IET Collaborative Intelligent Manufacturing

*Research Article*

IET Journals
The Institution of Engineering and Technology

# Analysis of metamodels for model-based production automation system engineering

eISSN 2516-8398
Received on 14th February 2020
Revised 30th April 2020
Accepted on 12th May 2020
E-First on 17th June 2020
doi: 10.1049/iet-cim.2020.0013
www.ietdl.org

*Suhyun Cha[1] ✉, Birgit Vogel-Heuser[1], Juliane Fischer[1]*

[1]*Technical University of Munich, Boltzmannstr. 15, 85748 Garching, Germany*

✉ *E-mail: suhyun.cha@tum.de*

**Abstract:** In the current Industry 4.0 era, automated production systems (aPS) comprising of multi-disciplinary artefacts all closely interwoven are required to adapt to various and varying requirements from customers and environment which introduce additional complexity. Model-based engineering on the premise of metamodelling is regarded as a promising paradigm to handle this complexity to engineer aPS. Although various metamodels appear to solve problems in different viewpoints on systems, the absence of a core metamodel causes inconsistencies between the metamodels and hinders common understanding of the system and model reuse. In this study, the authors analyse existing metamodels from different research groups and present inconsistencies among them explicitly which support the necessity of the core metamodel. Considering properties of aPS together with relevant standards, the authors present a demonstration of analyses on exemplary metamodels and a set of criteria to understand the various aspects of the aPS metamodels as the first step towards the core metamodel. Feasibility of creating a universal metamodel of aPS domain is discussed, and the authors claim the necessity of having a common understanding of core concepts of aPS to support the cross-disciplinary reuse of existing metamodels and, accordingly, the compatibility of the instalment and operation of components.

## 1 Introduction

In the current Industry 4.0 (I4.0, which is derived from the German term Industrie 4.0 [1]) era, automated production systems (aPS) including manufacturing machines and logistics are required to adapt to various and varying production requirements (functionally) and operate in a smarter way (non-functionally) with the longer lifecycles [2]. Although the requirements could be described in short sentences, these imply usually complicated functional requirement, e.g. flexible and real-time production facility configuration, including non-functional requirement, e.g. increased production efficiency or human-friendly interactions. These various requirements lead to the increased complexity of aPS [3–5].

To handle this increased complexity efficiently, engineering using models (model-based or model-driven) is regarded as a promising paradigm for aPS engineering [2, 6]. Under the model-based paradigm, the requirements can be considered already in the design phase, and possible danger or risk in the real implementation can be avoided in advance by validating the models. Modelling has always been an engineering method in the field of production automation [7] appearing in different forms such as topology drawings, mathematical equations or graphics of processes. Regarded as a promising way to handle the problems efficiently during engineering processes and supported by the software tools [7], lots of model-based approaches have been introduced.

The International Standard Organization (ISO) defines the term model as ‘M is a model of S if M can be used to answer questions about S’ [8]. The Object Management Group (OMG) defines it as ‘A model represents some concrete or abstract thing of interest, with a specific purpose in mind’ [9]. Stachowiak [10] has derived three fundamental properties of a model: mapping (as a representation of something), reduction (by highlighting the relevant attributes and eliminating irrelevant ones), and pragmatism (usable for a specific purpose and timing). Although many define the term in different ways, a model could be conclusively defined as a representation of an original entity for a specific user concerning a specific intention at a specific point of time.

As the prefix meta means ‘beyond’ etymologically [11], metamodel means a model in the higher and more abstract degree beyond models. With this base, like model, there are different definitions of term such as by ISO [8] as ‘a metamodel presents the architecture description elements that comprise the vocabulary of a model kind’, by OMG specifications (e.g. [12]) as ‘a metamodel is a model that defines a modelling language and is also expressed using a modelling language’, by Mellor *et al.* [13] as ‘it defines the structure, semantics and constraints for a family of models’, or by Clark *et al.* [14] as ‘a metamodel should be capable of describing a language's concrete syntax, abstract syntax and semantics’. Summarising these different definitions, a metamodel could be defined as a representation of an abstract syntax used to represent semantic contents of the model kind.

Different viewpoints are decided by stakeholders and concerns [8], and lead to different metamodels, meaning different types of representation of the system-of-interest. However, metamodels vary, same object-of-interest, or more specifically its aspect-of-interest, should be described consistently to achieve efficiency over approaches. Especially, this applies to the essential parts, which all the same types of systems include commonly and appear more often in the metamodels correspondingly. We give a small exemplary case of unified modelling language (UML) specification [15], in which various metamodels are defined for different aspects of the language. As one type of behaviours, the StateMachine package defines concepts of event-driven behaviours. Within a state machine, a Trigger enables a path traversal by existing between a state and a transition (Fig. 1*a*). These elements are all explicitly visible within StateMachine metamodel. The mechanism of how this Trigger gets in effect is defined in the Event metamodel, which is the actual occurrence and defined as a core concept underlying all behavioural models in UML (Fig. 1*b*). Although the Event does not appear explicitly within the StateMachine metamodel, the separately defined elements can be associated with other core concepts. If they all were separately developed without any consolidating organisation like OMG, they might have had unassociated names for each and some rules should have been defined to describe the relevant associations. As they are developed on common ground, this huge metamodel could be consistent as a whole. Over model-based aPS engineering

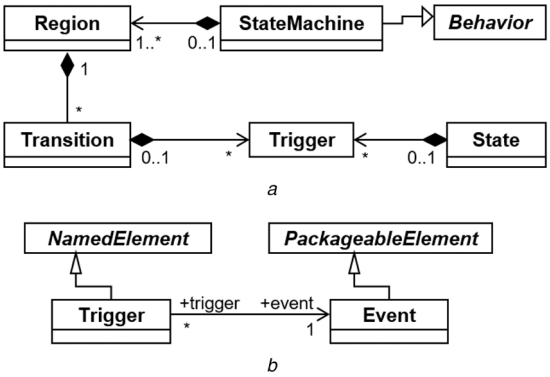


**Fig. 1** *Exemplary UML metamodels [15]: separate but consistent*
***(a)*** Partially excepted StateMachine metamodel, ***(b)*** Partially excepted Event metamodel

methodologies, it is observed that a huge variety of metamodels appears to solve different problems, but not with a commonly accepted abstraction of these all, which is namely a core metamodel. Various metamodels with the absence of the core metamodel may hinder the common understanding of stakeholders, e.g. different vendors of components who enable connections of the concepts from different sides as a very intuitive example.

