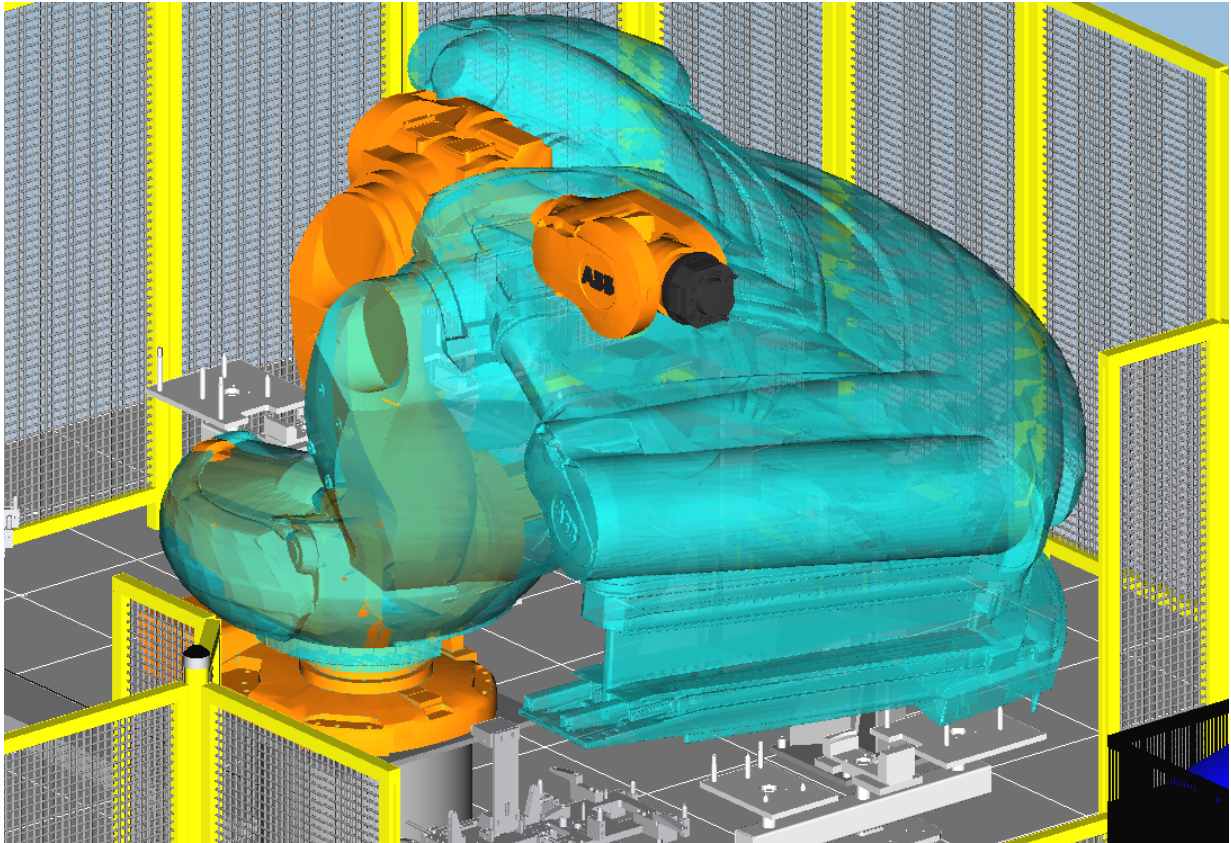


CHALMERS



Development of an Event Based Robotic Simulation implemented in Process Simulate

Master of Science Thesis

SAMIR DALILI
MARCUS PERSSON

Department of Signals and Systems
Division of Automatic Control, Automation and Mechatronics
CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg, Sweden 2009
Report No. EX100/2009

REPORT NO. EX100/2009

Development of an Event Based Robotic
Simulation implemented in Process Simulate

A Master of Science thesis conducted at Chalmers University of Technology in
collaboration with Volvo Aero Corporation

SAMIR DALILI
MARCUS PERSSON

Department of Signals and Systems
Division of Automatic Control, Automation and Mechatronics
CHALMERS UNIVERSITY OF TECHNOLOGY

Göteborg, Sweden 2009

Development of an Event Based Robotic
Simulation implemented in Process Simulate
SAMIR DALILI
MARCUS PERSSON

© SAMIR DALILI, MARCUS PERSSON, 2009

Report No. EX100/2009
Department of Signals and Systems
Division of Automatic Control, Automation and Mechatronics
Chalmers University of Technology
SE-412 96 Göteborg
Sweden
Telephone: + 46 (0)31-772 1000

Abstract

Simulation is playing a more and more important role in today's automation industry, where it saves development time and costs. Volvo Aero has recognized a new demand for automation when a new production method is introduced. In the future Volvo Aero wishes to simulate more of their manufacturing processes compared to what they do today. In the simulation three parts of an airplane engine article called the Turbine Exhaust Case will be measured, marked and fixtured.

This thesis involves determining if these processes can be used in the real robot cell, it also involves deciding the location of all the resources in the robot cell. Process Simulate was used as the simulation software. The robot cell was first simulated as a sequence based model and then converted to the final event based model. Additional software that was used to facilitate the work was Supremica and Sequence Planner.

From the results of the simulation a final layout was obtained that fulfilled the requirements from Volvo Aero. The main reason to create a simulation before building the real robot cell is that erroneous design parameters easily can be corrected, without costly modifications of the real robot cell.

Contents

Abstract	I
Contents	II
Preface	V
1 Introduction	1
1.1 Limitations	2
1.2 Purpose	2
1.3 Background	2
2 Theory	6
2.1 Robotic Simulation	6
2.2 Manufacturing Processes	6
2.2.1 Eddy current inspection	6
2.2.2 Direct Part Marking	7
2.2.3 Machine Vision	8
2.2.4 Fixturing	8
2.3 Software	10
2.3.1 Process Simulate	10
2.3.2 Architecture	10
2.3.3 Data Structure	11
2.3.4 Delmia Automation	13
2.4 Supervisor Control Theory	13
2.5 Supremica	15
2.6 Sequence Planner	15
3 Implementation	17
3.1 Installation of Process Simulate	17
3.2 Modelling and kinematics in Process Simulate	17
3.2.1 Modelling of grippers and pedestals	17
3.2.2 Adding kinematics to the fixture	18
3.2.3 Defining the gripper as a gripping tool	18
3.3 Robot placement	20
3.3.1 RobotSmartPlace	20
3.4 Sequence based simulation in Process Simulate	20
3.5 Event based simulation	23
3.5.1 Logic	23
3.5.2 Pallet switch	25
3.6 Modelling of the robot cell sequences using Sequence Planner	31
3.7 Modelling of the pallet and fixture switching process in Supremica	32

4	Result	35
5	Discussion	41
5.1	Modelling	41
5.1.1	Fixture gripper	41
5.1.2	Automatic Fixturing Model	41
5.2	Comparison between Simulation Models	42
5.2.1	Naming Convention	42
5.2.2	Sequence Based Model	42
5.2.3	First Event Based Model	43
5.2.4	Final Event Based Model	43
5.2.5	Model Differences	44
5.3	Logic	44
5.3.1	Robot signals	44
5.4	Problems	45
5.4.1	Dummy attachment	45
5.4.2	The rotational bug	45
5.4.3	Fixture	46
5.5	Additional Work	47
5.5.1	Automatic Path Planner	47
5.5.2	Pallet switch in Supremica	48
5.6	Methodology	48
5.6.1	Workflow methodology	48
5.6.2	Sequence Planner	51
6	Conclusions	54
	References	55
	Appendix A	57

Preface

In this thesis Process Simulate has been used to simulate a robotic cell which will be built in Trollhättan at Volvo Aero's facilities. Fixturing, measuring and marking processes are simulated. The work has been carried out from March 2009 to October 2009 at the Department of Signals and Systems, Division of Automatic Control, Automation and Mechatronics at Chalmers University of Technology. Petter Falkman was the supervisor and examiner at Chalmers and Mats Hansson was the supervisor at Volvo Aero.

Aknowledgements

We would like to say our thanks to our supervisors, the department of Signals and Systems for their support and to Johan Nordling for his help with the installation and support of Process Simulate. A special thanks to the people at Volvo Aero for showing their interest in our thesis.

Göteborg October 2009
Samir Dalili, Marcus Persson

1 Introduction

To find more economically justified solutions, drastic actions may sometimes be necessary. Volvo Aero is manufacturing a Turbine Exhaust Case in an airplane engine. This specific part is located in the exhaust area of the engine and acts as a bracket to hold the turbine axle. The turbine exhaust case is therefore not revolving unlike for example turbine blades. The current production method is very expensive and the competition between suppliers for this type of product is low. The reason for the high price is that it only is a handful of suppliers in the world that are able to deliver the item that Volvo Aero demands. The new production method connects sub-parts by laser welding to form the Turbine Exhaust Case. Volvo Aero intends to divide the Turbine Exhaust Case into 39 pieces, which puts higher pressure on the production technologies in terms of quality and accuracy. The fact that the Turbine Exhaust Case is a critical safety part in an airplane engine sets extra high demands on the production methods used, when it comes to traceability of parts throughout the whole products lifecycle. A simplified version of the Turbine Exhaust case is shown in figure 1.1. This new procedure makes outsourcing possible to a wider range of suppliers, in this manner lowering expenses. This thesis focuses on the fixturing, measuring & marking processes and the development of an event based robot simulation in Process Simulate.

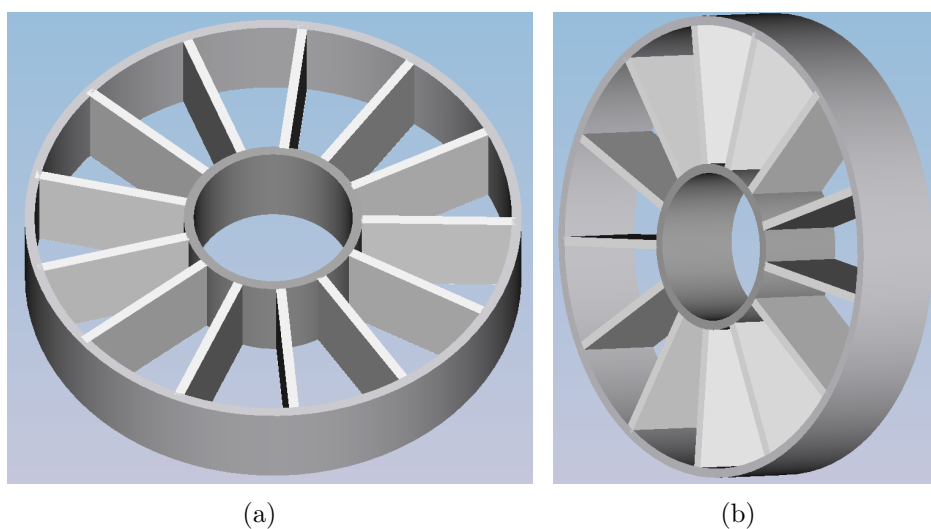


Figure 1.1: Simplified model of the Turbine Exhaust Case.

1.1 Limitations

This thesis is targeted towards persons with general scientific knowledge. There are a few limitations in this master thesis:

- Simplified parts are used, this is because the real parts are confidential.
- The fixture in this thesis is not the final fixture that will be used in the working robot cell.
- In a sense all robot programs using the simplified parts are incorrect, however the operations still display that the processes works properly but will need modifications when implemented.
- Volvo Aero will not be able to use the thesis robot paths without modifying the critical points in the robot paths. A critical point can be e.g. gripping positions when picking up parts.

Considering the limitations the simulation still gives satisfying results because finding the real parts gripping paths and positions is not the main purpose. This situation also applies for the collision tests with the real parts.

1.2 Purpose

The main purpose of this master thesis is to determine if it is possible to simulate and realize this robot cell and use it the way Volvo Aero has intended to. The sub-purposes are to find a suitable position for the fixturing robot and to create the logic for all robots and devices that are used in the work cell. Questions posed during the thesis work were:

- Can the robot operations be collision free?
- Where should the resources be located in the robot cell?
- Present additional tools if needed for the simulation?
- Is it facilitating to model the automaton for the pallet switch in Supremica?
- Is there any need for a Sequence Planner plug-in to Process Simulate?

1.3 Background

Today Volvo Aero buys articles like the Turbine Exhaust Case from suppliers. The main issue with this is that there are very few suppliers in the world that can manufacture such a large construction. Due to the fact that the part currently is bought in one piece there has not been a reason strong enough to look at automation. Volvo Aero will in the future buy smaller sub-pieces of the Turbine Exhaust Case and laser weld the parts together.

The amount of parts now increases from handling one large piece to handling thirty nine small pieces, and a great number of fixtures. This changes the need for automation greatly and Volvo Aero has decided to build a test robot cell where separate processes can be implemented and evaluated before they are incorporated into the manufacturing line.

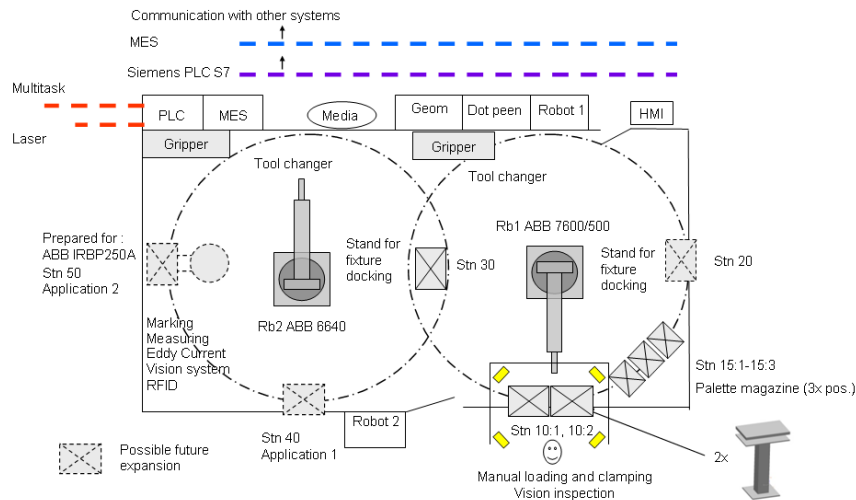


Figure 1.2: The simple drawing of the robot cell that outlined how the modelling could be done.

The test cell that is investigated in this thesis threats only a limited part of the whole process chain. In order to make the parts ready for the fixturing process, which will be investigated in this thesis, the parts must first be:

1. The molding from the suppliers must first be fixtured to make it ready for the milling process.
2. The milling process removes unwanted residue from the molding process.
3. The parts are then brushed to correct surface defects.
4. The parts are cleaned before the welding process begins which is crucial to get a high quality weld.
5. The fixture that each part was placed in are then removed and three of these parts are moved to a larger fixture.

The larger fixture mentioned above will be used in the thesis work. In order to understand the robot cell layout the operating sequences of what will commence in the robot cell is described here. More details on the manufacturing processes can be found in section 2.2. The positions in the list below can be found in figure 1.2. In the beginning of the modelling work, a rudimentary sketch was provided from Volvo Aero, this can be seen in figure 1.2.

1. A human operator loads three parts into the pallet table at position 10:2.
2. Robot 1 switches to the part gripper, which can lift all the three different parts.
3. Robot 1 moves the parts from position 10:2 to the fixture in position 10:1.
4. Robot 1 switches to the fixture gripper, which is used to move the fixture.
5. The fixture is closed by a human operator, and the parts are securely fixated in the correct positions.
6. Robot 1 moves the fixture to the common worktable at position 30.
7. Robot 2 mounts the measurement equipment and measures that the fixating process has been correctly done.
8. The parts are fake welded together.
9. Robot 2 mounts the inspection equipment for the weld, the eddy current tool. Robot 2 then performs the eddy current measurements.
10. Robot 2 changes to the marking equipment, the dot peen tool and then marks the welded part.
11. Robot 1 moves the fixture from position 30 back to position 10:1 where a human operator opens the fixture and removes the part.

In figure 1.3 three of the thirty nine parts that make up the whole Turbine Exhaust Case is shown, it is these three parts that are described in the sequence of operations list above. Due to that the parts consists of highly complex surfaces there are problems when gripping the parts and it is also very difficult to find appropriate reference points.

Robot cell simulation is an imitation of a real robot cell. To create and optimize automated manufacturing systems, simulation based software are necessary [4]. When using simulation software to design different scenarios and setups the overall time decreases compared to if the robot cell is constructed with traditional methods, e.g. if a robot cannot reach a gripper placed on a tool stand. In a simulation this problem is solved relatively quickly compared to a real robot cell where the tool stand have to be de-attached from the floor and re-attached to a new location and the robot have to be reprogrammed to get to the new location. In the simulation the repositioning problem and the required robot path is solved with a relocation of the tool stand and the affected robot paths points. A more critical example would be if the robot collides with a resource. In the real robot cell the robot and the resource may be damaged and this can result, if unfortunate with a non functional robot or resource. However in the simulated robot cell robots or resources will never be damaged, the solution is just a simple press on the reset button. A simulation based robot cell is if used correctly a more economical justified approach when constructing

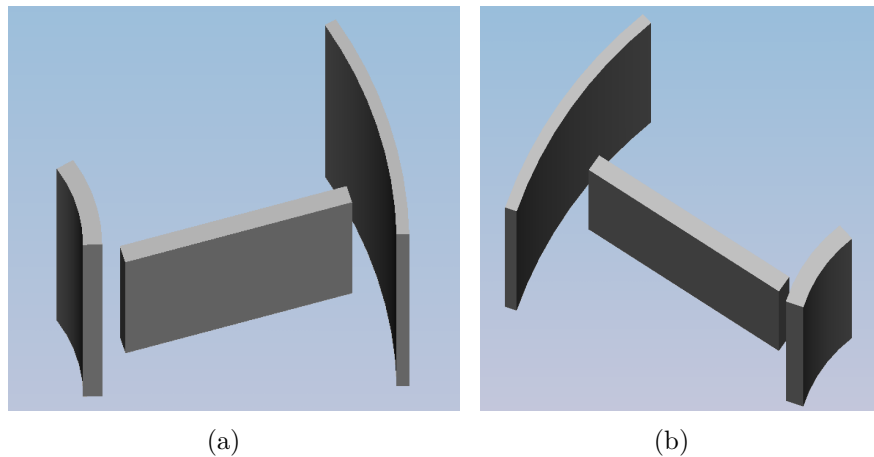


Figure 1.3: Simplified model of the three parts that will be fixtured together.

a real robot cell. A simulation's main benefit is that in a simulation there is a possibility to test if the planned robot cell layout, tools and parts can function together. The cost for this is negligible compared to if the robot cell were constructed and tested in reality.

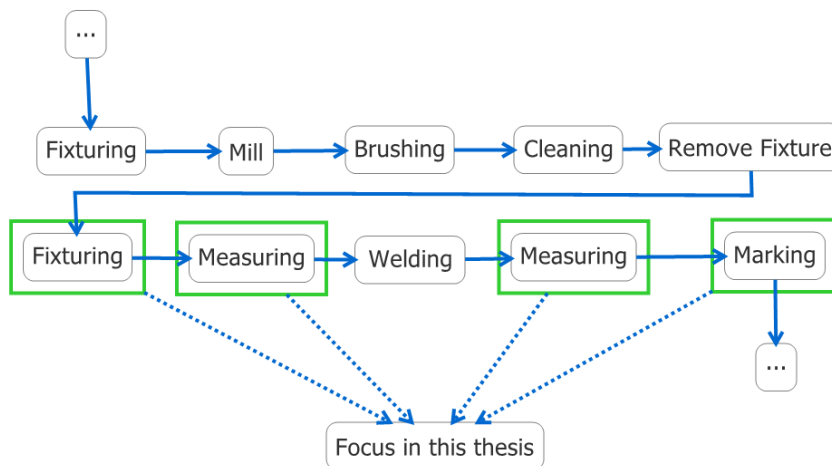


Figure 1.4: Part of the Turbine Exhaust Case production line.

The robot cell in this thesis is intended to be used as a test cell to evaluate different scenarios. This thesis focuses on the fixturing, measuring and marking processes, as seen in figure 1.4. When assembling the Turbine Exhaust Case a number of different processes are required. Instead of directly incorporating all the processes in the production line Volvo Aero uses a test cell in the first stage to guarantee that the production methods function properly.

2 Theory

2.1 Robotic Simulation

Rapid design and flexible manufacturing require that concepts, tools and methods of the process planning are fully applied. Using a computer to simulate the work cell before it is uploaded to the robots removes the guesswork whether a concept is possible or not. It also removes unrealistic expectations on technical specifications [3].

A robotic work cell simulation gives designers and decision makers the ability to find the optimal solution for a specific design, having been able to evaluate the different possible alternatives. If a change is to be made about a robot cell, the changes are incorporated much faster in a simulation compared to if it was done on the real work cell. Executing the simulation before uploading it to the robot controller gives a higher safety, both to operators and machines compared to if the programming were done online.

2.2 Manufacturing Processes

The manufacturing processes that will be used in the real robotic cell are not fully decided yet, in terms of exact vendor names and how long the different processes will take.

2.2.1 Eddy current inspection

Eddy current inspection is a technique that is non-destructive to the material that is being investigated, this is due to that the measurement is performed with electromagnetic fields. This method can be used to test for surface and sub-surface defects but it can also be used to test metallurgical properties. The technique is widely used in automotive, chemical processing, power, and aerospace industries to find cracks and other weld defects in components. Properties of the material such as permeability, conductivity and hardness can also be assessed with eddy current inspection measurements [10].

