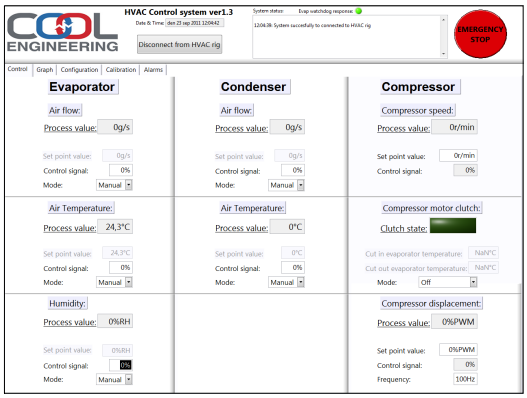
CHALMERS



DEVELOPMENT OF CONTROL SYSTEM FOR MODULARIZED HEATING, VENTILATION AND AIR CONDITIONING TEST RIG - INTRODUCING LABVIEW AS A GRAPHICAL USER INTERFACE

THESIS FOR THE DEGREE OF MASTER OF SCIENCE

Marcus Carlberg

Department of Product and Production Development
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2011

Development of control system for modularized heating, ventilation and air conditioning test rig - Introducing LabVIEW as a graphical user interface

MARCUS CARLBERG

Department of Product and Production Development

Cover:
The cover picture illustrates the old and new control system for the modularized HVAC test rig.

Published and distributed by:
Chalmers University of Technology
Department of Product and Production Development
SE-412 96 Gothenburg, Sweden
Telephone +46-(0)31-772 10 00

Print:
Chalmers Reproservice
Gothenburg, Sweden 2011

Abstract

This master thesis concerns the implementing of new software in a new control system for a modularized test rig for automotive heating, ventilation, and air conditioning, henceforth HVAC. The studied test rig is used for development testing of automotive HVAC components, and therefore many specific test cases are investigated. The test rig includes numerous actuators and sensors testing and measuring the properties of the main components (evaporator, compressor and condenser) according to the different test cases. The test rig is operated by test engineers who currently use an outdated control system based on a manual console.
The current control system of the test rig has reliability problems. Due to numerous modifications on the rig for each new test case, significant time is spent on troubleshooting the complicated control system. The idea here is that renewing and modularizing the control hardware and replacing a large amount of the sensors and actuators, would reduce the amount of downtime and increase the reliability. Another identified problem is that important functionality is also missing, e.g. active safety functions.
Development of control systems that efficiently support the test process and the user's needs is a difficult task. The data collection method used was mainly observations and three semi-structured interviews with real users of the test rig, i.e. test engineers. Use cases, process models, information models and system architectures were studied when developing the control system. Alternative solutions of the new control system were developed and evaluated. After partial implementation of the new control system, various validation processes were conducted, e.g. usability tests with the users of the test rig. The validation process aimed at ensuring that the new control system efficiently supports the test process and the user's needs.
The new control system is based on a LabVIEW graphical user interface and three Beckhoff PLCs (Programmable Logic Controller) corresponding to the three test rig modules. The usability tests showed that the new control system efficiently supports the test process and the user's needs. The reliability of the test rig is proven to be increased with the test rig modularization and the new calibration process for the regulators. The conclusion is therefore that the amount of downtime of the test rig would be reduced after a full implementation of the new control system.

Keywords: HVAC test rig, control system, LabVIEW, PLC, user interface.

Preface

This master thesis was carried out at Cool Engineering as a part of developing a new control system for their automotive HVAC test rig. I would like especially to thank my supervisor M.Sc. Eng. Magnus Blomstrand and the employees at Cool Engineering for their support and help during the project. I would also like to thank my tutor, M.Sc. Eng. Anna Tidstam, and my examiner Professor Johan Malmqvist at Chalmers.

Gothenburg, October 2011

Marcus Carlberg

Table of Contents

Chapter 1

Introduction

In this introductory chapter, the background and purpose of this master's thesis is presented together with the objectives to be met and limitations of the work. Furthermore, an outline of the thesis is provided.

1.1 Background

The background presentation of the master thesis is divided into an introduction to the company where the thesis was carried out at and then an overview of the project background.

1.1.1 Company background

This master thesis work was done at the consulting firm Cool Engineering, situated in the city of Gothenburg, Sweden. Cool Engineering was founded in 1989 and is a subsidiary of the Etteplan group since 2008. Cool Engineering primarily provides solutions and consulting in the area of automotive HVAC *(Heating, Ventilating, and Air Conditioning)* systems and engine optimization for mainly the European vehicle and workshop industry. The core businesses focus on automotive components regarding in-house development testing, development testing at customer site, building advanced customized testing equipment, and technical consulting. Cool Engineering is a LabVIEW alliance partner with National Instruments since 1996.

1.1.2 Project background

The oldest test rig at Cool Engineering, built in 1989, is a complete system for development testing and performance and lifetime analysis of different HVAC systems. A high

degree of flexibility is built into the rig construction to be able to test different automotive HVAC systems, reducing the need of any extra tubing or other equipment. This gives the possibility to test the actual solution as it would be built in the car, resulting in a more accurate test. The great benefits of using this system compared with performing a full-scale wind tunnel test with a whole car, are that a test can be done at a lower cost and earlier in the development phase without influencing the test's end result.

Cool Engineering reached a point when the test rig needed a development of the original control system. The control hardware of the system was renewed and new control software needed to be developed before it could be integrated with the test rig.// The PC is the user interface placed in the control room and the PLCs are located in three different control cabinets on the test rig.

The test rig can be divided into three main modules: Compressor, Evaporator, and Condenser. The compressor represents the steering of the vehicle's HVAC system. It pumps the refrigerant medium and therefore controls the entire test object system giving it the name of the Compressor unit. The Evaporator module represents the wind tunnel that simulates the climate conditions inside the car that the vehicle's evaporator unit is exposed to. The third module is named the Condenser, placed in the second wind tunnel simulating the outside climate that the condenser unit is normally exposed to.

The graphical user interface that will be developed is explained further on in the report, being known as the LabVIEW GUI, and the software for the PLCs as the control software.

1.2 Problem definition

The current control system has operational problems, during a testing session there is usually a significant downtime because of time spent with troubleshooting the control system. Important functionality of the control system is missing, e.g. it lacks active safety functions. Also the maintenance of the regulators within the system is too complicated. The regulators today work with non-calibrated process values which make the testing process very slow. There are also problems with controls that no longer have any functionality, or with what functionality some have. The task of this project is to develop control software fulfilling current and future needs of Cool Engineering's HVAC test rig with respect to user friendliness, maintainability and reliability. The project has embraced a deeper exploration of the requirements and building up the software by using a structured development process. The developed software should also be analyzed regarding whether it could be used as a general platform in similar hardware environment, and which parts need to be modified to fulfil its new tasks. The term User friendliness can in this case be broken down into level of automation, optimization of control parameters, presentation of values and control possibilities supporting experimentation, and understanding of the test object. Maintainability is described as

architecture based on individual modules, easy access to configuration parameters in the PLCs, and prepared and designed code for adding future hardware and functionality. These factors should result in a higher reliability and maintainability of the overall system.

1.3 Objectives

The objectives of the thesis were to investigate the user and system needs when developing a new control system with a graphical user interface introduced. The control system should be integrated with the new modularized hardware, developed by Cool Engineering. Important product aspects were to:

- Make the control system support and not slow down the HVAC testing process.
- Reduce the test rig's down time due to reliability problems with the control system.
- Develop a LabVIEW GUI platform that can efficiently be used for future configuration and refinement.
- Develop a PLC platform that can efficiently be used for future configuration and refinement.
- Ensure high integrity and clear interfaces between individual modules.
- Place critical processes and parameters in PLC modules, e.g. regulators and calibration parameters.

1.4 Limitations

The limitations of the thesis arose from initially coding all the PLCs within the test rig; due to time limitations and its similarity with the compressor and condenser PLCs, only the evaporator PLC was implemented. An initial requirement is that the new system hardware must consist of a standard PC with LabVIEW 2011 software from National Instruments and three Beckhoff PLC's with Beckhoff's owns TwinCAT software.

1.5 Time Schedule

Table 1.1 presents the approximate time schedule followed during this project. Most of the different steps overlap to a certain degree. The time schedule is based on recommendations for the product development process from Ulrich and Eppinger. [1] During

the first part of the thesis between 21 March and 29 May the pace of the work was only 50% and from 30 May until 21 October the pace was 100%.

Table 1.1: Approximate time schedule.

Time Period			Task
March 21	–	April 22	Literature survey
April 4	–	April 22	Requirement analysis and definition
April 25	–	April 27	System architecture design
May 2	–	June 7	Design user interface
June 8	–	August 19	Design PLC software
August 22	–	September 19	System validation
April 22	–	October 16	Writing of report
November 14			Final presentation

1.6 Thesis outline

The current outline of the report follows the logical development of the project.

In Chapter 2, a theoretical framework of the project is stated, connected with the purpose of the project. It begins with a theoretical and technical background behind the planning and the development of the project, and then focuses on the different methods used in each individual project step.

Chapter 3 gives a description of the development strategy used in this project.

The pre-study in Chapter 4 presents how the system specification was made and how the software architecture was created. This is rounded off with a presentation of different initial user interface concepts as a part of the software architecture.

Chapter 5 will cover the steps of implementation of the main control system and how the first PLC subsystem was programmed together with an initial heuristic evaluation.

In Chapter 6 an initial function test is performed to validate the overall functions and the integration of the different subsystems. A GUI test is conducted to ensure an adequate user interface.

The result from the implementations and the evaluations is followed up with a discussion of the final results in Chapter 7.

Ending with Chapter 8, the conclusion reconnects with the objectives according to how well the structured development process fulfilled the product aspects of the new testing system, and finally gives a recommendation for future work.

Chapter 2

Theory

The aim of this chapter is to introduce the reader to the theory behind the methods and technologies used in this thesis. The chapter follows the order in which the theory was implemented in the project, summarized in the development strategy in Chapter 3.

2.1 Automotive HVAC systems

In the automotive industry an HVAC system is more or less included as a standard component of the vehicle. The coupé in all types of vehicles, from modern buses to personal cars, is equipped with an HVAC system, also known as the AC system. The HVAC system has two main functions, to keep a steady comfortable temperature and to ventilate the air. To be able to maintain these functions the system consists of three main modules; see Figure 2.1. The overall system can be briefly described as one evaporator and one condenser with refrigerant flowing through these units transporting heat. The refrigerant is compressed by a compressor to transport more heat in the system and is regulated by an expansion valve; see Figure 2.1. The evaporator is the heat transfer unit that is connected to the airflow inside the coupé of the vehicle. Its purpose is to absorb heat from the air to create a comfortable air temperature inside the vehicle. This means that the evaporator needs to be cooler than the air that it should spontaneously cool down. The transported refrigerant is in a liquid state and is then evaporated into gas when absorbing heat in the evaporator under constant pressure and temperature. The air that passes through the evaporator can be either fresh air from the outside of the vehicle or recirculated air from the inside. The purpose of the compressor is to suck cold and low-pressure refrigerant vapour from the evaporator and compress it into a warmer high-pressure gas. To control the compressor, the displacement inside the compressor can be varied if the motor's speed of rotation (RPM) differs. The refrigerant is pumped further on into the condenser unit where the heat from the refrigerant is conducted to the air outside the vehicle. Here a change of state of the refrigerants is

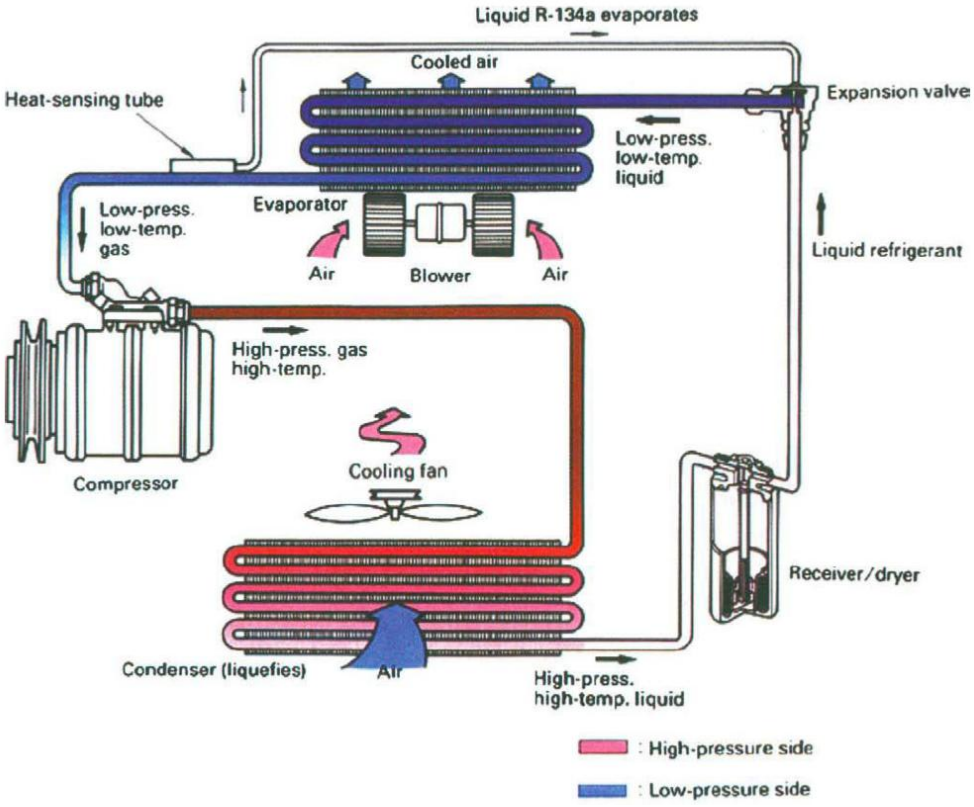


Figure 2.1: An automotive HVAC system. [2]

seen again, going from gas into a liquid form. The refrigerant is now circulated back to the evaporator via an expansion valve, closing the loop of the cooling system. The heat transfer principle is relatively simple and should be easy to handle, but this is not the case in the automotive industry; during normal operating conditions of the vehicle, the external circumstances for all the main modules can affect the HVAC system. Hence the HVAC system must be able to handle cases such as driving in extreme heat with a low airflow through the condenser unit and without recirculation of the coupé air. [2] [3]

2.2 The Waterfall model

The waterfall model is software development procedure, the outline of the model is that the different phases are distinguished as individual activities and are said to be plan-driven. The different steps are very high-level, starting with the requirement definition,

system and software design, implementation and unit testing, integration and system testing and finally operation and maintenance. These steps overlap do not overlap but there is always a possibility to reverse back in the process.
During the requirement specification phase the background of the project and its requirements are investigated; this can be done in several ways, and is adapted to each specific case. The system and software design contains activities such as establishing software architecture, designing a graphical user interface and so on. During the implementation the actual coding of the software takes place, and is usually the part that consumes the majority of the project resources.
In the unit testing step everything from individual code modules to smaller subsystems is tested to confirm its functionality and quality. Thereafter in the system integration stage these modules or other software parts are brought together to create the end product. This is later tested in the system testing phase, comparable with a factory acceptance testing or a full-scale test.
The last phase is the operation and maintenance time of the software; in this phase new versions and updates are released. [4]

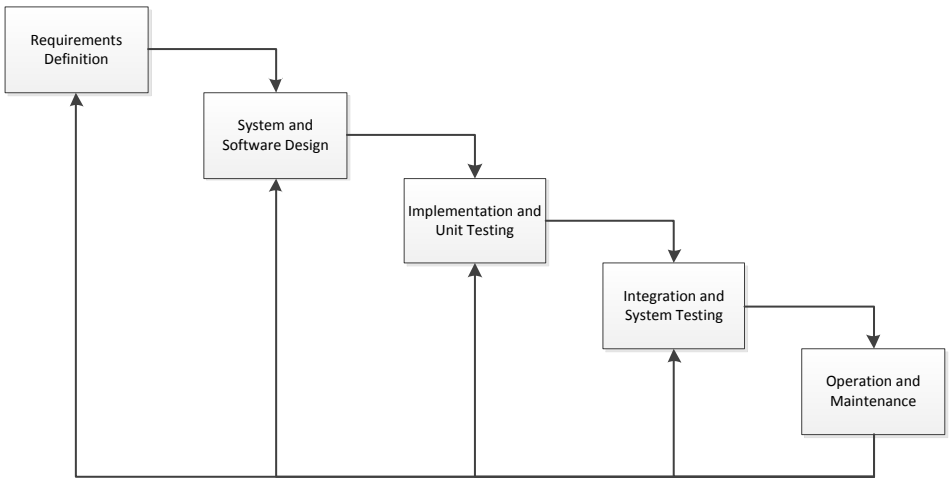


Figure 2.2: Waterfall model for software development

2.3 Agile programming

Agile programming is a rapid software development process which is supposed to result in useful software after a short period of time. The method is highly iterative and differs a lot depending on the programmer. The common outline of the method, though, is that the process does not need any specific or comprehensive specification; this is normally developed during the programming phase as the project proceeds. The system is released in a great number of versions increasing in functionality after each version. [4]

2.4 Requirement elicitation and analysis

This is an iterative process for creating a requirement specification based on interaction with both stakeholders and the systems, in order to understand what are the constrains, the required performance and so on for the application. To find out the needs of the stakeholders and the end-users, a process model for requirement elicitation and analysis is followed:

1. Requirement discovery – In the initial step contains interaction with stakeholders, such as interviewing and observations.

2. Requirement classification and organization – All the impressions from the interaction with the stakeholders are documented and organized into groups and related to each other.

3. Requirement prioritization and negotiation – The result from the documentation is brought back to the stakeholders for discussion and negotiation and prioritization. Conflicting requirements need to be resolved and reach a compromise for what should be included.

4. Requirement specification – The requirements are documented into a formal or informal requirement specification which usually is revisited later on during the overall project process, or this cycle is performed again.

This is a difficult process and normally needs to be revisited several times to discover and understand all the requirements. [4]

2.5 LabVIEW 2010

LabVIEW 2010 is a software tool for coding measurement and control software. Its name is short for Laboratory Virtual Instrumentation Engineering Workbench, and is

software made by National Instruments. [9] It is a platform and development environment for automating processes and measuring in laboratory environments. The programming of the software is graphical instead of pure text and is simply called "G". The environment is built up around one front panel which serves as the GUI, one block diagram containing all the code, and a connector panel. All programs made in LabVIEW are called VI, which stands for Virtual Instrument. These may be the main executive code of a program or a subVI used in another VI which can be compared with a sub routine in a textual language. The code is constructed by dragging and dropping graphical representations of lab equipment and wiring these together. By using these graphical representations in the block diagram, it is easier for inexperienced coders to develop their own small applications. But the width of the language also makes it possible to create more extensive applications for more advanced data acquisition or process automation. Compared with many other development environments, parallel execution is easily achieved, making the G language very powerful. An additional strength of LabVIEW is that it is easily integrated with various types of measurement instrumentation and data acquisition hardware. [10]

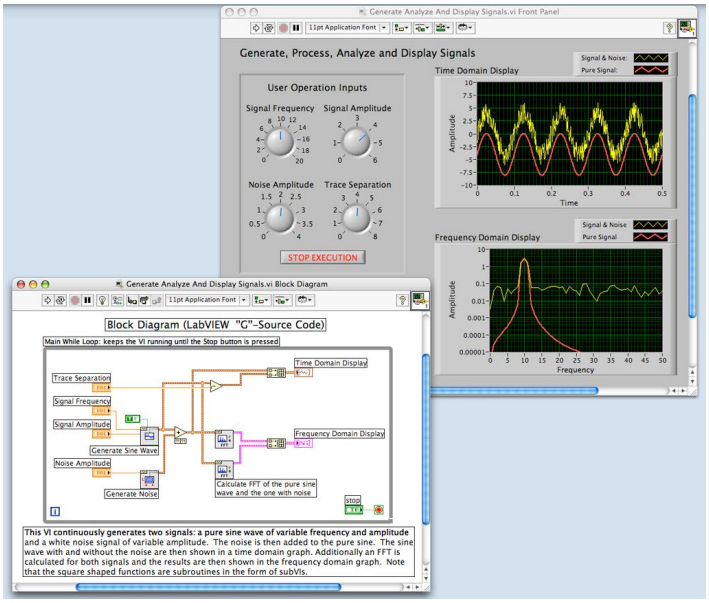


Figure 2.3: A simple LabVIEW program

2.6 Producer/Consumer loop

A Producer/Consumer loop is a design pattern used in LabVIEW for handling multiple processes simultaneously, often operating at different frequencies. The pattern can be seen in Figure 2.4 and is a further development of the Master/Slave pattern, with the big difference that the Producer/Consumer loop can share data between multiple loops in a more flexible way. [11] The pattern can be described as two different loops, the producer and the consumer. The producer communicates with the consumer by using a data queue. The producer sends different data into the queue depending on what task that needs to be done by the consumer. It is common that the producer loop contains an event structure which is triggered by events created within the LabVIEW system. An event can be created by a key pressed down or a change of a parameter's value. One of the most common events is the so-called 'time out' event triggered by a timer connected to the Producer loop, described in this project as a time-out event. When the queued-up data reach the consumer loop, it handles the first data and responds to them and continues down in the queue. E.g. when an event is triggered in the producer loop it sends data to the queue giving an order to run a certain part of code within the consumer loop. A common problem with the producer/consumer loop is that the queue gets overloaded with data, which means that the program will not perform its task properly. This can be avoided by using multiple producer/consumer loops working parallel. [11]

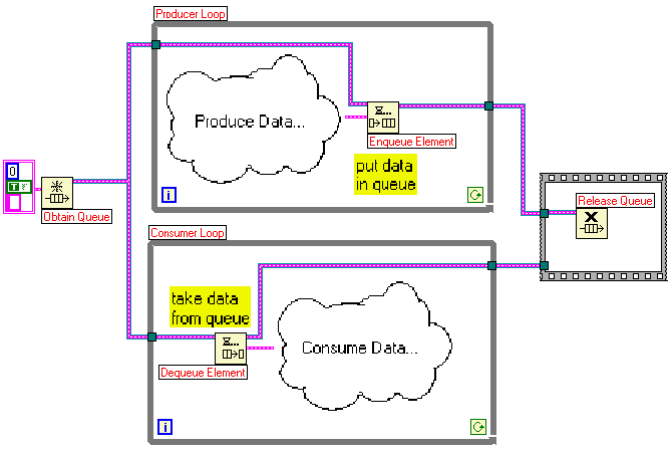


Figure 2.4: The producer consumer loop

2.7 TwinCAT v.2.11

TwinCAT is the development environment from Beckhoff Automation for their PLC and motion units. According to the supported IEC 61131-3 standard, TwinCAT can handle the programming languages; Sequential Function Charts (SFC), Ladderdiagram (LD), Instruction List (IL), Function Block Diagram (FBD), and Structured Text (ST). The software contains the basic functions for coding and debugging, but some of the interesting features of TwinCAT are the possibility to connect to run-time system via TCP/IP or Fieldbuss and online change of parameters within the running code. [12]

2.8 Think-aloud evaluation

The think-aloud method is an evaluation method where the evaluator is exposed to tasks or scenarios to be conducted or solved on the tested GUI. [14] During the session the participant should say what is passing through his or her mind when interacting with the human-machine system. The comments are recorded, and in some cases also the screen motions or facial expressions are filmed. The clear advantage of this method is the insight into how the user is reasoning when performing representative tasks. But to gain as much as possible from the test, it is important that the evaluator feels comfortable enough to be fully able to think aloud. The evaluation session spans from 60 to 90 minutes with performing different tasks of different significance. The method can in some evaluatiosn be successfully combined with cognitive walk-throughs. [5] [14]

Chapter 3

Research approach

This chapter introduces the reader to the development methods and principles used, and to the order in which they were applied in this project.

3.1 Development Plan

The development strategy plan was established to improve and define the development process, which was refined as the project proceeded. It is presented in a short paragraph form in the logical order, showing the different phases and the main methods or processes that were included in these phases. The outlines of the strategy is based on the water-fall model, which is a structure for software development, but the implementation steps of the project had an agile character to increase the pace of the project.

1. Literature Survey - To embrace the theoretical background of the project, a literature survey was made, based on internal material at Cool Engineering and external sources. In the study, methods that could be implemented in the thesis were investigated further in detail. The theory behind the methods and principles in the thesis is presented in Chapter 2.

2. Pre-study of requirement definition and concepts - As the requirement definition step of the waterfall model this phase of the project included the most methods. The reason for involving so many methods was to create a wide picture of what the requirements of the system are. To enhance the quality of the end result, a considerable part of the time was reserved for this phase; see time schedule 1.1.

    - Interviews and observations - As an initial and very broad method, interviewing was used to create a picture of who the stakeholders are. What the stakeholder requirements are, and the system requirements, needed to be acknowledged. First, small unstructured interviews in combination with

observations were made to gain a basic understanding of the test rig. As a second step, the structured interviews were made with the main stakeholders to get everything documented.

- Affinity diagram - As a method to refine the results of the interviews, an Affinity diagram was made. This helped to extract the most important statements, see which once were similar and group these together. The results of the diagram were interpreted into the requirement specification.
- System vision - To describe and get a clear picture in a natural language of what the stakeholder wanted to achieve with the software, a system vision was established. The vision was revisited during the development as a reminder to continue the development process in the correct direction. It served as a good tool for external communication of the development work.
- UML class diagram - The class modelling tool clarified the relationship of the different entities of the overall control system. As a first step in creating an architecture, it helped in defining and organizing all the functions that needed to be included.
- UML Use case - The use case was a good method that gave an overall picture of how the users and subsystems interacted with each other during operation of the current system. In this case there was a need to document more precisely which persons were active in each step and which subsystems were involved.
- IDEF0 - In order to get a deeper understanding of what the different processes around a system consist of, the processing modelling method was helpful. It served as a tool for digging deeper into the processes in a structural manner and to finally document these processes in a standardized way.
- Requirement classification - To organize all the requirements according to what type they are, a requirement classification tree was used. The background reason for using this was to be able to get a clearer structure in the requirement specification document and to be more precise when communicating the requirements.
- Requirement validation - In order to ensure the quality and validity of the requirements this process was included. This step was used as a tool for negotiating the prioritization of each requirement and ensuring that they were formulated correctly. The end result from this method was the first full requirement specification.

3. System architecture design - As the first part of the system and software design in the waterfall method, an architecture for both the control software of the GUI and

the PLCs was established. Various different architecture types were considered but the ones that would fit the system were chosen.

- Layered software architecture - the motives behind choosing this architecture are explained in section 4.6.1.
- Repository architecture - the motives behind choosing this architecture are explained in section 4.6.2.

4. Design user interface - This was the system implementation step according to the waterfall model development process. This phase of the project has been very agile in the development, meaning that small releases of the software were made and refined.