The main contribution of this study is to indicate existing metalevel inconsistencies over aPS metamodels explicitly and to reason the factors of different viewpoints on a system which have been somehow assumed but not clearly visible so far. To coordinate the metamodels in this domain, the first step towards the core metamodel would be to abstract common properties of our target system. Therefore, we visit the characteristics of aPS and their engineering processes and present classification criteria for aPS metamodels based on them. These are demonstrated with different research groups' metamodels. These criteria will enable engineers from aPS domain to judge or analyse whether an existing metamodel is suitable to fulfil or extend for their needs. At the same time, there will be a starting point to organise metamodel landscape in aPS engineering field and to have the foundation of the core metamodel underlying all aPS metamodels.

The outline of this study is as follows: first, model-based approaches for aPS are shortly reviewed together with the inconsistency management approaches and relevant standards in Section 2. In Section 3, characteristics of aPS and their engineering are introduced to enhance the insight on aPS metamodels. After addressing research questions and research methods including introducing the target metamodels in Section 4, Section 5 follows it by analyses of aPS metamodels to present and reason metamodel inconsistency cases by proving hypothesised causes. In Section 6, the summarised findings are discussed together with the threats to validity and metamodel classification criteria derived from aPS properties are presented followed by the conclusion and outlook in Section 7.

## 2 State of the art

This section addresses states of related researches about model-based approaches. The focus is put first on model-based aPS engineering approaches (Section 2.1) and narrowed down to metamodel inconsistencies (Section 2.2). Relevant standards and guidelines endeavouring to align individual approaches are introduced (Section 2.3) followed by a summary in Section 2.4.

### *2.1 Model-based engineering for aPS*

As model-based engineering is regarded as a promising paradigm for the development of aPS to handle the complexity [6], various model-based approaches appear along with the aPS engineering flow from requirement analysis to system maintenance and evolution. Introduction to these approaches detail is omitted here. Instead, reviews on model-based aPS/cyber physical system (CPS) engineering approaches are found in various literature, e.g. Vyatkin [16], Fay *et al.* [17], Vogel-Heuser *et al.* [2] or Shi *et al.* [18]. Considering the variety of the metamodels of the approaches for aPS engineering, some researchers have already started to look at the bigger picture trying to achieve insights into common understandings among the various approaches. An overview of different metamodels for system engineering and the modelling languages is introduced by Reichwein and Paredis [19]. Interestingly, in their outlook, they have considered handling inconsistencies of heterogeneous metamodels as one of the future challenges in model-based system engineering (MBSE). While it becomes more serious in aPS domain as model-based engineering gets more common methodology than before, the envisioned directions are in the wider scope of system engineering which requires precise elaboration to be applied in aPS domain. Vogel-Heuser *et al.* [2] present a view on aPS development along with the engineering cycles with MBSE though the inconsistencies commented in their work are laid mainly in model level. Witte *et al.* [7] present a more philosophical view with relating models, concepts and modelling languages mainly discuss interchange format among the different models and modelling software. Recently, Pietrusewicz [20] classifies various metamodels in the scope of CPSs. In his work, comparisons and classifications of metamodels depending on engineering activities and modelling workflows are presented but characteristics aPS are not considered.

### *2.2 Inconsistency management over metamodels and models*

Different metamodels are developed from different viewpoints depending on the concerns causing issues of model inconsistencies. Various stakeholders in an aPS building project understand and implement their own parts based on relevant metamodels simultaneously in most cases due to the multi-disciplinary aspect. Actually, inconsistencies over models are likely to occur and have to be carefully considered to guarantee a high quality of the final system [5] as well as consolidation of collaborative data for smart system operation and management as presented, e.g. in [21]. On the one hand, maintaining consistency or traceability is obtained by the model transformation. For a continuous engineering process, model transformation has been often applied in the automation software development process (e.g. from observed/classified pattern [22], component-based model [23], or SimulinkTM model [24]) or test cases (e.g. from UML State Charts [25] or sequence diagram [26]) to save development time and cost with fewer faults than manual transformation. As a syntactic standardised data exchange among the different presentations, AutomationML [27] is suggested, and accordingly, the approaches weaving different models and languages through this have appeared such as [28]. One remark is that model transformation is defined on the premise of the inter-metamodel consistencies which have been either assumed to be consistent or handled by full knowledge regarding the target metamodels (usually well-known) to build transformation rules.

On the other hand, approaches to manage inter-model inconsistencies are suggested. Derler *et al.* [29] have counted ‘component consistency’ and ‘model divergence’ as the challenges of modelling CPS highlighting the evolving models in parallel. So the assumption is that model semantics are already consistent but the models get inconsistent during the engineering. Similarly, the sequential model-based design steps for CPS suggested by Jensen *et al.* [30] is emphasising keeping consistencies among models by transforming a model into another one. Feldmann *et al.* [5] have introduced a semi-automatic supporting method of inter-model inconsistencies of aPS by bridging target models with a specific model called ‘Link’. In their work, they also classify inconsistency management approaches into logical reasoning (i.e. defining the relationships and constraints in logics), rule-based reasoning (i.e. acquired rules and patterns), and synchronisation approaches (i.e. capturing conflicts by transformation). Vogel-Heuser *et al.* [31] have found that inconsistencies can be detected easier with coupled technical models by a priori connections or post hoc tracing considering the perspectives of the collaborative human behaviours. They [32] have also presented the multi-view collaboration method using knowledge-base and shown a research

