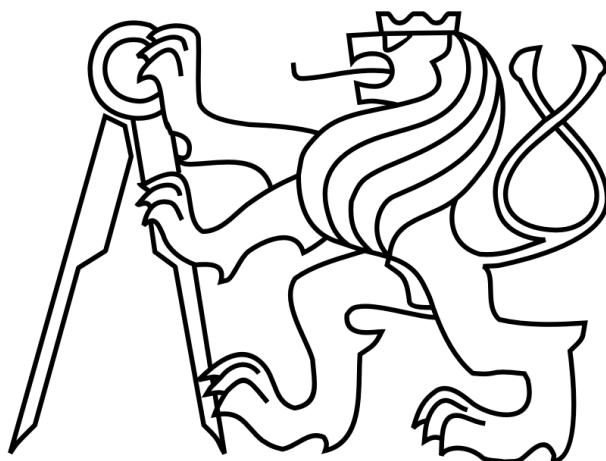


CZECH TECHNICAL UNIVERSITY IN PRAGUE
FACULTY OF ELECTRICAL ENGINEERING



DIPLOMA THESIS

Simulation of Production Processes

Prague, 2013

Author: Bc. Miroslav Konopa

Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Control Engineering

DIPLOMA THESIS ASSIGNMENT

Student: **Bc. Miroslav Konopa**

Study programme: Cybernetics and Robotics
Specialisation: Systems and Control

Title of Diploma Thesis: **Simulation of production processes**

Guidelines:

This master thesis focuses on the formal description of production processes and their simulation. A production line, inspired with real technology, will be designed, implemented in process simulation software and tested.

1. Do a research of existing techniques of the formal description of processes with focus on production processes.
2. Get acquainted with the process simulation software from Siemens. Focus on the possibility of the export and import to be able to connect the simulation software to external formal descriptions.
3. Design a production line inspired with real technology and implement it in the process simulation software. Part of the line will be composed of real devices such as conveyors or robots.
4. Do the integration of the entire production line and perform tests of its functionality. Whenever possible, verify also the implemented simulation against the formal description.

Bibliography/Sources:

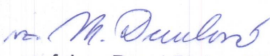
Siemens. Process Simulate operating manual.

Diploma Thesis Supervisor: Ing. Pavel Burget, Ph.D.

Valid until the summer semester 2012/2013


prof. Ing. Michael Šebek, DrSc.
Head of Department




prof. Ing. Pavel Ripka, CSc.
Dean

Prague, February 13, 2012

Declaration

I hereby confirm that I wrote this diploma thesis on my own and that I listed all used materials in references.

Prohlášení

Prohlašuji, že jsem svou diplomovou práci vypracoval samostatně a použil pouze podklady (literaturu, projekty, SW atd.) uvedené v příloženém seznamu.

Prague, 2nd January 2013

Miroslav Konopa

Poděkování

Tímto bych chtěl poděkovat svým rodičům a rodině, bez jejichž lásky, trpělivosti a podpory by tato diplomová práce nemohla vzniknout. Dále bych chtěl poděkovat především svému vedoucímu diplomové práce, panu Ing. Pavlovi Burgetovi, Ph.D. za jeho odborné vedení, připomínky a poznámky v průběhu řešení diplomové práce. Rád bych také poděkoval i všem ostatním, kteří mi poskytli cenné rady a pomoc ve chvílích, kdy jsem jí potřeboval nejvíce. Děkuji Vám všem.

Acknowledgment

I would like to thank to my parents and family. Without their love, patience and support this thesis could not be done. Furthermore, I would like to thank to my supervisor Mr. Ing. Pavel Burget, Ph.D. for his professional guidance, suggestions and hints during the course of this work. I would also like to thank to all others without whose help it would be difficult for me to create this thesis. I thank you all.

Abstrakt

Cílem této diplomové práce byl návrh výrobní linky, formální popis jejích výrobních procesů a simulace. Část výrobní linky je složena ze skutečných zařízení.

Tato výrobní linka byla navrhována se třemi robotickými pracovními buňkami jako svařovací linka na auta. Simulace byla vytvořena v simulačním nástroji Process Designer a Process Simulate v režimu virtuálního uvedení do provozu. Externí formální popis byl vytvořen v sekvenčním funkčním grafu (SFC) v nástroji Totally Integrated Automation portal pro programovatelný logický automat (PLC). Tento popis byl propojen do simulace přes OPC server. Simulace také obsahovala skutečný robot KUKA KR5 arc, který byl přes průmyslovou síť PROFINET připojen do PLC a následně i do simulace.

Abstract

This master thesis focuses on the design of product line, its formal description of processes and simulations. Part of the line is composed of real devices.

This product line was designed as the car weld line with three robot work cells. Simulation was created in Process Designer and Process Simulate in Virtual Commissioning mode. External formal description was created in Sequential Functional Chart (SFC) in Totally Integrated Automation portal for Programmable Logic Controller (PLC). This description was connected into simulation via OPC server. Simulation also contained a real robot KUKA KR5 arc, which was connected to PLC via industrial network PROFINET and subsequently into simulation.

Contents

1	Introduction	1
1.1	Outline	1
1.2	Objectives	2
2	Specification of the Modeled System	3
2.1	Informal description	3
2.2	Semiformal description	3
2.3	Formal description	4
2.4	Implementation of the Modeled System	5
2.4.1	Informal description	5
2.4.1.1	Workspace 1	5
2.4.1.2	Workspace 2	6
2.4.1.3	Workspace 3	7
2.4.2	Formal description	7
2.4.2.1	Robotic work cells	7
2.4.2.2	Car flow	7
2.4.2.3	Safety	9
3	Tecnomatix	10
3.1	Overview	10
3.2	Architecture and installation	13
3.2.1	Architecture	13
3.2.2	Installation	15
3.3	Process Designer	15
3.3.1	Models	16
3.3.2	PERT diagrams	17
3.3.3	In-Process Assembly	18

3.3.4	Implementation of a project	18
3.3.4.1	Creating of a new project	18
3.3.4.2	Reservation of a project	18
3.3.4.3	Adding Collections	19
3.3.4.4	Setting of a Working Folder	19
3.3.4.5	Adding of a Study Folder	20
3.3.4.6	Creating of 3D libraries	20
3.3.4.7	Design of a factory concept - resources and operations .	21
3.3.4.8	Design of a factory concept - parts and compound parts	22
3.3.4.9	Creating of a new study	23
3.3.4.10	Adding of resources to Resource tree	23
3.3.4.11	Adding of operations to Operation tree	24
3.3.4.12	Adding of parts to Product tree	24
3.3.4.13	Creating of a layout	24
3.4	Process Simulate	26
3.4.1	Modeling	27
3.4.2	Kinematics	27
3.4.2.1	Kinematics Editor	27
3.4.2.2	Tool Definition	28
3.4.2.3	Reach Test	28
3.4.2.4	Smart Place	29
3.4.3	Operations	29
3.4.3.1	Compound Operation	30
3.4.3.2	Object Flow Operation	30
3.4.3.3	Device Operation	30
3.4.3.4	Device Control Group Operation	30
3.4.3.5	Gripper Operation	30
3.4.3.6	Pick and Place Operation	30
3.4.3.7	Weld Operation	30
3.4.3.8	Continuous Operation	31
3.4.3.9	Non-Sim Operation	31
3.4.3.10	Robot Path Reference Operation	31
3.4.3.11	Swept Volume	31
3.4.3.12	Interference Volume	31
3.4.4	Welding	32

3.4.4.1	Create Weld Points	32
3.4.4.2	Pie Chart	32
3.4.5	Robotics	33
3.4.5.1	Robot Controller	33
3.4.5.2	Robot Program	34
4	Simulations	37
4.1	Time-based Simulation	38
4.1.1	Implementation of Time-based simulation	38
4.1.1.1	Adding kinematics to a gripper	38
4.1.1.2	Material Flow in Workspaces	38
4.1.1.3	Welding Robotic Operations	39
4.1.1.4	Gantt Chart	44
4.2	Event-based Simulation	44
4.2.1	CEE Simulation	45
4.2.2	Virtual Commissioning	45
4.2.2.1	Appearances	46
4.2.2.2	Signals	46
4.2.2.3	Sensors	48
4.2.2.4	Logic blocks	49
4.2.2.5	OLP programing	50
4.2.2.6	Connection to PLC	51
4.2.3	Implementation of Virtual Commissioning	53
4.2.3.1	OPC settings	53
4.2.3.2	Generation of appearances	54
4.2.3.3	Adding of sensors	55
4.2.3.4	Overview of signals	55
4.2.3.5	Robot simulation	58
4.2.3.6	SFC program	58
5	KUKA robot	60
5.1	KUKA KR5 arc	60
5.2	Description	60
5.2.1	Robotic assembly	61
5.2.2	Technical data	61

5.2.3	Axis data	62
5.2.4	Robotic Controller KRC4	62
5.3	Configuration of the robot	63
5.3.1	KCB configuration	64
5.3.2	KLI configuration - PROFINET	64
5.3.3	Mapping of variables for Automatic External mode	66
5.4	Commissioning of robot	66
6	Conclusion	69
	Bibliography	73
A	Content of the Attached CD	I
B	Formal Description - Tables	II

List of Figures

2.1	Basic visual description of the modeled system	5
2.2	SFC Description	8
3.1	Portfolio of a digital factory	11
3.2	Tecnomatix groups according to use	12
3.3	Tecnomatix architecture	14
3.4	Tecnomatix data structure	14
3.5	Base stones of Process Designer and Process Simulate	16
3.6	Project, Collections, Study folder, Working folder, Libraries	19
3.7	Navigation Tree - Resource, Part and Manufacturing Features library . .	21
3.8	Navigation Tree - Factory design (resources and operations)	22
3.9	Navigation Tree - Factory design (products)	23
3.10	Navigation Tree - New study	24
3.11	Workspaces	25
3.12	Workspaces	25
3.13	Entire product line - top view	25
3.14	Process Simulate - Navigation, Object, and Operation Tree	26
3.15	Process Simulate - Kinematic Editor	28
3.16	Process Simulate - Reach Test	29
3.17	Process Simulate - Smart Place	29
3.18	Process Simulate - Swept Volume	31
3.19	Process Simulate - Pie Chart	32
3.20	Process Simulate - Robot Controller Simulation (RCS) module	34
3.21	Process Simulate - Robot Program Inventory	35
3.22	Process Simulate - Download Program	36
4.1	Process Simulate - Kinematics of a gripper	39
4.2	Sections of a Welding Line - Car flow	40

4.3	Sections of a Welding Line - Plate flow	40
4.4	Designed weld points	41
4.5	Welding operations of a rob cell of Workspace 1	42
4.6	Welding operations of a rob cell of Workspace 2	43
4.7	Welding operations of a rob cell of Workspace 3	43
4.8	Gantt chart of time-based simulation	44
4.9	Signal Viewer - Example	47
4.10	Logick Block - Example	49
4.11	Robot signals	50
4.12	OLP signals - Example of using	51
4.13	Settings of CEE simulation/OPC simulation	52
4.14	Virtual Commissioning - Connection to PLC	52
4.15	Creating of S7 connection in TIA portal	54
4.16	Selecting of a reachable OPC server in Process Simulate	54
4.17	Defined material flow	55
4.18	Designed photoelectric sensors	56
4.19	Whole product line	59
4.20	Whole product line	59
4.21	Whole product line	59
5.1	Configuration of KUKA Kr5 arc	60
5.2	Main assemblies of the robot [18]	61
5.3	Hardware configuration of the KUKA Robotic Controller 4 (KRC4) . . .	63
5.4	WorkVisual - Kuka Controller Bus settings	64
5.5	WorkVisual - Settings of PROFINET	65
5.6	WorkVisual - Mapping of KUKA Controller signals into PROFINET network	65
5.7	Downloaded robotic program to the KRC4 controller	67
5.8	Commissioning of the real robot	68

List of Tables

4.1	Description of welding operation of Workspace 1	41
4.2	Description of welding operation of Workspace 2	42
4.3	Description of welding operation of Workspace 3	42
4.4	Signals - Central control of light stacks	56
4.5	Signals - Light sensors	57
4.6	Signals - Car flow signals	57
4.7	Signals - PLC communication with robot controllers	57
4.8	Robot programs	58
5.1	Technical data of the robot KUKA KR5 arc	61
5.2	Axis data of the robot KUKA KR5 arc	62
B.1	Description of steps and transitions - Start	II
B.2	Description of a robot cell in Workspace 1	III
B.3	Description of a robot cell in Workspace 2	IV
B.4	Description of a robot cell in Workspace 3	V
B.5	Description of a car flow to Workspace 1	V
B.6	Description of a car flow to Workspace 2	VI

Chapter 1

Introduction

Simulations of production processes are getting widely extended due to possibilities of computer technology in last decades. Simulations are used in optimization, experiments, visualizations, development of factories. They save time and money of manufacturers, protects property and health of employees (ergonomic simulations).

In this thesis I specialized in virtual commissioning. This simulation enables to create digital factory before its real commissioning. User defines production processes based on resources, operations, products and also signals contained in the real factory. This approach enables to optimize factory and design its whole environment (production processes, material flows, robot programs, signal definitions) in the development phase before starting the testing phase with real hardware. Above that Virtual Commissioning creates connection between virtual reality and real environment via Programmable Logic Controller. Therefore simulation can contain also real devices, robots and signals.

1.1 Outline

I divided this work into five thematic parts. In the first chapter I carry out objectives of my work. In the second chapter I explain general distribution of formal methods at first. Then I analyze implemented informal and formal descriptions of the modeled system (welding line).

Chapter 3 describes basic concepts of the Tecnomatix product, its architecture and process of installation. Further, basic procedures how to use Process Designer and Process Simulate are explained. Among them there is creation of a new project, new study, layout, etc. for Process Designer, and a short description of general tools in Process Simulate,

which are used later in this thesis.

In Chapter 4 there are depicted time-based and virtual-based simulations. These simulations are based on the specification of the welding line from Chapter 2.

Chapter 5 describes real robot KUKA KR5 arc and concentrated especially on this configuration and commissioning with the simulation. Its specification, configuration and commissioning with simulation.

Chapter 6 concludes this thesis.

1.2 Objectives

This section describes the main objectives of this thesis briefly:

- Design a concept of the digital factory and its utilization.
- Get acquainted with the process simulation software from Siemens called Tecnomatix.
- Model a 3D simulation of production processes in simulation software. This simulation will be adapted to be used with external controller.
- Find a way how to connect simulation to an external controller performing the control of the welding line, which implements the algorithm as specified formally in Chapter 2.
- Select a suitable method to define the external PLC program of the simulated welding line.
- Get acquainted with KUKA KR5 arc robot. It is necessary to have user experience with its programming and control.
- Integrate the whole product line with real robot, namely KUKA KR5 arc and its commissioning.

Chapter 2

Specification of the Modeled System

This chapter contains description of informal, semiformal and formal methods. Further there is also implementation of the modeled system.

2.1 Informal description

It uses a nature language or pictures and tools which enable understand to described system. It exploits an unstructured description and a structured description. The unstructured description contains free text, so it is not necessary to have any special knowledge of the methodology. The structured description contains text tables. This specification has some disadvantages such as lack of clarity and impossibility of automatic analysis and processing. On the other hand it is possible to create it without any special tools.

2.2 Semiformal description

The semiformal description is a description of a modeled system, which has the character of a precise syntax and free semantics. Graphical representation is well-arranged in comparison with free text. It has possibility of automatic analysis and processing. Semiformal methods are, for example: Event-driven Process Chains(EPC), Unified Modeling Language(UML). See [1] or [2].

2.3 Formal description

Formal description is a part of wider concept of formalization called Formal methods, which consist of:

- Formal description
- Formal development and verification
- Formal analysis

The formal description is a description of a modeled system. It exploits accurate mathematical operations and expressions and also it has a character of a precise syntax and semantics. This technique enables to specify the modeled system conclusively. This concept can be a pattern for creation of a real system.

After creation of a formal description this concept can be used for developing a unique real modeled system. Consequently, this developed modeled system can be verified against formal description. Based on a formal description we can also proof some properties of description.

We use formal analysis to analyze formal description against its logical concept trying to proof a correctness of the described system. One of the critical fields of this concept is, for example, verifying the verifiers.

In this work I concentrate especially on a formal description. There are many of those existing techniques, see [3]. I deal with Sequential Function Chart (SFC).

The Sequential Function Chart is the general graphical programming language used for programmable logic controllers. It was based on the GRAFCET which is based on the binary Petri nets. The SFC logic consists of :

- Steps
- Transition conditions
- Links between steps and transitions and also conversely

In steps the actions are defined. The possibility of progress from one step to other steps through links is controlled by transitions (their conditions). The S7-Graph, made by Siemens, is a program that allows us the SFC graphically and it is contained in the Totally Integrated Automation (TIA) Portal in the Step7 (an engineering tool). The logic itself is programmed using a subset of the Ladder Logic (LAD) or the Function Block Diagram (FBD).

2.4 Implementation of the Modeled System

I designed a product line of cars with three welding workspaces. Firstly I defined informal description of the modeled system. After I created formal description of the modeled system in Sequential Function Chart. To imagine the defined product line I described it basically in figure 2.1.

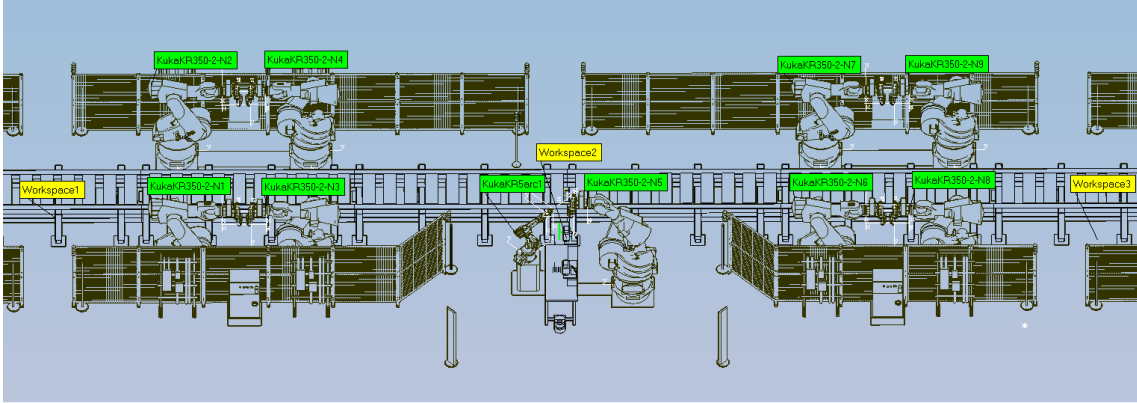


Figure 2.1: Basic visual description of the modeled system

2.4.1 Informal description

This subsection contains a verbal description of the system, which has been modeled in this thesis. It is a welding line for a car body, consisting of three workspaces connected with a conveyor, which transports the car.

In this section there is a verbal description of my modeled system.

2.4.1.1 Workspace 1

- Robots R1, R2, R3, R4 wait for a new car in Workspace 1. The orange light on the light stacks is on. Robots are READY.
- The conveyor runs and in time when a car has come to Workspace 1 light stacks light green at Workspace 1 - robots start to RUN and at the same moment the conveyor stops moving.
- Robots R1, R2, R3, R4 move out from the HOME position and it continues to weld spot locations on a skeleton of a car with the OPEN weld guns. After follow

welding operations of robots on a car, exactly on the back glass frame and the front glass frame, which runs simultaneously.

- After ending robotic operations the robots move back to the HOME position. Light stacks light orange which means that robots are READY for new operations.
- The conveyor moves a skeleton of a car on frame to Workspace 2.
- If anyone goes to Workspace 1 through light curtains in Workspace 1 the robot cell stops running. The light stacks light red and the robots are in the FAULT state and waiting for the end of the FAULT state.
- After ending of the FAULT state light stacks light green and robots are in the RUN state.
- If a human operator press the button to stop sending cars on frame to Workspace 1 and the conveyor stops to accept a new car on frame into Workspace 1.

2.4.1.2 Workspace 2

- Robots R5 and KUKA KR5 arc wait for a new car in Workspace 2. Light stacks light orange in Workspace 2 and robots are in the READY state.
- Robots start working and light stacks light green. Robots are in the RUN state.
- KUKA KR5 arc picks a part (a plate) from the table and place it into the marker.
- The marker signs the plate which KUKA KR5 arc holds in the marker.
- KUKA KR5 arc takes the plate from the marker and place it on a skeleton of a car. After it KUKA KR5 arc moves to the HOME position.
- Robot R5 moves to the placed plate and welds it. After it R5 moves to the HOME position.
- Light stacks light orange and robots are in the READY state waiting for new operations.
- The conveyor moves a skeleton of a car with the plate (assembled parts) on the frame to Workspace 3.

- If anyone goes to Workspace 2 through light curtains or the range of the light scanner in Workspace 2 robot cell stops to run. Light stack light red and robots are in the FAULT state waiting for ending of the FAULT state.
- After ending of the FAULT state light stacks light green and robots are in the RUN state.

2.4.1.3 Workspace 3

Workspace 3 is analogical to Workspace 1. There is only one difference that robots R6, R7, R8, R9 weld a roof of a car skeleton and the conveyor moves a skeleton of a car with a plate(assembled parts) on frame out of Workspace 3.

2.4.2 Formal description

In this section there is a description of the modeled system according to Sequential Function Chart (SFC), see figure 2.2 . Sequential Function Chart was made in the Totally Integrated Automation (TIA) portal with signals defined later in this work. Logic is defined in steps (S) by setting and resetting required values. Transitions (T) are represented by the ladder logic, such as AND, OR, SET, RESET etc. We can split this algorithm into two logical parts. The first one contains robotic work cells. The second one contains car flows. At the start and during the algorithm there are general steps which are described in table B.1. (Tables are located in Appendix B) .

2.4.2.1 Robotic work cells

After transition T2 algorithm branches into four branches. Their represent robotic cell of Workspace 1,2 and 3. Description of branches for Workspace 1 and 2 are in table B.2 and B.3. Branch of Workspace 3 is analogical to branch of Workspace 1.

2.4.2.2 Car flow

After transition T21 algorithm branches into 4 section for a car flow to Workspace 1, 2, 3 and out of Workspace 3. Description for a car flow to Workspace 1 and 2 is visible in table B.5 and B.6. Branches of Workspace 3 and 4 are analogical to Workspace 2.

