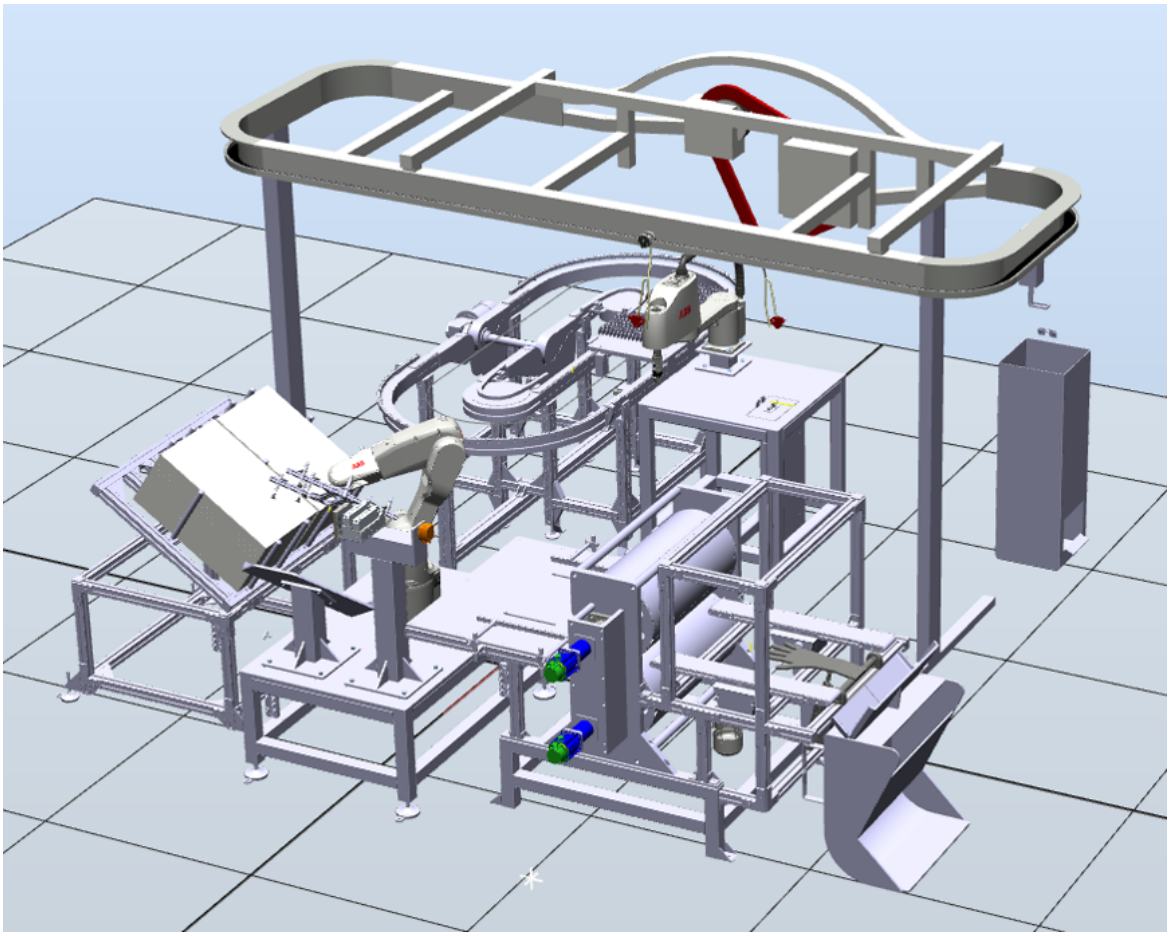

Virtual Commissioning of Smart Factory

Sanna Älegård and Stefan Knutsson EX047/2017

Department of Signals and systems
Chalmers University of Technology
Gothenburg, Sweden 2017

Master's thesis 2017:NN

Virtual Commissioning of Smart Factory

SANNA ÄLEGÅRD
STEFAN KNUTSSON

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Signals and systems
Chalmers University of Technology
Gothenburg, Sweden 2017

Virtual commissioning of Smart factory
SANNA ÄLEGÅRD
STEFAN KNUTSSON

Supervisor: Martin Fabian, Department of Signals and Systems
Examiner: Martin Fabian, Department of Signals and Systems

Master's Thesis 2017:NN
Department of Signals and Systems
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 76 296 3789

Cover: Overview of the virtual 3D-model showing the layout of the Smart factory modelled in RobotStudio.

Gothenburg, Sweden 2017

Acknowledgements

We would first like to thank Björn Magnusson, Magnus Seger and Anders Spaak at ABB for all the software support within RobotStudio and AutomationBuilder. They have been very helpful providing new useful components, licenses, software and overall help on technical questions regarding model building and connection to the PLC.

We would also like to thank our supervisors at ÅF, Niels Glamheden, Andreas Buhlin and Pär Ström for their support in the project. They have been very supportive providing general guidance on how to carry out and structure the project as well as moving us in the right direction regarding who to contact about different subjects.

From an academic point of view we would want to thank our supervisor and examiner at Chalmers, Martin Fabian for giving helpful feedback on the project and report. This has made sure that the report reaches a certain academic level.

Lastly we want to thank all the people at Smarta fabriker for giving us the opportunity to work with an interesting project like this. We would specifically like to thank Richard Hedman and Johan Bengtsson at GTC and Smarta fabriker. They have provided technical support, arranged meetings when needed and held all the thesis groups and the overall project together. They have always been available to answer questions and have provided weekly support. We have also been given opportunities to meet a lot of new people both within the academic world and industry.

Abstract

Virtual commissioning is a method to test and validate a factory, plant or line in a virtual 3D-model before physically building it. Incorporating virtual commissioning in automation projects, before physical commissioning, is a large and widely discussed research area today. This report presents the results and conclusions from building a virtual commissioning model of a miniature smart factory that has not yet been built. The model has been developed in ABB's RobotStudio and has been shown to be useful when evaluating the physical construction, robot paths and PLC-program. By testing the PLC-program in the model in an early stage of the project it has been possible to detect software errors that probably would have been found later at the commissioning stage, where mistakes are much more costly and there is a lot of pressure to start production as fast as possible. The report also covers the state of virtual commissioning including possibilities and challenges, by presenting a research study including interviews with experienced people within the subject. The study together with conclusions from developing the model shows that virtual commissioning is a topic that has a lot of potential but is still in an early stage with some challenges to overcome. The main challenges are that building a virtual commissioning model today is too time consuming and that the software available today need to be developed further for easier use and connection between the virtual PLC and the VC-model.

Keywords: Virtual Commissioning, Automation, Virtual model, Software-in-the-Loop, Emulation, Simulation, RobotStudio, Automation Builder.

Contents

1 Introduction

This section presents the background to this project and why it was initiated. Purpose, research questions to be answered in this thesis and limitations of the project are also introduced.

1.1 Background

With a market that is constantly changing and fluctuating, a lot of countries have developed some sort of strategy to raise their competitiveness. One example is Germany's Industry 4.0, where 4.0 refers to the fourth industrial revolution. With the strategy the goal is to develop a *smart factory* where everything is connected, closely related to IoT (Internet of Things). By having every product in the production line carrying information about where it is going and how, the factory can organize itself. The goal of Industry 4.0 is a production with shorter lead times, fewer errors and more flexibility (1). Sweden's counterpart to Industry 4.0 is called Smart industry and was initiated in 2016 (2).

Project Smart factories is a project initiated by GTC (Göteborg tekniska college) and is a collaboration between the industry and academy to contribute to the Smart industry strategy. The project aims to build up and spread knowledge about industrial digitalisation. This is done partly by building a demonstrative miniature smart factory as an exhibition at Universeum Science Centre. The factory will be used to show how technology and digitalisation can be used in production. The project of building the factory is divided and performed by students at different levels as thesis projects. The different projects include factory layout, programming of the automated process, electrical design, building the physical factory and to perform a virtual commissioning to name some. The lastly mentioned project, virtual commissioning, is the project that is covered in this thesis.

A virtual commissioning involves the development of a dynamic 3D-model of the factory which acts as a virtual twin. In a virtual commissioning the entire factory is tested in a virtual environment by connecting the model to a PLC (programmable logical controller). Virtual commissioning has been a growing research area the past decade and the benefits of it compared to the conventional way with only a physical commissioning are many (3). The code can be tested and debugged before even starting to build the factory and a lot of time that would be spent on-site in the conventional process, is saved. This decreases the commissioning time and cost significantly. The PLC- and robot-program used in the virtual environment is the same as the one used in the physical factory so once they are validated the idea is to just transfer it to the physical factory when it is built. In experiments described in (4) it has been estimated that by including a VC (virtual commissioning), it is possible to save 10-30% of the commissioning time and increase the quality of engineering solutions.

1.2 Purpose

The purpose of this MSc project is to perform a virtual commissioning of the Smart factory, where the actual PLC- and robot-code interacts with a simulated model of the factory. The main responsibility for the authors of this MSc thesis is to build the model, to write the robot programs and perform simulations, whereas another thesis group is in charge of writing the PLC-program. Ideally by performing a virtual commissioning the commissioning time will be short, compared to the conventional way with only a physical commissioning, leading to less costs. Another important aspect of a virtual commissioning is safety since different scenarios and robot paths can be tested in the virtual environment. Furthermore this project will lead to more knowledge about virtual commissioning applied to a real factory start up.

The main goal of this MSc project is to build a virtual commissioning model of the factory that can be used to test the construction, robot paths and PLC code. The purpose is to show how a virtual commissioning of a small scale factory can be done and to show the advantages and disadvantages of doing a virtual commissioning today. The virtual factory will be used as a demonstrator to illustrate how digitalization can be used and is thus an important

part when increasing knowledge about the area.

1.3 Research questions

Several research questions naturally arise in this context. The most important of those are:

- How should a Virtual commissioning model be built in an efficient and structured manner?
- Which are the main components and which details can be left out in the virtual commissioning model?
- What are the most important parts to test in a virtual commissioning? How should the PLC code be validated?
- Which are the major challenges when performing a VC today? What needs to be developed further and what has to happen before VC will be common practice?

1.4 Delimitations

In this project the following delimitations had to be made:

- The physical commissioning will lie outside the time scope of this project and evaluation of it will therefore not be a part of the project, even though that could yield in some interesting input to the results and discussion.
- This project only focuses on one specific software for development of the model, namely RobotStudio and thus comparisons to and/or evaluations of other software will not be within the scope of this project.
- The design choices of the factory is not to be evaluated or decided by the authors of this thesis. Simulation is usually used to decide on things such as placement of different parts and optimizing regarding different parameters, but this will at most be a very small part of the project.
- Servo drives and communication modules will not be included in the simulation model.

2 Theoretical framework

This section presents the background theory behind different virtual commissioning concepts and the steps needed to perform a virtual commissioning. Different strategies and concepts on how to plan a project in general and a virtual commissioning project specifically are also presented which are used as a base for planning of this MSc project. Since programming of the robots is also within the scope of this project, some background theory of robots and robot-programming is also presented.

2.1 Virtual commissioning concepts

Virtual commissioning (VC) is a way of testing the control program and the construction of a manufacturing system in a virtual environment, before physical commissioning (5). The concept has been a research area for several years and is a growing topic among manufacturers around the world. The expected benefits from performing a virtual commissioning are less debugging and correction during physical commissioning, which is usually a critical and stressful period of a project. Mistakes found during commissioning, either in the construction or control program, could result in high costs and long commissioning times.

In (6) a VC project is identified as three interconnected subsystems: 1. the mechanical design (including actuators, sensors and behavioural description of the system), 2. the control system, including input and output signals and 3. the connections between the sensors/actuators and the control.

There are four different constellations when performing a commissioning of a plant (5):

1. real plant and real controller, the traditional way of testing the control program during physical commissioning.

2. real plant and virtual controller ("Reality in the loop")

3. virtual plant and real controller, usually referred to as HiL (Hardware in the loop), where the real controller is needed in advance but a VC can be performed before building the physical plant.

4. virtual plant and virtual controller, usually referred to as SiL (Software in the loop), where no real controller is needed in advance.

When referring to virtual commissioning, one of the constellations 3 or 4 in the list above is used.

To perform a virtual commissioning, a model of the system is needed which has to be connected to either a simulated or real controller, usually a PLC (programmable logical controller). One way of designing the virtual model is described in (7) and consists of splitting the model into two separate parts: a physical model (geometric model with kinematics for programming of motions) and a logical device (a behavioral model that interacts with the PLC). This makes is possible to have a concurrent design process and according to (7) the design process consists of four steps; 1) process design, 2) physical device modeling, 3) logical device modeling, and 4) system control modeling. The process design step is done first, and the other steps can be done concurrently.

2.2 Planning strategies

One type of planning strategy widely used today in product development as well as in many other situations is Visual planning, which originates from the lean philosophy (8). The lean concept has its roots in Japan and the Toyota production system where the main goal is to reduce waste, where waste in this context can refer to loss of time and resources for example. The most used visual planning tool originates from the Toyota production system

and is a planning tool that is easy to use and where tasks and deliverables are organised in time and shown on a physical board. Regular meetings are held to update the involved people on progress and delays (9). One of the pros of using visual planning is that the involved people get a better understanding of what is happening in the project. Furthermore it has been shown that a higher degree of cross functional collaboration has been achieved together with a better workload distribution among the involved (9). The cons on the other hand are that it can be hard to see causal links between tasks and/or deliverables and that all involved have to be quite centralized geographically for this planning strategy to work (9).

In today's society, an increasing amount of components and systems are digitalised to be accessed from anywhere through the internet. This is true also when talking about planning strategies, where various planning tool software have been developed, based on the principles described above. Instead of having a physical board, the board is now digital and online, but with the same appearance and functionality as the physical counterpart. One pro of using a digital board compared to a physical, is that you can store history regarding for example late deliveries, which is harder when using a physical board. Another advantage with the digital board is that you can update the board from anywhere at any time, which can be convenient when working in larger projects. A digital board makes the planning activities accessible for all people involved from their own computer.

2.3 Data collection

Data about the plant is required to be able to build the virtual commissioning model, but also to test the validity of it. Data collection includes identification of relevant input parameters to the model, collecting information to represent these parameters, converting raw data, and documentation for future use and reference (10). The data can be divided into three categories: A) Available, B) Not available but collectible and C) Not available and not collectible (11). Category C data can be estimated but it is important to be careful when handling data of this category. One useful technique when the data is estimated is to apply a sensitivity analysis to it.

Data that is required to perform a virtual commissioning are in general (6):

- Extended 3D model including geometries, kinematics, electrics, electronics and controller programs.
- Layout with exact placement of equipment.
- Material flow involving SOP (Sequence of operations) and dependencies between different processes and operations.
- Control system. This may be the real control system (PLC) or an emulation software to test the model with.
- Detailed information about the inputs and outputs of the plant.
- Definition of extra functionality such as a safety system.
- Communication protocols for example between the control system and the simulation model.

2.4 Robots

The first industrial robots were introduced in the 1960's, and today they are becoming increasingly common in industry and other fields such as health care (12). The definition of a robot is according to (12) a mechanical manipulator that is flexible and can be re-programmed to perform different tasks. In (12) it is mentioned that there are two main parts in the definition: 1) a mechanical manipulator and 2) it is flexible and able of performing a variety of different tasks. One reason why robots today is used instead of humans are health issues, where for

example robots can work in places containing radioactive radiation or in other environments that would not be possible or healthy for human beings. Another aspect of this is monotone or heavy tasks where operator injuries may occur (12). Another reason why robots are becoming increasingly used is because of higher quality- and production requirements, where the robot can be superior to a regular worker since they never get tired and they can repeat the same task over and over again, with high accuracy and repeatability. Of course some of the properties that make the robot superior also make it flawed in certain situations. A robot only does what it is programmed to do, which is good in most cases but if something unpredictable happens the robot cannot always cope with it and it is believed that a human worker generally has an advantage in those situations (12).

2.4.1 Mechanical configuration of robots

A robot consists of links and joints where there are two different kinds of joints: *prismatic* joints move in a linear motion and *rotational* joints that move in a rotational manner between two links. The links are the static structural parts that acts as a connection between the joints. The energy needed to make a robot move the joints is most commonly electrical, but pneumatic and hydraulic solutions also exist (12). Robots are often split into five main categories which are cartesian, cylindrical, spherical, horizontal articulated and vertical articulated robots. How a robot is categorized depends on which type of joints the robot consists of, the orientation of the links as well as the order of the joints (12).

