Work Package 4

Engineering methodology

Deliverable D4.3

Integration of GRACE multi-agent systems with manufacturing CAE systems

Document type: Deliverable
Document version: Final
Document Preparation Date: 30/11/2012
Classification: Public
Author(s): SIEMENS
File Name: Deliverable 4.3 – Integration of GRACE multi-agent systems with manufacturing CAE systems_v1.0.pdf

Project Ref. Number: 246203
Project Start Date: 01/07/2010
Project Duration: 36 months
Website: www.grace-project.org

Project funded by the European Commission under the “Seventh Framework Programme” (2007-2013)
Contract n° NMP2-SL-2010-246203
<table>
<thead>
<tr>
<th>Rev.</th>
<th>Content</th>
<th>Resp. Partner</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>Document structure &amp; chapter 1</td>
<td>SIEMENS</td>
<td>15/10/2012</td>
</tr>
<tr>
<td>0.2</td>
<td>Adding chapter 2</td>
<td>SIEMENS</td>
<td>23/10/2012</td>
</tr>
<tr>
<td>0.3</td>
<td>Adding chapter 4</td>
<td>SIEMENS</td>
<td>26/10/2012</td>
</tr>
<tr>
<td>0.4</td>
<td>Adding chapter 3</td>
<td>SIEMENS</td>
<td>12/11/2012</td>
</tr>
<tr>
<td>0.5</td>
<td>Adding chapter 5</td>
<td>SIEMENS</td>
<td>16/11/2012</td>
</tr>
<tr>
<td>0.6</td>
<td>Adding Glossary and internal revision</td>
<td>SIEMENS</td>
<td>19/11/2012</td>
</tr>
<tr>
<td>0.8</td>
<td>Revision</td>
<td>SIEMENS</td>
<td>21/11/2012</td>
</tr>
<tr>
<td>0.9</td>
<td>Draft of Deliverable available to all partners for review</td>
<td>ALL</td>
<td>22/11/2012</td>
</tr>
<tr>
<td>1.0</td>
<td>Include review comments and prepare submission</td>
<td>SIEMENS</td>
<td>30/11/2012</td>
</tr>
</tbody>
</table>
# Table of contents

Table of contents........................................................................................................................................... 3  
Table of figures.................................................................................................................................................. 6  
Glossary.............................................................................................................................................................. 9  
1. Introduction.................................................................................................................................................. 13  
2. Concept for engineering tool analysis........................................................................................................ 15  
   2.1. Concept of perspectives......................................................................................................................... 15  
      2.1.1. Perspective – Engineering Activity Chain...................................................................................... 17  
      2.1.2. Perspective – Tool chain................................................................................................................. 18  
      2.1.3. Perspective – Information exchange ............................................................................................... 18  
   2.2. Applicability of the modeling approach............................................................................................... 19  
      2.2.1. Definition of process model............................................................................................................. 19  
      2.2.2. Analysis of re-use capabilities.......................................................................................................... 20  
      2.2.3. Analysis of tool chain coverage and continuity .............................................................................. 20  
      2.2.4. Analysis of lossless information flow.............................................................................................. 20  
   2.3. Consideration of engineering process consistency.............................................................................. 21  
      2.3.1. Fractions and gaps within engineering processes ......................................................................... 24  
      2.3.2. Engineering process evaluation .................................................................................................... 27  
   2.4. Tool chain scenarios............................................................................................................................... 32  
      2.4.1. Best of bread tools........................................................................................................................... 32  
      2.4.2. Integrated tool chain........................................................................................................................ 33  
      2.4.3. All-in-one tools................................................................................................................................. 33  
3. Description of selected engineering tools ............................................................................................... 35  
   3.1. AutomationML ...................................................................................................................................... 35  
      3.1.1. Tool description............................................................................................................................... 35  
      3.1.2. Mechatronical concepts.................................................................................................................. 38
3.2. Bentley Microstation................................................................................................. 40
  3.2.1. Tool description ................................................................................................. 40
  3.2.2. Mechatronical concepts .................................................................................. 43
3.3. Comos / Automation Designer ............................................................................. 45
  3.3.1. Tool description ................................................................................................. 45
  3.3.2. Mechatronical concepts .................................................................................. 50
3.4. JADE ...................................................................................................................... 54
  3.4.1. Tool description ................................................................................................. 54
3.5. NetBeans IDE ........................................................................................................ 58
  3.5.1. Tool description ................................................................................................. 58
3.6. Protégé .................................................................................................................. 61
  3.6.1. Tool description ................................................................................................. 61
3.7. Tecnomatix Plant Simulate .................................................................................... 64
  3.7.1. Tool description ................................................................................................. 64
  3.7.2. Mechatronical concepts .................................................................................. 71
3.8. Tecnomatix Process Designer .............................................................................. 73
  3.8.1. Tool description ................................................................................................. 73
  3.8.2. Mechatronical concepts .................................................................................. 77
3.9. Tecnomatix Process Simulate .............................................................................. 79
  3.9.1. Tool description ................................................................................................. 79
  3.9.2. Mechatronical concepts .................................................................................. 80
3.10. Tecnomatix RobCAD ......................................................................................... 82
  3.10.1. Tool description ................................................................................................. 82
  3.10.2. Mechatronical concepts .................................................................................. 88
3.11. TIA Portal ........................................................................................................... 89
  3.11.1. Tool description ................................................................................................. 89
  3.11.2. Mechatronical concepts .................................................................................. 91
4. Engineering Tool chain based on general engineering workflow ........................................ 93
5. Additional tool requirements for industrial engineering tools supporting GRACE MAS-architecture .............................................................................................................. 95
   5.1. Case studies.................................................................................................................. 95
   5.2. Additional tool requirements....................................................................................... 96
References .......................................................................................................................... 98
Appendix A – Engineering tool evaluation ........................................................................ 102
Appendix B – Modeling examples for engineering tools .................................................. 103
Appendix C – Engineering tool chain based on OEM specific engineering workflow ........ 104
Appendix D – Bearing insertion station – Whirlpool documentation .................................. 105
Appendix E – Bearing insertion station – Comos documentation ......................................... 106
Table of figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Relations between Engineering process and engineering artifacts</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>Intention and goals of GRACE WP 4</td>
<td>13</td>
</tr>
<tr>
<td>3</td>
<td>Design decision dependability</td>
<td>16</td>
</tr>
<tr>
<td>4</td>
<td>Engineering artifact flow</td>
<td>22</td>
</tr>
<tr>
<td>5</td>
<td>Problem location in the engineering process</td>
<td>24</td>
</tr>
<tr>
<td>6</td>
<td>Possible evaluation of information logistics</td>
<td>25</td>
</tr>
<tr>
<td>7</td>
<td>AutomationML base architecture</td>
<td>36</td>
</tr>
<tr>
<td>8</td>
<td>Possible modeling of mechatronical units by AutomationML</td>
<td>39</td>
</tr>
<tr>
<td>9</td>
<td>Microstation main window</td>
<td>41</td>
</tr>
<tr>
<td>10</td>
<td>Element parameter example and tool box examples</td>
<td>42</td>
</tr>
<tr>
<td>11</td>
<td>Cell file management and use</td>
<td>42</td>
</tr>
<tr>
<td>12</td>
<td>Information sets covered by MS</td>
<td>43</td>
</tr>
<tr>
<td>13</td>
<td>Cell definition structure</td>
<td>44</td>
</tr>
<tr>
<td>14</td>
<td>Cell example</td>
<td>45</td>
</tr>
<tr>
<td>15</td>
<td>Base object structure with project structures</td>
<td>46</td>
</tr>
<tr>
<td>16</td>
<td>Relation between base object and planning object</td>
<td>47</td>
</tr>
<tr>
<td>17</td>
<td>Information sets covered by ENGINEERING ARTIFACT</td>
<td>51</td>
</tr>
<tr>
<td>18</td>
<td>Implementation of engineering artifact within SIMATIC AD</td>
<td>53</td>
</tr>
<tr>
<td>19</td>
<td>Containers in the JADE Platform</td>
<td>55</td>
</tr>
<tr>
<td>20</td>
<td>Graphical User Interface of the Remote Monitoring Agent</td>
<td>56</td>
</tr>
<tr>
<td>21</td>
<td>Graphical User Interface of the Sniffer Agent</td>
<td>56</td>
</tr>
<tr>
<td>22</td>
<td>Graphical User Interface of the Introspector Agent</td>
<td>57</td>
</tr>
<tr>
<td>23</td>
<td>Graphical User Interface of the Director Facilitator</td>
<td>57</td>
</tr>
<tr>
<td>24</td>
<td>NetBeans runtime container (see <a href="http://platform.netbeans.org/tutorials/nbm-runtime-container.html">http://platform.netbeans.org/tutorials/nbm-runtime-container.html</a>)</td>
<td>58</td>
</tr>
<tr>
<td>25</td>
<td>NetBeans user interface</td>
<td>59</td>
</tr>
<tr>
<td>26</td>
<td>NetBeans Swing GUI Builder</td>
<td>60</td>
</tr>
</tbody>
</table>
Figure 27: NetBeans project structure................................................................. 61
Figure 28: Screenshot of the Protégé editor...................................................... 62
Figure 29: Details of the Protégé editor............................................................. 63
Figure 30: Protégé architecture diagram related to the 3.x series of Protégé...... 64
Figure 31: Use of Tecnomatix Plant Simulate in Engineering Phases ............. 65
Figure 32: Windows for variables and methods of objects............................ 66
Figure 33: Attribute configuration of moveable objects .................................. 67
Figure 34: Attribute configuration of material flow objects........................... 68
Figure 35: Class library example..................................................................... 70
Figure 36: Network example ........................................................................... 71
Figure 37: Information sets reflected by Tecnomatix Plant Simulate ............... 72
Figure 38: Basic Implementation for Mechatronical Units in TPS .................. 73
Figure 39: Software architecture of PD............................................................ 74
Figure 40: PD user interface ........................................................................... 76
Figure 41: PD library example ........................................................................ 77
Figure 42: Information sets of engineering artifact covered by PD .................. 78
Figure 43: Example of a engineering artifact in PD ....................................... 79
Figure 44: Tecnomatix Process Designer software structure ......................... 79
Figure 45: Information sets covered by PS ..................................................... 81
Figure 46: Engineering artifact implementation in PS ..................................... 81
Figure 47: Geometry modeling in RobCAD .................................................... 83
Figure 48: Kinematics editor ......................................................................... 84
Figure 49: Behavior modeling ....................................................................... 85
Figure 50: Main modeling objects of RobCAD .............................................. 86
Figure 51: Basic work cell ............................................................................. 86
Figure 52: Robot examples ............................................................................ 87
Figure 53: Controller example in RobCAD .................................................... 88
Figure 54: Engineering artifact information representation in RobCAD ........ 89
Figure 55: Start window of TIA Portal ............................................................ 90
Figure 56: Information sets covered by TIA Portal ........................................ 91
Deliverable D4.3
Integration of GRACE multi-agent systems with manufacturing CAE systems

Figure 57: engineering artifact structure in TIA Portal ....................................................... 92
Figure 58: Engineering tool chain based on general engineering workflow ......................... 94
Glossary

The following section defines the understanding and uses of different terms within this document.

**Domain** - An area of knowledge that

1. is scoped to maximize the satisfaction of the requirements of its stakeholders,
2. includes a set of concepts and terminology understood by practitioners in that area and
3. includes the knowledge of how to build (software) systems in that area

**Engineering** – Following (Encyclopedia Britannica 2011) engineering is “the creative application of scientific principles to design or develop structures, machines, apparatus, or manufacturing processes, or works utilizing them singly or in combination; or to construct or operate the same with full cognizance of their design; or to forecast their behavior under specific operating conditions; all as respects an intended function, economics of operation and safety to life and property.”

**Engineering process** – An engineering process is a coordinated course of knowledge use actions. It is divided into a set of phases targeting different levels of detail and concreteness of the engineering objects.

**Mechatronic engineering process** – A mechatronic engineering process is an engineering process where the involved engineering objects are mechatronic units and mechatronic systems.

**Engineering artifact** – An engineering artifact is an object created or used within one or more actions of an engineering process. Within early phases of engineering processes engineering artifacts are usually of informational nature while in later phases they can also be of physical nature.
Deliverable D4.3
Integration of GRACE multi-agent systems with manufacturing CAE systems

**Engineering tool** - An engineering tool is a software system designed to model, process, and store plant components or facets within an engineering process.

**Manufacturing Industry** - Task and aim of manufacturing engineering is the production of geometric determined solid bodies (workpiece, assembly group, products) with preset attributes by application of various manufacturing methods.

**Mechatronic system** - A mechatronic system is a closed system that realizes by its actuating elements, its sensor system and its control a defined (usually physical) behavior within a manufacturing system. It is composed of one or more mechatronic units and may be processed like a mechatronic unit during the engineering.

**Mechatronic engineering activity** - A mechatronic engineering activity (general engineering activity) is an action within the mechatronic engineering process. It is assigned to one or more phases of the engineering process, executed by one involved engineering role, and has impact on the design of a mechatronic system or unit.

**Mechatronic modeling concept** - A mechatronic modeling concept is an approach to map mechatronic systems or units to plant components or facets by structure and relationship.

**Mechatronic unit (MU)** - A mechatronic unit (plant component) is a mechatronic system that can be described by its functionality and is composed of software (control) and hardware (mechanical, electrical, and further construction) objects.

**Figure 1:** Relations between Engineering process and engineering artifacts
Model - A model is a representation of a real system or a couple of systems. Models have three significant features:

1. Representation. A model is always a representation of natural and artificial originals which could be models, too.
2. Reduction. A model does not have all attributes and aspects of the original, just those, which seem to be relevant to the model user or creator.
3. Intended purpose. Creating a model has always a purpose defined by the intended use of the model. The usage defines the model relevant parts and aspects.

A model is featured by abstraction which means the conscious reduction of reality to stress model features important for the modeler or for the model intention.

Modules – In the context of this deliverable the use of the word module should be understood as a collective term to describe the entirety of both agents and mechatronic units. This is done to cover all application cases e.g. agents interacting with agents, agents interacting with mechatronic units, vice versa and mechatronic units interacting with mechatronic units.

Plant - Plants are large assemblies of technical devices to produce certain technical products usually in a fully or semi-automatic way. Plants execute technological processes which produce the technical products from several processing steps.

Plant components - Reusable unit, which is completed and encapsulated, with defined interfaces to the outside. A component is an aggregation of building blocks.

The following terms of multi-agent-systems are defined according to (Instituto Politécnico de Bragança 2011):

Agent - An autonomous component, that represents physical or logical objects in the system, capable to act in order to achieve its goals, and being able to interact with other agents, when it doesn’t possess knowledge and/or skills to reach its objectives on its own.

Distributed Control System (DCS) - A dedicated system used to control manufacturing processes that are continuous or batch-oriented. DCSs are connected to sensors and actuators, normally through Programmable Logic Controllers (PLCs) and use set-point control to control the flow of material through the plant.
**Holon** - An identifiable part of a (manufacturing) system that has a unique identity, yet is made up of sub-ordinate parts and in turn is part of a larger whole.