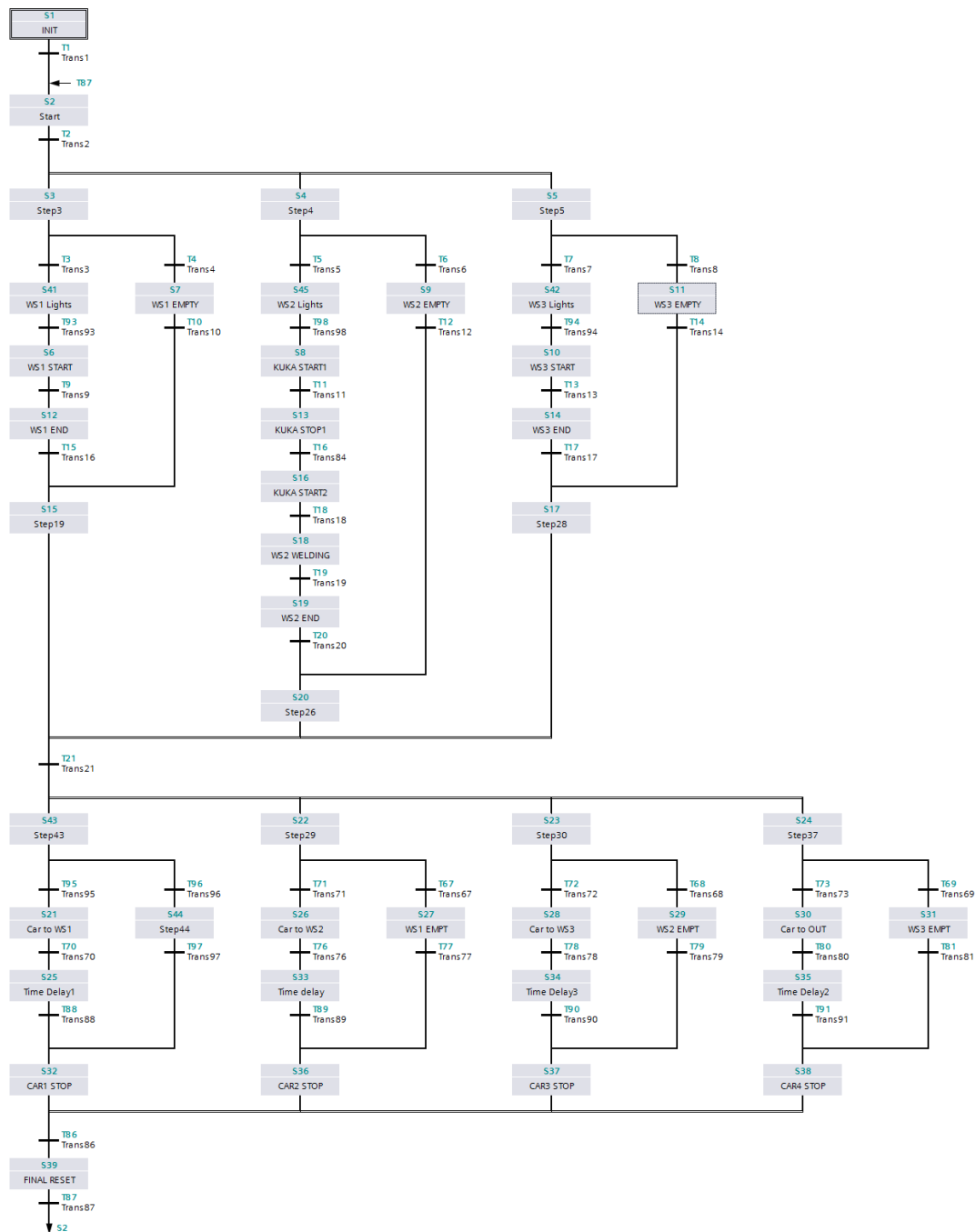


Figure 2.2: SFC Description

2.4.2.3 Safety

Safety description which was defined in subsection 2.4.1 was created separately to algorithm defined in figure 2.2. When a safety event occurs in Workspace 1, 2 or 3. Robot cell stops to work and light stacks are in the FAULT state (red color). After safety event ends robots continue to work and light stacks are in the RUN state (green color).

Chapter 3

Tecnomatix

This chapter briefly describes basic terms relating to the digital factory environment, which is called Tecnomatix.

3.1 Overview

Tecnomatix is a comprehensive portfolio of digital manufacturing solutions that deliver innovations by linking all manufacturing disciplines together with product engineering – from process layout and design, process simulation and validation, to manufacturing execution. Tecnomatix is built upon the open Product Lifecycle Management (PLM) [4].

The Product Lifecycle Management provides access to product and process knowledge in the frame of the whole life cycle of a product (conception, design, manufacture, transportation, utilization, disposal, recycling). The PLM is originally based on Computer-aided Design (CAD), Computer-aided Manufacturing (CAM) and Product Data Management (PDM).

Tecnomatix is a part of the Siemens PLM Platform (see figure 3.1) and it is categorized into groups such as:

- Part Planning and Validation - Part Manufacturing Planner, Machining Line Planner, Press Line Simulation, Virtual Machine Tool
- Assembly Planning and Validation - it is exploited in Process Designer, Process Planner, Process Simulate Assembly, Process Simulate Human, Jack (Control of assembly processes, ergonomics, etc.)



Figure 3.1: Portfolio of a digital factory

- Robotics and Automation Planning - it is exploited in Process Designer, Process Simulate Robotics, Robcad, Process Simulate Spot Weld (Robotic production process, etc.)
- Plant Design and Optimization - it is used in Factory Cad, Factory Flow, Plant Simulation (Optimization of production processes, etc.).
- Quality and Production Management - Dimensional Planning and Validation (DPV), Variation Analysis (VSA), CMM Inspection, Manufacturing Execution Systems (MES), HIM/SCADA

Detail explanation of those groups is possible to find, for example, in [5] or [6].

Further categorization according to use is mentioned in the lecture of Mr. Carvan [5]. The first category surfaces data link and rough planning via Process Designer or TCM Process Planner. The second category surfaces simulation and detailed planning via Process Simulate Assembly, Process Simulate Robotics or Robcad, Process Simulate Human or Jack. The third category surfaces design and optimization via Factory CAD, Factory FLOW and Plant Simulation. All groups and elements are connected, see 3.2

Software Factory CAD is a superstructure of AutoCad. We can use it to a quick

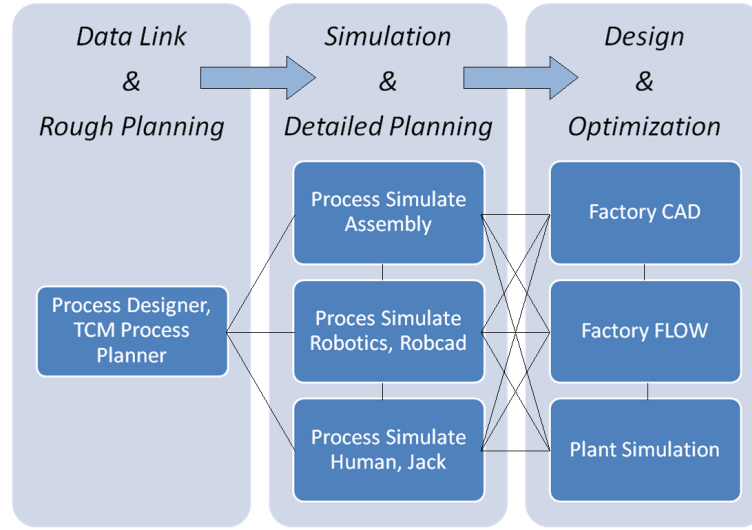


Figure 3.2: Tecnomatix groups according to use

modeling of a 3D layout of production and to project production halls, workstations etc. Factory FLOW is also a superstructure of AutoCad. We use it to representation of a material flow, graphical and numerical cost of transportation of individual variants etc. Plant Simulation is suitable for optimization of industrial factory processes in 2D layout. It contains dynamical simulation, exploiting of people and machines, identification of thin places in workstation, transport of supply, verification of strategies, occupancy of stores, optimization of production processes, etc.

Software Jack includes primarily a biomechanical model of a human being. It serves to higher efficiency and improvement of ergonomics in workplace, its analyzing, improvement/elimination of occupational diseases. It is possible to use special electronic gloves for transferring of human movements.

Software Process Designer serves as a rough planner for operations, resources and products used in 3D simulation, balancing of product lines, etc. On the other hand Process Simulate serves as a detail planner for accurate analysis, simulation of operations, collision planning, time analysis, ergonomic analysis, Offline Programming of Robots (OLP), Virtual Commissioning (VC), etc.

In this work I concentrate especially on Process Designer and Process Simulate. Both platforms are widely described here in section 3.3 and 3.4 respectively.

One of the main reasons of a virtualization is the reduction of costs and improvement, higher quality and faster Product Lifecycle Management process. It is also related to rule 1:10:100, which generally says that 1 euro spent on prevention will save 10 euro on

correction and 100 euro on failure costs. The cost of failure will increase if the error is discovered in later phases of development or even after manufacture [7]. We can either model solution which exists or model new product lines, halls, etc. Then we can optimize the solution without necessity of real devices, technologies and create variants to exploit our real technology maximally. Therefore, it is not necessary to stop the product line for a long time to build-up a new solution because we have already prepared everything in the PLM SW solution.

3.2 Architecture and installation

3.2.1 Architecture

The basic Tecnomatix configuration consists of three tiers: Oracle Database (tier 1), eMServer (tier 2), and client (tier 3), see figure 3.3.

According to [8] Oracle Database manages data and controls access for different users. The Oracle database server runs a single Oracle instance. The instance contains the processes and memory for the database. A single instance can contain multiple schemas, but eMServer can work only with one schema at a time. The eMServer is an Oracle client which manages conne.

The eMServer manages connection between Oracle database and client (application). The eMServer provides services and modeling of elements according to the rules and logic of the manufacturing process. Technologically, the eMServer is an application server hosted by Microsoft's COM+ technology. In addition, eMServer requires access to a file server (Systemroot), for non-database files such as documents, 3D components, and engineering files used in Tecnomatix applications [8]. For example, if we want to add models to Process Designer or Process Simulate, 3D models have to be placed in System root.

In the third tier there are all applications(clients) that use eMServer and utilize Application Programming Interface (API), such as Process Designer, Process Simulate, Plant Simulation and so on.

In Tecnomatix, elements are described uniquely according to identification ID instead of name. That is why we can have more objects in Process Simulate with identical names, such as Resources or Products, because their identification ID is different.

It is also really important to understand data structure in Tecnomatix environment.

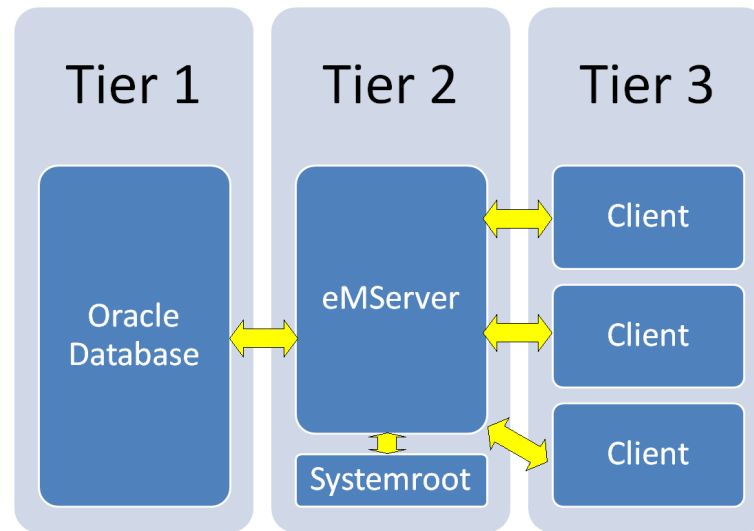


Figure 3.3: Tecnomatix architecture

According to [9] Oracle Database is divided into schemes. These consist of projects which are each made up of objects/nodes structured in trees. These nodes are products, operations, resources and manufacturing features that together define the manufacturing process. A tree can only contain a group of the same nodes, e.g. resources. Finally, attributes can be attached to the nodes, e.g. files with 3D data or AutoCAD files, see figure 3.4.

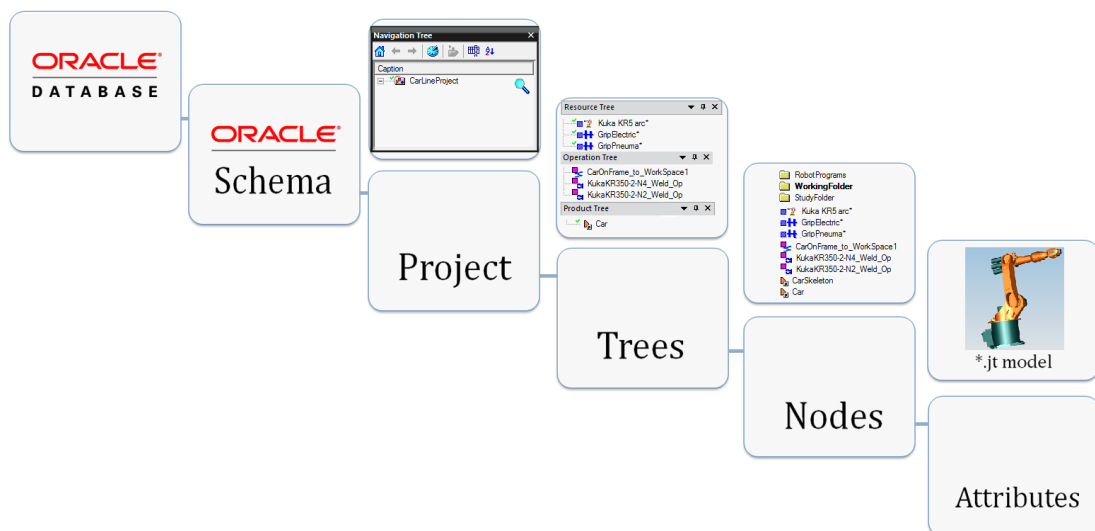


Figure 3.4: Tecnomatix data structure

3.2.2 Installation

Possibilities of installation are due to the 3 tier architecture relatively complex. I chose a standalone(client/server) installation which means that all 3 tiers of the Tecnomatix architecture are installed on one machine. I did it in following steps:

- Installation of Oracle Database 11g Release 2 (11.2.0.1.0) for Windows 7(x64).
- Creation of Oracle instance and configuration of it for the eMServer. It is possible manually or by using of special PERL's scripts. Oracle database does not support underscore conventions in operating system Windows. It can be very time-consuming to recognize this problem.
- Installation of eMServer and Client Application of Tecnomatix 10.1 (x64).
- Selection of an eMServer System Root.
- Creation of Oracle schema for eMServer.
- Connection of the Oracle schema into the Oracle instance.
- Installation of Hotfixes.
- Installation of Tecnomatix License Server and registration the license of Tecnomatix product.

If there is a problem during the installation of Oracle Database Server we can uninstall it using a special tool provided by the manufacturer. The tool is a common part of Oracle Database, but we can download it also as a separate tool. This tool runs in command line and we have to deconfigure the Oracle Database instance with their listeners, which we have already configured before.

3.3 Process Designer

We use it Process Designer to rough planning of production process. It allocates resources, products and operations of our product line.

The Process Designer also provides an integration of the eMserver and its planning tools with the 3D environment. Main components of Process Designer are Navigation Tree, which includes all the data of the Product Operation Tree, Object Tree, Manufacturing

Features (i.e. weld points) and Resource Trees. This data contains main build stones of Process designer(Process Simulate), which are resources, parts and operations, see figure 3.5. It is important to memorize the colors of elements, because they are presented everywhere in Process Designer and Process Simulate. Resources, products and processes are blue, orange and pink respectively.

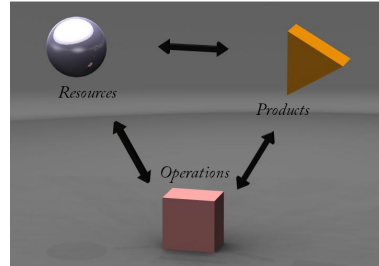


Figure 3.5: Base stones of Process Designer and Process Simulate

Certain functionality of Process Designer and Process Simulate is common for both of them. Project created in Process Designer is then transferable to Process Simulate and conversely.

Process Designer and Process Simulate are object oriented graphical environments.

3.3.1 Models

A default 3D model used by Tecnomatix is *.jt format. We can view these models without using of Tecnomatix in a freeware program JT2go viewer. We can create models in any reachable tool, such as Catia, NX, etc. The final model has to be converted after modeling in converter. The converter requires the license of the modeling program and of Tecnomatix, which means that you can not convert models from Catia, AutoCAD, etc. without having licenses of this programs. There is only one exception: Tecnomatix itself contains a converter from older *.co format to newer *.cojt.

Basic kinematic 3D models of robots are downloadable on the Global Technical Access Center (GTAC) of Siemens. Tecnomatix itself does not contain any 3D models and it is necessary to create it. To get access to the JT-files it has to be placed in either a *.cojt or *.co folder and placed in the system root.

3.3.2 PERT diagrams

The Program (Project) Evaluation and Review Technique (PERT) is a method to analyze the involved tasks in completing a given project, especially the time needed to complete each task, and to identify the minimum time needed to complete the total project. It works with optimistic, realistic and pessimistic estimation of every process. It was developed for the U.S. Navy Special Projects Office in 1957 to support the U.S. Navy's Polaris nuclear submarine project, more in [10]. It exploits a theory of probability to calculate duration of project and its dispersion. Based on this techniques it is possible to estimate processes which are critical from viewpoint of the given project.

The PERT viewer in Process Designer provides a graphical environment to examine logic diagrams of operations and dependencies, it includes resources, products, processes, manufacturing features and links between them. Main components are:

- Operation Box (operation) - name and type of operation, duration, resources, manufacturing features, etc.
- Source - it is a source of part that is delivered (produced) by a source
- Sink - it is a sink of part that is delivered (consumed) to a source
- Interface - it is a gateway of parts into/out of compound operations(set of operations)

Basically every operation can have some resources, which we assign to Operation box of a given operation. Between two operations (Operation boxes) or operation and Source/Sink, there can be a Flow if an Operation, Source or Sink precede to another one. The Flow can be a wearer of a product produced/consumed on output/input side of operation respectively. If there is no operation which produces a product for the following operation, which requires this product, we can add Sources element to bring it to the chart. Analogically, if we have an operation which produces a product and we do not use/want following operation which consumes a product in chart, we can add Sink elements.

PERT Diagram is connected with Gantt diagram in Sequence editor of Process Simulate and Material Flow Viewer of Process Simulate. If Iwe made a change in Process Simulate the change appears also in PERT diagram in Process Designer.

3.3.3 In-Process Assembly

According to [11] the overall workflow for using the IPA technology consists of creating an assembly tree and loading a related process. An IPA is a collection of assembled parts that all together make up a compound of parts. For example, if a car is an IPA, a door and a window can be parts in collection of IPA. When the IPA is created and contains the wanted parts, its content can be seen in Process Designer under the IPA Viewer.

In my thesis I did not work with In-Process Assembly parts but it plays an important role in Process Designer. To read more information about IPA I recommend following master thesis [12].

3.3.4 Implementation of a project

As I mentioned before Process Designer is a rough planning tool and Process Simulate is a detailed planning tool, which are both interconnected. That is why some settings made in Process Simulate appear also in Process Designer settings and conversely. To create and develop a project in Process Designer, the steps are in following subsections.

3.3.4.1 Creating of a new project

After opening the Process Designer a new empty project is created. I named it CarLineProject. The file of the CarLineProject is viewed in Navigation Tree in a tree structure as shown in figure 3.6.

3.3.4.2 Reservation of a project

To be able to work with the CarLineProject, it is necessary to Check Out a given project. Process Designer/Simulate is a client of eMServer that enables more people to work on one project. To limit a simultaneous work on the same parts of a project the following approach has been used:

- Check Out - reservation of a selected part of a project
- Check In - canceling reservation and saving a selected part of a project. It is possible to save it like a same version or create a new version (or a new branch) of a project.
- Cancel Check Out -canceling reservation without saving a selected part of a project. Selected part will be back to the previous Check In version.

If the project is checked out, its files should be marked with green mark ✓ and we can start to work with it, see figure 3.6. I worked on one project alone so I checked out the whole project with its hierarchy. To save changes to eMServer without Check In functionality we can use "soft" saving. It is direct upload of changes to eMServer. If a user decides to take those saving changes back he can use a Cancel Check Out functionality.

3.3.4.3 Adding Collections

Collections are basically file folders and they are a type of nodes in Process Designer/Simulate, as all elements in tree structures in Tecnomatix. If a project is checked out we can add them in a Navigation Tree under CarLineProject. I named my Collections, such as Libraries, Factory, RobotPrograms and WorkingFolder, see figure 3.6. Collections create basic structure of the project in the Navigation Tree. Special type of file node is a Study Folder.

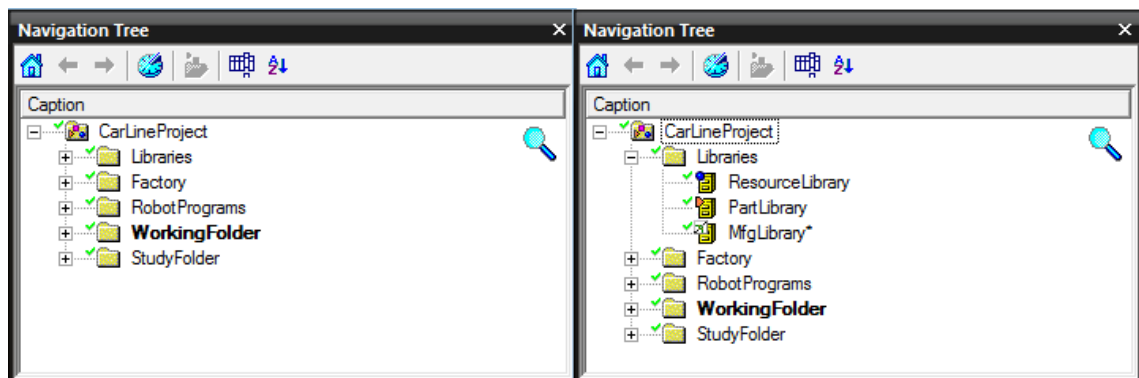


Figure 3.6: Project, Collections, Study folder, Working folder, Libraries

3.3.4.4 Setting of a Working Folder

Working folder has a special function. It is a stock of all nodes, objects, etc. which were not included in operations, products and processes defined in a hierarchy (structure) of Process Designer. There is a possibility to choose which collection will be the Working folder. On the other hand if it is not defined it is going to be generated automatically (according to user name $\langle USER \rangle$ Folder). It appears in Navigation Tree. The Working folder is marked by bold font, see figure 3.6.

