Available online at www.sciencedirect.com

ScienceDirect

Procedia CIRP 57 (2016) 321 – 326

www.elsevier.com/locate/procedia

49th CIRP Conference on Manufacturing Systems (CIRP-CMS 2016)

A systematic approach to OPC UA information model design

Florian Pauker[a,*], Thomas Frühwirth[b], Burkhard Kittl[a], Wolfgang Kastner[b]

[a] *Institute for Production Engineering and Laser Technology - TU Wien, Karlsplatz 13, 1040 Vienna, Austria*
[b] *Institute of Computer Aided Automation - TU Wien, Karlsplatz 13, 1040 Vienna, Austria*

[*] Florian Pauker. Tel.: +43-158801-311-382; Fax: +43-158801-311-99; *E-mail address:* pauker@ift.at

Abstract

Current trends in manufacturing focus on the use of Information and Communication Technologies (ICT). The physical world and its virtual representation are increasingly converging, which leads to Cyber-Physical Production Systems (CPPS) in the manufacturing environment. CPPS synergize conventional production technology as well as ICT, allowing machines and products to exchange information, trigger actions and control other components autonomously. Therefore, seamless communication between physical objects of the shop floor and various computer systems is required. The Reference Architecture Model Industrie 4.0 (RAMI4.0) provided by the Plattform Industrie 4.0 specifies requirements for CPPS consisting of Industrie 4.0-components. In such systems, a major goal is to enable communication of I4.0 components among each other via industrial networks. For this purpose, RAMI4.0 suggests that each component has a virtual representation and uses Service-Oriented Architecture (SOA) based communication with I4.0-semantics. This paper describes a systematic approach to OPC Unified Architecture (OPC UA) information model for representing the static and dynamic behavior of manufacturing systems. Moreover, the approach is generic in the sense that it can be used to define information models for multiple target technologies, such as OPC UA, MTConnect and others. It even allows to reuse large parts of the generated models for similar manufacturing utilities and various target technologies. Therefore, we first present a concept for system analysis and design by using the Unified Modeling Language (UML), which is widely accepted for interdisciplinary work. The information gathered is then transformed to OPC UA information models which serves as target technology in this paper. The purpose of this approach is to simplify the process of defining virtual representations of manufacturing systems. Applying the presented concept allows transformation of classic manufacturing systems into CPPS with SOA-based communication and semantically rich virtual representations of individual components. It is therefore well suited to meet the requirements specified by RAMI4.0.

Peer-review under responsibility of the scientific committee of the 49th CIRP Conference on Manufacturing Systems
Keywords: OPC UA; MDA; CPPS; SOA; information model; RAMI4.0; virtual representation

1. Introduction

Current manufacturing systems are faced with increasing use of Information and Communication Technologies (ICT). One of the most significant developments in that field are Cyber-Physical Systems (CPS). CPS are a new generation of systems with integrated computational and physical capabilities [1]. They are characterized by a combination of real (physical) objects with information processing (virtual) objects which can communicate via (industrial) communication networks [2].

The introduction of CPS into the manufacturing environment leads to Cyber-Physical Production Systems (CPPS). CPPS synergize conventional production technology and ICT, allowing machines and products to exchange information, trigger actions and control other components autonomously. This includes information exchange across all levels of production, from sensor to processes through machines up to production and logistics networks [3].

Several R&D topics arise from the design and implementation of CPPS. The most important ones are to create standardized interfaces for vertical and horizontal data exchange through value added networks and define the ”virtual twin” (i.e., the virtual representation of a physical object) of a manufacturing system. To realize the idea of flexible production systems, new methods are required which support the fusion of the virtual and real sub-systems. These methods have to be robust and fit for the future to be useful in rapid changing environments, such as modern production facilities. [4]. The Reference Architecture Model Industrie 4.0 (RAMI4.0) [5] provided by the Platform Industrie 4.0 offers solutions to the previously named challenges. It defines a Reference Architecture Model, which precisely describes Industrie 4.0 compliant production equipment. In addition, RAMI4.0 describes future production networks consisting of Industrie 4.0 components (I4.0-components). These I4.0-components have Service-Oriented Architecture (SOA) communication capabilities and a virtual

Peer-review under responsibility of the scientific committee of the 49th CIRP Conference on Manufacturing Systems
doi:10.1016/j.procir.2016.11.056

representation which adds semantic information to the physical object. Semantic describes the relation between signifiers (e.g. words) and their denotation, what they stand for.