2.4.2 Controller

The controller of the robot acts as the brain of the system and is where the signals from all the encoders positioned in the joints are gathered to get position and orientation. The controller calculates the end-effector position and orientation by using the orientation of the single joints, the length of the links between the joins and the assumption that the links between the joints are rigid (12). That the joints are assumed rigid is a simplification and means that no bending occurs in the links. Two of the reasons why it is possible to consider the links between the joints to be rigid are that most of the industrial robots today still have links that almost experience no deflection within their specific load capacity and because the error margin that you get by deflection of the links is so small compared to the joint stiffness (13).

2.4.3 Movements

Robots mainly have two strategies of moving, where the first is called point-to-point movement. The controller executes this movement by calculating how it needs to reconfigure the joints to reach the target point. Depending on the controller manufacturer and what kind of code they have implemented, the controller either calculates first how fast each of the joints need to change so that all joints finish at the same time, or it does not do this and the joints will in most cases finish their movements at different times (12). One thing to keep in mind with the basic point-to-point movement when different speeds are used for the joints to make them finish at the same time, is that for short distances the robot moves more or less linearly. The other strategy for moving the robot is by continuous path control and here the robot should follow a defined path between the points, managed either by defining many points along the path with a small distance between them or by using continuous interpolation to make the robot follow the defined path (12).

2.4.4 Singularities

A problem that may occur when programming robot paths is singularities. There are different kinds of singularities where some depend on joints aligning, for example if joint number four and joint number six align there will occur a singularity problem, sometimes referred to as wrist singularity (14). The problem that occurs when joint number

four and joint number six align is that the joint movement does not have a unique solution (the result of a rotation of 7 degrees in axis four is the same as a rotation of 7 degrees in axis six) and can be countered by an opposite movement in one of the other axis (14). Other singularities that may occur is for example that a very small movement will require infinite speed of a joint or that the robot tries to reach outside its work area. What typically happens when moving into or close to a singularity is reduced movement and can lead to a reduction of speed since the movement in an arbitrary direction cannot be done. This problem is also known as losing a degree of freedom (15; 16).

2.5 Modelling

Building up the virtual factory or modelling the factory consists of two phases called high level modelling and low level modelling, described in this section.

2.5.1 High level modelling

High level modelling is the process of composing the virtual commissioning model from already created mechatronic parts. These parts are called low level components and should be fully functional with the necessary sensors and actuators. If a library with all necessary low level components already exists, the creation of the VC-model is fairly easy, with some additional effort on setting up an I/O-system and dragging the low level components into the correct place in the 3D-environment, according to the defined layout. Robot models is an example of usually pre-existing low level components created by the vendor, that can be imported into the high level model.

2.5.2 Low level modelling

Low level modelling is the creation of mechatronic models based on CAD data created either by the manufacturer or the plant designers. The low level modelling can be divided into three different stages: geometrical modelling, functional modelling and electrical modelling (5).

Geometrical modelling
After the CAD data has been imported to the simulation software, some adjustments might be necessary as a first step. When importing CAD data into the simulation software, the model might be unstructured and thus the first step is to create an appropriate hierarchical structure for the model. This involves manual structuring of CAD data into objects, sections and components. Another step that needs to be taken either before importing the CAD data or inside the simulation software is simplification of CAD parts. Usually the CAD components are unnecessarily detailed for a simulation which can cause too much load on the CPU and inaccurate simulations. Some simulation software has integrated simplification tools but if it is not sufficient manual work may be necessary to remove selected unnecessarily parts of the models. Features that may be removed to reduce the complexity of the model are holes, bosses or labels whereas parts that may be removed from the model are small or other selected parts and hidden invisible parts (5).

Functional modelling
The functional modelling step includes adding kinematics and sensor functions. The software usually has integrated functionality to add movements such as translations, rotations and gripping or sensor functions of different variants such as optical, inductive, capacitive or light barrier. The sensors and actuators have to be manually placed on specified geometries. The functional modelling step results in functional parts such as cylinders, sensors, tools such as grippers etc.

Electrical modelling
This is the last step of the low-level modelling and results in a complete mechatronic model including inputs and outputs on actuators and sensors. These outputs are to later be used when communicating with a control program.

The software may have a graphical interface, making it easy to add and edit the signals and get an overview of the design.

2.6 Communication

During the late 1970's and the beginning of the 1980's protocols and means to connect automation devices where developed. In the beginning, most protocols were brand specific meaning that only devices from the same brand could be connected to each other. This was seen as a problem and a need to develop a generic way to be able to connect devices regardless of the manufacturer arose. In the late 1980's this was realized by using the same local area network used by regular computers to also connect different automation devices regardless of manufacturer by development of the Open System Interconnection (OSI) standard. OSI is not a communication protocol itself but defines a standard framework for communication protocols (17). New communication protocols began to develop fast and became widely used to be able to decentralize bigger systems and by this being able to have distributed nodes which for example leads to less wiring and easier rebuilds to name two of the benefits (18).

There are three main types of topologies for how a network between different nodes can be set up; star-, ring- and bus network. Depending of what kind of topology is being used there are pros and cons regarding for example redundancy. Using a ring network instead of a bus network would result in redundancy if the wire between two nodes breaks or is removed, since the signal can still make it to all the nodes and the network can in that case be seen as working like a bus network. On the other hand using a ring network as opposed to a bus network requires an extra wire from the "last" node to the "first". In Figure 1 the different layouts together with some more pros and cons can be seen (19).

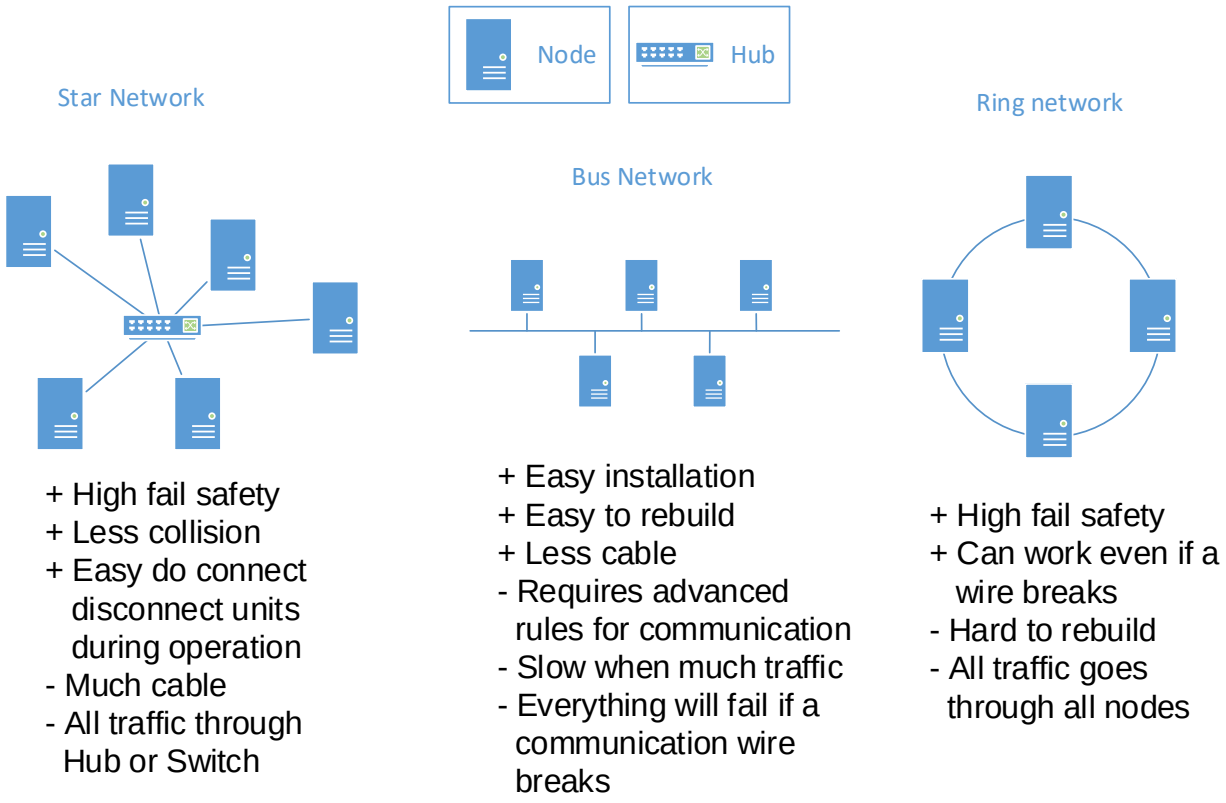

Figure 1: Different topologies with pros and cons.

In the beginning of the 21th century Ethernet based buses were introduced which were welcomed by the industry. Two of the reasons were because this new technology allowed the use of regular network cables and routers and access points could be used instead of more expensive special equipment. The performance also got better with the Ethernet based solutions and can nowadays reach speeds up to 1 Gbit/s (19). When mentioning different communication protocols and their speed, where speed usually refers to the response time, it is common to make a distinction between soft and hard real time. Hard real time has a faster response rate than soft real time and when hard real time communication is used no packet loss is allowed whereas in soft real time some packet loss is allowed (20). Another definition is that hard real time signals must be correct and meet the deadline every time and soft real time is when the deadline can be missed occasionally without making the system unreliable (17). In Figure 2 the difference between TCP/IP, traditional fieldbuses and Industrial Ethernet response time range can be seen as well as an approximation of the difference between soft and hard real time in relation to response time.

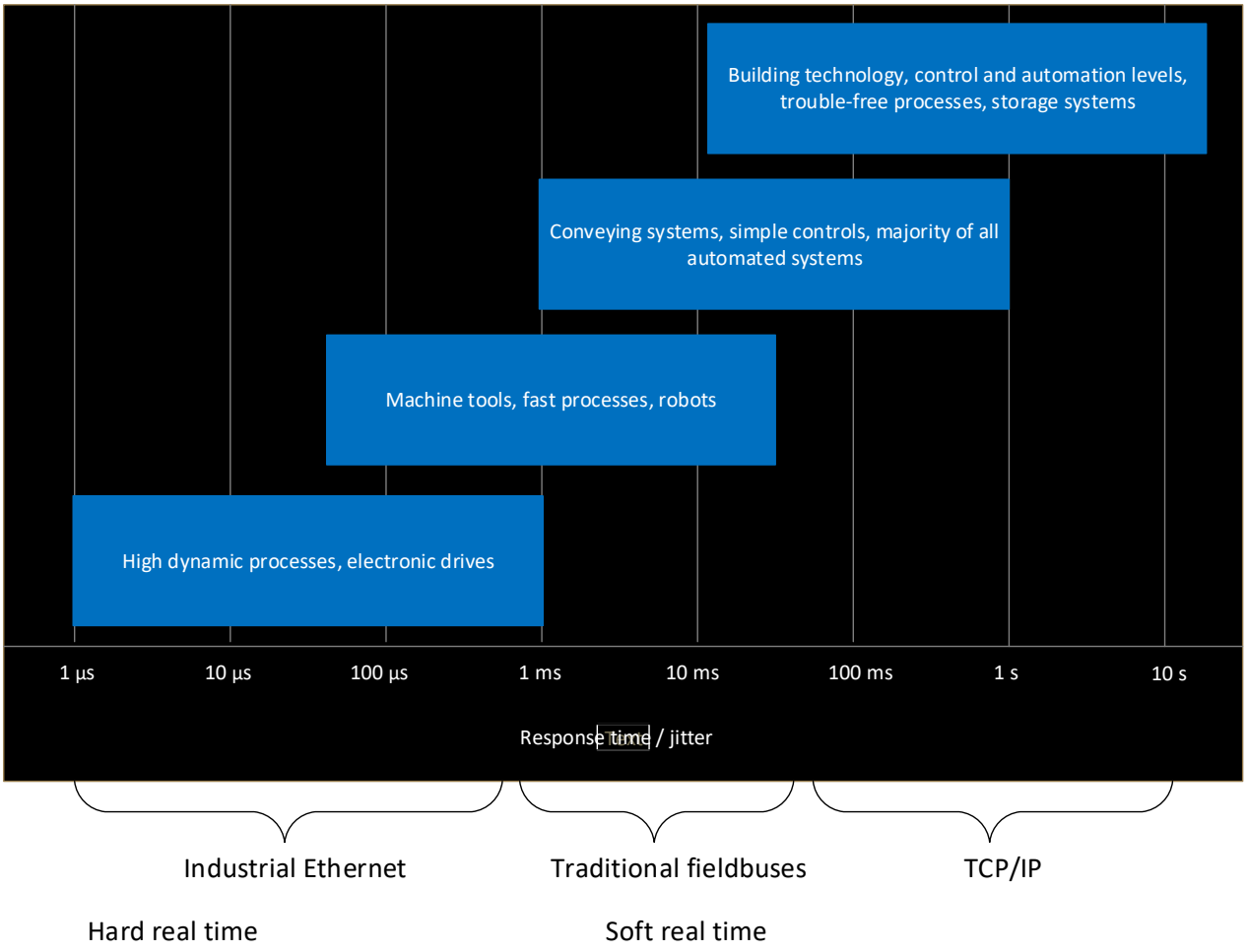

Figure 2: Response time comparison between different communication protocols.

When Ethernet based buses became common, new protocols had to be developed where one of the main reasons was because the common measures to avoid data collision inside the Ethernet standard caused irregular time delays (21). Three common Ethernet based buses used today are EtherCAT, ProfiNet and EthernetIP. When using Ethernet based buses one benefit is that it is usually quite easy to integrate it into the regular network. ProfiNet for example has three different communication services, TCP/IP for communicating with higher level systems, ProfiNet RT to communicate in real time with other automation applications and lastly ProfiNet IRT which is the fastest service with a response time in the sub-microsecond range (22).

2.7 Verification and validation

Verification and validation is needed to assure that the simulation model has a certain level of accuracy depending on the demands of the model. To differentiate between verification and validation two usual definitions are:

- Validation: are we building the right system?
- Verification: are we building the system right?

This means that verification is a way of assuring that the model is correct according to some sort of specification and validation is a way of assuring that the model is accurate enough according to the user demands of the model. This also means that validation is a way of testing the specification itself and that the user defines if the model is satisfactory. There is thus no certain way of knowing when a model is validated but it is more about reaching a certain level of confidence that the model will serve its purpose. Balci O. says in (23) that model verification is 'ensuring that the computer program of the computerized model and its implementation are correct' and that validation is 'substantiation that a model within its domain of applicability possesses a satisfactory range of accuracy consistent with the intended application of the model'. In (24) Sargent R.G divides the system into the real world and the simulation world and describes how verification and validation is an iterative process closely connected to the model building process. Figure 3 shows a simplified picture of how the validation, verification and model building can be interconnected in a simulation project. For a more detailed version of Figure 3, the reader is referred to (25).

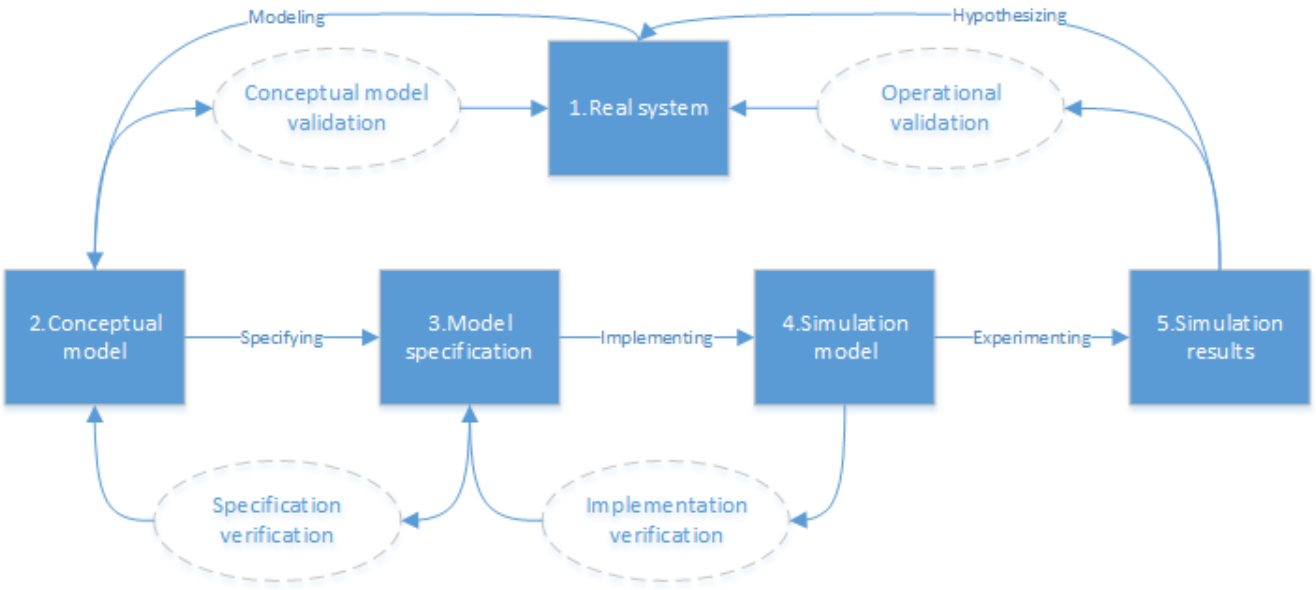

Figure 3: Presenting how model building, verification and validation can be interconnected in a simulation project.