3.3.4.5 Adding of a Study Folder

To work with Product, Resource tree and Operation tree we have to use studies. To use Process Simulate I worked with Robcad Study.

Study Folder is a special file node which is reserved only for studies. Under this node we can place only Gantt Study, Line Simulation Study, Locational Study Robcad Study and Simple Detailed Study nodes. I named it the StudyFolder. See figure 3.6.

3.3.4.6 Creating of 3D libraries

To create a 3D library we have to place all models under system root of the eMServer. If Process Designer is active at the moment of adding of models under system root, Process Designer must be updated. Each model has to have a parent file with *.co or *.cojt suffix to be visible for Process Designer/Simulate. There are two possibilities of creating libraries:

- Manual definition of nodes and manual addition of models to nodes
- Automatic definition of models and manual choosing of nodes to models

First method is quite time-consuming because all is done manually. The second method is faster cause we can exploit tool Create Library in Process Designer.

In my project I defined three basic library nodes: Resource Library node, Product Library node and Manufacturing Features node. I placed it under a Collection node named Libraries, see figure 3.6.

For better structuring I created next seven Resource Library nodes under the main Resource Library node. I named them according to their functionality. Under each node I placed new nodes of given resources, such as Container, Conveyor, Device, Gripper, Gun, Human, Robot, Tool Prototype node and I connected them with models, see figure 3.7. It is also possible to define and develop new nodes and add them to Process Designer.

The product library was created analogically. I added four Part Prototype nodes and connected them with models under Part Library node, see figure 3.7. The Manufacturing Features Library was created automatically by projecting of designed weld points on a part in Process Simulate, see figure 3.7. But it is also possible to add it directly in Process Designer.

Models should contain a TuneData.xml in their files under system root. If not we can generate it by Update Engineering Library tool. The TuneData.xml contains their

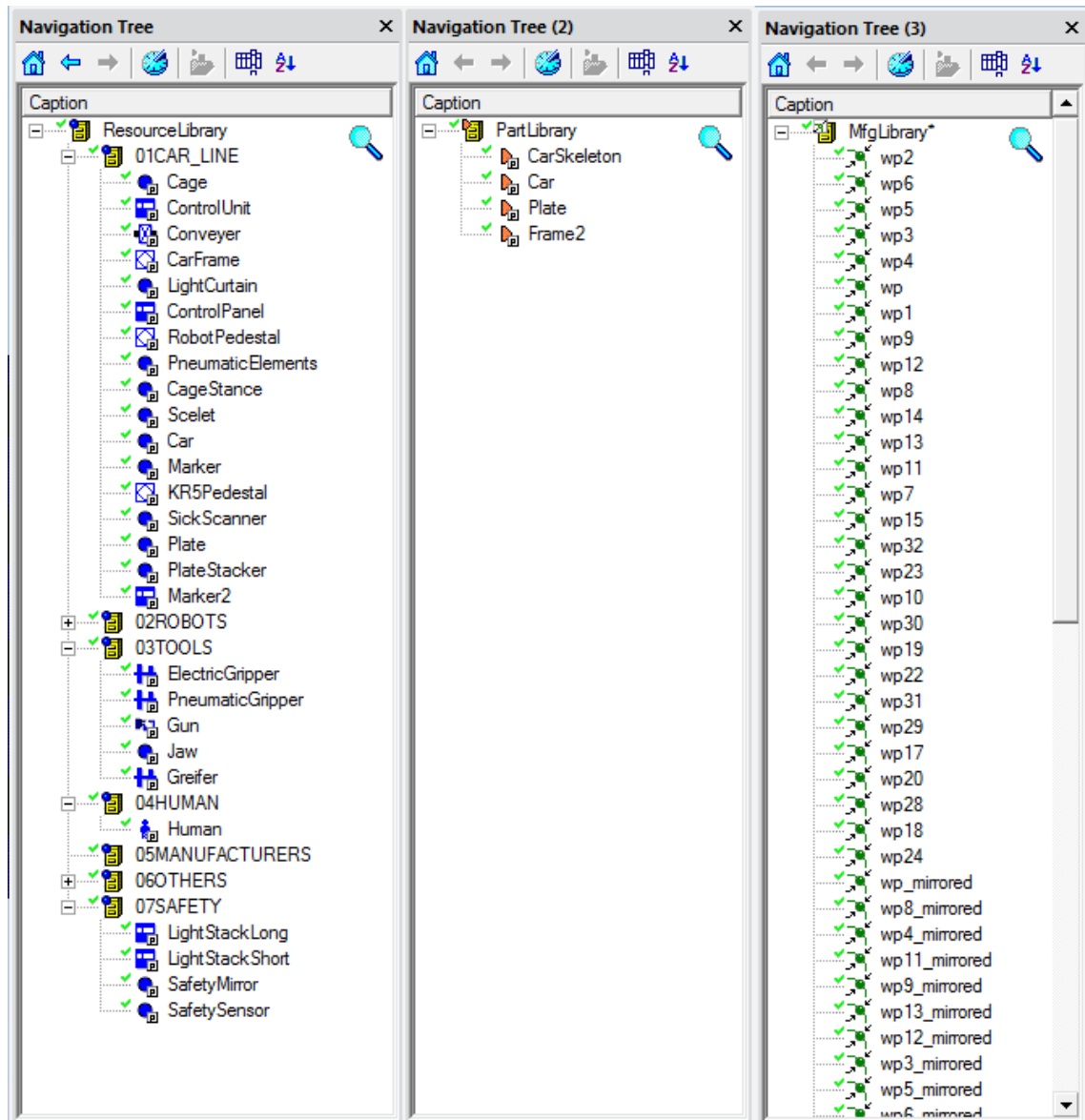


Figure 3.7: Navigation Tree - Resource, Part and Manufacturing Features library

external ID, node type, etc. in a study. We can also generate library preview by Create Library Preview (Process Simulate) to better well-arranged properties of models connected in nodes.

3.3.4.7 Design of a factory concept - resources and operations

In Process Designer there are resource or operation nodes which define places from zone to whole factory (Plant, Zone, Line, Station). According to those nodes I designed a factory concept. Factory contains one hall. This hall contains one welding line and

contains three workspaces. I used twin objects which means that structure of my factory defined in resources is mirrored also in processes. After renaming of these nodes in operation or resource structure we can transfer those names to the resource or operation structure respectively by using of Synchronize Process Objects tool. I placed this resource and operation concept of factory under Collection node called Factory. This concept is later expanded by new children nodes of specific resources and operations in studies, see figure 3.8.

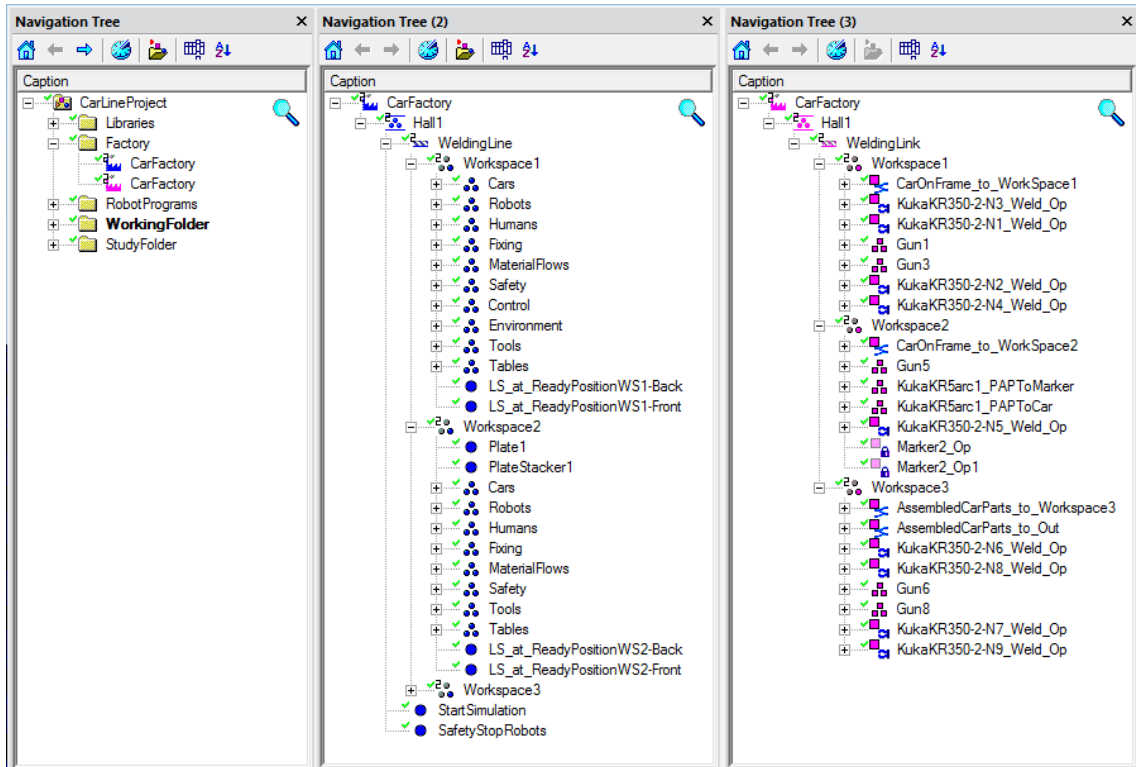


Figure 3.8: Navigation Tree - Factory design (resources and operations)

3.3.4.8 Design of a factory concept - parts and compound parts

I created a Compound Part node named *AssembledCarParts* under Collection node named *Factory*. *AssembledCarParts* then includes one instance of a part named *Plate* and and Compound Part node named *CarOnFrame*, which includes instance of *CarSkeleton* part and *Frame* part from created Product Library, see figure 3.9.

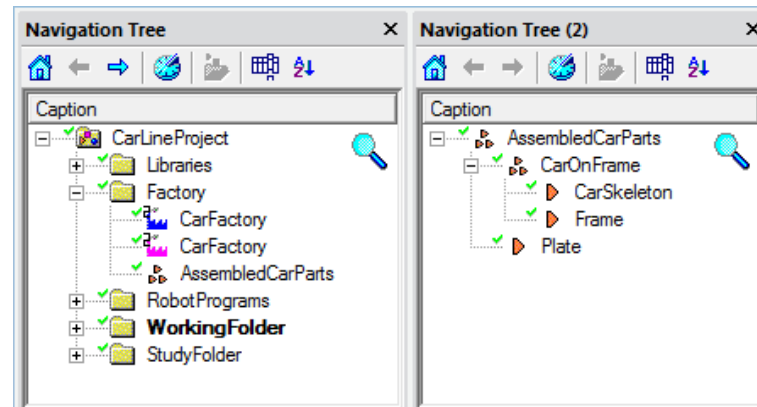


Figure 3.9: Navigation Tree - Factory design (products)

3.3.4.9 Creating of a new study

We create a new Robcad Study node under Study Folder node. Robcad study is a name of the study. It follows from connection of Robcad robotic software and Process Simulate. It enables to use standard (time based) mode of Process Simulate. Line Simulation Study enables to use LineSimulation (event based) mode of Process Simulate. Since version of Tecnomatix 10.1 there is no need of merging Robcad Study to LineSimulation Study to use Line Simulation mode of Process Simulate.

After creating of a Robcad study node I named it CarStudy05. Then I created instances of Factory file (CarFactory - resources, CarFactory - processes, AssembledCarParts - products) , which I placed under Robcad study node, see figure 3.10. When the step before is done we can load the study which means that resources, processes and products are loaded into Resource Tree, Operation Tree and Product Tree.

As you can see final Robcad study node named DiplomaStudy contains also instances of robotic programs under its study node. Robotic programs were created in Process Simulate.

3.3.4.10 Adding of resources to Resource tree

After loading the resources into the Resource tree we can edit resource structure directly. We can add instances of new resources from Resource Library and we can also see them in Graphic Viewer.

I added instances of resources under Workspace 1, 2, 3 as you can see in figure 3.8.

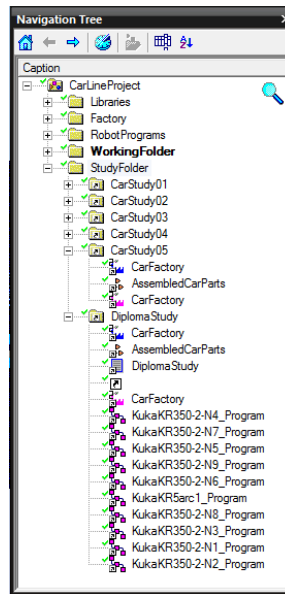


Figure 3.10: Navigation Tree - New study

3.3.4.11 Adding of operations to Operation tree

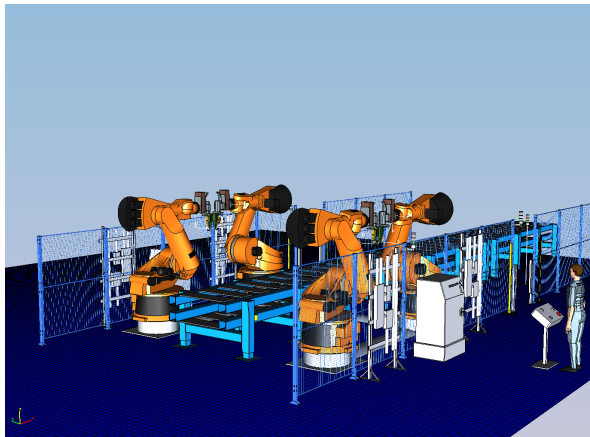
After loading the processes into the Operation tree we can edit operation structure directly. We can add instances of new operation from Operation Library and we can also see them in Graphic Viewer. But I did not work with Operation Library. I edited operation structure directly in Process Simulate during creation of processes. I added instances of processes under the Workstation1, the Workstation2, the Workstation3 as you can see in figure 3.8.

3.3.4.12 Adding of parts to Product tree

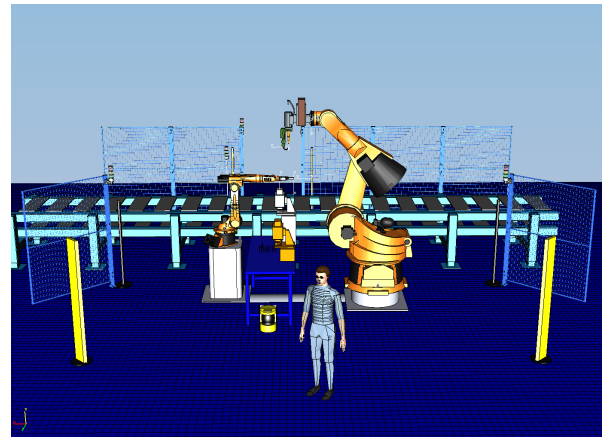
After loading the parts into the Product tree we can edit product structure directly. We can add instances of new parts from Part Library and we can also see them in Graphic Viewer.

3.3.4.13 Creating of a layout

To create a layout of workstations I used many tools, such as Measuring, Placement, Duplicate Objects, Mirror Layout, Sections, Kinematics, Selection and so on. There is a very useful tool called Snapshot Editor to save placement of object in the layout. It creates snapshots. After changes we can easily load back the chosen snapshot with the given placement of objects. Results are visible in figures 3.11(a), 3.11(b), 3.12(a), 3.12(b),

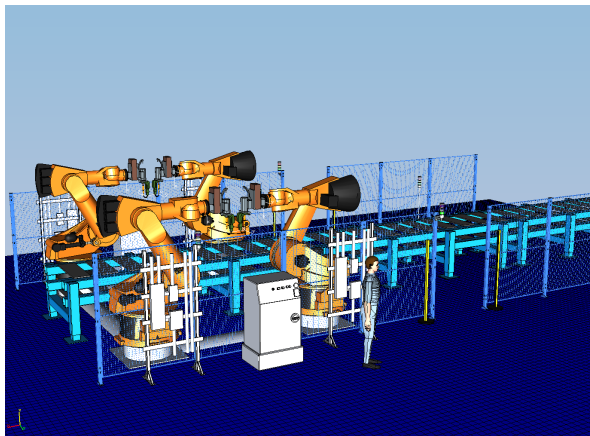


(a) Workspace 1

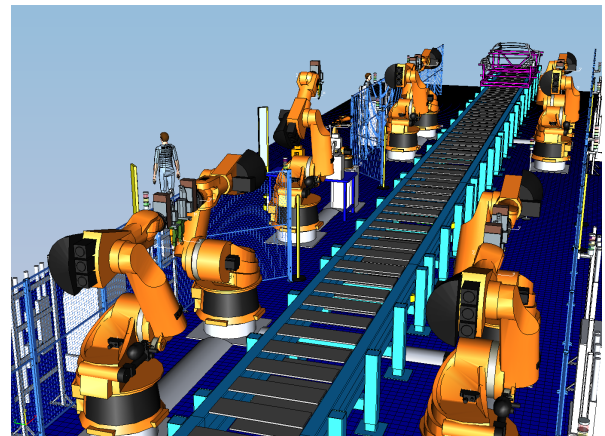


(b) Workspace 2

Figure 3.11: Workspaces



(a) Workspace 3



(b) Entire product line - side view

Figure 3.12: Workspaces

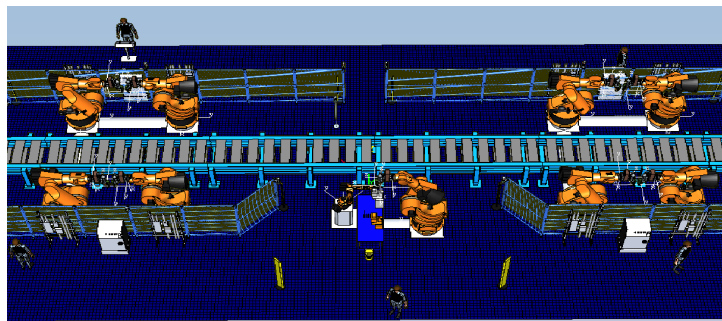


Figure 3.13: Entire product line - top view

3.13.

3.4 Process Simulate

Process Simulate provides an integration of the eMServer and its planning tools with the 3D environment. Main components of Process Simulate are the Navigation tree, which includes all the data of Object Tree (Resources, Parts, Appearances, etc.) and Operation Tree. Object Tree contains a hierarchy of objects in the study. Operation Tree contains only operation structure and operations of objects in the study. See figure 3.14.

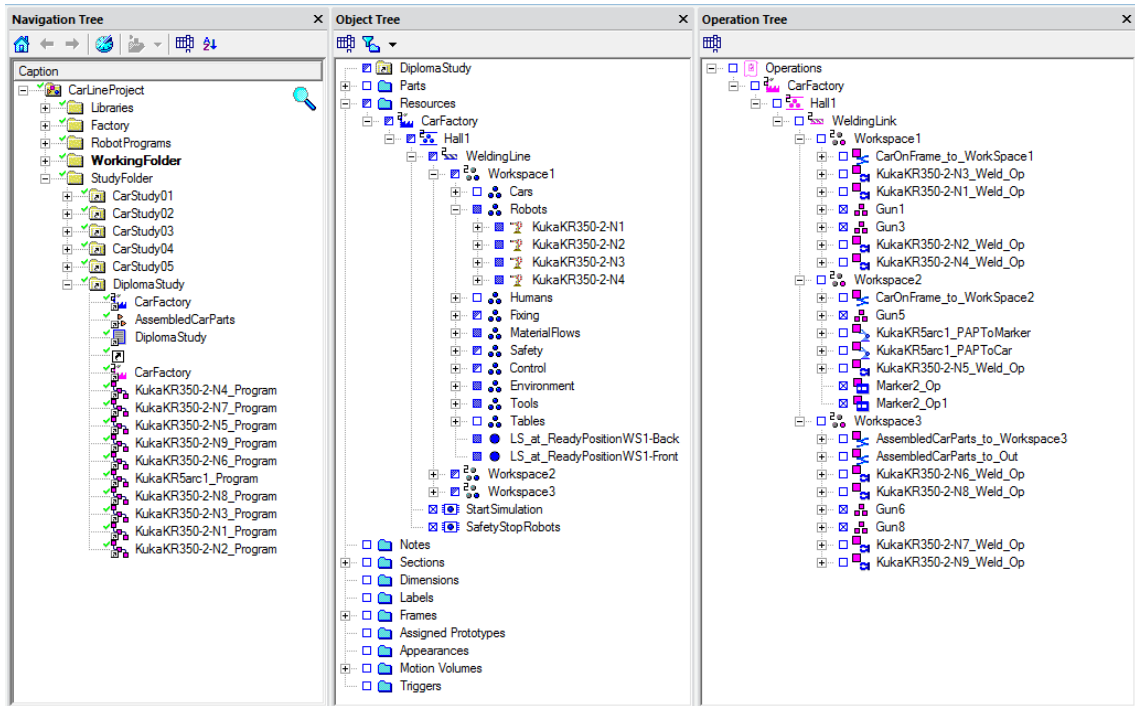


Figure 3.14: Process Simulate - Navigation, Object, and Operation Tree

The environment of Process Simulate is modular and allows creating plug-ins according to Tecnomatix.NET API [13]. Example of work with plug-ins is visible in diploma thesis [9]. Author of this thesis created a plug-in to extract information from the software to *.xml file.

Operations are activated in Sequence Editor or Path Editor. Sequence Editor serves for an overall simulation, whereas Path Editor is used to tune up a given operation, especially robotic operations and programs.

Process Simulate has many tools and settings which were used during examination of this work. I made an overview of the most important methods and tools from the point of view of creating this work. These tools are presented in the main menu bar of Process Simulate.

3.4.1 Modeling

In Process Simulate we can edit or create 3D models. To do that there is the Set Modeling Scope option on a given model. If a model is set to modeling, a small icon with M appears in the left down corner of a given object node in Object Tree. After finishing the modeling, we should set End Modeling option and the small icon with M will disappear on a given object node. Modeling tool itself enables to create 2D models (polylines, curves, circles, etc.) and 3D models (boxes, cylinders, cones, spheres, torus) and it has functionality such as Unite, Subtract, Intersect, Mirror Objects, Duplicate Objects etc. On the other hand, to create large projects I recommend to use special modeling programs, such as NX from Siemens.