**Multi-agent systems (MAS)** - Society of agents that represent the objects of a system that interact with each other to reach the system objectives.

**Moby** - is an electronic RFiD tag attached to the pallet carrying the products along the production line, identifying univocally each product. The tag contains information that can be accessed by a RFiD reader.

**Product** - In the sense of this work, a product is not only related to final products commercialized by OEM companies, but also intermediate parts/components that comprise complex products. Examples of products are washing machines or drums.

**Computer-aided Engineering (CAE)** - is the broad usage of computer software to aid in engineering tasks. It includes computer-aided design (CAD), computer-aided analysis (CAA), computer-integrated manufacturing (CIM), computer-aided manufacturing (CAM), material requirements planning (MRP), and computer-aided planning (CAP).

**Engineering Activity** - An activity is a task that needs to be accomplished within a defined period of time.
1. Introduction

The Deliverable 4.3 is the last output of work package 4 “Engineering methodology”. Within this work package the suitability of typical industrial CAE tools for modular factory automation will be evaluated. Thus, a tool chain for the GRACE engineering methodology can be foreseen and additional tool requirements may be identified. This will be done within three steps:

- Comparison of developed engineering methodology for decentralized manufacturing systems to available engineering concepts of selected CAE tools
- Derivation of (additional) requirements to CAE tool concepts
- Prototypical implementation of interfaces of CAE tools to GRACE platform enabling generation of configuration data for GRACE production process control and quality control

Main Objective: A new engineering methodology for decentralized manufacturing control systems based on the GRACE MAS platform

**Figure 2: Intention and goals of GRACE WP 4**

Within task 4.1 a general engineering process reference model for manufacturing systems was defined, which was developed and evaluated for domains like automotive production lines or home appliance production lines. It was then adapted and extended towards a detailed GRACE specific engineering process for home appliance production lines including process and quality control. The model was created by

- dissecting the engineering of the production line in its phases,
• decomposition of the working processes (e.g., layout planning) within the individual phases into individual activities and
• identification of their relations, and influences (causes and effects) with system properties of the process & quality control system.

This model and especially the general engineering activities presented in Deliverable 4.1 will serve as a basis for the tool evaluation presented within this deliverable D4.3.

Like the previous deliverables of WP 4 also Deliverable 4.3 will be divided in two parts. The intention is to present the general results within a public part, but to deepen the analysis in parallel within the confidential appendixes.

With the help of this bisection GRACE project will provide a public suitability analysis of CAE tools regarding the GRACE engineering methodology. In Addition the Appendixes will use confidential information of the consortium partners for further discussion of these results.

This document is structured as follows. The concept of general engineering activities was already introduced as an important prerequisite within deliverable 4.2. Chapter 2 will describe how these activities can be used for the evaluation of engineering tools. Within chapter 3 we will give an introduction to twelve CAE tools respectively data platforms used within the engineering tool chain. Additionally the tools will be evaluated regarding their coverage of mechatronical data as presented in Deliverable 4.2.

Chapter 4 will show how these tools may be applied as part of a tool chain at the general engineering workflow presented in Deliverable 4.2.

Based on these results chapter 5 will present additional tool requirements derived from the analysis.

Within Appendix A a detailed analysis of the industrial CAE tools will be shown based on the general engineering activities presented in Deliverable 4.1. Appendix B will deepen the tool understanding by showing typical modeling examples for some of these tools. Thus, Appendixes A and B will complete the tool analysis carried out in chapter 3.

Respectively, Appendix C will complete the tool chain analysis by applying the tools to the OEM specific engineering workflow presented in Deliverable 4.2 - Appendix A.

As a last part the Appendixes D and E will provide insights to the modeling of the bearing insertion station of the washing unit production line of Whirlpool in Naples. Appendix D will show which results were used to engineer the station for the first time. Appendix E will show the results of the reengineering of the production line within COMOS. Thus, interfaces between a typical all-in-one industrial CAE tool and the GRACE architecture will be shown by reengineering this part of the production line based on the GRACE engineering methodology.

Additional interfaces / influences of industrial CAE tools with / to GRACE based modular manufacturing system can be also found within the tool descriptions and the evaluation of the general engineering activities for each tool. Special interfaces used within GRACE e.g. the interface between GRACE MAS and Labview® or GRACE MAS and GraDaCo database will be shown in more detail within deliverable 5.1.
2. Concept for engineering tool analysis

Within this chapter the idea of an analysis method for tool evaluation based on engineering workflows will be shown. The method itself has been used by Siemens Corporate Technology more than a hundred times to evaluate tool suitability for special customer use cases. As the methodology contains highly confidential knowledge and intellectual property, only the basic idea will be shown. Additionally three typical tool scenarios will be discussed.

2.1. Concept of perspectives

The following section will present a set of perspectives on engineering processes relevant for their evaluation and optimization.

The engineering process of production systems is executed within an engineering organization and consists, usually, of a set of design decisions which have to be taken (Helmus 2003),(Grundig 2009).

VDI Guideline 3695 (VDI 3695) defines an Engineering Organization (EO) is an engineering firm or engineering subunit of a company (supplier, plant manufacturer, plant operator) executing one or more engineering processes. A project team cooperating for a limited time (also across companies) is also regarded to be an EO.

VDI Guideline 3695 additionally states, that for an EO there are key aspects of relevance related to the quality of an EO. The most relevant key aspects from authors point of view are the workflow, the methods used, the tools, the organizational structure, and the economics.

To capture the structure and behavior of an EO the procedure model of the EO can be exploited. The procedure model of an engineering process determines a sequence of engineering activities taking the necessary design decisions to incrementally enrich the intended technical solution, the required input and output information for the design decisions, tools to be used, and respective responsible human roles. It formalizes the engineering process and examines the technical content of an EO’s process. The use of a procedure model thus facilitates a structured engineering process, which optimizes the engineering and data workflow and the resource utilization. Besides the technical tasks, a procedure model should also describe activities in project management, quality control, change management as well as verification and validation of the engineering solution.

For the appropriate execution of the design decisions within the engineering process a set of preconditions have to be fulfilled. These preconditions include:

- the availability of the required input information for the decision activity in the required quality at the right moment,
- the execution of the design decision by a team of well trained and skilled persons, and
- the adequate tool support for the efficient execution of the design decision.
Within an EO design decisions have to be taken reflecting the incremental enrichment of the technical solution by knowledge and skill use. These design decisions are part of engineering activities. For each engineering activity the following conditions will hold: to enable the execution of the engineering activity a (possibly empty) set of predecessor engineering activities have to be finished, special information have to be available, appropriate engineering tools should be useable. Humans with the required skills and knowledge should execute the engineering activities. The engineering activity will create information reflecting the taken design decisions and provide them for a (possibly empty) set of successor design decisions within further engineering activities. This structure is depicted in the following Figure 3.

Thus an engineering process can be seen from different positions facing the different requirements to the design decision to be taken. This can be reflected by the term of a perspective on an engineering process.

A perspective on an engineering process models one structural, organizational, technical or further aspect of an engineering process covering the complete lifecycle of an engineering process. It subsumes all process objects within an engineering process related to the considered aspect. Thus, a perspective on an engineering process has to be seen as the set of required process objects of a defined object type (set of object types) were the object type (set of object types) defines the considers aspect of the engineering process execution describing needs for an efficient and correct execution of the engineering process. Perspectives do not need to be disjunctive.

Following this definition a perspective on an engineering process can be defined by the process objects covered by this perspective. Thus, within different engineering organizations there might be different perspectives following the different industrial fields and its different bordering conditions for engineering processes. Nevertheless, there are some perspectives which are common for all engineering processes. Three of these perspectives are the given in the next subsections.

**Figure 3: Design decision dependability**
The use of the perspectives and its semantics within an engineering process analysis depends on the application case. Examples for the semantic use are given in the following Chapter 3.

2.1.1. Perspective – Engineering Activity Chain

The perspective “Engineering Activity Chain” subsumes all process objects of an engineering process related to the sequence of necessary engineering activities for the intended engineering result. Thus, it covers among others

- the engineering activities itself with the design decisions taken within,
- the predecessor-successor-relation of engineering activities, and
- the timing requirements of the engineering activities like minimal and maximal duration or earliest and latest starting and end point.

In addition this perspective has relations to the other perspectives named in this chapter. Here it will be especially relevant:

- which engineering tools should be used for the execution of the design decisions, and
- which engineering artifacts are required for the execution for the design decisions and which are created during the design decisions.

For the analysis of this perspective the following table can be exploited:

Table 1: Activity perspective

<table>
<thead>
<tr>
<th>Activity</th>
<th>Name of the activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Short description of the design decisions taken within the engineering activity</td>
</tr>
<tr>
<td>Timing</td>
<td>Definition of durations, start time and end time of the activity</td>
</tr>
<tr>
<td>Predecessor activities</td>
<td>Activities creating the engineering artifacts required for the activity</td>
</tr>
<tr>
<td>Successor activities</td>
<td>Activities requiring the engineering artifacts developed within the activity</td>
</tr>
<tr>
<td>Involved human resources</td>
<td>Description of required engineering roles with competences, knowledge, and skills required for the execution of the engineering activity</td>
</tr>
<tr>
<td>Exploited tools</td>
<td>Description of engineering tools usable or required for the execution of the engineering activity</td>
</tr>
<tr>
<td>Required artifacts</td>
<td>Description of engineering artifacts required for the execution of the engineering activity</td>
</tr>
<tr>
<td>Created artifacts</td>
<td>Description of engineering artifacts developed within the execution of the engineering activity</td>
</tr>
</tbody>
</table>
2.1.2. Perspective – Tool chain

The perspective “Tool chain” subsumes all process objects of an engineering process related to the sequence of engineering tools used for the execution of the intended engineering activities. Thus it covers among others

- the type of the engineering tools required,
- the input and output artifacts consumed or created by the engineering tool,
- the data formats consumable or producible by the tool and
- the skills and knowledge required.

For the analysis of this perspective the following table can be exploited:

**Table 2: tool perspective**

<table>
<thead>
<tr>
<th>Tool</th>
<th>Name of the tool</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Short description of the tool</td>
</tr>
<tr>
<td>Tool type</td>
<td>Give the category of tools the tool belongs to</td>
</tr>
<tr>
<td>Involved human resources</td>
<td>Description of engineering roles with competences, knowledge, and skills required for use of the tool</td>
</tr>
<tr>
<td>Required artifacts</td>
<td>Description of engineering artifacts consumable by the tool</td>
</tr>
<tr>
<td>Created artifacts</td>
<td>Description of engineering artifacts developed by the tool</td>
</tr>
<tr>
<td>Input formats</td>
<td>Name data formats consumable by the tool or input technology used</td>
</tr>
<tr>
<td>Output formats</td>
<td>Name data formats created by the tool or output technology used</td>
</tr>
</tbody>
</table>

2.1.3. Perspective – Information exchange

The perspective “Information exchange” subsumes all process objects of an engineering process related to the exchange of engineering artifacts between different engineering tools along the tool chain, different human resources within the engineering organization, and among different engineering activities. These process objects can be of virtual nature (i.e. information coded within IT artifacts) or physical nature (parts of the physical system implementation). Thus this perspective can be described by naming the engineering artifacts exchanged including

- the type and semantics of the artifact exchanged,
- the data format and/or technology used for the artifact exchange1,
- the tools able to create the artifacts,
- the tools able to load and use the artifacts,

---

1 The technologies addressed here can have a very broad range from definition of file types over characterization of database types, used XML schema, and exploited communication protocols up to communication architectures like OPC.
the humans and its skills and knowledge required to create and use the artifacts.

For the analysis of this perspective the following table can be exploited:

**Table 3: artifact perspective**

<table>
<thead>
<tr>
<th>Artifact</th>
<th>Name of the artifact</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Short description of the artifact</td>
</tr>
<tr>
<td>Type of artifact</td>
<td>Give the category of artifact, the artifact belongs to</td>
</tr>
<tr>
<td>Semantics of artifact</td>
<td>Give the meaning of the artifact within the engineering process executed</td>
</tr>
<tr>
<td>Data format</td>
<td>If the artifact is exchanged by a file give the file format and the definition of the content structure of the file.</td>
</tr>
<tr>
<td>Exchange technology</td>
<td>If technologies are used for the artifact exchange going beyond the use of files name the technologies used and the type and version of software and hardware exploited for the implementation of the technologies should be given.</td>
</tr>
</tbody>
</table>

| Involved human resources | Description of engineering roles with competences, knowledge and skills required for development and use of the artifact |
| Creating tools | Name the tools possibly creating the artifact |
| Consuming tools | Name the tools possibly consuming the artifact |
| Creating activity | Name the engineering activities possibly creating the artifact |
| Consuming activity | Name the engineering activities possible consuming the artifact |

**2.2. Applicability of the modeling approach**

Based on a set of completed tables of all three types for an engineering organization a network of engineering activities, artifacts, tools and human roles can be characterized. It depends on the application case how the information within the tables is interpreted. In some cases it can be regarded as property or feature in some other cases as requirement. In the following some examples of application cases are named.

**2.2.1. Definition of process model**

As mentioned in (VDI 3695) one central criterion for EOs is the definition of process models for the project dependent and project independent engineering process. In both cases it has to be defined:

- which engineering activities have to be executed in which sequence
- which artifacts are required for these engineering activities,
- which artifacts are created by these engineering activities,
- which human roles are required for these engineering activities, and
- which tools have to be exploited for these activities.
In this case the Engineering Activity Chain perspective is the leading perspectives providing the properties and all further perspectives are seen as requirements to the engineering organization and the engineering process execution.

### 2.2.2. Analysis of re-use capabilities

Following (VDI 3695), another interesting criterion for EOs are the re-use capabilities of engineering artifacts. Therefore, it has to be determined, which engineering artifacts created in one engineering project can be exploited in other engineering projects. To analyze this capability the Information Exchange perspective is the leading perspective. It has to be evaluated (in this order):

- which engineering activities have to be executed in which sequence
- which artifacts are created by these engineering activities, and
- which artifacts are required for these engineering activities.

If there is an artifact created which can be used in another engineering process later on, this can be identified.

### 2.2.3. Analysis of tool chain coverage and continuity

Based on (VDI 3695) the tool chain coverage of an EOs and its continuity can be one criterion for the EO quality. To analyze the tool chain the following information have to be collected:

- which engineering activities have to be executed in which sequence
- which tools have to be exploited for these activities.

Here a main focus has to be the amount of engineering activities covered by one tool and its continuity in the activity chain as well as on the import and export formats (and technologies) supported by the different tools at the points were different tools will be used to execute the design decisions of the engineering activities been in predecessor – successor – relation. In this case the import and export formats (and technologies) are seen as the driving requirements.

### 2.2.4. Analysis of lossless information flow

Finally, one interesting criterion is the lossless information flow within an EO. This lossless information flow is given in the case were the logistics of engineering artifacts within an EO is correct and free from problems (see chapter 2.3).

