Generating metamodel-based descriptions of automation components in AutomationML

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

***Abstract*—The variability of modern automation systems is getting the key factor in determining the efficiency in manufacturing. The modularity of the employed automation components contributes significantly to the level of variability by enabling multiple compositions of automation components. In order to enable a high modularity, each automation component should be modeled as an integrated mechatronic model containing information from different disciplines. This paper describes a recursive concept allowing the structured generation of manufacturer-specific models of automation components. The concept is based on a metamodel in AutomationML and implemented in a prototypical tool for evaluation purposes.**

I. Introduction

The manufacturing industry currently experiences drastic changes leading mass production towards customer-specific products [1]. In order to satisfy the customers' requirements, a high range of personalized products with low production volumes is pursued. The high level of product diversity leads compulsorily to adaptable automation systems which can be rearranged to pending productions requirements with reduced time effort, therefore reducing the general downtime of a system, and leading in consequence to a higher availability and productivity of the automation systems. By using fully integrated and autonomous automation components [2], the resulting cyber-physical production system (CPPS) shows a better adaptability [3]. These autonomous components support Plug & Produce [4] and allow a fast and cost-effective [4] [5] rearrangement of the automation plant.

One key factor for Plug & Produce is a standardized description of the component containing information, for example, about its capabilities and how they are controlled. Modeling components in a standardized way facilitates their controlling and cross-vendor interchangeability. AutomationML (AML) offers a way of modeling automation components such that the models are stored and exchanged through different parties involved in the engineering process of an automation plant [6]. By combining AML with a model driven engineering (MDE) approach, an efficient and seamless engineering process is enabled. Furthermore, the responsibilities for modeling tasks are clearly allocated on specific phases in the engineering process. However, the crucial point with MDE is the generation and use of valid models. A gap in the application of the models between the different partners involved in the engineering process occurs and needs to be closed.

This paper introduces a recursive concept for generating manufacturer-specific models of automation components stored in AutomationML. The concept has been implemented in a tool called *AML Component Generator* for demonstration and evaluation purposes. A metamodel presented in [7] acts as a standardized engineering data model and provides basic information used for the generation of manufacturer-specific models. The concept closes the gap in component modeling between component and plant manufacturers.

The rest of the paper is organized as follows. Section II gives an overview of available work in the field of component-based engineering and application of AML in the industrial automation domain. Section III seizes on an engineering approach using a metamodel which contains integrated mechatronic models. The concept for generating model-based descriptions of automation components is presented in IV whereas the implementation of the concept is shown in section V. Section VI concludes the paper giving overall results and future work.

II. Related Work

Vyatkin et al. [8] put forward state of the art requirements and methods for software intensive systems in the industrial automation domain. For handling the complexity of systems in general, model based engineering is recommended. Further refining this, a component-based approach is suggested for improving re-usability and design patterns are advocated for handling variance. Vogel-Heuser et al. [9] analyze this from the industry's point of view and describe modularity, reuse, and variant management as the important requirements. Feldman et al. [10] take this further to apply this concept to hardware and software components for increasing re-usability and modularity. Considering all these aspects, a standardized model-based engineering approach is presented in our previous work [7]. A metamodel containing component models is proposed for integrating the plant engineering process. A base structure of the metamodel model is made available in the standard data exchange format in the industrial automation domain AutomationML [11] [12], which acts as the base engineering data model for all the disciplines involved. For realizing a component model which brings inter-disciplinary interaction and collaboration, a mechatronic component model integrated with a central behavior model is introduced in [13] extending the approach described in [7]. Here the behavior model acts as the core of the model which is familiar for