The process is mainly built on five steps; modeling, specifying, implementing, experimenting and hypothesizing. The first step, modeling, is where a conceptual model is developed based on the real system. The real system includes theories of the system and data about the system relationships, which the conceptual model is validated against in the conceptual model validation step. The conceptual model is a mathematical, logical or verbal representation of the system. A detailed simulation model specification is then created which thoroughly describes how the conceptual model should be implemented in a computer system and how the software design should look. The specification is verified against the conceptual model and when it is satisfactory the implementation process begins

where the simulation model is developed. The simulation model is the conceptual model implemented in a computer system and it is verified against the model specification. When the simulation model is satisfactory, experiments are applied to it to yield data which is compared to data from the real system in the operational validation step. The operational validation is performed to ensure that the model output behavior is accurate enough for the models intended purpose. The verification and validation process is iterative and repeated until a satisfactory model is developed.

2.7.1 Validation techniques

In (24) Sargent describes some of the most common validation techniques used for simulation model checking:

- Animation: The behavior of the model is graphically displayed and reviewed.
- Comparison to other models: one example is that simple results from the simulation model is compared to results from an analytic model.
- Event validity: does events occur as often in the model as they do in the real world?
- Extreme condition tests: the model should give a reasonable output result for any extreme case. For example if a production site simulation receives zero input material the output should usually be zero as well.
- Face validity: People that have deep knowledge about the system are asked whether the model and its behavior is reasonable.
- Historical data validation: If data exists parts of it can be used to build the model and parts of it used to compared the results from the model.

2.7.2 Acceptance tests

Traditionally in automation projects the testing is mainly carried out as a two-step process where the first step is a FAT (Factory acceptance test) which is an acceptance test in the development environment. The test is carried out by the supplier. The FAT is to make sure all components function and that all data sheets, manuals, electrical schemes exists. The FAT documents should fully cover the specification of the system. When the FAT is satisfactory a SAT (Site acceptance test) is carried out by the user on site after complete installation. The SAT should repeat the FAT to control that nothing has been damaged while shipping (26) and ensure that the factory is fully functional for start of production.

When including a VC in an automation project a third acceptance test should be included before the FAT and SAT, namely a VFAT (virtual factory acceptance test). The VFAT is carried out in the virtual environment to test the PLC-logic, placement of components, functionality of components etcetera. By including a VFAT a lot of software- and construction errors can be found before ordering parts and before commissioning. When the VFAT is completed the parts can be ordered and the factory built for FAT.

3 Description of the Smart factory process

The Smart factory will produce cardboard virtual reality goggles and consists of two production lines where one of the lines delivers plastic lenses and the other produces and delivers cardboard goggles. A conceptual model of the factory is shown in Figure 4.

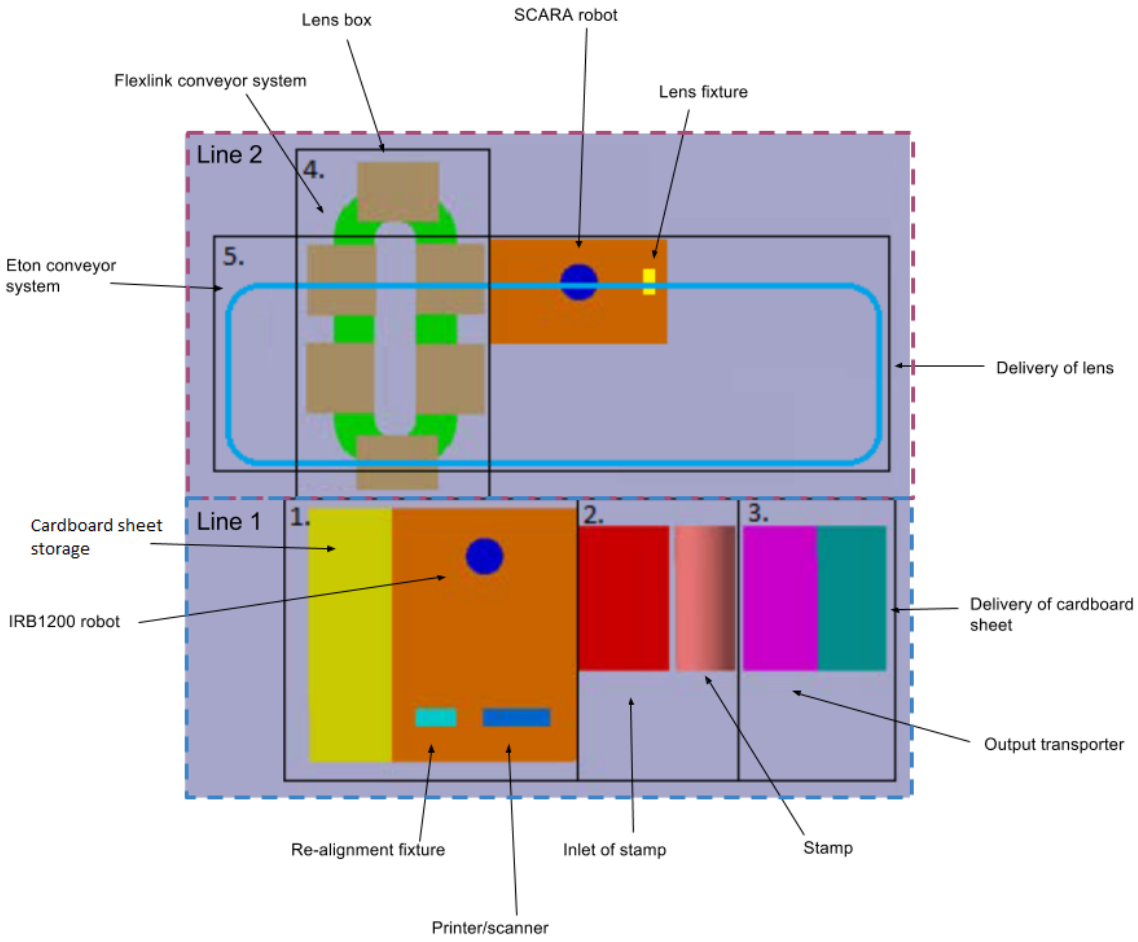

Figure 4: Conceptual model of the factory, as seen from above.

As can be seen each production line has been divided into modules with a total of five in the whole factory. Since the factory is only used to demonstrate technology, the process is not optimized for cycle time in any way. It is instead intended to be visual for the viewer with a lot of movements and actions, that may sometimes seem unnecessary. The two production lines are further described in the subsequent subsections.

3.1 Production line 1: production and delivery of cardboard goggles

Module 1: cardboard storage, robot, printer and scanner
The first module mainly consists of two storage units containing a number of cardboard sheets (approximately 600), one ABB IRB1200 robot, a printer and a scanner. The production of cardboard goggles starts with the robot, which is equipped with a suction cup tool, picking up a sheet from one of the storage units. The sheet is then transferred to a realign-station where the robot places the sheet in a fixture and performs a retake operation. The

retake operation and the fixture is used to ensure consistency of the orientation of the sheet for the subsequent operations. When the robot has performed the retake operation and the sheet is correctly oriented the sheet is transferred to the printing station. The robot moves the sheet three times with a linear movement in front of the printer, for the printer to print two QR-codes plus one additional print of text decided in advance by the customer. The QR-codes are printed to make the sheet act as an information carrier throughout the process, and the codes are therefore verified to be correct by the scanner before the sheet is passed on to the next module. If the QR code cannot be read by the scanner the sheet is scrapped and the customer gets to order a new product.

Module 2: input and stamp
After the QR codes have been printed and verified the robot transfers and places the sheet at the inlet of a rolling stamp which is where Module 2 starts. The inlet consists of an ultrasound sensor letting the PLC know that a sheet has arrived. When the sensor is active and the robot has moved away from the inlet, the cardboard sheet is pushed by an actuator into the rolling stamp where the goggles are cut out. The motors of the stamp cylinders are activated when the sheet has reached a second sensor close to the stamp. The cylinders act both as a stamp and a transporter of the sheet so that the rotation of the stamp cylinders will move the sheet forward to Module 3. The velocity of the inlet actuator has to be chosen carefully and cannot override the speed that the stamp will move the sheets with.

Module 3: output and delivery of cardboard sheet
After the sheet has been stamped it is transported by a conveyor towards the delivery. At the start of the output transporter there is a third ultrasound-sensor letting the PLC know that the sheet has arrived which activates the transporter. At the end of the transporter there is an actuator and another sensor telling the PLC that the sheet has arrived at the actuator. The transporter is then turned off and the actuator flips the cardboard sheet into a delivery box where the customer may pick it up. The delivery box is equipped with a sensor to make it possible to count the number of sheets in the box. This is to be able to know if the box is full.

3.2 Production line 2: delivery of plastic goggle lenses

Module 4: Flexlink conveyor system
The delivery of lenses starts with a conveyor system delivered by Flexlink. It contains five pallets moving around on a ellipse-shaped conveyor. The pallets carry boxes containing 50 lenses each, positioned in three rows. The pallets, which are plastic, contains a number of metallic pieces mounted on the bottom which are used to activate inductive sensors that are positioned at certain positions of the conveyor.

The conveyor system contains two stops controlled by the PLC which are used to stop the pallets at specific positions. One stop is placed at the position where the box should be for the robot to pick a lens, and one stop is placed a short distance before the picking position so that the subsequent pallets will not hit the pallet that the robot is picking from. A collision could create some unwanted movements and the robot might not be able to pick the correct lens.

An inductive sensor and a fixing device is placed at the picking position. The sensor indicates that the pallet is in correct position and is thus used as a check before fixating the pallet. The fixating device raises the pallet a few millimeters from the conveyor belt and fixates it, so that no unwanted movements from the conveyor belt disturbs the picking operation. When the box is fixated the robot is allowed to pick a lens.

Aligned with the two stops there are two inductive sensors which are used when programming the Flexlink system. They can be used to trigger when to stop the pallets, to count how many pallets have passed a stop and so on. The control program is written so that after one lens has been picked from a box the pallets move one entire lap until the same box stops at the picking position again. This is done until the box is empty and then the process continues with picking the first lens in the next box. When all boxes are empty some manual re-filling of the boxes is needed.

Module 5: SCARA robot, Eton conveyor system and delivery The fifth module consists of a SCARA robot standing

on a table, a lens fixture on the same table which is to be moved up and down by an actuator and a conveyor system delivered by Eton systems which transfers the lenses to the delivery on a conveyor with a height of two meters.

When the box is in correct position for picking a lens and an order has been placed the SCARA-robot will receive a signal to pick a lens from the box. The robot uses an electrical clamping tool to pick the lens which is then placed on a fixture on the robot-table. When the lens is in place and the robot has moved to its home position, an actuator raises the fixture towards the Eton conveyor-system. This movement transfers the lens to one of the carriers hanging from the conveyor. The Eton system receives a start signal from the PLC, indicating that the robot is in a safe position for the carrier to move, and transports the lens to the delivery position. To receive the lenses the customer needs to scan the QR-code on his/her sheet and place his/her hands in the delivery station which will activate a sensor telling the Eton system to drop the lens. When the customer receives the lens the last step is an assembly station where the goggles will be manually assembled by the customer.

4 Methodology

In this section the methodology is presented that was followed to fulfill the goals and to answer the research questions of this thesis.

4.1 Planning strategy

To make use of all the benefits of virtual commissioning and be effective when building the virtual model, it has been important to work both together with the other groups and suppliers to gather information and specifications. To spread and acquire information to other groups, a common vision and plan is required.

4.1.1 Visual planning

This MSc project is a part of the larger scale project Smart factories, which includes several other thesis workers, companies and institutions. All thesis workers are more or less dependant on each other and thus a good planning tool is necessary to work efficiently. Project Smart factories utilizes a planning tool called YoLean based on the LEAN concept. It is a visual tool where every user puts up digital post-its on a digital board. The post-its have different colors depending on its character, i.e if it is a milestone, activity, a meeting or something else. The activities should be possible to carry out in 3-5 hours and they should be planned one week ahead while the milestones are larger tasks and may be planned to be finished further away in time. The activities may also be connected to a specific milestone. The board is open to everyone involved in the project which makes it easy for groups depending on each other to plan together. The concept is also based on weekly meetings where everyone involved meets to discuss how the past week went, what should be done the next week in terms of activities and milestones and what everyone needs from each other to reach their goals. This tool makes it easy to discover and correct problems early and make the project work more efficiently.

The authors of this thesis have been mostly dependant on one more thesis group, namely the group responsible for the PLC program. Therefore at the start of the project it was important to set up milestones regarding the practical work together with them. The smart factory was in the beginning of the project divided into different modules and sections, making it easy to set a milestone for when each module should be done by each group. Milestones regarding the report was also included in the planning tool even though it is not relevant for other groups.

4.1.2 Project plan

The beginning of the project consisted of a planning- and design-phase where data collection and decision making about the factory process flow was done. Everyone involved in the development of the factory need to have the same information so a mutual document folder including all the documents describing the system was needed, which had to be continuously updated throughout the project. The document folders include I/O-lists, flowcharts, layout plan, CAD-parts, component lists etcetera. This was the first phase of the project, the design phase. Next phase was the modelling phase where the static model is further developed to a dynamic model, module wise according to the modules in Section 3. The parts need to have the correct input- and output signals according to the design plan and I/O-lists. Following the modelling phase was the testing phase where the factory was tested through simulations. This was done throughout the entire project by testing each part separately early on and then together with other parts later on and lastly with the real and final PLC code. Figure 5 shows a principal plan of how the project has been performed. The blue boxes are tasks that involves the authors of this thesis whereas the grey boxes are tasks performed by other people.

Figure 5: Principal chart of the planning process of the project.

Throughout the entire project the design, modeling and testing phases have been concurrent and iterative so that the factory has been tested part by part and if a test reveals that some part of the design needs to be changed we are back in the design phase, making changes in the design, modelling it and testing it again. The PLC programmers have been working in parallel with the modelling and testing, to increase the efficiency of the project, both by the ability to test parts of the PLC code early on and by having a platform, the dynamic model, for common understanding of the process among the groups. After the programming and testing is done and satisfactory, the factory is to be built.

4.2 Data collection

The Smart factory has not yet been built and data about the factory was therefore gathered via technical specifications, asking companies about their products and at some points where data could not be found it was estimated/guessed. The layout was handed out in an early phase of the project and has since then been constantly updated, which has required consistent communication to make sure the latest version of the layout and component list is available. Due to layout changes there has been a lot of re-work when building the virtual commissioning-model.

4.3 Robot programming

This section describes how the robots in the factory have been programmed and how they communicate with the PLC.

4.3.1 IRB1200

The IRB1200 is a 6 axis robot, in this project used for handling the cardboard sheets. The process consists of picking the sheet from one of the storages, performing a re-take procedure, passing the printer and scanner and then placing the sheet at the input of the stamp. If one of the QR-codes shows to be faulty the sheet is instead placed in a scrapping bin. A flowchart of the process can be found in Appendix A. Figure 6 shows a 3D view of the robot together with the surrounding environment.

