Strukturform 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 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

2.2.4.4 Problemstellung

Durch das das Programm sehr weit ausgereift ist, gibt es prinzipiell kein Problem ausser kleinere Anpassungen, wie z.B. die Feldbus Objekte zu integrieren. Dies wird zum entsprechenden Zeitpunkt realisiert, wenn 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 dem 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 wurden, erscheint ein Fenster mit der Auflistung der Tests, welche nicht ohne Fehler 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 nicht ohne Fehler kompiliert werden konnten

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

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

Vor der Ausführung

- Mit Angabe des Pfades werden alle Tests aufgelistet, die mit der aktuellen Controller Generation lauffähig sind, siehe Abbildung 4
- Aus dieser Auflistung können die gewünschten Tests ausgewählt werden

- Es gibt verschiedene Filtermöglichkeiten, welche die Testauswahl anpassen (Mehrfachfilter möglich)
- Die aktuelle Testkollektion kann abgespeichert oder eine zuvor gespeicherte kann geladen werden
- Weiter können spezifische Testeinstellungen wie beispielweise «soll der Test bei einem Fehler abgebrochen werden», «wie viele Informationen soll der Report liefern» vor der Ausführung definiert werden

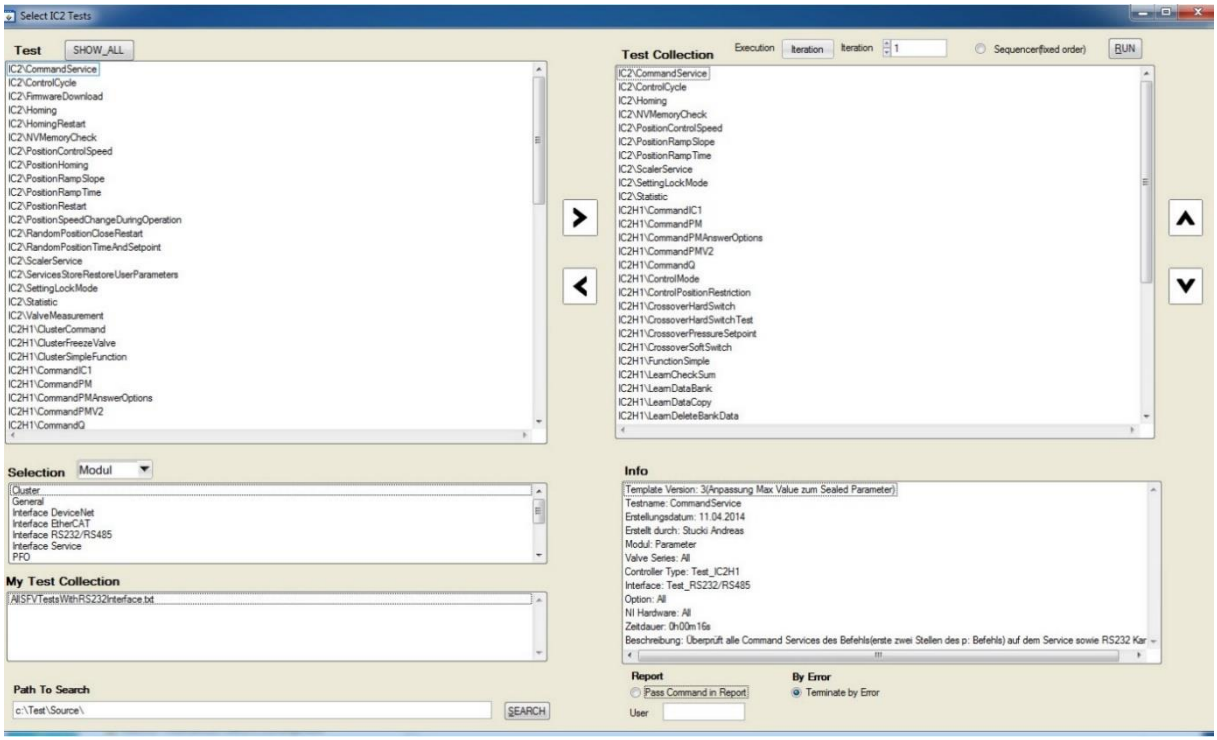


Abbildung 4: Ansicht der TTIC2 Oberfläche für die Auswahl der Testkollektion

Während der Ausführung

- Während die Testkollektion abläuft, zeigt das Reportfenster, den aktuellen Test sowie die bereits ausgeführten und die anstehenden siehe Abbildung 5
- Weiter wird jedes Testergebnis notiert
- Wird ein Fehler detektiert, so wird diese Fehlermeldung rot hervorgehoben
- Der User kann den Ablauf der Testkollektion abbrechen