    - The eight golden rules of interface design guidelines - These guidelines were used during the implementation of the GUI and the background system. They help e.g. to maintain consistency, use correct terminology, and so on. These guidelines are well recognized in the field of GUI design, which was one of the main reasons for including them in the project.
    - LabVIEW coding - In the project background by the project owners stated the system requirement that GUI control software should be developed in the LabVIEW environment.

5. Design PLC software - this was the implementation step of the PLC software whose code base should be used in all PLCs with minor modification.

    - Structured text coding - As a system requirement it was stated that the coding had to follow the company standard at Cool Engineering of using only structured text code in the PLCs.
    - Integration verification with User interface and IO-modules - As a stage gate of the implementation phase this was included to verify that the communication between the LabVIEW GUI and the PLC was functioning.

6. System validation - In the scope of this master thesis, this was the last step of the waterfall model covered in this project. Before handing over the project its functionality and performance needs to be validated in order to ensure its quality and to reflect on the previous method's results. A broader range of validation methods or principles has been used, so as to be able to create a strong validation of the whole system. During this step some incremental development of the software was included in the validation process to improve the final result.

    - Heuristic evaluation of GUI - The first real evaluation of the GUI was made using the Heuristic evaluation method developed by Nielsen [8]. It served as

an initial filter to find flaws in the interface that could be fixed to enhance the final result in the think-aloud tests.

- Think-aloud GUI evaluation - In this validation the think-aloud method was used. This is a simple tool to document easily how good the interaction between the evaluator and the user interface is. In situations where a low number of evaluators are available this method could highlight the majority of the system characteristics. The documentation of the evaluation was a video recording of the computer monitor together with a voice recording of the evaluator's comments. The evaluator was informed how the method works and shown a think-aloud video-protocol for a deeper understanding of how the method works.
- System function validation - As the final step in the evaluation process, a function validation was performed on the functions that were possible to test. By using this method it could be verified that measuring, configuring and controlling the processes is possible. The system performs well during the evaluation and has opened the door to continually expanding the implementation of the system to the remaining PLC units. The test was a mixture of top-down and bottom-up integration starting with testing individual functions of the system, making measurements then adding control functions etc. for a systematic approach. The first unit tested was the one that is probably most frequently used to be exposed to maximum testing according to Sommeville's rule of thumb. The testing was divided into two different tests due to no possibility to test on the actual HVAC system. The results were observed and discussed together with a test engineer to judge the system performance according to the project objectives.

Chapter 4

Pre-study - System specification

In this chapter the system specification process is presented, walking through the requirement process, resulting in a requirement specification and software architecture and interface design.

4.1 Requirement exploration

As a step of the development strategy, a system specification was made; in this pre-study the background research of the current system is presented. This is complemented with the voices from the system stakeholders which are interpreted in an affinity diagram. The background information and the affinity diagram were used as input to the task analysis. All these results were finally broken down into a first full system requirement specification. From the system specification software architectures were created and an interface design was presented.

4.1.1 The HVAC test rig at Cool Engineering

The test rig and its control system which was modified in this project were built in 1989 and have been changed throughout the years. The system is built up around two wind tunnels and a compressor control unit. This is described schematically in Figure 4.1.

To simulate, the interior coupé climate that the evaporator is exposed to is created inside the so-called evaporator tunnel. Air representing the coupé climate recirculates and is manipulated by heating it and adding humidity inside the tunnel to create different test cases. To change the relative humidity a steam humidifier is used; the unit boils water and distributes it by opening a valve. To change the temperature, the air is warmed by an electrical heater. The heater control signal is a very slow PWM *Pulse-width modulation* signal with a period time of several seconds.

The condenser tunnel represents the outside climate that the condenser is exposed to. This tunnel can heat up the air and exchanges air continuously with fresh air from outside the building. To control the rate of exchanged air an orifice, described as Orifice 1 in Figure 4.1, is opened manually. If the hole is wide a low part of the air will go through the return duct and thus also the supply duct. With a small orifice opening, less air can pass through the orifice and a lower rate of the air recirculates in the tunnel. The heater unit used in the condenser tunnel is of the same type as the one used in the evaporator tunnel. The second orifice in the condenser tunnel, Orifice 2 in the figure, creates a drop pressure that makes it possible to calculate the airflow using the Bernoulli principle. The same type of orifice is used in the evaporator tunnel to calculate the airflow.

In the compressor unit, three things are controlled: the clutch between the compressor motor and compressor, the compressor motor speed, and the displacement inside the compressor. The motor is a normal synchronic electrical motor with a motor drive unit which uses a control signal of a voltage level between 0 and 10 VDC. The clutch is steered with a solenoid that is activated by a digital high/low signal. The displacement inside the compressor is varied by a PWM control signal that operates on different frequencies depending on the manufacturer of the compressor test object.

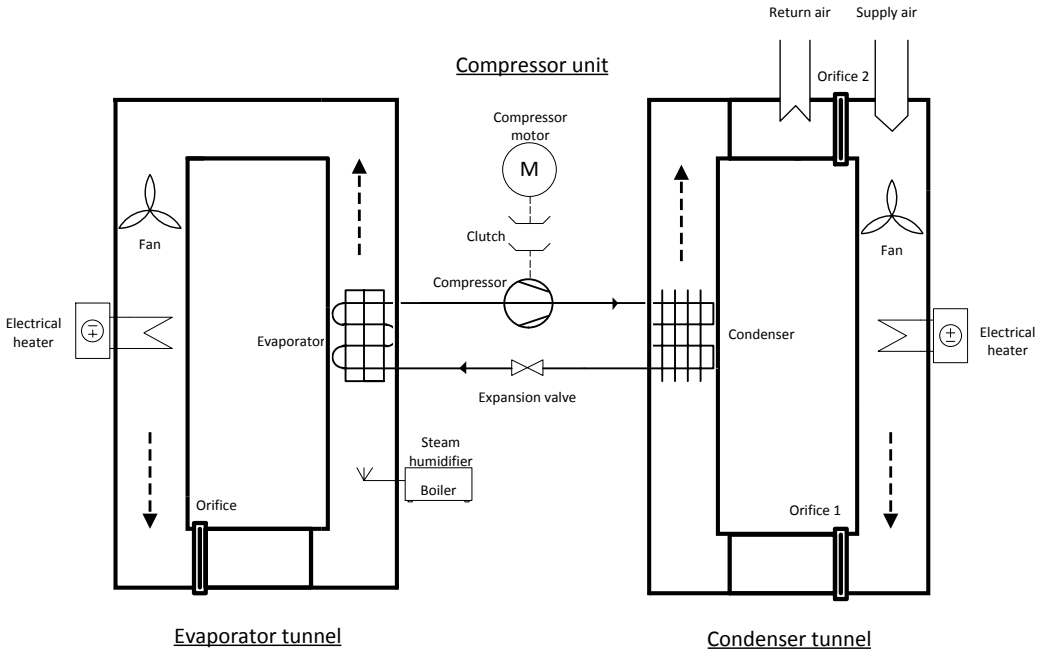


Figure 4.1: Schematic description of the HVAC test rig

The control panel for the entire system is placed inside a control room with an overview of the actual test rig. It can be seen in Figure 4.2; the layout of the panel is divided according to the different units of the test rig. /par During the years, the system has been upgraded and repaired, and today many objects on the panel have no function and external small control panels have been placed next to it. The motor drive was changed for the compressor motor and the old control panel unit was not compatible with the new drive which explains the solution. The compressor motor speed is steered by the grey unit in the bottom left corner in Figure 4.2. To steer the humidity inside the evaporator a small unit is place in the middle on the original panel.

The evaporator and condenser tunnels heaters are steered by two similar PID temperature controllers from Eurotherm. The controller today uses very old PID parameters, which results in a non-optimized and slow system, the process value presented on the control unit is not calibrated and cannot be calibrated. This means that the system cannot be controlled properly. /par

The fans in the evaporator and condenser unit use the same motor drives and control panels, They are PID controllers that can be set according to a certain RPM on the fans, and not according to mass of air flowing through the system. From the evaporator part of the panel also the relative humidity level is steered by a PID control loop, which performs well except that the valve solenoid has a very high hysteresis when steered. To control the compressor, the user can set that clutch state manually on/off or in cycling mode steered externally by a data acquisition computer manually on/off. The displacement inside the compressor is steered by a PWM control signal created in one of the control units placed on the panel, and the frequency is set by another unit. There is no active safety in the system.

4.1.2 Stakeholders

The stakeholders were identified as an initial step of extracting the user needs and requirements.

Test engineer

The test engineer is the main operator of the test rig, so the user interface has the main impact on this stakeholder. This group also does development testing, maintenance and configurations on the rig such as calibrating parameters and optimize control parameters. The test engineers have good background knowledge of the HVAC processes and the technology inside the test rig. This gives a possibility to create a more powerful system and with more advanced configuration possibilities. A faster and more configurable control system will decrease the testing time and give the possibility to make more advanced and experimental tests.

Figure 4.2: The old control panel of the HVAC test rig

Manager

The management wants to increase the lifetime and uptime of the test rig combined with faster configuration and maintenance of the system. There is also a great interest in keeping both the software and hardware of the system modularized to enable the possibility to move the control system to an entirely new HVAC test rig in the future.

Maintenance engineer

This stakeholder is involved in the larger, normally planned maintenance and modifications of the system and needs to be able to access and understand how the software is built up and how to implement new hardware to the test rig. It is common that this person also is a test engineer.

Customer

The customer is an external person who is usually involved in the planning and the result phases of the testing process. Many of the customers also participate during the testing season, which brings new aspects into the picture. All decisions in the user interface will have an indirect effect on the customer's perception of the test. The

system, especially the GUI needs to give a good impression of being correct and reliable for the end customer to gain trust in the professionalism of the test process.

4.1.3 Informal interviews and observations

To get a better understanding and to create a more natural conversation to reveal uncovered product aspects, informal interviews was performed, which increased the trust and understanding of the stakeholders. One test engineer also demonstrated the test rig and presented some of the usability problems mainly related to the control panel. Persons from all stakeholder categories were involved in these discussions and observation sessions. These interactions was documented and served as a good basis for constructing a relevant structured interview guideline.

4.1.4 Structured interviews

The structured interviews became both of an explorative and a confirming kind, which gave a good basis for collecting raw data to finally result in a requirement specification. Three interviews were performed with individuals who can in this context be described as lead users; two of these interviews were with test engineers and the third with a maintenance engineer. The questions in the interviews can be found in Appendix A. The initial questions were on a general level, such as "What should be presented on the screens?" and "What should be controlled on the screens?" which was followed up with more specific questions. The interviews lasted between 20 and 35 minutes.

4.1.5 Affinity diagram

The results from the interviews and observations were refined and organized in a modified version of an affinity diagram to extract the essential requirements and characteristics of the control system. The interpreted data were grouped, but no interconnection between the groups was established which the original affinity diagram contains. The result can be found in Appendix B. By grouping the statements into groups, it was easier to see what the most important product aspects were. These groups later formed the subcategories in the requirement specification.

4.2 System vision

By using structured methods ranging from affinity diagrams to UML process modelling the purpose of this system requirement is to explore the spectrum of demands and needs more deeply in the very beginning of the development process. Following this

path will hopefully bring more value into the process in the beginning, so as to shorten development time and give a better end result of the new system. The vision with the new software is to provide the possibility to modernize the test rig, which holds a great potential for improvement. The user interface is today lacking in support of the testing process for the test engineer. There is a possibility to raise the level of automation in this part of the system, but also to present the data from the process in a more clear and flexible way, helping the test engineer to gain deeper knowledge of the tested HVAC system faster and easier. The new system is to contain a new level of automation for reaching steady state and integrating safety functions. The new system should cover more maintenance aspect coming to modularity and configuration. All calibration and control parameters stored in the PLCs should be accessible and configurable to ease regular maintenance. The system should be evaluated on its ability to be used as a template for designing basic control systems based on a PC containing LabVIEW that communicates with the test rig PLCs.

4.3 Process model

The testing processes are described by using use case and IDEF0 diagrams. They present the general testing process on a high level and can be compared with the IDEF0 diagrams on the second level. The use case gives a basic view and understanding of the different roles involved in the testing process. The IDEF0 tables shows the process in greater detail, giving a deeper knowledge of which individual steps are involved and their different mechanism. The information flow between these steps was also visualized efficiently. These tables can be found in Appendix C. Maintenance of the system is a common process within the system, this have not been documented into a process model since it varies too much from time to time. A model of it would not give a clear or full picture of the interaction.

The general testing process

The process can be described as development testing, and the initiation begins with a request from a customer of a new HVAC test; see Figure 4.3. A discussion takes place between the parts represented by the customer, the test engineer and the management is made regarding the test object and the test plan. After reaching an agreement the test engineer can proceed with the test plan and the test object(s) as a basis for configuring the HVAC test rig. Before starting the actual testing process, a function test of the new rig configuration is made. The dynamics of the system is then verified and hopefully ready for the real test.

The actual test can vary, but in general it is common that the customer is present during the whole or parts of the testing process. This brings in new aspects such as that

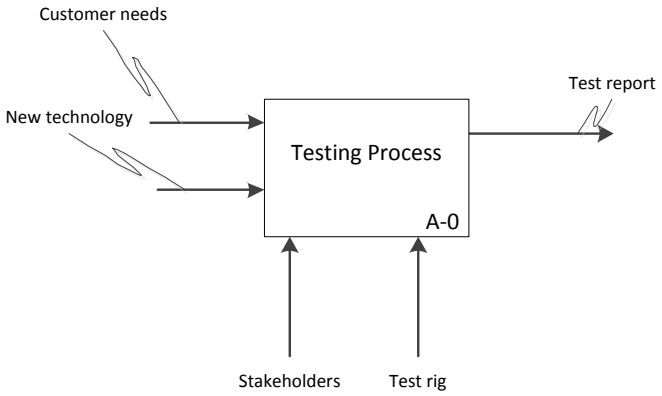


Figure 4.3: The general testing process IDEF0 A-0.

the user interface must look appealing and trustful even though the customer is not directly involved in operating the system. The process can be seen in Figure 4.4. Two of the most common test made on the test objects are the Complete performance test and Ice guard optimization. In the Complete performance test, the test object is put into extreme operating situations such as transferring high amounts of heat in combination with a low RPM on the compressor unit. In the Ice guard optimization test it is tested when ice appears on the evaporator and whether the ice guard reacts before this. The list of possible test can be made long and this introduces the need for a flexible system. The details of these specific tests and the rest of the general test process can be found in Appendix C. All measurements data from the test rig are logged in CoolSeq which is analyzed by the test engineer

4.4 Requirement classification, organization and validation

With the result of the information collection from the stakeholders and the process modelling a requirement specification was created. The different requirements were organized according to their topic and the type of requirement by using a requirement classification tree. [4] As a final step of the requirement elicitation and analysis a formal requirement reviewing session was made to ensure the validity of the system requirements, together with persons from the management and the test engineers. Step by step each requirement was inspected and discussed, mainly regarding validity, consistency

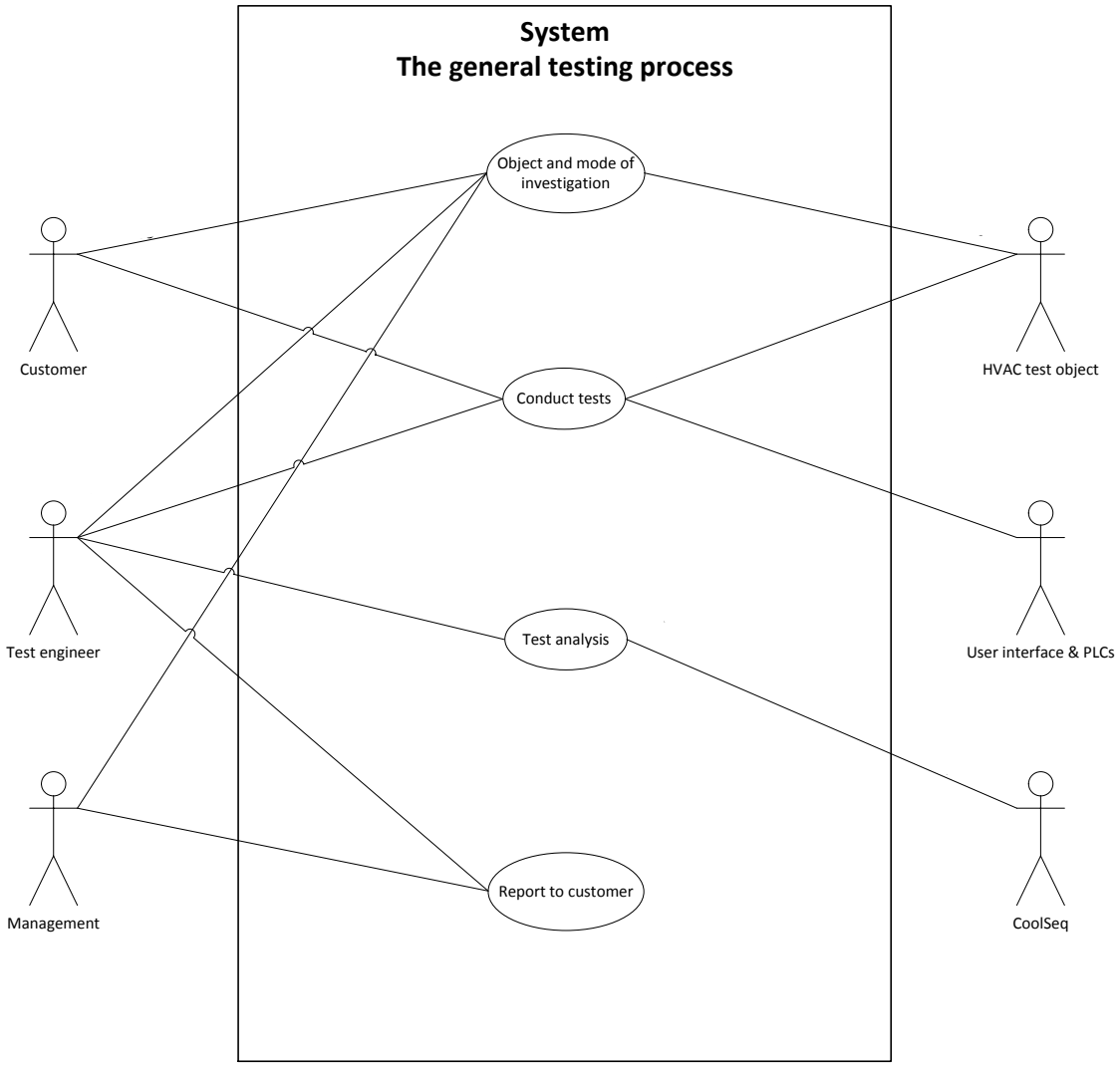


Figure 4.4: The general testing process use case.

and completeness. The priority of each requirement was negotiated. The verification method was also covered in the session, but with less focus on it due to the simple nature of the requirements in this case.

4.5 System requirement result

The system specification was based mainly on the information collected by informal and structured interviews with four people at Cool Engineering and technical documentation provided from the company. The outcome of this has been compiled by using the affinity diagram, information model, use cases and IDEF0 diagrams. The final result is a document that summarizes the majority of the testing processes needs.

4.5.1 Final version of requirement specification

The requirements was defined in different ways depending on whether they were user requirement or system requirements. These were then classified according to a requirement specification tree developed by Sommerville. [4] The requirements are described in a natural language format and shown in Table 4.1 and Table 4.2; this is a modified version of the natural language format used by Sommerville. The final requirement specification can be found in Appendix E.

Table 4.1: Template for user requirements

Name of specification		
Function	–	*What is the function?*
Description	–	*How is the function performed?*
Stakeholders	–	*Who is affected by the requirement?*
Priority	–	*From a scale 1 to 5*
Inputs	–	*What data inputs does it need?*
Output	–	*What is the output?*
Action	–	*What actions will be taken by the result?*
Requirements	–	*What will the function need to function?*
Side Effects	–	*What are the side effects?*
Requirement class	–	*What type of requirement according to the classification tree*
Verification method	–	*How should the requirement be verified?*

Table 4.2: Template for system requirements

Name of specification		
Description	–	*What is it and how is the function performed?*
Requirement class	–	*What type of requirement according to the classification tree*
Stakeholders	–	*Who is affected by the requirement?*
Priority	–	*From scale 1 to 5*
Verification method	–	*How should the requirement be verified?*

4.5.2 System architecture

The system architecture has been specified in the system requirements in the very beginning of the project and has not been a part of the development process; see Figure 4.5. But by having this configuration of several individual computers, advantages could be gained such as scalability and a higher fault tolerance. There is also a risk of higher complexity and security faults due to communication problems between the units.

The basic architecture of the system can be described as a control system with user interface acting as the overall system controlling and sharing data with the underlying PLCs and CoolSeq. The data contain steering and feedback signals from actuators or sensors closely linked to the actuators of the test rig. The user interface sends and receives control data from the different PLCs via Ethernet over the local area network, making the PLCs easier to access; this type of communication makes the main control system less dependent on specific PLC manufacturers. The PLCs communicate further on with individual actuators and sensors placed in the system. PLCs have the role of doing the active steering with the different units in the system. The data from the PLCs to the user interface are passed on to CoolSeq, which is logging all the activity in the test rig.

4.5.3 Information model

The information model is based on UML and is presented in Appendix D. This version is still on a general level and specific attributes are still not presented in the different classes. The system is based on the two classes; User interface and PLC module, which are both composed of several subclasses. The reason for not including any attributes in the model is that the system design was still not settled and should give a picture of the decomposition of the system components.

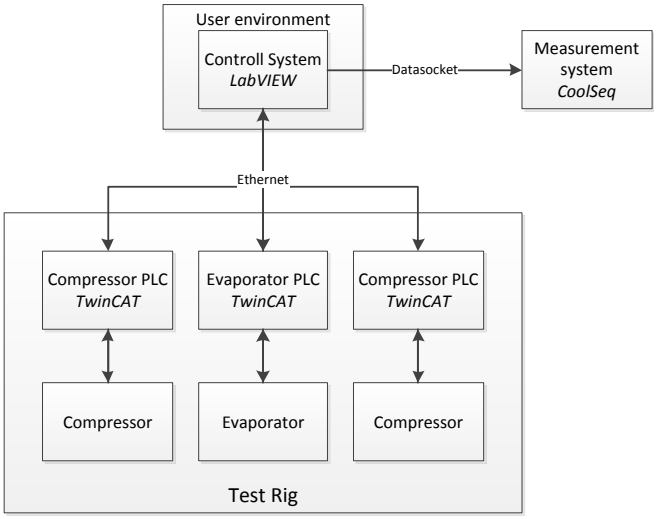


Figure 4.5: Basic system architecture.

4.6 Software architecture

To develop the control software for the HVAC-system and potentially reuse it in other system, a well-designed software architecture was of the highest essence. According to Bass an explicit design and documentation of the architecture will influence the possibility for large-scale reuse of the software. [15] The critical factors for this current type of system are Performance, Safety and Maintainability according to the requirement specification. These factors have been used as guidelines in the architecture design, but other factors not mentioned have also been considered to a certain degree. [16]

4.6.1 Control system

An event-driven system has been chosen by using the standardized design pattern for Producer-consumer loops which has been chosen as a backbone in the architectural design. In this case, multiple Producer-consumer loops are used as individual parallel working modules to run the system. This architecture can be seen in Figure 4.6. Each relevant external or internal event triggers a module of code assigned to each event in the consumer, creating a second level of modules handled inside the Producer-consumer loops. These abilities of LabVIEW give the possibility to create a layered architecture to describe the overall control system. The first basic layer is the background execution

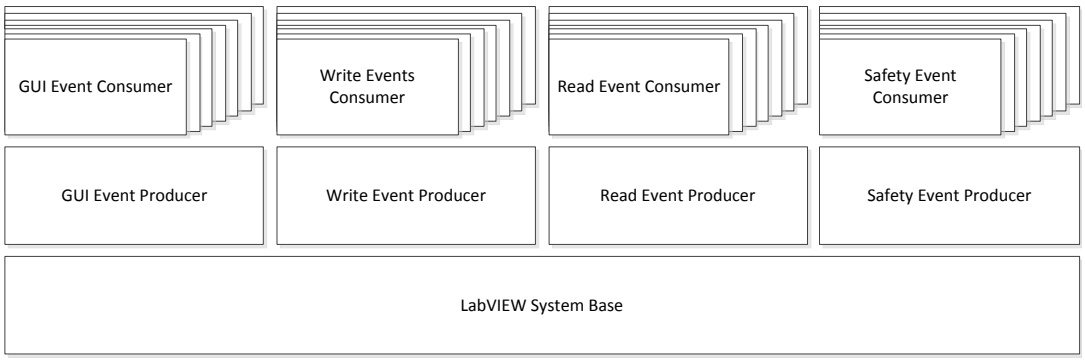


Figure 4.6: The Control system layered architecture.

that is built in LabVIEW, the second layer is each producer, and the third is the final layer containing different modules of code as a response to each event triggered inside the consumer./par To handle the limitation of how many cases can be managed by each Producer-consumer loop regarding the execution time, the loops need to be divided and organized into different categories. Each loop is limited to a clear category of events that should be handled to ensure that all critical cases are covered. In this architecture the events connected to the GUI are handled by one assigned loop. The sending data to each PLC are maintained by a second loop due to the high frequency of communication steered by a timeout event. The third handles reading data from each PLC, which are also put into a single loop due to the communication frequency. The fourth loop is a safety-critical loop which catches the rare but critical events and responds to these.

4.6.2 PLC

The different PLCs presented in Figure 4.5 were coded in the structural text language according to the standard at Cool Engineering. Together with other requirements a basic design of the software in a flowchart was constructed(see Appendix F), based on fundamental repository architecture. By using the repository model it is clarified that the individual functions do not interact with each other directly, only through the main program and the shared global variables. The architecture is presented in Figure 4.7. The model is used for describing a system that handles and shares a large amount of data with tightly integrated subsystems, which in this case fits well. The structure of the template consists of a main program which follows a procedure close to the industrial standard, which can briefly be described by looping, scan input status,

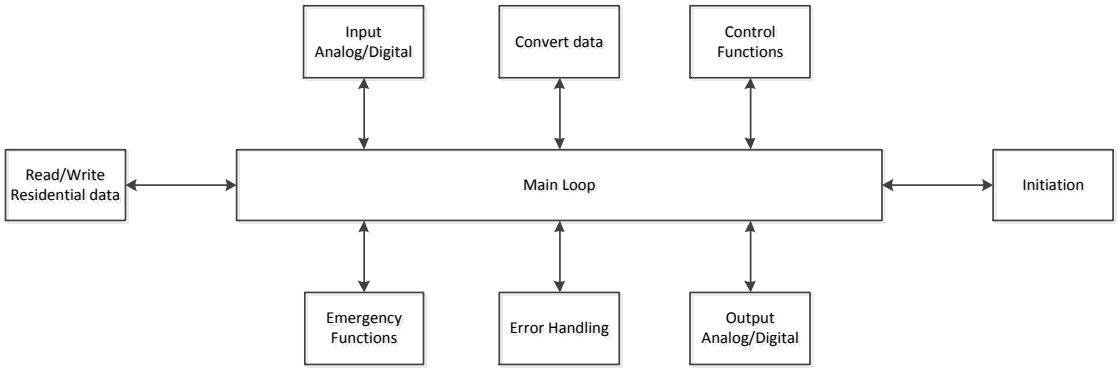


Figure 4.7: The PLCs repository architecture.

execute program and update output status. In the main program, each process step contains sub-processes. It is here that the specific program for the HVAC control system is created by inserting individual control algorithms the specific system is created. The different control functions were divided into sub functions only sharing the data handling functions, such as reading and writing to the actuators. To follow the architecture most of the variables are declared globally to handle the large amount of data so it can be accessed fast by the sub functions. The main motive behind this architecture is that, when adapting the code for each PLC, only certain exchangeable modules should only be affected. This covers the shared variables and the control functions for each actuator steered by the PLC.