This semantic is necessary, because the exchange of information between two or more I4.0-components requires unequivocal semantics. RAMI4.0 suggest OPC UA, because it can be used for both, communication as well as to define virtual representations. OPC UA is the newest standard provided by the OPC Foundation, because of its communication capabilities.

In complex manufacturing systems several different entities are cooperating and each one needs a digital description (information model) in a semantic way.

While the Unified Modelling Language (UML) is established as the major modeling language in software development, there are several standards being used in the manufacturing environment. For each of them, the model and the code must be written and maintained separately, which requires high manual effort.

Model-driven techniques, which are already used to design complex software systems tremendously simplify the process of information model design. The main concept of a model-driven approach is to separate the functional description and the implementation [6]. Thus, it is not necessary to repeat the process of defining an application or the system's functionality and behavior each time a new communication technology (OPC UA, for example) comes along.

This paper describes a model driven approach to OPC UA information model design for the virtual representation of systems. It thereby focuses on the manufacturing environment. The paper precisely specifies the workflow needed to generate these virtual representations. This workflow briefly introduces the necessary diagrams and tools for modeling and code generation.

The remainder of the paper is structured as follows: Chapter 2 gives an introduction to the current standardization efforts, the "RAMI4.0 " (chapter 2.1). The connection between RAMI4.0 and OPC UA and UML is also described (Chapter 2.2 & 2.3). Chapter 2.4 explains the basics of the MDA. Chapter 3 presents a model driven approach for OPC UA information model design. It describes all MDA phases in detail, including requirements analysis, platform independent modeling, platform specific modeling (OPC UA information modeling) and code generation. Chapter 4 summarizes the paper and gives an overview of the next steps to be done in the future.

2. State of the art and basic technologies

2.1. Reference Architecture Model RAMI4.0

The German Industrie 4.0 platform, consisting of Zentralverband Elektrotechnik- und Elektronikindustrie e.V (ZVEI), Verband Deutscher Maschinen- und Anlagenbau (VDMA), and Bundesverband Informationswirtschaft, Telekommunikation und neue Medien (BITKOM), has published a first version of a Reference Architecture Model for Industrie 4.0 (RAMI4.0) [5] which precisely describes Industrie 4.0-compliant production equipment.

RAMI4.0, consists of a three-dimensional coordinate system that describes all necessary aspects of Industrie 4.0.

The right horizontal axis shows the hierarchy levels known from IEC 62264 [7] standard for enterprise IT and control systems. These levels represent the different entities grouped by the functionality which can be found in factories.

In addition to the hierarchical levels defined by IEC 62264, the layers labelled "Product" representing a workiece and Connected World (the connection to the Internet of Things and Internet of Services) were defined to represent all hierarchical levels in an Industrie 4.0 environment.

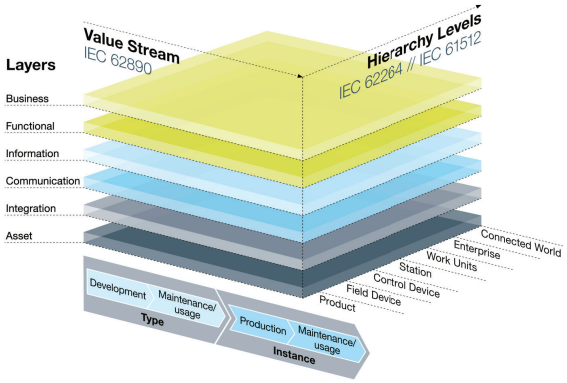

Fig. 1. Reference Architecture Model RAMI4.0 [8]

On the left horizontal axis, the life cycle of facilities and products is shown. These different life cycle phases are based on IEC 62890 for life-cycle management. The differnce between instance and type are clearly described by RAMI4.0. A type becomes an instance when the design and prototyping is completed and the product is ready for beeing manufactured.