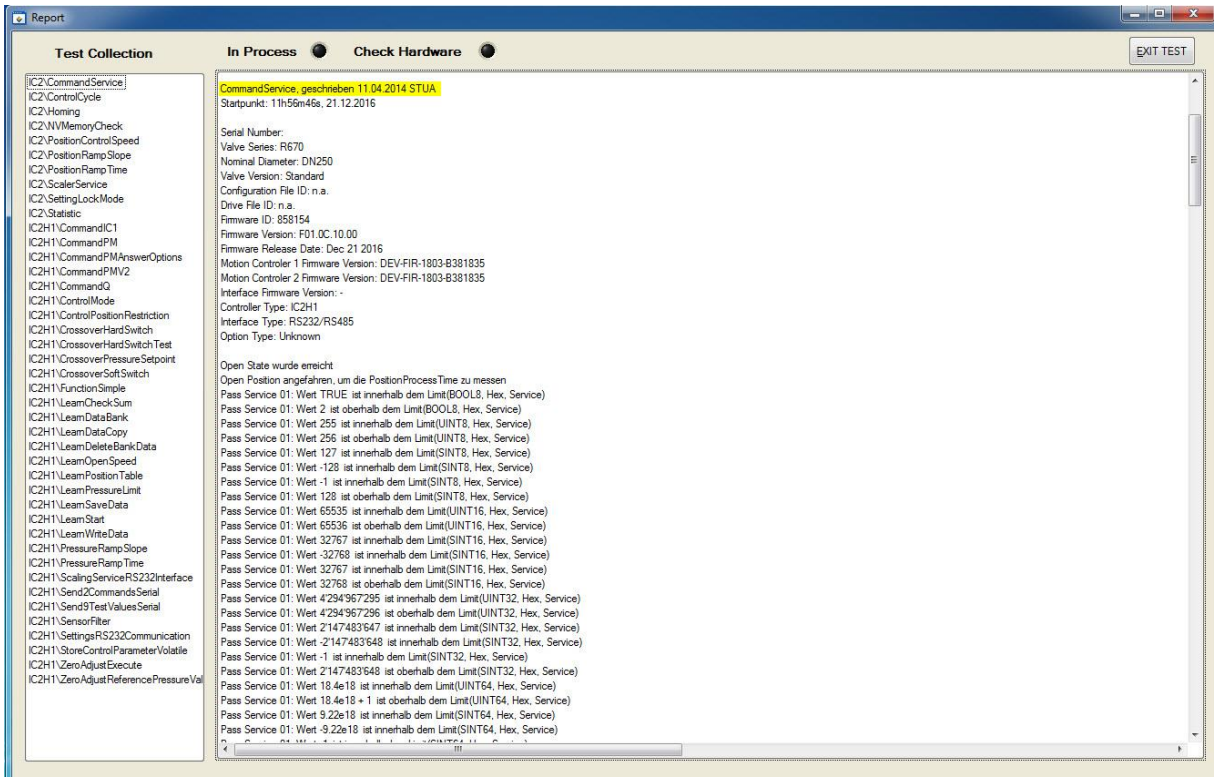


Abbildung 5: Report Ansicht währendem die Tests ausgeführt werden

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.

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 Text-file 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
- Auslesen der Testergebnisse geschieht im Reportfile

2.2.6.4 Problemstellung

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 und zeitaufwändig nach fehlerhaften Testdurchläufen durchsucht werden muss. Das Reportfile enthält zudem alle Testschritte und erreicht dadurch eine sehr grosse Datenmenge. Aus diesem Grund wird nur der letzte Report bei einer Ventil Firmware Freigabe in der SVN abgelegt. Dies erschwert die Auswertung der Tests enorm. Zusätzlich werden auch Zertifizierungsfiles abgelegt, diese erhöhen die Datenmenge weiter und kommen deshalb nochmals erschwerend hinzu.

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.

2.2.7 SoftwareVersionsDatabase

2.2.7.1 Ist-Zustand

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 hinterlegt, wie z.B. Ventil, Motion Controller sowie auch Interface. Dabei wird auch definiert, welche Ventil Firmware mit welchem Motion- und Interface Firmwaren 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

2.2.7.4 Problemstellung

Die Datenbank enthält aktuell die 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 welcher die Firmware Informationen gelesen und bearbeitet werden. 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.

2.2.8.2 Stärken

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

2.2.8.3 Schwächen

- Noch keine Benutzerverwaltung (Diese gebraucht, um gewisse Informationen zu verstecken)

2.2.8.4 Problemstellung

Die Applikation übernimmt aktuell Funktionen der Datenbankmodellierung. Weiter werden auch die Datenbanktypen in der Applikation verwendet.

3 Zielsetzung

Im ersten Unterkapitel 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 ist im Kapitel 2.2.2.4 näher beschrieben. Somit fiel der Entscheid die Arbeit auf die gesamte Testumgebung zu erweitern.

3.1 Ziele zum Startzeitpunkt

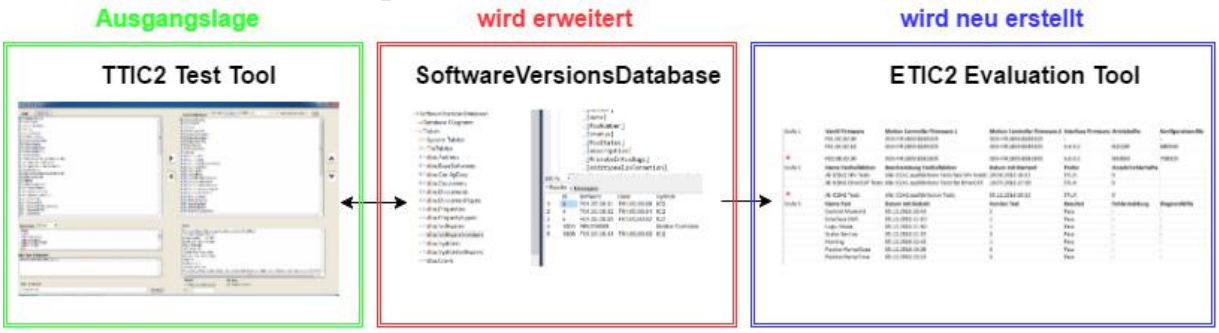


Abbildung 6: Ursprüngliches Konzept Masterarbeit

Zielsetzung nach Stand der Disposition besteht darin die bestehende 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 Database Oberfläche erstellt.

Das Hauptaugenmerk der Arbeit liegt in der Erstellung des Evaluation Tool Integrierter Controller 2 (ETIC2). Die Applikation gibt die generierten Testresultate vom TTIC2 wieder. 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.

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 z.B. geschaut werden, ob eine Firmware Version mit mehreren Motoren Firmwaren geprüft wurde und somit mehrere Versionen kompatibel ist.

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 immer aktuell sind. Als Weiteres wurde auch 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 sein muss eine lauffähige Testumgebung als Resultat der Arbeit liefern zu können. 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.

