Available online at www.sciencedirect.com

ScienceDirect

Procedia CIRP 00 (2021) 000–000

www.elsevier.com/locate/procedia

54th CIRP Conference on Manufacturing Systems

Modelling protocol gateways for cyber-physical systems using Architecture Analysis & Design Language

Patrick Denzler[a], Daniel Scheuchenstuhl[a], Daniel Ramsauer[a], Wolfgang Kastner[a]

[a] *Automation Systems Group,TU Wien, Vienna (1040), Austria*

* Corresponding author. Tel.: +43-1-588801-18311 ; fax: +43-1-588801-18391. *E-mail address:* patrick.denzler@tuwien.ac.at

Abstract

Developing, configuring, and deploying legacy protocol gateways for existing cyber-physical systems to improve interoperability remains challenging. A possible solution is using model-driven engineering languages such as the Architecture Analysis & Design Language (AADL) that enables combined modelling of software, hardware, and communication for distributed systems. Experiences made while creating an OPC Unified Architecture (OPC UA) / Data Distribution Service (DDS) gateway indicate the suitability of AADL to model complex software artefacts. Moreover, a proposed process reduces the development and configuration effort for platform-specific gateway instances. A challenge to be addressed is the generation of executable code for resource-constraint devices.

Keywords: Cyber-physical systems; Architecture Analysis & Design Language (AADL); OPC Unified Architecture (OPC UA)

1. Introduction

Interoperability and connectivity are essential for the realisation of the Industrial Internet of Things (IIoT) and Cyber-physical Systems (CPSs) [33, 23]. To enable seamless information exchange at all levels of a manufacturing organisation, the IIoT proposes a flat communication architecture to consolidate previously separate or isolated data islands [33]. However, one of the remaining challenges is the conversion of existing operational technology (OT) (e.g., programmable logic controllers (PLCs)) and Supervisory Control and Data Acquisition (SCADA) systems) with information technology (IT) (e.g., Enterprise-Resource-Planning (ERP) tool suites). Historically, OT and IT environments utilise different types of technologies to fulfil their specific needs on, for example, real-time capabilities or safety [17]. Newer technologies such as cloud and fog computing, time-sensitive networking (TSN), and middlewares (e.g., OPC Unified Architecture (OPC UA), Data Distribution Service (DDS)) aim to close the OT/IT gap [1].

Despite all technological advances, an often overlooked fact is that replacing current industrial automation systems is not a feasible solution (e.g., high cost). The long-life cycles in this area have created a decade's old landscape of various communication protocols that are still widely used and fully functional [31]. Typical solutions that combine older protocols with new technology are connectors, proxies, or gateways. However, such solutions are complicated software artefacts that require significant development effort and are difficult to maintain [30].

Specifically, building a gateway requires in-depth knowledge about the bridged protocols and the network environment. Adding to the complexity, each gateway instance needs to be configured and deployed to different types of devices from the cloud to the edge of a network [11]. A possible solution to reduce the effort for gateway development, configuration and deployment is utilising model-driven engineering (MDE).

MDE is standard in the development of embedded and distributed systems [18] where software and hardware closely interact [40]. In MDE, all or parts of a system are generated from models [32] created in modelling languages such as MARTE Unified Model Language [8], System Modeling Language (SysML) [15] or Architecture Analysis & Design Language (AADL) [13]. In particular, AADL aims to describe software and hardware components of a system independently.

Developers in avionics, robotics and embedded systems design are familiar with AADL; however, there is little experience with AADL in industrial automation. In this context, the following research aims to investigate the feasibility of using AADL as a modelling language for industrial automation and, thus, reducing the complexity in creating and maintaining gateways. In a first step, the research examines the ability of AADL to model complex software artefacts for different target platforms (code generation) and the possibility of automatically adapting models (configuration) by modelling an OPC UA/DDS middleware gateway.