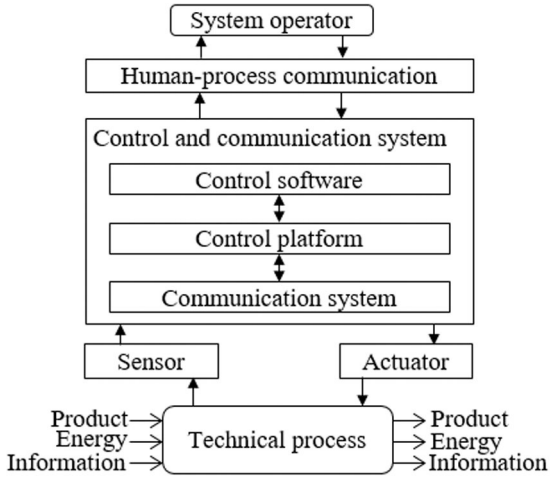


**Fig. 2** *Technical process and technical system [46]*

demonstrator use-case, still the method works in model level. As many approaches are developed for software engineering like [33], Wolfenstetter *et al.* [34] introduce a software tool called TRAILS (standing for traceability, integration, and life-cycle management support) which aims to convert existing information from different domain-specific model into a uniform format by defining the relationship between the metamodels like Link model. Similarly, correspondence and consistency rules can be defined for a set of metamodels of the targets for flexible views in virtual single underlying metamodel (VSUMM) suggested by Kramer *et al.* [35]. As seen in one of the UML model examples in [36], these approaches usually target resolving model-level inconsistencies of major languages like well-known UML profiles also assuming that metamodels are already consistent which is valid for UML profiles. However, as highlighted, metamodel consistency is not assured among independently developed approaches. Hardebolle and Boulanger [37] have summarised approaches about abstract syntax and semantics of modelling languages in the computer science field and concluded that capturing common semantics is a mandatory step to handle heterogeneous models as we see the problem in the same way.

### *2.3 Standards as common grounds*

Existing standards might take a role for being bases of consistency. ISO 42010 (originated from IEEE 1471) [8] defines systems and software engineering architecture description. There, the concept of architecture description and the correspondence of architecture framework entities. This can give a framework of aPS metamodelling as a conceptual base. More specific into the mechatronics system, IEC 61512 [38] (also known as ISA 88) defines reference models for batch control in the process industry. For example, a 'unit' is defined as a basic block in the sense of procedure and its technical operation. VDI 2193 [39], a communication language for I4.0, defines elements (I4.0 component) in the sense of network entities. For a seamless communication of separately delivery parts, standardised protocols like open platform communication unified architecture (OPC-UA), are developed and many relevant approaches to adapt them to the varying requirement of the production environment (e.g. [40] for flexible manufacturing). IEC 62264 [41] (also known as ISA 95), a standard for enterprise-control system integration, defines equipment very broadly in the viewpoint of system integration over functional hierarchies. The closest standard to our aim would be IEC 62390 [42] which defines automation devices as a component consisting of hardware, software, function and interface. Standards help for common understanding, but they mostly define entities in too abstract level by their nature and are not easy to interconnect each other, e.g. automation devices in IEC 62390 can be a little bit more specified regarding the basic structure and function of aPS. VDI 5100 [43] is a German software architecture guideline for intralogistics systems (also known as SAIL). These standards could guide regarding the metamodel consistencies to some extent; however, still they are quite higher abstract level to be considered in the engineering level as a core metamodel. Examples conforming this guideline are presented later to show metamodel level inconsistencies.

### *2.4 Discussion*

The relationship between metamodels can be defined in an *ad hoc* way by having a specific mapping between selected models as in some approaches introduced or many model transformation approaches. However, standardisation of the system models, not for all levels, but for core concepts, would be necessary to ensure semantical homogeneity over a wider range of metamodel semantics to achieve efficiency during aPS engineering. The International Council on Systems Engineering (INCOSE) also highlights and envisions the necessity of encompassing sophisticated model-based methods in the two recent trend reports [44, 45].

Although there are varieties and complexities of aPS, aPS have common characteristics in technical and engineering processes. Despite a number of standards, semantically inconsistent metamodels to handle different concerns are being developed missing a commonly accepted architecture as a set of terminologies or rules for coupling metamodels for aPS engineering. In this I4.0, the importance of compatibility of the components gets more highlighted to realise self-x systems since automatic integration or reconfiguration can be implemented based on the compatibility which requires information matches or well-defined transformation rules. Thus, this paper poses the following research questions: RQ1 – do modelled objects in the metamodels describe the same entity in the same way?; RQ2 – what makes the difference in the appearance of the metamodel on the same entity? and RQ3 – how do domain-specific metamodels focusing on the same concern differ?

## 3 Essential characteristics of aPS and engineering them

The aPS are defined in [2] as '…comprised of mechanical parts, electrical and electronic parts (automation hardware) and software, all closely interwoven, and thus represent a special class of mechatronic systems …' and 'aPS, a particular type of mechatronic system on which this paper focuses, is designed-to-order systems. These are complex manufacturing systems, and they have a typical lifetime in operation of several decades' in the technology point of view. In the functionality point of view, aPS refer to the manufacturing and logistics plants like production lines for automobiles or bottling beverages focusing on the production automation. Since a metamodel reflects a specific part of a system concerning the specific intention of the stakeholder, the concerns are related to the objectives of corresponding engineering steps. Therefore, understanding the characteristics of systems, i.e. aPS, and their engineering is necessary to understand the properties of metamodels. In this section, these characteristics are introduced as concerns and contexts of the metamodels in MBSE scheme.

### *3.1 Technical process of aPS*

A technical system implements its corresponding technical process by relevant components (Fig. 2) and this is the realm of our interest in aPS engineering. The system status is changed by the technical process with relevant parts (de)activated. Thus, the system structure, as well as, different states make the system to be seen differently in each engineering process. The two axes of the technical process and engineering process could categorise metamodels since system(part)-under-interest and the concerns of the stakeholder can be defined with these two axes.

The processes proceeded within aPS are technical processes based on the definition in [47] – a process by which involved object, i.e. matter, energy or information, is altered in its state. Thereby, a technical process is the totality of all operations in which material, energy or information is converted, transported or stored [46]. Automation [47] in general requires access to information from this technical process (via sensors) to influence