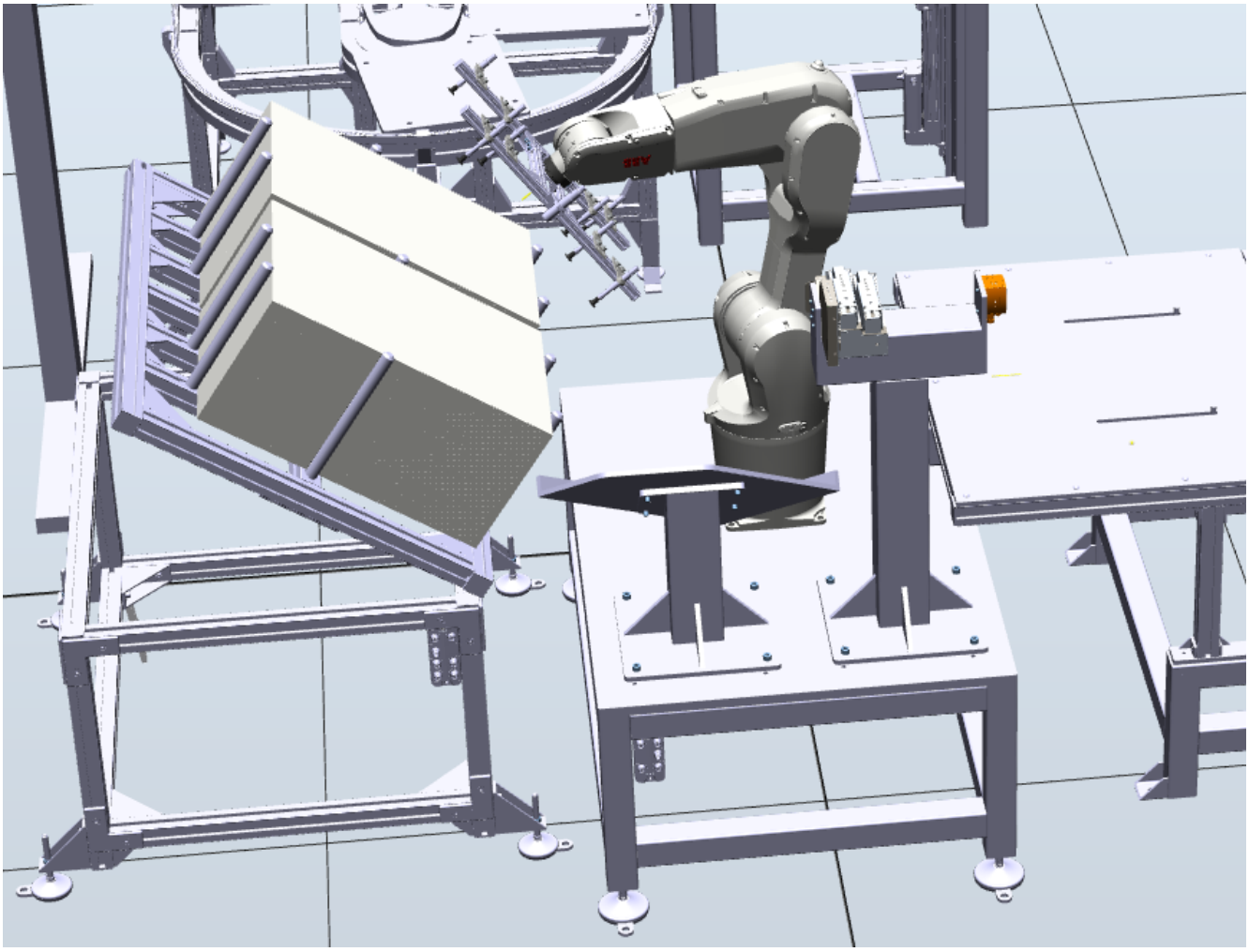

Figure 6: 3D view of robot together with its surrounding environment.

As can be seen there are quite a lot of components in a small area and it is important to verify that the placements of all components are valid. More specifically it is important to make sure that the robot can reach all of its targets without any collisions and without moving into singularities. RobotStudio has a feature called collision checking where different collision sets can be defined which was used to validate the created paths.

The robot is equipped with a vacuum gripper used to pick up the sheets and an UGR500 ultrasonic retro-reflective sensor from IFM to measure the distance to the sheets while they are in the storage. A vacuum guard is connected to the tool and is used in the picking-operations, detecting whether the sheet is attached or not.

From the moment that the sheet is being transported from the storage until it is placed on the input of the stamp, a trap routine is active. A trap routine is a routine in the robot controller that monitors a condition for a specified time during operation and if during this time the condition for the routine is met the program will execute the set routine in the robot. In this case the IRB1200 will have a trap routine that monitors so that if the vacuum for the sheet disappears for some reason and the vacuum guard goes low the IRB1200 will stop executing and print an error message on the pendant. This is so that the program at any time will stop execution if there is no sheet to work with.

Work objects
A work object is a coordinate system, mostly used to simplify programming and jogging of the robot. A work object coordinate system consists of a user frame and an object frame where the object frame is related to the user frame and the user frame is related to the task world frame. When a work object is defined it is attached to the

specified object in the simulation model. Robot targets can then be defined within the work object that the robot is working on, and if the object in the model is moved, the work object and targets moves with it. This means that the targets will not have to be redefined every time some object around the robot is moved. When having work objects it is also possible to jog the robot within the work object coordinate system instead of within the world frame.

In this MSc project it was decided that four work objects were sufficient to have:

- Wobj_fixture
- Wobj_Printer
- Wobj_input
- Wobj_magazine

The work objects were defined so that if it is found later that the component attached to it has to be moved the targets do not have to be changed within that work object . This makes it possible to make smaller changes in the layout without having to re-program the robot, thus making the robot program more flexible for changes.

Handshake with PLC
In parts of the program where the robot has to interact with the PLC it is important to have a handshake procedure. A handshake is used to make sure that both the PLC and the robot has the same information before performing a task. The robot acts as slave and PLC as master, which means that the robot is passive, waiting for the PLC to start communication.

The handshake is performed before picking a sheet from one of the storages, before printing, before scanning, after scanning to see if a satisfactory print have been achieved and when the sheet is placed at the stamp input. Figure 7 presents a flowchart showing a general handshake procedure. It will be slightly different depending on which operation is going to be performed.

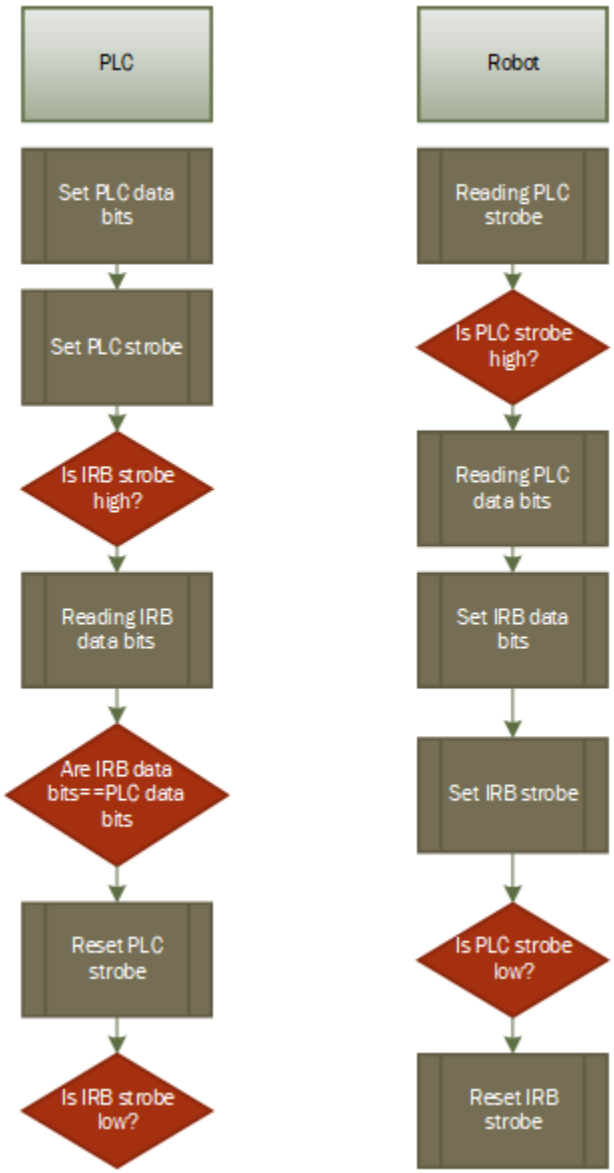

Figure 7: Flowchart showing the general handshake procedure.

The PLC data bits are used to indicate which operation is going to be performed, where the different operations is shown in Table 1 and the strobe is used to indicate when the robot or PLC is allowed to read the bits.

Value data bits	Event/Process
0	Pick left storage
1	Pick right storage
2	Reset offset picking position (if storage is refilled)
3	Printing
4	Scanning
5	QR code ok
6	QR code not ok (scrap product)
7	Robot moved from stamp input

Table 1: Presenting the different values and corresponding event/process of the bits used for communication between the robot and PLC.

There is one strobe on both the IRB1200 side and PLC side, and one set of data bits on each side as well. When the PLC strobe is high the robot is allowed to read the data bits. When the bits are read the robot sets the same bits and sets its strobe high indicating that the PLC can read the bits. If the PLC gets the same bits back as it sent it replies with setting the PLC strobe low. When the robot read that the PLC strobe is low, it resets its own strobe and starts with the process indicated by the data bits. The handshake is thus done.

4.3.2 IRB910

In Production line 2 a IRB910 Scara-robot is used for handling lenses. The movement consists of picking a lens from the box that is at the picking position and placing it in a fixture where it will be transferred to one of the Eton carriers. In Appendix B a flowchart of the IRB910 process is presented.

The SCARA-robot acts as slave in the same way as the IRB1200 and thus the PLC is master. A handshake is needed for the SCARA-robot before picking a lens and before putting it in the lens fixture, to guarantee that there will be no collisions between either the robot and the lens fixture or between the robot and one of the product carriers. The handshake is also used for the robot to know which lens to pick. The PLC keeps track of which lens position index, ranging from 1 to 50, that the robot should pick from. When the path is safe for the robot to move the PLC sets the data bits to the correct index and sets the strobe allowing the robot to read the bits. The different handshake operations for the SCARA-robot are shown in Table 2.

Value data bits	Event/Process
1	Pick lens at index 1
2	Pick lens at index 2
.	.
.	.
.	.
50	Pick lens at index 50
51	Put lens in fixture

Table 2: Presenting the different values and corresponding event/process of the bits used for communication between the SCARA-robot and PLC.

Handshake number 51 is finished when the robot has moved to a safe home position and triggers the PLC to send a signal to Eton that a carrier may be sent down to position for transferring the lens to the Eton system. When the carrier is in a proper position and the robot is in a safe position the PLC activates the lens fixture actuator which moves the lens fixture up to hand over the lens to the product carrier.

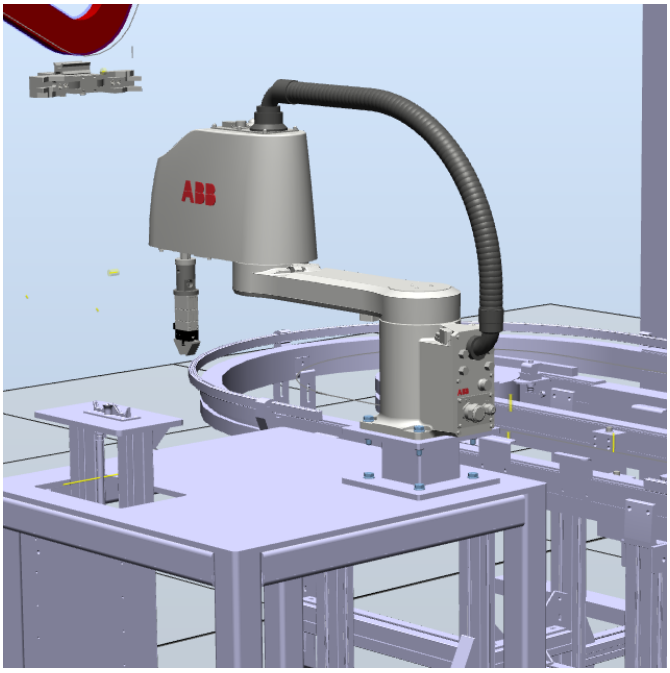

Figure 8: SCARA-robot and surrounding environment.

The robot program is not flexible for mistakes such as misplacement of lenses or missing lenses. This is because there are no sensors mounted on the tool. Sensors could detect if a lens has been picked up or to use a search operation to find a product. The program is thus based on exact placement of the lenses, but if a lens is missing in one of the boxes this will instead be indicated by an ultrasound sensor placed on the table next to the the lens fixture.

4.4 Modelling of Smart factory

As mentioned in Section 2 modelling is a big and crucial part of a virtual commissioning project and has thus been the part where the majority of the time and effort has been spent in this project. The modelling process is described in this section.

4.4.1 Low-level modelling

In this project the CAD-model was created and delivered by the mechanical engineers. The CAD-models were imported to RobotStudio where some parts needed transformation, to become dynamic/electrical parts. RobotStudio has a feature called SmartComponent which enables the user to create low-level components within the software. A smart component contains one or more child smart components such as logic gates (AND, OR, NOT etc), logical expressions, sensors of different kinds, attachers, detachers and manipulators among others. It is a combination of these that makes up the part or function needed for the wanted application. RobotStudio has a graphical interface where drag-and-drop programming can be done to easily be able to create I/O-signals and connect the different child components.

For the IRB1200 robot a suction cup tool is going to be used and thus had to be defined in RobotStudio for simulation. How this tool was modelled will now be presented to show an example of how modelling can be done in RobotStudio. The suction cup 3D-model can be seen in Figure 9. It consists of eight suction cups and a built in "vacuum guard" used for knowing if a sheet is attached to the tool or not.

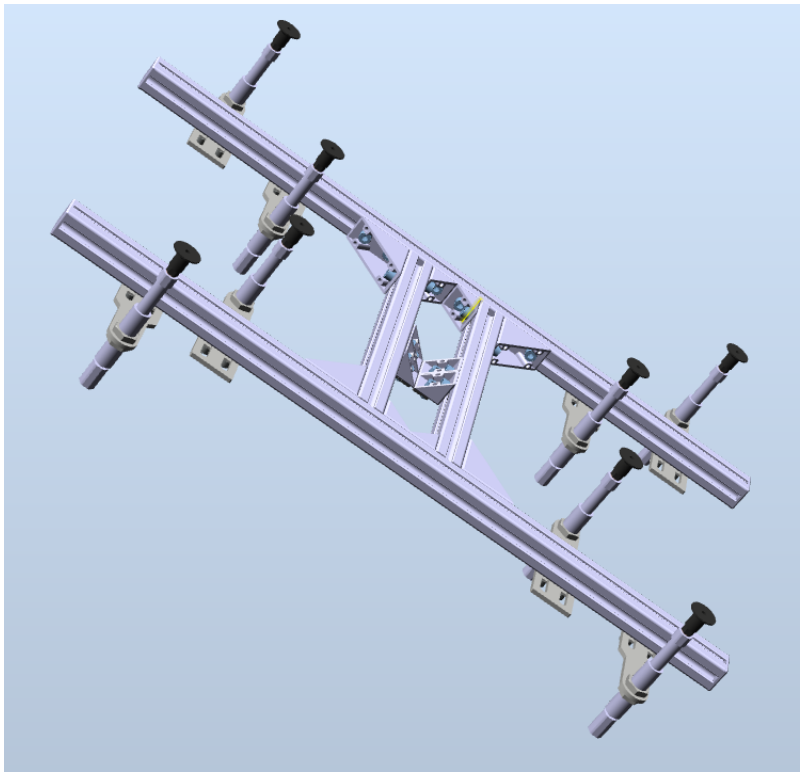

Figure 9: 3D-model of vacuum tool.

In the VC model the tool consists of an attacher to attach objects, detacher to detach objects, a line sensor attached to the 3D-model, used to simulate the vacuum guard and a SR-latch to set an output signal saying if a sheet is attached to the tool or not. The graphical design view of the vacuum tool can be seen in Figure 10.