978-1-5090-5320-9/17/$31.00 ©2017 IEEE

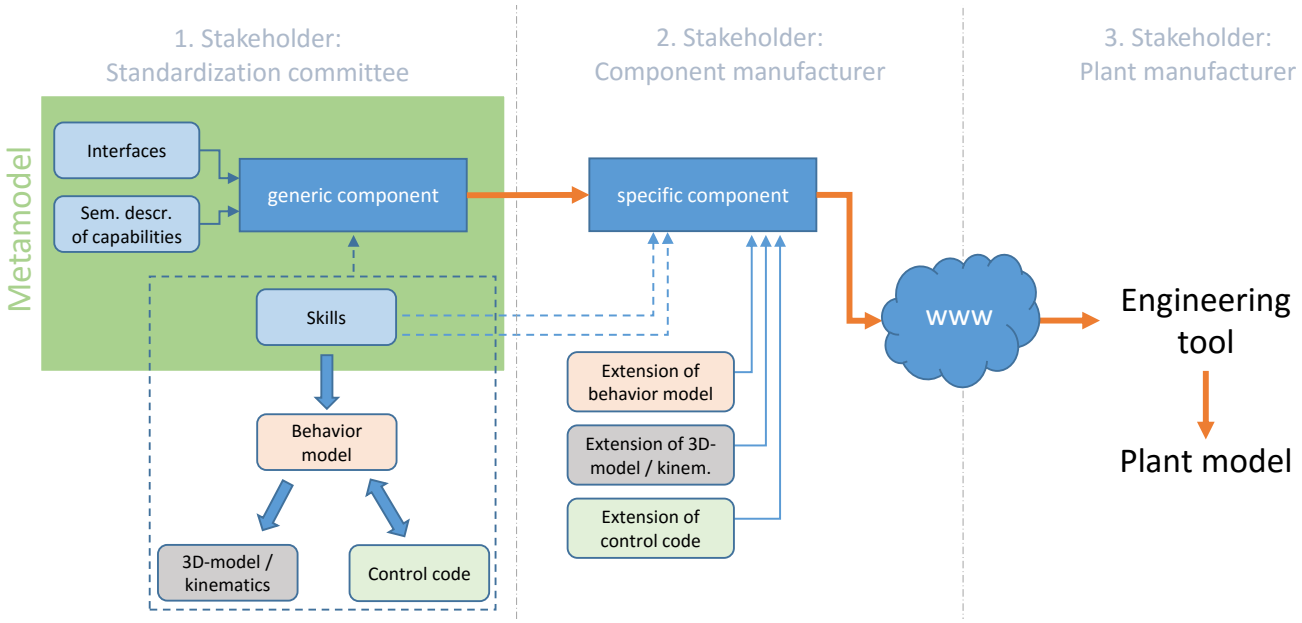

Fig. 1. Engineering steps using the object-oriented approach presented in [7]

each discipline involved in the engineering process, since the behavior model represents the functionality of the component in an abstract way. Vathoopan et al. [14] propose that this aggregated engineering data could be also applicable in the operation stage of the component, such as for raising error during operation and for creating simulation. The proposed metamodel including component models requires multidisciplinary data such as control code, 3D data model, simulation data etc. to be integrated within it. Thus component models implemented in AML are quite complex with multidisciplinary data and require a lot of external and internal links to realize the interdisciplinary interaction and dependencies. There is a range of work in literature analyzing AML models and editors for implementing aspects such as requirement engineering [15], project configuration, virtual commissioning [16], simulations [17] etc., but using the AML Editor for a complex model that brings the multidisciplinary aspects has not been analyzed so far.

III. Principles for a Standardized Model-Based Component Design

In [7] a tripartite engineering approach has been presented. The model-based approach concentrates on the creation of standardized models of automation components to support interchangeability of similar components in a manufacturing plant. This approach partitions the modeling tasks on different stakeholders, depending on their expertise. Therefore, clear scopes for each stakeholder can be derived.

Figure 1 depicts the modeling process and the three main stakeholders involved in the engineering of automation plants. The first stakeholder is a standardization committee being responsible for the creation of a metamodel. It consists of generic component models, which are universally valid, manufacturer independent, highly abstracted, and cross-domain models of automation components. These generic components are later used to generate manufacturer-specific models. The metamodel also contains libraries defining standardized interfaces, standardized capabilities (so-called skills), and further information such as semantic descriptions of the capabilities and physical attachment constraints. Skills can be interpreted as service-definitions and are linked through the behavior model to the corresponding sections in the control-code, the service-implementations. All these elements together with a component-specific behavior model, a 3D-model with kinematics, and a template for the control code are used to create an integrated mechatronic model of a generic component. These artifacts from different domains are interlinked in order to obtain cross-domain interdependencies [13]. The proposed concept of the standardization committee and standardized generic component models are a first proof of concept and try to resolve the standardization deadlock described in [18] by making the first step towards a standardized model.

