# Model based Virtual Startup of Automation Systems

Uwe Schob (uwe.schob@iwu.fraunhofer.de), Ralf Böttcher,
Fraunhofer IWU,
Reichenhainer Straße 88, 09126 Chemnitz

Torsten Blochwitz, Olaf Oelsner, ITI GmbH Dresden
Marek Winter, USK Karl Utz Sondermaschinenbau GmbH Limbach-Oberfrohna

## Abstract

This paper deals with the model generation for automation systems. A generic solution is presented, which analyses existing Computer Aided Engineering (CAE) documents and thereof automatically generates simulation models. They are described as Modelica models and use a special library, customized for the automation branch. It contains common elements which can be parameterized with very few efforts. Additional components for the communication to the control are available. They support several widely accepted technologies like OPC, Profibus, Profinet, analog and digital signals.

The generation process focuses on being as automated as possible. It is a general approach, which could be applied to various types of simulation with different levels of detail. Its integration into a software prototype shows the feasibility and provides first practical feedback.

*Keywords:*

*Automation; Model Generation; Machine-simulation; Modeling; CAE-analysis; Virtual Startup*

## 1 Introduction

Global product diversification increases the demand for more flexible and thus more complex manufacturing plants. Reducing their time to market is of utmost priority in order to remain competitive. Developing and deploying highly automated machines is a most demanding task requiring specialists from different domains. Plans for the mechanical construction, the electrical diagrams as well as control programs are created under deadline pressure followed by the assembly and startup phase.

Several trends arose to meet the above mentioned requirements. The introduction of high level paradigms to restructure the development process itself may be considered as the most future-oriented solution as seen in [1, 2, 3]. Software tools implementing these ideas are based on unified data models to simplify the inter-domain associations [4, 5]. Another method of supporting the machine development is the extensive use of simulation to improve the design quality. Simulations may be applied to verify early design decisions and help finding flaws [6].

## 2 Motivation

A common practice in plant development is simulating the machines or parts thereof. Such virtual machines are used to verify control programs and thus increase their overall quality in an early stage. This method is called virtual startup and may already be performed without real machine hardware, leading to a reduction of the time needed for the real machine deployment as stated by Wünsch [7].

This assumption is only true, if the time needed to create simulation models does not surpass the expected gain at a later phase. Especially in the field of special purpose machines, where only few repetitive systems are constructed, the task of model generation is too time consuming. This is also stressed by Bergert in [8]. The task of creating the machine model is regarded as additional work to the normal development and creates error prone solutions.

As stated above, the idea of virtual startup lacks optimizations allowing its application to broader areas. To meet the problem of the time consuming model generation task, one has to be reminded that most of the required information already exists. It is contained in the construction and manufacturing plans, e.g. Computer Aided Design (CAD) drawings and circuit diagrams, which are available in digitalized form. They are stored and managed in various CAE, Enterprise Resource Planning and office applications.



DOI: 10.3384/ecp09430120

One feasible solution is to use readily available interfaces of the domain-specific applications as data access. They provide the data in digitalized and reusable form. As stated in [9], the interoperability between software systems is a matter of designing appropriate adapters, which translate one set of data elements into another one.

# 3 State of the art

## 3.1 Machine development

The process of developing a machine or a whole plant consists of several phases as seen in Fig. 1. These are mainly, the mechanical construction, the electrical, pneumatic and/or hydraulic design as well as control programming. According to actual guidelines, the phases should be performed in a parallel way to reduce overall-time and inter-domain errors [1]. Despite the obvious advantages, the idea of a holistic mechatronic design is far away from being common knowledge or a widely accepted method. Each domain-specialist performs his respective tasks for himself with his favored software tools. His results remain independent unless they are communicated to other engineers, often in an informal manner. The outcome is a set of different electronic CAE-documents, for example CAD-drawings, circuit diagrams, pneumatic plans as well as control programs, whose association is as good as the previous inter-domain communication was.

