

## Automatic Coordination and Collision Avoidance Using Sequence Planner and Process Simulate

*Master of Science Thesis in Automation*

Muhammad Sualeh

Umair Qadeer

Department Of Signal and Systems

*Division of Automatic Control, Automation and Mechatronics*

CHALMERS UNIVERSITY OF TECHNOLOGY

Göteborg, Sweden 2011

EX071/2011



Master of Science Thesis in Automation  
REPORT NO. EX071/2011

# Automatic Coordination and Collision Avoidance Using Sequence Planner and Process Simulate

Muhammad Sualeh  
Umair Qadeer



Department of Signals and Systems  
*Division of Automatic Control, Automation and Mechatronics*  
CHALMERS UNIVERSITY OF TECHNOLOGY  
Göteborg, Sweden 2010

# Automatic Coordination and Collision Avoidance Using Sequence Planner and Process Simulate

© Sualeh, Umair, 2011

Report No. EX071/2011

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

Göteborg, Sweden 2011

## ABSTRACT

In recent years consistent growth of market has created a demand for higher production rates. To keep up with this competitive industry, Volvo also has to introduce new car models on regular basis with higher capacity and speedy productions.

Production lines at Volvo comprises of several robot stations which leads to more operation time and low production capacity. Also these robot stations are subjected to design modifications when there are new car models. Spot welding is one of the operation for which Volvo wants to reduce operation time and design a separate generic welding station to avoid modifications for different models.

In cooperation with Volvo, models of the spot welding stations were obtained with configurations of five robots and 50 seconds cycle time for each station respectively. Volvo encouraged redesigning of stations by addition of robots and redistribution of welding tasks so that spot welding operation is completed within two stations with same cycle time of 50 seconds. Thus such a station is successfully designed in a simulation tool by addition of four robots, making it to total of nine robots per station which enables us to achieve the desired goal.

Since putting in more robots makes the station flexible but also highly complex as the collisions of robots in the mutual exclusion zones may and can take place. Keeping this in view, new methodology is to be derived which will help to avoid collisions by sequential coordination of robots. This can be done by automatic detection of such zones and setting additional conditions on robots. So collision and blocking free stations can be placed in production lines.

Hence, in this thesis the proposed method for automatic coordination was implemented by developing GUI for collision free operation through simulation tool Process Simulate. Further the verification check for any deadlock was done by using verification tools Sequence Planner and Supremica.

So the suggestions made to Volvo & Chalmers Uni. are that due to overall height limitations for welding station, robots should be installed in two levels but studies for three level robot station can be of more interest. Also there is space for further development of Sequence Planner tool with respect to interpretation of analysis.

Keywords:

Tecnomatix, Process Simulate, Swept Volume, Sequence Planner, weld operation



## ACKNOWLEDGEMENTS

We would like to thank our examiner Peter Falkman for his consistent support in this work. We would also like to thank Bengt Lennartson and our co-supervisor Mohammad Reza Shoaie who always discussed and advised us in our work.

Our special thanks to Stefan Axelsson for his efforts and cooperation in execution of this work.

We would also like to thank our parents whose continuous efforts lead us here and our friends who kept us motivated and energetic.

Muhammad Sualeh  
Umair Qadeer





# TABLE OF CONTENTS

<b>Abstract</b> .....	i
<b>Acknowledgements</b> .....	iii
<b>Table of Contents</b> .....	v
<b>Notations</b> .....	1
<b>Introduction</b> .....	2
<b>Project Scope</b> .....	4
<b>Purpose</b> .....	4
<b>Objectives</b> .....	4
<b>Approach</b> .....	5
<b>Limitations</b> .....	5
<b>Background</b> .....	6
<b>Process Simulate</b> .....	6
<b>Modeling in Process Simulate</b> .....	8
Creation of a New Study.....	8
Creation of Operation .....	9
Constraints of Weld locations.....	9
Projection of Weld Locations .....	10
Verification of Weld location Orientation .....	10
Positioning of Robots.....	10
Assigning Weld operations to robots.....	10
<b>Sequence Planner</b> .....	11
<b>Extended Finite Automata</b> .....	12
<b>Supremica</b> .....	13
<b>Modeling</b> .....	14
<b>Provided Model</b> .....	14
<b>Specifications for new model</b> .....	15
<b>Complex Model Design</b> .....	15
<b>Methodology</b> .....	19
<b>Operation model</b> .....	19
<b>Collision Detection Algorithm:</b> .....	20
<b>Guards Insertion and Generation of Non-blocking Controller</b> .....	21

<b>Implementation</b> .....	22
<b>Division of Weld Operations (GUI)</b> .....	22
<b>Collision Avoidance (GUI)</b> .....	24
<b>Volume Creation</b> .....	25
<b>Collision Test and Generation of XML File</b> .....	<b>Error! Bookmark not defined.</b>
<b>Contribution</b> .....	29
<b>Conclusion</b> .....	30
<b>References</b> .....	31

## NOTATIONS

- .NET** A programming infrastructure created by Microsoft for building, deploying, and running applications and services that use .NET technologies, The .NET Framework contains three major parts; the Common Language Runtime, Framework Class Library and ASP .NET
- action** If a guard condition is true and the event occurs, actions in updating the variables may follow.
- API** Application Programming Interface implemented by a software program enabling the program to interact with other software.
- C#** Object-oriented programming language developed by Microsoft.
- EFA** Extended Finite Automata which is an augmentation of an ordinary automaton, where transitions are associated with guards and actions.
- event** Represents an incident that causes an automaton to move from one state to another.
- guard** A set of conditions that need to be fulfilled for an event to be enabled.
- GUI** It is a graphical (rather than purely textual) user interface to a computer.
- plug-in** A set of software that makes it possible to add new features to a larger software application.
- Process Simulate** The process simulation application in the Tecnomatix environment.
- weld operation** It is an operation assigned to a robot in Process Simulate in simulation.
- swept volume** It is an object created by Process Simulate indicating the space that any resources occupies while performing any operation.
- Sequence Planner** A formal verification tool developed at Chalmers University of Technology.
- Supremica** Modeling and analyzing tool for sequence of operations.
- Tecnomatix** An application for e.g. part manufacturing, resource planning and plant simulation owned by Siemens Product Lifecycle Management Software Inc.
- XML** Widely used programming format. It stands for eXtensible Markup Language.

## INTRODUCTION

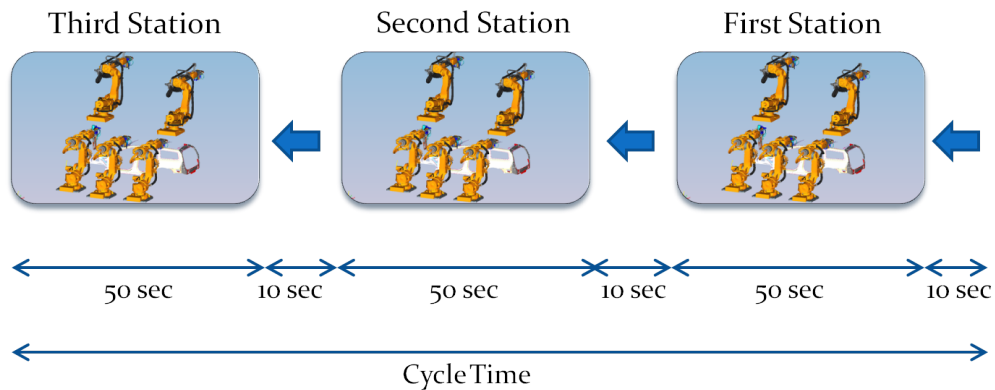
Today, companies are facing a demand of higher production rate and shorter time-to-market. On the other hand they need to introduce new products more often to compete in the market. Since the production systems are based on a product therefore whole system is to be redesigned for each new product. The activities involved in preparing these systems include many different tasks. These tasks range all the way from the early product design to the final preparation of the shop floor. To reduce the time spent in these tasks, companies have adopted a trend towards the manufacturing systems that are flexible.

There are some requirements on the flexible systems. The most important are cycle time and collision avoidance. Cycle time shall not only be short but it should be the same as the rest of the systems so that a balanced flow through the entire manufacturing system is achieved. Hence, resources like robots are subjected to work most of the time to keep the efficiency high. Such resources being in vicinity may end up in a collision or increased cycle time, so the designers have to consider these requirements in preparing the flexible systems.

The preparation tasks are traditionally carried out using different types of software by different companies. Simulation tools are used to simulate the system and to study its requirements. However, the result of this work is used in a manual manner such as software indicates a collision and designer changes the design or inserts a condition to avoid it. It becomes tedious as the complexity of the system grows, thus some automatic mechanism is required to be developed within the simulation tool for this task.

This thesis is carried out in cooperation with Volvo Car Corporation (henceforth VCC). VCC's vision is to reduce the cycle time of the spot welding process by taking a step towards flexible system. The most important factor that encouraged VCC for taking this step is limited production space for future. To overcome the complexities and to fulfill the requirements of flexible systems VCC is also interested in the research of automated methodology.

The models of spot welding station were provided by VCC. In the current configuration, shown in figure 1, spot welding of the car body is carried out in three identical stations. Each station has five robots that perform welding. Each station has the equal cycle time of fifty seconds. The VCC encouraged to redesign the configuration with addition of robots and redistribution of weld spots among them, such that the task of spot welding is completed in two stations and time of each station remains within fifty seconds.



*Figure 1: Three stations currently installed for spot welding operation*

Designing a complex station to be collision free needs an automated method to foresee the possibility of collisions. Designer may create a collision free station and never encounter any collision in the simulation but in reality due to delay or robots being out of synchronization may result in destruction. For this reason coordination among all the robots is required. Finding such situations and to set additional conditions for tackling them manually is practically impossible.

In the present thesis the proposed method for automatic coordination of a collision-free flexible manufacturing system [1] is implemented. This method automatically detects all the possible collisions and sets additional conditions on the corresponding operations of the resources to avoid them. It also describes a way to verify that collision free station does not get into a blocking situation. The simulation tool in which it is implemented is Process Simulate.

Motivation of this study is to utilize the functionalities of Process Simulate. The main idea is to automatically generate swept volumes of the resources that are the spaces they occupy while performing their operations. The intersecting regions of these volumes are the mutual exclusion zones, which more than one resource at a same time may occupy.