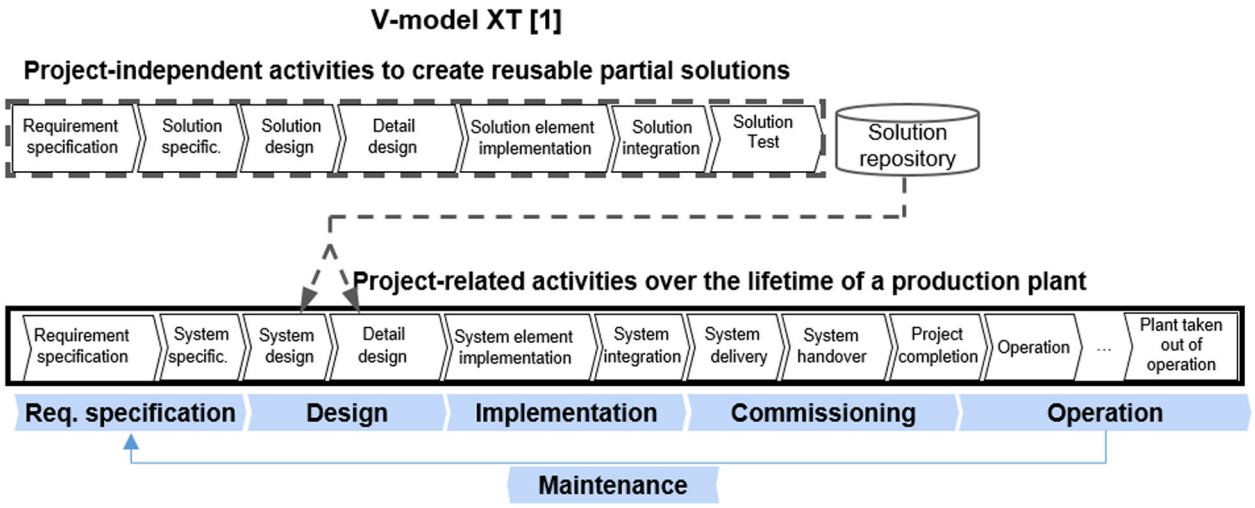


**Fig. 3** *V-Model XP with separation of the project-independent/project-related activities (elaborated from [2])*

the technical process (via actuators) based on its behaviour. Here, the material, energy and information are the objects of the technical processes. Also, human interaction is regarded as an important element in aPS since a human is one of the main elements contained in the aPS to be specialised out of overall scope of automated systems as an entity who develops, follows the process event, control and influences process, and handle the faults [46]. This classical concept of process has been a base for the automation pyramid as known (defined in ISA 95), which extends the basic technical process by including supervisory control level (supervisory control and data acquisition), overall manufacturing execution management level (manufacturing execution system), and business management level (enterprise resource planning, ERP).

### *3.2 Engineering process of aPS*

Since aPS, a particular type of mechatronic system, is designed-to-order system, each aPS engineering process goes through specific engineering methods depending on the own requirements and objectives of the system. The engineering life cycle can be presented as Fig. 3. Based on the project-independent activities, customer-specific projects are designed and developed in particular (lower part in Fig. 3).

Although the overall process of engineering from requirement specification to the maintenance in the coarse level is similar to the other systems, like pure software system, one of the big difference is that aPS have physical substances with masses. To be integrated as one aPS, most pieces machinery are shipped in part, assembled and commissioned on-site [48]. This specific engineering process leads to the necessity of adaptation during commissioning as well as start-up [2]. Therefore, this step cannot be overlooked in aPS engineering process. Various components are gathering or generated from various disciplines (or their combination). This leads to maintenance activities in various forms. Also, longer life cycle (usually decades) and corresponding ageing cause components from different disciplines to go through re-engineering and modernisation (mechanics: 5–40 years, electrics: 10–15 years, software: weeks–years). Some reasons could be, e.g. physical wear and tear, changing requirements, or technology trends and these cause changes in different disciplines with different cycles [2]. This is also a distinctive characteristic of aPS engineering.

### *3.3 Characteristics of aPS and the properties of models and metamodels: what, when and why*

A model represents a technical system in a technical process statically or dynamically. Therefore, the technical system, including interacting entities (e.g. external systems or engineers) if necessary, is what a model describes. A metamodel is built to capture the same sort of description. Then, when and why is a model generated? In the MBSE, the engineering processes can be separated into the project-independent activities and the project-related activities (Fig. 3). Within the project-independent activities, reusable solutions are developed and these are taken to be used in the project-related development process. Along this engineering process, models are the artefacts of each engineering process. Therefore, when and why a model is generated could be answered by the juncture and the objective of the project-related engineering activities, respectively. In the case of metamodels, the objective of the metamodel would be same as the model while the timing of the metamodel generation and use would be different. Most probably the generation of a metamodel would be during the project-independent activities to develop an approach for the same sort of the problems and its use (i.e. modelling) would happen during the project-related activities.

## 4 Research questions and study design

Metamodels appear in different forms depending on stakeholders and concerns. Inconsistencies in not only values (model level) but also semantics (metamodel level) hinder the efficiency throughout the aPS engineering lifecycles. Getting one step deeper into the metamodel levels, we discover explicit metamodel level inconsistencies and investigates possible reasons in this paper.

### *4.1 Research questions and hypothesis*

Metamodels reflect the design concept, and thus modelling mistakes can seldom be a reason for this fundamental level inconsistency. Rather, design rationale and individual background could be. To clarify the direction of the investigation and the goals, the research questions (RQs) to be addressed connected to the hypotheses as follows:

- RQ1: Do modelled objects in the metamodels describe the same entity in the same way?

  ○ (H1) There are inconsistencies across metamodels regarding the objects' names and their semantics.

  H1.1) There are objects in the metamodels named differently though semantically they mean the same.
  H1.2) There are objects in the metamodels, which are named the same but mean different semantics.
- RQ2: What makes the difference in the appearance of the metamodel on the same entity?

  ○ (H2) The different objectives and backgrounds of the architect decide the different use of the elements across the metamodels.

  H2.1) Metamodels for analysis of the already existing artefacts and generation of the new artefacts appear in different constitutions.

H2.2) Metamodels designed by different discipline-based architect appear in different constitutions.

- RQ3: How do domain-specific metamodels focusing on the same modelling object differ?

  ○ (H3) Various abstraction levels appear across the metamodels, which is related to the underlying design rationale of the corresponding software architecture.

