Integrated Behavior Models for Factory Automation Systems[*]

Jewgenij Botaschanjan Benjamin Hummel
Institut für Informatik, Technische Universität München
Boltzmannstr. 3, 85748 Garching b. München, Germany
{botascha,hummelb}@in.tum.de

Thomas Hensel Alexander Lindworsky
Institut für Werkzeugmaschinen und Betriebswissenschaften, Technische Universität München
Boltzmannstr. 15, 85747 Garching b. München, Germany
{thomas.hensel,alexander.lindworsky}@iwb.tum.de

Abstract

Despite the large amount of models for different aspects of factory automation systems, many of these models target at individual and in most cases static aspects of the system, such as the geometry or its electric parts. There is a lack of suitable description methods, which integrate these individual models to a behavior model including spatial aspects and the handling of material. Furthermore, it is important that this model keeps the link to the more detailed individual models and is sufficiently formal in order to allow an automated analysis. This paper provides a solution to this problem by introducing a model which addresses both spatial structure and behavior and is based on a thorough mathematical theory. Complementary, we report on a tool realization of the modelling theory and explain how the model supports the development of mechatronic systems.

1. Introduction

In the the last decades machine manufacturers faced new challenges due to the increasing number and complexity of customer demanded functions as well as the raised share of PLC software in automation systems [28]. The substitution of purely mechanical operation principles by mechatronic solutions has led to a strong interdependence of the technical disciplines [17]. The question arises, how the development, validation, and testing of such complex systems can be supported. In the early phases of development, often no suitable system model is available, which could be used for validation. It is still common that the automation system is not even tested before the regular commissioning phase, when a comprehensive system test is usually not possible, since the work takes place under high time pressure. Furthermore, changes to the system are often extremely cost-intensive or even not possible at this stage. Above all, failure scenarios and erratic behavior are only examined rudimentarily due to the lack of time, missing specification of possible errors, and the risk of damaging the machine. Thus, errors are not identified and may appear during the operation phase [10].

Despite the mechatronic character of modern production systems the development processes are still focused on mechanical engineering, which has a leading position. The development process is built up in a sequential manner, with one discipline building upon the results of the previous one. Normally, the mechanical engineers are defining the functions of a machine at the beginning of the development process. Subsequently the machine is designed by the different departments, which complement the previous models by their own descriptions, such as electric wire plans or software code. However, using this sequential process misses many opportunities for cost reduction or improvement from interdisciplinary cooperation. Without a common model which can be understood and extended by engineers from all disciplines, the sequential process is hard to come by.

The envisioned common model would act as a link between engineering departments and capture all aspects of the machine (mechanics, electrics, controller), albeit in an abstract and simplified fashion. Then, design ideas can be explored and discussed with the customer more easily. By refining this model, by adding more details, and finally using it as a basis for discipline-specific models (*e. g.*, CAD models), synergies emerging from mechatronics can be exploited and the overall development process gains flexibility. Finally, such a model can serve as the basis for valida-

[*]This work is partially funded by the AIF e.V. and the German Ministry of Economics and Technology (BMWi) in the context of the AutoVIBN project.

tion and verification by observing its simulation and using it as a counterpart for software testing or stress estimations.

Contribution This paper introduces integrated behavior models, which combine static/structural, dynamic/behavioral, and spatial aspects of a (mechatronic) system into one common model. These models are sufficiently abstract to be created fast and early on, but still detailed enough in order to be executable and support early validation by simulation. We provide a detailed description of the model elements used, their interplay and interpretation, and the reasons for designing the model the way it is. Furthermore, we report on various development activities, which can be supported by the described model, including documentation purposes, early validation, tracing and checking consistency between other engineering models, as well as testing of the controller software. As a proof-of-concept, tool support for modeling and most of these activities has been built.

Outline The next section relates the model described here to existing work in this field and elaborates on the differences to the presented approach. Sec. 3 describes the model in detail and explains some consequences of design choices. In Sec. 4 different activities are sketched that can be supported by using the integrated models, and finally we conclude and provide an outlook on planned work in Sec. 5.

2. Related Work

Most modeling approaches for industrial automation concentrate on the control software aspect and abstract from the physical processes in the environment of controllers. From the methodological point of view such approaches consider only one fragment of mechatronic systems and fail at integrating the different involved disciplines. Also, for an early and reliable validation of control software, *e. g.*, by simulation, test-case generation, or HIL/SIL (hardware-/software-in-the-loop), the behavior of its physical environment has to be "somehow" mapped to observable sequences of input signals. The control-related approaches offer no guidance for this procedure. Next, we compare our approach with works, which aim at a integrated description of mechatronic systems.