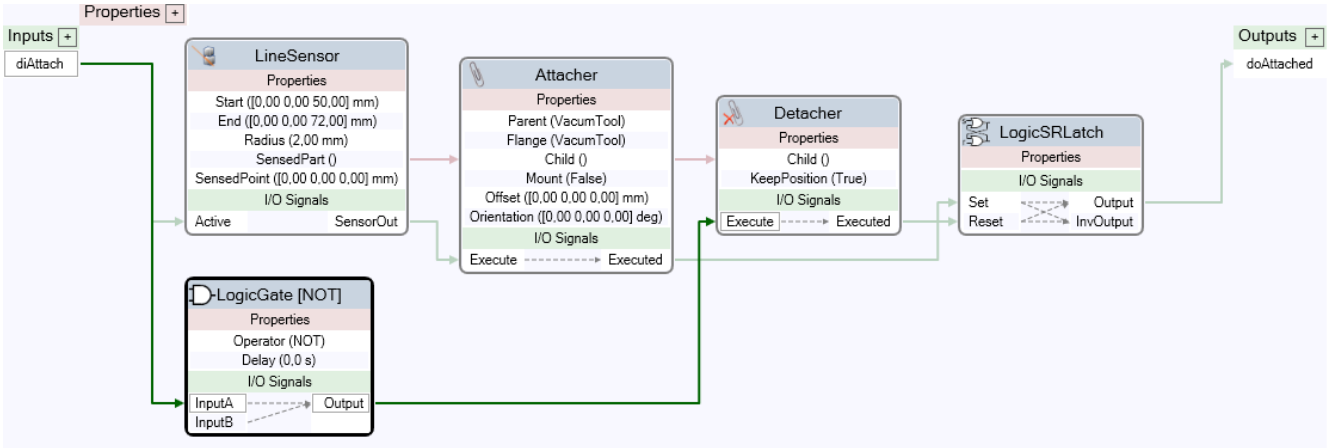

Figure 10: Internal logic of robot vacuum tool.

The Smart component's input signal, diAttach, is connected to a real output port on the robot internal DeviceNet device and the output signal doAttached is similarly connected to one of the input ports of the DeviceNet device. Connections between different components, such as robots, tools and PLC, are done in what is called Station logic, which can be seen as a virtual wiring between devices/components and the example with the vacuum tool is shown in Figure 11.

I/O Connections

Source Object	Source Signal	Target Object	Target Signal
IRB1200	doVacuum	SC_VacumTool_2	diAttach
SC_VacumTool_2	doAttached	IRB1200	diVacuum

Figure 11: Connections between IRB1200 robot and vacuum tool.

The view in Figure 11 is different from the Design view in Figure 10 but still represents I/O-signals and connections. The user can thus choose which view is the most convenient to work with.

For other more complex parts such as the stamp more logic was needed, since there are more parts connected to each other and more dependencies to handle. The movement of the input device before the stamp depends on the position of the robot and placement of the sheet. An angular velocity has to be transferred from the PLC to the model and transformed into a linear velocity for the sheet when taken over by the cylinders. The start of the cylinders is also dependent of the placement of the sheet.

Another problem is to create "physics". In reality the sheet would automatically be brought into the stamp by the cylinders, but since no physics engine is included in RobotStudio this movement has to be explicitly programmed with logical expressions and several other smart components making the modelling part very complex even for modelling simple movements.

4.4.2 High-level modelling

Since the static model was imported as an assembly with all parts in their correct place there was no need for placement of parts except for some minor adjustments. The high-level modelling part mostly consisted of creating signals and connections between the PLC and other components. This was done in a Smart component called ConnectivitySmartComponent, where all PLC signals needed in the model were defined. The signals need to have the same name in the simulation model as in CodeSys, which is the PLC-programming environment, to be able to connect to them. ConnectivitySmartComponent is what establishes a connection between the virtual PLC and the model of the factory.

Figure 12 shows how the ConnectivitySmartComponent looks. VAC500 is a variable used to connect to the correct address of the virtual PLC via TCP/IP.

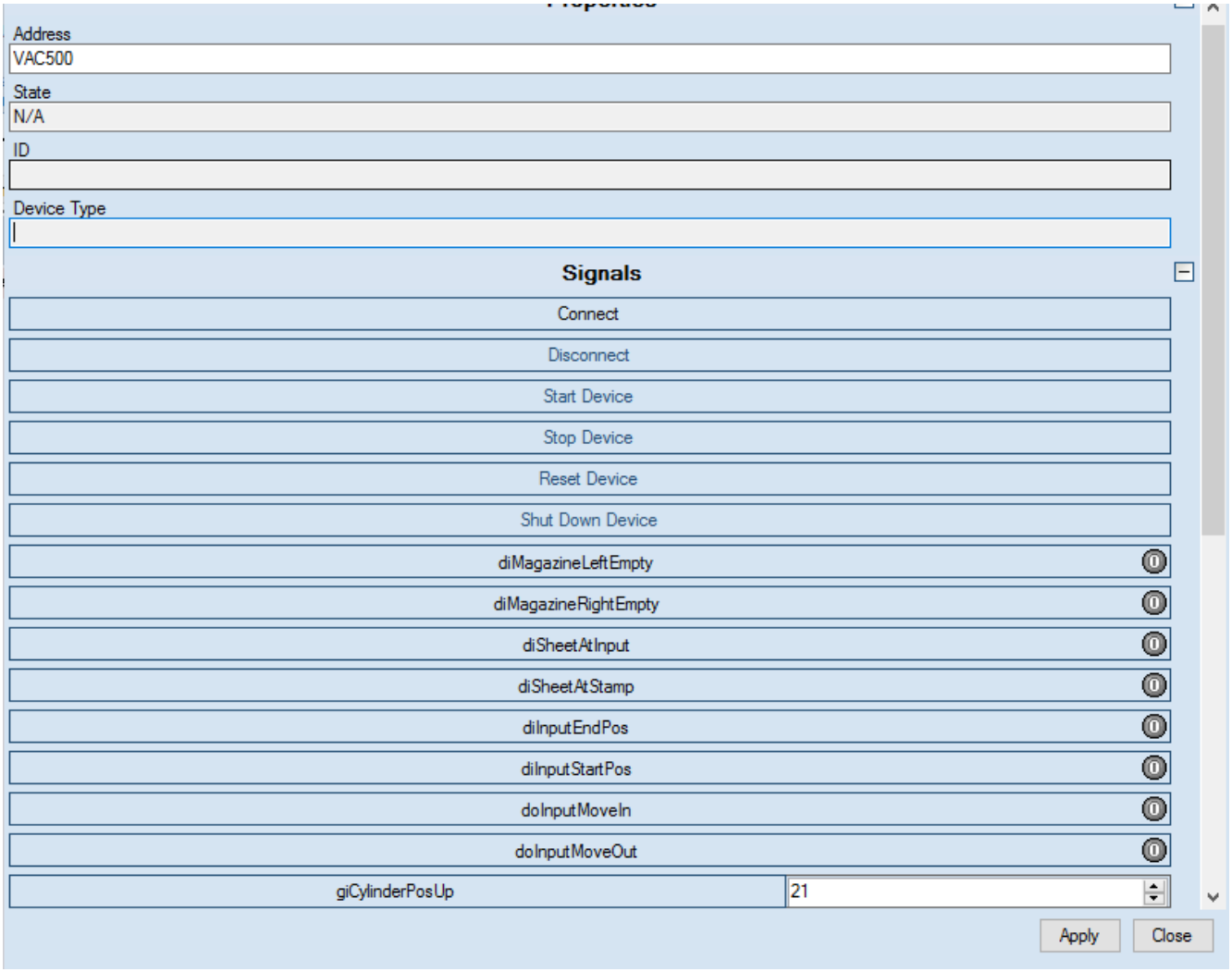

Figure 12: View of how the ConnectivitySmartComponent looks like.

The inputs and outputs defined in ConnectivitySmartComponent are the inputs and outputs to the PLC. The PLC signals are connected to one of the robots, the stamp, an actuator or some other components in order to get the correct functionality. Boolean signals are defined in the model as digital inputs/outputs and int/doubles are defined as group inputs/outputs.

4.5 Communication protocols

In the physical factory most of the communication between devices such as sensors, robots and actuators will go through Profinet. The exceptions to this are the servo drives for the cylinders in the rolling die cutter that will communicate with the PLC using EtherCat and some other, mostly diagnostic features, are sent via TCP/IP to the supervising system of the factory. Even though the majority of the devices communicate via Profinet as stated above and it would be favorable to be able to emulate this, this will not be done in this project because this would require all the nodes and devices to be emulated and have a compatible protocol on how the different devices should interconnect. This is not available today for all the devices. An example of what is meant by having a compatible protocol is how time should be handled between the different devices being emulated. Should they run free or be synchronized, should time slices be used or not, etcetera. All these kinds of options have to conform between the different devices and what has been found is that it is only a few manufacturers that even supply emulators for

their products and that there probably is no easy way to connect devices from different manufacturers.

In this project the PLC, the robot controllers and the control panel have been emulated while the other devices, parts and movements such as sensors, product carriers etcetera have been simulated inside RobotStudio. The virtual PLC is a standalone part of Automation Builder (27), in which the virtual PLC is configured. The configuration includes which CPU that should be used and what modules such as I/O- and communication- modules the system will have.

Automation Builder is also where configuration such as IP-addresses of a Profinet module is configured and where variable names can be connected to hardware addresses for easy use in the programming phase. When configuration of the hardware is done and it is time to start programming Automation Builder will start CODESYS v2.3 where the configurations from Automation Builder and all the necessary libraries needed will be loaded. When the programming is completed and it is time to load the program into the virtul PLC the first thing to do is to start an emulator from Automation Builder. After it is started the program can be loaded into the emulator from CODESYS where the protocol used to load the code into the emulator is the same that would be used for the physical PLC, namely TCP/IP. The virtual PLC is now loaded with the program and it is time to connect the virtual model built in RobotStudio to the virtual PLC. Figure 13 shows a visual compilation of how all devices and applications are connected.

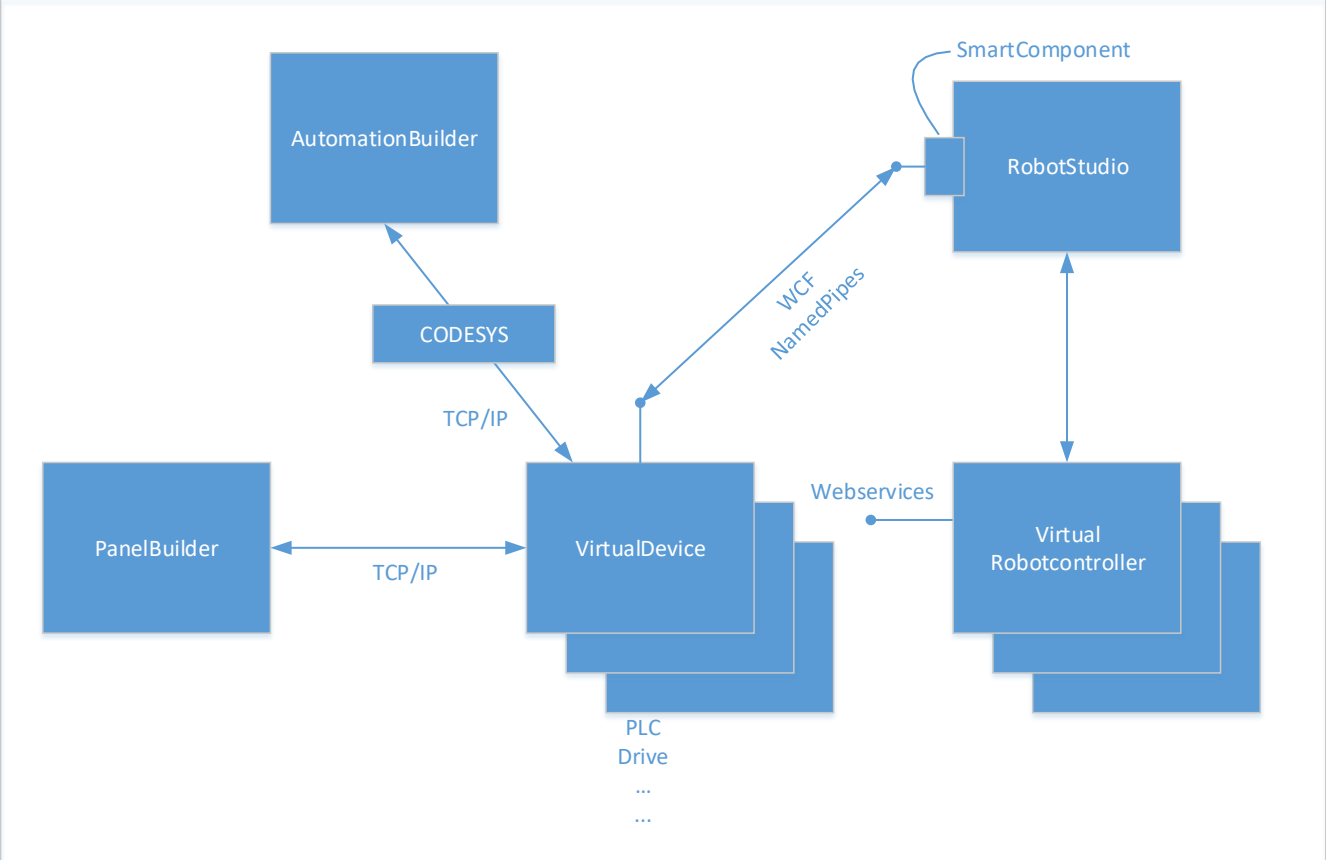

Figure 13: Coupling between different applications.

To connect to the virtual model created in RobotStudio a SmartComponent developed by ABB is used. The smart component works as a sort of plugin in RobotStudio which makes it possible to establish a connection between the Virtual PLC and the SmartComponent inside RobotStudio using Windows Communication Foundation (WCF) NamedPipes. The virtual robot controllers are also standalone processes outside RobotStudio and communicate through an ABB developed protocol, in the virtual robot controllers as in the real controller there is the possibility to connect to the controller through different web services which will be used later in the physical factory by the

supervising system to get information from the controllers. In the factory there should also be a Human Machine Interface (HMI) of a control panel, the HMI is connected to the virtual PLC in the same way as it would with a real PLC by using TCP/IP.

4.6 Verification and validation

The model has been divided into different modules, introduced in Section 3 and one good approach is to test the model components separately and then in modules, before testing the entire factory. Each component has thus been tested separately to verify that their internal behavior functions before connecting them to other parts of the model. In RobotStudio this can be done by manually triggering inputs to the components and verify that movements and other signals are accurate. When a component is verified to function correctly, it can be added to the high level model which then has to be verified as well. Some parts of the model had to be verified by studying the technical specification. By working in modules both in the model and in the PLC-program it has been possible to test small parts of the program before the entire model is completed. When one module in the simulation model has been verified it has been possible to test the PLC-code of that module.

The last part of verification and validation in this MSc project has been to perform a VFAT where the entire virtual factory was tested and validated. In the VFAT the entire process flow was checked and compared to the process specification. Furthermore the most obvious cases are considered that could happen in the physical factory such as breakdowns, emergency stops, parts that stop to function etcetera.

5 Final simulation

The results from the VFAT shows that the model is functional and the communication with the virtual PLC works. The model can visualize a completely automated process from an order being placed to a sheet and plastic lens being delivered to the customer. It can handle several cases such as when there is a need for refill of material, when one of the boxes are empty on the Flexlink conveyor and when QR-codes are faulty. Figure 14 shows the VFAT protocol that was used as a checklist for the VFAT in this MSc project.

