AVANTI

ITEA 2 – 12036

Test methodology for virtual commissioning based on behaviour simulation of production systems

Work Package 1

Analysis, requirements, scenarios

Task 1.1, Task 1.2

Requirements definition and definition of typical scenarios

Deliverable D1.1

State of the Art and Use Cases

Document type	: Deliverable
Document version	: Final
Document Preparation Date	: 30.04.2014
Classification	: Public
Contract Start Date	: 01.11.2013
Contract End Date	: 30.06.2016

ITEA 2

INFORMATION TECHNOLOGY FOR EUROPEAN ADVANCEMENT

EUREKA

Final approval	Name	Partner
Review Task Level	Riedl	ifak
Review WP Level	Riedl	ifak
Review Board Level	Bär	Daimler

Executive Summary

The described state of the art gives an overview of technologies and aspects that enable service oriented architectures, with the main focus on networked embedded systems. To establish an entire service oriented ecosystem, a broader range of technologies is necessary. Chapter 1 introduces into the approach of Virtual Commissioning. The following chapters summarize the relevant technologies and the engineering approaches and point out the relevance to AVANTI. Each chapter shows the characteristics of an individual technology and shows relations to research topics of AVANTI.

Chapter 2 is dedicated to mechanical models and identifies the required information for Virtual Commissioning from this perspective.

Chapter 3 is focused on existing description technologies. They will be considered as basis for the description of component and machine behavior.

The state-of-the-art of simulation tools are summarized in the fourth chapter, whereas the fifth chapter goes in detail for aspects of automatic test generation.

The sixed chapter describes the complete engineering process for the integration of the specific specialized tool and approaches describes before.

Deliverable D1.1 is a collection of contributions from the experts involved in the project. It shows therefore also a snapshot of the state of the art at the beginning of the project. Mutual understanding has increased writing this deliverable, which provides a starting point for the detailed work in the other work packages.

The other work packages will partly precise and extend the technical content of the specific topics, so that the complete knowledge of the technologies can only be acquired reading D1.1 and the other deliverables.

Contents

1 Introduction

The project AVANTI would like to contribute to enhance the Virtual Commissioning (VC) by means of physical based simulation, improved behavior models and a mostly automatically derivation of test cases in order to enhance and accelerate the engineering of production systems.

The goal of Virtual Commissioning for factory automation is to develop a realistic virtual model of an automation system before the real setup is realized and then run virtual simulations on the model to test the functionality of the system via “what-if” scenarios. The major benefits are the shortening of the time spent from planning to real setup and early detection and debugging of errors in the system, both leading to significant reduction in costs. For this reason, Virtual Commissioning of automation systems has gained importance in industry and academia during the last decade [1].

In the following sections, the research and development efforts on Virtual Commissioning are discussed under two sub-sections:

a) Commercial Solutions: the commercial software packages available in the market,
b) Academic Solutions: a brief summary of Virtual Commissioning systems developed under EU-funding and some other academic work.

The results of the examined research and development efforts are organized in sections very close related to the working package structure of AVANTI.

2 Mechatronic Model

2.1 Introduction

The Virtual Commissioning is a method with is an early testing and optimization of PLC programs (Program Logic Control) possible based on a so-called mechatronic model. This model represents the 3d-geometry and the behavior of all components into the plant. Figure 1 shows a schematically structure of the mechatronic model and the communication interface between the PLC and robot control with him.

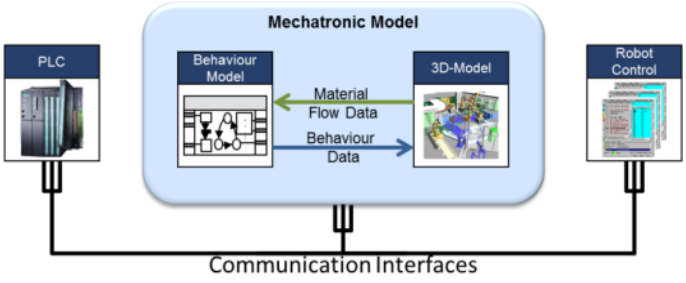

Figure 1 : Mechatronic Model

Furthermore, the mechatronic plant model can divided into the behavior model and the extended 3d geometry model. These tow sub-model communicate internal with each other, as example transfer of position values from the behavior model to the 3d geometry model, and external with a PLC, as example transfer signal from the PLC to the behavior model. In

addition, the mechatronic model can communicate with a robot controller, when a robot existing into the plant.

2.2 Extended 3d geometry model

The extended 3d geometry model, also called 3d model, is used for visualization of movements into the plant during the Virtual Commissioning. With the position values from the behavior model, as example, will be animated the movements of the plant component during the simulation. This means, the signals from the behavior models controlled the kinematic of all components into the plant. For this propose, the signals from the behavior model must be assigned to the components into the 3d model. The actual 3d model and kinematic definition will be taken from the design model of the plant. In addition to this information, the extended 3d model must have the material flow information, which defines the transfer of parts into the plant. Thereby, the position information can be given back to the behavior model the sensors need to be integrated additionally into the 3d model.

2.3 Behavior Model

The behavior model represents the logical and temporal behavior of all components into a plant. Each behavior model requires information from outside. This information is provided about interfaces to the behavior model. On basis of PLC outputs and inputs signals will be calculated from the behavior model the kinematic values for each component in the plant. The behavior model gives for each simulation step the current positions or velocities of a component, so that it can to perform a movement.

To carry out the Virtual Commissioning, the first tests took place directly with the cad tool. But these tools have strength in the area of construction and are suitable for the VC only to a limited because the achieved on account of the complexity very limits quickly. Also, incorporating your communication for the required interfaces is very difficult. The result of some attempts shows that a “one-toll” System is not suitable for complex plant simulation. That’s why the solution that the cad- and behavior-data must be transferred into other tools which can be connected to some communication interfaces. The best practice method for the optimal performance is to split the mechatronic model and the robot control in more than one tool. To find out something about the quality and consistency of control programs in the VC the tools must be connectable to a real PLC, what means that the tools must have a communication interface. In addition have to the different formats of cad tools, which are described in detail in one of the next section, from the tools that used for VC are support.

This chapter, therefore, has a closer look at each of the original disciplines which mechatronics stem from. It is subdivided into IT, logical attributes and behavior, in order to cover computational models, electronic attributes and behavior, to cover the field of electrical engineering and mechanic attributes and behavior, to discuss the state of the art in models for mechanical engineering. In addition, special models such as fluidic models and attributes that do not fit the classification discussed here are presented.

2.4 IT, logical attributes and behavior

The relevant qualities that need to be modelled in the IT and logical behaviour are typically communication and signal information. Another relevant input is program code (typically for a PLC, see below), as well as the specific behaviour of discipline specific interfaces. The latter might be sensors or proprietary IT systems.

IT attributes in Virtual Commissioning today focus on the modelling and simulation of control programs. The consistency and correctness of control programs is checked by simulating

their run time behaviour. To this end, the PLC (programmable logic controller), that is to run the control program, is emulated. A related topic is the simulation of the behaviour of a field bus system.

The IT structure of current VC solutions is based on communication interfaces to allow data exchange between concurrently running simulation programs. Examples for these data exchange interfaces include the RSServer by Rücker or the Y200 Server by mewes und Partner. These servers marshal signal exchange between geometrical 3D models. Furthermore, information about the flow of materials and the movements of all devices is processed.

To connect to the PLC, interfaces like A780 of WinMod (Mewes und Partner) can be utilised. Simulation of the participants in field bus communication need to be emulated as well, for a setup using a Siemens PLC a Simba Box can be utilised to this end, e.g.

In general, it can be said that system parameters and architecture must be representable. Also, signal interface for sensors, components and bus systems must be provided between software environments which are used. The relevant debugging functions must be provided. And finally, class diagrams and state charts must be available.

Links:

- http://www.robsim.de/programme.html
- http://winmod.de/de/index.php?page=feldbusemulation
- http://www.industry.siemens.com/services/global/en/it4industry/products/simulation/simba_profibus/pages/default_tab.aspx

2.5 Electronic Attributes and Behaviour

Relevant information regarding electrical engineering consists of information such as the current, voltage and power of an electrical flow, electronic circuits in the form of E-CAD (electrical computer aided design) as well as discipline specific data about signal tables, bus systems and the like.

In Virtual Commissioning, no information on electrical engineering data is processed as the state of the art. However, it is possible to evaluate these attributes by reverse engineering factual cycle times, e.g.

With the help of E-CAD data (i.e. signal tables), it is possible to define the behaviour model of an arrangement. This can be transferred into macros for simulation tools like WinMod.

Links:

- http://winmod.de/de/index.php?page=peripherisimulation
- http://winmod.de/de/index.php?page=anlagensimulation

2.6 Fluidic Attributes and Behaviour

2.6.1 Overview

The consideration of fluidic system is of relevance beyond the classical mechatronic approach, because of the relevance of fluidic systems in automation. Relevant properties

that should be modelled for fluidic systems are fluidic circuit diagrams with attributes such as overall fluidic consumption, function and debugging and testing information, simulation models for single components and circuits containing power, flow, speed, acceleration and shearing forces and interface behaviour with mechanical and electrical properties, such as valve switching times and control flow.

Currently, the only aspects of this information that are considered for Virtual Commissioning by the original equipment manufacturers are the movement of the actors and the feedback from the sensors. However, as the state of the art, information about the movements and the freedom of movement in the CAD (computer-aided design) Tools is declared separately. A standardised library would be helpful, which for example models a gripper with valve, degrees of freedom of movement as well as pre-defined stop positions.

2.6.2 Fluidic models for circuits

Models for circuits have a high diffusion rate. These models aim to set up, test and debug fluidic circuits. To build up a model of a fluidic system the models of the single components are connected among each other. The component models are stored in libraries.

This kind of model does not aim to provide a deep physical simulation to allow a fast calculation. However these models can have a quite high resolution that consider many fluidic aspects (e.g.: flow resistance, flow rates, forces, pressures and fluidic leakage) which is sufficient for a large number of applications. An example circuit diagram is given in Figure 2. For physically more precise simulations, reference is made to fluidic component models.

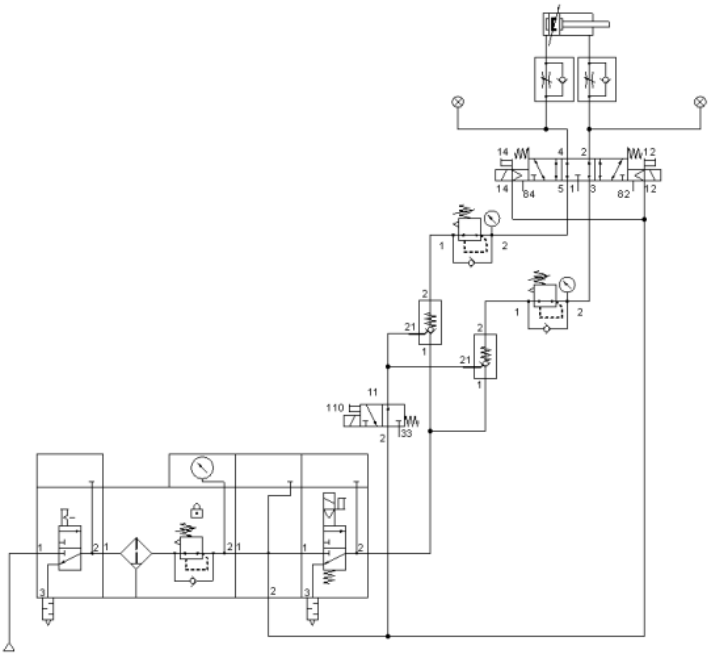

Figure 2: Example of a pneumatic circuit

Behind these models is a mathematical abstraction with varying depth. The mathematical model describes the component behavior with differential, algebraic and discrete equations. Object-oriented description languages (e.g. Modelica) allow describing and connecting of different mathematical models.

A tool to generate and simulate models of fluidic circuits is FluidSIM by Festo Didactic. It provides a graphical user interface to set up, test and debug fluidic circuits in an interactive way, as shown in Figure 3. Because of its simple handling it is often used for didactic purposes. Also, this tool allows its user to connect the created models to PLCs via OPC-Interface or DDE. It's possible to simulate pneumatic and electric circuits at the same time and with their connections among each other.

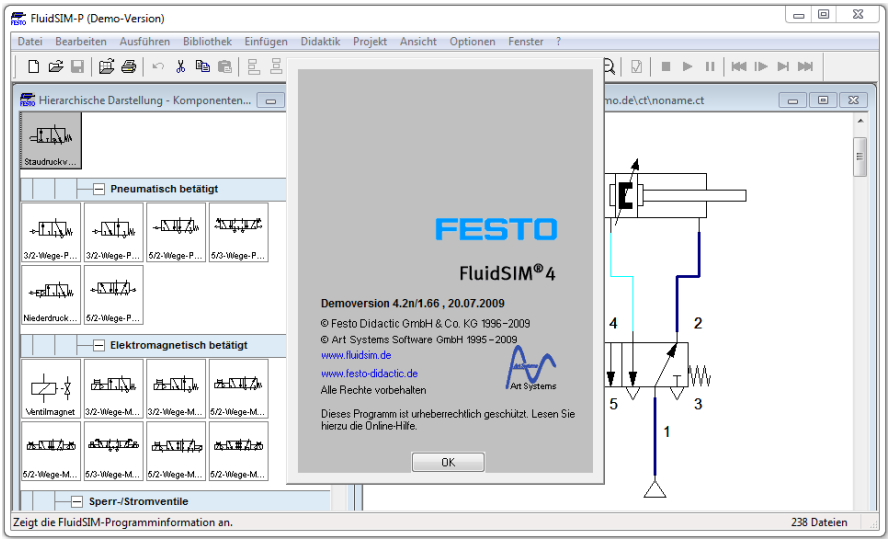

Figure 3: FluidSIM 4.2

Links

- http://www.festo-didactic.com/de-de/lernsysteme/software-e-learning/fluidsim/fluidsim-4-pneumatik-einzel-und-mehrfachlizenzen.htm?fbid=ZGUuZGUuNTQ0LjEzLjE4LjU5MS40NzQw [German]:

2.6.3 Fluidic component models

Fluidic component models have a focus different from fluidic models for circuits insofar as they focus on the single components inside a fluidic circuit. It simulates single components with appropriate accessories (e.g. pneumatic cylinder and its check valve with choke). They are utilized in order to select a component suitable for a specific task. Target qualities for this selection might include the determination of the feasibility of a solution, the calculation of a solution's characteristics, optimization of the cycle times of a circuit, minimization of the energy consumption of a circuit and the reduction of the kinetic energy in the stop positions (for a cylinder, e.g.), thereby increasing safety and longevity. Also, the models can help to find the best settings for an already chosen component (e.g. for damping).