The layers on the vertical axis describe the decomposition of a machine into its properties structured layer by layer. These representations originate from ICT, where properties of complex systems are usually broken down into layers.

Within these three axes, all crucial aspects of Industrie 4.0 can be mapped, allowing objects such as machines to be classified according to the model. Highly flexible Industrie 4.0 concepts can thus be described and implemented using RAMI4.0.

The reference architectural model allows for step-by-step migration from the present into the future world of Industrie 4.0. Cross-vendor data exchange is necessary for communication between machines or between machines and workpieces. This requires unified semantics including a common syntax for data. RAMI4.0 proposes OPC UA for realizing such communication.

2.2. OPC Unified Architecture - OPC UA

OPC Unified Architecture (OPC UA) [9] is the latest OPC standard specification provided by the OPC Foundation. It is used for interconnectivity in state-of-the-art industrial automation technology [10].

Classic OPC is implemented in almost all industrial automation systems and, thus, well accepted. The three major improvements of the UA are platform independence, the capability to exchange data beyond local networks and the specification of a generic information model, which is a basis for domain specific

information models [11]. Therefore it is widely accepted as an enabling technology for Industrie 4.0.

The OPC UA specification consists of 13 parts. The first seven parts are related to the core specifications e.g. the concept, security model, address space model, services, information model, service mappings and profiles. The parts eight to

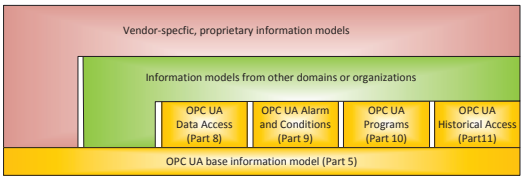

Fig. 2. OPC UA

thirteen are related to access type specifications like data access, alarms and conditions, programs, historical access, discovery and aggregates. The most important ones necessary for user-specific information models are depicted in Fig. 2. The address space model defined in Part 3 of the specification [12] is the meta model of OPC UA. The base component of the meta model are nodes. Several node classes are defined specializing the base node class (e.g. objects, variables, etc.). Each node has a set of attributes depending on the node class. Some attributes are mandatory and some are optional. For example, each node class requires a ”NodeId” uniquely identifying the node, while the ”Description” is optional [13].

Besides the communication part, data modeling is the second fundament of OPC UA. In [9], the base principals of UA modeling are listed as follows:

- Using object-oriented techniques with type hierarchies and inheritance.
- Type information is provided and can be accessed the same way as an instance.
- Full meshed network of nodes allows to connect the information in different ways.
- Extensibility of the type hierarchies as well as the references types between nodes.
- No limitations of modeling in order to allow an appropriate information model design.
- OPC UA information modeling is always done on the server.

These principals allow to generate simple, but also very complex, OPC UA information models.

2.3. Unified Modeling Language - UML

The Unified Modeling Language (UML) specified by the Object Management Group (OMG), is a graphical language for visualization, specification and documentation of artefacts of software systems [14]. UML is nowadays not only accepted as major modeling language for IT-software design, but is also often used as a tool for interdisciplinary work. UML's priority is to specify terms and relationships between them in form of so called ”models” [15]. The latest specification, UML 2.5, has many types of diagrams which are broadly grouped into two categories: structural and behavioral diagrams [16]. Behavioral diagrams include dynamic diagrams, e.g. sequence diagram or state-chart diagram, including a few that represent special aspects of interactions. Structural diagrams like class diagrams, object diagrams, etc. are used to model static information.