The operations in Process Simulate cannot be visualized in a way that information of collisions or conditions is viewed. Therefore these operations are modeled in Sequence Planner [2], a tool developed at Chalmers for verification of discrete systems. This tool can also transform the modeled operations into EFAs (Extended Finite Automata) [3] to verify controllability or blocking situations using Supremica [4].

## ***Project Scope***

The scope of the project is to develop an interactive interface within process simulate that automatically generates the sweep volumes of robots, finds their intersection and puts additional conditions on the operations that correspond to the intersecting volumes.

## ***Purpose***

The purpose of this thesis is to implement a method for automatic generation of controllers for a non-blocking and collision free manufacturing system.

The intention is to redesign a system with reduced cycle time using the proposed method, and to embed it into a simulation tool for future use.

## ***Objectives***

The objective of this thesis is

- Developing a toolbox for automatic generation of non-blocking controller for a collision-free manufacturing cell in the Process Simulate that performs,
  - Identification of operations in Process Simulate
  - Generates volumes for operation assigned to the resources
  - Performs pair-wise intersection test over all volumes
  - Identifies the sequences of operations in the Process Simulate model and create the corresponding Sequence Planner model
  - Adds interlocks (collision avoidance guards) based on the pair-wise intersection test on the corresponding operation model in the Sequence Planner
  - Generates a non-blocking controller for the collision-free manufacturing system
  
- Designing of a lean collision-free manufacturing cell in Process Simulate such that,
  - Weld stations are reduced from three to two
  - Cycle time of each station remains within fifty seconds
  - No robot is idle for more than 10% of cycle time
  - Minimum distance between moving parts is 50mm
  - Designed station has a height below 7 meters
  - Specification of each robot to be considered since any type, make and number of robots can be used
  - Designed station is collision free

## *Approach*

The model of weld station provided by the VCC will be redesigned such that the weld operation is completed within two stations instead of three. More robots will be added to each station and weld spots will be equally distributed among them.

The requirements of this station will be tackled by the proposed methodology that is discussed in [1]. This methodology will be implemented in a way that it can readily be used in future for the design of such systems to be non-blocking and collision free.

Implementation of this method involves the usage of functionalities offered by Process Simulate. Tecnomatix.NET API will be used to interact with Process Simulate v9.1.2. This API allows the use of buttons, commands and functions within Process Simulate environment. C# programming language will be used to develop a GUI within Process Simulate. The developed GUI will operate Process Simulate on user's commands. Each command from the user will trigger series of tasks following the algorithms, like generation of volumes, intersection test and etc.

To check that the developed manufacturing system is non-blocking, the defined sequences of operations along with their conditions will be automatically exported to the Sequence Planner [2] via XML. Operation models can be viewed and verified using Sequence Planner.

Model of operations can be verified and synthesized in Supremica [4] to check Controllability and non-blocking where additional guards can be suggested to the original models to guarantee the required behavior.

## *Limitations*

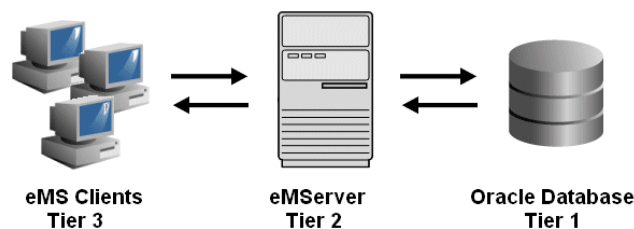
This thesis in terms of collision avoidance does not contain the collisions that occur with static objects. It is also out of scope of this thesis to find the optimized path for the robots to avoid collisions and to perform certain operation.

## BACKGROUND

### *Process Simulate*

Process Simulate is a simulation tool among the suite of engineering study tools that are collectively known as Tecnomatix that is owned by Siemens Product Lifecycle Management Software Inc [6]. It provides an environment where simulation of a manufacturing process can be created and planning of processes, resources and parts can be carried out virtually. To automate the functions performed in Process Simulate and to retrieve the information, Tecnomatix .NET API is used. This requires the knowledge of software structure. Organization of software is based on three layers that are database, server and clients. This is commonly referred as 3-Tier Architecture.[5][6]

A single Oracle Database that contains several oracle accounts (schemas) acts as a server to the multiple eMServers (middleware) each of which further interacts with the client's (eMS Client) desktops connected to a single eMServer.[7]

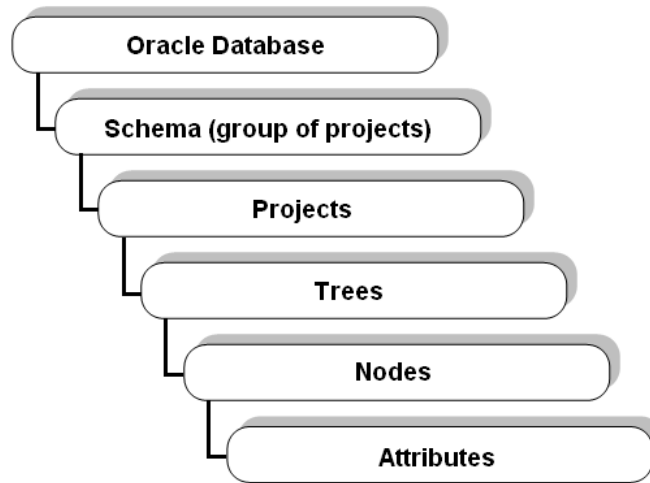


*Figure 2: 3-Tier Architecture*

First tier is the Oracle Database Server containing schemas where each schema holds certain type of data, the main task is to manage and to control data update. Second tier is the eMServer that operates as a middleware, it provides services to the clients, such as, to retrieve data from the database. Third tier consists of different clients that can be applications within Tecnomatix.[8]

Schemas in the database hold the projects that are comprised of parts, operations, resources and manufacturing procedures in the form of trees/objects of nodes. Each tree in a project consist of similar type of nodes, they contain the information of other type of nodes as attributes.





*Figure 3: Data structure in Tecnomatix.*

In order to create a simulation in Process Simulate there are number of steps that are followed. In the beginning of the simulation design, typical type of study is created e.g. line simulation, robcad, etc. A study is a small portion of the project that can be loaded into the simulation environment such as CAD models of parts, resources and operations.

After the creation of appropriate study type, operations are created and defined. Operation is referred to as the path of motion of any object such as device operation, weld operation, etc. Having done with the definition of operations they are combined to form a complete process. Each sequence can be demonstrated and verified. The paths of operations in sequences can be monitored and altered while analyzing the design. [8][6]

In this thesis it is required to obtain the robotic operations, to generate their swept volumes and then to sequence them. For that purpose Tecnomatix .NET API can provide an application to act as a client, in the environment of Process Simulate.

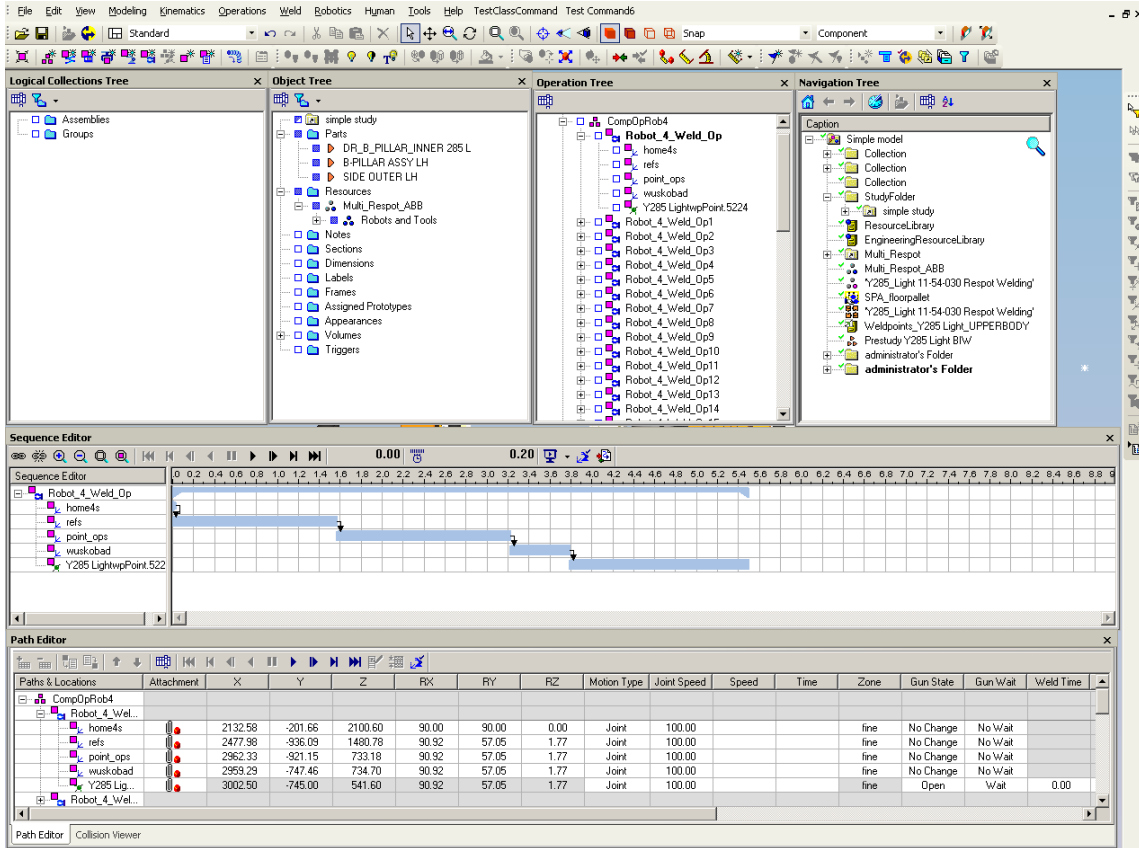


Figure 4: Trees in process simulate.

## Modeling in Process Simulate

The work flow of the simulation design, in Process Simulate, is comprised of series of steps that starts from definition of resources, definition of parts, assignment of operations, and sequencing of operations down to final deliverable. Key steps will be discussed here that were involved in the thesis work.