For the evaluation of this fact it has to be evaluated:

- which engineering activity covering which design decision has to be executed in which sequence
• which artifacts are required for these engineering activities,
• which artifacts are created by these engineering activities,
• which tools have to be exploited for these activities, and
• which human roles are required for these engineering activities.

In a first stage the Information Exchange perspective is considered as a requirement for the Tool Chain perspective. It will be investigated whether a defined data format for an artifact can be provided or consumed by a tool.

In the second stage the Engineering Activity Chain perspective and the Tool Chain perspective are considered. It is evaluated whether the available human resource roles will provide all required skills, capabilities, and knowledge required by tools and engineering activities.

For more information on this analysis method see chapter 2.3.

2.3. Consideration of engineering process consistency

The engineering process of production systems is characterized by a sequence of design decisions contained in engineering activities. By this sequence of design decisions the engineering artifacts required to characterize, model, parameterize, program, and implement the production system are established. For the execution of the design decisions appropriate tools are exploited which are used by humans (see Figure 4).
Thus, for an appropriate engineering process the migrating engineering artifacts and the information contained within should be considered. To ensure a proper information logistic as stated in VDI guideline 3695, it will be necessary to enable the provision of

- the right information (artifact)
- in the right quality (data type, file type)
- at the right moment
- to the right tool and
- to the right human able to interpret them.

Problems will occur, if this information logistic will not have the required quality. This may be caused by the following problems.

**Problem 1:** The information required is not available. This can happen, if a required input to an engineering activity cannot be provided by a human, tool or activity of the scope of activity of the considered engineering chain.

**Problem 2:** The information cannot be provided in the right quality. This happens, if an engineering artifact is available only in a format that cannot be
automatically imported to the engineering tool used for the next design decision.

In such a case there are two options possible. The first one is a data transformation executed by another software system mapping the available data format to a data format importable to the intended tool. The second one is a manual integration of the information to the intended tool.

In both cases problems resulting from the necessary data conversion may happen resulting from human error-proneness.

Problem 3: The information can only be partially provided in the right quality. This happens, if the engineering tool to be used has either a data model not able to interpret the information contained in the provided artifact or

Problem 4: The information cannot be provided in the right moment. This will happen, if the information or the artifact required will be created by a predecessor activity which is not completed at the moment the information/artifact is required.

Problem 5: The used tool is not the right tool for the intended design decision. This will happen on the one hand, if the scope of activity of the tool will not contain the intended design decision. On the other hand this will happen if the tool chain at all is not suitable for the sequence of design decisions.

Problem 6: The involved human has not the right skills or knowledge for the intended design decision. This will happen, if the human is either not able to use the intended tool or to interpret the information provided in the right way necessary for the design decision.

Problem 7: The provided information within the engineering process are not consistent. This will happen, if the work within an engineering tool will lead to not consistent information within this tool or the parallel work within more than one tool will result in contradicting artifacts.

The location of the seven problems in the engineering process is depicted in Figure 5.
Within the following subsections these problems will be considered in more detail. To enable the selection of appropriate measure to avoid these problems based on mechatronical concepts two classes of problems named fractions and gaps within the engineering process of production systems are considered. Therefore the terms fraction and gap will be defined, typical examples are given based on the results of the questionnaire, a methodology for evaluation of engineering organizations and the identified 7 problems of information logistics within the engineering chain. Finally, possibilities to eliminate the fractions and gaps by exploiting mechatronical concepts are named.

2.3.1. Fractions and gaps within engineering processes

Within this section the terms information fraction and information gap with respect to engineering execution are defined. Some basic examples for fractions and gaps are given.
An information fraction is given in the case where information provided to an engineering activity cannot be used within this engineering activity by reasons of data provision process. Information fractions are given in the case were the provided information are either not the right information for the engineering activity, not in the right data format thus not readable, not consistent, or not available in the right moment (delayed).

An information gap is given in the case, were the information provided to an engineering activity cannot be used within this engineering activity by reasons of decision process execution. Information gaps are given in cases were tool cannot execute engineering activity based on the information (the tool is not appropriate for the design decision based on the provided information) or human cannot execute engineering activity based on the information (the humans have not the right skills, knowledge, or competences to execute the design decisions based on the provided information).

Facing the migration and use of information within an engineering organization the risk of fractions and gaps is related to the amount of separations within the four different perspectives at the same qualitative point in time of the engineering organization. Thus risk factor can be defined as follows.

Figure 6: Possible evaluation of information logistics

Figure 6 provides a qualitative visualization of the analysis results of an example engineering process. In this figure two problems can be easily seen. Between project phases C and D there is a kind of fraction. Within each perspective there is a switch to another entity responsible for the project. Thus, there is a risk of losing information at this point of the project resulting in errors or double work. The second problem visible is that there is no tool available to execute engineering activity B.C. This also might result in problems of the information chains since design decisions cannot be taken creating information.

Exploiting this thinking the following definitions can be taken defining measures for the quality of an engineering organization.
The separation degree of an engineering organization is the number of separations within the perspectives “Engineering Activity Chain”, “Human Resources”, “Tool Chain”, and “Information exchange” at the same point in time. It is evaluated by the number of activity changes in the Engineering Activity Chain” perspective, the number of different person changes in the “Human Resources” perspective, the number of tool changes in the “Tool Chain” perspective, and the number of artifact changes in the “Information exchange” perspective.

Mapped to the problems defined above information fractions are related to problems 1, 2, 3 or 7 and information gaps are related to problems 5 and 6.

Examples of fractions are cases, in which the information exchange among the different involved engineering tools cannot exploit a common data format for the different tools. They result in the high number of paper and .pdf based interfaces as indicated in the questionnaire evaluation.

Typical examples are:

- the transmission of geometry and kinematics information from mechanical engineering to simulation or virtual commissioning in that the used tool will not support the same graphic data format,
- the transmission of signal data from mechanical engineering over electrical engineering to PLC and robot programming in that appropriate data formats are missed, and
- the roundtrip engineering of robot cells between offline robot programming and simulation based collision detection.

For more information about typical existing fractions see the vision paper of the AutomationML consortium (AutomationML e.V. 2007).

Examples of information gaps are usually not sufficient skilled humans, not appropriate tools or its combinations.

Typical examples are tools not directly intended for an engineering decision like

- the use of excel for the design of hierarchical plant structures,
- the use of PLC programming tools for requirement specification, or
- the use of UML class diagrams for the modeling of PLC programs,

or humans not qualified for the engineering decisions like

- electricians execution the design of behavior models of plant components for virtual commissioning of PLC programs or
- PLC programmers executing mechanical engineering.

The criticality of an information fraction or an information gap is the level of impact of this fraction/gap on the proper execution of the intended design decision.
The value of the level of criticality depends on the intention of its use can be given as a real value or an integer value with or without an admissible range. In any case it is an engineering organization related measure to prioritize the quality of information fractions/gaps within an engineering organization with respect to the execution of countermeasures based on the possible impact on the engineering organization quality or efficiency.

2.3.2. Engineering process evaluation

Within this section a methodology for engineering process assessment is given enabling the identification and evaluation of information fractions and information gaps. It is based on the different views on engineering processes given in 2.1.

**Step 1.** As the first step the tables of the perspective “Engineering Activity Chain” have to be completed by discussion with the persons involved for the engineering organization. It will be important to detail for each identified engineering activity the sets of design decisions made within the description line, the sets of involved human roles with skills and knowledge required to take the design decisions, used tools and required and created artifacts in the relevant lines as given in the following figure.

**Table 4: Filling of activity table**

<table>
<thead>
<tr>
<th>Activity</th>
<th>Name of the activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Short description of the design decisions taken within the engineering activity</td>
</tr>
<tr>
<td>Timing</td>
<td>Definition of durations, start time and end time of the activity</td>
</tr>
<tr>
<td>Predecessor activities</td>
<td>Activities creating the engineering artifacts required for the activity</td>
</tr>
<tr>
<td>Successor activities</td>
<td>Activities requiring the engineering artifacts developed within the activity</td>
</tr>
<tr>
<td>Involved human resources</td>
<td>Description of engineering roles with competences, knowledge, and skills required for the execution of the engineering activity</td>
</tr>
<tr>
<td>Exploited tools</td>
<td>Description of engineering tools usable or required for the execution of the engineering activity</td>
</tr>
<tr>
<td>Required artifacts</td>
<td>Description of engineering artifacts required for the execution of the engineering activity</td>
</tr>
<tr>
<td>Created artifacts</td>
<td>Description of engineering artifacts developed within the execution of the engineering activity</td>
</tr>
</tbody>
</table>

**Step 2.** In the second step the tables of the perspective “Information Exchange” have to be completed. For each artifact which is output artifact of an engineering activity and an input artifact of another engineering activity it has to be clearly indicated

a) which tools will create the artifact by taking which design decision and export it using which data format and
b) which tool will consume the artifact by taking which design decision and import it using which data format.

as given in the following table.

Table 5: Filling of artifact table

<table>
<thead>
<tr>
<th>Artifact</th>
<th>Name of the artifact</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Short description of the artifact and the design decisions the artifact is involved in</td>
</tr>
<tr>
<td>Type of artifact</td>
<td>Give the category of artifact, the artifact belongs to</td>
</tr>
<tr>
<td>Semantics of artifact</td>
<td>Give the meaning of the artifact within the engineering process executed</td>
</tr>
<tr>
<td>Data format</td>
<td>If the artifact is exchanged by a file give the file format and the definition of the content structure of the file</td>
</tr>
<tr>
<td>Exchange technology</td>
<td>If technologies are used for the artifact exchange going beyond the use of files name the technologies used and the type and version of software and hardware exploited for the implementation of the technologies should be given</td>
</tr>
<tr>
<td>Involved human resources</td>
<td>Description of engineering roles with competences, knowledge and skills required for development and use of the artifact</td>
</tr>
<tr>
<td>Creating tools</td>
<td>Name the tools possibly creating the artifact</td>
</tr>
<tr>
<td>Consuming tools</td>
<td>Name the tools possible consuming the artifact</td>
</tr>
<tr>
<td>Creating activity</td>
<td>Name the engineering activities possibly creating the artifact</td>
</tr>
<tr>
<td>Consuming activity</td>
<td>Name the engineering activities possible consuming the artifact</td>
</tr>
</tbody>
</table>

Step 3. In the third step the step the tables of the perspective “Tool Chain” have to be completed. For each tool it has to be indicated

a) which data format the tool can import and which data format the tool can export by containing which information (It should be considered that in contrast to step 2 here the capabilities of the tools need to be named and not the intention of the engineering chain.)

b) which design decisions can be taken by humans using this tool and which skills and knowledge will the human require for the design decisions.

Table 6: Filling of tool table

<table>
<thead>
<tr>
<th>Tool</th>
<th>Name of the tool</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Short description of the tool including the design decisions possibly taken by using the tool</td>
</tr>
<tr>
<td>Tool type</td>
<td>Give the category of tools the tool belongs to</td>
</tr>
<tr>
<td>Involved human resources</td>
<td>Description of engineering roles with competences, knowledge, and skills required for use of the tool</td>
</tr>
<tr>
<td>Required artifacts</td>
<td>Description of engineering artifacts consumable by the tool</td>
</tr>
<tr>
<td>Created artifacts</td>
<td>Description of engineering artifacts developed by the tool</td>
</tr>
<tr>
<td>Input formats</td>
<td>Name data formats consumable by the tool or input technology used</td>
</tr>
</tbody>
</table>
Step 4. In the fourth step the information fractions can be evaluated. Therefore, the tables of the perspective “Tool Chain” and the tables of the perspective “Information Exchange” will be compared. The tables of the perspective “Information Exchange” will be considered as required exchange capabilities which have to be supported by the engineering organization. The tables of the perspective “Tool Chain” are considered as provided exchange capabilities which are supported by the tool chain of the engineering organization.

No information fraction based on Problem 1 is given if the following property holds:

Property 1: (For all engineering artifacts having an engineering activity possible consuming the artifact there is an engineering activity possibly creating the artifact).

This definition does not reflect the artifacts which are provided from outside of the process or artifacts created but not used on any further activity of the process. Such artifacts would be considered as fractions. This can be omitted by integrating additional activities modeling the information exchange of the EO with its environment like suppliers and customers.

No information fraction based on Problem 2 is given if for each case an artifact which has to be exchanged between two engineering activities the following property holds:

Property 2: (The two activities exchanging an artifact are executed by the same engineering tool) OR (The intersection of the set of data formats useable to export the artifact from the tool creating the artifact and the set of data formats useable to import the data to the tool consuming the artifact is not empty).

No information fraction based on Problem 3 is given if the following property holds:

Property 3: (Property 1 holds) AND (the information set which can be modeled by the data format used to export the artifact from the tool creating the artifact and the information set of the which can be modeled by the by data format used to import the data to the tool consuming the artifact contains at least the information set required to model the artifact to be exchanged).
If Property 1, 2 and/or 3 are violated for an artifact to be exchanged, there is an information fraction based on this artifact.2

Step 5.

In the fifth step the information gaps are identified. Therefore, the tables of the perspective “Engineering Activity Chain” and the perspective “Tool Chain” will be compared. The tables of the perspective “Engineering Activity Chain” will be considered as required capabilities to execute a design decision which have to be supported by the engineering organisation. The tables of the perspective “Tool Chain” are considered as provided capabilities which are supported by the tool chain of the engineering organisation.

No information gap based on Problem 5 is given if the following property holds:

Property 4: {For all design decisions to be executed within an engineering activity there is an engineering tool named within this activity with the capability to execute the design decision}.

No information gap based on Problem 6 is given if the following property holds:

Property 5: {For each design decisions to be taken holds: for all skills required to take the design decision there is a human role providing this skill} AND {For each design decisions to be taken holds: for all knowledge required to take the design decision there is a human role providing this knowledge} AND {For each design decisions to be taken holds: for all competences required to take the design decision there is a human role providing this competence} AND {For each tool to be used for a design decisions holds: for all skills required to use the tool there is a human role providing this skill} AND {For each tool to be used for a design decisions holds: for all knowledge required to use the tool there is a human role providing this knowledge}.

