Integrated engineering based on Modelica

Andreas Hofmann, Nils Menager, Issam Belhaj, Lars Mikelsons

Angaben zur Veröffentlichung / Publication details:

Hofmann, Andreas, Nils Menager, Issam Belhaj, and Lars Mikelsons. 2015. "Integrated engineering based on Modelica." In *Proceedings of the 11th International Modelica Conference, Versailles, France, September 21-23, 2015*, edited by P. Fritzson, 893–901. Linköping: Linköping University Electronic Press. https://doi.org/10.3384/ecp15118893.

Integrated Engineering based on Modelica

Andreas Hofmann[1] Nils Menager[1] Issam Belhaj[2] Lars Mikelsons[1]

[1]Bosch Rexroth AG, Germany, {andreas.hofmann7,nils.menager, lars.mikelsons}@boschrexroth.de
[2]Dassault Systèmes, France, Issam.Belhaj@3ds.com

Abstract

The academic society claims the use of virtual engineering (i.e. simulation) since many years. Nevertheless, it is de facto rarely ever used in the automation industry. This paper presents an approach and a toolchain for an integrated, digital engineering workflow including virtual commissioning, shown at a real industrial example. In particular, a new method for virtual commissioning that allows to drop all real-time requirements is presented.

Cyber-physical production systems rising with the concepts of industry 4.0 have a complexity that conventional engineering methods cannot bear. Therefore, the time has come to finally use model-based systems engineering methodologies that were proposed years ago, e.g. (Verein Deutscher Ingnieure (2004)). Nevertheless, the automation industry acts very conservative towards new technology. This is mainly due to the distrust that model-based methods can be used in an economic manner. Within the development cycle in the automation industry CAD models are used, since they save costs compared to construction by hand. During other stages of the development cycle, virtual models are considered to be of little or no use, since the effort for modeling those images of real systems is assumed to excel the benefits. This prejudice can only be overcome by lowering the effort for modeling or increasing the value of generated models.

In this paper models generated in early development phases are re-utilized within later stages of the development cycle, like application engineering and commissioning. The re-use of models for virtual commissioning is in particular possible due to coupling of a Rexroth PLC and a (possibly non real-time) Modelica simulation using a new Modelica library. In order to obtain an development cycle that is as integrated as possible, transitions between different phases in the development cycle are tackled. First, starting with CAD data it is shown how to automatically generate a physical representation of a machinery in Modelica. Using the physical interfaces of Modelica the model can easily be extended by drive models from component manufacturers. In combination with Bosch Rexroth PLCs, a transition towards the commissioning phase without further adaptions (e.g. complexity reduction for real-time application) is possible employing a new Modelica library. To show the entire potential of an integrated engineering workflow based on Modelica, an approach for creating control code based on a Modelica model of the control algorithm is given. By demonstrating those methods in the industrial application example of a bottling machine, it is disclosed that the assumptions of a high effort for creating simulation models, as mentioned introductory, can be disproved.
Keywords: integrated engineering, virtual commissioning, code generation, RFLP

1 Introduction

1.1 Motivation

Industry 4.0 is the central topic in automation industry. Controlled plants are replaced by highly automated, networking and self-regulating cyber-physical systems. Conventional development methods do not match for this complexity. Instead, those new rising challenges need to be faced by new product development methods.

Model-based System Engineering is something very natural. Before building a machine or more general a technical system in nearly every case a model is set up. However, in the automation industry in many cases this model is a mind model rather than a virtual model. Clearly, most of the benefits that apply for virtual models also hold for mind models, but to a lower extend. Thus the benefits of simulation (low cost experiments, available at any time, ...) are well known and highly valued, but the effort for virtual modeling is estimated higher than the saving in time.

Within a typical development cycle in the automation industry, simulation models are used for the 3D design. However, during the dimensioning of suitable components simulation is utilized only sometimes. During the control design of the complete system virtual models are hardly ever used. The reason for using models for the

design is quite obvious; using a CAD software clearly saves a lot of time compared to construction with pencil and paper. Nevertheless, a dynamic model helps to avoid over-dimensioning and thus may save costs. Last but not least usually it is not worth the effort setting up a model for control design.

Within the automation industry simulation will only earn its space if the benefits, i.e. the savings in time at last, outperforms the effort that is required for creating virtual images of the technical system. Currently, the effort for modeling is felt as very high, because every role in the development cycle models start from scratch. Models from other engineering phases are not re-used, sometimes even models from other engineering disciplines are ignored. Nevertheless, in order to successfully develop those cyber-physical systems that are rising as mentioned initially, an interaction of the different physical domains is required.