To manage complexity, UML is organized into language units comprising a set of modeling elements [17]. The core language units for the later described MDA approach are the class, state-chart, component and use-case models.

2.4. Model-driven Architecture - MDA

In the Model-driven Architecture (MDA) models are the central elements of the software development process. The aim is to derive platform-specific models from platform-independent models using a highly automated process. This reduces the effort of software development and the adaptation to new technologies [6]. Any additions and changes to the model will be automatically reflected the next time the code is generated. MDA models can then be used for the production of documentation, acquisition specifications, system specifications, technology artifacts (e.g. source code) and executable systems [18]. Currently the main disadvantages of MDA is, that available tools are not able to automatically generate 100% of the code. So there is always a need to adapt the code manually after the generation process.

UML is the preferred modeling language for MDA. The most important advantage of using UML for Model-driven-Architecture is that it proposes a format for information exchange between tools and applications. Aside from UML, XML Metadata Interchange (XMI) [19] and the Meta-Object Facility (MOF) [20] are the most important standards related to MDA.

MDA defines three levels of abstraction. The Computer Independent Model (CIM), the Platform Independent Model (PIM) and the Platform Specific Model (PSM) [21].

A CIM is also often referred to as domain model because it uses a vocabulary that is familiar to the Subject Matter Experts (SMEs) [22]. It shows exactly what the system will do, but do not respect technology-specific information, to remain independent of the system implementation [23]. The CIM plays an important role to close the gap between domain experts and the information technologists responsible for implementing the system [23].

A PIM specifies the system in more detail but still maintains a sufficient degree of generality to allow its mapping to one or more platforms. Usually this is done by defining a set of services, which do not consider technical details. The realization of these services in a platform specific way is done with other models [23].

A PSM combines the specifications defined in the PIM with the details required to stipulate how a system uses a particular type of platform. The PSM has to contain all relevant information, i.e. the static structure and dynamic behavior of the system, using different Domain Specific Languages (DSL). Automated tools may be used to perform the PIM-PSM translation [21,23].

3. MDA approach to OPC UA information model design

The approach presented in this paper is depicted in Fig. 3. It offers a MDA based method using UML class, state-chart,

