Towards an Integrated Plant Engineering Process Using a Data Conversion Tool for AutomationML

Milan Vathoopan, Benjamin Brandenbourger, Amil George*, and Alois Zoitl
fortiss GmbH
Guerickestr. 25, 80805 Munich, Germany
{vathoopan, brandenbourger, zoitl}@fortiss.org
*Technical University of Munich
Boltzmannstr. 3, 85748 Garching, Germany
{george}@in.tum.de

***Abstract*—The development process of modern automated manufacturing plants requires a concurrent engineering process that integrates different engineering disciplines. Envisioning a concurrent development process, we propose an engineering process based on an AutomationML metamodel. The proposed metamodel contains standardized mechatronic component models, which acts as the base engineering data for the disciplines and the tools involved in the process. Since the tools operate on discipline-specific data structures, a data conversion tool is introduced which parses AutomationML mechatronic model and produces specific data interpretable by the tool employed. The benefits of this approach are evaluated by generating an automation project in CODESYS Application Composer, employing the metamodel and data conversion tool with the proposed engineering process.**

I. Introduction

Today's manufacturing companies are equipped with automation systems composed of complex mechatronic components [1]. The development of a modern automated plant necessitates interaction and collaboration of different disciplines involved [2] such as automation engineering, mechanical engineering, electrical engineering, software engineering, etc. Frequent updates and adaptations of a plant, required for meeting today's market requirements [3] demands coordination and collaboration of these disciplines even during operation stage. Different engineering tools are necessary for providing adequate support for these engineers to develop their solution. As these tools represent domain related information specific to the disciplines involved, they operate on different data structures and thus form a heterogeneous tool landscape [4]. Creating a seamless engineering process requires integration of this heterogeneous tool landscape. AutomationML (AML) [2] is an engineering data exchange format developed for the tools used in automation systems engineering process. AML provides a data format which can integrate various aspects of plant engineering such as requirement engineering, production engineering, process engineering, etc., keeping automation engineering as the core.

Considering mechatronic components based automation systems, a metamodel based standardized engineering process is proposed in our previous work [5]. The metamodel is implemented in AML envisioning the integration of a heterogeneous tool landscape and data exchange among the disciplines involved in the process. The metamodel contains mechatronic component models, which incorporates necessary engineering aspects required for a mechatronic component.

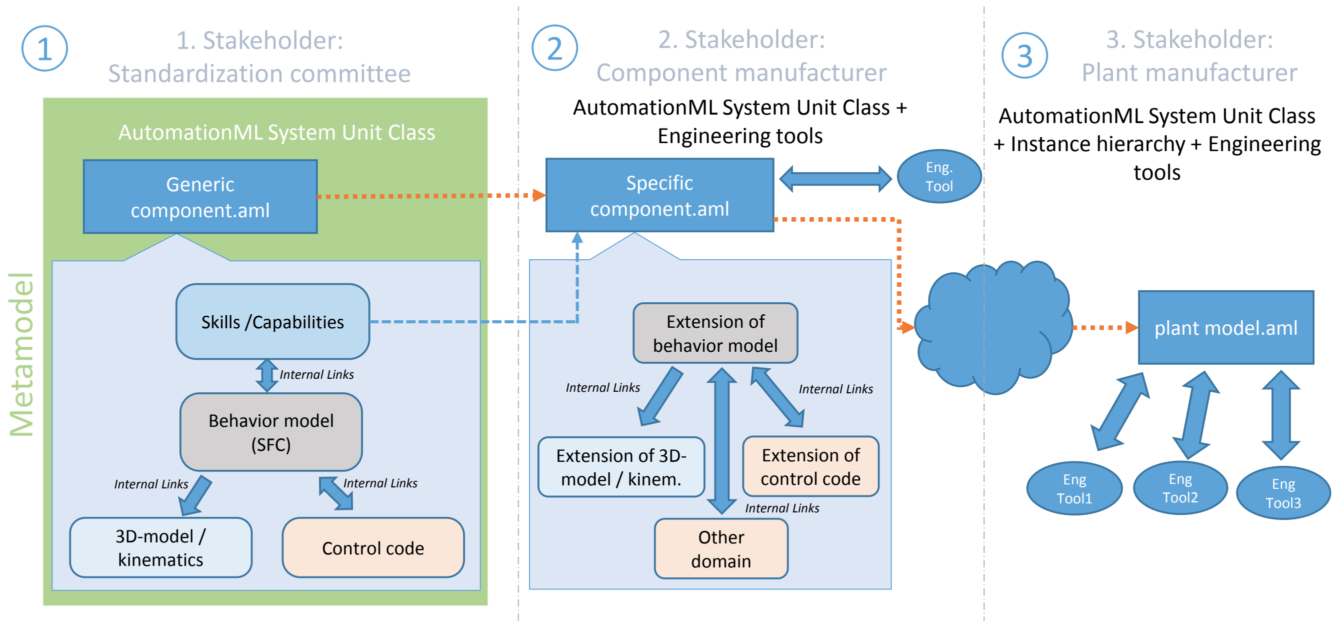
This work implements the concept of heterogeneous tool integration with AML presented in [5] by employing a conversion tool in the engineering process. The conversion tool examines the semantics used in AML metamodel, and map the data required by domain specific tools in a format interpretable by the tool specified. Thereby the conversion tool reduces development effort for the domain specific engineering tools. A prototypical implementation of the conversion tool is presented with an example of the engineering tool CODESYS Application Composer (AC)[1]. The advantages of employing AML component models and a conversion tool in the development process is evaluated with AC and the standardized engineering process. AML aspects necessary for the tool AC are only considered in this paper and other detailed aspects of AML are out of the scope of this paper.