The second modeling step is held by component manufacturers. This stakeholder takes care of the creation of manufacturer-specific component models. For supporting the interchangeability through standardized component models, manufacturer-specific component models are always based on a generic component model. Specific components contain additional manufacturer-specific information in form of extensions of the given data by the generic component. Additional domain-specific information such as electrical or pneumatic data can also be added at this step in modeling. By following this modeling approach, each manufacturer-specific component fulfills the minimum requirements given by a generic component and defined by the standardization committee. Therefore, the interchangeability of components originating from different manufacturers but using the same generic component as source-element is ensured. The second stakeholder also has the possibility to create own libraries of specific components which again can recursively be used as a

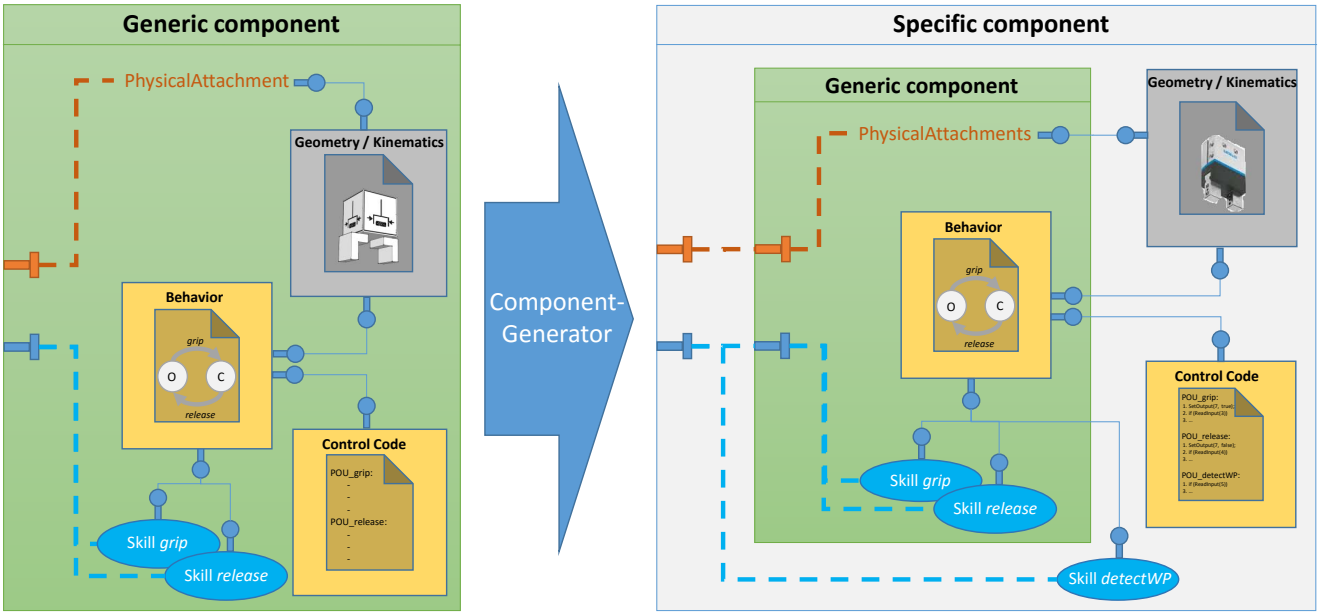

Fig. 2. Procedure of manufacturer for creating specific components based on premodeled generic components

source for modeling even more specific components.

After completing the modeling of all the manufacturer-specific information, the specific component is provided to the last stakeholder, namely plant manufacturers. These plant manufacturers use engineering tools to create a plant model consisting of different domain-specific models. In the engineering approach presented in [7], the plant manufacturer picks the components from different manufacturers. As long as each manufacturer has modeled its specific components relying on generic components, an interchangeability of similar components is supported. In order to get the specific models into the existing tools in industry, another supporting application presented in [19] is used. This application performs a model transformation from AML into the target tool-specific proprietary format and imports the information into the engineering tool.