The calculation of the fluidic component model brings as results the dynamics of the component, as for example its position at a given time, speed and acceleration. Furthermore, the pressure and temperature of the fluid can be calculated. The fluid's flow rate through the component is calculable, as is the electric current that is used to control valves. Of importance for the feasibility of a solution are the forces and moments in damping and guiding.

2.6.4 Model Interfaces

Fluidic models have several interfaces to the domains of other models considered in this chapter. However, few of these interfaces are currently addressed in a way that would allow

a seamless integration of model information. Interfaces that need to be addressed are for example the interface to the electronic circuit model. This interface can provide information on the activation of valve controls. Kinematic models interface in the calculation of forces such as shearing forces. Physical models can provide information on the forces that come to bear at a certain point of time. And 3D CAD models can be used to provide information on the flow rates.

The information exchange between these models still has to be automated as of the state of the art. The relevant models are illustrated in Figure 4.

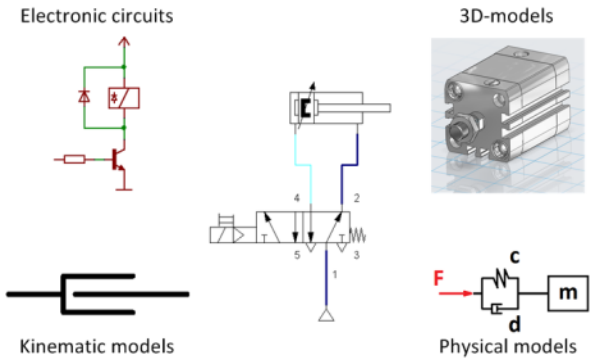

Figure 4: Model interfaces in fluidic components

2.7 Mechanic Attributes and Behaviour

Mechanical attributes that are considered for Virtual Commissioning include the type of material used in an arrangement, the weight of components and the tolerances involved. Of high relevance are the kinematics, as represented in 3D models that define joints and moving parts, sometimes only available in diagrams. The physical behaviour as a combination of forces and effects is another point of consideration.

A lot of information can be gotten from the M-CAD (mechanical computer-aided design) data of the components involved in an arrangement. The M-CAD data is available in many different formats, not all of which are provided by every solution. Specific interfaces are another subject of heterogeneity, such as screwing, welding or gluing interfaces.

Links:

- http://www.cadenas.de/elektronischer-produktkatalog/intelligente-cad-modelle [German]

2.8 Further Specific Attributes

In addition to the different disciplines considered above, there are additional attributes that warrant consideration for a broad mechatronic approach to Virtual Commissioning. For example, the purchase of components for an arrangement that is subject to VC can be facilitated by specific information in the form of product classifications such as eCl@ss and STEP or catalogue exchange information such as BMEcat. The description technologies of chapter 4 are another example.

Furthermore, the documentation of components and arrangements produces textual information in the form of written documents. These documents, in addition to the information

described above, might contain instructions for use and installation, commissioning, safety and standard compliance which might of relevance. These documents can be accessed in various ways, sometimes stored on or with the component in question, sometimes accessible online.

Final consideration for the mechatronic model is the information that is carried over the life cycle of components and arrangements. Information from the dimensioning can be relevant for the VC of an arrangement. The configuration and programming of controllers and other intelligent devices has an impact on a system's behaviour. Information from VC should be utilised for the regular commissioning, information from operation, condition monitoring and maintenance can be mirrored back to dimensioning and commissioning of future arrangements. Recycling of arrangements might free up components for future usage. Therefore, the mechatronic model needs to be maintained and optimised over the different phases of an arrangement's life cycle.

Links:

- http://www.eclass.de/eclasscontent/index.html.en
- http://www.iso.org/iso/standards_development/technical_committees/list_of_iso_technical_committees/iso_technical_committee.htm?commid=54158

3 Description technologies

3.1 Electronic Device Description Language (EDDL)

3.1.1 Introduction

Years ago, field devices were shipped with so called electronic data sheet or manual, describes the field device in detail. This includes the data types, the data ranges, the default values, dependencies of the parameters, the online access, vendor specific functions etc. A further development of this data sheet is the Electronic Device Description, mainly used for integration of the device into the automation system and for maintenance tasks. EDD is based on the Electronic Device Description Language (EDDL), which is international standardized in IEC 61804 parts 3, 4 and 5. With the EDD it is possible to describe the device in a formal way including the parameters, communication aspects, functions and the user interface. Special software (EDD Application or EDD Host) that can read and interpret the EDD is then able to access every device that has an EDD. Latest improvements enhance the EDDL to describe also simulated behaviour of the device. By means of this, offline configuration and system behaviour may be more exact.

EDDL allows establishing libraries with predefined device behaviour and localisation of the user interface. In addition, standardized strings may be separated in so called dictionaries.

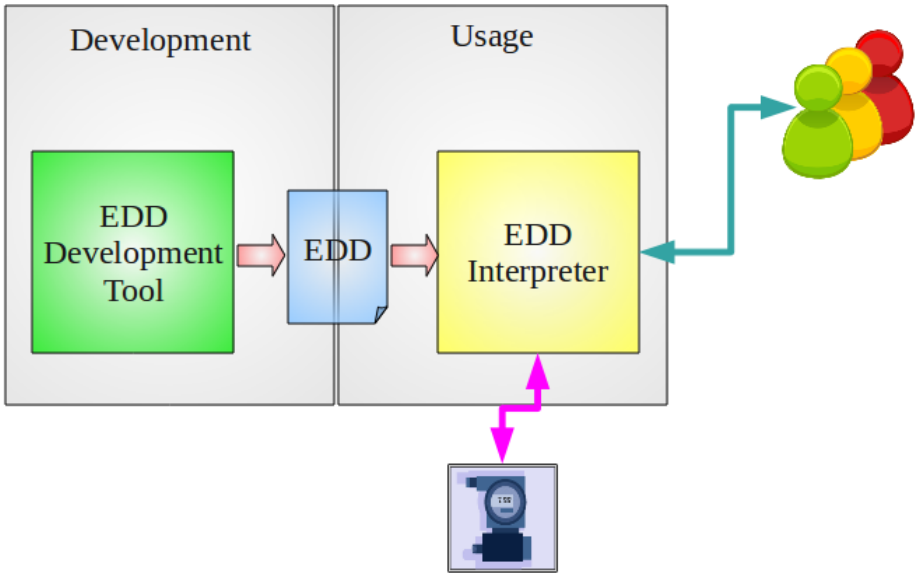

Figure 5: Integration of an EDD on source code level

Figure 5 demonstrates the usage of an EDD on source code basis. In order to protect the intellectual properties of the companies, the encoded file format (DD, see Figure 6) is also well defined and is improved in FDI project.

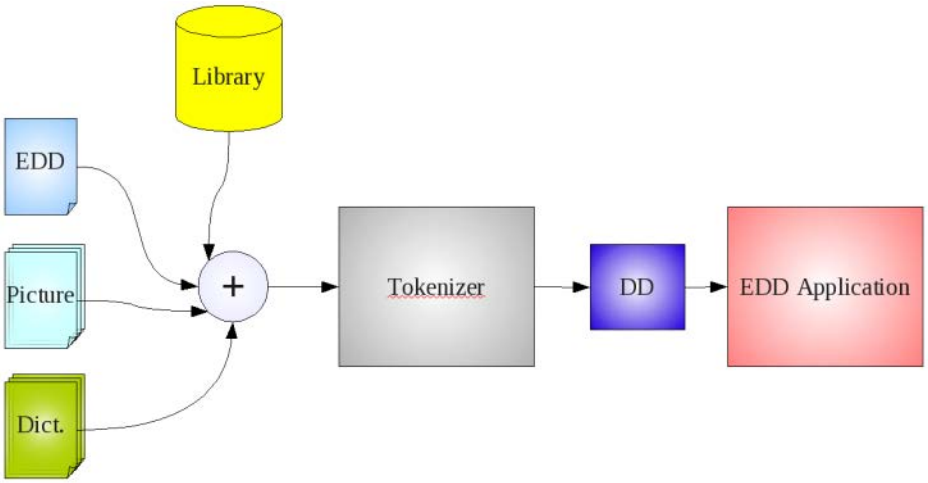

Figure 6: Integration of an EDD by means of an encoded file format

Existing tools are dedicated for EDD development and its usage in real commissioning for parameterization. In principle the EDD application can also be used for device simulation.

3.1.2 References

3.1.2.1 Relevant publications

- Riedl, M.; Naumann, F. †: EDDL – Electronic Device Description Language. 2011, Oldenbourg Verlag, ISBN 978-3-83563106-9
- Großmann, D; Braun, M.; Danzer, B; Riedl, M: FDI - Field Device Integration, Handbuch für die einheitliche Integrationstechnologie, VDE Verlag, Nov. 2013, ISBN 978-3-8007-3513-6

- http://www.eddl.org/
- http://www.fdi-cooperation.com/

3.1.2.2 Relevant products or links to company statements

- Siemens: SIMATIC PDM
- Emerson: AMS Suite
- HART Communication Foundation: DD Integrated Development Environment
- Fieldbus Foundation: DD Integrated Development Environment
- FDI Cooperation: FDI Package Integrated Development Environment
- ifak / ifak system GmbH: isEDD Workbench, http://www.ifak-system.com/kommunikation-automation/werkzeuge/produkte/isedd-workbench/

3.1.2.3 Latest version of relevant specifications, RFCs, standardization activities

- IEC 61804-3, Function blocks (FB) for process control and Electronic device description language (EDDL) - Part 3: Electronic Device Description Language (EDDL), Ed. 3, 2012
- IEC 61804-4, Function blocks (FB) for process control and Electronic device description language (EDDL) - Part 4: Implementation, Ed 1.0, CD will be published in 2014
- IEC 61804-5, Function blocks (FB) for process control and Electronic device description language (EDDL) - Part 5: EDDL Builtin library, Ed. 1, 2012

3.1.2.4 Relevant projects

- EP 26951 NOAH Network Oriented Application Harmonization

3.1.2.5 Network activities

- FDI: EDDL Maintenance Group

3.2 Basic system and modules for design and simulation tools (BAPSI)

BAPSI is an ifak-framework for the development of a basic software system for the simulation of dynamic systems and related modules, which can be used to develop rapidly and inexpensively customised software applications. Such applications, set up using this basic simulation system are planned to be used primarily in engineering practice. The applications will focus on design, simulation and optimisation of process engineering plants. Appropriate software tools can be developed, which can be used in the design process for small and medium scaled plants by end users and consulting companies.

BAPSI support also the block oriented approach, but for complex control sequences, other descriptions are more appropriate. E.g. sequence function charts are one example of a better suited approach. In the recently available system, a generic Petri-net approach is used. Petri-nets allow describing single sequences as well as parallel running coordinated control sequences. Figure 7, the graphical model of a sequencing batch reactor (SBR) controller is presented. The controller consists of places (states) and transitions (flow of marks/states). To combine the Petri-net with function blocks, transitions might have a condition input. To generate control outputs, a selection of places can be used to generate multiple output signals by an output matrix (block YMux) which defines the values of the output signals for each possible state. The transition blocks have optionally an internal timer to allow a simple time based propagation of states.

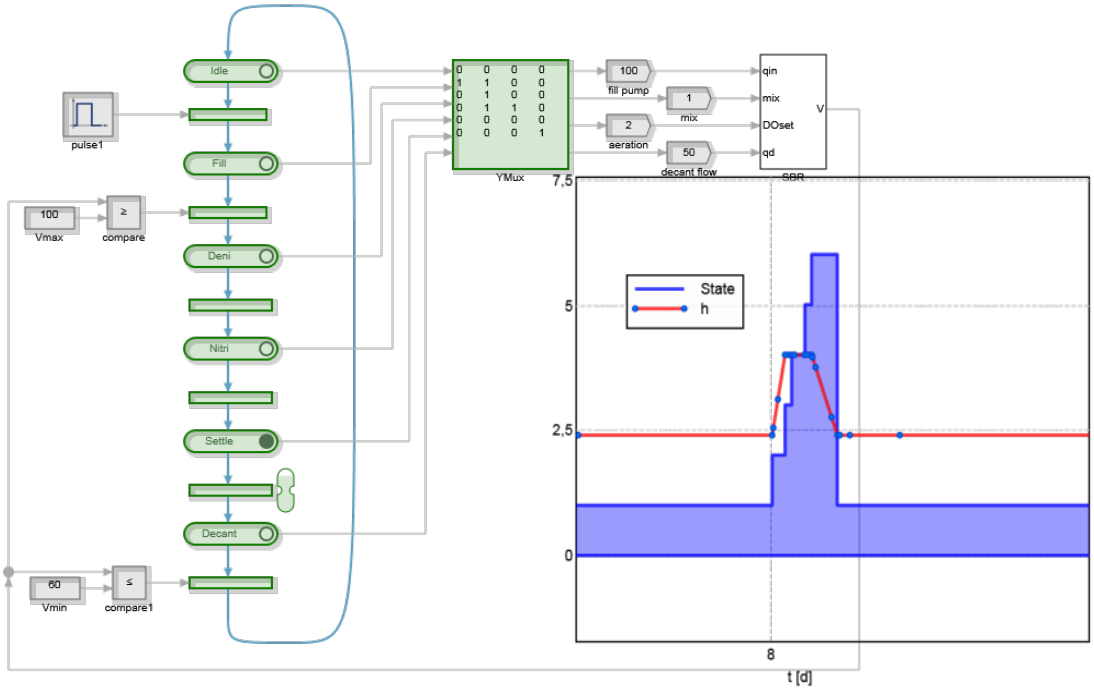

Figure 7: Petri-net to describe a SBR controller (left), state and SBR volume (right)

The described system behaviour will be saved in an XML file following proprietary schema definitions.