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 introduces our envisioned engineering approach using mechatronic component models. The concept for tool integration is presented in Section IV. The implementation and evaluation of the concept with the engineering tool AC is presented in Section V. Section VI concludes the paper giving overall results and future scenarios.

II. Related Work

Modern automation systems are software intensive and their functionality is largely determined by the software executed on embedded computers. Software intensive engineering approaches are in development in the industrial automation domain [6]. Several methods have been analyzed for reducing the complexity of engineering in automation systems. Vogel-Hauser et al. [3] put forward modularity, reuse and variant management as the key requirements for a software intensive

[1]https://www.codesys.com/products/codesys-engineering/application-composer.html



1
1. Stakeholder: Standardization committee
AutomationML System Unit Class
Generic component.aml
Metamodel
Skills /Capabilities
Internal Links
Behavior model (SFC)
Internal Links
Internal Links
3D-model / kinematics
Control code
2
2. Stakeholder: Component manufacturer
AutomationML System Unit Class + Engineering tools
Specific component.aml
Eng. Tool
Extension of behavior model
Internal Links
Internal Links
Extension of 3D-model / kinem.
Extension of control code
Internal Links
Other domain
3
3. Stakeholder: Plant manufacturer
AutomationML System Unit Class + Instance hierarchy + Engineering tools
plant model.aml
Eng Tool1
Eng Tool2
Eng Tool3

Fig. 1. Component based engineering approach using AutomationML

development methods in automation engineering domain. For ensuring reuse of hardware and software components Feldman et al. [7] propose a modular development, where a module is an aggregation of components from different disciplines. VDI defines guidelines for engineering organization improvements with certain devices, flow charts, and operational procedures [8]. Analyzing further requirements such as enabling interdisciplinary interaction [9] and continuous data management [10], avoiding inconsistencies and redundancies [11] etc., a metamodel based standardized engineering approach is introduced in [5]. The concept of the metamodel presented is valid through different stages of a plant engineering process starting from conception to normal operation of a plant. The envisioned aspects are elaborated by using the characteristics of AML in an abstract way. Extending this approach a functionally integrated mechatronic component model and corresponding implementation in AML is presented in [12], [13]. Here a behavior model is used for functionally integrating multi-disciplinary aspects of a component. AML has been explored by many people for showing its modeling aspects [14], [15] and representing multi-disciplinary aspects [16], [17]. Süß et al. [18] propose AML data model for virtual commissioning in automotive plant engineering. Lüder et al. [19] present representing graph-based structures in AML for loss less exchange of production related data. Schleipen et. al apply AML for representing skills of production plants in [20]. In this paper, we combine modeling and application aspects of AML shown in above studies with our component based plant engineering process and evaluate AML for a discipline specific tool integration.