3.4.2 Kinematics

3.4.2.1 Kinematics Editor

In Process Simulate we can create kinematics of 3D models in Kinematics Editor. There is a precondition that intended 3D model has to be set to modeling (see Modeling). Kinematic model consists of kinematic chain of links and joints. Kinematic joint are revolute and prismatic joints and we can select their movement limits, speeds and accelerations. Joint is defined between two links (parent link and child link), where link is a part or more parts of a given 3D model. As an example, kinematic chain of a robot KUKA KR5 arc is defined, see figure 3.15. Revolute joints are marked j1-j6 and links are marked k1-k7. Colors of links correspond to colors on robot. After creating a kinematics model, it should appear a *kin_graph.xml* file appear in the file folder of the corresponding model in system root. It contains data of the created kinematics chain.

After creating the kinematic chain we can define different poses of such device.

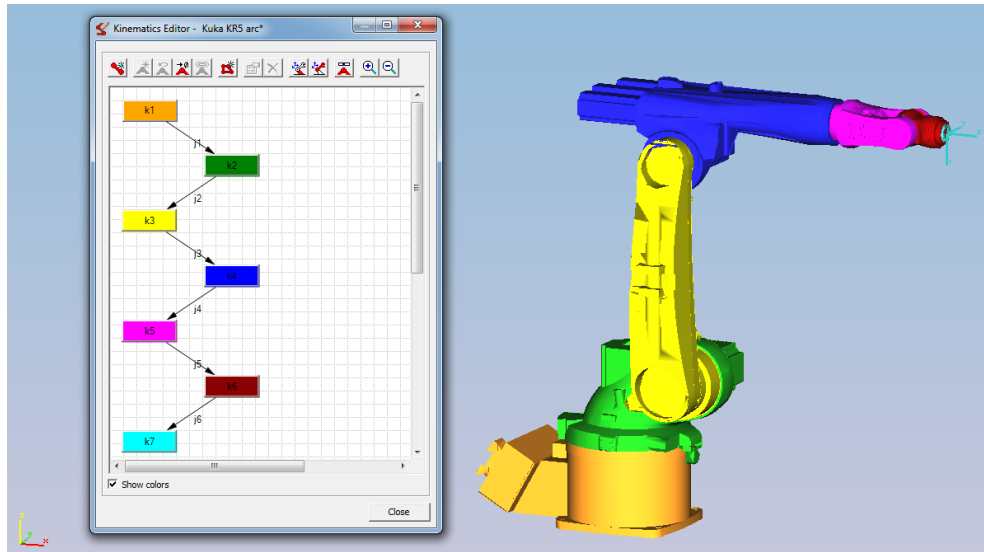


Figure 3.15: Process Simulate - Kinematic Editor

3.4.2.2 Tool Definition

In Process Simulate we can define a 3D model as a tool in Tool Definition. The tool means an object that can be attached to a robot to enable it to perform a task. Tool types are Gun, Servo Gun, Pneumatic Servo Gun, Gripper, Paint Gun. There is also defined Tool Center Point and its Base Frame. There is a precondition that intended 3D model should define required kinematics and it has to be set for modeling (see Modeling subsection). In Process Simulate there is no possibility to use intended 3D model as a tool without its tool definition.

3.4.2.3 Reach Test

To verify that robot is able to reach defined locations or weld points we can use Reach Test tool. We select a given robotic operation which we test. Reachable points are marked with \checkmark symbol and unreachable points with \times symbol as you can see in figure 3.16 . We can also choose if a given robot is able to reach these locations with its joint limits (Joint Indication Limits).

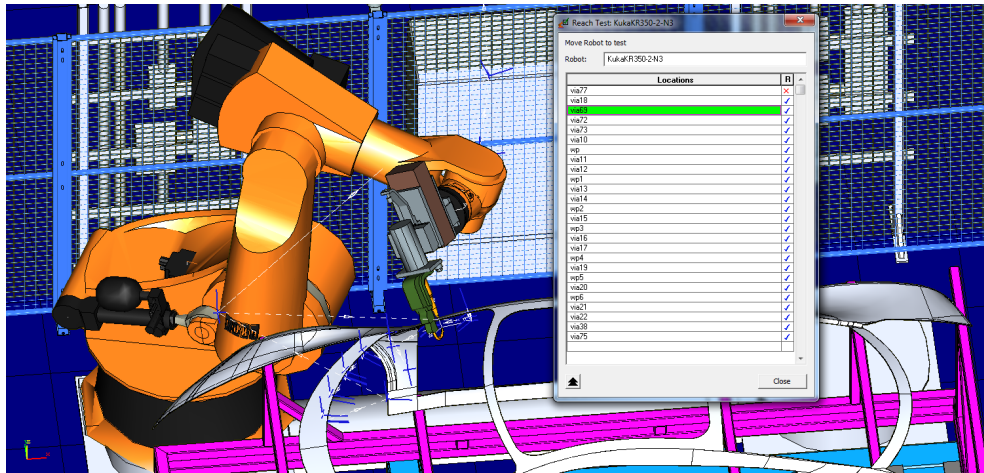


Figure 3.16: Process Simulate - Reach Test

3.4.2.4 Smart Place

Smart Place enables to find optimal location for a robot. It calculates all positions of robot basement in a square field which are reachable for a selected robotic operation. See figure 3.17. Blue squares are reachable positions of robot for a given robotic operation and red squares are unreachable positions.

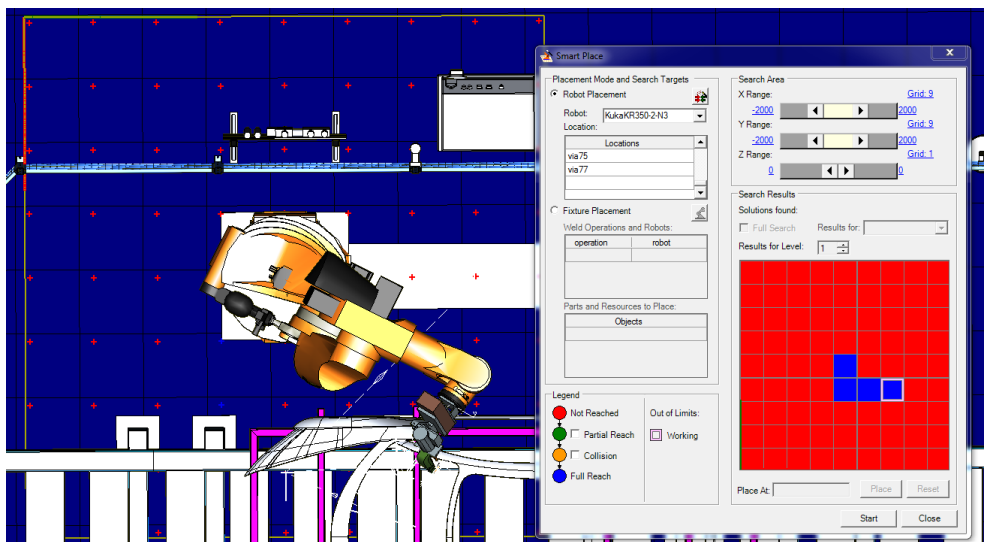


Figure 3.17: Process Simulate - Smart Place

3.4.3 Operations

Process Simulate contains following types of operations.

3.4.3.1 Compound Operation

Compound Operation is a hierarchical node that consists of other operations and that can include also other compound operations.

3.4.3.2 Object Flow Operation

Object Flow operation enables to move an object from one place to another.

3.4.3.3 Device Operation

Device operation enables to move a device from its one pose to other pose. A device is an object which has defined its kinematic chain and poses, where the pose is a set of joint values. For example, a gripper can have a position OPEN which is marked as a first pose and a position CLOSE which is marked as a second pose. After we can define a Device operation to open or close mentioned gripper.

3.4.3.4 Device Control Group Operation

In Process Simulate we can define a Device Control Group of devices with similar signs or behavior. This operation has a possibility to determine a common operation for all devices in Device Control Group. So devices move from one pose to other pose.

3.4.3.5 Gripper Operation

Gripper operation is defined on a gripper. Gripper can then grip objects and release them.

3.4.3.6 Pick and Place Operation

Pick and Place operation is defined for a robot with defined gripper. It enables to move an object from one place to another place.

3.4.3.7 Weld Operation

Weld operation is defined for a robot with defined weld gun. It enables to create a weld operation on parts, objects, etc.

3.4.3.8 Continuous Operation

Continuous operation is defined for a robot with tool. It surfaces laser welding and continuous glue applications.

3.4.3.9 Non-Sim Operation

It is an empty operation that enables to reserve a time interval for future operation or just for a non-sim operation which we exploit in a different way (i.e. precondition of transition in CEE simulations, which will be defined later).

3.4.3.10 Robot Path Reference Operation

It enables to tune robotic program in line simulation mode of a given robot.

3.4.3.11 Swept Volume

Swept Volume is an envelope of a selected operation. We can use it to control working space of robots, devices, etc. as shown in figure 3.18.

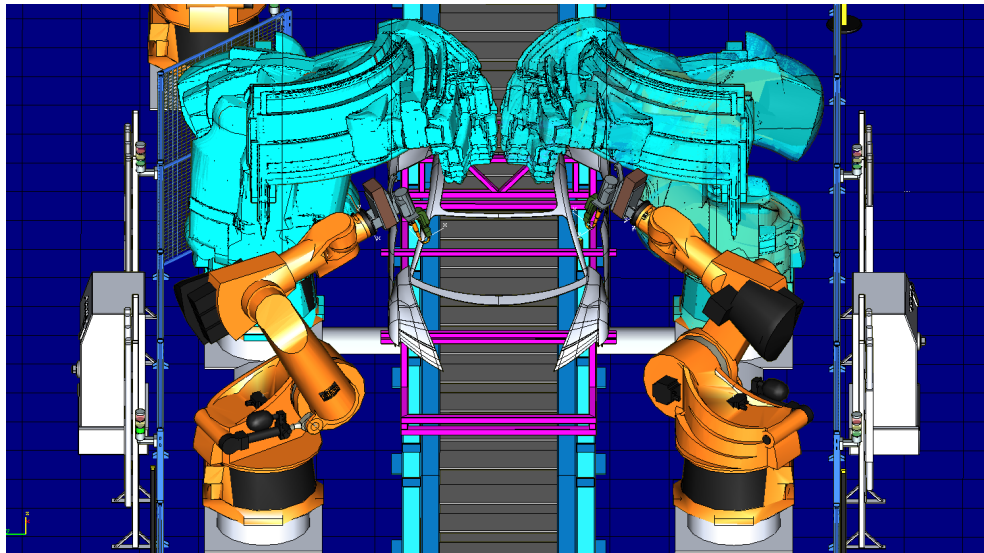


Figure 3.18: Process Simulate - Swept Volume

3.4.3.12 Interference Volume

Interference Volume is an intersection of two Swept Volumes. Based on that we can check if the operation space of two devices intersects.

3.4.4 Welding

3.4.4.1 Create Weld Points

Create Weld Points is an option of Process Simulate which is a direct way how to add weld points on parts and, subsequently, to define weld operation on these points. These points must be projected before doing this. In Process Designer or Process Simulate there is automatically created a Manufacturing Features Library in Working Folder in Navigation Tree.

The system assigns the projected weld points and attaches them to the selected part. Projection of a weld point adds orientation to the weld point and creates a weld location which specifies the perpendicular and approach axes to the location. Before projection the system indicates weld points by a small cross . After projection the weld points have a frame representation [14].

3.4.4.2 Pie Chart

Pie Chart enables to describe the approach vector for a weld gun to a selected weld or path location. It also shows if a given approach vector is reachable or unreachable. See figure 3.19.

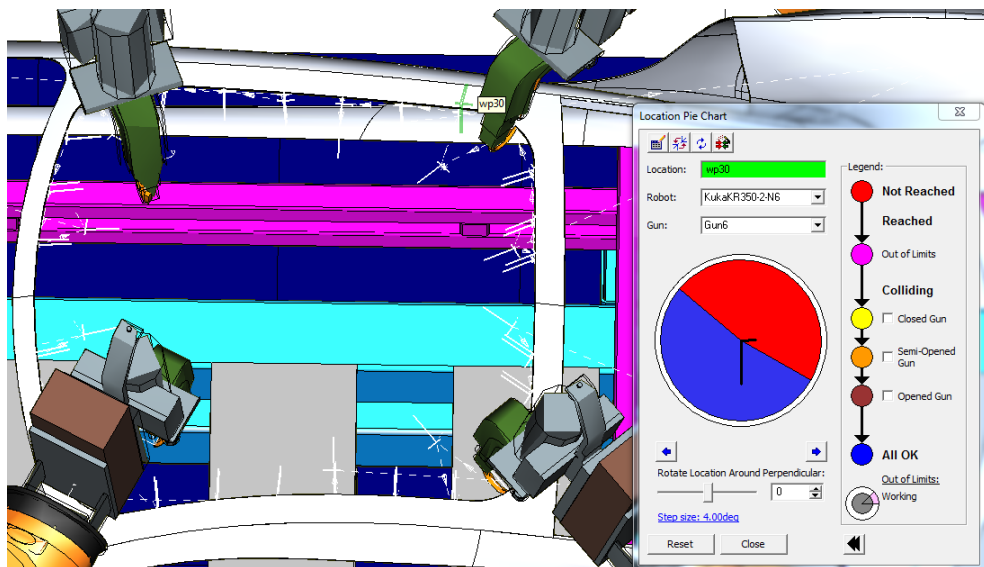


Figure 3.19: Process Simulate - Pie Chart

3.4.5 Robotics

3.4.5.1 Robot Controller

Robots in Process Simulate have set a default robot controller. It can be used with all robots. Its accuracy is around 80 percent compared to the real controller of a robot. Positions of points are same for the real robot controller and default robot controller but a path between them can be different. Alternatively we can use a customized controller package for our robot type which contains:

- Robot Controller Simulation module (RCS module)
- Off Line Programming package (OLP package)

The RCS module for each robot is produced by the robot manufacturer and supplies accurate data for motion planning and inverse kinematics. The manufacturer employs exact knowledge of the robot's characteristic and limitations to create accurate joint values for robot motion and to plan the optimal path of motion between points in space. The RCS module passes this information back to Process Simulate in order to create a Realistic Robot Simulation (RRS) [14].

To use RCS module for creating an RRS we should install OLP package of our robot, which is an add-in for Tecnomatix application. I used Tecnomatix 10.1 KUKA KRC 64bit v2.19 for KUKA KR5 arc which supports KRC4 controller and its package name is KUKA-Krc. It contains functionality, such as :

- Simulation Controller - In my case Kuka-Krc
- Custom Teach Pendant - It enables, for example, to change joint locations etc. for a given position of a robot (robot can reach some position with different joint locations)
- Application for downloading robotic programs - I am able to generate a Kuka program for my robot with *.src, *.dat, *.olp and *.log files.
- Application for uploading robotic programs - I am able to upload robotic program back to the Process Simulate and tune it up.

After installing the OLP Controller Package we can install RCS module. In my case I used RCS module rcs_kuka_WIN_ KRC8.2R_R01_00 with Manufacturing Data

(MADA) Krc8.2_r01_V1.0.0.b95(Krc4-b95-KUKA8_2_RRS1Structure) and TuneAddOn application rcs_kuka_WIN_TuneAddOn_07_08_2012. Mapping tables of given RCS modules and Off Line Controller packages are available on the Global Technical Access Center of Siemens. Accuracy of this modules is 95 percent compare to a real robot controllers. See figure 3.20.

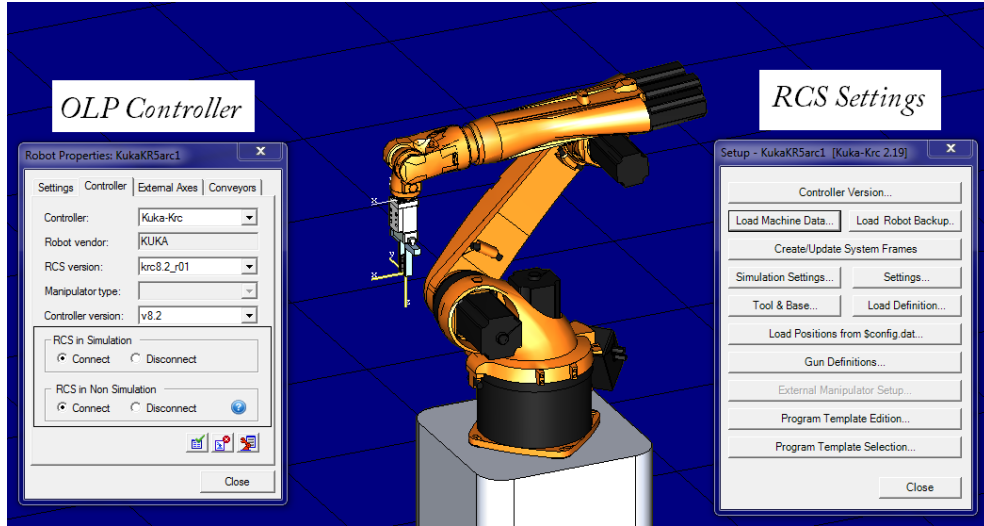


Figure 3.20: Process Simulate - Robot Controller Simulation (RCS) module

According to [15] a simulation is always based on assumptions and simplifications. Due to the orientation on the RRS1 standard a simulation with the KUKA.RCS module has the following limitations:

- The RCS of KUKA robot cannot represent the dynamic characteristic of a robot (e.g. oscillation, vibration).
- The inputs and outputs of a robot can not be simulated via RCS module
- The external friction and inertia of a system (e.g. holding force of a gripper, forces generated during spot welding) cannot be simulated in RCS module
- The RCS module cannot simulate the flexible connection (e.g. cable package)

3.4.5.2 Robot Program

In Process Simulate we can work with robotic programs in Program Inventory. We can create a new one, edit them, delete them, download program to a shop-floor robot and also upload a robot program to Process Simulate. To use robotic program with its

robotic controller, program has to be set as default. Programs set as default are marked in bold. After that they can be called, for example, in line simulation mode according to the Path number of a given program, etc as shown in see figure 3.21

To be able to download (generate) a program into the selected robot we can use a default controller but the program is quite general and it could require other changes. That is why we should use the selected controller of a manufacturer. In my case it is KUKA-Krc (OLP Controller Package). In robotic program at lease a tool and the basement of the robot should be defined. The editor of the robot program is directly in the Path Editor tool. See figure 3.20, 3.21 and 3.22. During a download program to shop-floor robot of a KUKA controller there are created four files, such as *< programName > .src*, *< programName > .dat*, *< programName > .olp* and *< programName > .log*.

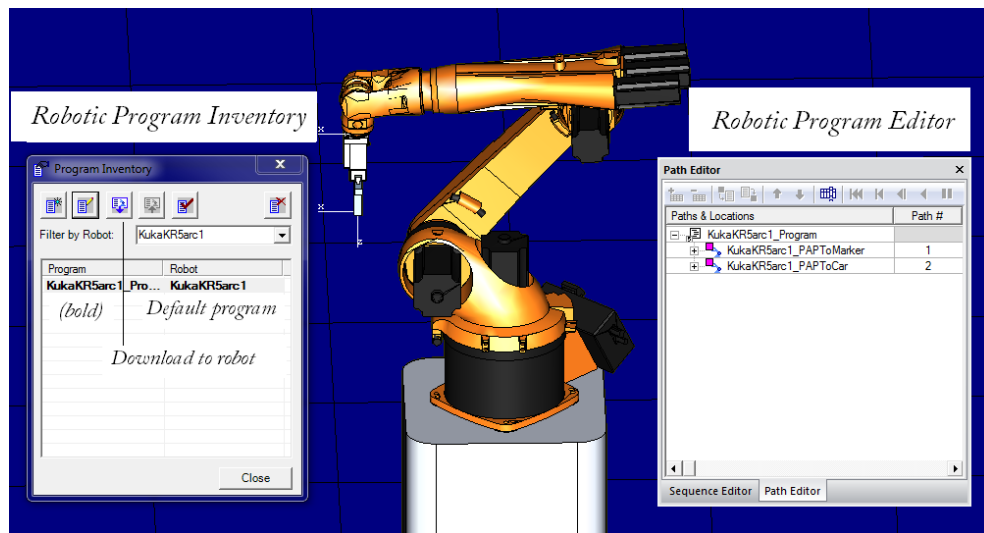


Figure 3.21: Process Simulate - Robot Program Inventory

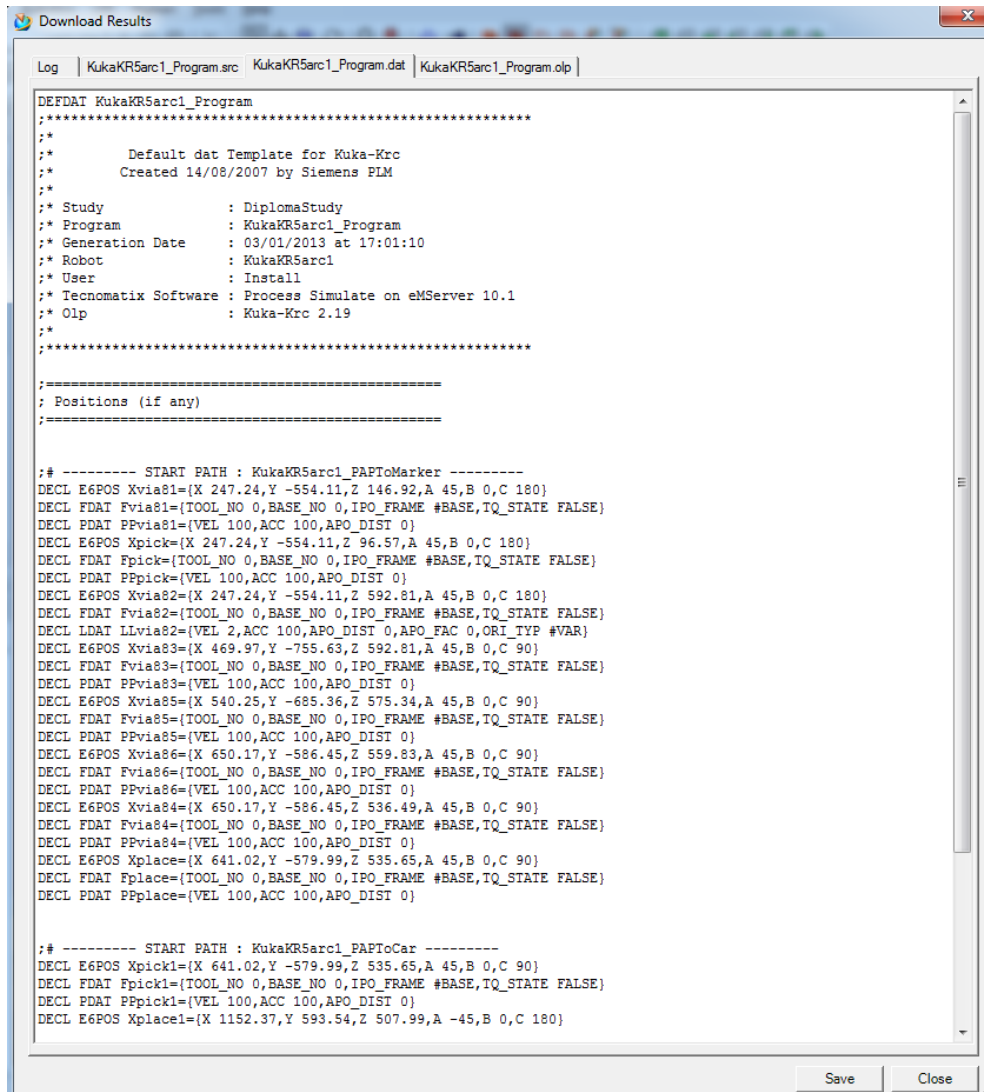


Figure 3.22: Process Simulate - Download Program

Chapter 4

Simulations

According to [15] the word Simulation is defined as follows in VDI guideline 3633. Simulation is the replication of a system with its dynamics processes on the basis of an experimental model in order to apply the acquired knowledge to the real world. A simulation is always based on assumptions and simplifications. Simulations are divided into:

- Process Simulation - this type of simulation is used for planning complex production and material flow systems.
- Finite Element (FEM) simulation are used primarily for calculating the stability or load capacity of a system.
- Graphical 3D simulations - 3D simulations of kinematics systems that can display and optimize the motions of a production system. You can also use this software to optimize the layout and predict cycle times.