  H3.1) Metamodels with the same objective appear in different forms depending on the design rationale of the overall control software in the same application domain, like intralogistics.
  H3.2) If the objective of the model even in one application domain like intralogistics is different, the resulting metamodel is different in regard to the level of abstraction of the modelled concepts

### *4.2 Introduction of the considered metamodels*

Some published metamodels from aPS engineering methodologies are selected from the automation community to be analysed (Table 1). To have a wide range of application cases, metamodels are selected based on several aspects of the aPS;

- *Control software*: IEC 61131-3 [49] is the standard for control software on PLC. As the proportion of the aPS functionalities realised in software is increasing [50], efficient and effective handling of automation software via model-based approaches regarding, e.g. automated generation or change management, is highlighted [2]. Two metamodels related to this language are selected regarding automatic generation and analysis.
- *I4.0*: as the current trend in automation, most design principles and enabling technologies are actively researched for I4.0 during recent years [1]. Two metamodels are selected focusing on two main features of I4.0 which are flexible collaboration and reconfiguration
- *Enterprise management*: cost reduction is the main driver of aPS engineering techniques [16] during not only developing them but also maintaining for their long life cycle. One metamodel is selected to cover the engineering cost in the viewpoint of enterprise management.
- *Specific subdomain (intralogistics)*: intralogistics systems are considered as an especially mature subdomain of aPS since the modules are loosely coupled and formed by application of internet of things paradigm even a decade before I4.0 [51, 52]. Selected metamodels are based on the domain-specific software architecture (SAIL) and show inconsistencies even within a small domain.

To find the evidence of hypotheses and prove them, selected metamodels considering supplementary ones are investigated regarding their aims and design rationale in detail. Over the target metamodels, terminologies and their semantics are figured out and compared. Also, the counter parts of the comparison targets are projected to each other for the structural difference analyses.

## 5 Analyses on metamodel inconsistencies

For each hypothesis to explore the research questions, considered exemplary metamodels are analysed in the following (Table 1). They are referred with acronyms, and some additional metamodels are introduced with brief explanations to support and strengthen the proofs.

### *5.1 Inconsistencies of the syntaxes and semantics (H1: H1.1. and H1.2)*

Some objects are named differently in different metamodels even though they are semantically the same (H1.1). That is, various synonyms appear to represent the same entities for the metamodels (or part of them) describing the same system. For example, in SemAnz40 [55], aiming at providing a semantic basis for a mechatronics system, a system-of-interest is divided into functional units, which are further decomposed into components. In this metamodel, a structural joint of components is named as interface for flows of material, energy and information as well as a geometric joint for assembly. On the other hand, there is another metamodel RC4PA [56] for plant automation focusing on the reconfiguration mechanism of the system. In this metamodel, the control system is divided into several mechatronic components. These components are connected (i.e. referring to or referred by) variables through the IO class which represents input/output of the signals. Therefore, IO means the same sense with the information flow of interface in SemAnz40. Additionally, we can find another synonym in other metamodels like interdisciplinary behaviour model [60] which is designed for formal verification of the system behaviours by extending software behaviour models to the hardware levels. Similarly to SemAnz40, the system structure is decomposed further into components as essential structural elements. In this metamodel, components are connected to the others through port instead of interface.

In reverse, different objects are also observed to be called with the same names (H1.2). In IntraMAS [58], where interface class is found in resource part representing an interfacing point to hand over the materials as downed by modules. In SemAnz40, the comparable parts of these interface and module in IntraMAS would be interfaces and components with a broader meaning. If an engineer uses these metamodels, he/she should be careful not to be confused with the element names. In both cases, which are naming and semantic mismatch, the inconsistency introduces inefficiency. Also, H1.1 and H1.2 are true and, consequently, H1 is true.

One possible reason for these inconsistencies could be that terminologies and definitions, which are firmly established as

**Table 1** Description about exemplary metamodels

| Aspects | Description |
|---|---|
| control software | IEC 61131-3 metamodel for code generation (CG4IEC) [53]: defines the information in the IEC61131-3 program to generate the code automatically from the model |
| | IEC 61131-3 metamodel for family mining (FM4IEC) [54]: defines the information in the IEC61131-3 program to compare and analyse variants and versions of the PLC software |
| I4.0 | metamodel for semantic modelling (SemAnz40) [55]: defines semantical bases to support semantic modelling of aPS in the various viewpoint |
| | plant automation system metamodel for defining control system reconfiguration (RC4PA) [56]: defines the reconfiguration of IEC 61131-3 software programs to ensure availability at runtime in case of controller failures |
| enterprise management | change effort estimation metamodel using Karlsruhe Architectural Maintainability Prediction (KAMP4aPS) [57]: defines the structural description of the aPS to apply KAMP method to derive a list of tasks for a change considering functional hierarchy diversity on aPS domain derived by pure software engineering viewpoint |
| specific domain (intralogistics) | intralogistics multi-agent system metamodel (IntraMAS) [58]: defines the structural composition of material flow systems including routes and transportation allowing the knowledge base (system layout, component and task) required to control flexible material flow systems |
| | metamodel for automated material flow modules (AutoMFM) [59]: defines material flow modules to generate the control code of intralogistics systems |

unified concepts in aPS domain, are hardly found. Actually some standards such as the device reference model [61] (which has its own base on IEC 62390) define core concepts, e.g. Device – as a module which gives specific functions like measuring or controlling – containing interfaces to be able to exchange broad-ranged information like field control or diagnosis. However, the naming of each element from these all metamodels is not exactly and firmly tied into each other. In other words, an engineer knowing one of them should spend some time to interpret the meaning of the other metamodel elements so that he/she could find the corresponding element.

When there are inconsistencies between metamodels regarding its syntaxes and semantics, metamodel elements can be synchronised either in *ad hoc* ways by having a specific mapping between selected ones only (like transformation rules between metamodels) or in consolidated ways by compiling a unified terminology dictionary. Although the former might require small work comparable to the latter only for mapping between target models, this one-to-one mapping causes redundant mapping over the pairs of the metamodels. Therefore, a common understanding of the considered aPS domain would enhance the understandability and reuse of the existing metamodels as well as the detection and resolution of inconsistencies over metamodels used in different disciplines or different phases of engineering.

### *5.2 Concern types of metamodel: analysis of the existing artefacts versus generation of the new artefacts (H2.1)*

In metamodels, elements of the system-of-interest appear with different features in the metamodels depending on the concern types (H2.1). We introduce two examples with a narrow focus on the IEC 61131-3 language regarding: code generation transformed from UML model (CG4IEC), and code comparison as a prepossessing step to family mining (FM4IEC).