3.2.1 References

3.2.1.1 Relevant publications

- http://www.mathworks.com

3.2.1.2 Relevant products or links to company statements

- SIMBA classroom
- SIMBIA Professional

3.2.1.3 Latest version of relevant specifications, RFCs, standardization activities

- V. 1.0

3.2.1.4 Relevant projects

- KURAS, nidA200, NoNitriNox, … (BMBF)

3.2.1.5 Network activities

- Hochschulgruppe (HSG) "Simulation"

3.3 MATLAB / SIMULINK

3.3.1 Introduction

Simulink is a well popular simulation tool using the MATLAB® high-level language offered by Mathworks. MATLAB is both, the programming language and an interactive environment for numerical computation, visualization, and programming. The language uses ASCII characters and will be saved as an XML file, recently also as compressed file.

The contents of the MATLAB file represent the whole system modelled in the tool. A system will be modelled in blocks connected to each other to define the data flow. There are predefined blocks and it is possible to define own blocks. The propriety programming language and XML schema will be maintained by Mathworks.

3.3.2 References

3.3.2.1 List of conferences covering relevant topics

- MathWorks Automotive Virtual Conference
- MathWorks Automotive Conference
- MATLAB Virtual Conference

3.3.2.2 Relevant publications

- http://www.mathworks.com

3.3.2.3 Relevant products or links to company statements

- MATLAB
- Simulink

3.3.2.4 Latest version of relevant specifications, RFCs, standardization activities

- Release 2013b

3.3.2.5 Relevant projects

- -

3.3.2.6 Network activities

- Diverse user groups

3.4 CAEX

3.4.1 Introduction

CAEX (Computer Aided Engineering eXchange) is an XML-based data format for relational object information, in particular in the domain of plant automation. It is standardized in IEC 62424 with the motivation of overcoming the incompatibilities between individual vendor's description formats for process and control engineering tools. While CAEX is not bound to modelling entities of a particular nature, we assume that a plant model is considered. Such a model is commonly called a *topology*.

CAEX natively supports several object oriented concepts. The most prominent is aggregation (or composition) which allows for hierarchical models. Moreover, classes, instances, relations and interfaces are supported; a further prominent feature is the association of elements with roles (platform-independent semantic data). The language is extensible, i.e. other modelling concepts can be added to the language as seen fit.

The model of a particular plant, or a part thereof, is called an *instance* in CAEX. An instance may be hierarchically composed of the instances of its constituents. Such hierarchies are the intended domain of CAEX.

To simplify the reuse of sub-models across different instances, a library concept is supported. Three library-types are pre-defined by CAEX, they are

- System Units

- A system unit is a self-contained physical or logical module of the system. As such, it may contain again attributes, interfaces and internal elements, i.e. a system unit may be hierarchically structured. System unit libraries allow for vendor-specific catalogues with elements that are integrated into a full system description.
  - Interfaces
- An interface models the connection of objects in order to exchange physical units (workpieces, energy) or data (control, measurement).
  - Roles
- A role is an abstract functionality that may be provided by an element. Roles allow the modelling independent of implementation details, i.e. based on abstract functionality alone. On the other hand, roles allow adding semantic value to a part of the plant model. Also, the same element may realize different roles. For example, a conveyor belt that connects two manufacturing cells removes pieces from one cell, while it feeds the other, thus is implements different roles with respect to the different cells.
- While CAEX models relation among parts of a plant, it does not allow to model the inner workings of these elements. However, any part of an instance or library can be linked with external data, which may define further details of the modelled element. This also allows to model different aspects of an object by different languages and integrate them via CAEX.

3.5 COLLADA

3.5.1 Introduction

COLLADA (COLLaborative Design Activity) is an XML-scheme format for 3D asset exchange. The specification language was introduced as an exchange format for digital content in the digital games industry and was later adopted for visualization purposes of other domains, such as the automation domain. The design explicitly aims at a text-only language, i.e. COLLADA does not encourage the inclusion of binary data whatsoever. Version 1.5 of COLLADA has been standardized as ISO/PAS 17506:2012.

A COLLADA document consists of a *scene* which is assembled from instances of *library*-elements. Libraries describe the elements of a 3D model aspect-wise, by considering independent element properties (geometry, mass), dependent element properties (joints) and scenic properties (lighting, animation). Of the features (i.e. libraries) that are supported by COLLADA, we elaborate on the three we deem the most important for the Avanti-project.

- Geometry
  The spatial extensions of an object are described by a *boundary-representation* model. Such models represent objects by successively joining outlines of a lower dimension in a higher dimension. Somewhat simplified, points (0D) are joined to yield edges (1D), which are joined to yield lines and shapes (2D), which are joined to yield bodies (3D). More sophisticated bodies can be constructed using parameterized curves, i.e. splines.
  This boundary-representation approach allows modelling solids in an unambiguous way, in particular with respect to solid and hollow parts, as compared to wireframe models.
- Kinematics
  Kinematics describe the restrictions that interconnected rigid bodies, i.e. the solid geometric entities described above, impose on each other's movement. Kinematics do not take the forces that cause movement into account, but focus on the degrees of freedom of the parts of the model and their connection by joints. COLLADA supports

two types of joints natively, namely prismatic and revolute, from which more complex types of joints can be derived.
Objects that are connected by joints can be grouped to form a kinematic chain. This chain is fully described by enumerating its rigid components, the relative displacement of consecutive components, the connecting joints, and the attachment of each joint to the components it connects.

- Physics
  COLLADA supports the description of basic rigid body dynamics. This includes force fields, e.g. to model gravitational pull. To aid physics engines with the computation of collision responses, parameterized primitive shapes – such as box, cylinder or sphere – can be associated with elements. Moreover, the convex hull of an element can be specified in form of a convex mesh grid.

3.5.2 References

- https://collada.org

3.6 PLCopenXML

3.6.1 Introduction

PLCopen is a none-profit, vendor- and product-independent association active in Industrial Control. It supports the use of international standards in this field and has several technical and promotional committees. The activities of PLCopen are based on the IEC 61131-3.

PLCopen specifications are normally based on IEC 61131-3. Especially the definition of the XML exchange format goes beyond the scope of IEC 61131-3. The scope is to provide a technical basis for user needs to exchange their programs, libraries and projects between development environments (engineering tools). PLCopen defined an open interface between all different kinds of software tools, which provides the ability to transfer the information that is on the screen to other platforms. This screen information does not only contain textual information, but also graphical information, like where the blocks are and how they are connected. The main goal is shown in Figure 8.

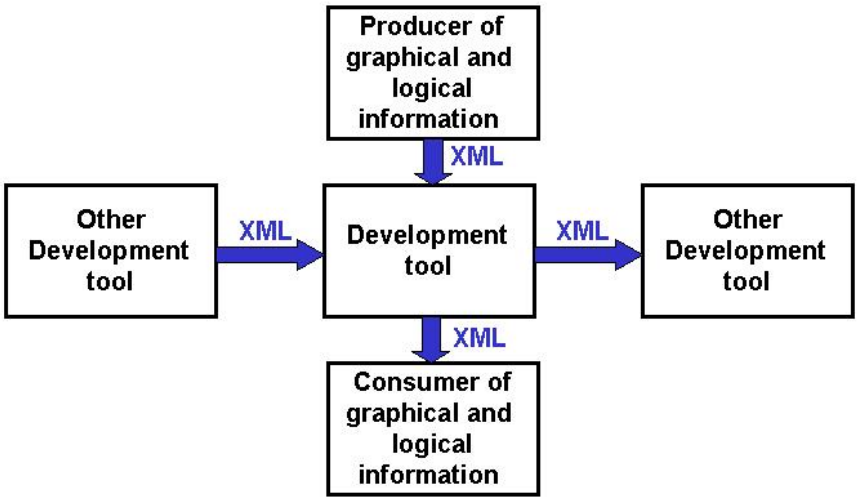

Figure 8: Usage of PLCopen XML exchange format

The exchange format supports following items:

- Textual Programming Languages – Instruction List (IL) and Structured Text (ST)
- Graphical Programming Languages – Ladder Diagram (LD), Function Block Diagram (FBD)
- Structural Language – Sequential Function Chart (SFC)
- Graphical Information, like place and position, and routing of connections
- Comments
- Program Organization Units – (User Derived) Functions and Function Blocks, Programs
- (User Derived) data types
- Project information (layered structure)
- Mapping information
- Supplier specific information

The definition of the exchange format allows exchanging a complete controller program. The exchange format is defined as an XML schema. The latest enhancements of IEC 61131-3 Ed. 3 (object orientation) are not covered by the exchange format.

In addition to the PLCopen exchange format, German standardization committees at DKE

Important for AVANTI is the possibility to express dynamic behaviour by means of SFC. SFC is similar to Petri-nets a good basis for deriving test cases from the behaviour description. Furthermore PLCopen XML exchange format is the preferred exchange format used in AutomationML (see section 3.7) to describe components with programmed or dynamic behaviour. Disadvantageously is the missing intellectual property protection.

3.6.2 References

3.6.2.1 Latest version of relevant specifications, RFCs, standardization activities

- PLCopen XML schema, version 2.0, in December 2008
- IEC 61131-3: Programmable controllers - Part 3: Programming languages, Ed.3, 2013

3.6.2.2 Network activities

- Activities in PLCopen association
- Activities in AutomationML association
- Joined work between the associations

3.7 AutomationML

AutomationML (Automation Markup Language, sometimes abbreviated *AML*) is a free and open format for storing automation data. It is intended as an exchange format in heterogeneous toolchains. AutomationML is not a language per se but rather a collection of three (free and open) XML-based languages that are domain-specific to the different aspects of automation resources. In a nutshell, we have

AutomationML = CAEX + COLLADA + PLCOpen,

where

1) CAEX describes the topology of a production plant, down to the component level (e.g. a conveyor or robot),
2) COLLADA describes the geometric, kinematic and possibly physical properties of the components and
3) PLCopen describes the control logic of the connected PLCs.

In particular, CAEX serves as the top level format that associates components with each other as well as with their physical and functional aspects.

The “interface”-primitive provided by CAEX is used to reference plant components within CAEX, e.g. for specifying a signal used for inter-component communication. Moreover, COLLADA- and PLCOpen-documents are referenced via interfaces, too. To this end, AutomationML provided special interface classes, giving such interfaces a semantic.

Specific engineering aspects are predefined by AutomationML, using CAEX primitives. Such are *Ports* (bundled interfaces), *Facets* (restrictions to a subset of attributes and data; intuitively: views on data) and others. Likewise, a catalogue of *roles* for plant components is predefined in AutomationML. A role is an attribute that provides semantics for a component; roles are, e.g. *transport*, *gate* or *tool*. A single component may be associated with several roles.

3.7.1 References

- https://www.automationml.org

3.8 Resource Description Framework (RDF)

3.8.1 Introduction

An open issue of description languages is the missing semantic of the described artefacts. Normally this lack is avoided by means of specific instance data names, e.g. by defining specific entry points into the description. Nevertheless this approach is not quite well. The description of semantics is a wide research field addressed by computer science. One of the most popular approaches to describe facts from the real world including the semantics are ontologies, known languages are Web Ontology Language (OWL) and also Resource Description Framework (RDF). OWL uses the basics of RDF but is enhanced and quite more complex. In principle OWL is able to describe each artefact of the real world and also reasoners are able to find answers inside the described domain. But establish a consistent ontology by means of OWL is a very complex process. Thus RDF shall be considered in here representatively, because AVANTI has a clear defined domain.

RDF can be used in order to express knowledge about relationships between exiting objects or items. Figure 9 gives an overview about the relationships between basic terms in information modelling (according to [Eppl11]). Objects are described by their properties. Each properties will be describes more precise by their characteristics.

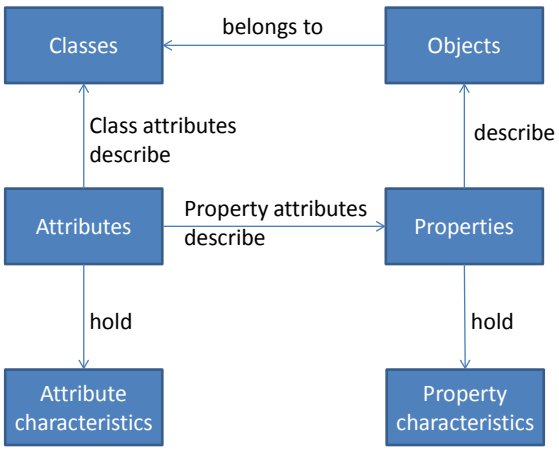

Figure 9: Relationships between the basic terms

RDF was developed to provide a simple data model, easy to manipulate by different applications. The formal specification of the semantics can be enhanced by new vocabulary using URIs. An RDF document describes directed graphs consisting on the set of nodes. Basis graph is a triple: <subject, predicate, object>. This means, if two objects are in a relationship, then there is a triple in the RDF. Subjects are also objects. So implicitly a more complex graph can be built. From the sentential calculus point of view, RDF describes true statements.

RDF provides an easy to use mechanism to express relationships, but the semantic is undefined. The following small example demonstrates this; Figure 10 shows the graphical representation.

<'Daimler', 'builds', 'Car'>

<'Smart', 'builds', 'Car'>

<'C180', 'is a', 'Car'>

<'E220', 'is a', 'Car'>

<'C180', 'has', '4 wheels'>

<'E220', 'has', '4 wheels'>

...