By extensive recycling of simulation models the cost for developing those virtual images can be reduced significantly. Utilizing Modelica as modeling language renders this re-use possible. Models can not only be transferred and used between different domains easily. Modelica also, since it is a describing language that needs a compiler to generate executable code anyway, can be used as basis for code generation for hardware targets like PLCs. Through its open interfaces it is also possible to include external libraries that provide features further than for dimensioning of a machinery. The models can also be re-used for virtual commissioning in coupling with real industry controls.

By integrating Modelica in the model-based system engineering methods, like the RFLP approach, a distinct improvement towards willingness to simulate of the automation industry can be made. Dassault Systèmes 3DEXPERIENCE platform (3DXP), see (Dassault Systèmes (2015)), that incorporates the RFLP approach in combination with Modelica, allows on the one hand to work with a clearly structured cross-domain development process and on the other hand renders the re-use of simulation models and the extend of model-based engineering towards the stages of application engineering and virtual commissioning possible.

1.2 Outline

In the following section a short overview about the RFLP approach is given. Although this method alone is not suitable lowering the perceived effort for creating virtual models, using Modelica as modeling language allows to extend the benefits, especially for automation industry. Section 3 focuses on the area of code generation for PLCs based on Modelica models. After a general definition of code generation is given and possible fields of application are discussed, a toolchain is presented, which can be used to execute arbitrary C/C++ code on a Rexroth PLC. The subsequent chapter addresses the idea of virtual commissioning. Basic concepts are described and a new approach with focus on model-based system engineering is given. In section 5, the previously described technologies are combined in an example. Starting from CAD, the approach of integrated model-based engineering is shown for a bottling machine. In the last chapter the integrated engineering methodology is summarized and the paper is closed with an outlook about further development.

2 The RFLP Approach

During the process of product development, nowadays, several disciplines interact with each other. Since it is difficult to manage such concurrent multidisciplinary engineering processes it is necessary to provide products that meet the customer requirements or to create a structured development process in order to integrate all the disciplines and specialty groups into a coherent team effort. The RFLP approach, c.f. (Kleiner and Kramer (2013)) (**R**equirements engineering, **F**unctional design, **L**ogical design, **P**hysical design) as system engineering process based on the V-cycle design process, see Figure 1, permits to simplify matters by defining a system based on its fundamental aspects through essential views and their relations.

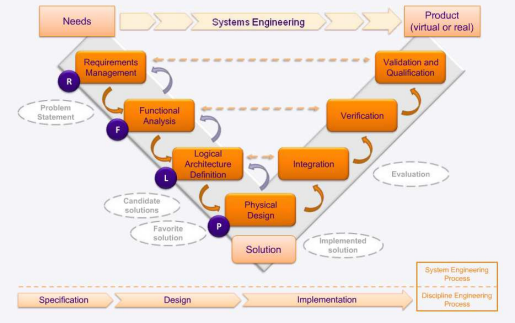

Figure 1. The steps of RFLP within the V-cycle design process.

Facilitating cross discipline communication between customers, different engineering departments, partners and suppliers, the RFLP approach provides a common view to all. RFLP ensures traceability and provides decision support in a highly collaborative environment. Furthermore, this methodology ensures that the final products meet the customers requirements in a cost effective, timely and qualitatively efficient way.

Within the **requirement engineering** customer and stakeholder needs are defined. Characteristics and activities the system has to satisfy are concentrated. Hence, the

DOI 10.3384/ecp15118893

approach permits following the life cycle of each requirement and validating each step of the engineering process. Additionally, requirements that contain physical quantities can be directly linked to those models of functional design, in which their implementation is intended.

In the **functional design** phase all capabilities of the technical system are decomposed into elementary functions. The intention of the technical system is formalized into a structure that can be allocated to technical solutions. Behavior models can be part of the functional design defining how components transform inputs to outputs. Thereby, a validation of requirements is rendered possible.

From the functional decomposition the logical architecture of the technical system is derived in the **logical design** phase. Different technological solutions corresponding to required functions are analyzed and compared during this step. Each technical solution is based on subsystems, components and their interfaces according to the technical requirements, customer needs and expected functions. Like in the functional design phase, behavior models can also be embedded into each logical component. The logical design is the main structure during the conceptional phase. Based on modular definitions, all different engineering disciplines are brought together.