Bonfé *et al.* [1] extends UML-RT in a way which allows the capsules to be also specified by bond graphs. By this, continuous mechanical and electrical issues can be captured within an object-oriented modular system description. However, every single capsule can be modeled either as a discrete (*e. g.*, by a Statechart) or as a continuous one (using bond graphs). This limitation does not exist in the presented approach. Also, no explicit modeling of material and material flow is provided by [1].

The language and tool *Modelica* [26] supports modeling of physical systems with continuous and discrete parts. The disadvantage of Modelica is the high level of detail needed for the system description: masses, torques of inertia, and friction coefficients are not known in the early development phases. The present approach supports different levels of abstraction; it allows describing individual components of the same system at different levels of detail.

Model Integrated Mechatronics (*MIM*) [24, 25] is a component-based model, in which a component consists of mechanical, software and resource parts. The behavior of each part is specified by a simulator. The tool Archimedes [24] is a prototypical realization of MIM. Unfortunately, no details are provided on how these simulators interact with each other. Furthermore, the lack of an explicit integrated model limits this approach to simulation and does not support more advanced analysis methods.

The *MEDEIA* project [23] aims at integrating different descriptions of mechatronic components as well as of hierarchical and structural layout of mechatronic systems. Thereby, the behavioral aspect is left unspecified and, consequently, the integration of different views and components to an overall system behavior remains an issue.

Petri-nets are often proposed (*cf.* [11, 29] and references therein) as a modeling formalism for mechatronic systems. High-level Petri-nets or Activity Diagrams allow concise and intuitive description of data and control flow between distributed constituents of a system and actors in its environment. However, to the well-known limitations of Petri-nets belongs the lack of composability (*cf.* Sec. 3.6 for an in-depth discussion of this topic). Thus, this formalism cannot be considered as an appropriate choice for modeling modern mechatronic systems.

The present work applies a special type of communicating hybrid I/O automata for modeling the behavior of mechatronic components. Hybrid automata seem to be the most natural choice since they allow to model both the continuous physical processes and discrete controller behaviors and incorporate the notion of composition. The general models of hybrid automata [13, 18] do not provide explicit modeling constructs for material flow and collisions, which constitute an essential behavioral aspect of mechatronic systems. For the modeling of mechatronic systems the *global* and *PLC automata* were proposed in [7] resp. [9]. Although, we were not able to find a trace set-characterization of global automata, they seem to be a generalized version of Henzinger's hybrid automata [13]. The relationship to Lynch's TIOA [18] is unclear. PLC automata provide an operational semantics for the Duration Calculus [27]. They are concerned with modeling of control software only. Both formalisms lack of explicit support for material flow and collisions – the first-class citizens of the present approach.

The existing tools and approaches for HIL/SIL simulation, also called virtual commissioning [8, 20, 21], do not consider the integration of their models and methods into the broader context of system development. This greatly enhances the effort of their application and produces redundancies in the project data. The present work aims at significantly reducing both the effort and redundancies.

3. Integrated Behavior Modeling

This section describes our approach for modeling the behavior of mechatronic or cyber physical systems in the domain of factory automation. It extends on earlier versions described in [3, 16]. We give an overview of the modeling elements used, their interaction, and their intuitive meaning. A more formal view on the semantics of a part of this model can be found in [15].

The behavioral aspects of the model are based on the FOCUS theory [6], which is an asynchronous stream-based semantical framework for formalizing reactive systems, and especially on its tool realization AutoFOCUS [5, 22]. The spatial description of the system are influenced both by 3D-CAD tools and ideas from spatio-temporal logics [12, 19].

3.1. Foundation

The model builds upon two fundamental notions: *types* and *space*. Types are used for internal computations and to describe signals exchanged between components. In this context a type has a name, a set of valid values (its *carrier set*), and a set of operations. More complex type systems (*e. g.*, including inheritance) can be mapped on this simple scheme. For practical purposes we usually limit the types to Boolean, numerical (integer and real values), enumeration, and tuple types, which seem to be sufficient in the context of industrial automation.

To describe the shape of a system, we introduce the notion of space, which consists of a description of valid volumes, operations for merging volumes and checking them for collision, and a set of valid transformations (see [15] for a formalization). For our abstract models we usually limit the volumes to those which can be constructed by simple geometric objects (cuboids, cylinders, spheres) and only allow affine transformations (rotations and translations), however, more complicated geometry systems including *constructive solid geometry (CSG)* or spline based modeling systems can be mapped to our notion of space as well.