CG4IEC [53] aiming to generate a runnable object-oriented PLC project in sequential function chart (SFC) defines the PLC platform specifications with configuration and resources including variables and program organisation units (POUs). We focus on the way of variable declarations. In this metamodel, possible variable types are defined firmly as enumeration depending on the functional module types (e.g. FunctionBlock or Method). When we see another metamodel, FM4IEC [54] also depicting IEC 61131-3 control software with a top-level view on a PLC project, the metaclasses works as a container of information to be compared regarding similar variants and versions of PLC projects for its code comparison purpose. In this metamodel, variables and their declaration appear as ones of the main comparison elements; however, restrictions of the variable types depending on the POU types do not appear. This is because restrictions on allowed syntaxes of elements are necessary to assure the validity to generate runnable code whereas this is not an issue for code comparison since the code already fulfils these requirements of being executable. For example, variable types or statement orders should be valid to make the code executable while no explicit measures are taken to ensure any certain rules in FM4IEC for code comparison. Another worthy point to compare is comment which is observed in FM4IEC but not in CG4IEC. Some implementation history or other information regarding variants could be described in comments, such as a version number or the versioning date. This is important information source to analyse the variants of the code. In contrast, this is not regarded as necessary for code generation because the comment of the generated code would not affect the software execution in any way. Therefore, we observe that same elements-of-interest are described with different features depending on the concerns. Thus, hypothesis H2.1 holds.

Some information can already be obtained or at least prepared to be consistent during earlier engineering steps for the later steps if it is already known that relevant information will be required later. For example, within the engineering process of the aPS, code generation is done in implementation steps while code comparison is a reverse engineering activity during the maintenance steps. Variants information can be prepared during the code generation in some information holder like ‘comment’, if it is foreseen, so that this could be used during the analysis steps. Although not all the engineering processes can be strictly defined or must not, engineering processes and terminologies can be standardised to some course degree to relate approaches and corresponding metamodels, which will bring higher efficiencies.

### *5.3 Effect of the background (H2.2)*

A metamodel might appear in different constitutions depending on the discipline base of the metamodel designer (H2.2). We show two aPS metamodel examples: KAMP4aPS and FM4IEC.

The KAMP4aPS [57] aims to estimate the effort to implement a change by deriving a task list. To analyse the change propagation over the components, a structural description of the aPS is necessary. This constitution is defined in two levels: the coarse level structure (‘abstract’ metamodel) and the finer level structure (‘specific’ metamodel) with further subclasses of possible component types. Comparing this to the other metamodels like SemAnz40, SysML4Mechatronics [62], or RC4PA in the context of the structural description, this metamodel would seem flat to the aPS engineer's viewpoint. Several reasons can be addressed for this. For example, ‘abstract’ metamodel provides only the very coarse level of differentiation of the system parts without any further abstraction of the features. It is a contrast that the compared metamodels already capture features of system parts in the coarse level with the abstraction (e.g. separating software from hardware and connecting these over interface). In addition, the ‘specific’ metamodel in KAMP4aPS highly depends on the detailed change propagation rules which work only for a specific system (or the same sort) while the comparing metamodels in the finer level are still applicable to broader system types.

A similar distinction could be found in another pair of metamodels: FM4IEC and a control flow analysis metamodel of PLC programs [63]. Both metamodels depict IEC 61131-3 SFC in which the elements of steps and transitions compose the program. Since the steps and transitions appear alternatively, a program is established by indicating adjacent elements of different types (from step to transition or vice versa). At this point, in the control flow analysis metamodel, SFCstep class are defined as floating nodes and transitions (Edge class in the metamodel) connect the steps by indicating sources and targets (sourceObject and targetObject properties in the metamodel). Differently from this, in FM4IEC steps and transitions are indicated bidirectionally: a transition indicates source/destination steps and a step indicates incoming/outgoing transitions. Although direct adjacency indication allows quicker data resolving of the predecessor/successor of each step or transition, this is not the way that system engineers would take since this might cause information inconsistency between the step's and the adjacent transition's indicator.

Mainly the KAMP4aPS and FM4IEC metamodels are developed by the designers from pure computer science field who might see every hardware component equally most probably. It might not be intuitive for the metamodel designers, who do not have enough experience of engineering this type of system, to abstract components differently regarding the hardware characteristics or to expect behavioural element inconsistencies. Thus, H2.2 is true.

Metamodels can be designed in different ways; however, these need to be acceptable to the field engineers who would utilise the metamodels during the engineering with the concrete values and data. Especially, as the broader disciplines are integrated during aPS engineering and the boundaries are getting blurred among the disciplines [1], a core metamodel of aPS would provide a blueprint of the system even to the engineers from another field with a consistent language.

### *5.4 Domain-specific metamodels with regards to levels of granularity (H3)*

Metamodels might appear differently, even in a sub-domain of aPS for the same modelling objective (H3.1). The AutoMFM metamodel [59] is designed to generate the control code of an intralogistics system modules dynamically adaptable to changes with conformance to a software architecture guideline called SAIL

**Table 2** Research questions and related hypotheses evaluation

| Criterion | Result and validity | Evidence |
|---|---|---|
| RQ1: Do modelled objects in the metamodels describe the same entity in the same way? | | |
| (H1) there are inconsistencies across metamodels regarding the objects' names and their semantics | TRUE (strongly valid) | H1.1, H1.2 |
| (H1.1) there are objects in the metamodels named differently even though semantically they mean the same | TRUE (strongly valid) | SemAnz40, RC4PA |
| (H1.2) there are objects in the metamodels, which are named the same but mean different semantics | TRUE (strongly valid) | SemAnz40, IntraMAS |
| RQ2: What makes the difference in the appearance of the metamodel on the same entity? | | |
| (H2) the different objectives and backgrounds of the modeler/architect decide the different use of the elements across the metamodels | TRUE (strongly valid) | H2.1, H2.2 |
| (H2.1) the objective of the metamodel can be separated into analysis of the already existing artefacts and generation of the new artefacts as one of the cause factors of the difference across the metamodel | TRUE (strongly valid) | CG4IEC, FM4IEC |
| (H2.2) metamodels designed by different discipline-based architect appear in different constitutions | TRUE (strongly valid) | KAMP4APS, FM4IEC |
| RQ3: How do domain-specific metamodels focusing on the same modelling objective differ? (narrowed down to intralogistics) | | |
| (H3) metamodel inconsistencies appear even within a smaller application domain | TRUE (strongly valid) | — |
| (H3.1) metamodels with the same objective appear in different forms depending on the design rationale of the overall control software in the same application domain, like intralogistics | TRUE (moderately valid) | AutoMFM, IntraMAS |
| (H3.2) different objectives affect on the level of abstraction of the modelled concepts even in one application domain of like intralogistics. | TRUE (strongly valid) | AutoMFM, IntraMAS |