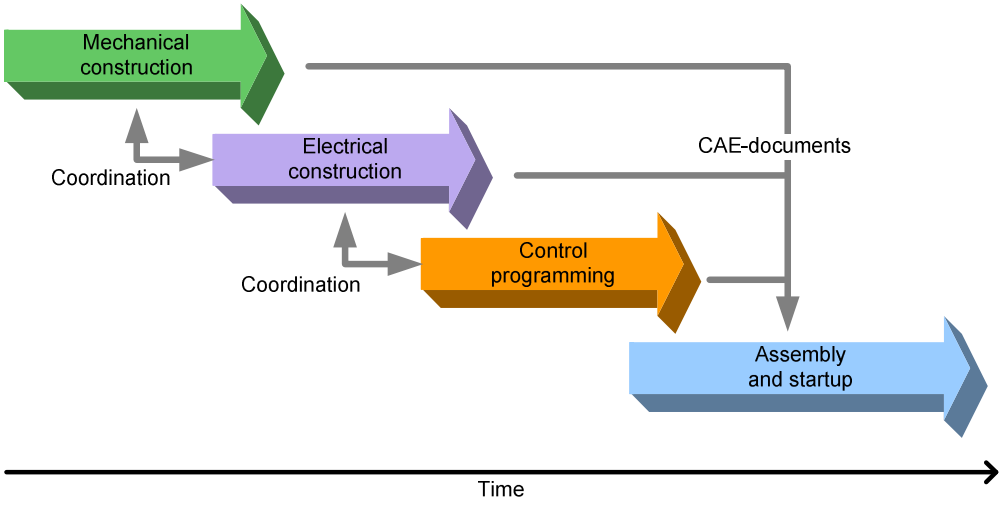


**Figure 1:** Machine development process

## 3.2 Assembly and startup

By using the generated assembly documents, the commissioning of the required components and products begins. Depending on the product delivery times, the machine is being assembled more or less in time. As soon as the first functional parts are assembled and electrical connected, its startup begins. Design flaws, wiring mistakes, non-documented changes and even programming mistakes mostly manifest in the startup phase. Beginning at this step, the domain specific decisions are forced to work together and thus often highlight inconsistencies. A missing sensor for process-control might lead to changes in the mechanical structure of the machine as well as its wiring and the control program. Most of all, these late corrections take a lot more time than they would have required during the design-time.

## 3.3 Simulation

A commonly accepted method to reduce design flaws and inter-domain problems is to simulate the desired system or parts thereof. Depending on envisaged results, different kinds of simulation are used. They range from finite-elements-method to test mechanical strain over multi-body-systems to calculate kinematic behavior of geometric bodies and up to behavior-simulation of whole machines. Simulation may take place as soon, as the first specifications are available. Based on parts of the assembly documents, simulation models may be defined. Results are used for dimensioning machine parts or to verify its functionality.

## 3.4 Virtual startup

A behavioral model of a machine used for simulation is called a virtual machine. Virtual startup is the process of driving a virtual machine or plant model with real control hardware, as defined by [8]. Cur-

rently available commercial tools like WinMOD or Virtuos [10, 11] emphasize the importance of early control program tests as their main goal. One of the main hindrances of a wide acceptance is the additional effort of creating the simulation models [12]. Repeating machine parts, the heavy use of predefined library components and modification of existing simulation models allow savings in the modeling effort.

The remaining effort is still considerably and requires engineers, which are familiar with all machine-aspects expected within the simulation results. This counts especially in the field of special purpose machines, where the repeated construction of the same machine is more than unusual.

Unfortunately, no currently available tool allows a sophisticated reuse of CAE-documents from all design phases as a base for simulation models.

# 4 Solution

## 4.1 Starting point

The software tools currently used for documenting the different design phases are manifold. They do not share a common data depository nor are they able to export their content into a universal data format. However, most of them include application programming interfaces (API), which provide a limited access to their internal data structures [13, 14]. This can be used to algorithmically acquire a subset of the available information, which is contained in the CAE-documents. Each design domain provides a different set of information.

The mechanical construction defines, of which parts a machine is composed and how they are interconnected. It focuses on the geometric design of bodies. Additionally, their assembly hierarchy is defined by various constraints.

The electrical construction determines, by which means electrical energy is transformed into useful energy. Actuators manipulate the work piece and sensors provide feedback about it. Every machine component that needs electrical energy has to be documented in circuit diagrams.

Similar to the electric are the pneumatic and hydraulic constructions. They also transform source energy into useful energy. Only the sources, pressurized air respective pressurized fluid, are different. The resulting diagrams and schematics contain all machine parts, which are connected by tubes.