Within the **physical design**, an entire virtual representation of the real technical system is modeled. The logical components are expanded by the dynamic characteristics. Mechanical models are for instance established with CAD, wires can be included, electrical drive behavior imaged or hydraulic flow represented. Eventually, the behavior of the holistic, complete cross-domain technical system can be verified against the requirements, that were defined during the requirement engineering.

2.1 3DEXPERIENCE Platform

Although RFLP can support to keep the product development cycle of a technical system clearly arranged, the limitations of virtual models that were initially introduced still remain. Furthermore, the re-use of models especially in the logical and physical phase can be difficult, if no elementary interfaces are given.

Dassault Systèmes incorporates the RFLP approach within their Business experience platform 3DXP. Since this tool provides all the different steps of product development, the interface issue between the different steps of RFLP vanishes. Furthermore, 3DXP uses Modelica for the logical and physical modeling of the technical system. This allows on the one hand to re-use the created connections from the logical model to the physical model and on the other hand an integration of different domains can be easily performed, since Modelica is the most sophisticated modeling language for modeling complex cross-domain physical models.

2.2 Extending 3DXP towards an holistic model-based engineering

Apart from the previously stated advantages of RFLP over the product development process, the integration of Modelica offers further possibilities. As described at the beginning, effort for creating virtual models hinders many customers from automation industry to use those methods. However, this effort can be significantly reduced by expanding the use of virtual models and by reutilizing simulation models for this enhancements. Especially in the automation industry, where the area of application engineering and commissioning are one main focus during development, simulation offers vast benefits.

3 Code generation

This section deals with code generation as a characteristic of model-based engineering. In the first part, the definition and important fields of application are discussed. After that, a toolchain to use code generation with Rexroth industrial controllers is presented.

3.1 Definition and fields of application

To realize an integrated model-based development, code generation out of simulation models is one important key feature. Code generation allows the transfer of knowledge from one development phase into following ones, e.g. between the system design and the commissioning. This technique makes it possible to re-use information, which is already available during the design phase and generally stored in a simulation model, during the commissioning. Generated code can be used for various fields of application.

The most common one is Rapid Control Prototyping. Nowadays, new technical systems are, in a first step, mostly designed virtually using modeling and simulation. Therefore, a simulation model of the system is set up inside a simulation environment. To investigate the dynamical behaviour of the plant, a control algorithm is added inside the simulation environment. After the system is completely designed and tested virtually, these models are in general not used any further. Instead, the control algorithm is re-implemented from scratch using PLC programming languages. Besides the fact, that a re-implementation needs additional time and therefore causes costs, it is always a potential error source. These disadvantages can be avoided, if the already existing controller inside the simulation model is re-used. This can

be realized by generating code out of the model and executing it on the PLC.

Besides the use of controller models on the PLC, it is also possible to use plant models on the PLC. One field of application is model-based diagnosis. The plant model is simulated in parallel to the operation of the machine. The model is used to calculate the expected dynamical behaviour of the machine, while the actual behaviour is gathered using sensors. As soon as differences between the calculated and measured values occur, this may refer to an error inside the machine. In case, that special error models are available, even the specific error type might be determined.

Code generation out of plant models can also be used for modern control strategies like Model Predictive Control. Model Predictive Control solves an optimal control problem (dynamic optimization problem) in every cycle online on the controller. The differential equations of the system, which are included in the model, are used as constraints of the optimization problem. The solution of the problem is an optimal input on the system in the next time step and minimizes a user-defined cost function, which includes the control aim.

Modelica models are perfectly suitable for code generation. As Modelica is a describing language, a compiler is needed to generate executable code. If the commercial Dymola compiler is used, C code is generated. In case of the OpenModelica compiler, both, C and C++ code, can be selected. While the Dymola code generation is a black box and can therefore not be modified, the code generation inside the OpenModelica compiler is template-based and can be easily adapted. This allows to generate specific code even for different hardware targets.

3.2 Toolchain for code generation using the OpenModelica compiler

Bosch Rexroth has an own code generation module inside the OpenModelica compiler, which generates C++ code. Using a flag, it is possible to generate code for specific industrial PLCs (e.g. IndraControl XM22). In comparison to an offline simulation, several functions of the application programming interface (API) of the controller have to be integrated into the code. The generated code contains only the model. Additional information, how the occuring model equations should be solved, is not included in the model. Therefore, a simulation runtime is necessary. This runtime includes the numerical integration methods and manages the entire simulation, e.g. handles occuring events. Bosch Rexroth developes also a runtime, written in C++, which supports the simulation of models generated from OpenModelica.

