HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

COMPUTER SCIENCE

TECAN.

.NET Qualification-Tool for Pipet-Instruments

Bachelor Thesis

Department of Computer Science
University of Applied Science Rapperswil

Spring Term 2012

Author(s):	Andreas Zollinger
Advisor:	Prof. Dr. Luc Bläser
Project Partner:	Tecan Schweiz AG, Männedorf
External Co-Examiner:	Dipl. Inf.-Ing. ETH Jean-Daniel Merkli
Internal Co-Examiner:	Prof. Dr. Andreas Steffen

TECAN.

Index of Contents

Part I - Management

The first part of this documentation contains all formalities of this project including the abstract and management summary.

- Aufgabenstellung
- Erweiterte Copyright-Erklärung
- Abstract and Management Summary

Part II – Documentation

The second part contains all the documentation done during the bachelor thesis. These documents are all based on the Tecan SOP.

- Project Development Plan
- Product Requirements Document
- Software Specification
- Use Case Specification
- Software Structure Design (Architecture)
- Software Graphical User Interface Design
- Software Detail Design
- Test Case Plan and Reports

Part III – Appendix

Additional documents are placed in the appendix.

- Project Conclusion
- Traceability Matrix
- Definitions, Acronyms and Abbreviations
- Poster

TECAN

Part I - Management

The first part of this documentation contains all formalities of this project including the abstract and management summary.

Aufgabenstellung Bachelorarbeit für Andreas Zollinger:

.NET Validierungs-Tool von Pipettier-Maschinen

1. Auftraggeber und Betreuer

Diese Bachelorarbeit findet in Zusammenarbeit mit der *Tecan Schweiz AG* statt.

Ansprechpartner Auftraggeber:

- Joas Leemann, Tecan Schweiz AG , Software Architekt, joas.leemann@tecan.com

Betreuer HSR:

- Prof. Dr. Luc Bläser, Institut für Software, lblaeser@hsr.ch

2. Ausgangslage

Die Tecan AG stellt unter anderem Pipettier-Robotermaschinen her, welche über verschiedene Robotermotoren verfügen. Zum Zeitpunkt der Entwicklung einer neuen Maschinenserie müssen die unterschiedlichen (z.B. Kraft, Offsets oder PID-Regler) Parameter der Motoren so bestimmt werden, dass die Robotersteuerung im produktiven Betrieb zuverlässig und mit der geforderten Dynamik und Präzision funktioniert.

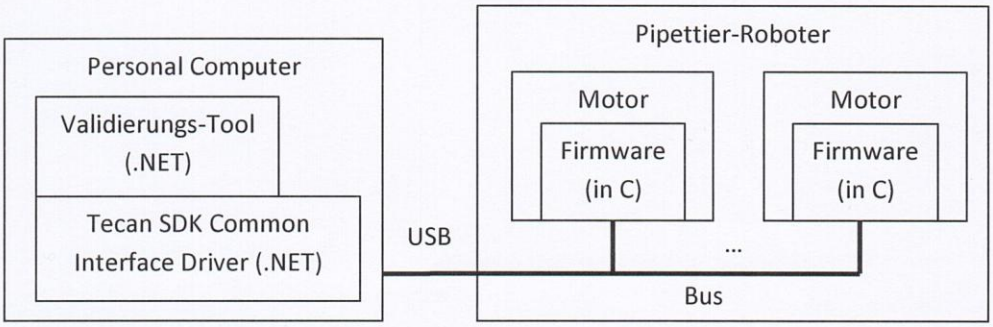
Zur Validierung der Parameterkonfiguration sind aufwendige Tests an den Maschinen notwendig, die mit einem Tool zu automatisieren sind. Das Validierungs-Tool ermöglicht folgende Vorgehensweise:

1) Ausgehend von einer Input-Konfiguration richtet das Tool die Maschinen mit den definierten Parametern ein.
2) Danach werden automatisierte Tests ausgelöst, bei denen Bewegungen in verschiedenen Varianten(schnell, langsam, kurz, lang etc.) auf verschiedenen Achsen und schliesslich verschiedenen Maschinen in wiederholten Testrunden durchgeführt werden.
3) Die verschiedenen Maschinen-Werte (Positionen, Ströme etc.) werden bei den Bewegungen kontinuierlich gemessenen und vom Tool gesammelt.
4) Das Tool wertet kontinuierlich die Messwerte aus und verarbeitet diese zu diversen zu Statistiken, die zeitgleich mit dem Testbetrieb dargestellt und von dem User mit den Achsen-Spezifikationen verglichen werden können.
5) Am Schluss oder während der Validierung können Reports und Aussagen generiert werden und exportiert werden.

Basierend auf den Resultaten der Validierung können Tester die Konfiguration überprüfen und allenfalls revidierte Konfigurationen in eine neue Testrunde schicken.

Zum jetzigen Zeitpunkt gibt es in Tecan bereits einige LUA-basierte Skriptlösung zur Parametrisierung, die jedoch nicht zu einem Tool integriert ist (diverse manuelle Schritte sind nötig) und auch nicht die produktiv relevante Umgebung (Tecan .NET Basisframework) für die Tests einsetzt.

Aus diesem Grund soll im Rahmen dieser Bachelorarbeit ein einheitliches Validierungs-Tool für Pipettier-Robotermaschinen auf der Technologiebasis von .NET (mit C# und WPF) entwickelt werden. Diese soll den gesamten Validierungs-Workflow wie oben beschrieben unterstützen. Das Tool verwendet dazu direkt das existierende .NET Treiberframework (Common Interface Driver) der Firma Tecan, welches direkt als .NET Assembly in das Tool eingebunden werden kann. Die Übergabe der Parameterdaten an den Treiber erfolgt über ein XML-Format.



Situierung des .NET Validierungs-Tool

3. Ziele und Aufgabenstellung

Die Aufgabe dieser Arbeit ist es, ein .NET-basiertes Tool zur automatisierten Validierung von Parameterkonfiguration auf Tecan Pipettier-Robotermaschinen zu implementieren. Der Funktionalitätsumfang orientiert sich an das existierende LUA Skripte sowie an die Anforderungen der Benutzer.

Folgende spezifische Ziele werden vorgegeben:

- Aufnahme der genauen Anforderungen für das neue Validierungs-Tool.
- Entwurf des GUI- und Bedienkonzepts in Absprache mit den Parametrisierungs-Spezialisten (Screenshots, Abläufe, wenn nötig mit Prototyp).
- Entwicklung des Validierungs-Tools in C# auf Basis von .NET 4.0, dass die gesamte Validierungs-Workflow unterstützt. Das neue GUI soll in .NET WPF implementiert werden. Die Funktionalität umfasst insbesondere die automatisierte Einstellung der Maschinen, Starten und Überwachen der Tests, Sammlung und Auswertung der Messwerte, Online-Visualisierung von Statistiken und Werten, Generierung und Export von Reports der Messresultate.

4. Zur Durchführung

Mit dem HSR-Betreuer finden in der Regel zweiwöchentliche Besprechungen statt. Zusätzliche Besprechungen sind nach Bedarf durch die Studierenden zu veranlassen. Besprechungen mit dem Auftraggeber werden nach Bedarf durchgeführt.

Alle Besprechungen sind von den Studenten mit einer Traktandenliste vorzubereiten und die Ergebnisse in einem Protokoll zu dokumentieren, das dem Betreuer und dem Auftraggeber per E-Mail zugestellt wird.

Für die Durchführung der Arbeit ist ein Projektplan zu erstellen. Dabei ist auf einen kontinuierlichen und sichtbaren Arbeitsfortschritt zu achten. An Meilensteinen gemäss Projektplan sind einzelne Arbeitsresultate in vorläufigen Versionen abzugeben. Über die abgegebenen Arbeitsresultate erhalten die Studierenden ein vorläufiges Feedback. Eine definitive Beurteilung erfolgt auf Grund der am Abgabetermin abgelieferten Dokumentation.

5. Dokumentation

Über diese Arbeit ist eine Dokumentation gemäss den Richtlinien der Abteilung Informatik zu verfassen (siehe https://www.hsr.ch/Allgemeine-Infos-Diplom-Bach.4418.0.html?&L=0). Die zu erstellenden Dokumente sind im Projektplan festzuhalten. Alle Dokumente sind nachzuführen, d.h. sie sollten den Stand der Arbeit bei der Abgabe in konsistenter Form dokumentieren. Die Dokumentation ist vollständig auf CD/DVD in 3 Exemplaren abzugeben. Auf Wunsch ist für den Auftraggeber eine gedruckte Version zu erstellen.

6. Termine

Siehe auch Terminplan auf https://www.hsr.ch/Termine-Diplom-Bachelor-und.5142.0.html?&L=0

20.02.12	Beginn der Bachelorarbeit, Ausgabe der Aufgabenstellung durch die Betreuer.
Mai 2012	Fotoshooting. Genauere Angaben erteilt die Kommunikationsstelle rechtzeitig.
08.06.12	Die Studierenden geben den Abstract für die Diplomarbeitsbroschüre zur Kontrolle an ihren Betreuer/Examinator frei. Die Studierenden erhalten vorgängig vom Studiengangsekretariat die Aufforderung und die Zugangsdaten zur Online-Erfassung des Abstracts für die Broschüre. Die Studierenden senden per Email das A0-Poster zur Prüfung an ihren Examinator/Betreuer. Vorlagen sowie eine ausführliche Anleitung betreffend Dokumentation stehen unter den allgemeinen Infos Diplom-, Bachelor- und Studienarbeiten zur Verfügung.
13.06.12	Der Betreuer/Examinator gibt das Dokument mit dem korrekten und vollständigen Abstract für die Broschüre zur Weiterverarbeitung an das

	Studiengangsekretariat frei.
15.06.12	**Abgabe des Berichtes an den Betreuer bis 12.00 Uhr.**
	Fertigstellung des A0-Posters bis 12.00 Uhr.
15.06.12	**HSR-Forum, Vorträge und Präsentation der Bachelor- und Diplomarbeiten, 13 bis 18 Uhr**
06.08. - 25.08.12	Mündliche BA-Prüfung
28.09.12	Nachmittag Bachelorfeier und Ausstellung der Bachelorarbeiten

7. Beurteilung

Eine erfolgreiche Bachelorarbeit zählt 12 ECTS-Punkte pro Studierenden. Für 1 ECTS Punkt ist eine Arbeitsleistung von ca. 25 bis 30 Stunden budgetiert. Für die Modulbeschreibung der Bachelorarbeit siehe auch https://unterricht.hsr.ch/staticWeb/allModules/19419_M_BAI.html.

Für die Beurteilung sind die HSR-Betreuer verantwortlich.

Gesichtspunkt	Gewicht
1. Organisation, Durchführung	1/6
2. Berichte (Abstract, Mgmt Summary, technischer u. persönliche Berichte) sowie Gliederung, Darstellung, Sprache der gesamten Dokumentation	1/6
3. Inhalt *)	3/6
4. Mündliche Prüfung zur Bachelorarbeit	1/6

*) Die Unterteilung und Gewichtung von 3. Inhalt wird im Laufe dieser Arbeit mit den Studierenden festgelegt.

Im Übrigen gelten die Bestimmungen der Abt. Informatik zur Durchführung von Studienarbeiten.

Rapperswil, den 20. Februar 2012

Der verantwortliche Dozent

Prof. Dr. Luc Bläser
Institut für Software
Hochschule für Technik Rapperswil

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

Erweiterte Copyright-Erklärung

Ich/wir erkläre(n) hiermit,

- dass ich/wir die vorliegende Arbeit selber und ohne fremde Hilfe durchgeführt habe(n), ausser derjenigen, welche explizit in der Aufgabenstellung erwähnt sind oder mit dem Betreuer schriftlich vereinbart wurden,
- dass ich/wir sämtliche verwendeten Quellen erwähnt und gemäss gängigen wissenschaftlichen Zitierregeln korrekt angegeben habe(n).
- das ich/wir keine durch Copyright geschützten Materialien (z.B. Bilder, Messdaten) in dieser Arbeit in unerlaubter Weise genutzt habe(n).

Ort, Datum:

Stäfa, 11.06.2012

Name, Unterschrift:

Abstract and Management Summary

Project-Name: **Qualification Tool**

Project Number: **-**

Subject: **-**

Abstract

Tecan Schweiz AG develops various instruments and devices to support the daily tasks in a laboratory. Many of these instruments have motor-driven parts which move items (water tubes, tablets of wells etc.) from one station to another. During engineering, numerous motor control parameters need to be evaluated to program an optimal machine configuration. These motor control parameters have must be validated for their qualification. The current parameter qualification tool has some major shortcomings: Firstly, it is not implemented with standard Tecan C# library, the Tecan Base SDK; it is written in the less known scripting language LUA. Secondly, the current LUA script tool can only execute one test at a time. After each test the results have to be gathered and stored by hand. As this leads to unnecessary work that could be automated, Tecan aims to develop a new qualification tool in C# using the common Tecan Base SDK.

The task of this bachelor thesis is to determine the requirements and specifications of this new qualification tool and to design and implement a first version of the tool in C# and with the Tecan Base SDK.

In the first phase of the project, all the requirements for the new tool have been analyzed and formally defined. In the second phase, the essential functionality required for the tool has been implemented. As main approach to solve the problem the concept of a strategy files has been developed. Within a strategy XML file the different tests for different motors are defined. Inside the qualification tool the user can select one strategy and one or more motors he wants to test. The tool will then test all selected motors in the most parallel way possible till the user stops the process. It gathers all generated data and stores them. On a report view it presents a first overlook over the gained data.

As a result of the project, the first version of the qualification tool has been completed. Test runs can be executed and will gather data fully automated in an endless loop. Old test runs can be loaded and continued if the same instrument configuration still is present. In the future many features can be added, as example automated error recovery. Current implementations can be enhanced, as example the current gathered data is stored in a XML file. Storing them in a (online) database would be more performing.

Management Summary

Current Situation

Tecan Schweiz AG develops various instruments and devices to support the daily tasks in a laboratory. Many of these instruments have motor-driven parts which move items (water tubes, tablets of wells etc.) from one station to another. One of the main tasks of the motion control department is to determine parameters which guarantee top performance and a long life of these motors in each single use case. To get these so-called motion parameters this department has developed a great and well working tool environment. However, there is a tool which isn't optimal at all.
After the motion control team has determine a set of motion parameters they have to verify that these parameters fulfill certain specifications, as example that they don't use too much power or that the parameters are still good, even after thousands of moves. The motion control department currently uses a tool which can measure data and stores them very unproductive, as the tool only can execute one measuring at a time. This leading to the fact that one member of the team has to sit in front of an instrument the whole day and stupidly executes the same tasks all 30 seconds.
Unfortunately in addition to the unproductive way of doing the verification, the tools of the motion control department don't use the software made by the software department, which will be used in productive use.
That's why an automated tool which uses the Tecan Base SDK is highly needed. The tool will automatic start different tests on an instrument at the same time, collects all the data and evaluates the motion parameters on the fly.

Task and Approach

As the Tecan Base SDK shall be used, the new tool is written in C#. With help of this library communication with the instrument and its motor can be done without great effort.
To realize the main task of the new tool a concept called "strategy" is introduced. A strategy contains information of an axis. How the axis shall be initialized? What test runs can be executed with an axis? Are there dependencies and constrains for an axis? All this questions are answered in a strategy file. These files can be generated by a trained person. The new developed qualification tool can read in the strategy file and interprets it. The tool generates an automated test process that can run ad infinitum. The gathered data will be stored in a file and a first analysis is done on the fly. To support the motion control department even more the tool will do statistical calculation to better rate the motion parameters.

Conclusion

In the first phase of the project the requirements have been collected and formally defined. A paper prototype was used to involve the motion control department early in the project in the development of the graphical user interface. In the second phase, a first implementation of the tool has been design and implemented. The strategy concept worked and is capable of mirroring test ideas to a file format.
At the end of the project a first version of the tool has been completed. The tool is fully usable and can replace the current solution used by the motion control department. However there are missing components which are vital for an automated tool that shall run for hours. For example an error recovery has to be implemented.
Although it still misses some feature, the new qualification tool proves already that it is wanted, as other departments have also declared their interest for this tool. Be it for automated life cycle tests or for a burn in script replacement.

Outlook

As mentioned above, the new qualification tool still needs further implementation. There are also additional features that would improve this tool even more and would make it to one of the indispensable ones.

Additional features could be:

- Automated error recovery.
- Store data to a central data base instead.
- E-mail notification on certain events (number of moves reached, error occurred).
- More statistical calculations to support the verification process.
- Special test cases (measuring belt tension, recording noise).

TECAN

Acknowledgments

I would like to express my gratitude to …

- … the whole motion control team for their support with firmware and motion control advice. Especially in relation to the Paper Prototyping approach, even if it was maybe a little bit too “esoteric” for some.
- … the Tecan Base SDK team for their support with the SDK.
- … Remo Kälin and Bernhard Pfund for giving good input from the customer’s side.
- … Joas Leemann advisor and project leader of the qualification tool.
- … Prof. Dr. Luc Bläser from the Institute for Software (IFS), HSR Rapperswil, for supporting in many ways, especially with threading and code reviews.
- … my family and friend for supporting me during the bachelor thesis.

TECAN.

Part II - Documentation

The second part contains all the documentation done during the bachelor thesis. These documents are all based on the Tecan SOP.

1 Project Development Plan

The Product Development Plan defines the activities, the resources and quality practices for the development of the qualification tool.

2 Product Requirements Document

The purpose of the Product Requirements Document is to clarify who the persons in interests are and what they expect from the outcome of this project.

3 Software Specification

The Software Specification defines the behavior of the application or its subsystem.

4 Use Case Specification

The Use Case Specification document describes the main use cases of the qualification tool and how different users are supported.

5 Software Structure Design (Architecture)

The Software Structure Design describes the software architecture of the qualification tool. It comprises the structural design of the SW components and design decisions and conventions.

6 Software Graphical User Interface Design

The GUI design describes the look and behavior of the windows and screen. It further describes the connection between the user controls and the functions required in the SWS.

7 Software Configuration Management Plan

The purpose of the Software Configuration Management Plan is to describe the configuration management during the different phases of the SW Product Life Cycle.

8 Software Detail Design

The Software Detail Design establishes a lower level design of the software identified in the Software Structure Design and completes the software design documentation providing sufficient information from which programmers can code.

9 Test Case Plan and Report 1

10 Test Case Plan and Report 2

The Test Plan and Test Report documents shall offer the possibility to formally test a specific version of the qualification tool.

Project Development Plan

Project-Name: **Qualification Tool**

Project Number: **-**

Subject: **-**

	Author	Reviewer	Approver
Name	Andreas Zollinger	Luc Bläser	Joas Leemann
Function	Software	Supervisor HSR	Project Leader
Date / Visa			

Table of Contents

1 Introduction

1.1 Purpose of this Plan

This *Product Development Plan* defines the activities, the resources and quality practices for the development of the qualification tool. It is a 'living document' that has to be updated according to the ongoing project work.
The objectives for this product development plan include:

- Defining the development approach and strategy.
- Describing how design control requirements will be met
- Clarifying interacting responsibilities.
- Defining the quality practices ensuring that the product will meet the customers' expectations and that Tecan's quality standards will be met.

To get a total overview, the Product Requirement Document (see Ref. [1]) is needed.

1.2 Scope of this Plan

This Document is written in the Concept Phase and is first released at M2. After first release changes will be documented in the Document Change History.

1.3 Intended Audience

This report is written for the PL, the supervisor from HSR and as general documentation for further usage from the customer.

1.4 Definitions, Acronyms and Abbreviations

Definitions, acronyms and abbreviations can be found in the global table (see Ref. [2])

1.5 References

Ref #	*Description*
Ref. [1]	Product Requirement Document for Qualification Tool, 02_ProductRequirementDocument.doc, V1.0
Ref. [2]	Definition, Acronyms and Abbreviations for Qualification Tool, 90_DefinitionAcronymsAbbreviations.pdf, V1.0
Ref. [3]	Design History File for Qualification Tool, 92_DesignHistoryFile.xls, V1.0

1.6 Document Change History

Date	*Version*	*Change*	*Author*
2012-03-02	1.0	Initial Version	AnZo
2012-03-16	1.1	3.6.1 Updated dates in table. 3.8.2 Update milestone dates @M2 3.8.3 Update milestone costs @M2	AnZo
2012-04-09	1.2	3.6.1 Updated dates in table. 3.7 Added Risk: strategy concept wont fulfill requirements 3.8.2 Update milestone dates @M3 3.8.3 Update milestone costs @M3	AnZo
2012-05-18	1.3	3.8.2 Update milestone dates @M4 3.8.3 Update milestone costs @M4	AnZo
2012-06-12	1.4	3.1.1 Added expert. Added BePf to the FW team. 3.2 Added BePf to the graph and described his position. 3.8.2 Update milestone dates @M5 3.8.3 Update milestone costs @M5	AnZo

1.7 Development Process and Deviations

This Documentation is based according to the following version of the SOP Product Development:

V4.6.

Deviations from that version won't be documented, as the project is a small one and would not be qualified enough in time and complexity for all the needed documents.

1.8 Assumptions and Constraints

Assumptions and Constraints	*Impact if not true*
The project will be further developed after this research project.	Research project would not have an economic output. It would be shift to a feasibility study.

2 Product Overview

2.1 Product Description

Existing and newly developed components of the product of this project are indicated in the table below.

Type	*Name*	*Develop / Re-use*
SW	Qualification Tool	New development
SW	Tecan framework for motor drivers	Re-use
SW	Tecan Firmware	Re-use
HW	Existing motors	Re-use
HW	Simulation kit	Re-use

3 Project Overview

3.1 Project Organization

3.1.1 Project Team Definition

The Project Team of this project consists of the members listed in the table below:

Function	*Name*	*Initials*	*Member of Ptm or associated*	*Name of Deputy*
Project Leader (PL)	Joas Leemann	JoLe	N/A	Michael Keller
Application Software Engineer (SWE)	Andreas Zollinger	AnZo	N/A	N/A
FW Engineer (FWE)	Remo Kälin	ReKa	N/A	Christian Meier
FW Engineer (FWE)	Bernhard Pfund	BePf	N/A	Christian Meier
FW Engineer (FWE)	Simon Schoettl	SiSc	N/A	Christian Meier
FW Engineer (FWE)	Nebojsa Vrcelj	NeVr	N/A	Christian Meier
Supervisor HSR	Luc Bläser	LuBl	N/A	N/A
Expert HSR	Jean-Daniel Merkli	-	N/A	N/A

3.1.2 Trainings

Andreas Zollinger ha to be introduced to the current used LUA script. The training is organized by Andreas Zollinger himself.

3.1.3 External Development Partners

No external development partners are involved in this project.

3.1.4 VAR Client Projects

No VAR Client exits within this project.

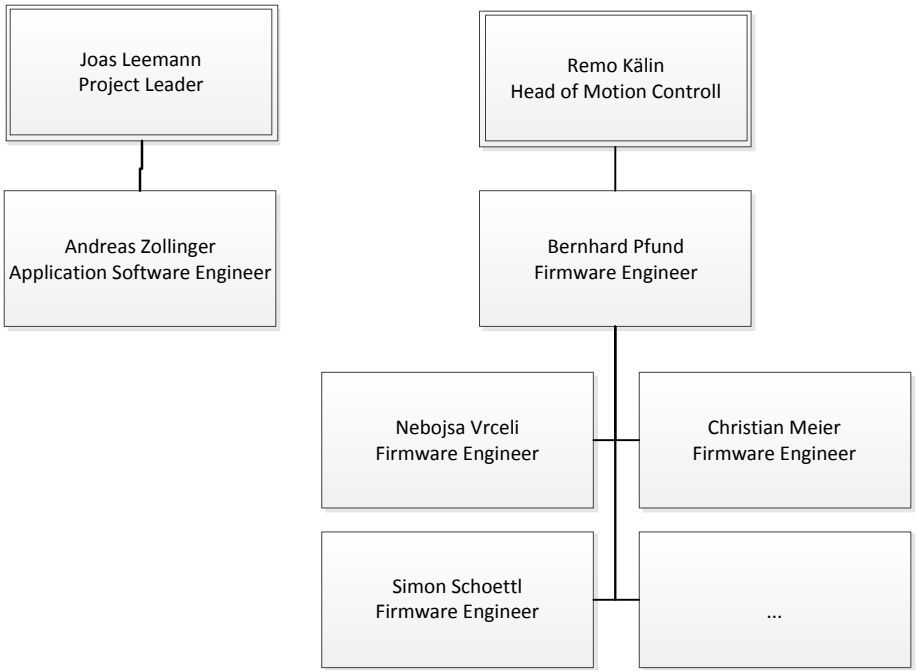
3.1.4.1 Organization at Tecan

No general organization description needed. For internal structure see chapter “3.2”.

3.1.4.2 Organization at the VAR Client

N/A

3.2 Internal Structure



Joas Leemann fulfills the role as PL. At Tecan Schweiz AG he is working as a software architect.
Andreas Zollinger is the executing engineer, responsible for all the working tasks. Inside Tecan Schweiz AG he is working as a software developer.
Remo Kälin and his team provide support considering the firmware. He is regarded as the costumer. At Tecan Schweiz AG Remo Kälin is the head of the motion control skill group. He will be supported by Bernhard Pfund who is the contact person to the firmware team for Andreas Zollinger.

3.3 Confidentiality Disclosure Agreements (CDAs)

CDAs concluded for this project are listed in the table below:

Subject of the CDA	*Address of External Party*	*Name and Phone Number of Responsible Person of external Party*	*Name of Responsible Person within this Project at Tecan*	*Signing Date*
N/A	N/A	N/A	N/A	N/A

No CDAs have to be done.

3.4 Additional specific Project Deliverables

The documents listed in the following table are released at the given date in a first version. Further changes are still possible and will be documented inside each document.

Deliverable	*Date available*	*Responsibility*
PRD	2012-03-09	AnZo
SW Configuration Management Plan	2012-03-16	AnZo
SWS	2012-03-23	AnZo
SW GUI Design – Paper Prototype	2012-03-26	AnZo
SSD	2012-03-30	AnZo
SW GUI Design – Pre-Final	2012-04-20	AnZo
SW GUI Design – Final	2012-05-04	AnZo
SW Unit Test Case Plan	2012-05-18	AnZo
SW Unit Test Case Report 1	2012-05-25	AnZo
SW Unit Test Case Report 2	2012-06-01	AnZo

3.5 Project Controlling

Control Point	*Method*	*Responsibility*
Requirements Management	Requirements are written at the start of the project in the PRD. Minor changes can be adapted after consulting the PL. Late major changes cannot be considered.	AnZo
Schedule Control	In each week one or more meetings with the PL the progress will be monitored. Within meetings every week the actual progress will be discussed with the supervisor from HSR.	AnZo
Quality Control and Quality Assurance	Quality checks are made on a regular basis after each milestone together with the PL and the supervisor from HSR.	AnZo

3.6 Reviews and Audits Out of the Development Process

3.6.1 Reviews and additional Assessments

Design Reviews are executed inside the project team by reviewing design documents by experts of the own team and other teams. The review minutes and the respective signatures will document the review.

Item to be reviewed	*Responsible Person*	*Review Committee*	*Phase*	*Status*
PRD	AnZo	JoLe, LuBl	Concept Phase	2012-03-09
SW Configuration Management Plan	AnZo	JoLe, LuBl	Design Input	2012-03-26
SWS	AnZo	JoLe, LuBl	Design Input	2012-03-30
SSD	AnZo	JoLe, LuBl	Design Input	2012-03-30
SW Unit Test Case Plan	AnZo	JoLe, LuBl	Validation / Testing	To be done
SW Unit Test Case Report 1	AnZo	JoLe, LuBl	Validation / Testing	To be done
SW Unit Test Case Report 2	AnZo	JoLe, LuBl	Validation / Testing	To be done

3.6.2 External Audits

All new external partners must become qualified.

External partner	*Responsible – Project team*	*Responsible – TQM*	*Documentation for qualification*	*Date*
N/A	N/A	N/A	N/A	N/A

No new external partner exists.

3.7 Major Project Risks

Project risks are indicated in the table below for this project:

Risk	*Probability (low, medium, high)*	*Severity (low, medium, high)*	*Mitigation*	*Remarks*	*Identification Date*
loss of AnZo (illness, death or other causes)	very low	high	-	Very unlikely project would be stopped	2012-02-24
HW is not available when needed	high	medium	A test station with a limited subset of axes can be used. Tests with whole system cannot be guaranteed to be available.		2012-03-02
HW has malfunctions during usage time	medium	medium			2012-02-24
Project cannot be finished due to lack of time	low	medium	Requirements are chosen so the basic implementation can be done.		2012-02-24
To less time for a long time test, as the product is itself is designed for long time tests.	high	low	Long time tests are no part of the bachelor thesis. Reasonable sub steps can be tested.		2012-03-02

3.8 Planned Milestones and Costs

3.8.1 Phase Description

3.8.1.1 Kickoff Phase

During the kickoff phase the main idea on the topic was defined.

3.8.1.2 Concept Phase (M1 – M2)

The Goal of the concept phase is the creation and definition of product requirements. In addition in this phase initial concepts for the new product are created and the general technical feasibility is estimated.
The Product Requirements Document is the main deliverable of this phase.
A Project Development Plan, this document, is also developed in this phase. The PDP lives on through the whole project.
In the concept phase, the Design History File is opened. The DHF index serves as documentation planning for the software documentation. All additional documents that are generated during the project must be stored in the DHF.
The concept phase is completed with the milestone M2.

3.8.1.3 Design Input Phase (M2 – M3)

The goal of this phase is to define all Design Inputs necessary.
The final Product Requirements Document is the main deliverable of this phase.
During this phase SW Specifications and SW Structure Design are also deliverables.
The design input phase is completed with the milestone M3.

3.8.1.4 Design Output Phase (M3 – M4)

In the design output phase the software is developed on the basis of insights from the input phase. The goal is to realize prototypes and to confirm the matching of these with the specifications through the verification.
The design output phase is completed with the milestone M4. The traceability between requirements and specifications has to be done with a design review.

3.8.1.5 Validation & Testing Phase (M4 – M5)

The goal of the validation and testing phase is the completion of the design verification and validation.
The traceability is completed and verified.
The completeness of the implementation of requirements in specifications and their successful verification is checked.
The validation and testing phase is completed with the milestone M5.
M5 is also the end of this project. After M5 only tasks considering the student project from HSR are planned.

3.8.2 Milestone Dates

Milestone dates for this project are indicated in the table below:

Milestone	*Expected Date @M1*	*Expected Date @M2*	*Expected Date @M3*	*Expected Date @M4*	*Expected Date @RfV*	*Expected Date @M5*	*Actual Date*
M1	2012-02-27	N/A	N/A	N/A	N/A	N/A	2012-02-27
M2	2012-03-12	2012-03-12	N/A	N/A	N/A	N/A	2012-03-12
M3	2012-04-02	2012-04-02	2012-04-02	N/A	N/A	N/A	2012-04-02
M4	2012-05-14	2012-05-14	2012-05-14	2012-05-14	N/A	N/A	2012-05-14
RfV	N/A	N/A	N/A	N/A	N/A	N/A	N/A
M5	2012-06-04	2012-06-04	2012-06-04	2012-06-04	N/A	2012-06-04	2012-06-04
M6	N/A	N/A	N/A	N/A	N/A	N/A	N/A

3.8.3 Milestone Costs

Forecasted and actual costs of this project are indicated in local currency (man-hours) in the table below:

Cost-to-date (Actual Total Cost) @ Each Milestone:							
	@M1	*@M2*	*@M3*	*@M4*	*@RfV*	*@M5*	*@M6*
Internal:	12	49.5	121.5	276.0	N/A	359.0	N/A
External:	0	0	0	0	N/A	0	N/A
Material:	0	0	0	0	N/A	0	N/A
Total:	**12**	**49.5**	**121.5**	**276.0**	N/A	**359.0**	N/A
Cost-to-date (Actual Cost from Last to Actual Milestone) @ Each Milestone:							
	@M1	*@M2*	*@M3*	*@M4*	*@RfV*	*@M5*	*@M6*
Internal:	12	37.5	72.0	154.5	N/A	83.0	N/A
External:	0	0	0	0	N/A	0	N/A
Material:	0	0	0	0	N/A	0	N/A
Total:	**12**	**37.5**	**72.0**	**154.5**	N/A	**83.0**	N/A
Forecast (Total Project Cost from BOI to M6) Estimated @ Each Milestone:							
	@M1	*@M2*	*@M3*	*@M4*	*@RfV*	*@M5*	*@M6*
Internal:	360	367	376	391	N/A	408	N/A
External:	0	0	0	0	N/A	0	N/A
Material:	0	0	0	0	N/A	0	N/A
Total:	**360**	**367**	**376**	**391**	N/A	**408**	N/A
Forecast (Project Cost to Next Milestone) Estimated @ Each Milestone:							
	@M1	*@M2*	*@M3*	*@M4*	*@RfV*	*@M5*	*@M6*
Internal:	42	63	126	63	N/A	N/A	N/A
External:	0	0	0	0	N/A	N/A	N/A
Material:	0	0	0	0	N/A	N/A	N/A
Total:	**42**	**63**	**126**	**63**	N/A	N/A	N/A

3.9 Production Quantity

The output will be an executable software package with no quantity requirements.

3.9.1 Project History

This chapter is of informal character only. Its intention is to keep track of the most significant events during the project in order to extract from time to time the lessons learned and to improve the situation.
It specifically contains major changes requested, refused or executed during development phase. This document is also a summary of all changes with the responsible person and time of implementation.

Event / Date	*Decision*	*Consequences*

3.10 Substitution Plan

During development of the qualification tool the motion control department will be introduced to the new SW.
The tool should be able to replace the current solution at the end of the project.
Maintenance and support has to be defined and is no part of this bachelor thesis.

4 Project Documentation

4.1 Documentation Deliverables for VAR Projects

Document Title	*Purpose*	*Intended Audience*	*Responsibility for:*		
			Writing	*Review*	*Approval*
N/A	N/A	N/A	N/A	N/A	N/A

Not required for this project.

4.2 Design History File (DHF) on the Server

Files are saved on the Tecan network server.
A DHF exists in the documentation dictionary of the project. See Ref. [3]

4.3 DHF in Hardcopy

At the end of the project a hardcopy will be delivered to the PL and the instances of HSR who require a hardcopy.

4.4 Traceability

The traceability will be insured in this project by Andreas Zollinger.
The traceability will be saved in an excel sheet.