The work presented in this document concentrates on the modeling steps between the first two stakeholders and the work performed by the second stakeholder. The presented approach supports the closing of the gap between the stakeholders in order to enable seamless engineering. The introduction of a new tool supporting the described engineering approach ensures the correct modeling of manufacturer-specific components.

IV. Concept for Generating Manufacturer-Specific Models of Automation Components

In order to create a manufacturer-specific model of an automation component, the model of a generic component needs to be extended. This section presents the modeling steps and exemplifies them with the help of a gripper component as depicted in figure 2. Figure 2 is a zoom-in of the arrow between the first two stakeholders and the extension happening at the second stakeholder as depicted in figure 1.

Within a generic component, the single domain-specific models such as the behavior model, the 3D-model with kinematics, the skills, and the skeleton of the control code are interlinked with each other and together represent an integrated mechatronic model as presented in [13]. The functionality of the modeled component can only be accessed by skills, such that a functional engineering approach is ensured [20] and abstract I/O-signals are encapsulated in the control code. The behavior model acts as a central entity in the integrated mechatronic model. Called skills or input signals from the periphery trigger changes in the behavior model. In turn, changes in the behavior model trigger kinematics of the 3D-model and call program organization units (POU) or function blocks (FB) of the control code. In this way, an automation component is described from the perspective of different domains which are interlinked with each other.

In the left part of figure 2, a generic gripper component is modeled. It consists of two skills *grip* and *release* which can be triggered by an external event (see lower interface on left side). The behavior model which is represented here as a finite state machine (FSM) has two states *open* and *closed* and the transitions *grip* and *release* corresponding to the two skills. The transitions are also linked to the kinematics *openJaws* and *closeJaws* of the 3D-model, whereas the 3D-model offers a defined *PhysicalAttachment* allowing the physical fixation of the automation component to some other device. Furthermore, each transition is linked to a POU or FB in the control code, which, at a later modeling stage, will contain the actual source code for controlling the hardware. All these interlinked domain-specific models represent the integrated mechatronic model of a generic component.

The presented concept for generating manufacturer-specific models of automation components uses the generic component model and extends the model with manufacturer-specific information. This model-based approach ensures the validity

of the manufacturer-specific component models but arises the problem of consistency between generic and specific models. Therefore, restrictions concerning altering the generic model exist such as:

- The content of existing POUs/FBs can be adapted, but not deleted. Additional POUs/FBs can be added.
- Cross-domain links between domain-specific artifacts can be altered but not deleted.
- The behavior model of the generic component can only be extended. Specific components manifest always the basic behavior defined in the generic component (see fig. 3).

Another advantage of the model-based approach consists in drastically reducing the engineering effort of the second stakeholder as only the extensions respective to the generic component need to be modeled explicitly. In the given example, the specific component (right part of fig. 2) is a composition of a generic component with some adaptations. The generic 3D-model is replaced by a more precise manufacturer-specific 3D-model, whereas the existing links to the behavior model and PhysicalAttachment remain unchanged. The link between the behavior model and the control code also remains unchanged, as the skeleton of the source code from the generic component containing empty POUs or FBs is only filled with manufacturer-specific information. This modeling step corresponds to the phase where the manufacturer contributes its intellectual property (IP) and exact knowledge of the automation component to the integrated mechatronic model. The skills defined in the generic component (*grip*, *release*) will remain in the specific component. Depending on the requirements, additional skills and corresponding POUs or FBs can be added to the specific component (e.g. *detectWorkPiece*) and linked to the behavior model. The behavior model of the generic component is taken over in the specific component and extended as required, but without altering and therefore affecting the original behavior model. Figure 3 shows an approach of the extension of a behavior model without altering the basic behavior. Additional limit or detection sensors and their digital signals are incorporated in the behavior model as *TriggerConditions* known from extended finite state machines [21]. Another approach of extending the behavior model is the introduction of additional states within a transition [22]. In both cases the basic behavior model remains unaltered and therefore ensures consistency and the correct behavior defined in the generic component by the standardization committee. A more precise analysis of behavior inheritance is beyond the scope of this paper.