For the execution of Modelica models on industrial controllers, a toolchain is available, see (Menager et al. (2015)). This toolchain uses the OpenModelica compiler, which generates C++ code as described before. To run this code on the hardware, the code has to be compiled for the operating system on the controller. Bosch Rexroth industrial controllers use VxWorks as real-time operating system. Hence, to execute the code, a VxWorks compiler is needed. In this toolchain, the WindRiver VxWorks compiler is used to compile both, the model code and the simulation runtime, into a library. Using the Motion Logic Programming Interface (MLPI), which is used as interface to the controller, the code can be simply connected to an existing PLC application. The integration is realized with a function block, which offers the inputs and outputs for the data exchange between the executed code and the PLC. Additional information about MLPI can be found in (Engels and Gabler (2012)).

Of course, not only code generated from the OpenModelica compiler can be executed on the PLC. In general, any C or C++ code can be run on the hardware. This includes the C code, which is generated from Dymola. However, this code has to be modified manually to implement the necessary interfaces of the controller's application programming interface.

4 Virtual Commissioning

4.1 Fundamentals of Virtual Commissioning

Up to 25% of the project period of a plant and therefore a big share of the costs are needed for commissioning. Especially troubleshooting of controller application software dominates this part of the engineering process, see Figure 2.

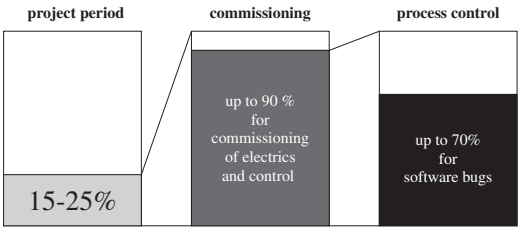

Figure 2. Amount of commissioning time on product development period, based on (VDW (1997))

Although software is a central issue already, commissioning times will most likely increase with Industry 4.0. The rise of connected, highly automated and cyber-physical systems will increase the number of software parts within a plant and thus have significant impact on commissioning times. This is, on the one hand, based on growing contents of software in mechanical and automation industry as well as horizontal and vertical networking, cf. (BMBF, 2013). On the other hand increas-

DOI 10.3384/ecp15118893

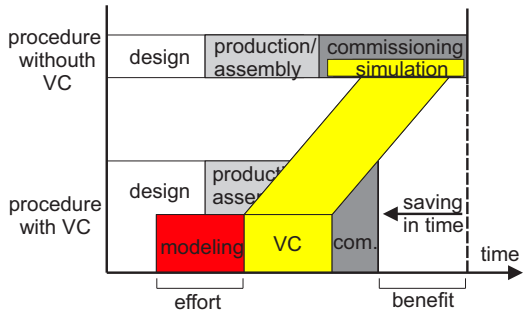

Figure 3. The basic idea of virtual commissioning, based on (Wünsch (2008))

ing complexity of software projects will also lead to rising commissioning times. The method of virtual commissioning can help reduce commissioning times significantly.

In this context virtual commissioning (VC) is understood as a method for testing functionality respectively operation sequence of a plant with the assistance of a model of the system. Basic idea of the VC is the coupling of the simulation model to a real or virtual control. Using simulation, the operability of the control application can be checked in early stages of the development. Software errors can be eliminated during production and assembly of the plant and time as well as expenses can be saved during the real commissioning, see Figure 3.

Furthermore, VC assists to increase the quality of the control application, c.f. (Wünsch (2008)). However, in order to obtain valid results the VC approach has to secure that the deterministic and consistent behavior of the PLC is represented.

4.2 Interaction between Simulation and Control

In the area of automation industry VC is typically used in the context of coupling a simulation model of the plant to a virtual or real PLC. Hence, the two approaches Hardware-in-the-Loop (HiL) and Software-in-the-Loop (SiL) are common.

Software-in-the-Loop describes the coupling of a virtual model of the plant with a virtual image of the control. Since there is no real hardware control within the setup, all models can be simulated on the same computer and no real-time requirements prevail. It follows that arbitrary complex models can be used for this kind of VC and especially models from previous engineering steps are suitable. However, this setup does not ensure deterministic execution of the control application. A full check of the system behavior is not possible.