The obtained results indicate the suitability of AADL in industrial automation. However, a final evaluation was not possible as the intended code generation based on the modelled gateway failed. Nevertheless, the identified issues add to the MDE body of knowledge in the automation systems context. Another contribution is the gateway generation and configuration process that reduces the development effort and builds the ground for further studies.

Section 2 presents background information and related work, followed by Section 3, introducing the AADL model language. The described methodology in Section 4 lays the ground for the OPC UA/DDS middleware gateway modelling in Section 5 and the discussion of the findings in Section 6. Section 7 concludes the paper.

2. Related Work

There is a vast body of research about AADL due to the diverse application possibilities of the language in domains where real-time, embedded, fault-tolerant, secure, safety-critical, software-intensive systems are relevant [14]. The main focus lies on embedded systems where software and hardware interact. In [38], AADL supports the model-driven design of an avionics CPS. The study shows several benefits of AADL in modelling the physical part of the CPS. Ziani et al. [40] focus on examining existing research related to MARTE, SysML, and AADL and are concerned with modelling embedded systems that have restrictions on memory, autonomy, and processing. The results indicate that AADL is most suitable for such tasks. Huges et al. [21, 20] demonstrate the usability of AADL in an MDE rapid prototyping process for space rugged embedded systems.

Zhang et al. [39] studied research concerning the reconfiguration of systems. AADL was used to model system configuration architectures and reconfiguration behaviours in integrated modular avionics systems in their studies. In the reconfiguration context, AADL is often used in combination with other approaches, e.g., Timed Petri Nets (TPNs) [34], or model checking [16] to find deadlocks, carry out reachability analysis or detect faults based on missed deadlines during reconfiguration. A recent study looks into using AADL to represent the implementation architecture of systems under different designs to improve verification and analysis of reconfigurable integrated electronic systems [24].

A closely related topic to reconfiguration and MDE is code generation based on models. The authors of [22] propose a code generation model-based framework that provides the flexibility to generate different source code for different devices. Their approach uses AADL to capture the platform's hardware/software architectural aspects, including sensors and actuators. An algorithm then generates platform-dependent code based on code snippets provided by a repository. Previous research built code generators to transform AADL models to C-code [25]. Similarly, the work done in [3] presents the generation of ROS-based software from AADL models for complex robot systems. In [2], the same authors extend their work to an entire toolchain to produce executable code. Whermeister et al. [37] present a nearly identical model-based approach with the difference that they use UML/MARTE.

Other research is using or analysing AADL for different purposes. Delanote et al. [7] use AADL to model a UDP/IP protocol stack and point out the lack of generic component concepts to model such applications. Other studies model middlewares for Distributed Real-Time systems [36] or a Fog Computing Platform (FCP) [4]. The latter one applies the AADL models to implement an industrial use case based on conveyor belts.

3. AADL as a Modelling Language

AADL is an Architecture Description Language (ADL) that aims to describe and analyse system designs before development and during the system lifecycle. The language targets real-time embedded systems, where hardware and software components are tightly coupled, and interaction analysis is required. Version 1.0 of AADL was published in 2004 and defines the textual syntax and the semantics of the core language [13]. SAE International standardises AADL. Several additional annexes specify, for example, how to graphically model with AADL, the meta-model and the XML/XMI interchange format. The AADL ecosystem also includes an open-source tool environment (OSATE) [5] based on Eclipse. Its plug-in mechanism allows the development of additional analysis and generation tools. Those tools provide the means for analysing availability & reliability, security, data quality, real-time performance, and resource consumption. It is possible to study individual models or the overall system and identify potential exploits or flaws within the overall design, which results in reduced development efforts and increased quality.

The central element of AADL is a hierarchical system model encompassing the entire system's architecture with all its components and their relationships. AADL distinguishes between three component categories in the system model: software, hardware, and hybrid (also called composite components). This combination enables AADL to model the hardware, software as well as communication components of a system.