During the measurements an energized coil is moved over the particular component that is being inspected. This in turn induces eddy (circulating) currents on the conducting surface. In figure 2.1 the rotating currents can be seen. These induced currents are located at the surface near the coil. The magnetic flux from the eddy currents opposes the magnetic flux from the coil. When the eddy currents on the surface are affected by material variations or surface discontinuities, the impedance of the coil changes which also affects the phase of the voltage across the coil. If these changes are measured it is possible to identify characteristics of the object that is being inspected.

The frequency of the magnetic field determines the penetration depth into the material that is being inspected. When the frequency is increased, the penetration is reduced, which then increases the eddy currents near the surface of the object which can be read about in [9].

Volvo Aero will use a probe that is shaped like a pen, when used in the robotic cell the pen will not bend. The eddy current inspection tool will follow a distinct path along

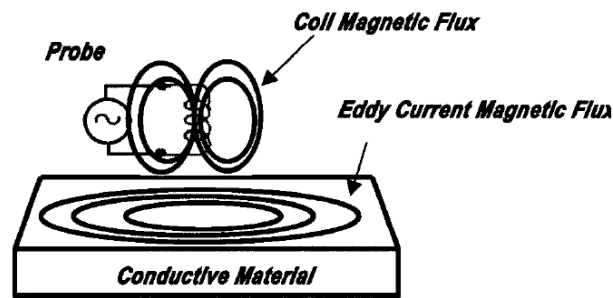


Figure 2.1: Induced eddy currents in object [9].

the weld seam. When following this path the probe will not be exposed to shear strain or normal strain, the pen will therefore not deform. Process Simulate is able to simulate this non deforming behavior. When used in the simulation this manufacturing process will be visually accurate and correspond to the real robot cell. During the simulation it will not be possible to get any measurement readings from the eddy current tool which would be the case in the real robot cell.

2.2.2 Direct Part Marking

When producing a component in an airplane engine it is due to safety reasons crucial that the products lifecycle is fully documented and that it is possible to follow the part throughout the manufacturing process. In industry today products from tires and engine pistons to surgical instruments are directly marked, the technique is also used in the struggle to eliminate counterfeited products. The industry has not yet established a clear standard for direct part marking which competes with RFID (Radio Frequency IDentification) marking, according to [22].

To mark a part there exist several different methods, e.g. electro-chemical etching, laser-marking and dot-peen marking. Which one to choose depends on several factors among them it can be mentioned:

Part Function

The function of the part being used is very important when selecting the marking method. For parts that are going to be used in safety critical operations it is recommended that non-intrusive methods are used when applying the marking to the part. Safety critical parts can be found in e.g. aircraft engines and control systems, high pressure systems and pyrotechnical systems.

Part Geometry

Flat surfaces are favored over curved surfaces when it comes to marking. This is due to that the marking process becomes simpler with a flat surface.

Part Operating environment

The marking method must survive the lifespan of the product that is being marked. Test should be performed in the expected conditions to ensure that the marking will be sufficient even at the expected product lifecycle length.

Part material

The primary selection criteria for the marking method should be the material type, more information can be found in [12].

Volvo Aero has chosen to use a dot-peen marking system to mark the parts, this method uses a marking needle that engraves the identification on the part. The marking equipment will be mounted on one of the robots. The marking needle will be oscillating in real life, this is something that will not be implemented in Process Simulate. It is possible to do but does not add any vital information to the simulation, only unnecessary complexity. The parts that will be marked in the simulation will not be visually altered even though dot-peen marking is an intrusive method.

2.2.3 Machine Vision

Machine vision is an umbrella term used to describe a wide array of vision system types. Machine vision systems is used in today's industry in order to process, analyze and understand images from an industrial process. Emulating human vision is not necessary when dealing with industrial problems, it might even overlook a simpler more elegant solution. In order to develop a vision system it demands a wide variety of different techniques and disciplines, among other: electronic engineering, mathematic engineering, physics etc. Automated inspection and measurements are the main fields for industrial vision systems and accounts for more than half of the total market. The reason for this is that retro-fitting an old inspection system is less expensive and simpler than building a new robot cell and integrating these two advanced techniques. In most applications the cost of the vision system is small compared to the total cost, however it is very important that the vision system does not restrain the other operations in the manufacturing system. False expectations on the machine vision system might be the case if not all implications are considered [2].

Volvo Aero will use a vision system to measure that the parts are fixtured correctly, which is possible to do in Process Simulate. No actual image will be recorded by the virtual vision system in Process Simulate. No alternation will be done to the parts when measuring the positions of the parts, which makes it a manufacturing process that is correctly simulated in Process Simulate.

2.2.4 Fixturing

Fixtures is a vital component in many manufacturing processes, they are used in machining and part fabrication. During the fabrication it is important that a part is fixated securely and that the part can be located accurately. There are generally five conditions that has to be fulfilled in order to fixate a part in a fixture [6]:

1. The part fixated should be completely restrained in the fixture.
2. It should be possible to accurately locate the part in the fixture.
3. The part should not deform when applying e.g. clamps to hold the part.
4. There should be no interaction between the fixture and tools that are used during manufacturing.
5. The part should be easy to remove and insert into the fixture.

It is common that the part is exposed to an excessive force to be held in place during machining. This can cause problems if the material in the part can deform, or if the surface would be damaged. It is common today that milling and lathe work are done automatically but not as common that a part is fixtured by a machine and not an operator. The design of a fixture can be split into two separate processes, fixture synthesis and fixture analysis. After the synthesis of the fixture is made, it has to be verified to work with e.g. the machining tools in the manufacturing process. Analysis can be divided into four different levels [21]:

Geometric analysis

Collisions between the fixture, machining tools and the part are here checked to ensure that the fixture will work as intended.

Kinematic analysis

The correct location of reference points are important to reassure that the work-piece is located correctly. The fixturing points should also be confirmed to be positioned correctly to oppose forces caused from the machining.

Force analysis

When the fixture is closed, the reactive forces at the fixturing points should be investigated to make sure that the forces are adequate to hold the part in place during machining.

Deformation analysis

If the part being fixtured is easily deformed it is important that a deformation analysis is performed to determine the plastic or elastic deformation of the part when it is e.g. being clamped or machined.

Volvo aero uses a fairly complex fixture to clamp the three parts, however this will cause no problems when simulating it in Process Simulate. The parts will be attached to the fixture by internal commands and not by the clamping forces like in the real robotic cell.

2.3 Software

2.3.1 Process Simulate

Process Simulate is part of the larger Tecnomatix application suite from Siemens PLM. Process Simulate can be used to design, analyze, simulate and optimize processes in the manufacturing industry. Other software in the Tecnomatix family includes for example Plant Simulate and Process Designer.

2.3.2 Architecture

Process Simulate depends on the following three applications to function properly:

Client

Process Simulate itself or another application in the Tecnomatix family.

eMServer

A dedicated application which controls the communication between the clients and the database.

Database

An Oracle database that store project information.

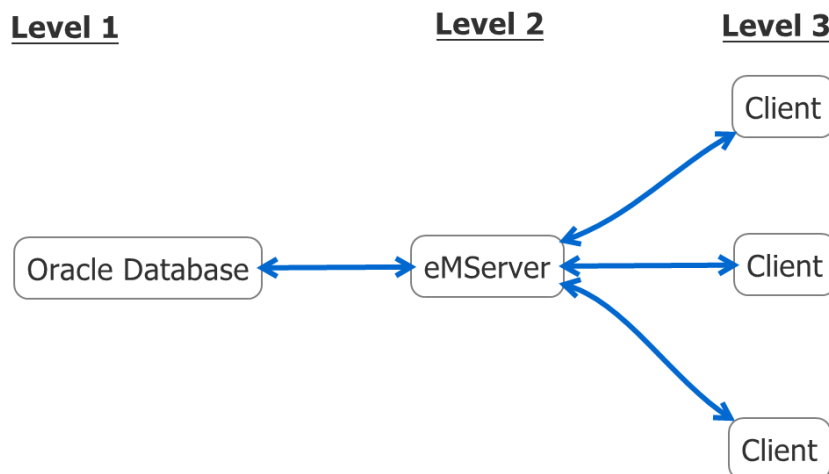


Figure 2.2: The communications between the different tiers.

The communication between the different applications can be seen in figure 2.2. According to [18] the database is at the lowest level of three and makes sure that the data is kept persistent. The Oracle database is used for its ability to manage data and control access for different users.

The eMServer is the core of the software, it provides everything from element modelling to programming logic of the manufacturing process. It also supplies services to the clients such as connections to the database and the behavior, e.g. resources in the manufacturing process. The eMServer perceives each client as a standalone client which is not conscious about the multiplicity of the other clients.

The third level is any application that utilizes the eMServer. As an example there is Process Simulate and Process Designer. In practice any software that uses the API (Application Programming Interface) can function as a client in the architecture. The Process Simulate client working environment can be seen in figure 2.3

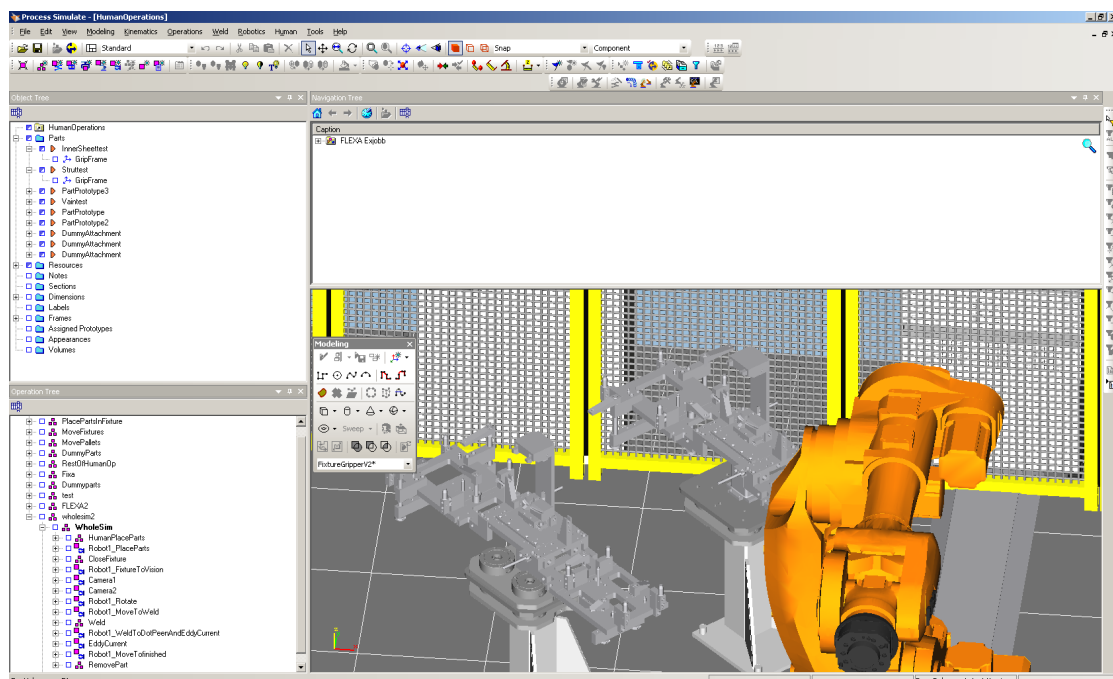


Figure 2.3: Process Simulate working environment.

2.3.3 Data Structure

Process Simulate is as previously mentioned a multi-user tool which allows several users to work on the same project but not on the same branch at the same time. The data is stored in the database and in the system root. The system root is a shared folder on a

file server, which all clients has access to. It is here e.g. all the CAD (Computer Aided Design) model data is stored.

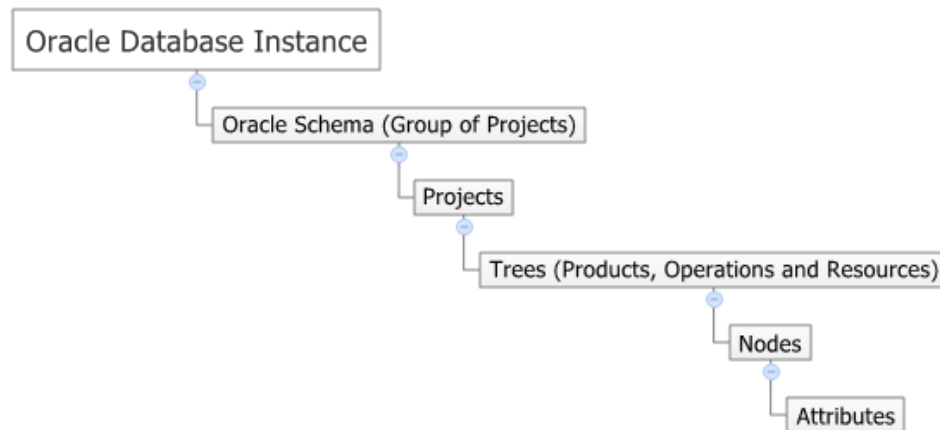


Figure 2.4: The Oracle schema.

In figure 2.4 the following objects are used to describe the data structure [18]:

Schema

The schema contains the projects. Several schemas can be created with different numbers of projects. An user can only open the projects available in the user-defined schema. A schema can be seen as a container for the projects.

Project

The project contains all the objects that is required to fully characterize the manufacturing process. When accessing a project the user can only work with one project at a time.

Trees

Trees are subdivided into Products, Resources and Operations, e.g. a product can be an engine, a resource can be a robot and an operation can be a robot sequence.

Nodes

Each tree can be represented as a node. The difference between a tree and a node is that a node has a specific type, e.g. product, resource or operation.

Attributes

Each node contains attributes. An attribute can be e.g. the robot cell location of a CAD model, an attachment between a set of products or the color of a resource in the robot cell.

2.3.4 Delmia Automation

Delmia is another type of PLM software and was introduced in the year 2000. Delmia is targeted towards the manufacturing industry where it is being used to simulate virtual models of different manufacturing processes. It is also possible to optimize and control the production systems. Delmia uses kinematics for the different robots and the plc connected devices. Delmia can simulate both large and small manufacturing cells, where several products and robots are used together to describe the entire cell. Delmia has a library with the most popular robots available [8].

2.4 Supervisor Control Theory

The Supervisor Control Theory developed by Ramadge and Wonham (1987) is a general approach for controlling discrete event systems. Many technical systems in everyday life can be described using discrete states and events that change these states. Events are transitions between different states and they can be controllable or un-controllable. If an event is controllable it executes in a deterministic way while uncontrollable events can occur randomly. A dead-lock state is a state where no transition between states can occur.

Finite state Automaton is a graphical representation of a discrete event system. It contains a finite number of states, and transitions (events) between these states. In figure 2.5 a simple example of an automaton is presented. This automaton contains two states and two events, the states could for example represent if a door is open or closed. The events could be a person pulling or pushing the door.

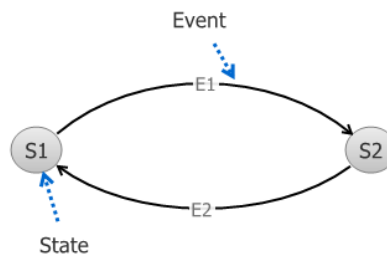


Figure 2.5: Example of a finite state automaton.

A plant describes everything that a modelled discrete event system can accomplish. A plant should only consist of the necessary states and should not include anything that can be confused with the specification. This is to prevent unnecessary states and large calculations. It is not clear how a plant should be modelled. The level of detail and the structure of the language are common problems when creating a plant [7].

The full synchronous composition between two finite state automata is denoted $A_1 || A_2$ and it states that events common in both automata can occur simultaneously. A specification describes what the given discrete system should do. Specifications can be partial or

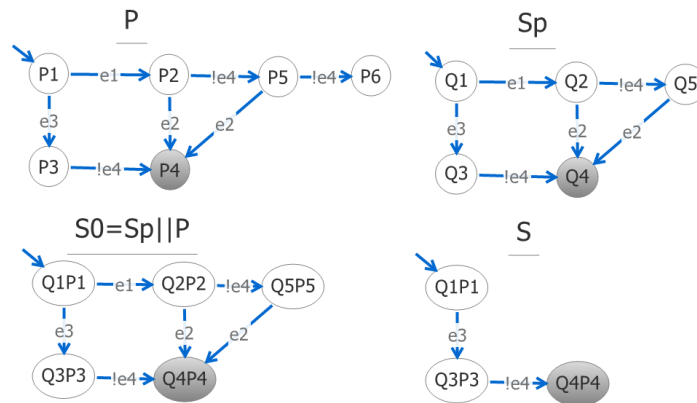


Figure 2.6: Example of a plant, supervisor, synchronization between them and the synthesised supervisor, with inspiration from the introduction to discrete event systems course held at Chalmers University of Technology.

total. A partial specification only describes a part, e.g. the preferred part of a plant, while a total specification describes the entire plant. States can be marked in the specification to prevent dead-lock states in the final supervisor. A marked state is a desired state which the specification wants the plant to end up in, a specification can have several marked states.

In figure 2.6 an exclamation mark before the event name indicates that the event is uncontrollable. The plant, marked with P , in figure 2.6 contains six states, labeled from $P1$ to $P6$ with its initial state $P1$ and a marked state $P4$. To the right of the plant an automaton of a specification is shown, it is similar to the plant model but does not contain the uncontrollable event $!e4$ from state $Q5$. $S_0 = S_p || P$ is the synchronized automaton between the plant model P and the specification S_p , it can be noted that the state name of the new synchronized automaton now contains both the state which the plant and the supervisor is in. The state $P6$ in the plant is removed because the specification S_p does not contain the event $!e4$ in its state $Q5$. The event still exists in the plant so it would be possible for the plant to execute the event, even though the specification S_p says otherwise. This is where the need for a supervisor becomes clear. Supervisors are created from plants and specifications. A synthesized supervisor is only able to avoid un-controllable events that exist in forbidden states. When the supervisor is synthesized all dead-lock states are removed. The synthesized supervisor in figure 2.6 is denoted S and the small automaton is the result of removing uncontrollable states which else could have made the plant end up in a dead-lock state. More information about Supervisor Control Theory could be found in [7].

To implement this theory in large industrial processes a computer tool is needed to calculate the supervisors, because the supervisor tends to hold a large set of states. One of this tools is Supremica, which are described in section 2.5.

2.5 Supremica

Supremica is a program developed at Chalmers University of Technology. Supremica is used for synthesis, verification and simulation of discrete event systems. The finite state automaton is the basic model of Supremica and the software is under constant development. Supremica can utilize extended finite state automata when setting up plants and specifications. This is similar to regular finite state automata but contains additional features such as guards, actions and variables. The guards on the events contain conditions whether an event is allowed to execute or not. Supremica manages the automata by two different methods. The first method uses modularity to simplify the main problem into smaller sub-problem that together solves the main problem. The second method addresses the problem with an efficient data structure which is a binary decision diagram [1]. In figure 2.7 the Supremica environment can be seen, for further information see [13].

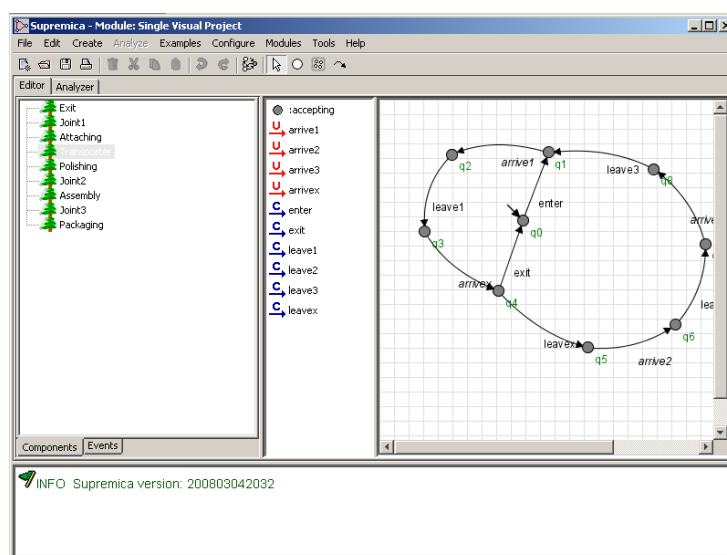


Figure 2.7: Supremica working environment.

2.6 Sequence Planner

Sequence planner is software developed at Chalmers University of Technology. It incorporates a flexible way to setup operations and their pre- and postconditions. The operations can be shown from several perspectives with different graphical representations of the pre- and postconditions. The main focus of Sequence Planner is to model operations that represent starting and finishing of processes. An operation's graphical representation is a box with its name on it. After at least two operations has been created it is possible to organize the operations such that the constraints of the model is fulfilled. This can either be done by writing pre- and postconditions or by drawing a line between two operations.

The reason for this is that some processes are easier to visualize graphically and others are better understood if a condition is written. Sequence Planner can export its data to Supremica for verification and optimization. Sequence Planner is still under development. Figure 2.8 shows the working environment of Sequence Planner. The intended workflow of Sequence Planner [14]:

- Gather general data that could be used to define clear borders on the model that will be created. This data can originate from e.g. a specification or data from a robot cell.
- A model is created in Sequence Planner with the use of the collected data, this model is then analyzed in Supremica for verification and optimization.
- The final stage is to convert the resulting model, which is verified and optimized to PLC-code that can be used in the manufacturing cell.