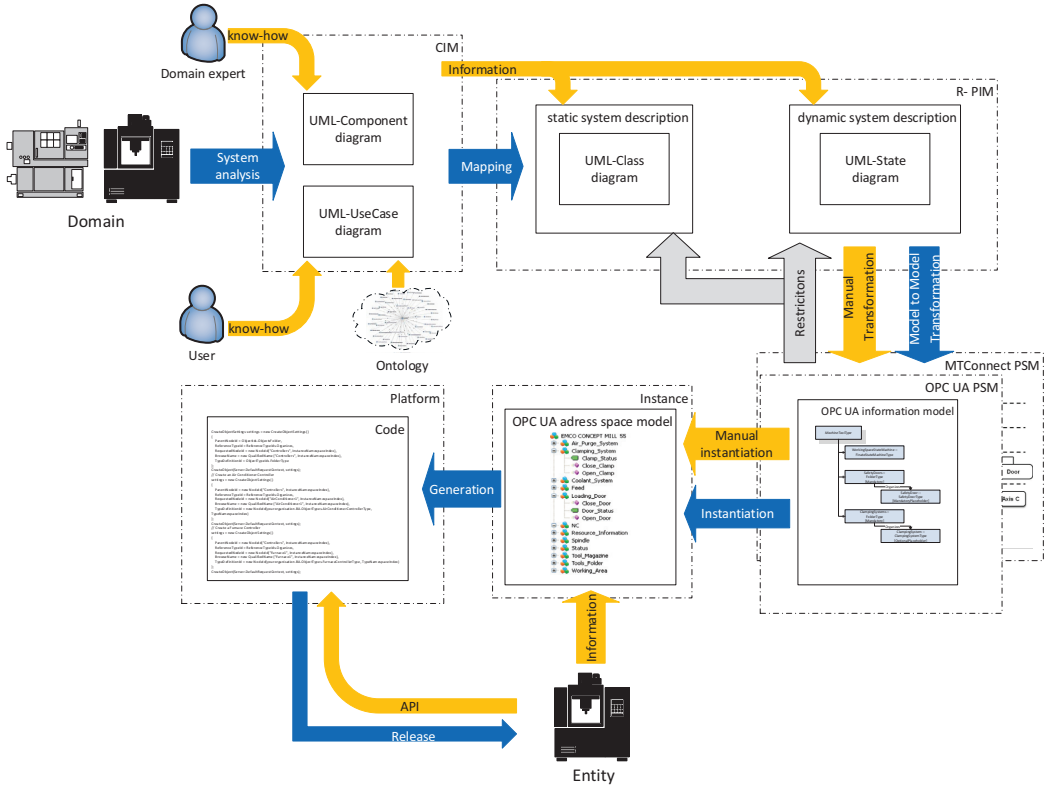

Fig. 3. MDA based workflow for OPC UA information model design

use-case and component diagrams for virtual representation of manufacturing systems. Using the presented approach allows OPC UA information model design in the manufacturing domain.

MDA is only an approach, systematic methods for each domain use-case have to be defined to exploit it [24]. However, which models should be used for different applications in the CIM and PIM is not always clear. Kriouile et.al [25] give an overview of different UML diagrams used for describing PIM and CIM. Despite the lack of appropriate modeling diagrams for using a MDA in manufacturing domain, there are also problems in OPC UA information modeling and implementation.

The development and implementation of OPC UA interfaces for virtual representation as ”virtual twin” of systems needs know-how from several domain experts. In particular, the information model design and its implementation need a lot of effort and increase the initial investments required to get familiar with the OPC UA technology. To reduce development and implementation effort Software Development Kits (SDKs) from different manufactureres are used. These manufacturer mostly offer special tools for OPC UA information model design and, in some cases, to generate runnable code. These tools use different versions of XML-schemata for OPC UA information model representation. Therefore, the developed information models are often not compatible with other solutions available on the market. MDA seems to provide a proper solution for this problem.

The approach describes the complete engineering process of OPC UA information models from system requirements to code generation and deployment (cf. Fig. 3). The blue arrows in the graphic show the suggested workflow creating virtual representation of systems with OPC UA. In an ideal manner, this workflow is done automatically but currently there is a lack of automatic mechanisms. The yellow arrows represents knowledge and actions done by humans. In an optimal MDA, there is only a single information flow into the CIM, while all other models and necessary code is generated automatically. As this is currently not possible, we need information from system experts and users to generate the PIM and code for one entity. The major difference between MDA and the presented approach is that we have to take care of restrictions (gray arrows in Fig. 3) given by the technologies used in the manufacturing domain. This restrictions are respected by the Restricted Platform Independent Model (R-PIM) explained in Chapter 3.2.

3.1. Domain description - CIM

We use UML component and use-case diagrams for describing the target domain. With the component diagrams we define the system boundary and identify sub-systems necessary for modeling the PIM.

The use-case diagram describes the system functionality in an abstract way. With this description it is possible to define the operations of the system. These operations provide the basis for defining the operations in the class diagrams of the PIM. UML component diagrams are used to describe the static structure of the system. With the component digram it is possible to define the system boundaries and also identify sub-systems.

In a subsequent step, the interfaces between the different systems can be described in detail. The components defined in the diagram are the basis for the classes defined in the PIM.

The CIM to PIM transformation is currently done manually and requires additional knowledge from domain experts and system users (cf. Fig. 3).

3.2. Restricted platform independent description - R-PIM

The presented approach uses UML-class diagrams and UML-state charts to specify the dynamic behavior of manufacturing systems. The main elements are classes with related attributes and operations and the relationships between the classes. To identify the classes, attributes and operations necessary for the UML class diagramm we use, in addition to the information defined in the CIM, object-oriented modeling methods such as Object-Oriented Software Engineering (OOSE) [26] or the Object Modeling Technique (OMT) [27].