By relating these data sources to each other, the major part of the machine behavior can be estimated. Nevertheless, these data source are not sufficient to form a complete simulation model. Some associations are often undocumented, as they are trivial noticeable with a human understanding of the machine. For instance, the relation of a linear drive in a circuit diagram and its geometric representation in a CAD-drawing is algorithmically not obvious. Even the positioning of sensors within the machine is seldom to be found in CAE-documents. In order to being able to generate a meaningful simulation model, additional data needs to be supplied manually.

Relating the different data sources takes place as part of a data analysis resulting in groups of logical connected objects. Each of them is then to be translated into a behavioral identical simulation element. By this translation, simulation models representing the supplied data sources are created. Fig. 2 shows this translation process in a simplified form.

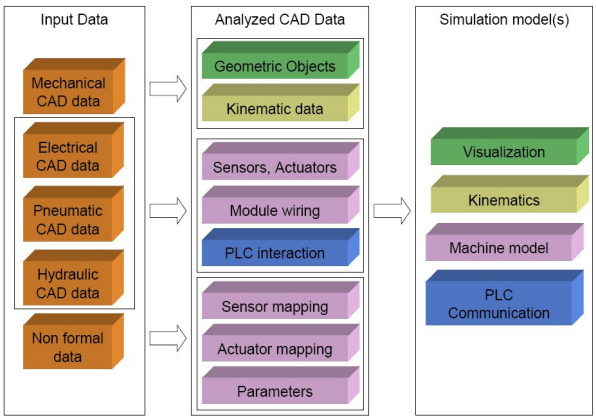


**Figure 2:** Transforming data sources

## 4.2 Transformation process

Looking into the transformation process in more detail reveals four separate steps as shown in Fig. 3. Their starting point are the different data sources, which are preferably supplied automatically. Partial information needs to be fed manually.

1) The data acquisition targets the available CAE-documents. A program module uses the APIs of each involved software tool to read out the raw information, a user has previously entered during his normal workflow. The output represents the actual state of the machine being developed.
2) During the mapping, the raw information parts are analyzed and grouped based on their resource mark, defined by [15]. Through them, logical connected units can be recognized. For instance, an electric relay in the circuit diagrams is represented by a conductor and one or more switches which all share a similar resource mark.

3) The manual input is a necessary step which requires information by the user. The informal data exchanged through the inter-domain coordination, as shown in Fig. 1, has to be formalized. A functional decomposition of the machine is provided as the desired machine development state. This will be supplemented by simulation-parameters not otherwise available. Additional associations are created between actuators and their respective geometric object, thus allowing a 3D-visualization during simulation based on the CAD-drawings.
4) The final step in creating a simulation model is the consistency check. It is performed based on the nominal and the actual machine development state. A comparison shows common mistakes such as:
   a) specified but not yet realized machine functions
   b) designed but unspecified functions
   c) wiring and tubing changes not consistent with the specification
   d) possible communication / wiring problems between control program and connected actuators and sensors.

   The results are reported to the user as recommendation. Updating the data sources in this development stage is optional regarding the generation of a simulation model. Nevertheless would a later error search be simplified, if obvious design flaws are removed.

In addition, the consistent logical elements are mapped to ready Modelica elements. By trying to apply rules of a larger set to an analyzed logical group, a corresponding Modelica library element can be found. The rule set to be used depends on the complexity of the logical group and the available element library. Different kinds of mapping rules may apply:

a) Exactly one circuit element is matched with exactly one simulation element.
b) Exactly one circuit element is matched with two or more simulation elements. This helps expressing more complex circuit elements through a combination of several simple simulation elements.
c) Two or more electric (or pneumatic) elements are mapped to a single simulation element.
d) A set of circuit elements is matched by another set of simulation elements.
e) Defining the rule set responsible for a mapping is a one-time task. It might be reused for following machine developments, as long as no additional unmapped electrical components exist in the CAE documents.