4.7 Graphical user interface

The GUI was designed according to the specification and the guidelines the eight golden rules of interface design. [6] In early graphical prototypes created in Microsoft Visio (see Figure 4.8(a)), a main top window with basic controls and indicators was designed, and complemented with a tab control containing controls and indicators for more specific tasks. In Figure 4.8(b) the second interface concept can be seen; this concept was not followed up because that the tab control in the first concept was more convenient and used less screen space. It should be mentioned that it is also easier to see which view is active with the big buttons to the left in concept 2. The system is designed for a target group with high knowledge of the system they are using and expert users. The GUI is not especially adapted for very novice users according to the user profiles that are confronted with this software. From the system requirement specification five gen-

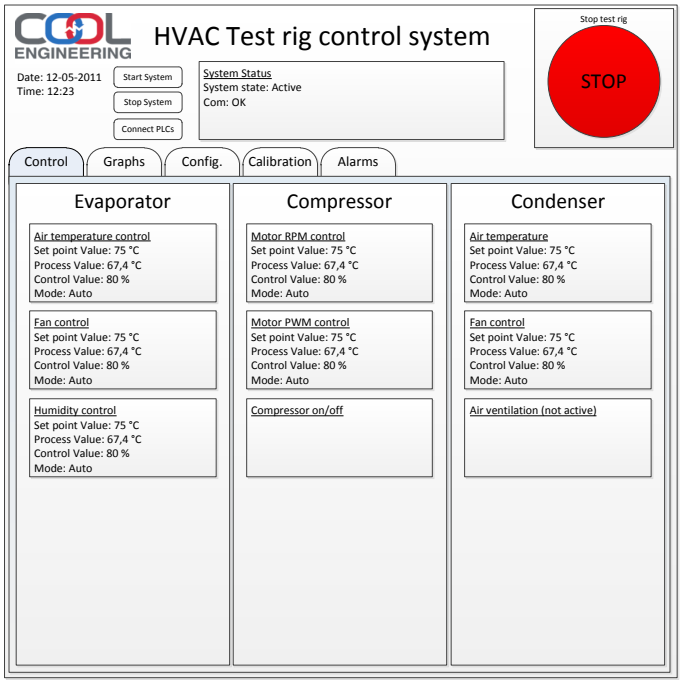
eral functions were identified; "Controlling the actuators", "Showing graphical trends", "Advanced configuration of the system", "Calibration of sensors", and "Configuration of alarm functions". These general functions were divided into individual tabs for a natural arrangement of the system.

4.7.1 Main view

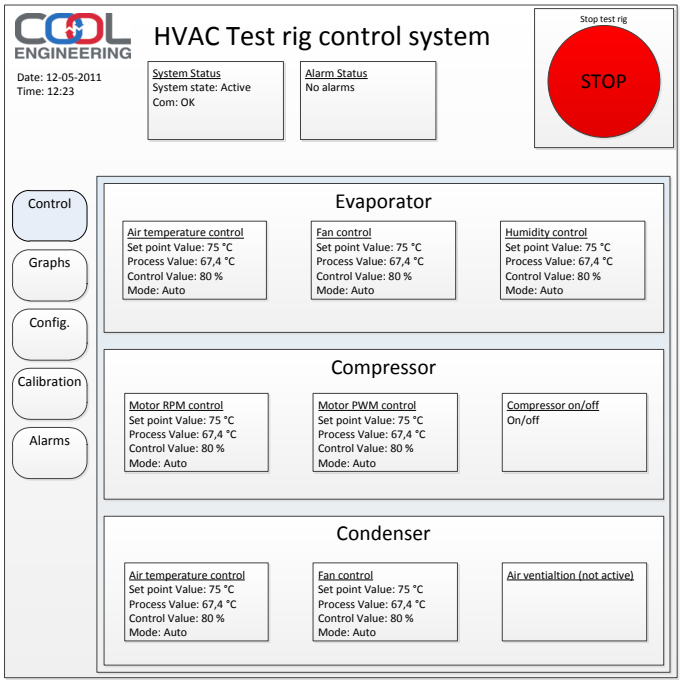
The main view window contains essential controllers and indicators that need to be available continuously for the user when operating the system. In this section the GUI contains basic and safety-critical functions and information that make it important to keep the information and functions low and relevant to serve its purpose fully. The view contains mainly controllers to start and stop the system, connect to the PLCs, and an emergency stop. It also has a text box for indicating time and date, and most importantly, a text box presenting the system status. This text box serves as a feedback for functions within the main view, but also gives feedback from the specific functions placed in the tab pages below. The system status box also serves as a short-term logger of error events created in the system. In the main view a picture of the Cool Engineering logo is placed, to identify and increase the feeling of professionalism for the stakeholders. This company logo is meant to be exchanged if the software is used by another company, i.e. implemented externally by the Cool Engineering Company.

4.7.2 Control tab

The control tab is placed as the first tab according to its priority and usage. It should serve as the operating view for the user and gives the feeling of control according to Fleischanderl's rule of a centre of control. [6] To make the interface more visually pleasing, a symmetry and consistency of the controls was striven for by using a general actuator control design, presented in Figure 4.9. The user has a bottom-up process to identify the relevant information while operating the system. The idea behind placing the process value directly under the darkly highlighted title of the controller is that the eye tends to focus from dark areas to light areas and from big objects to smaller ones. [14] But also following from the upper left corner down to the bottom right corner, enhance the guidance of the eyes' focus. The user can also find the set point signal next to the process value to reduce the short-term memory load. By carefully choosing the most relevant controls in the control tab the need of short cuts in the system is reduced also for the experienced user.



(a)



(b)

Figure 4.8: a)GUI concept 1 b)GUI concept 2, of the main view and the control view

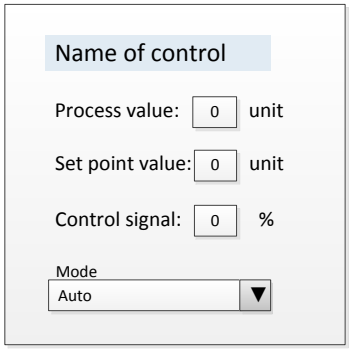


Figure 4.9: The control module used in the control tab.

4.7.3 Graph tab

The graph tab shows the trends of the signals within the system; it serves as a complementary tool to the control tab during normal operation of the system but also supports the calibration process done in the calibration tab and PID configuration. It is placed as the second tab due to frequent usage during both operation and tuning in configuration parameters. It is presented in Figure 4.10. The user can choose up to four channels simultaneously manually or by using predefined sets of parameters to offer shortcuts for the user. The predefined sets are placed at the top left to be found fast, and then the result is presented at the right. If the user needs to manually pick any channels, this can be done in the roll-down menus placed under the predefined sets. If a predefined set is chosen, it can temporarily be changed in the roll down menus. Screenshots can be taken of the current graph view which is directly saved in a folder specified by the user. The placement of the screen capture button is at the bottom left as a result of starting with choosing the channels at the top left and viewing the result at the right to be changing the eye focus to the bottom left corner.

4.7.4 Configuration tab

This tab is more rarely used, and therefore the experienced user becomes a more intermittent user in relation to the functions presented here. This reduces the needs of short-cuts in the interface, but grouping and clear consistency is desirable. All different configuration functions are grouped and divided in a natural way. The level of error handling is increased in the PID-configuration to avoid accidents. The various configurations are placed to the right and are in some cases combined with hints and further explanations. The tab can be seen in Figure 4.11.

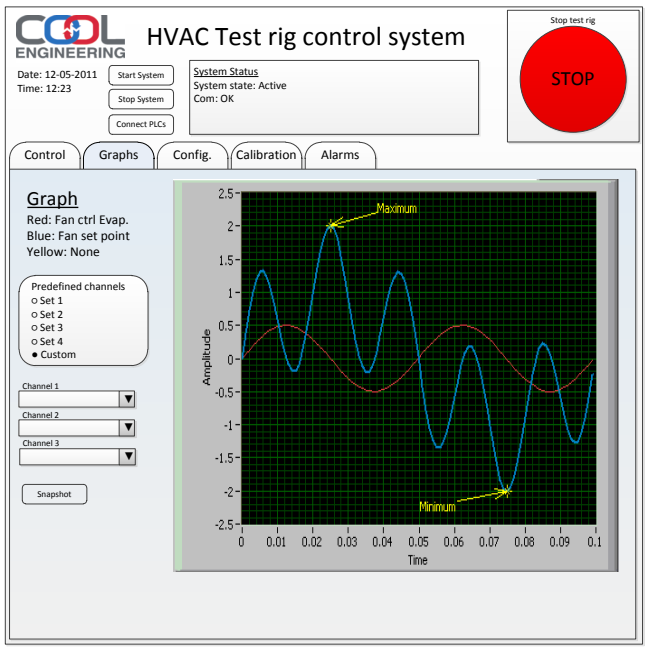


Figure 4.10: The graph view design

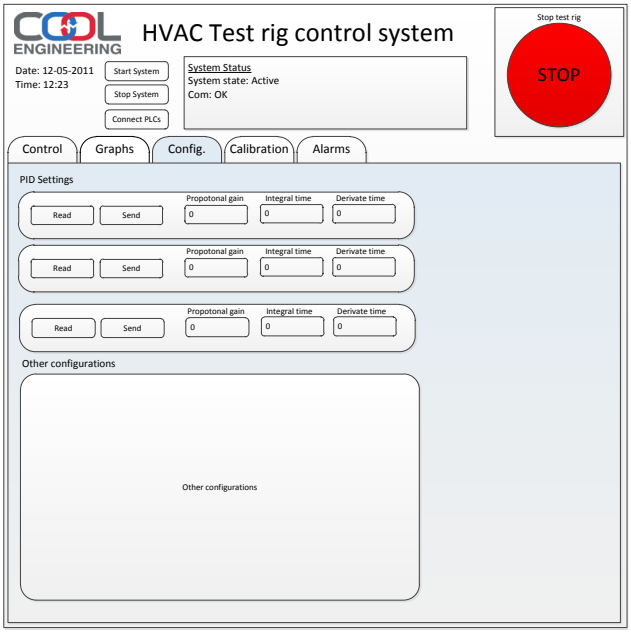


Figure 4.11: The configuration view design

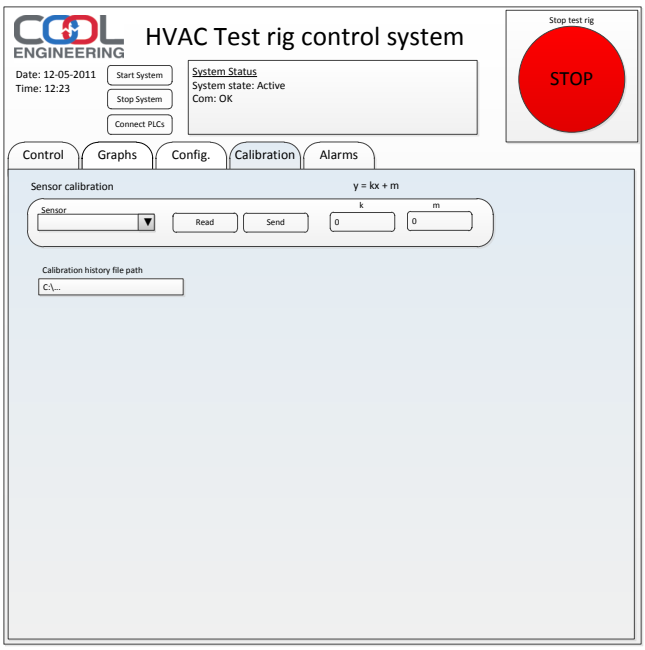


Figure 4.12: The calibration view design

4.7.5 Calibration tab

The calibration tab is seldom used, and therefore the characteristic of the GUI call for an interface that is more self-explanatory due to less experienced users, in the same manner as the configuration tab. As a result of this the error prevention functions are prioritized; e.g. the user cannot load any new configuration without reading in the old settings from the current PLC. This is to avoid the possibility that the user will load severely wrong data in to the PLC. The data of the calibrations are automatically saved in a text file which is chosen in the calibration tab interface. The calibration tab can be seen in Figure 4.12.

4.7.6 Alarm tab

The last tab is normally not used in the daily work, but definitely needs to be included the GUI. The user should be given the possibility to configure two customized alarms, and also to (de)activate the pressure guard function. To ease the configuration of the customized alarms, the left-to-right principle is used, starting with picking the variable of interest, followed by what value it should be compared with and what message should be published. At the far right, the final decision to activate the alarm is made.

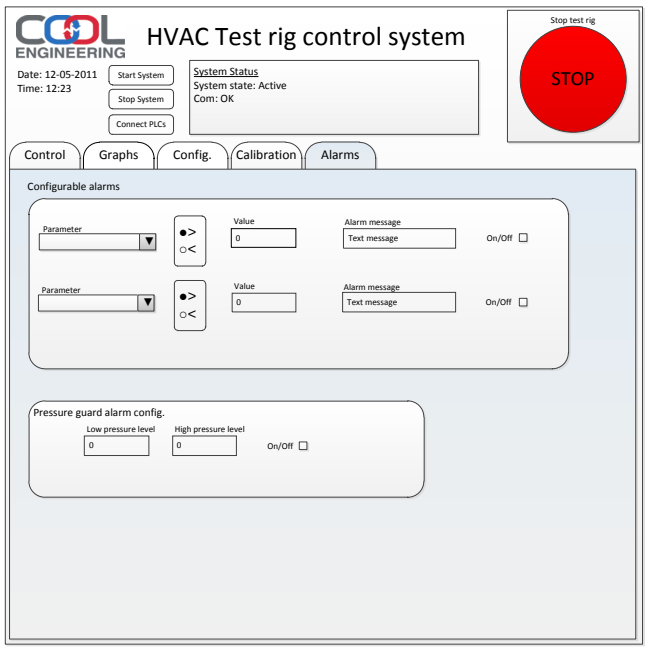


Figure 4.13: The alarm view design

In the second section of the alarm tab, specific predefined alarms can be configured; in this case there is only a need to configure the pressure guard of the HVAC system. The design of the tab can be seen in Figure 4.13.

Chapter 5

System Implementation

This chapter presents how the software was implemented based on the stated requirements and software architecture. A validation of the design guidelines and the integration of the systems will also be found in this chapter.

5.1 Programming the GUI control system

The coding of all the software was an agile development phase wrapped in the implementation step of the waterfall model, using the output from the pre-study and resulting in software that was ready for validation.

The implementation of the GUI control system can be described as two parts, the programming of the interface and the development of the background control code. The interface followed in the requirements, especially the information model and the GUI specification. The interface was based on two different window frames with one top view representing the "main view" and then a tab control showing different windows of control and configuration. See Figure 5.1. The background code is based on four different Producer/Consumer loops, handling events produced in the system. These four event loops are divided according to the information model: the GUI, Send data, Receive data, and safety functions loops. By using these four loops, the system can process four parallel tasks simultaneously without slowing down the system. It also uses a number of external subroutines which handle the complex data-sharing within the system. See Figure 5.2 for an overview of the G-code.

5.1.1 GUI loop

The GUI loop manage a high number of different cases which are directly or indirectly associated with the GUI. Mainly these events catch up inputs from the user and update the screen with data from the underlying PLC systems. The data from the underlying

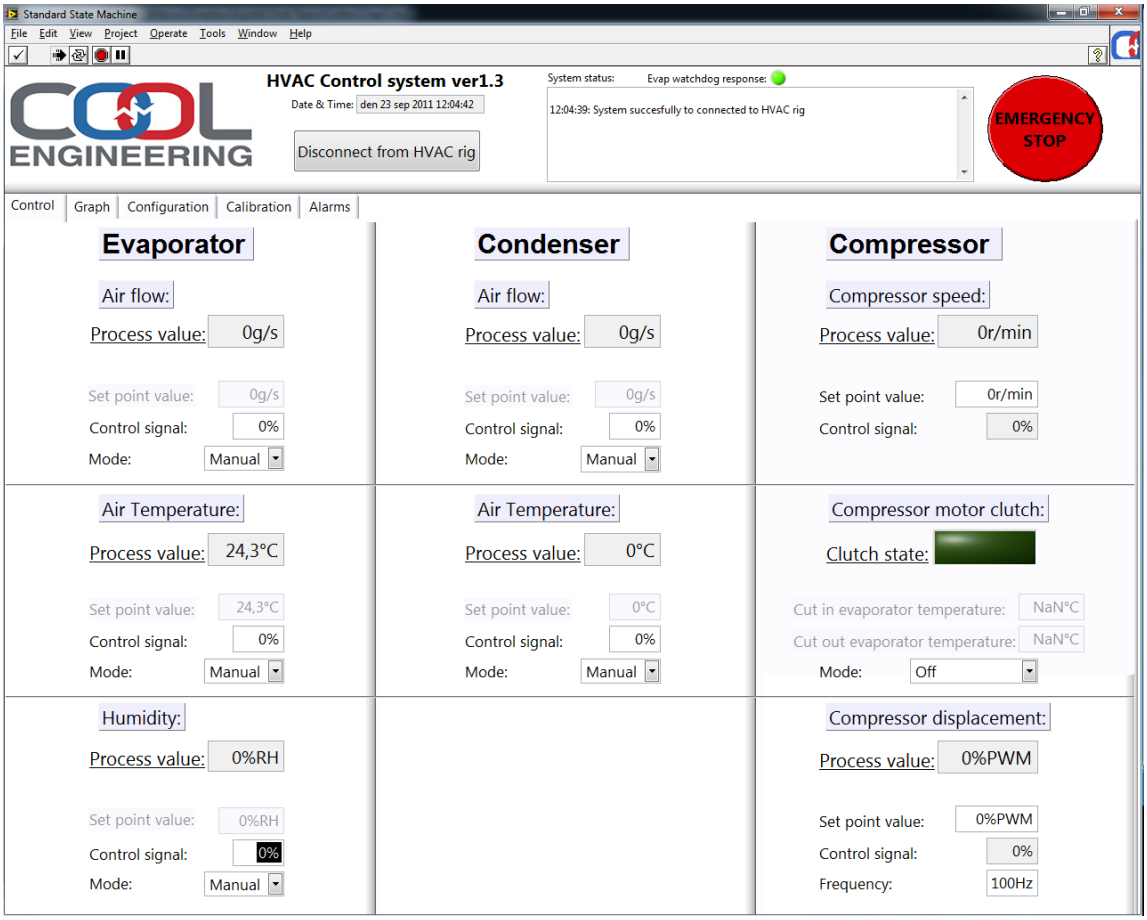


Figure 5.1: The main view on the top and the bottom frame with the tab control.

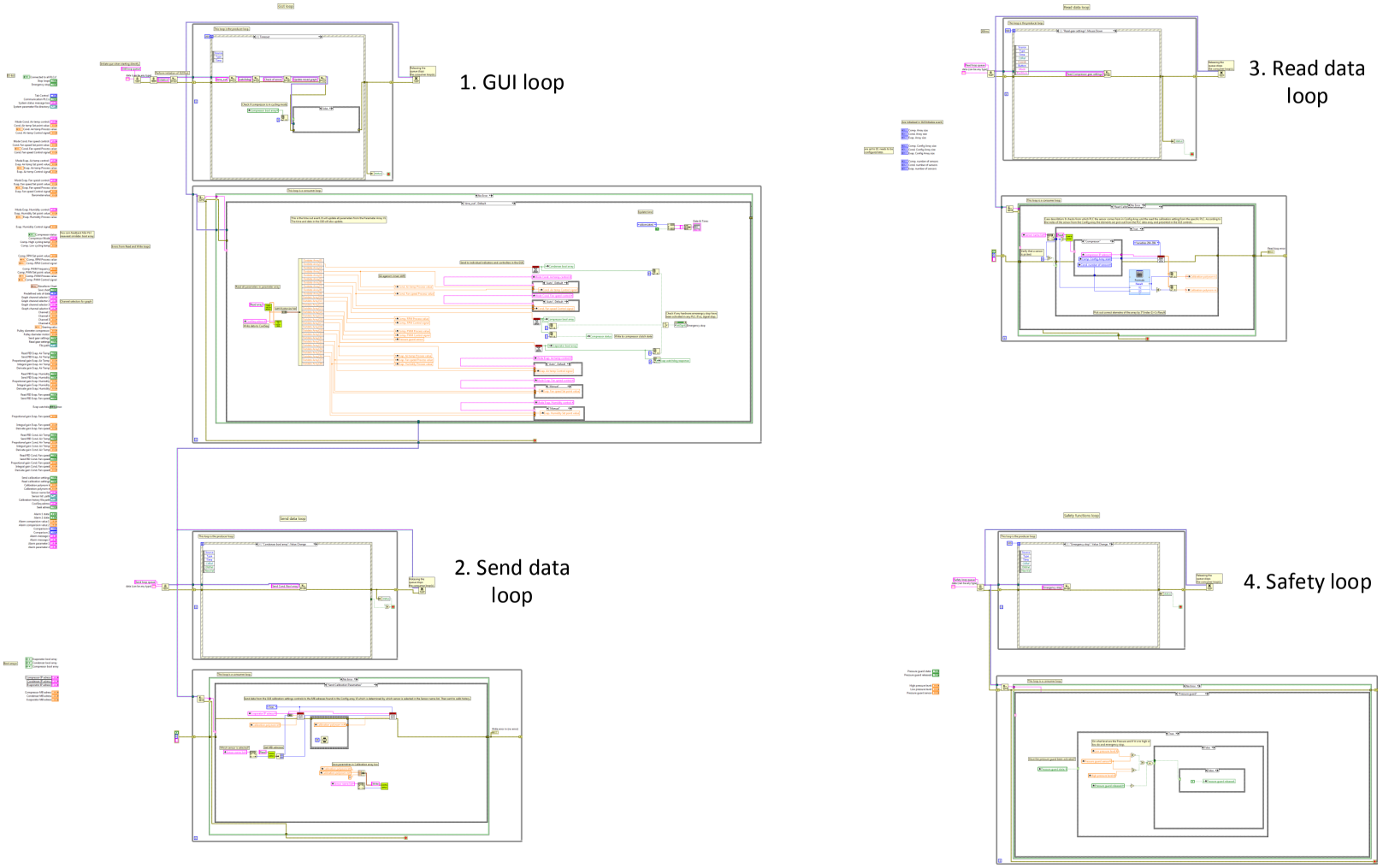


Figure 5.2: Overview of the LabVIEW code for GUI control system with the four producer/consumer loops and the variables placed outside these.

systems that are received from the Read data loop, saved in a common parameter list and then presented in the GUI by a timeout event with a frequency of 20 Hz. When input from the user is acknowledged by the basic LabVIEW system, it triggers first an event in the GUI loop which then manually triggers the Send-data loop, which processes and sends it. Indirect functions are such as the initiation and stopping of the system which organizes data and sends it to the underlying PLCs.

5.1.2 Send-data loop

Actions performed in this loop are directly connected to a writing of data or connecting to the PLCs. The events are activated in three different ways, either by manually triggering from the GUI loop, by value changes of specific parameters, or by send buttons. The reason for using different types of event triggers is related to the appropriate level of automation; sensitive values for the PLC configurations must be actively sent by the user, by pressing a send button. Changes in operating values such as temperature levels etc. are automatically updated by the system when the user is finished editing the value. It must also be mentioned that this loop does not use any time-out event to perform any actions.

5.1.3 Read-data loop

This loop frequently reads all data from the PLCs in an event triggered by a time counter to serve the GUI loop with updated data with a frequency of 5 Hz. The reason for using only 5 Hz is to avoid overloading the network communication. The loop also reads specific configuration settings from the PLCs on demand from the user by pressing the read-button; see Figure 5.9 and for read calibration see Figure 5.10.

5.1.4 Safety loop

The safety loop supervises the incoming data and compares the critical values with allowed levels. It also controls the software emergency stop button in the GUI; if it is activated it will send data to the loops to stop and also respond to the user of the result. The loop only manages a low number of cases to ensure that the risk of queued events is minimized, to maintain a low response time. The loop will actively take control of the system and stop it in a case of emergency. Current functions are handled in this loop:

- Emergency stop
- Pressure guard
- RPM guard

- Configurable alarms

5.1.5 External subroutines

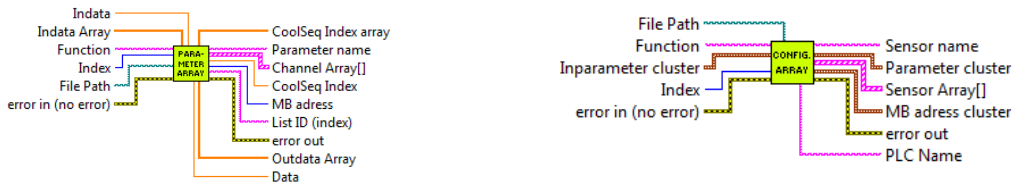
A number of tasks performed in the consumer loops use external subroutines specifically made for this software. These handle different types of sharing or saving data both externally and internally in the software.

Parameter array.vi

This subroutine handles all the data from all control and indicator parameters in the system; it is possible to read and write data to different parameters and to read all the parameters' names from it. The motive behind this subroutine is that several loops need to access common data simultaneously. Each parameter has been assigned a name, a Modbus address, and an index number saved in a specific Excel file. All these data and the temporarily stored current value of the parameter can be accessed in this subVI. This Excel file can then be easily be updated if a parameter is added or removed from the system, and can be accessed without any need to access the LabVIEW code.

Config array.vi

The config array subroutine shares almost the same function as the parameter array subVI, with the difference that it handles the parameters connected to the systems configuration such as PID-settings, calibration parameters etc. This routine is not so often called by the system, because the settings of the system are rarely changed. All the information connected to the parameters is also saved in a separate Excel file.



