A bi-directional Interface enabling cross-disciplinary Engineering with RAMI 4.0 and AutomationML

Christoph Binder and Christian Neureiter
Josef Ressel Center for Dependable System-of-Systems Engineering
Urstein Sued 1, A–5412 Puch/Salzburg, Austria
(firstname.lastname)@fh-salzburg.ac.at

Arndt Lüder
Otto-v.-Guericke University
Universitätsplatz 2
D-39106 Magdeburg, Germany
arndt.lueder@ovgu.de

***Abstract*—The Reference Architecture Model Industrie 4.0 (RAMI 4.0) has been proposed to faciliate the discussion and contribute to the standardization of future industrial systems, especially in the area of basic engineering. In order to provide the resulting system model to the heterogeneous detailed engineering disciplines, AutomationML appears to be one of the technology drivers, giving the possibility to exchange respective engineering artifacts. Concluding, the utilization of this standard after applying the theoretical engineering concepts of RAMI 4.0 would strongly enhance the interoperability between industrial systems aligned to the reference architecture on the one hand and provide a practical implementation of its theoretical concepts on the other hand. Therefore, this work-in-progress Paper introduces the first implementation of a bi-directional interface, which interconnects basic engineering with detailed engineering of such systems. As model-based systems engineering (MBSE) with either RAMI 4.0 or AutomationML is currently under broad investigation, already established approaches are used to verify the application of the developed interface. An initial real-world example is taken for use to validate the research goals or challenges of the resulting engineering tool-chain, which can strongly contribute to engineering practitioners of flexible production systems by maxing out the heterogeneous tool-landscape.**

***Index Terms*—Reference Architecture Model Industrie 4.0 (RAMI 4.0), Model-based Systems Engineering (MBSE), Industrial Internet of Things (IIoT), AutomationML, Model Transformation**

I. Introduction

Under the term of Industry 4.0 or Industrial Internet of Things (IIoT), flexible production in lot size 1 has gained increasing importance in recent years. This resulted in new ways of value creation throughout the whole product life-cycle or intelligent decision-making supported by ubiquitous interconnection. With those new possibilities, efficiency and effectiveness in regard to production systems are gaining more and more interest, as global players as well as small and medium-sized enterprises (SMEs) are required to be sustainably competitive. The mentioned aspects result in new ways of thinking when setting up contemporary production lines. Hence, conventional approaches reach their limits when dealing with the complexity resulting from this trend.

Several methodologies emerged recently, like versioning [1] or simulation [2], promising new opportunities to deal with complex production systems. One of those methodologies has been proposed with Model Based Systems Engineering (MBSE), aiming to support the construction of such systems throughout all engineering phases by applying domain models [3]. Hence, to ensure the application of MBSE, modeling languages are inevitable when dealing with the complexity whilst planning, designing, realizing and maintaining flexible production systems [4]. As different languages deal with different aspects of the systems, heterogeneous modeling tools have been introduced, each of them having particular advantages. While either domain-internal as well as domain-crossing aspects are important to consider throughout the engineering life-cycle of those systems, comprehensive all-in-one tools might be too limited in scope to deal with all their facets, rather flexible tool-chains should be established [5].

An example for such a domain-specific approach is Reference Architecture Model Industrie 4.0 (RAMI 4.0) and its corresponding modeling tool, the RAMI Toolbox [6]. Their goal is to structure an industrial system according to different aspects and more granular parts in the sense of basic engineering. This means, a high-level draft of the production system and its plant topology is prepared for detailed engineering disciplines, like electrical or mechanical engineering. By using interoperability layers, hierarchy levels and the product's life-cycle during the development of the flexible production system, the modeling paradigms *separation of concerns* as well as *divide and conquer* are considered. The current modeling methodology including a development process and a modeling framework of the RAMI Toolbox are specifically targeting the engineering aspect of such a system and developing it top-down from a higher perspective to a detailed one, as requested by Model Driven Architecture (MDA).