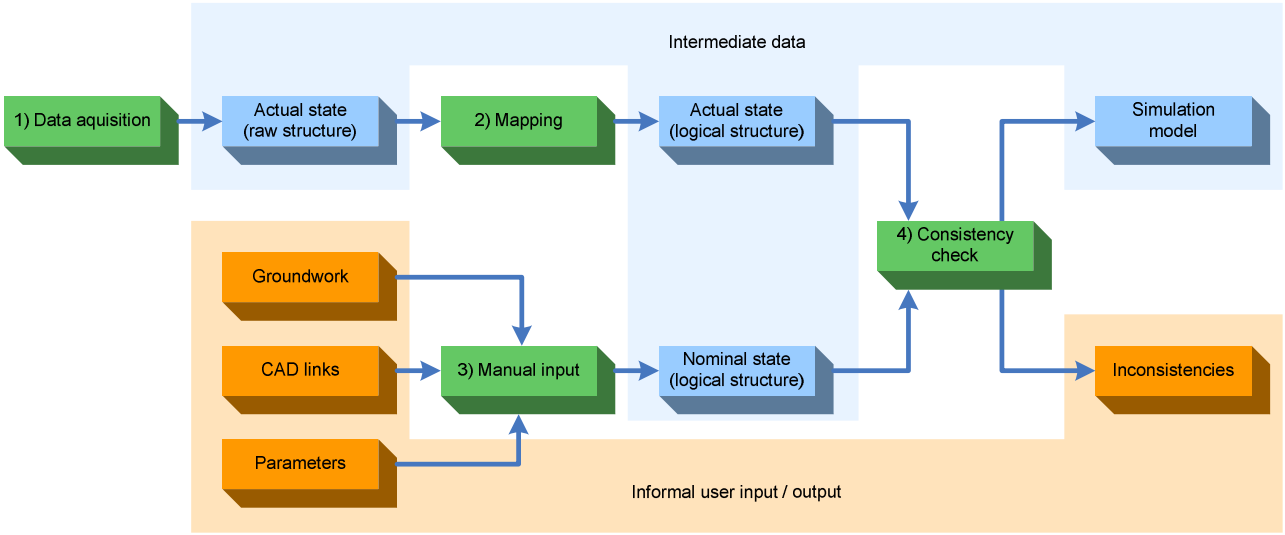


**Figure 3:** Transformation process in detail

### 4.3 Simulation usage

An important impact on the allowed simulation model complexity is the application purpose. The main goal of this work is the virtual startup of a machine by using its real control program. Although the programming of programmable logic controls (PLC) is standardized in [16] not all control vendors conform to it. Additionally, various hardware dependant features are proprietary, thus making it difficult to simulate the control itself in software. Based on this assumption, hardware-in-the-loop is the means of choice for the simulation usage. The interface be-

tween the virtual machine and the real control hardware are the PLCs input and output signals. An exemplary signal flow through a circuit diagram is shown in Fig. 4.

In order to feign a real machine to the control, the simulation has to meet several requirements:

1) All output signals written by the control need to be used within the simulation. Respective need all input signals read by the control to be provided as actual simulation outputs.
2) The simulation must not progress faster than the real control.
3) The complexity of the simulation model has to be chosen to be as time-efficient as possible.
4) The signal-transmission to and from the control must not influence the control program

Although the control hardware is a hard real-time system, the virtual startup itself does not strictly requires it. Of course, if the simulation is working in a soft real-time mode, its average calculation time should not exceed the controls cycle time. Exceeding the cycle time is critical for the validity of the simulation results only if the control is performing operations, depending on strictly timed feedback.

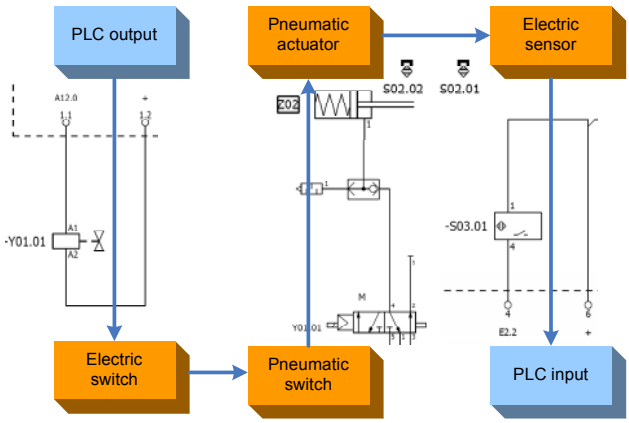


**Figure 4:** Signal flow

### 4.4 A customized Modelica library

Although various Modelica libraries exist, the above mentioned requirements of runtime optimized simulation elements need to be met. Respecting and modeling all physical effects is possible, but not necessarily relevant. A new library was created to incorporate machine devices modeled more behavioral than physical. It was named after and designed for the special needs of the automation branch. An excerpt is shown in Fig. 5.