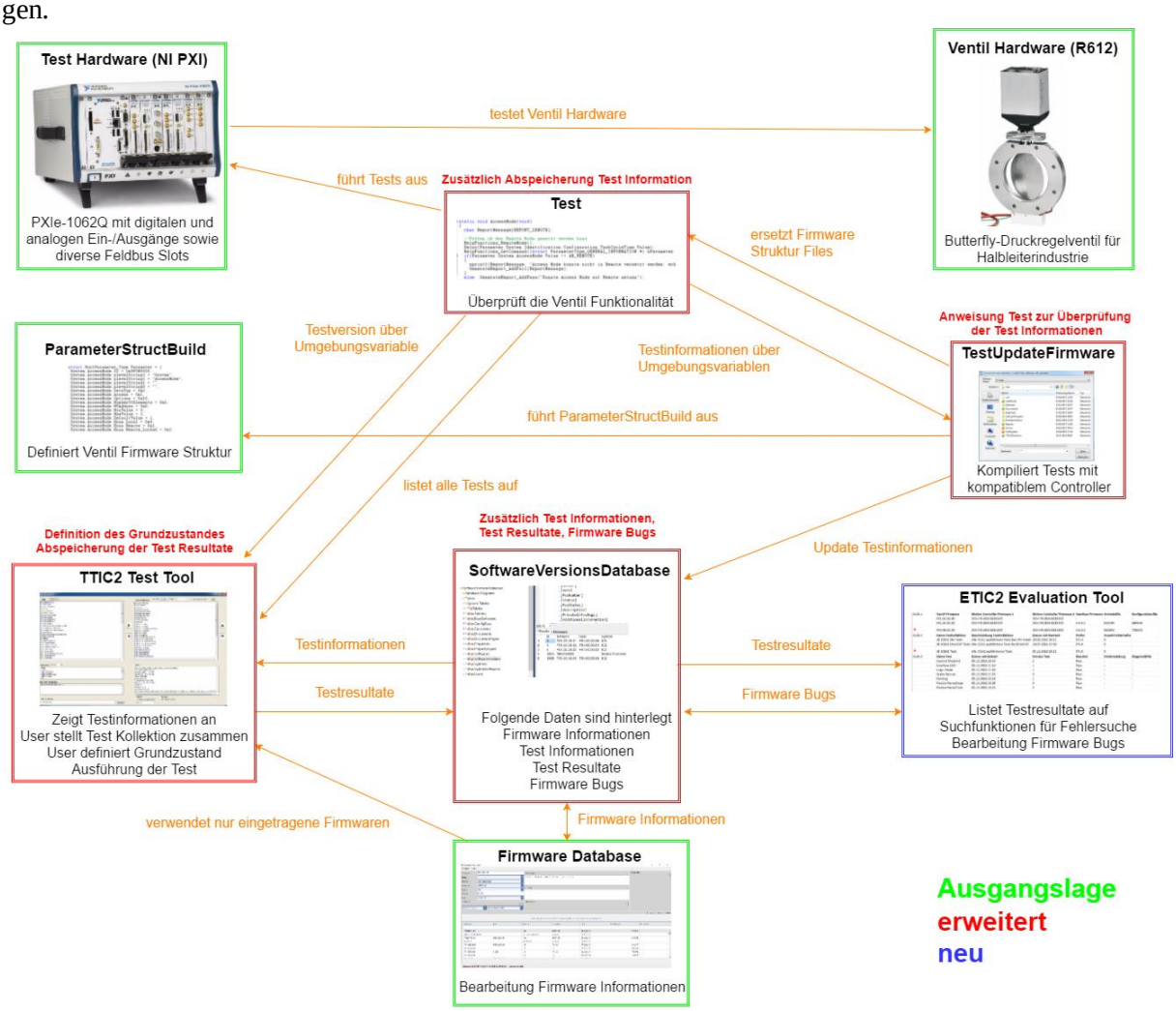


Abbildung 7: Konzept Masterarbeit Testumgebung komplett

Die Testinformationen bezüglich des Tests sowie die Anforderungen an die Ventil Hardware soll 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, bevor die Tests ausgeführt werden, aktualisiert. 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. Weiter soll ein Report zu einem bestimmten Grundzustand mit allen Testresultaten per Knopfdruck erzeugt werden können.

3.3 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.

3.4 Qualitative Ziele

- Das TTIC2 soll für die Anzeige der Testinformationen immer auf die aktuellsten im Test definierten Beschreibungen zugreifen, sowie auf die Auswertung, ob die angeschlossene Ventil Hardware für den Test ausreicht. (Bei einem Release Prozess)
- Das TTIC2 muss vor Ausführung der Test Kollektion die definierten Firmwaren und Parameterfiles (Drive- und Configuration File) automatisch die Ventil Hardware für den Grundzustand updaten.
- 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.~~

3.5 Aufgabenbegrenzung

- Die Ventil Hardware ist als Verständnishilfe im Konzept zu finden und nicht Teil der Arbeit
- Die Test Hardware wird verwendet um die Ventil Hardware anzusprechen, ist aber nicht Teil dieser Arbeit.
- Durch das ParameterStructBuild Programm können die Parameter Informationen der Firmware ausgelesen werden, was den Anforderungen entspricht.
- 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 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, was aktuell ausreicht.
- ETIC2 wird für den internen VAT Verwendungszweck entwickelt und nicht für den kommerziellen Gebrauch konzipiert.

4 Auswertung der Testresultate (Beschreibung der Arbeit)

Das Hauptkapitel, welches die Umsetzung der Arbeit dokumentiert, ist einerseits in Unterkapitel der einzelnen betroffenen Testumgebungselemente chronologisch unterteilt. In der nächsten Ebene werden die grösseren Arbeitsschritte separat aufgelistet. In der untersten Ebene wird nach der Methodik gleich noch die konkrete Umsetzung beschrieben.

4.1 SoftwareVersionsDatabase

Bei der Modellierung wird das Tool MySQL Workbench eingesetzt. Damit 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 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.

4.1.1 Modellierung Firmware Informationen

4.1.1.1 Methodik

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 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. D.h. es gibt nur Einträge, wenn es sich beim aktuellen Firmware Eintrag um eine Ventilfirmware handelt.

4.1.1.2 Umsetzung

In der Abbildung 8 ist das aktuelle SoftwareVersionsDatabase Modell ersichtlich.

