Behavior Modeling of Automation Components using cross-domain Interdependencies

Benjamin Brandenbourger, Milan Vathoopan, and Alois Zoitl
fortiss GmbH
Guerickestr. 25, 80805 Munich, Germany
{brandenbourger, vathoopan, zoitl}@fortiss.org

***Abstract*—Within the multi-disciplinary engineering process of a cyber-physical production system, the information about the plant is increasing steadily. Different departments work on separate domain-specific models and tools allowing a specialized view of the same system. However, data exchange and linking the heterogeneous information across domains is a critical issue. This paper investigates behavior modeling of automation components out of interdependencies coming from different disciplines such as mechanics, electrics, and software engineering. By this, the interlinking of the domain-specific models, as an enabling step towards early cross-domain validation and facilitated system evolution, allows a common understanding of the correct functionality of the system. The concept will be explained through the example of a pneumatic stopper and evaluated in AutomationML, an emerging standard in automation, to store and exchange artifacts between domain-specific engineering tools.**

I. Introduction

When it comes to Industrie 4.0 [1], manufacturing systems are driven from mass production towards highly customized goods with a lot size close to 1 [2]. Faster commissioning and shorter down-times support the main goal of efficient production for best rate of yield. Cyber-physical systems such as intelligent components and machines present a new perspective for this enterprise environment by offering a modularized encapsulation and flexible applications. Their increasing capabilities enable the exposition of their smart behaviors as services, allowing a service-oriented paradigm across all layers of the enterprise manufacturing environment [3], [4]. But in industry practice, domain-specific engineers use heterogeneous and especially isolated tools with barely no interaction and data exchange capabilities [5].

In this context, the goal of this paper is to create an integrated mechatronic model of an automation component encompassing different domain-specific models and their consistent interdependencies with each other [6]. This common model is composed of various heterogeneous artifacts created by different disciplines such as mechanics, electronics, and information technology [7]. The progression and transformation of the artifacts are an issue for maintaining existing interdependencies along the engineering phases. The presented approach uses the behavioral model as the center of the mechatronic model while connecting the domain-specific artifacts together in order to ensure a dedicated unit behavior which can be provided to an overall system [8]. The behavioral model describes the behavior of an individual component, combined components, stations, and eventually the complete production system in a particular physical state [8]. The presented work concentrates on behavioral models of individual components but can later be expanded at each object scale and design stage. The idea is based on a model-based method presented in [9] which encompasses seamless engineering with continuous data management.

978-1-5090-1314-2/16/$31.00

The remainder of this paper is structured as follows. Section II gives an overview of available work in the field of behavior modeling and its integration in the bigger context of reconfigurable manufacturing systems. Background and definitions of used terms are described in Section III. Section IV describes the approach for linking the behavioral model to other domain-specific models while Section V deals with the implementation of the approach in AutomationML (AML). In Section VI the advantages of the approach are presented and discussed. Finally, Section VII offers the paper conclusion.

The work presented in this paper has been supported by the German Federal Ministry of Economic Affairs and Energy (BMWi) as part of the research program ”Autonomik Industrie 4.0” in the research project OPAK (Funding no. 01MA13012F).

II. Related Work

Hummel introduces integrated behavior models in his work, which combine structural, behavioral, and spatial aspects of a mechatronic system into one common model [10]. He uses the models, which are based on a thorough mathematical theory, for automated analysis and early validation by simulation. Contrary to our approach, Hummel focuses on centralized composed automation systems and not on decentralized automation components. Furthermore, the structural view is the central point of connection, whereas we approach the track of having the component's behavior as a hub.

Berardinelli et al. investigate the combination of AutomationML and Performance Model Interchange Format (PMIF) [11]. PMIF is a common representation devised in the performance engineering domain for model-based system performance analysis and simulation based on Queueing Network Models. The goal is an early performance validation of cyber-physical production systems by linking the two models. The approach of linking the AML-model with the PMIF-model is similar to the presented work. However, Berardinelli et al.

provide no details on describing the behavior of automation systems.