### Creation of a New Study

This is the first Step in the creation of a simulation model, there are various types of studies available in the Process Simulate, the two most commonly used studies are Robcad Study and Line Simulation Study. The Robcad Study is time based and all the operations are synchronized, whereas the Line Simulation Study is event based. It is also called as CEE (Cyclic Event Evaluation), where logical conditions on the basis of signals can be applied. Common in all the studies are parts, operation, and resource trees, which represent a desired smaller area of the complete project, are loaded into the simulation environment. In this project Robcad Study is used and it contains the parts of

one side of the vehicle (parts), set of weld spots (operations), floor pallet on which weld operation is performed, and set of three types of ten robots with weld guns (resources).

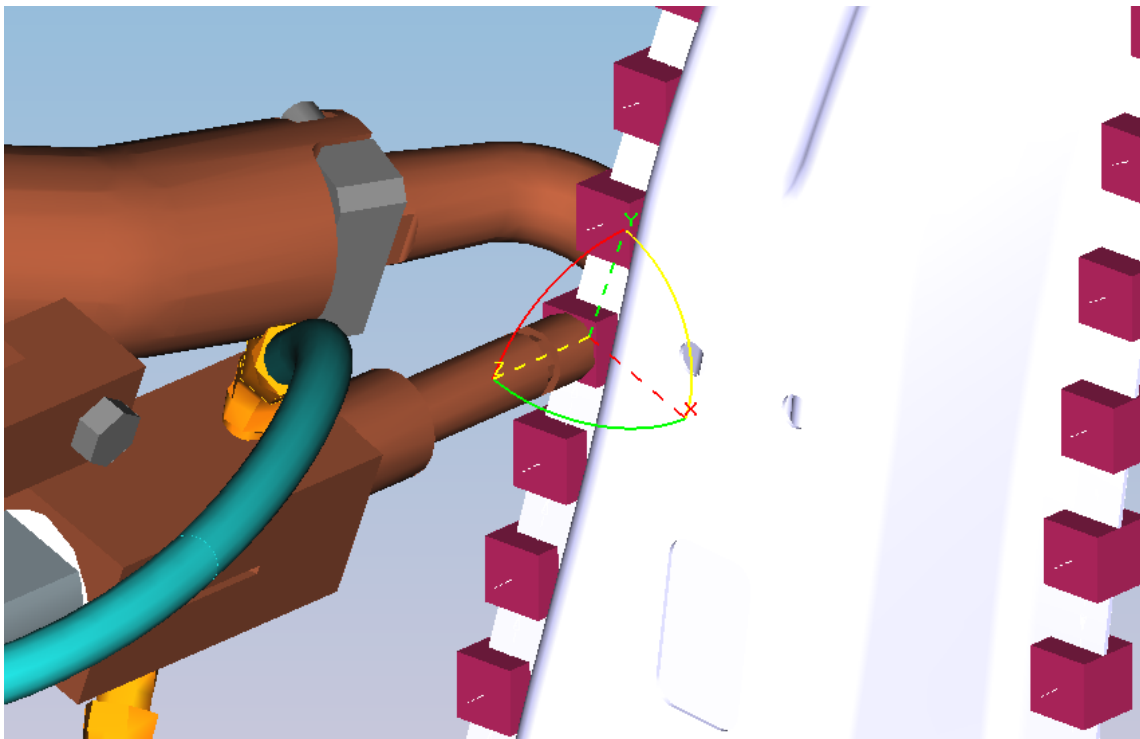
#### *Creation of Operation*

An operation is typically related to a path of motion of an object. It can be a robot welding, a human activity, or a part moving freely through space. Operation in this project is the path of robot through which robot has to weld the spots in sequence. It is also called as weld operation.

A weld location in Process Simulate is a type of frame that contains the position and orientation of the TCP (Tool Center Point) frame at a weld point. Weld locations are generated with their origins placed on the Manufacturing Features. The origins of the locations are the actual welding points.

#### *Constraints of Weld locations*

Orientation of a weld point is very important for the quality of the welding therefore, one axis must be perpendicular to the surface on which welding is to be performed and another axis to define the approach angle, that is the direction through which the weld gun should approach the location.



*Figure 5: in this case z-axis is the perpendicular axis along which weld-gun impacts, its rotation determines the approach angle.*

### *Projection of Weld Locations*

To automatically set the orientation of the weld points considering the above mentioned constraints there is a function in Process Simulate that projects the weld locations, these locations however can later be changed especially approach vector needs to be adjusted in later stages of simulation design.

### *Verification of Weld location Orientation*

In this step the specified gun is taken to all the preliminarily projected weld points to verify the perpendicular and approach vectors. This step is repeated time and again to ensure that gun itself or its part is not colliding with the object.

### *Positioning of Robots*

This is the most delicate task as there are a number of constraints that are to be considered while placing the robots,

1. Estimation of the number of robots that are required to complete the task in particular time. For example, if there are 400 weld points and each weld point takes approximately 3 seconds and available time is 100 seconds then roughly 10 robots can perform this task each having 20 weld spots.
2. Reachability of the weld-points is also equally important. Considering the above mentioned example; the targeted 20 weld points for a robot should be in its reach.
3. Robots should be positioned such that they do not collide.
4. If robots are mounted on the levels, walls or upside down, then their own specifications should be met.

### *Assigning Weld operations to robots*

After positioning of robots, the targeted weld operations are assigned to the robots and simulation checks are performed. Sequence of welding can also be adjusted and planned. For this particular step Process Simulate has a built in function by the name of Weld Distribution Center, this function automatically detects and assigns the weld spots that are in the reachability area of a robot. Through this function the number of weld spots assigned to a robot can also be visualized. However sequencing of these weld-operations is manually carried out.

There is also need of via points in each weld operation; they are the locations which the TCP of the robot tracks to reach the weld locations. Via points are required in the beginning and ending of the weld operation so as to avoid the collision with the parts on which welding is performed. They are also used to alter the path of robot to avoid the collisions with other robots. Each weld operation has at least two via points.

### *Collision Detection*

When weld operations are distributed among the robots, simulations are run time and again. Initially a single robot's operation is loaded into the sequence editor. Behavior of a robot can be observed in the graphical viewer. All undesirable moves in terms of collisions or tool adjustment can be corrected. When all the robots are performing well, then whole station is loaded into the sequence editor for the study of collisions among the robots.

For collision detection, Process Simulate offers a function by the name of Collision Viewer. This function can be used to check the feasibility of the operations planned in the assembly process and to ensure that the process is collision-free. Designer can use the Collision Viewer to display planned collision sets that are of particular interest and to hide others. For example, two robots are far apart from each other and there is no chance of their collision, so there is no need to put them in same set. When simulation is run with collision detection functionality, it generates a report for collisions which can later be used.

By following all the mentioned steps above a collision free station can be designed, but only if it is assumed that robots in a real station will perform exactly at the same time instances and at same speed. This assumption is impractical since any robot in a real station can experience a failure or even a delay resulting in a massive collision.

### *Sequence Planner*

Sequence Planner is a Java based application developed by department of Signal and systems at Chalmers University. In this application instead of manual construction of the sequences they are viewed on the basis of relationships among the operations. The sequence of robot operations can be visualized using different views. It helps to understand the relationship between the operations and also the overall system behavior [8].

In this thesis Sequence Planner is used to view the sequence of operations of the robots in a work cell sharing the common zone on the basis of the relations. In figure 6, the model of robotic operations is shown. The green boxes represent the operation, and text written on the top of each green box is the precondition for that particular operation. This condition needs to be fulfilled before the execution.

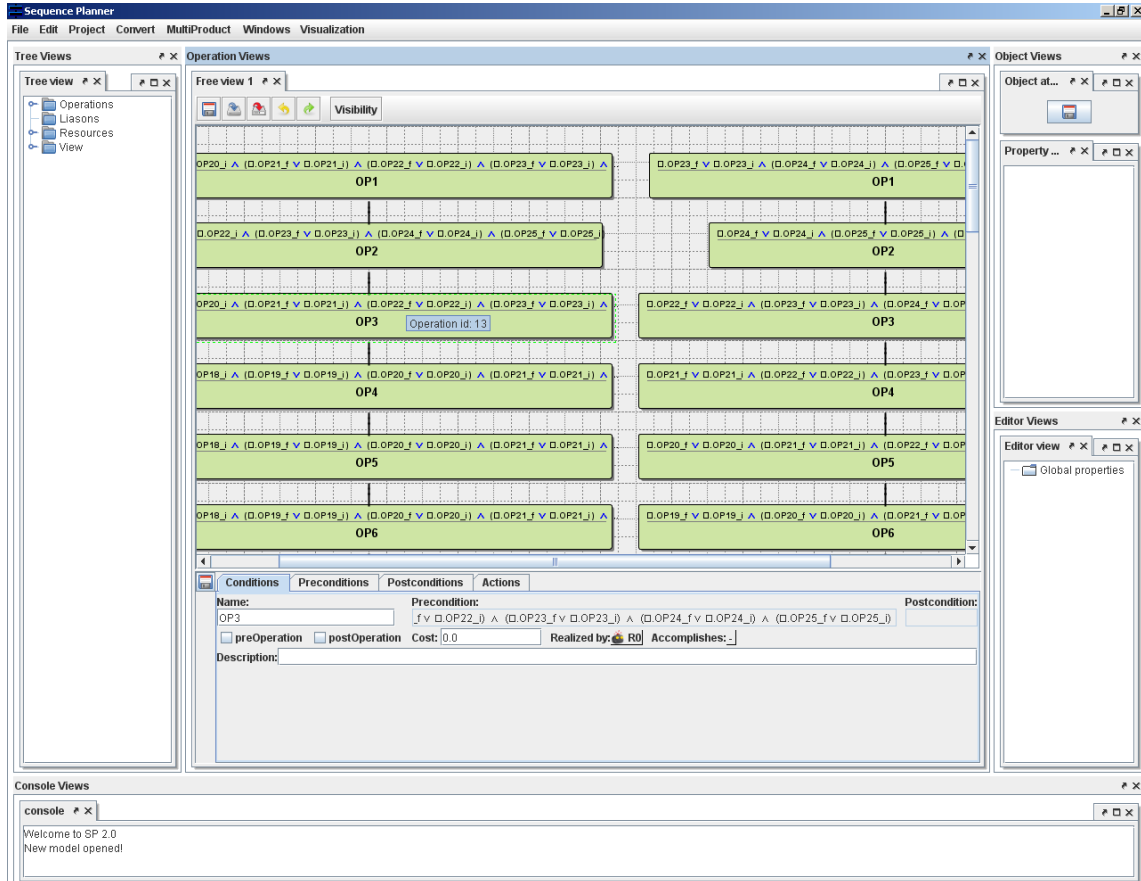


Figure 6: Sequence Planner

### Extended Finite Automata

Such systems whose state space is discrete and state changes are associated with occurring of an event at a discrete point in time, can be regarded as Discrete Event System (DES). They can be formally be represented in the form of Deterministic Finite Automata (DFA). DFA is a 4-tuple  $A = \langle Q, \Sigma, \rightarrow, q_0 \rangle$ , where  $Q$  is a set of finite states,  $\Sigma$  is a set of events,  $\rightarrow$  is a state transition matrix and  $q_0$  is the initial state.