Basically, the generation of a manufacturer-specific component model consists of replacing the 3D-model, extending the control code and if required the behavior model and skills, and especially preserving existing links from the generic component to ensure the integrity and consistency of the new integrated model.

V. Implementation of the Concept

Following the engineering approach presented in section IV, each manufacturer-specific model must be derived from a

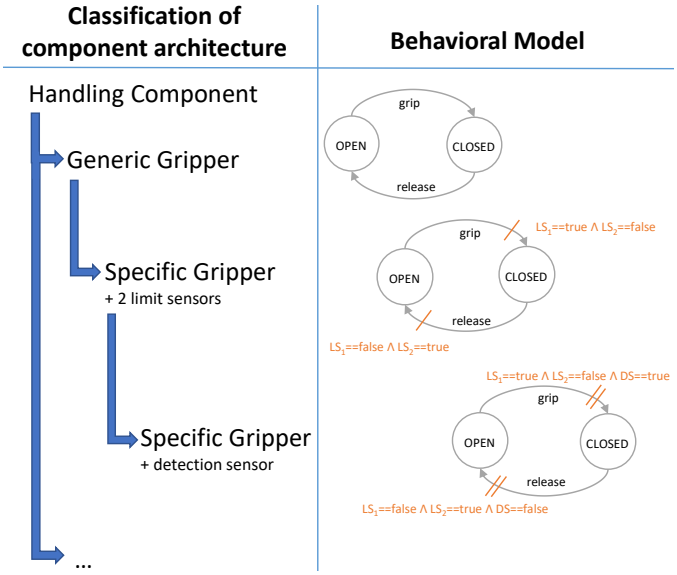

Fig. 3. Extension of a behavior model with additional sensors

generic component stored in the metamodel provided by the first stakeholder. In [7] a metamodel implemented in AutomationML is presented. AML is a a neutral, XML-based data format for storing and exchanging plant engineering information [11]. The AML association provides a tool called *AutomationML Editor* [6] which allows the visualization, creation, and editing of AutomationML/CAEX files. The *AutomationML Editor* can be used for performing the modeling steps of section IV for the creation of manufacturer-specific component models, but the complexity of the necessary modeling steps is too high and error-prone. Therefore, a new tool called *AML Component Generator* has been developed which closes the gap between the standardization committee and component manufacturers and supports component manufacturers during modeling specific components as presented in section IV. The employment of the *AML Component Generator* is less error-prone as the user is logically lead through the modeling steps and mandatory modeling tasks are performed automatically. Besides the higher intuitiveness given by the GUI (see fig. 4), the tool embraces different domains into the final integrated mechatronic model stored as a manufacturer-specific model. Nevertheless, the *AML Component Generator* uses the *AutomationML Engine* [23], a framework representing the CAEX data model in form of a C# class structure and containing classes and methods for manipulating CAEX objects such as classes and instances.

The *AML Component Generator* is designed in such a way that, first of all, the metamodel must be loaded (see fig. 4-1). The metamodel contains all the entities needed for modeling and defined by the standardization committee such as generic components, skills, and roles. When selecting a generic component, the corresponding classification symbol following the approach presented in [24] is shown. Furthermore, the skills (fig. 4-3) and roles (fig. 4-4) assigned to the selected generic component are displayed and can not be altered. Roles represent in this case the semantic description of