In this work I concentrate on Process Simulation and Graphical 3D simulation which are included in Process Simulate software.

Process Simulate itself includes time-based simulations and event based simulations. Time-based simulation is enabled in Standard Mode and partly also in Line Simulation Mode of Process Simulate. On the other hand event based simulations are enabled only in Line Simulation Mode of Process Simulate. Nevertheless according to my experience complexity of settings of an event-based solution exponentially rises with requirements compared to time-based simulations.

4.1 Time-based Simulation

Main build stones of a time-based simulation are resources, products and operations. Time-based simulation is limited by its duration of operation and it is strictly defined to one scenario of a given simulation. It means that defined simulation strictly leads to one solution. Because an event-based solution is quite complex I made a rough time-based simulation at first to check the behavior of the a designed model of the production line. Logic of time-based simulations is based on Gantt chart. It describes a sequence of operations in simulations.

4.1.1 Implementation of Time-based simulation

To implement a time-based simulation I had to do many settings and modeling, such as defining of kinematics (a gripper, a marker, light stacks), modeling of all operations (robotic operations, material flow operations), designing of weld points (and its approach angles and position), defining of a Gantt chart for created operation and so on. Some of them are briefly described in following subsections.

4.1.1.1 Adding kinematics to a gripper

General description of Kinematic Editor was mentioned in section Process Simulate of this work. To define a kinematic model of the gripper I defined links called BASE (fix part of gripper), link1 and link2 (movable jaws of a gripper). Between the BASE and link1 there is the prismatic joint j1. Between BASE and link2 there is the prismatic joint j2. Joint j2 has also defined the kinematic function $j2 = (-1) * D(j1)$ which says that if the kinematic joint j1 moves to left kinematic joint j2 moves to right. See figure 4.1.

After defining of the kinematics it is necessary to define the kinematic model in Tool Definition as the gripper. We also determine its Tool Center Point (TCP) frame and its Base frame.

4.1.1.2 Material Flow in Workspaces

Material flow of a car is in Weld Line divided into four section between points from A to E, see figure 4.2. And material flow of a plate is divided into two sections between points from F to H, see figure 4.3. These sections are

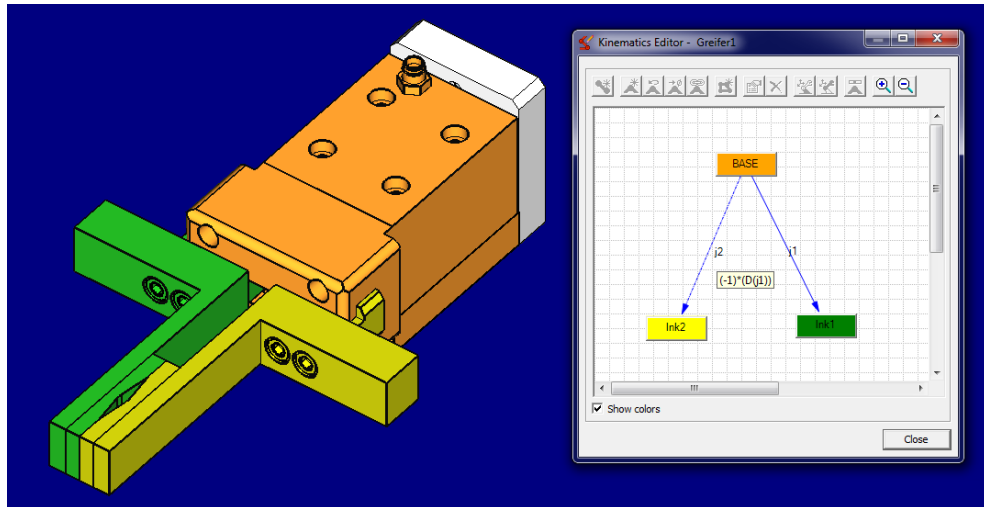


Figure 4.1: Process Simulate - Kinematics of a gripper

- From A to B - define a material flow of the operation CarOnFrame_to_Workspace1, where a car on frame flows from location A to B.
- From B to C - define a material flow of the operation CarOnFrame_to_Workspace2, where a car on frame flows from location B to C.
- From C to D - define a material flow of the operation AssembledCarParts_to_Workspace3, where a car on frame with the welded marked plate flows from location C to D.
- From D to E - define a material flow of the operation AssembledCarParts_to_Out, where a car on frame with the welded marked plate flows from location D to E.
- From F to G - define a material flow of the robotic operation KUKAKR5arc1_PAPToMarker, where KUKA KR5 arc picks and places a plate under the marker.
- From G to H - define a material flow of the robotic operation KUKAKR5arc1_PAPToCar, where KUKA KR5 arc moves the plate from the marker and places it on a car skeleton.

4.1.1.3 Welding Robotic Operations

In welding line there are defined basic welding and material flow operations. In Workspace 1 robots N1, N2, N3 and N4 weld a car skeleton, exactly on the back glass

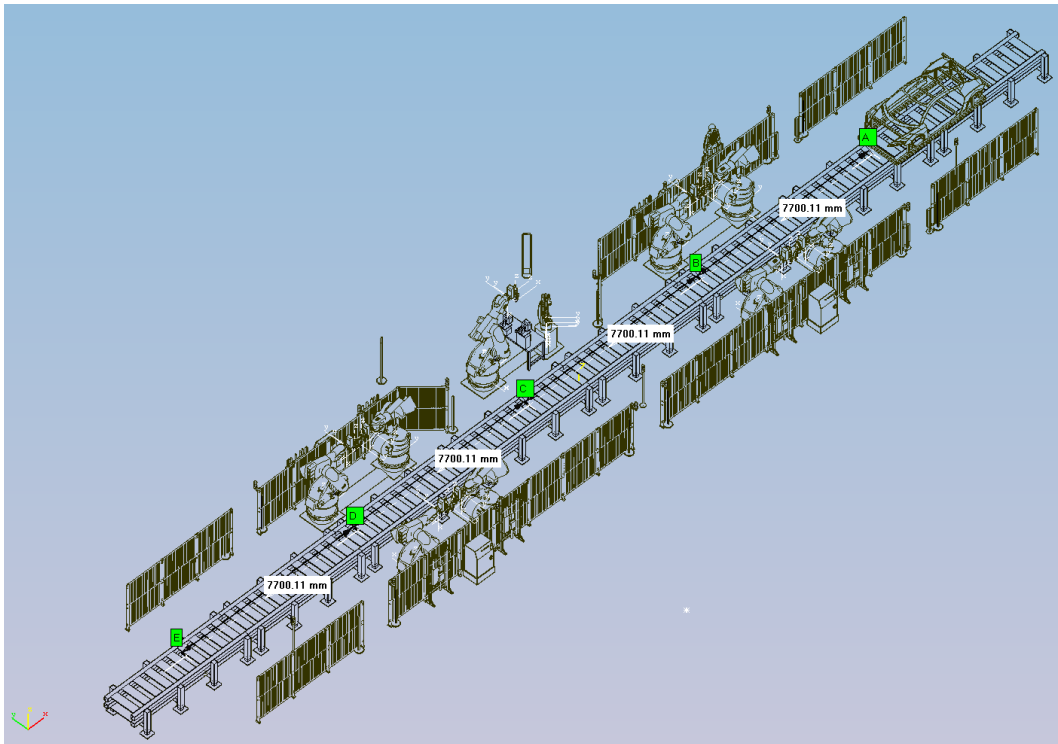


Figure 4.2: Sections of a Welding Line - Car flow

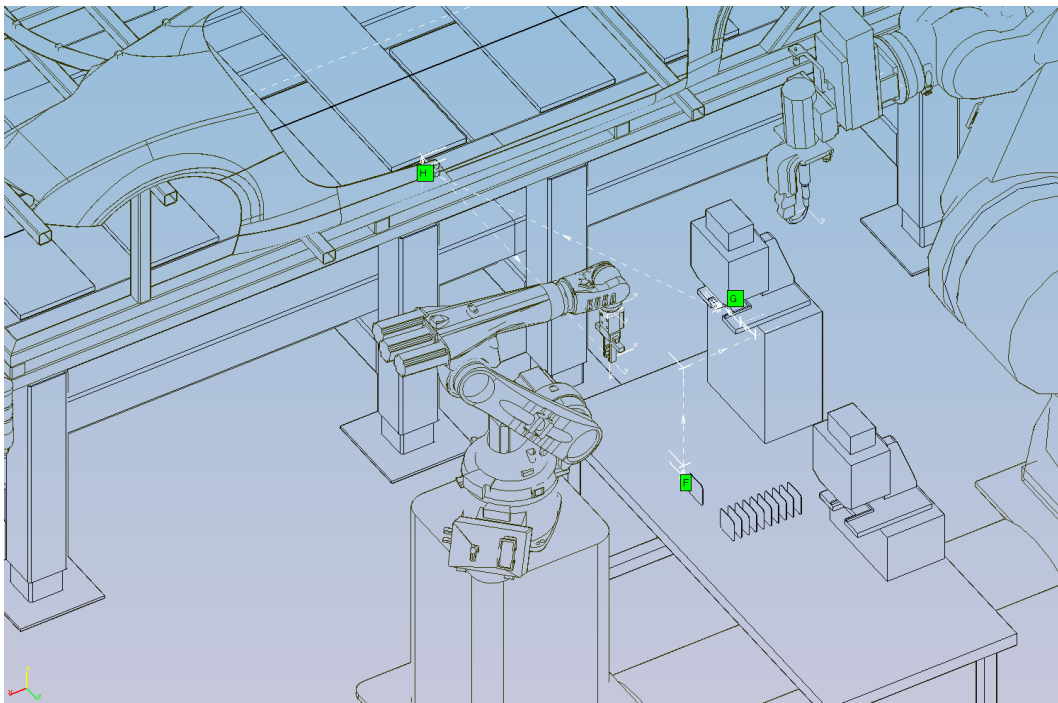


Figure 4.3: Sections of a Welding Line - Plate flow

frame and the front glass frame. In Workspace 2 robot N5 welds a plate to the car on the right bottom side down of a car. In Workspace 3 robots N6, N7, N8 and N9 weld a roof of skeleton of a car. The designed weld points are visible in figure 4.4. They are marked by red cubes on the car skeleton.

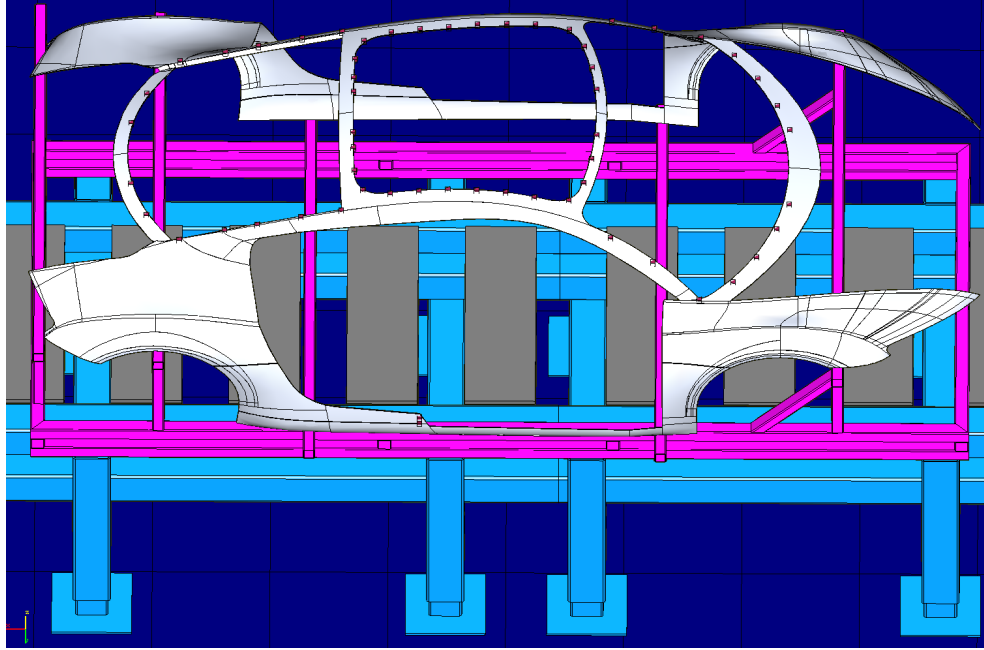


Figure 4.4: Designed weld points

According to the designed welding points there were defined welding operation of Workspace 1, 2 and 3. For Workspace 1 these operation are described in table 4.1 and visible in figure 4.5 (green welding and approaching locations).

For Workspace 2 these operation are described in table 4.2 and visible in figure 4.6 (green welding and approaching locations).

For Workspace 3 these operation are described in table 4.3 and visible in figure 4.7 (green welding and approaching locations).

Name of operation	Type of operation	Resources
KUKAKR350-2-N1_Weld_Op	Weld operation	KukaKR350-2-N1, Gun1
KUKAKR350-2-N2_Weld_Op	Weld operation	KukaKR350-2-N2, Gun2
KUKAKR350-2-N3_Weld_Op	Weld operation	KukaKR350-2-N3, Gun3
KUKAKR350-2-N4_Weld_Op	Weld operation	KukaKR350-2-N4, Gun4

Table 4.1: Description of welding operation of Workspace 1

Name of operation	Type of operation	Resources
KUKAKR350-2-N5_Weld_Op	Weld operation	KukaKR350-2-N5, Gun5

Table 4.2: Description of welding operation of Workspace 2

Name of operation	Type of operation	Resources
KUKAKR350-2-N6_Weld_Op	Weld operation	KukaKR350-2-N6, Gun6
KUKAKR350-2-N7_Weld_Op	Weld operation	KukaKR350-2-N7, Gun7
KUKAKR350-2-N8_Weld_Op	Weld operation	KukaKR350-2-N8, Gun8
KUKAKR350-2-N9_Weld_Op	Weld operation	KukaKR350-2-N9, Gun9

Table 4.3: Description of welding operation of Workspace 3

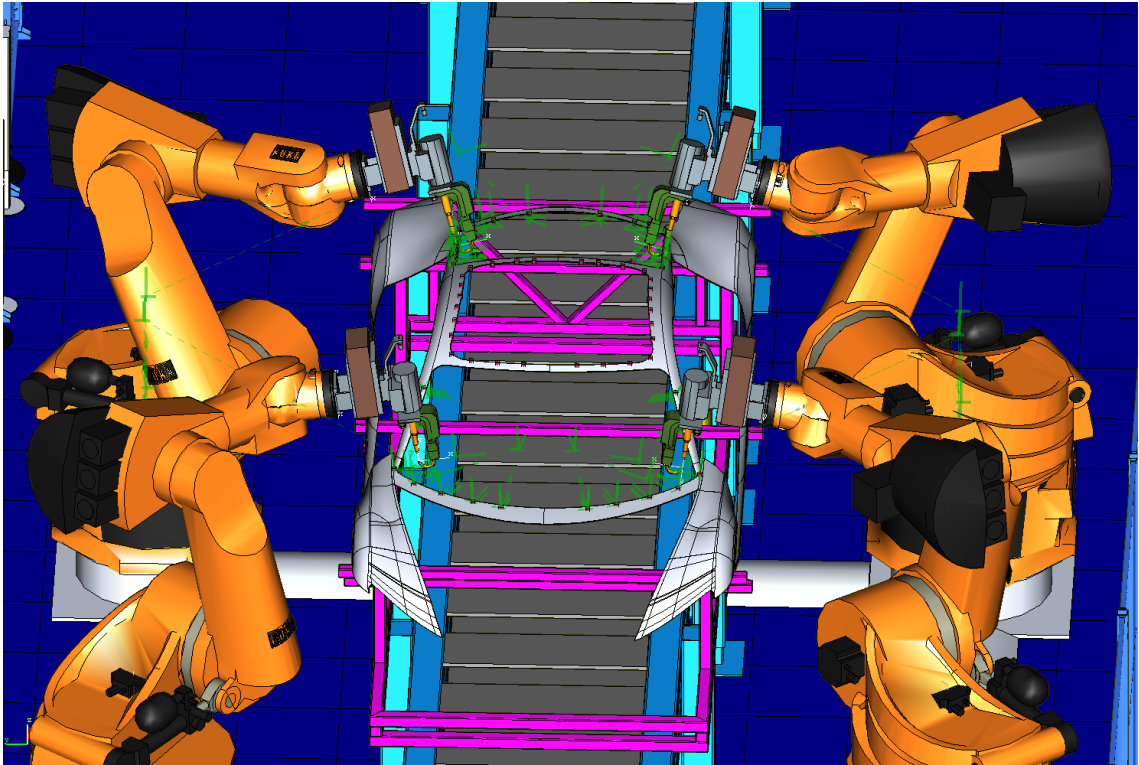


Figure 4.5: Welding operations of a rob cell of Workspace 1

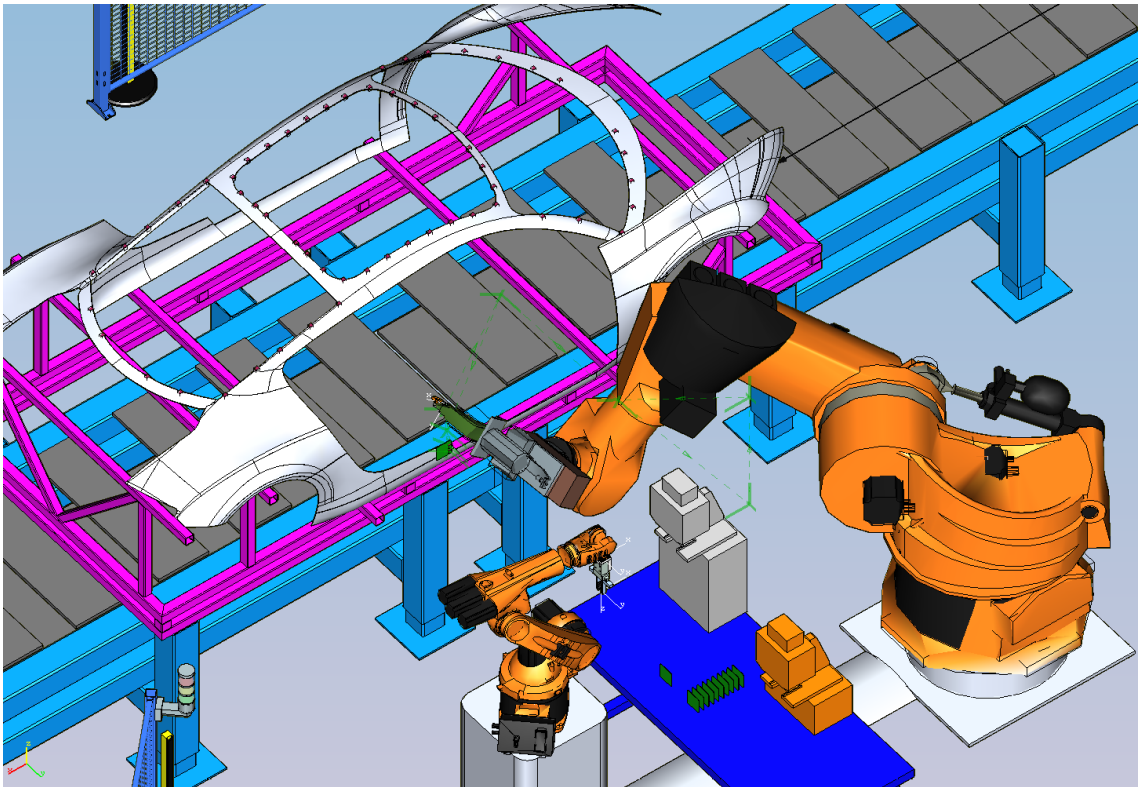


Figure 4.6: Welding operations of a rob cell of Workspace 2

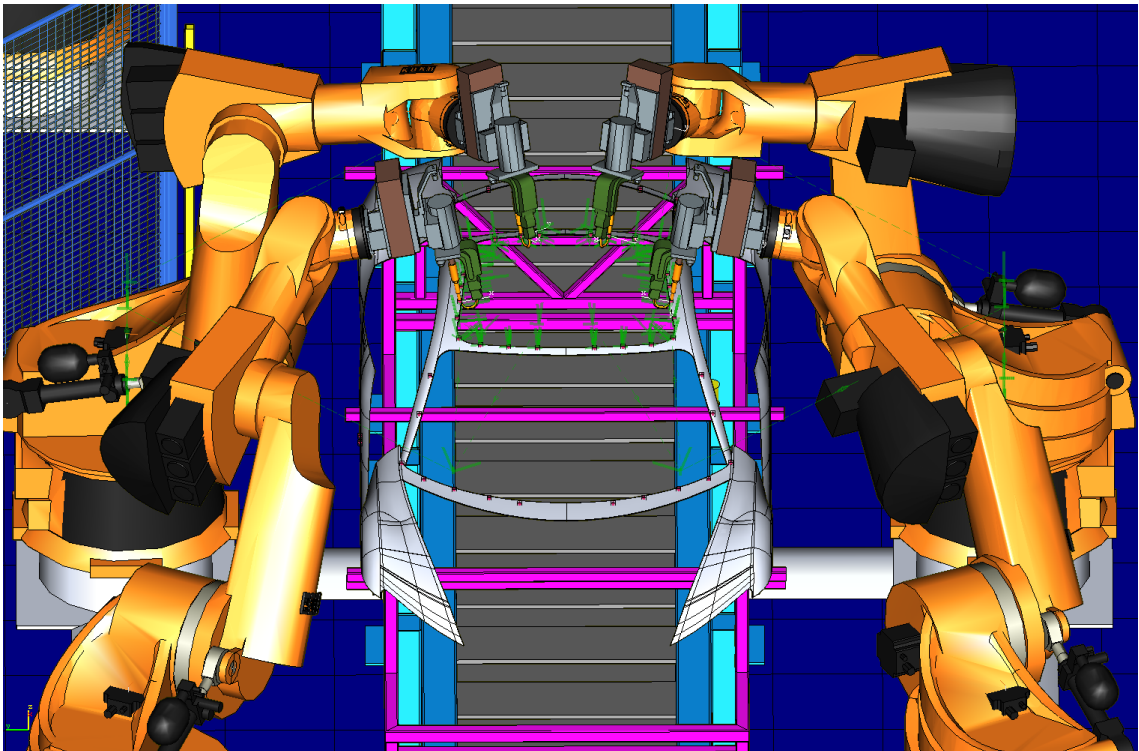


Figure 4.7: Welding operations of a rob cell of Workspace 3

4.1.1.4 Gantt Chart

I used Gantt Chart for defining the sequence of time operations, see figure 4.8 . There is defined which operations follows after previous operations and also which operations run simultaneously. We can create link by small icon of a chain. After that I was able to run this simulation. Its duration is 152 seconds. In this simulation I did not define operations of light stacks, safety technologies (light curtains and light scanner) and neither sensors. I did it later in the event-based simulation with another approach. In this simulation I also used robots which are controlled by robotic operations. In comparison robotic controller controls robots in event-based simulations. This process is comparable to real approach.