An Extended Finite (EFA) is an ordinary finite automata augmented with variables, guard formulas and action functions[3]. The variables are updated by the occurrence of events that trigger the functions at the transitions. If on these variables guards (preconditions) are applied, a transition is enabled and fired when its guard is satisfied.

An EFA is defined as a 7-tuple  $A = \langle L \times V, \Sigma, \mathcal{G}, \mathcal{A}, \rightarrow, (\ell_0, v_0), M \rangle$ , The set  $L \times V$  is the extended finite set of states, where  $L$  is a finite set of discrete locations and  $V$  is the finite domain of an m-tuple of variables,  $\Sigma$  is a nonempty finite set of events.  $\mathcal{G}$  is a set

of guard predicates over  $V$ ,  $\mathcal{A}$  is a set of action functions from  $V$  to  $V$ , where each function maps the present variable values to the variable values of the next state.  $\rightarrow$  is a state transition relation,  $(\ell_0, v_0)$  is the initial state, and  $M$  is a set of marked states.

The event set, is comprised of two disjoint subsets, the controllable and the uncontrollable events. A controllable event can be inhibited by the supervisor whereas uncontrollable event cannot be inhibited. When an event occurs in the plant that is uncontrollable, the supervisor must be able to follow it, or risk losing control over the system. Additional information of EFAs and their working can be found here [9].

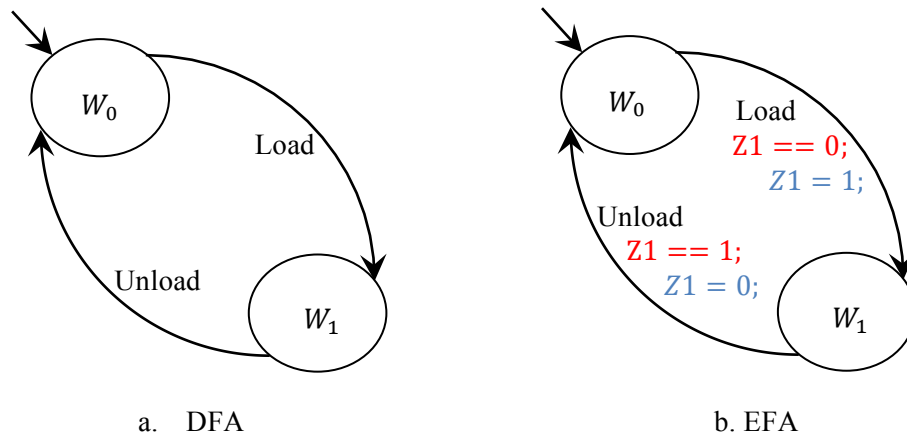


Figure 7: (a) Deterministic Finite Automata and (b) Extended Finite Automata are shown, both have two states  $W_0$  and  $W_1$  and both have same events Load and Unload. In EFA along with event there is a guard formula in red and actions in blue.

In figure 7(b), EFA has one variable constituting the guard and action. For event Load to be enabled, the guard formula has to be true, i.e. variable  $Z1$  is 0. When the guard condition is true and a transition is taken, the variable is updated to  $Z1$  to 1. There can be more than one variable with logical relations that can be used in guards.

### ***Supremica***

Supremica is a tool for formal verification and synthesis for control systems, its typical usage is building of uncontrolled system, behavior of closed-loop system by creating specifications, generation of supervisor and generation of PLC executable code.[6] Information on verification and synthesis of discrete event systems can be found here [10].

# MODELING

## *Provided Model*

For this project the provided model of welding station was in robcad study. In simulations only right half of the vehicle was spot welded, assuming other half to be at exact symmetry. It also contained 3D models of ABB, KUKA and COMAU robots all in separate compound resources but at same positioning and operations.



Figure 8: Snap shot of the provided model, only ABB robots are shown and base for two robots on the top, is removed for clarity.

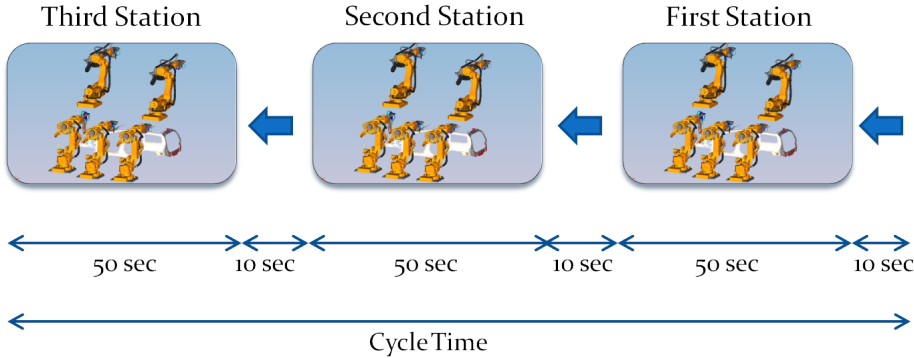


Figure 9: Three stations currently installed for spot welding operation



In figure 8, compound resource of ABB robots is shown. It has two levels, robots on the top are ABB 6650 having different reachability regions then three ABB 6640 on the bottom. Provided model contained five robots per station and three stations to complete the spot welding of approximately 375 weld spots, that is, one side of the vehicle. Complete welding station is shown in figure 9. Model also contained compound resources of COMAU and KUKA with same positioning and operations (weld point distribution).

### ***Specifications for new model***

Provided model was to be improved on the basis of the following specifications:

- Spot Weld operation was previously being completed in three stations. The objective was to perform this task within two stations in order to reduce the occupation of space.
- The cycle time of each station should not exceed 60 seconds, of which 10 seconds are for arrival of a fixture with the vehicle body, its adjustment and brushing of the weld guns.
- No robot should be standby for more than 10% of the cycle time that is 0 to 5 seconds for which a robot can halt or wait for other robot.
- Minimum distance between moving parts of the robots should be around 50mm.
- The designed station should have a height below 7 meters.
- There was freedom in selection of type and make of robots and number of robots. However ABB was preferred and selection was to be based on the basis of pay load for the weld guns that weighs around 100 to 150kg.
- Each robot has its own specification and flexibility like some models can be mounted upside down, others cannot, this was to be considered.
- Main task was that the prepared station should also be collision free.

### ***Complex Model Design***

Designing of the model started with average time consumption of one weld spot that is about 2.5 seconds. If one weld spot takes around 2.5 second then in one station robot can weld at most 18-20 weld spots. This implies that one robot can weld up to maximum 40 weld spots in two stations. Then 375 spots should be done by around 10 robots in two stations. If delays and time consumed by the rotation of weld gun in a non linear curve are considered then 10 robots is a realistic solution.

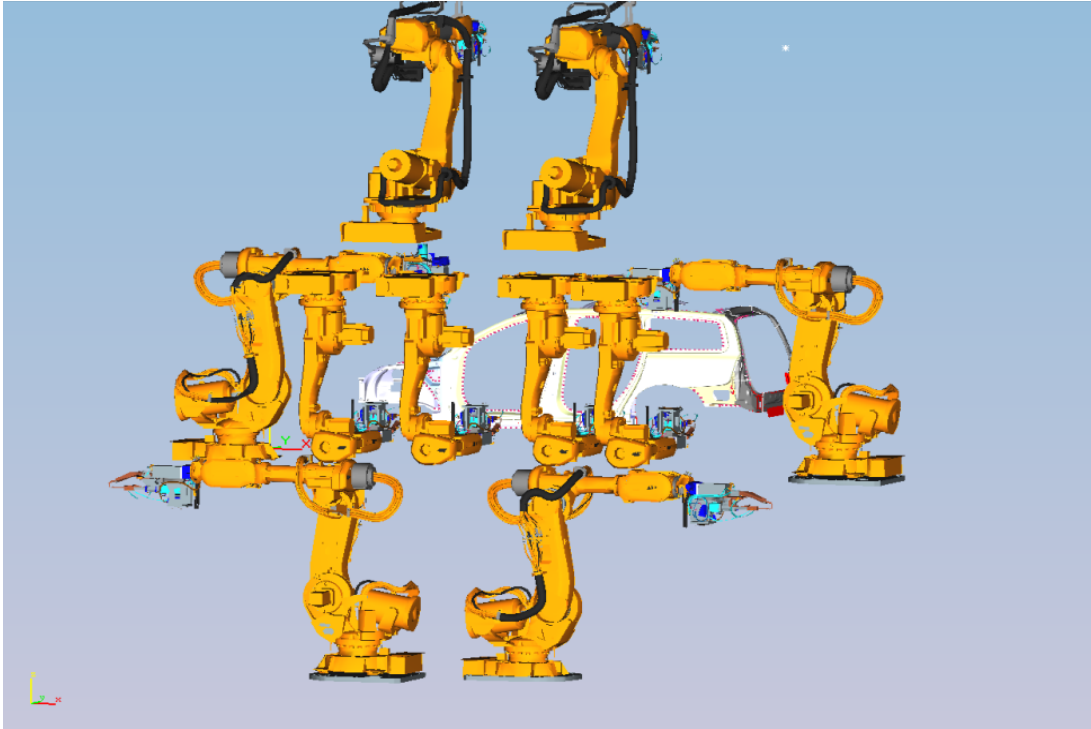


Figure 11: Snap shot of the initially designed model having 10 robots two ABB 6650 on top, four ABB 6620 along centre (upside down) and four ABB 6640 one on the rear and back and two on the bottom.