The desired level of detail of each model element was chosen element-wise, depending on the devices real functionality. By initially creating all basic elements, such as relays and switches, more complex devices could be built upon them. A later described machine example was created through the normal development process and taken as reference for filling the automation library. It contains pneumatic and electric elements and is still being extended.

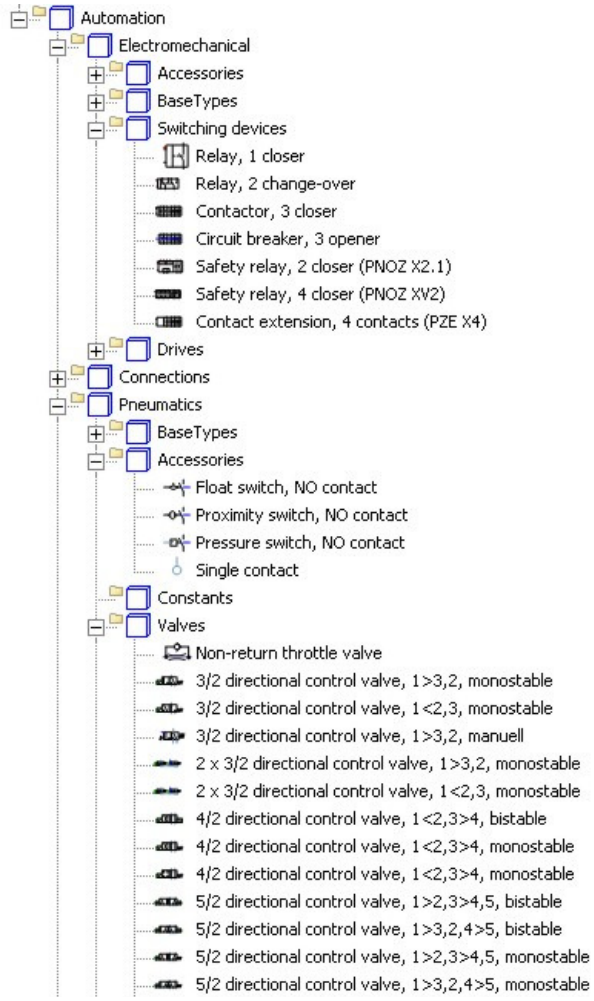


**Figure 5:** Overview of the automation library

## 5 Implementation

### 5.1 Connected tools

During a current research project, the mentioned resolution is being implemented. A difficult decision had to be made, which software tools are to be used. A market survey revealed several CAE-tools, whose features were roughly similar, concerning the domain specific development requirements. By examining the available API-functionality according to their flexibility and complexity, the choice felt on the most promising tools:

1) Autodesk Inventor as CAD software,
2) EPlan Electric P8 and its Fluid-addon as tool for designing circuit and pneumatic diagrams and

3) ITI SimulationX as modeling and simulation tool [17].

The missing connection between all these software tools was a program module, which managed the transformation process and its underlying data sources. In consequence, a standalone application (CADSIMA, Fig. 6) was developed, that was able to connect and handle the data flow from and to each API. In addition it can be used as coordination tool between the domain specific design steps, through which necessary informal data is acquired.

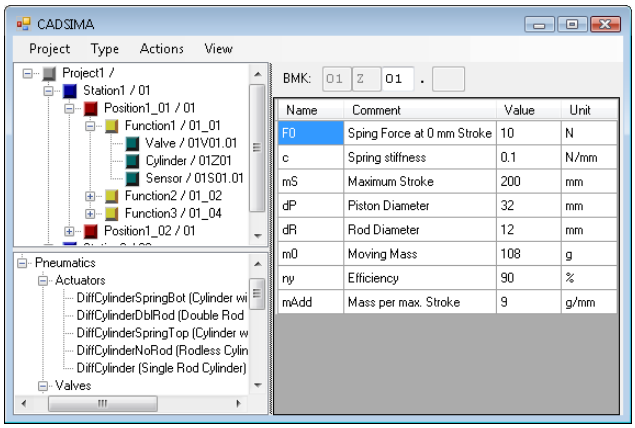


**Figure 6:** Screenshot of newly tool Cadsima

The nominal state can be entered by the user and is stored in a database. This allows the distributed access to the project data. CADSIMA creates a SimulationX model based on the results of the mapping and consistency checks.