Hardware-in-the-Loop is a VC approach in which a real PLC is coupled with a simulation model of the plant. Therefore a real-time bus and a real-time operating system is required. Of course the simulation model also has to satisfy this demands. Hence, models from other stages of the product development process are not suitable. The body of acquired knowledge in form of the simulation model needs to be discarded and a new virtual representation of the plant is required. Though, in contrast to SiL, determinism is provided, since the real control is part of the infrastructure.

In the context of an integrated engineering approach as stated preliminary, none of the prevalent methods is eligible. Either the validity of the simulation is not sufficient or simulation models from previous engineering steps can generally not be used.

4.3 Extending virtual commissioning towards MBSE

Typical coupling strategies are very limited in a simulation based development approach. In order to use VC within a model-based system engineering approach, both benefits of SiL and HiL need to be joined. For Bosch Rexroth PLCs this can be achieved using their OpenCore Interface. Bosch Rexroth OpenCore Interface, c.f. (Bosch Rexroth (2015)), is a universal port with direct access to the control and motion kernel of Rexroth industrial controls. With this technology applications can be written, using high level languages like Java or C++, that allow to join drive and control systems and conventional IT environments, see (Engels and Gabler (2012)). For the purpose of virtual commissioning the OpenCore interface is implemented in Modelica in the library mlpi4Modelica whic allows to access the control within a simulation.

Virtual commissioning with Bosch Rexroth PLCs is based on HiL abrogating the real-time requirements. This is achieved using the OpenCore technology which can interact with the motion kernel of the control. This allows to set the control in some simulation mode and tasks that are triggered by the motion cycle event, managed by the internal clock of the PLC, are no longer executed cyclically. Instead they are activated by an external trigger signal from the simulation and are launched, precisely once, during the next motion cycle. Thus the real-time requirements repeal and the complex infrastructure with real-time operating system and real-time bus can be replaced by a general simulation pc and a common ethernet connection. However, the consistent and deterministic behavior of the control is secured. Triggering the control from the simulation allows to utilize arbitrary complex models of the plant, which are in general not real-time capable.

4.4 Modelica library mlpi4Modelica

The library mlpi4Modelica provides functionality to directly access Bosch Rexroth PLCs and read as well as change symbol variables or parameters within the PLC program. The library itself is divided into three parts.

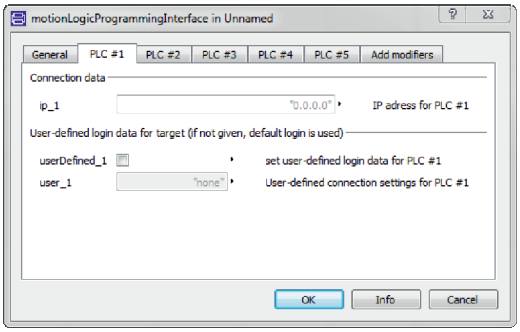

Figure 4. Parameter window of the MotionLogicProgrammingInterface model

The first part is a component called MotionLogicProgrammingInterface, which creates the connection to the PLC, c.f. Figure 4 over ethernet connection and thus is mandatory. Currently up to five industry controls can be connected. However, this limitation is due to Modelicas GUI-performance and more controls could be easily added by extending the component.

The second part of the mlpi4Modelica library is represented by the model mlpiCoupler and the package IOCoupler4Modelica. The former serves as a gate, which uses the connection data from MotionLogicProgrammingInterface, and converts the signals into valid information for the PLC. It also translates the Modelica datatypes to the required datatypes on the control if possible. Furthermore, the mlpiCoupler enables/disables the simulation mode of the control and regulates the trigger signal. The package IOCoupler4Modelica includes an analog and a digital interface model, that can be connected to the mlpiCoupler. Within those, for every input and output, the parameter name or symbol variable name on the control respectively the PLC program is defined. Using the modelica language element connectorSizing allows on the one hand to have not linked inputs and on the other hand to have multiple interface models connected to the mlpiCoupler, c.f. Figure 5. As a consequence, arbitrary signals can be exchanged with the PLC.

The last portion of the library is the Library package, which contains all functions of the OpenCore Interface that are required for simulation purpose. Those functions, originally written in the language C, are wrapped within Modelica and allow to write own models that can access the control or the PLC application. In addition to this, when using a C/C++ code generation, this renders

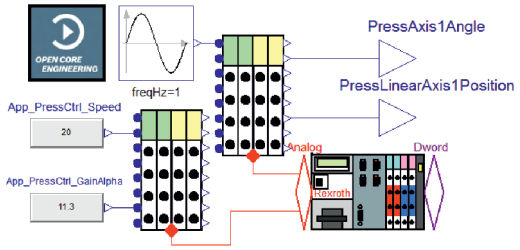

Figure 5. Example model with use of the mlpiCoupler and two interface models.

creating of controller modules or even PLC application directly from the model possible as described in Section 3. It is also depicted in the following section.