(a) (b)

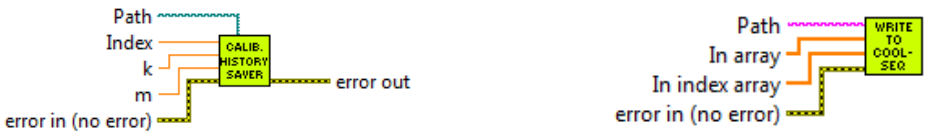
Figure 5.3: a) SubVI Parameter array b) SubVI Config array, with their symbol and connections that are used in the LabVIEW code.

Calibration history saver.vi

This subroutine is called in case of an update of the calibration parameter in the system. The subroutine writes the new configuration values together with the current date and time to an Excel file, so that the calibration history can easily be accessed and viewed. The input to the subVI is only an index that can be accessed from the config array subVI, the calibration data, and the path to the Excel calibration history file. This sub routine is highly independent, so it can be reused in other LabVIEW programs.

Send to CoolSeq.vi

This subroutine acts as a communication bridge for sharing selected process and control values from the main control program to the CoolSeq data logging system. Selected data are acquired and sent via National Instruments' own data protocol Datasocket over the LAN to CoolSeq.



(a) (b)

Figure 5.4: a) SubVI Calibration history saver b) SubVI Send to CoolSeq, with their symbol and connections that are used in the LabVIEW code.

5.1.6 User interface

The interface was programmed according to the design established in the requirement specification and further refined. The screen can be divided into two areas; the main top view with graphical object that can always be seen, and the tab control containing different tabs views which will be presented later on.

Main view

The main view was refined during the validation step, but before this phase this frame contained start and stop buttons and a separate "connect to PLC" button. When pressing

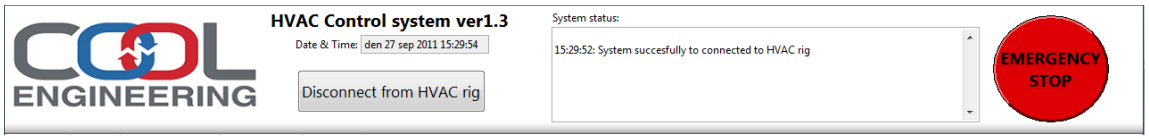


Figure 5.5: Main view of the graphical user interface.

start, the user started up the GUI so that the local settings and data could be initiated, but the control system did not interact with the test rig's PLCs. To operate the test rig, the user had to press the "connect to PLC" button and then manipulate the parameters in the interface. This was later simplified during the validation process that is described in Chapter 6. In the main view messages about the system status are directly presented in the system status text box. In every timeout event of the GUI loop errors that have occurred in any of the four loops are presented status text box. Otherwise the main view turned out to be as stated in the requirement specification, see Chapter 4.

Control view

The control view went through perhaps the most refinement during the development. The final version has a layout based on how the tasks are performed during the general testing situation. This was the final modification that was performed also during the evaluation process. Before entering the validation process, the layout was based on how the old panel was designed; see Figure 4.2. In the control tab, each main actuator within the test rig system had an assigned control module; see Figure 5.7. They were kept to one design in order to be as consistent as possible when applying them on different actuators. It contains three main components; the process value, the set point value, and the control signal. All modules except the compressor clutch control are based on this module design. All actuators with a PID control with the possibility to be turned off also have a roll-down list where it can be chosen whether the actuator should be controlled automatically or manually; this enables and disables the PID code in the PLCs. The process values font was increased in size to make it easier to read and all parameters' units were moved inside the text box to make the interface cleaner.

Graph view

The graph view in Figure 5.8 uses a waveform chart to present the data graphically, which is designed by National Instruments. It has the functionality that it is simple to change the scales in the diagram and the time period. In the waveform chart the user can right-click on the data line and change the appearance of the lines according

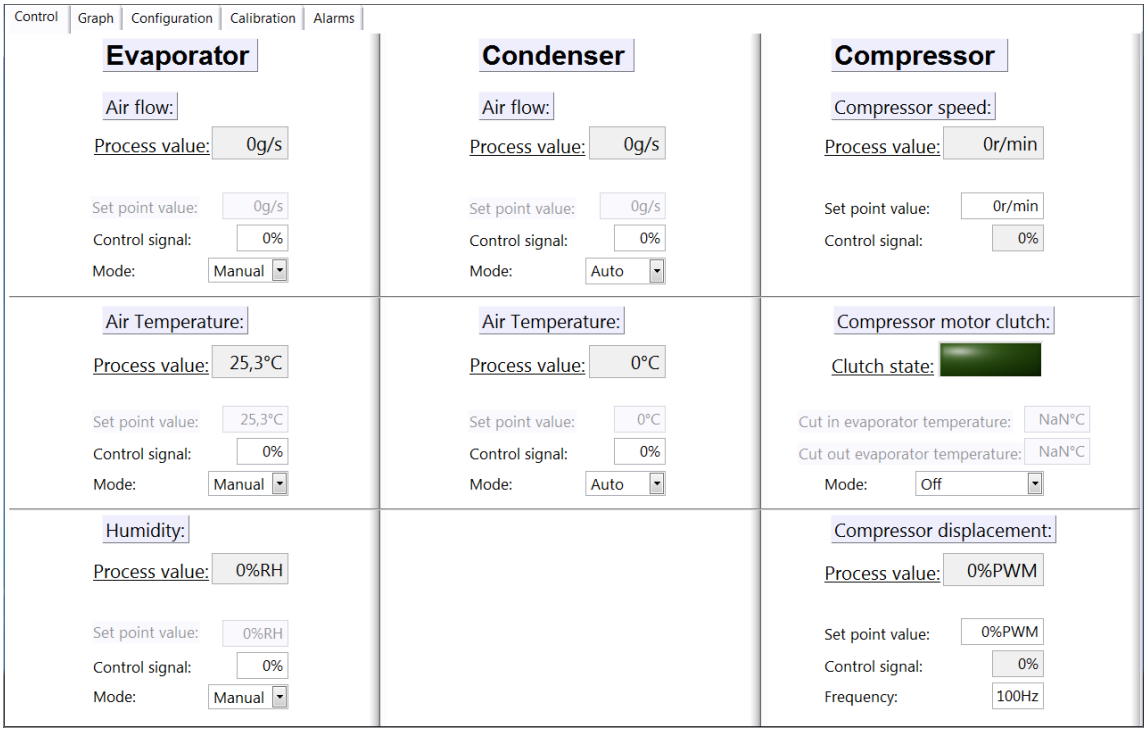


Figure 5.6: The control view of the actual GUI.

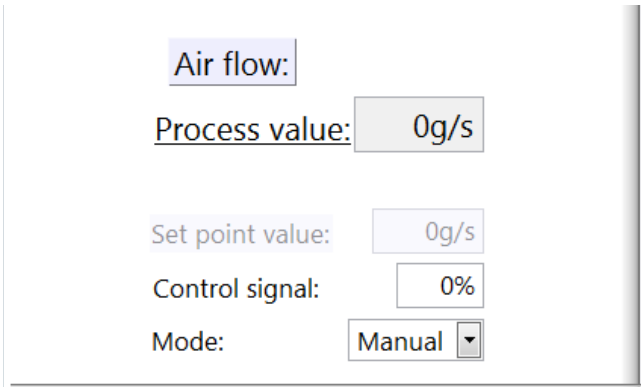


Figure 5.7: The control module of the airflow that is used in the control view.

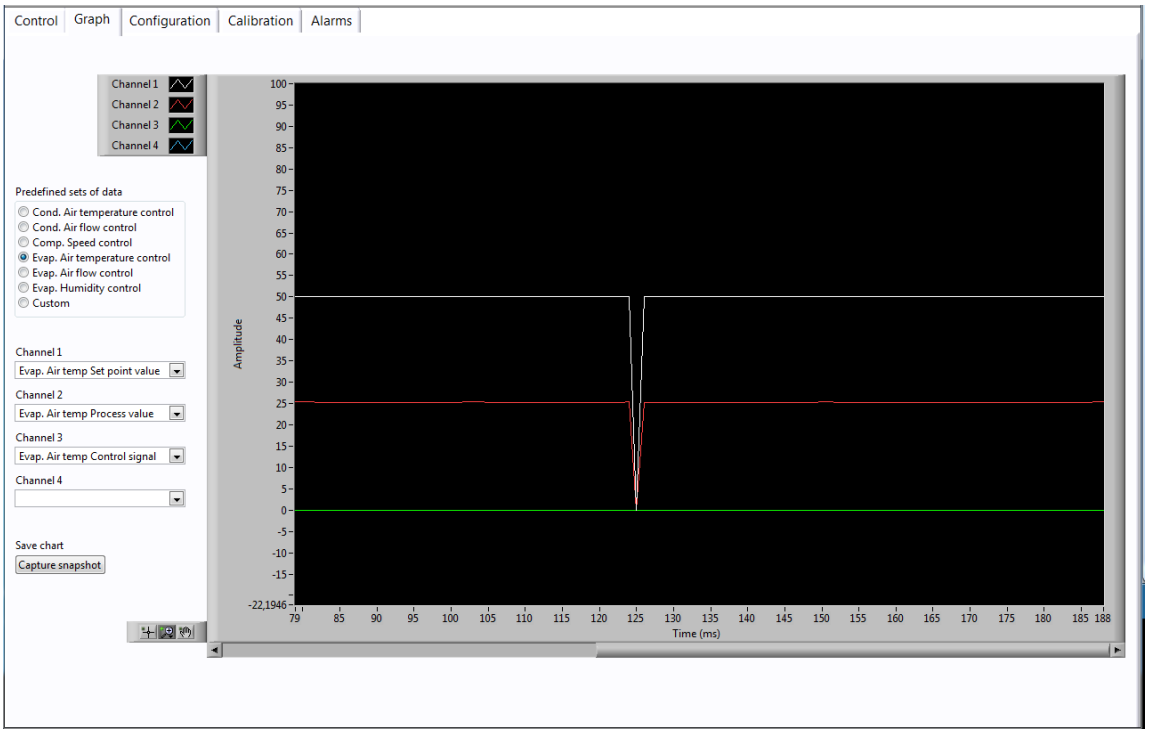


Figure 5.8: The graph view of the actual GUI.

to what the user prefers. The graph handles at most four channels simultaneously, as stated in the requirement specification. When the user saves a chart a save-file dialogue will prompt, so that the user can choose directory and file-name. The old functionality was that the user only pushed the save button and the picture was saved with a time stamp in a predefined folder that was set in the configuration tab.

Configuration view

The configuration view functions were grouped according to the specification. The terminology in this view was changed during the evaluation period to match better what the operators are familiar with. Incorrect values cannot be typed in to the different text boxes, which helps the user to avoid internal errors within the software. This contains a lot of functionality which can make the interface hard to overview when searching for a specific function. When changing any file directory or data socket address, the user will be guided by a dialogue window according to the operating system's standard. A barometer value is entered manually every time during the testing session; this func-

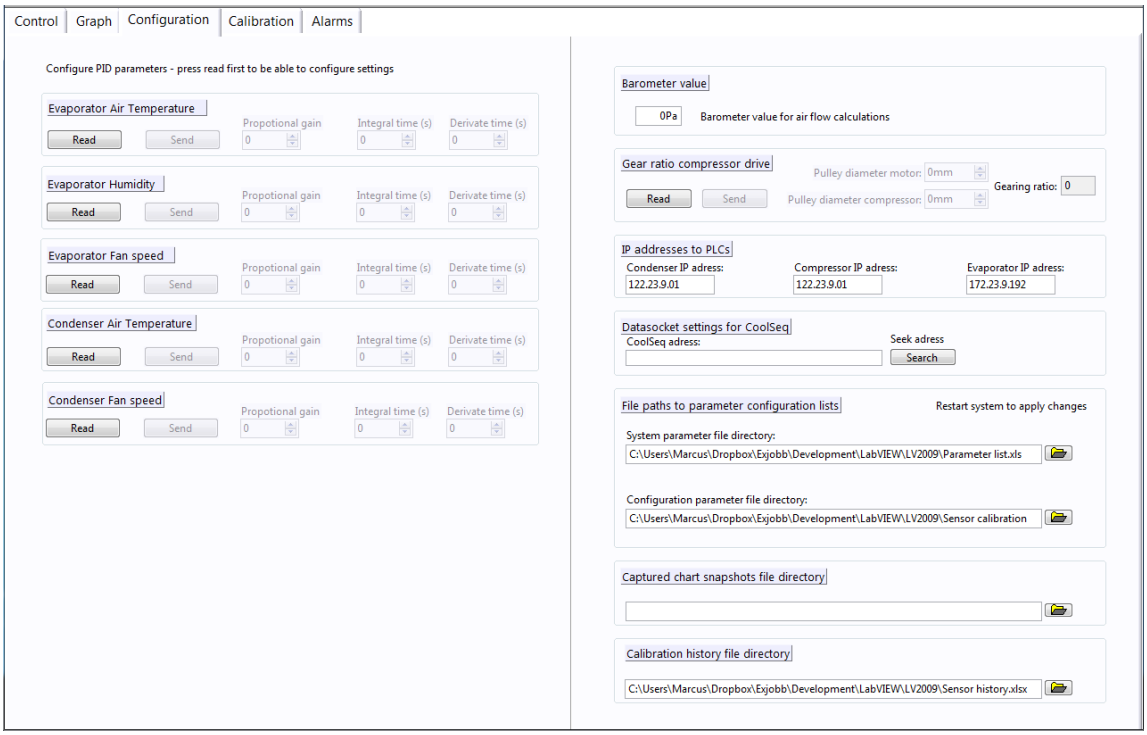


Figure 5.9: The configuration view of the actual GUI.

tion will be removed later in the development of the test rig's hardware, where this measurement will be replaced by a pressure sensor.

Calibration view

The implementation of the Calibration view (see Figure 5.10), is straight-forward in this part of the GUI. The calibration history file directory was moved to the configuration tab next to the other directories to increase consistency in the interface. All sensor names were changed from being very hard to understand in the old system to be more intuitive for the user when searching for one. The entire range of names is loaded from the config parameter subVI to ensure that it always uses updated metadata of the parameters. If a new parameter is added in the configuration Excel file connected to the configuration parameters, the GUI system will directly handle it without any configuration except a basic calibration. When a sensor is calibrated, it will trigger an event in the GUI loops which save the calibration settings into the history Excel file by using the calibration history saver subVI.

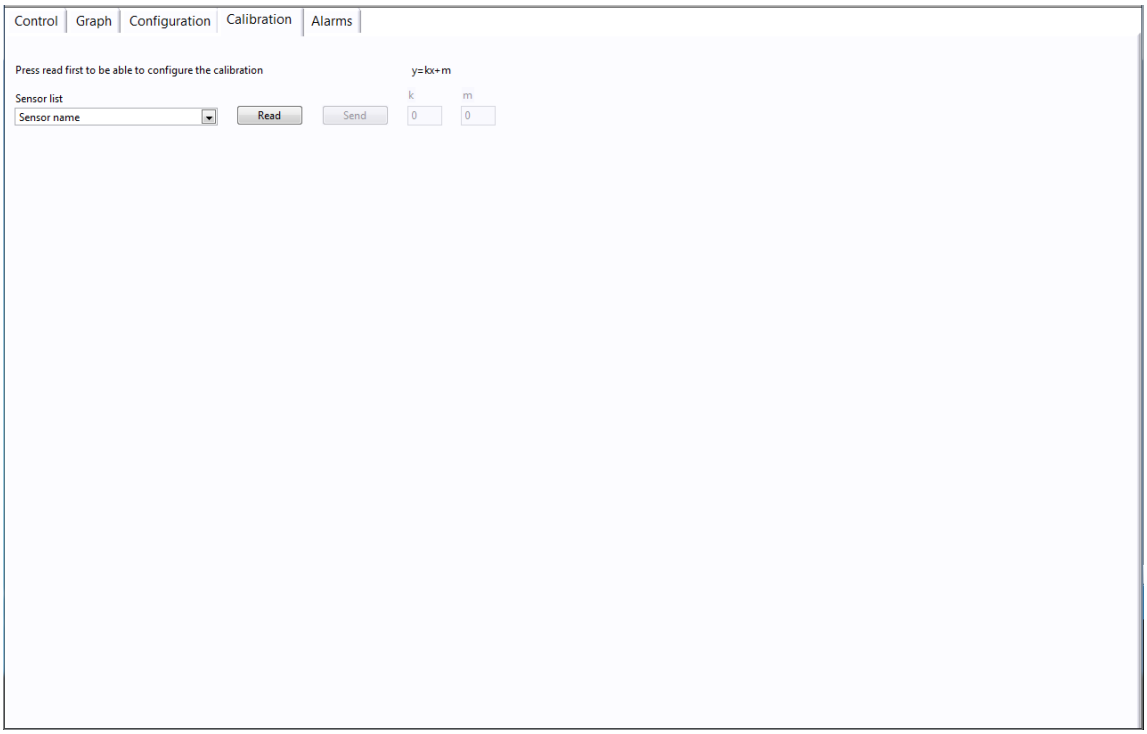


Figure 5.10: The calibration view of the actual GUI.

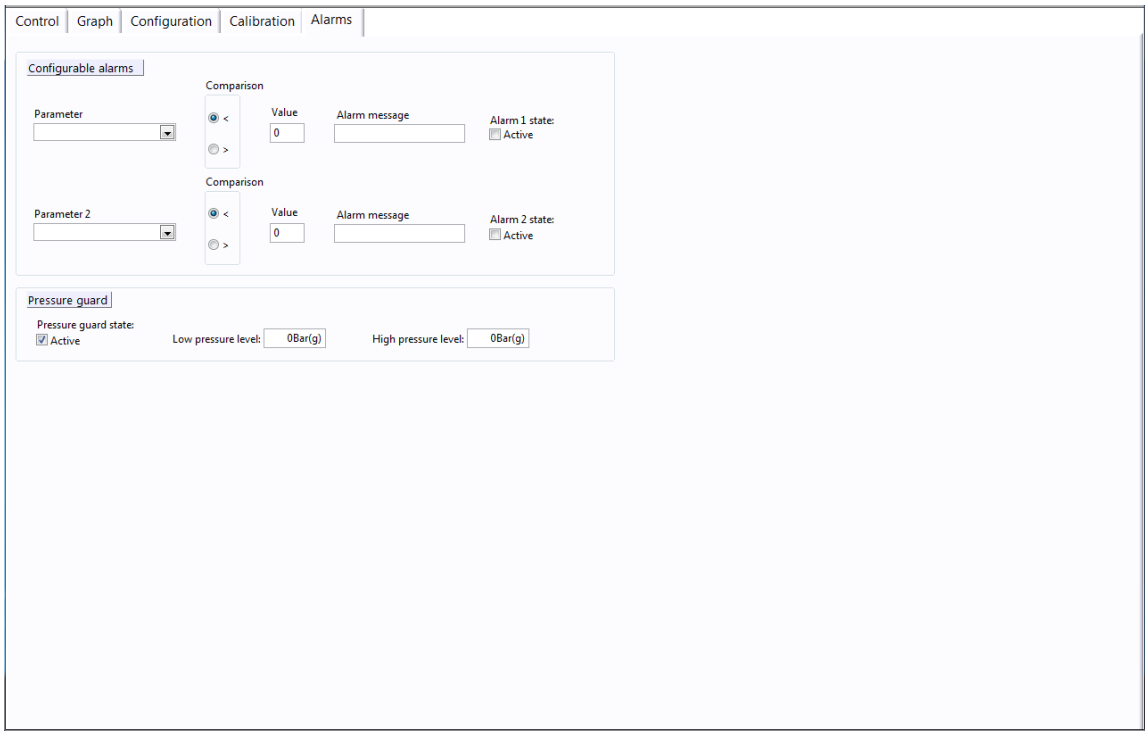


Figure 5.11: The alarm view of the actual GUI.

Alarm view

In the alarm view, two different types of alarms are configured; the background functionality in this section is run entirely by the safety loop to ensure fast response to critical situations. The configurable alarm uses parameters that are loaded from the parameter subVI for operational parameters, which means that all parameters used in the system can be selected and monitored. If an alarm exceeds the allowed level the user will get a popup message that the alarm triggered and that it will be deactivated until it is manually activated again. The reason for deactivating the alarm is to avoid letting the alarm trigger repeatedly with a short time interval, making the interface unusable. The second type of alarm is the predefined alarms such as the pressure guard, this type will compared with the configurable alarm execute code to handle the critical situation. If the pressure guard sees that the pressure exceeds the maximum allowed level, it will stop the system by sending a control signal with a zero value to all actuators. There is also another predefined alarm that monitors the compressor's RPM.

5.2 Heuristic evaluation of GUI

Based on the theory presented by Galitz,a review of the LabVIEW GUI was made. [14] The evaluator was the author of the thesis. The validation of the interface started with the first heuristics of design, going through all the different views systematically. It was used as an initial validation of the GUI before starting the upcoming think-aloud test with various stakeholders. By starting with this, the intention was to eliminate the flaws in the GUI and ensure that all the guidelines had been followed.

5.2.1 Heuristic evaluation of GUI result

The result from the evaluation is presented in a natural language according to each of the design heuristics. Some repetition might occur in the result from each heuristic evaluation, but it can also be seen that it is hard to make a clear distinction between some of these guidelines.

Visibility of system status

The graphical interface has a clock that continuously updates every second when the system is started. The system status text box indicates that the system is up and running by posting messages that it is connected to the system. The connect push button has also changed text from "Connect" to "Disconnect". In the control view the numbers from the process values update if the system is connected to the test rig's PLCs. If the system loses connection, the internal error message handler will present a connection error message in the system status text box; the system can also present other error messages. Looking in the graph tab, it updates at a high frequency when any channel is chosen. If the control system is connected to the test rig it will also continuously update the process values in the control tab. Thus it is possible to see if the system is in operating mode and if it is working correctly.

Match between system and real world

When this evaluation was made, the layout of the control view was built up according to the old placement of the controls on the old control panel. The terms used for these control modules in the layout are familiar, following the industry standards as far as possible. In each control module the place of all graphical objects try to have a logical order based on the left-to-right and top-down design principles. The units used in the old system have been reused in the new control system to enhance the feeling of using the same system as the one before this project. Feedback from inputs, e.g. noise, from the real system is picked up since the operator is placed in the control room with a close overview of the test rig.

User control and freedom

The system has a basic stop and an emergency stop function, so if necessary it is simple for the user to make a controlled stop. To reach full user freedom it should be possible to undo most of the actions that have been made in the interface. When doing one or a sequence of actions, it is not supported to undo these directly by pressing an undo button, but all actions can in most cases be redone manually.

Consistency and standards

The measurement units and the technical terminology follow the same standard throughout the whole control system. Related or similar functions use the same layout in order to be able to recognize and avoid user errors, especially in the control modules. Also the calibration and PID configurations and compressor pulley settings use the same layout and controls to read and write data to the PLCs. Because of the critical data, the user has to go through a more difficult reading/writing process compared with being able to write any value to the system. The so-called control modules in the control tab try to use the same layout so as to create a more reliable and consistent system.

Error prevention

A large range of error prevention functions are built into the basic LabVIEW system which have been implemented in various places in the interface. Identified as a main potential error source are typing mistakes and values in the control view. The user cannot enter any obviously invalid values, e.g. entering 103% to a control signal, as the system will either correct the value to the closest valid value or ignore the entered value and keeps the old one. For critical values, such as calibration parameters and so on, the user must actively read the old values stored on the current PLC to the LabVIEW GUI before it can be changed. This will prevent the user, from instead of typing a new value from the beginning, always editing the old value.

Recognize rather than recall

Short instructions are placed in strategic places as a complement to the labels of the graphical objects. The user does not have to try to remember how to use rarely used functions. Most of these strategic places are functions that might cause problems or where it may be hard to understand the reasoning behind them. But it should be mentioned that the configurable alarm function lacks instructions due to a potentially non-intuitive design.
The user can easily retrieve old configurations used both in the PID configuration and in the calibration configuration. For less safety-critical configurations, the current used

value is always presented in the interface and can be edited directly, so the user will not have to remember to the old configuration before editing in the control view when changing from automatically mode to manual mode in a control module; the interface also remembers the current value of the set point or control signal, depending on the current mode.
Problems that can be found in the interface, in relation to the start-up process of the test rig are remembering to type in settings for current air pressure and so on.

Flexibility and efficiency of use

For the experienced user keyboard tabbing between the different views in the tab control frame is missing, as also between values that are typed in a sequence. But the order of all tab names is according to which views are used most commonly, which is placed to the far left. This definitely speeds up the process of controlling and monitoring the interface, by having the tab names placed in this order.
It can be seen in the graph tab that a shortcut for predefined sets of channels is applied, making it easier for the user to select all channels related to one control module, which means; set point, process value, and control signal.

Aesthetic and minimalistic design

The control view only contains the most relevant information, trying to keep down the amount of information presented in that view and striving for a minimalistic design. By highlighting the process values and trying to reduce the focus on the set point and control values, the views should form a cleaner interface. Different categories of functions are divided into different views; some of these views do not hold much of functionality or information, but it helps the interface to be more divided according to frequency of use and function, resulting in a clean interface. This makes it easier for the user to locate functions and information faster without affecting aesthetics negatively. Less frequently occurring information concerning the system is presented in the system status text box, to keep the user informed about the system but not overwhelming the rest of the interface with information. This avoids confusion and distraction when the user is performing a task. But the division of the main top frame and the tab control frame makes it always possible to see the system status text box.

Help user recognize, diagnose, and recover from errors

The user interface manages and presents error messages in the system status text box partly by using LabVIEWs built-in error message system. The error is presented in plain text together with an error number to identify the error. With this number the user can read further about the error on National instruments' homepage if it is a standard error.

Most of the errors are presented with a possible cause of the error helping the user to recover from the situation. Errors that are specific for this control system, which have been programmed during the project, are also presented in the system status text box. In some specific cases automatic recovery is initiated when an error occurs; the most significant example is the connect recovery process which tries to recover the connection between the LabVIEW system and the test rig's PLCs.

Help and documentation

The current system does not support any help documentation; instead it uses tool tips which emerge when hovering the mouse pointer over a graphical object. Small instructions are also placed at various strategic locations in the interface that are visible all the time, e.g. in the calibration and configuration view.

5.3 Programming PLCs

Only one of the PLCs was programmed in this thesis, due to limited time and the small difference of code between the units. The PLC to be programmed was the unit steering the Evaporator tunnel. This evaporator PLC is the most complicated unit among all the three PLCs and was therefore chosen to be implemented. The main motive was to develop a generic PLC software that could with simple modification be implemented on all three PLCs within the test rig system.