## 5.2 Analysis and mapping example

The general data analysis and mapping into model elements have been implemented for the selected software tools. Their internal data structures showed a diversity of objects, whose properties need to be evaluated in detail during the analysis. The resulting mapping mechanism and corresponding rules are derived thereof and explained in the following paragraph.

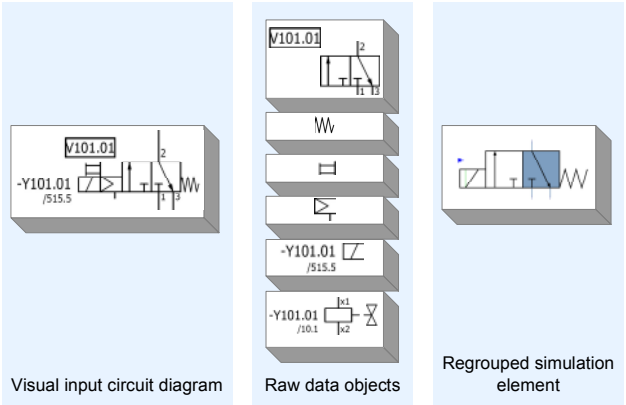


**Figure 7:** Mapping example

Fig. 7 shows the analysis and mapping of a pneumatic monostable 3/2-way valve. A visual inspection of the corresponding circuit diagram page displays one element with two visual resource marks. The internal data model reveals that it contains 5 separate objects:

1) the valve itself with resource mark V101.01,
2) a spring on the right side,
3) a manual reset at the left upper side,
4) a pneumatic trigger on the left side,
5) an electric trigger at left side with the different resource mark Y101.01 and
6) a reference to an inductor at another page, which represents the wired element.

A rule to find such valves looks for each of the 6 elements and if found, signals the creation of the corresponding simulation element. Wires and tubes are stored as connections between pins in the circuit diagrams. They are also mapped to pins of the simulated pneumatic valve.

## 5.3 A first use case

Current work focuses on providing the resources to transform small machines or parts thereof. The resources meant are appropriate library elements and defining rule sets for them. This work is a necessary step prior to transforming actual CAE documents. The reference project of a pick and place handling machine, as shown in Fig. 8, contains 3 separately controlled axes and a not displayed conveyer. Looking at the mechanical characteristic, it contains about 30 mechanical parts. Its circuit and pneumatic diagrams include on 87 pages roughly 170 articles represented by more than 1300 separate symbols. About 25% of these symbols are only for a display purpose and not actually wired to other components. Unfortunately, as worst case every remaining symbol needs to be transformed into a simulation element.

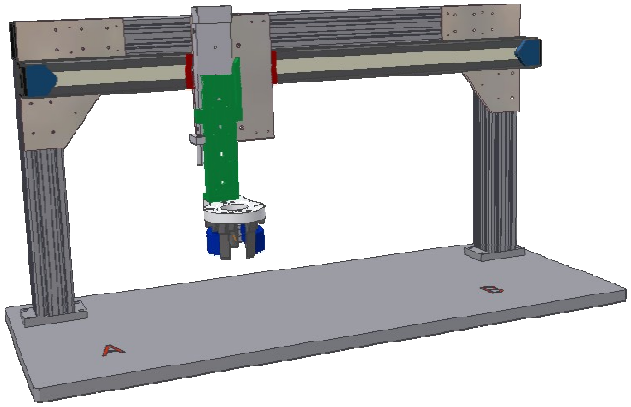

**Figure 8:** Handling machine for pick and place

### 5.4 Virtual startup

A setup for a virtual startup contains a real control, without additional peripheral equipment and a pc running the generated simulation model. The pc will communicate with the control hardware via OPC, overwrites the controls inputs and reads its outputs. No additional hardware as the above mentioned is required, although actual displays or human machine interfaces can be connected to the control hardware, if desired. The control program itself needs a minor change to accept the missing peripheral equipment. The programs write and read access is not influenced by it. Despite that, the program remains identical to the one used in the real machine.