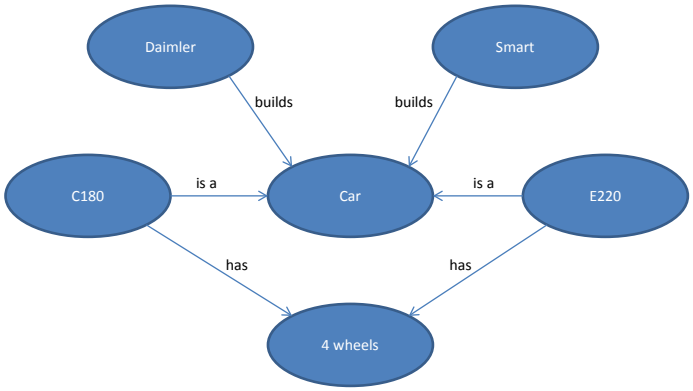

Figure 10: Example with 6 statements as a graph

The lack of type and property information shall be eliminated by an improvement called RDF Vocabulary Description Language Schema (RDFS). Within RDFS it is possible to enhance the descriptions by semantics by means of type information (classes) and properties. But it do not express transitive, inverse or symmetrical features.

3.8.2 References

3.8.2.1 List of conferences covering relevant topics

- ISWC – International Semantic Web Conference
- SWAT4LS – Semantic Web Applications and Tools for Life Sciences
- WIMS – International Conference on Web Intelligence, Mining and Semantics
- SemTechBiz – Semantic Technology & Business Conference

3.8.2.2 Relevant publications

- [Epp11] U. Epple. Merkmale als Grundlage der Interoperabilität technischer Systeme. At – Automatisierungstechnik 59 (2011), Juli, Nr. 7, 2011.
- Pascal Hitzler, Markus Krötzsch, Sebastian Rudolph, York Sure: Semantic Web. Grundlagen. Springer, Berlin u. a. 2008, ISBN 978-3-540-33993-9.
- Shelley Powers: Practical RDF. O'Reilly, Beijing u. a. 2003, ISBN 0-596-00263-7.

3.8.2.3 Relevant products or links to company statements

3.8.2.4 Latest version of relevant specifications, RFCs, standardization activities

- http://www.w3.org/RDF/
- http://www.w3.org/1999/02/22-rdf-syntax-ns

3.8.2.5 Relevant projects

- Theseus (BMWi)
- FOAF – Find of a Friend
- SemanticGov – EU, FP6-2004-IST-4-027517
- SemProM (BMBF)
- MEDPILOT (DFG, BMfG)
- GRENPILOT (DFG)

3.8.2.6 Network activities

- NEPOMUK - Networked Environment for Personalized, Ontology-based Management of Unified Knowledge, Open-Source Framework

3.9 eCl@ss/PROLIST

3.9.1 Introduction

eCl@ss is a cross-industry product data standard for classification and clear description of products and services. Meanwhile it covers also the product data description of PROLIST, because both associations have been joined. The list of properties described by these specifications achieves seamless integration of the workflows of all parties involved in plant life cycle management. The machine-readable descriptions of the properties of process automation components are relevant for engineering purposes and shall assist different phases of the engineering.

eCl@ss has established itself as the only ISO/IEC compliant industry standard nationally and internationally.

Besides its classical applications such as procurement, controlling and distribution, eCl@ss release 8.0 also shows particular strength regarding company-wide process data management as well as engineering.

The eCl@ass / PROLIST approach defines a basic set of well-defined elements, where the semantic is clear. In principle other descriptions can refer to these defined elements in order to reuse the existing semantic. DIN cooperates with DIN.

3.9.2 References

3.9.2.1 Latest version of relevant specifications, RFCs, standardization activities

- Version 8.0

3.9.2.2 Network activities

- eCl@ss e.V. association

3.10 MathML

3.10.1 Introduction

MathML is an XML based description format in order to describe mathematical equations. It is a recommendation of the W3C math working group and part of HTML5. The language has in focus to represent mathematical notations and capturing both its structure and content. Thus no rules exist how to resolve the equations at runtime and to describe the context and runtime artefacts. This means there is no binding to external information sources and to resolvers.

3.10.2 References

3.10.2.1 Relevant publications

- RFC of W3C

3.10.2.2 Relevant products or links to company statements

- Several editors and convertors (e.g. to TeX)

3.10.2.3 Latest version of relevant specifications, RFCs, standardization activities

- Language Version: 3

3.10.2.4 Network activities

- W3C

3.11 Modelica

3.11.1 Introduction

Modelica is a freely available, object-oriented language for modelling of large, complex, and heterogeneous systems. It is suited for multi-domain modelling, for example, mechatronic models in robotics, automotive and aerospace applications involving mechanical, electrical, hydraulic control and state machine subsystems, process oriented applications and generation and distribution of electric power. Models in Modelica are mathematically described by differential, algebraic and discrete equations. No particular variable needs to be solved for manually. A Modelica tool will have enough information to decide that

automatically. Modelica is designed such that available, specialized algorithms can be utilized to enable efficient handling of large models having more than one hundred thousand equations. Modelica is suited and used for hardware-in-the-loop simulations and for embedded control systems.

3.11.2 References

3.11.2.1 List of conferences covering relevant topics

- International Modelica Conference

3.11.2.2 Relevant products or links to company statements

- CyModelica (CyDesign Labs)
- Vertex (CyDesign Labs)
- Converge (CyDesign Labs)
- Dymola (Dassault Systemes)
- MOSILAB (Fraunhofer FIRST)
- SimulationX (ITI GmbH)
- MapleSim™ (MapleSoft)
- OpenModelica (Open Source)

3.11.2.3 Latest version of relevant specifications, RFCs, standardization activities

- Language Version: 3.3

3.11.2.4 Organisation

- http://www.Modelica.org/

4 Simulation tools for physics based Virtual Commissioning

4.1 Physics simulation

Physical simulation is the modification of simulated objects' properties according to the laws of physics. There is an enormous amount of physical effects which can be simulated and during the last decades several mostly domain specific methods for creating physical models and simulating the effects of interest have been developed and optimized for their respective field of application.

Well known physics simulations methods from an engineering perspective are for example:

- Finite-Element-Method (FEM)
- Finite-Difference-Method (FDM)
- Multi-Body Simulation (MBS)

All of the aforementioned methods have established themselves as valuable tools in engineering processes and lots of research has been made to enhance their accuracy and efficiency. A common property of these methods is that they rely on numeric methods to approximate a high accuracy solution for partial differential equations on irregular grids. This process is computationally expensive and currently no solution is known which is able to produce results for arbitrary systems interactively let alone in real time.

Virtual commissioning relies on supplying a real or simulated controller with the same inputs as a real plant would generate in order to allow testing the given control software and answering questions regarding the achieved production throughput and machine usage.

This also constitutes the first requirement towards any physics simulation which is to be used in VC. The simulation must be able to produce reliable results in real time.

4.2 Collisions

Interactive physics simulations are focused on providing good results for the interaction of rigid bodies by means of collisions. To achieve this, the simulation software generally loops through the process of

- Collision detection and subsequent
- Collision handling

before updating the internal states.

Optionally, the current state of the system can be visualized after each iteration of this cycle, but this is not part of the simulation process and it remains to be evaluated if visualizing every state is necessary in Virtual Commissioning. It might be more suitable to disregard the visualization and using the saved computational resources for improving the simulation results.

4.2.1 Collision detection

Many actions in a simulated environment originate from physical contact between two or more simulated objects. So the primary step in interactive physics simulations is to determine which objects in the virtual scene collide.

There are two different approaches to determining collisions:

- Continuous collision detection (a priori)
- Discreet collision detection (a posteriori)

In the a posteriori case, the physical simulation advances by a small time step, then checks if any objects are intersecting, or are close enough to each other that they are considered to be intersecting.

At each simulation step, a list of all intersecting bodies is created, and the positions and trajectories of these objects are corrected to account for the collision.

This method is called “a posteriori” because the actual instant of collision is missed, and the collision is only detected after it has actually happened.

In the a priori methods, the collision detection algorithm is able to predict very precisely the trajectories of the physical bodies. The instants of collision are calculated with high precision, and the physical bodies never actually interpenetrate. This is called “a priori” because the instants of collisions are calculated before any updates to the configuration of the physical bodies and so collision will be detected before they actually happen.

Benefits of discreet collision detection:

- The collision detection algorithm need not be aware of the possibly complex nature of the system, since it merely acts on a fixed state of the system represented by a simple list of physical bodies.
- The collision detection algorithm doesn't need to evaluate friction, elastic collisions, non-elastic collisions and deformable bodies.

- Due to the lack of time as a variable to the collision detection algorithm, discreet algorithms are one dimension simpler than the continuous algorithms.

The challenging part of discreet algorithms is a posteriori calculation of a correct collision response. A worse problem occurs if the discrete step is not related to object's relative speed. A collision could go undetected if a small fast moving object approaches another thin object. If the time step is too big the first object passes through the second unnoticed since there never is an interception.

The benefits of the a priori algorithms are increased fidelity and stability at the cost of having to use numeric algorithms to approximate the collision instant, since it cannot be directly computed in reasonable time.

4.2.2 Collision Response

4.2.2.1 General

Computation of collision responses is based on the results of collision detection, which at least include a reference to the objects colliding and the coordinates of their contact points.

When handling collisions, two major fields of research have to be distinguished, the first being soft body collisions and the second one being rigid body collisions.

Soft body collisions are a much complex and computationally expensive and have for a long time played no role in interactive physics simulations because no methods for approximating them in real time were available. This has changed due to increased research efforts and new approaches exploiting the advances in parallel processing using graphics cards and the available APIs (CUDA, OpenCL, …). as for example in TRAMES. (4.5.1)

For rigid body collision responses the following two different calculation approaches are available.

4.2.2.2 Impulse based collision response

Impulse based collision handling starts by determining one pair of closest points for every pair of colliding objects. At these points an impulse is generated and applied which causes the objects to separate at these points. Unfortunately this can lead to new collisions which must be resolved subsequently. This is the reason why impulse based collision handling is continued in a loop until all collisions are resolved. Furthermore the limitation of solving only one collision contact at a time is the reason for instable behavior when handling resting contacts. When two bodies rest on each other there is usually more than one contact, in an ideal case of a box lying on a table there would be an infinite number of contacts, but only one contact is solved at a time which leads to a vibrating motion. There are different approaches to stabilizing the collision resolution behavior when using impulse based techniques but there is also a completely different approach which does not suffer from this instability.

4.2.2.3 Constraint based collision response

Constraint based collision responses are based on the non-penetration constraints which are created for each contact that occurs. These constraints are unilateral and can be formulated as a linear complementarity problem (LCP). Solving this LCP gives the resulting contact forces. An alternative approach is formulating these constraints as a quadratic program and solving it by using quadratic programming algorithms.

4.3 Requirements for physics simulation frameworks

Solving classical physics problems using numerical integration is a common and widely used technique and solving the ordinary differential equations describing these problems has been thoroughly researched. There are efficient methods which can solve special forms of ODE to any desired level of accuracy.

Unfortunately the equations of motion of constrained mechanical systems are differential algebraic equations (DAE) of index 3 and numerical solution methods for those do not have the same maturity status as ODEs.

Eventhough there are numerical integration methods for non-smooth problems like collisions, contacts and dry friction, selecting a system for interactive simulation of physically correct behavior, to a degree suitable for Virtual Commissioning, is a non-trivial task since the requirements for interactive physics differ from those of standard numerical integration methods.

Interactive physics simulation for VC is a real-time, dynamic, nonsmooth, problem. In this context also, overall stability and speed have priority over local accuracy as long as the error is bounded and predictable.

The resulting requirements have already been categorized and collected in [8]. When applied to industrial contexts the following requirements must be met:

- Real time:
  Because of the interactive context of VC, i.e. the virtual environment being connected to some outside entity, being a real controller, a simulated controller or a HMI-Panel, the simulated system must behave continuously regarding time, time must flow uniformly, at least as seen from outside of the virtual environment. Additionally updates must be predictable and fast. Refresh rates must be fast enough to simulate sensors which are required for correct operation of connected controllers.
- Stability:
  There are two different kinds of stability required from physics simulations. On the one hand, as the main field of interest of VC is testing new systems, the results of a simulation must be deterministic and so reproducible. On the other hand the simulated systems resemble real mechatronic systems so unstable solutions resulting in oscillating movement, exploded geometries and other well-known problems from computer games physics engines must be avoided. Since dealing with physical systems, total energy is the natural measure of stability. Integrators should be biased so that numerical energy is monotonically non-increasing, or oscillatory within global bounds when the system is conservative.
- Dynamic reconfiguration:
  Plant behavior is neither known in advance nor bounded in variations in any way, and can include arbitrary changes to the physical problem under consideration at any time. For instance, a cutting machine might cut sheet metal. This discrete event deletes one body and creates two new ones while the simulation is running. Likewise, a badly programmed robot can crash into another one at any point in time, thus dynamically adding several non-penetration conditions to the simulated system. The engine must thus be nearly stateless and designed to handle addition and deletion of bodies, constraints, and forces, or arbitrary changes in physical parameters, at any point in the simulation, without requiring too much computational effort. Some persistence and state information can be useful in reducing average computational

costs but this information should be easy to reconstruct when it must be deleted, and not be essential for the computation of the next configuration.

- Non-smooth geometry and non-penetration conditions
  Solids must not be allowed to interpenetrate and this restriction must be imposed at the geometric level. But geometric models are not generally convex and may have arbitrary discontinuous shapes. This may result in sharp edges, kinks, and corners so that the surface normals of these models are not continuous functions. Thus, non-smooth analysis is required to correctly determine contact regions and normals.
- Impacts
  Non-penetration conditions lead to very large and rapid changes in the velocities when collisions occur. Impacts. These are instantaneous at time and spatial resolutions used in interactive applications. The physics engine must correctly process localized step discontinuities in velocities. For instance, collision between two objects is an instantaneous event at the time scales considered. During the actual impact, which might last just a few microseconds, the velocities change abruptly but the positions are not altered significantly.
- Contacts and friction:
  Objects in contact are subject to dry friction and so are also subject to discontinuous transitions between static and kinetic friction. Such non-smooth phenomena must necessarily be reproduced in interactive physics and therefore, discontinuous velocities are expected and must be processed correctly. This leads to a situation in which accelerations are not well-defined everywhere, since a finite instantaneous change in velocities implies an infinite acceleration and thus, infinite forces as well. The physics engine should therefore be based on discrete time-stepping involving positions, velocities and impulses — time integrals of forces — but not directly on the differential equations relating accelerations and forces.
- Accuracy:
  Judging whether witnessed movements in a complex production plant simulation accurately resemble the movements of the real system is beyond the abilities of a human spectator. So care must be taken to ensure that chose quality, in terms of computational errors, can be guaranteed.
- No tuning:
  Since when working with data from real mechanical systems, including its engineering data, all masses, friction coefficients etc. are known or relatively easy to acquire. The physics simulation should be able to work with these values and not require any adjustments to produce valid simulation results.