Another approach presented by Legat et al. is based on interface behavior modeling of design artifacts from various disciplines [12]. The approach supports automatic verification of the functional conformance of the artifacts. The different scopes such as context (mechanical engineering), platform (electric/electronic engineering discipline) and software (software engineering) are linked together to a system viewpoint which includes the whole mechatronic system. However, the behavior is modeled indirectly and implicitly by linking items of different scopes together. A discipline-independent behavioral model does not exist.

Shah et al. [13] pursue the approach of capturing dependencies between different domains through the use of a common view of the system in SysML. The necessary semantics from each of the domains are included within SysML by using profiles which do not modify the underlying SysML metamodel. In this work, the behavioral model is not taken into account explicitly but modeled implicitly in Modelica. Nevertheless, the main idea of creating dependencies between different domains through the use of a common view of the system in a description language is similar to the presented approach in this work.

In [14], the authors describe a manufacturing process in AML adopting the Product-Process-Ressource (PPR) view. A behavioral description of PLC software deployed on mechatronic components by means of sequential function charts (SFC) is also provided. The authors follow the same approach on modeling resources which contain all process actions executable by the resource with their control and behavior. However, the exact interlinking between control code, behavior and other models is not described in detail.

In summary, our presented approach differentiates from the above mentioned works by using an explicit behavioral model which forms the center of the integrated mechatronic model of an automation component. Domain-specific models influence the behavior model by linking internal objects from different domains together. Especially the combination of so-called skills, representing executable functions, with the behavioral model is a novel approach.

III. Domain-specific models

The different disciplines involved in the engineering of an automation component use a heterogeneous tool landscape which needs different views on the component realized in different domain-specific models. These models are created and worked with in separate engineering tools. In previous work [9], we proposed a multi-level modeling approach based on a metamodel. A metamodel is a 'model used to model modeling itself' [15]. It means, that this metamodel is a model that specifies valid concepts which can exist within the domain of industrial automation. The metamodel enables the modeling of integrated mechatronic models of automation components on different levels of details. For this, various libraries containing domain-specific information (e.g., skills, control code snippets, mechanical interfaces, ...) are embedded in the metamodel. Manufacturer specific components inherit from the mechatronic models and support a correct-by-design technique.

This work focuses on four different models contained in the integrated mechatronic model. More domain-specific models could be used (e.g., EPLAN), but would not create added value for understanding the presented approach. The four models described in domain-specific languages (DSLs) represent:

3D-model and Kinematics: A 3D-model contains a graphical representation of the automation component and is created by the mechanical engineer. Additionally, kinematics give information about moving space and trajectories.

Control Code: Control code controls the hardware of the automation component with human-readable computer instructions and is created by the software and electrical engineer. The control code is subdivided into sections such as program organization units (POUs) from IEC-61131 which contain commands for reading inputs and setting outputs.

Skills: Skills are a standardized encapsulated representation of abilities offered by the automation component. These basic automation functions are defined by the software engineer. In more complex systems, basic skills can be orchestrated to composed skills which are in the most cases application-specific. The presented approach is based on functional engineering that supports the use of descriptive functions instead of abstract I/O-values. The exact way of how a basic skill works is programmed in the corresponding POU in the control code which interacts with the hardware over inputs and outputs.

Behavior: The description of how a component should behave is realized in a behavior model. A behavior model reflects the expected correct behavior for example in form of a state machine or a sequential function chart. It can be created and extended by any discipline as the correct behavior is discipline-independent.

IV. Modeling Interdependencies between domain-specific models

The four models introduced in Section III describe domain-specific views on the modeled automation component. The integrated mechatronic model aims at including the different domain-specific models and especially creating inter model relationships. These relationships are used to model cross-domain dependencies of the same object stored in different domain-specific models. We are not using model transformation to map the different domain-specific models and languages to other models.