III. Component Models for a Standardized Engineering Process

A standardized engineering process as presented in [5] requires generic component models, which further allows development of manufacturer specific components and complete plant models. The overall engineering process based on component models is shown in Fig. 1. We propose a standardization committee (first stakeholder), who is responsible for the creation of the metamodel containing generic component models. Generic components are universally valid, manufacturer independent, highly abstract, and cross-domain models of automation components. The generic component integrates aspects from different domains such as mechanical, electrical, and software. A basic model of a generic component composed of domain-specific artifacts as presented in [13], is shown in Fig. 1 (1). This model contains 3D-data with kinematics, a generic behavior, standardized capabilities (so-called 'skills'), and a minimal structure of the control code. A component manufacturer holding the role of the second stakeholder builds manufacturer specific components based on the generic component models. For this purpose manufacturer-specific information is added to the available generic models until all specific aspects of the component are modeled (see Fig. 1 (2)). Thus a specific component implements all detailed aspects applicable to a manufacturer specific component. It contains an extended version of the behavior, 3D kinematic and control code and further artifacts when required. The proposed engineering approach ensures a component based development also for the third stakeholder, the plant manufacturer. The plant manufacturer works with completed models of components from the specific manufacturer or can also make use of generic components in order to create complete plant structures.

The engineering approach envisions interaction and collaboration of the disciplines and tools involved in the process, for which the component models are implemented in the standardized data exchange format AML. The standardization committee (the first stakeholder) creates an AML metamodel with categorized generic components [21]. The generic components are created as system unit classes, using different standards artifacts propounded by the AML consortium. This AML metamodel containing generic component models is then provided to the next stake holder-the component manufacturer. The component structure in the generic component model is realized using internal elements and internal links. The 3D model is integrated with an external COLLADA interface. An external PLCOpenXML interface tethers the skeleton of the control code and behavior model. The component manufacturer extends these artifacts and provide specific component system unit classes to the third stakeholder which is depicted in Fig. 1. The second stakeholder can import or export AML data models to some engineering tools, if necessary. The third stakeholder uses the specific component models available in the AML system unit classes to form the plant hierarchy in AML and also uses the same for integrating the heterogeneous tools involved in the process.

IV. Concept for Heterogeneous Tool Integration in Plant Engineering Process

The standardized engineering approach recommend to use the metamodel implemented in AML as the central data base. Different discipline specific details are described within the metamodel and the inter-dependencies are available for all the disciplines involved. To ensure model consistency, the component manufacturers always work based on the metamodel available from the standardization committee. A completed description of a specific component and the standard libraries are made available to the third stakeholder e.g. through the manufacturer's website as shown in Fig. 1. This ensures the third stakeholder is also working on the same standardized model, and the same model is used to integrate their different engineering groups and tools. Thus the three stakeholders can be better integrated into the engineering process with a collaborative engineering approach.

The AML metamodel in our approach contains components which exhibit real mechatronic characteristics such as behavior, control code and kinematics within a single component description given as a system unit class. In a plant engineering process several tools are employed and each tool understands only the data format specific to this tool. If a tool used in a specific domain (e.g. a PLC programming tool) has to understand the metamodel, the data irrelevant to this tool has to be filtered out from the component model and only the data relevant to the target tool has to be presented. Thus for integrating these tools with minimum engineering effort, we proposes a conversion tool. The conversion tool parses the AML component description and performs a data conversion, to produce a neutral data format such as XML, PLCopenXML, etc., supported by the specific tools. The tool analyzes the domain specific information and their dependency with other domains stored in AML and provides only data relevant and importable by the specific target tool. In the next stage the converted data is imported by the target tool in a tool specific manner, such that automatic generation of projects is