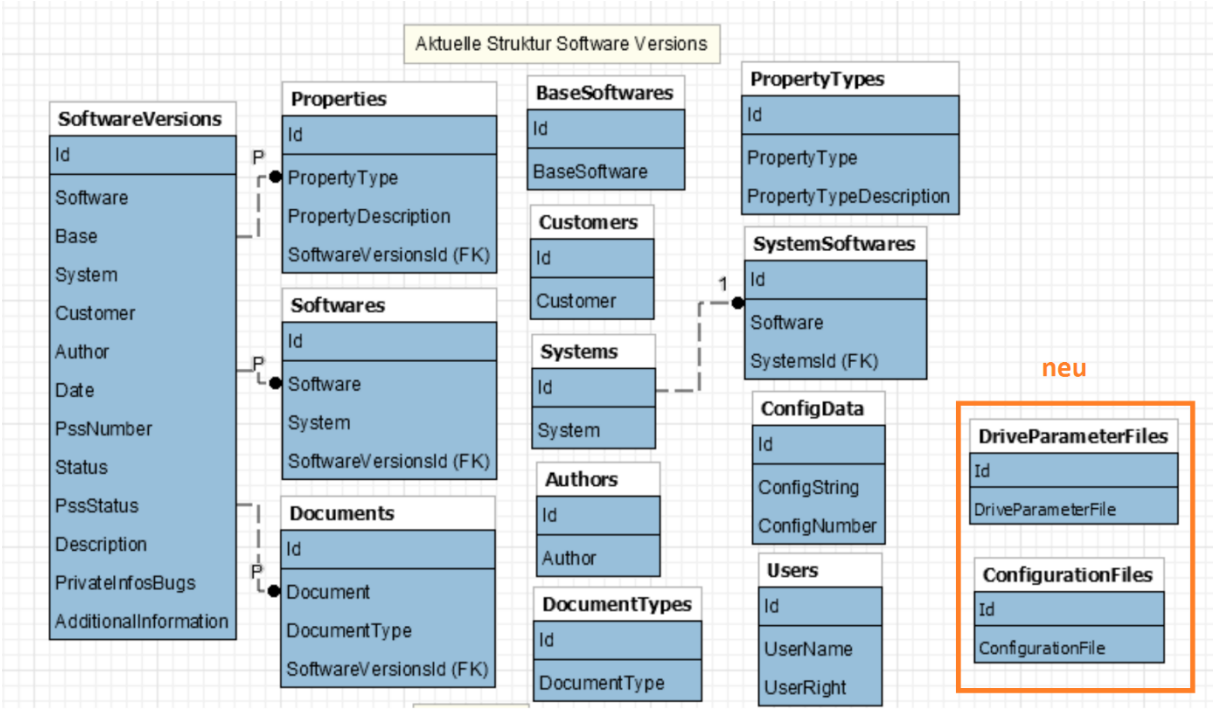
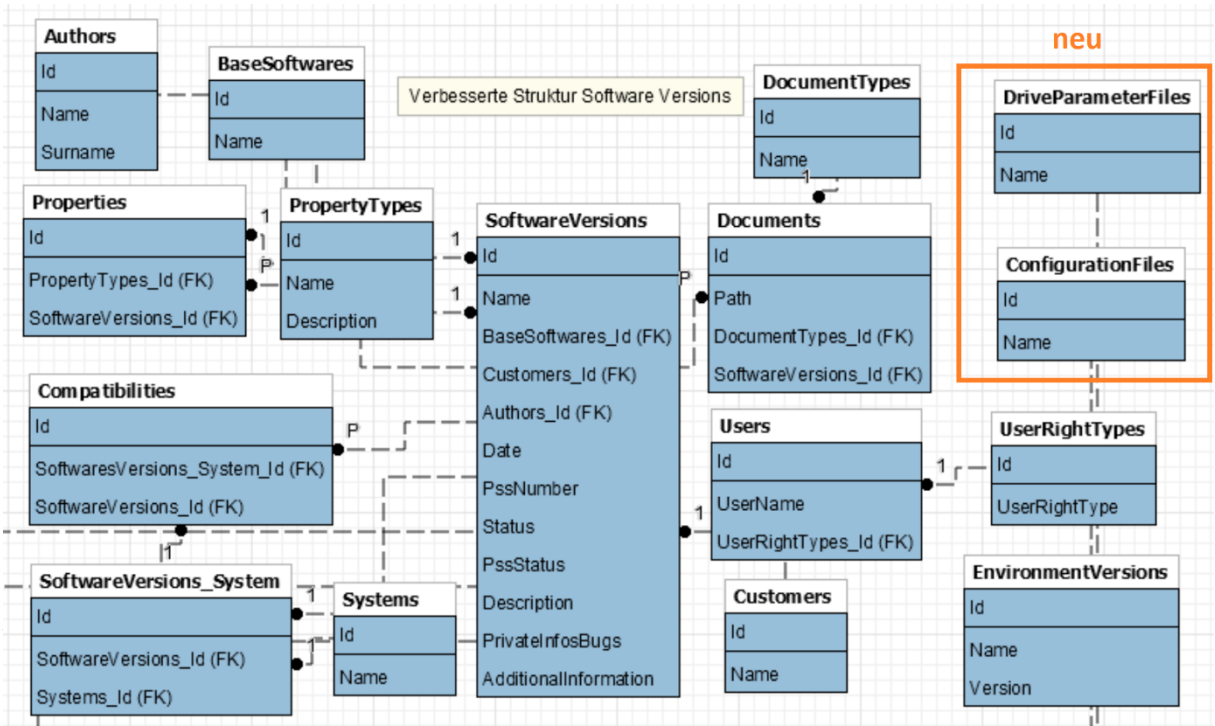


Abbildung 8 Aktuelle SoftwareVersionsDatabase Struktur

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 Tests 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 Abbildung 9 ersicht-



lich ist.

Abbildung 9: 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 auf die Namensgebung der einzelnen Attribute. Der Tabellennamen 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 dieses Modell auszuführen, muss die Oberfläche des Firmware Verwaltungstool angepasst werden. Dies kann aus Zeitgründen nicht weiterverfolgt werden. Somit wird in dieser Arbeit noch mit dem ursprünglichen Modell weitergearbeitet. Unter Kapitel 7.1 ist beschrieben, wie später der Umbau auf das neue Modell realisiert werden soll.

4.1.2 Modellierung Testinformationen

4.1.2.1 Methodik

Die SoftwareVersionsDatabase wird nun erweitert um die Testinformationen abspeichern zu können. Dabei handelt es sich einerseits um Informationen, die den Test näher definieren. Diese sind:

- Name
- Beschreibung
- Erstellungsdatum
- Autor

Anderseits 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

4.1.2.2 Umsetzung

Die Umsetzung der Modellierung bezüglich Testinformationen ist in Abbildung 10 ersichtlich.