5.3.1 Main loop

As input for how the main loop should be coded the flowchart developed in the software architecture (see Appendix F), was used. This main code is looped every 20 milliseconds, and is built up around a case structure of two states. The first state, named state 0 in the code, is the initial state where the PLC waits for getting clearance from the LabVIEW GUI system to start up and enter the operating mode. Briefly explained, the PLC reads the Boolean operators from the external LabVIEW system, containing the start indicator, and if the system is to start it reads the initial values for the parameters and changes the state in the case structure to enter operating mode. The loop sequence will never return to the initial state again until the system is ordered from LabVIEW GUI to stop.

In the operational mode, named state 10 in the code, the loop starts with reading all external analogue and digital input from the sensors and the Boolean arrays from LabVIEW. The system then updates from all public configuration parameters accessed externally by LabVIEW to the residential memory space of the PLC, so that it can be accessed after the PLC has been shut down one or more times. The next step is to read

the PID settings from the residential memory and then perform the safety functions, which are one watchdog function and the function which checks if the emergency stop is pressed down. After this, the system runs the controller functions, which from the input data, will calculate a response which is used in the output section. The output code will write to the Boolean array and the analogue and digital ports on the PLC rack, setting voltages or current levels to the controlled actuators.

The last task in the operating state is to check if a normal stop has been ordered by the LabVIEW GUI system; if so, the system will reset all variables to the actuators and change from operating to initial state, waiting to start again. The code can be found in Appendix G.

Reading inputs functions

The input section consists of three sub-functions, used in the main loop. All analogue signals connected to the PLC's analogue input bus can be read on different software addresses defined together with all the input parameters. The raw data from the analogue input bus have a resolution of 32767 and are not calibrated values. Therefore the first step is to add the calibration parameters to the raw data and then divide everything by the resolution to get the actual value from the sensors e.g. temperatures, RPMs and so on. The input from the digital signals needs less processing than analogue; if it is a numerical value, only the calibration parameters are added, and if it is a Boolean input no processing needs to be done. The last input sub-function is the Boolean array reader; this function converts a decimal numerical value to a binary numerical value which is written to different Boolean variables. The reason for using a Boolean array is to use only one Modbus address instead of one for each individual for all Boolean variables. The Boolean array is used in the communication between the LabVIEW GUI and the PLC.

Update data functions

This group of functions performs the very simple task of updating values to and from the residential values. The sub-function Update to Resident Data saves the current configuration of the PID setting and all calibration values that can be accessed after the PLC has been shut down. All configuration variables that are updated via the LabVIEW GUI are saved and accessed via the residential memory to ensure a robust system. In the sub-function PID setting, the function accesses all the PID settings in the residential memory and changes all values for the time settings from seconds to milliseconds, and then converts the data type from an integer to the specific time data type. The reason for the time conversion is that the PID functions in TwinCAT need this format on the setting input, but in the GUI seconds are preferred.

Safety functions

The most critical functions of the PLC are found in this section. To check if the emergency button connected to the PLC have been pressed down, the function reads that digital input and then responds if necessary to call for the sub-function "Reset variable after emergency stop", which basically writes directly to all actuators to stop as fast as possible. The second safety function is the watchdog, which has the task of checking if the LabVIEW GUI system is active, as it may either can be hung or have lost communication with the specific PLC. The watchdog sends a Boolean signal and the GUI should respond to that by sending another Boolean signal within a certain time. If the LabVIEW system does not succeed in that, the watchdog will also call the "Reset variable after emergency stop" function to shut down the system until it gets contact with the LabVIEW system and is started again.

Control system functions

This section can be seen as the main part of the operating state in the main code; here all the control functions are called which will respond to the input of the system. In the control function, the function will first call for advanced calculation sub-functions, e.g. calculation of airflow through an orifice if needed, and then call the specific PID function if the actuator is to operate in automatic mode or, in case of manual write directly to the actuator's control signal. The control function can also contain other specific functions, and the reason for this control function is to collect all the sub-functions for one actuator in one place. This part is where the major changes are made when adapting the code for different systems to be steered.

Writing output section

The output section has a lot in common with the input section; see 5.3.1. It also has the same disposition of functions for the Boolean array, analogue and digital output. In the analogue output function, all control signals that are created either manually by the user or by the PID functions are values between 0 and 100.These numbers are divided by 100 and multiplied by the resolution of 32767 to create the raw output signal, which can represent either a voltage or current level. All outputs for the evaporator's actuators are analogue. In the digital output section, only one output is defined for the evaporator unit; it is the signal that "drives" the emergency stop, which is always set to a "true" value. Finally, the output section also has the Boolean array writer functions, which add together all binary values and convert to a decimal value that can be published for the LabVIEW GUI system via one Modbus address.

5.4 Integration verification

As a stage gate in the development process, an integration test was performed when the first full version of the LabVIEW GUI system was made together with the first PLC software for the Evaporator PLC. The integration can be described as a mixture between top-down and bottom-up integration with weight on top-down integration. The verification process was performed in a less formal way due to the incremental development and testing approach, but the outline of the sequence was starting with the basic communication between TwinCAT and LabVIEW over the Modbus interface. A second step was to verify that the analogue and digital in- and output ports were functioning correctly in relation to the software. In both steps more functions were added in the software for debugging as the testing proceeded. The result from the verification was that the LabVIEW GUI system needed some modifications to make the communication function correctly.

Chapter 6

System evaluation

The following chapter presents how the system evaluation was performed together with the results.

6.1 Think-aloud GUI evaluation

After the heuristic evaluation was made a second GUI design validation was made based on the think-aloud method. This validation is presented in the next sections.

6.1.1 Test plan

As a part of evaluating the software, a GUI test plan was created to get the most out of the evaluation sessions. The test was performed in a late stage of the development process. The format of the testing process is based on Galitz recommendations. [14]

6.1.2 Scope

The test was completed in a late phase of the development, bringing the possibility to perform the test on the actual software. The time plan for the evaluation was one to two days. The GUI of the tested software cannot be considered to be novel to any high degree which will influence the evaluation in that the outline of the test will look standard and confirmatory. The tested system can be considered as a relatively not safety-critical system with a low number of different users.

6.1.3 Purpose

The main target with the evaluation is to confirm that the GUI follows the user requirements of good GUI design. It should also serve as an indicator of whether the software

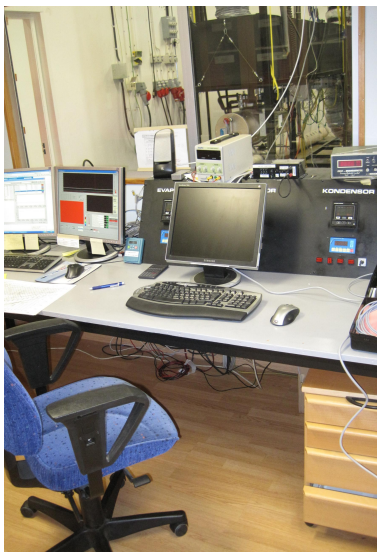


Figure 6.1: Test location with the computer setup to simulate the future control system.

fulfils the requirement specification. To some extent the purpose of the evaluation is to identify flaws and problems in the design even though, in this late stage of the project, the main motive is to confirm rather than explore the GUI. The performance goal of the testing is to evaluate mainly the use of the screen view regarding the general screen layout, navigation, functionality identification etc. The result is to serve as a fact basis for the conclusion on whether the system fulfils its purpose and requirements.

6.1.4 Test location

The evaluation will take place in the actual control room for the HVAC system at Cool Engineering using the correct monitor size. The test location can be seen in picture 6.1.

6.1.5 Scenarios

Due to limited time only the most important scenarios were tested. The functionality performance testing was included as a secondary objective. The scenarios in the test were:

1. Start-up and set-up of a new system, gear settings etc. The compressor wheel diameter is 100 mm and the wheel diameter on the motor is 85 mm. Configure barometer value to current value of 101300 Pa. Compressor PWM frequency

50 Hz. The pressure guard alarm should trigger between minimum 2 Bar and maximum 25 Bar.

2. Starting the system and trying to reach steady state with this set of stated parameters. The parameters should be as follows. For the Evaporator tunnel: Air temperature Auto mode 30°C, Air flow Auto mode 300 g/s, Humidity Auto mode 50%RH. Condenser: Air temperature Auto mode 40 °C, Air flow manual mode 45% control signal. Compressor: Clutch cycling mode between 20 °C and 30 °C. Speed control 300 RPM. Compressor displacement 30% PWM.

3. Do an emergency stop.

4. Restart system after emergency stop - Return to the state in task 2 after doing an emergency stop.

5. Configure PID settings - Adjust Evaporator Fan Speed control to P=1,2 I=3 D=2.

6. Configure and active the configurable alarms - Activate alarm 1 with the Evaporator temperature control value; it should alarm if over 45% and send out the alarm message "Control value over 45%."

7. Configure calibration settings - Evaporator temperature sensor with the addition of a positive offset of 30 °C.

8. Change history file of calibration settings - Change to the Excel file "new calibration history" placed on the desktop.

9. Change path of graph snapshot and then take a snapshot of a graph - Change to save the files in the "new snapshots" folder on the desktop.

10. Change IP settings for one PLC unit and reconnect to it - use the IP-address 192.3.10.192.

6.1.6 Closing interview

The testing session was followed up with a short pause and then a short interview covering the user's experience of the system that the evaluator just faced. The questions were:

- Which task was the hardest task to perform?
- How do you experience the overall impression of the system?
- Could you relate the physical process of the test rig with the control system?

- Was it easy to understand how to perform the tasks?
- How was it to read the screen from a distance?
- Was the screen layout helpful?
- Was relevant information presented?
- Did the system respond well to input?
- Did the system seem to be reliable?

6.1.7 Results

Three evaluations were made which resulted in an incremental development of the graphical user interface. After each evaluation session of the GUI, the most obvious and simple flaws was removed to be able to gain as much as possible from the next evaluation, bringing new improvements into focus. This explains why some of the initial issues are not recognized by the evaluators in the following sessions.

First evaluation, Test engineer 1

The first evaluator, Petter, can be described as a stakeholder from the test engineer category, with long experience of the specific test rig and the physics behind HVAC processes.

Before starting, the evaluator had a few minutes to click around in the GUI to familiarize himself with its appearance. The whole testing process was successfully recorded and lasted for one hour including the closing interview.

In the first task, how to start up the system was not clear for the evaluator, forgetting to press first "Start" and then "Connect" to get the system in an operating mode. See Figure 6.2. The evaluator got some advice on how to finish the task from the session leader.

Most of the tasks were performed without any significant problems; the evaluator in this test had difficulties in finding some of the control objects, even though the subject had a good understanding of what actions to perform to finish the tasks. The problems occurred when the user had to identify certain objects that were located in different views, during tasks involving configurations that usually are rarely made. The evaluator spontaneously pointed out certain labels using terminology that was inappropriate or could be confusing. But in line with the stakeholder description of the subject, he could still understand what the actual meaning was and handle the situation correctly. These suggestions of change were so obvious that they were changed before the next GUI test to avoid having the evaluator focus on old flaws and not detect new ones. They were:

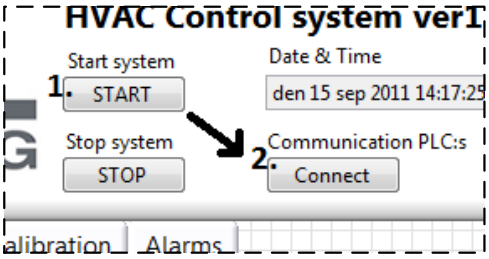


Figure 6.2: Cut out view of the startup buttons on the GUI.

- Terminology used in cycling mode of compressor
- Text indicating if an alarm is active or not
- Terminology used in the gear ratio of compressor pulleys
- Numbers of precision in value indicators and controls in various places in the GUI was adjusted

In the closing interview of the evaluation, the evaluator thought that the connection between the physical process and the GUI was easy to understand; but as mentioned before in the development process, the need for a "process picture" was addressed. This process picture would act as a complement to the control tab for monitoring of the process. In the closing interview, the test situation was discussed regarding how the layout of the control tab was and how it looked from a normal distance. Fast navigation between the values was a problem, but onces the information was found it was easy to read. The evaluator thought of using a bigger font on the process values, which are the most interesting current data when monitoring the system during testing.

When navigating between the tabs it was requested to make the labelling bigger than the original, which also would help navigating fast and from a distance./par

The order of the control modules' layout in the control screen did not feel natural for the evaluator, which conflicts with the old layout used on the old control system. In the testing process the user starts with activating the airflow and then starts steering the air temperature; the background reason for this working order is the risk of overheating the heaters if no air flows through. In the interface at this stage the temperature control module is placed on the top and then the air flow control comes as the second module below.

But the overall impression from the evaluator of the GUI was positive; the placement of the controls and functions was logical and clear. No important functionality was missing and the user felt that he was in charge of the system and got a robust feeling from it. The system responded well to input even though it was not fully connected to the test rig, creating no data from the sensors. He did not experience it as hard to learn the GUI.

The evaluation gave a good input for especially fast and easy changes that was made before continuing with the next evaluation sessions. It also gave an initial picture of whether the overall software had fulfilled the basic parts of the requirement specification.

Second evaluation, Maintenance engineer

The background of this evaluator, Erik, is a certified LabVIEW developer with expert knowledge of the LabVIEW coding language. Erik qualifies as a maintenance engineer in the stakeholder profiles.

Before starting the actual test, the evaluator was informed about the think-aloud method and got a minute to explore what the GUI contains. The evaluation was finished successfully and lasted 35 minutes, which can be compared with the first test that lasted an hour.

When the test started, the evaluator forgot to start up the GUI and connect to the PLCs before configuring the system with diameter of the compressor pulleys and so on. The evaluator had to stop and redo the first task before being able to continue. It was not intuitive that these initiation buttons needed to be pressed and what sequence it should be done in. From this emerged a discussion of whether the number of start-up buttons could be reduced so that one "connect to PLCs" was enough. /par

As in the first evaluation, new recommendations of small adjustments in the GUI were identified with a different character than in the first evaluation with a test engineer. It was clear that the evaluator had a deep knowledge of the LabVIEW software capabilities which could be better implemented in the HVAC control software.

The appearance of the control tab could be changed; the evaluator missed a trend indicator of the process value. This could be indicated perhaps by a curve or by colour-coding positive and negative trends. Also the arrangement inside the control module could be changed; the user would find that the set point value would be directly under the process value, on the third level the control signal value could be placed, and at the bottom the mode selector between manual or automatic control.

Moreover, to navigate on the grey background was hard when the other graphical objects had similar colours. This made it harder to find and focus on the correct information from a distance. Perhaps the theme of all the graphical objects shoulde be changed from the "System" theme to the new theme "Silver light" which was introduced

in LabVIEW 2011. The "Silver light" theme used buttons with stronger borders and more modern icons, which might enhance the usability of the software.

Some sort of process picture was missing for the software; it could keep a very simple level that could be used in monitoring the system, reducing the information presented to a low level with just the most relevant information, mainly process values. /par The picture of the process layout helps the user to find the needed information faster on the screen.

After performing an emergency stop, it would be appreciated if there were a function to automatically return to the old state of the parameters, because of the high number of settings that are lost when pressing the emergency stop.

The subject of the evaluation felt that the system acted reliably and responded well to input; with software expert background, he could see that the software was performing well within the aspects of communication and CPU usage. All tasks were still performed well except the first one, indicating that a novel user of the test rig could perform basic tasks, which often is the case when maintenance engineers are adjusting the test rig.

Larger adjustments that should be considered to be made before the first release, or in the further future development of the software, are to change from the "System" theme to the "Silver light" theme, add a process view tab, and reduce the number of steps in the upstart sequence for the user.

Third evaluation, Test engineer 2

The third and final evaluation was with Johan, who has a background as the main operator of the test rig. Johan has great experience of the automotive HVAC system, but also a strong knowledge of how the sensors and actuators of the test rig are built up and operate.

In the first task the upstart sequence was simplified just to start the VI, which always has to be done and then instead of pressing start system in the GUI, just to press Connect to HVAC rig. This simplification still did not work out without any problem. The evaluator did not read the task well and, in combination with inexperience of the system, forgot to start up the system and connect to the test rig before starting to configure the rest of the test. The rest of the test revealed no significant problems; the evaluator did not react to any of the changes that been made in the GUI after the previous test. Directly after finishing the first task, the test engineer pointed out that the frequency value was placed out in a way unnatural to him. He felt that it did not belong in the control tab, but should be moved to the configuration tab where it would make more sense. The value,to him, was more of a configuration value than a control value.

When performing the second task the user found the entire control module's placement on the screen logical and easy to understand and find; it was finished very fast with few spontaneous comments. On the third task, doing an emergency stop, the com-

ment from the evaluator was that he found the feedback from the GUI to be in sufficient that the emergency stop had been completed. Even though the operator of the HVAC rig would hear it shutting down, the confirmation to the user should be clarified. /par

When returning from the emergency stop state, the evaluator found it good that the user has to retype all the control values again. This ensures that the user avoids returning to a hazardous machine state. Other control systems have the function of returning to the previous state of the machine before the automatic emergency stop. During the interview, the evaluator was presented with the idea of rearranging the control view layout similarly to what the first evaluator, Petter, suggested in the first test. The control module's placement should be changed from following the layout of the old systems control panel to be more connected to how the general testing situation is performed. This means that the airflow control units would be placed on the top instead of the air temperature control, because setting the airflow is the first thing to do when starting a testing session. The evaluator found it more logical to follow the task-oriented layout, and also proposed that the evaporator and condenser modules should be placed next to each other instead of having the compressor placed in the centre of the control view./par

In the configuration of the PID settings, the evaluator found it good that the user needs to read in the current values from the PLC to avoid mistyping and that it has a separate send button. What the evaluator was missing was some sort of pop-up message asking "Are you sure that you want to send these settings?" to guarantee that the user would not send any values that are incorrect.

The new naming of all parameters used in the system has been changed to a more logical manner, according to the evaluator. In the old documentation of all channels, the naming did not follow the development of the test rig, resulting in bad naming and channels that are no longer used. As a novice user of the new system, the evaluator expressed that the learning curve of the new control system is very short, and the design of the GUI appearance gives the impression of a thought behind all objects.

The calibration history file also supports the quality assurance of the testing process; in every testing session the test engineer can in a simple way access the current calibration parameters and store them together with the testing result.

Compared with the old system, the evaluator found the GUI to be more reliable and give more control to the test engineer while greatly increasing the functionality of the control system. Future development and maintenance can be performed in a more simple and reliable way.

6.2 System function verification

6.2.1 Test plan

To verify the capabilities of the integrated system, a test was performed. By setting up one PC with the GUI and one PLC with evaporator software, it was tested together with the related hardware of the physical process.

6.2.2 Scope

The test included testing the LabVIEW software together with one PLC and its related sensors and actuators. The test was made on a lab assembly that simulates the evaporator tunnel according to the real HVAC test rig system. The main difference from the real system was the optimal PID parameters compared with the integration test system.

6.2.3 Purpose

By conducting a functionality test it was verified that the system performs as it should and could be used as a basis for continued development, integrating the remaining PLC etc. The separate tests were as follows;

First test - Measuring and controlling the airflow

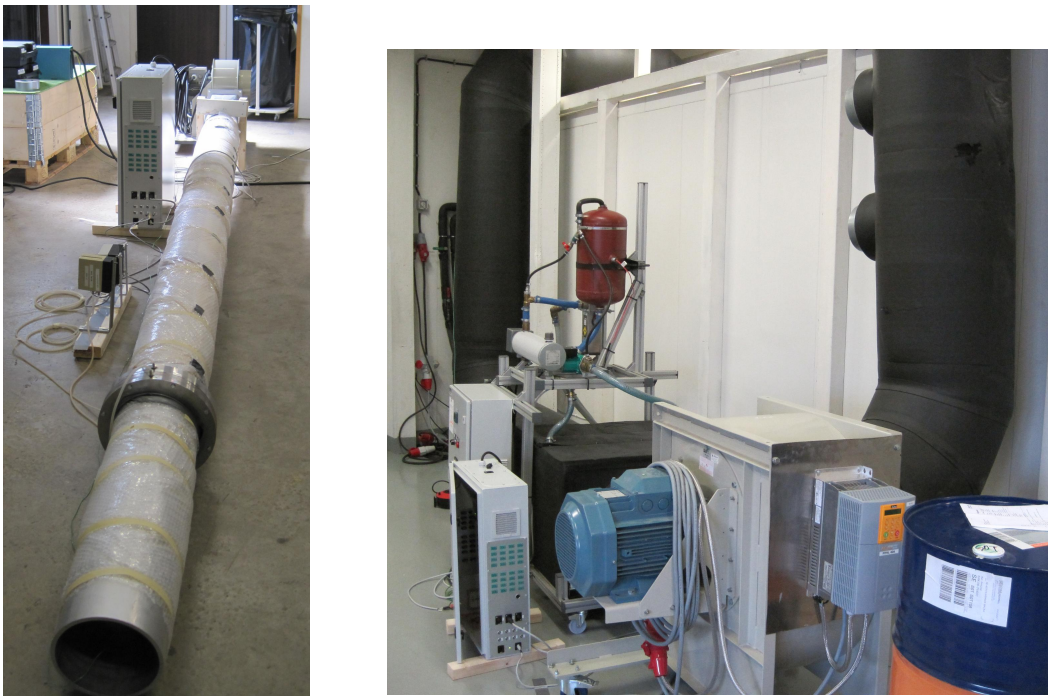
The main purpose of this test was to verify that the algorithm for calculating the airflow was working properly and that the PID loop can handle the fan control. The configuration of the test can be seen in Figure 6.3(a).

Second test - Measuring and controlling heater

This test verified that the heater control was functioning correctly. By setting a constant airflow through a heater the system should measure and calculate the control signal. It should be noted that this test was performed on a different physical system compared with the airflow test. The configuration can be seen in Figure 6.3(b).

6.2.4 Airflow test setup

As mentioned in connection with the purpose, this was a validation test of the airflow measurement and control, which was simulated on a test set-up that consisted of three main components: the pipe and orifice assembly, the fan and motor assembly, and the control cabinet with the evaporator PLC inside. When the air flow measurement



(a) (b)

Figure 6.3: a) The setup of the airflow test. b) The setup of heater test.

was validated, the control function was changed to an automatic mode to validate the function of the airflow PID controller.

6.2.5 Heater test setup

This test was made on an existing set-up for a climate chamber which consists of the second test fan unit and the second test heater unit; see Figure 6.4(b). This was an assembly with a fan unit and a heater connected in series. The heater unit was governed externally by a control signal from the evaporator control cabinet. The fan speed was set to a fixed airflow and the heater and thermo elements were to reach a steady state of the air temperature. By using a fixed airflow external sources of error could be reduced through the testing process.

6.2.6 Equipment

For each test the equipment used is presented here with a picture, product data and its role in the testing.

Thermocouple first and second test

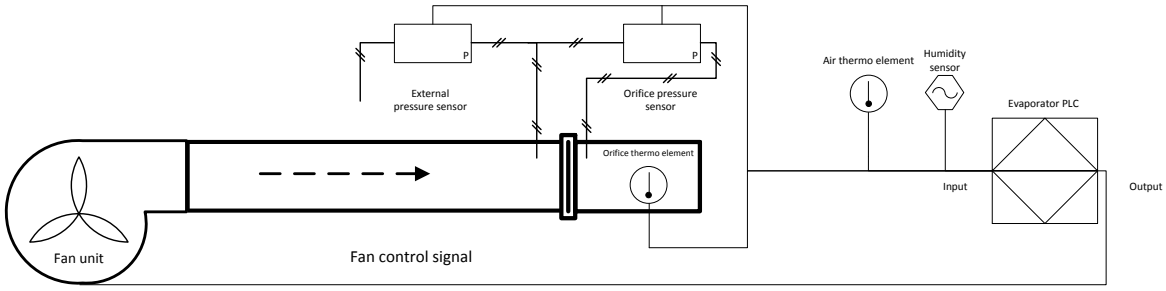
For all measurement of temperature standard thermocouples of type K were used. The principle of operation is that all conductors generate a voltage difference when exposed to different temperatures, also known as the thermoelectric effect. The thermocouple can measure between -200 and 1350 °C. The orifice temperature sensor can be seen in Figure 6.5.

Differential pressure sensor FSM first test

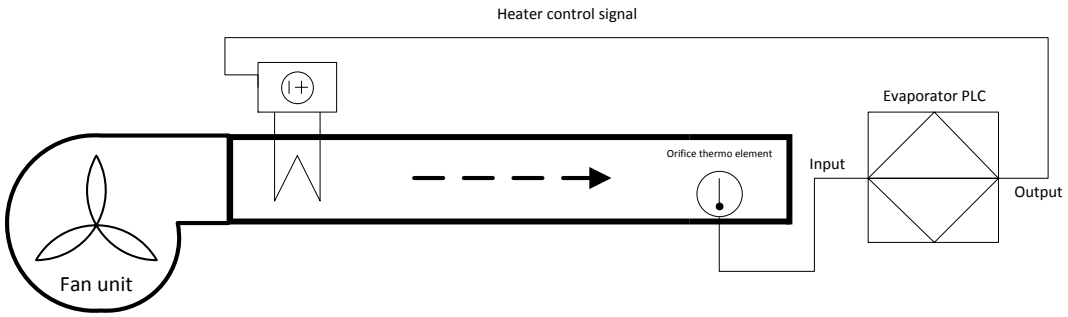
The sensor measures the difference between two pressure levels. It is made by FSM Elektronik GmbH with the name Pressure Transmitter type DPSAZ and serial number 00151209CRo. It has a measuring range between 0 and 25 hPa and has a nominal signal output from 0 to 5 VDC. See Figure 6.6(a).

Differential pressure sensor WIKA TRONIC first test

The sensor has the same function as the sensor in section 6.2.6 and was produced by Alexander Wiegand GmbH & Co with the name WIKA TRONIC 891.09.1968 and serial number 08928001. It has a nominal measurement output between 0 and 10 VDC. See Figure 6.6(a).



(a)



(b)

Figure 6.4: a) The schematic setup of the airflow test. b) The schematic setup of heater test.

Figure 6.5: The green cable is the thermo couple placed inside the tube with the orifice in the background

(a)

(b)

Figure 6.6: a) FSM pressure sensor to the left and the WIKA TRONIC pressure sensor to the right, connected with tubing from the orifice plate. b) The HygroFlex humidity sensor used in the airflow test.

Humidity sensor first test

For measuring the relative humidity a HygroFlex from Rotronic with serial number 60681866 was used. It returns a value between 0 and 10 VDC representing the relative humidity in %. See Figure 6.6(b).

Frequency drive first test

The control unit for the fan is a product from SSD drives of the model 650 series and with a serial number of 115652507002095. This is a standard general-purpose AC drive for three-phase units up to 7,5 kW. This unit is controlled by a remote signal of 0-10 VDC. See Figure 6.7(a).

Fan unit first test

The fan unit is a centrifugal fan from Sodeca SA, model CMP-1025-2T-4, driven by a 3 kW Siemens three-phase AC synchronous electrical motor. See Figure 6.7(b).