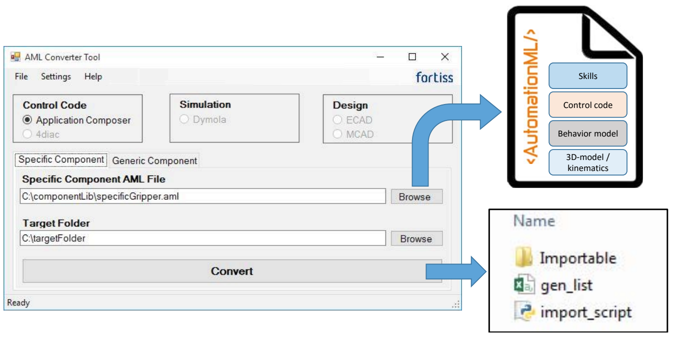

AML Converter Tool
Control Code: [✓ Application Composer] 4diac
Simulation: Dymola
Design: ECAD MCAD
Specific Component AML File: C:\componentLib\specificGripper.aml
Target Folder: C:\targetFolder
Ready

Fig. 2. Procedure for end-user applying the presented conversion tool

possible. A screen shot of the prototypical implementation of the conversion tool is shown in Fig. 2, the functional principle of it is elaborated subsequently with an example.

V. Evaluation with CODESYS Application Composer

The prototypical implementation of the converter tool is created for CODESYS Application Composer (AC) following the standardized engineering process. AC is a tool from Smart Software Solutions GmbH that provides a hardware-independent solution for developing control applications based on IEC 61131-3 [22]. Using AC, control applications can be composed by connecting existing modules and their parametrization. This model of development is efficient for application variants with recurring modules. A module typically represents a mechatronic component but also can be a software function. A module also contains sub-modules, parametrization, task assignment, I/O information, and visualization. AC also integrates a plugin called Depictor, which uses COLLADA models of components and provides a visual simulation for virtual commissioning. The characteristics of an AC module is applied in our approach for implementing control aspects of a mechatronic component. The proposed engineering process in Sec. III requires control modules for generic and specific components to compose a plant hierarchy and control application. The control module of a generic component implements abstract control characteristics of a generic component, and that of a specific component implements the specific control characteristics of a specific component.

As explained in Sec. III, the standardized engineering approach proposes AML as the core data model, which can be used for data exchange among different engineering domains and tools. The mechatronic model of a component implemented in AML contains an abstract implementation of the control code (implemented as a PLCopenXML interface), a CAD model (implemented as a COLLADA interface) and other discipline specific information. From the component model implemented in AML, aggregating certain aspects like control code, CAD data, and the hardware structure can give rise to a corresponding mechatronic control module implementation in AC. AC provides a python based scripting environment which provides APIs for creating different artifacts inside a module such as sub modules, parametrization, task assignment etc. So the tool processes the AML data, aggregates data required for creating a module in AC, converts

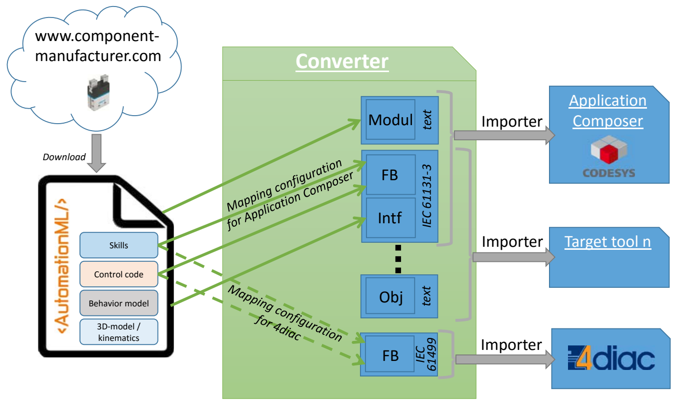


Fig. 3. Functional principle of the AutomationML data conversion tool

it into a data format which is interpretable for the python scripting environment provided by AC, and also creates a python script. The python script can be used in AC scripting interface to generate an AC project. A detailed explanation of this sequence is given below.