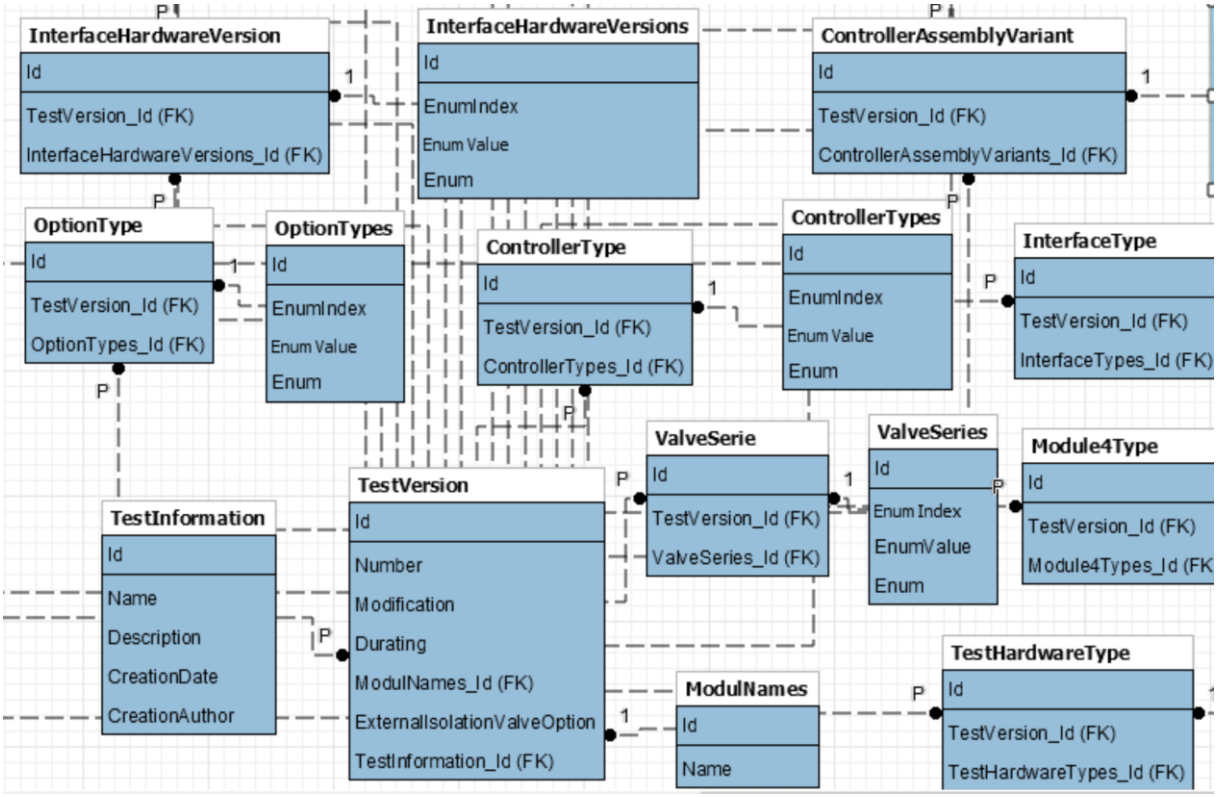


Abbildung 10: Erweiterung SoftwareVersionsDatabase zur Speicherung der Testinformationen

Die Haupttabelle ist die TestInformation, in welcher es genau einen Eintrag für jeden Test gibt. Hier sind zudem die generellen Daten des Tests hinterlegt. Erweitert wird das Modell mit den einzelnen Versionen eines Tests. So kann 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.

Zudem 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 der Modul Information kann es sein, dass mehrere Einträge einer Hardware Anforderung definiert werden sollen. Wie z.B. 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 kann mit Hilfe eines Wertes mehrere Einträge definiert werden.

4.1.3 Modellierung Testresultate

4.1.3.1 Methodik

Das ETIC2 zeigt die Testresultate zu den einzelnen Ventilfirmwaren. Hierbei sind die Ventilfirmware Informationen von Interesse bei der Auswertung. Dazu sind die Testresultate in derselben Datenbank zu finden. Weiter wird erreicht, dass nur eingetragene Firmwaren 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
- Testversion
- Testresultat
- 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.

4.1.3.2 Umsetzung

In der Abbildung 11 ist die Erweiterung der SoftwareVersionsDatabase zu sehen, um die Testresultate abzuspeichern.

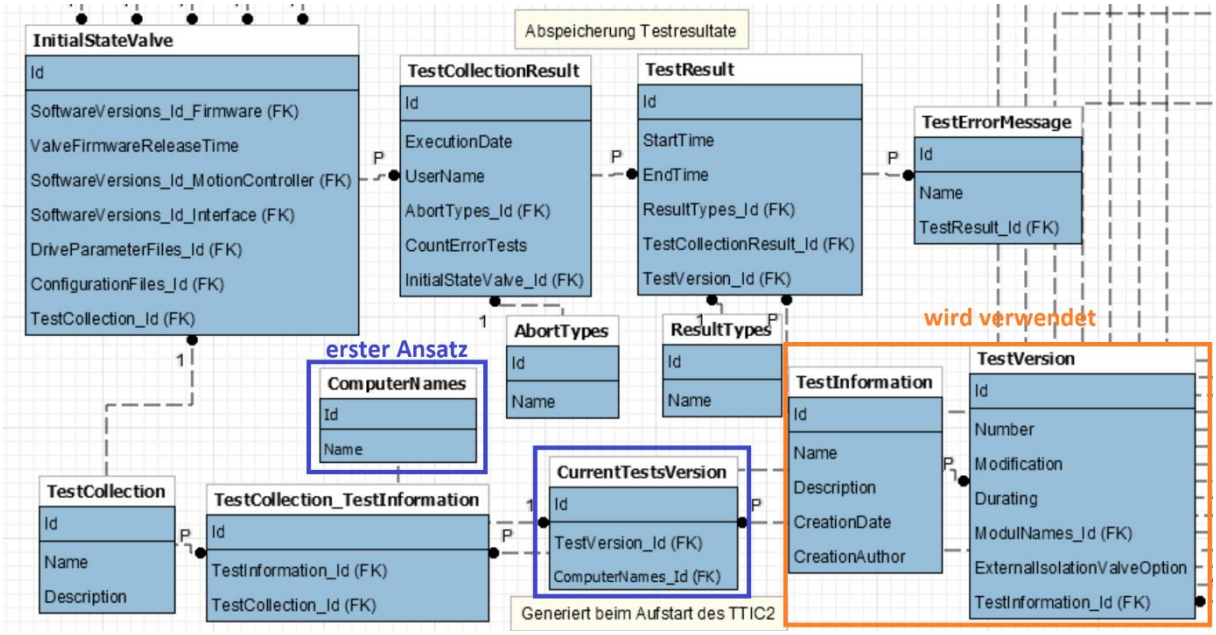


Abbildung 11: Erweiterung SoftwareVersionsDatabase zur Speicherung der Testresulate

Dabei übernimmt die InitialStateValve Tabelle die Funktion einen Grundzustand vor dem Start der Test Kollektion herzustellen. In der ETIC2 Applikation werden diese Informationen 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 diesem 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 Kollektion, die ausgeführt wurde, hinterlegt. Was der Anwender hier interessiert ist die Anzahl fehlerhaften Tests. Weiter ist auch ersichtlich, ob der User den Testablauf abgebrochen hat.