Another example targeting the industrial domain and ensuring integrity throughout the whole value creation network is AutomationML [7]. Originally developed for exchanging data between manufacturing engineering tools via XML-structures, AutomationML has been successfully used in applications that go widely beyond its traditional restricted purpose, like security risk identification [8] or hybrid assembly workplace generation [9]. Moreover, considering the engineering data exchange logistics [10], AutomationML can ideally be applied to transfer engineering information of flexible production systems to its various application areas within the tool-chain, especially in detailed engineering. Thus, the link between the engineering concepts of RAMI 4.0 and AutomationML could strongly contribute to future production systems engineering.

Therefore, the contribution of this paper proposes a bi-directional interface between the RAMI 4.0-cube and AutomationML, allowing to either export a developed system to the Computer Aided Engineering Exchange (CAEX)-file or import an external optimized system stored to such a file. This means, a production systems engineer could export engineering artifacts, such as plant topology models, from the RAMI Toolbox to AutomationML for use in other tools along the tool-chain. The other way round, a tool-chain designer might integrate the RAMI Toolbox into an AutomationML-based engineering tool-chain by applying its implemented interface. Hence, special focus is set on usability, as MBSE usually is a manual process having a lot of repetitive tasks. By doing so, the automation potential of this approach is investigated and validated with an excerpt of a real-world case study and evolutionarily developed by applying the Agile Design Science Research Methodology (ADSRM) [11].

To address these aspects, the remainder of this work-in-progress paper is structured as follows: In Section II, the background about AutomationML and RAMI 4.0 as well as the related work in this area is explained in more detail. The development, implementation and application of the interface itself in its current research state is delineated in Section III. Finally, in Section V the results of the study are summarized and a conclusion is given.

II. Related Work

A. RAMI Toolbox

The three-dimensional layout of RAMI 4.0 allows the description of industrial systems according to multiple perspectives [4]. The "Interoperability Layers", the "Life Cycle & Value Stream" axis as well as the "Hierarchy Levels" are each addressing different aspects of the system, as their names imply. From top to down, the viewpoints of the system are aligned, while the other two perspectives deal with the well-known automation pyramid as well as the system's life-cycle.

The RAMI Toolbox however has been implemented as Add-In for Enterprise Architect (EA) with the goal to make RAMI 4.0 applicable. Therefore, it includes all domain-specific aspects of the reference architecture and provides a proprietary metamodel, a UML-profile as well as a Domain Specific Language (DSL). With the help of this toolbox, industrial systems engineering is supported in different ways, mostly with regard to usability and automation. The graphical user interface (GUI) of the RAMI Toolbox allows users to provide them with all functionalities and a specific guideline how to develop such a system. The diagrams can automatically be created with the DSL elements by navigating through the panes. Additionally, interfaces to other tools or model checking functionalities are integrated within the RAMI Toolbox.

B. AutomationML

Originally, AutomationML has been developed with the goal to enhance the data exchange between engineering tools targeting the manufacturing area [7]. The object-based arrangement of plant components introduced by this standard allows to structure them according to multiple granularity levels, enabling the decomposition into single elements or complete manufacturing cells. In more detail, an XML-based data format arranges the respective information with regard to the CAEX data format and its object-oriented paradigm. This enables to associate engineering tools and disciplines by storing all engineering information of multiple domains, like mechanical, information or electrical engineering, within a single point of truth [12].

By now, AutomationML has a broader application than its original purpose, as it has paradigmatic been applied in a data exchange logistics for engineering networks by exploiting data integration [10]. To store the information accordingly, AutomationML introduces four major concepts of differentiating object-based components within a flexible production system. At first, RoleClasses describe the abstract system architecture regardless of its technical implementation and thus deal as foundation for other objects. The semantic is associated to the system element with the help of this class. Next, the SystemUnitClasses have to be defined based on the available system components within the delimited area or domain, like company-specific libraries. The so-called InterfaceClass specifies all interfaces and data exchange standards within the industrial system. Finally, Instance Hierarchies store all information including instances of system components within a particular project [13].