3.2. The Basic Model

The main element of the model is a *component*. A component can be used to describe individual parts of the system as well as the entire system. By composition (*cf.* Sec. 3.6), multiple components can be combined into a single component, which allows the assembly of components for larger parts of the systems from the components for smaller parts. This composability is central to the model as it helps in reducing complexity in modeling and analyzing a system by decomposing it into a hierarchy of smaller components, which can be modeled and analyzed in isolation. Components are also used to bridge the gap between spatial properties and behavior models as shown in Fig. 1.

A component consists of a *syntactic interface* (its structural aspects), which is described next, and a *semantic interface* (its behavioral description), which is detailed in the following subsection. The structure of a component is given by sets of *input* and *output ports*, *parts*, *detectors*, and *movers*. The input and output ports describe the communication endpoints of the component, where communication includes signal exchange between controllers via a common bus as well as the transmission of the number of revolutions from a motor to the belt conveyor it is driving. The ports are annotated by a type giving the kind of messages exchanged.

The spatial aspects of the component are given by the parts, which describe the dimensions and shape of the component. These are the portions of the system traditionally modelled in 3D-CAD tools, although we usually use simplified geometry for our abstract models. Locations in space where the component can observe and react to the presence of other parts are marked by detectors. These are used for example to model the light ray of a photoelectric barrier or the covered range of a proximity sensor. Both parts and detectors are assigned volumes of our space system. The remaining elements, movers, denote facilities which affect the position of other parts or material. They correspond to actuators in the real system and are associated with an axis and kind of motion (linear or rotational).

3.3. Behavior Specification

To specify the behavior of a component, we use hierarchical communicating hybrid state machines. These consist of discrete control states, state variables, and transitions. Control states are connected by transitions, which model discrete events in the system. Transitions can be guarded by Boolean expressions over the state variables and cause these variables to be updated. The state machine is hierarchical, *i. e.*, states may contain further states, to ease the structuring of complex behavior. Being communicating means that the guards of transitions may also depend on the messages received at the input ports of the component whose behavior is specified. Additionally, a transition may cause messages to be sent at the output ports.

The hybrid part of the state machine is mostly used to describe motion of the system's parts. For this, each state can be augmented by linear differential equations in the state variables, which describe the continuous behavior between discrete steps. Note that we are using hybrid automata only for conceptional reasons to obtain a clear separation between discrete and continuous change and to simplify notation. As mentioned in Sec. 3.7 the interpretation and evaluation of these state machines is performed in discrete time, which can lead to artifacts due to approximation er-

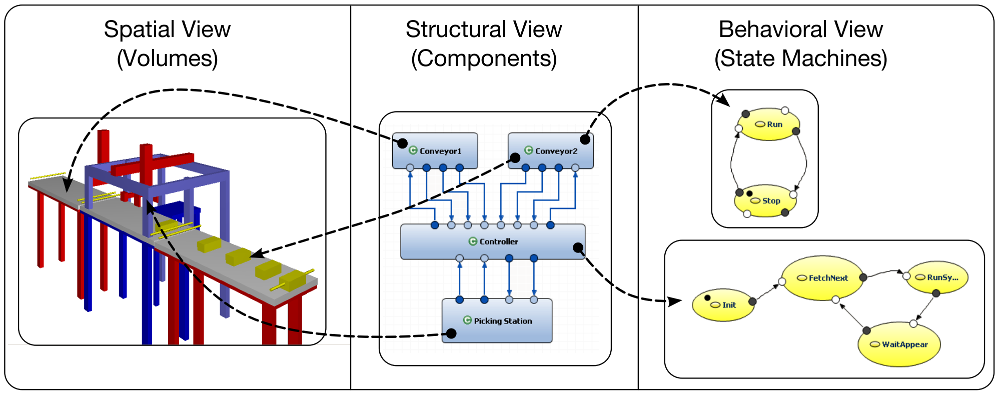


Figure 1. Linkage of Different Views in the Integrated Model.

rors, which, however, often can be neglected at our level of abstraction.

Detectors are integrated with these automata by interpreting them as read-only Boolean state variables, which are automatically set by the simulation environment based on actual collisions between parts and the respective detector. Similarly, each mover corresponds to a state variable, into which the amount of motion along the mover's axis is written into. Usually these mover variables are used in the differential equations to enforce motion as long as the component is in a certain state.

In addition to these state machines, our framework supports the usage of alternative specification techniques (both from a theoretical point of view, and in terms of our tooling infrastructure). We are currently experimenting with alternative description techniques, such as tabular notations, graphical traverse path specification, or HMI definitions for interactive user input during simulations. While state machines are the most generally applicable, other techniques can be more efficient for certain behaviors. Additionally we support the concept of behavior modifications, where a behavior description is augmented by an additional model. This allows to use different description techniques for orthogonal aspects of the behavior. We are using this technique to describe possible errors for components representing hardware. These components can be used to test whether the controller is capable of ensuring safety even in the presence of hardware defects. The details of this technique are given in [2].