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 CurrentTestsVersion Tabelle. Diese wird bei jedem Start des TTIC2 Programms neu gefüllt. 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 4.3.1 näher beschrieben wird. Das TTIC2 löst diesen Umstand neu, dass jeder Test mit Hilfe einer Umgebungsvariable seine aktuelle Version liefert. Diese wird beim Aufstarten der Applikation in einer Liste hinterlegt. Durch diesen Lösungsweg wird die CuttentTestsVersions 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 den Zeitlauf des Tests können sich die Anforderungen verändern, da neue Testfunktionen hinzukommen können.

Im Modell sind auch die Testinformationen ersichtlich. Da es für die 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 kann. Die Testinformationen bezüglich Hardware Anforderungen wie auch Beschreibung des Tests werden in den einzelnen Tests definiert.

4.1.4 Modellierung Firmware Bugs

4.1.4.1 Methodik

Aus den erhaltenen Testresultate, welche im ETIC2 angezeigt werden, können Rückschlüsse auf Firmware Fehler gezogen werden. Diese sollen auch im ETIC2 verwaltet werden. 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

4.1.4.2 Umsetzung

Die Umsetzung der Firmware Bugs Modellierung ist aus der Abbildung 12: Erweiterung SoftwareVersionsDatabase zur Speicherung der Firmware Bugs

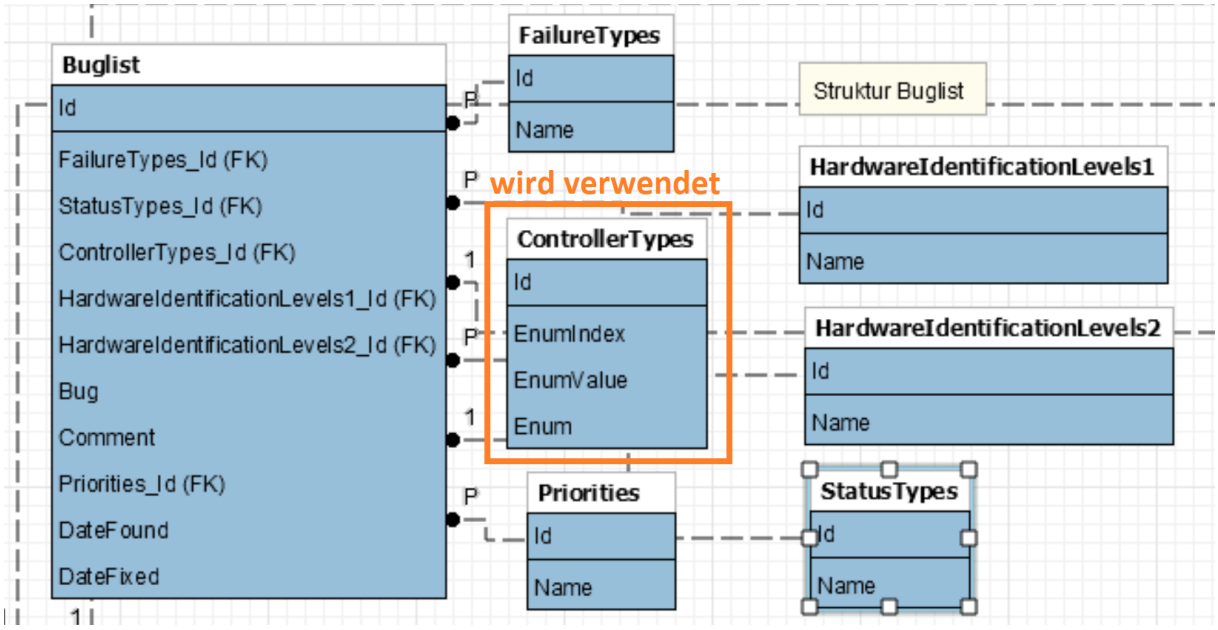


Abbildung 12: Erweiterung SoftwareVersionsDatabase zur Speicherung der Firmware Bugs

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 ist, noch offen oder das Fehlverhalten nicht behoben werden kann oder soll. Mit den ControllerType und HardwareIdentificationLevel1 und 2 Felder werden die verwendete Ventil Hardware gekennzeichnet bei welchem der Fehler aufgetreten ist. Neben dem Controller wird in der Hardware Identifikationslevel 1 zwischen dem verwendeten Interface, Option oder der Ventilreihe unterschieden. In der zweiten Ebene wird diese Eigenschaft konkretisiert. Im Falle der Ventilreihe, wird in der zweiten Ebene Einträge wie FaceSeal, SFV oder R655 eingetragen. Dabei wird dies nur eingetragen, wenn der Fehler nur in der definierten Ventilreihe auftritt. Weiter wird das Fehlverhalten genauer beschrieben, wie auch unter Bemerkungen ergänzende Informationen unter welchen Umständen der Fehler ausgelöst wird. 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 wird das Datum des ersten Auftretens des Fehlers eingetragen sowie, wenn der Fehler behoben werden konnte, das Datum an welchem der Fehler in der Software behoben werden konnte.

4.2 Test

Wie schon im Grundlagen Kapitel erwähnt, sind die Tests in CVI realisiert worden. Diese Programmierumgebung bietet ein SQL Toolkit an, mit welchem ein einfacher Zugriff auf die SQL Datenbank ermöglicht wird. Der ursprüngliche Ansatz sah vor, das in C# mit dem Entity Framework der Zugriff auf die Datenbank erfolgen soll. Der Nachteil dieser Lösung liegt in der Schnittstellendefinition, um dies zu vermeiden wird nun das SQL Toolkit eingesetzt.