A. *Converter*

The AML file contains data in a neutral XML-based format, so the AML data can be parsed using regular XML parsers. But the converter used in our approach is based on the AML Engine, a C# framework representing the CAEX data model of AML [23]. AML Engine provides methods for parsing standard AML aspects such as role classes, interfaces etc. Though the present implementation is only for AC, a pool of target tools is conceivable with the principle applied. The AML file describing the component model is loaded and a target directory is also chosen. The target directory is for the converted data specific for the target tool selected.

Since AC is based on IEC 61131-3, AC provides PLCopenXML as a neutral format for importing and exporting projects and application related data. The method applied for converting data from AML to AC is thus based on the standard IEC 61131-3 as shown in Fig. 3. The converter specific for the AC generates data formats in the form of PLCopenXML for all the aspects except module description. Module description which corresponds to hardware related data is not specified in the IEC 61131-3 standard and is given in a proprietary description format. Since the engineering process followed is the same as proposed in Sec. III, the output of the converter tool depends on the entities required inside AC for creating a mechatronic control module.

1) AC entities for creating a mechatronic control module: The following entities are required for creating control aspects of a mechatronic component in AC as explained above.

Function Block (FB): Is used to supply one or more values during execution. FB can extend other FB and implement interfaces.

Interfaces: Are used to enforce certain properties on a FB

Module Declaration (MD): Is defined in AC using a proprietary description language, which enables programming hardware aspects of control. It contains information like metadata, I/O, slots, PLC mapping etc.

Image Pool: Is for giving a symbol or image representation to a mechatronic component

COLLADA Data: Is for visualizing a component and for virtual commissioning.

2) Mapping AML-information to AC: A component model as shown Fig. 3 in AML contains structural composition, control code implemented in the form of skill, and COLLADA interface and meta data such as type of the component, its properties etc., which has to be converted to the corresponding entity in AC. With each mechatronic component class in AML a MD is created, with a FB which implements the software, an interface to the skill which it implements and the path to the COLLADA document. An Image pool required for holding the symbol is also created.

The MD, which is created in proprietary description format corresponds to a control hardware in AC and contains the following artifacts:

- The header data and system unit class attributes of a component in AML such as its category, manufacturer details and basic descriptions mapped to metadata section of the MD.
- Each technology attribute of a component model in AML mapped to Param section in the MD
- Each digital output of an AML component model mapped to Section OUTPUT in SEC IO of MD
- Each digital input of an AML component model mapped to Section INPUT in SEC IO of the MD
- Additionally PLC mapping information from AML mapped to SEC DeviceGenerator

The software functionality of a control module is implemented with FBs, as done inside any IEC 61131-3 standard programs. FBs are thus created as PLCopenXML files inside importable directory. The component data from AML mapped to a FB within the AC:

- Attached to each component in AML are the skills, each skill is mapped to a function within the FB. The control code attached to the skill is used as the function definition.
- Corresponding to each digital input and output, variables are defined in the FB
- Implementation of interfaces
- The behavior model mapped as a FB, along with the path to the COLLADA external interface in AML model implements visualization. In the current implementation, the behavior model is used only for creating simulation of a component with CODESYS Depictor.

Additionally the symbol attribute from AML component model is mapped to the image pool.

The converted files are placed in the directory `importable` as shown in Fig. 2. And a python script specific for the AC is also produced named as `import_script.py`, which can be executed in AC.

B. *Importer in CODESYS Application Composer*

The presented importer tool is an engineering tool specific application that uses the APIs provided by AC to create entities