5 Appendix

N/A

Product Requirements Document

Project-Name: **Qualification Tool**

Project Number: **-**

Subject: **-**

	Author	Reviewer	Approver
Name	Andreas Zollinger	Luc Bläser	Joas Leemann
Function	Software Engineer	Supervisor HSR	Project Leader
Date / Visa	2012-03-26	2012-03-30	2012-03-26

Table of Contents

1 Introduction

1.1 Purpose

The purpose of the PRD is to clarify who the persons in interests are and what they expect from the outcome of this project.

1.2 Scope

This document is created in the Concept Phase of the project timeline. At M2 (see Ref. [1]) the document has a valid status and goes into version 1.0. After M2 the PRD can be modified with entries in the change history.

1.3 Definitions, Acronyms and Abbreviations

Definitions, acronyms and abbreviations can be found in the global table (see Ref. [3])

1.4 References

Ref #	*Description*
Ref. [1]	Project Development Plan for Qualification Tool, 01_ProjectDevelopmentPlan.pdf, V1.0
Ref. [2]	http://www.lua.org/
Ref. [3]	Definition, Acronyms and Abbreviations for Qualification Tool, 90_DefinitionAcronymsAbbreviations.pdf, V1.0

1.5 Document Change History

Date	*Version*	*Change*	*Author*
2012-03-09	1.0	Initial Version	AnZo

2 Intended Use of the Product

2.1 Background

One of the core businesses from Tecan Schweiz AG are the instruments which are built up of many different electrical motors. At the start of the development of a new instrument the motors have to be configured with different parameters. This step has to be done to ensure the correct function of each motor and the task it has to fulfil.
One motor has many different parameters like force values considering gravitation, offsets or the parameters for a PID controller. All these parameters are evaluated by the Motion Control department.
These parameters have to be verified with help of long time tests and statistic calculations. This process is currently done with help of LUA scripts (See Ref. [2]). A data dump, generated by the script, have to be imported into an excel sheet and have to be analysed with help of other statistic SW modules.
This verification process can be supported by introducing a new tool, handling execution, analysing and reporting together in one solution.

2.2 New Solution

The qualification tool should replace the current LUA scripts and it's manually handling of text data dumps. Reports can be generated without great effort. Exporting into various formats, for example excel sheets, are automated.

3 Basic Conditions for Definition of Requirements

3.1 Clients, Customers and other Stakeholders

3.1.1 Clients for the Product

Client and customers don't differ. Look at the following chapter.

3.1.2 Customers for the Product

The Motion Control skill group is the customer of the qualification tool. They need the software to verify determined parameters for each electric motor.

3.1.3 Other Stakeholders

The Instrument Software skill group is responsible for the Tecan Base SDK and has the role as a consultant concerning implementation of the software.
The system engineers are also interested in the tool for their long life time tests they have to do.

3.2 RA Strategy

No RA Strategy is needed for this project.

3.3 Users of the Product

3.3.1 User Groups

3.3.1.1 Parameter Tester

Members of the motion control skill group evaluate parameter sets for each individual axis. The task of a parameter tester is to verify that these parameter sets meet the specification defined for these axes.

3.3.1.2 Life Time Tester

Life time testers want to clarify if each axis can reach there specified life time usage, as example a minimum amount of total distance. He has written test scripts that will run for weeks.

3.3.1.3 Strategy Master

The strategy master has an expert knowledge how test runs should be built up and in which order different test should be carried out. He is the person in charge and all strategy changes have to be done by him.

3.3.2 User Characteristics

The following characteristics are used to describe the user groups:

Name	*Description*	*Range*
Axis Parameter Knowledge	The understanding what axis parameters are and how the behaviour of an axis will change if one of the parameters will be mutated.	little – average – expert
Responsibility	Describes how much influence in political decisions a user group has. The higher the responsibility is the more the user can change and specify processes, for example how a test run should look like.	little – average – high
Attitude towards new technology	Describes how easy a person can be motivated to use new software. The changeover to the new qualification tool can be made smoother if the user is willing to learn new things	conservative – neutral – enthusiastic
Abstraction wanted	How much abstraction a person wants to move a drive. Some people want measure the current who powers the motor, others just want to press a button and it should move.	deep – neutral – abstract
HW capability	Has the person the allowance and the knowledge to alter the HW. Does the person know how the HW is built?	no permission – little – expert

Characteristic	*Parameter Tester*	*Life Time Tester*	*Strategy Master*
Axis Parameter Knowledge	expert	little	expert
Responsibility	average	little	high
Attitude towards new technology	neutral	neutral	neutral
Abstraction wanted	neutral	deep	neutral
HW capability	no permission	expert	little

3.3.3 User Priority

From the table above the priority of the different user groups could be determined:

3.3.3.1 Key User: Parameter Tester & Life Time Tester

The parameter tester and the life time tester can be merged to one single user. They have different goals they want to reach with this tool but both can provide from the functionality that is implemented for the other part. They also can start and handle the new tool exact the same way. A separation of this two user groups is not necessary.
They will use the tool to fulfil their individual tasks and generate a report.

3.3.3.2 Second Level User: Strategy Master

The strategy master has to define how a test run should look like. With help of the new tool he can specify the single steps which should be carried out later by the testers.
After he has defined the strategies he no longer uses the tool, that's what testers are for. He still will maintain the strategies and add new one.
He is interested in the reports generated by the tool after the tests are done.

3.3.4 User Participation

The strategy master is participating during the concept and design input phase. He can deliver the knowledge how the tests should be carried out and will define some of the strategies during the project.
The testers will be interviewed as end users so the new tool is exactly designed for their needs. They also can support the implementation with explaining the current implementation.

3.4 Constraints to the Product

3.4.1 Solution Constraints

The qualification tool shall run on Windows XP operating system and on Windows 7 64Bit operating system.
The qualification tool shall run using the Tecan Base SDK only, independent of a project where the Tecan Base SDK is used. Special strategy configurations for individual projects have to be done.

3.4.2 Implementation Environment

A standard Tecan software development workspace is quite sufficient for implementing the new software. In the first phase of implementation a simulation box with attached axes is needed. In the later or end phase of implementation and during testing a complete instrument would be preferable.

3.4.3 Interface to other Applications

The qualification tool uses the Tecan Base SDK.

3.4.4 Commercial Off-the-Shelf Packages

N/A

3.4.5 Anticipated Workplace Environment

The parameter tester and life time tester work normally with notebooks to be independent from location. The notebook is connected to an instrument where the tests shall run. After starting the tool the parameter tester and life time tester will most likely walk away and the tool must be able to work on its own.
A parameter tester or a life time tester rarely has direct contact to the customer of the instrument.

3.4.6 Anticipated problems

No physical problems are expected.

3.4.7 Critical Dates and Opportunity Windows

15. June 2012 is the delivery date for the project. At 12:00am the work has to be finished.

3.4.8 Other Input for generating Requirements

N/A

3.5 Costs

In the context of costs in this project the currency is always man-hour if not specified different.

3.5.1 Total Costs

As a bachelor thesis is valued with 12 ECTS points the total costs shall lie around 300-360 man-hours.

3.5.2 Material Costs

This project should not raise additional material costs as the infrastructure and development tools are already available.

3.6 Warranty

This project will not provide any warranty because it is not possible to carry out the long-time life cycle tests within the limited time of this project. Additional work after the end of the bachelor thesis has to be done to gain a status where warranty can be ensured.

3.7 Relevant Trends and Assumptions

LUA scripts are not in the standard skill portfolio of a new motion control member. Every new team member has to learn first this script language. This costs time where the employee is not capable of doing productive work. Further on it is not possible to verify a LUA script as everybody can manipulate the script freely to their wishes.
Sooner or later a change to another solution has to be done. This projects replaces some of these LUA scripts.

3.8 Further Development and Extensibility

After this project further development is possible.
A short list with possible additional features:

- Use more than one instrument simultaneous.
- Upgrade the GUI frontend for faster working.
- Use other inputs (audio, vibration) for validating parameters.

3.9 Work Context and Workflow

3.9.1 Work Context

Tecan Schweiz AG develops medicinal instruments which helps a laboratory technician by taking over some of the tasks in an automatically procedure. During development of this instruments each single motor inside has to be parameterized. It is utterly depending on what task and what circumstances a motor is used. So for each single motor-“type of use”-combination different parameters have to be determined. These parameters have to be verified. After defining a specific set of parameters for each use case these parameters are used by the firmware and the Tecan Base SDK.

3.9.2 Workflow

The workflow is always the same:

1. Ask the user to select which axes should be tested.
2. Perform the test according to the strategy.
3. Generate statistics at frequent intervals.
4. Wait till the user stops the test procedure.

3.9.3 Working Procedure

To verify a set of parameters following steps have to be done:

1. Organize an instrument and verify that a connection to the software is available.
2. Choose the axes that have to be verified.
3. Choose different options to define the test strategy.
4. Start the test.
5. Stop the test.
6. Get the reports.

4 Requirements

ID: Unique ID number
Description: Description of the requirement
Priority 1: These requirements have to be fulfilled to be able to use the tool.
Priority 2: These requirements are nice to have in the bachelor thesis.
Priority 3: These requirements are nice to have in later versions.
Source: Name of the issuer of this requirement
Date: Date of issuing the requirement
Supporting Info: References to additional supporting information
Comment: any further comments on the requirement
Proposal: for implementation / specifications

4.1 Use Tecan Base SDK

Priority:	*1*	*Date:*	*2012-02-27*	*Source:*	*AnZo*	*ID:*	*PRD 1*
Description							
The Tecan Base SDK shall be used by the new product. It supports almost all needed functions.							
Supporting Info							
If it does not support a functionality an own implementation can be done.							
Comments							
-							
Proposals							
-							

4.2 One Click Application

Priority:	*1*	*Date:*	*2012-03-03*	*Source:*	*AnZo*	*ID:*	*PRD 2*
Description							
The product shall be designed in a way that a user can start the test with one click after starting up.							
Supporting Info							
The user also shall have the possibility to set options before starting the test run. See PRD 3							
Comments							
-							
Proposals							
-							

4.3 Select Axes to Test

Priority:	*1*	*Date:*	*2012-03-03*	*Source:*	*AnZo*	*ID:*	*PRD 3*
Description							
The user shall be able to select which axes he wants to include during the test run.							
Supporting Info							
Axis not selected for a test run but are within the dependencies of one selected axis must be available to move to support the test run.							
Comments							
-							
Proposals							
-							

4.4 Scheduler

Priority:	*1*	*Date:*	*2012-03-03*	*Source:*	*AnZo*	*ID:*	*PRD 4*
Description							
The tool shall be able to calculate an optimal test run according to the selected axes of the user (PRD 3) and the strategy defined by a master user (PRD 5).							
Supporting Info							
-							
Comments							
-							
Proposals							
-							

4.5 Define Strategy

Priority:	*1*	*Date:*	*2012-03-03*	*Source:*	*AnZo*	*ID:*	*PRD 5*
Description							
A master user shall be able to define strategies which define the order and parameters of a test run.							
Supporting Info							
-							
Comments							
Under normal circumstances a normal user shall not be able to modify the strategy.							
Proposals							
-							

4.6 Select Strategy

Priority:	*2*	*Date:*	*2012-03-03*	*Source:*	*AnZo*	*ID:*	*PRD 6*
Description							
A user shall be able to select a strategy defined by a master user (PRD 5) if more than one is available for the combination of the currently connected instrument and the selected axes to test (PRD 3).							
Supporting Info							
-							
Comments							
-							
Proposals							
-							

4.7 Continuous Reporting

Priority:	*1*	*Date:*	*2012-03-03*	*Source:*	*AnZo*	*ID:*	*PRD 7*
Description							
The tool shall continuously generate a report.							
Supporting Info							
These reports shall be visible to the user during the execution of the tests. The user shall be able to execute an export of the report to specified documents anytime during or after a test run.							
Comments							
The GUI shall not “freeze” during a test run. The user shall be able to still interact with the report view.							
Proposals							
-							

4.8 Error Recovery

Priority:	*1*	*Date:*	*2012-03-03*	*Source:*	*AnZo*	*ID:*	*PRD 8*
Description							
The tool shall try to auto recovery from an occurred error on an axis.							
Supporting Info							
If the axis cannot be recovered the test shall be excluded from the test run. Also all axes that are affected by an axis in the error state have to exclude all the tests which have a dependency.							
Comments							
Error and the recovery shall be explicit documented inside the report.							
Proposals							
-							

4.9 Time Scheduler

Priority:	*2*	*Date:*	*2012-03-03*	*Source:*	*AnZo*	*ID:*	*PRD 9*
Description							
The tool shall contain a time scheduler who can pause and resume a current running test.							
Supporting Info							
This also shall be possible with switching off the instrument, the tool and the computer in between.							
Comments							
-							
Proposals							
-							

Software Specification

Project-Name: **Qualification Tool**

Project Number: **-**

Subject: **-**

	Author	Reviewer	Approver
Name	Andreas Zollinger	Luc Bläser	Joas Leemann
Function	Software Engineer	Supervisor HSR	Project Leader
Date / Visa			

Table of Contents

1 Introduction

1.1 Purpose of the Document

The SWS describes the external behavior of the application or subsystem identified. It also describes nonfunctional specifications, design constraints and other factors necessary to provide a complete and comprehensive description of the specification for the software.

1.2 Scope

This document is generated during the “Design Input” phase and is first released at M3. Changes after the first release will be documented in the document history.

1.3 Definitions, Acronyms and Abbreviations

Definitions, acronyms and abbreviations can be found in the global table (see Ref. [3])

1.4 Referenced Documents

Ref #	*Description*
Ref. [1]	http://www.lua.org/
Ref. [2]	Use Case Specification for Qualification Tool, 04_UseCaseSpecification.pdf, V1.1
Ref. [3]	Definition, Acronyms and Abbreviations for Qualification Tool, 90_DefinitionAcronymsAbbreviations.pdf, V1.0
Ref. [4]	Traceability Matrix for Qualification Tool, 91_TraceabilityMatrix.pdf, V1.0
Ref. [5]	SW Graphical User Interface Design for Qualification Tool, 06_SWGUIDesign.pdf, V1.0
Ref. [6]	SW Structure Design (Architecture) for Qualification Tool, 05_SWStructureDesign.pdf, V1.0

1.5 Document Change History

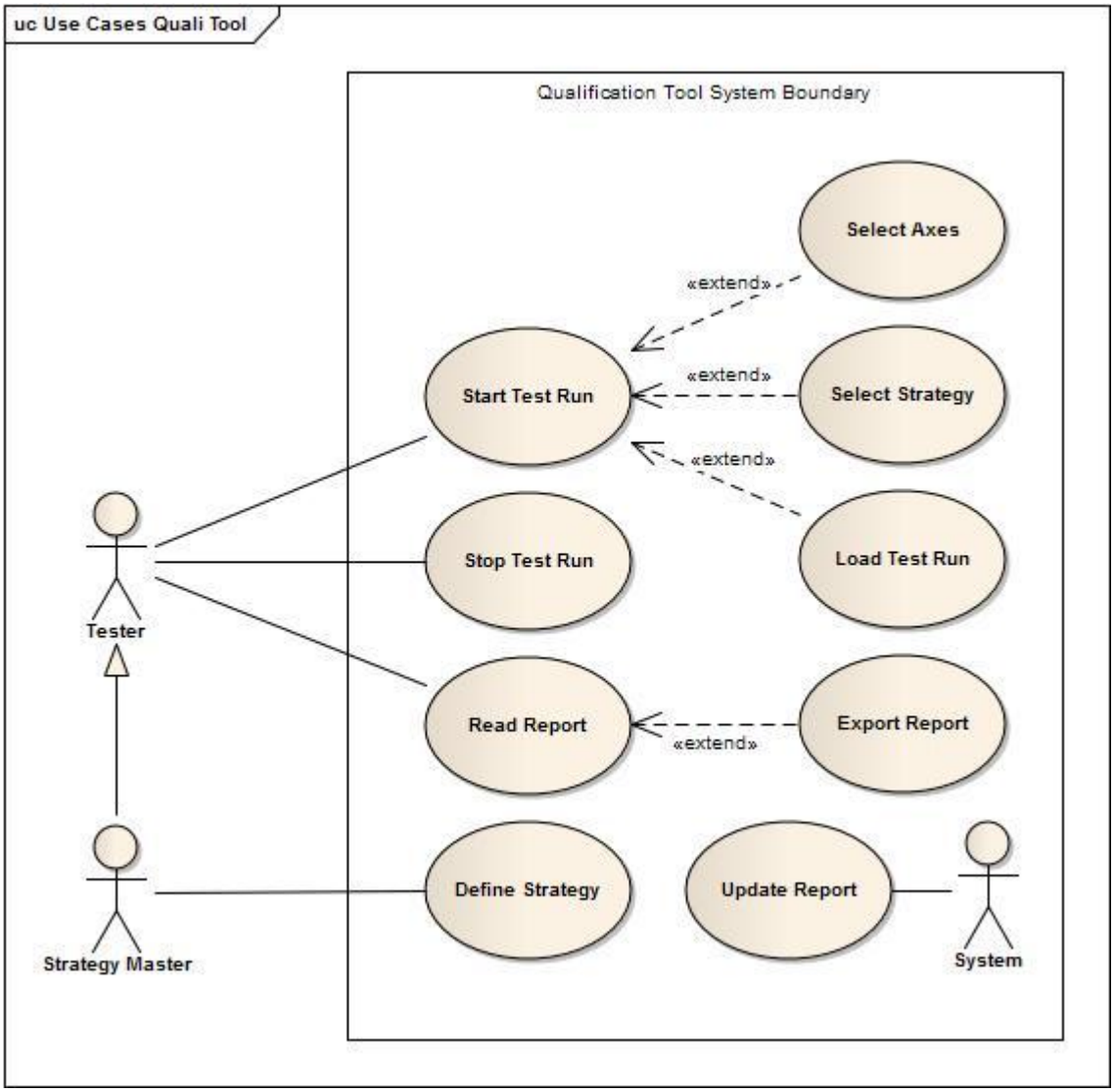
Date	*Version*	*Change*	*Author*
2012-03-19	1.0	Initial Version	AnZo

2 Overall Description

One of the core businesses from Tecan Schweiz AG are the instruments which are built up of many different electrical motors. At the start of the development of a new instrument the motors have to be configured with different parameters. This step has to be done to ensure the correct function of each motor and the task it has to fulfill.
One motor has many different parameters like force values considering gravitation, offsets or the parameters for a PID controller. All these parameters are evaluated by the Motion Control department.
These parameters have to be verified with help of long time tests and statistic calculations. This process is currently done with help of LUA scripts (See Ref. [1]). A data dump, generated by the script, has to be imported into an excel sheet and has to be analyzed with help of other statistic SW modules.
This verification process can be supported by introducing a new tool, handling execution, analyzing and reporting together in one solution.

2.1 Use Case Model Overview

In this chapter a short overview of the use cases is shown. Fully described use cases are defined in the UCS (see Ref. [2]) document.



2.1.1 Use Cases

ID: Unique ID number
Name: Name of the use case
Description: A short description what the use case does in general.

ID	*Name*	*Description*
UC01	Start Test Run	Let the tester start a test run.
UC02	Select Axes	Let the tester select what axes shall be included in the test run.
UC03	Select Strategy	Let the tester select a strategy for the test run.
UC04	Load Test Run	Let the tester load an older test run.
UC05	Stop Test Run	Let the tester stop a current running test run.
UC06	Read Report	Let the tester read a report of a test run.
UC07	Export Report	Let the tester export the current report to various formats.
UC08	Define Strategy	Let the strategy master define strategies.
UC09	Update Report	The system calls an update of the report periodically during a test run.

2.1.2 Actors

Name: Name of the actor.
Description: Description of the actor.

Name	*Description*
Tester	The tester is responsible to carry out the test run. He has to assemble all the different parts, as example a working instrument and a computer with the current software installed. He knows how to handle the qualification tool.
Strategy Master	The strategy master is a special tester. He has the same knowledge and skills as a tester. In addition he can define strategies and has the responsibility for them.
System	The system has to update the report with the statistic data periodically.

2.2 Physical Characteristics

Name	*Specification*
Code language	.net 4.0, C# 4.0
Data file storage	XML
Operating system	Windows XP & Windows 7 (32 & 64 Bit)

3 Specifications

ID: Unique ID number
Name: Name of specification
Description: Detailed description about the specification
Priority: 1 – Must have (work for this project)
2 – Nice to have (work for this project)
3 – Nice to have (after project)
4 – Additional ideas for the future (after project)

To fully understand the specification it is mandatory to read the explanations about the component naming and meaning in the software structure design (See Ref[6]).

3.1 Functionality

3.1.1 Handle Test Run

Name	Start Test Run	*Priority*	1	*ID*	SWS 1
Description					
The tester can start the test run with a click on a “Start Test Run” button. The “Start Test Run” button changes to a “Stop Test Run”					

Name	Stop Test Run	*Priority*	1	*ID*	SWS 2
Description					
The tester can stop the test run with a click on a “Stop Test Run” button. The “Stop Test Run” button changes to a “Start Test Run” The Instrument ends its currently running test run cycle. The test run is then still open, reports are still visible.					

Name	Restart Test Run	*Priority*	2	*ID*	SWS 3
Description					
If a test run is open and the same configuration (instrument axes / strategy) is available the test run can be restarted. The tester can then start the test run with a click on a “Start Test Run” button, as seen in the SWS “Start Test Run”.					

Name	Save Test Run	*Priority*	1	*ID*	SWS 4
Description					
The system automatically saves the test run configuration and its process after each test run cycle. For each test run one file will be generated.					

Name	Load Test Run	*Priority*	2	*ID*	SWS 5
Description					
The tester can load a test run which is saved by the system.					

3.1.2 Configure Test Run

Name	Select Axes	*Priority*	1	*ID*	SWS 6
Description					
The tester can select which axes he wants to include into the test run. If a not included axis has a dependency to an included one, it still has to be present and working to be able to carry out the test run.					

Name	Select Strategy	*Priority*	1	*ID*	SWS 7
Description					
If more than one strategy is available for the current set of included axes the tester can select which strategy should be used.					

3.1.3 Running Test Run

Name	Initialize Process	*Priority*	1	*ID*	SWS 8
Description					
After starting the test run each axes is initialized, so it goes over into a working state. If every axis could be initialized a test procedure starts to test out if each axis can reach all position needed during the test run.					

Name	Test Run Cycle	*Priority*	1	*ID*	SWS 9
Description					
The System calculates little sets of moves which can be carried out in sequence. These sets have something in common like “3 different tests on one axis 5 times”. This current running test run cycle will be carried out to its end if the tester stops the test run.					

Name	Simple Scheduler	*Priority*	1	*ID*	SWS 10
Description					
A very simple scheduler calculates the test run cycles. After this step the scheduler triggers the test run cycles and waits till the next one can be started.					

Name	Parallel Axes Tests	*Priority*	2	*ID*	SWS 11
Description					
Axes that have dependencies to each other are in one dependency group. If more than one dependency group exists, these axes will carry out their tests in parallel. They have their own test run cycles.					

Name	Automatic Error Recovery	*Priority*	2	*ID*	SWS 12
Description					
If it happens that an axis has an error (due to various reasons, as example a crash into another object or over current on the motor) the system should automatically try to recover from that state. Each step (error occurs, error recovery, succeed/fail) is mentioned in the report.					

Name	Suspend Axes from Test Run	*Priority*	2	*ID*	SWS 13
Description					
If an unrecoverable error occurs on a axis, all dependencies are to be checked and axes which are not able to operate normally are suspended from the current test run.					

3.1.4 Reporting

Name	Continuous Reporting	*Priority*	1	*ID*	SWS 14
Description					
The system writes down the statistic data configured by the software and collected by the firmware after every move. These data is written down in XML files.					

Name	In-Tool Reporting View	*Priority*	1	*ID*	SWS 15
Description					
Inside the tool the tester can view all statistic data for each axis. Besides of the pure values different charts are available. The tester can export the current viewed report anytime. See SWS “Export Reports” for more information.					

3.1.5 Strategy

Name	Strategy Files	*Priority*	1	*ID*	SWS 16
Description					
The strategy master can create, edit and delete strategy files. For each strategy one file exists. This files are written in XML.					

Name	Write Protection	*Priority*	1	*ID*	SWS 17
Description					
The strategy can only be edited by the strategy master. To ensure that only strategies are loaded who are approved by the strategy master a special function is implemented in the system to verify an strategy with help of a checksum. If the checksum is not correct the strategy will not be loaded.					

3.2 Safety

No specifications concerning safety exist in this project.

3.3 Software driven Alarms and Warnings

Name	Informing Email Service	*Priority*	4	*ID*	SWS 18
Description					
The tester can register an email to one or more of the following events: • User defined time expired • Axis in error state. • Axis could be recovered. • Axis in fatal error state (not recoverable). • Specification of one of the axes is exceeded.					

Name	Reporting Errors and Warnings	*Priority*	2	*ID*	SWS 19
Description					
The system will include errors and warnings generated during a test run automatically into the current test data. So they will be always visible inside the report.					

3.4 Operator Messages

No operator messages have be specified.

3.5 Security

No specifications concerning security exist in this project.

3.6 Usability

Name	One Click Application	*Priority*	1	*ID*	SWS 20
Description					
The application should usable for any tester who is also able to work with the development environment of Tecan Schweiz AG. After starting up the application the tester is able to start a test run with just one click.					

Name	Strategy Overview	*Priority*	1	*ID*	SWS 21
Description					
The tool shows all available strategies to the strategy master and enables him to fast open one.					

Name	Strategy Editor	*Priority*	3	*ID*	SWS 22
Description					
The tool supports the strategy master in building strategies with an in-tool editor for strategies.					

3.7 Reliability

Name	Reliability	*Priority*	1	*ID*	SWS 23
Description					
The system has to be resistant to common axes errors. If possible an error recovery must be executed.					

3.8 Performance

Name	Time to Start a Test Run	*Priority*	1	*ID*	SWS 24
Description					
After starting up the application it needs only 2 seconds to start a test run for a trained tester.					

3.9 Installation, Methods of Operation and Maintenance

Name	Installation	*Priority*	1	*ID*	SWS 25
Description					
The software has a copy deployment during this project. The Tecan Base SDK has to be installed. All necessary files can then be copied to the appropriate place.					

3.10 Attributes

No attributes have to be specified for this project.

3.11 Inputs and Outputs

Name	Instrument Configuration	*Priority*	1	*ID*	SWS 26
Description					
As the tool uses the Tecan Base SDK also the used instrument configuration file of this framework is used.					

Name	Strategy Input	*Priority*	1	*ID*	SWS 27
Description					

The strategy master defines a strategy.
One file per strategy is generated.
The strategies are written using XML.
This strategies containing a list of axes and they again contain:

- Name of a module number (also name of the axis).
- Dependency information to other axes.
- The parameter can be changed for test runs.
  These values can have a set of concrete values or a range, in which case the system will choose a random value in the range for each single test action.
  - Speed
  - Acceleration
  - Deceleration
  - Profile (Sinus or Linear)
  - Distance
  - Position of dependent axes
- Specification:
  - Speed
  - Acceleration
  - Deceleration
  - Settling Time
  - Positioning Accuracy (dynamic and static)
- A general description
- A text shown in a message box on starting the test run.

Name	Export Reports	*Priority*	1	*ID*	SWS 28
Description					

The tester can export the current opened report to various export files:
Supported formats are:

- PDF / XPS (Priority 1)
- Excel CSV (Priority 2)
- Charts, Pictures (Priority 3)
- Web Service (Priority 4)

3.12 Design Constraints

The new product uses the Tecan Base SDK. So some design constrains are given:

- SW language: .net 4.0 C# 4.0
- WPF
- Development Tool: Microsoft Visual Studio 2010

3.13 Online User Documentation and Help System

The base of an online user documentation will be done.
A detailed documentation will not be part in this project.

3.14 Applicable Hardware

The software should be usable by all hardware that can be used with the Tecan Base SDK.

3.15 Purchased Components (SOUP Components)

Windows XP 32 Bit
Windows 7 64 Bit

3.16 Interfaces

3.16.1 User Interfaces

Name	Test Run OK/NOK Feature	*Priority*	1	*ID*	SWS 29
Description					
The UI must contain a view that a tester who looks at the UI can register if everything is OK or not OK (NOK). One aspect of this feature is the coloring. Following colors are used: • Green: Everything is OK. • Yellow: One or more axes had a problem, but error recovery worked. • Red: One or more axes are currently excluded from the running test run because of an error and failed recovery tries.					

Name	Statistic OK/NOK Feature	*Priority*	1	*ID*	SWS 30
Description					
The UI must contain a view that a tester who looks at the UI can register if the statistics of axes are OK (as in the specifications or better) or not OK (NOK). One aspect of this feature is the coloring. Following colors are used: • Green: Everything is OK. • Red: Statistic data of one or more axes is over the specified values.					

3.16.2 Hardware Interfaces

The HW interfaces are handled by the Tecan Base SDK.

3.16.3 Software Interfaces

Name	Tecan Base SDK	*Priority*	1	*ID*	SWS 31
Description					
The software uses the Tecan Base SDK and its functionality.					

3.16.4 Communications Interfaces

Communication to the FW will be handled by the Tecan Base SDK.

3.17 Licensing

No licensing specifications have to be done for this project.

3.18 Legal, Copyright and Other Notices

The developed product is property of Tecan Schweiz AG.

3.19 Applicable Standards

Applicable standards concerning the GUI are handled in the Graphical User Interface Design. See Ref. [5]

4 Traceability

The traceability is handled inside the global traceability matrix file. (See Ref. [4])

5 Appendix

n/a

Use Case Specification

Project-Name: **Qualification Tool**

Project Number: **-**

Subject: **-**

	Author	Reviewer	Approver
Name	Andreas Zollinger	Luc Bläser	Joas Leemann
Function	Software Engineer	Supervisor HSR	Project Leader
Date / Visa			

Table of Contents

1 Introduction

1.1 Purpose

This document describes the software use case specification for the qualification tool. It defines the ways how the tool should be realized.

1.2 Scope

This document belongs to the qualification tool project. This document is generated during the “Design Input” phase and is first released at M3. Changes after the first release will be documented in the document history.

1.3 Definitions, Acronyms and Abbreviations

Definitions, acronyms and abbreviations can be found in the global table (see Ref. [1])

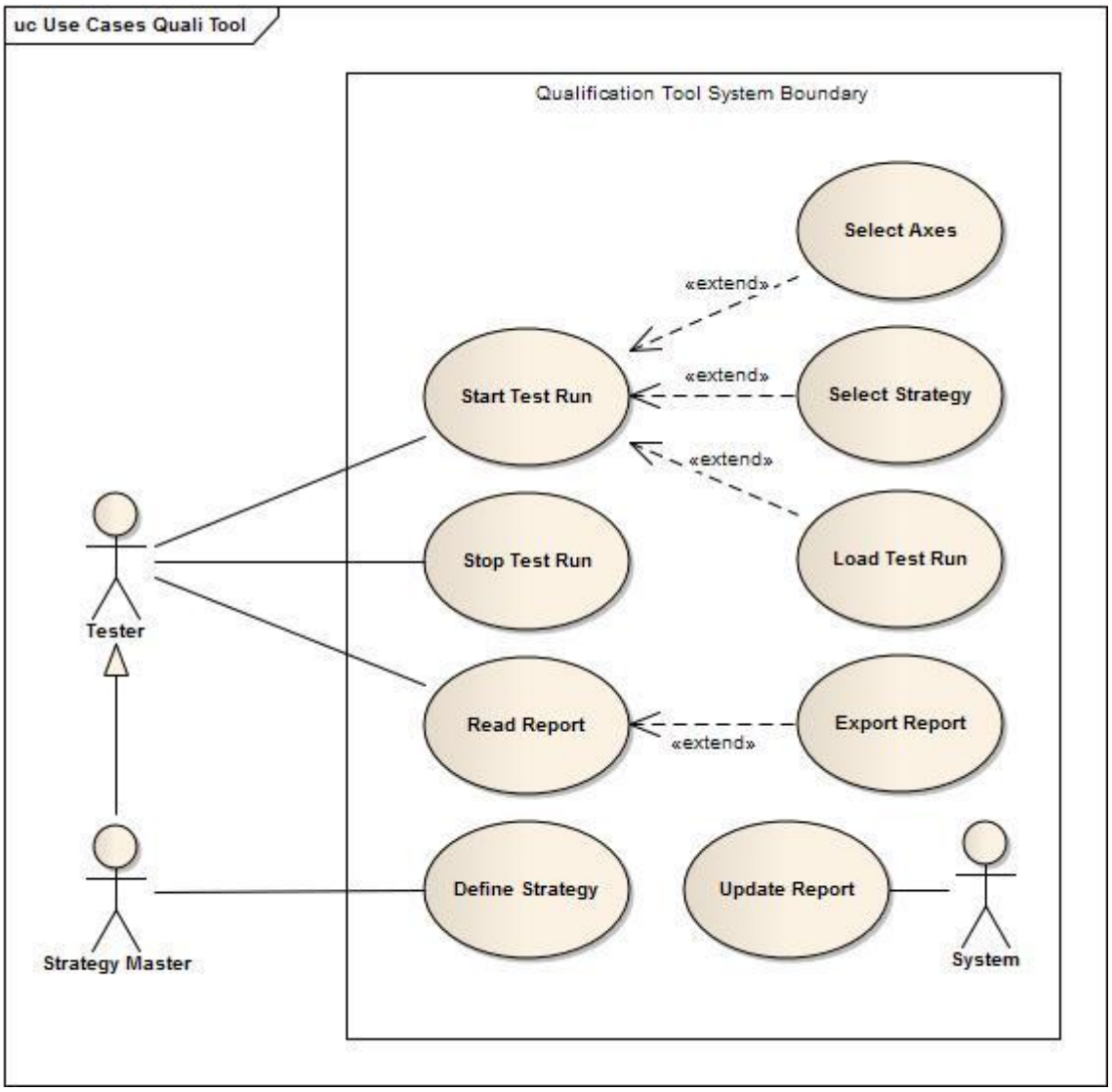
1.4 References

Ref #	*Description*
Ref. [1]	Definition, Acronyms and Abbreviations for Qualification Tool, 90_DefinitionAcronymsAbbreviations.pdf, V1.0
Ref. [2]	Traceability Matrix for Qualification Tool, 91_TraceabilityMatrix.pdf, V1.0

1.5 Document Change History

Date	*Version*	*Change*	*Author*
2012-03-18	1.0	Initial version	AnZo
2012-03-29	1.1	Added one UC: UC09 Update Report Added User: System Edit 3.9.5 Table is not split over page anymore	AnZo

2 Use Cases Overview



2.1 Use Cases

ID: Unique ID number
Name: Name of the use case
Description: A short description what the use case does in general.