5 Application Example

Bottling machines, see Figure 6, serve as good example for performing the previously described steps and extensions.

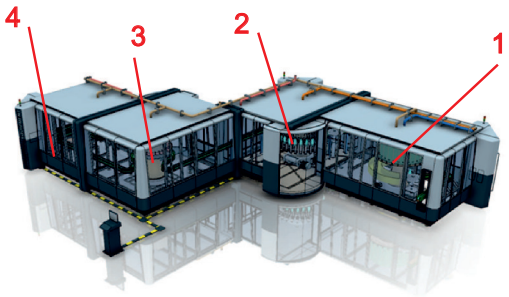

Figure 6. Image of the complete bottling machine, consisting of a rinsing machine (1), a filling machine (2), a capping system (3) and a labeling machine (4).

In this contribution the model-based system engineering approach is shown for the filling part. However, all steps are also suitable for the other machines since the methodology is, as stated before, of general purpose. The challenge in this kind of machinery is to have the bottle infeed system, the rotary filler and the bottle outfeed synchronized. Since every part has its own driving motor, the synchronization has to be performed by the PLC. However, since each motor has its own position and velocity control, the interaction has to be further investigated in order to eliminate errors due to contouring errors.

5.1 Development of the bottling machine model

The generation of dynamic models of the machinery is one big task, that hinders the use of simulation models. Within 3DXP this problem can be eliminated, at least for the mechanical part. Having the CAD data of the technical system available in 3DXP, a mechanical Modelica

DOI 10.3384/ecp15118893

model (joints, bodies, e.c.) based on the library CATIA-MultiBody can be generated, see Figure 7.

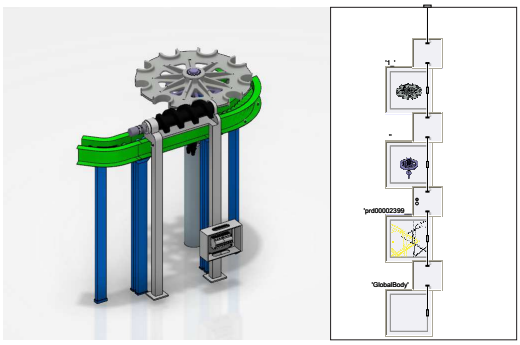

Figure 7. 3D model of the infeed system (left) and the generated Modelica model (right).

Driven joints within the 3D model are represented by joints with flanges. Therefore, a simple extension of this mechanical model is possible.

For the filling machine, as described here, the driven joints are connected to motor models from Bosch Rexroth, that contain motor specific characteristics, like torque and motor speed limitations and fine interpolation that is done by a motor controller.

The whole setup of the dynamic model of the filling machine can be seen in Figure 8.

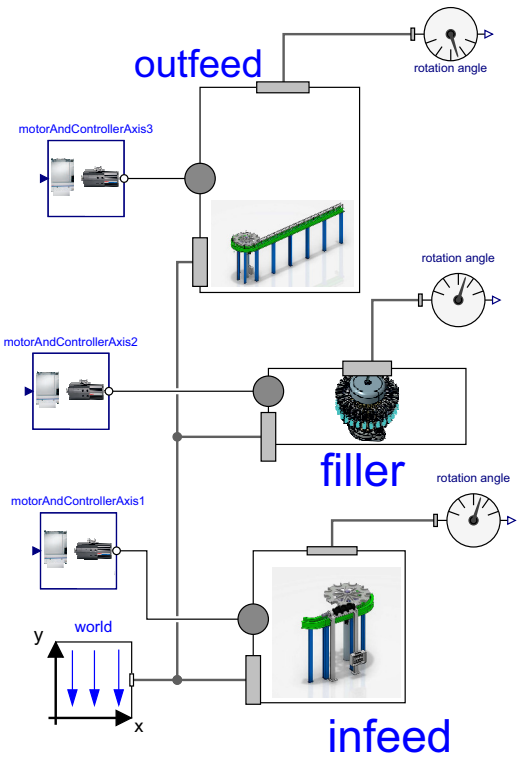

Figure 8. Dynamic model of the filling machine, divided into the mechanical model derived from CAD and the Rexroth drive models.