Tubing and orifice plate first test

The tubing used in the contraption is an aluminium pipe of 4 metres with a diameter of 200 mm and thickness of 8 mm. The thickness of the pipe ensures that it keeps its round and straight form, to avoid air turbulence before and after the orifice plate. The orifice is a steel plate with a diameter of 138 mm that creates a drop pressure when the air passes through, enabling a flow measurement according to the Bernoulli principle. See Figure 6.8.

Heater unit second test

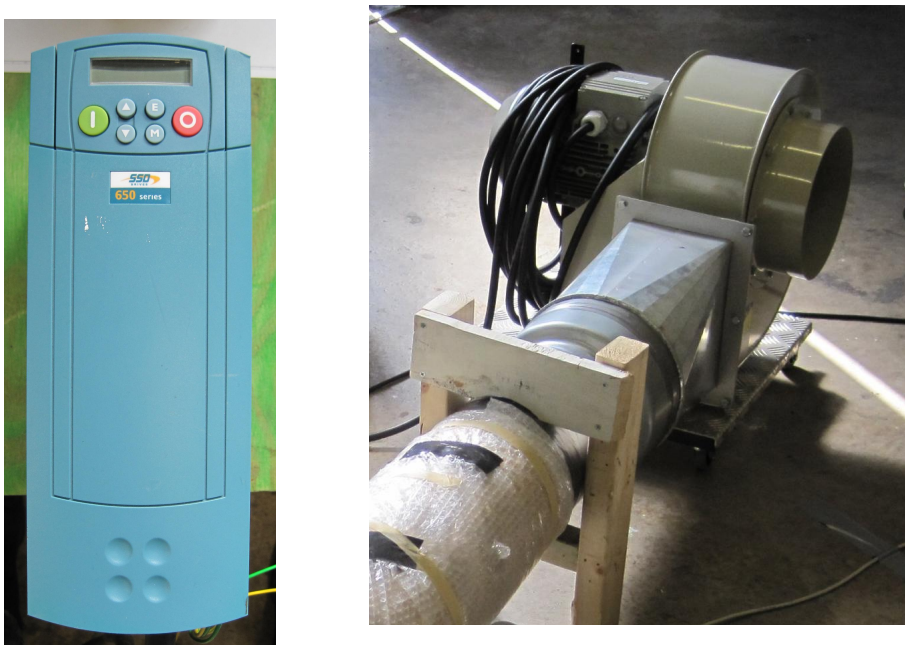
The heater unit used in the second test is an own built heat exchanger by Cool Engineering with a 9 kW electrical heater. It is steered remotely by a 4-20 mA control signal. See Figure 6.9(a).

Fan unit second test

In the heater test an axial fan unit from SODECA model CMR-7650-2T with an 11 kW ABB three-phase AC synchronous motor is used. See Figure 6.9(b).

6.2.7 Results

The results from the function validation result are divided into the two different tests that were made.



(a) (b)

Figure 6.7: a) Frequency drive to steer the fan unit with a control signal from the evaporator unit. b) The fan unit connected with the tubing.

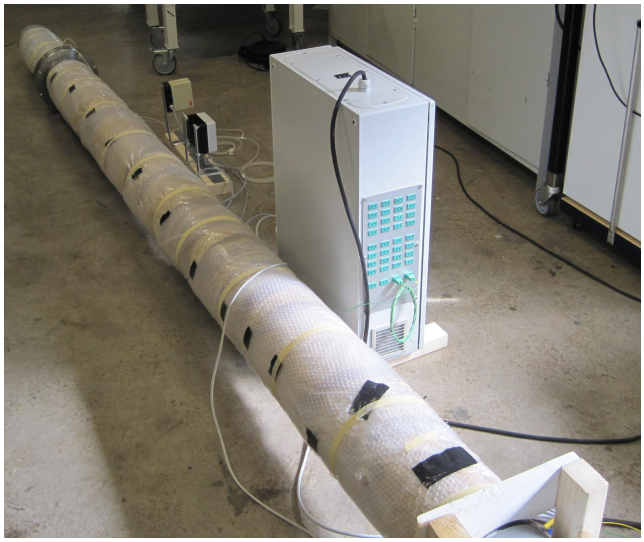


Figure 6.8: The tubing with the orifice plate in the far end.

(a)

(b)

Figure 6.9: a) The heater unit connected to the climate chamber. b) The fan unit connected with the tubing.

Airflow test

The orifice calculation method did not work properly and was verified by running the fan in manual mode to keep a steady level of the airflow. For measuring the airflow, a calculated method is used, based on the temperature, humidity and pressure differences over an orifice. This formula is complex, containing a number of external measurements. These data were verified and it could be seen that the signals from the pressure sensors was wrongly scaled. Additional errors of a logical type were found and removed in the calculation code. After these changes the airflow calculation worked as it should. It was also found that there were problems with communication stability between the LabVIEW and the PLC units. The control loop was tested and could be configured, and operated as it should without any significant problems. The control parameters were not optimized but could still reach steady state within a reasonable time. A snapshot from where the system reaches steady state can be found in Figure 6.10 where the set point-, process, and control-value can be seen taken in the graph view of the LabVIEW GUI. A test engineer confirmed the stability of the airflow functions and that the system could reach steady state faster and easier compared than the old control system on the test rig. The test engineer is the same person that participated in the third GUI evaluation; see Chapter 6.1.7.

Heater test

The heater test was successful, showing the ability to measure and steer the temperature. It should be mentioned that this heater test was relatively simple compared with the airflow test. Here the measurement from the air temperature was easy to make and the control signal was simple to steer. Its functionality was also confirmed by the same test engineer as in the airflow test. The conclusion about the system is that it is a slow system with a slow time response, as suspected. The only adjustment during the testing was changing the data source for the process value due to a wrongly chosen temperature sensor port.

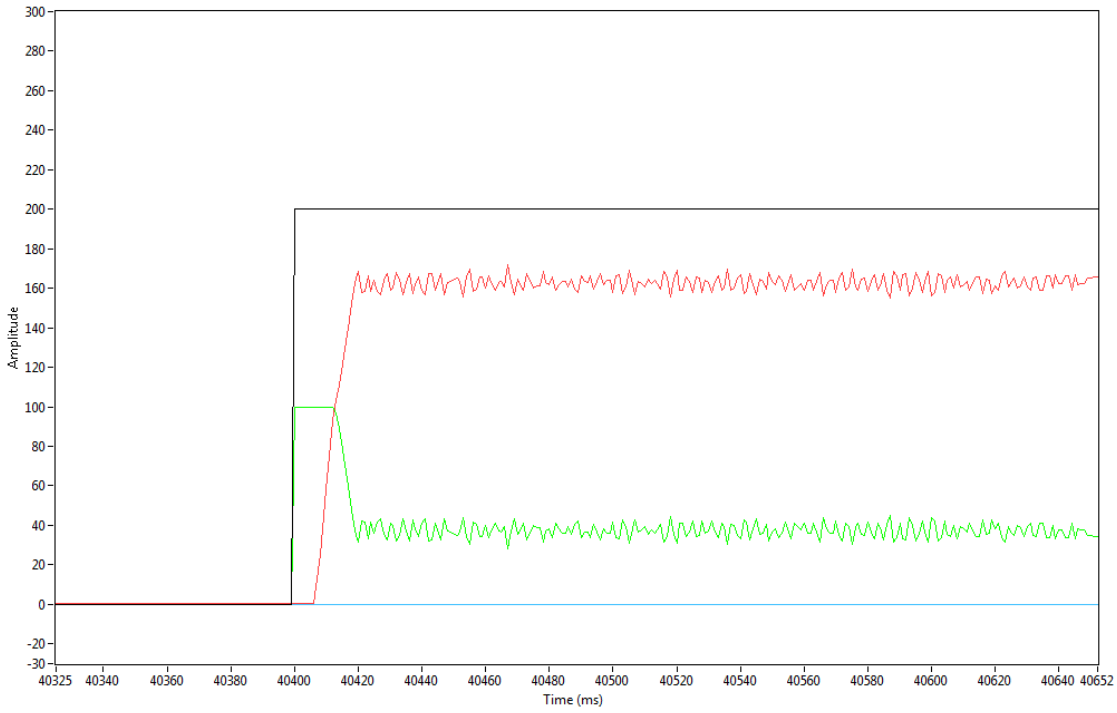


Figure 6.10: Data from the airflow test. The black line is the set point, the green the control signal and the red line the process value. The y-axis unit is gram of air

Chapter 7

Discussion

This chapter presents the discussion of the final result and the methods that have been used throughout the thesis.

The final result can be described as fulfilment of the requirement specification, but what remains is how well the new control system will perform when fully implemented. The conclusion that it would do well is based on a subjective evaluation of the functional test made by the test engineers. This should be considered as a qualified guess of its future performance regarding the objective of downtime and reliability. Also concerned is the objective of future configurations and refinement. The modular design and clear interfaces between modules indicate that the system would probably reach the performance goals when fully implemented, based on the current information from the evaluations. The tested modules of the airflow function and the heater functions have a lot in common with other modules in the system. Moreover, the basic code for the PLCs was validated and thus also the LabVIEW GUI through the evaluations.
In the discussion of general fulfilment of the objectives, it should be mentioned that the new system compared with the old system reaches all goals. But the current old system is underperforming, which indicates that only making a comparison is not appropriate. To avoid an incorrect evaluation, objective evaluation methods were used, which did not involve putting the old and new system in relation to each other. Hence the objectives regarding a clear and efficient LabVIEW GUI could be fully examined, and so also the basic reliability of the new control system concerning the placement and communication of critical parameters between the modules. By simulating certain scenarios in the GUI evaluation, it could be seen that the LabVIEW GUI supports the testing process satisfactorily even though a real test on the test rig was not performed.

Chapter 8

Conclusion and recommendation for future work

The conclusion of the thesis and a recommendation for the future work are presented in this chapter.

8.1 Conclusion

The conclusion that can be drawn from the evaluations is that the control system supports the test process and the user needs. From the functionality evaluation, the conclusion is that with a fully implemented system the downtime could be significantly reduced. The requirement specification within the final scope of the project has been fulfilled, including the project's accomplishment of documentation and implementing the majority of the current and future needs of the software. The methods used in the pre-study of the HVAC test rig and the test process proved to be valuable in the implementation of the LabVIEW GUI and the PLC code. /par -By using this structured development strategy, it could be seen that the agile coding phase reached a satisfying end result. This probably helped to shorten the development time and simultaneously enhance the quality of the result.
In the GUI evaluation, the evaluators experienced the new LabVIEW GUI as simpler and faster to start up compared with the old panel system. The test engineers, who should be seen as experienced users of the test rig, could start up the system faster and the inexperienced maintenance engineer could also reach operational mode without any major problems. This proves that a clear and efficient LabVIEW GUI has been reached.
By using a modularized system, a safer and more reliable overall control system has been obtained. This is because clear interfaces and safety functions have been incorporated in both the LabVIEW GUI and the PLCs. If a critical situation emerges, all

actuators in the test rig can cooperate for a controlled shut down of the HVAC test rig. By including a watchdog function within the system, all control units can act by themselves if one unit ceases to respond, or if the communication connection between them is lost. All critical parameters, e.g. calibration values are stored locally in the PLCs to ensure that the new system serves the test process in a reliable way.
Compared with the old control panel, the LabVIEW GUI can be configured and maintained more simply. It now shows calibrated process values and the controllers can be configured easier and faster according to the test engineers. Sensors and actuators can be added in the PLC control software without affecting the rest of the system negatively. This aspect, together with the modularized structure, shows that future configurations and refinement can be made in a simpler way than with the old control panel.

8.2 Recommendation for future work

This type of general software does not have a specific ending point when it comes to further development. For this specific project, the next step will be to implement the PLC software on the condenser and compressor unit and to perform a function evaluation on these. To adapt the evaporator PLC code for the other PLCs should be relatively easy since the development had a focus on creating generic and modularized software. The LabVIEW GUI can be further developed with more functionality, especially in the background of the software, including safety and reliability functions and so on. Also the general development methods within the software development can be refined further, compared with classical product development which focuses more on physical products. Hopefully this project can serve as a case in the future development of these development strategies.

References

[1] K. Ulrich and S. Eppinger, *Product Design and Development*. McGraw-Hill/Irwin, fourth ed., 2008.

[2] S. Daly, *Automotive air-conditioning and climate control systems*. Elsevier Butterworh-Heinemann, Amsterdam, Boston, first ed., 2006.

[3] WAECO Svenska AB, "Teknisk dokumentation för AC-anläggningar i fordon," 2000. Swedish technical handbook for automative HVAC systems.

[4] I. Sommerville, *Software Engineering*. Addison-Wesley, eighth ed., 2007.

[5] B. Shneiderman and C. Plaisant, *Designing the User Interface (Strategies for Effective Human-Computer Interaction)*. Addison-Wesley, fourth ed., 2005.

[6] G. Fleischanderl, "User interface requirements for knowledge acquisition and modeling," *Workshop ECAI*, 2000.

[7] M. Bohgard, S. Karlsson, E. Lovén, L.-Å. Mikaelsson, L. Mårtensson, A.-L. Osvalder, L. Rose and P. Ulfvengren, *Arbete och teknik på människans villkor*. Prevent, first ed., 2008.

[8] J. Nielsen, "10 heuristics for user interface design." http://www.useit.com/papers/heuristic/heuristic_list.html, accessed August 30, 2011.

[9] National Instruments, "What is ni labview?." http://www.ni.com/labview/whatis/, accessed August 25, 2011.

[10] "Is LabVIEW a general purpose programming language? - developer zone - national instruments." http://zone.ni.com/devzone/cda/tut/p/id/5313, accessed September 29, 2011.

[11] National Instruments, "Application design patterns: Producer/consumer - developer zone - national instruments." http://zone.ni.com/devzone/cda/tut/p/id/3023, accessed July 10, 2011.

[12] Beckhoff Automation, "Beckhoff new automation technology." http://www.beckhoff.se/se/default.htm?twincat/default.htm, accessed August 9, 2011.

[13] "Standards." http://www.plcopen.org/pages/tc1_standards/, accessed August 9, 2011.

[14] W. O. Galitz, *The Essential Guide to User Interface Design*. John Wiley & Sons, Inc. , New York, second ed., 2002.

[15] P. C. L. Bass and R. Kazman, *Software Architecture in Pratice*. Addison-Wesley, Boston, second ed., 2003.

[16] J. Bosch, *Design and Use of Software Architectures*. Addison-Wesley, Harlow, UK, 2000.

Appendix A

Interview questions

Interview

User requriements

1. What should be presented on the screen?
    a. Configurability of data presentation?
    b. Possibility to compare data? (have two types of data in same graph?)
    c. How often should it be updated?
2. What should be controlled on the screen?
    a. Automation functions for steady state?
    b. Manual functions?
    c. Detailed functions?
    d. Alarms and safety functions
    e. Maintenance functions on PLCs – calibration PID etc
        i. Possibility to see status of PLC?

System requirements

Technical documentation

3. How should it be formed?
    a. Comments in code?

Go from PLC to PLC and PC

4. What data needs to be send to the CoolSeq computer?
    a. If it sends any values
5. How should the data from the PLCs be sent?
    a. hardware
    b. Format
    c. Freq
6. What parts is configured from test to test
7. What parts is configured in the long term?
8. PLC hardware – Beckhoff 9100 with KL-series

Compressor PLC

9. Actuators
    a. What should be controlled
        i. Prepare to control second compressor in future?
    b. How should it be controlled?
    c. Which control processes?
10. Sensors
    d. What and how?
11. Safety functions
    a. Overheat?

b. Overpressure in system?

Condensator PLC

12. Actuators
    a. What should be controlled
    b. How should it be controlled?
    c. Control process?
13. Sensors
    d. What and how?
14. Safety functions

Evaporator PLC

15. Actuators
    a. What should be controlled
    b. How should it be controlled?
    c. Control process?
16. Sensors
    d. What and how?
17. Safety functions

Appendix B

Affinity diagram

Control Parameters

The user always wants to see the control signal, momentum over compressor and air temperature after evaporator.

PWM of compressor should be showed and controlled on main screen

On/off values for motors pwm duty cycle.

On/off Button to compressor

3(4) Air temperatures should be showed and controlled on main screen

2 (3) Airflows extra evaporator airflow should be showed and controlled on main screen

level of moisture in evaporator tunnel should be showed and controlled on main screen

RPM of the compressor should be showed and controlled on main screen

Test object expansion valve level should be controlled

Compressor actuators:
RPM motor
RPM Compressor
Torque on compressor (calculated value from torque on motor)
On/off compressor
PWM duty cycle

Compressor controlled:

regulator of rpm on motor

motor speed control

compressor on/off

Future Configuration

There is a possibility that ventilation fans will be introduce to the condenser tunnel, manual control

Plans to introduce more moisture sensors in the evaporator

eventually cooling of in air control

Data Presentation

Various data should be able to be chosen from the data channels and some calculated values.

The axis should be configurable and zooming for detecting steady state.

It (data presentation) needs to be very flexible in that way that the user can place several channels in the same graph and that the scales are individual and configurable for each channel.

Manual Control

Lock the heaters on a specific level.

Possibility to lock the fans in the tunnels to a specific rpm.

Air moisture could be locked manually.

Possibility to set values to other additional functions such as a possible future ventilation fan for the condenser tunnel.

Configuration

Possibility to configure PID parameters

PID parameters should be configurable.

PID parameters needs to be change due to problems with hysteresis in some actuators from test to test.

Gearing ratio between motor and compressor

From test to test the duty cycle of the PWM signal to the compressor engine Sometimes needs to be changed.

Update frequency

The presented values should be able to change updating frequency, usually its 2-4Hz but up to 10 Hz.

Update frequency on graphs: 2 Hz max 4 Hz

Internal updating speed of the system could work in 10 Hz.

Updating speed: 2,3 Hz for numbers and 5 Hz for the graph.

Organization

All data shall be sent to CoolSeq

There is a company standard for the plc project layout.

There is a Labview block written by erik to pick values from PLC to LabVIEW.

CoolSeq updates at a 4 Hz rate too. Might be problems with data socket.

The user experiences the testing situation stressful because of all the values to keep track on simultaneously.

Automation

Cycling of compressor is a possible active automation.

There should be a possibility to turn on and off control loops and set manual values on actuators to avoid resonance and also give the possibility for further experimentation.

Automatic ramping of air temperature when testing "ice guard".
Automatic Test sequences for testing program.
Some extra configured regulators in software

There is a wish for two "blank" alarms that should be able to configure from Labview, pick channels as input and some action for output.

Maybe a watchdog function

Control Presentation

All the actual values and the desired values for the control loops is showed

Also % of the control signal.

Pressure guard

safety function for over pressure that stops the system.

Code for emergency stops should be included to reset all values in the plc. The emergency stop breaks the power mechanically.

Pressure release function- can stop the compressor fast and easy from operators screen.

Calibration

Calibration of sensors, by polynomial or interpolation of table.

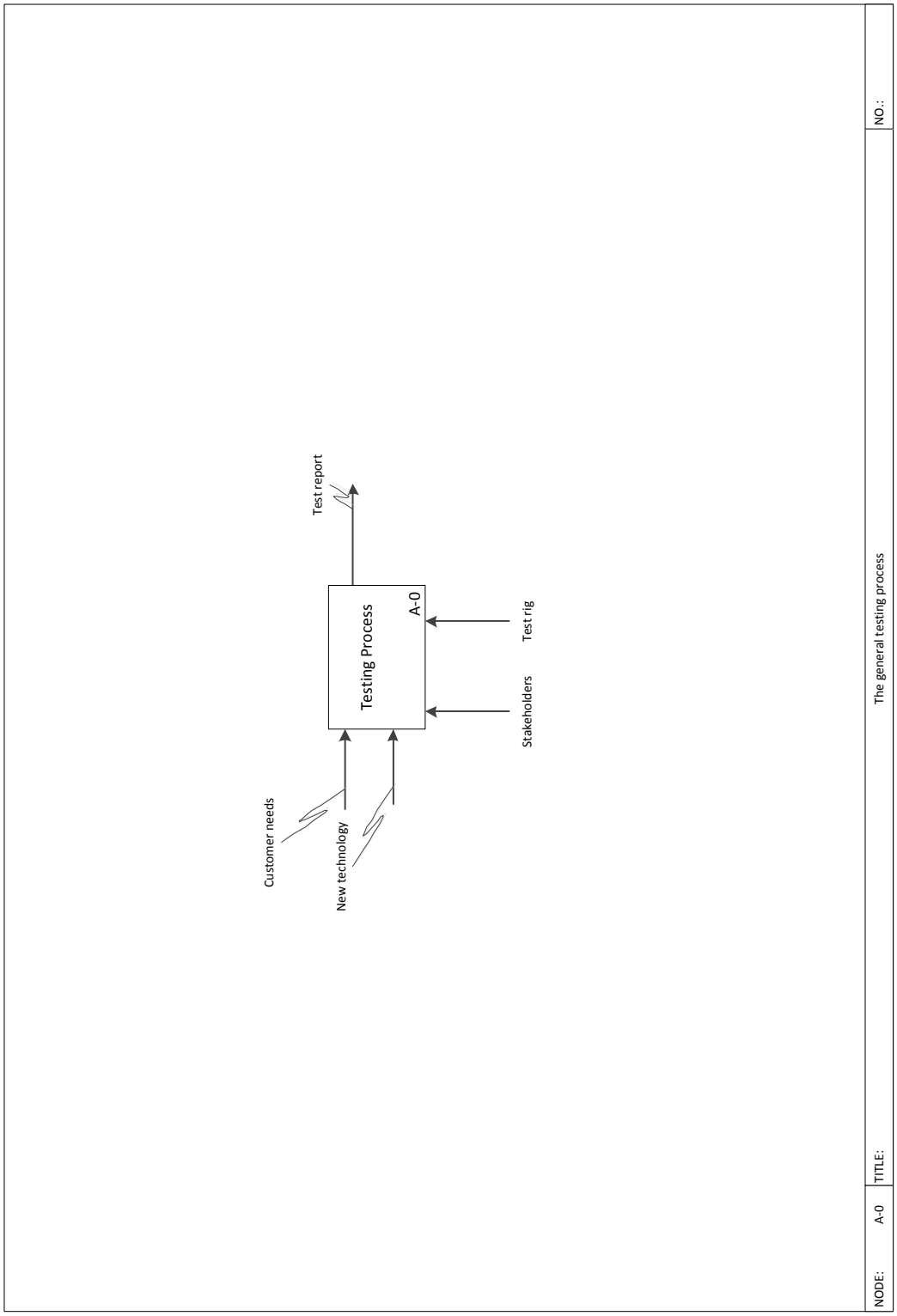
Should be made in such way to prepare for external software of calibration can reach the values.

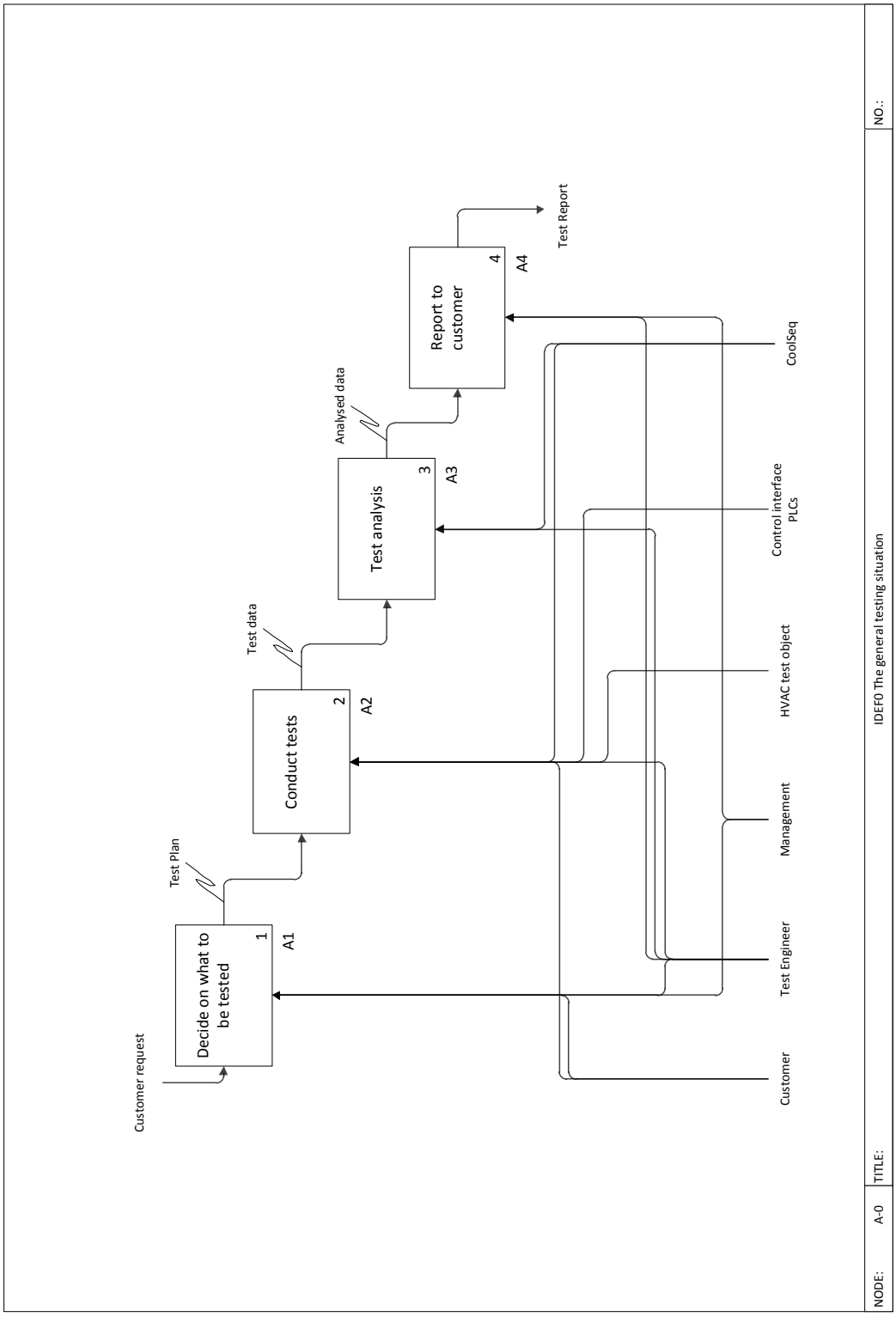
Should be able to see history on calibration and have the possibility to export the polynomials.