ID	*Name*	*Description*
UC01	Start Test Run	Let the tester start a test run.
UC02	Select Axes	Let the tester select what axes shall be included in the test run.
UC03	Select Strategy	Let the tester select a strategy for the test run.
UC04	Load Test Run	Let the tester load an older test run.
UC05	Stop Test Run	Let the tester stop a current running test run.
UC06	Read Report	Let the tester read a report of a test run.
UC07	Export Report	Let the tester export the current report to various formats.
UC08	Define Strategy	Let the strategy master define strategies.
UC09	Update Report	The system calls an update of the report periodically during a test run.

2.2 Actors

Name: Name of the actor
Description: Description of the actor.

Name	*Description*
Tester	The tester is responsible to carry out the test run. He has to assemble all the different parts, as example a working instrument and a computer with the current software installed. He knows how to handle the qualification tool.
Strategy Master	The strategy master is a special tester. He has the same knowledge and skills as a tester. In addition he can define strategies and has the responsibility for them.
System	The system has to update the report with the statistic data periodically.

3 Detail Use Case

3.1 Description of the Following Descriptions

This section contains the explanations of the different points the following chapters are about.

3.1.1 Characteristic Information

This chapter defines information that pertains to this particular use case. Each piece of information is important in understanding the purpose behind the Use Case.

3.1.2 Main Success Scenario

This Scenario describes the steps that are taken from trigger event to goal completion when everything works without failure. It also describes any required cleanup that is done after the goal has been reached. The steps are listed in a table.

3.1.3 Scenario Extensions

This is a listing of how each step in the Main Success Scenario can be extended. Another way to think of this is how can things go wrong. The extensions are followed until either the Main Success Scenario is rejoined or the Failed End Condition is met. The Step refers to the Failed Step in the Main Success Scenario and has a letter associated with it. I.E. if Step 3 fails the Extension Step is 3_A.

3.1.4 Scenario Variations

If a variation can occur in how a step is performed it will be listed here.

3.1.5 Related Information

The following table gives the information that is related to the Use Case.

3.2 Start Test Run

3.2.1 Characteristic Information

Goal In Context:	The tester starts a test run.
Scope:	System
Level:	Task
Pre-Condition:	Axes to test must be available. A strategy exists for all selected axes. No test run is currently running.
Success End Condition:	The test run starts to run.
Failed End Condition:	Test run cannot be started.
Primary Actor:	Tester
Trigger Event:	User action

3.2.2 Main Success Scenario

Step	*Actor*	*Action Description*
1	Tester	Starts the tool.
2	System	Presents all available axes and strategies to the tester.
3	Tester	Starts the test run.
4	System	Begins with the test run routine.
5	System	Stores the test run configuration for later usage.

3.2.3 Scenario Extensions

Step	*Condition*	*Action Description*
3_A	No available axes exist	3_A1: System does not allow starting a test run.
3_B	No available strategies exist	3_B1: System does not allow starting a test run.

3.2.4 Scenario Variations

Step	*Variable*	*Possible Variations*
-	-	-

3.2.5 Related Information

Schedule:	M4
Priority:	Must
Performance Target:	-
Frequency:	Every time a test run should be started.
Super Use Case:	-
Sub Use Case(s):	Select Axes Select Strategy Load Test Run
Channel To Primary Actor:	Direct calling
Secondary Actor(s):	-
Channel(s) To Secondary Actor(s):	-

3.3 Select Axes

3.3.1 Characteristic Information

Goal In Context:	The tester can select what axes shall be included to the next test run. In a normal case all available axis will be tested and this use case will not be needed.
Scope:	System
Level:	Sub-Functionality
Pre-Condition:	The system has to be in a state that the use case “Start Test Run” can be executed.
Success End Condition:	Axis could be selected / deselected.
Failed End Condition:	Axis could not be selected / deselected.
Primary Actor:	Tester
Trigger Event:	User action during the “Start Test Run”.

3.3.2 Main Success Scenario

Step	*Actor*	*Action Description*
1	Tester	Selects / deselects an axis.
2	System	Alters the test run configuration.

3.3.3 Scenario Extensions

Step	*Condition*	*Action Description*
-	-	-

3.3.4 Scenario Variations

Step	*Variable*	*Possible Variations*
-	-	-

3.3.5 Related Information

Schedule:	M4
Priority:	Must
Performance Target:	-
Frequency:	Rare: normally all available axes will be tested.
Super Use Case:	Start Test Run
Sub Use Case(s):	-
Channel To Primary Actor:	Through “Start Test Run”
Secondary Actor(s):	-
Channel(s) To Secondary Actor(s):	-

3.4 Select Strategy

3.4.1 Characteristic Information

Goal In Context:	The tester selects a strategy for the set of axes he has selected for the next test run. In the normal case just one strategy exists for an instrument.
Scope:	System
Level:	Sub-Functionality
Pre-Condition:	The system has to be in a state that the use case “Start Test Run” can be executed.
Success End Condition:	The strategy could be changed to the new selected one.
Failed End Condition:	The strategy could not be loaded.
Primary Actor:	Tester
Trigger Event:	User action during the “Start Test Run” use case.

3.4.2 Main Success Scenario

Step	*Actor*	*Action Description*
1	Tester	Selects a strategy.
2	System	Loads the strategy and alters the test run configuration.

3.4.3 Scenario Extensions

Step	*Condition*	*Action Description*
-	-	-

3.4.4 Scenario Variations

Step	*Variable*	*Possible Variations*
-	-	-

3.4.5 Related Information

Schedule:	M4
Priority:	Must
Performance Target:	-
Frequency:	Very rare: normally just one strategy exists for a test run so the tester cannot choose between two different strategies.
Super Use Case:	Start Test Run
Sub Use Case(s):	-
Channel To Primary Actor:	Through “Start Test Run”
Secondary Actor(s):	Strategy Master
Channel(s) To Secondary Actor(s):	Through “Define Strategy”: The strategies defined by the strategy master are available to the tester during this use case.

3.5 Load Test Run

3.5.1 Characteristic Information

Goal In Context:	The tester can load an older test run. If the same instrument configuration is available, the test run can be restarted with the use case “Start Test Run”. The tester has also the possibility to read the reports of the old test run with the use case “Read Report”
Scope:	System
Level:	Sub-Functionality
Pre-Condition:	The system has to be in a state that the use case “Start Test Run” can be executed.
Success End Condition:	A test run could be loaded.
Failed End Condition:	A test run couldn’t be loaded.
Primary Actor:	Tester
Trigger Event:	User action during the “Start Test Run” use case.

3.5.2 Main Success Scenario

Step	*Actor*	*Action Description*
1	Tester	Wants to load an old test run.
2	System	Presents all available old test runs.
3	Tester	Tester selects a test run.
4	System	Loads the old test run.

3.5.3 Scenario Extensions

Step	*Condition*	*Action Description*
2_A	No test runs exist.	2_A1: Systems informs the user that no test runs exists.
4_A	Test run cannot be loaded due to not available axes	4_A1: Not available axes will be tagged. Restart of the test run is not possible.
4_B	Test run cannot be loaded due to not available strategy	4_B1: Strategy will be tagged. Restart is possible if user can select another suitable strategy.

3.5.4 Scenario Variations

Step	*Variable*	*Possible Variations*
-	-	-

3.5.5 Related Information

Schedule:	M4
Priority:	Want
Performance Target:	-
Frequency:	Variable. From once a day to once in a month.
Super Use Case:	Start Test Run
Sub Use Case(s):	-
Channel To Primary Actor:	Through “Start Test Run”
Secondary Actor(s):	-
Channel(s) To Secondary Actor(s):	-

3.6 Stop Test Run

3.6.1 Characteristic Information

Goal In Context:	The tester can stop the current running test run anytime. The test run will not end immediately. It ends the current test run cycle and then stops.
Scope:	System
Level:	Task
Pre-Condition:	The system is currently in a running state started through the “Start Test Run” use case.
Success End Condition:	The system has ended the last test run cycle and finally stops then.
Failed End Condition:	System was not able to stop in correct order.
Primary Actor:	Tester
Trigger Event:	User action

3.6.2 Main Success Scenario

Step	*Actor*	*Action Description*
1	Tester	Tells the system to stop.
2	System	Finish it current test run cycle.
3	System	Stops the test run.

3.6.3 Scenario Extensions

Step	*Condition*	*Action Description*
-	-	-

3.6.4 Scenario Variations

Step	*Variable*	*Possible Variations*
-	-	-

3.6.5 Related Information

Schedule:	M4
Priority:	Must
Performance Target:	-
Frequency:	Variable. From once in a day to once a month.
Super Use Case:	-
Sub Use Case(s):	-
Channel To Primary Actor:	Direct calling
Secondary Actor(s):	-
Channel(s) To Secondary Actor(s):	-

3.7 Read Report

3.7.1 Characteristic Information

Goal In Context:	The tester can read the report of the current loaded test run. This option is during test run after the test run was started with “Start Test Run”, after the test run was stopped with “Stop Test Run” or after the test run was loaded with “Load Test Run”.
Scope:	System
Level:	Task
Pre-Condition:	One of these: “Start Test Run” “Stop Test Run” “Load Test Run”
Success End Condition:	Report can be read by the tester.
Failed End Condition:	Report could not be read by the tester.
Primary Actor:	Tester
Trigger Event:	User action

3.7.2 Main Success Scenario

Step	*Actor*	*Action Description*
1	User	Tells the system to show the reports.
2	System	Shows all available reports.

3.7.3 Scenario Extensions

Step	*Condition*	*Action Description*
-	-	-

3.7.4 Scenario Variations

Step	*Variable*	*Possible Variations*
-	-	-

3.7.5 Related Information

Schedule:	M4
Priority:	Must
Performance Target:	-
Frequency:	Often. several times during a test run.
Super Use Case:	-
Sub Use Case(s):	-
Channel To Primary Actor:	Direct calling
Secondary Actor(s):	-
Channel(s) To Secondary Actor(s):	-

3.8 Export Report

3.8.1 Characteristic Information

Goal In Context:	The tester can export the current report to different export formats.
Scope:	System
Level:	Task
Pre-Condition:	The system has to be in a state that the use case “Read Report” can be executed.
Success End Condition:	The report could be exported.
Failed End Condition:	The report couldn’t be exported.
Primary Actor:	Tester
Trigger Event:	User action

3.8.2 Main Success Scenario

Step	*Actor*	*Action Description*
1	Tester	Tells the system to export the current report.
2	System	Shows all available axes and export formats to the tester.
3	Tester	Selects the axes to export and what export format should be used.
4	System	Exports the selected axes to the selected export format.

3.8.3 Scenario Extensions

Step	*Condition*	*Action Description*
4_A	Could not generate a report.	2_A1: Systems informs that the generating of the report was not successful.

3.8.4 Scenario Variations

Step	*Variable*	*Possible Variations*
-	-	-

3.8.5 Related Information

Schedule:	M4
Priority:	High-Want
Performance Target:	-
Frequency:	Variable. From several times a day to once in a month.
Super Use Case:	Read Report
Sub Use Case(s):	-
Channel To Primary Actor:	Through “Read Report”
Secondary Actor(s):	-
Channel(s) To Secondary Actor(s):	-

3.9 Define Strategy

3.9.1 Characteristic Information

Goal In Context:	The strategy master can define strategies. These strategies are later on used by the system to carry out the test runs. The strategy master is able to generate new strategies or load old ones and edit them.
Scope:	System
Level:	Task
Pre-Condition:	-
Success End Condition:	A strategy was changed to the needs of the strategy master.
Failed End Condition:	Changes to the strategy couldn't be saved.
Primary Actor:	Strategy Master
Trigger Event:	User action

3.9.2 Main Success Scenario

Step	*Actor*	*Action Description*
1	Strategy Master	Creates new strategy / opens old strategy
2	System	Creates new strategy file
3	Strategy Master	Edits the strategy
4	Strategy Master	Saves the strategy
5	System	Saves the changes to the file
6	Strategy Master	Closes the strategy.

3.9.3 Scenario Extensions

Step	*Condition*	*Action Description*
-	-	-

3.9.4 Scenario Variations

Step	*Variable*	*Possible Variations*
-	-	-

3.9.5 Related Information

Schedule:	M4
Priority:	Must
Performance Target:	-
Frequency:	Rare: the strategy master should only create them once for each possible test and edit them in a first phase.
Super Use Case:	-
Sub Use Case(s):	-
Channel To Primary Actor:	Direct calling
Secondary Actor(s):	Tester
Channel(s) To Secondary Actor(s):	The use case “Select Strategy” can only present strategies which are defined by the strategy master.

3.10 Update Report

3.10.1 Characteristic Information

Goal In Context:	The system must update the report data periodically.
Scope:	System
Level:	Task
Pre-Condition:	-
Success End Condition:	The report data is updated.
Failed End Condition:	The report data could not be updated.
Primary Actor:	System
Trigger Event:	Triggered by the system itself after certain conditions. This could be after enough test data was collected or the user asks for a report currently not available.

3.10.2 Main Success Scenario

Step	*Actor*	*Action Description*
1	System	Copies all test data to a local copy.
2	System	Starts to calculate the statistics.
3	System	Collect all other information for a report.
4	System	Publish a new report.

3.10.3 Scenario Extensions

Step	*Condition*	*Action Description*
-	-	-

3.10.4 Scenario Variations

Step	*Variable*	*Possible Variations*
-	-	-

3.10.5 Related Information

Schedule:	M4
Priority:	Must
Performance Target:	-
Frequency:	Very often: about all 5 to 10 seconds.
Super Use Case:	-
Sub Use Case(s):	-
Channel To Primary Actor:	System calls itself.
Secondary Actor(s):	-
Channel(s) To Secondary Actor(s):	-

4 Remarks

The following table provides insight to additional comments and remarks.

RemarkID	Remark
-	-

5 Open Issues

The following table provides insight to any unresolved problems or questions. These are the things that seem to apply but could not be fit into this use case on this pass.

Issue ID	Issue Description
-	-

6 Traceability

The traceability is handled inside the global traceability matrix file. (See Ref. [2])

Software Structure Design (Architecture)

Project-Name: **Qualification Tool**

Project Number: **-**

Subject: **-**

	Author	Reviewer	Approver
Name	Andreas Zollinger	Luc Bläser	Joas Leemann
Function	Software Engineer	Supervisor HSR	Project Leader
Date / Visa			

Table of Contents

1 Introduction

1.1 Purpose

This document describes the software architecture of the qualification tool. It comprises the structural design of the SW components and design decisions and conventions.

1.2 Scope

This document belongs to the qualification tool. This tool is needed to validate motor parameters against their specification. These parameters are determined by the motion control skill group and they also will test them using the qualification tool.
This document is generated during the “Design Input” phase and is first released at M3. Changes after the first release will be documented in the document history.

1.3 Definitions, Acronyms and Abbreviations

Definitions, acronyms and abbreviation can be found in the global table (see Ref. [1])

1.4 References

Ref #	*Description*
Ref. [1]	Definition, Acronyms and Abbreviations for Qualification Tool, 90_DefinitionAcronymsAbbreviations.pdf, V1.0
Ref. [2]	Use Case Specification for Qualification Tool, 04_UseCaseSpecification.pdf, V1.1
Ref. [3]	00143_05238 C# Programming Guidelines Version 1.0
Ref. [4]	00143_05028 SSD Tecan Base SDK Version 1.0
Ref. [5]	SW Detail Design for Qualification Tool, 08_SWDetailDesign.pdf, V1.0

1.5 Document Change History

Date	*Version*	*Change*	*Author*
2012-03-28	1.0	Initial Version	AnZo

2 System Context

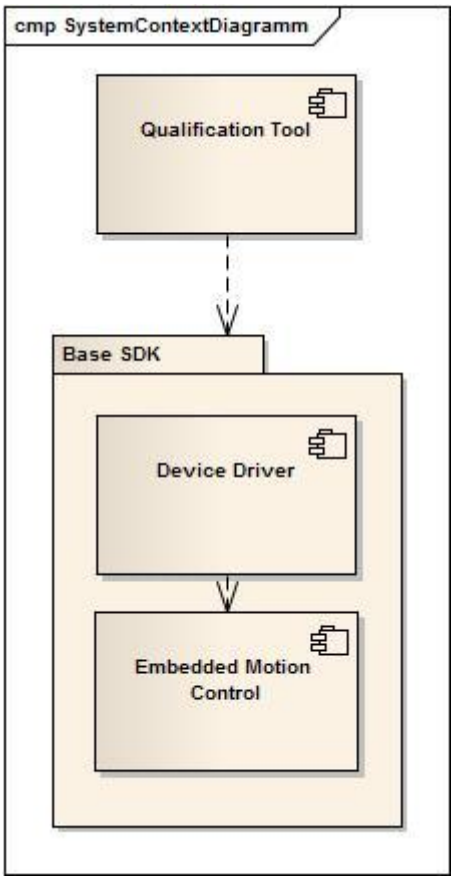


Figure 1: System Context Diagram

2.1 Qualification Tool

The qualification tool component provides the functionality to validate the parameters. It uses the Tecan Base SDK to carry out these tasks.

2.2 Base SDK

The Tecan Base SDK provides drivers and hosts a driver manager used to implement application software to control one or more Tecan instruments. To do that, these drivers communicate to embedded firmware modules.
This document does not describe this package in detail, because it is mentioned just for giving context to the qualification tool.

2.2.1 Device Driver

The device driver controls devices which are specific to the application.
This document does not describe this package in detail, because it is mentioned just for giving context to the qualification tool.

2.2.2 Embedded Motion Control

The Motion Control is a firmware module which handles the communication between the PC and the Tecan instrument on the embedded site. It further contains a firmware module which controls motion drives.
This document does not describe this package in detail, because it is mentioned just for giving context to the qualification tool.

3 Functional View

3.1 Functional Context

The qualification tool provides the functionality to validate motor parameters in an automatized test environment. The continuous generated reports support the testers in their task.

3.2 Functional Areas

3.2.1 Testing Axes

The qualification tool can calculate a continuous test cycle for each individual axis. It stores all data assembled during this tests and stores them.

3.2.2 Reporting

The stored data gained during the test cycles are evaluated and used to do statistic calculations for each individual axis. The results are integrated into a reporting continuously.

3.3 Use Cases Overview

In this chapter a short overview of the use cases is shown. Fully described use cases are defined in the UCS (see Ref. [2]) document.

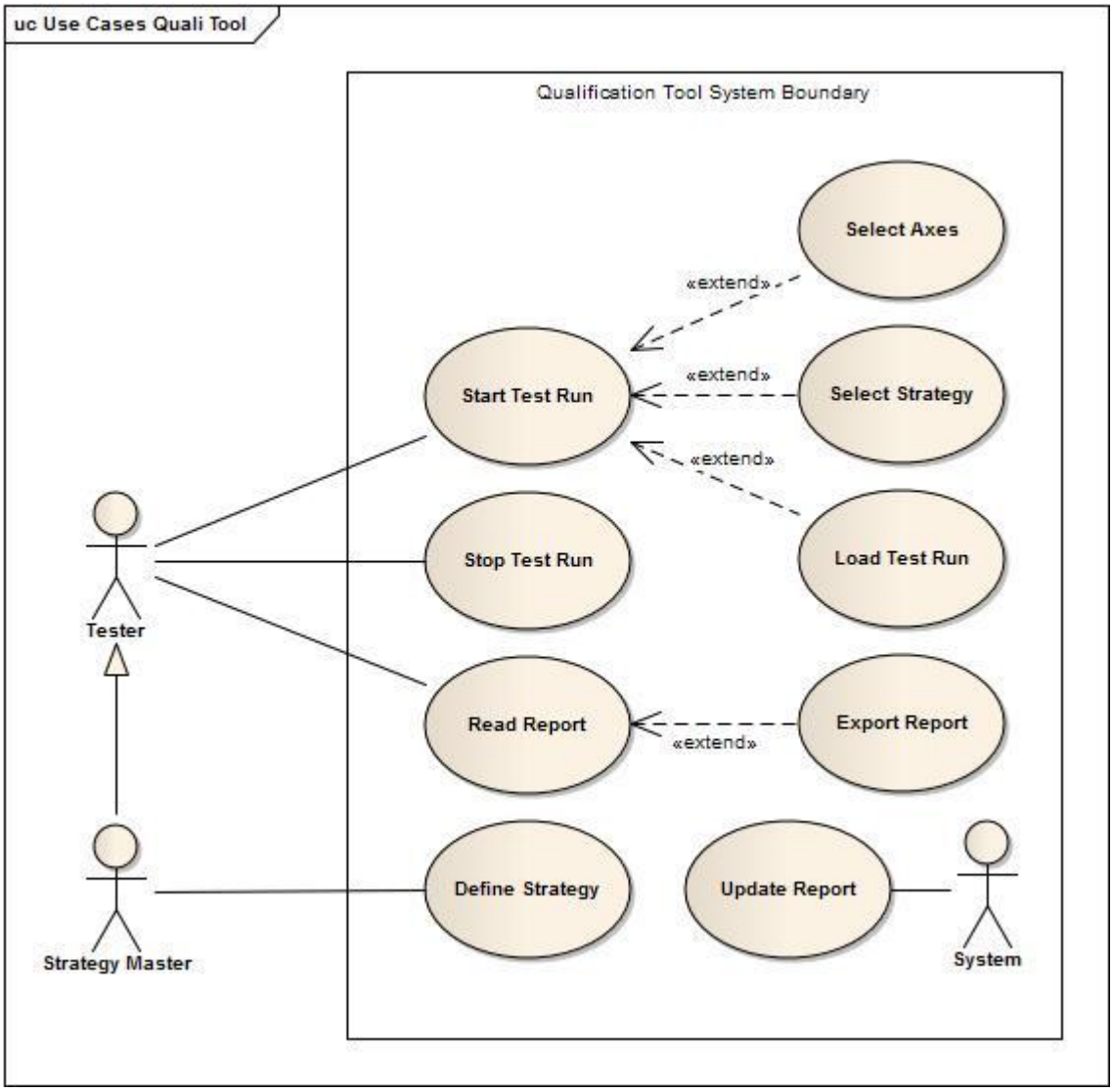


Figure 2: Use Case Diagram

3.3.1 Use Cases

ID: Unique ID number
Name: Name of the use case
Description: A short description what the use case does in general.

ID	*Name*	*Description*
UC01	Start Test Run	Let the tester start a test run.
UC02	Select Axes	Let the tester select what axes shall be included in the test run.
UC03	Select Strategy	Let the tester select a strategy for the test run.
UC04	Load Test Run	Let the tester load an older test run.
UC05	Stop Test Run	Let the tester stop a current running test run.
UC06	Read Report	Let the tester read a report of a test run.
UC07	Export Report	Let the tester export the current report to various formats.
UC08	Define Strategy	Let the strategy master define strategies.
UC09	Update Report	The system triggers an report update mechanism periodically.

3.3.2 Actors

Name: Name of the actor.
Description: Description of the actor.

Name	*Description*
Tester	The tester is responsible to carry out the test run. He has to assemble all the different parts, as example a working instrument and a computer with the current software installed. He knows how to handle the qualification tool.
Strategy Master	The strategy master is a special tester. He has the same knowledge and skills as a tester. In addition he can define strategies and has the responsibility for them.
System	The system acts according to the actions from the user. Single exception is the periodical update of the reports.

4 Process View

The different process views are explained with the following state diagrams

4.1 Application States

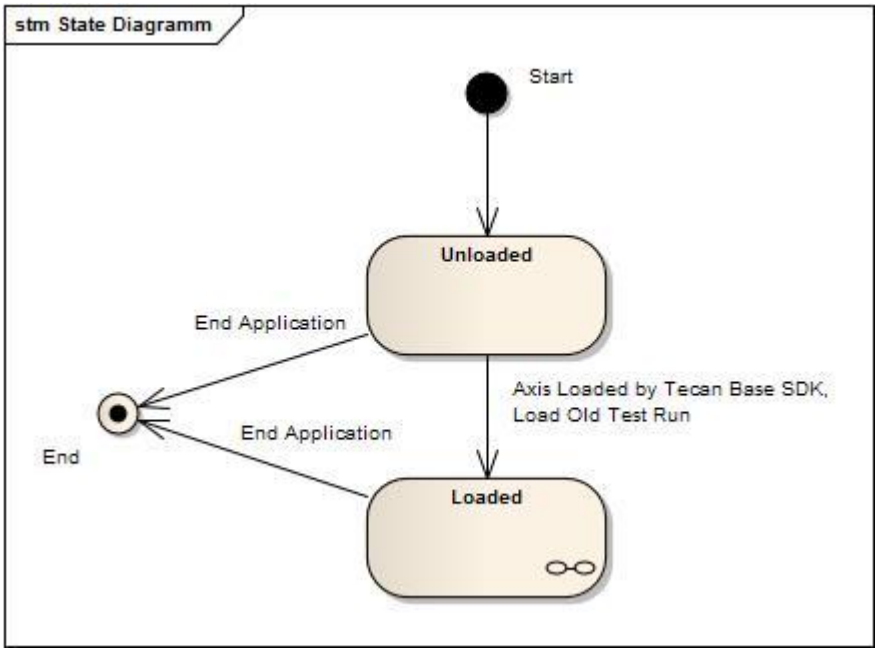


Figure 3: Application State Diagram

4.1.1 Start – End

After the application has started up it can be ended from both main states at any time.

4.1.2 Unloaded State

After starting up the tool is in an unloaded state. No axes are registered.

4.1.2.1 Axis Loaded by Tecan Base SDK

The application is able to be started without the Tecan Base SDK service running in the background. However, provided that the service is running and one or more axes are connected to the computer the application will change over the “Loaded State Machine” automatically. All usable axes are presented to the user.

4.1.2.2 Load Old Test Run

The user has also the possibility to load an older test run. In this case the used axes of the old test run are loaded into the application.

4.1.3 Loaded State Machine

If one or more axes are loaded the application is in the “Loaded State Machine”.

4.2 Loaded State Machine

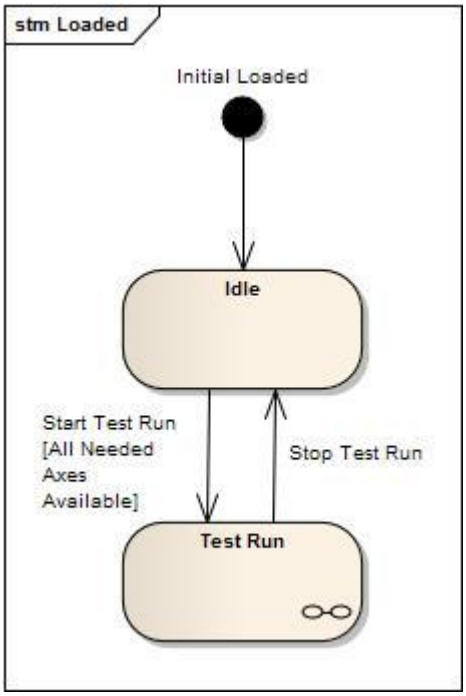


Figure 4: Test Running State Diagram

4.2.1 Start – End

At entering the “Loaded State Machine” will move over to the “Idle State”. As seen in the parent state diagram the application can go over to an end state at any time.

4.2.2 Idle

After loading one or more axes the application is still in an idle state. Reports of the loaded axes can already be viewed by the user.

4.2.2.1 Start Test Run

If all axes selected by the user for testing are available the test run can be started. The application will switch over to the “Test Run State Machine”.

4.2.3 Test Run State Machine

This state machine will execute test runs one by one in an endless loop.

4.2.3.1 Stop Test Run

The user can stop a current running test run any time. However the current running test will be finished first. Thereafter the application will move back to the “Idle State”.

4.3 Test Run State Machine

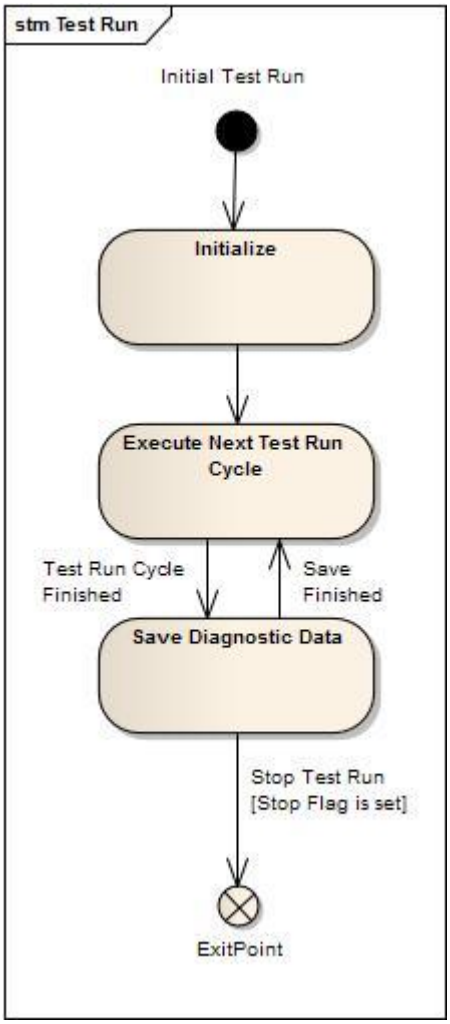


Figure 5: Test Cycle State Diagram

4.3.1 Start – End

By starting the test run the application will start with initializing.
Ending the state machine is possible after each test run cycle when the diagnostic data was saved.

4.3.1.1 Initialize State

During this state all included axes needed for all possible test runs are initialized to guarantee a working environment.
Initialization also contains first range checks and move checks if everything is OK.

4.3.2 Execute Next Test Run Cycle State

The system will take the next test run cycle in line and execute the tests. After finishing all actions inside the test run cycle the system moves over to the “Save Diagnostic Data State”.

4.3.3 Save Diagnostic Data State

After each test run cycle or special axis test the collected diagnostic data will be saved.

5 Non-functional View

5.1 Runtime non-functional requirements

5.1.1 Monitoring

Monitoring the system is one of the main aspects of this tool. Each single step will be saved as statistic data. This also counts for error or warning messages. The system is build up in a way that exception cases are caught, evaluated and logged. In case of an exception which cannot be handled the application will log this and then shut down.

5.1.2 Auditability

The persistence saving format is xml. With a simple but strong base structure the auditability is ensured.

5.1.3 Scalability

The tool itself is “unbranded”. It would work with every single axis. The combination of the instrument configuration file and the strategy file define the borders of the capabilities of the tool. If more tests are needed just the strategy and the execution part of the tool have to be expanded.

5.1.4 Management

One or more axes can be grouped up to a device. Axes and devices can have different versions.

5.2 Non-runtime non-functional requirements

5.2.1 Flexibility

The program uses a fine granularity of assemblies and interfaces to be as flexible as possible. Functions can be modified or enhanced by providing changed or newly created modules.
The Tecan Base SDK supports this functionality innately.

5.2.2 Localization

No localization is done. The default language is English. The tool and all outputs are in English.

5.2.3 Extensibility

The qualification tool uses strategy files for defining test runs. If a new axis or a new type of instrument should be tested the strategy master just have to add more strategy files and the tool will also be able to handle the new instrument.

6 Architectural Constrains

6.1 IDE

The qualification tool is developed with Microsoft Visual Studio 2010.

6.2 Coding-Languages

C# / .NET 4.0 for PC based SW will be used.

6.3 Coding-style-guide

The coding-style-guide for C# is specified in an external document. See Ref. [3].

6.4 WPF/GUI

WPF is used for GUI.

7 Architectural Principles

7.1 Key Layers

The qualification tool environment is divided into following layers:

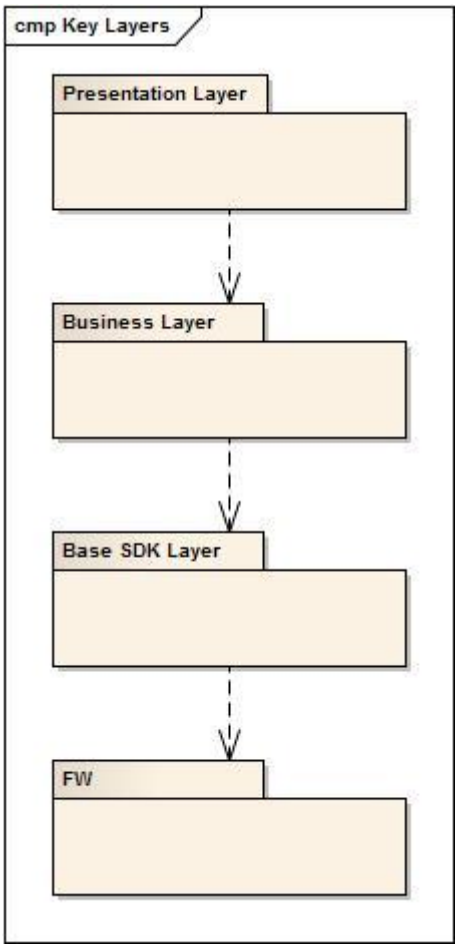


Figure 6: Key Layers

The flow of logic goes from top to down. The lower layers have no knowledge or do not reference to any layers above them except if the higher layers register themselves to lower layers. The abstraction goes from bottom to top. The lower layers implement smaller components and the higher layers combine these components to modules.

7.1.1 Presentation Layer

The presentation layer implements the GUI suitable for the needs of validating the axes.

7.1.2 Business Layer

The business layer contains all the logic to carry out the test runs and generating the reports. For these tasks it also uses functionality provided by the Tecan Base SDK.

7.1.3 Base SDK

The layer of the Tecan Base SDK contains drivers and services which are used to implement specific applications.

This document does not describe this layer in detail, because it is mentioned just for giving context to this tool.

7.1.4 FW

The firmware layer contains the software modules which run on the embedded processors in the instrument.
This document does not describe this layer in detail, because it is mentioned just for giving context to this tool.