3.4. Material Handling

An essential aspect of industrial automation is the treatment of material, which covers an extreme range from bricks and bottles to partially assembled circuit boards or car bodies. To model and analyze such systems, we must be able to describe and simulate this material and especially its interaction with the system's components.

To describe a single piece of material, we use the same component-based description technique as for the system itself. This is done to not increase the number of model elements further and also because the material can be as complex as the system itself. The main difference is that for a material component there can be multiple instances in the simulation and instances have to be created and destroyed dynamically. So called *entries* are used to introduce new (material) components to the simulation, while *exits* are used to remove them. Both are associated with a spatial volume to describe where components are inserted or discarded.

For the interaction between components of the system and the material there are two questions. The first is *when* interaction does occur, the second is *what kind* of interaction it will be. To determine, when material is in "interaction range", we are using detectors (which are already used for interaction, as they sense the parts of material components as well). Each component is complemented by *binding conditions*, which are basically predicates over the detectors. A material component is bound (and thus interacting), if the set of detectors of a component with which it collides fulfills this predicate. In this case the detectors either form the surface of some gripping- or friction-based transportation device, or the detection region of a sensor. For example a material might only be interacting with a gripper component, if it is gripped from both sides, *i. e.*, activates both gripper detectors. Detectors can also be completely deactivated on a per-state basis, *e. g.*, to model switching off a magnetic field.

The kind of interaction is defined by the binding condition as well and either connects the material to a mover of this component, causing the material to be moved as long as contact to the detectors is not lost, or temporarily connecting ports of the material component with those of the system's component. The later can be used to model complex sensors (such as RFID scanners), which read some information from the material, or components which change the state of the material component. Using port based communication, complex interactions can be handled without extending the meta-model.

3.5. Motion and Collision

As argued before, we consider the spatial aspects of the system to be highly important for the automation domain. This especially includes the dynamics, *i. e.*, the motion of the system's parts and material objects over time. As parts represent solid objects, their volumes may not overlap, *i. e.*, collide with each other. We use the term *collision* to indicate an actual intersection of volumes here, not only for non-penetrating "touching". Thus, just as in the real world, motion initiated by components might not be performed due to blocking by other parts.

In our model we interpret the motion initiated by components (which is realized by writing to the mover's variable the "amount" of motion required) as a request. Whether this request is fulfilled is determined by the simulator in a separate phase. In this phase possible collisions caused by movements are calculated and motion requests are partially cancelled based on the results. Often, this cancellation propagates along the component hierarchy, as an entire robotic arm might be blocked if its gripper touches an obstacle. The same holds for material, which has been bound to a component's mover. However, for these bindings we differentiate between weak and strong bindings. Weak bindings indicate that the motion of the underlying component is not affected if the bound material can not move. This is used, for example, for belt conveyors, which will continue to move and transport other objects on its surface even if one material object is blocked by a barrier. Strong bindings in contrast also cause the moving component to stop. An example for this is material gripped by a robot, which can not complete its movement if the object it handles is stuck. If material would be bound to multiple movers, strong bindings always prevail. In case of multiple equally strong bindings, we decided to choose only one binding non-deterministically but fair. This way each binding will eventually be active and make some progress. However, more complicated resolution schemes for multiple active bindings are imaginable.

3.6. Composition

One of the important aspects in engineering is composition, *i. e.*, creating a new part by combining existing ones. This allows to handle complex problems by dividing them into manageable parts and assembling the solution from them. Furthermore, existing building blocks can be easily integrated into new designs by composition. Thus, composition is central for our approach.

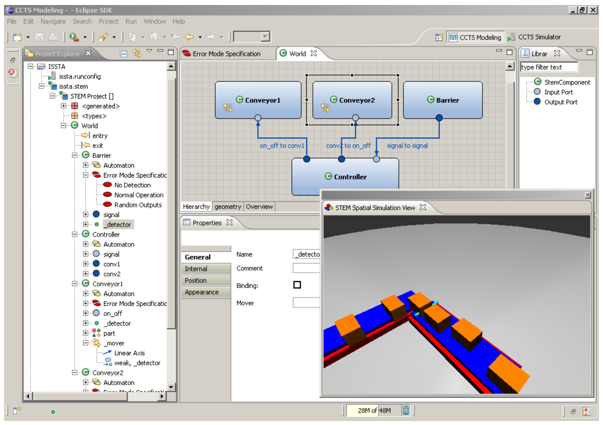


Figure 2. The Modeling Environment AF/STEM

When composing two or more components, their input and output ports can be connected by *channels* (if they transport the same type) to model communication between these components. Additionally ports can be associated with the ports of the surrounding (composed) component. By this the syntactic interface of the composed component is formed by a subset of the ports of all its sub-components. The parts, detectors, and movers of the composed component are just the union of the respective elements of the subcomponents. A component may introduce additional (static) parts, and may apply a transformation to its subcomponents. Thus composition for parts is similar to the approaches used in mechanical CAD systems. Furthermore, a component can be connected to the mover of a sibling component during composition, which makes the component follow every transformation of the respective mover.