Systems and components can have several sub-components/systems. A sub-component can inherit features and properties from its parent. Abstraction in AADL is possible as each component defines a type and an implementation separately. The type represents and defines how a component interacts with other components (e.g., interfaces), whereby the implementation of a component defines its functionality and internal structure. Furthermore, AADL allows specifying properties for each type of software, hardware, and hybrid component. For software components, such properties may be the dispatch protocol used or the period of some executed thread. Hardware components might benefit from timing requirements or bus bandwidth and processor frequency. These properties enable more realistic modelling of components.

3.1. Modelling Basics

Modelling in AADL is either done graphically or textually. The graphical part of the language gives a better overview of the system model. Each architecture type (software, hardware and composite) is a subelement of the system implementation and modelled with a case-insensitive generic syntax. Each architecture type contains several component types: *Process, Thread, Thread group, Subprogram Data* (software), *Processor, Bus, Memory, Device* (hardware) and *Composite* (composite). AADL also supports importing external previously created packages (e.g., libraries, source files) with public and private namespaces.

3.2. Component Modelling in AADL

The software components allow modelling processes, threads, data, and subprograms. Equally to processes within any operating system, processes in AADL represent executable application instances assumed to run on a processor. Threads are always a part of a process and communicate with features and connections. Features provide a common communication link with ports as interfaces that trigger events or represent constant data communication. Furthermore, features may also use component access, subprogram calls, and parameter interfaces (connectors) instead of ports to enable communication between components. Connections provide a common communication construct that defines one communication interface's mapping to another communication interface (one feature to another feature, for example). AADL also offers other features such as data types definition or component modes; For more information, the interested reader is referred to Feiler et al. [13].

To model the hardware components, AADL specifies four distinct types: processors, buses, memories and devices. Each component has a device and a bus type identifier. The first one represents an Input/Output (I/O) of a component. The latter enables a component to act as a hardware bus and establish communication between all connected hardware components.

The actual binding between the hardware and software architecture happens with binding properties. Each property maps a software component to the hardware architecture and specifically assigned usable resources. For this purpose, three types of binding properties exist, namely processor binding, memory binding, and connection binding.

To cover components that are not pure software or hardware related at the time of definition and may even be composed of several subsystem models, AADL allows the modelling of composite components. These hybrid constructs make the model of a system more identical to the concrete system. The main benefit of these components is that they combine software and hardware components to guarantee a flexible modelling process for software/hardware architecture interactions.

4. Methodology

As a methodological frame, this paper uses a design and creation research strategy that mainly focuses on developing new artefacts for creating new knowledge [27]. Such artefacts include constructs, models, methods and instantiations [6]. The focus lies either on the artefact itself (e.g., the IT application incorporates a new theory), the artefact as a vehicle to create new knowledge (e.g., the IT application in use), or on the process to build an artefact to create knowledge [19].

In this context, our research focuses on the creation process of an artefact (i.e., an OPC UA/DDS gateway) with AADL as the model language. The aim is to create knowledge about the suitability of AADL for industrial automation and specifically for reducing development efforts. Partly following the UML systems development methodology [29], the OPC UA/DDS gateway specification provided the foundation of the modelling process. As a modelling environment, the Open Source AADL Tool (OSATE) [5], and for code generation, the Ocarina tool suite [35] were used.

The evaluation criteria are the accuracy of the modelled gateway compared to the specification and the accomplished functionality. As a framework for this assessment, the gateway specification [28] stipulates conformance points to judge the quality of implementation. Moreover, by creating and testing different gateway instances, the configuration and deployment abilities of AADL are assessed.

5. Modelling an OPC UA / DDS Gateway

As indicated beforehand, modelling an OPC UA/DDS middleware gateway, specified by the Object Management Group (OMG) [28], builds the base for evaluating the suitability of AADL for IIoT. Middleware gateways represent a form of gateways that deal with complex mappings between advanced middleware systems essential in modern CPS and are, therefore, a suitable evaluation artefact. Moreover, such gateways can operate on different target systems (e.g., server, fog node) and different positions in the network. In fog computing environments, gateways are dynamically reconfigured and deployed [11].