During the simulation run, the control programmer is connected to the real control hardware as well. He performs the same startup steps as with a real machine. Its current state can be inspected through the simulator and its built-in signal-visualizations. Interactive changes are available through overwriting runtime parameters.

## 6 Conclusions and Outlook

This article described a method, which transforms an input data set, e.g. the circuit diagram of a manufacturing system, to an output data set, e.g. a simulation model of the same machine. It detailed the structure of the source data model, the components to which they will be transformed and finally, how a simulation run can be performed. The target model library consists of Modelica elements combined with communication components which are used for a hardware-in-the-loop simulation.

This approach drastically reduces the effort in creating machine models and thus enables even the special purpose machine manufacturer to use the virtual startup. A system test for validating the control program may be undertaken prior to assembly of the real machine.

Actual outcomes for the startup phase, e.g. higher program quality or shortened startup time, and for the machine development process as whole are outstanding. Hence, future works should focus on providing reliable feedback about the methods effectiveness in the field of application.

Experiments on measuring the numerical performance of larger machine models should be undertaken prior to extending the automation library to support more complex development projects. The usage of a faster communication protocol and appropriate hardware, e. g. Profibus or Profinet, would allow a better integration of feedback controlled systems working with smaller cycle times.

This work originated in the cooperation of the companies USK Karl Utz Sondermaschinenbau GmbH Limbach-Oberfrohna, ITI GmbH Dresden and the Fraunhofer Institute for Machine Tools and Forming Technology IWU during the joint research project “Depromes”, funded by the Sächsische Aufbaubank.

## References

[1] Verein Deutscher Ingenieure: VDI 2206 - Entwicklungsmethodik für mechatronische Systeme, Juni 2004.

[2] J. Bathelt: Entwicklungsmethodik für SPS-gesteuerte mechatronische Systeme, ETH Zürich, 2006.

[3] CADsys: FOD - Prozessketten übergreifende Produktkonfiguration, online, 2006.

[4] Verein deutscher Maschinen- und Anlagenbauer: Baukastenbasiertes Engineering mit Föderal, 2004.

[5] B. Grimm et al.: Universelles Datenaustauschformat, A&D Kompendium, 2008.

[6] M. Ehrenstraßer et al.: Virtuelle Werkzeugmaschinen für die Simulation, wt Werkstattstechnik online, Jahrgang 92 (2002)

[7] G. Wünsch: Methoden für die virtuelle Inbetriebnahme automatisierter Produktionsysteme, Technische Universität München, 2007.

[8] M. Bergert and C. Diedrich: Durchgängige Verhaltensmodellierung von Betriebsmitteln zur Erzeugung digitaler Simulationsmodelle von Fertigungssystemen, in Automation Kongress, 2008.

[9] U. Schob: Werkzeuge und Methoden zur mechatronischen Modellierung von Produktionsanlagen, Technische Universität Chemnitz, 2007

[10] Mewes & Partner GmbH: WinMOD, online, http://www.mewes-partner.de/www/eng/, 2009

[11] ISG Industrielle Steuerungstechnik GmbH: ISG-virtuos, online, http://www.isg-stuttgart.de/virtuos.html?&L=1, 2009

[12] G. Reinhart: Teilautomatisierter Aufbau von Simulationsmodellen, wt Werkstattstechnik online, Jahrgang 97 (2007)

[13] EPLAN Software & Service GmbH & Co. KG.: EPLAN API 1.0, application documentation, 2006

 

[14] Autodesk Inc.: Autodesk Inventor API, application documentation, 2004

[15] Deutsches Institut für Normung: DIN EN 61346-2 Industrielle Systeme, Anlagen und Ausrüstungen und Industrieprodukte - Strukturierungsprinzipien und Referenzkennzeichnung, 2000

[16] International Electronical Commission: IEC 61131-3 Programmable controllers - Part 3: Programming languages, 2003

[17] ITI GmbH: SimulationX, online, http://www.iti.de/cms/en/simulationx.html, 2009

 