



Graphical modeling notation for data collection and analysis architectures in cyber-physical systems of systems



Emanuel Trunzer*, Anne Wullenweber, Birgit Vogel-Heuser

Institute of Automation and Information Systems, Technical University of Munich, Garching, Germany

ARTICLE INFO

Keywords:

Big data in robotics and automation
 Factory automation
 Cyber-physical systems of systems
 Systems modeling
 Data collection
 Graphical modeling notation

ABSTRACT

Industrie 4.0 and data analytics blur the separation of operational and information technology that prevailed for industrial automation over the last decades. Decentralized control systems for production plants and robot cells collaborate actively with higher-level systems for big data analytics. In parallel, the complexity of designing and operating a system architecture for data collection and analysis increases dramatically as more experts from different domains get involved. Graphical modeling notations facilitate the design process by formalizing implicit knowledge, but currently do not exist for the combined description of field layer and data analytics. Modeling the system architecture, relevant constraints, and requirements early during the design process can increase the efficiency of the system development and deployment, especially as experts with various backgrounds are involved. In this contribution, a new graphical notation is introduced and evaluated in three industrial case-studies. The notation describes the underlying hardware and software components of cyber-physical systems of systems, the flow of data, and relevant constraints. The evaluation proved that the notation is powerful in supporting the engineering of data collection and analysis architectures in industrial automation. Future work is related to extending the scope of the modeling approach to include safety applications and real-time considerations on the field level.

1. Introduction

Industrial automation is currently influenced and challenged by various trends like Industrie 4.0 (I4.0), application of big data principles, and increasing demand for flexibility in production [1]. In contrast to the classical system hierarchy, which clearly separates plant control from superordinate IT systems, new paradigms and developments such as Ethernet TSN and data analytics rapidly break up this layered structure. This dramatically increases the complexity of designing and maintaining connected cyber-physical systems of systems (CPSoS) [2]. For instance, complex condition monitoring applications for multiple robots require the expert knowledge of production engineers, as well as the algorithmic understanding of data analysts and the know-how of IT architects to size networks and platforms [3–5]. One of the central aspects of I4.0 is, therefore, the integration of data and systems along the life cycles of CPSoS, their hierarchical structure, and supply chains [6,7].

However, due to their different understandings of specific terms in these domains, diverse and conflicting terminology, and differing educational backgrounds, information is lost at the interfaces of the disciplines [8]. Visual notations and representations support

information exchange, structure existing knowledge, and increase understanding, especially for complex applications [9]. Such a graphical notation does not exist yet for the field of data collection and analysis architectures for CPSoS, where operational technology (OT) and information technology (IT) coalesce with their very specific characteristics and an integration of data from various industrial information systems is necessary. The contribution of this paper is, therefore, to propose a graphical notation for data collection and analysis architectures that supports the engineering of industrial big data analytics for connected and distributed automation systems.

The authors define the term *system architecture* as a connected system of machines, services, and higher-level IT applications, as well as networks with their respective hardware and software components [10]. Additionally, the term *data collection and analysis architecture* is defined as such a system architecture with the purpose of collecting and integrating data from machines in the field as well as from superordinate IT systems for data analysis.

The remainder of this contribution is structured as follows: in the next section, the requirements for graphical notations for data collection and analysis architectures are derived. Next, in the state-of-the-art, the lack of graphical notations for data collection and analysis

* Corresponding author.

E-mail addresses: emanuel.trunzer@tum.de (E. Trunzer), anne.wullenweber@tum.de (A. Wullenweber), vogel-heuser@tum.de (B. Vogel-Heuser).

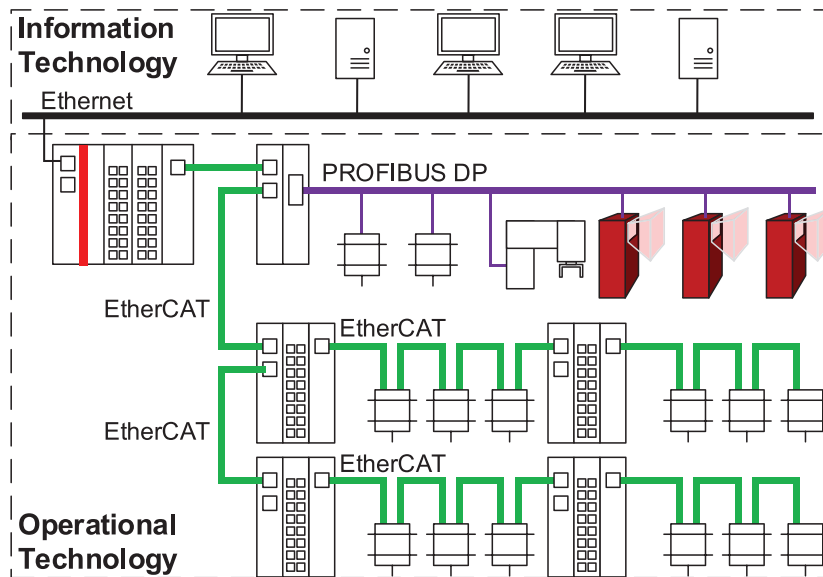


Fig. 1. Simplified network layout of a typical CPSoS consisting of IT and OT domains with various types of connected devices and networks.

architectures is discussed. Afterward, a notation concept is developed and evaluated in three industrial case-studies. In the last section, the results are summarized and an outlook on further work is given.

2. Requirements for a modeling notation for data collection and analysis architectures

Data collection architectures deal with various heterogeneous hardware devices and related software in IT and OT that are connected through different types of networks and fieldbuses (cf. Fig. 1 for an example). The OT domain, with its focus on control of production cells and robots, is bound to deterministic real-time communication between the distributed systems. Violation of real-time-constraints could cause malfunctions and harm to humans [11]. On the other hand, data analytics and superordinate systems from the IT domain often do not require hard real-time communication as a comparably high amount of latency t_L and jitter σ_j^2 are acceptable. Therefore, such systems are often connected via a non-real-time Ethernet. Collecting and analyzing data from such distributed and networked systems of systems is challenging because of the large number of connected systems (up to several hundred) with often more than 1000 in- and outputs per system. Moreover, the complexity of the underlying constraints (e.g., acceptable latency, transmission rates of networks, constrained computing power) needs to be considered [12]. A notation has to provide the means to model this multitude of devices, networks, and software functionalities (R1).

For systems with a significant number of connected devices and analyses, the flow of data becomes very complex. For both, IT architects, as well as data analysts, following the route of data through the systems in order to size hardware nodes or determine influences on the quality of data becomes almost impossible using only a system description. Consequently, the flow of data through the system should be captured by the modeling notation. Additionally, relevant information about the type of data (integer, float, etc.), as well as the state of data (batch or streamed data), has to be represented (R2).