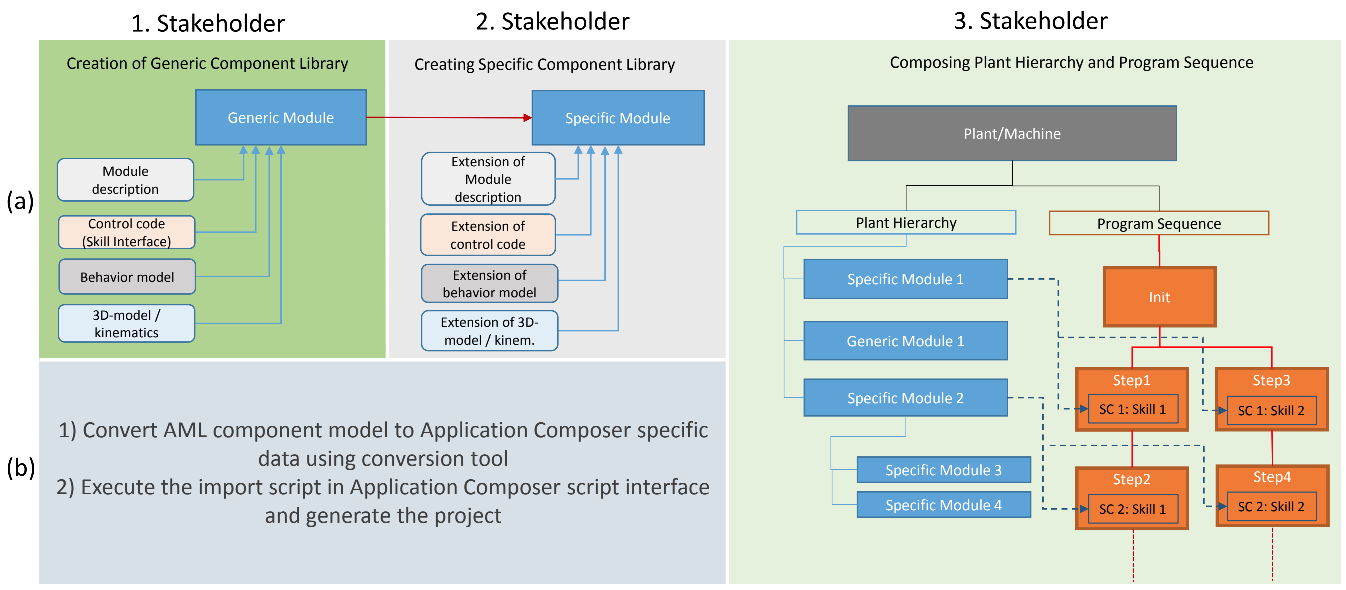

1. Stakeholder
2. Stakeholder
3. Stakeholder
(a)
Creation of Generic Component Library
Generic Module
Module description
Control code (Skill Interface)
Behavior model
3D-model / kinematics
Creating Specific Component Library
Specific Module
Extension of Module description
Extension of control code
Extension of behavior model
Extension of 3D-model / kinem.
Composing Plant Hierarchy and Program Sequence
Plant/Machine
Plant Hierarchy
Program Sequence
Specific Module 1
Generic Module 1
Specific Module 2
Specific Module 3
Specific Module 4
Init
Step1
SC 1: Skill 1
Step3
SC 1: Skill 2
Step2
SC 2: Skill 1
Step4
SC 2: Skill 2
(b)
1) Convert AML component model to Application Composer specific data using conversion tool
2) Execute the import script in Application Composer script interface and generate the project

Fig. 4. Building a control application (a) Using Aplication Composer Only, (b) Using Application Composer + AutomationML conversion tool.

within AC. As the approach uses a separate converter tool that takes care of transforming the AML model to a format that is suitable for the target engineering tool, the job of the importer is reduced to setting up a project structure, creating the corresponding objects, and finally checking for any errors or inconsistencies. The python script interface accepts inputs in the form of PLCopenXML and also text format. Accordingly a complete AC project is created using the data from the directory `importable`. A generic component inside AC contains only a generic declaration of the hardware module with corresponding FB which implements this module. It is also provided with corresponding interfaces for skills, which implements the control logic in a generic way. The metadata components are implemented with parameters of a MD. A specific component inherits from the generic components and implements specific functionality of a specific vendor. This implements the actual control code, extends hardware structure, properties etc. As mentioned above the `import_script.py` which is an output of the converter tool can be executed in the scripting interface of the AC. This script uses data available in the directory `importable` to generate a project containing the same classification of completed mechatronic components in AML system unit classes.