7.2 Layer Details

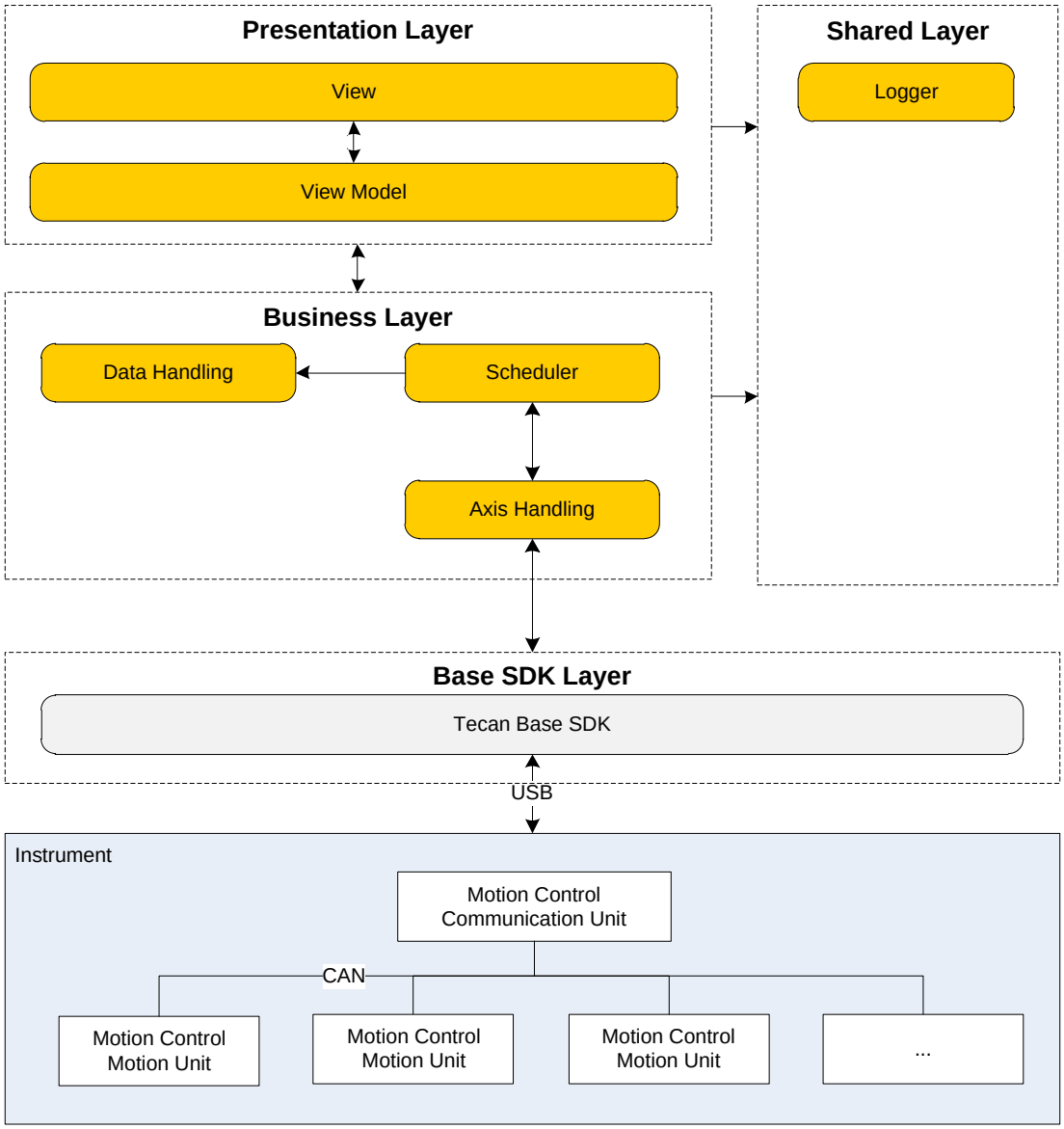


Figure 7: Layer Details

7.2.1 Presentation Layer

7.2.1.1 View

View Package contains all the GUI logic and presentation data.

7.2.1.2 View Model

View Model knows how to convert data from the business layer that the view layer can present them. The view model also interprets user actions and forwards the input to the business layer.

7.2.2 Business Layer

Business layer is responsible for the business logic. It is the knowledge holder and knows what function under what circumstances have to be executed.
Just the axis handling unit is able to use the functionality of the Tecan Base SDK.

7.2.2.1 Scheduler

The scheduler unit is responsible for the test runs. This package has the knowledge to calculate optimized test cycles and is able to start them. It further collects the data of each every test action and passes them over to the data handling unit.

7.2.2.2 Data Handling

The data handling has two main tasks.
First, it is responsible to generate the statistics and so the reports. The view model units can get the data for presentation from this unit. Further on the scheduler unit filling this package with the collected data.
The second main task is the whole file handling itself. The different files involved for a test, as example the strategy file, is handled by this unit. Also file exports of reports are done in this package.

7.2.2.3 Axis Handling

This unit provides all needed functionality of the Tecan Base SDK to the scheduler unit.

7.2.3 Shared Layer

The shared layer just contains one single component, the logger. This static component can be used by every other component.

8 Logical View

8.1 Presentation Layer

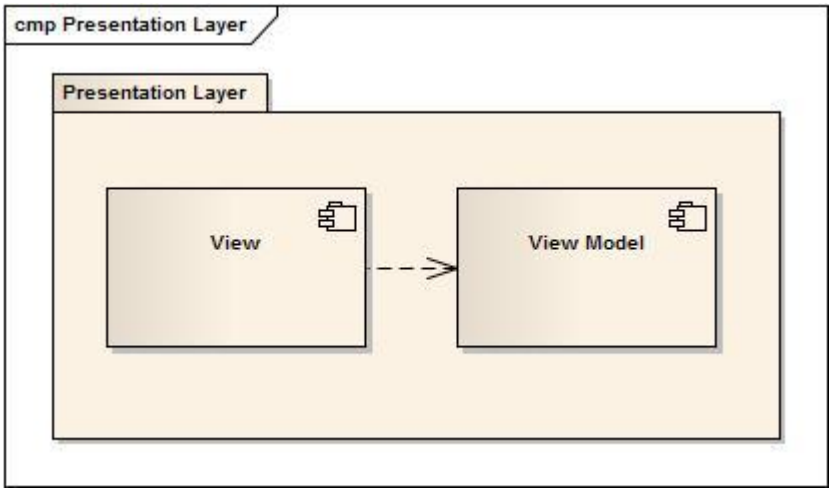


Figure 8: Presentation Component Package

8.1.1 View

This component package contains all views, templates and user controls. The view is connected to the view model.

8.1.2 View Model

The view model connects the view with the business logic. It interprets the user inputs and knows how to handle them and how to push them further down to the business layer. It pulls the data needed for the view also directly from the business layer.

8.2 Business Layer

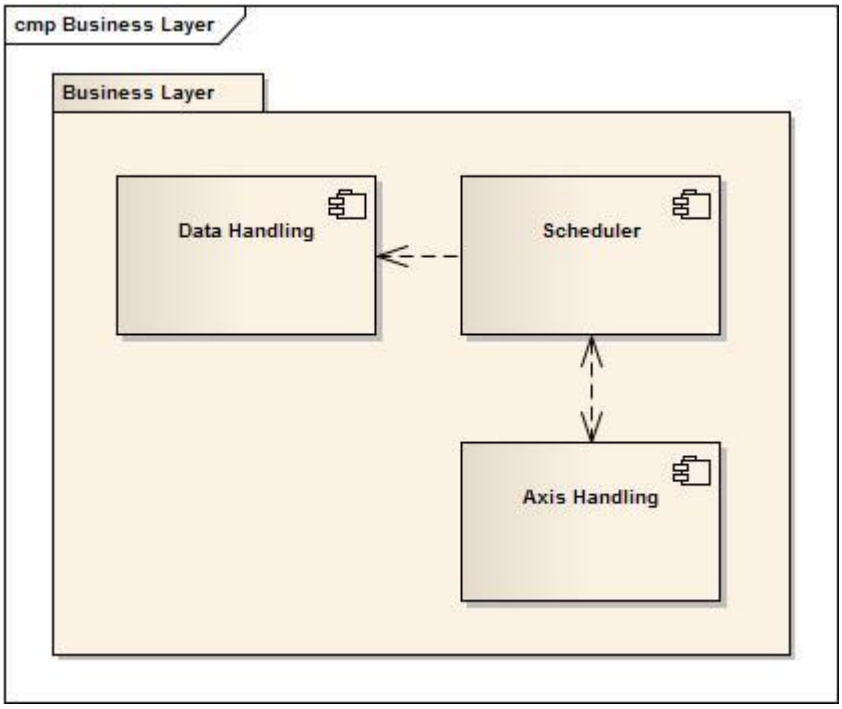


Figure 9: Business Component Package

8.2.1 Data Handling

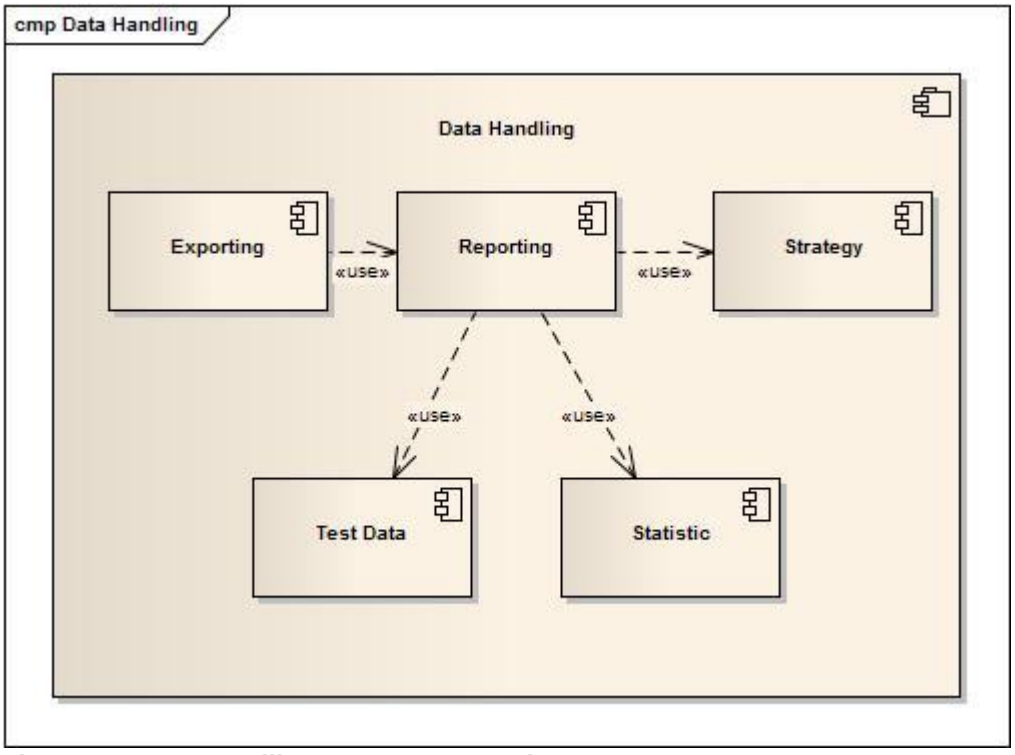


Figure 10: Data Handling Component Package

8.2.1.1 Reporting

The reporting component is responsible for gathering all information and assembles them. To generate a report it uses other components of this component package. The strategy component is used to get the requirements and assumptions of a test run. The test data component is used to get the gathered data. The statistic component is used to calculate statistic data out of the gathered test data. For further information about this connection see the according point in the design view chapter.

8.2.1.2 Exporting

The exporting is capable of generating files of various types containing the results of the reporting component.

8.2.1.3 Strategy

The strategy component is capable of handling the strategy files. It can interpret the strategy and provide them to other components. The scheduler components use the strategy to calculate the next test moves. The reporting component uses the strategy to create reports.

8.2.1.4 Test Data

The test data component has two main tasks to fulfill.
Mainly it is the container for the storage of the diagnostic data generated during a test run. This data it can provide to the reporting component. See the according design view chapter for further explanation concerning this connection.
The second task is to safe the test configuration and all its assembled data to a file. These files are used to be able to load already done tests from the past.

8.2.1.5 Statistic

The statistic component is responsible for the needed statistic calculations for the reporting component.

8.2.2 Scheduler

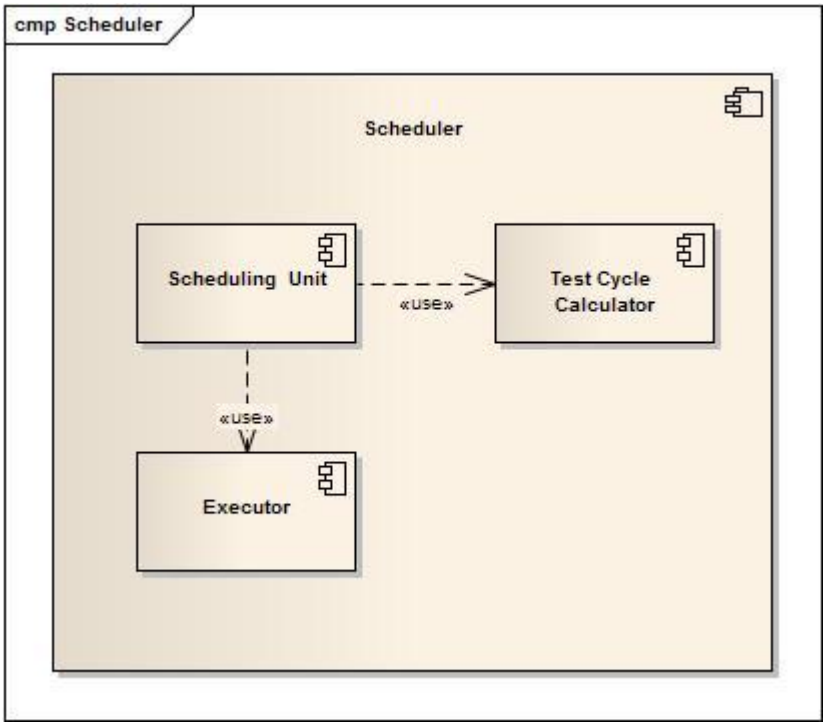


Figure 11: Scheduler Component Package

These three components have special connections. See the according chapter in the design view section.

8.2.2.1 Scheduling Unit

The scheduling unit is the centerpiece of the qualification tool concerning test runs. Most of the communication of the different layers and components are handled by this part.
The scheduling unit knows which axes are included into the current test run and what the next steps will be. It uses the test cycle calculator to get the next test steps. The executer component is used to carry out these test steps.

8.2.2.2 Test Cycle Calculator

The test cycle calculator can calculate optimal test cycles for each individual axis and provides them to the scheduler unit.

8.2.2.3 Executor

The executor component will carry out an test cycle it has get from the scheduler unit.

8.2.3 Axis Handling

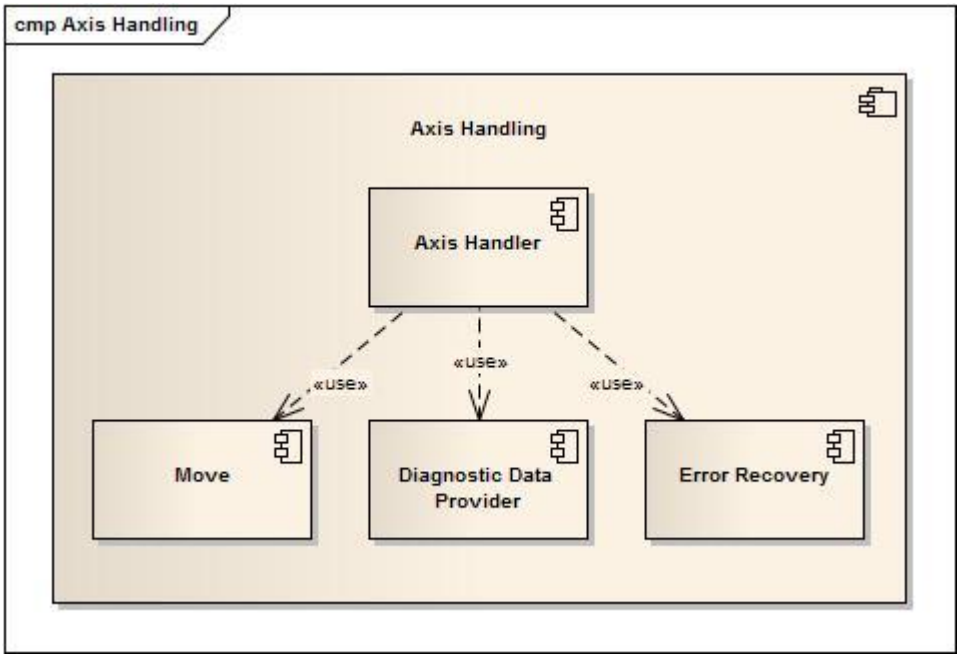


Figure 12: Axis Handling Component Package

8.2.3.1 Axis Handler

The axis handler acts as a façade towards the scheduler components.
This component knows the correct order for caring out a test step. It uses the other three components for carry out these test steps.

8.2.3.2 Move

This component knows how to move an axis. It can carry out normal moves as also specials ones, like initializing an axis.

8.2.3.3 Diagnostic Data Provider

The diagnostic data provider component knows how to gather the diagnostic data used by the data handling components.

8.2.3.4 Error Recovery

The error recovery component can try to recover single axes from an error state.

9 Interface View

The Tecan Base SDK remoting services will be used to.

For further information can be obtained from the Tecan Base SDK documentation. (See Ref. [4])

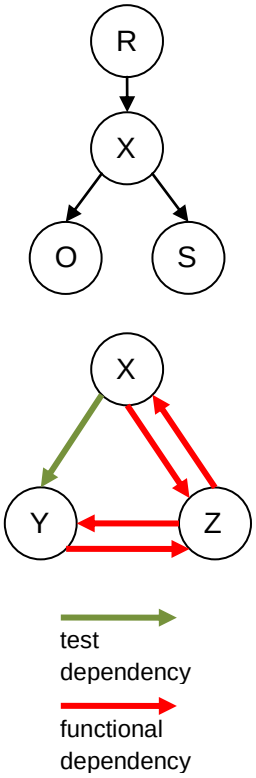
10 Design View

10.1 Qualification Tool Specific Naming

10.1.1 Strategy

A strategy contains all needed elements for running a test.
It knows for each single axis which test should be executed. This test can be reference to standard tests predefined or also specific test cases.
The strategy also contains the dependency description of one axis to other ones. For an example has the Z axis which is moving up and down on many instruments a dependency of the Y and X axis because the Z axis may have a “hole” where the robotic arm is able to move its full range. So the Z axis is only allowed to move down if the X and the Y are on the correct position. This also may be true in the opposite way: the X or the Y axes are only allowed to move if the Z axis is fully retracted, otherwise it may collide with the instrument.
A detailed description of the strategy concept is written in the SW Detail Design document (see Ref [5]).

10.1.2 Dependency Group



Because of the possibility of dependencies of one axis to others dependency groups are created. With an axis (as example axis R) has a dependency to another one (axis K) and this axis again has more dependencies to more axes (axis O and S) a tree like model of the axis can be build up.

But there are also axes with two-way dependencies. As example the X, Y and Z axis may have dependencies to each other, functionally wise or test wise.
A functional dependency describes a dependency for axis that won’t work if the other axis is not available and are not on the right position.
A test dependency describes that the position of the other axis may change the test result. This could also be a strategy point. As example: “do same test with X once with the Y maximal position and once for the Y minimal position”. The test can still be carried out if the test dependency axis is not available but it will be marked in the report.

Figure 13: Dependency Diagrams

The two examples above are already two examples of dependency groups. Inside a dependency group tests are carried out in sequent. The tests of different dependency groups are executed in parallel.

A detailed description of the strategy concept is written in the SW Detail Design document (see Ref [5]).

10.1.3 Test Run

The test run can be started and stopped by the user. One test run xml file will be generated per test run. The user has the possibility to load a test run file and continue the test run if the same instrument configuration is connected to the computer where the qualification tool is executed. Inside the test run file following data is stored:

- Included axis to the test
- Strategy used
- Collected statistic data
- Occurred errors and warnings

10.1.4 Commands

There are move and communication commands.

A prose example a move command:
"X move 300mm with 20 mm/s and 5 mm/s^2"

A prose example for a communication command:
"Configure Log Point"

10.1.5 Action

One or more commands can be assembled to one action. If the action is used to collect statistic data the action is called test action.

A prose example of a test action and the containing commands (also in prose):
"Move X forth and back 300mm with 100% speed/acceleration"

- "Configure Log Point"
- "X move 300mm with 20 mm/s and 5 mm/s^2"
- "Read statistic data"

A prose example of an action:
"Move X to position 123.45" (to enable a Z move)
"Initialize X"
(Error) "Recovery X"

10.1.6 Test Cycle

A test cycle contains one or more actions. A test cycle is the smallest part visible to the user. If the user stops a test run the current test cycle will be executed to the end and all statistic data will be saved.

A prose example of a test cycle and the containing actions (also prose):
"Test Z Axis, Speed/Acceleration Normal-Iteration, Distance Full"

- "Move X to position 123.45"
- "Move Y to position 123.45"
- "Move Z forth and back 300mm with 100% speed/acceleration"
- "Move Z forth and back 300mm with 50% speed/acceleration"
- "Move Z forth and back 300mm with 20% speed/acceleration"

10.2 Report Generating

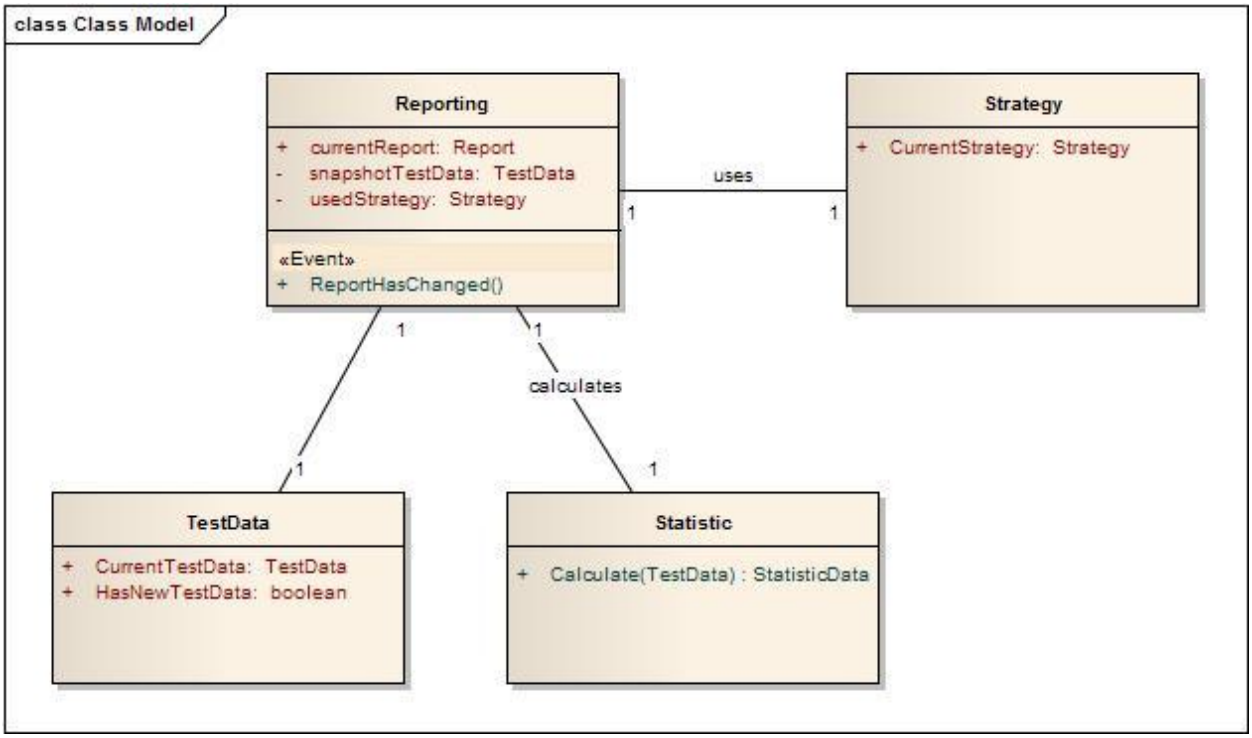


Figure 14: Report Generation Simplified Class Diagram

The reporting component is designed in a way that the system should not be blocked.

10.2.1 Test Data Handling

The test data component will be altered after each test cycle. So it changes really often. Calculating all the reports could take some time. During this time the test data should not be blocked. To encounter this problem the test data will be locked by the reporting component just for a snapshot copy. The further time consuming calculations will be done on with help of this snapshot. Then the test data will be released immediately.
Each time the test data is altered a flag will be set to indicate that the data has changed.

10.2.2 Continuous Reporting

The reporting waits till the flag of the test data is set. Then it locks the test data, creates a local copy and releases the test data immediately. With the copied data it uses the statistic component to calculate all needed values. The test data is used to know which value should be calculated (defined in the strategy).
The reporting component also provides an event other instances can subscribe. Every time a new report is available this event will be fired. The instances can then access the reporting component to get the new report if needed.

10.3 MVVM

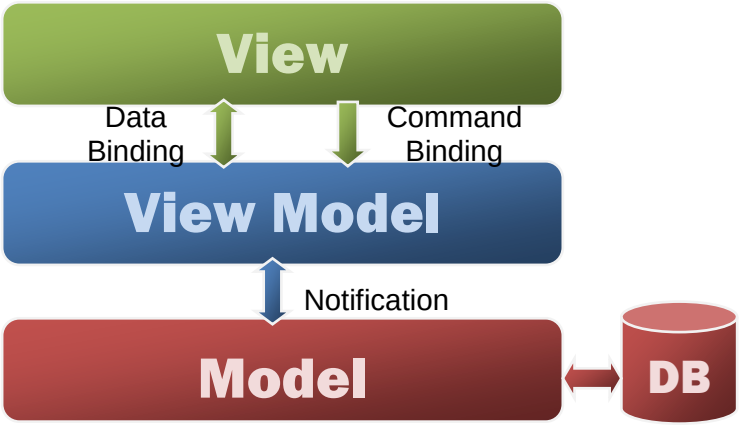


Figure 15: MVVM

The MVVM (Model – View – View Model) is a design pattern mostly used in WPF applications. The main purpose of the pattern is to isolate the business logic from the view.
Other patterns with similar functionalities are MVC (Model – View – Controller) or MVP (Model – View – Presenter).

10.3.1 Model

As in the MVC and MVP pattern the model has the task to organize the data access and business logic.

10.3.2 View

The view refers to all elements displayed in the GUI such as buttons, windows, graphics and other controls.

10.3.3 View Model

The new part of the MVVM pattern is the View Model. It is “a model of the view”, an abstraction of the view that also serves as in data binding between the model and the view. It converts the information provided by the model to a data structure used in the view and vice versa. It also interprets user actions and passes them on to the model.

10.3.4 Data Binding

Data binding is one or two way. The GUI control will always show the data the bounded model view property holds. If the user changes a value from a two way bounded property the setter from the model view property will be executed. This way changes can be forwarded from the Model View to the Model.

10.3.5 Command Binding

Commands are also direct bounded to a method in the View Model. They will be executed if the user uses the controls that are bounded.

10.3.6 Notification

The Notifications between the Model View and the Model are synchronal method calls or event based calls. View Models can subscribe them to a publisher class inside the Model and update their properties if anything changes.

11 Threading View

11.1 Threading Boundaries

11.1.1 GUI Thread

The graphical user interface is running in a GUI thread to guarantee no freezing windows.

11.1.2 Scheduler Threads

The scheduling unit runs in its own thread. The scheduler collects requests from executors to calculate new test cycles. The scheduler will then calculate them with help of the test cycle calculator.
For each dependency group also one thread will be generated. These threads are the executer components. They waiting still there is work (test cycles) which can be carried out. On execution and communication with the Tecan Base SDK this threads have to call the wait method. Therefore, they are running in an own worker thread so they won't block other processes.

11.1.3 Data Handling Thread

Also the whole data handling component package is running in its own thread. It won't block the GUI (running in the GUI thread) and the scheduler unit can still access the test data for updating. As no component needs to lock the test data for a longer time a simple lock with other components waiting is enough.

11.2 Producer Consumer Pattern

The producer consumer pattern describes a pattern where two processes share a common buffer. One component is responsible for filling this buffer and one consumes the content. The buffer is a queue; the content will be consumed according to the FIFO (first in first out) principle.

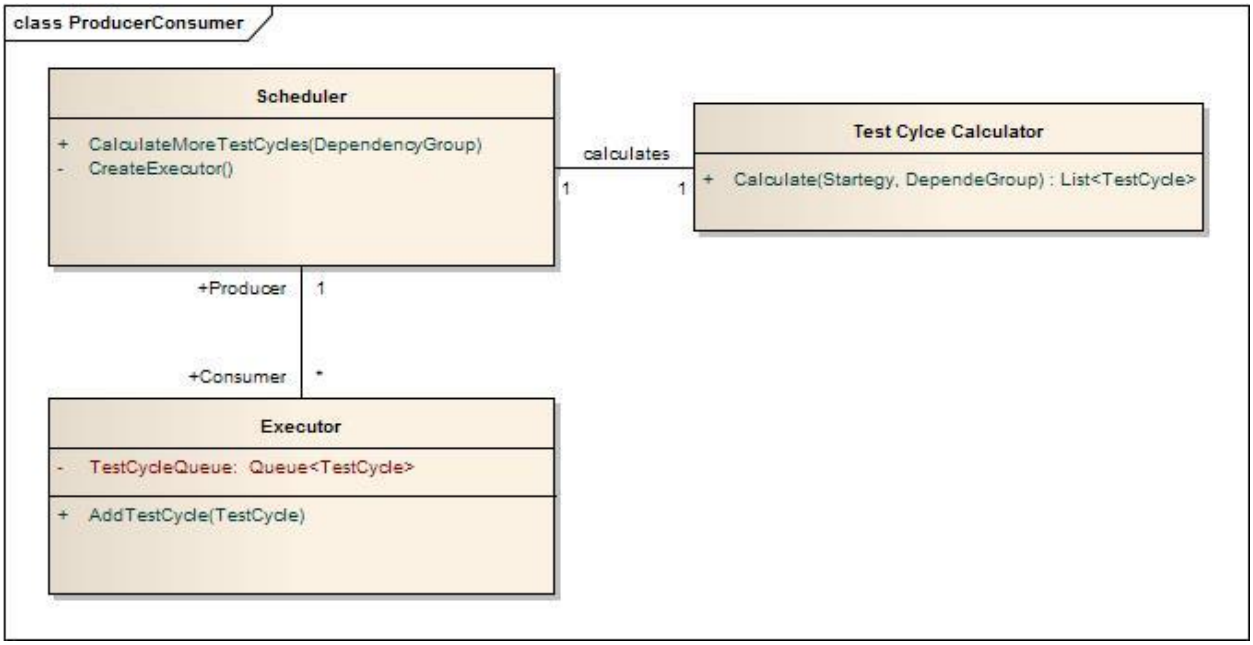


Figure 16: Producer Consumer Relationship of Scheduler and Executors

The scheduler is the producer and the executors are the consumers. The scheduler creates an executor per dependency group. Each executor owns a queue where the test cycles are stored which the executors shall execute. The scheduler calculates new test cycles with help of the test cycle calculator. This new created test cycles can be added to the executors queue through an adding method which guarantee thread save processing.

Executors have to run in their own thread. Communication and move handling with the Tecan Base SDK relies to some extend on thread waiting and thus would block other process.

12 Infrastructure View

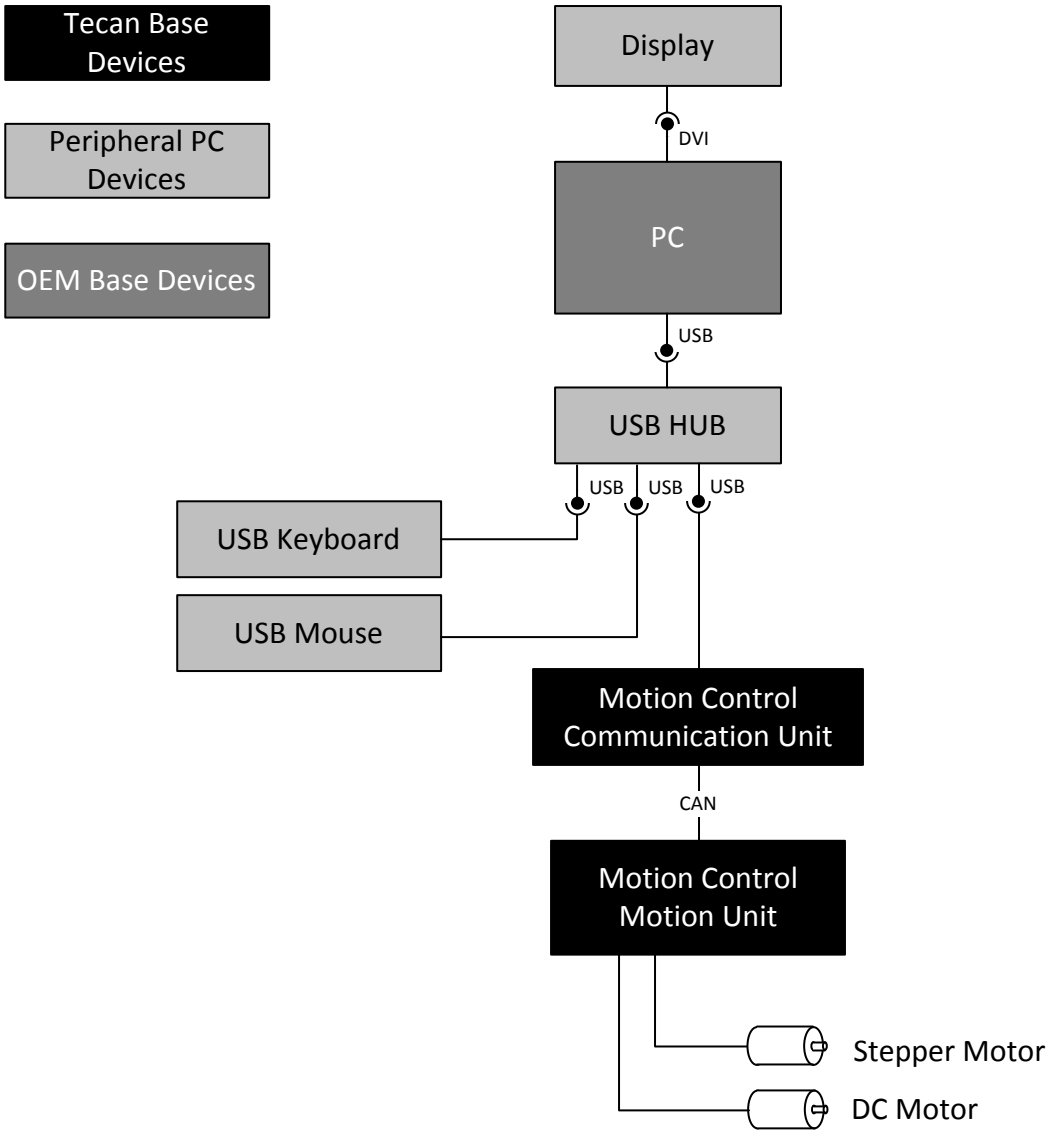


Figure 17: Infrastructure

The qualification tool is running on the PC. Normally a display is connected to that PC using a DVI or VGA port driven by the OS. To connect a Tecan instrument to the PC, it has to provide an USB port. Normally the PC hosts a USB hub and provides more than one USB port. To them may some other peripheries like a keyboard or a mouse is connected. The module controlled by an embedded motion control communication unit transfers commands from the USB port to the CAN bus and back. To the CAN bus devices can be connected. These devices are controlled by the embedded motion control motion unit. For those modules the Tecan Base SDK provides device drivers. Normally to the module controlled by the embedded motion control motion unit DC or stepper motors are connected.