In order for the notation to serve as an approach during the engineering and operation of a system, the requirements for, as well as the properties of, the system need to be represented. Experts from different domains can state distinct types of requirements which the data collection and analysis architecture should fulfill. For instance, while a data analyst can define the required sampling rate f_s of a variable, an IT architect is concerned about the security of data transmission. To evaluate the performance of a system in operation, the actual properties

have to be compared to the defined requirements. Hence, the means of stating requirements and properties should be part of the modeling notation (R3).

The clarity and applicability of the notation are of high importance, as the notation has to be usable by application engineers with their distinct fields of expertise (IT architects, data analysts, programmers), who are not familiar with all the aspects modeled (R4). In this contribution, the principles of Moody [9] are considered as best-practices for graphical notations because they deal explicitly with graphical notations.

Table 1 summarizes the requirements.

3. State-of-the-art in CPSoS and data analysis architecture modeling notations

Numerous researchers have investigated how I4.0 architectures provide data collection from systems, but only a small number of modeling approaches exist.

Architecture modeling languages provide a basis for the description of system architectures and help during the design phases. For instance, the international standard ISO/IEC/IEEE 42010 defines an ontology for architecture descriptions. An architecture description consists of several architecture views that address specific concerns, such as functionality, usage, system features, performance, and modularity [13]. Architecture viewpoints and model kinds constitute an architecture description

Table 1
Requirements for the graphical modeling notation.

Req.	Description
R1	Description of System Architecture Notation to model the relevant hardware (e.g., PLCs, computer systems, bus couplers), software (data analysis, forwarding, routing), and networks.
R2	Data Flow Description between Connected Systems Connected systems form complex networks where data is flowing between data producers, consumers, and manipulators. Notation to model these flows and types of data (data type, type of flow).
R3	Annotations for Requirements and Properties Timing, network protocols, and encryption requirements are relevant for system architectures. When in operation, the actual behavior can be captured. The notation needs to reflect these aspects.
R4	Clarity and Applicability for Application Engineering Clarity of the notation are of major importance for accelerating the engineering of system architectures.

language (ADL).

Medvidovic and Taylor [14], as well as Malavolta et al. [15], compare several existing ADL approaches. There is a lack of languages that can be used for distributed control systems in conjunction with data analysis. For instance, ArchiMate [16] defines an enterprise architecture modeling language that is capable of modeling information flows. However, aspects for modeling the distributed hardware architecture, specific timing requirements, and data formats for interoperability are needed. On the other hand, the Architecture Analysis & Design Language (AADL) [17], used for real-time embedded systems, includes timing requirements, but lacks consideration of legacy devices and does not provide a graphical notation on top of the generic UML (Unified Modeling Language).

The Object Management Group (OMG) specifies a UML profile for Modeling and Analysis of Real-Time and Embedded Systems (MARTE) [18]. The profile is tailored to describe the timing and non-functional requirements of embedded software and hardware systems. While the profile can be used to model the internals and behavior of embedded systems in-depth, including scheduling, hardware resources, and allocation, specific consideration of data analysis as well as distributed architectures are lacking. Additionally, as UML-MARTE is a UML profile, it lacks graphical elements added to the standard UML symbol library and relies on the available set of symbols in UML.

Other profiles for UML are UML-RT [19,20] for modeling of real-time systems, UML4IoT [21] for the IoT domain, and the approach by Tekinderdogan et al. [22] for the modeling of Data Distribution Service (DDS) deployment alternatives. All three lack support for data analysis as well as a graphical notation. Only the latter approach considers the communication architecture but is limited to DDS.

Terzic et al. [23] present a domain-specific language (DSL) consisting of a textual and a graphical representation. The DSL can be used to describe service-oriented architectures (SOA) that communicate over web services, but it lacks the means to describe hardware and networks.

Greifender and Frey [24] model networked control systems with a focus on communication-based delay. However, they do not capture the system with hardware and software components as well as additional annotations.

In [25], as well as in further work by the same research group [26], a modeling notation for distributed control systems (DCS) in the field of industrial automation is proposed. In another work [8], an adapted version of the modeling language is used for model-based engineering of DCS. These approaches capture timing requirements but fail to consider the data flows and the superordinate IT infrastructure in data collection and analysis architectures.

In summary, different approaches for describing distributed systems exist in their respective application areas. However, there is no graphical notation in the CPSoS domain that provides elements for describing distributed systems from the field level to data analysis, for properties and requirements of the systems, and data flow through these systems.

4. Graphical modeling notation concept

The concept comprises two viewpoints for separation of the concerns: one for a description of the system architecture with its hardware and software components and networks, the other to model the flow of data through the system. Both viewpoints can be annotated with requirements and properties. Additionally, a data mapping table allows capturing information and interdependencies related to data.

4.1. System viewpoint

The system viewpoint is based on the work by Vogel-Heuser et al. [25] and extended to capture information about software functionalities, additional types of signals (variables without a link to hardware signals and models) and a unique labeling system. Table 2

Table 2

System view: elements for the modeling of hard- and software components.

Symbol	Description
	Digital (square) or analog (circle) sensor/actuator signals.
	Calculated data/variable based on other signals/data.
	Model with parameters, e.g., from data analysis.
	Signal or variable element with type of signal and corresponding data type. Can be grouped to lists. UID placeholder to be replaced by a concrete value referring to the signal. Bus coupler with UID <i>BCI</i> for Profibus DP (UID of interface <i>DP1</i>) with eight digital inputs and eight digital outputs.
	PLC (UID <i>Machine3PLC</i>) with Ethernet interface and Profibus DP master. UID of Ethernet interface is <i>X1P1</i> and the UID of Profibus DP master interface <i>X2</i> .
	Computer (UID <i>Workstation</i>) with a single Ethernet adapter (<i>ETH1</i>), running a data translation software functionality (<i>Adap</i>).
	Network or fieldbus (thick) with connections to the devices (thin) and empty UID identifier.

shows a selection of the symbols. On the OT level, symbols for signals from analog and digital sensors and actuators are included. Additionally, values calculated from analysis on machine controllers, as well as the models (parameter sets) that are part of the data analysis, can be modeled. Annotations (arrows) link the signals and models with to location of their generation that could either be an in-/output or software functionality. The annotations include the type of signal/data and a unique name on the specific device. Solid rectangles represent hardware devices. Network and fieldbus interfaces share a diamond shape inside these rectangles, while master interfaces, commonly found in fieldbuses, are represented by a double diamond. If a device exhibits computational capacity, this is represented on the very left of a device symbol, either with the label PC or PLC. Software can only be executed on such devices, and a dash-dotted rectangle with a label is used for the respective software functionality (cf. Table 3). Networks are depicted as solid lines with thinner lines connected to the interfaces. All elements contain unique identifiers (UIDs) that identify the respective hardware systems, software functionalities, networks, and data elements.