4.2.1 Abspeicherung Testinformationen

4.2.1.1 Methodik

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. Anderseits sind dort die Anforderung an die Ventil Hardware hinterlegt. Die detaillierten Elemente der Testinformationen sind im Kapitel 4.1.1 ersichtlich.

Das Ziel des nächsten Abschnittes ist es den Funktionsumfang des SQL Toolkits vorzustellen. Die Abbildung 13 zeigt dabei die einzelnen Funktionen auf.

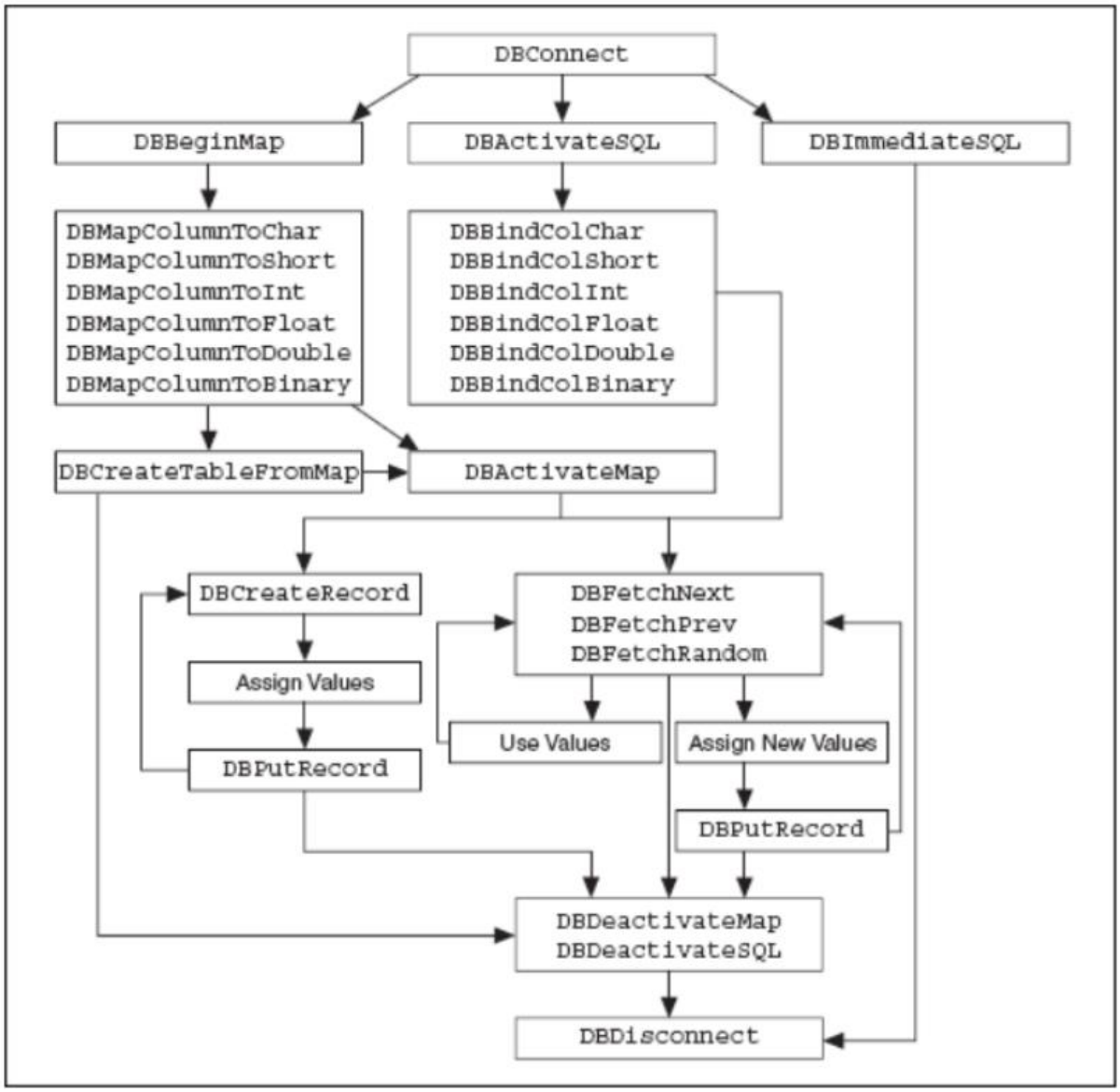


Abbildung 13: Funktionsumfang des SQL Toolkits

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.

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 ausgelesen werden und liegt dem Programm gleich zur Weiterverarbeitung bereit. Es ist nötig vor der Aus-

führung einer SQL Anweisung zuerst System Ressourcen anzulegen, welche nach der Ausführung wieder freigegeben werden. Hierbei handelt es sich um temporäre Files. (National Instruments Corporation, 2002)

4.2.1.2 Umsetzung

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

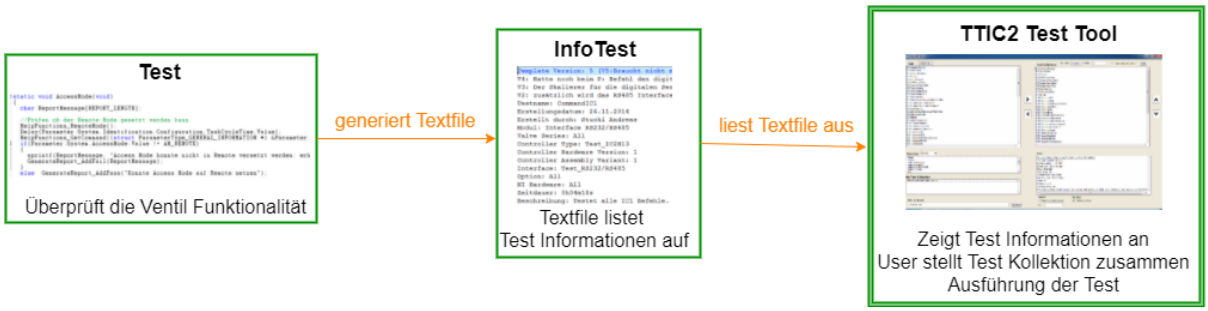
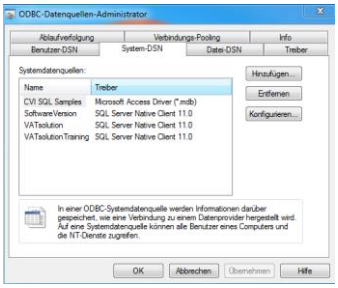


Abbildung 14: Ausgangslage Abspeicherung Testinformationen

Der Benutzer wählt beim Aufstarten des TTIC2 den Pfad aus, aus welchem die einzelnen Tests aufgelistet werden. Der Grund der Auswahl liegt darin, dass nicht immer die aktuellste Ventil Firmware getestet werden soll, sondern verschiedene Firmware Entwicklungen je nach Kunde. Nach dem Motto «never change a running system» und somit werden gewünschte Zusatzfunktionen vom letzten Firmware Stand aus implementiert.