13 Deployment View

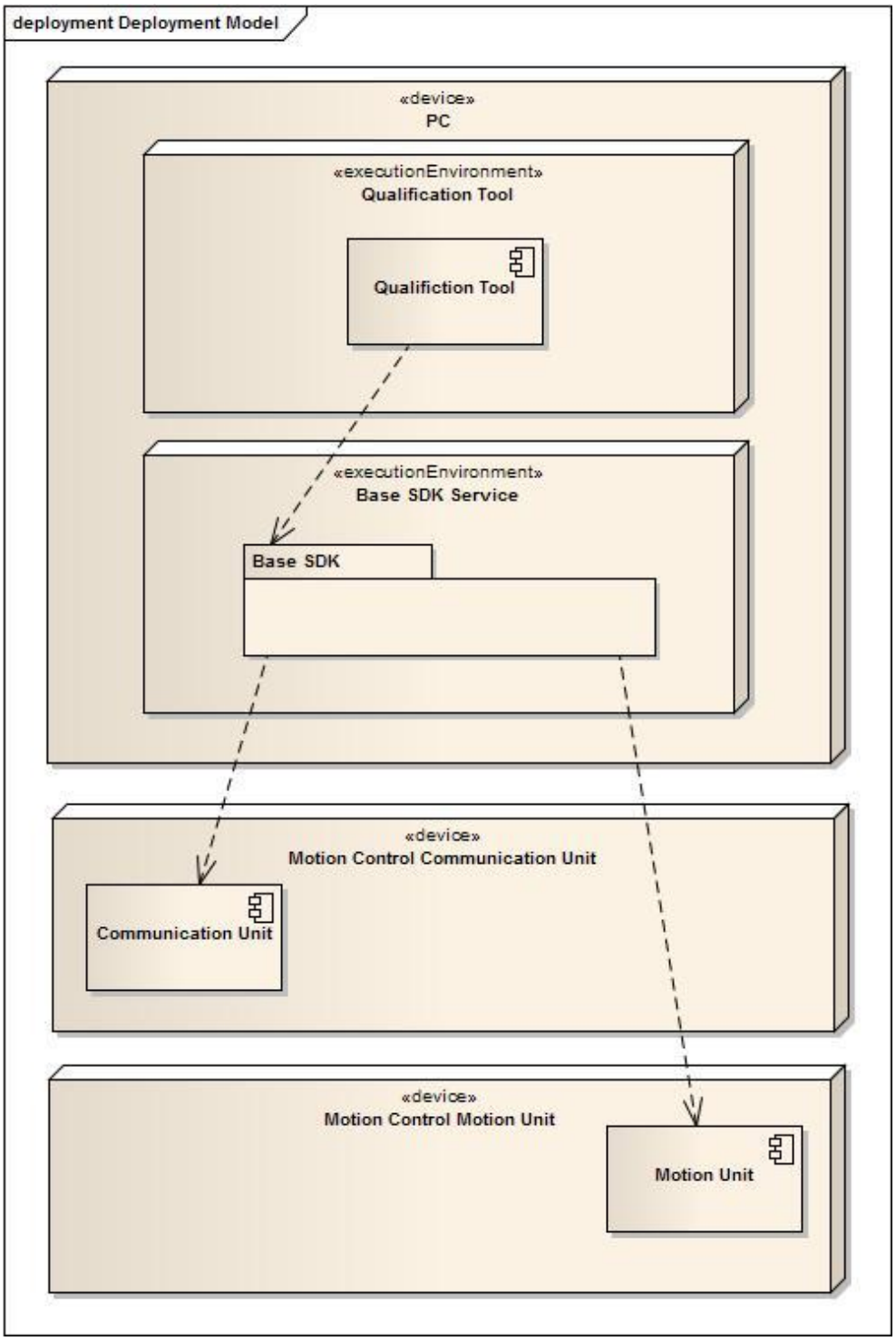


Figure 18: Deployment

The qualification tool runs in its own process on the PC. The Tecan Base SDK also runs in its own process. The embedded components (Communication Unit and Motion Unit) are executed on an embedded processor on dedicated PCBs. To install (download firmware binaries) and configure the firmware the Tecan Base SDK provides drivers.

14 Operational View

14.1 Logging

The logging system of Tecan Base SDK will be used.

14.2 Location of Base Binaries

The binaries of the qualification tool are located in:

```
Program Files\Tecan\QualificationTool
```

15 Security View

No specific security aspects exist for the qualification tool.

16 Data View

16.1 General Handling of Qualification Tool Files

It is highly recommended to save all files used and generated by the qualification tool on a common drive with backup.
The handling of publishing strategy files or test data files is not handled by the tool (in its first version).

16.2 Strategy File

Each strategy is saved in its own XML file.

The design of the strategy file is evaluated and implemented during the design output phase.

16.3 Test Data

The test data will be saved in an XML file. For each test run a new file will be generated. The tool can load one of these XML files to continue the test run. In this case no new file will be generated; the old one will be extended.

The design of the test data file is evaluated and implemented during the design output phase.

16.4 Instrument Configuration

The instrument configuration is handled by the Tecan Base SDK and so has to be available and fully functional.

17 Technology Section

C# and the.net framework V4.0 was chosen for the qualification tool because:

- The Tecan Base SDK shall be used.
- The target operating systems are Windows XP and Window 7 and C# and the .net framework are supported for them both in a 32 and a 64 bit version.
- The delivered binaries are CPU unspecific, the operating system cause if the binaries are executed in a 32 or 64 bit environment.
- C# is the latest development of the C language family.
- C# has features like garbage collection avoiding memory leaks and thus enhancing software stability.
- C# allows reusing legacy C and C++ code.
- C# allows interfacing to legacy C and C++ components.
- .net framework V4.0 is a well establish library.
- .net framework V4.0 contains feature rich technologies like WPF.

WPF is used for all GUI components, preferably by creating controls, allowing a state-of-the-art user interface.

18 Architectural Justification

18.1 General

The Tecan Base SDK has to be used and so many architectural decisions are already made, like using the Tecan Base SDK remoting services.

18.2 Presentation Layer

The MVVM patter is common used and for simply little tools like this one a simple but powerful support.

18.3 Simple but Effective Architecture

The tool shall be used by a team not trained in .net C#. They may have experience with the language, but this is no pre condition. Nevertheless they should be able to debug the tool if as example an error occurs. Also the further development is not secured to be done by one of the software team members.

19 Appendix

19.1 Table of Figures

Graphical User Interface Design

Project-Name: **Qualification Tool**

Project Number: **-**

Subject: **-**

	Author	Reviewer	Approver
Name	Andreas Zollinger	Luc Bläser	Joas Leemann
Function	Software Engineer	Supervisor HSR	Project Leader
Date / Visa			

Table of Contents

1 Introduction

1.1 Purpose

The GUI design describes the look and behavior of the windows and screen. It further describes the connection between the user controls and the functions required in the SWS.

1.2 Scope

This document belongs to the Qualification Tool. This document is generated during the “Design Input” phase and is updated during the “Design Output” phase.

1.3 Definitions, Acronyms and Abbreviations

Definitions, acronyms and abbreviation can be found in the global table (see Ref. [1])

1.4 References

Ref #	*Description*
Ref. [1]	Definition, Acronyms and Abbreviations for Qualification Tool, 90_DefinitionAcronymsAbbreviations.pdf, V1.0
Ref. [2]	Traceability Matrix for Qualification Tool, 91_TraceabilityMatrix.pdf, V1.0

1.5 Document Change History

Date	*Version*	*Change*	*Author*
2012-06-11	1.0	Initial version	AnZo

2 Paper Prototype

2.1 Why Paper Prototyping?

Goal of the paper prototype is to confront the customer early in the project with a possible design drawn on paper. Following advantages can be gained for this project:

- The customer can be included early in the developing phase. Big mistakes can be recognized early in the project and corresponding counter measurements can be set up to a low price (time).
- A paper prototype is done fast proportional to the time needed to do a first SW implementation.
- Because the design is made on paper, the customer criticizes more and generates more change requests, because it is “just” done on paper. Also the customer is in many cases more creative in supporting with ideas.

All these advantages results in a better and more usable tool for the customer.

2.2 Creating

The paper prototype is created with following tools:

- 4mm scaled paper
- pencil, 2B
- marker, red, orange and green

2.3 Interview

After the paper prototype is done two or three interviews will be done.
These interviews are filmed and analyzed. The interviewee will be absolute anonymous. The interview is held in the native language of the interviewee so the interviewee is not distracted. The interviewee is part of the department of the customer and knows the circumstances and dependencies of the occurring motors and modules he will be confronted with.
Prior to the interview questions and tasks were defined. The interviewee shall try to carry out these tasks.

2.3.1 Task 1: Start up and run a simple test run

2.3.1.1 Storyline

Described to the interviewee at the start:
The line manager of the interviewee has assigned a new task to the interviewee. The proband shall use the new qualification tool to gather some standard data for a new assembled instrument. The line manager is interested in the X, Y and Z motor of the gripping module and in the R motor of the centrifuge module.
The computer with the installed software and the hardware are already set up and ready to use.

Described to the interviewee after he has started the test run:
The pipetting module also initializes. Not described will be why. (Pipetting module has a dependency to the gripping module and so has also to be initialized.)

2.3.1.2 Expected Behavior

- starting up application and standalone server (Tecan Base SDK component)
- selecting/deselecting axes
- starting test
- stopping test

2.3.2 Task 2: Test with missing motors

2.3.2.1 Storyline

Described to the interviewee at the start:
Once again the line manager of the interviewee requests a test run on the same instrument as in task 1. This time all modules shall be tested. This test was executed once yesterday. The old test run shall be loaded and continued. Additional the short distance strategy shall be used. The exported reports from the tool shall be delivered after a 24 hours long test.

Described to the interviewee after starting up:
The pipetting module motors seems to have a problem (red dots). They cannot be checked (disabled checkbox). If the proband asks about the pipetting module (or if he mentions earlier during task 2 that he inspect the instrument if all is right) it will be described to him that in fact the hardware pipetting module is missing. After consultation the line manager the test shall be started without the pipetting module.

2.3.2.2 Expected Behavior

- starting up application and standalone server (Tecan Base SDK component)
- loading old test run
- asking about the pipetting module
- selecting/deselecting axes should not be done (all available motors already selected)
- starting test
- stopping test
- exporting the gathered data

2.4 Design

2.4.1 Environment Paper Parts

2.4.1.1 Desktop

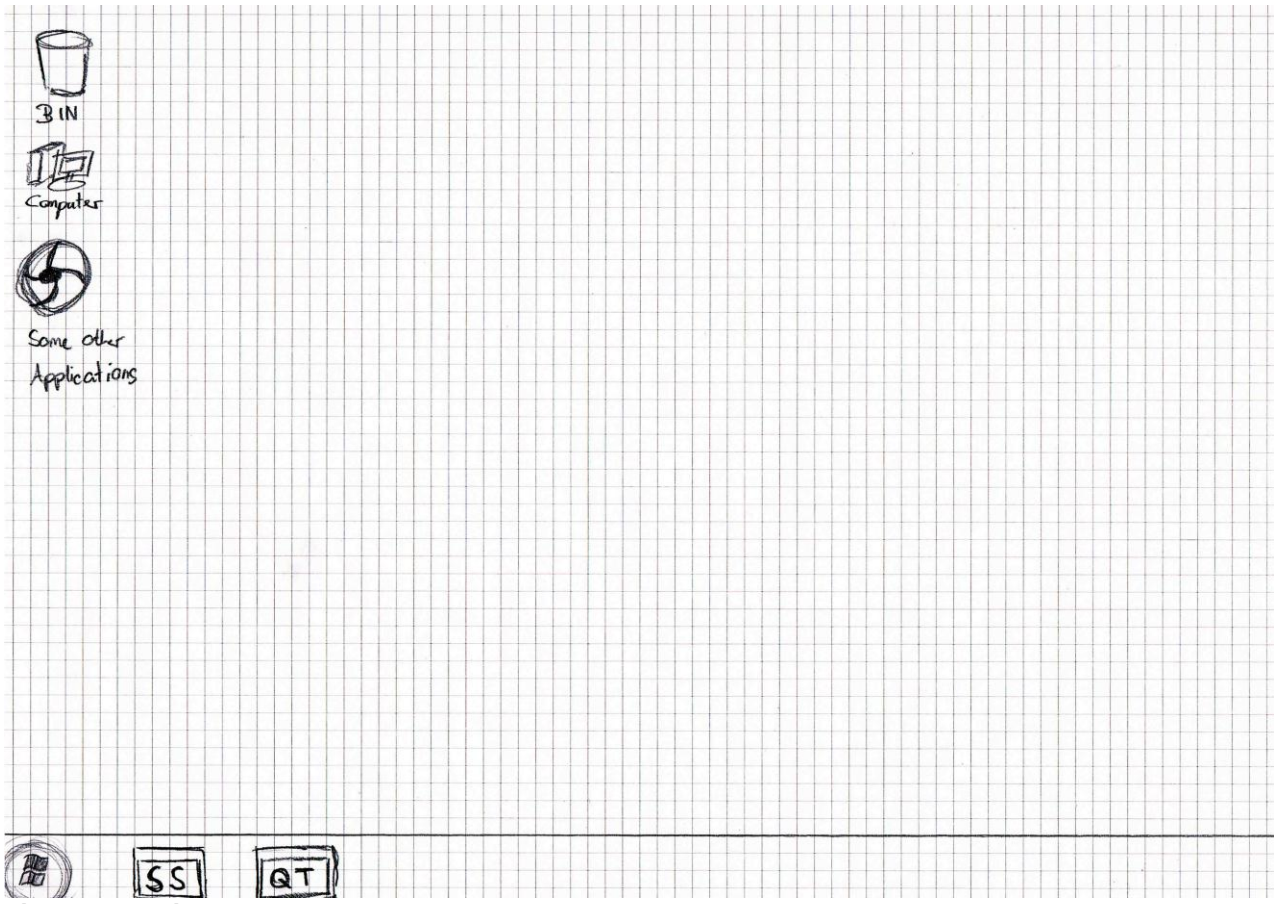


Figure 1: Desktop

“Figure 1: Desktop” shows the desktop screen of the computer used for the interview. The format of this paper is DIN A4 and symbolized the maximum computer screen size.

2.4.1.2 Taskbar

Most interesting for the interviewee is the task bar with the SS and QT icons. One is the standalone server used by the Tecan Base SDK the other one is the qualification tool.

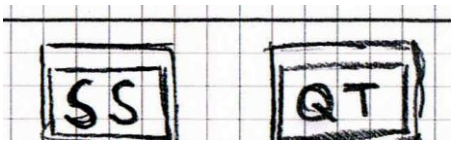


Figure 2: SS and QT not started

Figure 3: SS started

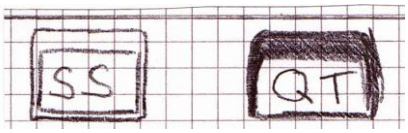


Figure 4: QT started

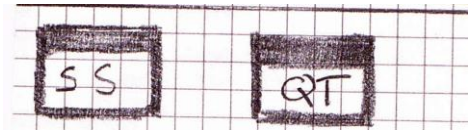


Figure 5: SS and QT started

The four figures “Figure 2: SS and QT not started”, “Figure 3: SS started”, “Figure 4: QT started” and “Figure 5: SS and QT started” represent the 4 different possible status of the two tools running. According to what application the interviewee starts the taskbar will change.

2.4.1.3 Standalone Server

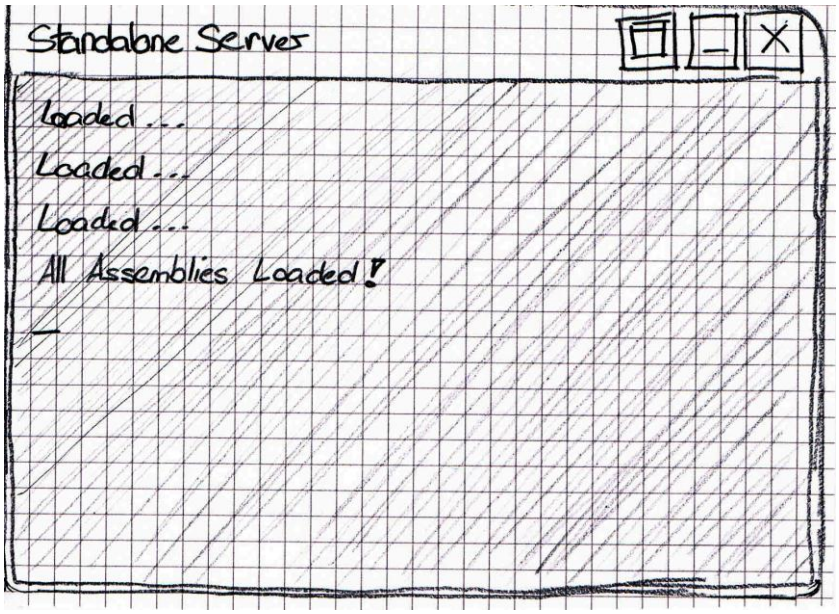


Figure 6: Standalone Server

This paper represents a commando line application used by the Tecan Base SDK.

2.4.2 Qualification Tool

2.4.2.1 Sections of the Screen

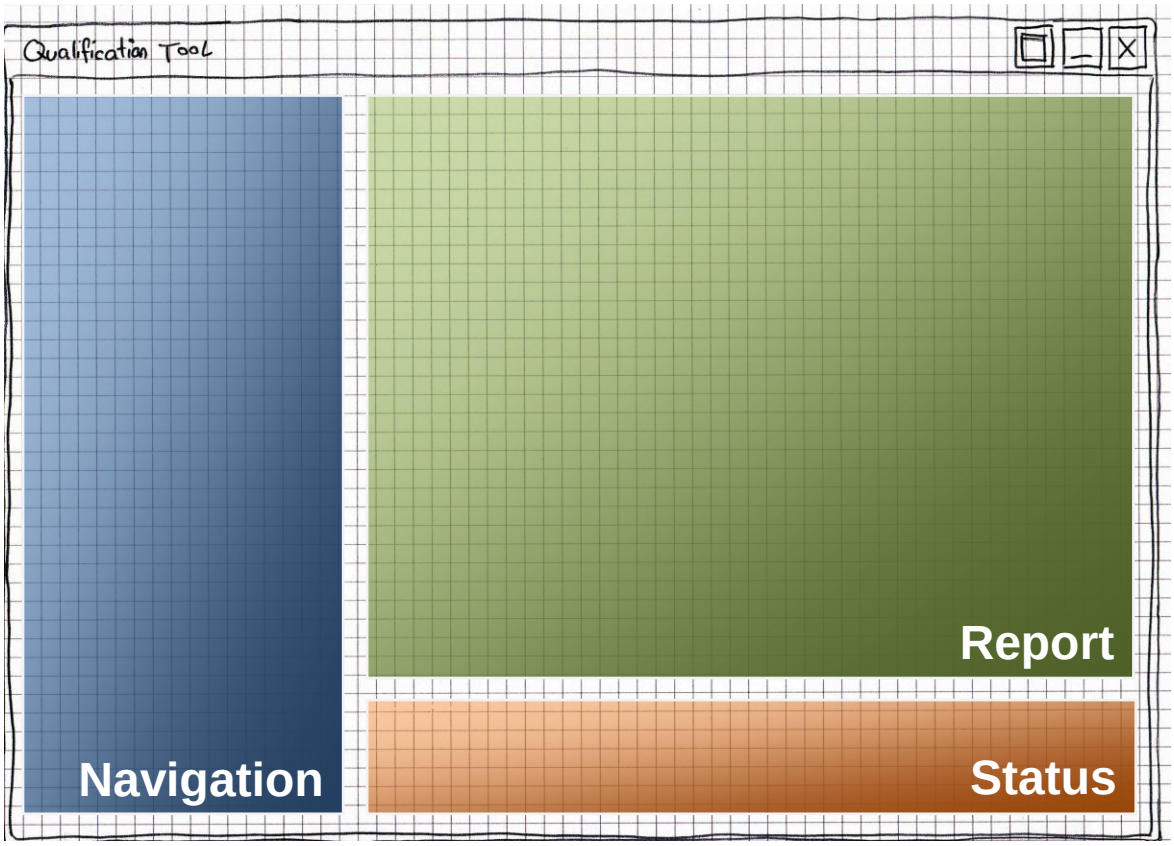


Figure 7: Sectioning

The main window will be spitted into 3 main regions. On the left is the navigation where mainly all user inputs will be handled. On the bottom is the status area where the user is informed about the status of the instrument and if any problems occurred. The biggest part is reserved for the report view. The charts and diagrams of the report will use much space that a trained user can use them.

2.4.2.2 Navigation

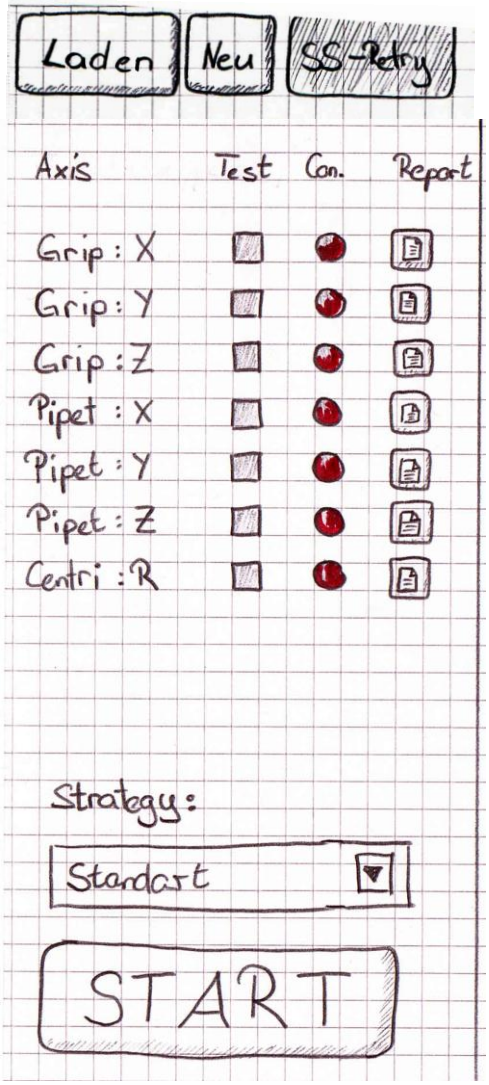


Figure 8: Navigation Section

The navigation sections itself is divided into another three parts.

The top most part containing two buttons to handle old test runs and one to thrigger a connection retry with the Tecan Base SDK service.

The part in the center is a list of all axes.

At the bottom a strategy can be selected through a drop down menu. The “start/stop” button (“START”) starts a test run or ends it if currently one is running.

2.4.2.2.1 Load

Figure 9: Load Button

With the load button an old test run can be loaded. With the loaded test run the already generated reports can be looked at. If the same instrument with the same configuration is connected the old test run can be continued.

2.4.2.2.2 New

Figure 10: New Button

With the new button the current selected axes and strategy will be reset. If an old test run is currently loaded it will be unloaded with a click on the new button.

2.4.2.2.3 SS-Retry

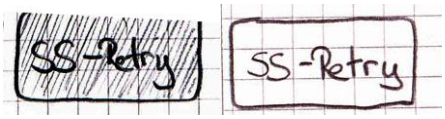


Figure 11: SS-Retry Statuses

The “standalone server retry” button can be clicked, if on startup of the qualification tool the Tecan Base SDK service was not running. With a click on this button a connection retry will be triggered. So a restart of the application can be avoided. If in the meantime the service was started, the button will be disabled if the connection try worked successfully.

2.4.2.2.4 Axis List

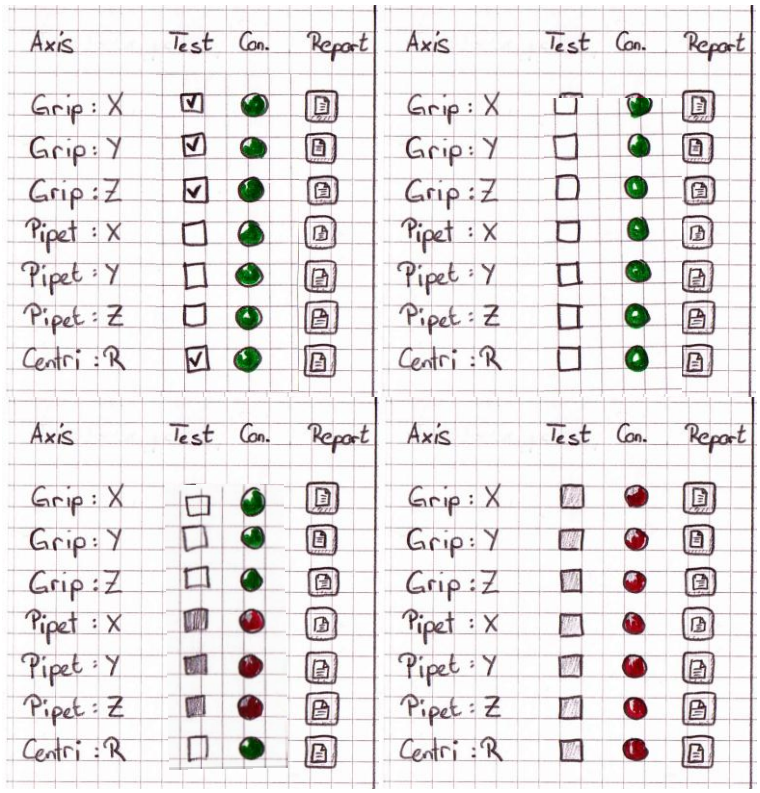


Figure 12: Examples Statuses of the Axis List

Next to the name of the axes each axis can be selected individually for the test through a check box. The connection “bulb” (can be red or green) indicates the connection and status of each individual axis. If something is wrong with the axis (bulb is red) the axis check box is disabled.

With a click on the report button of each axis the related test report will be displayed in the report section.

2.4.2.2.5 Start Stop Button

Figure 13: Statuses of the Start Stop Button

The start stop button can start or stop a test run. If before starting not enough information is available (as example no axis is selected) the start button is disabled.

2.4.2.3 Report

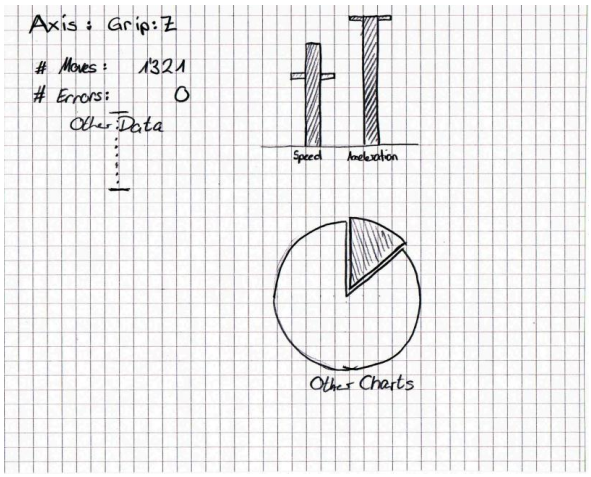


Figure 14: Report Screen Place Holder

At the early stage of the project where the paper prototype was done it was not defined what and how exactly the test results shall be provided. This paper acts as a general place holder for all report screens. The report section is mostly excluded from the paper prototype testing.

2.4.2.4 Status

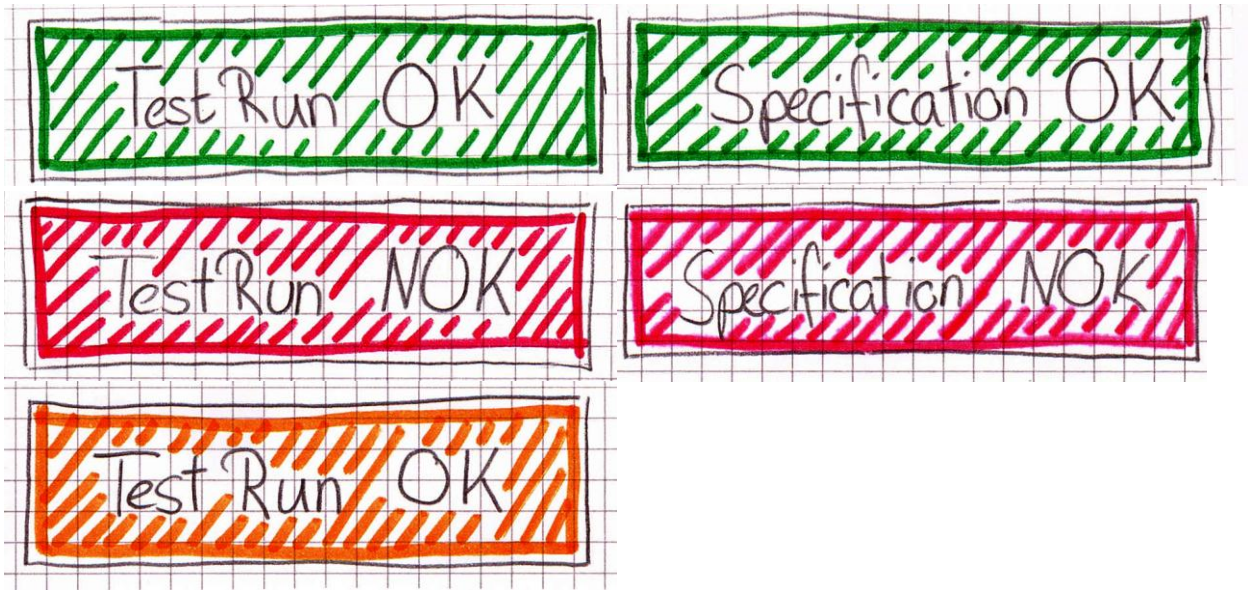


Figure 15: Status Statuses

At the bottom of the screen two eye catching status indicators are placed.
The test run status indicates the health of the instrument. A green color symbolized that everything is working, where the red one shows that something went wrong. The additional orange colors hints to a error that could be auto recovered by the qualification tool.
The specification status indicates if the gathered test data meets the requirements. Green indicates that all measured data fulfill the specifications, red that one or more values are not good enough.

2.4.2.5 Example View

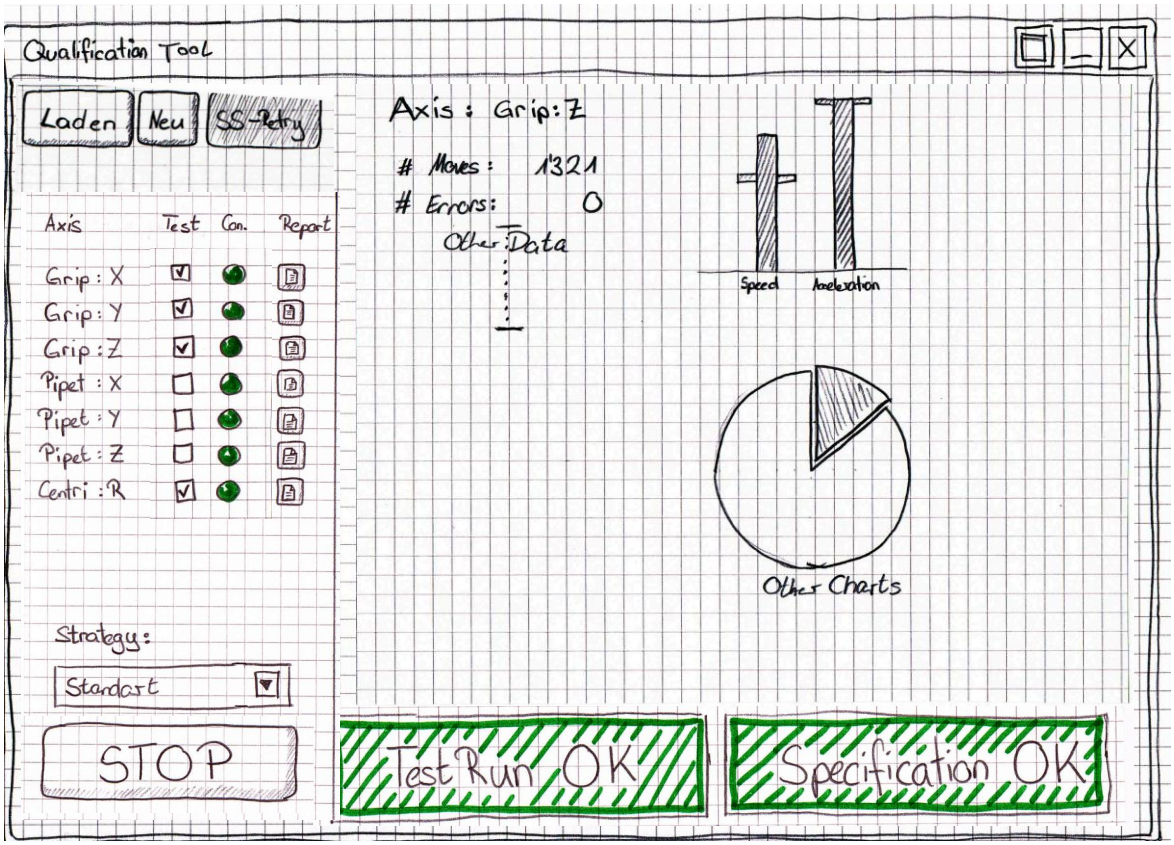


Figure 16: Example View

The screen above shows a possible view the interviewee can meet.

2.5 Interview

2.5.1 Interview 1

2.5.1.1 Main Problems

- The interviewee could not start the tool with the Tecan Base SDK service running in the background. The concept with the retry button was not self-explanatory. After helping the user could then continue with his task.
- The interviewee mentioned that the new button does not what he would suspect, the name is misleading.