4.2. Data flow viewpoint

The data flow viewpoint is inspired by the data flow notation by DeMarco's structured analysis (SA) [27], as well as the adaption SA/RT for real-time systems by Hatley and Pirbhai [28]. The elements of the data flow viewpoint are given in Table 4. The notation differentiates between data sources where data originates, and data sinks where data is not forwarded any longer. Transducers have the ability to forward or modify incoming data, but must provide all of these as output again. In addition, transducers may calculate additional values that are

Table 3
List of possible software functionalities.








Functionality	Description
AGGR	Aggregation of data from different sources without changes in protocol, format, and semantic.
DA	Data analysis functionality for extracting information and knowledge from data. May calculate values and models.
FORW	Software functionality to forward data to another system.
LEG	Existing legacy software component with an internal logic that may generate, consume, or manipulate data. If a legacy component can be decomposed into other software functionalities, these may be used instead of the LEG label.
MC	Machine control, typically a control application, running on a PLC. May calculate internal values from measurement signals.
ROUT	Message routing functionality to enable communication between the heterogeneous systems. Typically a middleware.
STOR	Storage functionality for saving data, information, and models.
TRANS	Translation between data protocols, formats, and semantics. Used to adapt incompatible and legacy systems.
VISU	Visualizes data for users (human-machine interface).

forwarded as well. A special node called *sinksources* ends the flow of entering data and does not forward the input data any longer, providing a new set of calculated data or models to other nodes instead. The notation differentiates between different types of flows relevant to data analysis. While streamed data is continuously flowing from one component to the other, often in small packages, batched data is a large bundle of data moved only from time to time. A data stream may be transformed into a batch of data by a system with a storage (buffer) functionality. All modeled elements need to be related to a concrete software functionality that executes the data handling or the networks/hardware systems used for transporting the data.

4.3. Annotations

Annotations allow experts from different domains to structure their requirements for the data collection architecture. Additionally, the actual behavior of an architecture in operation can be modeled as properties. This is especially useful when working in brownfield environments where specific characteristics are not easily customizable. Table 5 summarizes the different annotation elements. The notation differentiates between three types of property/requirement types based on the shape of the annotations: Time-related information, for instance communication latencies or sample rates, architectural information that defines types of data storage or scalability of components, and data flow-related information on protocols, semantics, or encryption. An

Table 4
Data flow view: elements for the modeling of data flows.

Symbol	Description
	Indicates that the component serves as source for the data flow. The element refers to a concrete software functionality from the system viewpoint hosted on a hardware system. The naming scheme for UID is (UID of the hardware system).(UID of SoftwareFunctionality).
	Specifies that the component serves as sink for the data flow. The element refers to a concrete software functionality from the system viewpoint hosted on a hardware system. The naming scheme for UID is (UID of the hardware system).(UID of SoftwareFunctionality).
	Indicates that the component serves as transducer for the data flow, i.e., that it provides a software functionality to convert, store, or forward data. May calculate additional data. The element refers to a concrete software functionality from the system viewpoint hosted on a hardware system. The naming scheme for UID is (UID of the hardware system).(UID of SoftwareFunctionality).
	Specifies that the component serves as sinksources for the data flow, i.e., that it receives data, processes it, and distributes these new data without distribution of raw data. The element refers to a concrete software functionality from the system viewpoint hosted on a hardware system. The naming scheme for UID is (UID of the hardware system).(UID of SoftwareFunctionality).
	Signal or variable element with type of signal. Can be grouped to lists.
	Data stream in the form of a continuous stream. The element refers to the network that transports the data or the respective hardware system if two software functionalities on the same system exchange data.
	Data stream in the form of batch packages. The element refers to the network that transports the data or the respective hardware system if two software functionalities on the same system exchange data.

example of the categorized information is given in Table 6. Properties are shown as single-bordered and requirements as double-bordered. An additional element describes the computational capabilities of devices if present. If an annotation refers to a system (hardware/software or node in the data flow), it is connected to this element by a gray line ending with a circle symbol. Alternatively, annotations that refer to networks or data flows use a square as the line end symbol. Signals are connected to a system by an arrow. To convey additional information, a UML comment element is used.

Consistency between the two viewpoints is ensured by giving unique identifiers to all components. Labels are placed on systems, networks, data flows, and nodes in the data flow viewpoint. For instance, a data source in the data flow viewpoint shares the same unique identifier as the hardware or software component where it was generated. Additionally, all signals, models, and variables are labeled with unique identifiers. In this paper, namespaces will be used to relate signals to the hardware and software. If a diagram becomes too large, it can be distributed over sub-diagrams. In this case, the sub-diagrams require unique identifiers. Connections between the diagrams can then be made with a special arrow symbol including the name of the diagram to which the connection relates (see Table 5, last two elements).

4.4. Data mapping table

As data can be modified on its way through the architecture,

Table 5
Annotations: properties, requirements, and linking elements.

Symbol	Description
	Actual communication or process-related time behavior.
	Specified communication or process-related time behavior.
	Actual architecture-related behavior of networks, hardware or software.
	Specified architecture-related behavior of networks, hardware or software.
	Actual data flow-related behavior of networks, hardware or software.
	Specified data flow-related behavior of networks, hardware or software.
	Annotation to describe actual computational capabilities of devices. Top line gives the manufacturer and name of the device (if applicable, otherwise the internal name of the device). Includes main characteristics of processing unit as well as available memory and storage. Can be used with a double outer line to represent a specification. Adapted from Hashemi et al. [29].
	Specification/property referring to a network or segment.
	Specification/property referring to a hardware component or software functionality.
	Arrow to relate signals, variables, and models to systems or nodes. Shape starts at variable element and ends with arrow at the system or node.
	Element for additional, non-formalized information. Based on the UML comment construct. Can be used to describe, for instance, the manipulation carried out by a translator or a data analyzer.
	Symbol to distribute networks on multiple sheets (system viewpoint). Used as an extension of a thick network line, pointing out from the drawing sheet. Arrow has to be used on both sheets.
	Symbol to distribute information on multiple sheets in data flow viewpoint. Arrow links a flow of information or annotations with another flow on a different sheet. Arrow has to be used on both sheets.

traceability of all variables has to be ensured. Therefore, the concept of data dictionaries [27,28] is adapted as so-called data mapping tables. The table includes the UID of the respective data element and allows the statement of additional information. For instance, variables may be identified by specific variable names inside systems instead of their UID.