to support modularisation of material flow modules. In the context of I4.0, alternative design rationales such as multi-agent systems are regarded as a suitable approach for a decentralised and flexible control architecture [64]. IntraMAS [58] applies this agent-based control architecture to support runtime change of logistics system's layout without error-prone re-engineering of the control software. IntraMAS describes the knowledge base required by a module agent to judge whether a requested job order can be fulfilled by the module. In contrast to AutoMFM which includes hardware device like sensors and actuators, IntraMAS metamodel omits this detail of hardware control but includes abstract information about the material flow modules instead, such as abilities or potential material interfaces. By this logical level description, the knowledge base can contain module-type independent information so that the agents of material flow modules can interact with each other. These two metamodels from the intralogistics domain demonstrate how metamodels could differ greatly even within a small sub-domain of aPS metamodels used with the same objective of supporting the PLC software development because the fundamental software control design rationale differs. Thus, H3.1 holds true.

Another aspect regarding RQ3 is that metamodels even in a small application domain with a different modelling objective handle the same core concepts in different abstraction levels (H3.2). One example of a core concept in the intralogistics domain is a module interface as a survey shows that the module interface development is one of the most critical issues for realisation [65]. We consider the two metamodels introduced above. In the AutoMFM metamodel, the module interfaces are described from a hardware point of view regarding the material flow interfaces (ModuleFlowInterface), similar to IntraMAS (Interface). Additionally in AutoMFM, a module sequence control viewpoint is considered which leads to control interfaces for details of the material transfer control such as specific event sequence (i.e. handshake) to build connections. In IntraMAS, it is crucial to detect neighbouring material flow modules to interact with while detailed information of the field device level is not of interest. In short, the metamodel represents the MFMs on a logical level. The Interface class is defined regarding its position and type of material transfer (i.e. either input, output or both) from a hardware point of view. This is sufficient for the agents to control the material flow consisting of a sequence of modules connected via their interfaces. Therefore, it holds true that depending on the objective of a metamodel, the same core concepts of a domain could be defined in different granularity levels even in a very narrow area (H3.2).

### *6.1 Summary of findings and validity analysis*

We have analysed various causes of metamodel differences starting with the research questions (summarised in Table 2). For RQ1, we have hypothesised that there are inconsistencies across metamodels regarding the objects names and semantics (H2.1, H2.2). We have verified this by showing the same elements constituting a system called differently, and disparate components called with the same name. This means a specific semantic is not always linked to a specific syntax. There could be observable syntactic or semantic conflicts between the metamodels as seen even in a small set of metamodels that we gathered as our demonstrators. Thus, the inconsistencies of semantics and syntaxes can be observed often which indicates the analysis result to be valid.

For RQ2, we have hypothesised that different objectives and backgrounds of the architect lead to different constitutions of metamodels. We verify this with two different objectives, which are analysing existing artefacts and generating new artefacts (H2.1). We have also shown that discipline-based views could be a reason for these differences on the metamodels (H2.2). We find that the two objectives could logically come to a hypothesised conclusion, and thus the H2 is conclusively valid. Still, H2 could be regarded as the answer to RQ2 partially since not all the factors leading to the different metamodels are exhaustively investigated. This hypothesis needs to be supplemented with more factors causing different metamodels. Also for H2.2, not all the backgrounds are counted as possible views on aPS though there are more disciplines involved in aPS engineering other than software and mechatronics views. Thus, the effect of objectives and background on the metamodel need to be observed in more aspects to answer the RQ2 soundly, which results in a moderate level of validity.

In RQ3, the observation scope has been narrowed down to a specific application domain of intralogistics. We have shown that metamodel appears in different forms depending on the design concept even in the narrowed sub-domain with the same objective of the metamodel (H3.1). Although the results are as hypothesised, more design rationales should be explored to validate this hypothesis more strongly. The effect of the different objectives on the level of abstraction in metamodels is observed (H3.2). It has been shown that constitutions appear affected by the design rationales and objectives. We have a limited pair of metamodels, but could easily observe such a different abstraction level. Therefore, the H3 could be said valid.

## 6 Discussion

### *6.2 Feasibility and effectiveness of creating a metamodel covering the aPS domain*

The analysis and the examples illustrate the variety of viewpoints about aPS. Syntactic and semantic inconsistencies between the metamodels are often observed and are basically due to different viewpoints including concerns and backgrounds. The objectives and disciplines, which differ in most problem-solving activities, affect even on the representation of essential components of aPS in metamodels. Theoretically, it would be possible to include all different aspects in the various abstraction levels within one large metamodel. However, this holds the risk of becoming inconsistent during the maintenance of the metamodel and being hard to understand: engineers using the metamodel would choose the parts, which are relevant to fulfil their tasks. However, they are required to understand the entire metamodel more than actually needed to sort out necessary and unnecessary parts. Most of all, scopes of concepts and resulting metamodels in aPS domain are extensive as seen. Therefore, one large closed aPS domain metamodel is not feasible practically when it comes down to supporting a specific engineer's tasks. Nevertheless, a joint, commonly accepted core metamodel representing the significant concepts (e.g. structural and behavioural) of aPS components would be beneficial for efficient and effective MBSE for aPS. The core metamodel plays a role of a common understanding ground of important concepts and terminologies. Newly generated metamodels based on it would be able to be consistent which leads to engineering efficiency. Although there have been some attempts to standardise the essential elements and concepts, e.g. SAIL for intralogistics domain or the device reference model in IEC 62390, it does not perfectly fit as an aPS core metamodel. It is still required to analyse aPS regarding common structural and behavioural architecture and define corresponding concepts considerately. This will enhance a shared understanding and enable interconnecting and expanding of developed metamodels.