The behavior of the composed component is then fully defined by the behavior of its subcomponents and the channels between them. From a theoretical point of view, composition can be interpreted as recursive equations over stream-processing functions [6, 15]. Practically, we interpret the composition of behavior as composition of the corresponding state machines, which is just an automaton product. To avoid the state space explosion, this product is used only implicitly in the simulator. We want to point out, that this notion of composition is an important difference to Petri-net based approaches, as, other than state machines, Petri-nets are not compositional, *i. e.*, given two Petri-nets there is no known automatic procedure to create a Petri-net that is equivalent to both of them running synchronously and in parallel.

3.7. A Note on Time

While mechatronic systems contain a large amount of control algorithms, which require a continuous model of time, on the level of abstraction we are interested in these only play a minor role and can be abstracted by discrete algorithms most of the time. Thus we decided to use a simple model of discrete time, where time advances in (not necessarily equidistant) ticks and events within a single tick can not be differentiated. Furthermore, components can send and receive at most one message on each port in a single tick. In our experience, most effects can be modelled and analyzed in this time model, as long as the length of a single time interval (tick) is chosen sufficiently small.

The limitation to this discrete time model simplifies both the understandability and analyzability as well as the creation of models and supporting tools. So, more care can be taken of the interplay of behavioral and spatial properties, which in our opinion is the central problem on a more abstract modeling level. However, extending the theory to more complex models of real or continuous time can be easily achieved following [4], although creating suitable tool support will be more challenging in this case.

3.8. Tool Support

To allow experimentation with these models, a prototypical editor for the model introduced so far has been implemented. A screenshot of the tool, called AF/STEM[1], is shown in Fig. 2. It consists of a basic type system, modeling support for components, automata, and simple geometry, as well as simulation and visualization support for these models. We also use the application to implement process support, which is described in the following section.

As components in industrial automation systems are usually used multiple times (*e. g.*, drives or belt conveyors), a concept for structured reuse is crucial for a modeling tool to avoid rebuilding the same components over and over again. For this we implemented generative libraries as described in [14] for our tool. Our implementation also allows the creation of parameterizable library elements, so for example a library element of a belt conveyor could support a variable user provided length.

4. Process Integration

Functional descriptions are often used in industrial practice. However, in most cases they are textual and, hence, not suited for computer-aided engineering. They serve as an initial system specification, but become outdated and obsolete as more detailed system models evolve. Simultaneously, these discipline-specific models become inconsistent, the maintenance and coordination efforts grow. Thus, the main purpose of the presented abstract mechatronic model is to allow decision-making engineering experts from different involved disciplines to coordinate and synchronize the development progress within an environment which offers

[1]`http://af3.in.tum.de/index.php/AF/STEM`

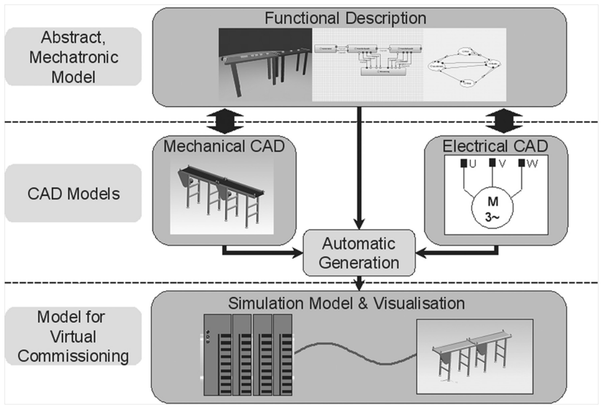
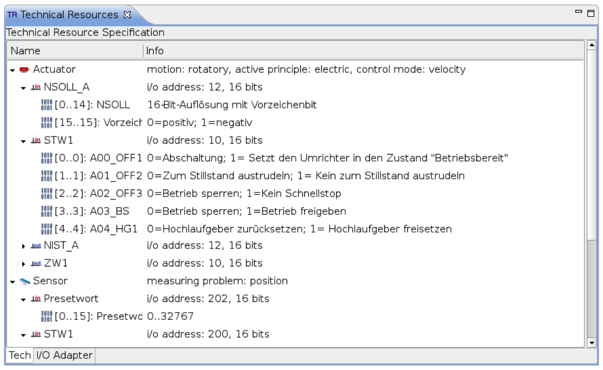