In contrast to the graphical elements presented previously, the data mapping table includes no graphical representation but serves as a dictionary to collect and structure additional information on data elements. Table 7 summarizes the columns of the data mapping table.

5. Evaluation of the notation in three industrial CPSoS case-studies

The evaluation of the graphical notation is carried out in industrial case-studies. Three different and typical applications of a data collection architecture in industrial practice are modeled with the help of industrial experts from different domains. The first use-case (A) uses a combined cloud and edge architecture for anomaly detection. Four to five production plants with one PLC each and several hundred in- and outputs per PLC are connected to a common cloud environment. The two other use-cases (B and C) describe alarm management systems for two kinds of production machines that support operators by preventing alarm floods and finding their root-causes. In use-case B, approximately 500,000 alarms are generated per year of operation with 200 distinct types of alarms. The alarm management system of use-case B is hosted in a public cloud by the original equipment manufacturer (OEM) of the machine, which offers additional diagnostic services. Therefore, the alarm messages of several hundreds of these machines, scattered over multiple customers and production sites, have to be transferred to the public cloud. In use-case C, the hosting of the alarm management system follows a hybrid approach with both private and public clouds. Customers can analyze data in their private cloud to ensure privacy. As an additional service, the OEM offers to combine data with datasets from similar machines to improve the quality of the analysis. One machine generates between 3,600,000 and 6,000,000 alarm messages per year, with there being approximately 40 machines per customer and production site. A total of 500 distinct alarms exist. Several customers connect their respective private clouds with the public cloud offered by the OEM. All three case-studies reflect typical applications of data collection and analysis architectures that interact with CPSoS and bridge IT and OT. In this contribution, the complex anomaly detection case-study (case-study A) will be discussed in more detail as an example. At the same time, the models of case-studies B and C can be found in full detail in the supplementary material.

Table 6
Properties and specifications ((T)ime, (A)rchitecture, and (D)ata).

Type	Name	Description
T	JITTER	Information on jitter σ_j^2 for data transmission from source to destination.
T	LATENCY	Latency t_l description for data transmission from source to destination.
T	PROCESS	Time for processing t_{proc} inside a system, for instance analysis or translation of semantics.
T	SAMPLE RATE	Sample rate f_s of a component to scan the data.
T	SAMPLE TIME	Sample time t_s of a component to scan the data.
A	REDUNDANCY	Information on redundancy/duplication of systems in order to improve reliability.
A	N_SAMPLES	Ability of a system to buffer or store a number of n samples.
A	SCALABILITY	Represents the number of similar configurations connected to the same network, while only giving one example.
A	TYPE	Specification on the type of a specific functionality, e.g., <i>stream</i> , <i>batch</i> , <i>hybrid</i> analysis/database.
D	AUTH	Authentication mechanism for establishing communication or data transfer, e.g., password-based or certificate-based.
D	ENCRYPT	Encryption used for securing a data transfer, e.g., AES (Advanced Encryption Standard).
D	PREPROCESS	Distributed preprocessing actions on involved systems, e.g., averaging or resampling.
D	PRIVACY	Privacy level of the transmitted data. This includes for instance normalization, resampling or the introduction of arbitrary noise.
D	PROTOCOL	Underlying protocol used for the communication.
D	SEMANTIC	Description of the underlying data semantic during transmission.

Table 7
Data mapping table.

Column name	Description
VariableUID	Unique identifier of a data element across all systems. It corresponds to the Name-attribute of the SA/RT [28].
SystemSpecificVariableUID	Unique identifier of a data element used in a specific system.
SystemUID	Unique identifier of the system the <i>SystemSpecificVariableUID</i> is valid for. Adapted Member-of-attribute of the SA/RT [28].
DerivedFromVariableUID	If data is based on other data (calculated, derived, composite, or used in the model), the original unique identifier of these data elements (VariableUIDs) can be given here. Otherwise empty. It can be multiple separated by commas.
Description	Optional description of a variable. It corresponds to the Comments-attribute of the SA/RT [28].
Type	Type of the variable, signal, or model, for instance, float, integer, boolean, or model.

5.1. Evaluation procedure

Based on the documentation provided as well as input from technical experts and IT architects, the brownfield production systems, without any additional data analysis, were modeled using the graphical notation. These diagrams were then adapted and extended with the help of data analysts. They stated which data is needed for the analysis and where the models should be deployed (edge, cloud). Additionally, they expressed additional requirements, e.g., for allowed latencies and sample rates.

In the next steps, IT architects drafted the adapted system architecture with additional data analysis components. Supplemental requirements, such as data security (encryption, authentication), communication (protocols, semantics), and system sizing (scalability, capacity of a storage), were specified and added to the diagrams.

The extended diagrams were then discussed with the experts in one joint session. This first part of the qualitative evaluation was to verify the correctness of the models. Afterward, a structured questionnaire with a total of 20 qualitative questions about the clarity of the graphical notation, its syntactic constructs, and its completeness was conducted in the joint session. The questionnaire was divided into four parts: syntax and completeness of the system viewpoint, syntax and completeness of the data flow viewpoint, mapping between the two views and annotation elements, and clarity of the graphical notation. In total, seven experts, which have profound and long-term expertise in their respective fields of application, were questioned. The selected experts are especially qualified to evaluate the notation as they have great industrial experience in the realization of data collection and analysis projects. Additionally, all have an interdisciplinary background from at least two domains relevant to the use-cases (technical experts, data analyst, IT architect, control engineer). The experts are, for instance, heads of information technology or senior engineers for digitization in their respective companies.

5.2. Case-study A: anomaly detection

Due to confidentiality, the boundary conditions and the conceptualized architecture of case-study A are modified slightly for this contribution (for instance, different protocols, single systems connected to other networks, or abstraction of company-specific information related to security configurations). Fig. 2 gives the conceptualized system architecture of the system, while Fig. 3 describes the corresponding flow of data through the system.

A production plant communicates in real-time over a PROFIBUS DP on the field level, which connects the PLC with various bus couplers. Additionally, a PC with a human-machine interface (HMI) hosts a visualization of process values and is connected to the same fieldbus. The PLC is a Siemens IPC627D industrial PC with two Ethernet ports and one PROFIBUS DP master interface. While the variables *Var1* to *Var9* are measured in the field, variables *Var10* to *Var14* are calculated in the machine control program that controls the plant. The PLC runs with a cyclic sample time of $t_s = 5$ ms (cf. Figs. 2(bottom) and 3 (lower left)). Based on *Var1* to *Var14* anomalies during plant operation should be detected.