I also used robotic operation instead of robotic controller. It is carried out in Virtual Commissioning simulation.

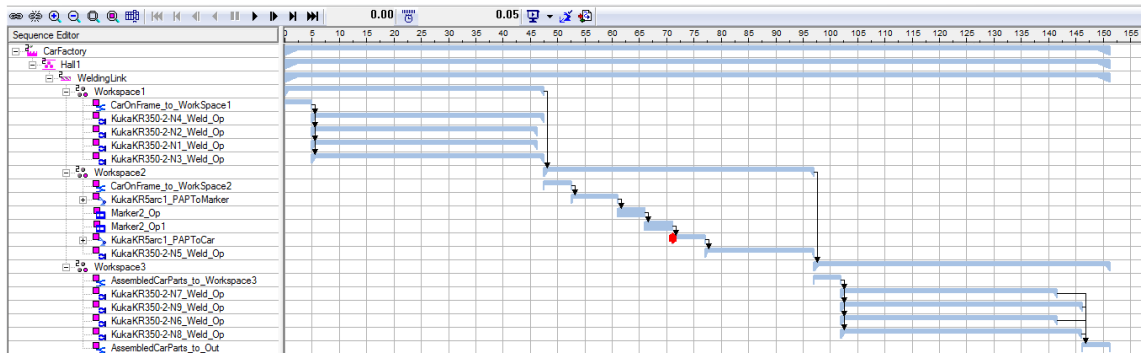


Figure 4.8: Gantt chart of time-based simulation

4.2 Event-based Simulation

The main build stones of event-based simulation are resources, appearances, operations and signals. Event-based simulation is endless simulation with many variants of scenarios. It means that one defined simulation can follow to many different solutions, which are controlled by different events during a simulation.

Event based simulations are subdivided into: Cyclic Event Evaluator (CEE) simulation and Virtual Commissioning (PLC) within this software. Nevertheless they are really similar. Main difference is in their logic. Cyclic Event Evaluator is a processor of the simulation engine and it runs cyclically in Process Simulate.

In Virtual Commissioning the external PLC (or a PLC Simulation) is a processor of

the simulation engine. . Therefore there are some differences between using of logic in Process Simulate.

In this thesis there are phrases Virtual Commissioning and OPC simulations equivalent.

4.2.1 CEE Simulation

Cyclic Event Evaluator simulation is closer to real processes of factory than time-based simulation, but further than OPC simulations. CEE simulations contains three logic types :

- Sequence transition condition - it is a combination of the time-based approach of simulations with transition condition signals. Therefore to use a transition there has to be defined a link between two operations (Gantt Chart) in Sequence Editor. If the transition condition is satisfied following operation is triggered. In the transition to multiple operations we can create simultaneous operations or variant branches. Each variant branch has its extra transition condition.
- Modules - modules are special logic blocks which behave like an internal PLC of Process Simulate. Module conditions control all devices connected to the PLC.
- Logic Block - logic block control logic of devices, robots and processes input signals. In a real product line not all of components are controlled by a robot or a PLC controller. For example, smart drilling machines or CNC machines contain their own logic. To determine this logic we use logic blocks.

We can combine all three types mentioned above and create a powerful simulation. For this is used combination of time-based approach with modules, transition conditions and logic blocks. Nevertheless I worked primarily in Virtual Commissioning (OPC simulations) controlled by the PLC in my master thesis. To read more information about CEE simulations I recommend following master thesis [16].

4.2.2 Virtual Commissioning

OPC technology is a core functionality of Virtual Commissioning. OPC connection enables to PLC program run on a real PLC and tune the real manufacturing line before its real commissioning. Production cell can be validated and analyzed in the design phase of

the virtual 3D environment. Therefore it is only a small step to create a new real product line.

Virtual Commissioning is sometimes also called OPC simulations.

In comparison with CEE simulations Virtual Commissioning contains only two logic types:

- Programmable Logic Controller - whole logic of a PLC can be used to control simulation in Process Simulate
- Logic Block - this logic block also control the logic of devices, robots and it processes input signals. Logic blocks in Virtual Commissioning are equal to logic block in CEE simulations.

We can combine both types arbitrarily, but we have to take into consideration watchdog of the PLC (time limitation of PLC's cycle). Logic Blocks of CEE and Virtual Commissioning are equivalent.

4.2.2.1 Appearances

Appearances are special parts which has to be generated each time the study is loaded or during the simulation. It follows from requirement of an endless event-based simulation. If I use a part in a production process of the time-based simulation I have only one instance of the part node, see section 3.3.4.1. I need as many instances as parts. But we want to simulate this production process in endless cycles with endless amount of one part. Therefore it was defined a new approach to parts in event-based simulations. We do not need as many parts as production cycles. We have only one part with endless amount of appearances of one instance generated during a simulation.

We can generate appearances, for example, by Material Flow Operation or manually in Operation tree on a selected operation with associated part to it. Then we have to ensure that the appearance will stay visible as long as required. The appearance is hold visible, for example, during the defined material flow. Definition of the whole material flow is done in Material Flow Viewer in Process Simulate. To save the initial position of an appearance, we have to delete it in required locations and generated it again.

4.2.2.2 Signals

Event-based simulations are controlled by signals. Based on it each operation or event can be driven. Process Simulate operates with four basic signals:

- Key signal - It is a signal defined only for Process Simulate and its simulations (user input to Process Simulate)
- Display signal - It is a signal which displays information
- Resource Input Signal - It is an input signal from a viewpoint of a Programmable Logic Controller (PLC). For example, if an operation has ended it will send a resource input signal to PLC, operation ended .
- Resource Output Signal - It is an output signal from a viewpoint of a Programmable Logic Controller (PLC). For example, if we want to run an operation in Process Simulate we will send a resource output signal to operation, operation starts .

The list of all signals is visible in Signal Viewer of Process Simulate, see figure 4.9. We can create new signals, edit them, filter them, define a connection to PLC via OPC server (mapping them). Then Simulation Panel tool serves for monitoring of chosen signals from Signal Viewer.

Signal Name	Memory	Type	Address	IEC Format	PLC Connection	OPC Connection	Resource
CarOnFrame_to_WorkSpace1_start	<input type="checkbox"/>	BOOL	No Address	No Address	<input checked="" type="checkbox"/>	local/OPC.SimaticNE	
CarOnFrame_to_WorkSpace2_start	<input type="checkbox"/>	BOOL	No Address	No Address	<input checked="" type="checkbox"/>	local/OPC.SimaticNE	
AssembledCarParts_to_Workspace3_start	<input type="checkbox"/>	BOOL	No Address	No Address	<input checked="" type="checkbox"/>	local/OPC.SimaticNE	
AssembledCarParts_to_Out_start	<input type="checkbox"/>	BOOL	No Address	No Address	<input checked="" type="checkbox"/>	local/OPC.SimaticNE	
KukaKR350-2-N1_Weld_Op_start	<input type="checkbox"/>	BOOL	No Address	No Address	<input type="checkbox"/>	local/OPC.SimaticNE	● KukaKR350-2-N1
KukaKR350-2-N3_Weld_Op_start	<input type="checkbox"/>	BOOL	No Address	No Address	<input type="checkbox"/>	local/OPC.SimaticNE	● KukaKR350-2-N3
KukaKR350-2-N6_Weld_Op_start	<input type="checkbox"/>	BOOL	No Address	No Address	<input type="checkbox"/>	local/OPC.SimaticNE	● KukaKR350-2-N6
KukaKR350-2-N8_Weld_Op_start	<input type="checkbox"/>	BOOL	No Address	No Address	<input type="checkbox"/>	local/OPC.SimaticNE	● KukaKR350-2-N8
StartSimulation	<input type="checkbox"/>	BOOL	No Address	No Address	<input checked="" type="checkbox"/>	local/OPC.SimaticNE	● StartSimulation
LightsControlWS1-HOME	<input type="checkbox"/>	BOOL	No Address	No Address	<input checked="" type="checkbox"/>	local/OPC.SimaticNE	● LightsWS1
LightsControlWS1-RUN	<input type="checkbox"/>	BOOL	No Address	No Address	<input checked="" type="checkbox"/>	local/OPC.SimaticNE	● LightsWS1
LightsControlWS1-READY	<input type="checkbox"/>	BOOL	No Address	No Address	<input checked="" type="checkbox"/>	local/OPC.SimaticNE	● LightsWS1
LightsControlWS1-FAULT	<input type="checkbox"/>	BOOL	No Address	No Address	<input checked="" type="checkbox"/>	local/OPC.SimaticNE	● LightsWS1

Figure 4.9: Signal Viewer - Example

Each operation in Operation tree has default defined the resource input signal (the end signal). We can also easily generate the resource output signal (the trigger signal of an operation) in Signal Generation menu of CEE. There are possibilities to generate trigger signals of robot, device, material flow and non-sim operations.

Moreover there is a possibility to generate device operations/signals automatically. So we can define operations or only signals of a given device. During this process there are defined pose signals, moving operations between poses and pose sensors for selected poses. If we choose only to generate signals, no operation is created in Operation tree and we can still control this device in an event-based simulation. In this way I defined

for example light stacks, weld guns, gripper and other devices which were not included in time-based simulation.

4.2.2.3 Sensors

By using sensors we are one step closer to the real factory environment. Creation of new sensors is located in CEE menu of Process Simulate. In Process Simulate there are five basic sensors in CEE menu:

- Joint value sensor - it allows to create sensor for a device or a robot. If joints of robot (or device) reaches a set value (or defined pose) sensor signal is activated. Set values can be accurate values, range of values and so on. It automatically creates a sensor signal with BOOLEAN values (true and false) only by definition of a sensor.
- Joint distance sensor - it enables on-line monitoring of a device joint. It has INT, DINT and REAL values.
- Property sensor - it enables to detect property of parts during simulation, such as temperature, color, barcode and so one. To allocate properties on part we use Property Projector. The property sensor signal has INT, DINT and REAL data type values.
- Proximity sensor - it enables to detect a mutual distance between two objects. Sensor has BOOLEAN values true and false. It is activated when defined object enter to certain set distance of a proximity sensor. Proximity sensor does not have a graphical representation and it is located to selected resource (object) in Process Simulate.
- Photoelectric sensor - it enables to monitor crossing of beam by objects. It has BOOLEAN values true and false. Photoelectric sensors consists of lens and beam. We can define diameter and width of lens and length of beam. To see defined beam in Graphical View of Process Simulate we have to use Display Detection Zone command in CEE menu. To work properly the sensor is activated (Activate Sensor in CEE menu) and also its signal has be used in logic (PLC or logic block). On the other hand it does not work and we cannot monitor its signal (sensor is inactive).

Sensors generate signals of defined data type only by definition of a sensor automatically.

4.2.2.4 Logic blocks

Logic blocks enable to process signals which come into input logic gateways called Entries. Afterwards logic block creates a combination of logic from the definition of Parameters, Constants and its default logic elements. After processing it sends the defined signals out from the output logic gateways called Exits, see figure 4.10 . Default logic elements are:

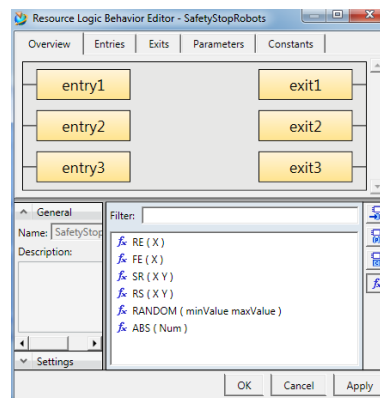


Figure 4.10: Logick Block - Example

- Raise Edge (RE) function - it has BOOLEAN value. It activates true value when the input signals changes from 0 to 1.
- Falling Edge (FE) function - it has BOOLEAN value. It activates true value when the input signals changes from 1 to 0.
- Set Reset (SR) function - it has BOOLEAN value and it requires two inputs (S and R). It activates true in output of SR element when S is 1 and R is 0. After it holds output activated as long as R is not activated. When both S and R are set to 1 output is set to 1.
- Reset Set (RS) function - it has BOOLEAN value and it is similar to SR function. There is only one exception that if S and R are set to 1 output is set to 0.
- Absolute Value Function (ABS) - its output is an absolute value of a number
- Random Number function (RAN) - it requires Min and Max values of generated random numbers

- User-defined functions - Process Simulate also enables to program logic of logic block in user-defined function interface. Logic can be written in programming languages (C#,etc.) and converted into *.dll file.

4.2.2.5 OLP programing

Robots can communicate with other robots, devices and PLC's. To be closer reality as much as possible Process Simulate enables to use software robotic controllers from manufacturer of selected robot. It has been already mentioned before in section 3.4.5 also with setting of a robot controller and creating of a robot program. A general default controllers for all robots in Process Simulate has accuracy around 80 percent.

The default controller has default list of robot Offline Programming Signals (OLP signals) which can be generated in Robot Signals viewer. It contains output signals Q from viewpoint of PLC, such as *startProgram* and input signals from viewpoint of PLC, such as *programEnded*. OLP signals are during creation also at the same time mapped to PLC signals. Analogically we can create new OLP signals and mapped them to PLC signals, see figure 4.11. Then OLP signals are used in logic of robotic programs according to the logic of chosen robotic controller. Robot programs are edited in Path Editor tool in Process Simulate.

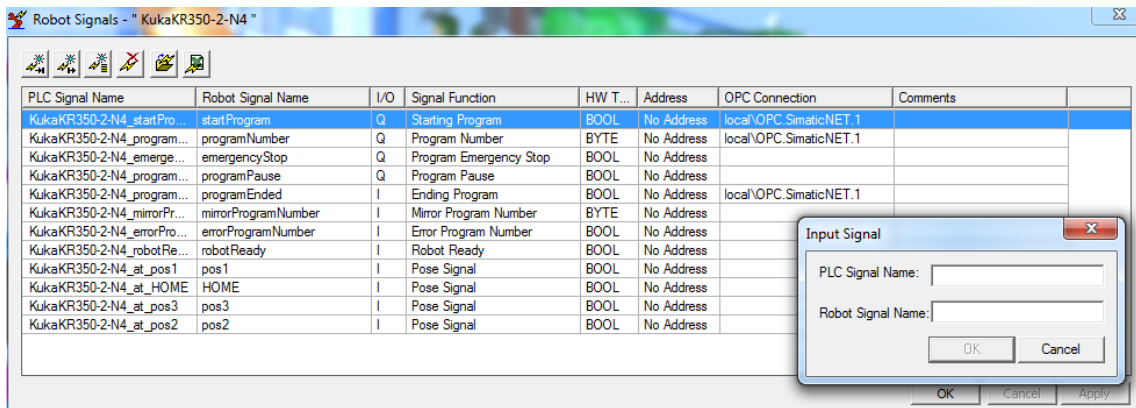


Figure 4.11: Robot signals

Default controller signals can be used in a following way. There are preconditions that path numbers have to be defined in Path Editor of selected program and also program has to be set as default in Robotic Program Inventory. I choose a robotic program from PLC (*programNumber*). Program number is mirrored back to PLC (*mirrorProgramNumber* and *errorProgramNumber*). If robot is ready (*robotReady*) its operation can start. PLC sends start program order (*startProgram*) signal. After ending of chosen robot program

robot sends signal to PLC (*programEnded*). There are also safety PLC signals, such as *emergencyStop* and *programPause*.

During robot programs operation there can be defined many others OLP commands with different functionality, such as synchronization (Send Signal, Set Signal, Wait Signal, Wait Time), tool handling (mount gun, unmount gun, wait device, go to state), macros and so on. See figure 4.12.

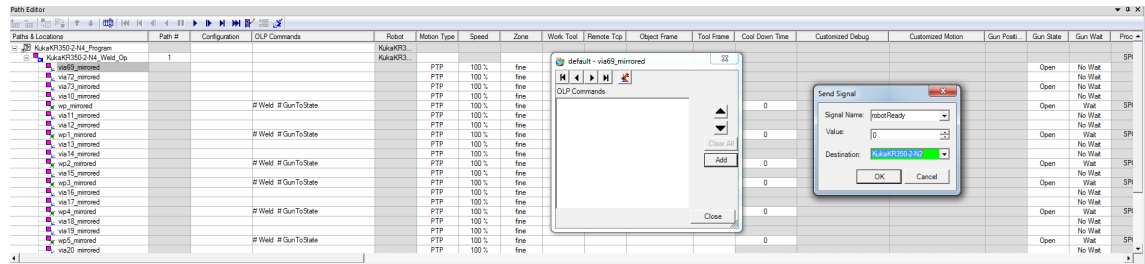


Figure 4.12: OLP signals - Example of using

4.2.2.6 Connection to PLC

In Process Simulate we can use emulation of PLC or real PLC, see figure 4.13. Emulation of PLC contains SIMIT and PLCSIM platform. PLCSIM does not need to create OPC connection and it is connected directly via COM-interface of Process Simulate. Process Simulate acts as an OPC client if we choose the OPC or SIMIT connection. See figure 4.14.

Object Linking and Embedding (OLE) technology enables linking objects. Based on this technology there was defined a special standard for automation industry called OLE for Process Control (OPC) standard. The OPC standard specifies a communication between arbitrary PLCs of different manufacturer and OPC server. According to [17] OPC Data Access specification provides access to read and write real-time data.

To get real-time data we connect to OPC server (I use Simatic.NET OPC server) as OPC clients. Therefore process of reading and writing is not real time on a client side. In client/server architecture OPC client is not synchronized in real time with PLC cycles. On the other hand this delays are insignificant (hundreds of milliseconds) in comparison with advantages of Virtual Commissioning.

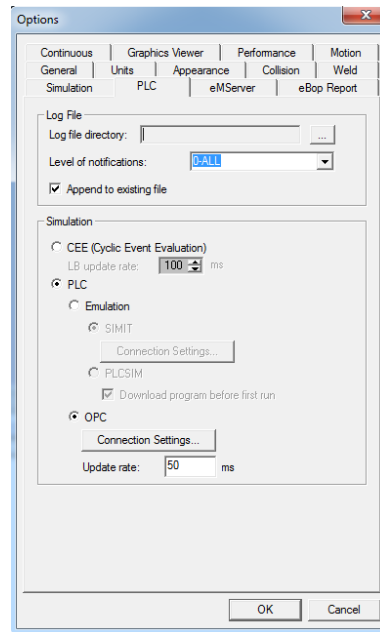


Figure 4.13: Settings of CEE simulation/OPC simulation

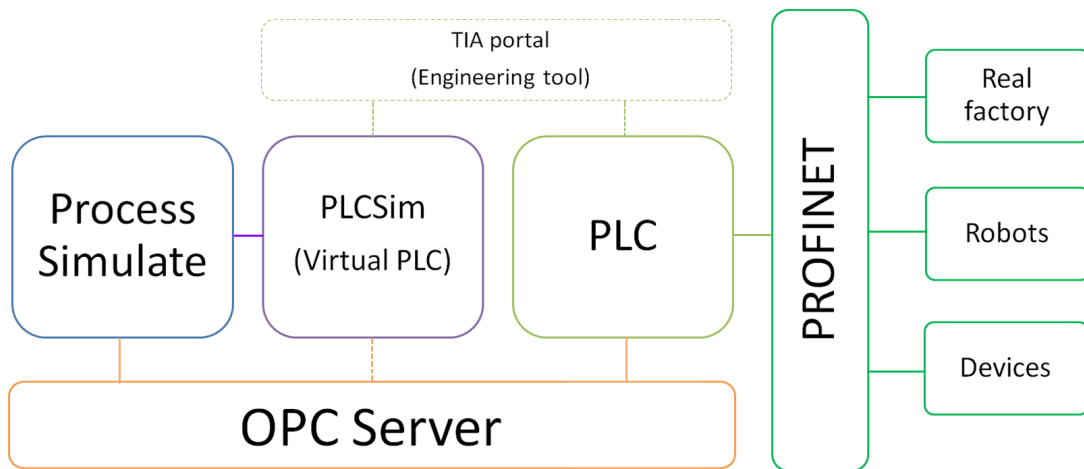


Figure 4.14: Virtual Commissioning - Connection to PLC

To better time description and synchronization there was developed SIMIT platform of OPC connection which brings many advantages. SIMIT acts as OPC server and it is synchronized with OPC simulation of Process Simulate (Process Simulate). Simit waits for OPC simulation before continuing to next step. It means that all required actions in a given virtual time finished before continuing.