All models of an automation system have in common, that the hardware and the correct behavior of the component form the basis [10]. The correctness of the domain-specific artifacts is verified and validated through debugging and testing of the code, using simulation, and applying correct-by-design techniques [16]. Therefore, the behavior model is used as the accumulated information storage while different domain-specific models directly influence the behavior model (see Figure 1).

The behavior model can be modeled in different ways, e.g., finite state machine (FSM), pulse diagram, sequence flow chart (SFC) etc. Depending on the requirements (e.g., timing constraints, parallelism) one model is more suitable than the other. In our approach and for better understanding, we assume the behavior model to be a FSM, even if timing constraint are more effort to be modeled.

The synthesis process of the behavior model consists of three steps:

1) Identifying the necessary, component-specific states of the behavior model.
2) Placing the transitions between the states describing the correct behavior.
3) Linking the information of the domain-specific models to the transitions and their triggers.

The last step shows clearly how the intra model relationships (here state transitions) on the behavior model are directly dependent on the given domain-specific models or influence them.

The following analyzes the behavior relevant content of the domain-specific models and the links to and from the behavior model (arrows in Figure 1):

A. Behavior - Kinematics

The invocation of the corresponding kinematics stored in the 3D-model can be derived out of the behavior model. A kinematic is performed when a transition is taken or a new state is reached in the behavior model (e.g.; when a pneumatic stopper unit changes from the state *retracted* to *extended*, the corresponding kinematic is called in the 3D-model).

B. Skills - Behavior

Calling a skill corresponds to a transition from one state to another in the behavior model. If the skill *moveOut* offered by a pneumatic stopper unit is called, the corresponding transition from the state *retracted* to *extended* is performed. If the FSM of the behavior model resides in a state which doesn't have any transition supporting the called skill, the skill is not performed. The converse ('a transition in the behavior model calls a corresponding skill') is out of scope of this paper.

C. Behavior - Control Code

A transition in the FSM of the behavior model can call a POU located in the control code. With this relationship the link between the components behavior and its hardware, controlled by inputs and outputs through POUs stored in the control code, is realized. Furthermore, as skills invoke transitions in the behavior model and a transition can invoke a POU-call, by transitivity a skill can call a POU which corresponds exactly to the main idea of how skills are implemented.

D. Control Code - Behavior

Input-values of the hardware influence the beginning or completion of transitions in the FSM of the behavior model. Therefore, inputs are modeled as *OnStateEntry* or *OnStateExit* conditions in the FSM (e.g., limit sensor needs to toggle before the stopper unit has reached a defined state; a filling level sensor needs to switch before another state can be reached).

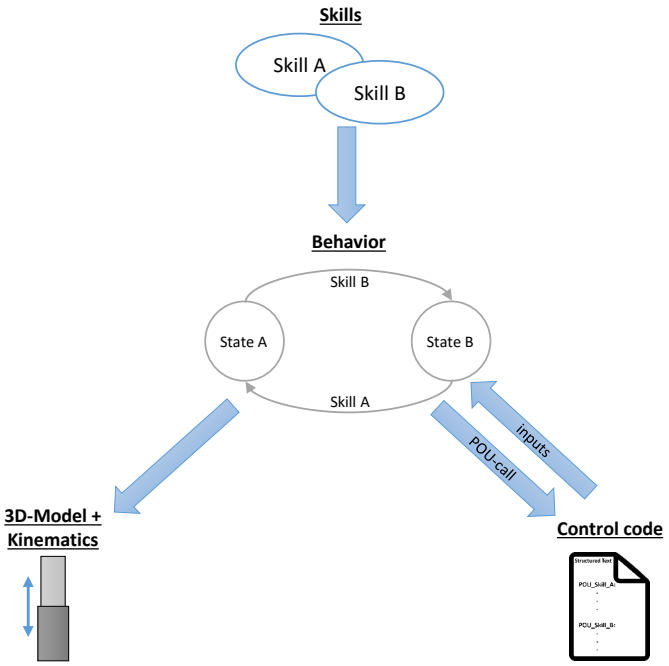

Fig. 1. Integrated, mechatronic model and interdependencies between behavior, kinematics, skills, and control code