Many manufacturing systems are event-driven, which means that they wait for the occurrence of some external or internal events. After recognizing the event, such systems performing the appropriate action, e.g. changing outputs or generating other events that trigger internal software functions. With role-based scenarios, sub-systems are described and help to define the dynamic behavior of the system with state diagrams. UML state charts are used to display these transitions from one state to another. These diagrams consist of states, and transitions with optional guards between them [28].

Established platform specific technologies (e.g. OPC UA, MTConnect) have been examined and a set of restrictions for platform independent modeling were defined. These restrictions lead to a R-PIM model which is not specialized for only one PSM, but with particular attention to OPC UA because of its role in the current I4.0 initiatives. To model the restrictions a meta-meta model may be used which is currently undefined and has to be defined in future steps. A simple example for a restriction is given by multiple inheritances, which may be defined in UML. In C# or other implementing languages used in PSM multiple inheritances can not be implemented and so is a restriction for the R-PIM.

The R-PIM model is extended with additional information needed for the specific PSMs. This information transform the R-PIM into an annotated R-PIM. Each node in OPC UA has the mandatory attributes NodeId, NodeClass, BrowseName and DisplayName. Thus each class of the PIM is annotated with these four parameters to meet the OPC UA requirements.

3.3. Transformation of R-PIM to PSM

Currently the transformation from UML to OPC UA is done manually but there are already solutions available for automated transformation. Rohjans et al. [29] describe an automatic transformation of UML-class diagrams to OPC UA address space models for power systems. They have developed a tool called ”CIMBaT” [30] for automatic code design. This solution only allows to model the static behavior of the system.

A solution where the static structure as well as the dynamic behavior of a system, including restrictions, can be described and transformed into platform specific applications is currently developed at the TU Wien. For transformation of UML to OPC UA, transformation rules are required, which have been developed during the current research program, but are not further explained in this paper. The transformation of the dynamic behavior follows the approach for Guarded State Machines in OPC UA [31].

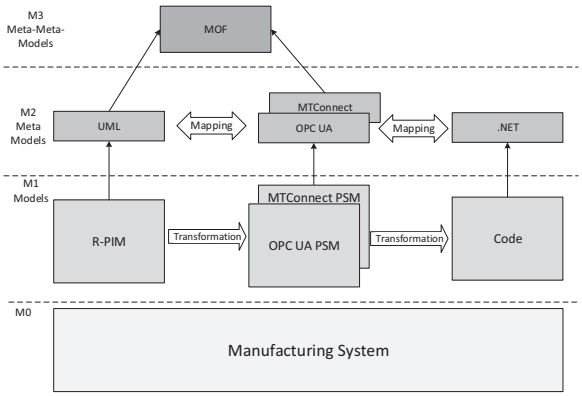

Fig. 4. Transformation process PIM to PSM

In conformity with [32] and [33], Fig. 4 shows the modeling tiers as well as the workflow from PIM to PSM and the final program code. The system layer M0 contains the Flexible Manufacturing System (FMS) as entity. This FMS is represented by a PIM that is defined in UML on Layer M2 and allows platform independent modeling on layer M1. Transforming this PIM to a desired platform is typically done by transforming PIM elements to PSM elements. A single element from the PIM may result in multiple PSM elements across multiple domains. A PSM for OPC UA conforms to the OPC UA meta-model, while a PSM for MTConnect conforms to the MTConnect meta-model. Each target platform can be implemented using different technologies and programming languages. Thus, a single PSM can be transformed into different programs (e.g. .NET based or C++).

The PIM as well as the PSMs on layer M1 describe the Manufacturing System on the system layer even if they are defined in different modeling languages.