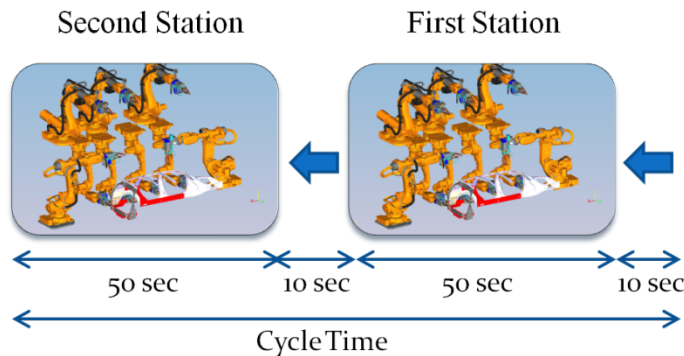


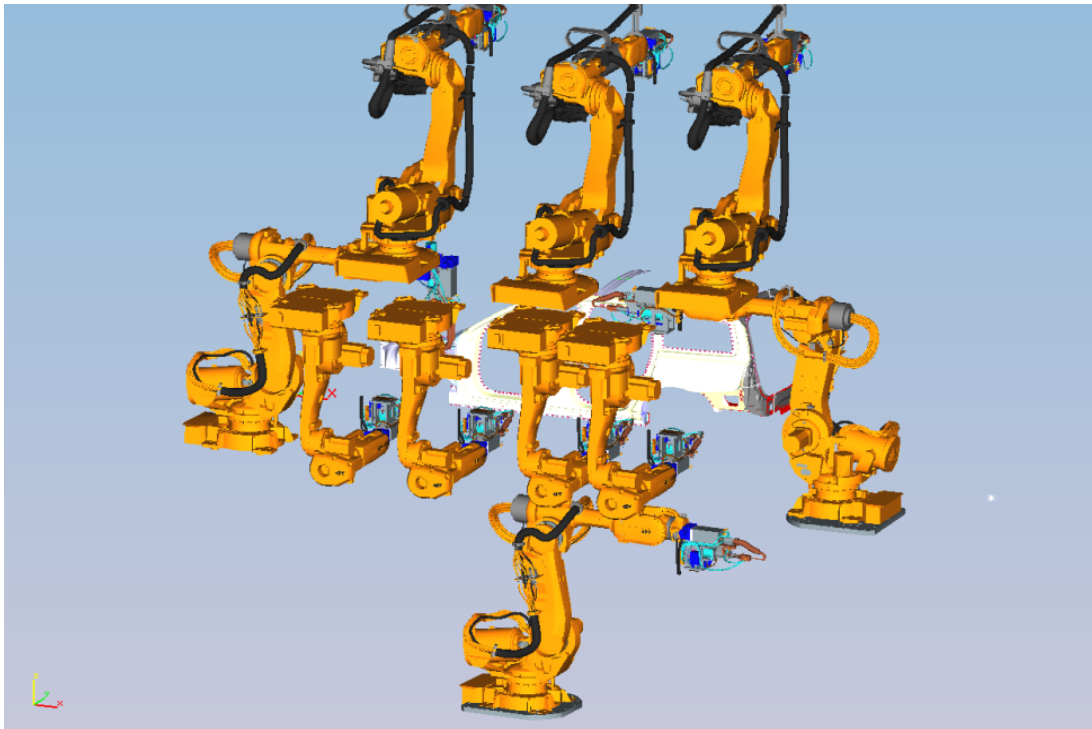
Figure 10: Proposed welding station having two identical stations

Adjusting ten robots in the place of five should have appropriate position where each can share equal amount of spots, so weld spot concentration areas were marked. Such areas were along the center and rear top. For the centre region there was a need of smaller robots that could mount on the wall or upside down, since larger robots have joints and elbows that must be maximum stretched such that they do not come in the way of other robots. For center region ABB 6620 robot was used, having smaller size, same payload and ability of being mounted upside down. For whole top region ABB

6650 were used having the reach ability area below the mounted surface. Two ABB 6640 robots for bottom and one for front and back each was sufficient since they had reachability for these regions of the vehicle.

After weld distribution in the model shown in figure 8, it was observed that bottom left robot got less weld spots and there were excessive spots on the rear top that required a robot there.

To move the bottom robot to the top, its type also had to change due to the difference in reachability area, so bottom left robot was removed and new robot was placed on the rear top location, this resulted in an equal distribution of the weld points in two stations while maintaining the cycle time of each station. This design is shown in figure 9.



*Figure 12: Snap shot of modified station with change of type and position of robot.*

Model in figure 12, satisfied all the given specifications but later it was informed that even if total height is within seven meters, in real station there was space occupied by the ventilators over the top. So this design was not acceptable since application of this design required major changes in the line like digging of the floor for the bottom robot, so suggestion was to omit the bottom robot along with its weld spots for this project.

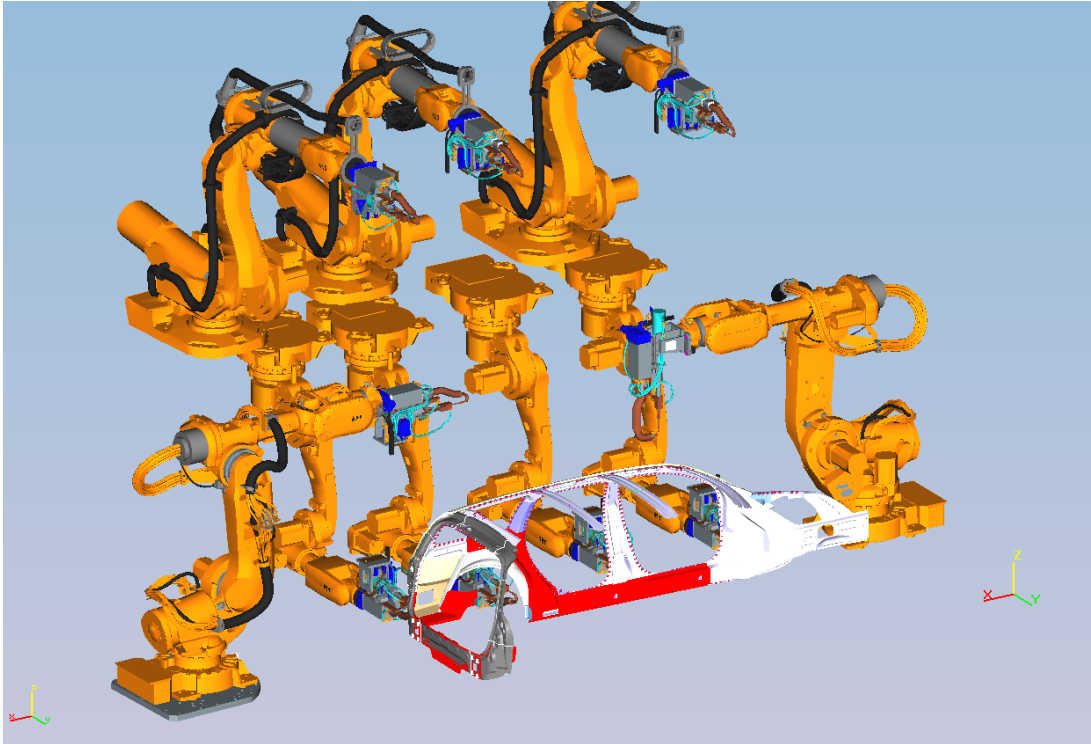


Figure 13: Snap shot of working station with removed bottom robot.

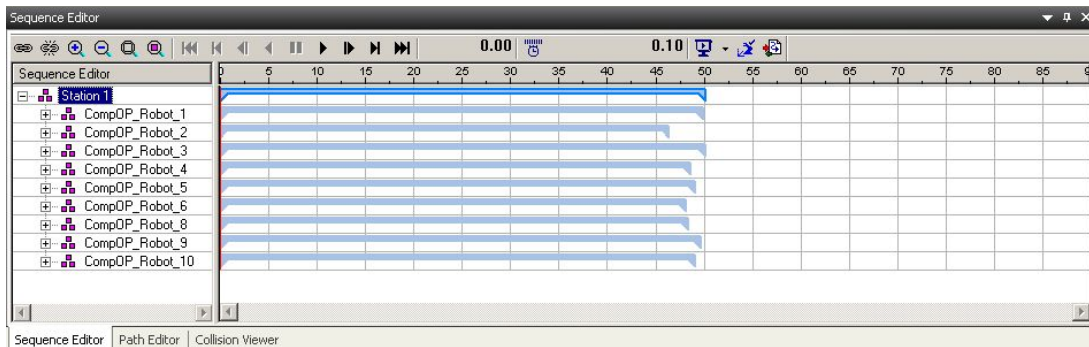


Figure 14: Snap of a Sequence Editor showing the cycle time of first station, where all the robots have active time between forty five to fifty seconds.

Figure 13 shows the final station that fulfilled all the specifications, neglecting the bottom robot with its assigned weld points. Figure 14 shows the cycle time of nine robots of first station. The time consumed by each robot in performing a weld operation within first station is represented by a bar. All the bars are restricted within forty five to fifty reading on the scale this means that all the robots complete their assigned weld operations within forty five to fifty seconds in first station.

## METHODOLOGY

In this thesis as described earlier, focus is on the two main requirements of flexible systems that are cycle time and collision avoidance. The proposed method [1] suggests that a resource has to wait until the mutual exclusion zone is free. There are many mutual exclusion zones and if a resource has to wait before entering such a zone to avoid a collision then cycle time is increased. This enhances the importance of sequence in which operations execute.

It is possible to set the sequence of operations in a way that the mutual exclusion zone is already free when another resource needs to enter it. Achieving such a design will result in a collision free station, but only in simulations. In real world a resource can experience a delay, which may result in a collision of all the resources in a complex station.

The proposed method [1] can address this problem and ensure the collision free station. Implementation of this method on the developed welding station will be discussed. The major advantage to use this method is that it does not require time information for any operation of particular resource e.g. which operation execute at what time instant. So if any resource experiences a delay in performing an operation then this method avoids the collision between different resources without considering the time delay. This might increase the overall cycle time but as an exception.

This method contains algorithms for collision detection and procedure for generation of non-blocking controller. First step in implementing this method is to model the operations.