Abstract, Mechatronic Model
Functional Description
CAD Models
Mechanical CAD
Electrical CAD
U
V
W
M 3~
Automatic Generation
Model for Virtual Commissioning
Simulation Model & Visualisation

Figure 3. Process Integration of AF/STEM



Name	Info
Actuator	motion: rotatory, active principle: electric, control mode: velocity
NSOLL_A	i/o address: 12, 16 bits
[0..14]: NSOLL	16-Bit-Auflösung mit Vorzeichenbit
[15..15]: Vorzeich	0=positiv; 1=negativ
STW1	i/o address: 10, 16 bits
[0..0]: A00_OFF1	0=Abschaltung; 1= Setzt den Umrichter in den Zustand "Betriebsbereit"
[1..1]: A01_OFF2	0=Zum Stillstand austrudeln; 1= Kein zum Stillstand austrudeln
[2..2]: A02_OFF3	0=Betrieb sperren; 1=Kein Schnellstop
[3..3]: A03_BS	0=Betrieb sperren; 1=Betrieb freigeben
[4..4]: A04_HG1	0=Hochlaufgeber zurücksetzen; 1= Hochlaufgeber freisetzen
NIST_A	i/o address: 12, 16 bits
ZW1	i/o address: 10, 16 bits
Sensor	measuring problem: position
Presetwort	i/o address: 202, 16 bits
[0..15]: Presetwc	0..32767
STW1	i/o address: 200, 16 bits

Figure 4. Technical Resource Model

design methods and validation support. Thereby, the role of our model is threefold (*cf.* Fig. 3): It serves as an input for more detailed discipline-specific models, it keeps these models in-sync throughout the development process, and it can be used to generate models for virtual commissioning.

4.1. Forward Engineering

Starting with customer and supplier discussions about machine functionality, engineers develop a first machine concept. Already during this step the abstract mechatronic model can give support by exploring technical solutions and discussing them with the customer. For these steps simulation support is crucial, as the customer usually does not understand the modelling language used.

In later design phases the machine model is enriched by more technical information about the mechanical and electrical parts, such as which actuators and sensors to use and the engine power required. This information together with the component structure will later be incorporated into the more detailed engineering models, such as mechanic and electric CAD. To capture this data before CAD modelling can be started, we allow components to be augmented by what we call the *technical resource model* (*TRM*, *cf.* Fig. 4).

It contains the technical details of a component's realization. In particular, it stores descriptions (*e. g.*, vendor, model number, principle of operation, *etc.*) of sensors and actuators to be used. For every sensor and actuator the signal-level interface can be defined. All this information allows an initial import of abstract mechatronic models into the mechanical and electrical models as well as automatic HIL/SIL generation. Presently, the module structure, simplified geometry, and the technical sensor/actuator data can be imported from AF/STEM into the NX[2] CAD system.

4.2. Tracing and Consistency

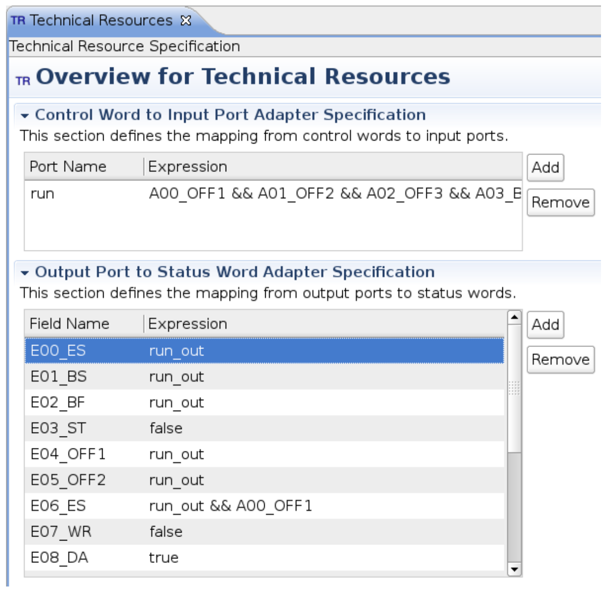
To relate elements of the abstract integrated model to parts in more detailed engineering models (*e. g.*, the different CAD models), our tool supports the creation and management of tracing files. These tracing files can be automatically generated during forward engineering or updated manually. This way the integrated model serves as a hub in the modeling landscape and allows elements from different models to be related to each other via this central model. The tracing links can be used for documentation purposes, rationale management, and consistency checking.