C. Modeling Language Mapping

A major prerequisite for developing the interface has already been proposed in [14]. In their work, the authors propose the mapping of AutomationML to SysML in order to enable cross-disciplinary modeling with those two languages. Thereby, they compared the different notations of the class-based SysML to the prototype-based AutomationML and developed a separate stereotype, which interconnects the two languages by profiling. However, while the result enables to model and export AutomationML files directly with corresponding EA diagrams, some issues have been pointed out, which lead back to the different application areas of each language.

While the mentioned work introduces the interconnection of those two methodologies for the first time, their approach has one drawback regarding to applicability. As they introduce own stereotypes and diagrams, a system modeled with SysML needs to be modeled with this proprietary approach again before being exported into an AutomationML-file. This includes some kind of redundancy and while most of the systems engineers are already conform with SysML, creating such files directly from this modeling language would strongly contribute to its usability. Because of these reasons and as the RAMI Toolbox itself partly makes use of SysML, the developed interface associates models directly to be applied by systems engineers in each of their discipline.

III. Design & Development

A. Interface Development

Before actually developing the interface between the respective modeling frameworks, similar concepts describing the same subject need to be compared and mapped if needed. While the upper layers of the RAMI 4.0 modeling framework, which is implemented in the RAMI Toolbox, describe contextual aspects with various domain-specific languages, the bottom layers are implemented with well-known Unified Modeling Language (UML)-diagrams or the Systems Modeling Language (SysML). This means, the technical system is decomposed into its single part at the end of the engineering process by applying an SysML Block Definition Diagram. In conclusion, this type of diagram appears to be the best matching to be transformed into AutomationML vice versa. Thus, Table I shows the results of the mapping between SysML and AutomationML. Thereby, the proposals of [14] are also considered, as they are introducing an own stereotype to link the respective concepts instead of mapping them directly.

TABLE I
Mapping between AutomationML and SysML concepts

Concept	AutomationML	SysML
Model	File	Model
Libraries	InstanceHierarchy	Package
Objects	InternalElement	Block
Interfaces	ExternalInterface	Port
Attributes	Attribute	Tag
Abstraction	Decomposition	Part Association
Connectors	InternalLink	Connector
Pattern	Role	Stereotype

The table indicates that an EA model is equal to an AutomationML file. Each InstanceHierarchy is realized with a Package, while SysML Blocks are translated to InternalElements. Following this principle, Ports realized the ExternalInterfaces Attributes are the same concept as Tagged Values in EA. InternalLinks are realized within the model with Connector relationships and the decomposition between the elements is extracted from the part associations. Finally, element Roles are implemented by stereotypes within EA.

After mapping the respective concepts, the interface itself could be implemented. As seen in Figure 1, this is done by providing a new function via the RAMI Toolbox GUI. This function realizes the bi-directional interface and offers an import as well as an export functionality. While explaining the whole source code of the interface would exceed the scope of this paper, the main functionality is outlined roughly. The export-function is thereby called on a SysML block and recursively finds all connected elements, ports and attributes, according to the previously defined correlations. The enclosing package deals as InstanceHierarchy to store all plant information. After finding the connected elements of the chosen block, a new AutomationML file is created and the listed elements are inserted one by one.

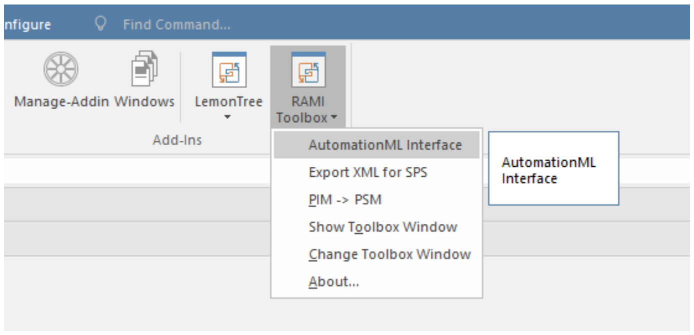

Fig. 1. RAMI Toolbox AutomationML interface

Therefore, Drath [15] proposed an C# Application Programming Interface (API) that is able to automate this step with minimal manual effort. This API can directly be implemented into the RAMI Toolbox as dynamic-link library (DLL), since the RAMI Toolbox itself is implemented in C#. Thanks to this DLL, difficult XML-transformations are abolished and resource consumption is optimized. After creating the AutomationML-file, it is saved to the designated storage space, where it could be used for further processing in other tools.