V. Evaluation

In [9] a metamodel supporting the different domain-specific models is presented. However, this metamodel needs to be extended with a class of links for being able to create the interdependencies between the domain-specific models. The metamodel has been implemented in AutomationML which supports natively the linking of objects. Herefore, so-called *VariableInterfaces* and *internal/external links* are used for modeling relationships within models (intra-model dependencies) and between different models (inter-model dependencies). They inherit an attribute called *refURI* describing a unified resource identifier (URI) and are of type *xs:anyURI*. The URI contains a fragment right after the delimiter '#' according to [RFC3986, ch.3]. This fragment addresses a variable within a file containing a domain-specific model. Similar to [8], this approach allows associating variables from 3D-models, control program, skills, and behavior models to each other by using AutomationML.

A pneumatic stopper unit is used as a simple application example. The stopper unit has two logical positions (*retracted* and *extracted*) which are detected by two limit sensors and offers the skills *moveOut* and *moveIn*. Furthermore, the 3D-model modeled in COLLADA consists of a static part and a movable piston and stores the kinematics for extracting and retracting the piston. The control code is structured in two POUs (*POU_moveOut* and *POU_moveIn*) and stored in a PLCopen XML file. Each POU contains the actual information

about setting the correct outputs for switching the valve which controls the piston.

Applying the presented approach to this application example results in a behavior model described in PLCopen XML with the two states *retracted* and *extracted*. The transitions correspond to the skills *moveOut* and *moveIn* and the onEntry-conditions of the states are linked to the corresponding input values of the limit sensors.

VI. Resulting Options

Besides efficient and time-saving integration of behavior models into integrated mechatronic models, more advantages can be deducted out of modeling interdependencies:

- **Scalability:** Multiple components can be combined into a single system by composition. The presented approach is also able to support more complex systems.
- **Correctness by construction:** Extending an integrated mechatronic model with more information (e.g. from a plant engineer or a manufacturer) will keep the basic behavior model unchanged and therefore correct.
- **Avoidance of malfunction:** Only skills which are modeled as transitions in the behavior model can be called in a specific state. The hardware can therefore be protected from faulty usage or programming by the plant engineer.
- **Automatic code generation:** Control code can automatically be synthesized out of the behavior model as this model contains all control code relevant information (e.g., inputs/outputs, timing, sequences, ...).
- **Simulation, virtual commissioning:** The integrated mechatronic model can be used as a virtual automation component which reacts on external factors such as digital inputs by performing for example a kinematic depicted in a visualization.
- **Error detection:** Inconsistent input-values or state changes which are not modeled in the behavior model point towards an occurring error.

VII. Conclusion

The understanding of the correct operation mode of an automation component is domain-independent. A behavioral model of this automation component can therefore be used from an early stage in engineering to model the correct operation mode. By means of a simple but intuitive application example, it has been shown how interdependencies between the different disciplines and the behavioral model, and the way of affecting the behavior has been modeled. Different options result out of this modeling such as better support for parallel engineering, cross-domain error-detection, and a common understanding of the correct operation mode.

Since the paper presents currently ongoing work, for a first step, some aspects are beyond the scope of this article, e.g. determination of the behavioral states, skills, and the electrical/electronic, inverse kinematic by linking the kinematic model to the behavior model as well as mechanical aspects have been considered in an abstract way. Despite the improvements for behavioral modeling achieved with the presented methodology, there is still a lot of work remaining to further improve behavior modeling. The future work will investigate the scalability of the approach by supporting inheritance of behavioral models through different abstraction layers in order to model more complex automation components and even complete automation systems. The usage of behavioral models needs to be discussed with industry partners in further detail. Especially the comparison of the presented expected behavior model with the input values from the hardware and deduction of error-states is a passionate topic. Furthermore, the creation and processing of sufficiently formalized and standardized behavioral models in order to allow automated analysis is another example of further work that the authors are aiming to elaborate.