During consistency checking, we automatically compare linked elements and their properties with each other to find deviations and also identify elements which are not covered by the tracing files. These checks can for example compare the types of actuator used (which can be found in the TRM), or the position of a part in the 3D-CAD with the one used in the integrated model. This way the divergence of the used models can be detected and avoided early, keeping all models meaningful during the entire development phase.

4.3. Virtual Commissioning

The testing of complex controller software is practically infeasible without the machine being controlled. To avoid testing at the real machine, which can only be performed after assembly and might damage the machine, today often virtual machine models are used. This is usually referred to by the term virtual commissioning or HIL testing and allows the software tests to happen earlier in the development process, thus often significantly improving software quality.

Unfortunately, the creation of virtual commissioning models is a time-consuming and error-prone task, since information needed is spread over different artifacts of the engineering process. This information is manually gathered and manually incorporated into the simulation models, producing redundancy and causing additional maintenance effort. In particular, for these virtual machine models one needs geometry from M-CAD, wiring plans and bus topologies from E-CAD, signal-level communication protocols from PLC project, as well as the rather implicitly available knowledge about the behavior of the used mechatronic components, material, and its flow through the plant.


Technical Resources

Technical Resource Specification

Overview for Technical Resources

Control Word to Input Port Adapter Specification

This section defines the mapping from control words to input ports.

Port Name	Expression
run	A00_OFF1 && A01_OFF2 && A02_OFF3 && A03_E

Output Port to Status Word Adapter Specification

This section defines the mapping from output ports to status words.

Field Name	Expression
E00_ES	run_out
E01_BS	run_out
E02_BF	run_out
E03_ST	false
E04_OFF1	run_out
E05_OFF2	run_out
E06_ES	run_out && A00_OFF1
E07_WR	false
E08_DA	true

Figure 5. I/O Adapter Specification

All ingredients listed above are covered and synchronized with the CAD tools by the presented abstract functional model. This fact allows us to automatically generate models for virtual commissioning. The signal-level interface from TRM is exactly the interface used for the communication between PLC and the physical system components. On the other side, components from our model communicate via logical messages. Thus, for the generation the TRM associates logical ports with expressions defined on the signal-level interface. These mappings between fields and ports are incorporated into *I/O adapters* (Fig. 5) which are generated with the rest of the abstract model and adapt the abstract system model to the protocol supported by the PLC program. Currently, we can generate HIL and SIL models running in the context of SIMATIC[3] solutions.

5. Conclusion and Outlook

We presented a novel integrated modelling approach for the specification of mechatronic systems, especially in the context of industrial automation. It is designed as a central artifact in the interdisciplinary development process of mechatronic systems and can serve as the basis for verification and validation activities. The approach is complemented by a tool prototype and a methodology, which coordinates and integrates mechanical, electrical, and control software development activities as well as domain-specific tools, like E-CAD or M-CAD.

[2]`http://www.plm.automation.siemens.com/en_us/products/nx`

[3]`http://www.automation.siemens.com/simatic`

Our approach permits modular development of mechatronic systems. The system is described by a set of interacting components, which encapsulate the functional behavior, geometry, and material flow. This fact facilitates reuse and library-based development.

The realization of the presented concepts in a tool prototype with an operational semantics permits early validation of system models by simulation, as well as the support of further quality assurance measures, like generation of test-cases and virtual commissioning models. The latter one is already realized for SIMATIC. Further contribution is the integration of our approach with existing development processes and CAD tools. This is of particular importance for acceptance of research results in practice, since there exists a significant amount of legacy models which have to be maintained. For the seamless integration with different design activities we presented consistency checking and data exchange mechanisms.

Our next step will be to gain more evidence about the scalability of our approach by exercising it on industrial-size case studies. A further important point is the integration of automatic test-case generation techniques and model checking back-ends. Finally, we plan the extension of our approach to support further activities of the engineering process. In particular, we consider the generation of PLC code and initial generation of more elaborated physic simulation models as promising fields of investigation.

References

[1] M. Bonfé, C. Fantuzzi, and C. Secchi. Unified modeling and verification of logic controllers for physical systems. In *Proc. of CDC-ECC'05*, 2005.

[2] J. Botaschanjan and B. Hummel. Specifying the worst case - orthogonal modelling of hardware errors. In *Proc. of ISSTA'09*, 2009. To appear.