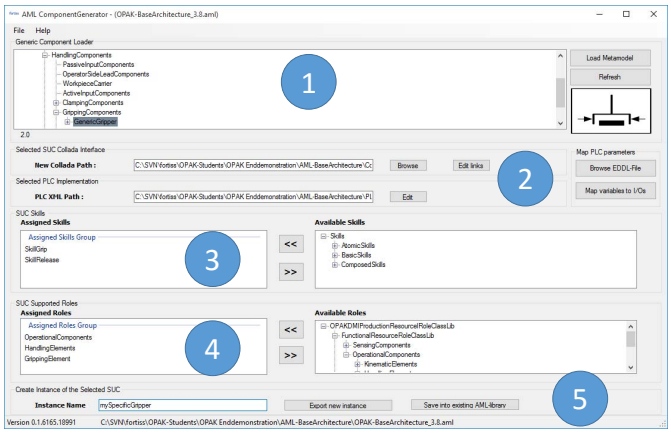

Fig. 4. Screenshot of *AML Component Generator* tool

the capabilities and can contain attributes according to the IEC 62424 [25] [26]. The next modeling step consists in choosing the 3D-model of the specific component and editing the given structure of the control code (fig. 4-2). Since a new 3D-model has been defined, existing links (e.g. *PhysicalAttachment*, links between behavior model and kinematik stored in the 3D-model) originating from the generic component are broken. Therefore, the broken internal links have to be repaired by defining the missing endpoint in the new 3D-model by using the *Edit links*-Button. For editing and populating the control code, a separate editor is opened which displays the skeleton of the control code given by the selected generic component. Afterwards, additional skills can be assigned to the new specific component as needed (fig. 4-3). Consequently, the corresponding POUs or FBs have to be created in the control code and linked with the skills. In fig. 4-4 additional roles can be assigned, if the capabilities of the new specific component surpasses the predefined roles of the generic component.

A. Saving the model information

After having parametrized the new specific component, the tool offers the possibility to either export or save the new model (fig. 4-5). Exporting is used for creating a ready-to-use package for the third stakeholder whereas saving allows the user to save the new model in an existing library for further posterior refinement.

The first use case in fig. 4-5 describes an immediate export of the specific component model into a self-contained AML-Container (fig. 5-UC 1.1). An AML-container [27] is a zip-file containing interlinked, component-relevant information modeled as a CAEX-file, the control code as PLCopen XML-file, the 3D-model as COLLADA-file, and the behavior model as another PLCopen XML-file. The AML-container is comparable to a dll-file, which can only be used by the third stakeholder through interfaces without having specific knowledge of the internal structure. In this way, the IP of the component manufacturer is protected and control code written by the component manufacturer is not changeable.

The other use case of the *AML Component Generator* describes saving the specific component as a System Unit Class (SUC) into a manufacturer-specific library (fig. 5-UC 2). The manufacturer-specific library has any internal structure required by the manufacturer, contains multiple component models, and can be interpreted as a class. Here, too, any manufacturer-specific IP can be stored as the library is not publicly available. Besides loading the metamodel, the *AML Component Generator* also supports loading such a manufacturer-specific library. Based on this library, additional modeling steps are performed as necessary and the final manufacturer-specific component model is exported into an AML-container (fig. 5-UC 1.2).

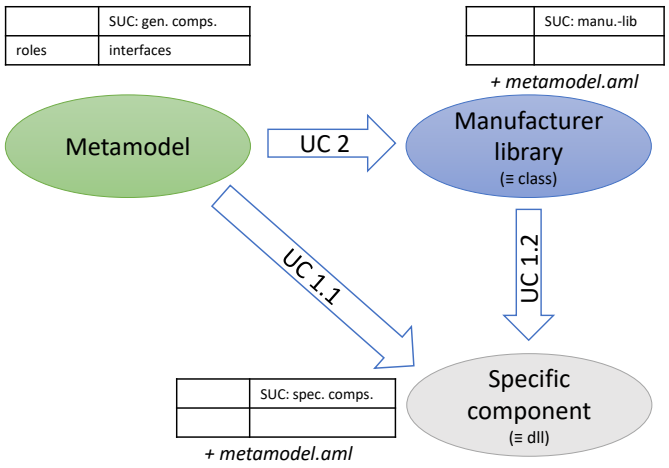

Fig. 5. Use cases (UC) for the tool *AML Component Generator*

At the end of UC 1.1 and UC 1.2, a manufacturer-specific component description arises which is provided to plant manufacturers (e.g. through the manufacturer's website (see fig. 1)). This description is imported by the plant manufacturer into its engineering tool (e.g. with the tool presented in [19]). At this point in engineering, the original metamodel comes into appearance again as the manufacturer-specific component contains references to AML-roles and AML-interfaces, which are only specified in the metamodel. Without the metamodel, the roles could not be interpreted properly.