5.1. OPC UA / DDS Gateway

In 2018, the OMG published a specification on bridging OPC UA and DDS in a standardised, interoperable, vendor-independent, and configurable way [28]. OPC UA is an object-oriented client/server-based information exchange standard for industrial automation systems that supports various IP-based networking technologies [26]. The request-response message pattern allows clients to access a server's services for manipulating *Nodes* by addressing their unique *NodeId*. DDS is a fourth-generation middleware that relies on the publish/subscribe communication model. With its data-centric approach, DDS transports data with high performance and consistency via a global data space accessible by all strictly decoupled subscribers and publishers [1].

The OPC UA/DDS gateway specification defines two separate bridges that connect the OPC UA and the DDS domain but operate unidirectionally based on the intended information control flow. In one direction, DDS applications can communicate with different OPC UA servers, navigate through their address space, manipulate the connected nodes and data. Complementarily, the second bridge allows OPC UA client applications to participate in the global data space of the DDS system as a publisher or a subscriber. The following subsections describe the modelling in AADL of the main gateway components. The complete model is available in a Git repository [12].

5.2. General Structure of the AADL System Model

Figure 1 presents an overview of the general structure of the OPC UA/DDS gateway as an AADL system model. The model consists of an array of multiple OPC UA to DDS and DDS to OPC UA bridges to create a bi-directional gateway. The advantage of multiple bridges lies in configuring each bridge individually (i.e., to enable separate communication) and improving the overall performance and latency (i.e., processing many requests and responses simultaneously). Moreover, the system model includes several other AADL models representing relevant components of the OPC UA and DDS middlewares. For OPC UA, there is a need for modelling OPC UA clients and servers and several datatype specifications. The specifications are distributed over several AADL files and categorised by OPC UA Service Set assignments. For DDS, the model incorporates general DDS datatypes specifications and the AADL model for DDS applications.

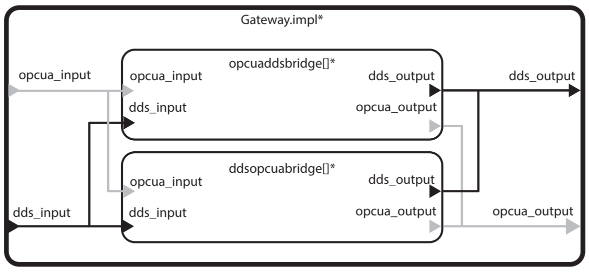

Fig. 1. General AADL model structure of the OPC UA/DDS gateway

5.3. OPC UA to DDS Bridge

The OPC UA to DDS bridge establishes the communication between participants of the DDS domain on one side and OPC UA servers on the other side. DDS applications can access (read, write, modify) and receive notifications on resources provided through the address space of an OPC UA server. A DDS application sends a request to the bridge's DDS endpoint, and the bridge forwards the request to the integrated OPC UA client, which in turn sends it to the requested OPC UA server. The DDS application would receive an answer from the OPC UA server if the request was successful. In principle, the integrated DDS endpoints act as an interface for DomainParticipants of DDS to publish and subscribe to topics and receive notifications on these topics. The OPC UA clients operate in the same way for the OPC UA servers on the other side. For both sides, the bridge is a standard participant in the corresponding DDS or OPC UA environment. To efficiently and correctly handle incoming requests and responses, the OPC UA to DDS gateway specification defines three types of mappings (*OPC UA to DDS Type System, OPC UA to DDS Service Sets*, and *OPC UA to DDS Subscription*) for all kind of interactions and communication. In AADL, those mappings were generically implemented as DDS domain participants that specify DDS data types as I/Os and OPC UA clients with OPC UA datatypes as I/Os. The OPC UA/DDS gateway specification also defines a configuration interface for these mappings so that any types of mappings can directly be adapted and configured to the immediate requirements of the user.