The counterpart, importing an AutomationML-file into an EA model, follows the same principle. After choosing the import-function via the RAMI Toolbox GUI, an AutomationML-file could be selected. This file is subsequently traversed and all InternalElements with their correlations are stored within the EA package. Additionally, a separate SysML Block Definition Diagram is created, which displays all imported blocks, ports, attributes and connections. This allows to directly use the imported elements to be interconnected with the already existing RAMI 4.0 model or further edit them with the EA modeling tools.

IV. Application

In order to evaluate the implemented interface, a superficial case study of the Siemens Fischertechnik industrial plant model is applied. This allows to investigate the usability and applicability as well as correct functionality. In more detail, this case study makes use of the newly implemented punching station, which has been integrated into the original production line according to the peculiarities of RAMI 4.0. With the help of the interface, the resulting SysML block can be exported and externally processed, for example with the AML Editor, or the resulting AutomationML-file can be imported again into the industrial plant model. Figure 2 indicates the comparison between the imported model and the exported file. The image shows that all related blocks are also implemented as InternalElements, which also counts for the ports respectively ExternalInterfaces. The hierarchy between the elements has also been correctly implemented. In addition, not visible in the image is that the attributes are also successfully transferred either by importing or by exporting them. The figure also shows one major drawback of the approach. As currently only instantiated systems can be considered within RAMI 4.0, only the InstanceHierarchy could be investigated.

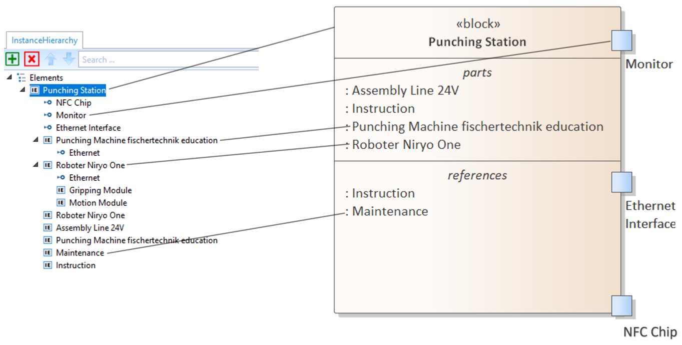

Fig. 2. Comparison between SysML model and AutomationML file

This also entails that the corresponding supported RoleClasses, SystemUnitClasses or InterfaceClasses could not be linked to the existing instances. In future reference architecture definitions, the interplay between AutomationML and RAMI 4.0 could strongly be enhanced by also including abstract or company-specific aspects.

V. Discussion, Conclusion & Future Work

As future production systems are addressing a lot of different aspects throughout the whole value creation network in the context of Industry 4.0 or IIoT, more and more complexity emerges when developing them. Thus, multiple engineering tools need to work together, as each one addresses a separate part of the complex system, which reduces the complexity by breaking the system down. With the aim to close this gap in current approaches, this work-in-progress paper proposes a bi-directional interface between RAMI 4.0 and AutomationML. While developing such a system could be achieved with RAMI 4.0 and the RAMI Toolbox, the resulting system model could be transferred to other engineering tools with AutomationML. Therefore, the current state of this interface is presented, which automatically exports AutomationML-files from previously modeled systems or imports SysML Block Definition Diagrams from previously created AutomationML-files. This interface is evaluated towards usability and applicability by making use of a superficial industrial plant case study, which should be enhanced in future.

The current version of the RAMI Toolbox[1] helps system engineers by modeling a production system according to the characteristics of RAMI 4.0. In its current state, only the instance of such a system could be modeled, either as it is currently build or as it should be build in future. However, no reference architectures, targeting specific domains or domain-specific aspects, are currently defined. Hence, the AutomationML-file as well as the interface could also consider the InstanceHierarchy of the system to be described.