B. Performing final checks

Before generating the AML-container or saving into a manufacturer-specific library, the *AML Component Generator* performs a couple of checks for ensuring the integrity and consistency of the new integrated component model. First of all, the structure of the control code is examined. All given POUs or FBs must be populated with some content, whereas the *AML Component Generator* can not check if the content is correct. The correctness is only ensured by the software engineer working for the component manufacturer. The second check concerns the links. All existing links in the generic component must be kept alive; either by maintaining the links (e.g. between skills and the behavior model from the generic component) or by connecting the links to a new object in the specific component. What applies in this regard, too, is that the *AML Component Generator* has only the possibility to check the existence of a valid link, not if the link is connected

to the right objects. In order to support the engineers creating specific component models for component manufacturers, it is advisable that the standardization committee uses meaningful terms for links or kinematics in the generic component, such that it is easily derivable to which internal object the links in the generic component were relating to.

The application of meaningful terms is also crucial for the semantic description of the capabilities stored in AML-roles and defined by the standardization committee. Those roles represent the common interface between an exporting and importing party [6]. In the presented case, the exporting party is the component manufacturer who exports a manufacturer-specific model of an automation component for a plant manufacturer, the importing party. Drath introduces in [18] four maturity levels, defining full proprietary semantics up to standard data models and describing in this way how many exchanging partners have the same understanding of a semantic description. The last check performed by the *AML Component Generator* concentrates on the correct usage of standardized roles in order to rise the maturity level.

VI. Conclusion and further work

In this paper, we have presented an approach for generating model-based descriptions of manufacturer-specific automation components in AutomationML. The implementation in form of the tool *AML Component Generator* facilitates seamless engineering between component manufacturers and plant manufacturers. By using a metamodel which holds generic information for automation components, the correct-by-construction composition of manufacturer-specific component models enables the interchangeability of the modeled components by preserving basic features. The implementation of the concept was successfully evaluated together with a component manufacturer and CODESYS Application Composer by using an additional tool [19] which transforms specific AML component models into the proprietary format of the target engineering tool.

The *AML Component Generator* is currently under investigation for extending some features such as creating an AML-Editor plugin, importing EDDL-configurations for getting available I/Os of the modeled component, or adding manufacturer-specific information in the form of a reference to an external document. Furthermore, the link-handling for new elements such as additional skills is missing. The redefinition of existing links originating from generic components must be optimized as well.

References

[1] H. Kühnle, "Post mass production paradigm trajectories," *Journal of Manufacturing Technology Management*, vol. 18, pp. 1022–1037, 2007.

[2] J. Sztipanovits, X. Koutsoukos, G. Karsai, N. Kottenstette, P. Antsaklis, V. Gupta, B. Goodwine, J. Baras, and S. Wang, "Software engineering in industrial automation: State-of-the-art review," *IEEE Transactions on Industrial Informatics*, vol. 100, no. 1, pp. 29–44, 2012.

[3] K. Thramboulidis, "Challenges in the Development of Mechatronic Systems: The Mechatronic Component," *IEEE Int. Conf. on Emerging Technologies & Factory Automation*, pp. 624–631, 2008.

[4] G. Reinhart, S. Krug, S. Huttner, Z. Mari, F. Riedelbauch, and M. Schlogel, "Automatic Conguration (Plug & Produce) of Industrial Ethernet Networks," *9th IEEE Int. Conf. on Industry Applications*, pp. 1–6, 2010.

[5] V. Hammerstingl and G. Reinhart, "Unied Plug&Produce Architecture for Automatic Integration of Field Devices in Industrial Environments," *IEEE Int. Conf. on Industrial Technology (ICIT)*, pp. 1956–1963, 2015.

[6] R. Drath and D. Weidemann, *Datenaustausch in der Anlagenplanung mit AutomationML: Integration von CAEX, PLCopen XML und COLLADA*. Springer-Verlag, 2009.

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

[8] V. Vyatkin, "Software engineering in industrial automation: State-of-the-art review," *IEEE Transactions on Industrial Informatics*, vol. 9, no. 3, pp. 1234–1249, 2013.

[9] B. Vogel-Heuser, C. Legat, J. Folmer, and S. Rösch, "Challenges of parallel evolution in production automation focusing on requirements specification and fault handling," *at-Automatisierungstechnik*, vol. 62, no. 11, pp. 758–770, 2014.