Function	What should be checked?	Test	OK	Not OK	Comment
Stn1.Elipse	Does startorder from Elipse work?	Check that PLC receives startorder and starts process	x		
Stn1.PLC	Programstart PLC	Check that PLC starts the process	x		
Stn1.Sensors magazine	Does PLC handle empty/not empty magazine?	Check that correct process is running depending on sensor signals	x		
Stn1.Tool IRB1200	Does the tool work?	Check that robot can pick and release sheet	x		
Stn1.Tool IRB1200	Does robot handle if vacuum guard goes inactive	Routine that checks vacuumguard and sends error signal	x		
Stn1.Printer	Does printer work?	Scan print	x		Elipse handles printer
Stn1.Scanner	Does scanner work?	Check status signal from elipse		x	No signal received from Elipse
Stn1.Sensor input	Is start signal set depending on sensor signal and robot status signal?	Check sensor signal, robot signal and PLC signal	x		
Stn1.Actuator input	Does communication between PLC and actuator work?	Check PLC signal, movement of actuator and and input signals to PLC	x		
Stn1.Stamp	Does program handle if stamp stops movement while sheet is in stamp?	Checkpoint		x	Not implemented in RobotStudio model
Stn1.Stamp	Does communication between PLC and servo work?	Check signals and movement of stamp cylinders	x		
Stn1.Sensor output	Is start signal set depending on sensor signal?	Check sensor signal and PLC signal	x		
Stn1.Actuator output	Does communication between PLC and conveyor work?	Check PLC signals and movements of conveyors	x		
Stn1.Sensor flip	Is stop signal and actuator start signal set depending on sensor signals?	Check sensor signal and PLC signal	x		
Stn1.Actuator flip	Does communication between PLC and actuator work?	Check PLC signal, movement of actuator and and input signals to PLC	x		
Stn1.Elipse	Is sheet delivered to customer?	Check signal sent to elipse when flip is up	x		
Stn1.Robot handshake	Does handshake between robot and PLC work?	Check flowchart and signal exchange	x		
Stn1.Scrapping	Does PLC and robot handle faulty QR-codes?	Check PLC signal to robot and robot movements	x		
Stn1.Robot	Does robot start in a safe state?	Check trap routine. Robot can not move until it is within a certain area	x		
Stn2.Elipse	Does start order work?	Check that PLC receives startorder and starts process	x		
Stn2.Flexlink conveyor	Does start of conveyor work?	Check PLC signal and conveyor movements	x		
Stn2.Flexlink stops	Does communication between PLC and stops work?	Check signals and movement of stops	x		
Stn2.Flexlink stops	Are the stops in correct position?	Check if plates are stopped?	x		
Stn2.Flexlink inductive sensors	Are sensors in correct position?	Checkpoint(sensor before pick pos., plate in pos. and at pick pos.)	x		
Stn2.Flexlink cylinders	Does communication between PLC and cylinders work?	Check PLC signal, cylinder movements and input signals to PLC	x		
Stn2.Robot	Does the tool work?	Can robot pick and drop lens?	x		
Stn2.Robot	Does the robot pick correct lens?	Check communication between robot and PLC and robot movements	x		
Stn2.PLC	Does the program handle if a box is	Check if lens position is reset and	x		
Stn2.PLC	Does the program handle if all boxes are empty?	Check PLC signal and conveyor movements	x		
Stn2.Sensor lens fixture	Is sensor in correct position?	Check sensor signal when lens is in	x		
Stn2.Actuator lens fixture	Does communication between PLC and actuator work?	Check PLC signal, movement of actuator and and input signals to PLC	x		
Stn2.Eton	Does eton start signal work?	Check eton signal and movements	x		
Stn2.Eton	Does the "Clear" signal work?	Check signal and position of carriers	x		
Stn2.Eton	Does the "Drop" signal work?	Check signal and if lens is dropped	x		
Stn2.Eton	Does the "Carrier at delivery" signal work?	Check signal and position of carriers	x		
Stn2.Eton	Does the "Carrier at picking position" signal work?	Check signal and position of carriers	x		
Stn2.Elipse	Is "delivered"-signal set when lens is dropped?	Check PLC signals and if lens is dropped	x		
Stn2.Delivery	Is sensor in correct position?	Checkpoint	x		
Stn2.Handshake	Does handshake between robot and PLC work?	Check flowchart and signal exchange	x		
Stn2.Robot	Does the program handle if the robot fails to deliver lens to fixture?	Check sensor after robot movement is done. Send error signal		x	Function is not implemented
General	Does robot send/PLC receive status signals	Checkpoint		x	Function is not implemented
General	Does the robots have safety zones?	Check robot safety zones		x	Function is not implemented
General	Does factory handle emergency stop?	Check if factory shut down		x	Function is not implemented
General	Does factory handle controlled stop?	Check if factory does a controlled shut down		x	Function is not implemented

Figure 14: VFAT protocol

The protocol shows which cases that were covered and what needs to be further developed. Further developments could be both in the simulation model, robot program and in the PLC-code.

The VFAT that has been conducted does not include the supervising system and is thus only to show the communication between the PLC and the VC-model. The variables needed to communicate with the superior system are still in use in the PLC-program though and therefore to be able to run the simulation successfully these have to be manually set. Some parts were not possible to simulate with the real PLC-code, so the program had to be simplified a bit for the simulation to run.

6 Challenges and possibilities of Virtual commissioning

To be able to answer the 3rd and 4th research question in Section 1.3, concerning the challenges, possibilities and the state-of-the-art of virtual commissioning, interviews have been conducted with experienced people within the area. The interviews have been semi-structured with open questions and are summarized in the next three subsections. Conclusions from all interviews are summarized and combined in Section 6.4.

6.1 Interview Jesper Halmsjö, ÅF

Jesper Halmsjö is a consultant at ÅF (Ångpanneföreningen) currently working at a project at SKF in Gothenburg. He has a background in production and automation which is the area in which he finished his master degree at Chalmers University of Technology. He completed his master thesis in 2016, which was a collaboration with SKF. SKF was at the time in the design phase of building a new production cell and Jesper's job together with his thesis partner was to create a virtual commissioning model of the cell. They built two versions of the virtual commissioning model using two different softwares, ABB's RobotStudio (28) and Xcelgo's Experior (29). They managed to create a well functioning virtual commissioning model in both softwares and they also came up with ideas on how one should perform a virtual commissioning in an efficient way and what needs to be improved both from the suppliers of components and from the developers of the modeling/programming software.

When asking Jesper if virtual commissioning is something that is being used or asked for today, the answer is basically no, not yet. His estimation is that within 10 years it will be common practice and that these upcoming years a lot will happen. As usual it will be the big car manufacturers, such as Volvo, that will control this evolvement and specifically Volvo has already started using virtual commissioning in some of their projects. This has pushed leading companies on the market such as KUKA and Siemens to develop their modelling/programming software. If more companies starts demanding VC in their projects other big software companies such as ABB will have to follow.

Since projects involving VC is unusual it is hard to say what the work process usually looks like. Every company has their own approach. The people responsible for the VC-model usually are in the middle of everything though, and need to have knowledge about all parts. Most important is to have close contact with the group responsible for defining signals and the purchasing department. To gain the most benefits out of a VC it should be involved from the start of a project. This way every new design choice can be tested and evaluated early and the work progress will be iterative. Usually today the model is built after all design choices are made, making it too late to gain full effect of the model.

One problem with building a VC-model today is that it is very time consuming and to be able to efficiently build the model it is needed to come to a drag-and-drop stage. This would mean that the low level components already are available from the suppliers, ready to be imported to the high level model. Today it may be difficult to even get the CAD-geometries from the suppliers, making it harder to create an accurate model and slowing down the model building process. Another problem is the amount of signals that somehow has to be imported to the model. The software should include support for auto-generation of signals making it possible to import the signals defined by the electrical engineers. This is not supported in all software today. There are also too few functions available in the libraries, making it hard too create more complex systems.

So, is it possible to make profit from performing a VC? Jesper's answer is that it is probably possible but it will have to be a large scale project, including several similar plants. The first plant model will take time to build but the subsequent plants can be built by drag-and-drop-modelling with some minor adjustments. And to test the plant in the virtual environment instead of at site is of course profitable. There is thus profit to be found but the software has to be more stable and easy to work in. As of today ÅF has only done VC as thesis projects. Some companies that have their own complete system and predefined layouts performs VC of their systems. One example is Flexlink that is a supplier of transport systems. It is a different situation compared to production cells though, where there are components from a lot of suppliers and a lot of different software involved.

6.2 Interview Magnus Seger, ABB

Magnus Seger is a digital factory engineer at ABB and has been working in the business for about 20 years. He started working with virtual commissioning 4-5 years ago when the subject was still in its early ages and he has tried a lot of different software such as ABB's RobotStudio, Excelgo's Experior and Visual Components (30). He mentions that the softwares most commonly used today are Siemens Process Simulate (31) and Dassault Systems (32) and that it is the most common in car manufacturing business in Germany. When asked about what changes he has seen happening the last couple of years in the area of VC Magnus says that it is all about simulating more. It is about testing the PLC-logic, that it is correct but also about visualizing robot movements, operators and simulating everything more exactly and realistically. Basically what we want is a digital twin of the factory. Until now the most common approach in virtual commissioning is to run HiL but what is desired now and in the future is to be able to have stable simulations running SiL, using a virtual controller and to include a VFAT in the project plan. This would make it possible to make all design choices and testing without needing any hardware in advance. Magnus says that we are not there yet though. Siemens for example have only one virtual PLC that can be used, and it is still very expensive.

What is missing today for companies to perform virtual commissioning projects in an effective way? Magnus says that for one thing there is a lack of software. There are basically no products on the market that are stable. But there are also problems with a market that has not yet adapted to these new methods. Companies are not ready to take risks and instead use old methods that they know works. Usually at commissioning it takes a lot of time to reach full production rates due to errors in the program or other mistakes. This is a critical time in a project which costs a lot of money. Building a model to test the program in in advance is something that could make the ramping period at commissioning much shorter, but that is very hard to convince companies to pay for. They are not there yet and not willing to change their process. This could be an effect of another problem that Magnus mentions, that it is very hard to quantify the results of a VC project. This is because every project is unique making it hard to quantify how much a VC project actually saves. Another problem that is commonly mentioned is that it takes a lot of time to build the VC models which Magnus agrees on. It takes a lot of time to build the model and the models are never re-usable. It is also an area where the person building the model have to have deep knowledge in a lot of separate areas to be able to build an accurate model, and even in that case a lot of mistakes and faults might be seen in the model.

To be able to perform a VC project Magnus strongly believes that the process has to be changed completely. Usually in an automation project there is a planning phase and a design phase followed by some engineering and lastly coding and commissioning. Usually commissioning takes a lot of time, a lot of mistakes are found and it costs a lot of money. Magnus thinks that the engineers and programmers must be involved much earlier in the project, ideally from start. The process has to be more parallel so that programming and testing is done iteratively throughout the project. This could mean that the total time of the project is longer but will yield in a significant shortening of the commissioning time which is the most crucial. Parallelization of the process and to introduce a VFAT is the most significant differences between the traditional way of how an automation project is done and how it should be done in the future, according to Magnus.

The most important things to test in a virtual environment before commissioning is of course the PLC logic but also the interaction between different devices. All the handshakes and signal exchange between for example robots, PLC and the supervising system. Often the programs appear to work well separately but when the systems are interconnected problems occur. Some VC-programs include a physics engine which yields a more realistic simulations and Magnus thinks that there are many pros of using that. Without a physics engine everything is ideal, and a lot of test cases are not possible or very time consuming to build up. A physics engine makes the simulations more realistic and good looking for the users. Magnus is convinced that in a near future VC will be common practice but it is up to the large manufacturers to decide and the others will follow.

6.3 Interview Anders Spaak, ABB

Anders Spaak works at ABB as a product engineer and is a part of the research and development (RD) department of RobotStudio in Gothenburg, which is the main site for coding and development of RobotStudio. Anders does not write any source code himself but instead works with testing new and existing RobotStudio versions where he gives feedback and acts as a coupling between the end user and the RD-department. He has also done consulting work and has earlier held educational classes in RobotStudio. Anders has worked at ABB with robotics since 1998 and has been working with RobotStudio since the start.

When asked about his previous experience with virtual commissioning he answers that it depends what is meant by virtual commissioning. He says that he has done a lot of offline programming and virtual commissioning of robots but not at the extent that is usually referred to today including PLC, servo drives etcetera for an entire factory or cell. Anders also says that what is being developed from ABB right now is the virtual PLC which can be connected and run with RobotStudio. However he says that there is still too much work to set up the model and connect the different drives, such as servo- and communication drives. He believes that there is still a bit to go before it is simple and fast enough to set up the connections and the virtual environment so that time and focus can be spent on coding of devices such as PLC and robots.

Anders is certain that VC will become common practice but says that it took approximately ten years for offline programming of robots to become accepted. He thus expects that it will take around the same amount of time for VC to become accepted on the market. We asked him what he thought about companies building libraries and manufacturers offering standard dynamic components to speed up the process of building the environment in the future. His answer is that he definitely thinks this will become a common thing and that companies are going to have this included in their business model, but he also says that if companies start to supply tested and verified code for the user to only put some unique details on top the need for a VC would decrease.

When asked about how he thinks a VC project should be carried out he replies that the dynamic model should be developed as soon as a conceptual model is chosen and that the mechanical designers should work side by side with the virtual model developers to be able to find mistakes and problems and make concurrent improvements. Anders says that how it works today is that the robot programmers usually build up the environment and do offline programming of the robots. He himself sometimes simulates a PLC using a robot controller to try to test more of the system but this code cannot be used in the PLC later and is only to try to get a bit farther in the robot programming. When the model developers and the mechanical designers are finished the virtual model should be handed to the programmers of robots, PLC etcetera so that they can start to test their code against the virtual model. Today typically the PLC programmers are not involved in the project before the physical factory or cell is constructed, which is the time when they start to program. After the coding is done a VFAT should be performed and be satisfactory. Before the VFAT no equipment of the factory should be ordered. He also points out that even though you can cut down on the commissioning time by including a VFAT there is always going to be some adjustments needed in the physical factory due to the complexity of reality.

6.4 Compilation of interview answers

Virtual commissioning is not yet widely used or asked for by customers. The market is not ready to pay for the risks. It could be possible to make profit out of a VC but it needs to be a large scale project with a lot of plants with similar layout. VC projects are often carried out as thesis projects today. The most recent changes in the area are that more realistic simulations are possible and a change from using HiL to using SiL can be seen.

What needs to be developed further?

- Software

  - More stable software.
  - Broader range of PLC and device specific emulators from manufacturers.
  - Software needs to be more optimized for simulation of more complex systems (the requirements on the computers are very high today).

- Modelling
  - Standardization of dynamic components
  - Larger library of components
  - Drag-and-drop modeling
  - Amount of signals is high, no standard way of importing/exporting, some sort of auto-generation needed.
  - Requires a lot of "expert knowledge" about everything.
  - Faulty models.

- Market
  - Market not ready. Needs convincing, time and a push from large companies.
  - Change of project process needed.

Most important to test:

- PLC logic.
- Signal exchange.

Process:

- There is no standard process of how to perform a VC project and every company has their own approach.
- VC-model people is in the middle (need knowledge about everything).
- Engineers/programmers should be involved from start.
- Process needs to be iterative and parallel.
- VFAT needs to be introduced.

7 Discussion

This section presents discussions about how this project has been conducted and the outcome of trying to answer the research questions presented in Section 1.3.

7.1 Discussion about how to build a VC model efficiently

It has been shown throughout the project that the time it takes to build a VC model is one of the most prominent problems in a VC project. There is yet no standardized method for how to build the model, and a lot of the softwares available today lack functionality and user-friendliness. The study has shown that to be able to efficiently build the model the dynamic components has to be available in advance for easy drag-and-drop modelling. If this is not possible, as is usually the case today, it has been shown very important to work close to the mechanical engineers, electrical engineers and robot- and PLC-programmers.

One reason that it might take a lot of time to build the model is due to changes of the design throughout the project, or that the planning phase was not performed carefully enough, yielding component lists, IO-lists etcetera that are faulty. Faulty IO-lists or components lists, could lead to the same work having to be done twice with, the only difference being change of components or signals. This takes up unnecessary time since these kind of errors could be avoided by making sure that the planning phase and design phase are carefully conducted. Another way to look at it is that changes in the design is a natural part of an automation project and thus the modelling part should be flexible enough for these kind of changes and that the software should be designed for easy replace of signals or components.

When discussing about virtual commissioning the opinion on what should be excluded and simplified in the model can differ. Some people want the model to be an accurate copy of the real factory including everything for example wires, tubes, lamps, etc., whereas others think simplifications and exclusions should be made to both save time and to make the simulation computationally less greedy.

7.2 Discussion about verification of VC model and VFAT

The verification part of the project could be divided into two parts; verification of VC model and verification of PLC-program. The verification of the model is crucial to do before starting to verify the PLC-program.