To make the fraction and gap evaluation more efficient the following checklist can be exploited:

(1) [Is the artifact consumed by an activity of the engineering process]

AND

[Is the engineering artifact not created by an activity or provided from the EO environment]

⇒ Fraction based on Problem 1

---

2 The consideration of Problem 7 is not possible without considering the semantics of the engineering artefacts and its content. Thus it will not be considered for fraction evaluation.
(2) [Is the artifact not created by the same activity of the engineering process consuming it]
   AND
   [the data format the engineering artifact is provided within by the creating tool is not the same data format the consuming tool can understand]
   ➔ Fraction based on Problem 2

(3) [Is the artifact not created by the same activity of the engineering process consuming it]
   AND
   [Is the intersection of the modeling capabilities of the export data formats from the creating tool of the artifact and the import data format of the consuming tool of the artifact smaller than required modeling capability to model the engineering artifact]
   ➔ Fraction based on Problem 3

(4) [Is there an engineering activity with a design decision to be taken within and a required tool with special engineering capabilities named to execute the design decision]
   AND
   [Is the intended design decision not part of the set of design decisions executable by the engineering tool]
   ➔ Gap based on Problem 5

(5) {{[Is there an engineering activity with a design decision to be taken within] AND {{{[Is there a competence required to take the design decision of the activity] AND [There is no human role named in the activity providing this competences]] OR
   {{[Is there a knowledge required to take the design decision of the activity] AND [There is no human role named in the activity providing this knowledge]] OR
   {{[Is there a skill required to take the design decision of the activity] AND [There is no human role named in the activity providing this skill]]}}
   OR
   {{[Is there an engineering activity with a tool to be used within] AND {{{[Is there a competence required to use the tool] AND [There is no human role named in the activity providing this competences]] OR {[Is there a knowledge required to use the tool] AND [There is no human role named in the activity providing this knowledge]] OR {[Is there a skill required to use the tool] AND [There is no human role named in the activity providing this skill]]}}
   ➔ Gap based on Problem 6
2.4. Tool chain scenarios

Within this chapter different scenarios of tool chain integration are described. These scenarios are important when it comes to gaps and fractions within the engineering process, as the degree of tool integration is crucial to the seamless information flow within the process (see section 2.2).

2.4.1. Best of bread tools

The lowest level of integration can be found in the “Best of Bread” philosophy. Here the engineering process is executed using the best fitting engineering tools for the individual engineering activities. Thus a tool chain has to be established. Within this tool chain the engineering artifacts have to be exchanged among tools using appropriate interfaces. The implementation of these interfaces can be made using proprietary or standardized data exchange formats (Diedrich et al. 2011), (Drath et al. 2011) were standardized data exchange formats should have the higher priority as they can provide larger benefits.

The most positive aspect of the “Best of Bread” philosophy is the adaptability of the tool chain to changing engineering process requirements and changing technology conditions. The replacement of tools within the chain by more suitable tools is very easy supposed the tools provide standardized interfaces for data exchange. But the tool chain is not providing an integrated data model for all relevant information. Data consistency and data integrity have to be ensured independent from the used tool chain. A tool independent implementation of a tool crossing engineering process management and a data consistency management system is required. Another disadvantage of this approach is the necessary efforts to keep consistency and interoperability among the tools in case of tool updates if no standardized data exchange formats are used.

Within the mechatronical engineering process a tool chain following the “Best of bread” philosophy has to consider the different engineering disciplines of the used tools. Each tool and its interface have to support the exchange of the relevant part of the plant components involved. The tool independent data consistency management system has to ensure the data consistency related to the engineering artifacts. Within this philosophy the different engineering tools have to implement less modular engineering concepts and general engineering activities. Relevant modular concepts are Hierarchies of Plant components, Modularization of Plant components, Level of Detail, Libraries of Plant components, Variants of Plant components, and Templates of Plant components. General engineering activities to be implemented should be Variant Development, Template Development, Instantiation, Engineering artifact Library Management, and User Guidance within the Engineering Process.

In contrast to the “One for all” philosophy the tool independent data consistency management system has to implement the modular concepts Facets Description of Plant components, and Dependencies between Facets, Facet Aggregation and the general engineering activities Consistency Management, and Change Management and Change Impact Analysis. This will make the project management using a tool chain much more complicated.
2.4.2. Integrated tool chain

The “Integration Framework” philosophy is in between the both philosophies of "best of bread tools" and "All-in-one tools". It tries to provide the benefits of both philosophies while not having its drawbacks. The “Integration Framework” philosophy is characterized by the combination of a centralized data management system implemented as middleware with a tool chain with standardized interfaces. Examples of such Integration Frameworks are the SIEMENS Team Center or the Engineering Service Bus (Moser et al. 2011). The Integration Framework will provide a syntactic and semantic integration of the data sets relevant for the engineering tools associated based on a common data model implemented within the middleware. Using this common data model and the standardized interfaces the main benefits of this philosophy are the easy adaptation of the framework to changing engineering process requirements and changing technology conditions while ensuring consistency within and among the engineering artifacts. The drawback of integration frameworks is seen in the required setup of the common data model at the moment of tool chain definition. In this moment the common data model has to be developed on the basement of the data models of the different involved tools. Within the mechatronical engineering process the integration framework can be based on the engineering artifact establishing the common data model. Thus the different tools involved will implement the modular concepts Hierarchies of Plant components, Modularization of Plant components, Level of Detail, Libraries of Plant components, Variants of Plant components, and Templates of Plant components and the general engineering activity Variant Development, Template Development, Instantiation, modular concept Library Management, and User Guidance within the Engineering Process. The underlying middleware and data storing facility should implement the modular concepts Facets Description of Plant components, Dependencies between Facets, Views on Plant component, Level of Detail, Facet Aggregation, Libraries of Plant components, Revisions of Plant component Data, Variants of Plant components, and Templates of Plant components and the general engineering activities Consistency Management, Variant Development, Flexible View Generation, Template Development, Instantiation, engineering artifact Library Management, User Guidance within the Engineering Process, and Change Management and Change Impact Analysis.

2.4.3. All-in-one tools

The highest level of integration can be found in the “All-in-one” philosophy. Here a comprehensive tool is used enabling the engineering organization to execute all necessary engineering steps (or at least the most) with one engineering tool and integrating all relevant engineering disciplines. Examples of such tools are EPlan engineering center, SIMATIC automation designer or COMOS. Within such an engineering tool the engineering artifacts are based on a tool covering integrated data model over all engineering disciplines. The engineering artifacts are stored and managed in a centralized way. Thereby, it is easy to ensure consistency within and among the engineering artifacts. Within such a tool it becomes possible to provide essential engineering process support. This may range from automatic engineering process management with process progress monitoring over change
Deliverable D4.3
Integration of GRACE multi-agent systems with manufacturing CAE systems

management and tracing up to automatic execution of discipline crossing engineering steps to name only some of the relevant capabilities. The major drawback of this philosophy is the tool complexity. This complexity makes an adaptation of the tool to changing engineering process requirements and changing technology conditions very complicated and slow. Within the mechatronic engineering process such comprehensive tools have to implement data structures enabling the design and use of plant components. Thereby, modular concepts like Hierarchies of Plant components, Facets Description of Plant components, Dependencies between Facets, Views on Plant component, Level of Detail, Facet Aggregation, Libraries of Plant components, and Variants of Plant components as well as general engineering activities like Consistency Management, Refinement of Plant components, Variant Development, Template Development, Instantiation, engineering artifact Library Management, User Guidance within the Engineering Process, and Change Management and Change Impact Analysis can and should be implemented by the universal tool.
3. Description of selected engineering tools

Within this chapter different selected engineering tools will be described. Additionally it will be shown if and (if yes) how mechatronic concepts are realized within these tools. The tools are listed in alphabetical order.

Notes:

- AutomationML itself is not a tool but a data exchange format. It is listed here to give insights in what is needed from data exchange formats if they are used as a data backbone for and integrated tool chain
- JADE, NetBeans IDE and Protégé are no typical industrial CAE tools. They are especially used within GRACE for the engineering of the MAS. Thus, they are shortly described but not analyzed in detail as the other industrial tools to give input for the additional tool requirements for industrial CAE tools supporting the GRACE MAS architecture.

3.1. AutomationML

3.1.1. Tool description

Within a typical engineering value creation chain, data exchange is a significant bottleneck. Due to these reasons, the neutral data exchange format AutomationML usable within the engineering process of manufacturing and process systems was developed to improve the exchange of data among the various used tools. This neutral data format follows the needs of engineers regarding interoperability of tools within the today’s heterogeneous tool landscape. Additionally, it intends to become an enabling technology for the topics „virtual commissioning“ and mechatronical engineering.

AutomationML is intended to store and exchange different information created and/or applied during the plant engineering. This information are usually created and maintained in different tools following various disciplines, guidelines, business rules and naming conventions. This leads to complex and cross linked data.

In this section the basic AutomationML concepts:

- Plant topology description,
- Geometry and Kinematic,
- Behavior description,
- References and relations,

are introduced as well as the main AutomationML architecture and the extended AutomationML concepts.

AutomationML supports the storage of the following basic concepts:
**Plant topology**

The plant topology describes the manufacturing system as a hierarchical structure of objects. This structure is stored in the AutomationML top level format. Different aspects of an item are directly associated to the corresponding AutomationML object. Each object has individual properties and interfaces. As basis data format for the plant topology CAEX (IEC 62424) is being used.

**Geometry and Kinematic description**

The geometry and kinematic information description of an object comprises its complete geometric 3D description, its physical interconnections, and its kinematical conditions. Geometry and kinematic information is stored by means of the data format COLLADA (Arnaud und Barnes 2006) of the industry consortium “Khronos group” as separate XML document. AutomationML defines special reference mechanisms to link CAEX data objects with a COLLADA file.

**Behavior description**

The behavior of objects is described as a sequence of actions. These are stored by means of the data format PLCopen XML (PLCopen Konsortium 2009) as SFC including the corresponding I/O-relations and logical variables. Variables of the SFCs can be “published” within CAEX. This allows storing high level interconnections between them in CAEX.

**References and relations**

AutomationML distinguishes between relations and references. References on the one hand describe associations from CAEX objects to externally stored files. Relations, on the other hand, specify interrelations between CAEX objects. These are additionally used in order to link information in the top level format which is stored externally. For this, corresponding interfaces must be “published” in CAEX. They can be linked with standard CAEX link mechanisms.

![AutomationML base architecture](image)

**Figure 7: AutomationML base architecture**

The basic architecture of AutomationML (see Figure 7) consists of the standards CAEX, PLCopen XML and COLLADA. Thereby CAEX is the top level format and the link between all information. It stores the plant topology, while COLLADA stores geometric and kinematic
information and PLCopen XML serves for the storage of sequences and behavior descriptions. This architecture results in an inherent distributed file structure.

The application of CAEX is an important part of AutomationML. AutomationML specifies the concrete application of the CAEX concepts. For this, AutomationML uses the following CAEX means and special attributes, classes, interfaces and object identification rules:

- **Object identification**: AutomationML defines how to identify AutomationML-objects and AutomationML-classes.

- **InterfaceClassLib**: Interfaces serves for the description of relations between objects. AutomationML delivers a predefined interface library, which contains a number of abstract interface classes for general automation systems. These classes are syntactically and semantically well defined and allow the modeling of user defined interface instances.

- **RoleClassLib**: A role library serves for the definition of abstract role classes. A role class explains the general functionality of a CAEX object within its context.

- **SystemUnitClassLib**: A system unit library allows storing user defined AutomationML classes. It is intended as user defined library of modeling objects. AutomationML does not specify a certain SystemUnitClassLib, but it defines rules how to design them.

- **InstanceHierarchy**: Instance hierarchies store concrete project data and are, therefore, the core of AutomationML data. The InstanceHierarchy is a hierarchy of object instances with its own individual properties, interfaces, references and relations.

Additional to the basic concept and the basic architecture AutomationML specifies the extended concepts:

- AutomationML Port Concept
- AutomationML Facet Concept
- AutomationML Group Concept
- Support of Multiple Roles
- Process-Product-Resource Concept

These concepts are explained below.

**AutomationML Port Concept**

The Port concept allows a high level description of complex interfaces. AutomationML ports consist of a set of AutomationML interfaces that belong together. They can be understood similar to plugs or sockets.

**AutomationML Facet Concept**

AutomationML facets allow the storage of a subset of attributes and interfaces of an AutomationML object. They can be considered as a view on engineering data.
**AutomationML Group Concept**

AutomationML Groups allow the storage of separate views on a subset of AutomationML objects. They can be used to filter objects of the plant tree for different engineering tools.

**Support of Multiple Roles**

In addition to CAEX, AutomationML defines how to specify multiple role support for an object instance. Multiple roles are of interest, if an object can have multiple functionalities. An example is an all-in-one printer that is a scanner, a printer and a fax device at the same time.

**Process-Product-Resource Concept**

The Process-Product-Resource concept allows high level structuring of engineering data based on a process-centric, product-centric or resource-centric view including relations between them.

3.1.2. Mechatronical concepts

Additionally to the introduced concepts above AutomationML provides means for the application of MUs. Therefore, a defined structure has been defined, see Figure 8.
With this structure the same information sets related to MUs as presented in Deliverable 4.2 can be applied. Thus, each information part is mapped to a so-called CAEX Internal Element of AutomationML with appropriate substructures as expressed in Figure 8. In contrast to the SIMATIC AD were the control system behavior is stored within the SIMATIC AD object hierarchy as SIMATIC AD objects AutomationML references PLCopen XML documents that contain the corresponding behavior information. The same holds for geometry in kinematics information which is modeled within COLLADA documents. Again the capital letters show how the different information sets of an mechatronical unit, compare Figure 17 und Figure 18, are integrated in the AutomationML data structure.

In contrast to the concept of the SIMATIC AD the AutomationML MU concepts maps basic semantically and structure information with the help of the role concept.
3.2. Bentley Microstation

3.2.1. Tool description

Bentley Microstation V8i trial version SELLECT series 2 (in the following only MS) is a CAD based engineering tool initially intended for construction of mechanical and later on also electrical systems. It can be applied to various application domains including manufacturing systems, process industry, architecture and buildings, city planning, and much more.

MS is a tool based tool, i.e. MS works on the basis of several tool internal tools developed for a special modeling purpose. The overall tool use is characterized by the, therefore, necessary switching between different tools and its detailed use.

Figure 9 represents the main window of MS. It contains

- the main menu bar for basic file functions like save and open, basic modeling functions like select and undo, basic element definitions, basic settings for models, starting tools, using utilities, window settings, and help;
- the primary tool box with tools for cell library management, model management, etc.;
- the view windows representing the working plain for the different models of a project;
- the menu for main model element parameters;
- the main modeling toolbox; and
- the windows for individual tool settings.
Models are seen as basic modeling means of MS. Within one project file different models can be defined which are used separately and independently. Each model contains its own working plain, a 1:1 size surface of the system to be modeled.

The models itself consist of different model elements which have several parameters to be described including graphic parameters like size and position, color and layer number, etc. (see left part of Figure 10 as example). Model elements can be grouped.

Model elements can be manipulated using tools which are organized in different tool boxes. MS provides different tool boxes for the creation of model elements, organization of model elements, manipulation of model elements and for the overall model management. The right part of Figure 10 displays the main tool box with the drawing tools part opened and the opened tools menu with different tools available.
An essential mean for the management of sets of model elements are cells. Cells are seen as reusable model elements or combination of model elements which are organized in desperate cell files which can be attached to a project or a model (see Figure 11).

Figure 10: Element parameter example and tool box examples

Figure 11: Cell file management and use
3.2.2. Mechatronical concepts

The application scope of MS is limited to the modeling of geometric systems, i.e. to CAD. Thus it will mainly cover geometry information. In addition it is able to represent wiring information, interface information and, by using different colors of the model elements, device hierarchies and its association to different engineering disciplines like mechanics, electrics and control. This is represented in Figure 12.