One can clearly see, that the motor inputs are not connected, at this point. For now, the model needs to be tested with synthetic stimuli. For the virtual commissioning of the complete system later on, it will be coupled to the real nominal axis values of the PLC and therefore receive real values.

5.2 Control design for the bottling machine

The control algorithm for the functionality of the bottling machine can be held simple. It consists basically of the states Initial, Manual, Synchronization, Automatic and Stop, see Figure 9, and was implemented in Modelica using the new State Machines language elements, (Modelica Association, 2014), (Elmqvist et al., 2012).

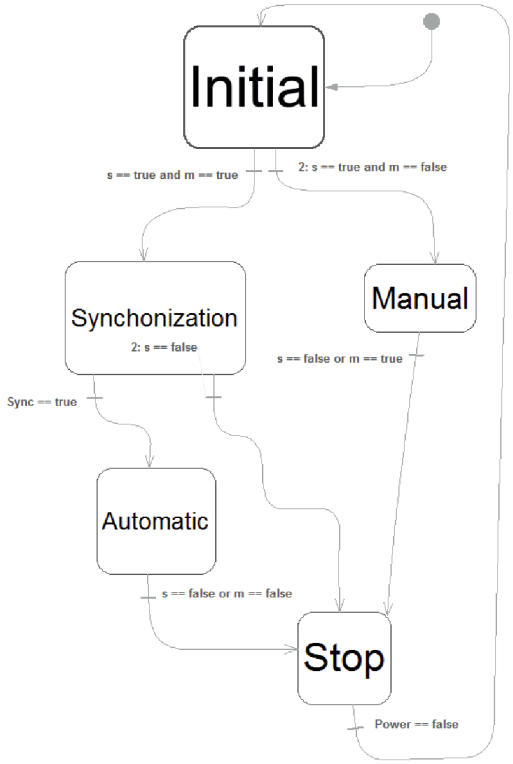

Figure 9. State machine of the bottling machine.

The transitions between the individual states are switched depending on the values of the boolean variables *start* and *automaticMode*.

State Initial
This state is active during boot up of the machinery and stays active until a transition either to State Manual (start == true && automaticMode == false) or State Synchronization (start == true && automaticMode == true) applies. Within this state, all three axis (infeed, filler and outfeed) of the filling machine keep in a resting position.

State Stop
Whenever the machine is running, either in state Manual, Automatic or Synchronization and a change to

any other running state or State Stop is applied, the State Stop is reached. All running axes are shut down. Switchover from State Manual (automaticMode == true || start == false) or State Synchronization (automaticMode == true || start == false) does shut down all axes individually. Transition from State Automatic (automaticMode == false || start == false) decelerates all shafts synchronously. After all axis remain in a resting position, the state automatically changes to state Initial.

State Manual
Within state Manual, all three axes of the bottle filling machine can be individually moved, depending on the values of boolean variables *axisInfeed*, *axisFiller* and *axisOutfeed* in combination with a real variable *manualSpeed*. This can be useful for manual positioning or testing of desired motor speeds.

State Synchronization and State Automatic
If the State changes from Initial to Synchronization, all Axis are moved to a synchronized position and the active state changes to State Automatic. Within the latter, all axis are accelerated in sync to a desired speed. This state represents the normal production of bottles.

All movements of axes that have been described previously are implemented using the Modelica library mlpi4Modelica. As a result, the complete state machine model can be used directly as a model for the real PLC. After generating C-Code from a model the compiled code can be transferred to the PLC as object program. In order to change values on the variables, e.g. manualSpeed, some functions block are created, see 3.

5.3 Virtual commissioning of the plant

In order to validate the behavior of the previously described controller algorithm and to examine the contouring error, that might inhibit a valid synchronization, the dynamic model is coupled to the real PLC. The interconnection is realized by the components of the library mlpi4Modelica, as described in section 4.

The output values from the mlpiCoupler are the axis nominal values that are calculated internally by the PLC. Those values are input for the internal interpolation of the drive models (motor and motor controller) from Bosch Rexroth. Their output torque drives the different mechanical parts of the filling machine. For synchronization the current angle of each axis is transferred to the PLC, see Figure 10.

After fully parametrizing the drive controller, the contouring error stays below a permitted deviation and the synchronization is not inhibited, c.f. Figure 11. Also the controller application module that was generated from the state machine model is working properly.