One thing that has been shown during this project is that it is hard and time consuming to test larger parts of the model without having access to PLC-code. Without PLC-code it is not possible to have a complete sequence of events, but instead one has to manually trigger different inputs or outputs to simulate actions. This has lead to some simple PLC-programs being written by the authors of this thesis, with the sole purpose of verifying the model, and then being scrapped. Since PLC-programming and development of the model should be conducted as two separate processes, it seems weird that model developers need to write PLC-code to be able to verify the model. Furthermore to be able to verify the model by using PLC-programs the complete setup with model and virtual PLC is needed throughout the project, which may not always be the case. There should thus be better ways within the modelling software to verify the model.

Only a simplified VFAT has been performed in this project, using the checklist presented in Section 5. It has been hard to know exactly how to structure a VFAT and what to test in what order. It is important to have a clear specification on demands of the system which is something that has not been available in this project. Instead the verification has been based on the authors of this thesis knowledge about the system and the process together with the group responsible for the PLC-program. Generally there is a customer that has some demands on the system being delivered, specified in a specification list.

Another aspect that should be stated and standardized is who should be responsible for the verification and the VFAT. Should it be the model developers, PLC-programmers or some other group that are experts in testing and has general knowledge about both modelling and PLC-programming? We think that the best way now is that the model developers are responsible for the verification of the PLC-code, but this requires some knowledge about PLC-programming and the code from the model developers side. This way of working could though lead to conflicts among the groups. Who should fix the errors that are found? One way is that the PLC-programmers and model developers work side by side during the verification phase, which was the method used in this project. When an error was found by the model developers, it was fixed by the PLC-programmers. This made sure that the programmers was responsible for the code throughout the entire project, not creating conflicts by some other group adding onto the code. The problem by working side by side like this is that the PLC-programmers may sit a lot of time just waiting for potential errors to be found, spending unnecessary time. Although this is an issue it was found that it was too complicated to work the other way around, having the PLC-programmers verifying the code using the model. The environment and connections between all programs were too complicated if one is not used to it, and the model is also very hard to work with if one has not worked with it from start. In future versions of the modelling software it may be possible to have the PLC-programmers use the VC model to test their program in. Probably the different softwares will be more user friendly and integrated into each other.

7.3 Sustainability and ethical aspects

A sustainable production is an important issue and includes both social, environmental and economical sustainability. Simulation of a plant or as in the case of this project, a complete emulation of the plant, can be used as a base for decision making, helping manufacturers to make ethical decisions. For example simulation or emulation can be used for optimizing the amount of material or energy use. A VC can also be used to detect errors in construction before ordering or building the components. It ensures that no component would go to waste.

Regarding social aspects including a VC has a lot of advantages. For example a lot of stress can be reduced since work can be made in advance at a PC at the office instead of on-site in the final phase of a project, simulation can be done to improve ergonomics of workers and to have an overall safer system. Safety critical systems can be tested in a safe virtual environment decreasing the risk of someone being harmed.

8 Conclusions and future work

This section presents the conclusions drawn from this MSc project and possible future work.

8.1 Conclusions

Considering the effort and time that one has to put into building a VC model today, and the difficulty of connecting the different VC-softwares, there seem to be quite a long time before VC will be part of the common working process. Most people agree that there are a lot that could be gained from doing a VC, but this is today mostly in theory. From both the practical work and the investigation of virtual commissioning in this MSc project, it is concluded that components that are needed for a full VC are not yet available on the market. This is one of the main problems. Another issue is that companies are not always willing to hand out all the information about their products.

To be able to build a VC model efficiently there is a need of standardization, that does not exist today. Virtual dynamic components have to be more easily accessible and have a standard format, for an easy model building process. There is also a need for a standardized working process. By including a VC and a VFAT the entire process is changed and it seems to be difficult to know how to include these things in the working process that exists today. Investigations on how to include these new ways of working should be made. For example, how to perform a VFAT, is an important aspect to investigate.

8.2 Future work

The model can be improved in many ways. Some of the movements of different parts can be more accurate and realistic, for example when the sheet is delivered to the customer. The robot programs and handshake between the robots and the PLC can also be further developed. One example is to include so that the PLC can demand status signals from the robot to see the state. This is not included today. Signals and actions to perform a controlled stop and emergency stop is also something that needs to be included in the future. Other than that there are some geometries that does not exist yet and has to be included in the future.

Furthermore it would be interesting to compare the real system results with the virtual model when the factory is built and running. By comparing cycle times and movements an evaluation can be done to see how accurate the model is and what the model lacks. The virtual model could also be used as a base if similar plants are to be built in the future. Using the existing virtual plant and just applying the changes needed to simulate the new factory would probably make it possible for a faster virtual commissioning and thus also start of production than building a virtual model of the new factory from scratch.

References

[1] F. Shrouf, J. Ordieres, and G. Miragliotta, "Smart factories in industry 4.0: A review of the concept and of energy management approached in production based on the internet of things paradigm," vol. 2015-, pp. 697–701, IEEE, 2014.

[2] Näringsdepartementet, "www.regeringen.se." http://www.regeringen.se/contentassets/869c75f458fc4585ab4ec8c13b250a07/informationsmaterial-smart-industri---en-nyindustrialiseringsstrategi 2016.

[3] W. Hofmann, S. Langer, S. Lang, and T. Reggelin, "Integrating virtual commissioning based on high level emulation into logistics education," *Procedia Engineering*, vol. 178, pp. 24 – 32, 2017. RelStat-2016: Proceedings of the 16th International Scientific Conference Reliability and Statistics in Transportation and Communication October 19-22, 2016. Transport and Telecommunication Institute, Riga, Latvia.

[4] R. Drath, P. Weber, and N. Mauser, "An evolutionary approach for the industrial introduction of virtual commissioning," in *2008 IEEE International Conference on Emerging Technologies and Factory Automation*, pp. 5–8, Sept 2008.

[5] P. Hoffmann, R. Schumann, T. M. Maksoud, and G. C. Premier, "Virtual commissioning of manufacturing systems a review and new approaches for simplification.," in *ECMS*, pp. 175–181, 2010.

[6] S. Makris, G. Michalos, and G. Chryssolouris, "Virtual commissioning of an assembly cell with cooperating robots," *Advances in Decision Sciences*, vol. 2012, 2012.

[7] M. Ko, E. Ahn, and S. C. Park, "A concurrent design methodology of a production system for virtual commissioning," *Concurrent Engineering*, 2013.

[8] G. Jansson, E. Viklund, H. Lidelöv, L. tekniska universitet, I. och hållbart byggande, and I. för samhällsbyggnad och naturresurser, "Design management using knowledge innovation and visual planning," *Automation in Construction*, vol. 72, no. 3, pp. 330–337, 2016.

[9] L. Lindlof, B. Soderberg, C. U. of Technology, C. tekniska högskola, D. of Technology Management, O. M. Economics, and O. M. Institutionen för teknikens ekonomi och organisation, "Pros and cons of lean visual planning: experiences from four product development organisations," *International Journal of Technology Intelligence and Planning*, vol. 7, no. 3, p. 269, 2011.

[10] A. Skoogh and B. Johansson, "A methodology for input data management in discrete event simulation projects," in *Proceedings of the 40th Conference on Winter Simulation*, pp. 1727–1735, Winter Simulation Conference, 2008.

[11] S. Robinson and V. Bhatia, "Secrets of successful simulation projects," in *Proceedings of the 27th conference on Winter simulation*, pp. 61–67, IEEE Computer Society, 1995.

[12] M. Shoham, *A textbook of robotics: 1, Basic concepts*. London: Kogan Page, 1986.

[13] J. Wang, H. Zhang, and T. Fuhlbrigge, "Improving machining accuracy with robot deformation compensation," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3826–3831, 2009.

[14] I. Adept Technology, "Six-axis robot configuration singularities." http://www1.adept.com/main/KE/DATA/Procedures/Singularity/Singularity.pdf, 2007.

[15] C. University, "Cs4733 class notes:kinematic singularities and jacobians." http://www.cs.columbia.edu/~allen/F15/NOTES/jacobians.pdf.

[16] O. Bohigas, M. Manubens, L. Ros, S. O. service, and S. $e-bookcollection$, *Singularities of Robot Mechanisms: Numerical Computation and Avoidance Path Planning*, vol. 41. Cham: Springer International Publishing, 2017;2016;.

[17] F. Danielsson, P. Moore, and P. Eriksson, "Validation, off-line programming and optimisation of industrial control logic," *Mechatronics*, vol. 13, no. 6, pp. 571–585, 2003.

[18] J. . Decotignie, "Ethernet-based real-time and industrial communications," *Proceedings of the IEEE*, vol. 93, no. 6, pp. 1102–1117, 2005.

[19] B. Meador, "A survey of computer network topology and analysis examples." `http:///www.cse.wustl.edu/~jain/cse567-08/ftp/topology.pdf`, 2008.

[20] S. Norden, G. Manimaran, and C. S. R. Murthy, "New protocols for hard real-time communication in the switched lan environment," in *Proceedings 23rd Annual Conference on Local Computer Networks. LCN'98 (Cat. No.98TB100260)*, pp. 364–373, 1998.

[21] J. Liu, L. Sun, and X. Chen, "The characteristics analysis about transmission time delay of industrial ethernet," in *2009 Second International Conference on Intelligent Networks and Intelligent Systems*, pp. 134–137, 2009.

[22] P. N. America, "§1.1 – coping with real time." `http://us.profinet.com/technology/profinet/`, 2016.

[23] O. Balci, "Validation, verification, and testing techniques throughout the life cycle of a simulation study.," *Annals of Operations Research*, vol. 53, no. 1-4, pp. 121 – 173, 1994.

[24] R. G. Sargent, "Verification and validation of simulation models," *Journal of Simulation*, vol. 7, pp. 12–24, 02 2013. Copyright - © Operational Research Society 2013; Last updated - 2013-10-08.

[25] R. G. Sargent, "Verification and validation of simulation models," *Journal of Simulation*, vol. 7, p. 15, 02 2013.

[26] G. I. M. Worm, J. P. Kelderman, T. Lapikas, A. W. C. van der Helm, K. M. van Schagen, and L. C. Rietveld, "The use of process simulation models in virtual commissioning of process automation software in drinking water treatment plants," *Water Science and Technology: Water Supply*, vol. 13, no. 5, pp. 1331–1339, 2013.

[27] ABB, "Automation builder." `http://new.abb.com/plc/automationbuilder/platform/software`, 2017.

[28] ABB, "Robotstudio." `http://new.abb.com/products/robotics/robotstudio`, 2017.

[29] Xcelgo, "Experior." `http://xcelgo.com/experior`.

[30] V. Components, "Visual components." `http://www.visualcomponents.com`.

[31] SIEMENS, "Process simulate." `https://www.plm.automation.siemens.com/en_us/products/tecnomatix/manufacturing-simulation/assembly/process-simulate.shtml`.

[32] D. Systèmes, "Delmia." `https://www.3ds.com/products-services/delmia`.

Appendix

A Flowchart IRB1200

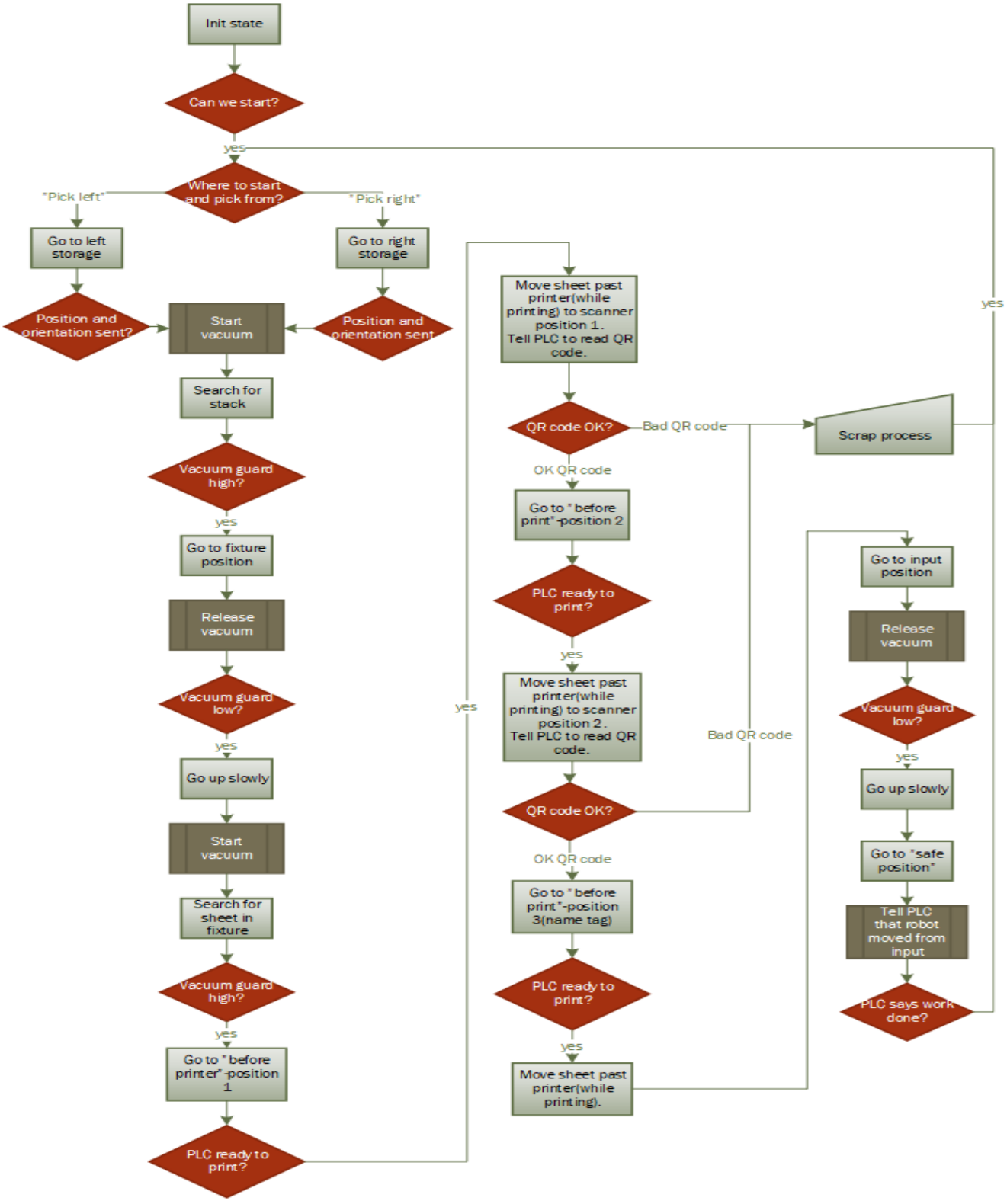

Figure 15: Flowchart of IRB1200 process.