4.4 Functional and non-functional properties of physics engines

In this section the functional and non-functional properties of different physics engines will be examined. Because of their ability to solve many physical problems of mechanical systems interactively, game physics engines have been in the focus of research projects like[7] and [9].

In [7] and [9] a selection of criteria and tests is presented which serves as a basis for judging the potential of different game physics engines. Based on these tests a set of candidates shall be chosen which will then be further evaluated.

The evaluation process will include rerunning the tests introduced by these researches because there has been a huge leap in development since 2006 especially in using parallel processing on recent graphics hardware and the results in terms of accuracy and performance are expected to vary greatly from the results in 2006.

4.4.1 Physics Engines selection

There is a broad variety of physics engines available, some free and open source others commercially licensed. While open source engines provide greater means of flexibility since they be extended and modified easily to provide the required functions, given the required physical knowledge and programming skills. On the other hand excluding commercial physics engines from the research due to their license terms may distort the results whilst considering and testing them still leaves developers with the option of deliberately choosing an open source engine and knowing the differences to be expected.

Physics engines selected for further consideration are:

- PhysX Version version 3.3 (Nvidia)
- Havok Physics 2013 (Havok)
- Bullet Versions 2.8 and 3.0 (http://bulletphysics.org)
- Open Dynamics Engine (ODE) version 0.12 (http://www.ode.org/)

Engine	License	Cost (Edu/Commercial)	Supported Platforms
NVIDIA PhysX	EULA	Free/Free[1]	Windows, Linux, OS X, iOS, PS3, PS4, Wii, … and more
Havok	EULA	Free[2]/non-free	Windows, Windows RT, Linux, Android, iOS, OS X, Xbox 360, PS3, PS4, Wii, ... and more
Bullet	zlib	Free/Free	Windows, Linux, iOS, Android, PS3, Xbox360. Wii
ODE	BSD or LGPL	Free/Free	Windows, Linux, PS3, PSP, OS X, Xbox 360

The determining factor for selected these engines as candidates for further evaluation were the tests made by Seugling06 and Boing09 as well as the engine's references of usage. The assumption was made that engine's which have actually been used in several projects tend

[1] Free for commercial usage up to annual revenue of $ 10.000.

[2] Free evaluation period including support for research projects.

to be more mature. Additionally only engines were considered which are still being actively supported and developed.

Engine	Simulation projects	Game Projects
NVIDIA PhysX	Acive Worlds, Microsoft Robotics Studio,	Call of Duty: Ghosts, Assassin's Creed, Crazy Machines 2
Havok	Kongsberg: Naval Training Environment, Nova Technologie: U.S. Army's Call for Fire Trainer, Lockhead Martin: Small Arms Training	Metal Gear Rising, The Last of us, Assassin's Creed III, Halo 4, Modern Combat 4: Zero Hour
Bullet	OpenSimulator, V-REP, Houdini, Poser, Modo	Toy Story 3, GTA 4, GTA 5, Deus Ex, Sony's Free Realms
ODE	OpenSimulator, Webots, ARS, V-REP	Resident Evil: The Umbrella Chronicles, World of Gooo, Stunt Marble Racers, Call of Juarez

There are different properties that influence the behavior of a physics engine. These include:

- the simulation paradigm,
- collision detection technique,
- collision response mechanism,
- type of numerical integrator,
- and which physical effects are considered by default (i.e. air resistance).

4.5 Industrial application of physics-based modelling for Virtual Commissioning of automated production systems

Physics-based modelling of automated production systems for Virtual Commissioning currently comprises multi-body simulation of rigid bodies and particles. Internally physics engines simplify rigid bodies as a system of particles considering rotation, whereas soft bodies are much more complex to handle. Due to their flexible shape, soft bodies can collide with themselves, cannot be subdivided into simple shapes, and underlay continuous change of center of gravity and moments of inertia. Soft bodies are a huge research field and have not been addressed in physics based modelling for VC yet. Industrial applications focus on physics based modelling for rigid bodies and particles.

4.5.1 Research projects with respect to physics based modelling for Virtual Commissioning of automated production systems

Some industrial-close research has been conducted in the area of physics based modelling of mechatronic systems. Different approaches focus on modelling and analyzing the behavior of mechatronic components, especially robots, e.g. in outdoor environments. With reference to VC the research was focused on flow of materials of production systems or defining a framework for executing VC for different types of production systems [1], [2], [3]. The types of production systems can be numerical machines or special purpose machines and vary in

size and functionalities. Some national and international research projects regarding physics based VC were conducted or are still ongoing, e.g. TRAMES, AutoPhyS. All of them aim at enhancing the concept of VC for different kinds of production systems with different areas of application.

Following some relevant research projects get briefly summarized:

- TRAMES (Transportprozesse in der Mechatronik-Simulation): this project aims at improving the simulation of transportation processes (material flow) of bulk goods using physics based modelling
- AutoPhyS (Auslegung automatisierter Produktionsanlagen im Automobilbau mit Hilfe der physikbasierten mechatronischen Simulation):
  the project focusses on establishing current research approaches of physics based modelling in development processes of automated production systems in the automotive industry
- PhySiMa (Methoden und Systeme zum Einsatz von Physikmodellen für die Simulation des maschinen- und anlageninternen Materialhandlings zur Virtuellen Inbetriebnahme):
  this project deals with developing phyics based models for VC of production systems with high material flow
- PhySiOS (Physiksimulation zur effizienten Auslegung mechanischer Ordnungsschikanen):
  this project also aims at simulating the physical behavior of bulky goods for virtual validation purposes, especially with respect to vibrating production systems
- A concurrent design methodology of a production system for VC
  The objective of this study is to provide a design procedure for VC between a real controller and a virtual plant The procedure consists of four steps: process design for defining Sequence of Operations (SOP), physical device modelling for simulation of geometry and kinematics, logical device modelling for checking controller behavior, and control logic design. Conventionally, the design of a production system is performed sequentially, which causes delays in development (e.g. electrical designers typically wait until mechanical design is finished). In this project, this problem has been solved by applying a concurrent design methodology to VC. They demonstrate the applicability of this methodology with a press machine and a robotic system in a real world scenario.
- Virtual Assembly Verification with Haptic Feedback
  This project aims to both improve the product design by early detection of assembly problems and provide a training environment for manual workers who perform assembly tasks in the automotive industry [17]. Virtual assembly of a Volkswagen car is chosen as a case study. This case study helps to monitor the human aspect of a manufacturing process without building costly prototypes. In addition to visual feedback, haptic feedback is used for enhancing the realism of the virtual interactions. In order to provide haptic feedback, a bimanual haptic device, which is developed by German Aerospace Center (DLR), is used. This device displays forces, which result from the interaction of the objects colliding in a virtual environment, to its user via two KUKA Lightweight Robots. In the virtual environment, collision detection and force computation is performed by the Voxmap-Pointshell Algorithm. This algorithm uses two different data structures, voxels and pointshells, for respectively representing static and moving objects. In the scope of this project, haptic and visual feedback modalities are compared and the superiority of haptic feedback is verified with both objective and subjective measures [18]

4.5.2 Physics based modelling for Virtual Commissioning of automated production systems in automotive industry

In automotive industry the application of physics based modelling of automated production systems for the purpose of Virtual Commissioning is still subject of research and is currently not an essential part of the VC process. Potentials for the application of the methodology physics based modelling are getting identified [4]. Current research approaches in automotive industry focus to simulate the behavior of pneumatic components used in automated production systems.

Essential for the usage of physics based modelling for the methodology VC is the capability of the physics engine to provide simulation results in real-time. Due to the hardware-in-the-loop approach of Virtual Commissioning, requirements for the response time of the modelled system are strict and limited to some milliseconds. Other multi-body simulation methodologies providing more accurate calculation of the physical behavior (e.g. FEM) require much more processing time and are thus not applicable in Virtual Commissioning. An example is shown in Figure 11.

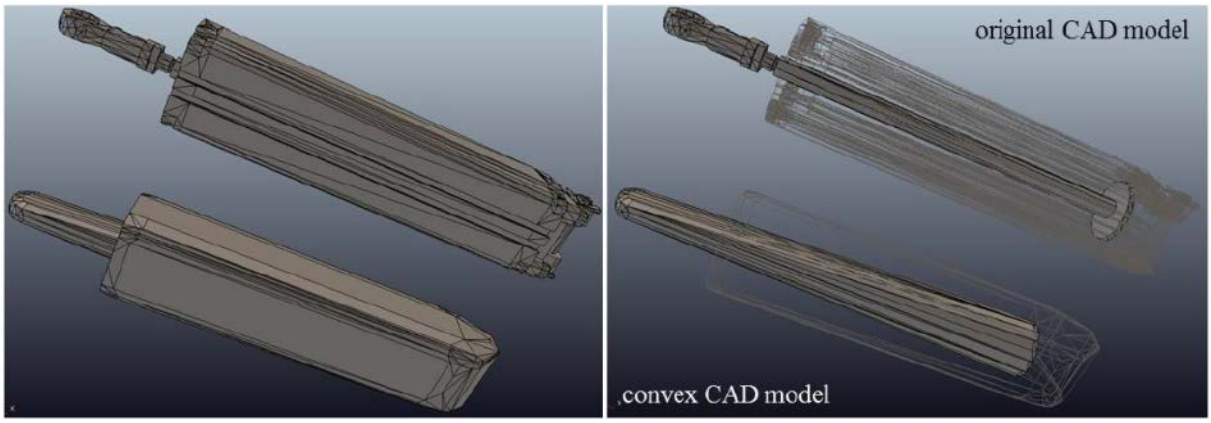

Figure 11: Convex model of a pneumatic driver [6]

For VC a model of the plant, the so-called mechatronic plant model is required. This model is classified into the expanded 3D geometry model and the behavior model. The expanded 3D geometry model thereby serves for the motion visualization of the plant components without considering their physical behavior. An expansion of the mechanic model by adding the collision model is presented in [5], the so-called physics-based plant model. With the help of this model collision detection and management under consideration of the physical characteristics becomes possible. This is based on the 3D geometry of the expanded 3D geometry model. The basic virtual design model (i.e. the CAD-model) of the production system simplified (see Figure 11) in the first place and afterwards gets moderately enriched in order to examine the system's dynamic behavior, e.g. unwanted collisions or friction, respectively. The system's dynamic behavior bases on physical equations (Newton, Coulomb) and can be simulated in real-time due to the support of optimized computer vision algorithms in many physics engines. As a consequence, manual modelling the object's dynamic behavior is obsolete.

Physics based modelling of automated assembly systems solely requires creating a kinematic model of the production system and physically parameterizing the system components and objects. Current research focusses on simulating the behavior of pneumatic components in automated assembly systems [6]. Due to the use of physics-based simulation assembly system's cycle time can be validated thoroughly.

4.5.3 References

4.5.3.1 Relevant publications

- [1] Lacour, F.-F. Modellbildung für die physikbasierte Virtuelle Inbetriebnahme materialflussintensiver Produktionsanlagen
- [2] Wünsch, G. Methoden für die Virtuelle Inbetriebnahme automatisierter Produktionssysteme
- [3] Reinhart, G. et al. Skalierbare Simulation kinematischer Strukturen in der physikbasierten Virtuellen Inbetriebnahme
- [4] Drescher, B., Stich, P., Kiefer, J., Strahilov, A. & Bär, T. Physikbasierte Simulation im Anlagenentstehungsprozess – Einsatzpotentiale bei der Entwicklung automatisierter Montageanlagen im Automobilbau. In: Simulation in Produktion und Logistik. HNI-Verlagsschriftenreihe, 2013. p. 271 – 282.
- [5] Spitzweg, M. Methode und Konzept für den Einsatz eines physikalischen Modells in der Entwicklung von Produktionsanlagen
- [6] Strahilov, A. et al. Simulation of the pneumatic behavior in the Virtual Commissioning of automated assembly systems
- [7] Boing, A., Bräunl, T. Evaluation of real-time physics simulation systems
- [8] Lacoursière, C. Ghosts and Machines: Regularized Variational Methods for Interactive Simulations of Multibodies with Dry Frictional Contacts
- [9] Seugling, A., Rölin M. Evaluation of Physics Engines and Implementation of a Physics Module in a 3d-Authoring Tool

4.5.3.2 Relevant projects

- AutoPhyS – Auslegung automatisierter Produktionsanlagen im Automobilbau mit Hilfe der physikbasierten mechatronischen Simulation (DFG)
- PhySiMa – Methoden und Systeme zum Einsatz von Physikmodellen für die Simulation des maschinen- und anlageninternen Materialhandlings zur Virtuellen Inbetriebnahme (DFG)
- TRAMES – Transportprozesse in der Mechatronik-Simulation (BMBF)
- PhySiOS - Physiksimulation zur effizienten Auslegung mechanischer Ordnungsschikanen (Bayerische Forschungsstiftung)
- A concurrent design methodology of a production system for Virtual Commissioning (A concurrent design methodology of a production system for Virtual Commissioning)
- Virtual Assembly Verification with Haptic Feedback (Virtual Assembly Verification with Haptic Feedback)

4.6 Commercial Solutions using physic-based simulation

The goal of Virtual Commissioning for factory automation is to develop a realistic virtual model of an automation system before the real setup is realized and then run virtual simulations on the model to test the functionality of the system via “what-if” scenarios. The major benefits are the shortening of the time spent from planning to real setup and early detection and debugging of errors in the system, both leading to significant reduction in costs. For this reason, VC of automation systems has gained importance in industry and academia during the last decade [1].In the following sections, the research and development efforts on VC are discussed under Commercial Solutions: the commercial software packages available in the market.