References

[1] VDI, "Statusreport Referenzarchitekturmodell Industrie 4.0 (RAMI4.0)," *http://www.vdi.de/industrie40*, 2014.

[2] U. Lindemann and G. Baumberger, *Individualisierte Produkte*. Springer, 2006.

[3] L. De Souza, P. Spiess, D. Guinard, M. Khler, S. Karnouskos, and D. Savio, "Socrades: A web service based shop floor integration infrastructure," *The internet of things*, pp. 50–67, 2008.

[4] D. Guinard, V. Trifa, S. Karnouskos, P. Spiess, and D. Savio, "Interacting with the SOA-based internet of things: Discovery, query, selection, and on-demand provisioning of web services," *IEEE Transactions on Service Computing*, vol. 3, pp. 223–235, 2010.

[5] A. Fay, S. Biffl, D. Winkler, R. Drath, and M. Barth, "A method to evaluate the openness of automation tools for increased interoperability," *Industrial Electornics Society (IECON)*, pp. 6844–6849, 2013.

[6] K. Thramboulidis, "Challenges in the development of mechatronic systems: The mechatronic component," *IEEE Int. Conf. on Emerging Technolgies and Factory Automation (ETFA)*, pp. 624–631, 2008.

[7] O. Kovalenko, D. Winkler, M. Kalinowski, E. Serral, and S. Biffl, "Engineering process improvement in heterogeneous multi-disciplinary environments with defect causal analysis," *Proceedings of the 21st EuroSPI conference*, pp. 73–85, 2014.

[8] S. Suess, S. Magnus, M. Thron, Z. H., U. Odefey, A. Strahilov, A. Klodwaski, and T. Baer, "Test methodology for virtual commissioning based on behaviour simulation of production systems," 2016, pp. 1–8.

[9] B. Brandenbourger, M. Vathoopan, and A. Zoitl, "Engineering of Automation Systems using a Metamodel implemented in AutomationML," in *International Conference on Industrial Informatics (INDIN)*. IEEE, 2016.

[10] B. Hummel, "Integrated Behavior Modeling of Space-Intensive Mechatronic Systems," *Institut für Informatik der Technischen Universität München*, 2010.

[11] L. Berardinelli, E. Maetzler, T. Mayerhofer, and M. Wimmer, "Integrating Performance Modeling in Industrial Automation through AutomationML and PMIF," in *International Conference on Industrial Informatics (INDIN)*. IEEE, 2016.

[12] C. Legat, J. Mund, A. Campetelli, G. Hackenberg, J. Folmer, D. Schuetz, M. Broy, and B. Vogel-Heuser, "Interface Behavior Modeling for Automatic Verification of Industrial Automation Systems' Functional Conformance," *at - Automatisierungstechnik*, pp. 815–825, 2015.

[13] A. Shah, A. Kerzhner, D. Schaefer, and C. J. Paredis, "Multi-View Modeling to Support Embedded Systems Engineering in SysML," *Hutchison, Kanade et al. (Ed.) Graph Transformations and Model-Driven Engineering*, vol. 5765, pp. 580–601, 2010.

[14] A. Lueder, L. Hundt, and A. Keibel, "Description of manufacturing processing using AutomationML," *Proc. of 17th IEEE Int. Conf. on Emerging Technolgies and Factory Automation (ETFA)*, pp. 1–8, 2012.

[15] O. M. O. F. (MOF), "Core Specification v 2.0," *http://www.omg.org/docs/formal/06-01-01.pdf*, 2006.

[16] V. Vyatkin, H.-M. Hanisch, C. Pang, and C.-H. Yang, "Closed-Loop Modeling in Future Automation System Engineering and Validation," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 1, pp. 815–825, 2009.