Cells are the main elements for the development of ENGINEERING ARTIFACTs within MS. Cells are a combination of model elements grouped and stored for reuse. A cell can be established by a combination of elements and/or cells which are grouped and placed in a cell library file of *.cel type.

Each cell is then usable as modeling element which can be placed, moved, adjusted, etc. like each other modeling element.

The hierarchy of cell elements and its association to normal elements is given in Figure 13.
A cell itself is initially seen as a set of elements. Each element within a cell has its own parameters. To make a cell to a engineering artifact it is essential to use these parameter to signalize the engineering discipline a cell element belongs to. For example the colours can be used to identify whether an element geometry belongs to the mechanical discipline or the electrical or the control engineering discipline. Figure 14 represents an example for this version. Here a complex cell is represented. Within this cell red coloured elements belong to the control engineering discipline (sensors and actuators) and white and orange coloured elements are or mechanical engineering discipline.
3.3. Comos / Automation Designer

3.3.1. Tool description

The SIMATIC AD is an engineering tool for digital engineering of Siemens AG. It is based on the software system COMOS. Within this work the Version 9.0 Vega with service pack 198 is used.

For a general description we refer to the SIMATIC AD handbooks and its antecedent versions like (COMOS Industry Solutions 2008).

Within (Barth 2009) and (Foehr 2010) a mechatronical engineering process centric application description of the SIMATIC AD is given.

This section will concentrate on tool concepts of the SIMATIC AD relevant for the implementation the modular conceptss and general engineering activities described above.

Generally, SIMATIC AD is based on the following set of tool concepts:

- Base objects and planning objects
- Base object project
- Planning project
- Inheritance
- Data views
- Queries
• Mass data processing
• Reports
• E-blocks
• Scripts

**Base object and planning object**

The SIMATIC AD follows an object oriented modeling philosophy. Therefore, basic objects are used as the fundamental information object were each other object is derived from. Each base object consists of attributes, symbols, interfaces and scripts as basic and not dividable information objects as well as possible further base objects. This is depicted in Figure 15.

Attributes are used to represent special information about the base object like parameters, names, etc.

Symbols are used as graphical representation of a base object within a specific information view. For example, a base object describing a drive has a special representation within a wiring plan.

Interfaces describe the possible connection places to other base objects independent of the connection nature. They can represent electrical, mechanical, logical or further connection possibilities.

Finally, scripts are programmable engineering artifacts useable to manipulate information covered by a base object. They will be described in detail later on.

Base objects describe classes of engineering artifacts ranging from physical objects like inductive sensors, stepping motors, or complete gantries, to logical objects like PLC programs, wiring plans, or Gantt based sequences usually. Thus base objects can be the result of the product business.

The structure of a base object is depicted in Figure 15.

![Figure 15: Base object structure with project structures](image)
A planning object is an instantiation of a base object. It is used to describe an individual engineering artifact. Thus it is used within the solution business and part of solution projects.

**Base object project and planning project**

The base object project is the library of engineering artifacts useable within engineering processes. It is the basis for the further work of an engineer within an engineering project which is stored in the base object library. The base object library is the source of the base object project. It is completed by the results of the product business as well as by evaluation of the results of the solution business regarding re-useable mechatronical units.

**Figure 16: Relation between base object and planning object**

Within the solution business the planning project contains all instantiated planning objects. The planning objects can be created by instantiation from a base object, completed with information, integrated in the project topology and finally used within further engineering activities in the planning project. This is depicted in Figure 16.

Following the VDI recommendation 3695 (VDI 3695) it is intended that the following cyclic process can be followed.

1. In a product business project
   - the base object project can be extended by creation of new base objects and
   - the resulting base objects can be written to the base object library.

2. In a solution business project
   - the base object project is an instantiation of the base object library,
   - the base objects can be adapted to the needs of the solutions business project by enrichment or new creation, and
   - the planning project is created by instantiation and enriching of base objects to planning objects while the base object library is unchanged.
3. In a project evaluating the results of an solution business project
   - the base object project can be changed and extended by creation and or
     enrichment of base objects and
   - the resulting base objects can be written back to the base object library.

Each object is located within two object hierarchies, location hierarchy and equipment hierarchy within a planning project. The location hierarchy defines the position of an object while the equipment hierarchy defines the hierarchy of mechatronical units.

**Inheritance**

The inheritance concept enables to inherit more detailed objects from existing base objects within the base object project. Here, the derived child base object will have all information/structures from its parent object a can get additional information/structures. Inheritance is not usable within the planning project.

**Data views**

Data views are views on information within base objects or planning objects. They provide a segmentation of information of an object. Thereby, information belonging to different engineering disciplines can be modeled separately. Examples for data views are register cards associated to objects (both planning objects and base objects) containing attributes.

**Working layer**

Working layers are a segmentation of project data access beyond the use of planning projects and base object projects. Working layers are usually specified by a project administrator defining the data access rules for the different persons involved in an engineering project. These working layers can be defined for planning projects as well as base object projects.

Changes within a working layer of a project will be valid only within this working layer. Thereby, the engineering activities of different persons can be decoupled without the problem of interferences of the engineering activities and the possible problem of changes within the own work caused by other workers.

Working layers can also be condensed to one after finalization of parallel engineering resulting in a common project.

**Query**

Queries are search and filter functions useable to search within projects. Starting from a previously selected object within the project all sub-objects with a previously defined property will be selected within a result list.

Queries can be combined with other tool concepts. Thereby, the quality of the search can be improved or the search results can be further processed. Here, especially scripts can be exploited.

**Mass Data Processing**
Mass data processing is a tool function useable to integrate or change the same engineering information within a set of previously selected objects. Similarly to queries within mass data processing objects are selected based on a previously defined property starting from an object of the project. All selected objects will then be extended / changed / … in the same way.

As queries, mass data processing can be combined with other tool concepts to improve the processing quality and efficiency. Therefore, the most important role plays the script concept.

**Report**

Reports are a tool concept enabling the presentation of information within a project following special presentation rules. They enable an engineering discipline conform presentation. Reports can be distinguished with respect to its ability to change objects presented within.

The first type of reports is the analyzing report. They analyze the information stored within a project with respect to special properties and present them in a predefined layout. But they are not able to change or add information within the project. Examples of these reports are signal lists or data sheets.

The second type of reports is the interactive report. This type of report extends the presentation capabilities of analyzing reports by the capability to change information. Thus, using this type of report information within the project including the object structure can be changed. Examples of such reports are function plans or wiring plans.

Interactive reports are the main instrument for the engineering activities oriented towards one engineering discipline like control programming, which is made by using function plans or the wiring planning using wiring plans.

**E-block**

E-blocks are a tool concept useable to implement the execution of special but multiple occurring engineering activities. They are based on user interaction. Based on user information special engineering activities are executed over a complete project or objects within the project, respectively.

Within the analyzed version of SIMATIC AD a set of basic e-blocks as well as several special e-blocks were available. The available basic e-blocks are:

- Merge of objects
- Move of objects
- Definition of object position within equipment hierarchy
- Definition of object position within location hierarchy
- Definition of implementation relation between objects
- Definition of sub-objects
- Execution of decision table
• Report execution
• Definition of terminal implementation relation
• Definition of channel implementation relation
• Definition of bridge relation between objects
• Connection of interfaces
• Change of base object

There is also an e-block version available usable as self-defined e-block.

Generally existing e-blocks can be extended by using scripts to include additional functionality. Thereby, user-defined engineering activities can be implemented manipulating planning objects in a special way by including, merging, or processing information.

Thus, scripts are the basement of the functionality of e-blocks. Each e-block contains its own script and extends this script by a user interface usable to provide user information to the script prior to script execution. For the standard e-blocks the user interface contains three parameters for source object target object and naming of standard e-block function. For self-defined e-blocks this user interface can be changed.

**Script**

In contrast to the tool concepts named above scripts can’t be used independently from defined objects. Usually they are embedded within the object structure by associating them to objects like reports or e-blocks. They are executed on a predefined point within the lifecycle of the object, they are attached to. Examples of embedding scripts to objects are scripts within reports or scripts which are embedded in e-blocks. One example of a special script is the script for generation of a unique object name called in the moment of object generation.

Scripts are programmed in the VBscript programming language. They can be changed within the base objects in the base object project. The complete script system of the SIMATIC AD can be changed following user needs including the integration of additional scripts for special purposes.

### 3.3.2. Mechatronical concepts

The realization and application of mechatronical concepts based on SIMATIC AD is based on the definition of mechatronical structures and information involved in ENGINEERING ARTIFACTs describing mechatronical objects as given in the literature (see Harashima et al. 1996, Thramboulidis 2008, Wagner et al. 2010) and within standards (see VDI 2206).

The information set covered by the engineering artifact can be classified following engineering discipline related (Kiefer 2007), plant structure related (Drath 2010), or data related (Schorn und Große) structures. But all of these approaches are not complete and have to be combined. The resulting structure is depicted in Figure 17.
The named information sets consists of:

- The process control data consists of all control relevant information including control code and control code specifications of any kind and signal information like signal list and variable definitions.

- The mechanical data cover all information about the mechanical construction including geometry and kinematics data.

- The electrical, pneumatic, and hydraulic data describe the electrical, pneumatic, and hydraulic construction of the MU including the connections and wiring of the different types and its plugs.

- The topological data cover the hierarchy of MUs and devices. They give an overview about the structuring and the interfaces within the hierarchy.

- The function describing data will give a functional description of the MU. This contains relevant functional parameters, and technological descriptions and guidelines, and functional models of the uncontrolled and controlled behavior of the MU.

- Finally, the generic data summarize further organizational, technical, economical and other data. They cover for example article codes and manufacturer identifications and addresses, weight and size of the MU, supply information for electrical and other power, costs for acquisition and maintenance, and user manuals.
For more information see (Lüder et al. 2010).

The modeling of these information sets within SIMATIC AD is based on the idea that MUs will provide functions to the overall system. These functions can be primary functions directly used within the manufacturing process like transportation functions in the case of conveyors, manufacturing functions in the case of machines, cells, or robots, or supporting functions in the case of clam sets. In addition to the primary functions there can be secondary functions required for correct behavior of the MU. Such functions can be maintenance functions or preparation functions like manufacturing parameter adjustment.

Within the suggested information structuring within a engineering artifact representing a MU the internal structuring of the functions as well as the structuring of the overall engineering artifact and its MU representatively should be standardized.

Each function is controlled by appropriate control code. This code implements a behavior specification given by a program organization unit (POU). To connect the functions to the underlying MUs and devices and to parameterize them, appropriate interfaces and parameters are defined. Hence, the engineering artifact representing the MU contains structures for these control information.

To provide the functions the MU consists of lower level MUs and devices which are given within the devices part. To be used from the higher level MUs each unit has an execution interface with appropriate parameters for the overall MU. Hence, the engineering artifact representing the MU contains structures for these interface information.

The behavior of the MU will be described within the sequences part of the information structure. Here models of controlled as well as uncontrolled behavior can be integrated for analysis purposes like virtual commissioning. Hence, the engineering artifact representing the MU contains structures for these behavior information.

Finally there is a separate sub-information set for the geometry and kinematics information.

The described structure is given in Figure 18.
The capital letters within Figure 17 and Figure 18 give the mapping of the different data sets relevant for the engineering artifact to the suggested representation in SIMATIC AD. Thereby, it can be seen, that:

- the engineering artifact itself contains functional parameters (F), functional descriptions (G), and generic data (J),
- the functions (primary as well as secondary) directly covers the technological descriptions (G),
- the function sub-structure POU gives the controlled behavior (I),
- the function substructure Code gives control code (B)
- the function substructure device interface + parameters gives parts of the signal information (A), electrical, hydraulic, and pneumatic data (D), and functional parameters (F),
- CAD data / kinematics gives geometry and kinematics information (C),
- the devices gives the topology information (E),
- the sequences give the controlled (I) and uncontrolled (H) behavior models\(^3\), and
- the execution interface and parameters gives further parts of signal information (A), electrical, hydraulic, pneumatic data (D), and functional parameters (F).

\[^3\] The uncontrolled behavior of an MU is the overall behavior reached by reaction of the MU on all arbitrary sequences of external inputs. It subsumes all behavior possibilities the MU contains. The controlled behavior of a MU reached by reaction of the MU on a special sequence of external inputs represented by a control program executed in one or more controllers, e.g. robot controller, PLC etc.
This described implementation of engineering artifact is used as basement for the following consideration of implementation strategies for the modular concepts and general engineering activities within SIMATIC AD.

3.4. JADE

3.4.1. Tool description

Multi-agent systems can be adequately developed using usual object-oriented languages, such as Java and C++. However, the development of multi-agent system solutions requires the implementation of features not supported by usual programming languages, such as message transport, encoding and parsing, white and yellow pages services, ontologies for common understanding and agent life-cycle management services, which increases the programming effort. The use agent development platforms, that provide the previously referred features and services, makes easier the development of agent-based applications and reduces the programming effort (and also providing a higher level of abstraction of some technical details).

A significant set of platforms environments for agent development is available on commercial and scientific domain, providing a variety of services and agent models, which differences reflex of the philosophy and the target problems envisioned by the platform developers. Among the broad number of available agent development platforms, the following platforms were analysed (these platforms share between them the fact of being FIPA compliance): ZEUS, FIPA-OS, Java Agent Development Framework (JADE), Grasshopper, JACK and April. Jade is probably the well-known and the most used framework to develop agent-based systems.

JADE is a Java-based architecture that uses the Java Remote Method Invocation (RMI) to support the creation of distributed Java technology-based to Java applications. JADE aims to simplify the development of multi-agent systems by providing a set of services and agents in compliance with the FIPA specifications, e.g. naming service and yellow-page service, message transport and parsing service, and a library of FIPA interaction protocols ready to be used (Belliflemin et al. 1999). Note that in the essence, the agents developed using the JADE platform are Java Threads, which makes the debugging of multi-threading very difficult; consequently, some tools have been developed to simplify the development of agent-based solutions, being every single tool provided by JADE packaged as an agent itself.

JADE provides the mandatory components defined by FIPA to manage the agent platform, which are the:

- Agent Communication Channel (ACC), which is responsible for the conversation channel, supporting the communications among the agents and offering interoperability of the components.
- Agent Management System (AMS), which provides white pages and agent life cycle management services (controlling the access to the platform, authentication, and registration), maintaining a directory of agent identifiers and states.
• Directory Facilitator (DF), which provides a piece of shared memory offering the yellow pages services as defined in the FIPA specifications, and the capability of federation within other DFs on other existing platforms. Using this tool, it is possible to register services and search for agents offering specific services.

In this way, the main container of a JADE multi-agent system application is composed of the ACC, AMS and DF agents, and by an RMI registry (that is used by the JADE for intra-platform communication), as illustrated in Figure 19.

![Figure 19: Containers in the JADE Platform](image)

JADE uses the concept of behaviours to model concurrent tasks in agent programming (Bellifemine et al. 1999).