Virtual Universe Pro by Virtual Commissioning is a 3D emulation software that allows manufacturers to test system behavior in real time [2]. It provides control software designers with the opportunity of testing PLC programs before a real setup is available. Different CAD programs are supported to create realistic industrial settings. Also physics-based tests are available during VC. Consequently, time, cost and risk can be reduced for commissioning.

Mechatronics Concept Designer (MCD) by Siemens is a bundle of NX. MCD is designed to establish strong connection between the design tools [3]**.** MCD also enables the designers to evaluate their designs and debug them. Using MCD, collaboration of the mechanical, electrical and automation software disciplines is achieved from the early stages of design. MCD provides physics-based simulation via NVIDIA PhysX and hardware in the loop simulation via OPC.

Experior by Xcelgo is another commissioning solution, which consists of several software platforms such as Virtual Automation, PLC Tester, Simulation, and High Level Emulation [4]. Virtual Automation Platform provides not only 3D visualization, but also physics or discrete-based simulation. PLC Tester allows connection to a PLC for controlling and testing virtual models. Using Simulation platform users can test the design, operation, and performance of their systems. On the other hand, High Level Emulation platform provides an interface between simulation models and the Material Flow Control module for testing of a system before physical realization.

3DCreate by Visual Components Oy enables to test material flow and robotic behavior on the same simulation platform, and visualize the complete manufacturing system [5]. Level of detail can be increased in the specific areas of the model when it is required to test behaviors. General layout validations, like collision detection, resource utilization, and cycle times are also available.

Delmia by Dassault Systems connects virtual and real worlds and provides optimization of build-to-order and lean production manufacturing systems [6]. Operating a virtual production system helps to track real-time production activities, perform schedule changes, launch new programs and introduces model changeovers, and schedule maintenance operations. Process simulation, physics-based simulation, and ergonomic analysis can be performed [7].

Tecnomatix by Siemens offers wide range of tools to bring together all the manufacturing disciplines that are necessary for production system design [8]. It is built upon the open product lifecycle management (PLM) foundation called Teamcenter manufacturing platform. As expected from all VC tools, Tecnomatix also aims to make the production system design more efficient. It is capable of 3D kinematic simulations, static and dynamic collision detection, sequencing of operations, automatic assembly path planning, line and workstation design, 3D interactive documentation and ergonomic analysis.

WinMod by Mewes & Partner GmbH is a software suite for VC which enables to a) emulate/simulate the real signal periphery, b) simulate the behavior of components and processes in real-time, c) visualize signal values, conditions, signal sequences, kinematic movements in real-time [9].

tarakos product suite, taraVRbuilder to design and visualize manufacturing systems. taraVRcontrol uses the designed systems from taraVRbuilder to visualize the system-behavior, connected to a control system. Collision detection is also available [10].

4.7 References

- [1] P. Hoffmann, R. Schumann, T. Maksoud and G. Premier, "Virtual Commissioning of Manufacturing Systems a Review and New Approaches for Simplification," in *24th European Conference On Modelling And Simulation*, Kuala Lumpur, 2010
- [2] Virtual Commissioning, 17 May 2012. [Online]. Available: https://www.youtube.com/watch?v=WRwOr4-vMBI. [Accessed February 2014]
- [3] Siemens, [Online]. Available: http://www.plm.automation.siemens.com/en_us/products/nx/mechatronics_concept_designer/. [Accessed Fenruary 2014].
- [4] Experior, [Online]. Available: http://www.xcelgo.com/29-2/experior/. [Accessed February 2014]
- [5] M. Sacco, G. Dal Maso, F. Milella, P. Pedrazzoli, D. Rovere and W. Terkaj, "Virtual Factory Manager," in Virtual and Mixed Reality - Systems and Applications - International Conference, Virtual and Mixed Reality 2011, Held as Part of HCI International, 2011
- [6] Dassault Systems, [Online]. Available: http://www.3ds.com/products-services/delmia/. [Accessed February 2014]
- [7] Haption, [Online]. Available: http://www.haption.com/site/index.php/en/products-menu-en/software-menu-en/rti-menu-en. [Accessed February 2014].
- [8] Siemens, [Online]. Available: http://www.plm.automation.siemens.com/en_us/products/tecnomatix/. [Accessed February 2014].
- [9] WinMod, [Online]. Available: http://winmod.de/en/index.php?page=systemplattform-2. [Accessed 21 February 2014].
- [10] tarakos GmbH [Online] www.tarakos.de

5 Requirements for test generation and test execution

5.1 Description of state-of-the-art and latest solutions, relevance to AVANTI

5.1.1 Test generation and model-based testing

5.1.1.1 Introduction

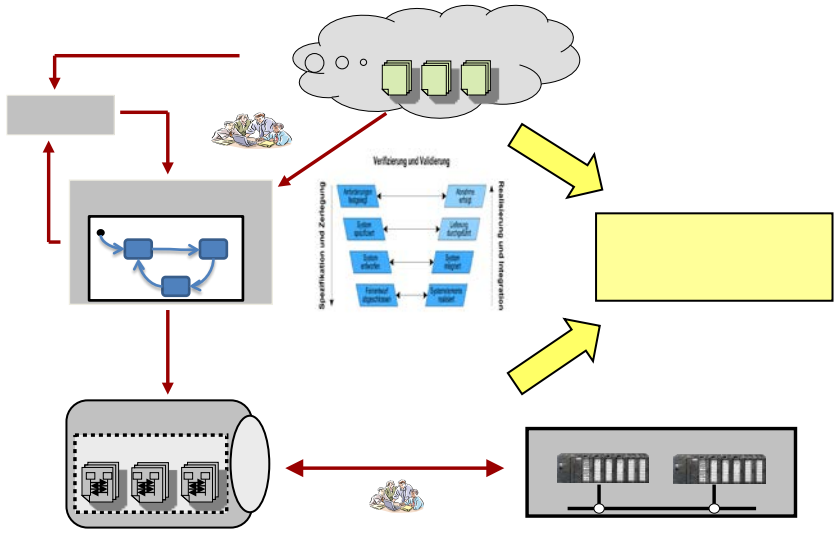

Figure 12: General model based testing process

The basic idea of model-based testing (*MBT*) is to transfer the actions involved with the testing process to an automated level as shown in Figure 12. This encompasses several steps of the testing process, namely test derivation, test execution and evaluation of testing results. The core of MBT is the availability of a model of the system, which allows for these actions. Necessarily, it must provide a formal representation of the requirements that shall be implemented with the system – i.e. a representation that can be processed algorithmically. This formal model is called the *specification model* of the system. Construction of the specification model is mainly a manual task as it usually integrates requirements from different sources that cannot be expected to be a) already present in a formal representation and b) compatible with respect to the chosen language for the specification model.

Ideally, the requirements are formalized and integrated into a single model, which supports the traceability of model components back to the requirements. The models best suited for discrete event driven systems are (finite) state models. Two prominent classes of this type are statecharts and Petri-Nets; the latter of which is particularly well suited for the formalization of PLC-programs. Either formalism is also supported by UML to some extent, which might allow a smooth transition from requirements that are pre-formalized in UML as state-diagrams (related to statecharts) and activity-diagrams (related to Petri-Nets). On the other hand, if the requirements are present in a suitable logic representation, notably temporal logic, the state model can be verified against the requirements, using model checking techniques, in order to verify that the (formalized) requirements are properly represented by the model.

Based on the specification model, a test system creates test cases, i.e. event sequences that must be correctly processed by the implementation. As state models are essentially annotated (directed) graphs, this comes down to covering the graphs by paths, which result

in input event sequences. The reasonable coverage objectives are state coverage and transition coverage; path coverage can generally not be enforced, since the graph model may (and usually does) contain cycles – which leads to an infinite set of possible paths.

The test system runs the implementation on these input sequences and evaluates the reaction of the implementation to generate a test metric. In case of a failed test, the test system traces the step which led to an error back to the model, i.e. a particular state or transition therein, and further back to the requirement that gave rise to this part of the model.

Procedure:

- Formalization of the requirements
  In order to allow for a formalization of functional requirements, certain principles of requirements engineering must be respected. In particular, requirements must be *consistent* (i.e. non-contradictory), *unambiguous* (preferably adhering to a naming convention) and – in order to evaluate testing results – *measurable*. Depending on the chosen modelling approach, the presence or absence of these properties will become apparent in the process of constructing the model, though, and can be enforced by iterating the requirements elicitation phase. Other properties, such as *structure* and *completeness*, are independent of the level of formality of the presentation and will carry over to the formalized requirements.
  Based on the requirements, a system is defined the meets the requirements. A system may be composed of any number of (possibly interacting) elements. In order to be suitable for the derivation of a specification model, this description must respect the very principles listed above for requirements.
- Derivation of the specification model
  The “pre-formalized” requirements / specifications are integrated into a model. There are several ways of doing so, which, however, come essentially down to constructing a transition system representation of the specified system. To this end, the modeller extracts the states and the inputs of the system from the requirements and the input-guided transitions among states.
  There are several possible approaches for building the mode; these are
  1) The *manual* and direct construction of the model. The resulting model is based on the modeler’s experience and view of the specifications.
  2) The *guided* transformation of requirements. To this end, the requirements may be put in a dependency relation, which is refined to a state transition system in a step-wise fashion. A technique as such is described in [2] where requirements are first translated to a tree-structure, which is then transformed to a transition system.
  3) The *automated* construction from an intermediate formal model. To this end, the specification is “implemented” in a model program. This program catches crucial properties (e.g. variable values) and communication primitives among actors, but abstracts from implementation details that in fact realize required or specified properties.

The model may be further transformed by adding detail to the model or abstracting from detail. For a transition system, this comes down to adding further states and transitions that capture behaviour (model-refinement) or by merging states into “super-states” that hide detail in black boxes (model-abstraction). This control of detail may as well be exercised on the specification-level but is arguably easier on the model-level.

In general the techniques of creating test cases with methods of model based testing can be used in all activities of a development process (like the stages in the V-Model, see

Figure 13). Notice that the abstraction level of the tests matches that of the model; this also implies that refinements on the model carry over to refinements on the tests.

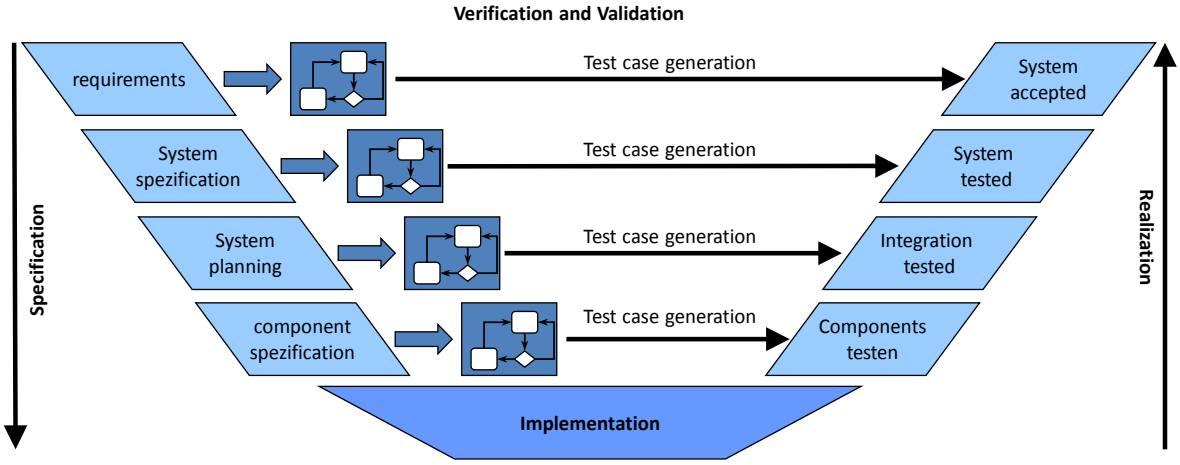

Figure 13: Model based testing and the V-Modell

5.1.1.2 State of the art, industrial tools

- The *Conformiq* tool suite consists of two applications, namely *Conformiq Modeler* and *Conformiq Designer*, to model the systems's nominal behaviour and to create tests from the model, respectively.
  The modeller tool allows to specify the model as a set of hierarchical statecharts or in form of program written in *QML*, the *Conformiq Modeling Language*, which resembles Java. The two model notations may also be combined.
  The designer tool creates test cases based on the model and the chosen coverage criterion. Based on the nature of the model, i.e. statecharts and/or QML, a coverage criterion can be chosen that puts emphasis on a particular aspect that is present in the model – such as state or transition coverage (relative to a state machine model) or control flow (relative to a QML model). The designer tool supports the import of statechart-models models from third-party tool, notably Enterprise Architect. Moreover, the tool may interact with the requirements engineering tool DOORS and a coverage criterion can be specified relative to requirements. This allows for the traceability of requirements down to test results. Tests are expressed in an abstract (language-independent) format, TTCN-3, and can be converted to several target-language via an adaptor.
- *MS Spec Explorer* is developed by Microsoft for integration with the Visual Studio -development workflow. It generates a transition system model based on a model program written in a dedicated language which resembles a programming language. Two such languages are supported for modelling, namely *Spec#*, an imperative language which is syntactically similar to C#, and *ASML*, which is a functional language. The use of a model-program supports developers who are not explicitly versed in working with strictly formal methods (i.e. transition systems) and are thus given the ability to express models in a familiar environment.
  In MS Spec Explorer, a model program specifies the system via sets of objects and actions that may alter members of these objects. Control structures, i.e. loops and branches, can be specified in these objects, as well as conditional access. The model

program is internally transformed to a transition system representation which can be examined graphically (but not edited). In order to make the representation visually comprehensible or to emphasize particular behavioural aspects, abstractions can be defined – which are called “projections” in the tool. Moreover, projections will also be applied by the tool automatically if the state space of the transition model is infinite – this might happen for unbounded variables.
Tests are derived from the transition model by state space exploration. This can be further controlled by the user. In particular, an intermediate layer allows to map actions of the abstract model with function calls (or returns) in the implementation code. This layer is necessary if the implementation API does not coincide with the action names of the model.