B Flowchart IRB910

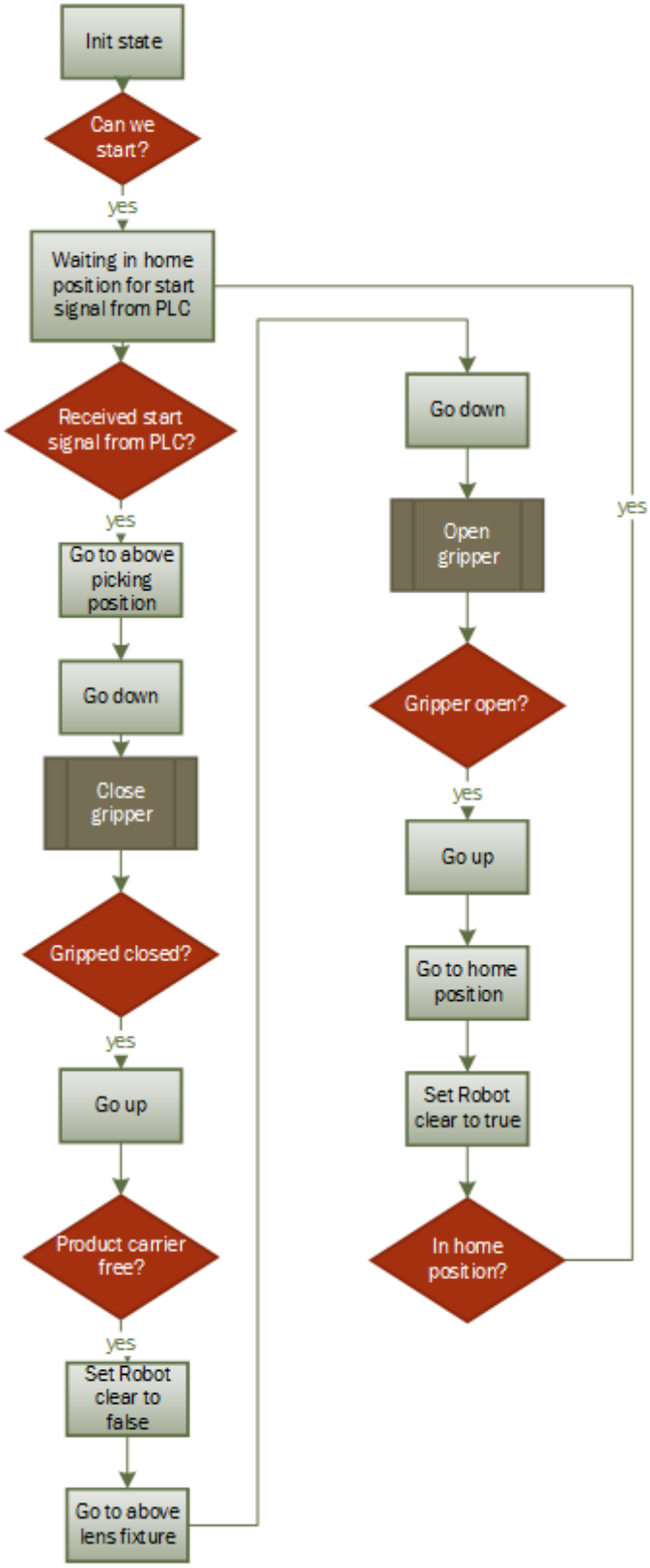

Figure 16: Flowchart of IRB910 process.

C RAPID code IRB1200

```
MODULE Module1
    VAR num pile_number:=1;
    VAR num loop_number:=0;

    VAR robtarget pFixturePos;
    VAR robtarget pSheetLeft:=Magazine_left_above;
    VAR robtarget pSheetRight:=Magazine_right_above;
    VAR robtarget pSheetLeftAbove:=Magazine_left_above;
    VAR robtarget pSheetRightAbove:=Magazine_right_above;

    VAR robtarget current_tool_pos;
    VAR errnum ERR_BATT:=-1;
    VAR intnum timeint;
    VAR intnum dovacuum_low;

  PROC main()
        SetDO doVacuum,0;
        SetDO doIRBStrobe,0;
        setGO goPLCComBits,0;
        CONNECT dovacuum_low WITH Lost_Sheet;
        ISignalDO doVacuum, 0 , dovacuum_low;
        ISleep dovacuum_low;
        init;
        WHILE TRUE DO
            WaitDi diPLCStrobe,1;
            WaitUntil giPLCComBits=0 OR giPLCComBits=1 OR giPLCComBits=2;   !0=pick left pile, 1=pick righ
            SetGO goPLCComBits,giPLCComBits;
            SetDO doIRBStrobe,1;
            WaitDI diPLCStrobe,0;

            IF giPLCComBits=0 THEN
                Pick_left;
                Fixture_proc;
                Print_scan;
                Input_stamp;
            ELSEIF giPLCComBits=1 THEN
                Pick_right;
                Fixture_proc;
                Print_scan;
                Input_stamp;
            ELSEIF giPLCComBits=2 THEN
                Reset; !resets the search position offset
            ENDIF
        ENDWHILE
ENDPROC

    TRAP Lost_Sheet
        TPWrite "Sheet dropped unexpected";
       EXIT;
    ENDTRAP


    PROC init()
```

```
        setDO doVacuum,0;
        SetDO doIRBStrobe,0;
        setGO goPLCComBits,0;
        current_tool_pos:=CRobT(\Tool:=VacumTool \WObj:=wobj0);
        if (current_tool_pos.trans.x < (Robot_Ready_To_Pick_Magazine.trans.x-100) OR current_tool_pos.trans
            (current_tool_pos.trans.y < (Robot_Ready_To_Pick_Magazine.trans.y-100) OR current_tool_pos.tra
                (current_tool_pos.trans.z < (Robot_Ready_To_Pick_Magazine.trans.z-100) OR current_tool_pos
                    Error_4800;
                ENDIF
    ENDPROC

    PROC Error_4800()
        VAR num errorid := 4800;
        VAR errstr my_title := "Robot in bad position";
        VAR errstr str1 := "Jog robot, To Start position!";
        BookErrNo ERR_BATT;
        ErrRaise "ERR_BATT", errorid, my_title, ERRSTR_TASK, str1,ERRSTR_CONTEXT,ERRSTR_EMPTY;
        ERROR
        EXIT;
ENDPROC

    PROC Pick_left()
        SetDO doIRBStrobe,0;
        MoveJ pSheetLeftAbove,v500,fine,VacumTool\WObj:=Wobj_magazine;
        SetDO doVacuum,1; !activates vacuum
        SearchL\Stop,diVacuum,pSheetLeft,offs(pSheetLeftAbove,0,0,500),v80,VacumTool\WObj:=Wobj_magazine;
        IWatch dovacuum_low;
        MoveL Offs(pSheetLeft,0,0,-40),v40,fine,VacumTool\WObj:=Wobj_magazine; !slow movement first part
        pSheetLeftAbove:=offs(pSheetLeft,0,0,-100);
        MoveL Magazine_left_above,v1000,z100,VacumTool\WObj:=Wobj_magazine;
        MoveL Magazine_left_above_2,v500,fine,VacumTool\WObj:=Wobj_magazine;
    ENDPROC

    PROC Pick_right()
        SetDO doIRBStrobe,0;
        MoveJ pSheetRightAbove,v500,fine,VacumTool\WObj:=Wobj_magazine;
        SetDO doVacuum,1;!activates vacuum
        SearchL\Stop,diVacuum,pSheetRight,offs(pSheetRightAbove,0,0,500),v80, VacumTool\WObj:=Wobj_magazin
        IWatch dovacuum_low;
        MoveL Offs(pSheetRight,0,0,-40),v40,fine,VacumTool\Wobj:=Wobj_magazine; !slow movement first part
        pSheetRightAbove:=offs(pSheetRight,0,0,-100);
        MoveL Magazine_right_above,v1000,z100,VacumTool\WObj:=Wobj_magazine;
        MoveL Magazine_right_above_2,v500,fine,VacumTool\WObj:=Wobj_magazine;
    ENDPROC

    PROC Reset()
        pSheetLeft:=Magazine_left_above;
        pSheetRight:=Magazine_right_above;
        pSheetLeftAbove:=Magazine_left_above;
        pSheetRightAbove:=Magazine_right_above;
        setDO doIRBStrobe,0;
    ENDPROC


    PROC Print_scan()
        IWatch dovacuum_low;
```

```
        MoveJ Printer_position_1_before,v500,fine,VacumTool\WObj:=Wobj_Printer;
        Handshake(3);
        MoveLdo Printer_position_1,v500,fine,VacumTool\WObj:=Wobj_Printer,doIRBStrobe,0; !in position to pi
        MoveLdo Scanner_position_1,v500,fine,VacumTool\WObj:=Wobj_Printer,doIRBStrobe,0; !in position to s
        Handshake_scanners;
        MoveL Printer_position_2_before,v500,fine,VacumTool\WObj:=Wobj_Printer;
        Handshake(3);
        MoveLdo Printer_position_2,v500,fine,VacumTool\WObj:=Wobj_Printer,doIRBStrobe,0; !in position to pi
        MoveL Printer_position_3_before,v500,fine,VacumTool\WObj:=Wobj_Printer;
        Handshake(3);
        MoveLdo Printer_position_3,v500,fine,VacumTool\WObj:=Wobj_Printer,doIRBStrobe,0; !in position to pi
        MoveLdo Scanner_position_3,v500,fine,VacumTool\WObj:=Wobj_Printer,doIRBStrobe,0;  !in position to
        Handshake_scanners;
        movej For_Reorient_For_Input,v500,fine,VacumTool\WObj:=Wobj0;
        MoveJ Reorient_For_Input,v500,fine,VacumTool\WObj:=Wobj0;
    ENDPROC

    PROC Handshake_scanners()
        Handshake(4);
        SetDO doIRBStrobe,0;
        WaitDI diPLCStrobe,1;
        IF giPLCComBits=5 THEN        !giPLCComBits=5: QR code okay
            Handshake(5);
            SetDO doIRBStrobe,0;
        ELSEIF giPLCComBits=6 THEN        !giPLCComBits=6: QR code not okay
            Handshake(6);
            Scrap_proc;
            SetDO doIRBStrobe,0;
        ENDIF
    ENDPROC

    !handshake procedure between robot and PLC
    PROC Handshake(num handshake_num)
        WaitDi diPLCStrobe,1;
        WaitGI giPLCComBits,handshake_num;
        SetGO goPLCComBits,handshake_num;
        SetDO doIRBStrobe,1;
        WaitDI diPLCStrobe,0;
    ENDPROC

    PROC Scrap_proc()
        movej For_Reorient_For_Input,v500,fine,VacumTool\WObj:=Wobj0;
        MoveJ Scrap_pos_above,v500,fine,VacumTool\WObj:=Wobj_input;
        MoveJ Scrap_pos,v50,z50,VacumTool\WObj:=Wobj_input;
        ISleep dovacuum_low;
        SetDO doVacuum,0;
        WaitDI diVacuum,0;
        WaitTime 5;
        MoveJ Scrap_pos_above,v500,fine,VacumTool\WObj:=Wobj_input;
MoveJ Position_Go_From_Stamp,v1000,z50,VacumTool\WObj:=wobj0;
MoveJ Robot_Ready_To_Pick_Magazine,v1000,z50,VacumTool\WObj:=wobj0;
        main;
    ENDPROC

    PROC Input_stamp()
        MoveJ Above_input,v500,fine,VacumTool\WObj:=Wobj_input;
```

```
        MoveL Input,v500,fine,VacumTool\WObj:=Wobj_input;
        ISleep dovacuum_low;
        SetDO doVacuum,0;
        WaitDI diVacuum,0;
MoveL Above_input,v500,fine,VacumTool\WObj:=Wobj_input;
        Handshake(7); !robot moved from input
        setDo doIRBStrobe,0;
MoveJ Position_Go_From_Stamp,v500,fine,VacumTool\WObj:=wobj0;
MoveJ Robot_Ready_To_Pick_Magazine,v500,fine,VacumTool\WObj:=wobj0;
    ENDPROC

     PROC Fixture_proc()
        MoveJ offs(Above_fixture,30,30,-30),v500,z100,VacumTool\WObj:=Wobj_fixture;
        MoveL offs(Fixture_pos,30,30,-30),v500,fine,VacumTool\WObj:=Wobj_fixture;
        ISleep dovacuum_low;
        SetDO doVacuum,0;
        WaitDi diVacuum,0;
        MoveL Above_fixture,v40,z10,VacumTool\WObj:=Wobj_fixture;
        SetDO doVacuum,1;
        SearchL\Stop,diVacuum,pFixturePos,offs(Fixture_pos,0,0,10),v80, VacumTool\WObj:=Wobj_fixture;
        IWatch dovacuum_low;
        MoveL Above_fixture,v40,fine,VacumTool\WObj:=Wobj_fixture;
        MoveL Above_fixture_2,v400,fine,VacumTool\WObj:=Wobj_fixture;
    ENDPROC

ENDMODULE
```

D RAPID code IRB910

```
MODULE Module1
    VAR robtarget start_pos_above;
    VAR robtarget start_pos;
    VAR num row_offset;
CONST robtarget Lens_row1_start:=[[227.157,371.099,-74.5],[0.5,0.5,0.5,0.5],[-1,-1,0,1],[9E+09,9E+09,9E+09
CONST robtarget Lens_row1_start_above:=[[227.157,371.099,-134.5],[0.5,0.5,0.5,0.5],[-1,-1,0,1],[9E+09,9E+0
CONST robtarget Lens_row2_start:=[[147.157,361.099,-74.5],[0.5,0.5,0.5,0.5],[-1,-1,0,1],[9E+09,9E+09,9E+09
CONST robtarget Lens_row3_start:=[[67.157,371.099,-74.5],[0.5,0.5,0.5,0.5],[-1,-1,0,1],[9E+09,9E+09,9E+09,
CONST robtarget Home:=[[650,0,50.2],[0.707106781,-0.707106781,0,0],[0,-1,0,1],[9E+09,9E+09,9E+09,9E+09,9E+
CONST robtarget LensFixturePos:=[[334,423.5,-74.819],[0.5,-0.5,-0.5,0.5],[0,-1,0,0],[9E+09,9E+09,9E+09,9E+
    CONST robtarget Lens_row3_start_above:=[[67.157,371.099,-134.5],[0.5,0.5,0.5,0.5],[-1,-1,0,1],[9E+09,9
    CONST robtarget Lens_row2_start_above:=[[147.157,361.099,-134.5],[0.5,0.5,0.5,0.5],[-1,-1,0,1],[9E+09,
    CONST robtarget LensFixturePosAbove:=[[334,423.5,-24.819],[0.5,-0.5,-0.5,0.5],[0,-1,0,0],[9E+09,9E+09,

PROC init_SCARA()
        setDO doOpenGripper,0;
     setDO doCloseGripper,0;
     setdo doSCARAStrobe,0;
     SetGO goSCARAComBits,0;
    ENDPROC

PROC Main()
    init_SCARA;
        WHILE TRUE DO
        MoveL Home,v1000,z100,SCHUNK_EPS_2\WObj:=wobj0;
        PickAndPlace;
        ENDWHILE
    ENDPROC

    PROC PickAndPlace()
       waitDI diPLCStrobe2,1; !wait for order from PLC to start pick lens from some position index
        IF giPLCComBits2 <=17 THEN
          start_pos_above:= Lens_row1_start_above;
          start_pos:= Lens_row1_start;
          row_offset:=1;
        ELSEIF giPLCComBits2 > 17 AND giPLCComBits2<=33 THEN
        start_pos_above:= Lens_row2_start_above;
          start_pos:= Lens_row2_start;
          row_offset:=17+1;
        ELSEIF giPLCComBits2>33 AND giPLCComBits2 < 51 THEN
        start_pos_above:= Lens_row3_start_above;
          start_pos:= Lens_row3_start;
          row_offset:=17+16+1;
          ELSE
            !knas
        ENDIF
            setGO goSCARAComBits, giPLCComBits2; !tell PLC that index is received
            setDO doScaraStrobe,1; !tell PLC to start reading bits
            WaitDI diPLCStrobe2,0; !start pick from index

           MoveJ Offs(start_pos_above,0,-20*(giPLCComBits2-row_offset),0),v1000,fine,SCHUNK_EPS_2\WObj:=Wo
            MoveL Offs(start_pos,0,-20*(giPLCComBits2-row_offset),0),v1000,fine,SCHUNK_EPS_2\WObj:=Wobj_le
              SetDO doCloseGripper,1;
```

```
        WaitDI diGripperClosed,1;
        setDO doCloseGripper,0;
          MoveJ Offs(start_pos_above,0,-20*(giPLCComBits2-row_offset),0),v1000,fine,SCHUNK_EPS_2\WObj:=Wob
        setDO doScaraStrobe,0; !handshake finished -> PLC start moving plates one lap
        Handshake(51); !check if it is safe to move lens to fixture(actuator has to be down, and no carrie
        MoveJ LensFixturePosAbove,v1000,fine,SCHUNK_EPS_2\WObj:=wobj0;
            MoveL LensFixturePos,v1000,fine,SCHUNK_EPS_2\WObj:=wobj0;
           SetDO doOpenGripper,1;
            WaitDI diGripperOpen,1;
            setdo doOpenGripper,0;
            MoveL LensFixturePosAbove,v1000,fine,SCHUNK_EPS_2\WObj:=wobj0;
        MoveJ Home,v1000,fine,SCHUNK_EPS_2\WObj:=wobj0;
        setDO doScaraStrobe,0; !handshake finished, safe to move fixture actuator
ENDPROC

    PROC Handshake(num handshake_num)
        WaitDi diPLCStrobe2,1;
        WaitGI giPLCComBits2,handshake_num;
        SetGO goSCARAComBits,handshake_num;
        SetDO doSCARAStrobe,1;
        WaitDI diPLCStrobe2,0;
    ENDPROC


ENDMODULE
```