[10] S. Feldmann, C. Legat, and B. Vogel-Heuser, "An Analysis of Challenges and State of the Art for Modular Engineering in the Machine and Plant Manufacturing Domain," *IFAC-PapersOnLine*, vol. 48, no. 10, pp. 87–92, 2015.

[11] R. Drath, A. Lüder, J. Peschke, and L. Hundt, "AutomationML-the glue for seamless automation engineering," in *IEEE Int. Conf. on Emerging Technologies & Factory Automation (ETFA)*. IEEE, 2008, pp. 616–623.

[12] *Engineering data exchange format for use in industrial automation systems engineering*, IEC 62714 Std., 2014.

[13] B. Brandenbourger, M. Vathoopan, and A. Zoitl, "Behavior Modeling of Automation Components using cross-domain Interdependencies," *Int. Conf. Emerging Technologies & Factory Automation (ETFA)*, 2016.

[14] M. Vathoopan, B. Brandenbourger, and A. Zoitl, "A Human in the Loop Corrective Maintenance Methodology Using Cross Domain Engineering Data of Mechatronic Systems," in *Int. Conf. on Emerging Technologies & Factory Automation*. IEEE, 2016.

[15] N. Schmidt, A. Lüder, H. Steininger, and S. Biffl, "AutomationML for user requirements fulfillment related to engineering process efficiency," in *40th Annual Conf. of the IEEE Industrial Electronics Society (IECON)*. IEEE, 2014, pp. 4902–4908.

[16] E. Yemenicioglu, A. Strahilov, H. Zipper, and M. Riedl, "Improving the transition and modularity of the virtual commissioning workflow with AutomationML," *4th AutomationML User Conference,*, 2016.

[17] S. Süß, S. Magnus, M. Thron, H. Zipper, U. Odefey, V. Fäßler, A. Strahilov, A. Kłodowski, T. Bär, and C. Diedrich, "Test methodology for virtual commissioning based on behaviour simulation of production systems," in *21st Int. Conf. on Emerging Technologies & Factory Automation (ETFA)*. IEEE, 2016, pp. 1–9.

[18] R. Drath and M. Barth, "Concept for managing multiple semantics with AutomationML - Maturity level concept of semantic standardization," *IEEE Int. Conf. on Emerging Technologies & Factory Automation*, 2012.

[19] M. Vathoopan, B. Brandenbourger, A. George, and A. Zoitl, "Towards an Integrated Plant Engineering process Using a Data Conversion Tool for AutomationML," *IEEE Int. Conf. on Industrial Technology*, 2017.

[20] C. Sünder, A. Zoitl, and C. Dutzler, "Functional structure-based modelling of automation systems," *Int. Journal of Manufacturing Research*, 2006.

[21] J. Belzer, A. G. Holzman, and K. Allen, "Encyclopedia of Computer Science and Technology," *CRC Press*, vol. 25, p. 73, 1975.

[22] C. Klein, C. Prehofer, and B. Rumpe, "Feature Specification and Refinement with State Transition Diagrams," *Feature Interactions in Telecommunications and Distributed Systems IV*, pp. 284–297, 1997.

[23] R. Drath, "Let's talk AutomationML - What is the effort of AutomationML programming," *IEEE Int. Conf. on Emerging Technologies & Factory Automation (ETFA)*, 2012.

[24] T. Helbig, S. Henning, and J. Hoos, "Efficient engineering in special purpose machinery through automated control code synthesis based on a functional categorisation," *Machine Learning for Cyber Physical Systems and Industry 4.0 (ML4CPS)*, 2015.

[25] *Representation of process control engineering - Requests in P&I diagrams and data exchange between P&ID tools and PCE-CAE tools*, IEC 62424 Std., 2016.

[26] M. Schleipen and R. Drath, "Three-view-concept for modeling process or manufacturing plants with AutomationML," *Int. Conf. on Emerging Technologies & Factory Automation (ETFA)*, 2009.

[27] E. Yemicioglu and A. Lüder, "Implementation of an AutomationML-Interface in the digital factory simulation," *AML user conf.*, 2014.