5.4. DDS to OPC UA Bridge

The second bridge provides the opposite communication direction from OPC UA to DDS. An OPC UA client application sends a request to the bridge's internal OPC UA server that forwards the request to the integrated DDS endpoint, which in turn sends it to the appropriate DDS application. If the request is successful, the OPC UA client application receives a response from the DDS application. The OPC UA server acts as an interface for the OPC UA clients of the OPC UA domain and replicates the DDS global data space using the nodes and references of its address space. OPC UA client applications may either use the View Service, the Subscription and MonitoredItems Service, or the Attribute Service Set to publish and subscribe to topics like any other ordinary DDS application.

For the DDS to OPC UA bridge, it is required to implement the DDS to OPC UA type system mappings and a precise information model that correctly represents the global data space of the DDS system within an address space of an OPC UA server. In the AADL model, the same approach was applied in the first bridge, including the potential for adaption and configuration. The OPC UA information model used the additional OPC UA AADL components.

5.5. Addressing Configuration and Code Generation

A benefit of an MDE approach is the possibility to create different implementations on the fly. The AADL model represents a framework that allows adding or removing new mappings and further information for more specific models. To address the stated intention to use AADL for reducing the development effort, the process as depicted in Figure 2 was implemented to instantiate gateways running on standard office computers. In the first step, a parser transforms the generic gateway model into a specific instance of a gateway where an XSD configuration file provides the required information. In this stage, the Ocarina tool suite can generate source code either in C or Ada. As stated before, the generated code is not specific to one platform. For the next step to create a gateway application, further "hard-coded" information is required, such as configuration informa-

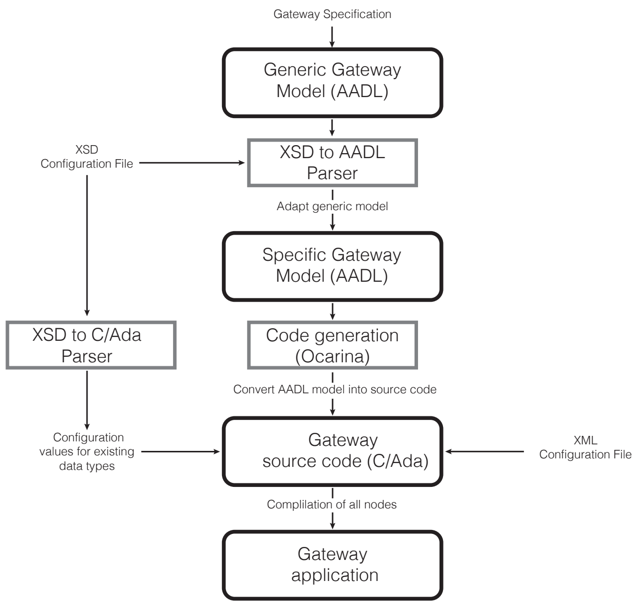
Gateway Specification
Generic Gateway Model (AADL)
XSD Configuration File
XSD to AADL Parser
Adapt generic model
Specific Gateway Model (AADL)
XSD to C/Ada Parser
Code generation (Ocarina)
Convert AADL model into source code
Configuration values for existing data types
Gateway source code (C/Ada)
XML Configuration File
Complilation of all nodes
Gateway application

Fig. 2. Gateway generation process

tion (e.g., server addresses, subscription topics). An XML file provides the configuration part concerned regarding the gateway environment, such as the network or the connected nodes. Additionally, it is necessary to specify internal data types that depend on the XML configuration. For this reason, the XSD file needs to be transformed into C/Ada to address this issue. The last step compiles the code into a gateway application.

6. Discussion

As mentioned beforehand, this research aims to assess the capabilities and suitability of AADL for IIoT and CPS. The experiences and obtained results made while implementing the OPC UA/DDS gateway indicate that AADL possesses the capabilities to model complex software artefacts. As a basis for this statement, the created AADL gateway model was compared with the conformance points specified by OMG.