Layer M3 of the figure consists of one unique meta-meta-model in order to define the language for specifying meta-models. In MDA, the Meta Object Facility (MOF) meta-meta-model is used to support interoperability between meta-models on layer M2.

3.4. Address space modeling and code generation

There are several tools with different levels of maturity available to create the address-space model and generate code. Most of them provide a graphical user interface for for modeling the address space, adding nodes and references and finally generate code. These tools not only speed up the implementation, they also increase the software quality by producing well structured and error-free code. As implementation is reduced to modeling and the code is generated automatically, even complex models can be implemented in a rapid fashion. Most tools are based on the related Software Development Kits (SDKs). In our proof-of-concept, we use a tool from Unified Automation, the UaModeler [34]. This tool supports all the SDKs from Unified Automation.

The UaModeler comes with three standard models and can be enhanced by user-defined models. These models can be edited, modified, and saved in an XML format. The UA Modeler is capable of generating code for multiple target languages. After creating the code for the specific entity, its necessary to manually implement the entity-specific functionality. An API is used for this purpose in many cases. After this step is completed the runnable server can be deployed. The two steps address space modeling and generating runnable code currently require a lot of time and effort because of the lack of available tools and APIs in the manufacturing environment.

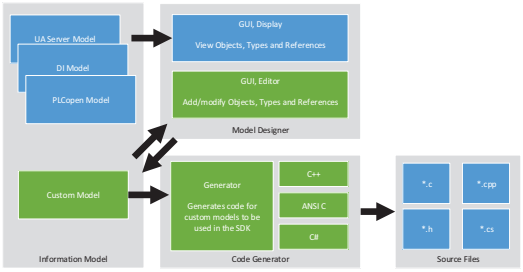

Fig. 5. Functions of state of the art OPC UA modeling tools [34]

4. Conclusion and further research

This paper presented an approach for OPC UA information model design based on Model-Driven Architecture. It showed the workflow for defining a virtual representation of manufacturing systems and the necessary steps for automatic code generation. Future work should help defining the CIM models. Ontologies as knowledge bases can be used instead of domain experts for automatic model design and transformation to platform independent models.

This paper only presented the realization of OPC UA information model design, but the overall approach is generic and it is possible to extend it with transformation rules for other communication and modeling languages available in the manufacturing domain (e.g. MTConnect). Additionaly the transformation of UML to OPC UA should be modeled in Model-to-Model Transformation (MMT) language and an add-In for Enterprise Architect should be developed.

5. Acknowledgement

The content of this paper is subject to current work within the FFG research project OPC4Factory (P843613). We acknowledge the support of the FFG through the Produktion der Zukunft research program.