This problem could be solved in future versions of Tecnomatix by adding of SIMBAPro interface which enables to connect simulations directly to real-time industry network

(PROFINET).

4.2.3 Implementation of Virtual Commissioning

In this thesis I created Virtual Commissioning based on the rough time-based simulation which was created in section 4.1. To work with OPC Simulation it was necessary to configure OPC server with PLC and connect Process Simulate as an OPC client. After I created material flow to generate appearances. I also generated signals of all operations and devices/robots contained in simulation (output (start) signals of flow operations, output/input signals of devices, robot signals). Later I defined sensors in my welding line. I also defined and set robotic programs in controller for robots N1-N9 and KUKA KR5 arc. At the end I created the PLC program in Sequential Functional Chart which controls simulation of production processes

4.2.3.1 OPC settings

To create OPC connection between PLC and Process Simulate there is necessary to follow several steps. I did it in following way:

- Installation of Totally Integrated Automation portal. It contains Step7 to configuration of PLCs.
- Installation of Simatic Manager. It contains modules for creating of the OPC server in Step7.
- Hardware and software configuration of the PLC in Step7.
- Hardware and software configuration of SIMATIC PC Station in Step7.
- Creating of S7 connection between the PLC and the OPC server in Step7. See figure 4.15.
- Downloading of the PLC configuration into the PLC in Step7.
- Downloading of the OPC server configuration into Station Configuration Editor (it triggers OPC server with defined configuration from created **.xdb* file)
- Creating of the OPC connection into Process Simulate (login to the OPC server as a client). See figure 4.16.

- Definition and mapping of signals connected to PLC in Process Simulate (Signal Viewer).

In case that signals change on PLC side we have to refresh configuration of the PLC and the OPC server. It means that we have to compile configuration of PLC and OPC server again and download it to them.

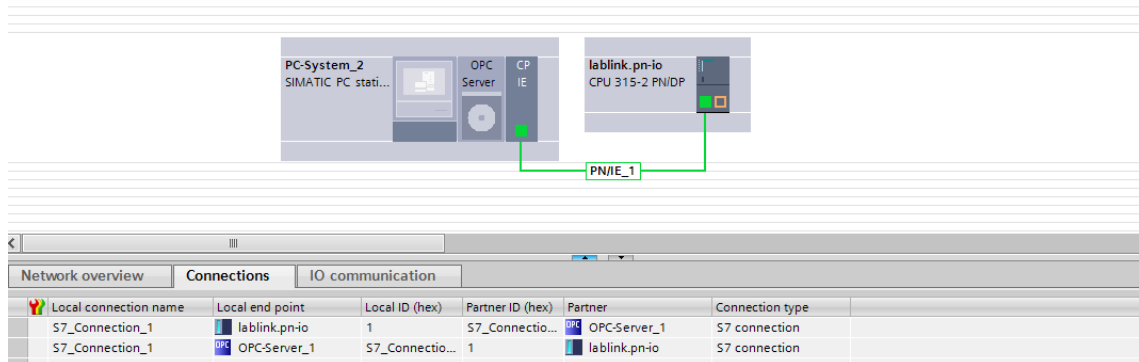


Figure 4.15: Creating of S7 connection in TIA portal

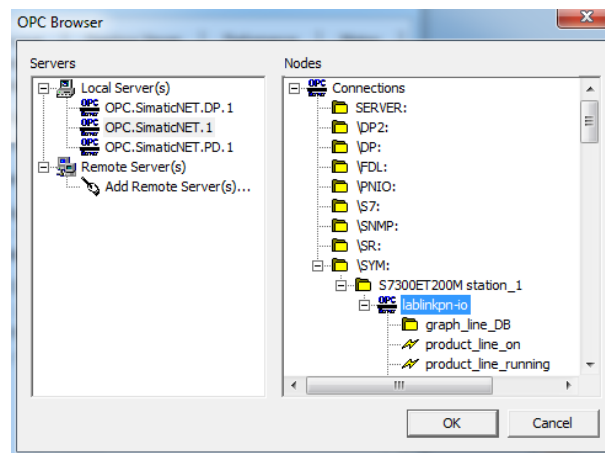


Figure 4.16: Selecting of a reachable OPC server in Process Simulate

4.2.3.2 Generation of appearances

Appearance approach has a limitation in event-based simulations. Simulations must be controlled with consideration of defined material flow. It means that control sequence must be strictly complied according to sequence of the defined material flow. To generate appearances I defined material flow in Material Flow Viewer according to description of the modeled system in section 2.4.1. See figure 4.17.



Figure 4.17: Defined material flow

4.2.3.3 Adding of sensors

In my graphical layout I added four pairs of photoelectric sensors on conveyor to check position of cars. They are active when the frame of a car flows across them. If both signals of a defined pair are active it means that a skeleton of a car on this frame is in required position of robotic work cell in Workspace 1, 2 or 3. The last pair of signals serves to recognition that we can turn off a signal which triggers material flow operation (conveyor) from Workspace 3 out. Designed placement of sensors is visible in figure 4.18.

I also worked with joint values sensors but their were defined automatically with their defined poses (Generate Device Operations/Signals tool).

4.2.3.4 Overview of signals

During development of my thesis I created around 300 signals. In final stage I worked with up to 220 signals.

In Simulation Panel I sorted chosen signals into groups - Car Flow, Car Position, Robot Programs, Welding, Welding Guns, Grippers, Pick and Place, Light Stacks and Safety Stop. The group of Robot Programs was subdivided into Workspace 1, 2 and

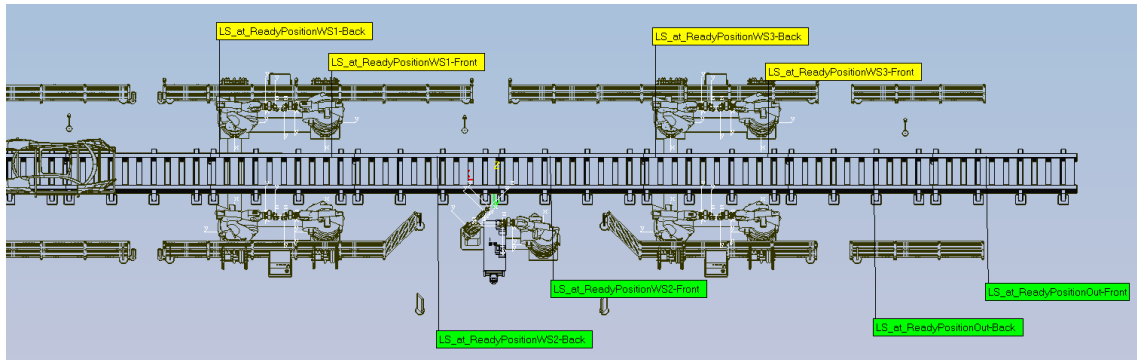


Figure 4.18: Designed photoelectric sensors

3 and Light Stack into Central Control WS1, WS2 and WS3. In this section I try to describe some important signals of this simulation.

In PLC logic there were also defined signals which are not defined in Process Simulate, such as an input signal StopCarButton. If StopCarButton is active my welding line stops to accept new cars to Workspace 1. After resetting of this signal cars start to flow to Workspace 1 again.

The largest group of signals contains signals of Light Stacks. Light signals of each Workspace (WS) covers 40 signals. From this reason I created special logic block which enables to control all lights in one workspace with 4 signals, see table 4.4.

Signal	I/O	Data type
LightsControlWS1-HOME	O	BOOL
LightsControlWS1-RUN	O	BOOL
LightsControlWS1-READY	O	BOOL
LightsControlWS1-FAULT	O	BOOL

Table 4.4: Signals - Central control of light stacks

Signals of photoelectric sensors are defined in table 4.5.

Really important group is the Car Flow group which defines output signals of car flow operations. See table 4.6.

Signal	I/O	Data type
LS_at_ReadyPositionWS1_Back	I	BOOL
LS_at_ReadyPositionWS1_Front	I	BOOL
LS_at_ReadyPositionWS2_Back	I	BOOL
LS_at_ReadyPositionWS2_Front	I	BOOL
LS_at_ReadyPositionWS3_Back	I	BOOL
LS_at_ReadyPositionWS3_Front	I	BOOL
LS_at_ReadyPositionOut_Back	I	BOOL
LS_at_ReadyPositionOut_Front	I	BOOL

Table 4.5: Signals - Light sensors

Signal	I/O	Data type
CarOnFrame_to_WorkSpace1_start	O	BOOL
CarOnFrame_to_WorkSpace2_start	O	BOOL
AssembledCarParts_to_Workspace3_start	O	BOOL
AssembledCarParts_to_Out_start	O	BOOL

Table 4.6: Signals - Car flow signals

Signal	I/O	Data type
KukaKR350-2-N1_startProgram	O	BOOL
KukaKR350-2-N2_startProgram	O	BOOL
KukaKR350-2-N3_startProgram	O	BOOL
KukaKR350-2-N4_startProgram	O	BOOL
KukaKR350-2-N1_programNumber	O	BOOL
KukaKR350-2-N2_programNumber	O	BOOL
KukaKR350-2-N3_programNumber	O	BOOL
KukaKR350-2-N4_programNumber	O	BOOL
KukaKR350-2-N1_programEnded	I	BOOL
KukaKR350-2-N2_programEnded	I	BOOL
KukaKR350-2-N3_programEnded	I	BOOL
KukaKR350-2-N4_programEnded	I	BOOL
KukaKR5arc1_startProgram	O	BOOL
KukaKR5arc1_programNumber	O	BOOL
KukaKR5arc1_programEnded	I	BOOL

Table 4.7: Signals - PLC communication with robot controllers

Robot program group contains signals which PLC uses to communicate with robot controller. In my case PLC sets concrete number of the robotic program in robotic controller and triggers it. After ending of selected robotic program robot controller sends input signal to PLC that program ended. This type of communication with simulated robot controller is equivalent to communication with a real robot. See table 4.7.

4.2.3.5 Robot simulation

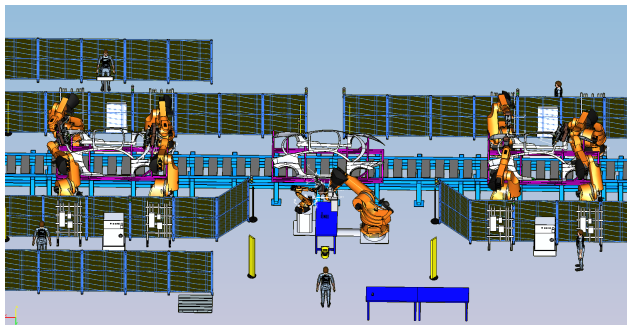
A precondition has to be satisfied to trigger robot programs. Robot program has to be set as default in robot controller and each operation within it has set its path number (program number). I defined following programs, see table 4.8.

Robot	Operation	Path number
KukaKR350-2-N1	KukaKR350-2-N1_Weld.Op	1
KukaKR350-2-N2	KukaKR350-2-N2_Weld.Op	1
KukaKR350-2-N3	KukaKR350-2-N3_Weld.Op	1
KukaKR350-2-N4	KukaKR350-2-N4_Weld.Op	1
KukaKR350-2-N5	KukaKR350-2-N5_Weld.Op	1
KukaKR350-2-N6	KukaKR350-2-N6_Weld.Op	1
KukaKR350-2-N7	KukaKR350-2-N7_Weld.Op	1
KukaKR350-2-N8	KukaKR350-2-N8_Weld.Op	1
KukaKR350-2-N9	KukaKR350-2-N9_Weld.Op	1
Kuka KR5 arc	KukaKR5arc1_PAPToMarker	1
Kuka KR5 arc	KukaKR5arc1_PAPToCar	2

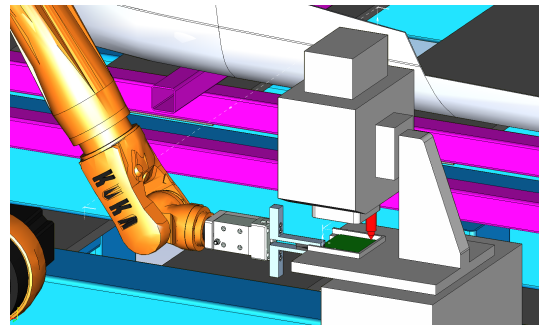
Table 4.8: Robot programs

4.2.3.6 SFC program

After definition of signals of the whole product line there were developed Sequential Functional Chart defined in section 2.4.2. This program runs in PLC and controls the whole logic of the production line. I created several views of this simulation. See figures 4.19(a), 4.19(b), 4.20(a), 4.20(b), 4.21(a), 4.21(b).

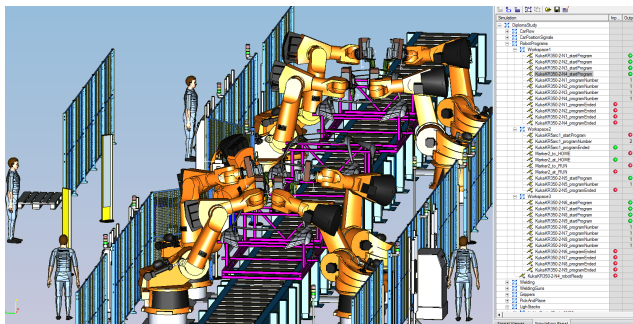


(a) View 1

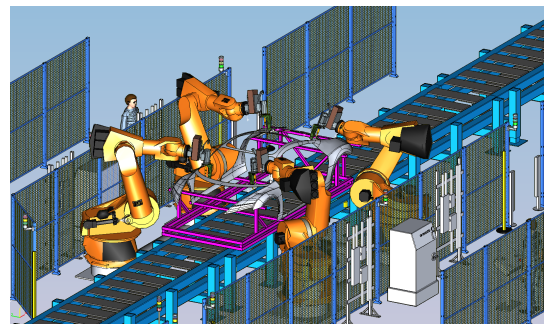


(b) View2

Figure 4.19: Whole product line

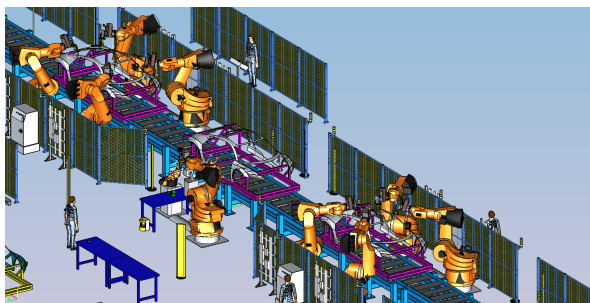


(a) View 3

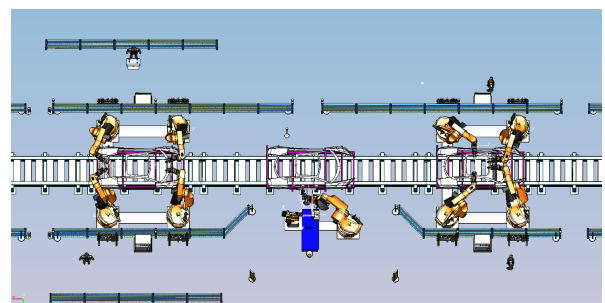


(b) View 4

Figure 4.20: Whole product line



(a) View 5



(b) View 6

Figure 4.21: Whole product line

Chapter 5

KUKA robot

5.1 KUKA KR5 arc

5.2 Description

The robot KUKA KR5 arc is a small-payload robot (5 kg). Its configuration consists of robot itself, connecting cables, a control system, software equipment and a manual teach pendant (SmartPad), see figure 5.1. We can also add additional axes to robot controller, for example a linear robot unit to extension the operational and manipulating space of the robot. Description of the robot in this chapter is based on [18]

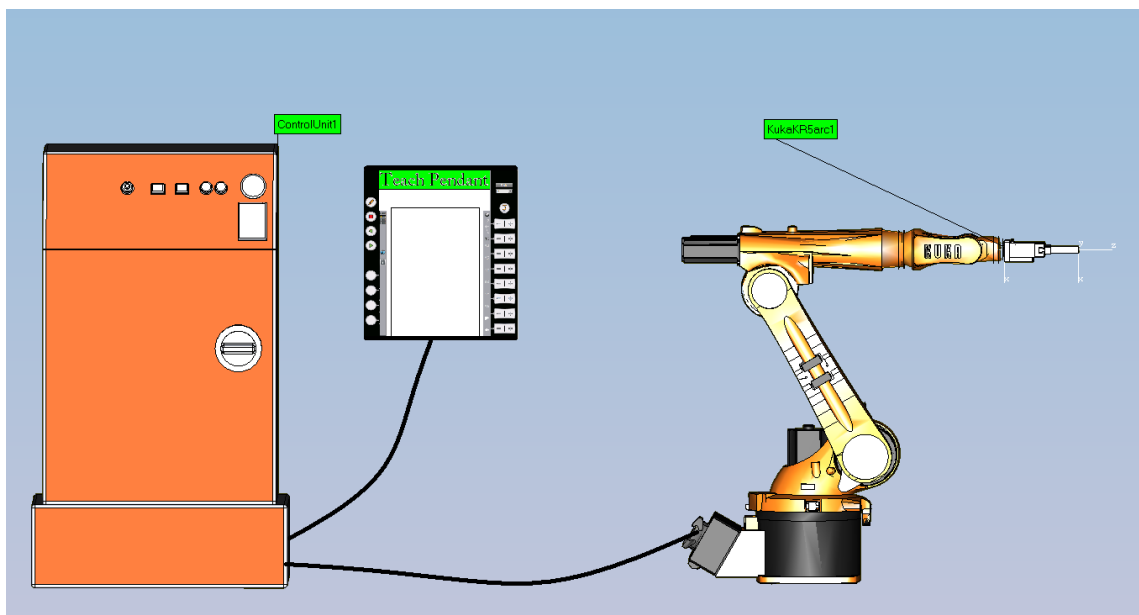


Figure 5.1: Configuration of KUKA Kr5 arc

5.2.1 Robotic assembly

. Robot itself is designed as a 6-axes jointed kinematic system. It consists of In-line wrist, Arm, Link arm, Rotating column, Base frame and Electrical installations. See figure 5.2.

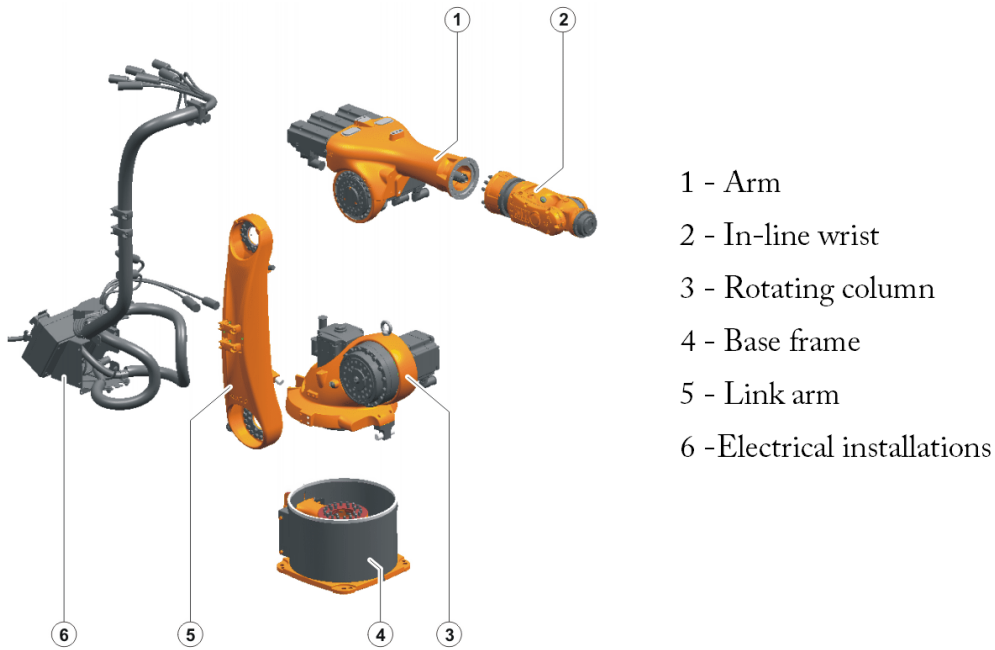


Figure 5.2: Main assemblies of the robot [18]

5.2.2 Technical data

Basic technical data are visible in table 5.1.

Number of axes	6
Volume of working envelope	8.4 m^3
Pose repeatability	0.04 mm
Weight	127 kg
Sound level	$< 75 \text{ dB}$
Max. total load	37 kg
Supplementary load, arm	12 kg

Table 5.1: Technical data of the robot KUKA KR5 arc

5.2.3 Axis data

KUKA KR5 arc has following axis data, see table 5.2

Axis	Range of motion	Speed with rated payload
1	+/-155°	154°/s
2	+65° to -180°	154°/s
3	+/-158° to -15°	228°/s
4	+/-350°	343°/s
5	+/-130°	384°/s
6	+/-350°	721°/s

Table 5.2: Axis data of the robot KUKA KR5 arc

5.2.4 Robotic Controller KRC4

In this section I used some information from specification [19]. KUKA Robotic Controller 4 (KRC4) contains Control PC, see figure 5.3. It has 3 basic Ethernet ports which are marked:

- KCB (KUKA Controller Bus) - it is a main control bus of the whole controller
- KLI (KUKA Line Interface) - connection to higher-level control infrastructure (PLC via PROFINET, etc.)
- KSB (KUKA System Bus) - internal KUKA bus for internal networking of the controllers with each other

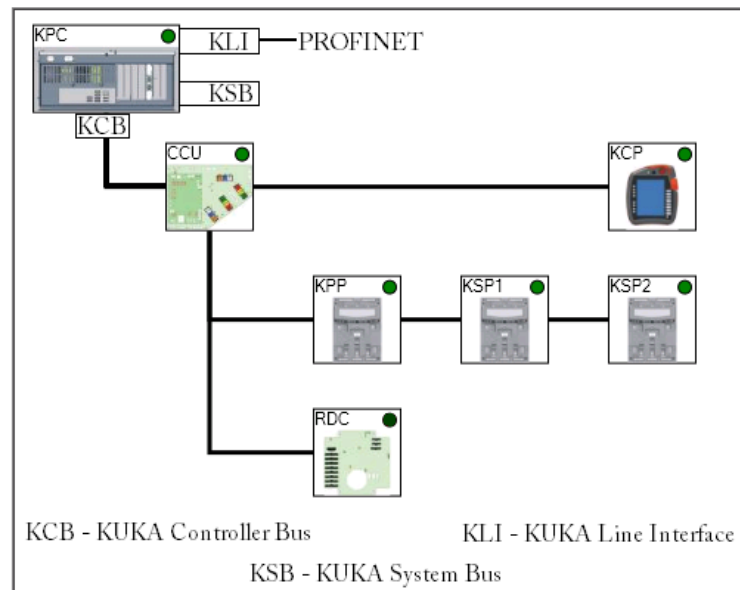


Figure 5.3: Hardware configuration of the KUKA Robotic Controller 4 (KRC4)

KUKA Controller Bus is connected to Cabinet Control Unit (CCU) which is the central power distributor and communication interface for all components of the robot controller. The CCU consists of the Cabinet Interface Board (CIB) and Power Management Board (PMB). Into the CCU there are connected:

- KUKA Power Pack (KPP) - it drives power supply with drive controller
- KUKA Control Panel (KCP) - it connects teach pendant (SmartPad) into KRC4 controller
- Resolver Digital Convertor (RDC) - it is used to detect the motor position data.