Um die Testinformationen auszulesen, sucht das TTIC2 in den Testordnern 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.

Das Konzept für die Abspeicherung der Testinformationen mit Integration der SoftwareVersionsDatabase sieht vor, dass bei der Ausführung des ParameterStructBuild geschaut wird, ob sich die Testinformationen verändert haben. Somit wird sichergestellt, dass vor einem Release der Firmware die Testinformationen aktualisiert werden.



Im nächsten Abschnitt wird die Programmierung mit Hilfe des SQL Toolkit vorgestellt. Die erste Aufgabe, die Verbindung zur SoftwareVersionsDatabase erwies sich schwieriger als angenommen. Das Beispiel von Seite des SQL Toolkits siehe Abbildung 16, dies mit Hilfe eines DSN Eintrages, erwies sich nicht als praktikabel. Da dafür Admin Rechte nötig sind, um den Eintrag neu auf einem Rechner erstellen zu können, siehe Abbildung 15Abbildung 16.

Abbildung 15: Erstellung eines DSN Eintrages

```
/* Connect to database (in this case dBase files) */
hdbc = DBConnect ("DSN=CVI SQL Samples");
```

Abbildung 16: Beispiel Datenbankverbindung mit Hilfe eines DSN Eintrages

Die Verbindung mit der Datenbank kann auch mit Angabe eines Connection Strings aufgebaut werden. (SQL Server 2012 connection strings, o. J.) Im nächsten Schritt werden Funktionen erstellt, um

```
string[200] = "DRIVER={ODBC Driver 11 for SQL Server};Server=chva0012;Database=SoftwareVersionsDatabase;Trusted_Connection=yes;";

/* Connect to database (in this case dBase files) */
hdbc = DBConnect (string);

/*Einträge auslesen*/
hstmt = DBActivateSQL (hdbc, "SELECT * FROM Authors");
if (hstmt > 0)
{
  resCode = DBBindColInt (hstmt, 1, &Id, &IdStat);
  resCode = DBBindColChar (hstmt, 2, authorLen, Author, &authorStat, "");
  if (resCode != DB_SUCCESS) {ShowError(); goto Error;}
  while ((resCode = DBFetchNext (hstmt)) == DB_SUCCESS)
  {
    printf("Id: %d, Author: %s\n", Id, Author);
    Delay(1);
  }
}
```

die Testinformationen in die Datenbank zu schreiben, welche im Kapitel 4.1.2.2 in der Modellierung beschrieben sind. Dabei wird prinzipiell mit der Explicit SQL Variante gearbeitet um Informationen aus der Datenbank auszulesen, welche im Code Ausschnitt Abbildung 17 näher gezeigt wird.

Abbildung 17: Code Ausschnitt einer Select Anweisung

In der ersten Codezeile ist der Truted Connection String für den SQL Server zu sehen. Nachdem eine Verbindung mit der Datenbank aufgebaut wurde wird mir dem DBActivateSQL einen Select Befehl auf die Authors Tabelle ausgeführt. Zu der Abfrage wird die Id der Tabelle sowie der Autor Namen gebunden um später die dazugehörigen Tabellenwerten zu erhalten. Mit dem DBFetchNext Befehl kann jeder einzelne Eintrag der Tabelle durchlaufen werden, bis keinen Eintrag mehr vorhanden ist. Mit dem dem DBDeactivateSQL werden alle Ressourcen wieder freigegeben und mit dem DBDisconnect Anweisung wird die Verbindung zur Datenbank getrennt.

4.2.2 Testinformationen an TestUpdateFirmware über Umgebungsvariablen

4.3 TestUpdateFirmware

4.3.1 Anweisung Update Testinformationen

4.3.2 Abfrage Testinformationen

4.3.3 Testinformationen in SoftwareVersionsDatabase schreiben

4.4 TTIC2

Methodik

Abbildung 18: 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.

Umsetzung

Das TTIC2 ist wie schon in Kapitel 2.2 beschrieben in CVI realisiert. In der Disposition war angedacht die Anbindung mit Hilfe des Entity Framework zu realisieren. Der Grund dies mit Hilfe des SQL Toolkits zu realisieren ist bereits im Kapitel 4.2.1.2 erwähnt worden.

4.4.1 Auslesung der Testinformationen

4.4.2 Hinterlegung des Grundzustandes

4.4.3 Abspeicherung der Testresultate

4.5 ETIC2

4.5.1 Design View Model

4.5.2 Codierung nach MVVM

4.5.3 Anbindung SoftwareVersionsDatabase

4.5.4 Ausgabe Bericht

5 Ergebnisse (Tool)

6 Diskussion (Was hat Funktioniert, was nicht)

7 Ausblick (offene Punkte, wie geht es weiter)

7.1 Umsetzung Überarbeitung SoftwareVersionsDatabase

Wie unter Kapitel 4.1.1.2 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.

7.2 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

Marugg, L. (03. 04 2010). PG_Info_Hardware. VAT Interne Präsentation. Haag.

National Instruments Corporation. (01 2002). Abgerufen am 17. 05 2017 von LabWindows/CVI SQL Toolkit Reference Manual: http://www.ni.com/pdf/manuals/370502a.pdf

(o. J.). Abgerufen am 17. 05 2017 von SQL Server 2012 connection strings: https://www.connectionstrings.com/sql-server-2012/

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

Abkürzungsverzeichnis

NI	National Instruments

Abbildungsverzeichnis

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.
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
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: …….....