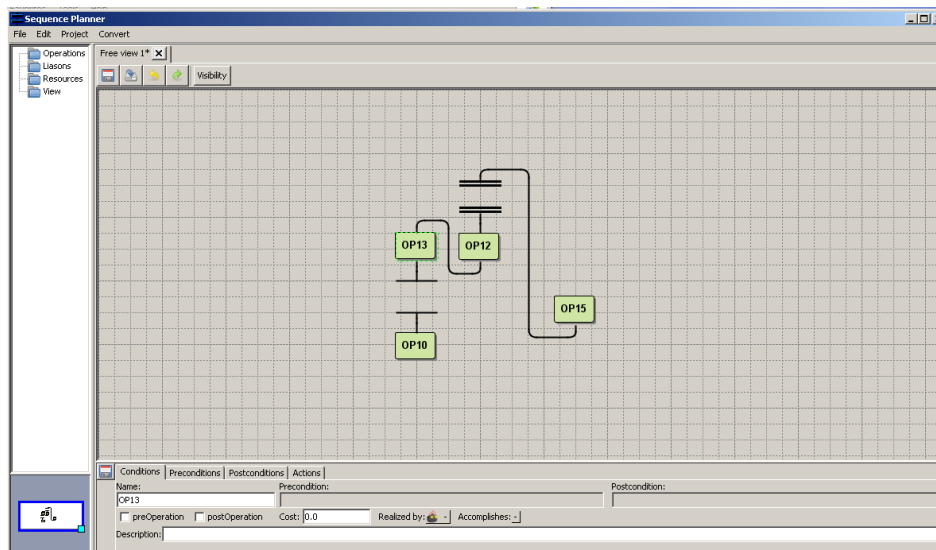


Figure 2.8: Working environment of Sequence Planner.

3 Implementation

3.1 Installation of Process Simulate

In section 2.3.1 it can be read that in order to run Process Simulate, an oracle database needs to be installed and setup properly, the eMServer must be running and the client has to be installed. Process Simulate version 9 has been used during this thesis work. When the master thesis work began no working environment was installed for Process Simulate. In order to start the project the following had to be performed:

- A windows 2003 x64 server was installed on a virtual environment.
- The oracle database was installed. Various scripts were run to make it compatible with the eMServer.
- The Process Simulate eMServer software had to be installed in the virtual environment.
- Process Simulate V9 clients were installed on the computers that were going to be used for creating the simulation.
- The networking environment in windows was configured in order to use the Process Simulate clients.

It is recommended by Siemens that the client/server configuration is setup in a network domain, due to restrictions on the Chalmers network this setup was impossible to use. Process Simulate was instead configured under the Windows workgroups environment.

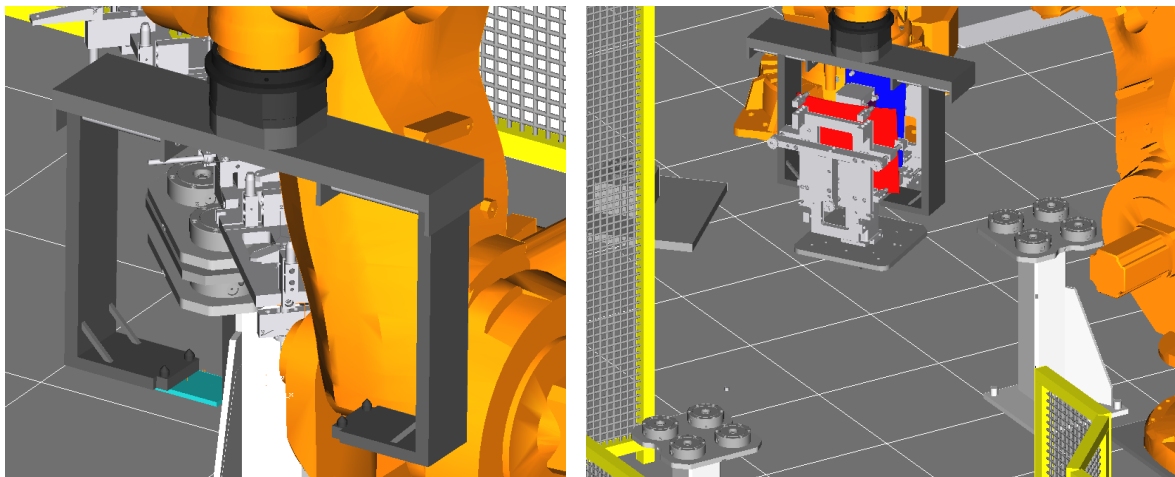
3.2 Modelling and kinematics in Process Simulate

Process Simulate contains a simple tool used for CAD modelling, due to license problems it has been difficult to import different CAD data formats into process simulate.

3.2.1 Modelling of grippers and pedestals

On each movable fixture or pallet table top a mounting system from Schunk [17] was used. The remaining problem was to decide height and location for each pedestal. On the locations where human interaction was necessary a height according to [23] was used. In this case it was decided to be 930 mm. The sequence of operations was the main influence on the pedestals placement in the robot cell.

When modelling the gripper it can be noted that the lift point on the fixture is placed under it. This is to minimize the forces acting on the foldable sides of the fixture, this can be seen in figure 3.1 where the gripper arms extends under the fixture to lift it.



(a) The modelled gripper

(b) The gripper holding a fixture

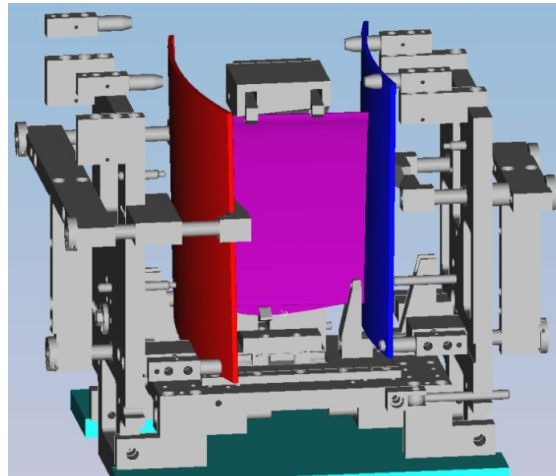
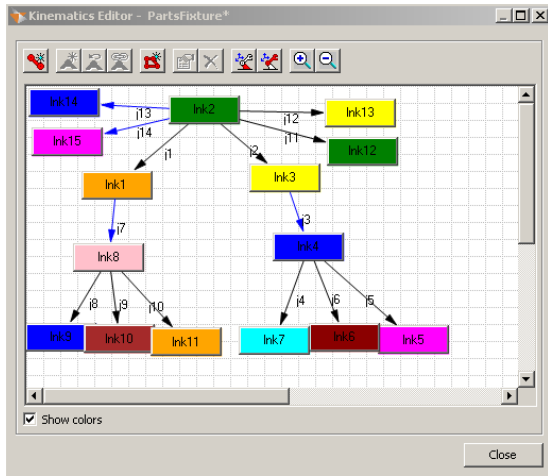
Figure 3.1: The gripper used by the robot to move fixtures and pallet tables.

3.2.2 Adding kinematics to the fixture

The fixture supplied by Volvo Aero contained 16 different movable joints, an image of the fixture can be seen in figure 3.2(b). A joint can be either prismatic or rotational. To create this kinematic chain in Process Simulate, the kinematic editor is used. The order of the kinematic chain is defined by the relationship between the different links. The parent link determines how the child link will move. When the correct kinematics is setup the device can for example be defined as a gripper or a weld gun. The reason to define a device is that these specific types can perform different tasks in Process Simulate e.g. to produce a spot weld a weld gun is necessary. The kinematic chain of the gripper can be seen in figure 3.2(a). As an example **Ink2** in figure 3.2(a) is the base of the fixture, and by following the kinematic chain to **Ink13** shows that it is a prismatic joint moving the entities associated with **Ink13**.

3.2.3 Defining the gripper as a gripping tool

To be able to use the gripper in pick and place operations the tool has to be defined as a gripper. To achieve this, the tool center point and the base frame, where it is attached to the robot, has to be chosen. In figure 3.3 the gripping entities in this case the gripper arms are highlighted in light blue. The yellow section is the entities that Process Simulate should not check for collisions with. When defining the gripper, the tool center point was placed in the center of the gripper to assure that rotations of gripped objects were simple to achieve.



(a) the kinematic links defining the fixture

(b) The fixture with inserted parts

Figure 3.2: Kinematic links and the fixture.

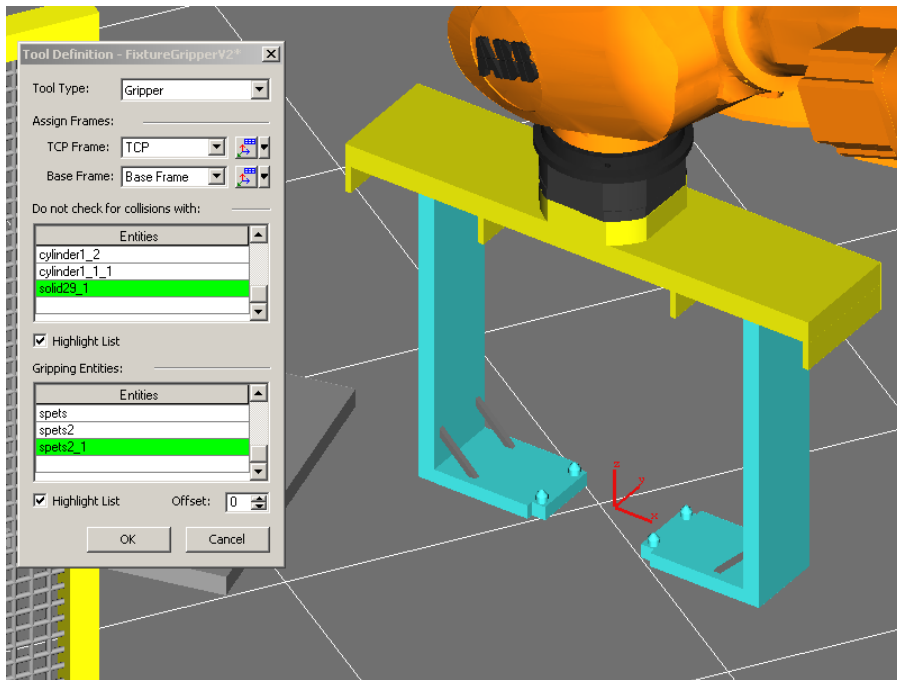


Figure 3.3: Screenshot showing the tool definition of the gripper.

3.3 Robot placement

A very rudimentary sketch of the robot cell layout was supplied by Volvo Aero, this sketch contained the parts and which robots to use. There were two robots, one ABB IRB7600 and one ABB IRB6640. The first one is a power robot that is designed to lift heavy loads and the second is a smaller more versatile robot. In the cell the IRB7600 is referred to as robot 1 and the IRB6640 is known as robot 2.

3.3.1 RobotSmartPlace

Before it is possible to build robotic movement, locations of the resources in the robot cell must be decided. The process started with placing non moving resources, i.e. tables and fixture pedestals. This placement was made arbitrary but with the intention to maximize the robot's flexibility. In this stage working height for certain tables were decided, basic data for this was collected from [23].

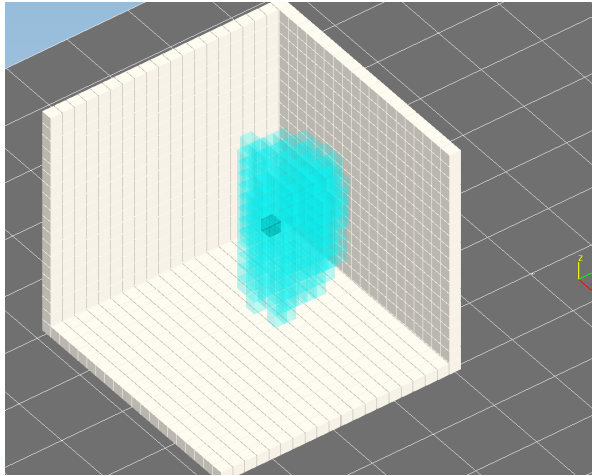
The resources were placed in vicinity to a thought center point of the IRB7600 robot, 2500mm from the center of the allocated cell. According to Volvo Aero the cell size should be a maximum of 10000×5000 mm.

To find an appropriate place for the robot's placement a program feature of Process Simulate called RobotSmartPlace were used. This application gives the user the ability to test a large variation of different placements for the robot. A number of crucial locations which the robot had to be able to reach were inserted as via points. A limitation in RobotSmartPlace is that it has no ability to change the tool center point when doing the simulation. In order to solve this problem RobotSmartPlace was run several times with different tools. The image acquired was overlaid and finally sketched in three dimensions. This was made to get an easier overview of the solution which can be seen in figure 3.4. It can be noted that the actual position chosen as placement for the ABB IRB7600 robot is marked with a darker box in figure 3.4.

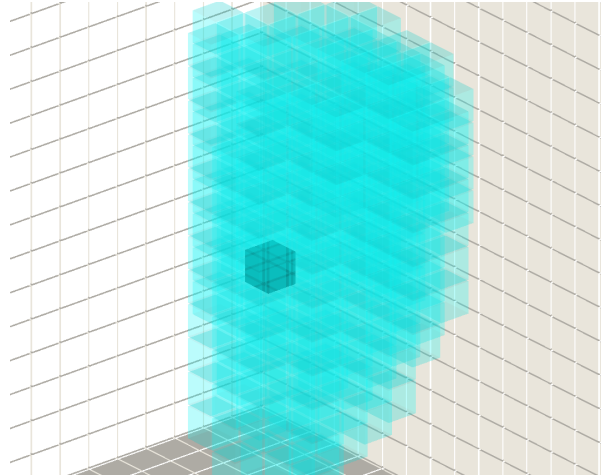
3.4 Sequence based simulation in Process Simulate

To be able to simulate a sequence in Process Simulate a **RobCadStudy** is required. This is a data type that contains information such as resources, parts and operations. In order to perform an event driven simulation in Process Simulate a different data type is needed, namely a **LineSimulationStudy**. The difference between the two is among other that a LineSimulationStudy does not contain parts but appearances. Appearances have to be generated each time the study is loaded or during the simulation. It is generally in the RobCadStudy the robotic movements are created and the collision checks between objects and robots are made.

The sequences are built in a Gantt chart and then linked together to form the simulation. Processes can be run in parallel or in serial. It facilitates that the robotic movement in the RobCadStudy is built correctly, because the increased complexity in the LineSimulationStudy makes the editing process more cumbersome. Figure 3.5 shows the sequence editor



(a) Overview image



(b) Close up image

Figure 3.4: Three dimensional visualisation of the possible locations for robot 1.

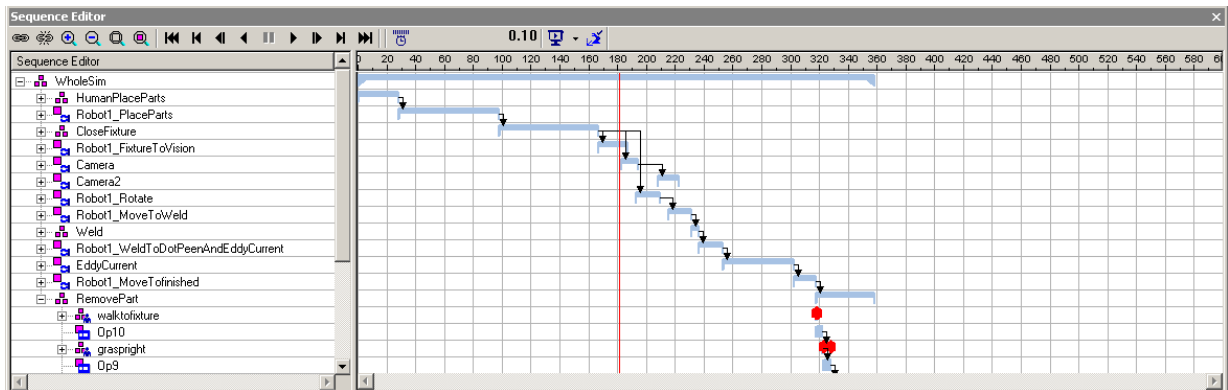


Figure 3.5: Sequence editor.

where the different sequences are placed and organized to make up the RobCadStudy operations. The figure shows that some sequences are run in parallel and some in serial. In this robot simulation most of the sequences are run in serial, due to the fact that robot 1 has many pick and place operations. While robot 2 has to wait for robot 1 to finish its operations.

Human operations

In Process Simulate it is possible to have a human model in the simulation. The length and weight of this individual can be set in the Process Simulate Human tool. According to [16] the average height for men in Sweden is 179.5 cm and the average weight are 82.4 kg, this has been adapted to the model. Creating human simulations in Process Simulate is very demanding, but the model can give a rough approximation if the working height in the different sequences is possible or will be too strenuous for the worker. No ergonomic studies have although been carried out in this master thesis. In figure 3.6 it can be seen that it is possible for a human to place one of the parts in the pallet.

Process Simulate Human cannot be used in Line Simulation Studies, therefore these operations needs to be recreated without using the human.

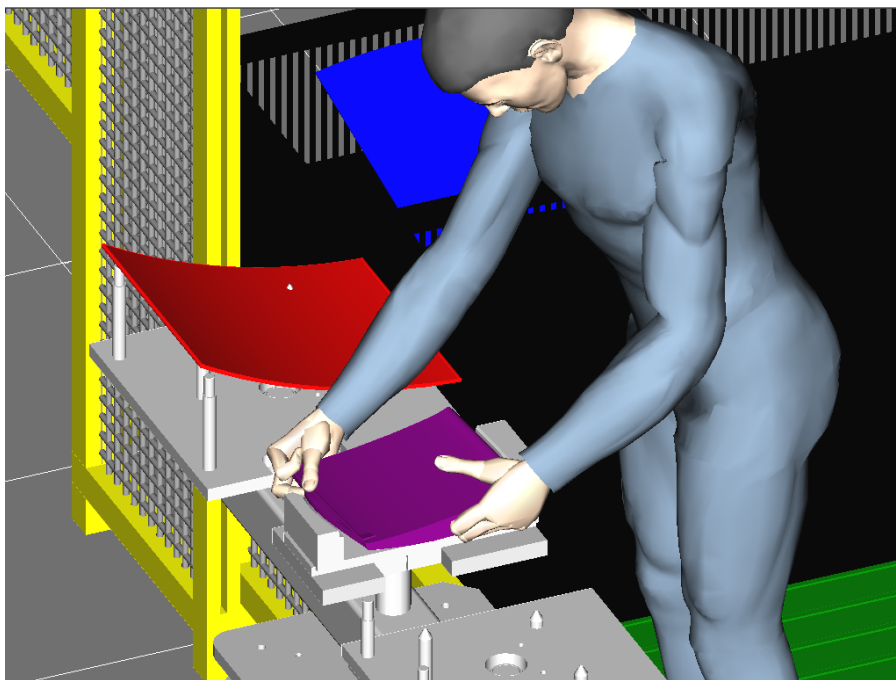


Figure 3.6: The Human Model placing one of the parts in a pallet.

3.5 Event based simulation

What distinguishes the event based simulation from the sequence based simulation is that the event based simulation depends on logic from the processes and on different events that occurs while running the simulation. The sequence based simulation only depends on time and a pre-defined sequence flow. The different events that can occur in an event based simulation can be signals from robots or signals from other PLC connected devices. This can create several unique solutions for one robot cell, compared to a sequence based simulation where there is always only one solution. The event based simulation is used to simulate how a real robot cell should function.

A LineSimulationStudy is created from a RobCadStudy, see section 3.4 for more information, using a tool called Merge Studies. This results in a new and modified sequence flow with default sequence pre-conditions for all sequences from the RobCadStudy. This conversion makes it possible to create the logic and the different signals in the robot cell.

3.5.1 Logic

The Logic in a robot cell simulation is controlled by three different logic types, which can be used to describe the behavior of the real robot cell. Each of these works independently of each other, but they can also be combined. The combination between modules and sequence transition conditions corresponds the most to how a real robot cell functions due to that the modules have a structure reassembling a PLC (Programmable Logic Controller). The three different logic types in Process Simulate are:

Sequence transition condition

The sequence transition condition controls the devices and operations that cannot use PLC signals, e.g. this type of condition can control when a part or a human operation is activated. Another example could be that two parts can be attached together by a transition condition when a signal from a welding robot is set to true, in figure 3.7 the transition condition window is shown.

Logic block

The logic blocks controls the robots logic and processes the input signals. For example a logic block can be used to model a swimming pools heating system, which changes the temperature to a predefined value based on inputs that are received from a thermostat. The logic block is the controlling device that decides when to activate the heating system. An overview of a logic block can be seen in figure 3.8.

Modules condition

The modules condition controls all devices connected to the PLC, such as robots and PLC connected devices. Modules are specified by signals from the entire robot cell, the modules consist of expressions that sets a result signal. The module logic can consist of if-statements and hierarchy. In figure 3.9 the module editor and the module viewer can be seen.

Sequence transition conditions

The sequence transition conditions in this thesis are used to keep track of which parts that should be generated throughout the process and which fixtures that should be closed, opened or welded. Dummy transition operations are used for the parts, because Process Simulate deletes a part from the simulation when all sequences of that part are completed. This could be when a part is generated and moved to a drilling station, when that operation is completed that part should not disappear before it is packed and not when the part is placed in the drilling station. There are also unique conditions for all sequences, this is a bulletproof way to prevent that more than one sequence transition condition is true at the same time in one operation.