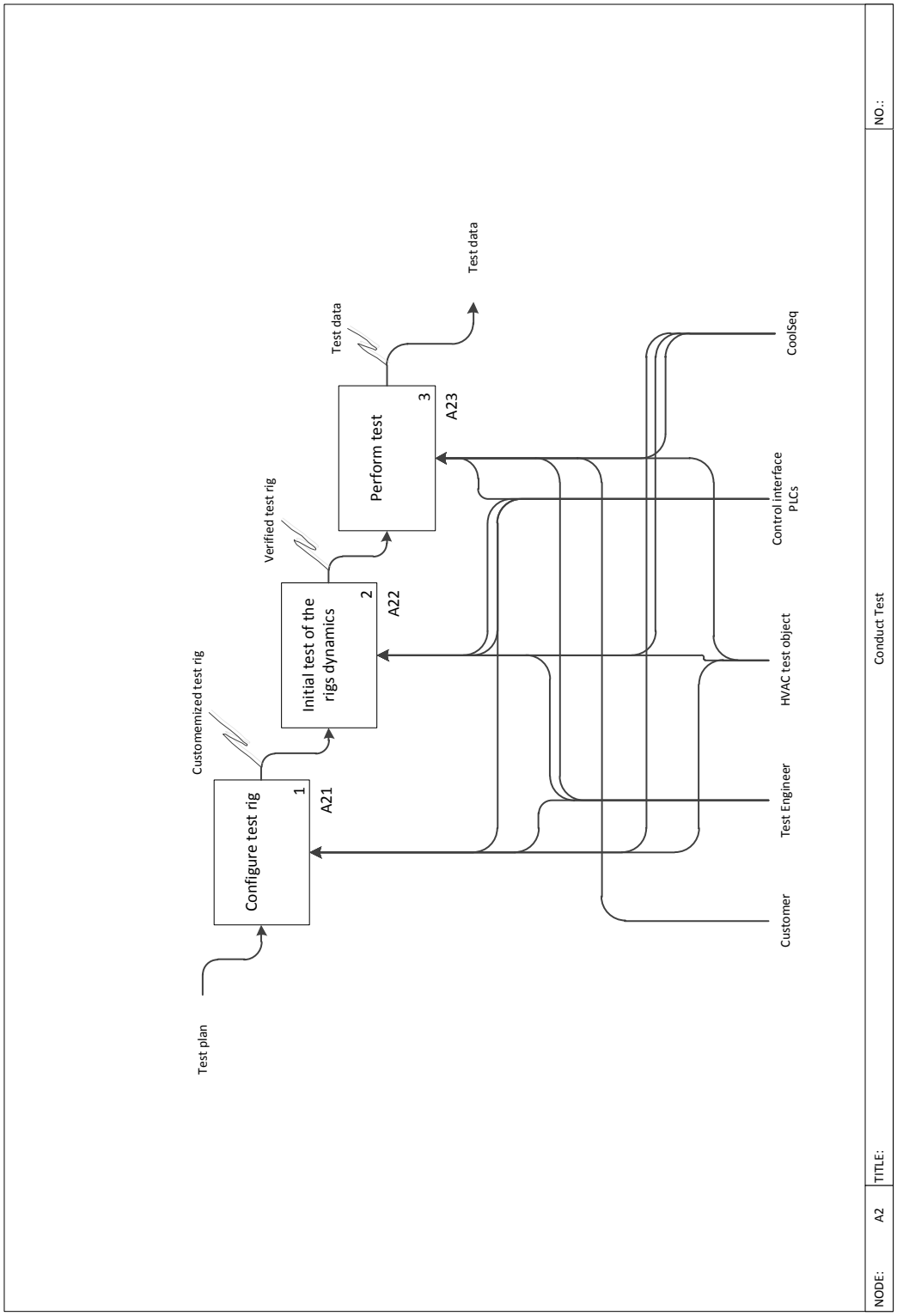
Appendix C

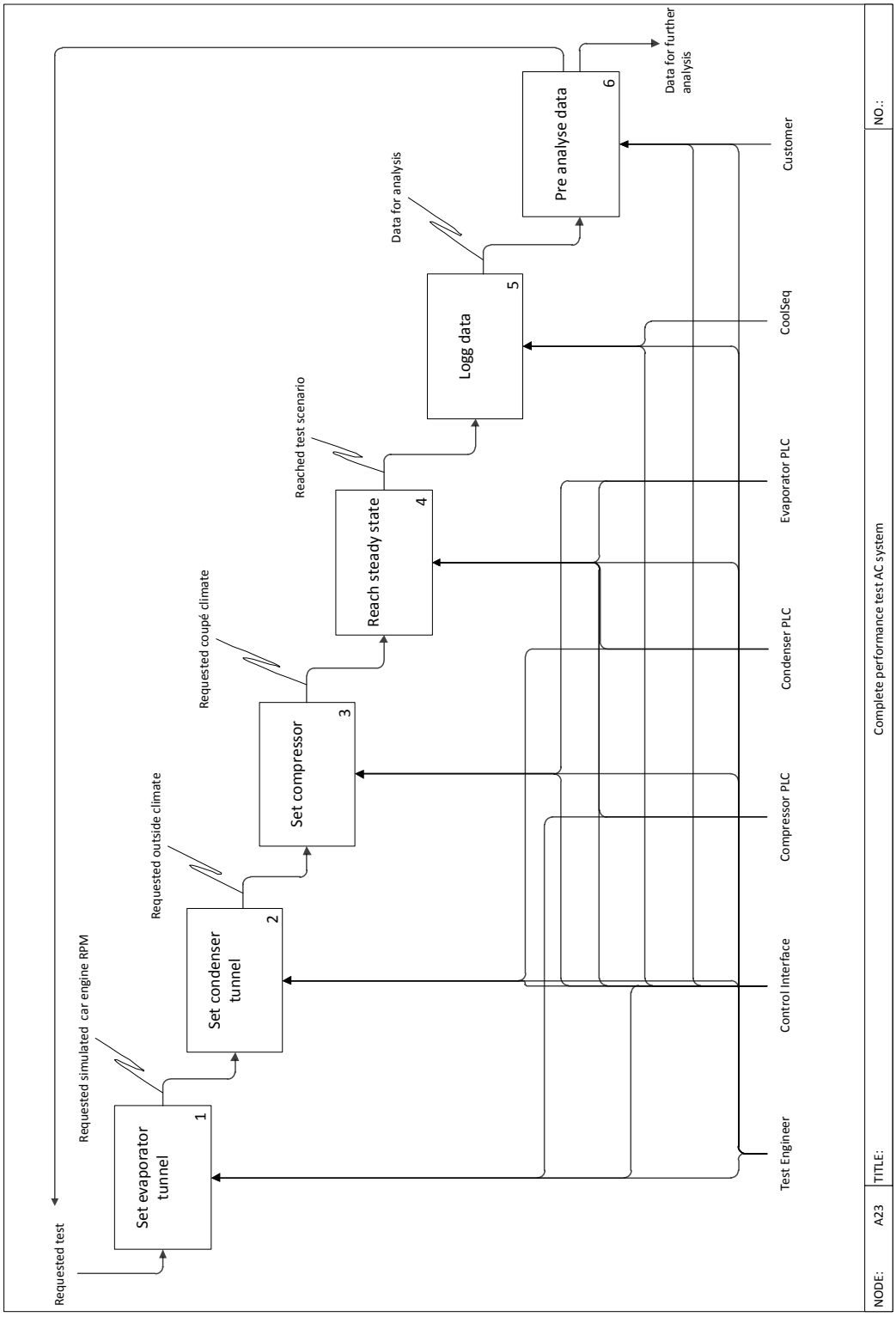
IDEF0 - The general testing situation

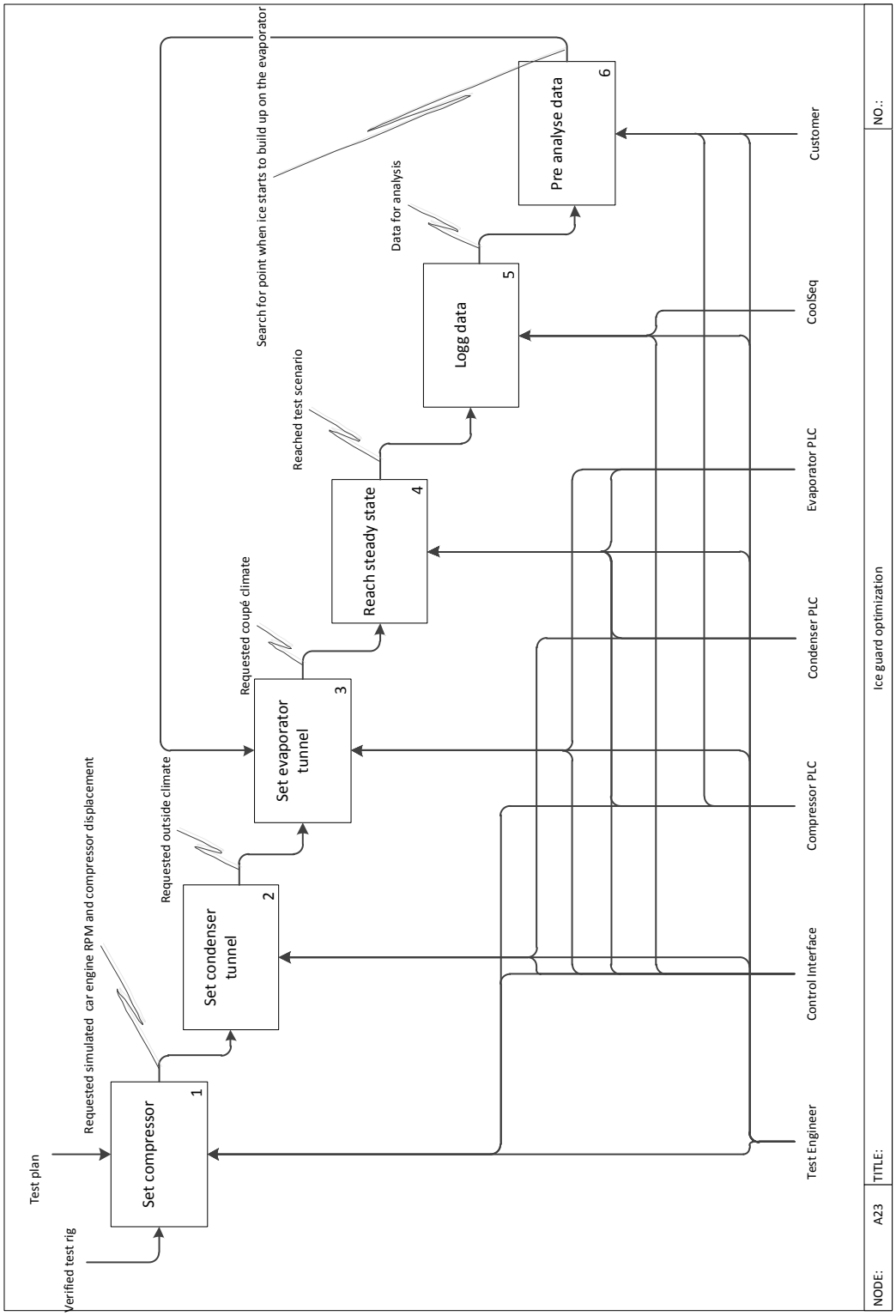
1. A-0 The general testing situation
2. A0 The general testing situation
3. A2 The general testing situation
4. A23 Complete performance test AC system
5. A23 Ice guard optimization
6. A231 Set compressor
7. A232 Set condenser
8. A233 Set evaporator

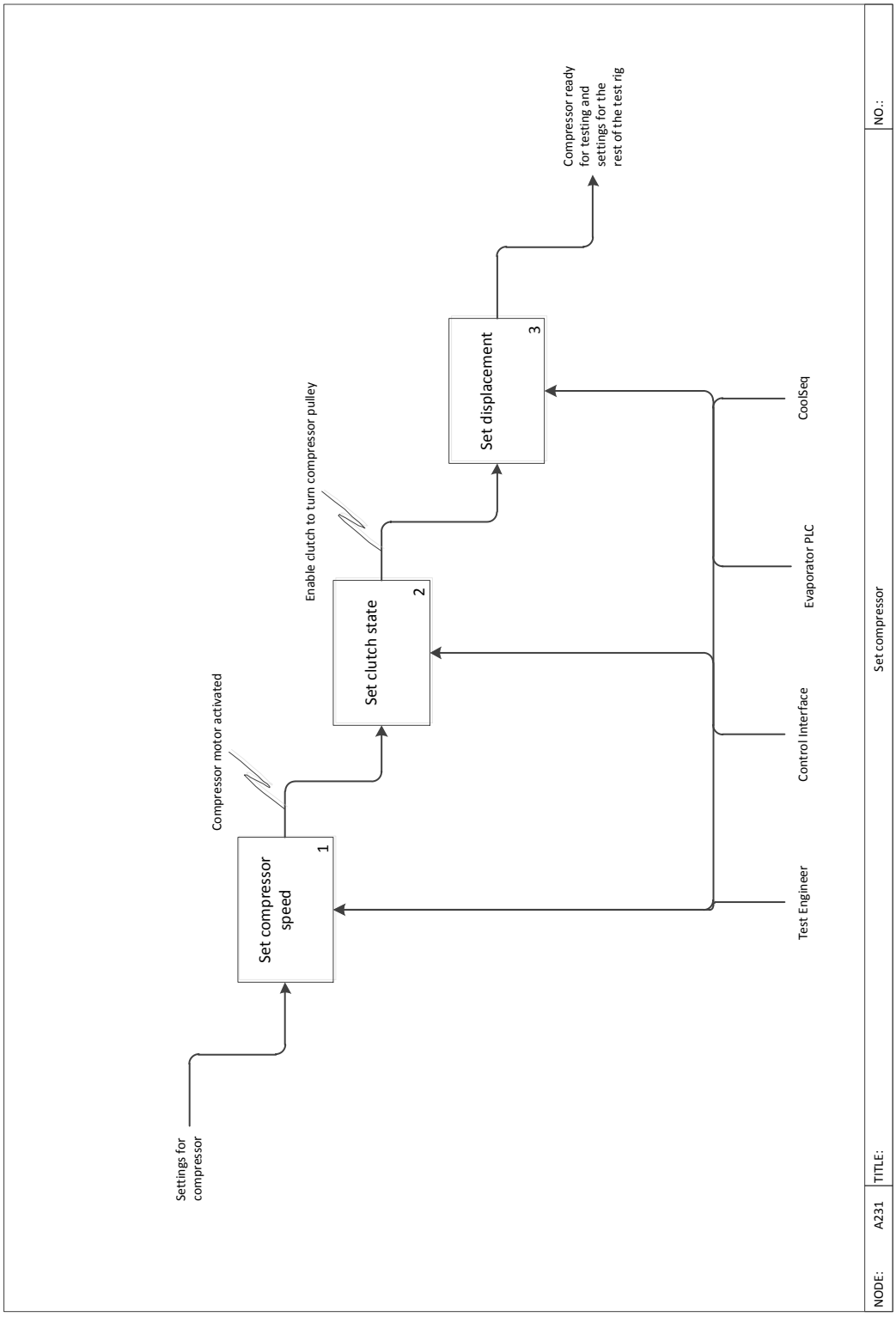

Test report
Testing Process
A-0
Test rig
Stakeholders
Customer needs
New technology
NODE:
A-0
TITLE:
The general testing process
NO.:

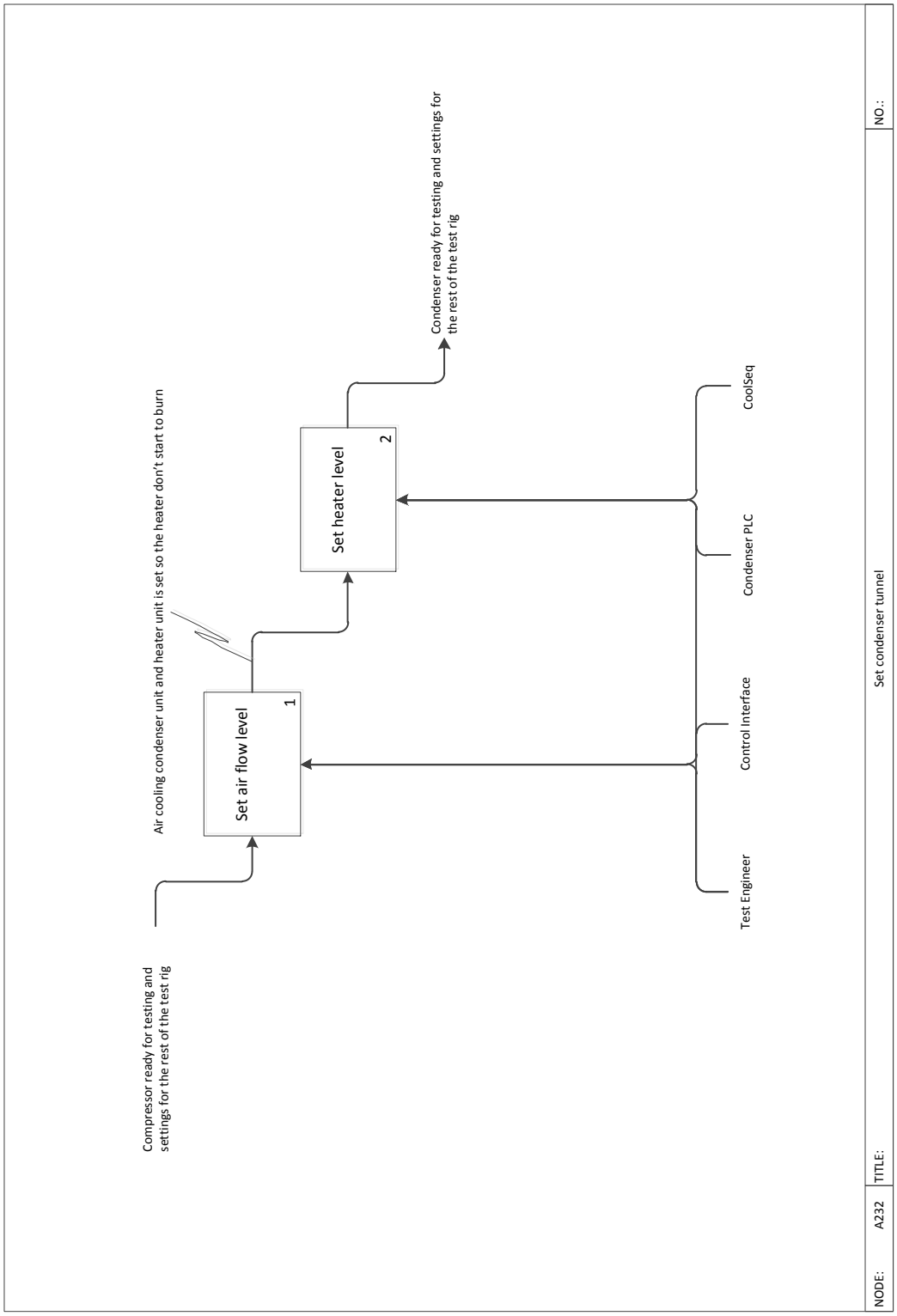

Customer request
Decide on what to be tested
1
A1
Test Plan
Conduct tests
2
A2
Test data
Test analysis
3
A3
Analysed data
Report to customer
4
A4
Test Report
Customer
Test Engineer
Management
HVAC test object
Control interface PLCs
CoolSeq
NODE: A-0
TITLE: IDEF0 The general testing situation
NO.:

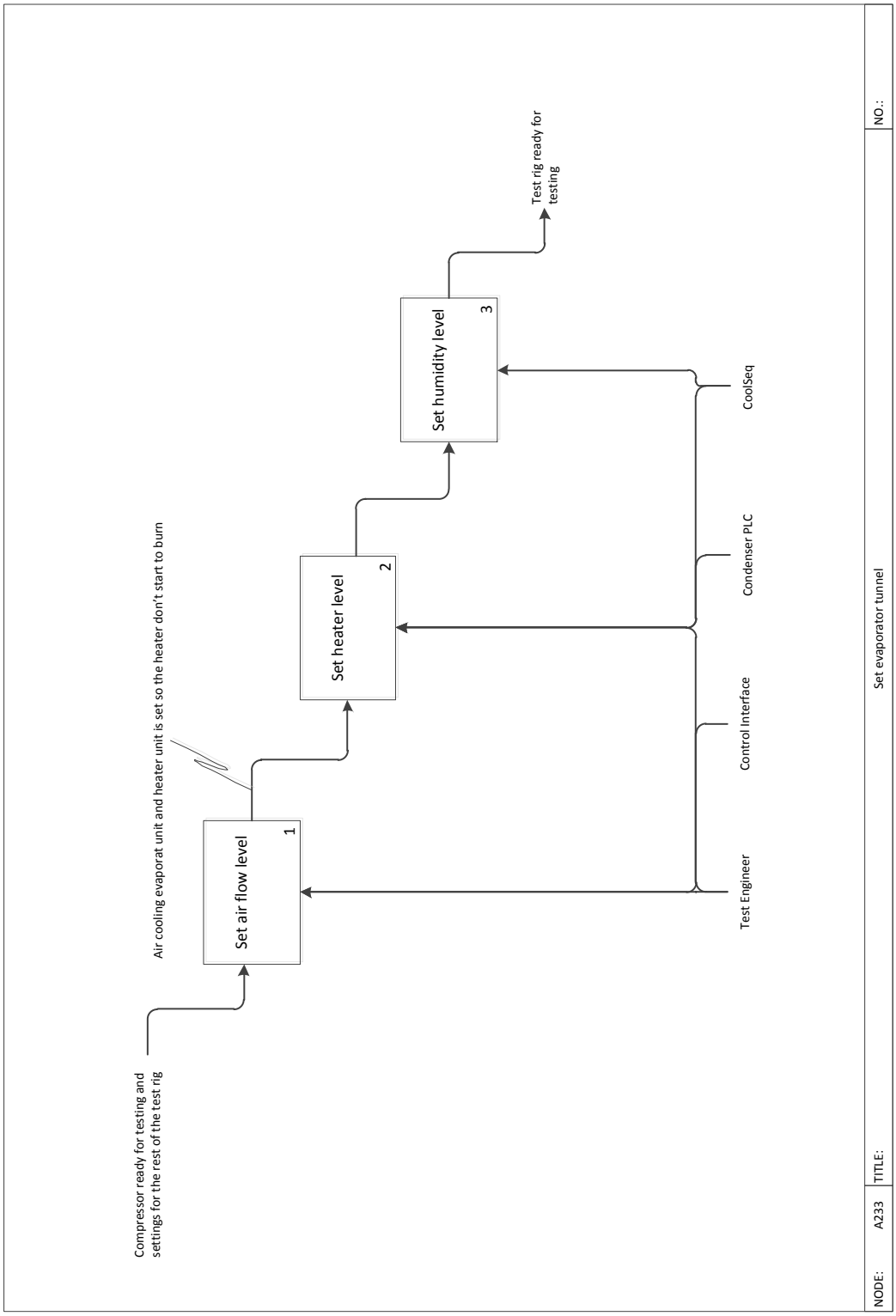

Test plan
Configure test rig
1
A21
Customemized test rig
Initial test of the rigs dynamics
2
A22
Verified test rig
Perform test
3
A23
Test data
Test data
CoolSeq
Control interface PLCs
HVAC test object
Test Engineer
Customer
Conduct Test
NODE:
A2
TITLE:
NO.:


Requested test
Set evaporator tunnel
1
Requested simulated car engine RPM
Set condenser tunnel
2
Requested outside climate
Set compressor
3
Requested coupé climate
Reach steady state
4
Reached test scenario
Logg data
5
Data for analysis
Pre analyse data
6
Data for further analysis
Customer
CoolSeq
Evaporator PLC
Condenser PLC
Compressor PLC
Control Interface
Test Engineer
NODE: A23
TITLE: Complete performance test AC system
NO.:


Test plan
Verified test rig
Requested simulated car engine RPM and compressor displacement
Requested outside climate
Requested coupé climate
Reached test scenario
Data for analysis
Search for point when ice starts to build up on the evaporator
Set compressor
1
Set condenser tunnel
2
Set evaporator tunnel
3
Reach steady state
4
Logg data
5
Pre analyse data
6
Test Engineer
Control Interface
Compressor PLC
Condenser PLC
Evaporator PLC
CoolSeq
Customer
NODE: A23
TITLE: Ice guard optimization
NO.:

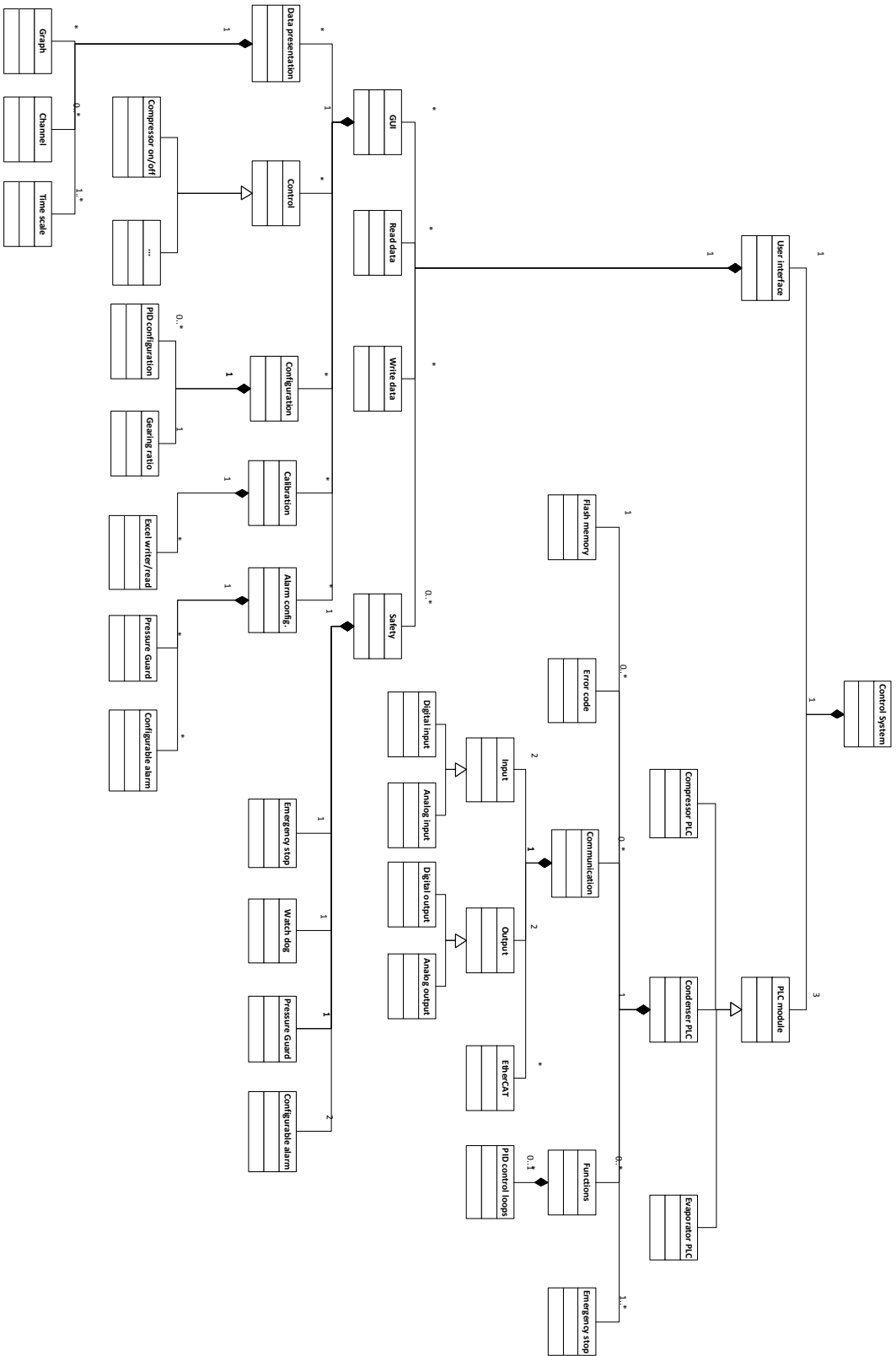

Settings for compressor
Set compressor speed
1
Compressor motor activated
Set clutch state
2
Enable clutch to turn compressor pulley
Set displacement
3
Compressor ready for testing and settings for the rest of the test rig
Test Engineer
Control Interface
Evaporator PLC
CoolSeq
NODE:
A231
TITLE:
Set compressor
NO.:


Compressor ready for testing and settings for the rest of the test rig
Air cooling condenser unit and heater unit is set so the heater don't start to burn
Set air flow level
1
Set heater level
2
Condenser ready for testing and settings for the rest of the test rig
Test Engineer
Control Interface
Condenser PLC
CoolSeq
NODE:
A232
TITLE:
Set condenser tunnel
NO.:


Compressor ready for testing and settings for the rest of the test rig
Air cooling evaporat unit and heater unit is set so the heater don't start to burn
Set air flow level
1
Set heater level
2
Set humidity level
3
Test rig ready for testing
Test Engineer
Control Interface
Condenser PLC
CoolSeq
NODE:
A233
TITLE:
Set evaporator tunnel
NO.:

Appendix D

Class model


Graph
Channel
Time scale
Data presentation
Compressor on/off
...
Control
PID configuration
Gearing ratio
Configuration
Excel writer/read
Calibration
Pressure Guard
Configurable alarm
Alarm config.
GUI
Read data
Write data
Safety
Emergency stop
Watch dog
Pressure Guard
Configurable alarm
User interface
Control System
PLC module
Flash memory
Error code
Communication
Functions
PID control loops
EtherCAT
Input
Digital input
Analog input
Output
Digital output
Analog output
Compressor PLC
Condenser PLC
Evaporator PLC
Emergency stop

Appendix E

Requirement specification

Organizational

PLC hardware

Description	The PLC hardware used as a new standard at Cool Engineering which is the Beckhoff 9100 series. The PLC programing should follow this constrain.
Requirement class	Organizational standards requirement
Stakeholders	Test engineer, maintenance engineer and management
Priority	1
Verification method	Evaluation

Communication between User interface and PLCs

Description	The PLCs shall communicate with the GUI based on EtherCAT.
Requirement class	Product reliability requirement
Stakeholders	Maintenance engineer
Priority	1
Verification method	Evaluation

Communication between User interface and CoolSeq

Description	The GUI shall communicate with CoolSeq via Datasocket.
Requirement class	Product reliability requirement
Stakeholders	Maintenance engineer
Priority	1
Verification method	Evaluation

Control software

Description	The control software implemented on the standard PC shall be based on LabVIEW 2010 runtime environment.
Requirement class	Product implementation requirement
Stakeholders	Test engineer and maintenance engineer
Priority	1
Verification method	Evaluation

IO-channels

Description	All IO channels are fixed for each individual PLC and needs to be followed.
Requirement class	Product implementation requirement
Stakeholders	Test engineer and maintenance engineers
Priority	1
Verification method	Evaluation

Internal update frequency

Description	The LabVIEW system shall update its parameters with a rate of 10 Hz.
Requirement class	Functional requirement
Stakeholders	Test engineer
Priority	2
Verification method	Evaluation

Documentation of code

Description	All code shall be well documented by commenting in the code and also a separate document describing the program in a natural language.
Requirement class	Documentation requirement
Stakeholders	Test engineer, maintenance engineer and management
Priority	1
Verification method	Evaluation

Control presentation

Presentation of process and set point value

Function	Show process and set point value
Description	The user interface shows the process value and the set point value grouped together in pares.
Stakeholders	Test engineer and customer
Priority	2
Inputs	Process value
Outputs	Setpoint value
Action	Updates the process value and updates the PLC if the set point value is changed.
Requirements	Communication between user interface and the PLCs
Side Effects	None
Requirement class	Usability requirement
Verification method	Evaluation

Presentation of control value

Function	The user interface shows the control value in % of maximum output of each actuator.
Description	The user normally needs to see the actual control value on the actuator to indicate its current state and the limitations in working area of the steer signal.
Stakeholders	Test engineer and maintenance engineer
Priority	2
Inputs	Current status on the actuators control value
Outputs	None
Action	The value is updated by reading values sent from the PLC and recalculated into % to get the control value.
Requirements	Communication between user interface and the PLCs
Side Effects	Might be confusing for a new user.
Requirement class	Functional requirement
Verification method	Evaluation

Automatic or manual mode on control

Function	Switch between automatic and manual mode of a functions
Description	Each control that is connected to a PID control loop shall have the possibility to go in manual mode by pressing a button to leave automatic mode and entering an control value instead of set point value
Stakeholders	Test engineer
Priority	2
Inputs	State of control loop
Outputs	State of control loop

Action	If the control loop is deactivated the user can edit the control value in % to steer the actuator
Requirements	Possibility to deactivate control loops and possibility to make the control value editable and recalculated to a signal to the PLCs.
Side Effects	Might be confusing and easy to forgotten in manual mode
Requirement class	Usability requirement
Verification method	Evaluation

Boolean input representation

Description	All the Boolean output from the user should be represented as standard OS-system on/off buttons.
Requirement class	Interface requirement
Stakeholders	Test engineer
Priority	3
Verification method	Evaluation

Numerical input representation

Description	The numerical output control should be represented as a LabVIEW "numerical control" with a relevant precision of decimals.
Requirement class	Interface requirement
Stakeholders	Test engineer
Priority	3
Verification method	Evaluation

Grouping of control parameters

Description	The control parameters should be grouped into the subgroups; Compressor, Evaporator and Condenser to avoid confusions. The old system had this control layout and has then created a mental model of the basic system layout for the user.
Requirement class	Procedural interface
Stakeholders	Test engineer
Priority	2
Verification method	Evaluation

Update frequency on numerical values

Description	The update frequency on numerical values shall be 2 Hz. The user experiences a faster update frequency hard to read.
Requirement class	Usability requirement
Stakeholders	Test engineer
Priority	2
Verification method	Evaluation

Data presentation

Comparing data in same graph

Function	Showing different curves in same graph
Description	The user needs to be able to compare data of different types in the same graph. Minimum two different data source should be able to be picked from any IO-channels.
Stakeholders	Test engineer and customer
Priority	2
Inputs	Data from any data channel
Outputs	Presentation in graph
Action	When a channel is picked it is showed in the graph and the scale function is activated
Requirements	Graphs in the user interface, access to data channels and flexible scales.
Side Effects	None
Requirement class	Functional requirement and data representation
Verification method	Evaluation

Flexible scales

Function	The time period of each curve in the graph is configurable
Description	For each type of data in all graphs the time scales should be configurable to be able to zoom and compare data more easily.
Stakeholders	Test engineer and customer
Priority	2
Inputs	Data channels
Outputs	Numerical value for scales
Action	After changing the numerical value the scale changes and the graph updates.
Requirements	Graphs with multiple data
Side Effects	More objects in user interface
Requirement class	Functional requirement and data representation
Verification method	Evaluation

Selection of data channel

Function	Possibility to pick data from all data channels or calculated values
Description	Each graph should have multiple data sources based on data directly from data channels or calculated values
Stakeholders	Test engineer and customer
Priority	2
Inputs	Data from data channels
Outputs	Data in graph
Action	After picking a data channel it should be presented in the graph window. If a calculated value is picked its starts to calculate the value and present it in the graph.
Requirements	Access to data channels and graphs
Side Effects	None
Requirement class	Functional requirement and data representation
Verification method	Evaluation

Update frequency on graphs

Description	The user interface shall update the graphs in 10 Hz to show values that are current which also indicate that the system is active.
Requirement class	Data representation
Stakeholders	Test engineer and customer
Priority	2
Verification method	Evaluation

Control Functions

Compressor on/off function

Function	The user should be able to turn of the power to the compressor motor easy. Both in normal use and in case of risk of a damaging the test rig.
Description	A button on the screen is pressed with a mouse click which send a stop signal to the compressor motor.
Stakeholders	Test engineer, maintenance engineer, customer and management
Priority	1
Inputs	Current state on compressor motor
Outputs	A high and low signal to compressor PLC.
Action	The user interface sends a signal to compressor PLC start or stop compressor motor.
Requirements	Connection to PLC.
Side Effects	The button might need to be presented on all “screens” in the interface.
Requirement class	Usability and safety requirement
Verification method	Evaluation

Current air humidity

Function	The current air humidity needs to be entered into the system manually.
Description	The system needs to now the humidity to be able to calculate the air mass flow inside the test rig. This is made by entering the humidity value provided by e.g. metrology institutes. This part will be removed in the future by putting more humidity sensors into the system.
Stakeholders	Test engineer.
Priority	2
Inputs	Numerical value entered in the user interface
Outputs	A numerical value sent to the PLCs.
Action	The system will send the humidity data to the condenser and evaporator PLCs to be used in the calculation of the air mass flow.
Requirements	Connection to PLC.
Side Effects	Low precision of the data.
Requirement class	Functional requirement
Verification method	Evaluation

Compressor rpm

Function	Controls the rpm of the compressor
Description	A numerical value of the rpm is set and sent to the compressor PLC. The rpm of the compressor will increase, pumping more coolant medium in the HVAC system to transport heat away.
Stakeholders	Test engineer
Priority	2
Inputs	Rpm output from motor from rpm pulse sensor
Outputs	Numerical value sent to compressor PLC is passed on to the motor control unit as a 0-5 Volt signal.
Action	When the User interface sends the data to the PLC it will update the motor control that will assure that the motor reaches the rpm.
Requirements	Connection between User interface, the compressor PLC and the motor control unit. The compressor PLC needs to translate the pulses to a rpm value.
Side Effects	None
Requirement class	Functional requirement
Verification method	Evaluation

Evaporator fan control

Function	Controls the air mass flow in the evaporator tunnel.
Description	The fan tries so to keep a stable air flow according to a set air mass which is dependent on the air pressure, temperature, and humidity.
Stakeholders	Test engineer
Priority	2
Inputs	Current status from motor control unit via evaporator PLC
Outputs	Desired value sent to motor control unit via evaporator PLC
Action	The fan will be controlled by sending a value to its motor control unit, which will increase the mass flow of air through the evaporator.
Requirements	Control loop in the PLC
Side Effects	None
Requirement class	Functional requirement
Verification method	Evaluation

Evaporator temperature control

Function	Controls the air temperature in evaporator tunnel
Description	The temperature level is set in the user interface and controlled by a PID-control loop. The control loop will use readings of the air temperature and steer towards the desired level.
Stakeholders	Test engineer
Priority	2
Inputs	Temperature readings from test rig.
Outputs	A numerical number sent to the heater unit via the evaporator PLC.
Action	The air temperature in the evaporator tunnel will change, giving a new test scenario.
Requirements	Fast readings of the temperature sensor and a stable PID-control.
Side Effects	One more control loop in the system that might cause resonance in the overall system.
Requirement class	Functional requirement
Verification method	Evaluation

Evaporator humidity control

Function	Control the air humidity in the evaporator tunnel
Description	The humidity level is set by opening a valve which sprays water steam into the evaporator tunnel.
Stakeholders	Test engineer
Priority	2
Inputs	Humidity readings close to the evaporator unit inside the evaporator tunnel.
Outputs	A voltage level opens the valve from the boiler unit.
Action	The valve to the boiler will open spraying steam in to the tunnel changing the air humidity will change, giving a new test scenario.
Requirements	Fast readings from the humidity sensor and a stable PID-control.
Side Effects	One more control loop in the system that might cause resonance in the overall system.
Requirement class	Functional requirement
Verification method	Evaluation

Condenser fan control

Function	Control the air mass passed through the condenser unit.
Description	The mass of air passing through the condenser unit affect the ability to transfer heat from the evaporator tunnel.
Stakeholders	Test engineer
Priority	2
Inputs	Air temperature and air pressure from the condenser tunnel.
Outputs	Numerical rpm value sent to the fan motor control unit via the PLC.
Action	The air mass passed through the system will give a new test scenario.
Requirements	Readings from sensors and a PID-control loop.
Side Effects	One more control loop in the system that might cause resonance in the overall system.
Requirement class	Functional requirement
Verification method	Evaluation

Condenser temperature control

Function	Controls the air temperature in condenser tunnel
Description	The temperature level is set in the user interface and controlled by a PID-control loop. The control loop will use readings of the air temperature and steer towards the desired level.
Stakeholders	Test engineer
Priority	2
Inputs	Temperature readings from test rig.
Outputs	A numerical number sent to the heater unit via the condenser PLC.
Action	The air temperature in the condenser tunnel will change, giving a new test scenario.
Requirements	Fast readings of the temperature sensor and a stable PID-control.
Side Effects	One more control loop in the system that might cause resonance in the overall system.
Requirement class	Functional requirement
Verification method	Evaluation

Configurable control loops

Description	Two control loops should be configurable in the LabVIEW software and controlled locally from the concerning PLC by selecting data channels and PID parameters. These loops shall be used in case of introducing temporary actuators to the system.
Requirement class	Functional requirement
Stakeholders	Test engineer
Priority	4
Verification method	Evaluation

Automation

(De)activating control loops

Function	All control loops should be able to be turned on and off individually.
Description	The user can turn on and off all PID-control loops by sending a signal to the PLC where the control is performed. If deactivated the control will go into a manual mode sending a steer signal instead of a desired value.
Stakeholders	Test engineer and maintenance engineer
Priority	3
Inputs	None
Outputs	A bit set and sent to the PLC
Action	Resonance in the system can be avoided and also gives a greater possibility of experimentation for the test engineer. Also the manual control will help the maintenance of the test rig.
Requirements	The user interface needs to be able to connect to the individual PLCs of the test rig.
Side Effects	Using to many control loops simultaneously might cause resonance in the HVAC system
Requirement class	Functional and reliability requirement
Verification method	Evaluation

Watchdog function

Function	The watchdog will mutually check if the PC and PLCs are operating as it should.
Description	The PC will send continuously data to the different PLCs and they will respond. If one of the PLCs doesn't respond it is an indication that the subsystem might be hung. If the PLCs don't get any data from the PC it will indicate that the PC is hung.
Stakeholders	Test engineer and management
Priority	3
Inputs	Respond from PLC
Outputs	Question to PLC if still alive
Action	If all systems communicate as they should the system will continue running. If a PLC doesn't get any data sent from the PC the PLC will shut down. If the PC doesn't get any data from any PLC continuously the PC will stop remaining PLCs.
Requirements	Continuous communication between the modules.
Side Effects	The system becomes more independent of the test engineer.
Requirement class	Efficiency requirement
Verification method	Evaluation

Pressure guard

Function	The pressure guard will alarm and stop the compressor when the pressure in test object have reach a to high level to avoid damage the system
Description	The pressure guard will activate if the pressure in the piping between the condenser and evaporator reaches a certain level set by the user.
Stakeholders	Test engineer and management
Priority	1
Inputs	Readings from the systems pressure sensor and a safety level set by the user
Outputs	A stop signal to the compressor motor to shut down fast
Action	If the level is reached the system will alarm the user by showing a popup message and shut down the compressor motor to avoid any damage. The user needs to confirm the popup message before being able to continue operating the system.
Requirements	The function need readings from the pressure sensor continuously and the possibility to control the compressor motor.
Side Effects	Risk of false alarms
Requirement class	Safety requirement
Verification method	Evaluation, a simulated value will be sent to the system to test if its react as it should.

Compressor RPM guard

Function	If there is a transmission failure between the compressor motor and the compressor it will stop the compressor motor to avoid damaging the motor.
Description	The transmission is based on a belt and might break or fail during the testing process. There is one pulse sensor for the motor rpm and one for the compressor rpm, if these are not indicating a correct rpm this indicates a failure.
Stakeholders	Test engineer and management
Priority	1
Inputs	Readings from the compressor rpm sensor and the compressor motor rpm sensor.
Outputs	A stop signal to the compressor motor to shut down fast
Action	If there is a failure the compressor motor shuts down and sends an alarm message to the user interface. When the alarm message has been accepted by the user the system will be able to operate the system again.
Requirements	The function need readings from the rpm sensors continuously and the possibility to control the compressor motor.
Side Effects	Risk of false alarms
Requirement class	Safety requirement
Verification method	Evaluation, a simulated value will be sent to the system to test if its react as it should.

Two configurable alarms

Function	The alarms should be connected to a data reading alarming if a certain level is reached
Description	A channel of data should be picked and connected to an alarm level. The user should also have the possibility to connect a message to the alarm.
Stakeholders	Test engineer and management
Priority	3
Inputs	Any data channel
Outputs	Message on the user interface screen
Action	The user will be informed about the alarm by a popup message
Requirements	Possibility to reach the data channels
Side Effects	Risk of false alarm or configuring alarms incorrect
Requirement class	Safety and usability requirement
Verification method	Evaluation, a simulated value will be sent to the system to test if its react as it should.

Updating PID control parameters

Description	The parameters in the old system are not updated and optimized. When introducing the new system these parameters shall be updated to reach steady state in the system faster during a testing session.
Requirement class	Efficiency requirement
Stakeholders	Test engineer and management
Priority	2
Verification method	Evaluation by test engineer if its faster

Configurability

Possibility to reach and configure PID parameters

Function	The different PID-control loops should be accessible for configuration
Description	The different PID-control loops will be placed in the different PLCs making them hard to reach with connecting a computer directly on the PLC hardware. To avoid this, the user interface shall be able to retrieve and send new parameters to the different control loops easily.
Stakeholders	Test engineers and maintenance engineer
Priority	2
Inputs	The PID parameters stored in the PLC memory
Outputs	New PID parameters sent to the PLC memory
Action	The new configuration of the PID parameters will hopefully enhance the control loop.
Requirements	Access to the PLC memory
Side Effects	Wrong data can be sent to the PLC.
Requirement class	Usability requirement
Verification method	Evaluation, retrieve and send data through the system.

Gearing ratio between motor and compressor

Function	The gearing between the compressor and the motor needs to be configured for each test
Description	The transmission between the compressor and the motor can differ from test to test due to different wheels on the compressor. This needs to be set to be able to control the rpm of the compressor.
Stakeholders	Test engineer and maintenance engineer
Priority	2
Inputs	Current value of the gearing stored in the LabVIEW software
Outputs	A new value on the gearing ratio stored in LabVIEW
Action	The calculation of the compressor rpm will be correct
Requirements	Access to the LabVIEW data based
Side Effects	None
Requirement class	Functional requirement
Verification method	Test of configuring gearing and set a rpm and verify on the hardware with a stroboscope

Future configurations

Preparing for future ventilation control on condenser tunnel

Description	There is a wish of introducing a controlled ventilation of the condenser tunnels air with a fan or variable orifice. Space in the LabVIEW GUI will be saved for this function.
Requirement class	Implementation requirement
Stakeholders	Test engineer and maintenance engineer
Priority	3
Verification method	Evaluation

Preparing for introduction of more humidity sensors in the evaporator and condenser tunnel

Description	In the current solutions the value of the humidity is picked from CoolSeq and will be written by the test engineer to the user interface manually. To reduce this source of error and increase level of automation in the system there is a wish to introduce new humidity sensors connected directly to the user interface. This will be prepared for in the code.
Requirement class	Implementation requirement
Stakeholders	Test engineer and maintenance engineer
Priority	2
Verification method	Evaluation

Calibration

Possibility to calibrate in the user interface

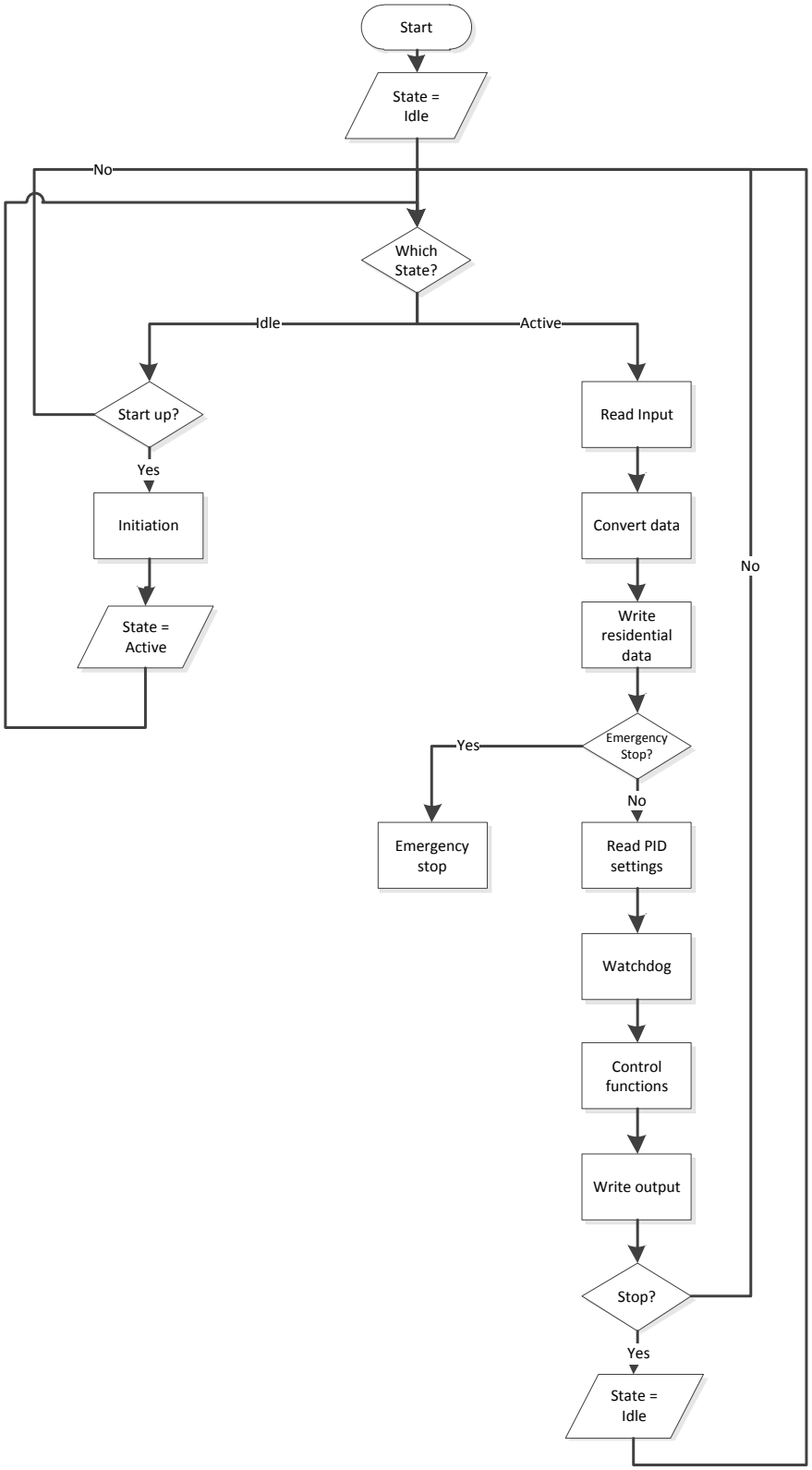
Function	Calibration of different sensors should be possible through the user interface.
Description	The user interface can retrieve and send calibration data from the different PLCs. The system should be able to handle calibration polynomials. This to avoid the need of accessing the PLCs externally to change the calibration settings.
Stakeholders	Test engineer and maintenance engineer
Priority	2
Inputs	Table of the calibration data stored in the PLC
Outputs	Calibration data to the PLC and to a file outside LabVIEW on the PC
Action	The calibration of the sensors will be updated and store the calibration data externally on the user interface PC.
Requirements	Access to the PLCs memory and the possibility to write to an external file.
Side Effects	None
Requirement class	Functional and reliability requirement
Verification method	Evaluation

Access of historical calibration data

Function	The system should provide the possibility to see the history of each sensors calibration.
Description	The calibration data for each sensor is stored externally and are accessible through the user interface showing the history of the calibration.
Stakeholders	Test engineers and maintenance engineer
Priority	4
Inputs	From each calibration update the polynomial or table is stored in a history file
Outputs	A table of polynomial or tables
Action	The history of the calibration will be presented giving a hint of when it's suitable for next calibration and if the sensors are stable of not.
Requirements	Access to calibration data and storage of data
Side Effects	Risk of data loss when stored externally
Requirement class	Functional and reliability requirement
Verification method	Evaluation

Appendix F

PLC Main loop flowchart


Start
State = Idle
No
Which State?
Idle
Active
Start up?
Yes
Initiation
State = Active
Read Input
Convert data
Write residential data
Emergency Stop?
Yes
Emergency stop
No
Read PID settings
Watchdog
Control functions
Write output
Stop?
No
Yes
State = Idle

Appendix G

PLC code Main loop

```
(* Main loop at 5 Hz *)

CASE State OF

0:  (*Initial state - Wait for start up*)
BoolArrayReader(); (*Check if start bit is activated*)

IF Start =TRUE THEN
(*Set initial values*)
InitialValues(); (*Puts all residential data in "working memory" which then allways uppdates
the residential data during run in UpdateToResidentData(). *)
State:=10; (*Got to next state*)
END_IF

10: (*Operating state - Main Program *)
(*Read input*)
AnalogInputSection();
DigitalInputSection();
BoolArrayReader();

(*Update data functions*)
UpdateToResidentData();
PIDSettings();

(*Safety functions*)
EmergencyStopCheck();
Watchdog();

(*Control functions*)
ControllerAirFlow();
ControllerAirTemperature();
ControllerHumidity();

(*Write to output*)
BoolArrayWriter();
AnalogOutputSection();
DigitalOutputSection();

IF Start = FALSE THEN (*Check if start bit is deactivated*)
(*System has been halted!*)
ResetVariablesAfterEmergencyStop(); (*Make sure all actuators are shut down*)
State:=0; (*Go to wait for start up state*)
```

```
END_IF

END_CASE
```