The communication among the agents is performed through message passing, where FIPA-ACL (Agent Communication Language) is the agent communication language to represent messages. JADE provides the FIPA SL (Semantic Language) content language and the agent management ontology, as well as the support for user-defined content languages and ontologies that can be implemented, registered with agents, and automatically used by the framework.

The Remote Management Agent (RMA) provides a Graphical User Interface (GUI) for the remote management of the platform, allowing monitoring and controlling the status of agents, for example to stop and re-start agents, Figure 20. The RMA allow a fully control of an agent life cycle from a remote host. When the Jade platform is started, a default container is created, which holds the RMA itself, the DF, and AMS.
The RMA agent provides a set of graphical tools (packaged as agents) to monitor the state of the agents and to support the debugging phase, usually quite complex in distributed systems, such as the Dummy, Sniffer and Introspector agents.

The Dummy Agent is a monitoring and debugging tool that allows to edit, compose and send ACL messages to agents, and to receive and view messages from agents. When the complexity of the multi-agent system increases, it is very useful to use a tool to check the exchanged messages between the agents. The Sniffer Agent, illustrated in Figure 21, is a debugging tool that allows tracking messages exchanged in a JADE agent platform using a notation similar to Unified Modelling Language (UML) sequence diagrams. It can be analyzed the type of message, the FIPA protocol, the ontology language and its contents.
The Introspector Agent, illustrated in Figure 22, allows monitoring and controlling the life-cycle of a running agent, its exchanged ACL messages (incoming and the outgoing messages) and the behaviours in execution (allowing to execute them step-by-step).

In distributed systems it is important to have a service of yellow pages, where agents register their services and skills to be found by other agents. In the JADE platform, this concept is named as Direct Facilitator (DF) following the FIPA specifications. This yellow pages services allow to see the details of agents registration, deregister the agents, modify some descriptions, or as the greatest utility of the yellow pages, look for a service that is performed by another agent. Figure 23 illustrates the screenshot of the DF.

JADE also provides other features such as good documentation and an active mailing list to support technical problems.
3.5. **NetBeans IDE**

3.5.1. **Tool description**

NetBeans is an integrated development environment (IDE), free and open-source. It is implemented in Java and can run on any operating system where it was installed a Java virtual machine (JVM). It is a tool to support the development of software which provides the programmer with a wide range of resources, such as compilers, interpreters, debuggers, as well as a large set of libraries and application programming interfaces (API). It also has an advanced multi-language editor with support for multiple languages, such as Java, C, C++, PHP, JavaScript, among others. The meeting of these tools allow streamline the programming process supporting the development of several applications, such as websites, mobile applications, stand-alone applications, distributed applications, among other.

In NetBeans the basic building block is modules. Modules are collections of related classes and interfaces descriptions with other internal modules. In fact, keyword is modularity. The “NetBeans runtime container” is the core of the platform and is the collective name of the modules presented in Figure 24. The constitutive modules of the IDE run on it and also the programmers' applications.

![Figure 24: NetBeans runtime container](http://platform.netbeans.org/tutorials/nbm-runtime-container.html)

A brief overview of each of these six modules is provided in following:

- **Startup** (org-netbeans-core-startup) – Provides the main method of programmer’s application, as well as all the code needed for starting it up.
- **Bootstrap** (org-netbeans-bootstratp) – Enables the runtime container to understand what a module is and how to load and compose them into one application.
• Filesystem API (org-openide-filesystems) – Gives programmer’s application a virtual filesystem.
• Module System API (org-openide-modules) – Gives access to the lifecycle of the modules in programmer’s application.
• Lookup API (org-openide-util-lookup) – Provides a generic communication mechanism for inter-modular interaction.
• Utilities API (org-openide-util) – Includes several utility classes shared between the other modules in the runtime container.

The user interface of NetBeans is composed by several regions as depicted in Figure 25.

![Figure 25: NetBeans user interface](image)

The central element is the editor that permits insert code. In the left top of the figure is possible to see the project tree. In the left bottom of the figure is possible to see a navigator that permits to explore the elements of a specific class (like the attributes and methods).

Swing is a standard UI toolkit on the Java desktop that can be used throughout NetBeans. GUI components can be dragged and positioned from a palette, onto a canvas. Clicking into GUI elements is possible to edit their properties directly in place. Figure 26 presents a view of the Swing GUI Builder, exhibiting the pallet of Swing components on the right.
In the center is possible to see the construction of the GUI of Product Type Agent from the Grace system. The GUI Snapshot permits to debug a Swing GUI application without looking into the source code.

In NetBeans all the work happens inside a project. The concept of project could be understood like a set of code files plus associated information, permitting to compile, execute and debug applications. From the point of view of visible organization is possible to distinguish two main sets of files. The first one is related to the internal organization of the project and is managed by NetBeans in an automated way based on user configurations. As shown in Figure 27, those files are stored in the nbproject folder.
3.6. **Protégé**

3.6.1. **Tool description**

The development of ontologies is a complex task that requires the support of proper frameworks which assist the creation or manipulation of ontologies and are able to express ontologies in one of many ontology languages.

Examples of relevant criteria for choosing an ontology editor are:

- The degree to which the editor abstracts from the actual ontology representation language used for persistence.
The visual navigation possibilities within the knowledge model.

The incorporation of methodologies and languages, in an easy way.

The ability to import and export foreign knowledge representation languages for ontology matching.

The licensing costs of the ontology editor.

The use of these tools may lead to an easier ontological learning and also a more productive task in the design of ontologies, supporting the concurrent work of the ontology engineers and the domain experts.

Several frameworks are currently available, namely OntoEdit, WebODE, Protégé and Hozo. Protégé is probably the most used tool for the development of ontologies, either the development from the scratch, and the merging, importing, querying and export of ontologies. It is a free, open-source platform that provides a suite of tools to construct domain models and knowledge-based applications with ontologies. In Protégé it is possible to create ontologies based on different types of expressiveness and languages; additionally, there are many plugins to be used with Protégé, e.g. to support the validation phase and to export the ontology in different formats.

Protégé can be used to edit and verify the ontology correctness, since it is a free platform and it provides all necessary characteristics to support a suitable abstraction and technical implementation of the GRACE ontology in a graphical manner.

Figure 28 illustrates a screenshot of the Protégé tool.

![Figure 28: Screenshot of the Protégé editor](image)

In next figure, it is illustrated the Protégé editor in a more detailed way, highlighting the different tools provided by the editor.
Deliverable D4.3

Integration of GRACE multi-agent systems with manufacturing CAE systems

Figure 29: Details of the Protégé editor

The Class tab is responsible to give to the user the opportunity to create/edit the ontological classes. In the Properties tab the editor allow to create the properties and connect to a specific class. It is possible to express the type of property (object, String, Integer, etc). The SPARQL, is the responsible to query the ontological model, is created a query very similar with (SQL) and then executed, the result can analysed on the right side of the Query sector. The Instance tab permit the creation of values, the classes can be instantiated to create not a Schema but data with some properties of the schema model. The user has the opportunity to install or create him-self more plug-in for the editor. These sections (Class tab, Instances tab, etc.) are very mouldable and we can change the position.

Protégé architecture was developed under Jena Framework. Jena is one of the most widely used Java APIs for RDF and OWL, providing services for model representation, parsing, database persistence, querying and some visualization tools. Protégé-OWL has always had a close relationship with Jena, just like it is illustrated in Figure 30.
3.7. Tecnomatix Plant Simulate

3.7.1. Tool description

Tecnomatix Plant Simulation (TPS)\textsuperscript{4} is a discrete event system-based simulation tool for modeling, simulation, and analysis of digital models of manufacturing and logistic systems. Based on this simulation tool

- the functional structure of a manufacturing system,
- the material flow within a manufacturing system,
- the temporal conditions of the use of manufacturing resources,
- the control strategy of the manufacturing execution system layer of a manufacturing system

can be modeled, analyzed, and optimized.

TSP enables the structured design of manufacturing system models based on system components and its connection and concretization. After development of the manufacturing system models, these systems can be evaluated by simulation experiments. Thereby, TSP can be applied for

- Validation of developed structures and control strategies,
- Comparison of existing and planned systems variants, and
- Optimization of existing and planned systems,

\textsuperscript{4}See http://www.plm.automation.siemens.com/de_de/products/tecnomatix/plant_design/plant_simulation.shtml
All related to global manufacturing system down to local manufacturing seats or parts of them.

TSP contains capabilities for the automatic generation of statistics and result visualization supporting the use of the simulation experiment results.

For that purpose this tool provides means to model manufacturing systems by

- defining the manufacturing system resources and its interconnection by material flow systems and
- defining the major control rules for the processing and the flow of work pieces including its timing conditions.

Facing this functionality range of the tool can be exploited within the phases Raw Planning and Detailed Planning as well as Use and Maintenance of the Solution Business as depicted in Figure 31.

The tool can provide 2D and 3D views of system models and works in an object-oriented manner. The basic elements of the modeling capabilities are the classes of material flow objects, resource objects, moveable objects, and information flow objects, as well as the programming capabilities for methods behind based on SimTalk. All objects, usable for the modeling of manufacturing systems, are contained in the class library of the tool.

**Classes and objects**

Objects are the basic elements of the modeling capability of TPS. These objects are instantiated from the classes given or modeled in the class library.

Each object has attributes (acting like variables) and methods, defined in the basic library, related to it. They are usable to control the object behavior from the simulation point of view and from the visualization point of view. They can be set / changed / implemented based on predeveloped dialog as depicted in Figure 32.
All objects have a common set of basic methods and basic attributes. Examples of these predefined methods are

- `derive` implementing the inheritance of classes within the class library,
- `updateDialog` implementing the writing of changed attribute values in the attributes after user intervention,
- `duplicate` implementing the duplication of a class within the class library,
- `createObject` implementing the instantiation of a class within a network,
- `openDialog` implementing the start of a user interaction with an object,
- `sendMessage` implementing the interaction among objects, and
- `deleteObject` implementing the end of life of an object in a network.

Further methods and attributes are usually object specific predefined or can be defined by users depending on the modeling intentions of the user.

Classes can be developed within the class library. Here classes can be derived from other classes and detailed as it is usually possible in object oriented approaches. For detailing a class user defined attributes and user defined methods can be developed (see a larger explanation on library structure and use below).

**Moveable objects**

Moveable objects are objects used to model work pieces, transportation units, etc. transported and processed within a manufacturing system. The predefined basic moveable objects are vehicle, transport unit, and transport assistance unit.

Moveable objects will “flow” during the simulation process over the material flow objects and the resource objects. Within this flow they will be controlled by these objects exploiting the methods of all related objects.
Moveable objects have a basic set of predefined attributes including for example a name, size information and statistics information. Additional attributes can be defined for moveable objects in the dialog depicted in Figure 33.

**Material flow objects**

Material flow objects are used to model the logistics components of the manufacturing system. Examples of material flow objects are turntables, transport lines, flow control stations, storages, buffers, sinks and sources, which are direct material flow related, as well as mounting stations and unmounting stations, which are processing stations.

All material flow objects have a common basic set of attributes containing a name, material flow control information, timing information and statistics information. Additionally, user defined attributes can be attached to them using the dialog depicted in Figure 34.

For control purposes user defined methods can be implemented for material flow objects.

Material flow objects can be distinguished according to their activity level. Active material flow objects will take up moveable objects, process, store, etc. them for a time span controllable by various timing modes, and transmit them to successor objects depending on control decisions. Examples of active material flow objects are buffers and transfer lines. In contrast to that passive material flow objects will take up moveable objects but not transfer them to further objects automatically. To pass moveable objects to successor objects passive material flow objects have to be explicitly triggered by calling an appropriate method.
Resource objects

Resource objects represent human centered objects within the manufacturing system. Thus working places, foot paths, workers belong to the resources. In addition, brokers and exporters will model services provided and consumed by humans within the system.

Usually, resources are associated to the material flow objects single station, parallel station, mounting station, and unmounting station.

Information flow objects

Information flow objects are used to model the flow and processing of information in parallel to the material flow. They are also used to control the material flow.

The main information flow objects are the following:

- Methods and variables are used to implement the control of the modeled manufacturing system. Variables are global accessible objects exploited for information transfer within the simulation run. Methods are user implemented methods controlling the material flow based on object attributes and variables. Methods are implemented in the script language SimTalk.
- Lists and tables like cards, tabs or queues are used to provide arrays of information in a structured and easy accessible way.
- Triggers and generators are used to control the creation of objects.
- The attribute explorer is used to manage the attributes of the objects during the simulation run.

---

5 Attributes cover values of specific semantics modeling the current state and the behavior of an object. They are completely dependent on this object. In contrast to this, variables are independent objects covering an value with a user defined semantics of a specific data type.

Date interfaces are used to exchange information with further applications and tools.

Further objects

In addition exist objects related to a complete simulation project. One of the most important of these objects is the event manager (Ereignisverwalter). This object is used to control (start, stop, parameterize) the simulation run of a complete simulation project. Within each simulation project there shall be only one event manager.

Class library

The class library is the central repository for all predefined and user defined classes useable within a modeling and simulation project. It contains the basic libraries material flow, resources, information flow, surface, moveable objects, and tools which are predefined within the TPS.

The class library can be extended by developing user defined class libraries exploiting inheritance, enrichment and network definition (see below) and by importing existing library parts developed by different sources.

Within the class library the different classes are organized within folders which can be hierarchically structured. But this structure does not provide a syntactical or behavioral relation among classes of the same folder or classes in the folder hierarchy. It is only for semantical user assistance. The inheritance relation defining syntactically end behavioral dependencies among classes is completely independent from the folder structure.

Nevertheless, the folder structure of the class library is used to enable export, import, and management of classes and class library parts and provides the user with an organization and search structure for objects to be used within modeling activities.

Networks

Networks are special objects used to model manufacturing system structures within. They are organized within the class library and can be structured in the same way as all other objects.
Deliverable D4.3
Integration of GRACE multi-agent systems with manufacturing CAE systems

Figure 35: Class library example
Network objects provide a modeling surface to position objects and networks within a visual observable area and connect them to a network. As networks can be embedded in networks exploiting special interface objects of networks a hierarchical structure of networks can be established following the hierarchical structure of a manufacturing system.

**SimTalk**

Simtalk is a script based programming language used to implement user defined methods. These methods are executed on an interpreter based mode during the simulation run of the modeled manufacturing system.

SimTalk provides basic programming structures for implementing decisions based on IF .. THEN .. ELSE and implementing cyclic processing. In addition it provides possibilities to access and use attribute and variable values.

### 3.7.2. Mechatronical concepts

The representation of the complete data set relevant for the mechatronical concept as defined in deliverable 4.2 is restricted within TPS. This is based on the specialized focus and information modeling range reflected within TPS. TPS enables the modeling of the information sets

- Hierarchy of manufacturing cells and resources,
- High level 3D CAD data of the manufacturing system,
- Functional models and functional parameters of the resources and products, and
- Signal information of the resources and products compared to the information sets given in deliverable 4.2.
In general all these information are related to the manufacturing execution control (MES) layer of the manufacturing system. All other information of the plant component are not expressible in TPS as indicated in Figure 37. This figure depicts the focus of the capabilities of the TPS to express ENGINEERING ARTIFACTs with respect to its specific focus of use.