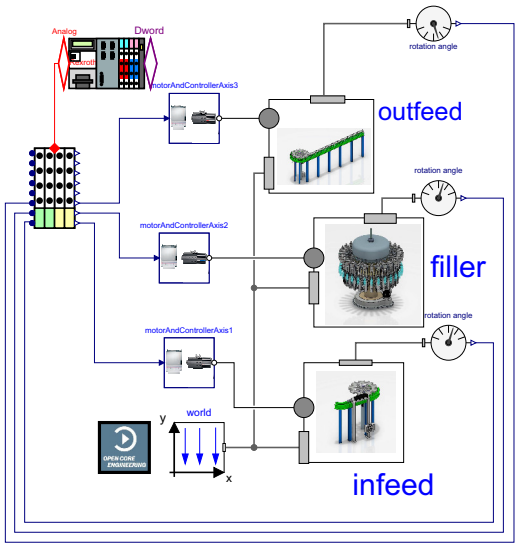

Figure 10. The complete dynamic model of the filling machine during virtual commissioning phase.

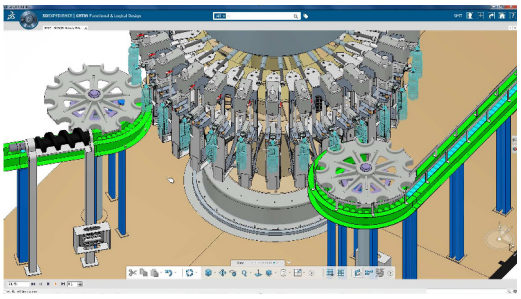

Figure 11. Filling machine in synchronized motion.

6 Conclusion and Outlook

The re-use of simulation models offers huge potentials. Utilizing those models in multiple phases of the product development cycle enhances productivity and reduces costs as well as time. Of course, the effort for creating the virtual images of a technical system remains. However, compared to the additional benefits that are available at no cost, this effort loses significance. Using Modelica as modeling language within the RFLP approach renders those enhancements of model use over the whole development cycle possible. In this contribution, code generation is used to create controller modules or whole PLC applications from simulation models. Furthermore, using the simulation model of the plant instead of the real machinery, the commissioning can be performed virtually without having produced any part yet. This is demonstrated using a bottling machine. Starting from the requirements and functional model of the plant, the steps of dynamic model generation, control code development from simulation model and virtual commissioning are described.

The benefit of simulation models does not reach the end of the line with commissioning. Since control code

DOI 10.3384/ecp15118893

can be generated from the Modelica model, it is also possible to run the simulation model on the controller during production. This enables features like model-based diagnosis or model predictive control. Also the generation of complex robot transformations which are elaborately derived by hand, can be automated using simulation models and Modelica.

References

BMBF. *Umsetzungsempfehlung für das Zukunftsprojekt Industrie 4.0*. German Feder Ministry of Education and Research, 2013.

Bosch Rexroth. Engineering Network: community for software developers. `http://www.boschrexroth.com/network`, 2015. Accessed 19-May-2015.

Dassault Systèmes. 3DEXPERIENCE platform. `http://www.3ds.com/about-3ds/3dexperience-platform/`, 2015. Accessed 19-May-2015.

Hilding Elmqvist, Fabien Gaucher, Sven Erik Mattsson, and Francois Dupont. State machines in modelica. *Proceedings of 9th International Modelica Conference*, 2012.

Elmar Engels and Thomas Gabler. *Universelle Programmierschnittstelle für Motion-Logic Systeme : Struktur, Funktionen und Anwendung in der Forschung und Lehre*. Tagungsband AALE 2012, 2012.

Sven Kleiner and Christoph Kramer. Model Based Design with Systems Engineering Based on RFLP Using V6. *Proceedings of the 23rd CIRP Design Conferenc*, 2013.

Nils Menager, Lars Mikelsons, and Niklas Worschech. Model-based engineering using Rexroth controllers and open standards. *Tagungsband Mechatronik 2015*, 2015.

Modelica Association. Modelica language specification 3.3 revision 1, 2014.

VDW. *VDW-Bericht: Abteilungsübergreifende Projektierung komplexer Maschinen und Anlagen*. WZL, 1997.

Verein Deutscher Ingnieure. VDI 2206: Entwicklungsmethodik für mechatronische Systeme, 2004.

Georg Wünsch. *Methoden für virtuelle Inbetriebnahme automatisierter Produktionssysteme*. Herbert Utz Verlag, 2008.