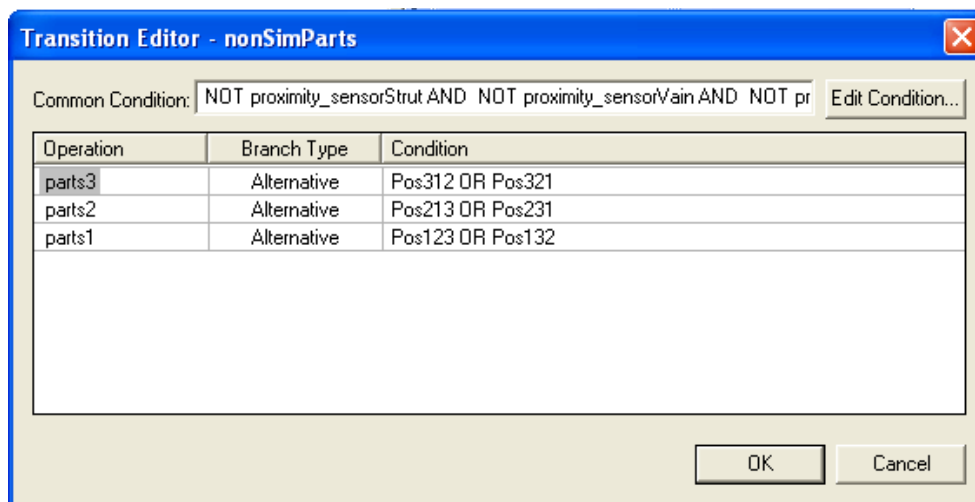


Figure 3.7: Transition editor window for the Strut, Vain and Inner sheet parts.

In figure 3.7 there is one common condition for the three possible part types and three conditions determining which part type sequence that starts if the signals are true. The common condition consists of an expression that controls if there are parts placed on the pallet table or in the fixture and that the fixture is in an open position.

Logic blocks

The logic blocks are used to connect and process the resource signals such as the robot signals. The logic block consists of entry and exit signals. The logic block uses constants and parameters, which are all uniquely defined. The parameters consist of expressions that sets the exit signals. In this thesis the logic blocks are used for the robot's select program and start program signals.

In figure 3.8 the logic block for robot 1 is displayed. There exist one entry signal for every selectable robot program, some constants that are used when calculating the parameters

and setting the exit signals. The main logic for robot 1 consists of two parameters **SelProg** and **StaPrBol**. The value expression behind **SelProg** consists of a boolean signal that is true when the program entry signal is true. This entry signal is multiplied with its own program value so that **SelProg** becomes larger than zero. When **SelProg** is larger than zero **StaPrBol** is set to true. The connections between the logic block signals and the robot signals:

- **SelProg** is connected to the exit signal **SelProgNum**.
- **StaPrBol** is connected to the exit signal **StaProg**.
- **SelProgramNum** is connected to the robot's programNumber signal.
- **StaPrBol** is connected to the robot's startProgram signal.

Modules

The Modules can be considered to act as an internal PLC with hierarchy that evaluates signal expressions. The modules can be used to define a signal as a result of a logical expression which consists of other signals and operators. All expressions are evaluated in every cycle and follows a hierarchy that prioritize from top to bottom as seen in figure 3.10. All modules can have if-statements that decide if that module is allowed to change a signal.

In figure 3.10(a) a section of the modules and the modules hierarchy for the robot cell is shown. In figure 3.10 the **RESETROBOT1** module consist of a condition that sets all robot 1 signals to false. This gets triggered from an if-statement shown in figure 3.9, when robot 2 sets its internal startProgram signal to true. This method is used to acknowledge that the signal from robot 1 has been received correctly and that robot 2 has started its chosen robot program and sets the received signal to false. The structure of the **RESETROBOT1** module contains all signals that can trigger robot 2 startProgram signal. These signals do not need to be set individually because the modules reset each other.

3.5.2 Pallet switch

The whole Turbine Exhaust Case part consists of 13 sections where each section contains the three parts that are fixtured in this thesis. Some of these sections are unique and therefore demands different fixtures. To be able to implement a simplified problem in the simulation, a pallet and fixture switching operation were constructed. In the simulation there is a possibility to change between three different pallet and fixture combinations.

The logic behind the pallet switch sequences came from reasoning. This was due to that the smallest possible solution was a manually derived supervisor, which can be seen in figure 3.12. In figure 3.11(a) the initial setup is displayed, position 1 is the pedestal in

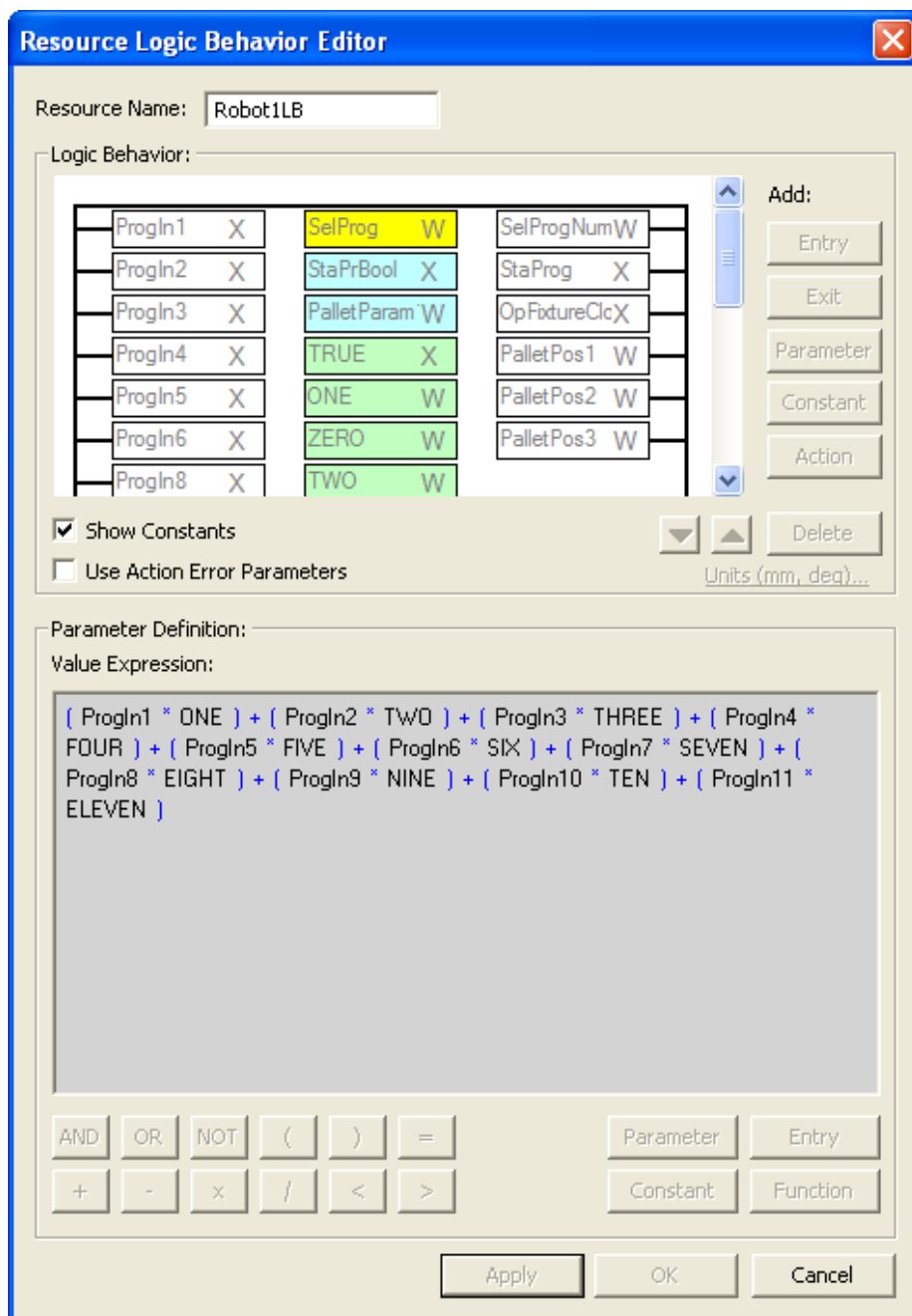


Figure 3.8: Resource logic behavior window for robot 1.

the working area. The working area pallet table stand will always be affected by a pallet switch, while the other two pallet table pedestals can be active or passive. The robot 1 and robot 2 common pallet table are being used as a temporary position while changing the pallets. If a pallet table pedestal is passive, it always needs knowledge about which pallet tables that are switching places. The side effect of this is that the passive pedestal will not have unique events which trigger the switching process.

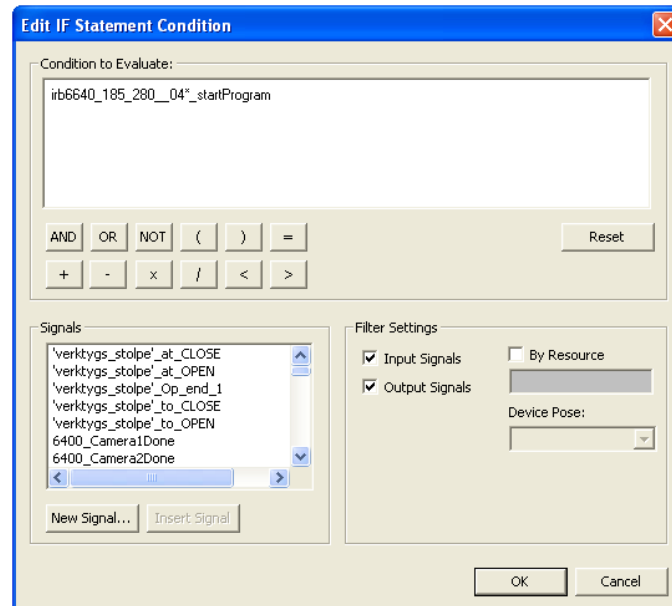
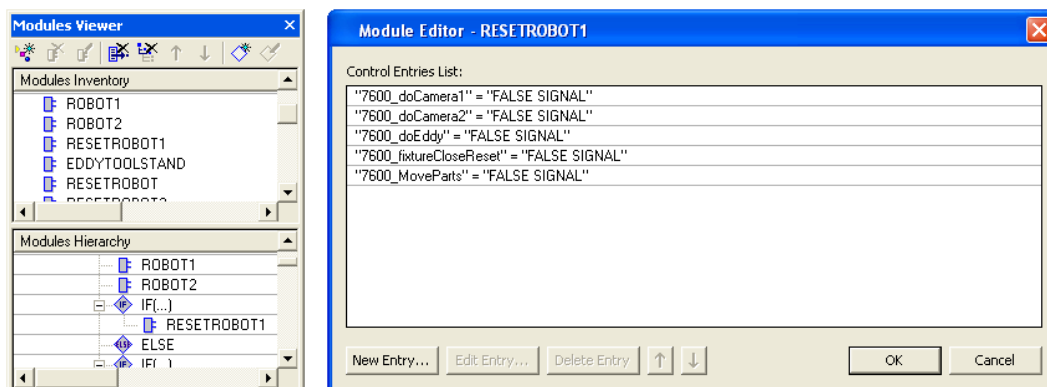


Figure 3.9: Edit if statement condition viewer for RESETROBOT1.



(a) Modules viewer

(b) Modules editor

Figure 3.10: Modules viewer for the robot cell & modules editor for RESETROBOT1.

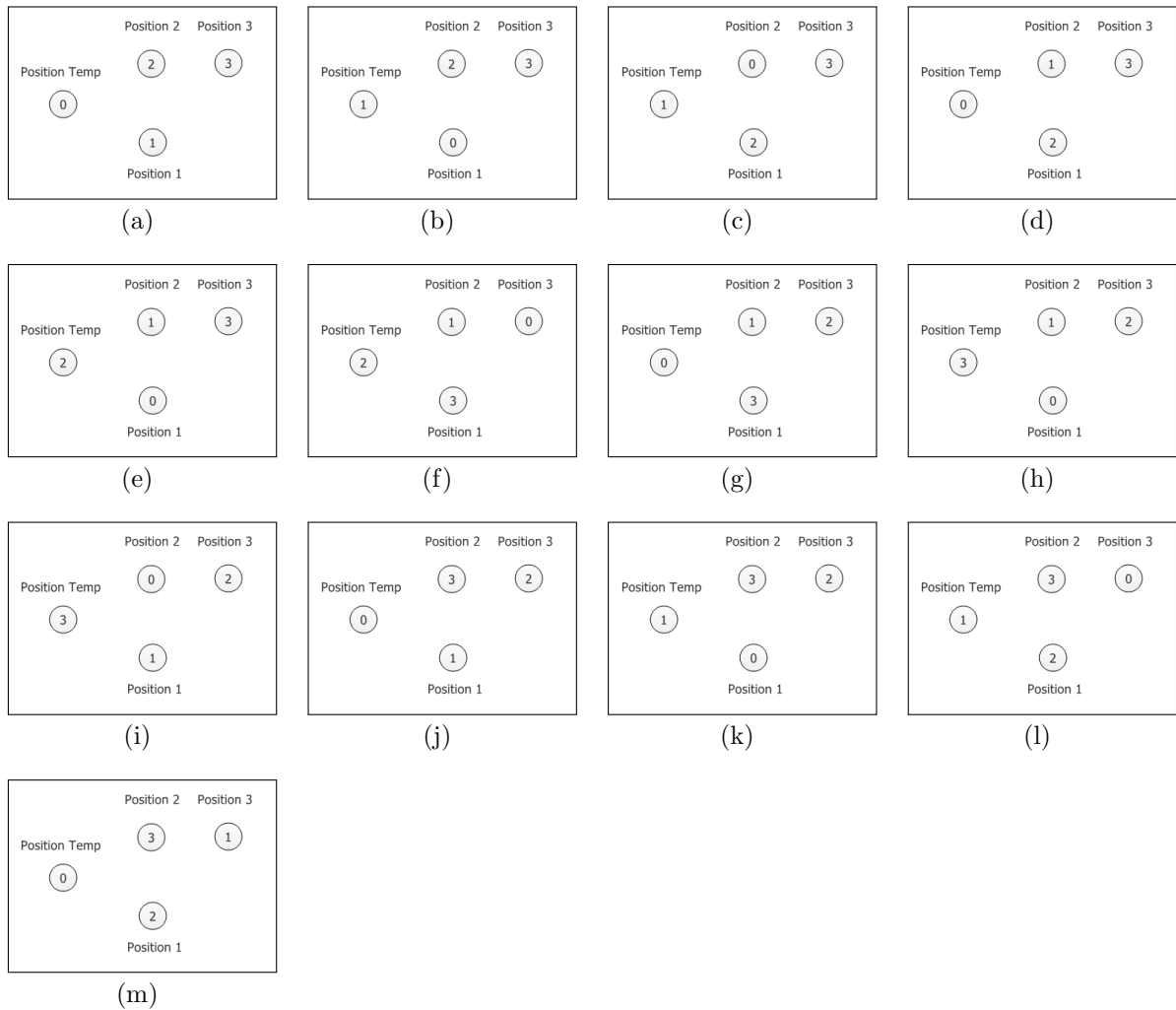


Figure 3.11: Sequence flow for the pallet switch operations with type 2, 3, 1, and 2 selected.

In Figure 3.11 the flow of the pallet switch operation is shown, by selecting type 2, 3, 1 and 2 in that specific order. The pallet position is defined with text close to the circles and the numbers inside the circles are the current pallet placed on the pedestals.

Description of the flow for the pallet switch sequences

The numbered list describes the operations when changing pallet types from type 1, 2, 3, 1 and 2.

1. In figure (a) the initial position of the pallets are shown with pallet 1 on pallet position 1, pallet 2 on pallet position 2 and pallet 3 on pallet position 3.
2. In figure (b) the selected type are changing to type 2 from type 1. The first robot

sequence is to transfer the pallet placed in position 1 to position temp. In this case pallet 1 is moved.

3. In figure (c) the robot continues with the operations and moves the selected pallet to position 1. Pallet 2 moves from position 2 to position 1.
4. In figure (d) the robot moves the pallet standing in the temp position to the available pallet stand. Pallet 1 from position temp to position 2. After this sequence the pallet switch is completed for the selected type.
5. In figure (e) type 3 is selected. As always robot 1 transfers the pallet standing in position 1 to position temp. Pallet 2 from position 1 to position temp.
6. In figure (f) the robot knows where pallet 3 is placed from the logic signals. The robot moves pallet 3 from position 3 to position 1
7. In figure (g) the robot moves the fixture in position temp to the available pallet stand. Pallet 2 from position temp to position 3. The operations are complete for type 3.
8. In figure (h) type 1 is selected. The robot moves pallet 3 from position 1 to position temp.
9. In figure (i) the robot knows that pallet 1 stands in position 2 from the signals. The robot moves pallet 1 from position 2 to position 1.
10. In figure (j) the robot moves pallet 3 from position temp to position 2. The operation is complete for type 1. The difference here compared to the initial position is that pallet 2 and pallet 3 have switched places. Pallet 2 is standing in position 3 and pallet 3 is standing in position 2.
11. In figure (k) type 2 is selected, this is done to show that the logic behind the pallet switch functions as described in 3.5.2. The robot moves pallet 1 from position 1 to position temp.
12. In figure (l) The robot knows from the signals that pallet 2 stands in position 3 and not in position 2. The robot moves pallet 2 from position 3 to position 1.
13. In figure (m) The robot moves pallet 1 from position temp to position 3 and the sequences are completed.

In figure 3.12 the automaton supervisor for the pallet switching function is displayed. The state name changes according to which pallet that are in which position, e.g. state 231 means that pallet 2 stands in position 1, pallet 3 stands in position 2 and pallet 1 stands in position 3. The events are marked with arrows and describe which pallet type that should be placed on position 1. The events that occur are highlighted with red. This is

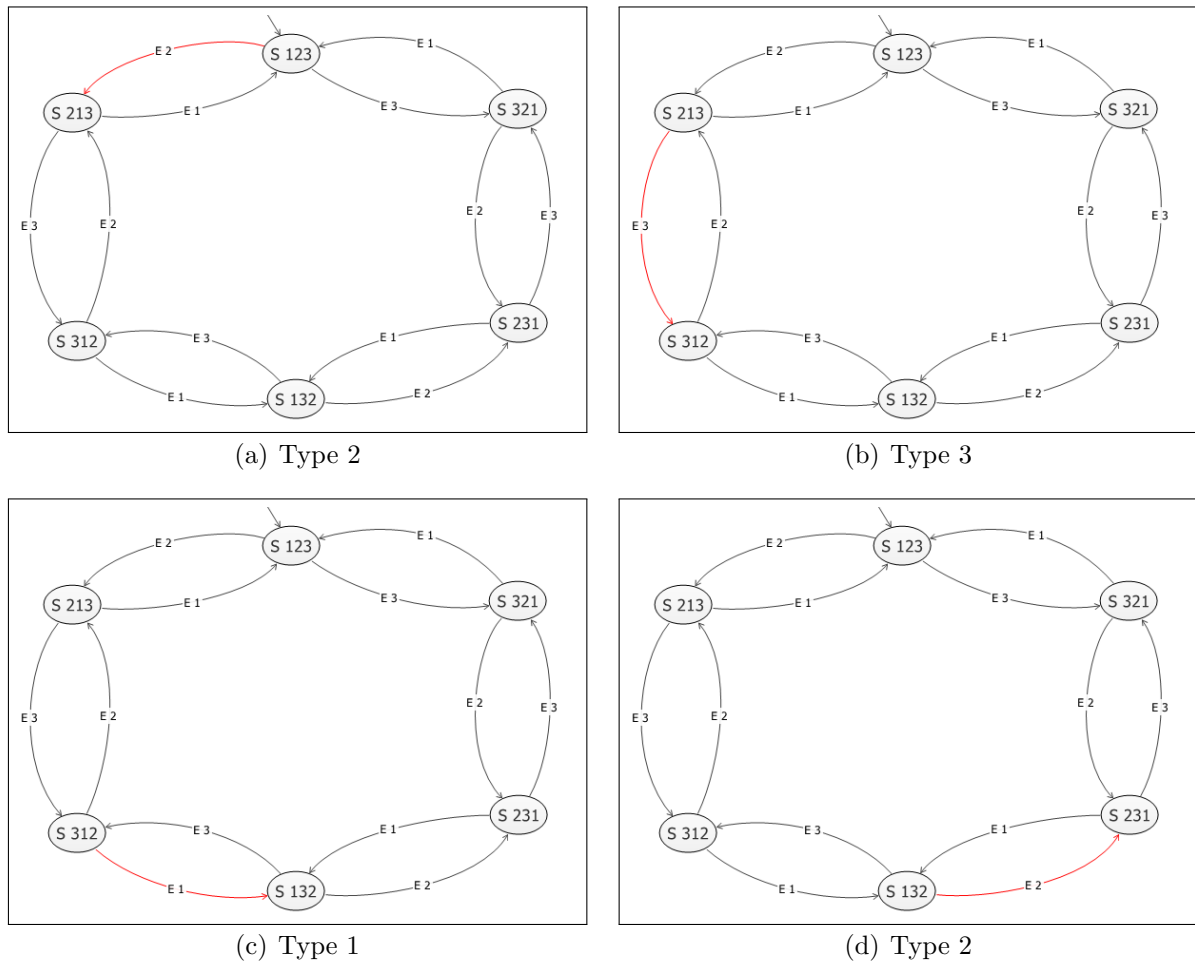


Figure 3.12: The automaton supervisor for the pallet switch function.

the events used in figure 3.11. The signals in the logic blocks implement this supervisor. The fixtures are moved in synchronization with the pallets, this means that the fixtures are being switched during the same procedure as the pallets.

3.6 Modelling of the robot cell sequences using Sequence Planner

As an alternative to create the sequence logic on paper or directly in Process Simulate the possibility exists to use Sequence Planner to setup the relations between the different operations. Sequence Planner supports hierarchical architecture and inputs, which makes the design process simpler when not the whole process is yet known or decided. An operation is something that executes in the robot cell, e.g. a sensor that goes high or a drill operation is completed. In this particular robot cell all the operations that was thought to be used, was created in Sequence Planner in no particular order. This is to

keep the order of the sequences as loose as possible, in order to utilize the optimization option in the software. In figure 3.13 the created operations can be seen, it can be noted that no relations between them are yet defined.