Data from the PLC is forwarded every 4th cycle ($t_s=20$ ms). Two forwarding functionalities are required on the PLC to forward information to superordinate systems and the visualization on the field level. As a huge amount of data has to be transmitted, and network bandwidth is limited, the raw data is compressed. Therefore, a data analysis functionality (*DA_CP*, cf. system viewpoint (*PLC1*) and data flow viewpoint (*CaseStudy_1_Data_1*, right path of data)) compresses the raw data before transmission. A similar model is executed on the cloud level (*PC3.DA_RC*) to decompress the data. The compression follows the approach presented by Kang et al. [30]. Two similar models are executed in parallel. If the fit of the compression model is good enough, no data is transferred to the decompression model, which calculates values based on the trained model. If the data deviates too much, an update of both models is necessary. In this case-study, a different model from a set of pre-trained models (*PLC1.modelSelection*) is chosen, and an offset is calculated in the compression model based on the raw data. These two variables are then communicated with the decompression model for updating it (data flow viewpoint, right path of data up to *PC3.DA_RC*).

For the communication with superordinate systems and inside the cloud environment (cf. system viewpoint), a middleware approach is considered. For the case-study, MQTT is chosen as the transport protocol with a central broker (routing functionality *RT* on *PC2* in the system viewpoint). Unifying the protocol and semantics of data transfer increases flexibility and simplifies the development and operation of the architecture greatly. As the brownfield infrastructure is not capable of communicating in the common protocol and semantics, *PC1* serves as a data adapter or wrapper between brown- and greenfield. Data adapters can transparently translate between protocols, for instance, OPC UA to MQTT, and different semantics of data, for instance, pressure in psi instead of Pa or different variable names. In the example, *TR1* not only translates the protocol from OPC UA to MQTT, but also alters the semantics of the data (from an information model named *PLC1_UA_1* to *PF.v7*). The respective identifiers represent internal information models that are embedded into the systems and designed during the engineering phase. The translators *TR1* and *TR2* therefore, translate the messages back and forth between the two protocol and semantic representations.

The decompressed data is stored inside the cloud in a storage (*PC2.ST*) for analysis. A data analyzer functionality on *PC4* (*PC4.DA_AD*) is used to train an anomaly detection model. Therefore, based on a manual trigger by the operator through the HMI on *PC5* on the field layer (*PC5.TriggerModel*), the buffered data from *PC2.ST* is analyzed, and a model for anomaly detection is trained. A maximum acceptable latency of $t_{l, \max} \leq 4$ s between data generation and anomaly detection is specified by the OEM to meet the production cycle time of the production machine.

The anomaly detection model could be deployed either on the cloud level (e.g., on *PC3*) or on the edge of the network (e.g., directly on *PLC1*). A cloud environment is more flexible to scale according to the actual needs of an application. However, in this case, the additional latency t_l introduced by data compression, transmission, and decompression, as well as the transfer of the results back to the field level, is unacceptable because a huge amount of data needs to be exchanged and the additional latency constraint. Therefore, the experts decided to

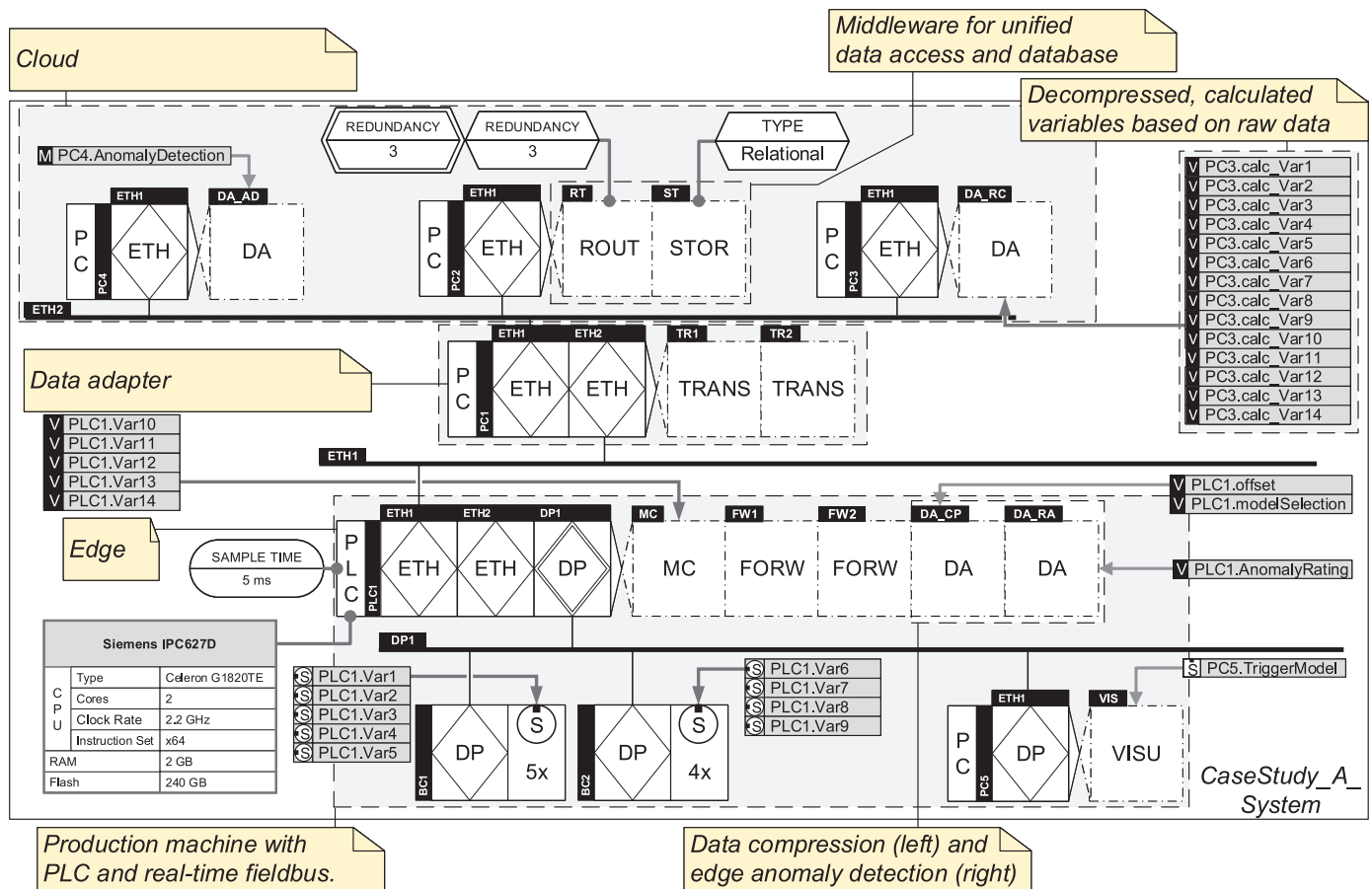


Fig. 2. Combined edge and cloud architecture (case-study A) modeled in the system viewpoint. Production machine with anomaly detection on edge level, cloud environment for model training, and data adapter in between to translate protocol and semantics.