Nevertheless, TPS enables the application of mechatronic concepts within the engineering of manufacturing systems in a basic way. A basic mechatronic information object can be established using a network object and embedding within the network object

- the underlying resources of the mechatronical units (independent of either they are objects or they are networks)
- the control for the underlying resources as methods
- the necessary control information to the underlying resources as tables, variables, queues, etc., and
- the material flow interfaces as input and output objects derived from the interface object within the material form object library.

This structure is exemplarily modeled in Figure 38.
This approach enables the hierarchical structuring of the manufacturing system combining material flow with control decisions. It will not provide an explicit information flow hierarchy but an implicit based on global variables if required.

Independent of the made statements of this section it has to be mentioned, that TSP is focused on only one special engineering activity within the engineering process of manufacturing systems, the simulation of plant structures and controls. This is reflected by the portion of the information set of the plant component expressible within TSP.

3.8. Tecnomatix Process Designer

3.8.1. Tool description

The Tecnomatix Process Designer (PD) is a database based software system for the high level planning of manufacturing lines, workflows, and manufacturing processes for complete manufacturing system. It is intended to be used for:

- Overall process planning
- Comparison of alternatives
- Optimization of manufacturing line throughput
- Design of different variants of manufacturing lines and cells
- Evaluation of change impacts
- Cost calculation
- Cycle time calculation

and further activities.

The PD is based on a three tire architecture as given in Figure 39. The lowest layer is established by an Oracle server containing a set of databases. Each data base can contain...
sets of projects all represented in similar schemata. Within each project tables for resources, operations, and product are distinguished.

The second layer is represented by a so-called eM-Server\(^7\). It represents an intelligent interface between PD clients and Oracle database providing the data access and user management capabilities of the PD. In addition it provides access to the so-called system root, a file path storing additional files outside of the Oracle database. This system root is used for example to store 3D information files. Each eM-Server is bound to one schemata used within the Oracle databases. Hence, for each used schemata for project data structuring one EM-Server has to be exploited.

The upper layer of the architecture is provided by the PD client establishing the user interface of the PD.

This architecture will enable different users to use the system concurrently exploiting the same database. In addition this architecture enables an easy extension by additional clients as well as databases.

---

\(^7\) This naming has historical reasons. The PD has been emerged from the tool eM-Planner of the company Tecnomatix. It has been extended and renamed after the acquisition of Tecnomatix by Siemens.
The user interface of the PD is based on the internal data structure of the PD covering three types of general information: resources, products, and operations. These three categories follow the Product-Process-Resource categorization with operations for processes.

Thus, the main window of the user interfaces consists of parts for the product tree, the operations tree, and the resource tree and additionally of parts for the navigation tree, the graphical representation of objects and finally sets of menus for different purposes. This is depicted in Figure 40.

The navigation tree structure is the starting point for the navigation within the process database. It represents the content of an opened project and the libraries exploited within. Here, all relevant data can be accessed. From this tree structure all relevant information can be extracted by users to add them to the other three tree structures for resources, products, and operations (by drag&drop).

The product tree provides the part-subpart hierarchy of a product. It is a special representation of the Bill of Material of the product. It is mainly exploited to represent the product structures of all products provided by the planned manufacturing system and its structuring in product groups.

The operation tree is a representation of all manufacturing steps executed by the planned manufacturing system it represents the Bill of Operations relevant for the product set of the manufacturing system. Thereby, it provides a hierarchical structure of the operations set with an operation – suboperation hierarchy.

The resource tree provides a hierarchical structuring of the manufacturing resources of the manufacturing system exploiting a resource – subresource relation. Starting with a complete manufacturing system as highest level the resource tree the resource tree enumerates all assets used within the manufacturing system following the mechatronical hierarchy of system – cell – function units – subunits – component – device. It also includes all manufacturing tools, relevant human resources, etc.

The content of the three trees for resources, operations, and products are interlinked. Thus, there are relations defining which part of a product is manufactured by which operation using which resource.

The graphical representation provides a 3D view on selected objects. It is based on the object selected within the navigation structure which contains the three trees named above. The graphic representation is synchronized with the three trees and highlights the object selected in a tree.
The PD provides a library concept for resources and operations as well as for complete manufacturing systems with products. Within this concept libraries are intended as means for information transfer between different projects as well as for reuse. Therefore, in a library usually resources are stored accompanied by operations usually executed within these operations.

For the development of libraries a special user role is defined. This role is responsible for the creation, update and deletion of objects within a library. The use of the library content is open for each normal user (only rendered by library visibility rules). Normal users can import a library at project start (or at any other moment of project execution) and use the content in the planning process. In this case it is intended that the user will create an asset hierarchy in the project and complete this hierarchy by using prototype objects from the library.

In addition to the named elements PD can contain 3D modeling objects for the visualization of resources and operations in the system root. These objects will be accesses in the case a planned system is visualized in the graphic representation.

The structure of system root and library is usually synchronized as given in Figure 41.
3.8.2. Mechatronical concepts

Within PD different concepts are strongly distinguished, directories, administrative structures and model elements. The model elements can be either resources with 3D geometries, operations, or products with 3D geometries.

The product elements consist of a hierarchy of product parts representing the structural properties of the part – subpart relation. Each product part may be related to a 3D visualization object enabling the visual representation of the product in the different creation stages in the graphic representation field of the user interface.

Resources are seen as assets of any type including manufacturing resources, tool, humans, etc. Like products they establish a hierarchy based on resource – subresource relations and are related to 3D objects enabling a visual representation.

Both, resources and products, are interlinked using operations. Operations represent manufacturing activities executed by resources to create product stages (mounting, processing, etc.). Operations will have a hierarchical structure as well as a time and logic based sequencing possibly represented by PERT or Gantt Charts.

In any case the three types of objects will render the capability of PD to represent plant components.
Following the information sets the plant component can contain the PD enables the storing of the information given in Figure 42.

The first information set belongs to the topology information. It gives the possibility to store device and component hierarchies within the resource hierarchy.

The second set belongs to the mechanical information. It enables the storing of geometry and partially kinematics information.

As part of the function describing information technology descriptions, functional parameters, controlled behavior and uncontrolled behavior can be stored.

Finally the PD can cover generic data mostly belonging to organizational data, technical data and economical data like device IDs, vendor IDs, material descriptions, weights, costs, handbooks, etc.

All other information possibly belonging to a engineering artifact cannot be expressed in PD. Nevertheless it is possible to exploit mechatronical concepts within PD. The basement for this is the basic representation of a engineering artifact in PD. This basic representation is given in PD by a combination of resource objects, operation objects, and product objects. This structure is either directly contained in a planning data object of the navigation tree, a planning data object of a library, or distributed over the resource, operation and product tree. The first version is given by the Sation_630 object in Figure 43 while the second version is represented by the equally named station objects of resource, product, and operation trees given in Figure 40.
3.9. Tecnomatix Process Simulate

3.9.1. Tool description

Tecnomatix Process Simulate (PS) is a database based engineering software system intended to be used for manufacturing system design and validation. Based on a three tier software structure with a database and a file server as backbone a Server provides clients with the capability to work on engineered plant models. These plant models consist of different components related to manufacturing resources, manufacturing processes, and manufactured products implementing the PPR concept. Thus, PS is based on the same structure as PD using the same file formats and databases (see section 3.8).

Figure 43: Example of an engineering artifact in PD

The inherent possible distribution of engineering artifact within PD is a strong problem for the use of mechatronical concepts in PD limiting its effectiveness.

Figure 44: Tecnomatix Process Designer software structure
PS is intended to be used for e.g.

- 3D based system development and validation,
- path planning, reachability tests, and placing of robots,
- 3D resource planning including control strategies, and
- Manufacturing system simulation.

### 3.9.2. Mechatronical concepts

As PS is already based on the same data structure as PD it enables a similar implementation of the engineering artifact structure. Following the distinction between products, resources, and processes these three types of objects have to be combined to a plant component. Nevertheless, following the strong focus of the intended functionality of PS the information modeling capabilities of PS are limited with respect to PD.

PS enables the application of the following information sets:

- Topological data representing the internal hierarchy of a manufacturing system resource structure.
- Function describing data are part of the process structure. They represent the behavior of a manufacturing process including operation sequences and its detailed parameters.
- Mechanical data are part of the 3d CAD data of the different resources. They represent the mechanical construction of the resources representing its geometry and kinematics. They have to be available in JT data format.
- Generic data representing additional technical information of the resources like weights or energy consumption.
These information sets are integrated within one planning alternative (Planungsalternative). Thus, such planning alternative will be regarded as ENGINEERING ARTIFACT.

Figure 45: information sets covered by PS

Figure 46: engineering artifact implementation in PS
3.10. **Tecnomatix RobCAD**

**3.10.1. Tool description**

Tecnomatix RobCAD is a CAD/CAM based tool for modeling, simulation, and analysis of industrial robots and other controlled automatic devices in manufacturing systems. Its main characteristic is the modeling of geometric and kinematic structures and behavior.

Thus, within this tool

- the functional structure of a robot based manufacturing cell,
- the topological structure of such manufacturing cells,
- 3D geometry data and kinematic structure of the robots and further agile objects of the manufacturing cell,
- the process structure and control structure of the robot and further agile objects, and
- additional data

can be modeled and analyzed. The main analysis capabilities are based on simulation.

RobCAD enables the creation and/or import of 3D CAD models and the combination of them with kinematic models and process models (defining the control structure of the robot) to execute a manufacturing cell simulation.

Thus, the modeling capabilities of RobCAD are based on modeling objects for geometry and kinematics information which in parallel enable the modeling of process control data, functional data, and topological data related to them. Exploiting these information objects within the simulation run of the robot cell the controlled movements and behavior of the robots in the cell is visualized. Thereby, it is possible to simulate the reachability of the agile objects (including robots), to avoid collisions with obstacles in the cell, and to generate information about the timing of behaviors precisely.

Beside simulation and analysis, RobCAD enables offline programming of industrial robot systems from various vendors to reduce or in some cases avoid the complexity of implementing new manufacturing programs in the real robot cell. Therefore, RobCAD enables the use of a set of interfaces to automatically generate vendor specific robot control programs for different types of robots.

RobCAD enables the description and sequencing of all processes and tasks, which are implemented with manufacturing resources (as for example robots, machines, manual activities). This ability makes visualization and optimization possible of the entire manufacturing cell.

Thereby RobCAD can be applied for

- Planning and simulation of new manufacturing cell projects
- Simulation and optimization of existing manufacturing robot projects
The tool can provide 3D views of robot system models and works in an object and function oriented manner. The basic elements of the modeling capabilities are CAD data, kinematic objects like links and joints, and controllers, as well as the programming capabilities based on robot languages.

All these objects are not contained in a defined library. RobCAD rather offers a selection of tools to create a project based library manually.

**Tool components**

RobCAD contains an adaptable set of integrated tool components each developed for a special modeling and simulation purpose. Thus there is a one to one mapping of the information structure for a RobCAD model and the tool component structure itself. The set of tool components can be adapted / extended depending on the available licenses and application case requirements.

**Geometry models**

For enabling 2D and 3D modeling RobCAD implements a number of CAD tools to realize the full range of creating CAD elements and integrate them in the project library. The 2D Sketcher allows the creation of 2D structures. These structures can be used for further 3D operations. The Surface tool can be used to create surfaces for previously defined 2D structures. The 3D Sketcher is able to create simple 3D objects like boxes, pyramids and spheres. These objects can be combined using Boolean operations to create more complex 3D objects. With the placement editor the previously created objects can be arranged in the right direction. These tools are depicted in Figure 47.

![Figure 47: Geometry modeling in RobCAD](image)

**Kinematics and behavior modeling**

RobCAD also implements a number of tool components to create, simulate and optimize the kinematic and functional behaviors of the robot.

With the kinematics tab it is possible to launch different operations to create a kinematic structure, to combine 3D elements to links, compose them with joints along defined axis, add a controller and define a robot based on them. It is depicted in Figure 48.
By adding the created robot to a cell it is possible to test and program the robot or respectively the robot system. Therefore, the motion and the path editor components can be used (see Figure 49).

The motion editor offers the possibility to drive the robot, single joints of the robot, or the tool center point (TCP) to special poses (positions), to mark these poses, and to save them within the project library. The TCP can be driven to every direction and coordinate as possible based on the modeled kinematic system. The complete kinematic structure will automatically calculate the necessary movements of the single joints/axis.

The saved positions of the TCP are available as locations, which can be used to combine them to paths with the path editor. Here also they can be transformed to programs. These can be executed in the motion editor, to execute a collision-, joint-limit- or reachability- test.
Locations can also be created by pick or value in the path editor directly. This enables the creation of exact positions. These positions can be mirrored, copied and deleted. Additionally paths can be reversed and interpolated. Thus the motion and path editors enable the modeling of complex motion paths of robots and its axis and to execute them.

**Modelling objects**

The main modeling objects of RobCAD are the objects named in the class diagram in Figure 50. It also depicts the internal topological structure in the workcell of RobCAD and the main objects robots and components.

The main modeling objects of RobCAD are

- Work cell
- CAD element
- Frame
- Link
- Joint
- Axis
- Controller
- Component
- Robot.
The Work cell is the starting point of each modeling activity. It is one representative of a mechatronical system in the manufacturing system, modeled in RobCAD.

A work cell is based on a modeling floor with a defined floor size and a floor grid with a base frame representing the centre of the cell. This is visualized in Figure 51. It is not allowed to execute more than one workcell in one session.

Within a work cell CAD elements and links as geometry objects and all further objects up to components and robots can be integrated successively.

CAD elements are the basic element for visualization of 3D geometries. The constituent physical appearance of a robot and can be defined in a hierarchical way by Boolean combination of different partially basic 3D geometries.
A frame is a cartesian coordinate system which can be defined within the work cell on an arbitrary position.

A link consists of at least one CAD element and a frame. It established the basic element for creating components. Created links can be saved in the project library by defining them as components. Links are created in the modeling menu in RobCAD with the kinematics tab.

In addition links can contain axis and joints to be used for kinematical connection of links as well as controllers to model the motion behavior of links.

A component is the combination of one or more composed links. It enables the multiple use of the same links within one work cell. It is possible to compose components by component aggregation.

A special type of component are robots as well as obstacles in the cell, work pieces or pallets.

A robot is a component setted of links and a kinetic structure between the links. The kinetic structure consists of the joints between the links, two links are chained with exactly one joint, the axis of the joints, the frames and the controller.

The robot can consist of a discretionary amount of joints, but it has to be at least one joint. For example Figure 52 depicts a 6-axis-buckling arm robot.

Figure 52: Robot examples

This figure also shows the possibility of components to be used to model special tools attached to the robot for manufacturing process execution. RobCAD enables to model the different types of tools like grippers, welding guns, etc..

As seen in the picture the robot has two frames: the base frame on the base and the tool centre point frame (TCPF) on the mounting point at the last end. To make the robot usable it is needed to attach a sub component, for example a tool, on the robot, specially on the TCP.