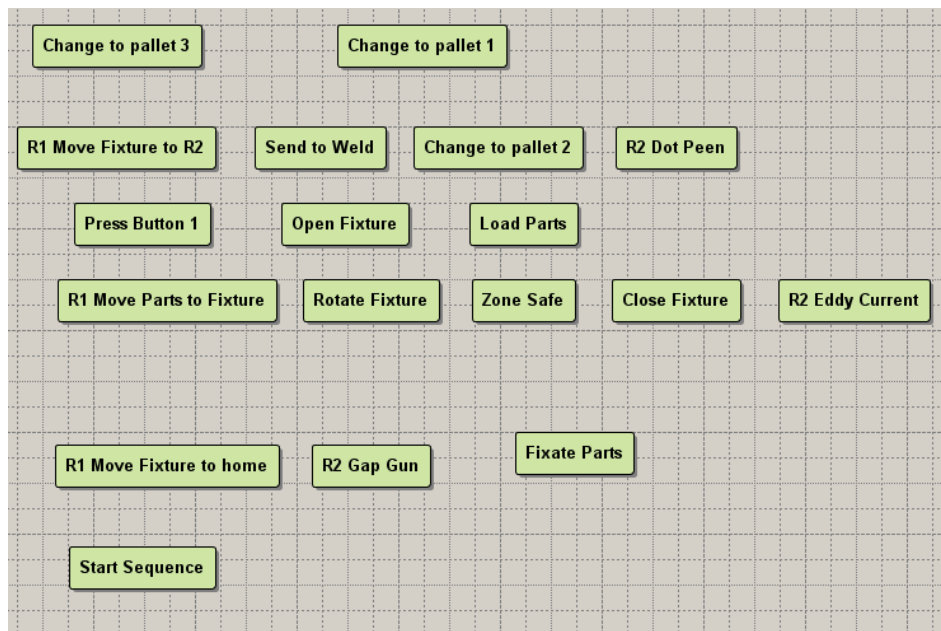


Figure 3.13: The different operations that was created.

The next step in the modelling process is that the different operations are sorted into larger sequence blocks. It was decided that the modelling level of the main sequence where the parts are fixated, would be more detailed than the sequences where the pallets are changed. Due to the hierarchical properties of Sequence Planner it was possible to insert the operations relating to the fixate part operations to the same operation which resulted in figure 3.14. The operations in this case can only be run alternative and it can be noted that **Fixate Parts** in figure 3.14 has a small + sign, this means it can be expanded to show the rest of the operations from figure 3.13.

In figure 3.15 the **Fixate Parts** operation is shown in greater detail. In figure 3.15 it can be seen that the Load Parts operation has a precondition that the **Start Sequence** has been finished. This is indicated with the text over the operation. In Sequence Planner both pre- & postconditions and sequential function charts can be used in the same graph depending on the preferred modelling technique. To handle the common zone between the robots an operation named Zone Safe is used, which is executed in parallel with the robot program that demands the zone. This modelling technique gives an overview of the problem. Sequence Planner also gives the option to verify that the operations are non-blocking and it can optimize the order of the sequences to achieve the shortest cycle times for a certain process.

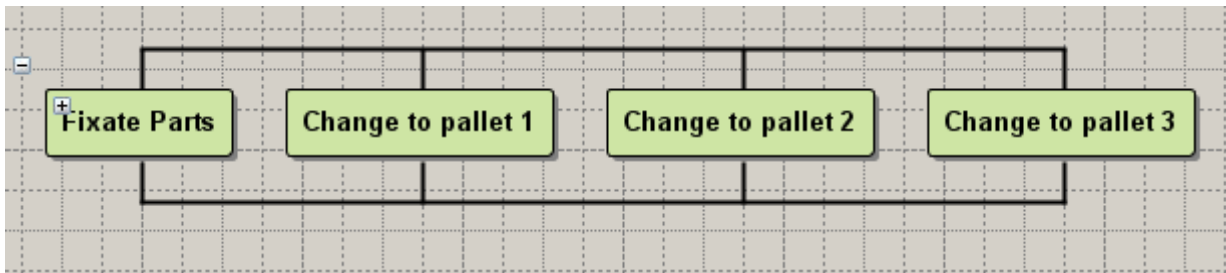


Figure 3.14: The alternating sequence.

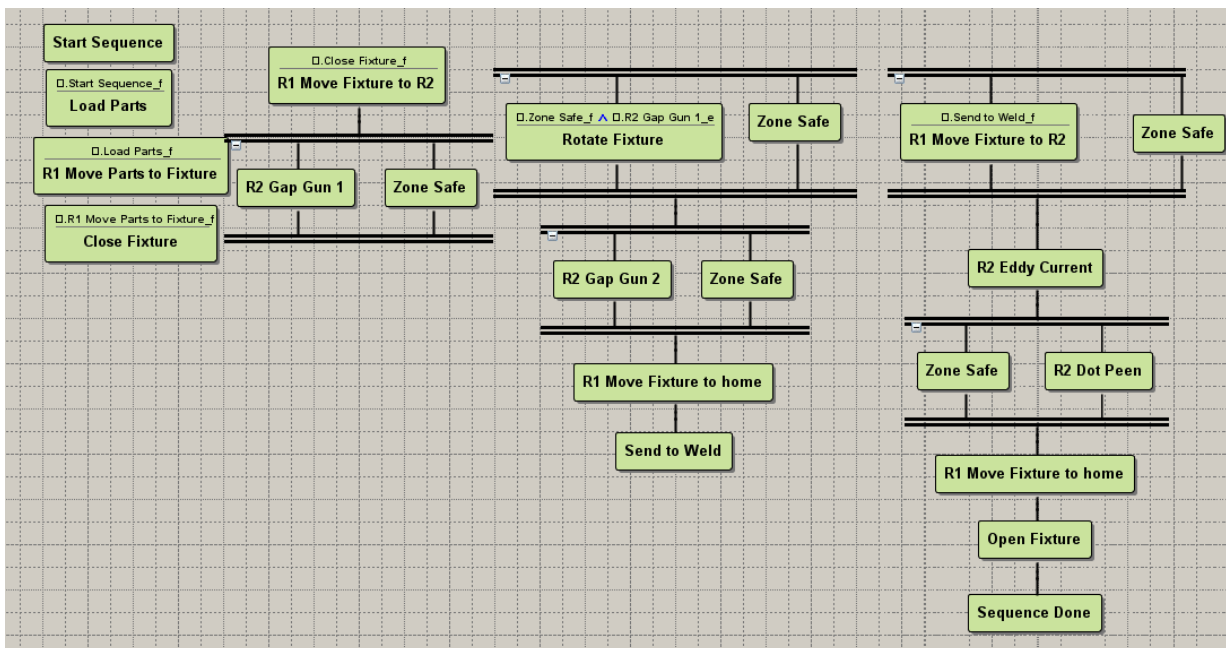


Figure 3.15: The Fixate Parts sequence in detail.

3.7 Modelling of the pallet and fixture switching process in Supremica

In order to model how the pallets and the fixtures are moved in Supremica an general description of the event names and their meaning is shown in figure 3.16. If more than three pallet and fixture sets are going to be used, the level of complexity increases drastically which increases the need to use a software tool, such as Supremica to find a suitable supervisor. In figure 3.16 the desired type is shown in the lower part of the figure and is numbered from 1 to n, in terms of the modelled robot cell this is either a fixture or pallet. The fixture and pallet has distinct places, in the robot cell location 1 is the fixture and 2 is

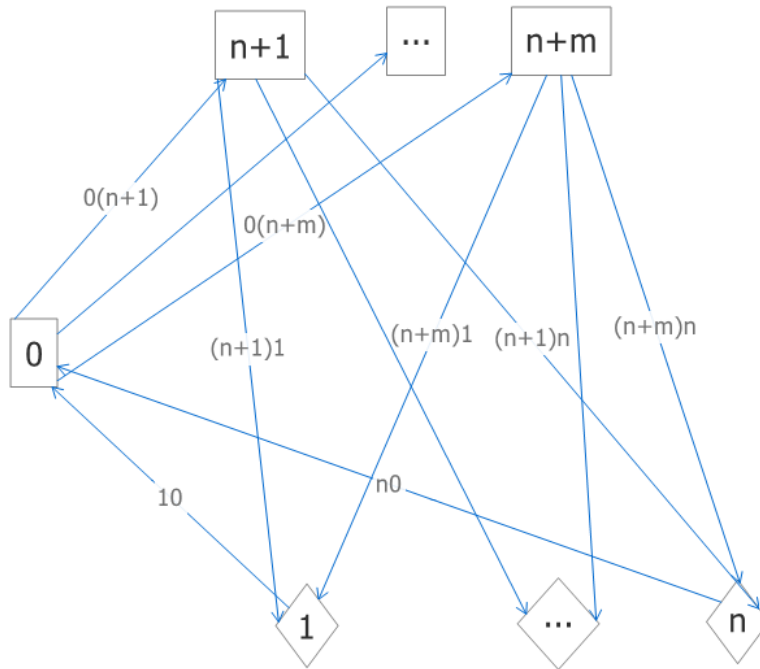


Figure 3.16: The general movement of the palletswitch.

the pallet. It is assumed that the number of desired types is a integer multiplied with the number of places in the upper part of the figure, i.e. $m = a \cdot n$, $a \geq 1$. The upper part of figure 3.16 in other words describe how many sets of types that there is, in the robot cell there was two pallets and two fixtures. They are located pair wise with the fixture having number $n + 1, n + 3, \dots, n + m - 1$ and the pallets having number $n + 2, n + 4, \dots, n + m$ i.e. the fixtures are located at odd numbers and the pallets at even numbers. The number 0 is the temporary location where a type, e.g. a pallet or fixture could be located during the switching process.

In order to describe the general model the events are labeled as following; an event e_{210} corresponds to a movement from location 1 to location 0 in figure 3.16 and 2 corresponds to which type number that is moved. In the robot cell this results in a movement of a pallet to the temporary location. To be able to adjust the number of types and how many sets of types that can be handled in the model the following variables are introduced:

$$l = \{1, 2, \dots, n\} \quad (3.1)$$

$$k = \{n + 1, n + 2, \dots, n + m\} \quad (3.2)$$

$$j = l \cup k \quad (3.3)$$

j' is the odd numbers in j and j'' is the even numbers in j , J is the active pedestal which is used to define the initial state in some automata.

In the plants the state numbers are equivalent of the fixture or pallet number located at that plants pedestal, i.e. there are one plant for each pedestal which can be seen as a memory for which fixture or pallet currently standing at that pedestal. As an example state $S1$ in plant 3.17 means that fixture 1 is located at the temporary pedestal. The plants were defined by everything that could possibly occur. All types can be located at the temporary pedestal, the initial state of the temporary pedestal is $S0$ meaning that it is empty in the beginning. The events that can place a type on the temporary pedestal are defined for all types with the events $ej'10$ and $ej''20$, this means that all different fixtures are moved to the temporary pedestal with the events $ej'10$ and the pallets with the events $ej''20$. The events defining the movement from the temporary pedestal to all storage pedestals are a combination between $ej'10$ & $ej'0k$ and $ej''20$ & $ej''0k$. In a similar way are the other pedestals defined, as seen in figure 3.18, 3.19 and 3.20. It can be noticed that the plants for pedestal in the working area only contains either a fixture or a pallet.

To get the desired marked states in the supervisor a specification controlling that the temporary pedestal is empty and a set of specifications controlling that the different types match each other when placed on the pedestals located in the working area. The specification for controlling that the temporary pedestal is empty contains the same events from the initial state in the plant for the temporary pedestal to make its first transition. The events to the initial state in the plant are the events that transition the specification back to the marked state. This can be seen in figure 3.21. There are type matching specifications of the same amount as the number of different sets, each of these specifies that their unique type matches in the model. These specifications have uniquely defined type events as seen in figure 3.22

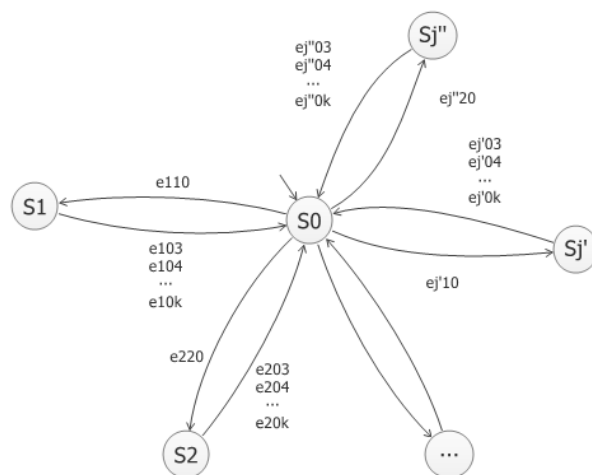


Figure 3.17: Automaton plant of the temporary pedestal.

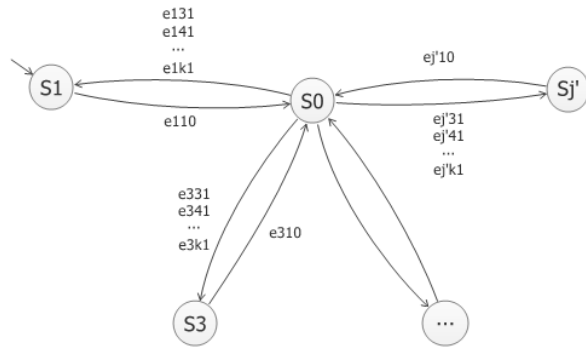


Figure 3.18: Automaton plant of the fixture pedestal in the working area.

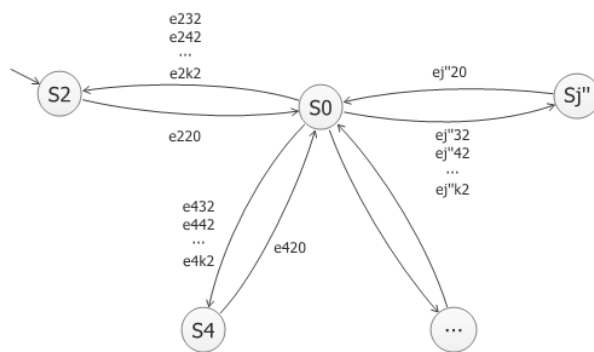


Figure 3.19: Automaton plant of the pallet pedestal in the working area.

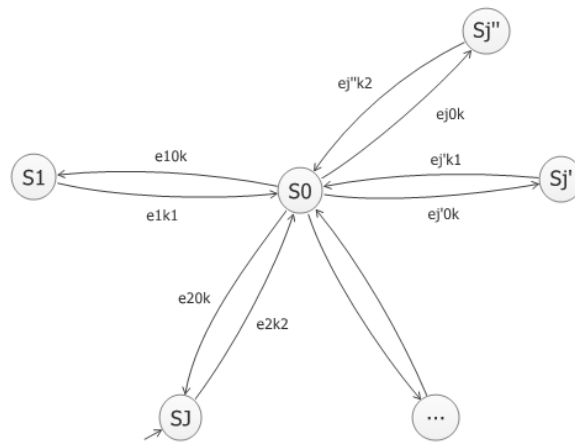


Figure 3.20: Automaton plant of the storing pedestals.

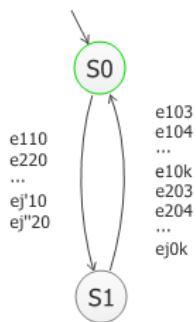


Figure 3.21: Specification of the temporary pedestal.

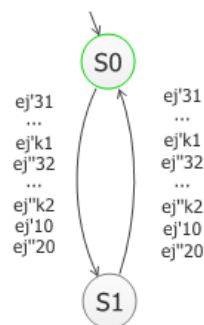


Figure 3.22: Specification of the different types in the working area.

4 Result

The final layout of the robot cell can be seen in figure 4.1. The robot cell has a length of 10000 mm and a width of 5000 mm. The fence surrounds this area. All the distances in the drawing are measured from a reference point that is marked with a red dot in figure 4.1. The red dot is positioned 4000 mm up and 2500 mm left from the lower left fence corner. The orientation and placement of the different resources can be found in appendix A.

The final position of robot 1 was determined by using the RobotSmartPlace tool as described in section 3.3. In figure 4.1 the location of robot 1 is shown, for a more detailed view see appendix A. Note that robot 1's position is elevated 600 mm above the ground.

There exist collisions between the gripper which lifts the parts, and the fixture. This result can be seen in figure 4.2, this might not be reliable because simplified part models were used.

A gripper to lift the fixture was constructed and tested, this gripper replaced the rotational work table that was intended to be used in an earlier version. The tool center point of the fixture gripper was placed in between the gripping entities to simplify rotation of the fixture. The fixture is also able to move the pallet tables, making it a versatile gripper.

Modelling the pallet switch operation in a general way resulted in the supervisor in figure 4.4, it can be noted that this supervisor is very large therefore the manually constructed supervisor was implemented. The used supervisor only works for the case in this thesis and not for an arbitrary number of pallets. The supervisor used in the robot cell can be seen in figure 4.3.

The final simulation model was delivered to Volvo Aero after the thesis presentation in Trollhättan.

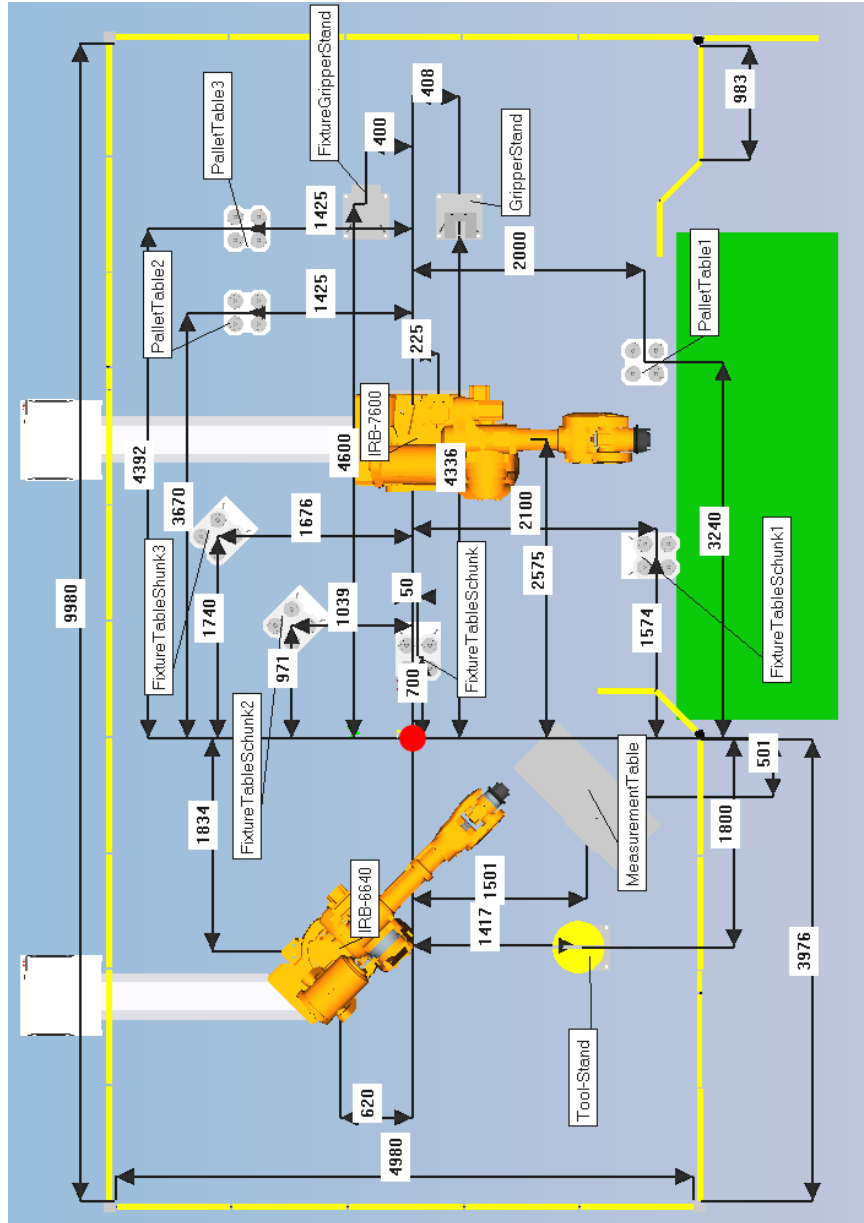


Figure 4.1: The final drawing of the work cell layout.

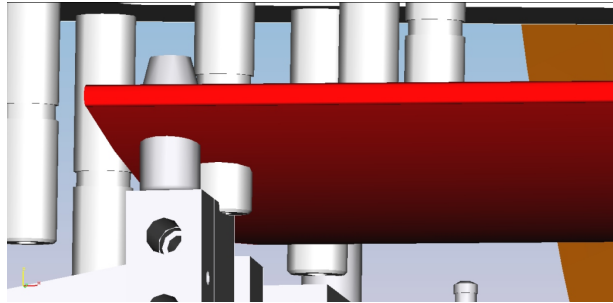


Figure 4.2: Collision between gripper and fixture.