execute the model directly on *PLC1* using the raw data to minimize communication overhead. This implies that the model *PC8.AnomalyDetection* has to be transferred through the different layers of the architecture and translated to the legacy representation (see highlighted path of data in the data flow viewpoint in *CaseStudy_1_Data_2*). The model then analyzes raw data with a sample time of $t_s = 100$ ms and calculates the probability of an anomaly (*PLC1.AnomalyRating*). This value is visualized on the HMI of *PC5* to support the operator. Validation of the actual properties and the fulfillment of the defined requirements have to be carried out either by simulation or in operation through monitoring.

The data mapping table (cf. excerpt in Table 8) serves as a dictionary and allows to trace the specific data elements through the systems. Furthermore, the *DerivedFromVariableUID* column reveals interdependencies between the data elements, e.g., in case a new data element is calculated based on multiple input data elements (cf. *PLC1.AnomalyRating*). The data mapping table can also be used to reflect data mappings that are necessary due to incompatible information models in a translator, e.g., the mapping of two data elements that flow into the translator to a single data element that flows out. In case changes to the system are planned, the data mapping table can be used to identify possible problems and to anticipate them during the redesign phase.

5.3. Case-studies B and C: alarm analysis

The full diagrams that depict the system and data flow viewpoints of case-studies B and C can be found in the supplementary material of this contribution.

5.4. Evaluation results

This section describes the evaluation results based on the summarized expert feedback for the three case-studies A, B, and C.

At first, the completeness of the graphical notation and its elements was evaluated for both viewpoints. All experts pointed out that all relevant information could be captured and structured using the notation. Both the system architecture, with its hardware and software elements (*R1*), as well as the data flow through the system (*R2*), could be expressed and structured. The differentiation between hardware devices and software functionality that is executed on this hardware in the system viewpoint was considered as very helpful to structure the system by all experts. The same is valid for the data flow viewpoint, where the distinction between the types of data handling (source, sink, transducer, and sinksource) is useful to follow the flow of data. Combining the two viewpoints, the data can easily be traced through the associated hardware and software systems. This greatly reduces the complexity when designing and sizing data collection and analysis architectures for all involved parties. Furthermore, the number of different constructs and symbols is relatively low and makes the notation manageable (*R4*).

Concerning the annotations as an essential part of the graphical notation during the specification and design of system architectures, the expert opinion was positive as well (*R3*). Especially for complex connected production cells and robots, latency requirements or protocol constraints can easily be structured and exchanged. All experts considered the categorizing of annotations (time, architecture, and data) as very helpful to separate concerns. Minor concerns were related to the absolute number of symbols, especially in extensive data flow viewpoints (raised by two experts). Here, the number of annotations can be huge in confined space. Grouping of annotations and references to

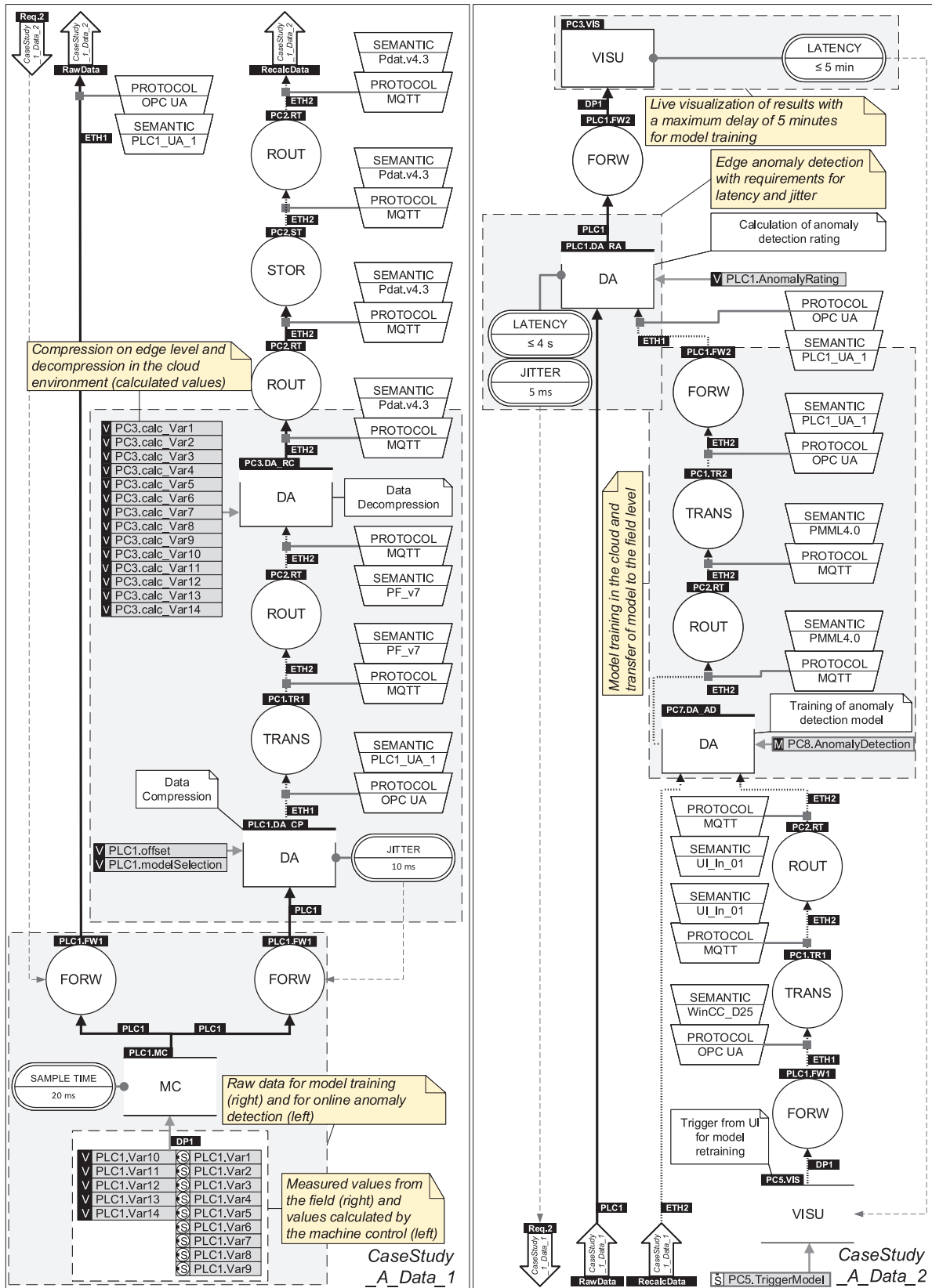


Fig. 3. Case-study modeled in the data flow viewpoint. The diagram is distributed over two sheets for better overview, arrows link the two sheets.

multiple data streams or nodes in the data flow will be considered for future versions of the notation. Furthermore, an interactive graphical editor may overcome this limitation as visibility of elements could be adjusted on-the-fly to provide experts only with the needed

information.