### *Operation model*

Each resource or each set of resources have their own sequence of operations. Sequence of operations (SO) is a set containing operations and SO is finished when all the operations involved are in the finished state. Without any restriction on individual operation the basic assumption is that all operations can be executed in parallel. The order in which operations execute or physical location of the resources sometimes reduces the flexibility and typically leads to the restriction on the operations. Therefore, some operations must wait until other operations are finished. These sequential restrictions on the order between different operations are expressed by logical precondition. An operation is an EFA such that the set of discrete locations  $Q_k = \{O_k^i, O_k^e, O_k^f\}$ , the event set  $\Sigma_k = \{O_k^\uparrow, O_k^\downarrow\}$ , the set of transition condition  $C_k = \{C_k^\uparrow, C_k^\downarrow\}$ , the transition relation  $\rightarrow_k = \{\langle O_k^i, O_k^\uparrow / C_k^\uparrow, O_k^e \rangle, \langle O_k^i, O_k^\downarrow / C_k^\downarrow, O_k^e \rangle\}$ , and the initial and marked locations are  $\ell_k^i = O_k^i$  and  $L = M$  as shown in figure 12. Also, it is assumed that all events are controllable and local; communication is done by variables.

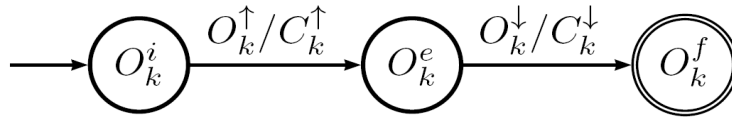


Figure 15: Three states of a model for operation; initial, execution and final

The transition for operation  $O_k$  from the initial discrete location  $O_k^i$  to the execution location  $O_k^e$  is enabled when the precondition  $C_k^\uparrow$  is satisfied, after which the transition can be fired and the start event  $O_k^\uparrow$  occurs. In the same way the completion event  $O_k^\downarrow$  can only occur when the post condition  $C_k^\downarrow$  is fulfilled.

For example if we have a single robot (resource) and two operation ( $O_1$  and  $O_2$ ) are assigned to that robot, where  $O_1$  is followed by  $O_2$ . The EFA models for these operations are illustrated in figure 13.

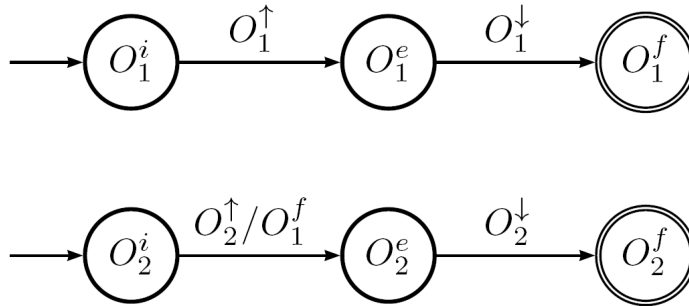


Figure 16: EFA models for operation  $O_1$  and  $O_2$

**Collision Detection Algorithm:**

This algorithm is presented in [1]; the objective of this algorithm is to detect collision between different resources sharing common regions in space. In the first part of this algorithm, three shapes for each operation of a resource are generated and this procedure repeats for each resource.

The shapes are the swept volumes of the resources while performing particular operation. The stand still position at the initial location, the sweep mode at the execution location and the stand still position at the final location of the operation.

In the second part of the algorithm, collisions are identified by pair-wise intersection test over the set of shapes generated from simulating the operations. The pairs with intersected shapes, excluding shapes from the same resource, are added to the set of pairs of colliding shapes  $\chi$ .

The algorithm is fairly simple to understand. Input of the algorithm is set of resources and the output is set of pairs of colliding shapes. Steps for this algorithm are given below.

**Input:**  $\mathcal{R}$ - the set of resources  
**Output:**  $\chi$ - the set of pairs of colliding shapes  
 $S$ - the set of generated shapes

```

foreach  $\mathcal{R}_l \in \mathcal{R}$  do
  foreach  $O_k \in \mathcal{R}_l$ . Operations() do
     $(S_k^i, S_k^e, S_k^f) \leftarrow \text{CreateShapeByLocation}(O_k^i, O_k^e, O_k^f)$ 
     $S \leftarrow (S_k^i, S_k^e, S_k^f)$ 
  end for
end for
while  $S \neq \emptyset$  do
   $S_i = S$ . RemoveFirst()
  foreach  $S_j \in S$  do
    if  $S_i$ . Resource()  $\neq$   $S_j$ . Resource() then
      if  $S_i$  intersects with  $S_j$  then
         $\chi \leftarrow (S_i, S_j)$ 
      end if
    end if
  end for
end while
return  $\chi, S$ 

```

### *Guards Insertion and Generation of Non-blocking Controller*

After getting the set of pairs of colliding shapes the procedure for interlocks generation starts. For each pair of colliding shapes, the corresponding forbidden states are identified. In order to avoid collision reaching the forbidden states are eliminated by adding guards to the corresponding events of the two involved operation models. The pseudo code for interlock generation is given below.

**Input:**  $\chi$  – the set of pairs of colliding shapes

```

while  $\chi \neq \emptyset$  do
   $(S_1, S_2) \leftarrow \chi$ . RemoveFirstPair()
   $(O_1, O_2) \leftarrow \text{DiscreteLocationsSetFromShape}(S_1, S_2)$ 
  for  $z = 1$  to 2 do
     $O_{t_1}^{l_1} = O_z$ . RemoveFirst()
     $k \leftarrow t_1$ 
    if  $(\exists O_{t_2}^{l_2} \in O_z : l_2 = f)$  then
       $k \leftarrow t_2$ 
    end if
     $C_k^\uparrow = C_k^\uparrow \wedge \sim O_{3-z}$ 
    if  $(\exists O_m^y \in O_{3-z} : y = f)$  then
       $C_k^\uparrow = C_k^\uparrow \wedge \sim O_m^e$ 
    end if
  end for
end while

```

In the first step of the algorithm, one pair of colliding shapes,  $(S_1, S_2)$ , is selected and also removed from  $\chi$ . For each of the colliding shapes,  $S_1$  and  $S_2$ , the corresponding locations are retrieved from the shapes. Since each shape represents a discrete location (or combination of discrete locations) in the related operations, to avoid collision, it is enough to forbid operations to enter the collision location concurrently. This condition is expressed by adding the safety guards to the transition condition  $C_k^\uparrow$  of operations. If the discrete location  $O_k^e$  is also forbidden to avoid an uncontrollable supervisor.

The input of the algorithm is a set of pairs of colliding shapes, for example the execution of first operation  $O_1^e$  of first resource is colliding with third operation  $O_3^e$  of second resource. So in first iteration, the pair of  $(S_1^e, S_3^e)$  is taken. The function *DiscreteLocationSetFromShape()* retrieves the location sets,  $O_1 = \{O_1^e\}$  and  $O_2 = \{O_3^e\}$ . In the next step, the condition  $\sim O_3^e$ , will be added to transition condition  $C_1^\uparrow$ , and the condition  $\sim O_1^e$  to the  $C_3^\uparrow$ , respectively.

After adding guards on the colliding operation, whole information is exported to readable format for sequence planner so that it can be visualized and verified using Supermica.

## IMPLEMENTATION

To implement the previously discussed method in Process Simulate, creation of interface is required. An interface through which a user can give commands and these commands can automatically perform series of steps following the algorithm. Two GUIs were created; first GUI is named as “Divide Weld Operations” and second as “Collision Avoidance”.

### *Division of Weld Operations (GUI)*

First, the specific amounts of weld points assigned to one robot are placed in one operation. Now the splitting of this operation in to small operations is necessary so that volumes for all these small operations are generated, by doing this the resolution increases. By increasing resolution commonly shared area is split into small operations, now both robots will check the presence of each other on specific area rather than ‘check and wait’ on complete common sharing area. This saves a robot from long idol time, hence solving one of the prime objectives of the thesis. Purpose of creating this GUI is to give choice to user to select the resolution size.



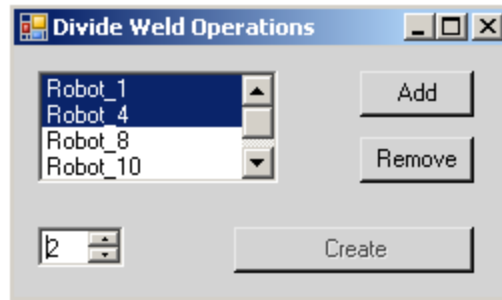


Figure 17: GUI for division of operations

The GUI shown in figure 17, will appear by pressing the button on menu bar in Process Simulate named as “Divide Weld Operations”. The selected resources from graphic viewer can be added to the “Resource List” by pressing “Add” button from GUI. Resources can also be removed from “Resource List” and by pressing “Remove” button in GUI. After adding required resources, the resolution can be set by specifying how many weld point each operation can have. This can be adjusted by using numeric up down counter on GUI. Operations of specific resolution for selected resources are created when “Create” button on the GUI is pressed.

In figure 18, division of operation is shown for better understanding. “Robot 10” has an operation with a name “Robot10”, having total of seven locations for operation. The names starting with “Y285 LightwpPoint” are the weld points and others are the via points. By using this GUI we split this operation into small operations where each operation has two weld points. To maintain the continuation each operation gets a trace of previous operation’s last location as its first, in the form of via point. In figure 17(b), operation “Robot\_10\_OP1” has a via point “viaRobot\_100” having the same location as the weld point “Y285 LightwpPoint 4128” of operation “Robot\_10\_Op0”. By using the described method, division of operations can be taken down to one weld point per operation, since Process Simulate does not allow creation of a weld operation with just one location.