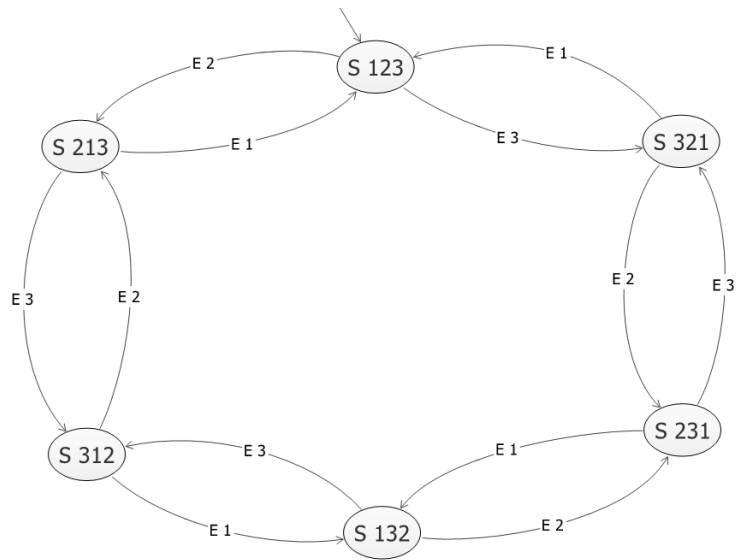


Figure 4.3: The automaton of the supervisor for the pallet switch sequences.

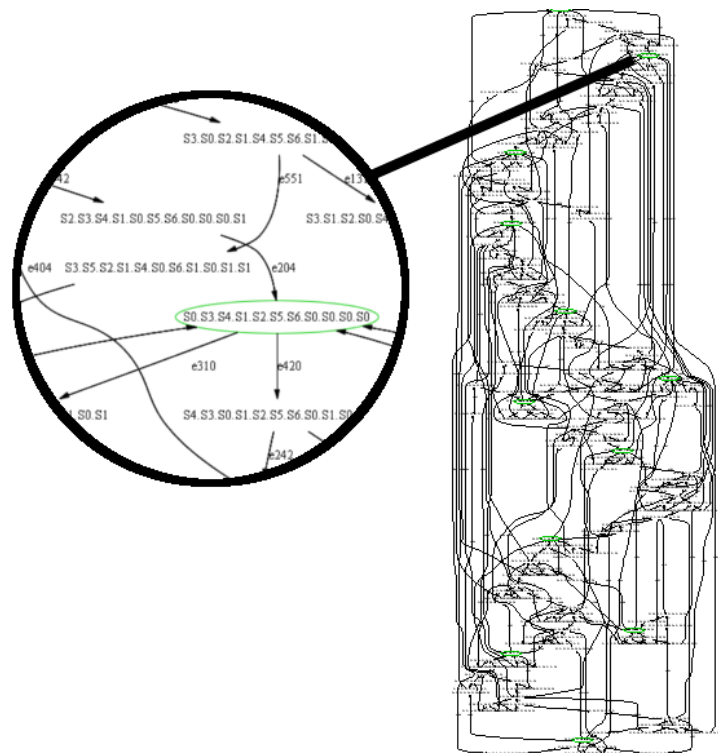


Figure 4.4: The automaton of the supervisor for the pallet switch sequences.

5 Discussion

5.1 Modelling

In this thesis not all of the models were fully developed and ready to be imported into Process Simulate. This gives the model greater flexibility and increase creativity, but if not updated regularly it might not comply with the desired end result.

5.1.1 Fixture gripper

When deciding the position of the TCP, considerations were made to locate it at the center of the gripper. This enables the gripper to rotate the same amount of degrees as the gripped object is being rotated. This simplifies construction of the robots movement when building robot paths, e.g. if a fixture should be rotated 180 degrees, the gripper will then rotate equally.

An earlier version of the fixture gripper needed a rotational table to perform the same action as above. The lifting point of this gripper was from the side of the fixture like a forklift, which made such a rotation impossible. The final fixture gripper instead lifts the fixture at the most robust area, its base. This is to minimize forces on the foldable sides of the fixture, thus increasing the precision of the mounted parts in the fixture.

5.1.2 Automatic Fixturing Model

There exists no realized solution on how the foldable sides of the fixture will be closed. A prototype device was made to test if this could be a solution in the future. Having a worker placing the parts in the fixture has both benefits and drawbacks. Due to the relative heavy sides, some sort of helping device has to be used. Different scenarios that could be implemented are:

- Let the robot close the foldable sides.
- Have an automatic closing and fixturing process.
- A semi-automatic fixturing resource where the human and the device co-operates.

The automatic fixture prototype developed can be seen in figure 5.1, and was developed to eliminate the human error as well as to remove the worker from the robot work area. There does although exist a few limitations with this approach:

- The electric motors used must be very exact to get the precision needed for the fixturing.
- More sensors are needed, which makes the robot cell more complex.
- Parts which has production errors are hard to discover.

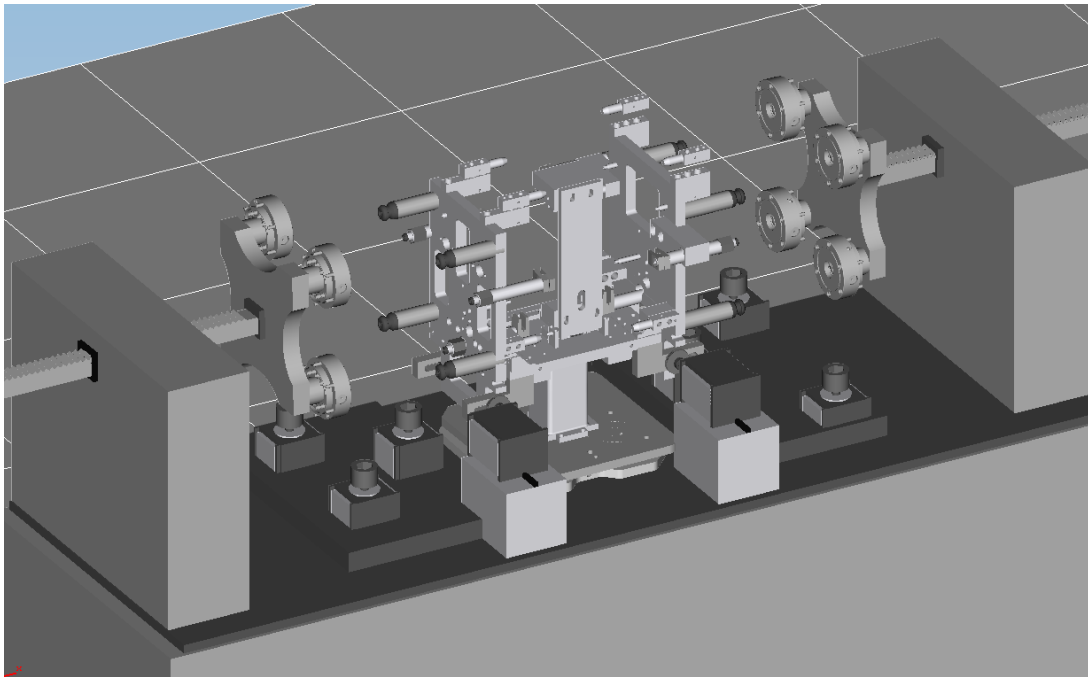


Figure 5.1: Screen shot of the automatic fixturing device.

5.2 Comparison between Simulation Models

5.2.1 Naming Convention

During the thesis it was discovered that it is very important to have a strict naming convention of signals, sequences and other types. This was to prevent confusion, especially when many persons are working on the same project.

5.2.2 Sequence Based Model

Benefits and Drawbacks

- + It is easy to get an overview of the simulation, due to the fact that the sequences only can unfold in a predetermined way.
- + Editing of the different sequences is convenient and moving forward & backwards in simulation time is simple.
- + No care is taken about events, which causes the model to be less complex.
- + Process Simulate Human can be utilized to e.g. ergonomic studies.
- + Building, moving and creating different sequences is less troublesome, this is where the main sequences of the robot cell should be created.

- Cannot simulate more than one scenario, even if a greater number is possible.
- Cannot be directly implemented in the real world.

5.2.3 First Event Based Model

Converting the RobCadStudy into a LineSimulationStudy yields the first event based model.

Benefits and Drawbacks

- + An effortless way to get a more realistic simulation, Process Simulate handles the conversion from the RobCadStudy automatically and the only condition that has to be altered to have a functioning simulation is the start condition.
- + Enables the possibility to create different scenarios that are triggered by certain events.
- + Can handle more than one instance of a part, this instance is then called an appearance, this enables cyclic simulations.
- Still uses Process Simulate internal logic to handle the conditions that triggers the different sequences.
- Emergency stop is not functional when using robots.
- Harder to edit robotic movement than if a RobCadStudy is used.

5.2.4 Final Event Based Model

The final simulation model was created after that robot logic was controlled by Process Simulate's logic blocks and modules.

Benefits and Drawbacks

- + More realistic than the previous models.
- + Robot sequences are triggered by logic blocks and not Process Simulate internal logic.
- + The model that is most likely to be offline programmed.
- The robot cell should be as complete as possible when the conversion is started. Otherwise the changes that have to be performed will be more time consuming compared to if they were created in the sequence based model.

- A large robot cell might demand severe time before it can be verified to work properly. This is due to that the number of states in the robot cell can increase rapidly when the number of possible events increases.
- Process Simulate Human cannot yet be used in an event based simulation.

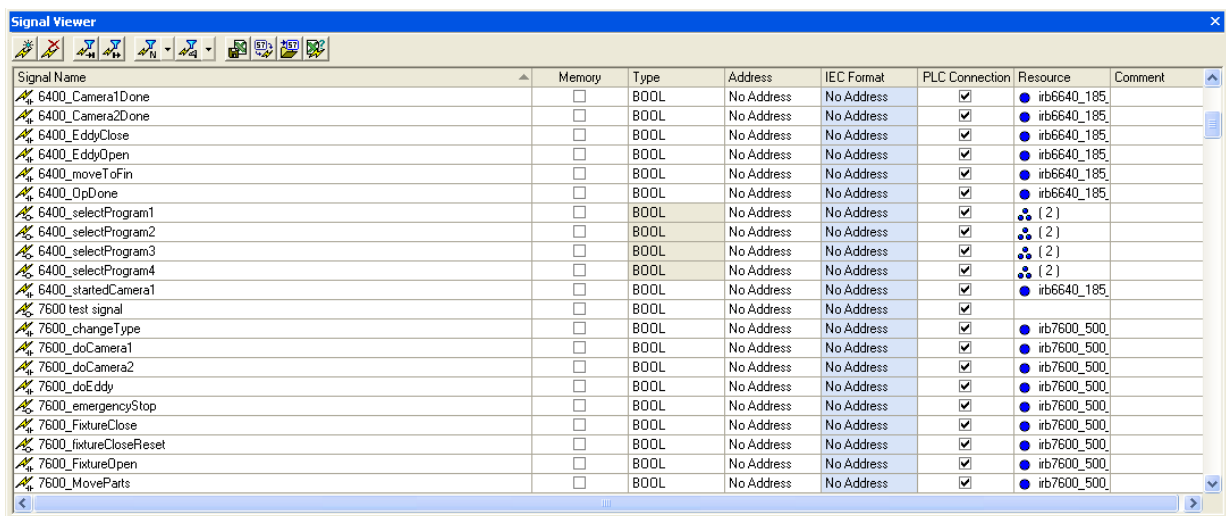
5.2.5 Model Differences

There are differences in the complexity of the modelling process depending on whether a sequence based or an event based model is used. The sequence based model is mainly used to analyze robotic movement, human processes and collision detection between objects. An event based model is constructed in two steps, where the first one uses sequences in combination with transition conditions. The next step is to convert the transition conditions to logic blocks and modules which in turn can be used for offline programming.

5.3 Logic

5.3.1 Robot signals

The main co-operation between the robots is their triggering signals. Some of these signals can be viewed in figure 5.2. These robot signals are set when the robot has finished or are about to finish its selected program.



Signal Name	Memory	Type	Address	IEC Format	PLC Connection	Resource	Comment
6400_Camera1Done	<input type="checkbox"/>	BOOL	No Address	No Address	<input checked="" type="checkbox"/>	ib6640_185	
6400_Camera2Done	<input type="checkbox"/>	BOOL	No Address	No Address	<input checked="" type="checkbox"/>	ib6640_185	
6400_EddyClose	<input type="checkbox"/>	BOOL	No Address	No Address	<input checked="" type="checkbox"/>	ib6640_185	
6400_EddyOpen	<input type="checkbox"/>	BOOL	No Address	No Address	<input checked="" type="checkbox"/>	ib6640_185	
6400_moveToFin	<input type="checkbox"/>	BOOL	No Address	No Address	<input checked="" type="checkbox"/>	ib6640_185	
6400_OpDone	<input type="checkbox"/>	BOOL	No Address	No Address	<input checked="" type="checkbox"/>	ib6640_185	
6400_selectProgram1	<input type="checkbox"/>	BOOL	No Address	No Address	<input checked="" type="checkbox"/>	(2)	
6400_selectProgram2	<input type="checkbox"/>	BOOL	No Address	No Address	<input checked="" type="checkbox"/>	(2)	
6400_selectProgram3	<input type="checkbox"/>	BOOL	No Address	No Address	<input checked="" type="checkbox"/>	(2)	
6400_selectProgram4	<input type="checkbox"/>	BOOL	No Address	No Address	<input checked="" type="checkbox"/>	(2)	
6400_startedCamera1	<input type="checkbox"/>	BOOL	No Address	No Address	<input checked="" type="checkbox"/>	ib6640_185	
7600_test signal	<input type="checkbox"/>	BOOL	No Address	No Address	<input checked="" type="checkbox"/>		
7600_changeType	<input type="checkbox"/>	BOOL	No Address	No Address	<input checked="" type="checkbox"/>	ib7600_500	
7600_doCamera1	<input type="checkbox"/>	BOOL	No Address	No Address	<input checked="" type="checkbox"/>	ib7600_500	
7600_doCamera2	<input type="checkbox"/>	BOOL	No Address	No Address	<input checked="" type="checkbox"/>	ib7600_500	
7600_doEddy	<input type="checkbox"/>	BOOL	No Address	No Address	<input checked="" type="checkbox"/>	ib7600_500	
7600_emergencyStop	<input type="checkbox"/>	BOOL	No Address	No Address	<input checked="" type="checkbox"/>	ib7600_500	
7600_FixtureClose	<input type="checkbox"/>	BOOL	No Address	No Address	<input checked="" type="checkbox"/>	ib7600_500	
7600_fixtureCloseReset	<input type="checkbox"/>	BOOL	No Address	No Address	<input checked="" type="checkbox"/>	ib7600_500	
7600_FixtureOpen	<input type="checkbox"/>	BOOL	No Address	No Address	<input checked="" type="checkbox"/>	ib7600_500	
7600_MoveParts	<input type="checkbox"/>	BOOL	No Address	No Address	<input checked="" type="checkbox"/>	ib7600_500	

Figure 5.2: Signal viewer window for a set of robot signals.

This method was used because it gave the most controlled scenario and made it possible to have parallel robot programs running at the same time, although a combination of triggering signals and unique sensor scenarios could be used as a safety precaution. The

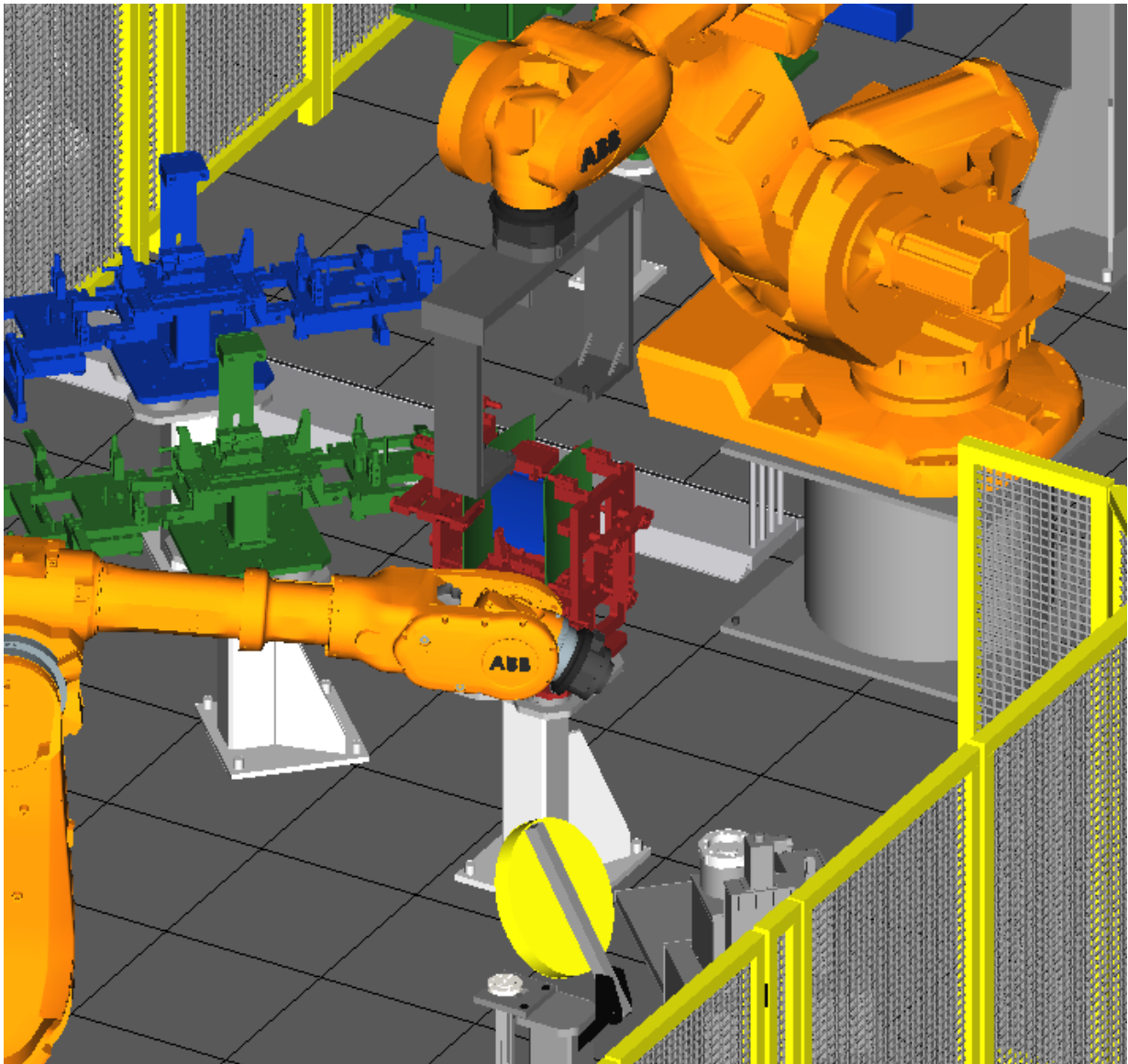


Figure 5.3: A picture of three operations running in parallel.

safety precaution however did not provide anything of interest in this thesis. Robot 1 and robot 2 can start each other's robot programs, e.g. when the **7600_doCamera1** signal is set to true, then robot 2 starts its **6640_Camera1** operation. This method also makes it possible to easily have many operations running in parallel, e.g. in figure 5.3 robot 1 has released fixture 1 on the temporary pedestal and is moving up and away from the common robot zone. The lid on the eddy current stand is moving to an open position to allow robot 2 to mount the eddy current tool and robot 2 is moving from its home position to the eddy current tool's mount position.

5.4 Problems

Under this section a few critical problems in Process Simulate will be discussed and how they were solved.

5.4.1 Dummy attachment

Moving parts with pick and place operations in Process Simulate is an easy task, it is quick and effortless. When it comes to moving fixtures the problem is that it is impossible to move when it is defined as a fixture. In order to be able to reuse the robot movements in the pallet switch sequences a solution had to be found, if not the number of sequences would swell considerably. The solution that was found was to use a dummy part and attach that to the fixture in the beginning of the sequence. This part then acted as the gripped object and the fixture followed due to the attachment to the part.

5.4.2 The rotational bug

Process Simulate contains like most other software some bugs. To name one of the more critical and unusual bugs there is the rotational bug. In the beginning of the thesis one of the computers used for simulation was running Windows XP in 64 bits mode. This was according to the manual a supported configuration. A very peculiar bug occurred at random which generated a lot of extra work. When loading a study, Process Simulate would at random rotate the resources, each independently rotated in very strange amounts. This made the robot cell non-working, and the whole robot cell had to be repositioned. This bug also affected the robot paths and the via-locations had to be rebuilt. No apparent solution to this problem could be found, even after consulting Siemens. The solution used was to reinstall the computer with a 32-bit Windows XP. This crucial bug can be seen in figure 5.4.

5.4.3 Fixture

The fixture used in the simulation is a prototype, below is a list of a few of the potential problems with the fixture:

- When folding the sides of the fixture the parts will collide before intended too, which makes the exact fixturing impossible. A solution to the problem could be to give the rotating axles sliding capabilities.
- No solution to move the fixture between the different stations is yet established in the real work cell.
- There does not exist any mechanism to help the human operator fold the sides of the fixture.
- The parts collides with the fixture itself when it is folding.