C. Evaluation

An evaluation of the approach, using AML metamodel, conversion tool and the standardized engineering process with AC is presented in Fig. 4. The development effort required, applying the standard approach in AC to create a component based engineering process is represented in Fig. 4(a). And Fig. 4(b) represents the simplified engineering process using AML data conversion tool along with AC. As depicted in Fig. 4(a), when using AC exclusively, every object has to be created from scratch. The same process as depicted in Fig. 1 has to be followed, which starts from creating generic control modules. Those generic modules have to be made available as a library so that specific control modules can be created inheriting from the generic modules. And finally generic and specific modules are used to compose (drag and drop) the complete plant hierarchy and program sequence. Using AML conversion tool the first 2 steps are eliminated using the script interface of AC, which automatically generate generic and specific control modules and related entities in AC. The generated specific and generic modules can be dragged and dropped to create the plant hierarchy and program sequence. Thus using this approach, the expertise is required only for the component manufacturer (second stakeholder) who provides encapsulated component descriptions in AML with standardized interfaces. The plant manufacturer (third stakeholder) uses the standardized component descriptions to model the plant, without in-depth engineering knowledge of the component.

As shown in Fig. 3, the principle applied in case of AC is reusable for other tools based on the standard IEC 61131-3. The PLCopenXML output of the converter tool can be reused for other tools in principle. For tools based on other standards such as 4diac [24], [25] based on IEC 61999, the approach presented in [26], [22] is applicable.

VI. Conclusion and Outlook

Nowadays seamless engineering of automation plants is a requirement for versatile manufacturing systems. The presented approach closes the gap between component manufacturers and plant manufacturers (engineers) by presenting a possibility to convert and import data stored in AML into a specific target tool. Automation components are modeled in AML by component manufacturers such that the recurring development effort of plant engineers is reduced. This approach envisages the plant engineers can concentrate on the engineering process and avoids unnecessary work of modeling the resources used. For avoiding inconsistencies and errors during development of a specific AML component model out of a generic component model as given in Sec. III, a component generator tool [27] is being developed in parallel to the conversion tool.

An example of the proposed engineering approach applied on CODESYS AC is shown in this contribution. A user experience with our engineering approach is not yet completed and seen as a future work. In order to increase user-friendliness and user-acceptance, the creation of a plugin for AC is also envisioned. At the moment, the presented approach supports only the conversion and import of automation component models. Integration of behavior model and CODESYS Depictor within module declaration is expected as an added feature for more flexibility of the whole process. Another useful feature envisaged is a possibility to import complete plant models stored in the instance hierarchy in AML into AC. But this necessitates also the export of a plant model from AC into an AML-file bringing interoperability of the model among different domains involved in the engineering process. Supporting other target tools such as IEC 61499 standard implementation 4diac is seen as an immediate future work, such that creation of distributed control code out of AML-descriptions is enabled. The proposed standardization committee, standardized component models and data conversion tool are first proof of concepts. We are planning to bring this concepts to the AML association for review and standardization.

ACKNOWLEDGMENT

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

REFERENCES

[1] P. Stich and G. Reinhart, "Mechatronic Sketching of Manufacturing Systems Using Physically Based Models: A Novel Approach for Simulation Based Systems Engineering," in *IEEE Symp. on Industrial Electronics & Applications(IndIn)*, 2013.

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

[3] B. Vogel-Heuser, C. Diedrich, A. Fay, S. Jeschke, S. Kowalewski, M. Wollschlaeger *et al.*, "Challenges for software engineering in automation," *Journal of Software Engineering and Applications*, 2014.

[4] S. Biffl, A. Schatten, and A. Zoitl, "Integration of heterogeneous engineering environments for the automation systems lifecycle," in *2009 7th IEEE International Conference on Industrial Informatics*. IEEE, 2009, pp. 576–581.

[5] B. Brandenbourger, M. Vathoopan, and A. Zoitl, "Engineering of Automation Systems Using a Metamodel Implemented in AutomationML," in *IEEE Conf. on Industrial Informatics (IndIn)*. IEEE, 2016.

[6] 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.

[7] 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.

[8] VDI/VDE, "Guideline 3695, engineering of automation systems, section 1," 2010.

[9] F. Himmler, "Function based engineering with automationml - towards better standardization and seamless process integration in plant engineering," *Wirtschaftsinformatik Proceedings*, no. 2, 2015.