In essence, there are four (two for each bridge) conformance points to assess the implementation quality and accuracy of the gateway. One pair provides basic and complete compliance requirements for the corresponding bridge. Fulfilling all conformance points symbolises a high accuracy comparing to the specification. For basic compliance, the first pair (OPC UA to DDS bridge) requires the implementation of the System and Service Set mappings, while for complete compliance, the full adherence to the Subscription Model is mandatory. For the second DDS to OPC UA bridge, basic compliance requires the DDS Type System and the DDS Global Data Space Mapping. Complete compliance adds a sub-clause demanding the reading of historical data from instance nodes.

Both modelled bridges fully comply with all conformance points, including the sub-clauses. Moreover, the model outperforms as it provides additional AADL packages such as a standalone OPC UA Client, the OPC UA View, Query and Method Service Sets, and OPC UA Attributes. This finding mirrors experiences made by other researchers that used AADL for modelling embedded systems in the avionics domain [38, 40] or when used to model a fog computing platform [4]. Problems to model software layers as described by Delanote et al. [7] could not be identified. However, implementing the OMG specified hybrid data types is challenging as required information is only available during the compilation of the final instance.

A second step assessed the capabilities of AADL to generate executable code and specific gateway instances based on the presented process in Figure 2. However, due to an internal error in the Ocarina toolchain, it was impossible to compile the complex data type mappings. The problem occurs during the transformation of the AADL code to C programming language, a problem other authors did not experience [21]. Until the final publication of this paper, it was not possible to identify the issue in Ocarina, and it was necessary to continue the trials with a reduced gateway model with fixed mappings. In further studies, the aim is to try other tools as described by Wehrmeister et al. [37] to assess the code generation capabilities of AADL.

In addition to the problems described, the process for creating specific gateway instances led to consistent results with the reduced gateway model comparable with results obtained by [39]. An adjusted instance of the generated gateway provided the foundation for a study to assess the gateway's capacities to transport automation data from OT to an IT environment [9]. Another issue in the proposed process is the use of XSD and XML files. Due to the complexity of the internal gateway mappings and parameters, the XSD and XML files require considerable formulation effort. This particular finding initiated the development of an external configurator to ease the creation process for the configuration files. The configurator and the gateway instance are part of a study presented recently [10]. Since the toolchain currently generates C-code without specific platform considerations, the suitability of AADL for platform-specific code cannot be conclusively assessed.

In summary, results indicate that the model language AADL provides capabilities to model complex software artefacts and specific gateway models. Conclusively answering the suitability of AADL for code generation and platform-specific deployment in IIoT environments is not reasonable and requires further research.

7. Conclusion

The paper explores the feasibility of using the modelling language AADL for IIoT and CPS applications by modelling an OPC UA/DDS gateway. It was possible to model the complex gateway in AADL with full compliance to the specification, yet the code generation for gateway instances was unsuccessful. The results indicate the suitability of AADL and contribute to the MDE body of knowledge applied in IIoT. Further studies need to address the code generation with AADL to continue the assessment and investigate other code generation tools specifically suitable for resource-constraint devices.

Acknowledgements

The research leading to these results has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 764785, FORA—Fog Computing for Robotics and Industrial Automation.

References

[1] Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M., Ayyash, M., 2015. Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications. IEEE Communications Surveys Tutorials 17, 2347–2376.

[2] Bardaro, G., Semprebon, A., Chiatti, A., Matteucci, M., 2019. From Models to Software Through Automatic Transformations: An AADL to ROS End-to-End Toolchain, in: 2019 Third IEEE International Conference on Robotic Computing (IRC), pp. 580–585.

[3] Bardaro, G., Semprebon, A., Matteucci, M., 2018. A Use Case in Model-Based Robot Development Using AADL and ROS, in: Proceedings of the 1st International Workshop on Robotics Software Engineering, Association for Computing Machinery, New York, NY, USA. pp. 9–16.