Experts had no problem differentiating between the different types of elements and annotations. Additionally, utilization of the same family of shapes for the specification of properties and requirements was

Table 8

Excerpt from the data mapping table of the case-study. Additional columns and definition of all variables omitted, indicated by (...).

VariableUID	SystemSpecific VariableUID	SystemUID	DerivedFrom VariableUID	...
PLC1.Var1	Var1	PLC1		...
...
PC8.	AnomalyModel	PC8	PLC1.Var1,	...
AnomalyDetection		
PLC1.	AnomalyResult	PLC1.DA_RA	PLC1.Var1,	...
AnomalyRating			...,	...
			PC8.AnomalyDetection	

pointed out as helpful without compromising the *perceptual discriminability* ($R4$) [9]. Nevertheless, four of the experts emphasized that the usage of color would be beneficial for differentiating the symbols of the notation.

In summary, the graphical notation is a powerful approach to structure information during the engineering and operational phases of cyber-physical systems of systems. In contrast to existing approaches, the notation can capture information from the OT as well as the IT domains. It contains constructs for the combined hardware and software architecture as well as the stream of data through all connected systems on different levels of the system hierarchy.

6. Conclusion and outlook

The introduction of I4.0 principles, the convergence of IT and OT, and the application of big data concepts in industrial automation dramatically increase the complexity of CPSoS and information integration from industrial systems. Aspects from a multitude of different disciplines have to be considered, especially when conceptualizing data analysis projects for production plants. Experts from these disciplines, with their varied knowledge and backgrounds, are all involved in the process of designing such an industrial data analysis architecture. In this contribution, a graphical modeling notation, which can support the engineering of interdisciplinary data collection and analysis architectures needed in the future, was developed and evaluated. The notation foresees two different viewpoints that describe the system architecture and the flow of data through the system, as well as a set of annotations to further specify and describe the architecture. The evaluation was carried out based on three different industrial case-studies together with domain experts. The evaluation consisted of a main functional part, in which the completeness of the modeling notation was considered, and a clarity part. The graphical notation proved to be a powerful approach to model distributed control systems in industrial automation. In contrast to existing approaches, it provides both a graphical notation as well as the means to structure information from IT and OT.

Still, the graphical notation can be extended in future work. At first, as it is based on the notation by Vogel-Heuser et al. [25], the modeling constructs could be unified between the different approaches to capture data analysis architectures (this approach), DCS [25] as well as safety aspects [26]. This could extend the scope of the notation dramatically while maintaining almost the same level of complexity.

Second, an extension of the graphical notation towards modeling of system dynamics can be of interest. This would allow experts to capture the internal behavior of systems as well as increase the understanding of conditional data forwarding. As the modeling notation is currently limited to a static view, modeling of the system dynamics can significantly increase the extent of the models and the information content. Possible directions could be the incorporation or adaption of UML activity or sequence diagrams into the modeling notation.

Additionally, a graphical editor is planned to make drafting and using the models easier, allowing the graphical notation to be embedded in the engineering workflow. Moreover, colors could increase the perceptual discriminability, and the visibility of elements could be

made adjustable. A subsequent step would be the introduction of a corresponding meta-model, in order to establish a domain-specific language (DSL). With such a DSL, the modeled information could be formalized and serve as a basis for a model-driven generation of the code for the communication inside the modeled architecture. As the systems, as well as the flow of data and the underlying protocols, can be characterized, the communication logic could be automatically generated to increase efficiency during the realization of data collection and analysis architectures.

CRedit authorship contribution statement

Emanuel Trunzer: Conceptualization, Methodology, Investigation, Writing - original draft, Visualization. **Anne Wullenweber:** Methodology, Investigation, Writing - original draft, Visualization. **Birgit Vogel-Heuser:** Conceptualization, Resources, Writing - review & editing, Supervision.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This contribution is part of project “M@OK” (machine@onlinetknowledge), which has received funding by the Bavarian Ministry of Economic Affairs, Energy and Technology (StMWi) under grant number IUK566/001. The authors thank Continental Reifen Deutschland GmbH, Dorst Technologies GmbH & Co. KG, GROB-WERKE GmbH & Co. KG, and HAWE Hydraulik SE for their participation in the evaluation.

Supplementary material

Supplementary material associated with this article can be found, in the online version, at [10.1016/j.jii.2020.100155](https://doi.org/10.1016/j.jii.2020.100155)

References

- [1] H. Kagermann, Change through Digitization: value Creation in the Age of Industry 4.0, in: H. Albach, H. Meffert, A. Pinkwart, R. Reichwald (Eds.), *Management of Permanent Change*, Springer Fachmedien Wiesbaden, Wiesbaden, 2015, pp. 23–45.
- [2] M. Dotoli, A. Fay, M. Miśkiewicz, C. Seatzu, An overview of current technologies and emerging trends in factory automation, *Int. J. Prod. Res.* (2018) 1–21, <https://doi.org/10.1080/00207543.2018.1510558>.
- [3] I. Avazpour, J. Grundy, L. Zhu, Engineering complex data integration, harmonization and visualization systems, *J. Ind. Inf. Integr.* 16 (2019) 100103, <https://doi.org/10.1016/j.jii.2019.08.001>.
- [4] B. Vogel-Heuser, D. Hess, Guest editorial industry 4.0—prerequisites and visions, *IEEE Trans. Autom. Sci. Eng.* 13 (2) (2016) 411–413, <https://doi.org/10.1109/TASE.2016.2523639>.
- [5] E. Trunzer, I. Kirchen, J. Folmer, G. Koltun, B. Vogel-Heuser, A flexible architecture for data mining from heterogeneous data sources in automated production systems, 2017 IEEE International Conference on Industrial Technology, (2017), pp. 1106–1111, <https://doi.org/10.1109/ICIT.2017.7915517>.
- [6] Reference architecture model industrie 4.0 (RAMI4.0), Deutsches Institut für