[10] O. Graeser, "Durchgängige datenhaltung schafft wettbewerbsvorteil," *Maschinenmarkt*, vol. 30, p. 31, 2014.

[11] K. Thramboulidis, "Overcoming mechatronic design challenges: the 3+1 sysml-view model," *Computing Science and Technology International Journal*, vol. 1, no. 1, 2013.

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

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

[14] L. Berardinelli, S. Biffl, E. Maetzler, T. Mayerhofer, and M. Wimmer, "Model-based co-evolution of production systems and their libraries with automationml," in *Emerging Technologies & Factory Automation (ETFA), 2015 IEEE 20th Conference on*. IEEE, 2015, pp. 1–8.

[15] S. Biffl, O. Kovalenko, A. Lüder, N. Schmidt, and R. Rosendahl, "Semantic mapping support in automationml," in *Proceedings of the 2014 IEEE Emerging Technology and Factory Automation (ETFA)*. IEEE, 2014, pp. 1–4.

[16] A. Lüder, N. Schmidt, R. Rosendahl, and M. John, "Integrating different information types within automationml," in *Proceedings of the 2014 IEEE Emerging Technology and Factory Automation (ETFA)*. IEEE, 2014, pp. 1–5.

[17] S. Faltinski, O. Niggemann, N. Moriz, and A. Mankowski, "Automationml: From data exchange to system planning and simulation," in *Industrial Technology (ICIT), 2012 IEEE International Conference on*. IEEE, 2012, pp. 378–383.

[18] S. Süß, S. Magnusy, M. Throny, H. Zippery, U. Odefeyz, V. Fäßlerz, A. Strahilovx, A. Kodowskik, T. Bär, and C. Diedrich, "Test methodology for virtual commissioning based on behaviour simulation of production systems," in *2016 IEEE 20th Conference on Emerging Technologies & Factory Automation (ETFA)*. IEEE, 2016, pp. 1–9.

[19] A. Lüder, N. Schmidt, and S. Helgermann, "Lossless exchange of graph based structure information of production systems by automationml," in *2013 IEEE 18th Conference on Emerging Technologies & Factory Automation (ETFA)*. IEEE, 2013, pp. 1–4.

[20] M. Schleipen, J. Pfrommer, K. Aleksandrov, D. Stogl, S. Escaida, J. Beyerer, and B. Hein, "Automationml to describe skills of production plants based on the ppr concept," in *3rd AutomationML user conference*, 2014.

[21] T. Helbig, S. Henning, and J. Hoos, "Efficient engineering in special purpose machinery through automated control code synthesis based on a functional categorisation," in *Machine Learning for Cyber Physical Systems*. Springer, 2016, pp. 67–74.

[22] M. Wenger, A. Zoitl, and G. Schitter, *Distributed Control Applications: Guidelines, Design Patterns, and Application Examples with the IEC 61499*. CRC Press, 2015, ch. Automatic Reengineering of IEC 61131-Based Control Applications into IEC 61499, pp. 97–122.

[23] R. Drath, "Let's talk automationml what is the effort of automationml programming?" in *Emerging Technologies & Factory Automation (ETFA), 2012 IEEE 17th Conference on*. IEEE, 2012, pp. 1–8.

[24] T. Strasser, M. Rooker, G. Ebenhofer, A. Zoitl, C. Sunder, A. Valentini, and A. Martel, "Framework for distributed industrial automation and control (4diac)," in *2008 6th IEEE International Conference on Industrial Informatics*, 2008.

[25] A. Zoitl, T. Strasser, and A. Valentini, "Open source initiatives as basis for the establishment of new technologies in industrial automation: 4diac a case study," in *2010 IEEE International Symposium on Industrial Electronics*, 2010.

[26] M. Wenger, "Model-Driven Re-Engineering of Control Applications," *Dissertation, Technische Universitt Wien*, 2016.

[27] B. Brandenbourger, M. Vathoopan, and A. Zoitl, "Generating Metamodel-based Descriptions of Automation Components in AutomationML," in *IEEE International Conf. on Industrial Technology (ICIT)*. IEEE, 2017.