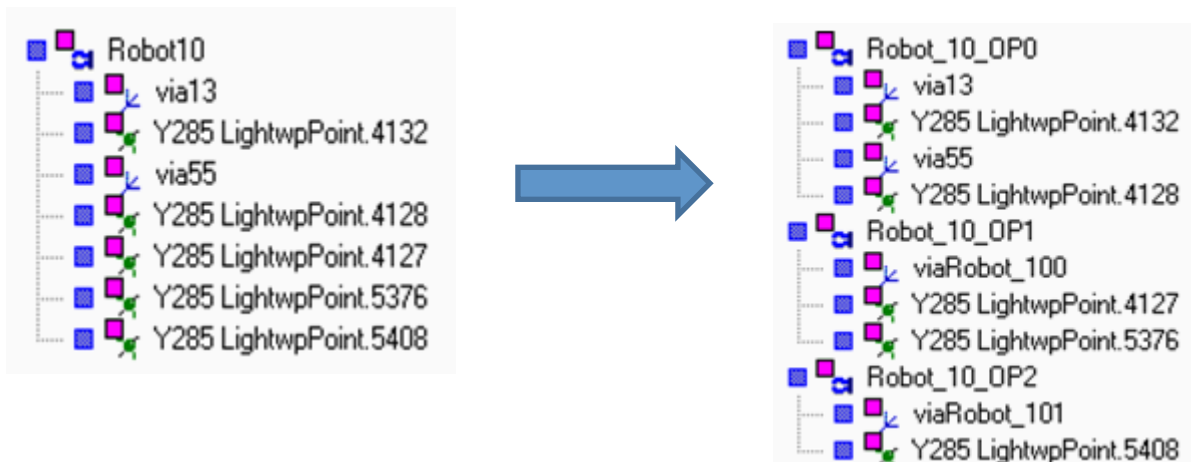


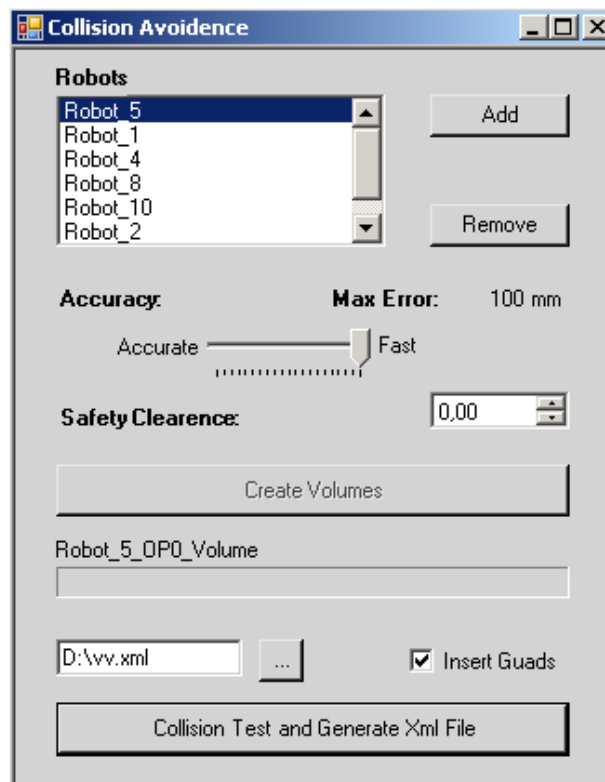
Figure 18: Division of operations in operation tree of Process Simulate

### ***Collision Avoidance (GUI)***

After creating small operation, the procedure for next task is as follows:

- Volume creation of resources corresponding to their operations
- Detection of collision between the generated volumes of the operations
- Export this information into xml to analyze it in sequence planner
- Insertion of collision-avoidance guards on the collided operations.

This GUI is shown in figure 19, it can be accessed by pressing the button on menu bar in Process simulate named “Collision Avoidance”. The selected resources in the graphic viewer can be added and removed from the GUI’s resources list by pressing “Add” and “Remove” button.



*Figure 19: GUI for Collision Avoidance; Volume Generation, Collision Test, Guards Insertion and export of XML for Sequence Planner*

It should be noted that only those resources can be added to the resource list that have simulating operations e.g. if it is tried to add a robot to the list that is not assigned any simulating operation with it then it will not be added because there is a filter programmed behind “Add” button which only allows resources with simulating operation or operations. Two buttons named “Create Volumes” and “Collision Test and

Generate Xml File” performs their functionalities on the selected resources in the resource list.

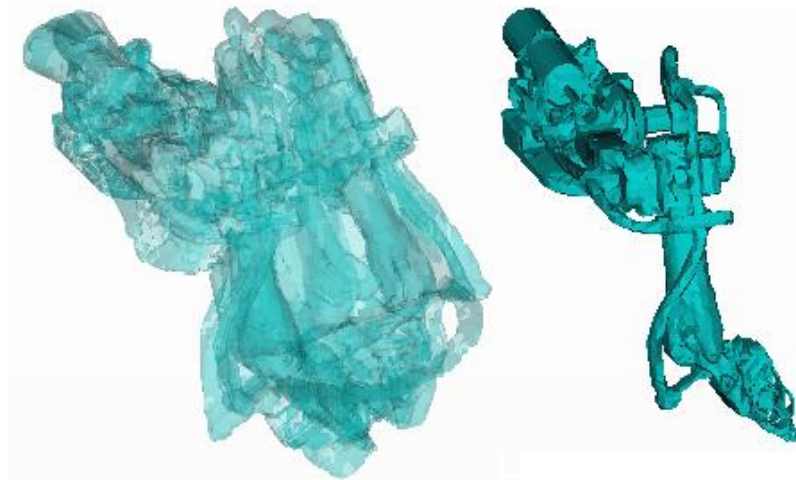
### ***Volume Creation***

This button creates the volumes for all simulating operations attached to the selected resource. The whole methodology explained above depends upon these volumes, based on these volumes collision detected and guards generated to avoid these collisions. There are two parameters which should be set before pressing these buttons, which are “Accuracy” and “Safety clearance”.

Accuracy is the measure of the refinement of volume. Less accurate volume object can be generated quickly since it has low resolution. Safety clearance is the measure of safety distance you want to select between two objects. It can be used if distance between the two resources less than a certain value is considered as a collision; in such case the generated volume is wider than the actual occupied space of robot.

Low accuracy and zero safety clearance will give us the rough and narrow volume. High accuracy value and increased safety clearance sometimes crashes the software with an error. Error occurs because this increases the resolution of volume and consecutive volume creation adds load to RAM and when this load increases to a certain limit then processor forces Process Simulate to stop.

Volume creation in Process Simulate using the provided tools is a time consuming and memory-intensive process. For this reason, we tried to automate it in such a way that less memory is consumed, perhaps it is not yet fully stable in case of high accuracy and increased safety clearance. However this GUI was developed and tested on a system that had virtual server that consumed half of the system’s resources, this problem may not arise in a computer systems with higher memory and processing power.



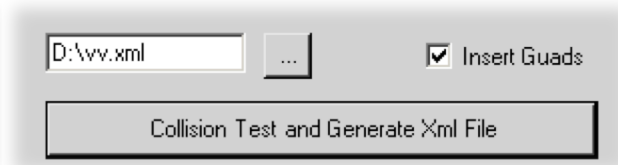
*Figure 20(a): Combination of volumes*

*Figure 20(b): Volume of one*

*of all operations of one robot*

*operation of same robot*

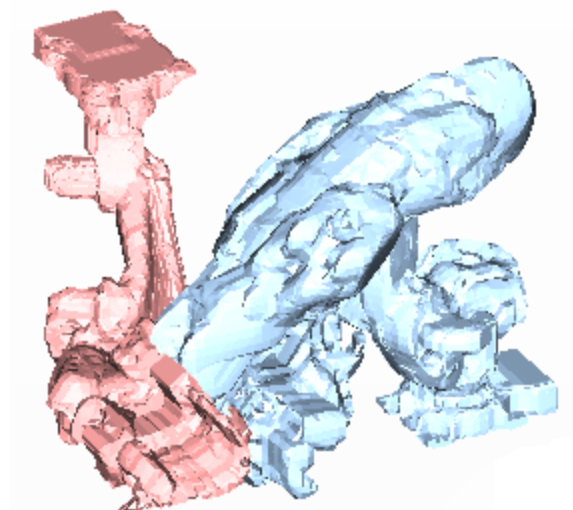
### ***Collision Test and Generation of XML File***



*Figure 21: Collision Test and Generation of XML File button in Collision Avoidance GUI*

The button shown in figure 21 is used to check the collision between resources on “Resource List”. Before pressing this button a location has to be selected where xml file is desired to be placed and if it is required to generate guards then the box “Insert Guards” should also be checked.

Initially the collision is detected between different resources by checking the minimum distance between their operations’ volumes. Figure 18 shows the volumes of two robots that are intersecting.



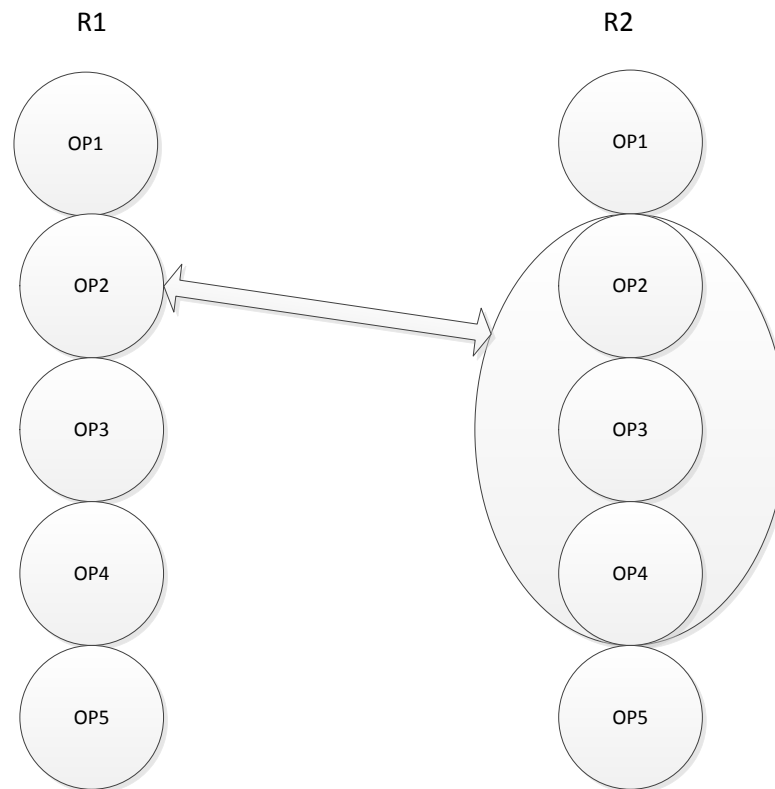
*Figure 18: intersecting volumes of two robots*

Each volume is associated with an operation of a resource and each volume may intersect more than one volume of the other resource. The information of each volume’s intersection is collected by repeating this step for all the volumes. This information is sufficient to know about mutual exclusion zones and involved resources.