Behind the KPP there are connected:

- KUKA Servo Pack 1 (KSP1) - Drive controller for axes 4 to 6
- KUKA Servo Pack 2 (KSP2) - Drive controller for axes 1 to 3

5.3 Configuration of the robot

WorkVisual is an engineering tool to configure hardware and software of a KUKA robot controller. I personally worked with WorkVisual 2.4 . It enables to configure

buses, safety control, mapping of signals, etc. It also contains programming environment for the KUKA Robotic Language(KRL).

5.3.1 KCB configuration

After creating of a new (KRC4) project we have to define its firmware version and amount of exchanging I/Os. To activate chosen controller we have to set it as active (a project can contains more controllers). Then we add new Bus structure from DTM selection (KCB structure). If DTM selection is empty we have to close the project and add new packages to it in DTM Catalog Management. After definition of KCB structure we have to add all elements connected to it in box of KRC4 controller and defined their topology in KCB settings, see figure 5.4.

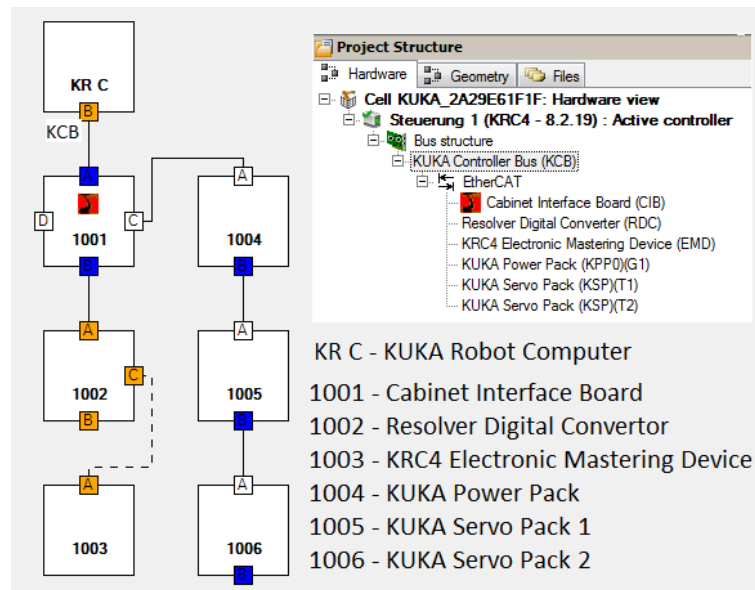


Figure 5.4: WorkVisual - Kuka Controller Bus settings

5.3.2 KLI configuration - PROFINET

PROFINET is the industrial standard which is based on Ethernet . It enables to exchange IO data in the real time. PROFINET consists of elements, such as a controller, devices and supervisors. The robot can behaves as a PROFINET controller or device. To be able to connect KLI interface into PROFINET IO network we have to install PROFINET package into the robot controller. It contains PROFINET IO Controller,

Device and ProfiSafe Device. Those PROFINET package supports only PROFINET IO Class A.

To configure the KUKA Line Interface as a PROFINET interface we have to add PROFINET into Bus structure from DTM selection in WorkVisual. After definition we open its communication settings, see figure 5.5. Robot controller is set as PROFINET device if Activate PROFINET device stack is marked. Next we choose a number of exchanged bits with PROFINET IO network. Equivalently the same thing is done later during configuration of the PROFINET network in Step7 (robot looks like device in Step7 and the number of exchanged bits is added like a module). We say to PROFINET controller how many bites will be exchanged with robot device. This settings automatically maps allocated number of bits into allocated bit on PROFINET network. After we choose the robot controller version and it bus cycle time.

After basic configuration of the robot into PROFINET network we have to map signals of the robot controller into signals of other devices or controller. See figure 5.6.

Communication settings | PROFINET | Device settings | Device Diagnostic

Network adapter: <undefined>

PROFINET device

☒ Activate PROFINET device stack

Number of device I/Os: 128

Device name: dce-KR5arc

Profinet version: v8.2

Bus cycle time: 8 ms

Bus timeout: 2000 ms

☐ Display diagnostic alarm as message

PROFINET controller

Bus cycle time: 8 ms

Bus timeout: 2000 ms

Figure 5.5: WorkVisual - Settings of PROFINET

Name	Type	Description	I/O	I/O	Name	Type	Address
SIN[1]	BOOL		←	→	01.01.0001 Input	BOOL	16.1
SIN[2]	BOOL		←	→	01.01.0002 Input	BOOL	2.0
SIN[3]	BOOL		←	→	01.01.0003 Input	BOOL	2.1
SIN[4]	BOOL		←	→	01.01.0004 Input	BOOL	2.2
			←	→		BOOL	2.3

1 bit(s) in 1 signal(s) selected

Figure 5.6: WorkVisual - Mapping of KUKA Controller signals into PROFINET network

When we satisfy above mentioned preconditions we can download this configuration

into the robot. Then we import the GSDML file of the robot (located on CD of WorkVisual) into Step7 and we configure PROFINET network in standard way.

5.3.3 Mapping of variables for Automatic External mode

Robot controller contains variables which are declared for communication with supervisor PLC (PGNO_VALID, \$MOVE_ENABLE, etc.). To be able to monitor these values in PROFINET network we have to map them into the range of *IN* and *OUT* which are allocated in PROFINET bits. To permanent settings *OUT* values of robot controller into log1 or log 0. We can use a value *OUT*[1025], which equals to log1 and *OUT*[1025] which equals to log0. The communication sequence with PLC and robot controller is described in KUKA System Software 8.2 documentation [20].

5.4 Commissioning of robot

A PLC can control a robot in its Automatic External mode. In this mode PLC acts as supervisor. The robotic program created in Process Simulate is downloaded directly to robot. Robotic program (*.src) and its locations (*.data) are downloaded under Program File folder, see figure 5.7. The definition of a tool and base frame of a robot is changed in System file folder in *config.dat* according to instructions in *.src file. In my case it includes a definition of the BASE (data, name and type), TOOL (data, name, type) and LOAD data.

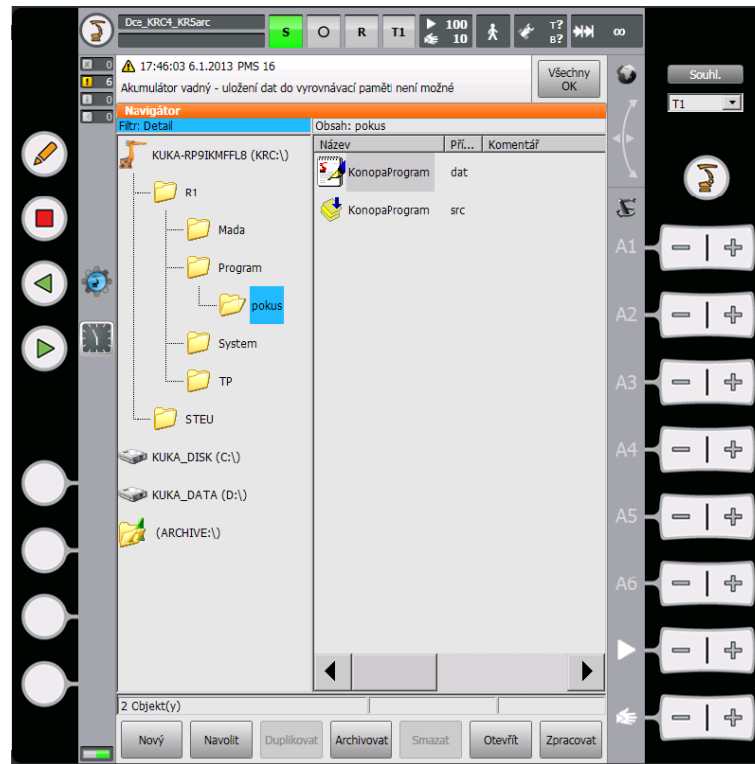


Figure 5.7: Downloaded robotic program to the KRC4 controller

There are some preconditions which PLC has to satisfied before the robotic program runs. In AUT EXT mode robot must define its cell.src program. The cell program assigns program numbers to defined robotic programs. It is similar to definition of paths in Process Simulate.

Then cell.src program must be activated in KUKA controller. This is equal to set on default robotic program in Process Simulate.

Next precondition is proper sequence of exchanging signals between the PLC and robot controller. This was precisely described in following diploma thesis [21]. Author also defined PLC program to trigger robotic programs in robotic controller. In my work I used this program to control robotic program of the KUKA KR5 arc. See figure 5.8.

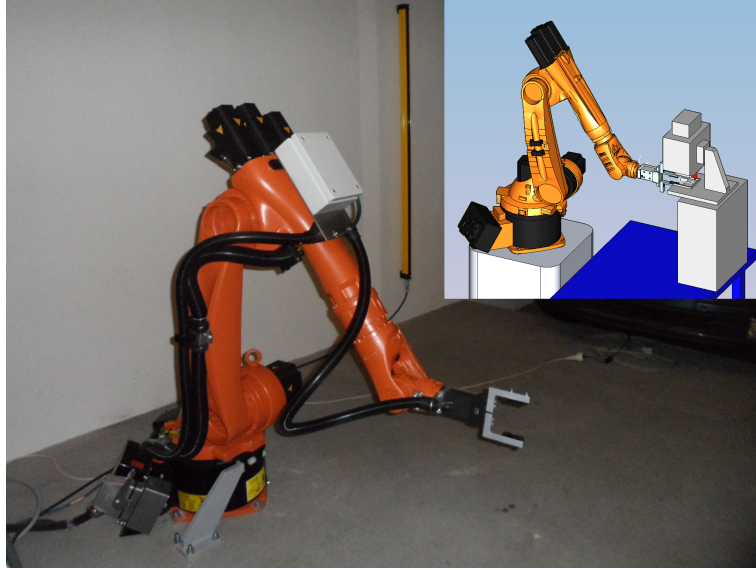


Figure 5.8: Commissioning of the real robot

The virtual robot in simulation and the real robot in the real environment do not have equal time. The real robot runs in real time whereas virtual simulation time can be for example ten times faster. To synchronize the virtual and real robot KUKA KR5 arc in simulation I created several WAIT and SET signals. If the virtual robot moves to position 1 it sets signal *virtualRobotInPosition1* and waits for signal *realRobotInPosition1* which is set by the real robot. Therefore faster robot waits for the slower one.

Chapter 6

Conclusion

During implementation of the project I used many different technologies. I designed a concept of the digital factory and its utilization also with the real robot KUKA KR5 arc. It is a welding line for a car body, consisting of three workspaces connected with a conveyor, which transports the car. This concept consists of 3 Workspaces. In Workspace 1 robotic cell of four KUKA KR350-2 robots welds a car body, exactly on the back glass frame and the front car frame. In Workspace 2 robotic cells contains one marker, a manipulating robot KUKA KR5 arc and one welding KUKA KR350-2 robot. KUKA KR5 arc picks a part from table, places it into the marker and holds it there. Marker device marks a plate and KUKA KR5 arc places the part on the car body. After the welding robot KUKA KR350-2 welds it to the car. In the third Workspace robotic cell of four KUKA KR350-2 robots welds a roof of the car body. This concepts also includes safety elements. I designed light stacks, safety fences, light scanner and light curtains.

To be able to implement this concept in simulation software I studied the possibilities and functionality of Process Designer and Process Simulate within the Tecnomatix package. I discovered how Process Designer and Process Simulate are connected. They have equal certain functionality which is given by its architecture. Both of them behaves as clients of the eMServer, which is a mediator between Oracle Database and API of Process Designer and Process Simulate. Oracle Database serves to saving data of simulations. During this phase I met great absence of study materials and references, because this software is used especially in large companies and I have been pioneer at the Faculty of Electrical Engineering at Czech Technical University in Prague. Process Simulate nor Process Designer do not contain any 3D models and most of them were created for the purpose of this work (with the exception of robots).

Process Designer is a tool which allocates products, resources and operations. It also

enables to plan operations from the viewpoint of a raw planning. On the other hand Process Simulate utilizes detail operations and simulations. It provides two different groups of simulations: Time-based and event-based simulation. Event-based simulations are divided into Cyclic Event Evaluator simulations and Virtual Commissioning in Process Simulate. Virtual Commissioning is the one which enables connection of simulation into the real word via Programmable Logic Controller. I decided to utilize this connection into the external PLC program.

Based on designed concept of the production line I created three levels of simulations in Process Simulate: Time-based simulation, Virtual Commissioning without the real robot KUKA KR5 arc and Virtual Commissioning with the real robot KUKA KR5 arc. The simplest one is the time-based simulation. Due to its possibility to connect it to the higher level of simulations (event based simulations) it was suitable to design operations in this mode.

To create Virtual Commissioning I designed a material flow in the product line. It is necessary to define visibility of appearances (parts) in event-based simulations. To work with logic I generated signals of operations, devices, robot controllers and device sensors (joint sensors). I also defined and placed new light sensors into product line. Sensors are activated when a car body cross the light beam. In event-based simulation I defined a new approach to operations of robot. In time-based simulation I controlled robots based on robotic operations. In event-based simulations I controlled robot based on communication with its controller. In this phase I defined up to 220 of I/O signals. To create connection between PLC and Process Simulate I defined Simatic.NET OPC server.

Therefore I was able to connect simulation to an external controller (PLC). The control algorithm implementation, running in the external controller, was created in Sequential Function Chart, which was also used to specify the control behavior. Based on the concept of the digital factory I connected the external controller, which controls the defined simulation in Virtual Commissioning mode of Process Simulate, via OPC connections..

At the end I configured a real KUKA robot KR5 arc and connected them into the PROFINET industrial network. Due that I was able to connect robot to PLC control program. After that I download a robot program from the virtual robot in Process Simulate into the real robot. Then I created parallel branches to the virtual program KUKA KR5 arc in SFC diagram, which enables to trigger both robots simultaneously during the simulation.

I did not verify the implemented simulation against the formal description. Simulation in Process Simulate does not run in the real time. Therefore movements of the real and virtual robot are not synchronized. To better synchronization of Process Simulate and PLC there exists SIMIT platform. It would also be possible to use SIMBApro as an emulator of an external PROFINET network, but at the time of working on this thesis, no software interface to Tecnomatix was available. However, for real-life applications using SIMBApro is recommended because it covers most of the time-dependent behavior of the system.

Bibliography

- [1] Štěpán Kuchař. Modelování podnikových procesů, 2012.
- [2] Lucie Pekárková. Techniky modelování a optimalizace podnikových procesů, 2007.
- [3] Wikipedia. Formal methods — wikipedia, the free encyclopedia, 2012. [Online; accessed 1-January-2013].
- [4] Sunway E-Systems. Tecnomatix, 2010.
- [5] Karel Carvan. Tecnomatix Úvodní webinář, 2012.
- [6] Bc. Jiří Kopenc. Simulace vybrané výrobní linky klimatizace jako komponenta digitální továrny, 2011.
- [7] totalqualitymanagement.wordpress.com. What is 1-10-100 rule?, 2009.
- [8] Simens Industry Software Inc. Tecnomatix Technologies Ltd. Tecnomatix 10.1 administration guide, 2012.
- [9] Nina Sundström. Diploma thesis: Automatic generation of operations for the flexa production system, 2010.
- [10] Wikipedia. Program evaluation and review technique — wikipedia, the free encyclopedia, 2012. [Online; accessed 1-January-2013].
- [11] Simens Industry Software Inc. Tecnomatix Technologies Ltd. Process designer 10.1 reference manual, 2012.
- [12] Andreas Larsson and Henrik Nilsson. Visualize an event-based simulation model made in process simulate, 2011.
- [13] Simens Industry Software Inc. Tecnomatix.net, 2011.

- [14] Simens Industry Software Inc. Tecnomatix Technologies Ltd. Process simulate 10.1 reference manual, 2012.
- [15] KUKA Roboter GmbH. Kuka.rcs module 8 installation and operating instructions, 2011.
- [16] Samir Dalili and Marcus Persson. Development of an event based robotic simulation implemented in process simulate, 2009.
- [17] Wikipedia. Ole for process control — wikipedia, the free encyclopedia, 2012. [Online; accessed 1-January-2013].
- [18] KUKA Roboter GmbH. Kuka kr5 arc specification, 2011.
- [19] KUKA Roboter GmbH. Kr c4 assembly instructions, 2012.
- [20] KUKA Roboter GmbH. Kuka system software 8.2, 2011.
- [21] Václav Brabec. Integrace a modelování systému distribuovaného řízení polohy a průmyslového robota, 2012.

Appendix A

Content of the Attached CD

Simulation of Production Processes in *.pdf format.

Appendix B

Formal Description - Tables

Element	Description
S1	Clear of all variables
T1	Check if simulation runs and also waits for the marker till it moves to the HOME position.
S2	It sets all light stacks in all workspaces to the READY state (orange color).
T1	It branches algorithm to parallel branches S3, S4 and S5. These branches represent each robot cell of a product line.
T21	It is an empty transitions which branches itself to step S43, S22, S23 and S24. Those branches represent a car to Workspace 1, 2, 3 and out respectively.
T86	It is an empty transition to step S39.
S39	It resets all information values of workspaces, such as variables (workspaces are empty/finished their activity).
T87	It is an empty transition to step S2 at the start of the algorithm.

Table B.1: Description of steps and transitions - Start

Element	Description
S3	Empty step which branches two parallel way. The first one is for T3 and the second one is for T4.
T3	Check if there is a car near to Workspace 1
T4	Check if there is no car near to Workspace 1
S41	It sets light stacks in Workspace 1 to the RUN state (green color)
T93	It is an empty transition to step S6.
S6	It sets a program number 1 of robots in Workspace 1 (R1,R2,R3,R4) and run their programs in robot cell
T9	It waits for ending of robotic programs in Workspace 1
S12	It sets light stack to the READY state(orange color) in Workspace 1 and also sets information that Workspace1 finished its work.
T15	It is an empty transition to step S15
S7	It sets information that Workspace 1 is empty
T10	It is an empty transition to step S15
S15	It is an empty step to transition T21

Table B.2: Description of a robot cell in Workspace 1

Element	Description
S4	Empty step which branches to transition T5 and T6
T5	Check if there is a car near to Workspace 2
T6	Check if there is no car near to Workspace 2
S45	It sets light stack to the RUN state(green color) in Workspace 2
T98	It is an empty transition to S8
S8	It activates a program number 1 of Kuka KR5 arc and runs it in Workspace 2
T11	It waits until a program number 1 of KUKA KR5 arc ended in Workspace 2.
S13	It runs the marker in Workspace 2
T16	It checks if the marker runs and continues to S16 in Workspace 2
S16	It sends the marker to the HOME position and runs program number 2 of KUKA KR5 arc in Workspace 2
T18	It waits until a program number 2 of KUKA KR5 arc ended in Workspace 2
S18	It sets a program number 1 of robot R5 and runs it in Workspace 2
T19	It waits until a program number 1 of robot R5 ended in Workspace 2
S19	It sets light stack to READY state(orange color) in Workspace 2 and also sets information that Workspace 2 finished its work.
T20	Transition to step S20
S9	It sets information that Workspace 2 is empty
T12	Empty transition to step S20 in Workspace 2.
S26	It is an empty step to transition T21.

Table B.3: Description of a robot cell in Workspace 2

Element	Description
S5	It is an empty step to transition T7 and T8 in workspace 3
T7	Check, whether there is a car near to Workspace 3
T8	Check, whether there is no car near to Workspace 3
S42	It sets light stacks in Workspace 3 to RUN state (green color)
T94	It is an empty transition to step S10
S10	It sets a program number 1 of robots in Workspace 3 (R6,R7,R8,R9) and run their programs in robot cell
T13	It waits for ending of robotic programs in Workspace 3
S14	It sets light stack to READY state(orange color) in Workspace 3 and also sets information that Workspace3 finished its work.
T17	It is an empty transition to step S17.
S11	It sets information that Workspace 3 is empty
T14	It is an empty transition to step S17.
S17	It is an empty step to transition T21.

Table B.4: Description of a robot cell in Workspace 3

Element	Description
S43	It is an empty step that branches itself into transition T95 and T96
T95	Check if there was not pressed a button for stopping incoming cars to product line (StopButton).
T96	Check if there was pressed a button for stopping incoming cars to product line (StopButton)
S21	It sends a car to Workspace 1.
T70	It waits for incoming car (information from sensors) to robotic cell of Workspace 1 and then goes to step S25.
S25	It is an empty step.
T88	It waits for a small moment to synchronize information.
S44	It is an empty step.
T97	It is an empty transition to step S32.
S32	It stops to flow a car to Workspace 1.

Table B.5: Description of a car flow to Workspace 1

Element	Description
S22	It is an empty step which branches into transition T71 and T67.
T71	Check if there is a car in rob cell in Workspace 1.
T67	Check if there is not a car in rob cell in Workspace 1.
S26	It sends a car to Workspace 2 from Workspace 1.
T76	It waits for incoming car (information from sensors) to robotic cell of Workspace 2 and then goes to step S33.
S33	It is an empty step.
T89	It waits for a small moment to synchronize information.
S27	It is an empty step.
T77	It is an empty transition to step S36.
S36	It stops to flow a car to Workspace 2.

Table B.6: Description of a car flow to Workspace 2