[4] Barzegaran, M., Desai, N., Qian, J., Tange, K., Zarrin, B., Pop, P., Kuusela, J., 2020. Fogification of electric drives: An industrial use case, in: 2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), pp. 77–84.

[5] Carnegie Mellon University, Software Engineering Institute, 2016 - 2020. Osate. Website: https://osate.org/. Accessed 29 June 2020.

[6] Checkland, P., 2000. Soft Systems Methodology: A Thirty Year Retrospective. Systems Research and Behavioral Science 17, S11–S58.

[7] Delanote, D., Van Baelen, S., Joosen, W., Berbers, Y., 2008. Using AADL to Model a Protocol Stack, in: 13th IEEE International Conference on Engineering of Complex Computer Systems (iceccs 2008), pp. 277–281.

[8] Demathieu, S., Thomas, F., André, C., Gérard, S., Terrier, F., 2008. First Experiments Using the UML Profile for MARTE, in: 2008 11th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC), pp. 50–57.

[9] Denzler, P., Ramsauer, D., Kastner, W., 2021a. Tunnelling and Mirroring Operational Technology Data with IP-based Middlewares, in: Presented at the 2021 22nd IEEE International Conference on Industrial Technology (ICIT), pp. XXX–XXX.

[10] Denzler, P., Ramsauer, D., Preindl, T., Kastner, W., 2021b. Communication and container reconfiguration for cyber-physical production systems, in: Accepted at the 2021 26th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), pp. XXX–XXX.

[11] Denzler, P., Ruh, J., Kadar, M., Avasalcai, C., Kastner, W., 2020. Towards Consolidating Industrial Use Cases on a Common Fog Computing Platform, in: 2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), pp. 172–179.

[12] Denzler, P., Scheuchenstuhl, D., . Readme [Source code]. URL: https://git.auto.tuwien.ac.at/fora/aadl_opcuaddsgateway.

[13] Feiler, P.H., Gluch, D.P., Hudak, J.J., 2006. The architecture analysis & design language (AADL): An introduction. Technical Report. Carnegie-Mellon Univ Pittsburgh PA Software Engineering Inst.

[14] Feiler, P.H., Lewis, B.A., Vestal, S., 2006. The SAE Architecture Analysis Design Language (AADL) a standard for engineering performance critical systems, in: 2006 IEEE Conference on Computer Aided Control System Design, 2006 IEEE International Conference on Control Applications, 2006 IEEE International Symposium on Intelligent Control, pp. 1206–1211.

[15] Friedenthal, S., Moore, A., Steiner, R., 2006. OMG systems modeling language (OMG SysML) tutorial, in: INCOSE Intl. Symp, pp. 65–67.

[16] Hametner, R., Winkler, D., Östreicher, T., Biffl, S., Zoitl, A., 2010. The adaptation of test-driven software processes to industrial automation engineering, in: 2010 8th IEEE International Conference on Industrial Informatics, pp. 921–927.

[17] Harp, Derek R and Gregory-Brown, Bengt, 2015. IT / OT Convergence Bridging the Divide. NexDefense , 23.

[18] Hästbacka, D., Vepsäläinen, T., Kuikka, S., 2011. Model-driven development of industrial process control applications. Journal of Systems and Software 84, 1100 – 1113.

[19] Hevner, A.R., March, S.T., Park, J., Ram, S., 2004. Design Science in Information Systems Research. MIS Quarterly 28, 75–105.

[20] Hugues, J., Perrotin, M., Tsiodras, T., 2008. Using MDE for the Rapid Prototyping of Space Critical Systems, in: 2008 The 19th IEEE/IFIP International Symposium on Rapid System Prototyping, pp. 10–16.

[21] Hugues, J., Zalila, B., Pautet, L., Kordon, F., 2008. From the Prototype to the Final Embedded System Using the Ocarina AADL Tool Suite. ACM Trans. Embed. Comput. Syst. 7.