The information of the sequence of operations and their intersection information for all the resources on “Resource List” are exported into XML format using the Sequence Planner Project schema file.

If the “Insert Guards” box is checked, then the guards are added based on the intersection information. Task is to insert a guard such that the operation in the colliding zone does not execute at the same time.

In this thesis guards are inserted in the form of OLP commands, along with the signals sent to the welding guns. A robot can send signal to a particular robot in the form of a variable name with a value and can also wait for a signal. These signals can act as logical conditions depending on which a robot may stop or perform an operation. By using send signal and wait signal, conditions can be inserted in the operations. Following example illustrates this mechanism.



*Figure 19: SOP of two resources R1 and R2 , volume of second operation of R1 in intersection with second, third and fourth operation volumes of R2*

Two resources R1 and R2 have five operations in sequence, shown in figure 19, just like in a welding station. Suppose that the second operation of first resource R1.OP2 is colliding with second R2.OP2, third R2.OP3 and fourth R2.OP4 operations that are consecutive.

Firstly, “insert guards program” goes to first resource R1; it checks each operation’s colliding information. Suppose that program arrives at R1.OP2, retrieve its colliding information and finds that it collides with three consecutive operations of R2 (R2.OP2, R2.OP3 and R2.OP4). So when R1 is executing R1.OP2 then R2 should not execute any of these operations or if R2 is in execution of R2.OP2 then R1 should not execute R1.OP2 until R2 finishes the operation R2.OP4. Figure 20 shows that how the guards will be inserted to deal with the mutual exclusion zone.

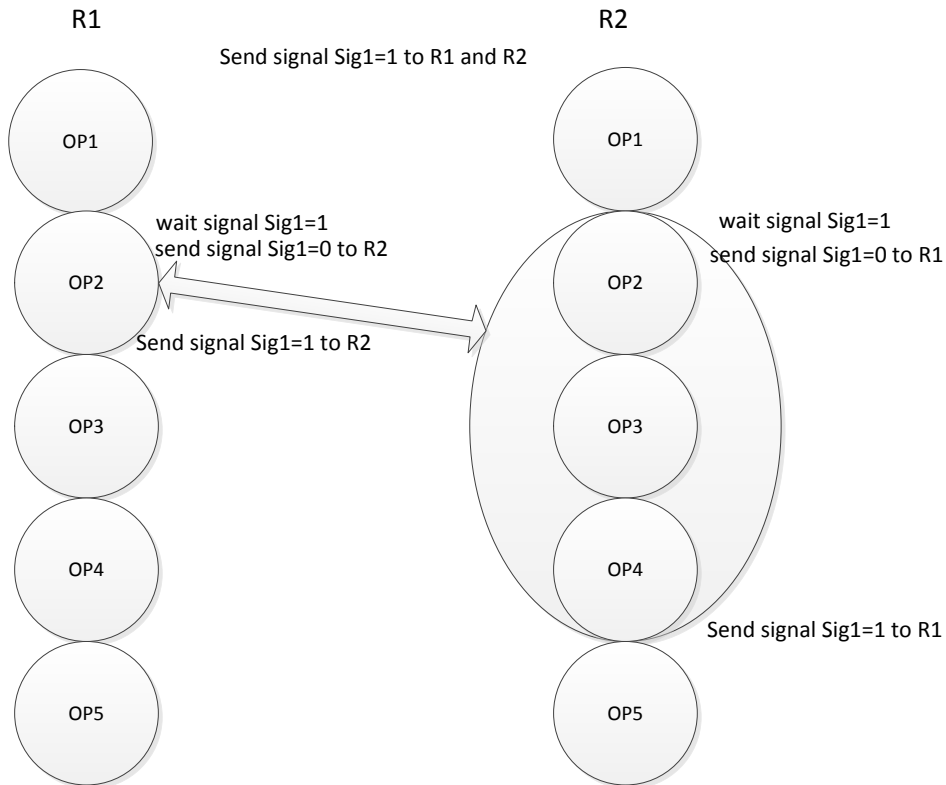


Figure 20: Guards on SOP of two resources R1 and R2, volume of second operation of R1 in intersection with second, third and fourth operation volumes of R2

The “Insert guards program” generates one signal with the name “Sig1” and value 1, and sends it to both robots in start of simulation. Now both resources R1 and R2 wait for this signal to have value “1” in start of operation 2. If R1 reaches first, it sends the signal to R2 with same name but changes the value of signal to “0”. Now for R2 “wait signal Sig1=1” command is not fulfilled so it stops until R1 sends back a signal to R2 “Sig1=1” at the end of its operation, and same happens in opposite scenario when R2 reach first.

Then “Insert guards program” does same with rest of operations of R1. The intersection check and insertion of guards is carried out in pairs. If the program checks and inserts guards for the intersection of R1 with R2, then checking of R2 with R1 is not necessary and is not done.

## CONTRIBUTION

This thesis work is unique in this regard that Process Simulate has currently no feature that automatically generates a collision free manufacturing cell, however there are functions that detects collisions and informs the designer about time instant and location of collision two such functions are discussed below.

### *Collision viewer*

The Collision Viewer is an essential tool for planning and optimizing the assembly process. Collision Viewer can be used to check the feasibility of the operations planned in the assembly process and to ensure that the process is collision-free. For example, when assembling an automobile body, one can use the Collision Viewer to answer questions such as:

- At what time and location two objects collided?
- Are all the moving parts within the safe distances that is there is no near miss?

User can use the Collision Viewer to display planned collision sets that are of particular interest and to hide others. For example, two robots are far apart from each other and there is no chance of their collision, so there is no need to put them in same set.

When running a simulation of user's proposed process, the Collision Viewer can indicate the collision curve of colliding objects. One can view the collision as a report or graphically in the Graphic Viewer. This enables user to make interactive corrections and to refine the process for optimal results.

This tool is not appropriate for the purpose presented in this thesis, since it only checks the simulation for collisions and near misses. It does not consider the collisions that may occur as a result of delay in any of the robots. There is no option available in this tool that can ensure that no robot will ever collide even if some delay or mechanical failure occurs in a robot.

### *Interference zone*

The Interference Zone is another tool in Process Simulate that checks for the zone common to a robotic operation and the Swept Volume previously created from another robotic program or operation.

The Interference Zone tool creates the Interference Zone by simulating a robotic program or operation and comparing it to a Swept Volume Object. Where the simulation results in a collision between the Swept Volume Object and the robotic program or operation, the tool creates Interference Zones. These can be used to detect and visualize

collisions that may occur during the execution of robotic programs or operations, when any object attached to the robot collides with the swept volume or any attached object which is visible.

This tool also gives the option of creation of notes for interference zones that mentions the time instant and operation point at which collision begins and where it ends.

Function of this tool is very close to the idea discussed in this thesis but, this tool only provides the information of the collisions. Designer has to visualize these collisions either graphically or through generated notes. To avoid these collisions designer has to manually make changes to the simulation or the robotic programs.

The developed tool in this thesis for collision avoidance is ideal for use in the design of complex manufacturing systems. It automatically detects the possible collisions and sets guards to the operations, ensuring that under any circumstances, collision will not occur. This tool can perform the division of robotic operations into smaller operations. Based on these operations sweep volumes of the resources can be generated. Intersection test of these volumes identify the collision regions. Guards are automatically applied to these regions such that no more than one resource can be in the intersecting region at the same time. External support of Sequence Planner and Supremica can ensure that system controller will not let it go into the blocking situation.

## CONCLUSION

### *Volvo Car Corporation*

Designed model of the weld station shown in *Figure 8* is a good option for any vehicle that requires weld spots around 800. It can be welded within two stations consuming time of 2 minutes, provided that levels of robots are increased from two to three. Same station can be used for new and different vehicles, if flexibility is introduced in positioning of robots for minor changes in two axis. To avoid collisions in the interlocking further investigation should be carried out that,

- How well robots react to the guards in OLP commands.
- How optimized paths for robots can be achieved automatically; it will reduce the cycle time.
- How to reduce the avoidable interlocking zones; it will reduce system complexity, risk and time.

### *Chalmers*

After successful automation in division of operations, volume generation, intersecting test and application of guards, simulation results are evident that proposed method for complex scenario performs well.

Implementation of methodology for collision avoidance required use of sequence planner that needs further development to correctly interpret the guards and blocking of the system. The developed channel between Process Simulate and Sequence Planner is one directional it should be investigated to make it bidirectional to create the controller in Process Simulate non-blocking.



## REFERENCES

- [1] M.R.Shoei, B.Lennartson and S.Miremadi. “Automatic Generation of Controllers for Collision-Free Flexible Manufacturing Systems,” 6th annual IEEE International Conference on Automation Science and Engineering, 2010, p. 368-373.
- [2] E. Ohlson and C. Torstensson, “Development, implementation and testing of Sequence Planner - A concept for modeling of automation systems,” Tech. Rep. EX/2009, Chalmers University of Technology, 2009.
- [3] M. Skoldstam, K. Akesson, and M. Fabian, “Modeling of discrete event systems using finite automata with variables,” in IEEE Conference on Decision and Control, 2007, pp. 3387–3392.
- [4] Supremica. <http://www.supremica.org>.
- [5] Siemens Product Lifecycle Management Software Inc. Tecnomatix.NET Manual, 2010.
- [6] Siemens Tecnomatix, 2010, <http://www.tecnomatix.com>.
- [7] Product Lifecycle Management Software Inc. eMServer Data Importing (via Process Designer) Student Guide, January 2008.
- [8] Product Lifecycle Management Software Inc. Process Simulate Basic Student Guide, January 2008.
- [9] C. A. R. Hoare, “Communicating Sequential Processes,” Communications of the ACM, vol. 21, 1985, pp. 666–677.
- [10] K. Akesson, M. Fabian, H. Flordal, and R. Malik, “Supremica – An integrated environment for verification, synthesis and simulation of discrete event systems,” in 2006 8th International Workshop on Discrete Event Systems. IEEE, 2006, pp. 384–385.