2.5.1.2 Suggestions

- Navigation should support the user with more states. Red and green may not be enough.
- E-Mail notification would be really nice.

2.5.2 Interview 2

2.5.2.1 Main Problems

- Also the concept with the standalone server retry was not understood. This user even went to the windows services and started the Tecan Base SDK manually, what resulted in the wished effect, but does not reflect the way the process was planned.
- The user was too confused with the fact that the pipetting module also was initialized. The interviewee immediately stopped the test run and tried all versions of checking and unchecking axes, even started to change strategies. The interviewee had to be interrupted in he's doing and the situation had to be clarified.

2.5.2.2 Suggestions

- E-Mail notification was also mentioned.
- The user must be better supported with the Tecan Base SDK service problem. Maybe introduce a status bar or something like that.
- Splash screen on startup is mandatory, because he would start the application twice if it would take too long.

2.6 Planned Changes from Paper Prototype

- Axis list must be more informative.
- Enhance the Tecan Base SDK service support.
- The tool shall suggest selecting the strategy first before selecting axes. This can be achieved by shifting the strategy drop down above the axis list.

3 SW Prototype

3.1 Design

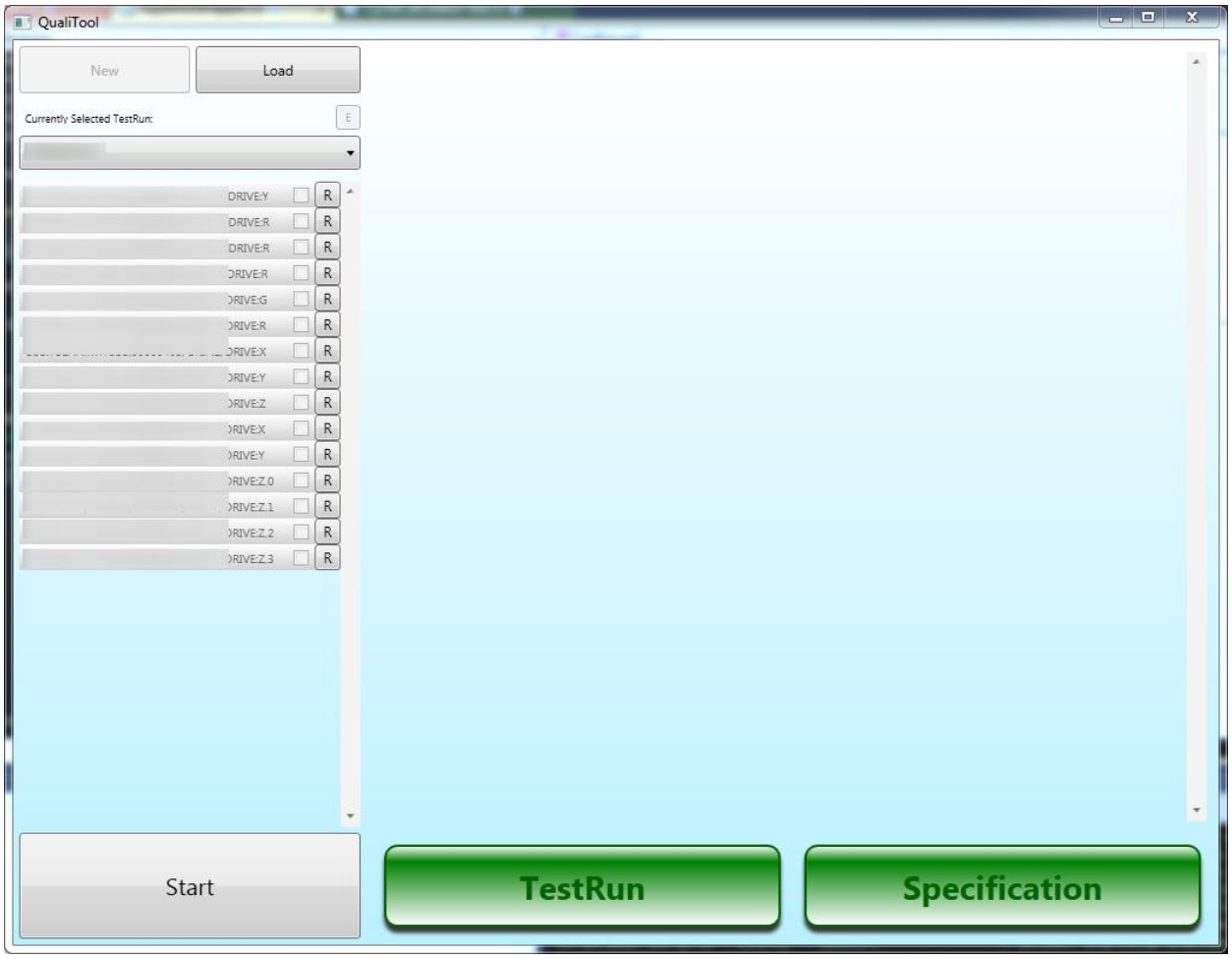


Figure 17: Startup View

The main ideas and design of the application was directly taken from the paper prototype. The structure was positively taken by the interviewee and by other gained informal feedback. As this is the case in this chapter only the main features of the GUI will be featured.

3.2 Strategy and Axis List

That the strategy drop-down is now above the axis list is one of the changes from the paper prototype. The axis list will display all axes described in the strategy. They are however not selectable.

Figure 18: Change Strategy

With help of this feature the user gets immediately feedback which axes can be tested with the strategies. Simple but common errors like misspelling axes can be detected quiet easily.

3.3 Axis Color

Axes have different colors according to the state they have.

3.3.1 In Strategy but not Connected

Figure 19: Axes in Strategy

Axes are grayed out and disabled if they are described in the strategy but not connected to the computer.

3.3.2 Connected but not in Strategy

Figure 20: Connected Axes

Axes which are connected to the computer but are not described in the strategy are also grayed out and disabled.

3.3.3 Available Axes in Strategy

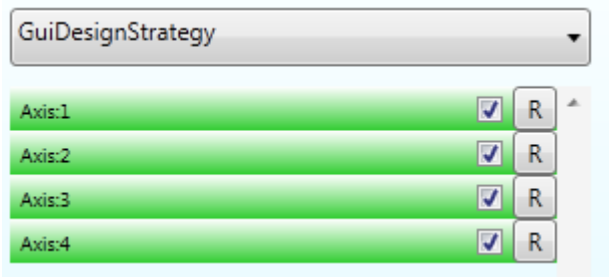


Figure 21: Available Axes in Strategy

Axes available in the strategy and connected with the computer will be checked automatically. They appear with a green background to signal that they are ok and will be tested.

3.3.4 Deselected and in Strategy

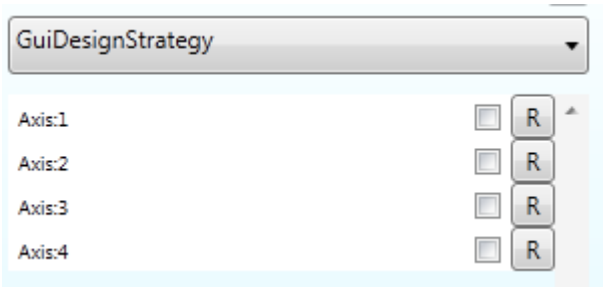


Figure 22: Deselected Axes Available in Strategy

If axes are defined in the strategy and are connected to the computer but were disabled by the user they appear with a white background are still selectable.

3.3.5 Axes with a Dependency

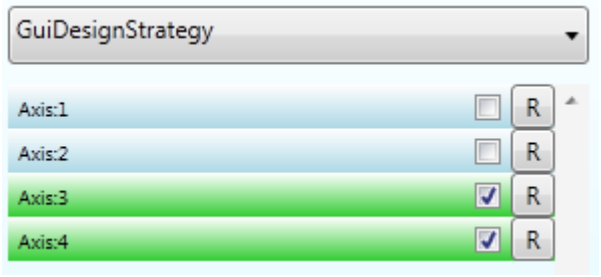


Figure 23: Axis with Dependencies

If an axis has some dependencies to other axes which are not selected, then the not selected axes which are still needed are colored in a light blue. This shall tell the user that the axis will not be tested, but still will be initialized and moved during test runs because there are axes who have a dependency to this axis.

3.3.6 Missing Axes

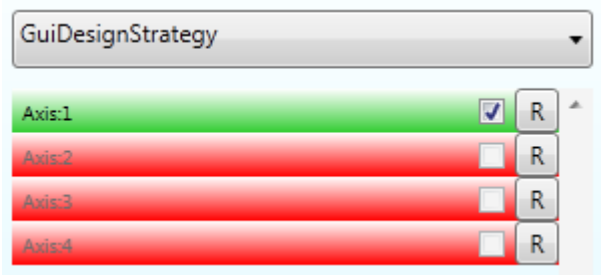


Figure 24: Missing Axes

If an axis has a dependency to another axis, but this axis is not connected to the computer, the missing axis will be higlighted with a red color. This indicates that the test run can not be started because the strategy describes a dependenc which is nto available.

3.3.7 Missing Axis in the Strategy

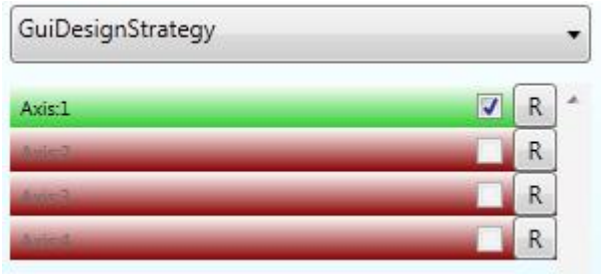


Figure 25: Missing Axes in the Strategy

As a last possibility it could happen, that an axis has a dependency described, but this dependency axis is not described itself in the strategy. This uncommon variant will be highlighted in a darker red.

3.4 Status Bar

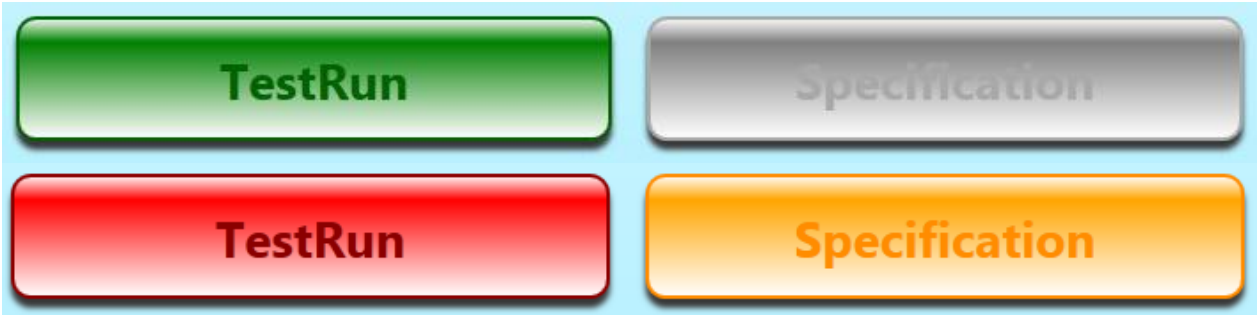


Figure 26: Status Bar Statuses

The status bar is designed quite big for its simple function and so should fulfill the requirement of an eye catching user control. Next to the normal green variant (everything is fine), the status could also be red (error state), orange (warning) or gray (offline/disabled).

3.5 Current Test Run and Exporting

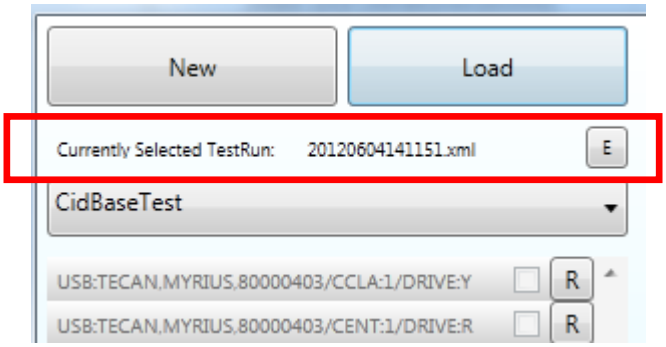


Figure 27: Current Test Run and Exporting

Right under the new and load button the current running test run is displayed. With the export button a standard save dialog opens where the user can save an exported version of the current test run.

3.6 Reporting

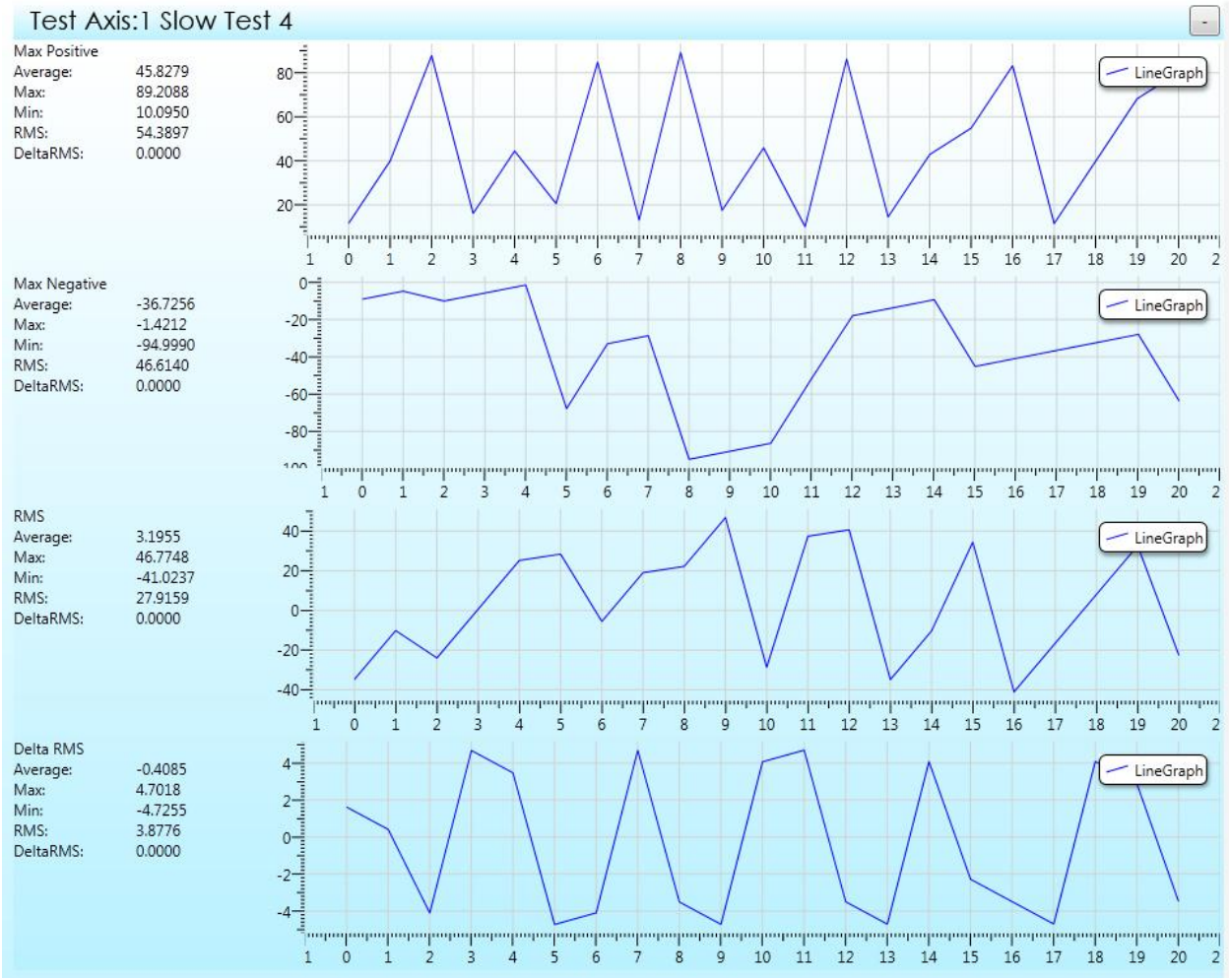


Figure 28: Reporting View

The test view displays the four values over time which will be reported from the firmware during one test. Additional information is provided on the left side of each line chart.

4 Applicable Standards

4.1 Implementation Technology

For implementation of the GUI .net WPF is used. With this framework the common used pattern MVVM will be used.

4.2 GUI Behavior

- The GUI provides the user immediate feedback, there is no “freeze screen”
- The GUI presents values as editable to the user only if they really are editable.
- The GUI only presents an information text that this section does not contain anything instead of a simple empty field. (Example: an empty list of motors.)
- The GUI uses simply validation and gives to the user (Example: the GUI validate if the input is a number on a number field but does not validate if the number is applicable.)
- Command inputs (like button or menu) are grayed out or even hidden if not applicable.
- Icons and labels should be chosen like other typical Microsoft products and have the expected behavior.

5 Traceability

The traceability is handled inside the global traceability matrix file. (See Ref. [2])

6 Appendix

6.1 Table of Figures

Software Configuration Management Plan

Project-Name: **Qualification Tool**

Project Number: **-**

Subject: **-**

	Author	Reviewer	Approver
Name	Andreas Zollinger	Luc Bläser	Joas Leemann
Function	Software Engineer	Supervisor HSR	Project Leader
Date / Visa			

Table of Contents

1 Introduction

1.1 Purpose

The purpose of this document is to describe the configuration management during the different phases of the SW Product Life Cycle of the Qualification Tool. It defines what has to be managed.

1.2 Scope

The scope of this document is to help and guide software developers and testers through the configuration management focusing on the software tools.

1.3 Definitions, Acronyms and Abbreviations

Definitions, acronyms and abbreviation can be found in the global table (see Ref. [2])

1.4 References

Ref #	*Description*
Ref. [1]	Project Development Plan for Qualification Tool, 01_ProjectDevelopmentPlan.pdf, V1.0
Ref. [2]	Definition, Acronyms and Abbreviations for Qualification Tool, 90_DefinitionAcronymsAbbreviations.pdf, V1.0

1.5 Document Change History

Date	*Version*	*Change*	*Author*
2012-03-12	1.0	Initial Version	AnZo

2 Description

2.1 Organization, Responsibilities and Interfaces

The responsibilities and roles within the project are described in the Project Development Plan of this Project (see Ref. [1]).

2.2 Tools, Environment and Infrastructure

2.2.1 Tools used during Development

Name	*Manufacturer*	*Unique Identifier (eg. Version or Release Date)*
Team Foundation Server (TFS)	Microsoft Corporation	Version 2010
Visual Studio	Microsoft Corporation	Version 2010
Enterprise Architect	Sparx Systems	Version 7.5 or newer

2.2.2 Development Environment

Windows XP 32 bit, Windows 7 32 bit and Windows 7 64 bit
.Net Framework 4.0

2.2.3 Target Environment

Windows XP 32 bit, Windows 7 32 bit and Windows 7 64 bit
.Net Framework 4.0

2.3 Control of Tools, Environment and Infrastructure

During development the tool is not compiled on a specific computer.
After the tool is released, compiling will be done on a PC where all the relevant tools are installed. The access to this building environment is limited.

2.4 Configuration Identification

2.4.1 Software Versions

Name of Configuration Item	*Version*	*Used in SW Version*
Tecan Base SDK	0.9.3	-

2.4.2 SOUP Components

Name	*Manufacturer*	*Unique Identifier (eg. Version or Release Date)*
Windows 7 64 Bit	Microsoft Corporation	-
Windows XP 32 Bit	Microsoft Corporation	

2.4.3 Build Mechanism

Complete source code and all additional files have to be in proper place.
The compiling will be done inside Visual Studio 2010.
The executable program is placed in the standard Tecan Base SDK place.

2.4.4 Versioning

No versioning will be done during this project.
At the end of the project the current software state will be labeled as draft version 0.1.

Software Detail Design

Project-Name: **Qualification Tool**

Project Number: **-**

Subject: **-**

	Author	Reviewer	Approver
Name	Andreas Zollinger	Luc Bläser	Joas Leemann
Function	Software	Supervisor HSR	Project Leader
Date / Visa			

Table of Contents

1 Introduction

1.1 Purpose

The SDD establishes a lower level design of the software identified in the SSD and completes the software design documentation providing sufficient information from which programmers can code.

If necessary, the SDD also describes the detailed design of one or more interfaces between one or more applications and other configuration items or critical items.

1.2 Scope

This document is generated during the design output phase and will be maintained till the end of the project.

1.3 Definitions, Acronyms and Abbreviations

Definitions, acronyms and abbreviations can be found in the global table (see Ref. [1])

1.4 References

Ref #	*Description*
Ref. [1]	Definition, Acronyms and Abbreviations for Qualification Tool, 90_DefinitionAcronymsAbbreviations.pdf, V1.0
Ref. [2]	Traceability Matrix for Qualification Tool, 91_TraceabilityMatrix.pdf, V1.0

1.5 Document Change History

Date	*Version*	*Change*	*Author*
2012-05-06	0.1	Initial Version	AnZo

2 Software Detail Design

2.1 Strategy

The strategy defines how a test run should be carried out. More than one strategy can exists for more than one test behavior. The user can select a strategy prior to starting the test run.

2.1.1 File Handling

Strategies are stored in form of XML files. Per strategy one file must be generated. On starting up of the qualification tool these files will be tried to deserialized and. All formally correct strategies are presented to the user.

2.1.2 XML Structure

2.1.2.1 Root

The root element of the XML is the strategy itself. The strategy has a `<Name>`. All other further details are stored in individual `<AxisStrategies>`.

```
<?xml version="1.0" encoding="utf-8"?>
<Strategy xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <Name>Test</Name>
  <AxisStrategies>
    ...
  </AxisStrategies>
</Strategy>
```

Example 1: Strategy XML Root Element with Sub Elements

2.1.2.2 AxisStrategies

For each axis a single strategy is described.

```
<AxisStrategies>
    <AxisStrategy Id="GRIP:1/DRIVE:X">
        ...
    </AxisStrategy>
    <AxisStrategy Id="GRIP:1/DRIVE:Y">
        ...
    </AxisStrategy>
    <AxisStrategy Id="GRIP:1/DRIVE:Z">
        ...
    </AxisStrategy>
</AxisStrategies>
```

Example 2: AxisStrategies containing some AxisStrategy

The Id attribute of the axis strategy has to be the same as the one defined in the instrument configuration file describing the axis. The instrument configuration file is done and provided outside from this project. The axis id has to contain the string “drive” to be recognized as a movable device.
In the “Example 2: AxisStrategies containing some AxisStrategy” are three axis defined.
The Id of these three axes can lead to the assumption that these three axes have something in common (like they belong to the same device). This could be, most likely it will be the case, but in the context of the strategy or the whole qualification tool no connection between axes are done.

For comprehension of this and further examples always the same device will be used to explaining the content:

- The GRIP is a (fictional) device able to grip (like a claw crane)
- The GRIP device has 3 motors, one for each direction in a normal orthogonal space
- This three motors are the three axes and named X, Y and Z

2.1.2.3 Axis Strategy

```
<AxisStrategy Id="GRIP:1/DRIVE:X">
  <InitStrategy>
    ...
  </InitStrategy>
  <TestStrategy>
    ...
  </TestStrategy>
</AxisStrategy>
```

Example 3: Axis Strategy

The `<AxisStrategy>` contains all the information and configurations that a test can be carried out successfully. The axis strategy is divided into two sections: the `<InitStrategy>` used for initialization of the axis and the `<TestStrategy>` used during the test run.

2.1.2.4 InitStrategy

```
<InitStrategy>
  <InitDependencies>
    <InitDependency To="GRIP:1/DRIVE:Z">
      <InitDependencyDescription Type="position" Value="67" />
    </InitDependency>
  </InitDependencies>
  <MoveAfterInit>
    <Position>1250</Position>
    <Speed>1</Speed>
    <Acceleration>1</Acceleration>
  </MoveAfterInit>
</InitStrategy>
```

Example 4: InitStrategy

The `<InitStrategy>` contains two main parts.
First is the `<InitDependencies>` part. This part will be described later in the dependency chapter.
The second part is the `<MoveAfterInit>` element. How an axis shall initialize is defined in the instrument configuration file. The question here is, where the axis should move to after the initialization is done. The qualification tool has no possibility to find out how the axis will be initialized, what steps are done in this process and where the axis is located after the routine is finished. With a move immediately after the initialization process to a position defined by the strategy the axis is in a known state.

2.1.2.5 TestStrategy

```
<TestStrategy>
  <TestDescriptions>
    <TestDescription>
      <TestDependencies>
        <TestDependency To="GRIP:1/DRIVE:Z">
          <TestDependencyDescription Type="position" Value="67" />
        </TestDependency>
      </TestDependencies>
      <TestCases>
        ...
      </TestCases>
    </TestDescription>
    <TestDescription>
      <TestDependencies>
        ...
      </TestDependencies>
      <TestCases>
        ...
      </TestCases>
    </TestDescription>
  </TestDescriptions>
</TestStrategy>
```

Example 5: TestStrategy

The `<TestStrategy>` part of an axis strategy defines how and what tests should be executed. The `<TestDescription>` section contains always a pair of two sections connected together. This two are the part of the dependencies and the part of the actual test cases. How the dependencies works will be explained in a later chapter. All `<TestDependencies>` defined are valid for all `<TestCases>` described in the same `<TestDescription>`.
Because some tests may have more, less or even conflicting dependencies than other tests, it is possible to define more than one `<TestDescription>` in the `<TestDescriptions>` element. With help of this construct more than one pair of dependency/test can be defined. Inside of the `<TestCases>` one or more `<TestCase>` can be defined.

2.1.2.6 TestCase

```
<TestCase>
  <TestProperties>
    <TestProperty Type="distance" Value="-300" />
    <TestProperty Type="speed" Value="1" />
    <TestProperty Type="acceleration" Value="1" />
    <TestProperty Type="deceleration" Value="1" />
  </TestProperties>
</TestCase>
```

Example 6: TestCase

The `<TestCase>` element contains a set of type-value pair properties. The type-value pair concept will be explained in a later chapter. With help of a set of `<TestProperty>` a test case can be described.
As an example, if the `<TestCase>` from above will be copied and the line:
`<TestProperty Type="speed" Value="1" />`
will be set to
`<TestProperty Type="speed" Value="0.5" />`
the same move will be tested, expect that the axis just move with 50% of speed.

2.1.3 Type Value Pairs

As seen in the structural description part above in three different places “type value pairs” are used. These three places are:

- Initialization Dependencies
  `<InitDependencyDescription Type="position" Value="67" />`
- Test Dependencies
  `<TestDependencyDescription Type="position" Value="67" />`
- Test Case Properties
  `<TestProperty Type="distance" Value="-300" />`

For the Type and the Value attribute are strings.
The Value attribute value is most of the time a double number, but other values like “max” or “true” can also be correct, depending on the Type value.
The Type attribute value can be selected from various keywords.

2.1.3.1 Type Keyword Table

The following table shows all current available keywords, the corresponding allowed Value attribute values, the conditions and where they can used.

Keyword	*Valid Value*	*Description*	*Special Conditions*	*Init Dependencies*	*Test Dependencies*	*Test Case Property*
position	double “max” “min”	For dependencies: the dependent axis has to be on this position. The position is an absolute value in the coordinate system of the own axis. In units (mm or °)	If the axis has no range (invinity) the value keywords “max” and “min” should not be used.	X	X	
distance	double	The relative distance an axis shall move. In units (mm or °)	Following keywords are prohibited if this one is used: range_max_distance range_min_distance target range_max_target range_min_target			X
range_max_distance	double	The axis will move a random distance where the upper limit is defined by this Type Value pair. In units (mm or °)	The range_min_distance Keyword MUST be used also. Following keywords are prohibited if this one is used: distance target range_max_target range_min_target			X

range_min_distance	double	The axis will move a random distance where the lower limit is defined by this Type Value pair. In units (mm or °)	The range_max_distance Keyword MUST be used also. Following keywords are prohibited if this one is used: distance target range_max_target range_min_target			X
target	double “max” “min”	The axis shall move exact to the absolute position defined in this Type Value pair. In units (mm or °)	Following keywords are prohibited if this one is used: range_max_target range_min_target distance range_max_distance range_min_distance			X
range_max_target	double	The axis will move to a random absolute position where the upper limit is defined by this Type Value pair. In units (mm or °)	The range_min_target Keyword MUST be used also. Following keywords are prohibited if this one is used: distance target range_max_distance range_min_distance			X
range_min_target	double	The axis will move to a random absolute position where the lower limit is defined by this Type Value pair. In units (mm or °)	The range_max_target Keyword MUST be used also. Following keywords are prohibited if this one is used: distance target range_max_distance range_min_distance			X
startposition	double “min” “max”	Optional keyword. The axis will move to this absolute position before the test move itself will be executed. In units (mm or °)	Can be used additional with the “distance” or” target” keywords.			X

endposition	double “min” “max”	Optional keyword. The axis will move to this absolute position after the test move was executed. In units (mm or °)	Can be used additional with the “distance” or” target” keywords. Will work together with the keyword “returnToStart” but may be unreasonable. The “returnToStart” keyword can be interpreted as an “endposition” with the starting position as value. Single exception is the use of the “repeat” keyword. See “repeat” keyword for more information.			X
speed	double “min” “max”	The speed that will be used for the test move. In % of the maximal value taken from the instrument configuration file.	The range keywords of speed are prohibited.			X
range_max_speed	double “max”	The speed used for the test move will be random within the boundaries where this value set the upper limit. In % of the maximal value taken from the instrument configuration file.	The range_min_speed Keyword MUST be used. The acceleration keyword is prohibited.			X
range_min_speed	double “min”	The speed used for the test move will be random within the boundaries where this value set the lower limit. In % of the maximal value taken from the instrument configuration file.	The range_max_speed Keyword MUST be used. The acceleration keyword is prohibited.			X
acceleration	double “min” “max”	The acceleration that will be used for the test move. In % of the maximal value taken from the instrument configuration file.	The range keywords of acceleration are prohibited.			X
range_max_acceleration	double “max”	The acceleration used for the test move will be random within the boundaries where this value set the upper limit. In % of the maximal value taken from the instrument configuration file.	The range_min_acceleration Keyword MUST be used. The acceleration keyword is prohibited.			X

range_min_acceleration	double “min”	The acceleration used for the test move will be random within the boundaries where this value set the lower limit. In % of the maximal value taken from the instrument configuration file.	The range_max_acceleration Keyword MUST be used. The acceleration keyword is prohibited.			X
deceleration	double “min” “max”	Optional keyword. If not defined, the acceleration keyword will be taken for decelerating. The deceleration that will be used for the test move. In % of the maximal value taken from the instrument configuration file.	The range keywords of deceleration are prohibited.			X
range_max_deceleration	double “max”	Optional keyword. If not defined, the acceleration keyword will be taken for decelerating. The deceleration used for the test move will be random within the boundaries where this value set the upper limit. In % of the maximal value taken from the instrument configuration file.	The range_min_deceleration Keyword MUST be used. The acceleration keyword is prohibited.			X
range_min_deceleration	double “min”	Optional keyword. If not defined, the acceleration keyword will be taken for decelerating. The deceleration used for the test move will be random within the boundaries where this value set the lower limit. In % of the maximal value taken from the instrument configuration file.	The range_max_deceleration Keyword MUST be used. The acceleration keyword is prohibited.			X

returnToStart	“true”	Optional keyword. If this Type Value pair is present, the axis will return to the start position after the test move is done. The value of the Value will be ignored! The presence of the “returnToStart” keyword is enough.	Will work together with the keyword “endposition” but may be unreasonable. The “returnToStart” keyword can be interpreted as an “endposition” with the starting position as value. Single exception is the use of the “repeat” keyword. See “repeat” keyword for more information.			X
repeat	double	Optional keyword. If the “repeat” keyword is used the same test move can be carried more than once successively. The value defines how many times.	Attention with the “returnToStart”, “startposition” and “endposition” keywords: Moving to start and end positions is only done once! At the very start and at the very ending of the test. In between the two keywords “startposition” and “endposition” are ignored. If the axis should move back to the starting point after each single test move the keyword “returnToStart” must be used.			X
statisticsource	Integer	The value describes from which statistic source the values should be taken. The list of all available sources can be read in the corresponding FW documentation.	This keyword is allowed to appear MORE THAN ONCE in a test case! But for each “statisticsource” keyword the same test case will be executed once!			X

2.2 Dependencies

Inside the `<AxisStrategy>` two times dependencies can be defined. A dependency describes the need of one axis to another axis. For visualization and as an example the GRIP device from the last example is used:

- X and Y have no dependencies to one another.
- But if X or Y wants to move, the Z axis has to be as much up as possible.
- And on the other side, if the Z axis want to fully extract to the bottom, maybe X and Y have to be on a special position

As said before two different types of dependencies exist: `<InitDependencies>` and `<TestDependencies>`. The main difference between these two dependency groups is that the `<InitDependencies>` are not allowed to have circular dependencies.

2.2.1 Initialization Dependencies

The dependencies during initializing are used the get the correct order of initializing. For the GRIP example it would look like this for initialization:

- First the Z axis has to initialize and move to the up most position.
- Than X and Y are allowed to initialize.

In the strategy xml file it would look something like this for the X axis:

```
<AxisStrategy Id="GRIP:1/DRIVE:X">
  <InitStrategy>
    <InitDependencies>
      <InitDependency To="GRIP:1/DRIVE:Z">
        <InitDependencyDescription Type="position" Value="67" />
      </InitDependency>
    </InitDependencies>
  </InitStrategy>
</AxisStrategy>
```

Example 7: Initialization Dependencies of X Axis

2.2.2 Test Dependencies

Test dependencies are allowed to have circular dependencies as the dependencies of each axis are viewed separately during a test. Before not all the test dependency constrains are fulfilled the test move will not be executed.

The test dependency with the GRIP example would look like this:

- For a X test:
  - Z has to be at the top.
- For a Y test:
  - Z has to be at the top.
- For a Z test:
  - X has to be on a special absolute position.
  - Y has to be on a special absolute position.

The Z test dependency part of the strategy file could be something like this:

```
<AxisStrategy Id="GRIP:1/DRIVE:Z">
  <TestStrategy>
    <TestDescriptions>
      <TestDescription>
        <TestDependencies>
          <TestDependency To="GRIP:1/DRIVE:X">
            <TestDependencyDescription Type="position" Value="200" />
          </TestDependency>
          <TestDependency To="GRIP:1/DRIVE:Y">
            <TestDependencyDescription Type="position" Value="200" />
          </TestDependency>
        </TestDependencies>
      </TestDescription>
    </TestDescriptions>
  </TestStrategy>
</AxisStrategy>
```

Example 8: Test Dependency of Z Axis

2.3 Strategy Examples

In this section two examples of possible strategies are written down.