- Other industrial tools
  - MBTSuite (http://www.mbtsuite.com/) by sepp.med
  - TestCast MBT edition (www.elvior.com)

5.1.1.3 General process for using within Virtual Commissioning

The procedure for generating test cases based on a specification model of the requirements is also useable within Virtual Commissioning for validating of components, systems of components, controls, robot programs, etc. . Thereby the biggest challenge is the implementation within a useful, practical relevant development process. In Figure 14 a general model based testing process for VC is illustrated.

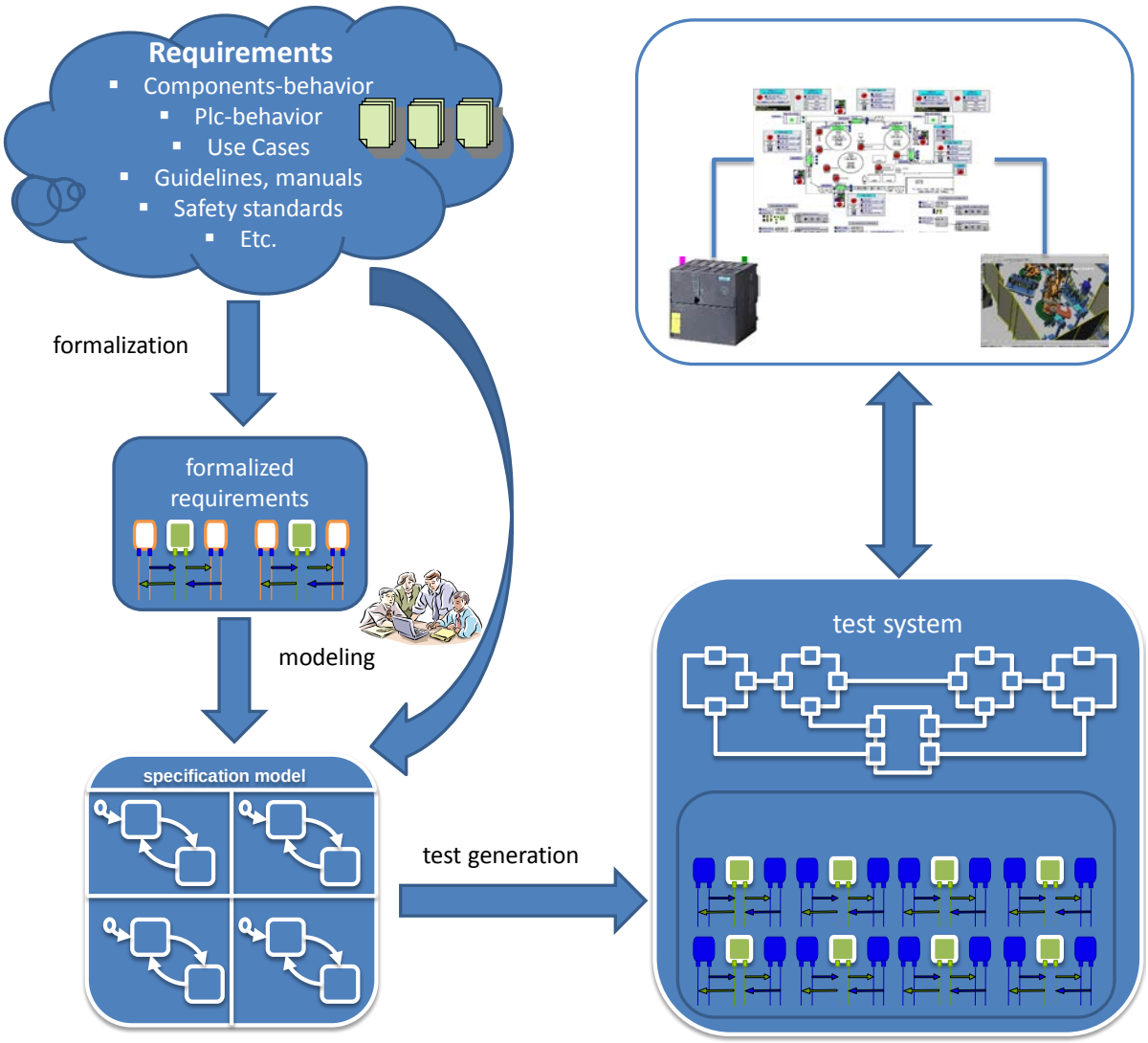

Figure 14: Process for model based test generation and automatic test execution within Virtual Commissioning

The biggest challenge will be the identification of relevant requirements on the behavior of single virtual components, of the controls (PLC, robot controls) and the whole automation system. Existing processes have to be analyzed in respect to

- Standardized descriptions of expected behavior (e. g. with Gantt Charts),
- Documents,
- Guidelines,
- Manuals,

Based on these information methods for the easy derivation of suitable specification models have to be developed. Specification models are used for generating test cases, but it is of huge importance that all created methods and tools have to be developed and evaluated with respect to demands of the end users.

5.1.2 (Automated) test execution

5.1.2.1 Introduction

In order to implement an efficient testing process with sufficient test coverage based on generated test cases an infrastructure for high automated test execution is necessary. Currently test execution is manually realized within VC. Focus is thereby the testing of PLC programs. Within AVANTI methods and tools for an efficient test infrastructure useable within VC have to be developed. The automated testing of virtual components, of (sub) systems of components and the whole automation system with PLCs and robots should be possible with these new approaches. For this the following questions have to be answered within AVANTI:

- How can virtual components/(sub) systems be stimulated?
- How can errors be created (signal deleting, signal manipulation, etc.)?
- How can virtual components/(sub) systems be observed?
- How can test suites be specified in a simple, flexible, reusable and standardized (if possible) way?

5.1.2.2 State of the art, available tools

Within the testing domain only a few relevant standards are available. The European Telecommunications Standards Institute has an established test standard with TTCN-3 (Testing and Test Control Notation, see www.ttcn-3.org [5]). TTCN-3 is widely used within the telecommunication domain. Currently there are much more domains using TTCN-3 as technology for testing activities. In [6] the current application areas of TTCN-3 are stated. In particular TTCN-3 is well suited to de-scribe and specify distributed test systems which are necessary for the testing of distributed test objects such as automation systems within VC. TTCN-3 is however also a very complex programming language, therefore TTCN-3 is not widely used within industrial applications.

Beside TTCN-3 there is another testing standard, the UML Testing Profile (UTP, [7]). Currently this standard is not well established, but some concepts are well useable for test specification and documentation. This has to be evaluated within AVANTI.

TTCN-3 Tools (see also http://www.ttcn-3.org/index.php/tools, some with UTP support)

- OPENTTCN (http://www.openttcn.com/)
- TTCN-3 Toolbox (http://www.devoteam.de/en/themen/testing/ttcn-test-solutions-and-beyond/)
- TESTCAST (http://www.elvior.com/testcast/introduction)
- TTWORKBENCH (http://www.testingtech.com/)

Generally within testing departments own, specialized testing tools are used and developed. These testing tools are developed and created with the special requirements and demands of the end users in mind. Thereby script languages like Ruby (https://www.ruby-lang.org/) and/or Python (http://www.python.org) are used. For the maintenance and extension of own testing tools and infrastructure an high effort is necessary. Beside this the using of own testing notations make it more difficult to reuse specified test suites and test cases.

Within the VC domain there are no well-known commercial and/or academic tools for the automated testing. Currently the manual test execution based on test specifications with office documents is realized within this domain. In AVANTI this state should be improved in order to realize a more automated testing process.

5.2 References

5.2.1 List of conferences covering relevant topics

5.2.2 Relevant publications

- [1] Ibrahim K. El-Far and James A. Whittaker. *Model-Based Software Testing*, in: Encyclopedia of Software Engineering, Wiley, 2001
- [2] R.G Dromey. *Formalizing the Transition from Requirements to Design*, in: Mathematical Frameworks for Component Software – Models for Analysis and Synthesis, He, Liu (Eds.), World Scientific Series on Component-Based Development, pp. 156-187, 2006.
- [3] M. Veanes, C. Campbell, W. Grieskamp, W. Schulte, N. Tillmann and L. Nachmanson. *Model-Based Testing of Object-Oriented Reactive Systems with Spec Explorer*. Microsoft Research, 2007.
- [4] Krause, J.: Testfallgenerierung aus modellbasierten Systemspezifikationen auf der Basis von Petrinetzentfaltungen. 2011, Shaker Verlag
- [5] European Telecommunications Standards Institute: Testing and Test Control Notation v3 (TTCN 3), 2009. [Online]. Available: http://www.ttcn3.org/.
- [6] ETSI: „TTCN-3 Application Areas," 2011. [Online]. Available: http://www.ttcn-3.org/ApplicationAreas.htm.
- [7] Object Management Group. *UML Testing Profile.* 2007. http://www.omg.org/technology/documents/formal/test_profile.htm/.

5.2.3 Relevant products or links to company statements

- www.conformiq.com

5.2.4 Latest version of relevant specifications, RFCs, standardization activities

- www.ttcn-3.org/index.php/downloads/standards

6 CAX process chain and interfaces

6.1 Description of state-of-the-art and latest solutions, relevance to AVANTI

6.1.1 Introduction

This chapter describes state of the art practices and latest solutions for engineering of automated assembly systems in automotive industries. *Virtual Engineering* and *Virtual Commissioning* as part of the workflow are explained.

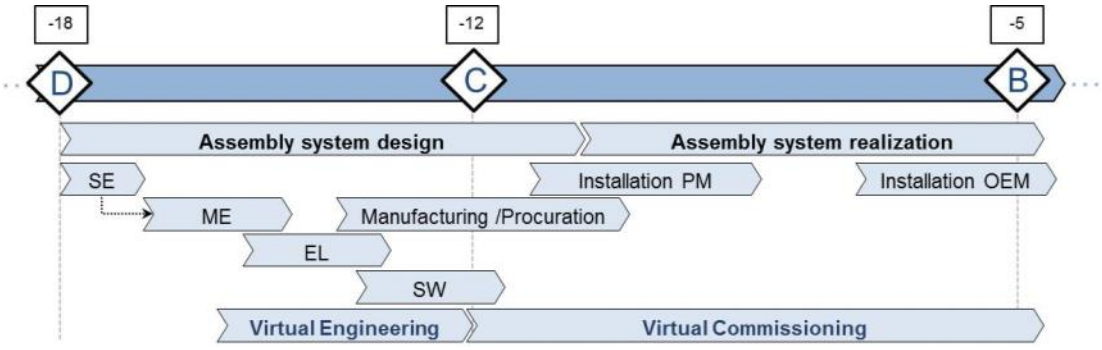

Figure 15: Development process of automated assembly systems

The engineering process of a highly automated body-shop system is complex, but during the last years, modularizetion, standardization and modern engineering tools have led to an improvement of the whole process, see Figure 15. In comparison to body-shop systems, automated assembly systems are characterized by high complexity, high product variety, and low standardization of their components. As a consequence the development process is unique for each single automated assembly system. In addition, the assembly system is not being conceptualized and designed by solely one vendor. Rather many collaborating partners and subcontractors are involved in order to provide a ready-to-operate assembly system to the Original Equipment Manufacturers (OEM's) shop floor. Based on interviews with leading experts of global operating plant manufacturers the assembly system development process was adumbrated and analyzed [1].

State of the art development processes of automated production systems in automotive industry consist of several stages, quality gates, and milestones, which are predominantly traversed sequentially. Average entire process duration is one year at a rough estimate, from acceptance of tender to finally encounter normal year volume of production. Based on the product specifications, pre-process-planning, and location-specific boundary conditions the OEM defines requirements for the automated assembly system and calls for bids from the plant manufacturers. Subsequently plant manufacturers and their subcontractors create a rough design, estimate costs, and submit a quote. After the tender process the OEM seals the deal with the selected plant manufacturer and proceeds to kick-off the development process.

At quality gate D of vehicle development process, 18 months prior to start of production (SOP), the plant manufacturer starts the assembly design phase (see Figure 16). Designing an automated assembly system can be considered as the design of a complex mechatronic product, starting with a systems engineering (SE) approach. Despite mechanical (ME) and electrical (EL) component design, software (SW) engineering is an essential part of the development process. All three design-threads are interdependent and cannot be treated separately. Thus, concurrent engineering approaches need to be applied in early process stage. Each stage may vary in length due to manufacturer-specific standards and business processes as well as assembly system complexity. Essential milestones in the early stages are conceptual design approval, mechanical design approval, and release for manufacturing. Subsequent milestones ensure proper installation of the assembly system at the plant manufacturer's production site, release for shipment to the OEM's shop floor, several test runs, and ultimately final acceptance of the assembly system on the OEM's part.

In mechanical and electrical component design many different digital tools (CAD, CAE, ECAD, etc.) are used in order to increase level of maturity and design quality early in the development process. Particularly, from conceptual design approval till functional software approval many important properties and parameters of the assembly system are defined and validated. This period offers many opportunities for impacting directly the quality of the final product and will be described in detail.

Developing and validating automated assembly systems using digital tools and methods increases quality and efficiency in ramp-up processes significantly. Both methodologies, Virtual Engineering and Virtual Commissioning, are state of the art for developing and validating body-shop production systems.