This leads to the future work of the mentioned approach. While it is planned to derive reference architectures regarding a particular sub-domain or company, the interface should also be extended in order to enable exporting or importing the InterfaceClassLib, the SystemUnitClassLib and the RoleClassLib. This should be done in future projects and validated with a more comprehensive case study of the Siemens Fischertechnik industrial plant model. While the applied research methodology for further enhancing the result of this paper is ADSRM, new outcomes or enhancements will be implemented as well as published step by step in small research cycles.

[1]The RAMI Toolbox is available at http://www.rami-toolbox.org/

Acknowledgment

The support for valuable contributions of Siemens is gratefully acknowledged. The financial support by the Austrian Federal Ministry for Digital and Economic Affairs and the National Foundation for Research, Technology and Development and the Christian Doppler Research Association as well as the Federal State of Salzburg is also gratefully acknowledged.

References

[1] R. Mordinyi and S. Biffl, “Versioning in cyber-physical production system engineering–best-practice and research agenda,” in 2015 IEEE/ACM 1st International Workshop on Software Engineering for Smart Cyber-Physical Systems. IEEE, 2015, pp. 44–47.

[2] W. de Paula Ferreira, F. Armellini, and L. A. De Santa-Eulalia, “Simulation in industry 4.0: A state-of-the-art review,” Computers & Industrial Engineering, vol. 149, 2020, p. 106868.

[3] L. E. Hart, “Introduction to model-based system engineering (mbse) and sysml,” in Delaware Valley INCOSE Chapter Meeting, vol. 30. Ramblewood Country Club Mount Laurel, New Jersey, 2015.

[4] M. Hankel and B. Rexroth, “The Reference Architectural Model Industrie 4.0 (RAMI 4.0),” ZVEI, 2015.

[5] M. Barth, R. Drath, A. Fay, F. Zimmer, and K. Eckert, “Evaluation of the openness of automation tools for interoperability in engineering tool chains,” in Proceedings of 2012 IEEE 17th International Conference on Emerging Technologies & Factory Automation (ETFA 2012). IEEE, 2012, pp. 1–8.

[6] C. Binder, C. Neureiter, and A. Lüder, “Towards a domain-specific approach enabling tool-supported model-based systems engineering of complex industrial internet-of-things applications,” Systems, vol. 9, no. 2, 2021.

[7] R. Drath, A. Luder, J. Peschke, and L. Hundt, “Automationml-the glue for seamless automation engineering,” in 2008 IEEE International Conference on Emerging Technologies and Factory Automation. IEEE, 2008, pp. 616–623.

[8] M. Eckhart, A. Ekelhart, and E. R. Weippl, “Automated security risk identification using automationml-based engineering data,” IEEE Transactions on Dependable and Secure Computing, 2020.

[9] M. Fechter and A. Neb, “From 3d product data to hybrid assembly workplace generation using the automationml exchange file format,” Procedia CIRP, vol. 81, 2019, pp. 57–62.

[10] A. Lüder, J.-L. Pauly, F. Rinker, and S. Biffl, “Data exchange logistics in engineering networks exploiting automated data integration,” in 2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA). IEEE, 2019, pp. 657–664.

[11] K. Conboy, R. Gleasure, and E. Cullina, “Agile design science research,” in International Conference on Design Science Research in Information Systems. Springer, 2015, pp. 168–180.

[12] A. Lüder, N. Schmidt, and R. Drath, “Standardized information exchange within production system engineering,” in Multi-disciplinary engineering for cyber-physical production systems. Springer, 2017, pp. 235–257.

[13] AutomationML consortium, “Whitepaper automationml part 1 - architecture and general requirements,” Tech. Rep., 2014.

[14] L. Berardinelli, S. Biffl, A. Lüder, E. Mätzler, T. Mayerhofer, M. Wimmer, and S. Wolny, “Cross-disciplinary engineering with automationml and sysml,” Automatisierungstechnik, vol. 64, no. 4, 2016, pp. 253–269.

[15] R. Drath, “Let’s talk automationml what is the effort of automationml programming?” in Proceedings of 2012 IEEE 17th International Conference on Emerging Technologies & Factory Automation (ETFA 2012). IEEE, 2012, pp. 1–8.