[3] J. Botaschanjan, B. Hummel, and A. Lindworsky. Interdisziplinäre Funktionsmodellierung im Anlagenbau. *ZWF Zeitschrift für wirtschaftlichen Fabrikbetrieb*, 01-02:71–75, 2009.

[4] M. Broy. Refinement of time. *Theoretical Computer Science*, 253(1):3–26, 2001.

[5] M. Broy, F. Huber, and B. Schätz. AutoFocus – Ein Werkzeugprototyp zur Entwicklung eingebetteter Systeme. *Informatik Forsch. u. Entwicklung*, 13(13):121–134, 1999.

[6] M. Broy and K. Stølen. *Specification and Development of Interactive Systems: Focus on Streams, Interfaces, and Refinement*. Springer, 2001.

[7] V. Deligiannis and S. Manesis. A survey on automata-based methods for modelling and simulation of industrial systems. In *Proc. of ETFA'07*, 2007.

[8] M. Deppe, M. Zanella, M. Robrecht, and W. Hardt. Rapid prototyping of real-time control laws for complex mechatronic systems: a case study. *J. Syst. Softw.*, 70(3):263–274, 2004.

[9] H. Dierks. PLC-automata: A new class of implementable real-time automata. In *Procs. of ARTS'97*. Springer, 1997.

[10] W. Eversheim. *Inbetriebnahme komplexer Maschinen und Anlagen*. VDI, 1990.

[11] L. Ferrarini. A theoretical framework to model and analyze manufacturing systems. In *Proc. of CDC'94*, 1994.

[12] D. Gabelaia, R. Kontchakov, Á. Kurucz, F. Wolter, and M. Zakharyaschev. Combining spatial and temporal logics: Expressiveness vs. complexity. *J. AI Research*, 23:167–243, 2005.

[13] T. Henzinger. The theory of hybrid automata. In *Verification of Digital and Hybrid Systems*, NATO ASI Series F: Computer and Systems Sciences 170. Springer, 2000.

[14] M. Herrmannsdoerfer and B. Hummel. Library concepts for model reuse. In *Proc. of LDTA'09*, 2009.

[15] B. Hummel. A semantic model for computer-based spatio-temporal systems. In *Proc. of ECBS'09*, 2009.

[16] B. Hummel and P. Braun. Towards an integrated system model for testing and verification of automation machines. In *Proc. of MiSE '08*. ACM, 2008.

[17] R. Isermann. *Mechatronic systems: fundamentals*. Springer, 2005.

[18] D. Kaynar, N. Lynch, R. Segala, and F. Vaandrager. *The Theory of Timed I/O Automata*. Morgan & Claypool, 2006.

[19] O. Kutz, F. Wolter, H. Sturm, N.-Y. Suzuki, and M. Zakharyaschev. Logics of metric spaces. *ACM Transactions on Computational Logic*, 4(2):260–294, 2003.

[20] A. Paoli, M. Sartiniy, and A. Tilli. Rapid prototyping of logic control in industrial automation exploiting the generalized actuator approach. In *Proc. of ETFA'08*, 2008.

[21] M. N. Rooker, T. Strasser, G. Ebenhofer, M. Hofmann, and R. V. Osuna. Modeling flexible mechatronical based assembly systems through simulation support. In *Proc. of ETFA'08*, 2008.

[22] B. Schätz, A. Pretschner, F. Huber, and J. Philipps. Model-based development of embedded systems. In *OOIS Workshops*, 2002.

[23] T. Strasser, M. Rooker, G. Ebenhofer, I. Hegny, M. Wenger, C. Sunder, A. Martel, and A. Valentini. Multi-domain model-driven design of industrial automation and control systems. In *Proc. of ETFA'08*, 2008.

[24] K. Thramboulidis. Model-integrated mechatronics - toward a new paradigm in the development of manufacturing systems. *IEEE Trans. Indust. Inform.*, 1(1):54–61, 2005.

[25] K. Thramboulidis. Challenges in the development of mechatronic systems: The mechatronic component. In *Proc. of ETFA'08*, 2008.

[26] M. Tiller. *Introduction to Physical Modeling with Modelica*. Kluwer Academic Publishers, 2001.

[27] Z. ChaoChen, C.A.R. Hoare, and A.P. Ravn. A calculus of durations. *Inform. Process. Lett.*, 40(5):269–276, 1991.

[28] M. Zaeh, N. Moeller, and W. Vogl. Symbiosis of changeable and virtual production - the emperor's new clothes of key factor for future success? In *Proc. of CARV'05*, pages 3–10. Utz, 2005.

[29] S. Zairi, B. Zouari, and L. Pitrac. A formal approach for the specification, verification and control of flexible manufacturing systems. In *Proc. of ETFA'07*, 2007.