- Normung e.V. Std. 91,3452016.
- [7] D. Gürdür, F. Asplund, A systematic review to merge discourses: interoperability, integration and cyber-physical systems, *J. Ind. Inf. Integr.* 9 (2018) 14–23, <https://doi.org/10.1016/j.jii.2017.12.001>.
- [8] A. Fay, B. Vogel-Heuser, T. Frank, K. Eckert, T. Hadlich, C. Diedrich, Enhancing a model-based engineering approach for distributed manufacturing automation systems with characteristics and design patterns, *J. Syst. Softw.* 101 (2015) 221–235, <https://doi.org/10.1016/j.jss.2014.12.028>.
- [9] D. Moody, The “physics” of notations: toward a scientific basis for constructing visual notations in software engineering, *IEEE Trans. Softw. Eng.* 35 (6) (2009) 756–779, <https://doi.org/10.1109/TSE.2009.67>.
- [10] E. Trunzer, A. Calà, P. Leitão, M. Gepp, J. Kinghorst, A. Lüder, H. Schauerte, M. Reifferscheid, B. Vogel-Heuser, System architectures for industrie 4.0 applications, *Prod. Eng.* 13 (3) (2019) 247–257.
- [11] P.A. Laplante, S.J. Ovaska, *Real-Time Systems Design and Analysis*, John Wiley & Sons, Inc, Hoboken, NJ, USA, 2011, <https://doi.org/10.1002/9781118136607>.
- [12] M. Wollschlaeger, T. Sauter, J. Jasperneite, The future of industrial communication: automation networks in the era of the internet of things and industry 4.0, *IEEE Ind. Electron. Mag.* 11 (1) (2017) 17–27, <https://doi.org/10.1109/MIE.2017.2649104>.
- [13] Systems and software engineering – architecture description, 2011, International Organization for Standardization Std. 42,010.
- [14] N. Medvidovic, R.N. Taylor, A classification and comparison framework for software architecture description languages, *IEEE Trans. Softw. Eng.* 26 (1) (2000) 70–93, <https://doi.org/10.1109/32.825767>.
- [15] I. Malavolta, P. Lago, H. Muccini, P. Pelliccione, A. Tang, What industry needs from architectural languages: asurvey, *IEEE Trans. Softw. Eng.* 39 (6) (2013) 869–891, <https://doi.org/10.1109/TSE.2012.74>.
- [16] The Open Group, ArchiMate 3.0.1 specification, 2017, [Online]. Available: <http://pubs.opengroup.org/architecture/archimate3-doc/>.
- [17] Architecture Analysis & Design Language (AADL), 2017, SAE International Std. AS5506C.
- [18] Object Management Group, UML profile for MARTE: Modeling and analysis of real-time embedded systems, [Online]. Available: <https://www.omg.org/spec/MARTE/1.1/PDF>.
- [19] B. Selic, Using UML for modeling complex real-time systems, in: G. Goos, J. Hartmanis, J. van Leeuwen, F. Mueller, A. Bestavros (Eds.), *Languages, Compilers, and Tools for Embedded Systems, Lecture Notes in Computer Science*, 1474 Springer Berlin Heidelberg, Berlin, Heidelberg, 1998, pp. 250–260, <https://doi.org/10.1007/BFb0057795>.
- [20] R. Grosu, M. Broy, B. Selic, G. Stefanescu, What is behind UML-RT? in: H. Kilov, B. Rumpe, I. Simmonds (Eds.), *Behavioral Specifications of Businesses and Systems*, Springer US, Boston, MA, 1999, pp. 75–90, https://doi.org/10.1007/978-1-4615-5229-1_6.
- [21] K. Thramboulidis, F. Christoulakis, UML4IoT—a UML-based approach to exploit IoT in cyber-physical manufacturing systems, *Comput. Ind.* 82 (2016) 259–272, <https://doi.org/10.1016/j.compind.2016.05.010>.
- [22] B. Tekinerdogan, T. Çelik, Ö. Köksal, Generation of feasible deployment configuration alternatives for data distribution service based systems, *Comput. Stand. Interfaces* 58 (2018) 126–145, <https://doi.org/10.1016/j.csi.2018.01.002>.
- [23] B. Terzić, V. Dimitrieski, S. Kordić, G. Milosavljević, I. Luković, Development and evaluation of microbuilder: a model-driven tool for the specification of rest micro-service software architectures, *Enterp. Inf. Syst.* 3 (5) (2018) 1–24, <https://doi.org/10.1080/17517575.2018.1460766>.
- [24] J. Greifeneder, G. Frey, DesLaNAS - a language for describing networked automation systems, 2007 IEEE Conference on Emerging Technologies & Factory Automation (EFTA 2007), 25.09.2007–28.09.2007, IEEE, 2007, pp. 1053–1060, <https://doi.org/10.1109/EFTA.2007.4416899>.
- [25] B. Vogel-Heuser, S. Feldmann, T. Werner, C. Diedrich, Modeling network architecture and time behavior of distributed control systems in industrial plant automation, 37th Annual Conference of the IEEE Industrial Electronics Society, (2011), pp. 2232–2237, <https://doi.org/10.1109/IECON.2011.6119656>.
- [26] M. Sollfrank, B. Vogel-Heuser, M. Fahimipirehgalin, Integration of safety aspects in modeling of networked control systems, Proc. IEEE International Conference on Industrial Informatics, IEEE Press, Emden, 2017, pp. 405–412.
- [27] T. DeMarco, *Structured Analysis and System Specification*, 21st ed., Yourdon Computing Series, Yourdon Press, Englewood Cliffs, 1979.
- [28] D.J. Hatley, I.A. Pirbhai, *Strategies for Real-Time System Specification*, Dorset House Publ, New York, NY, 1988.
- [29] M. Hashemi Farzaneh, S. Feldmann, C. Legat, J. Folmer, B. Vogel-Heuser, Modeling multicore programmable logic controllers in networked automation systems, IECON 2013 – 39th Annual Conference of the IEEE Industrial Electronics Society, IEEE, 2013, pp. 4398–4403, <https://doi.org/10.1109/IECON.2013.6699843>.
- [30] W. Kang, K. Kapitanova, S.H. Son, RDDS: a real-time data distribution service for cyber-physical systems, *IEEE Trans. Ind. Inf.* 8 (2) (2012) 393–405, <https://doi.org/10.1109/TII.2012.2183878>.