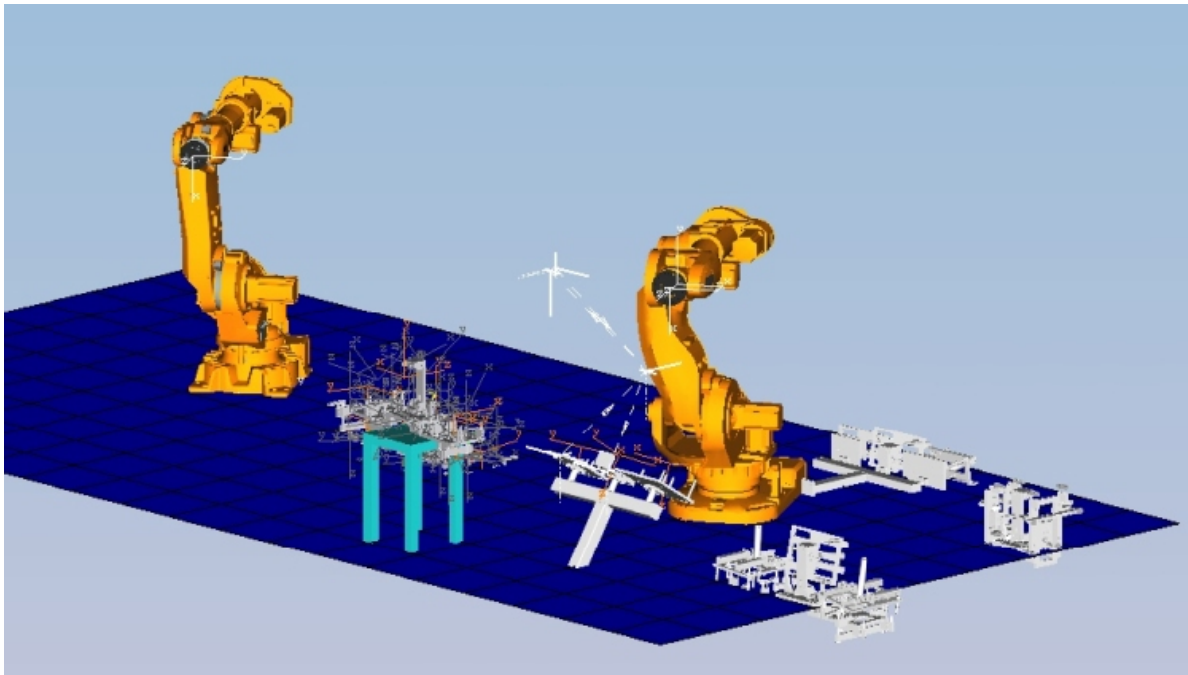


Figure 5.4: The rotational bug of the robot cell, notice that everything is rotated peculiar.

5.5 Additional Work

5.5.1 Automatic Path Planner

Automatic Path Planner is a plug-in to Process Simulate that enables the software to create collision free paths. This is something that was evaluated to see if it was possible to automatically create collision-free robotic paths between different points in space. The reason why this would be desirable is that the process of creating the paths is very time consuming. If it was possible to shorten the development time it would result in a greater variety of model scenarios to be tested under the same time span. Unfortunately the rendering time was too long, and the quality of the automatically generated paths too low to be used within the robot cell.

5.5.2 Pallet switch in Supremica

Unfortunately Supremica did not come in use for the implemented pallet switch operation. There were several tests, but the final supervisor always turned out to be larger, with more events and a more complicated event structure. The implemented supervisor in this thesis can only be used for three sets of pallets and fixtures and they are restricted to follow a specific execution order. Process Simulate does not support a supervisor implementation from Supremica therefore is it very difficult to use this type of solution.

If a supervisor was to be implemented in Process Simulate it is advised to write a plug-in which handles the decision which specific robot program numbers that is selected. As an example the pallet and fixture switching operation could use a plug-in that could be visualized as a black box. The black box contains the supervisor and a graph searching algorithm and a converter for the robot programs. An example of a search algorithm that could be used is Dijkstra's algorithm [20]. Process Simulate then sends a signal with the desired type to the black box. The box then calculates the path to the desired state and converts this path to a set of robot programs that are readable by Process Simulate. Process Simulate then executes the programs in the desired order. Time optimization is a possibility if the supervisor is implemented this way due to the usage of an optimization algorithm. To be able to implement this would be very time consuming and it is out of scope for this thesis.

5.6 Methodology

5.6.1 Workflow methodology

According to [3] a robot simulation can be subdivided into the following sections, see figure 5.5. The paper [3] contains a methodology about a particular software. In the section below a fairly accurate translation of Process Simulate has been done.

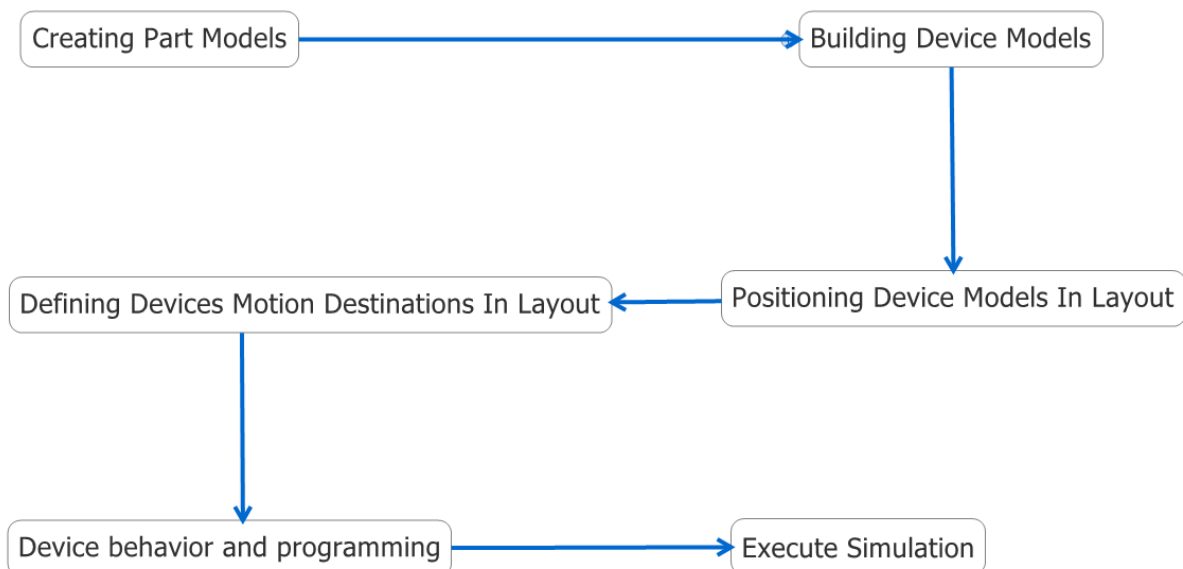


Figure 5.5: visualisation of simulation methodology according to [3].

To describe the procedure above in terms of process simulate the following remarks about figure 5.5 is made:

Creating parts models

Importing, or constructing parts in the built-in modelling editor in Process Simulate.

Building device models

Importing, or building robots, grippers, fixtures. This also incorporates defining kinematics for the devices, and setting the joints motion restrictions.

Positioning Device Models in Layout

Deciding where the different resources should be located.

Defining devices motion destination in layout

Inverse kinematic of the robotic movement is solved to generate the robot paths.

Device behavior and programming

Logic of the model is defined such that the sequences are executed correctly.

Execute simulation

Run the simulation in Process Simulate.

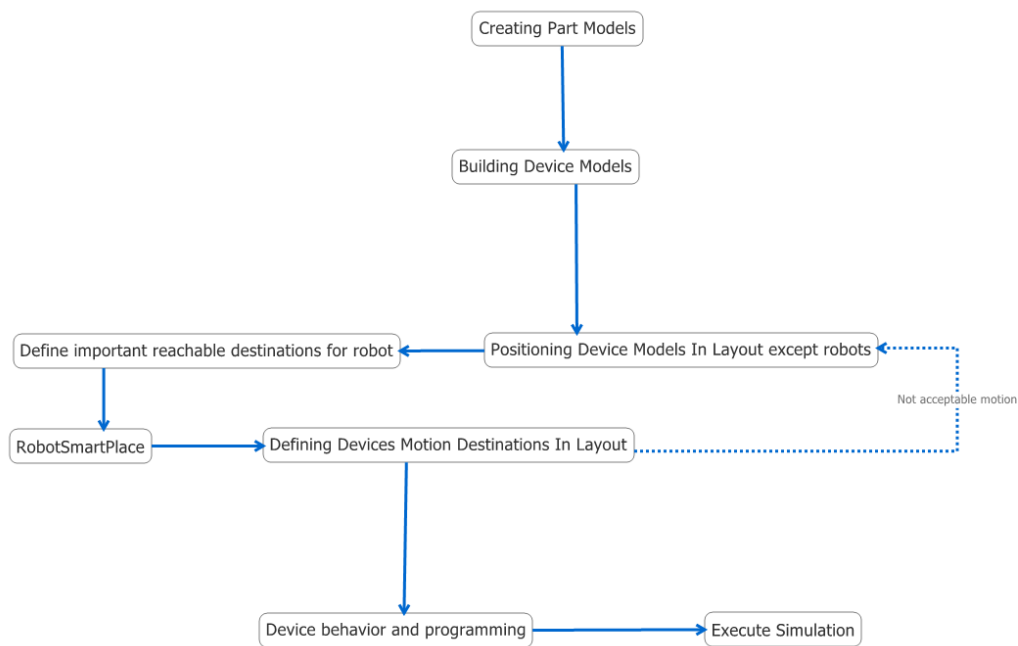


Figure 5.6: The suggested workflow.

According to 5.5 the procedure to create a new simulation is quite straight forward without any drastic re-works. An alternative formulation that is more suited to the workflow that emerged under this project is suggested in 5.6. The main difference being that this workflow

allows iteration of the positioning of resources. This was very important due to that it is hard to find an optimal placement for the resources. Instead there exist several possible solutions which each have its advantages and drawbacks. The suggested model incorporates the built-in tool RobotSmartPlace to iterate a large number of possible locations that the robot can be placed in. The non-robotic resources locations are i.e. decided before the locations of the robots are finalized. It is after this stage the more detailed robot paths are created. If the paths do not comply with the design specifications then repositioning of the resources in the robot cell is needed. The actual workflow that was used in this thesis

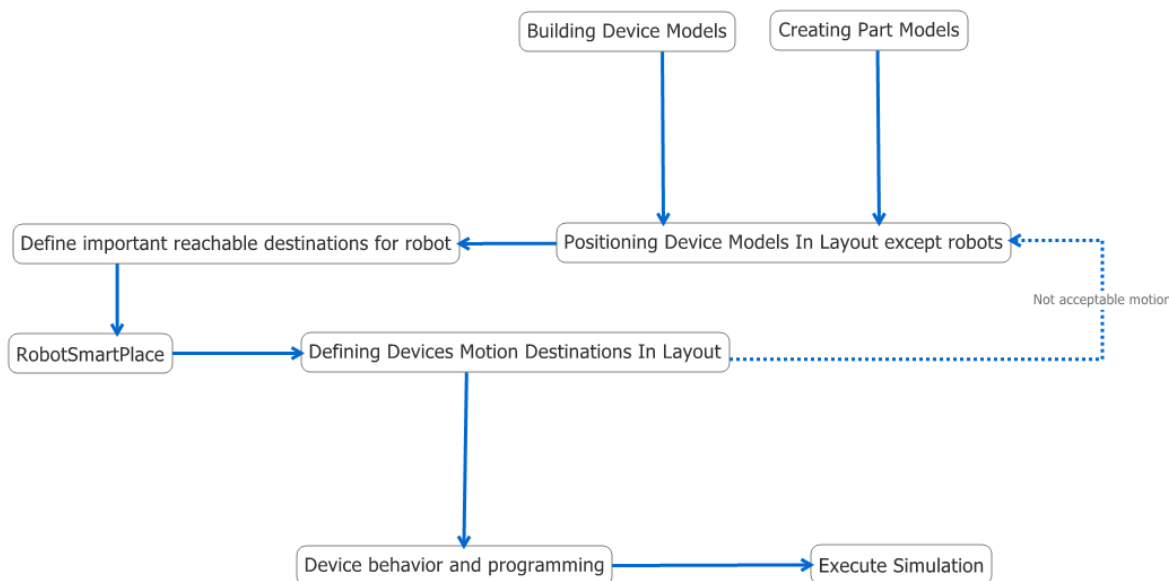


Figure 5.7: The workflow that was used in the project.

can be seen in figure 5.7, the only difference is that the parts and the resources was built in parallel and not in serial like in figure 5.6.

5.6.2 Sequence Planner

When building sequences in Process Simulate, it can either be done straight in the software but the sequences can also be managed in Process Designer. Process Designer has several different viewers to show the relations between the different sequences, resources and parts [19]. A viewer called PERT (Program Evaluation Review Technique) [5] is shown in figure 5.8. If Sequence Planner would be implemented into Process Simulate as a plug-in it has to add something that does not already exist in another software in the Tecnomatix family, or else it would not vindicate the work effort. Process Simulate has a plug-in called Line

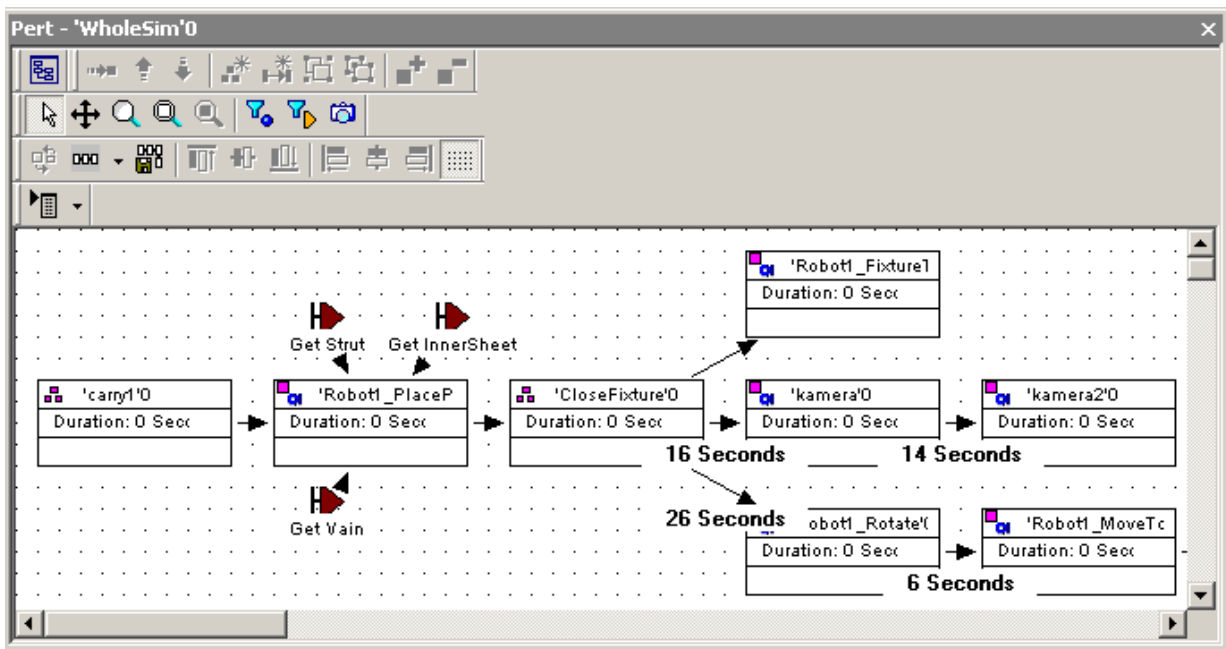


Figure 5.8: The PERT viewer.

Balancing which is an optimization utility for e.g. adjusting lines in a factory and finding bottlenecks. The optimization and verification that Sequence Planner has built in is based on automata theory and utilizes Supremica to execute optimization and verification. This is a new approach to Process Simulate and is something that is not yet implemented. The opposite procedure has been analyzed in [11], i.e. to import data from Process Simulate into Supremica to perform formal verification. In Sequence Planner only operations exists, but in Process Simulate there are several different data types, e.g. sequences, key signals and robot signals. To be able to have a working plug-in it would be necessary to define this in Sequence Planner or it would interfere with the data structure in Process Simulate. This could possibly disturb the modelling methodology of Sequence Planner. The difference between Sequence Planner and Process Simulate regarding operations can be seen in figure 5.9. In order to implement the operations from Sequence Planner into Process Simulate the operations has to be specified correctly to be compatible with the data types in Process Simulate, figure 5.9 clearly shows that Sequence Planner only has one data type but Process Simulate has many. The plug-in might be useful if a very complex robot cell is constructed but it is hard to see the real benefits with a simple robot cell.



Figure 5.9: Difference between data types between Sequence Planner and Process Simulate.

6 Conclusions

The main reason to create a simulation before building the real robot cell is that erroneous design parameters easily can be corrected, without alternating the robot cell. This also prevents expensive mistakes such as collisions between robots and resources, a computer only needs a reset of the simulation. Flaws that are difficult to discover in the design phase of a resource or operation can with the help of a simulation be determined easier.

When developing a robotic cell it is efficient to utilize a workflow during the process, the main advantages are that following a specific method saves time. A methodology to develop a robot simulation in Process Simulate has been achieved in this master thesis. It was discovered that it is facilitating to have the final CAD-models of the resources when building the robot cell. If the models are under development or non-existing it is important that the decision makers has an ongoing communication with the simulation developers, to keep them up to date.

The future use of the model could be to test new scenarios with different resources and operations, e.g. examine the milling process of the Turbine Exhaust Case. The test cell at Volvo Aero will be used in many different ways, why it would be wise to explore the possibilities with offline programming. If today's method is used then the same work will be done twice, once during the simulation and then again when programming the robot.

The importance of having a structured workflow cannot be stressed enough, but it is also important that the simulation is relevant in the products development. Some of the engineers at Volvo Aero uses a workflow known as KBA [15], knowledge based automation. This methodology could be of great benefit in the development of the Turbine Exhaust Case assembly line. It would be beneficial if all groups working with the problem had a closer co-operation and a mindset that the product that is designed will eventually be used in an automation line.

Future thesis work could be implementing other processes that will be used in the test cell or when Volvo Aero is developing the final assembly cell. Future work could also be evaluating Process Simulate's offline programming capabilities on the test cell. This also implies that the test cell must be identical to the simulation cell or else unforeseen errors may occur. If this problem arises the offline programming was irrelevant and the robots could have been programmed the conventional way.

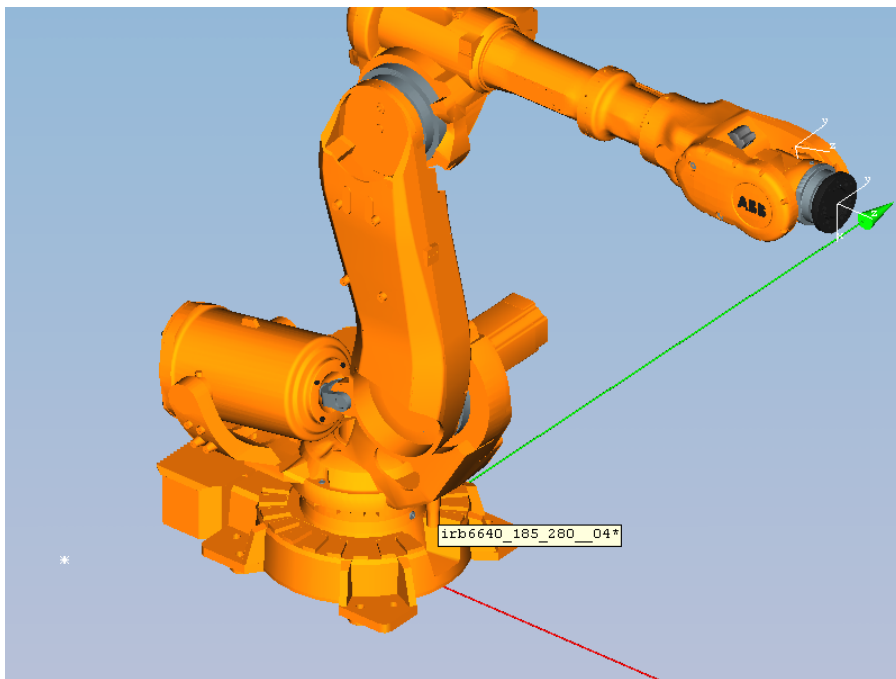
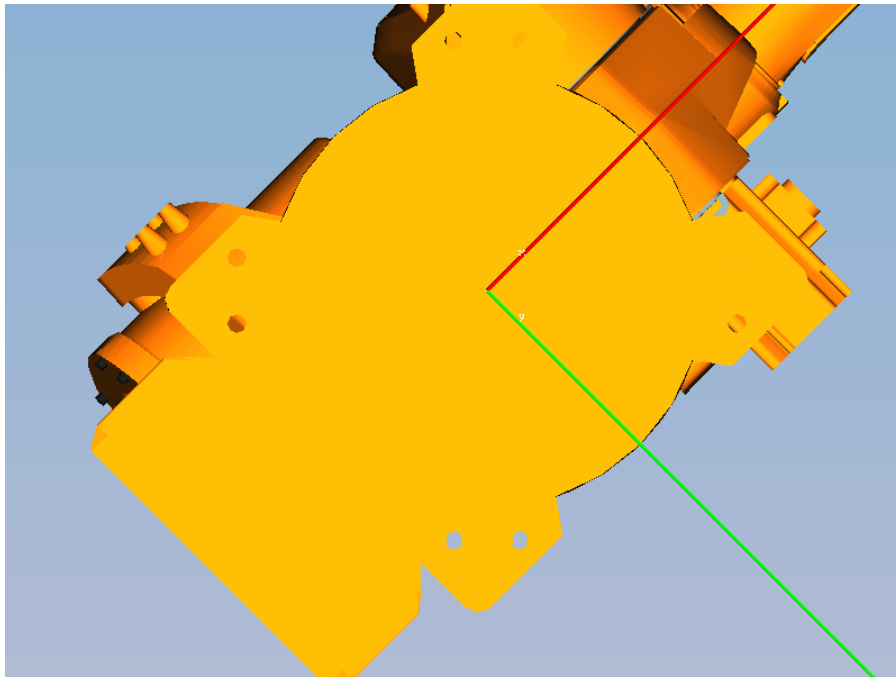
References

- [1] K. Akesson, M. Fabian, H. Flordal, and R. Malik. Supremica - an integrated environment for verification, synthesis and simulation of discrete event systems. In *Discrete Event Systems, 2006 8th International Workshop on*, pages 384–385, July 2006.
- [2] B.G Batchelor and P.F Whelan. Intelligent vision systems for industry. <http://elm.eeng.dcu.ie/~whelanp/ivsi/IVSI.pdf>, Accessed 2009-08-21.
- [3] F.S. Cheng. Design and optimization of cellular manufacturing systems: a methodology for developing robotic workcell simulation models. In *WSC '00: Proceedings of the 32nd conference on Winter simulation*, pages 1265–1271, San Diego, CA, USA, 2000. Society for Computer Simulation International.
- [4] J.J. Craig. Simulation-based robot cell design in adeptrapid. In *Robotics and Automation, 1997. Proceedings., 1997 IEEE International Conference on*, volume 4, pages 3214–3219 vol.4, Apr 1997.
- [5] G. Di Battista, E. Pietrosanti, R. Tamassia, and I.G. Tollis. Automatic layout of pert diagrams with x-pert. In *Visual Languages, 1989., IEEE Workshop on*, pages 171–176, Oct 1989.
- [6] D. Ding, Y. Liu, and M.Y. Wang. Automatic selection of fixturing surfaces and fixturing points for polyhedral workpieces. In *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, volume 2, pages 1147–1152 vol.2, 2001.
- [7] P. Falkman. Specification of Resource Allocation Systems; a STEP towards a unified framework, 2005.
- [8] P. Glimvert, E. Hartwig, A. Larsson, E. Lindskog, L. Lundh, and A. Ohldin. Virtuellt utveckling och idrifttagning av tillverkningscell med 3d-simulering. page 8, 5 2009.
- [9] K. Hench, W. Christensen, D. Gallant, R. Hinde, A. Lopez, C. Martin, and T. Phillips. Modeling eddy current analysis data to determine depth of weld penetration. In *Automation Congress, 2002 Proceedings of the 5th Biannual World*, volume 13, pages 77–82, 2002.
- [10] K.W. Hench, W. Christensen, A.A. Lopez, C. Martin, and T.T. Phillips. An optimization technique for the evaluation of eddy current inspection data to determine weld quality. In *Systems, Man and Cybernetics, 2005 IEEE International Conference on*, volume 2, pages 1535–1539, Oct. 2005.
- [11] C. Modig and F. Westman. Linking event-based simulations in process simulate with Supremica to perform formal verification sequences of operations at VCC, 2008.