[22] Kim, B., Phan, L.T.X., Sokolsky, O., Lee, L., 2013. Platform-dependent code generation for embedded real-time software, in: 2013 International Conference on Compilers, Architecture and Synthesis for Embedded Systems (CASES), pp. 1–10.

[23] Lee, E.A., 2008. Cyber Physical Systems: Design Challenges, in: 2008 11th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC), pp. 363–369.

[24] Lv, D., Qiu, G., Cai, Y., Lou, Y., Zhang, T., 2020. Research on Modeling of Reconfiguration for Integrated Electronic System, in: 2020 3rd International Conference on Unmanned Systems (ICUS), pp. 158–161.

[25] Ma, L., Gui, S., Luo, L., Yin, L., 2008. Research of Automatic Code Generating Technology Based on AADL, in: 2008 International Conference on Embedded Software and Systems Symposia, pp. 136–141.

[26] Mahnke, W., Leitner, S.H., Damm, M., 2009. OPC unified architecture. Springer Science & Business Media.

[27] March, S.T., Smith, G.F., 1995. Design and natural science research on information technology. Decision Support Systems 15, 251–266.

[28] Object Management Group, . OPC UA/DDS Gateway Version 1.0. URL: https://www.omg.org/spec/DDS-OPCUA/1.0. (2020, October 15).

[29] Rumbaugh, J., Jacobson, I., Booch, G., 1999. Unified Modeling Language Reference Manual. Addison Wesley Longman, Inc.

[30] Sauter, T., Lobashov, M., 2011. How to Access Factory Floor Information Using Internet Technologies and Gateways. IEEE Transactions on Industrial Informatics 7, 699–712.

[31] Sauter, T., Soucek, S., Kastner, W., Dietrich, D., 2011. The Evolution of Factory and Building Automation. IEEE Industrial Electronics Magazine 5, 35–48.

[32] Schmidt, D.C., 2006. Guest Editor's Introduction: Model-Driven Engineering. Computer 39, 25–31.

[33] Sisinni, E., Saifullah, A., Han, S., Jennehag, U., Gidlund, M., 2018. Industrial Internet of Things: Challenges, Opportunities, and Directions. IEEE Transactions on Industrial Informatics 14, 4724–4734.

[34] Suo, D., An, J., Zhu, J., 2011. AADL-based Modeling and TPN-based Verification of Reconfiguration in Integrated Modular Avionics, in: 2011 18th Asia-Pacific Software Engineering Conference, pp. 266–273.

[35] Telecom ParisTech, 2003 - 2009. Ocarina Toolsuite. Website: https://ocarina.readthedocs.io/en/latest/. Accessed 29 June 2020.

[36] Vergnaud, T., Pautet, L., Kordon, F., 2005. Using the AADL to Describe Distributed Applications from Middleware to Software Components, in: Vardanega, T., Wellings, A. (Eds.), Reliable Software Technology – Ada-Europe 2005, Springer Berlin Heidelberg, Berlin, Heidelberg. pp. 67–78.

[37] Wehrmeister, M.A., 2020. Generating ROS-based Software for Industrial Cyber-Physical Systems from UML/MARTE, in: 2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), pp. 313–320.

[38] Zhang, L., 2014. Convergence of physical system and cyber system modeling methods for aviation cyber physical control system, in: 2014 IEEE Int. Conference on Information and Automation (ICIA), pp. 542–547.

[39] Zhang, Q., Wang, S., Liu, B., 2016. Approach for integrated modular avionics reconfiguration modelling and reliability analysis based on AADL. IET Software 10, 18–25.

[40] Ziani, A., Hamid, B., Trujillo, S., 2011. Towards a Unified Meta-model for Resources-Constrained Embedded Systems, in: 2011 37th EUROMICRO Conference on Software Engineering and Advanced Applications, pp. 485–492.