2.3.1 GRIP Z Long and Short Test

This example uses the possibility to define two different sets of dependencies. The GRIP Z axis shall be tested with long moves where the X and Y axes also have to be on position and with short moves where the X and Y axes can be ignored.

```
<?xml version="1.0" encoding="utf-8"?>
<Strategy xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <Name>Example1_Z_LongAndShortDistanceTest</Name>
  <AxisStrategies>
    <AxisStrategy Id="GRIP:1/DRIVE:Z">
      <InitStrategy>
        <InitDependencies/>
          <MoveAfterInit>
            <Position>67</Position>
            <Speed>1</Speed>
            <Acceleration>1</Acceleration>
          </MoveAfterInit>
      </InitStrategy>
      <TestStrategy>
        <TestDescriptions>
          <TestDescription>
            <TestDependencies>
              <TestDependency To="GRIP:1/DRIVE:X">
                <TestDependencyDescription Type="position" Value="200" />
              </TestDependency>
              <TestDependency To="GRIP:1/DRIVE:Y">
                <TestDependencyDescription Type="position" Value="200" />
              </TestDependency>
            </TestDependencies>
            <TestCases>
                <TestCase Id ="LongDistance">
                  <TestProperties>
                    <TestProperty Type="distance" Value="-300" />
                    <TestProperty Type="speed" Value="1" />
                    <TestProperty Type="acceleration" Value="1" />
                    <TestProperty Type="deceleration" Value="1" />
                  </TestProperties>
                </TestCase>
            </TestCases>
          </TestDescription>
          <TestDescription>
            <TestDependencies />
            <TestCases>
                <TestCase Id ="ShortDistance">
                  <TestProperties>
                    <TestProperty Type="distance" Value="-10" />
                    <TestProperty Type="speed" Value="1" />
                    <TestProperty Type="acceleration" Value="1" />
                    <TestProperty Type="deceleration" Value="1" />
                  </TestProperties>
                </TestCase>
            </TestCases>
```

```
        </TestDescription>
      </TestDescriptions>
    </TestStrategy>
  </AxisStrategy>
  <AxisStrategy Id="GRIP:1/DRIVE:Y">
    <InitStrategy>
      <InitDependencies>
        <InitDependency To="GRIP:1/DRIVE:Z">
          <InitDependencyDescription Type="position" Value="67" />
        </InitDependency>
      </InitDependencies>
        <MoveAfterInit>
          <Position>200</Position>
          <Speed>1</Speed>
          <Acceleration>1</Acceleration>
        </MoveAfterInit>
    </InitStrategy>
    <TestStrategy>
      <TestDescriptions>
        <TestDescription />
    </TestStrategy>
  </AxisStrategy>
  <AxisStrategy Id="GRIP:1/DRIVE:X">
    <InitStrategy>
      <InitDependencies>
        <InitDependency To="GRIP:1/DRIVE:Z">
          <InitDependencyDescription Type="position" Value="67" />
        </InitDependency>
      </InitDependencies>
        <MoveAfterInit>
          <Position>200</Position>
          <Speed>1</Speed>
          <Acceleration>1</Acceleration>
        </MoveAfterInit>
    </InitStrategy>
    <TestStrategy>
      <TestDescriptions />
    </TestStrategy>
  </AxisStrategy>
 </AxisStrategies>
</Strategy>
```

Example 9: Example 1 GRIP Z Long and Short Distance

The first `<TestDescription>` defines two dependencies to X and Y. The second `<TestDescription>` does not need this dependencies as the test can be carried out everywhere on the worktable. Nevertheless, even if the X and Y axes are not tested in this strategy they must have an entry as an own axis containing a `<InitStrategy>` for successful initialization.

2.3.2 GRIP Z Random Test

The second example uses the range_max_xxx and range_min_xxx constructs of some available types.

```
<?xml version="1.0" encoding="utf-8"?>
<Strategy xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <Name>Example1_Z_LongAndShortDistanceTest</Name>
  <AxisStrategies>
    <AxisStrategy Id="GRIP:1/DRIVE:X">
      <InitStrategy>
        <InitDependencies/>
          <MoveAfterInit>
            <Position>200</Position>
            <Speed>1</Speed>
            <Acceleration>1</Acceleration>
          </MoveAfterInit>
      </InitStrategy>
      <TestStrategy>
        <TestDescriptions>
          <TestDescription>
            <TestCases>
                <TestCase Id ="RandomSpeed">
                  <TestProperties>
                    <TestProperty Type="distance" Value="200" />
                    <TestProperty Type="returnToStart" Value="1" />
                    <TestProperty Type="range_min_speed" Value="0.3" />
                    <TestProperty Type="range_max_speed" Value="1" />
                    <TestProperty Type="acceleration" Value="1" />
                    <TestProperty Type="deceleration" Value="1" />
                  </TestProperties>
                </TestCase>
                <TestCase Id ="RandomAccelerationAndDeceleration">
                  <TestProperties>
                    <TestProperty Type="distance" Value="200" />
                    <TestProperty Type="returnToStart" Value="1" />
                    <TestProperty Type="range_min_acceleration" Value="0.3"/>
                    <TestProperty Type="range_max_acceleration" Value="1" />
                    <TestProperty Type="speed" Value="1" />
                  </TestProperties>
                </TestCase>
```

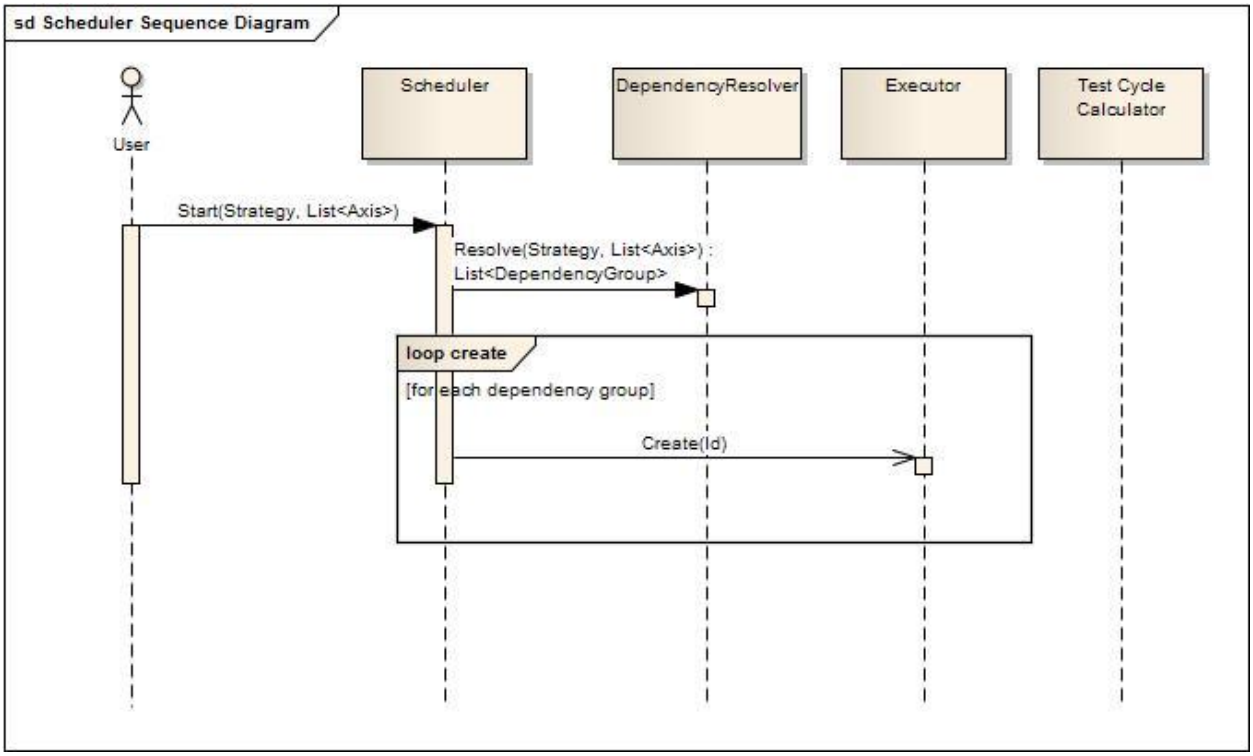
```
            <TestCase Id ="RandomDistances">
              <TestProperties>
                <TestProperty Type="returnToStart" Value="1" />
                <TestProperty Type="range_min_distance" Value="100" />
                <TestProperty Type="range_max_distance" Value="1000" />
                <TestProperty Type="acceleration" Value="1.0" />
                <TestProperty Type="deceleration" Value="1.0" />
                <TestProperty Type="speed" Value="1" />
              </TestProperties>
            </TestCase>
          </TestCases>
        </TestDescription>
      </TestDescriptions>
    </TestStrategy>
  </AxisStrategy>
 </AxisStrategies>
</Strategy>
```

Example 10: Example 2 GRIP X Random Test

In this example three test cases are defined. The first one selects random maximum speeds between 30% and 100% of the maximum defined speed in the instrument configuration. The second test case randomizes the acceleration. As the deceleration tag is not set, the same value will be taken for deceleration as for acceleration. In the last of the three test cases the distance is taken between 100 and 1000 units.

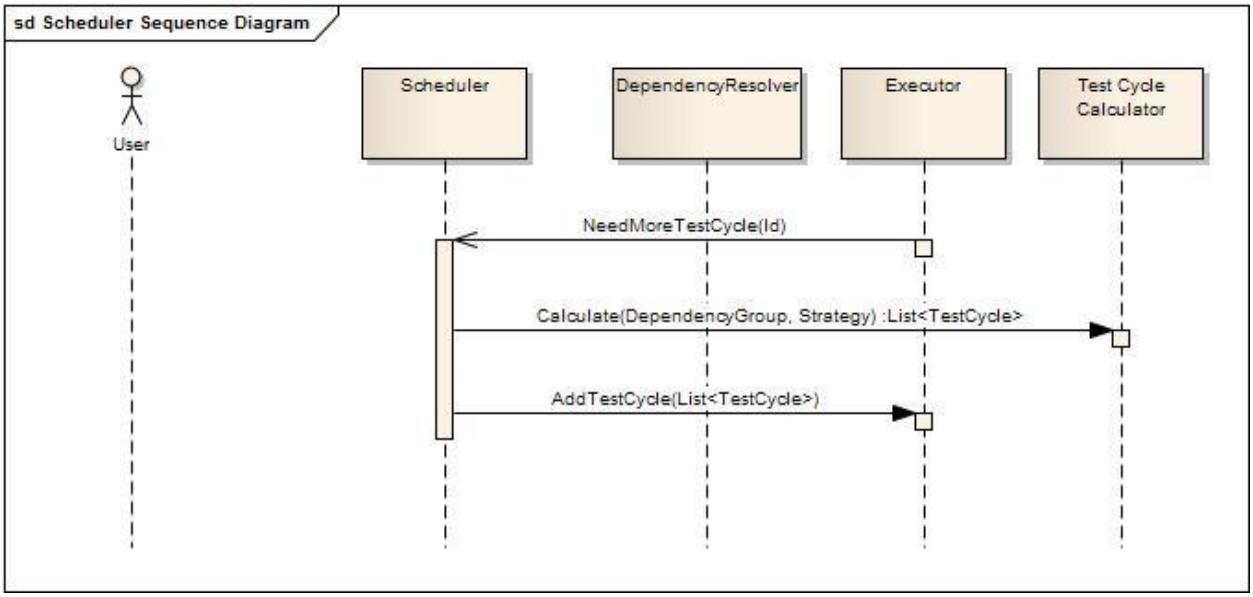
2.4 Scheduler Sequence Diagram

2.4.1 Start Sequence



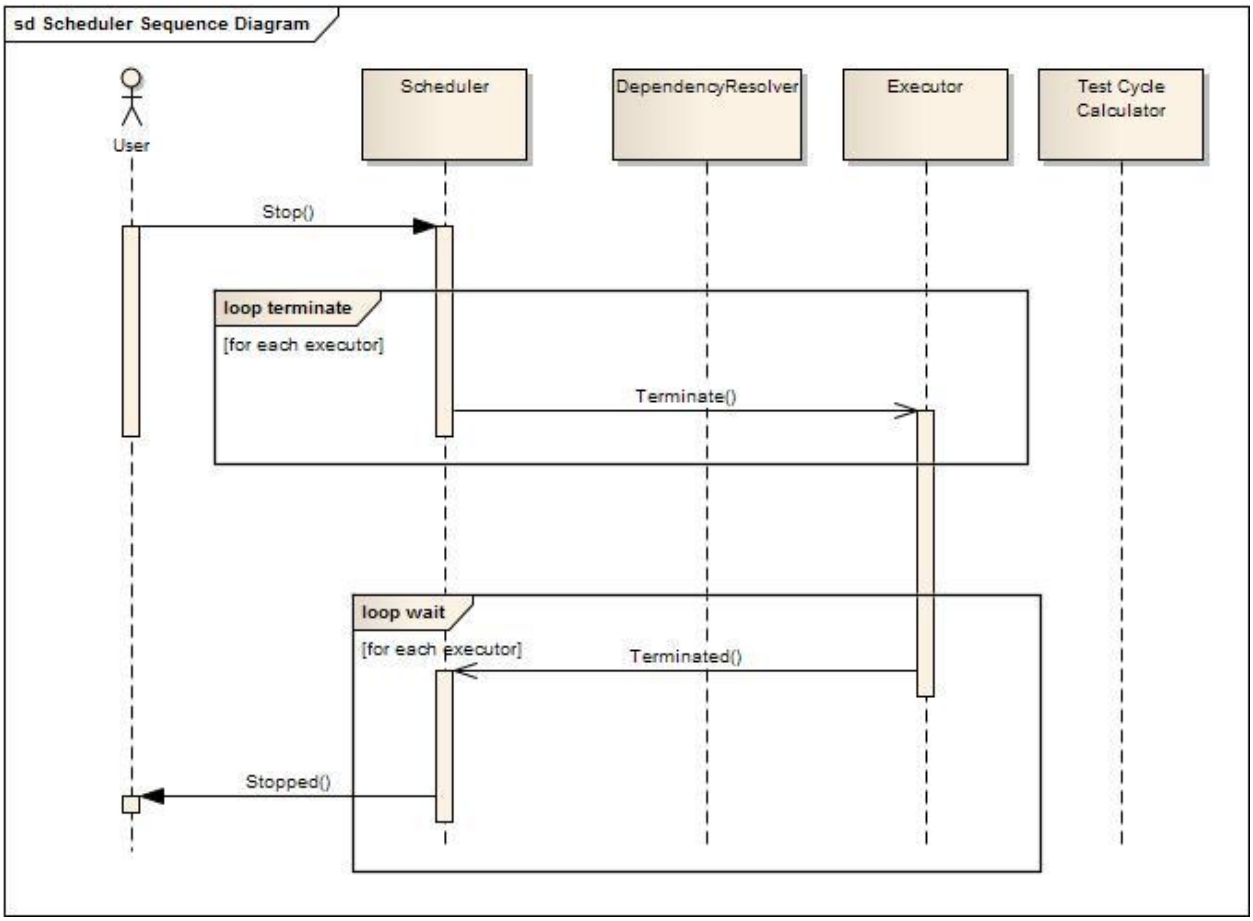
If the user starts a test run the scheduler will be started. The scheduler can use the dependency resolver to get all dependency groups. The dependency groups are determining according to the axes the user has selected and the current selected strategy. For each dependency group one executor will be created by the scheduler. The executors live in their own thread asynchronous to the scheduler.

2.4.2 Calculating Sequence



The executer will occasionally invoke an event that he needs more test cycle. The scheduler will then use the test cycle calculator to generate more test cycles. The test cycles are calculated on the basis of the selected strategy and the dependency group which ask for more test cycles. The scheduler adds the new calculated test cycles to a thread safe queue inside of the demanding executor.

2.4.3 Stop Sequence



If the user wants to stop the current running test run the scheduler will receive a stop command. The scheduler sends an asynchronous terminating command to each executor. The executors individually end their current test cycle and then finally terminate. The scheduler on the other hand waits till all executors have terminated and then signals the user that the test run has stopped.

3 Traceability Matrix

3.1 Downward Traceability

The traceability is handled inside the global traceability matrix file. (See Ref. [2])

4 Appendix

4.1 Table of Examples

Software Integration Test Case Plan-Report 1

Project-Name: **Qualification Tool**

Project Number: **-**

Subject: **-**

☐ Class/Unit Test Case

☒ Integration Test Case

☐ passed ☒ failed

Object(s) under Test: Qualification Tool V0.1

	Author	Reviewer	Approver
Name	Andreas Zollinger	Luc Bläser	Joas Leemann
Function	Software	Supervisor HSR	Project Leader
Date / Visa			

Table of Contents

1 Introduction

1.1 Purpose

The Test Plan and Test Report document shall offer the possibility to formally test a specific version of the Qualification Tool. Thus single test cases are defined in this document which the tester can carry out. The located bugs and errors can be feed back into the developing and bug fixing process of the tool.

1.2 Scope

This Test Plan and Test Report document scopes the whole Qualification Tool with all its functionality.
This Document is written in the Validation & Testing Phase.

1.3 Definitions, Acronyms and Abbreviations

Definitions, acronyms and abbreviations can be found in the global table (see Ref. [1])

1.4 References

Ref #	*Description*
Ref. [1]	Definition, Acronyms and Abbreviations for Qualification Tool, 90_DefinitionAcronymsAbbreviations.pdf, V1.0
Ref. [2]	Traceability Matrix for Qualification Tool, 91_TraceabilityMatrix.pdf, V1.0

1.5 Document Change History

Date	*Version*	*Change*	*Author*
2012-05-28	0.1	Initial Version	AnZo

1.6 Assumptions

The tester has to organize a working instrument with a working instrument configuration file compatible with the version of the qualification tool under testing. The tester must know how the instrument works. The tester musts know the different developer tool components of the Tecan Base SDK and how they work.

2 General Rules and Acceptance Criteria

- The test has to be done step by step.
- The tester has to log in writing the actual result of each test.
- During the execution of the test produced printouts must be attached to the paper version of this TPTR.
- The software is acceptable (the entire test is passed) only if all tests are completed and if all actual results correspond to the expected results (“Results as expected / Passed” has to be marked ☒).
- The software is unacceptable (the entire test is failed) if at least one of the actual test results does not correspond to the expected result (“Results not as expected / Failed” has to be marked ☒).
- Failures or errors have to be noted in the bug report rows.
- The tester has to sign and date after each test procedure.
- It is important that the tester has knowledge of the instrument, its operating software as well as the software being tested.

2.1 Additional Acceptance Criteria for Units with Safety Class C

There are currently no units with safety class C.

3 Test Equipment and Resources

3.1 Personnel

One tester is needed to carry out all de defined tests below.

3.2 Test Equipment

One working instrument.
One working and compatible instrument configuration file.
Tecan Base SDK v0.9.3.x

3.2.2 Personal Computer

3.2.2.1 Hardware

- USB interface (enough for all external devices and one instrument)
- Keyboard
- Mouse

3.2.2.2 Software

- Microsoft Windows XP 32 bit or Microsoft Windows 7 64 bit
- Tecan Base SDK

4 Test Case Plan and Test Case Report

4.1 Preparation

- Connect the instrument to the computer.
- Check with the Tecan Base SDK developer tools if all axes could be loaded.
- Test if each axis works fine with help of the Tecan Base SDK developer tools.

4.2 Standard Test Conditions

Standard test conditions are, if nothing else is mentioned, as follows:

- Computer and Instrument are powered on.
- No Tecan Base SDK component is running. (Expect of the LogViewer)

4.3 Test Procedures and Protocols

This section describes the test procedures and records the results of the tests. A test procedure is composed of one or more functions to be tested. Each function is structured into six items:

- Functionality
- Test conditions, starting point
- Stimuli, commands
- Expected results
- Actual results
- Passed / Failed
- Bug report

The actual result of each test must be logged and the boxes must be marked (☒) in writing by the tester. If a failure or an error appears during the test, the row “Bug report” must be filled out by the tester.

4.3.1 Start / Stop Test Run

4.3.1.1 Start Test Run / Stop Test Run		
Functionality	Start Test Run, Stop Test Run	
Test conditions, starting point	See 4.2.	
Stimuli, commands	a) start Qualification Tool b) start Test Run (wait till initialization of each axis is done) c) stop Test Run	
Expected results	a) tool starts up without error, all axes are checked for execution, one strategy is pre-selected b) all axes start to initialize, afterwards starting with test phase c) each axis group finish the current running test and stops afterwards	
Actual results	a) Start w/o any errors, all axes checked ,no strategy selected ! b) all initialize , than all start testrun. c) all groups ends current testcycle , than stop.	
Passed / Failed	Results as expected/Passed ☐	Results not as expected/Failed ☒
Bug report	no strategy pre-selected after start.	

4.3.1.2 Select Axes & Strategy		
Functionality	Start Test Run, Select Axes, Select Strategy	
Test conditions, starting point	See 4.2.	
Stimuli, commands	a) start Qualification Tool b) select/deselect some axes c) select another strategy d) start test run (wait till initialization of each axis is done) e) stop test run	
Expected results	a) tool starts up without error, all axes pre-selected, one strategy pre-selected b) deselected axes recognizable as such c) other strategy is seen as loaded d) selected axes start to initialize (in some case, depending on the selected strategy and selected axes also deselected axes initialize) e) each axis group finish their current running test and then stop	
Actual results	a) as in test 4.3.1.1 (bug ~~from~~ Strategy not selected still occurse) b) Axes are not checked anymore after deselecting. c) Other name seen in dropdown after selecting other strategy. d) all selected axes initialize & then start with test run. e) as in test 4.3.1.1	
Passed / Failed	Results as expected/Passed ☒	Results not as expected/Failed ☐
Bug report	/	

4.3.1.3 Restart Test Run		
Functionality	Start Test Run	
Test conditions, starting point	See 4.2. Additional Test Case 4.3.1.1 Start Test Run / Stop Test Run	
Stimuli, commands	a) restart test run b) stop test run	
Expected results	a) no initialization, testing should continue immediately b) each axis group finish their current running test and then stop	
Actual results	a) Starts w/o initialization. b) as in test 4.3.1.1	
Passed / Failed	Results as expected/Passed ☒	Results not as expected/Failed ☐
Bug report	/	

4.3.1.4 Load Test Run		
Functionality	Start Test Run, Load Test Run	
Test conditions, starting point	See 4.2. Additional Test Case 4.3.1.2 Select Axes & Strategy was done at least one time. Afterwards tool has to be quit.	
Stimuli, commands	a) start qualification tool b) load old test run c) start test run d) stop test run	
Expected results	a) tool starts up without error, all axes are checked for execution, one strategy is pre-selected b) after loading procedure only the selected axes of the loaded test run are selected, the corresponding strategy is loaded c) selected axes start to initialize (in some case, depending on the selected strategy and selected axes also deselected axes initialize) d) each axis group finish the current running test and stops afterwards	
Actual results	a) as in test 4.3.1.1 (still with bug) b) Axes selected as expected. c) As expected, with initialization. d) as in test 4.3.1.1	
Passed / Failed	Results as expected/Passed ☒	Results not as expected/Failed ☐
Bug report	/	

Name:	Signature:	Date:
Andreas Zollinger		2012-05-21

4.3.2 Reporting

4.3.2.1	Read Report on the fly	
Functionality	Read Report, Update Report	
Test conditions, starting point	See 4.2. Additional: Qualification tool is already running and a test run started	
Stimuli, commands	a) click on an axis reporting to see the reporting of this axis	
Expected results	a) axis report is shown, data on screen is updated on the fly with every test move the axis does	
Actual results	a) Report displayed, but no update!	
Passed / Failed	Results as expected/Passed ☐	Results not as expected/Failed ☒
Bug report	No update on view.	

4.3.2.2	Read Older Report	
Functionality	Read Report	
Test conditions, starting point	See 4.2. Additional: Qualification tool is already running and an old test run is loaded	
Stimuli, commands	a) click on an axis reporting to see the reporting of this axis	
Expected results	a) axis report is shown	
Actual results	a) Report displayed	
Passed / Failed	Results as expected/Passed ☒	Results not as expected/Failed ☐
Bug report	/	

4.3.2.3 Export Report		
Functionality	Read Report, Export Report	
Test conditions, starting point	See 4.2. Additional: Qualification tool is already running and an old test run is loaded	
Stimuli, commands	a) click on an axis reporting to see the reporting of this axis b) export the report	
Expected results	a) axis report is shown b) the report is exported into the selected file format to the selected location	
Actual results	a) Report is displayed b) no export option available !!	
Passed / Failed	Results as expected/Passed ☐	Results not as expected/Failed ☒
Bug report	There is no export option available on the entire screen!	

Name:	Signature:	Date:
Andreas Zollinger	A. Zollinger	2012-05-21

5 Traceability Matrix

The traceability is handled inside the global traceability matrix file. (See Ref. [2])

Software Integration Test Case Plan-Report 2

Project-Name: **Qualification Tool**

Project Number: **-**

Subject: **-**

☐ Class/Unit Test Case

☒ Integration Test Case

☐ passed ☒ failed

Object(s) under Test: Qualification Tool V0.1

	Author	Reviewer	Approver
Name	Andreas Zollinger	Luc Bläser	Joas Leemann
Function	Software	Supervisor HSR	Project Leader
Date / Visa			

Table of Contents

1 Introduction

1.1 Purpose

The Test Plan and Test Report document shall offer the possibility to formally test a specific version of the Qualification Tool. Thus single test cases are defined in this document which the tester can carry out. The located bugs and errors can be feed back into the developing and bug fixing process of the tool.

1.2 Scope

This Test Plan and Test Report document scopes the whole Qualification Tool with all its functionality.
This Document is written in the Validation & Testing Phase.

1.3 Definitions, Acronyms and Abbreviations

Definitions, acronyms and abbreviations can be found in the global table (see Ref. [1])

1.4 References

Ref #	*Description*
Ref. [1]	Definition, Acronyms and Abbreviations for Qualification Tool, 90_DefinitionAcronymsAbbreviations.pdf, V1.0
Ref. [2]	Traceability Matrix for Qualification Tool, 91_TraceabilityMatrix.pdf, V1.0

1.5 Document Change History

Date	*Version*	*Change*	*Author*
2012-05-28	0.1	Initial Version	AnZo

1.6 Assumptions

The tester has to organize a working instrument with a working instrument configuration file compatible with the version of the qualification tool under testing. The tester must know how the instrument works. The tester musts know the different developer tool components of the Tecan Base SDK and how they work.

2 General Rules and Acceptance Criteria

- The test has to be done step by step.
- The tester has to log in writing the actual result of each test.
- During the execution of the test produced printouts must be attached to the paper version of this TPTR.
- The software is acceptable (the entire test is passed) only if all tests are completed and if all actual results correspond to the expected results (“Results as expected / Passed” has to be marked ☒).
- The software is unacceptable (the entire test is failed) if at least one of the actual test results does not correspond to the expected result (“Results not as expected / Failed” has to be marked ☒).
- Failures or errors have to be noted in the bug report rows.
- The tester has to sign and date after each test procedure.
- It is important that the tester has knowledge of the instrument, its operating software as well as the software being tested.

2.1 Additional Acceptance Criteria for Units with Safety Class C

There are currently no units with safety class C.

3 Test Equipment and Resources

3.1 Personnel

One tester is needed to carry out all de defined tests below.

3.2 Test Equipment

One working instrument.
One working and compatible instrument configuration file.
Tecan Base SDK v0.9.3.x

3.2.2 Personal Computer

3.2.2.1 Hardware

- USB interface (enough for all external devices and one instrument)
- Keyboard
- Mouse

3.2.2.2 Software

- Microsoft Windows XP 32 bit or Microsoft Windows 7 64 bit
- Tecan Base SDK

4 Test Case Plan and Test Case Report

4.1 Preparation

- Connect the instrument to the computer.
- Check with the Tecan Base SDK developer tools if all axes could be loaded.
- Test if each axis works fine with help of the Tecan Base SDK developer tools.

4.2 Standard Test Conditions

Standard test conditions are, if nothing else is mentioned, as follows:

- Computer and Instrument are powered on.
- No Tecan Base SDK component is running. (Expect of the LogViewer)

4.3 Test Procedures and Protocols

This section describes the test procedures and records the results of the tests. A test procedure is composed of one or more functions to be tested. Each function is structured into six items:

- Functionality
- Test conditions, starting point
- Stimuli, commands
- Expected results
- Actual results
- Passed / Failed
- Bug report

The actual result of each test must be logged and the boxes must be marked (☒) in writing by the tester. If a failure or an error appears during the test, the row “Bug report” must be filled out by the tester.

4.3.1 Start / Stop Test Run

4.3.1.1 Start Test Run / Stop Test Run		
Functionality	Start Test Run, Stop Test Run	
Test conditions, starting point	See 4.2.	
Stimuli, commands	a) start Qualification Tool b) start Test Run (wait till initialization of each axis is done) c) stop Test Run	
Expected results	a) tool starts up without error, all axes are checked for execution, one strategy is pre-selected b) all axes start to initialize, afterwards starting with test phase c) each axis group finish the current running test and stops afterwards	
Actual results	a) No error on start up, all axes checked, strategy pre selected. b) All axes initialize, then starting with tests. c) All groups ending after finishing curren test.	
Passed / Failed	Results as expected/Passed ☒	Results not as expected/Failed ☐
Bug report	/	

4.3.1.2 Select Axes & Strategy		
Functionality	Start Test Run, Select Axes, Select Strategy	
Test conditions, starting point	See 4.2.	
Stimuli, commands	a) start Qualification Tool b) select/deselect some axes c) select another strategy d) start test run (wait till initialization of each axis is done) e) stop test run	
Expected results	a) tool starts up without error, all axes pre-selected, one strategy pre-selected b) deselected axes recognizable as such c) other strategy is seen as loaded d) selected axes start to initialize (in some case, depending on the selected strategy and selected axes also deselected axes initialize) e) each axis group finish their current running test and then stop	
Actual results	a) as in test 4.3.1.1 b) axes clearly visible as checked/unchecked. c) other strategy loaded. d) selected axes and additional axes (dependencies) start to initialize. e) as in test 4.3.1.1	
Passed / Failed	Results as expected/Passed ☒	Results not as expected/Failed ☐
Bug report	/	

4.3.1.3 Restart Test Run		
Functionality	Start Test Run	
Test conditions, starting point	See 4.2. Additional Test Case 4.3.1.1 Start Test Run / Stop Test Run	
Stimuli, commands	a) restart test run b) stop test run	
Expected results	a) no initialization, testing should continue immediately b) each axis group finish their current running test and then stop	
Actual results	a) only the selected axes (w/o the ones which have depend.) starting with test runs. b) as in test 4.3.1.1	
Passed / Failed	Results as expected/Passed ☒	Results not as expected/Failed ☐
Bug report	/	

4.3.1.4 Load Test Run	
Functionality	Start Test Run, Load Test Run
Test conditions, starting point	See 4.2. Additional Test Case 4.3.1.2 Select Axes & Strategy was done at least one time. Afterwards tool has to be quit.
Stimuli, commands	a) start qualification tool b) load old test run c) start test run d) stop test run
Expected results	a) tool starts up without error, all axes are checked for execution, one strategy is pre-selected b) after loading procedure only the selected axes of the loaded test run are selected, the corresponding strategy is loaded c) selected axes start to initialize (in some case, depending on the selected strategy and selected axes also deselected axes initialize) d) each axis group finish the current running test and stops afterwards
Actual results	a) as in test 4.3.1.1 b) ALL axes are selected → more than the old test run had selected. Had to deselect the ones not needed. c) Selected axes start to initialize and than start with testruns. d) as in test 4.3.1.1
Passed / Failed	Results as expected/Passed ☒ ☐ Results not as expected/Failed ☒
Bug report	After Loading all ~~test~~ axis are selected.

Name:	Signature:	Date:
Andreas Zollinger		2012-06-01

4.3.2 Reporting

4.3.2.1 Read Report on the fly		
Functionality	Read Report, Update Report	
Test conditions, starting point	See 4.2. Additional: Qualification tool is already running and a test run started	
Stimuli, commands	a) click on an axis reporting to see the reporting of this axis	
Expected results	a) axis report is shown, data on screen is updated on the fly with every test move the axis does	
Actual results	a) Report displayed, with updates	
Passed / Failed	Results as expected/Passed ☒	Results not as expected/Failed ☐
Bug report	/	

4.3.2.2 Read Older Report		
Functionality	Read Report	
Test conditions, starting point	See 4.2. Additional: Qualification tool is already running and an old test run is loaded	
Stimuli, commands	a) click on an axis reporting to see the reporting of this axis	
Expected results	a) axis report is shown	
Actual results	a) Report displayed	
Passed / Failed	Results as expected/Passed ☒	Results not as expected/Failed ☐
Bug report	/	

4.3.2.3 Export Report		
Functionality	Read Report, Export Report	
Test conditions, starting point	See 4.2. Additional: Qualification tool is already running and an old test run is loaded	
Stimuli, commands	a) click on an axis reporting to see the reporting of this axis b) export the report	
Expected results	a) axis report is shown b) the report is exported into the selected file format to the selected location	
Actual results	a) Report displayed b) Only the whole Report could be exported, but this worked.	
Passed / Failed	Results as expected/Passed ☒	Results not as expected/Failed ☐
Bug report	/	

Name:	Signature:	Date:
Andreas Zollinger	A. Zollinger	2012-06-01

5 Traceability Matrix

The traceability is handled inside the global traceability matrix file. (See Ref. [2])

TECAN.

Part III - Appendix

---

1 Project Conclusion

The Project Conclusion document gives a final conclusion about the bachelor thesis. Time management, achieved goals, an outlook, lessons learned and a personal statement are part of this document.

2 Traceability Matrix

This document contains the traces from the first requirements to the test cases.

3 Definitions, Acronyms and Abbreviations

4 Poster

Project Conclusion

Project-Name: **Qualification Tool**

Project Number: **-**

Subject: **-**

	Author	Reviewer	Approver
Name	Andreas Zollinger	Luc Bläser	Joas Leemann
Function	Software	Supervisor HSR	Project Leader
Date / Visa			

Table of Contents

1 Introduction

1.1 Scope

This document is written at the end of the project to summarize the goals achieved and give an outlook what has to be done in the near future.