The controller of the robot can be selected from a robot library implemented in RobCAD. For different cases different controllers are available, for example a controller for a nachi paint tool, as shown in Figure 53.
For simple executions it is also possible to use the default controller, it can control standard kinematics based on the well known Denavit-Hardenberg-Transformation of coordinate systems. The standard kinematics controller is not able to execute special tasks. For special purposes the control capabilities of RobCAD can be extended by adding additional user developed controllers. These controllers will implement the required kinematic calculations of coordinate systems.

3.10.2. Mechatronical concepts

The representation of the complete data set relevant for the mechatronical concept as defined in Deliverable 4.2 is restricted within RobCAD. This is based on the specialized focus and information modeling range reflected within RobCAD. RobCAD enables the modeling of the information sets:

- Hierarchy of robot workcells
- High level 3D CAD data of the robot workcell in the manufacturing system,
- Topological data such as the internal hierarchy of the ENGINEERING ARTIFACT, subordinate devices and interfaces to external devices
- Kinematic data of the robot systems
- Signal information and SPS function blocks
- Generic data as colors, weight, surface area...

cmpared to the information sets given in Deliverable 4.2.

In general all these information are related to the manufacturing execution control (MES) layer of the manufacturing system. All other information of the plant component are not covered by RobCAD as indicated in Figure 54. This figure depicts the focus of the capabilities of the RobCAD to express ENGINEERING ARTIFACTs with respect to its specific focus of use.
Nevertheless, RobCAD enables the application of mechatronic concepts within the engineering of manufacturing systems in a focused way.

RobCAD is focused on only two engineering activities within the engineering process of manufacturing systems, the simulation of robot structures and controls and the offline programming of existing robot cells. This is reflected by the portion of the information set of the plant component expressible within RobCAD.

A basic mechatronic information object can be established in RobCAD by the three different object component, robot and work cell. All of the combine to geometry and kinematics information as well as control information. In addition they constitute a hierarchical structuring as depicted by the aggregation relation in the class diagram given in Figure 50.

### 3.11. TIA Portal

#### 3.11.1. Tool description

The Totally Integrated Automation Portal (TIA Portal) is an engineering software system intended to be used for the engineering of complete control system projects including the engineering of the hardware system, the control application implementation, the communication system configuration, the control device configuration, and the HMI system implementation. Thus, it covers all engineering disciplines necessary to engineer and commission a complete control system for a production system.
The first release of TIA Portal brings together STEP 7 tools for programming and configuring SIMATIC controllers. Integrated into this environment is WinCC, the configuration tool for setting up Siemens’ extensive family of operator panels. Finally, drives can be set-up and parameterized in the same framework with StartDrive, a configuration tool for SINAMIC AC drives.

The main aim of the TIA Portal is the concentration on a set of possible engineering workflows enabling the user to engineering control systems step by step. Therefore, the TIA Portal provides different views and workflows.

Basic background of the TIA Portal is a common data base for all project related information of all engineering disciplines. This data base will be incrementally completed starting with the hardware configuration followed by control programming and HMI programming (see Figure 55).

In addition the TIA portal will provide a common look and feel over the different engineering disciplines and enable the free of breakage transfer of information between the disciplines exploiting the common data base.

The common data base will enable the use of common data tags over the different engineering disciplines. Thus, for example, creating a new variable in the control application and attaching it to a PLC input will enable to use this data tag within HMIs and within communication structures.
In addition TIA portal will support the development of distributed control applications by integrating different control hardware entities within one project an attaching control applications to them.

### 3.11.2. Mechatronical concepts

The TIA Portal is intended for control system engineering. Thus it covers all information which are relevant for control system. This will include:

- Control related data like control applications, program organization units, variables, device configurations, etc.
- Electrical data like wiring lists, communication device configurations, and energy supply,
- Function describing data for controlled behavior,
- Topological data related to the device hierarchy of the control system, and
- Generic data like order numbers, interface descriptions, etc.

The portion of the overall information set of mechatronical units/systems covered by TIA Portal is depicted in Figure 56.

![Figure 56: Information sets covered by TIA Portal](image-url)

The named information sets are integrated in device objects of a device and group of devices hierarchy which is used for project navigation. Hence, the device object and the group of devices object have been chosen as engineering artifact representation.
Within each device information and object sets are integrated as named in Figure 57.
4. Engineering Tool chain based on general engineering workflow

Within this chapter the engineering tool chain based on the general engineering workflow presented in Deliverable 4.2 will be shown. This tool chain will contain the tools presented within this deliverable. For sure this is only one exemplary tool chain, as in real application cases a multitude of tools is used within the engineering. Never the less it will be possible to show the basic context.

Figure 58 depicts the tool chain derived from the tool analysis. In the upper part the Activity layer shows the engineering activities proceeded within the general engineering workflow. The middle part shows the artifacts created within these activities. From left to right a qualitative time axis is given. Note that the length of each bar is not one to one translatable into a "consumed time". It is only a hint that the activities or artifacts are carried out / used during longer time periods.

The lower part of the figure shows the tools used to create artifacts. Here the height of the bars is an indication for the impact a tool has on the engineering results. E.g. the impact of AutomationML as a data format is quite low as data is only stored, but not generated within. Never the less there is a certain importance regarding the seamless information exchange.

Comparing the different tools to the tool scenarios given in chapter 2.4, most of the tools can be categorized as best of bread tools. Only Comos / SIMATIC Automation Designer and AutomationML are different. Comos is a typical all in one tool which can be easily seen as it is used without restrictions throughout the engineering process. AutomationML is also used in parallel, but as it is an data exchange format, it is not an all in one tool, but the basis for an integrated tool chain. Thus, assuming that all tools provide an interface to AutomationML (which most of them don't do in reality up to now) the beast of bread tools may be used within an integrated tool chain scenario.

Looking at the best of bread tools some of them are dashed in the early or later phases. This means that these tools may be used for creating these artifacts but it may not be their focus of work. Additionally it becomes obvious that the tools are kind of accumulating in the instantiation, implementation and first testing activities. This is owed to the fact that within these activities the whole bandwidth of engineering disciplines is used (mechanics, electrics, pneumatics, hydraulics, control, automation, ...). Thus different specialized tools may be used for different disciplines. Due to this reason also the Comos / SIMATIC Automation Designer is only highlighted with medium importance during the whole engineering process. Although it is used for different disciplines there are always specialized disciplines not embeded into the tool. Thus, it can be used for the "standard engineering activities" and will thus cover 90% of the engineering. But a 100% coverage will never be possible due to the high diversity within engineering business. Other tools like the TIA portal are again focused on one discipline (e.g. automation engineering) and provide all necessary sub-tools within this discipline. Thus, they have a big importance for these engineering artifacts.
Deliverable D4.3
Integration of GRACE multi-agent systems with manufacturing CAE systems

One thing that has to be mentioned is, that none of the industrial CAE tools provide capabilities for MAS engineering. Thus, tools like JADE, NetBeans IDE and Protégé have to be used here.

Figure 58: Engineering tool chain based on general engineering workflow

Note: Due to the size of the picture, Figure 58 will be available as separate .png file attached to this file.
5. Additional tool requirements for industrial engineering tools supporting GRACE MAS-architecture

For the evaluation of additional tool requirements for industrial CAE tools supporting a modular Grace based MAS-architecture, analysis were made for different industrial CAE tools. Additionally some Case studies were carried out. A short wrap-up of these studies will be done within section 5.1. Additional information is given in (AMENIA 2012).

Within the second section of this chapter the results of the case studies and tool evaluations will be used to identify further tool requirements.

5.1. Case studies

Within four case studies from different industries the usability and benefits of modular thinking have been investigated exploiting the analysis concept given in chapter 2.

The first case was related to a Natural Hydraulic Lime (NHL) production system EP executed by Fels-Werke GmbH Germany. Within the engineering organization analysis it was considered how the use of newly available TIA portal of Siemens and modular thinking will influence the engineering process. It could be shown that a set of engineering artifacts related to mechatronic components have a high potentials for reuse, the use of a common mechatronic structure will improve information exchange by providing a common semantics of information, and that a component library can improve reuse of engineering artifacts within control and HMI programming within TIA but also supports the component selection in the early stages of the engineering process.

The second case study was related to the engineering of railway control systems. The main question of the investigation was which impact mechatronic thinking and a mechatronic component library in a discipline spanning tool like the SIMATIC Automation Designer based on COMOS will have on the engineering process. It could be shown that a common engineering tool for all disciplines with a mechatronic engineering data model will ensure a lossless information exchange along the engineering process crossing the engineering disciplines. It will ensure engineering discipline integration based on common component information semantics and makes reuse of engineering artifacts much easier: Additionally, it will reduce engineering process risks by ensuring a much better version management of the project states.

The third case study was related to the engineering of a robot cell with a focus on virtual commissioning based on RobCAD. Within the procedure model for the engineering of a robot cell it could be shown that the main risk for information loss will be between special engineering activities were both tools and artifacts will change. The only glue at this position will be the cell planner. Thus this role has to have extensive skills covering all involved engineering disciplines. In addition it gets visible, that the considered tool will cover most of the engineering activities except electrical engineering and early phases. Thus it will automatically ensure a wide range of the process data consistency, lossless information exchange and the integration of engineering disciplines.
The last case study was related to the process engineering within machine tool industry. The process engineering in this case follows a strict sequence which cannot be improved by mechatronic thinking. But there are major improvement capabilities by organizing the reuse of engineering artifacts for material and immaterial ones. To ensure a proper reuse of artifacts it is necessary to know which artifact is used and created in which engineering activity. The procedure model is used to identify potentially reused artifacts like CNC programs, fixture CAD, and physical fixtures, as well as CAD of products. These artifacts are interrelated. Exploiting the Product-Process-Resource concept (Cutting-Decelle et al., 2007) and mechatronic concept the artifacts can be organized in an artifact library. In addition lossless information exchange can be ensured between different production orders and the engineering processes related to them.

5.2. Additional tool requirements

Based on the different tool scenarios, case studies and tool evaluations, several requirements to industrial CAE tools have been identified in order to be able to cope with engineering challenges involved in the engineering of modular production systems and especially for GRACE MAS based systems. These requirements should be regarded as a set of additional requirements which have not yet been fully implemented within industrial CAE tools. Never the less, tools are very different at their maturity level regarding modular design paradigms. This does not automatically mean that these tools are "bad" tools, as this is highly depending on the focus and application case of such tools.

Generally the integration and handling of data has to be improved. This is mainly focusing on data exchange aspects within an engineering process. As show in the three tool chain scenarios, tool landscapes might be characterized by scattered "tool islands" (best of bread). These islands nowadays are mostly independent or only loosely connected to each other. But following a modular design approach like MAS, engineering artifacts cannot be regarded independently. In fact several artifacts constitute different views or perspectives on the same engineering object. Thus changes in one perspective have to be shared with the others. This assurance of data actuality and consistency can only be obtained by enforcing the data integration. On step might be to drive development of different best of bread tools in a direction of an integrated tool chain. This can be achieved by providing a common data backbone like AutomationML. Another way might be the development and/or improvement of new/existing all-in-one tools.

As information about objects and data correlation between different perspectives might get very complex a clear data structuring is needed. This will also help avoiding information redundancies, thus improving engineering efficiency in parallel.

Another point regarding data integration becomes obvious looking at the field of collaborative engineering and supplier collaboration but also collaboration with stakeholders of later life-cycle phases (e.g. plant operator, maintenance, ...). Here not only the integration itself is important but also the possibility to have consistent simultaneous data access including multi-site engineering and specified user rights. Also versioning, variants handling and change management are typical requirements which are not new, but have to be better realized and integrated into manufacturing CAE systems.
Besides integration of data perspectives it is also important to improve the integration of different disciplines and new models within engineering tools. Scouting the current available industrial CAE tools, no tools for MAS engineering could be identified. This is also due to the fact that multi-agent systems are currently not typical industrial applied control paradigms. Thus, there is no real market need for this tool support. But this also leads to some chicken-egg-problems. As long as no industrial tools are available, industrial application is aggravated, as OEMs mostly rely on kind of industrial standard tools (as they also guarantee tool support for the next years). Thus they are not using MAS as free tools like JADE might be able to engineer those systems, but there is no guarantee that if the plant has to be changed in five years, this tool will be available, or even worse, compatibility with new control hardware etc. is given. This way they are obstructing the development of a MAS tool market.

But also other disciplines have to be better integrated e.g. Quality management (definition and check of Quality gates, milestones, etc.), ERP systems, office applications and so far. In case of GRACE engineering methodology also tools improving the handling of the MPFQ architecture might be helpful. This requirement, which at the first view seems to be very specific can be abstracted to another level. Within each company different engineers develop tons of models and methods to cope with certain problems. Generally there should be a way to bring these models, processes and methodologies into tools, thus not only enhancing problem solving based on this new model/process/method, but also connecting them to the "rest of the world". One example for this is the Java platform which was used within GRACE to ensure interoperability of JADE, NetBeans IDE and Protégé (each tool solving different problem aspects).

Another requirement, which has already been touched, is the standardization and reuse of modules. A project spanning standardization of modules will help to enable an also project spanning reuse of artifacts. This can be done e.g. by standardizing tool interfaces. A first approach for this was given in Deliverable 4.2. Together with a library system support this will ensure consistency of standard modules and also improve quality of the whole engineering process as tested and proven modules can be used. This aspect was also shown in the workflow bisection of the general and OEM specific workflows given in Deliverable 4.2 and D4.2 – Appendix A.

At the end there are also several other requirements which should be regarded. One problem, also addressed within GRACE, is the contextualization of information. Within GRACE this was done using ontologies for the MAS. Within the engineering these ontologies might also be used to contextualize information meaning. A second problem is the user guidance and user interface of engineering tools (e.g. automatic check of results, provision of guidelines / OEM specific workflows; localization, use of different scale-systems, integration of documentation handling, etc.).

Most of these requirements are somehow connected to each other, thus regarding them separately is not possible. This is also a big hurdle when it comes to their realization within engineering tools. Implementing one after another is not simple or even possible. Thus implementing them as a whole would mean high efforts for tool providers.
References

AMENIA Project Team (2012): Analysis and Evaluation in Fields of Applied Mechatronical Engineering for Industrial Plants AMENIA III – Siemens internal document


IEC 62424: Specification for Representation of process control engineering requeats in P&I Diagrams and for data exchange between P&ID tools and PCE-CAE.


VDI 2206, 06/2004: Design methodology for mechatronic systems.

inteGration of pRocess and quAlity Control using multi-agEnt technology

Project funded by the European Commission under the “Seventh Framework Programme” (2007-2013)
Contract n° NMP2-SL-2010-246203
Appendix A – Engineering tool evaluation
Appendix B – Modeling examples for engineering tools
Appendix C – Engineering tool chain based on OEM specific engineering workflow
Appendix D – Bearing insertion station – Whirlpool documentation
Appendix E – Bearing insertion station – Comos documentation