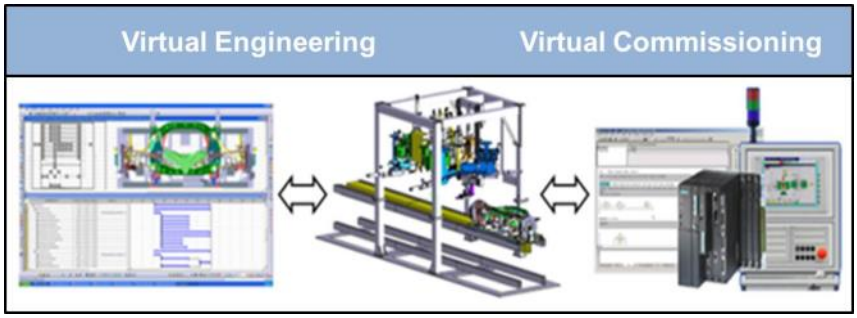

Figure 16: Virtual Engineering and Virtual Commissioning

6.1.2 Virtual Engineering

The methodology Virtual Engineering (VE) utilizes a kinematic model of the production system for visualization and simulation-based validation of system processes, cycle- time and collision for different product-variants [2].

Virtual Engineering of a production system includes the following tasks:

- Mechanical design and validation of the 3d-cad-model of the plant depending on the kinematics of the drivers (i.e. CATIA V5)
- Definition of process sequences and mechanical validation of assembly system and process (i.e. DELMIA V5, see Figure 17)

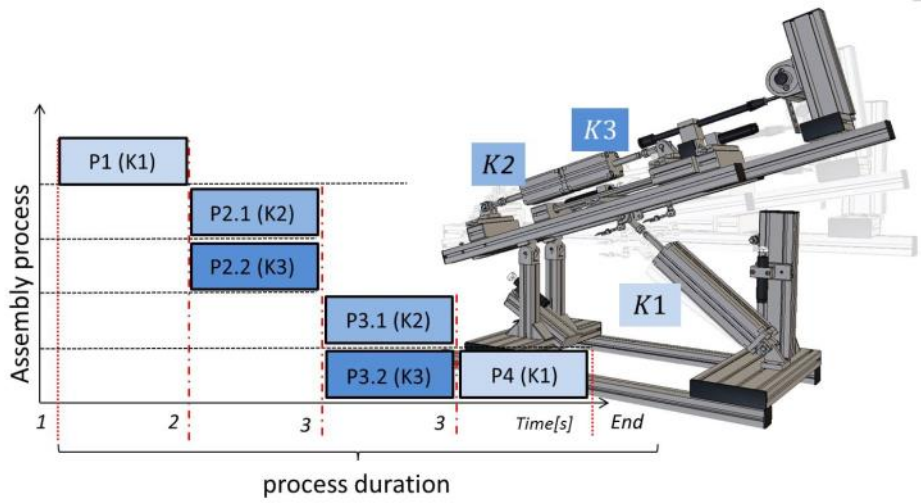

Figure 17: The process (left) and the 3d-cad-model (right) of an assembly system with three driver (K1-3)

6.1.3 Virtual Commissioning

Based on a mechatronic system model the methodology *Virtual Commissioning* aims at validating the system's control software by using a real Programmable Logic Controller (PLC) following the Hardware-in-the-loop (HiL) simulation approach. VC permits precocious evaluation, optimization and validation of the entire production system, particularly integration of product, production system's mechanical and electrical components as well as control software [6]. The goal of the validation is to create an environment that mimics the real automation hardware. The ultimate goal of emulation is to provide an environment for the

manufacturing automation controls engineer to validate their PLC (Programmable Logic Controller) ladder logic and HMI (Human-Machine Interface) files prior to system debug in the plant environment therefore improving quality and enabling a seamless transition from the virtual to physical environment. Another benefit is to deliver plant maintenance operators and machine conductors with realistic virtual environments for training themselves in safe and optimum conditions.

Virtual Commissioning of a production system includes the following tasks:

- Development of control programs (PLC and robotics) and HMI
- Definition of behavior models of components and of whole system (i.e. WinMOD)
- Coupling of control programs, behavior models and visualization in real-time (i.e. RF::Suite)
- Validation of the mechatronic assembly system (i.e. WinMod & RF::Suite)

6.2 RF::Suite

RF ::Suite is a package of several modular software tools that are used individually or in combination for the VC of robots or full system simulations, see Figure 18. The individual applications are perfectly matched and selected depending on project requirements. RF ::Suite enables that manufacturing cells already during the planning can be converted to virtual systems. The functions and processes can be investigated, hedged and improved. The data exchange can be offline and online (inter-process communication) and becomes realized with standards. Other applications can be easily integrated in functions of the RF ::Suite. RF ::Suite is successfully used by the project team of Rücker EKS and recognized partners in the automotive industry. A continuous improvement process ensures the continuous improvement of the software.

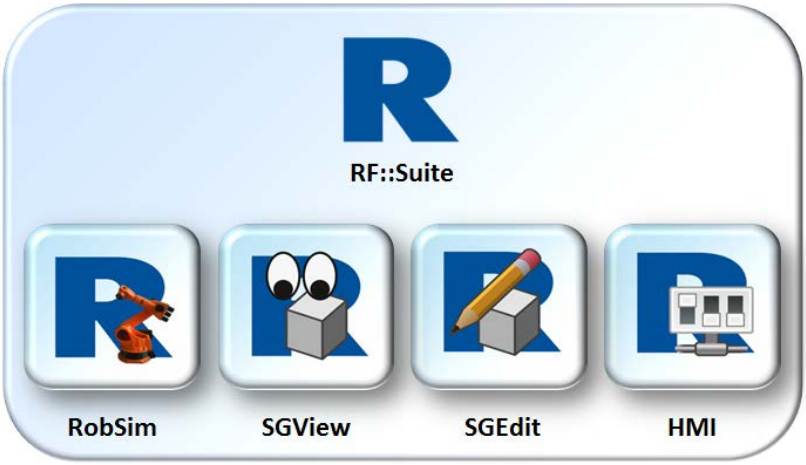

Figure 18: RF::Suite

6.2.1 RobSim

RobSim is applied to run and debug robot programs (online or offline). RobSim evaluates not only individual behavior of the robot, but also interactions between each of them, as well as interaction with the devices and controls of the periphery. This allows customization and robot simulation for all well-known industrial robots. In addition, RobSim supplies compilers for different robot programming languages, e.g. KRC for KuKa robots. The specific path planning is possible by means of plug-in. Other than real controllers, RobSim does not need complete robot programs and is able to work without a real robot. The robot program can be

tested completely with the focus of pointing out to the user all incorrect commands. Missing parts of the program can be added manually. Furthermore, RobSim is adapted to the interface of robot programmers, what enables convenient and detailed analysis of robot programs.

6.2.2 SGView

SGView is easy to use program for the visualization and animation of 3D geometric models. With SGView, either individual robots as well as complete production lines can be presented and therefore processes and concepts can be proofed and optimized before the real plant is established. With the help of Excel assistance complete material flow can be created in SGView and then simulated with associated signals.

6.2.3 HMI

HMI visualizes the online data exchange between different robot programs. Its focus is on the signals that are in reality exchanged between the robot and peripheral controllers or in the simulation between simulation programs. With HMI it is possible to integrate missing signals manually. This allows the examination of components, even if the PLC or the simulation is not yet available.

6.2.4 SGEdit

SGEdit is a program where it is possible to adjust the data structure of the 3D geometry models. SGEdit examines and edits the structure (hierarchical) of 3D geometric models. For easier presentation, not needed models can be removed and missing kinematics can be conveniently added as well. SGEdit is used in industrial applications for rapid correction of inconsistent data.

6.2.5 Interfaces between different applications

6.2.5.1 Visualization format jt

JT is a 3D data format developed by Siemens PLM Software (formerly UGS Corp.) and is used for product visualization, collaboration, and cad data exchange. It can contain any combination of approximate (faceted) data, exact boundary representation surfaces (NURBS), Product and Manufacturing Information (PMI), and Metadata (textual attributes) either exported from the native cad system or inserted by a product data management (PDM) system. It is probably the most widely used 3d visualization format for discrete manufacturing with over 4,000,100 JT-enabled licenses of software in use.

6.2.5.2 AutomationML

AutomationML (Automation Markup Language) is a neutral data format based on XML for the storage and exchange of plant engineering information, which is provided as open standard. Goal of AutomationML is to interconnect the heterogeneous tool landscape of modern engineering tools in their different disciplines, e.g. mechanical plant engineering, electrical design, HMI development, PLC, robot control.

AutomationML describes real plant components as objects encapsulating different aspects. An object can consist out of other sub-objects, and can itself be part of a bigger composition. It can describe a screw, a claw, a robot or a complete manufacturing cell in different levels of detail. Typical objects in plant automation comprise information about topology, geometry, kinematics and logic, where logic comprises sequencing, behaviour and control.

AutomationML incorporates different standards through strongly typed links across the formats:

1. Topology implemented with CAEX (IEC 62424)

Properties and relations of objects in their hierarchical structure

2. Geometry implemented with COLLADA of the Khronos Group

Graphical attributes and 3D information

3. Kinematics implemented with COLLADA

Connections and dependencies among objects to support motion planning

4. Logic implemented with PLCopen XML

Sequences of actions, internal behavior of objects and I/O connections

For future extensions, AutomationML is designed to integrate further formats using the same referencing mechanism.

6.2.6 Requirements towards future process chain

Nowadays, based on 3d-cad-models, it is possible to perform a so-called clash analysis, which allows the detection of collisions between two or more parts. Unfortunately by now it is not possible to monitor the impact of this collision to the colliding parts. To achieve this, the pertinent parts and especially their dynamics would have to be simulated during a collision. However, this is only necessary if the collision is a desired one and therefore is needed to perform a process in the assembly system. As an example, Figure 19 shows the collision between a moving piston and a shock absorber.

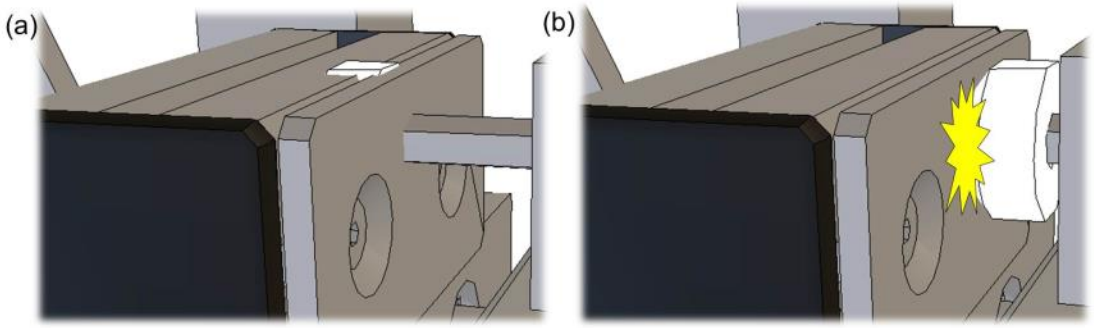

Figure 19: An example of collision between a moving part with (b) and without (a) a dynamic simulation

The needed time to prepare the simulation model for a dynamic simulation should be very short or in the best case zero. One possibility to achieve this is by integrating and fitting this type of simulation into the process chain.

6.3 References

6.3.1 Relevant publications

[1] Drescher, B., Stich, P., Kiefer, J., Strahilov, A. & Bär, T. Physikbasierte Simulation im Anlagenentstehungsprozess – Einsatzpotentiale bei der Entwicklung automatisierter Montageanlagen im Automobilbau. In: Simulation in Produktion und Logistik. HNI-Verlagsschriftenreihe, 2013. p. 271 – 282.

[2] Walla, W., Kiefer, J. Potentiale im Sonderanlagenbau am Beispiel der PKW-Hochzeit, In: 8. Fachkongress Digitale Fabrik@Produktion, Regensburg, 2012.

6.3.2 Relevant projects

- **Meda:** Due to continuously decreasing design times and increasing need for production flexibility new challenges arise in the process of designing production systems. In order to meet these changes research is being done to establish a design process which is based around mechatronic principles and ideas and supported by software tools supporting the use of mechatronic components. Key requirement for a mechatronic production system design process is the integration of all involved software tools by means of a common data exchange system, which guarantees lossless transfer of knowledge and design decisions throughout the entire design process.

The establishment of such a common data exchange format, based on existing industrial standards, was the main goal of the MEDA project. Using this data format should allow representing all information relevant to the design process. Additionally an interface engine has been developed which can be used by software tool vendors to export their product data to the newly developed exchange format and import any desired information from it. As part of this project a prototype will be developed to allow exchanging production process design data between tarakos' factory planning tool taraVRbuilder and process control visualization tool taraVRcontrol.

http://www.forschung-sachsen-anhalt.de/index.php3?option=projektanzeige&pid=14475

- **EVIE:** Entwicklung & prototypische Umsetzung eines Werkzeuges zur virtuellen Inbetriebnahme in allen Phasen des Entwurfsprozesses durch Kombination von Visualisierungssystemen & Verhaltensmodellierung - Modellintegration (EVIE - Modell)

The field of research in this project was production system design processes and design tools with relations to VC. Project goal was the creation of a software tool to facilitate Virtual Commissioning of production systems thought various stages of the design process. To achieve this an information model which integrated visual and behavioral information, a methods for visualization and behavioral simulation and a usage model have been created and implemented as an extension to existing software software tools like tarakos' taraVRcontrol.

The models and tools created help SMBs to efficiently widen the scope of their products by enabling easy integration with existing solutions established in the production system design process and so improve their situation in a competitive market.

http://www.forschung-sachsen-anhalt.de/index.php3?option=projektanzeige&pid=15723

7 Terms used

Abbreviation	Explanation
CoE	Cost of Energy
DIN	Deutsches Institut für Normung e.V.
DKE	Deutsche Kommission Elektrotechnik Elektronik Informationstechnik im DIN und VDE
EDD	Electronic Device Description
EDDL	Electronic Device Description Language
IEC	International Electrotechnical Commission
IT	Information Technology
MBT	Model Based Testing
OWL	Web Ontology Language
PLC	Programmable Logic Controller
RDF	Resource Description Framework
RDFS	RDF Vocabulary Description Language
URI	Uniform Resource Identifier
VC	Virtual Commissioning
XML	eXtensible Markup Language