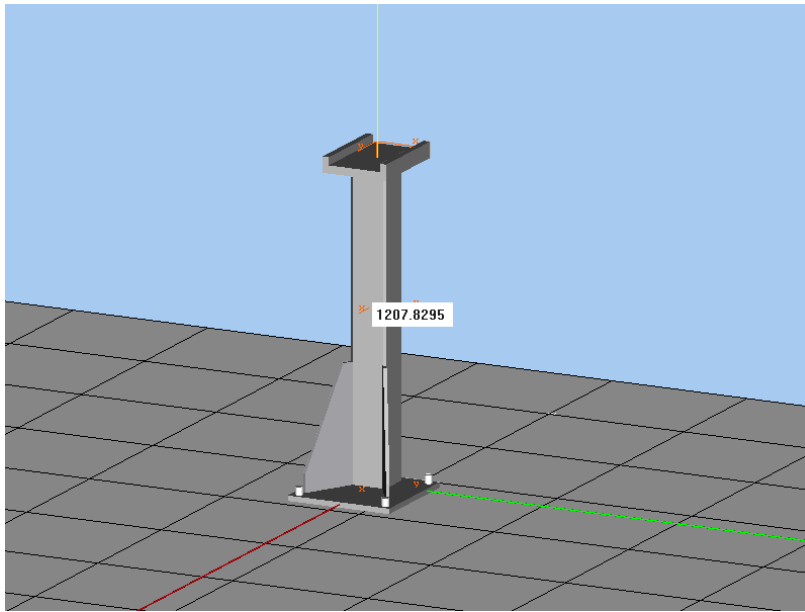
- [12] NASA. Applying data matrix identification symbols on aerospace parts. <http://standards.nasa.gov/released/6000/NASA-STD-6002A.pdf>, Accessed 2009-08-21.
- [13] Chalmers University of Technology. Supremica. <http://www.supremica.org/>, Accessed 2009-08-21.
- [14] E. Ohlson and C. Torstensson. Development, implementation and testing of sequence planner, a concept for modeling of automation systems, 2009.
- [15] Siemens PLM. Volvo aero harnesses knowledge to gain a commanding advantage. <http://www.plm.automation.siemens.com/CaseStudyWeb/dispatch/viewResource.html?resourceId=2172>, Accessed 2009-08-21.
- [16] SCB. Vikt och längd i befolkningen - www.scb.se:. http://www.scb.se/Pages/TableAndChart___47966.aspx, Accessed 2009-08-21.
- [17] Schunk. Schunk. <http://www.de.schunk.com/schunk/index.html?country=SWE>, Accessed 2009-08-21.
- [18] Siemens PLM. *eMServer System Administration Student Guide*, 8.2 edition, January 2008.
- [19] Siemens PLM. *Process Designer 9 Reference Manual*, 9 edition, 2008.
- [20] M. Sniedovich. Dijkstra's algorithm revisited: the dynamic programming connexion, 2006.
- [21] E.Y.T. Tan, A.S. Kumar, J.Y.H. Fuh, and A.Y.C. Nee. Modeling, analysis, and verification of optimal fixturing design. *Automation Science and Engineering, IEEE Transactions on*, 1(2):121–132, Oct. 2004.
- [22] B. Trebilcock. Direct part marking: The next hot trend in automatic identification. <http://www.mmh.com/article/CA6437021.html>, Accessed 2009-08-21.
- [23] Clemson University. Recommended workstation measurements. <http://www.clemson.edu/ces/departments/ie/documents/kimbler/WrkStaDim.pdf>, Accessed 2009-08-21.

Appendix A

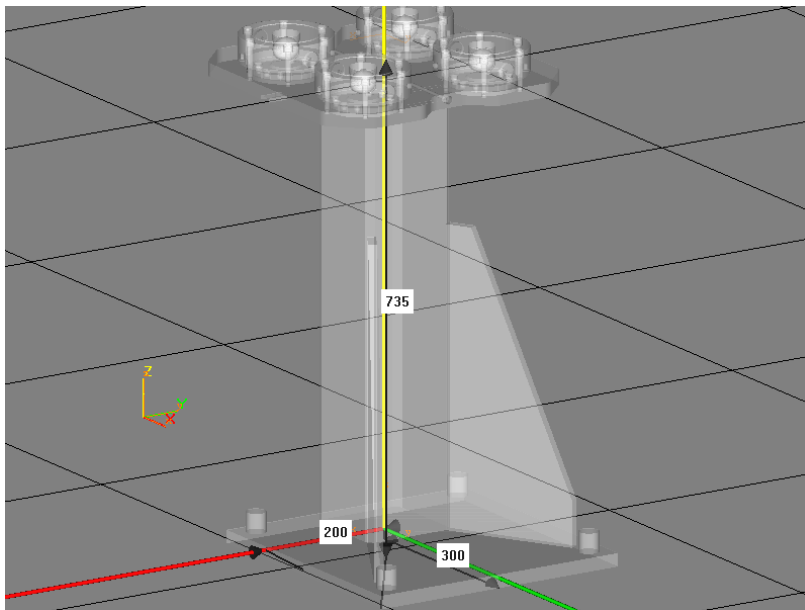
All measurements are in mm. Rotation of the parts are given relative to a right hand coordinate system, and the orientation in the pictures. The x-axis is denoted with red, the y-axis with green and the z-axis with yellow color.



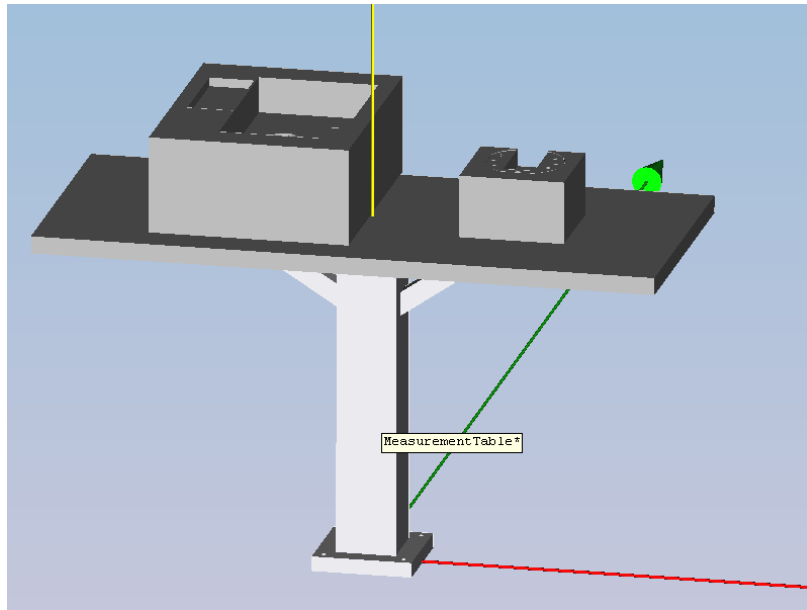
Part	X(mm)	Y(mm)	Z(mm)	Rot Z(deg)
IRB-6400	-1834	620	0	-45



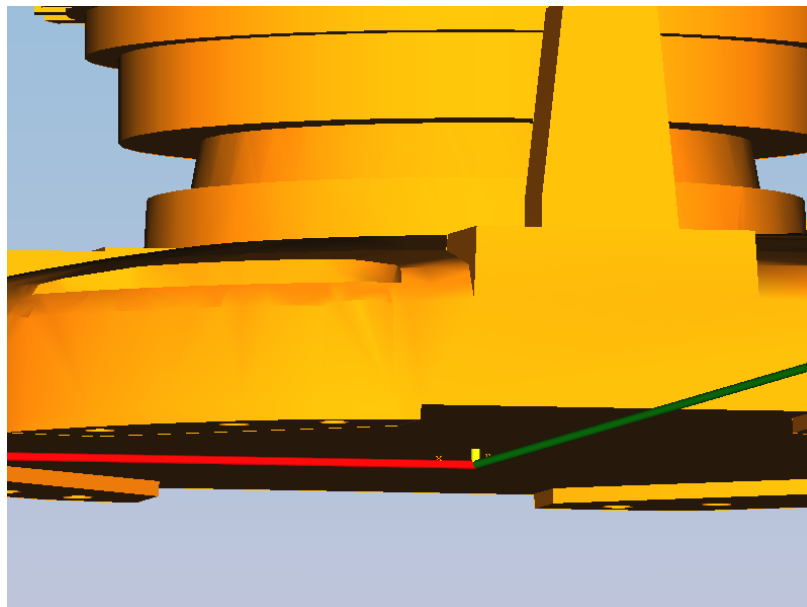
Part	X(mm)	Y(mm)	Z(mm)	Rot Z(deg)
TableFixtureGripper	4600	400	0	180

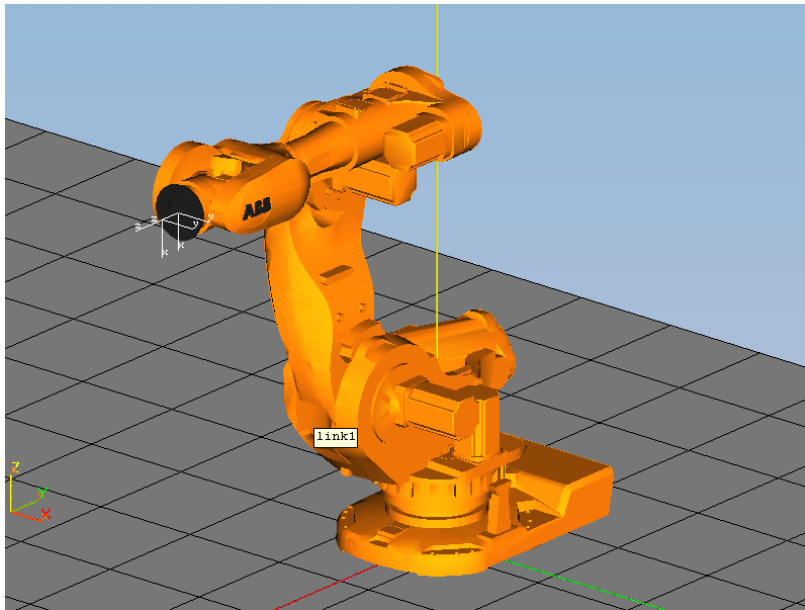


Part	X(mm)	Y(mm)	Z(mm)	Rot Z(deg)
FixtureTableSchunk	700	-50	0	-90

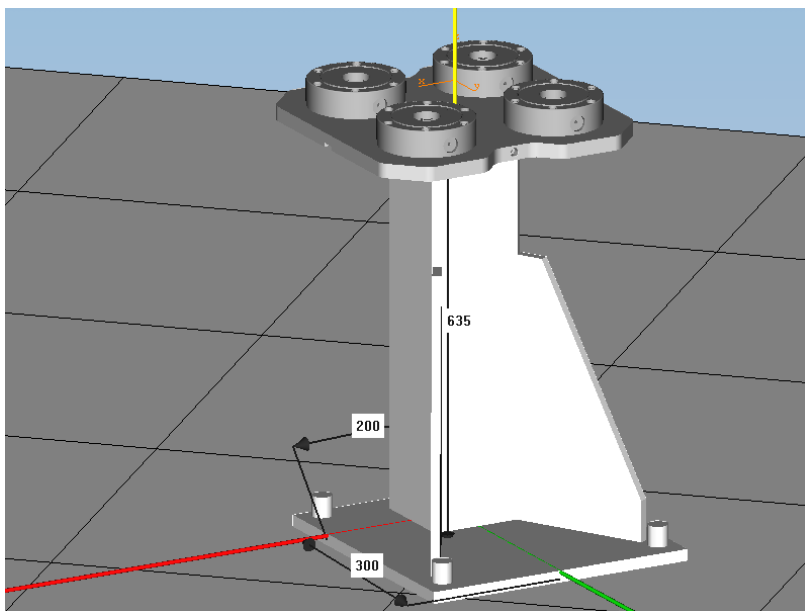


Part	X(mm)	Y(mm)	Z(mm)	Rot Z(deg)
MeasurementTable	-501	-1501	0	-45

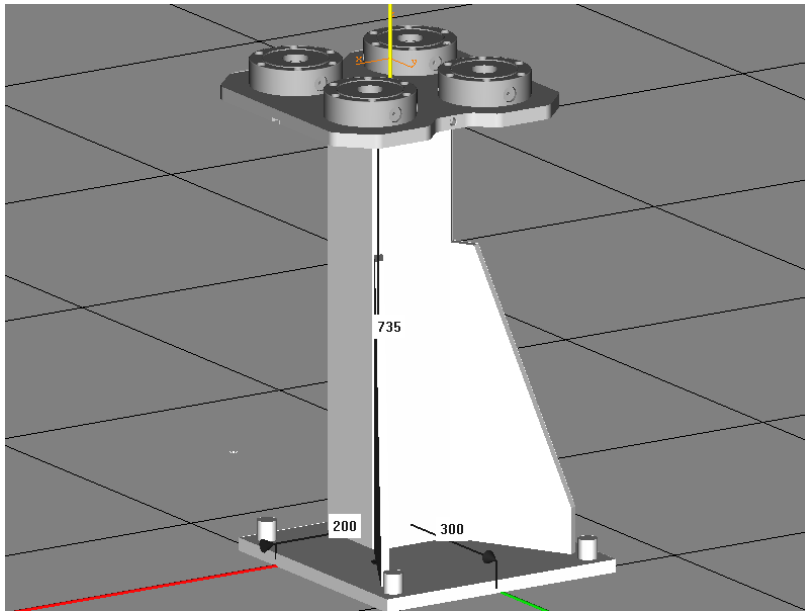




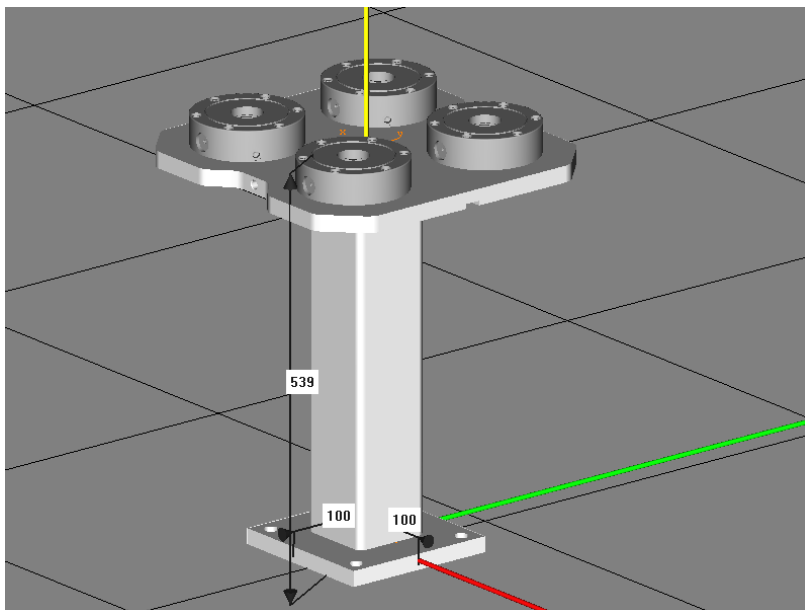
Part	X(mm)	Y(mm)	Z(mm)	Rot Z(deg)
IRB-7600	2575	-225	600	-90



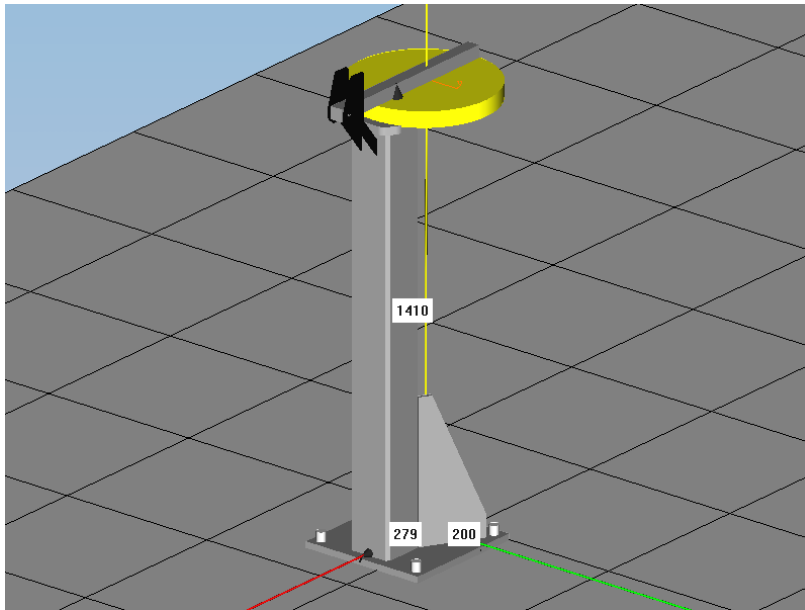
Part	X(mm)	Y(mm)	Z(mm)	Rot Z(deg)
FixtureTableShunk3	1740	1676	0	-135
FixtureTableSchunk2	971	1039	0	-135



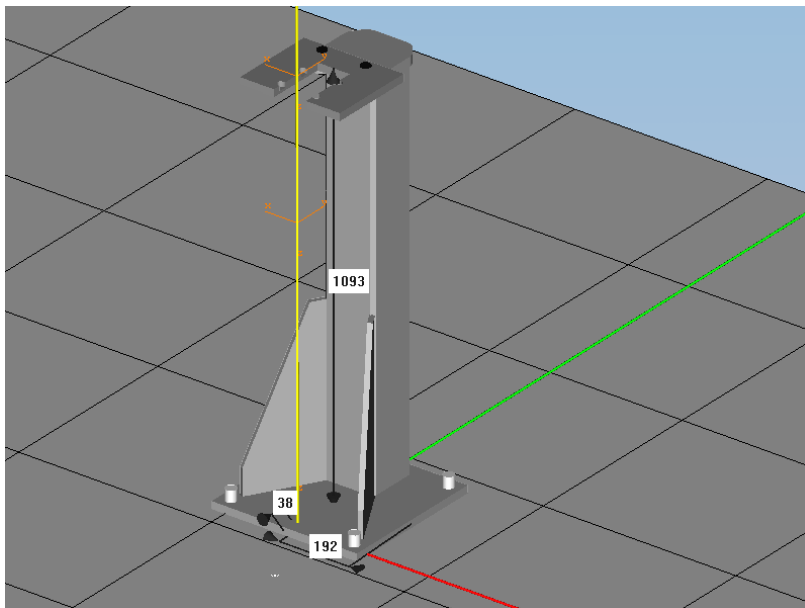
Part	X(mm)	Y(mm)	Z(mm)	Rot Z(deg)
FixtureTableShunk1	1574	2100	0	0



Part	X(mm)	Y(mm)	Z(mm)	Rot Z(deg)
PalletTable1	3240	-2000	0	0
PalletTable2	3670	1425	0	90
PalletTable3	4392	1425	0	90



Part	X(mm)	Y(mm)	Z(mm)	Rot Z(deg)
Tool-Stand	-1800	-1417	0	-90



Part	X(mm)	Y(mm)	Z(mm)	Rot Z(deg)
GripperStand	4336	-408	0	-90