References

[1] Baheti, R., Gill, H.. Cyber-physical systems. The impact of control technology 2011;12:161–166.
[2] Geisberger, E., Broy, M.. agendaCPS: Integrierte Forschungsagenda Cyber-Physical Systems; vol. 1. Springer-Verlag; 2012.
[3] VDI/VDE-Gesellschaft Mess- und Automatisierungstechnik, . Cyber-Physical Systems: Chancen und Nutzen aus Sicht der Automation. 2013.
[4] Monostori, L.. Cyber-physical production systems: Roots, expectations and R&D challenges. Procedia CIRP 2014;17:9–13.
[5] VDI/VDE-Gesellschaft Mess- und Automatisierungstechnik, . Status Report-Reference Architecture Model Industrie 4.0 (RAMI4.0). Tech. Rep.; 2015.
[6] Kempa, M., Mann, Z.A.. Model driven architecture. Informatik-Spektrum 2005;28(4):298–302.
[7] IEC 62264:2013, . Enterprise-control system integration. 2013.
[8] ZVEI, . Industrie 4.0: The Reference Architectural Model Industrie 4.0 (RAMI 4.0). 2015.
[9] Mahnke, W., Leitner, S.H., Damm, M.. OPC Unified Architecture. Springer Science & Business Media; 2009.
[10] Leitner, S.H., Mahnke, W.. Opc ua–service-oriented architecture for industrial applications. ABB Corporate Research Center 2006;.
[11] OPC Foundation, . OPC-UA Collaboration Overview. OPC Foundation; 2014.
[12] OPC Foundation, . OPC UA Specification: Part 3 Address Space Model. Version 1.00. 2006.
[13] Leitner, S.H., Mahnke, W.. OPC UAservice-oriented architecture for industrial applications. ABB Corporate Research Center 2006;.
[14] Booch, G., Rumbaugh, J., Jacobson, I.. Unified modeling language (uml). World Wide Web: http://www rational com/uml 1998;.
[15] Rumbaugh, J., Jacobson, I., Booch, G.. Unified Modeling Language Reference Manual, The. Pearson Higher Education; 2004.
[16] Object Management Group, . OMG Unified Modeling Language (OMG UML), Superstructure. Version 2.4.1. 2014.
[17] France, R.B., Ghosh, S., Dinh-Trong, T., Solberg, A.. Model-driven development using uml 2.0: promises and pitfalls. Computer 2006;39(2):59–66.
[18] Object Management Group, . Model Driven Architecture (MDA). Tech. Rep. MDA Guide rev. 2.0; 2014.
[19] Object Management Group, . XML Metadata Interchange (XMI) Specification - Version 2.5.1. 2015.
[20] Object Management Group, . OMG Meta Object Facility (MOF) Core Specification - Version 2.5. 2015.
[21] Brambilla, M., Cabot, J., Wimmer, M.. Model-driven software engineering in practice. Synthesis Lectures on Software Engineering 2012;1(1):1–182.
[22] Menken, I.. Model-Driven Architecture Complete Certification Kit - Core Series for IT. Emereo Publishing; 2013. ISBN 9781488508387.
[23] Truyen, F.. The fast guide to model driven architecture the basics of model driven architecture. Cephas Consulting Corp 2006;.
[24] Gherbi, T., Meslati, D., Borne, I.. Mde between promises and challenges. In: Computer Modelling and Simulation, 2009. UKSIM'09. 11th International Conference on. IEEE; 2009, p. 152–155.
[25] Kriouile, A., Gadi, T., Balouki, Y.. Cim to pim transformation: A criteria based evaluation. International Journal of Computer Technology and Applications 2013;4(4):616.
[26] Jacobson, I.. Object oriented software engineering: a use case driven approach 1992;.
[27] Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., Lorensen, W.E., et al. Object-oriented modeling and design; vol. 199. Prentice-hall Englewood Cliffs; 1991.
[28] Jeckle, M., Rupp, C., Hahn, J., Zengler, B., Queins, S.. UML 2 glasklar; vol. 1. Hanser Mnchen/Wien; 2004.
[29] Rohjans, S., Piech, K., Lehnhoff, S.. Uml-based modeling of opc ua address spaces for power systems. In: Intelligent Energy Systems (IWIES), 2013 IEEE International Workshop on. IEEE; 2013, p. 209–214.
[30] Rohjans, S., Piech, K., Uslar, M., Cabadi, J.F.. Cimbat-automated generation of cim-based opc ua-address spaces. In: Smart Grid Communications (SmartGridComm), 2011 IEEE International Conference on. IEEE; 2011, p. 416–421.
[31] Frühwirth, T., Pauker, F., Fernbach, A., Ayatollahi, I., Kastner, W., Kittl, B.. Guarded state machines in OPC UA. In: Proceedings of the 41st Annual Conference of the IEEE Industrial Electronics Society (IECON). 2015,.
[32] Bézivin, J.. In search of a basic principle for model driven engineering. Novatica Journal, Special Issue 2004;5(2):21–24.
[33] Atkinson, C., Kühne, T.. Model-driven development: a metamodeling foundation. Software, IEEE 2003;20(5):36–41.
[34] Unified Automation - UA Modeler. 2015. URL: https://www.unified-automation.com/products/development-tools/uamodeler.html.