1.2 Definitions, Acronyms and Abbreviations

Definitions, acronyms and abbreviations can be found in the global table (see Ref. [1])

1.3 References

Ref #	*Description*
Ref. [1]	Definition, Acronyms and Abbreviations for Qualification Tool, 90_DefinitionAcronymsAbbreviations.pdf, V1.0
Ref. [2]	Software Specification for Qualification Tool, 03_SWSpecification.pdf, V1.0

1.4 Document Change History

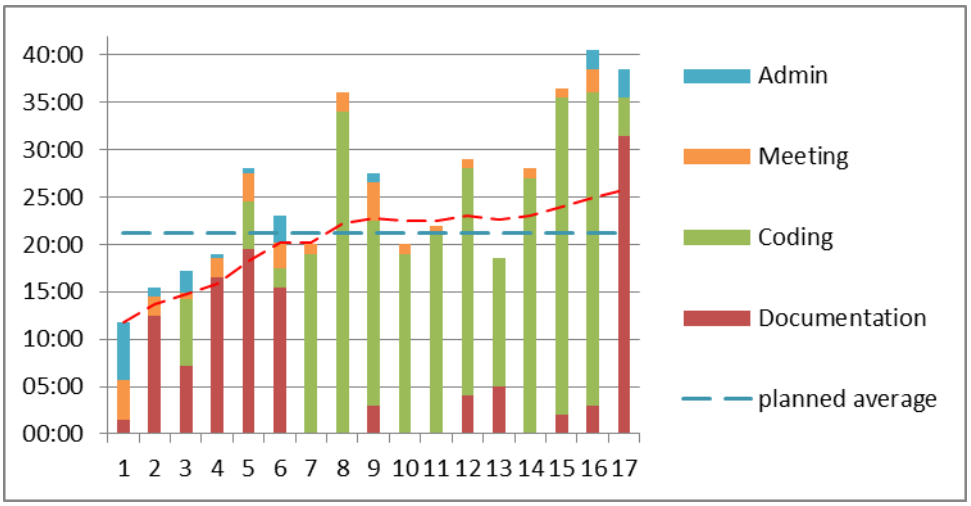
Date	*Version*	*Change*	*Author*
2012-06-2012	1.0	Initial Version	AnZo

2 Project Time Management

The HSR bachelor thesis module has a value of 12 ECTS points.
One ECTS equals a time effort around 30 hours. Thus, this module should have the outline of about 360 hours.
The bachelor thesis project is laid out over 17 weeks, resulting in a time effort around 21 hours per week.

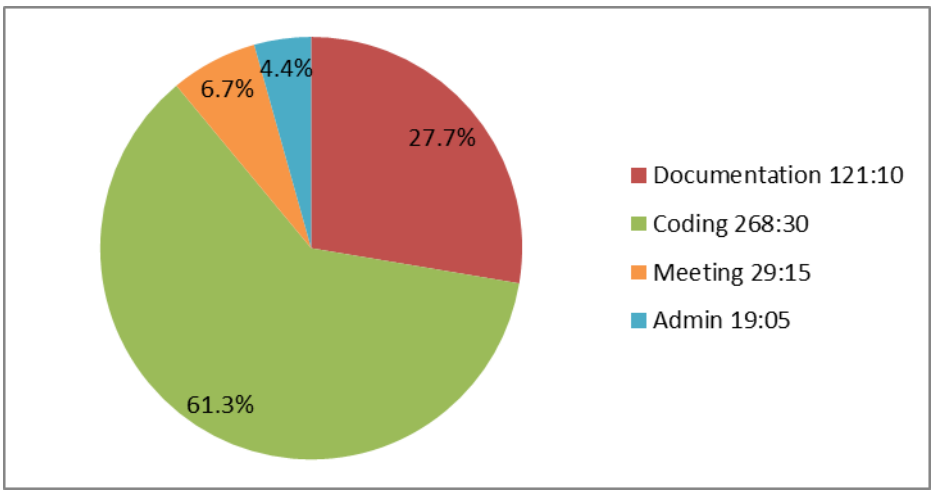
	Total Hours	*%*	*Hours / Week*
Estimated Time	360	100 %	~ 21
Time Worked	~ 440	~ 120 %	~ 25
Additional Work	~ 60	~ 20 %	~ 4

2.1 Time per Week



At the start of the project the required time of about 21 hours per week weren't used up. This was done fully aware that at the end of the project this additional time will have to be used. After the implementation phase has started, the average of hours per week overcame the 21 hours per week line really fast. At the end of the project where the delivery came closer (last 4 weeks) the overwork time fired the average hours per week up to its maximum.

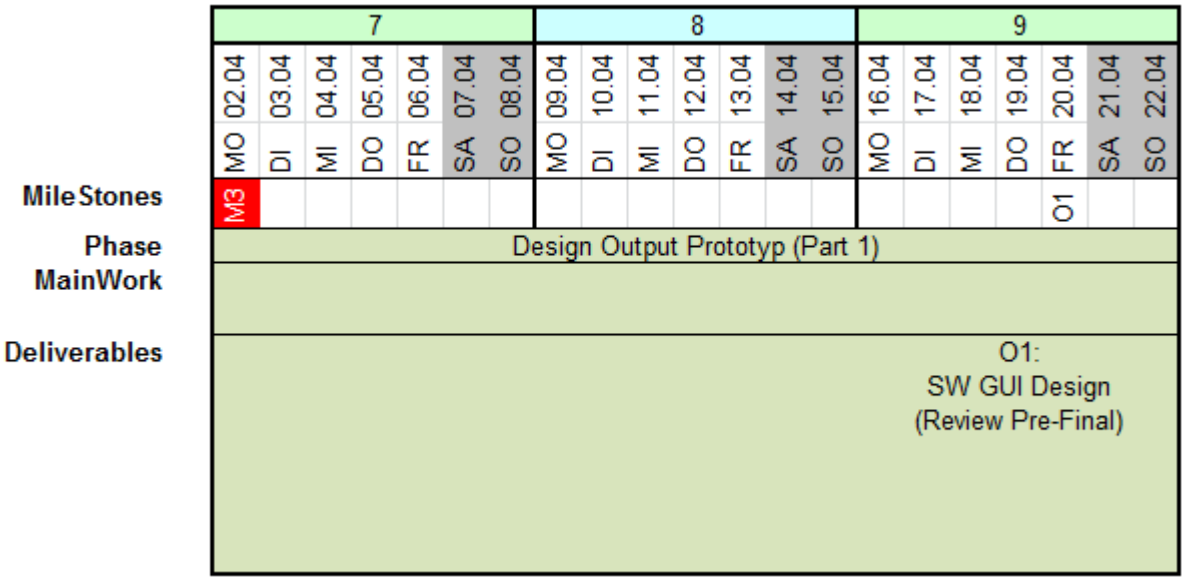
2.2 Time per Topic

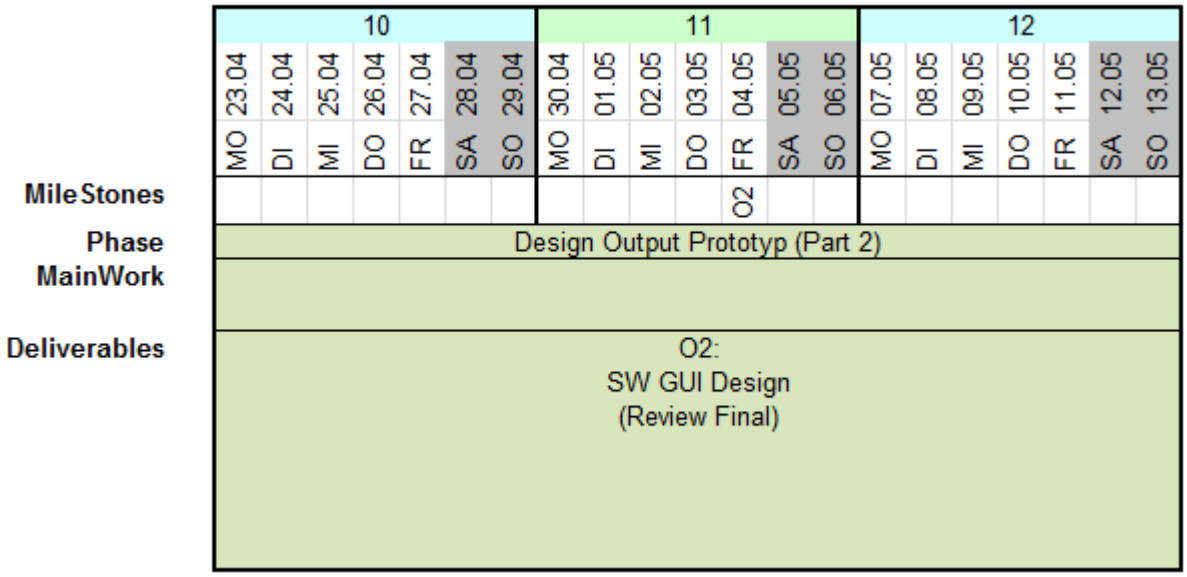


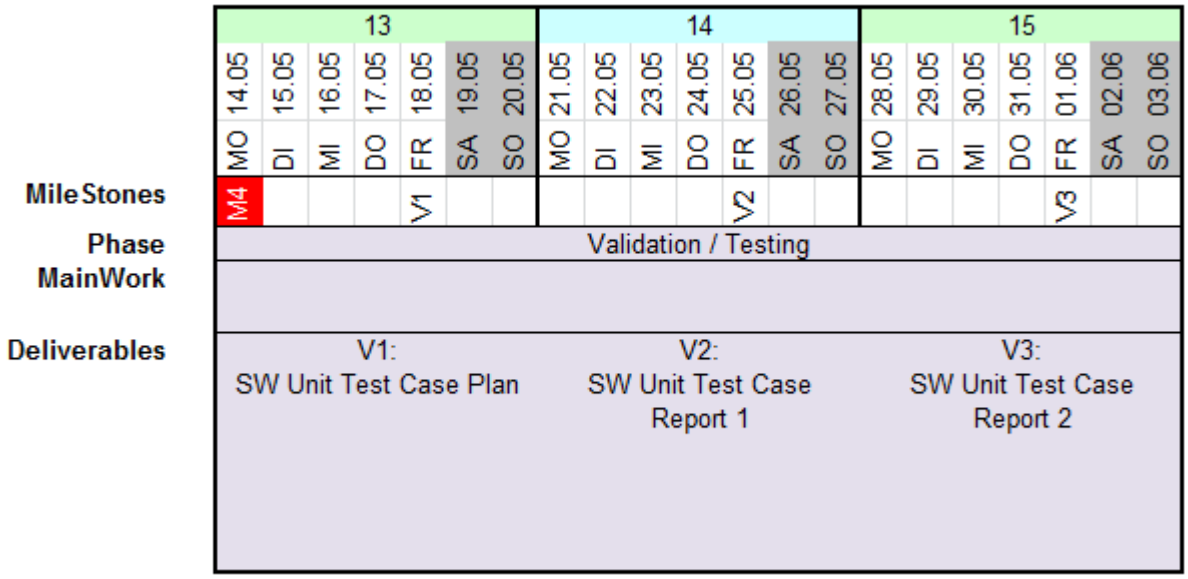
2.3 Project Time Line

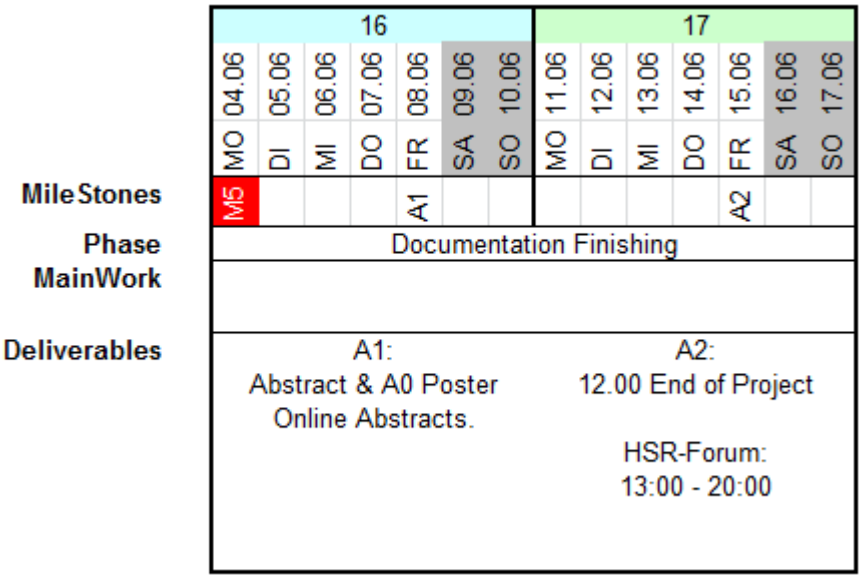
	1							2							3						
	MO 20.02	DI 21.02	MI 22.02	DO 23.02	FR 24.02	SA 25.02	SO 26.02	MO 27.02	DI 28.02	MI 29.02	DO 01.03	FR 02.03	SA 03.03	SO 04.03	MO 05.03	DI 06.03	MI 07.03	DO 08.03	FR 09.03	SA 10.03	SO 11.03
MileStones								M1				C1							C2		
Phase	KickOff							Concept Phase													
MainWork								Project Plan							Requirements						
Deliverables								C1: PDP (Project Development Plan)							C2: SW PRD (Product Requirements)						

	4							5							6						
	MO 12.03	DI 13.03	MI 14.03	DO 15.03	FR 16.03	SA 17.03	SO 18.03	MO 19.03	DI 20.03	MI 21.03	DO 22.03	FR 23.03	SA 24.03	SO 25.03	MO 26.03	DI 27.03	MI 28.03	DO 29.03	FR 30.03	SA 31.03	SO 01.04
MileStones	M2				I1							I2			I3				I4		
Phase	Design Input																				
MainWork	Specifications							Specifications, GUI & Architecture							GUI Architecture						
Deliverables	I1: SW Configuration Management Plan							I2: SWS (Specification) UCS (Use Case Specification)							I3: SW GUI Design (Paper Prototype) I4: SSD (Structure Design)						


7
8
9
MO 02.04
DI 03.04
MI 04.04
DO 05.04
FR 06.04
SA 07.04
SO 08.04
MO 09.04
DI 10.04
MI 11.04
DO 12.04
FR 13.04
SA 14.04
SO 15.04
MO 16.04
DI 17.04
MI 18.04
DO 19.04
FR 20.04
SA 21.04
SO 22.04
Mile Stones
M3
O1
Phase
MainWork
Design Output Prototyp (Part 1)
Deliverables
O1:
SW GUI Design
(Review Pre-Final)


10
11
12
MO 23.04
DI 24.04
MI 25.04
DO 26.04
FR 27.04
SA 28.04
SO 29.04
MO 30.04
DI 01.05
MI 02.05
DO 03.05
FR 04.05
SA 05.05
SO 06.05
MO 07.05
DI 08.05
MI 09.05
DO 10.05
FR 11.05
SA 12.05
SO 13.05
Mile Stones
O2
Phase
MainWork
Design Output Prototyp (Part 2)
Deliverables
O2:
SW GUI Design
(Review Final)


13
14
15
14.05 MO
15.05 DI
16.05 MI
17.05 DO
18.05 FR
19.05 SA
20.05 SO
21.05 MO
22.05 DI
23.05 MI
24.05 DO
25.05 FR
26.05 SA
27.05 SO
28.05 MO
29.05 DI
30.05 MI
31.05 DO
01.06 FR
02.06 SA
03.06 SO
Mile Stones
M4
V1
V2
V3
Phase
Validation / Testing
MainWork
Deliverables
V1:
SW Unit Test Case Plan
V2:
SW Unit Test Case
Report 1
V3:
SW Unit Test Case
Report 2


16
17
04.06 MO
05.06 DI
06.06 MI
07.06 DO
08.06 FR
09.06 SA
10.06 SO
11.06 MO
12.06 DI
13.06 MI
14.06 DO
15.06 FR
16.06 SA
17.06 SO
Mile Stones
M5
A1
A2
Phase
Documentation Finishing
MainWork
Deliverables
A1:
Abstract & A0 Poster
Online Abstracts.
A2:
12.00 End of Project
HSR-Forum:
13:00 - 20:00

3 Achieved Goals

3.1 Status Quo

At the end of the project the basic functionality is implemented. The tool is capable of initializing motors and carries out tests. For this an XML file is used which describes the dependencies and tests of a single axis.

The user can select one of the predefined strategies. He further can select which motors he wants to test. The tool will initialize fully automated all motors in the right order according to de defined dependencies inside the strategy file. After each motor is initialized the tool will start to execute the tests move. For this the tool first configures the firmware to monitor specific statistic data points and then executes a test move and immediately readouts the monitored data. The data itself will be collected and stored in an XML file.

The tool displays the measured data to the user. On a line chart the user can observe the change of the values over time. Additional information, for example the average or the maximal value, are also provided. The report view will be updated on the fly if a test run is currently running. The user can export the gathered data from one test run into a CSV file.

The tool as it is today can be used to carry out test runs and gather data. All priority one features could be implemented.

3.2 Missing Features

There are some important features missing. This section shall give a little overview over the task which were planned for this project but could not be reached. All of them were priority 2 or less. For additional features and what else is yet to come see the next chapter.

Error recovery is not implemented at all. There exist some standard procedures on what one can do if motors signal an error. In most of the cases an operator has to investigate the instrument in such a case. To map this knowledge of an operator into code is way more complex than the given time of this project would have allowed it. This fact became clear to me after I had done one or two simply recovery implementation tries. There is no single big error recovery procedure the tool can use to restore everything and handling each individual error case would be way too much time consuming. Therefore a good error recovery concept has to be done in the near future to enable automated error recovery.

One other big problem was the definition of criteria when the gathered data can state that the specification of a motor is satisfying. As stated by more than one member of the motion control team each individual motor in its application area has other specifications. These specifications can be defined beforehand, but maybe it is ok under some special condition that the specifications are not met. Other times the specifications are much too loose. What exactly shall be gained with an automated statistic comparison and how to present this to the user also has to be well defined first.

The third topic also has to do with the gathered data. With the current tool the data is taken and some simple calculations with them are done. For example the average will be calculated. But as a module who claims to do statistically calculation it should do a little more than just that. Things like filtering out outliers or calculating a standard deviation would be nice.

4 Outlook

This chapter shall give a rough overview what features are still missing or would be nice to have.

4.1 Performance Optimization

- Database
  Instead of storing the gathered data from the tool into an XML file, there would be an increase in performance if stored in a database. The database could be accessible for every tester and the test data could be saved in one place instead of hundreds of files.
- Threading
  As this project was my first one where I had to use a great amount of threads I'm well aware of that my implementation is surely not the best approach to implement threading based applications. Refactoring the threading part will increase the stability.

4.2 Additional Features

At this point only a small list of possible additional features is given. Another good source for features is the software specification document (see Ref.[2]).

- Additional Exporting Formats:
  - PDF / XPS
  - WebService
- E-Mail notification under special conditions.
- Additional type-value pairs in strategy file.
- Strategy-File Editor
- Error Recovery
- Test Data Specification Support
- Statistical Calculations

5 Lessons Learned

5.1 What Went Well

- Right from the beginning I did know that I wanted to use Ninject for injection. Sad but true, it is the first project where I used injection. And it worked very well. I am glad that I used it. The advantages are just too good to ignore them for further projects.
- As in my term project I used again the WPF framework Caliburn Micro. This time I could use some more features of it. Most positive point was that Caliburn Micro and Ninject worked hand in hand very well with each other.
- I could learn much about threading in this project. Until today I have always been afraid of threading, too much can be done the wrong way. Now I'm looking forward to work again on a project with many threads to learn even more.
- The communication with the motion control team and the Tecan Base SDK team went again really well. They were all eager to help and support me with every question I had.
- The whole project was challenging but always fun.

5.2 What Went Badly

- In the term project I had pointed out in my lessons learned that I had too little time planned for documentation. That is why I had planned more time for the documentation. At the start of the project I did took my time to do the documentation. But after the implementation phase had started I lost focus of the documents. At the end of the project I had too many documents still open and so a quite stressful time was unavoidable.
- Workplace situation was not optimal. During developing and for all my tests I had to use an instrument with many motors. At the place I organized for my bachelor thesis there was no such instrument. At my desk I have at Tecan for my normal work there is one, so I ended up using this one. But at my desk I had too much distraction from co-workers with problems related to my normal project at Tecan.

6 Personal Statement

To write my bachelor thesis at the place where I'm employed is an advantage. I'm very grateful for this opportunity. I could finish my term project in the last semester also at Tecan Schweiz AG. That's why I could guess quite well what tasks to expect in general. At the beginning of the term project I had no clue about the SOP from Tecan and was overrun by the mass of documentation I could have done. This time the SOP couldn't scare me anymore. I did know which the important documents were and where I had to set my priorities. The biggest advantage compared to the term project was the time. I had about 5 weeks more and so could also plan more time for the documentation, one of the negative points I had found for myself in the term project.

Nevertheless the bachelor project had a quite stressful start. At the beginning of the project the topic was not fixed. The first week was used entirely for finding an appropriate topic. Luckily, for me at least, the motion control team at Tecan Schweiz AG had a big task running at this time. They had to verify that the assembled axes worked well with the motion control parameters the team had elicited over the past few months. The process to gather the data was sophisticated and troublesome. The wish for an automated tool was really high at that time. As one may say, it was almost obvious that developing this tool was the logical conclusion. After the topic was fixed I could start successfully into the bachelor thesis.

As already mentioned, I did know what documents I had to generate in the first weeks. This did take away a lot of pressure and I could fully concentrate on the content. I developed the strategy concept and was quite happy with it. The idea of the strategy was accepted by my superiors, which also boosted my motivation. Documentation went really well, but during implementation I had to overcome some obstacles. There were some topics that I have never touched before, but I wanted to use them. First of all was the threading problematic I had to handle, because there was no way around it. As the Tecan Base SDK uses a lot of blocking waits I had to use a multi-threading approach. I do know the things I learned during the lessons at school and I had done my exercises, but nothing of this stuff had productive aspects. Secondly, I wanted to use injection, as I learned about Ninject one month prior to the bachelor thesis and was excited about the concept. At the end of the project, I am grateful that I did use the injection framework as it did simplify so many things, unit testing for example.

During the term project I had some problems with the Tecan Base SDK. But this lesson I have learned. If I had any problems during bachelor thesis I immediately asked the responsible person and did not wait a week or two just to learn that I used a wrong class or other stupid mistakes. Also, the experience I gained during the term project boosted the developing speed of this project significantly. I had almost no problems with the framework.

Summarizing, I had a good time during the bachelor thesis. As with every other project I had done it was stressful, but that is something I'm almost used to.

Traceability Matrix

Project-Name: **Qualification Tool**

Project Number: **-**

Subject: **-**

Table of Contents

1 Introduction

1.1 References

Ref #	*Description*
Ref. [1]	Definition, Acronyms and Abbreviations for Qualification Tool, 90_DefinitionAcronymsAbbreviations.pdf, V1.0
Ref. [2]	Product Requirement Document for Qualification Tool, 02_ProductRequirementDocument.pdf, V1.0
Ref. [3]	Use Case Specification for Qualification Tool, 04_UseCaseSpecifcation.pdf, V1.1
Ref. [4]	Software Specification for Qualification Tool, 03_SWSpecification.pdf, V1.0
Ref. [5]	Software Unit Test Case Plan and Report for Qualification Tool, 50_SWUnitTestCasePlanReport.pdf, V1.0

1.2 Document Change History

Date	*Version*	*Change*	*Author*
2012-06-15	1.0	First Version	AnZo

2 Tracing from PRD to UCS

Traces from the requirements (see Ref. [2]) to the use case specifications (se Ref. [3]).

PRD	*Requirement*	*UCS*
1	Use Tecan Base SDK	1,2,5
2	One Click Application	1
3	Select Axes to Test	2
4	Scheduler	1,2,3,8
5	Define Strategy	8
6	Select Strategy	3
7	Continuous Reporting	6,7
8	Error Recovery	1,5
9	Time Scheduler	1,4,5

3 Tracing from UCS to SWS

Traces from the use case specifications (see Ref. [3]) to the software specifications (se Ref. [4]).

UCS	*Use Case*	*SWS*
1	Start Test Run	1,3,4,8,9,10,11,12,13,19,22,23,24,25
2	Select Axes	6
3	Select Strategy	7,16,17
4	Load Test Run	5
5	Stop Test Run	2
6	Read Report	14,15,29,30
7	Export Report	18,27, 28
8	Define Strategy	20,21,26
	PRD 1	31

Software specification 31 cannot be traced upwards to a use case, the specification 31 “Tecan Base SDK” is to global and would better match directly with PRD 1.

4 Tracing from SWS to TPTR

Traces from the software specifications (see Ref. [4]) to the test plan and test report tests (se Ref. [5]).

SWS	*Software Specification*	*TPTR*
1	Start Test Run	4.3.1.1, 4.3.1.2
2	Stop Test Run	4.3.1.1, 4.3.1.2
3	Restart Test Run	4.3.1.3
4	Save Test Run	4.3.1.1, 4.3.1.2, 4.3.1.3
5	Load Test Run	4.3.1.4, 4.3.2.2
6	Select Axes	4.3.1.2
7	Select Strategy	4.3.1.2
8	Initialize Process	4.3.1.1, 4.3.1.2, 4.3.1.4
9	Test Run Cycle	4.3.1.1, 4.3.1.2, 4.3.1.3, 4.3.1.4
10	Simple Scheduler	4.3.1.1, 4.3.1.2, 4.3.1.3, 4.3.1.4
11	Parallel Axes Test	4.3.1.1, 4.3.1.2, 4.3.1.3, 4.3.1.4
12	Automatic Error Recovery	
13	Suspend Axes from Test Run	
14	Continuous Reporting	4.3.2.1
15	In-Tool Reporting View	4.3.2.1, 4.3.2.2
16	Strategy Files	4.3.1.1, 4.3.1.2, 4.3.1.3, 4.3.1.4
17	Write Protection	4.3.1.1, 4.3.1.2, 4.3.1.3, 4.3.1.4
18	Informing Email Service	
19	Reporting Errors and Warnings	
20	One Click Application	4.3.1.1, 4.3.1.2
21	Strategy Overview	
22	Strategy Editor	
23	Reliability	
24	Time to Start a Test Run	4.3.1.1, 4.3.1.2, 4.3.1.3, 4.3.1.4
25	Installation	
26	Instrument Configuration	4.3.1.1, 4.3.1.2, 4.3.1.3, 4.3.1.4
27	Strategy Input	4.3.1.1, 4.3.1.2, 4.3.1.3, 4.3.1.4
28	Export Reports	4.3.2.3
29	Test Run OK/NOK Feature	4.3.1.1, 4.3.1.2, 4.3.1.3, 4.3.1.4
30	Statistic OK/NOK Feature	4.3.1.1, 4.3.1.2, 4.3.1.3, 4.3.1.4
31	Tecan Base SDK	4.3.1.1, 4.3.1.2, 4.3.1.3, 4.3.1.4

Some software specifications do not have a trace to a test. On all such specifications the implementation has not even started. That's why no test was written for this specifications.

Definition, Acronyms and Abbreviations

Project-Name: **Qualification Tool**

Project Number: **-**

Subject: **-**

Table of Contents

1 Introduction

1.1 References

Ref #	Description
Ref.[1]	http://en.wikipedia.org/w/index.php?title=Non-disclosure_agreement&oldid=496652284 (2012-06-12)

1.2 Document Change History

Date	*Version*	*Change*	*Author*
2012-06-12	0.1	First Version	AnZo

2 Definitions, Acronyms and Abbreviations

BU	Business Unit
BP	BioPharma, BU of Tecan
CAN	Controller Area Network. Specification for a network.
CD	Clinical Diagnostics, BU of Tecan
CRC	Cyclic Redundancy Check; It is an error-detecting code designed to detect accidental changes to raw computer data, and is commonly used in digital networks and storage devices
CDA	Confidentiality Disclosure Agreements, legal contract between at least two parties that outlines confidential material, knowledge, or information that the parties wish to share with one another for certain purposes, but wish to restrict access to by third parties. See Ref.[1]
CFR	Code of Federal Regulations. Codification of the general and permanent rules and regulations of the United States of America.
COM	Common Object Model. Interprocess communication mechanism.
DHF	Design History File, compilation of documentation that describes the design history.
DLL	Dynamic Link Library
DVI	Digital Video Interface. Electric interface to transfer video data.
ECTS	European Credit Transfer System
FDA	Food and Drug Administration, agency of the HHS
FuMu	Funktionsmuster, Breadboard
FW	Firmware
GUI	Graphical User Interface
HHS	United States Department of Health and Human Services, cabinet department with the goal to protecting the health of all Americans.
HSR	Hochschule Rapperswil
I2C	I^2C, Inter-Integrated Circuit, multi-master serial single-ended computer bus
ICP	Instrument Communication Protocol. Communication protocol for data transfer to Tecan instruments.
IQ	Installation Qualification
IVD	In-vitro Diagnostic
IVDD	In-vitro diagnostics directive
LH	Liquid Handling
LUA	Portuguese for moon, scripting language
M	Milestone
NA	Not Applicable
NPV	Net Present Value
OEM	Original Equipment Manufacturer
OS	Operating System. SW to manage HW resources and executing application SW.
PC	Personal Computer.
PCB	Printed Circuit Board. Mechanically supports and electrically connects electronic components.
PID	PID Controller, proportional–integral–derivative controller
PL	Project Leader
PRD	Product Requirements Document
PT	Project Team
QM	Quality Management
RA	Regulatory Affairs
R&D	Research and Development

RfV	Ready for Validation
RS232	Recommended Standard 232. Specifies a serial communication protocol.
SCR	Software Change Request
SDK	Software Development Kit. A set of development tools that allows for the creation of applications for a certain HW platform.
SOP	Standard Operating Procedure
SOUP	Software Of Unknown Provenance
S&S	Setup and Service Software
ST	System Test Team
SW	Software
TBD	To be defined later
TPTR	Test Plan and Test Report. Document for definition and report of SW unit and integration tests.
UCS	Use Case Specification
USB	Universal Serial Bus. Specifies a serial communication protocol.
VAR	Value Added Reseller
VGA	Video Graphics Array. Electric interface to transfer video data.
WPF	Windows Presentation Foundation. .NET library for GUI coding.
XML	Extensible Mark-up Language. Text-based mark-up language for structuring hierarchical data.

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

COMPUTER SCIENCE

.NET Qualification-Tool for Pipet-Instruments

Bachelor Thesis Spring Term 2012

Subject Area: Software Engineering

Andreas Zollinger

Advisor: Prof. Dr. Luc Bläser

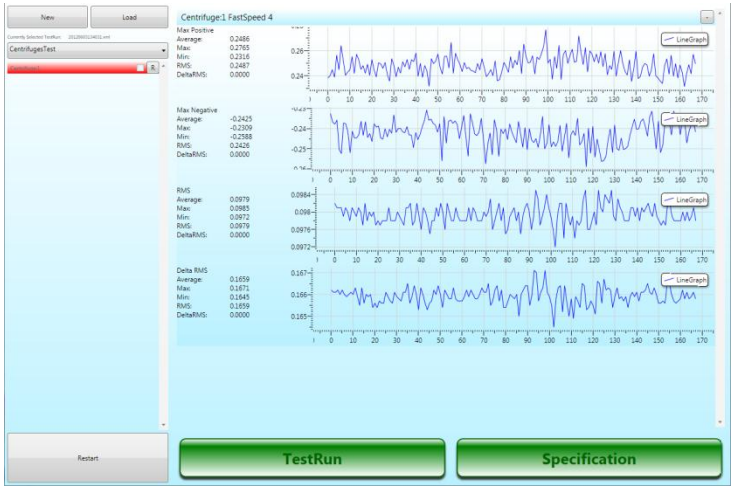
Co-Examiner: Jean-Daniel Merkli

Project Partner: Tecan Schweiz AG

Project Goal

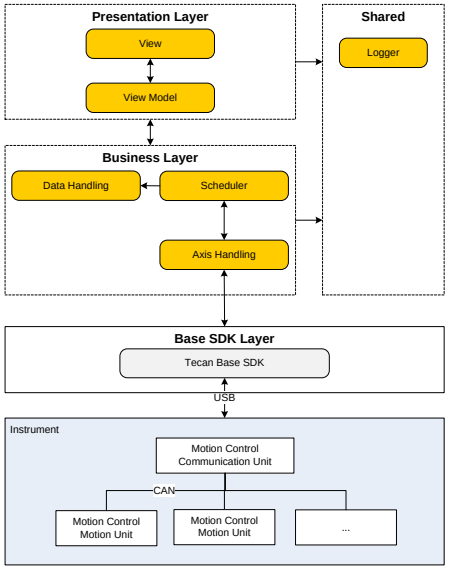
A new qualification tool for the automated validation of the motor control parameters on Tecan pipet instruments

- Detailed requirements analysis
- Design and implementation in C#, with WPF and Tecan Base SDK



Implemented Features

- Customizable test scenarios specified as “strategies” in XML
- Automated handling of concurrent test runs with declared constraints
- Gathering and reporting of test measurement results




```
<Strategy xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<Name>CentrifugesTest</Name>
  <AxisStrategies>
    <AxisStrategy Id="Centrifuge:1">
      <InitStrategy>
        <InitDependencies />
        <MoveAfterInit>
          <Position>0</Position>
          <Speed>1</Speed>
          <Acceleration>1</Acceleration>
        </MoveAfterInit>
      </InitStrategy>
      <TestStrategy>
        <TestDescriptions>
          <TestDescription>
            <TestDependencies />
            <TestCases>
              <TestCase Id="FastSpeed">
                <TestProperties>
                  <TestProperty Type="distance"
                                Value="3420" />
                  <TestProperty Type="speed"
                                Value="1" />
                  <TestProperty Type="acceleration"
                                Value="1" />
                  <TestProperty Type="deceleration"
                                Value="1" />
                  <TestProperty Type="statisticsource"
                                Value="4" />
                </TestProperties>
              </TestCase>
            </TestCases>
          </TestDescription>
        </TestDescriptions>
      </TestStrategy>
    </AxisStrategy>
  </AxisStrategies>
</Strategy>
```

Achieved Results

- Analysis and formal specification of the requirements
- Functional version of the new qualification tool
- Tool will replace current LUA script-based approach

Conclusions

Improvement of the Tecan machine validation process

- Repeated and customizable automated test runs
- Integration with the production-relevant Tecan Base SDK
- Compiled .NET application instead of LUA scripts

Future Work

- Advanced test scenarios (special moves)
- Storing of measurements in a common database
- Multi-instrument support for even faster test runs