### *6.3 Classification criteria for metamodels*

Considering the characteristics of aPS in the technical and engineering point of view including representative standards to abstract aPS characteristics, finer-grained classification of aPS metamodels could be developed. The collected criteria based on widely accepted standards and definitions including explicit division of the technical and engineering processes proposed for classifying metamodels are introduced in Table 3. These criteria are collected in the perspective of the engineer who would utilise metamodels. Through reviewing the criteria of the metamodels, they could figure out the characteristics of the target metamodel, utilise the existing ones, or extend them. These criteria can contribute to extract common properties of aPS as the first step towards aPS core metamodel. Core components and their composition can be considered regarding the given aspects of types and points as classified by these criteria.

The considered metamodels in this study selected from the automation community are exemplarily analysed regarding the criteria in the same table. This demonstration envisions how the

**Table 3** Metamodel classification criteria and selected metamodels with their properties on the criteria

| Criteria and categories | | Metamodels | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | Control software | | Industry 4.0 | | Enterprise management | Intralogistics | |
| | | CG4IEC [53] | FM4IEC [54] | SemAnz40 [55] | RC4PA [56] | KAMP4aPS [57] | IntraMAS [58] | AutoMFM [59] |
| (a) functional hierarchy levels [41, 66]: a technical process includes technical system and human–machine interaction (HMI) as well as, manufacturing operation and businessplanning | field level (sensor/actuator) | | | X | | X | | X |
| | control and communication level | X | X | X | X | | X | X |
| | HMI for operation of the plant system | | | | | | | |
| | manufacturing operation and control (with including HMI for manufacturing management) | | | | | | | |
| | business planning (ERP) | | | | | X | | |
| (b) engineering process steps [67] part 2 – Typical aPS engineering process is separated in to six general categories: | requirement engineering | | | X | | | | |
| | design | | X | X | X | | | |
| | implementation | X | X | | X | | X | X |
| | commissioning | | | | | | | |
| | operation | | | | X | | | |
| | maintenance | | X | | | X | | |
| (c) technical flow sorts [46]: if a model describes a technical process, the relevant flow object appears in the model implicitly or explicitly | material flow | | | X | | | X | X |
| | energy flow | N/A | N/A | X | | N/A | | |
| | information flow | | | X | X | | X | X |
| (d) material (refines c1): the product itself might appear in the metamodel to indicate effect to or by product properties, e.g. adaptability of the system regarding the product weight change. Even if the material flow is not depicted in the model, material can appear statically | | | | X | | | X | |

| Criteria and categories | | Metamodels | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | Control software | | Industry 4.0 | | Enterprise management | Intralogistics | |
| | | CG4IEC [53] | FM4IEC [54] | SemAnz40 [55] | RC4PA [56] | KAMP4aPS [57] | IntraMAS [58] | AutoMFM [59] |
| (e) information classes (refines c3) [55], this criterion gives the type information of depicted system entities | function | | | X | | | X | |
| | structure | X | X | X | X | X | X | X |
| | behaviour | | | X | X | | | X |
| (f) discipline range [46]: in the technical system point of view, disciplines can be classified. In addition, technical process itself is another type of discipline to control the technical system over time | mechanical | | | X | X | X | | X |
| | electrical | | | X | | X | | |
| | software | X | X | X | X | | X | X |
| (g) level of detail: the detail level depends on the concerns and the model usage. Therefore, it is hard to define absolute classification of the detail level but the user of the model would recognise the proper abstraction degree | | more detail | more detail | abstract | detail | detail | detail | detail |
| (h) aPS type [41] types of aPS can be separated in to these two types based on | discrete | N/A | N/A | N/A | N/A | N/A | X | X |
| | continuous (Inc. Batch)[a] | | | | | | | |
| (i) specific application domain: the applicability of the metamodel is also an issue to the model user to decide if it is suitable for his/her concern | general aPS | X | X | X | X | Δ[b] | | |
| | specific type of aPS | | | | | | intralogistics | intralogistics |

[a]Basic control is principally the same for batch and continuous processes since batch is a continuous system with infinitely state duration [68].

[b]See Section 4.5.

criteria are intended to be applied on metamodels. This could be a starting point to arrange a landscape of metamodels for aPS domain. Also, one can define the engineering properties within the criteria and assess metamodels regarding these properties as well as analysing their needs and choosing a metamodel regarding its properties in the other way around.

## 7 Conclusion and further works

Regardless whether models are to be in graphical symbolic forms for engineers to understand and represent systems for developing systems or to be in mathematical representation for systematic and mathematical problem solving, metamodel inconsistencies are problematic in both cases. Metalevel inconsistencies are not easy to be captured with shallow knowledge about the relevant metamodels but easy to confuse engineers. To achieve higher quality systems in the end, these have to be handled effectively and efficiently.

Within this study, model-based aPS engineering is focused regarding the metamodels. Explicit metamodel inconsistencies are investigated together with their potential causes. Based on the motivation of the challenging and time-consuming task of metamodelling, the analysis result leads to the necessity of core metamodel to promote the effective and efficient model-based aPS engineering with higher system quality. As a first step towards a core metamodel, a set of classification criteria for aPS metamodels is presented, which can serve as a basis to enable engineers to assess existing metamodels for their engineering tasks.

Having various metamodels on aPS is inevitable due to its broad range of disciplines and engineering activities. Containing all information required for the different engineering tasks arising throughout the entire development and life cycle of an aPS across all involved disciplines would not be feasible. However, we envision the possibility and benefits of having a common and abstract metamodel defining the terminology of core concepts of the domain to gain a common understanding of important concepts and to avoid unnecessary inconsistency management cost. Preliminarily, detailed analyses about existing metamodels should be executed regarding various aspects, such as characteristics, commonalities, distinctions and inconsistencies. Basically, the core metamodel would enable generation of modelling language with consistent concepts and terminologies. Furthermore, it could be reformed in various forms depending on the objectives to optimally support inter-model or inter-language consistency problems which might happen during handling different models such as model transformation or language unification. The metamodels grounded by the core metamodel could be efficiently linked, mapped, or even used to create new views by combining elements of existing ones by means of current metamodel linking approaches such as the introduced ones, namely TRAILS, Link models or VSUMM.

Another issue to be discussed in the further work is the maintainability of the metamodel. Due to the longer lifecycle of the aPS, the system itself, its corresponding model, and metamodel face many chances to change under the MBSE scheme. It is obvious that the model should reflect the system snapshot whenever a change happens. As seen, the detailed metamodel supports the engineering task to be more focused on the target and the problem. In the view of maintenance, however, it is fragile to the changes done on the system, i.e. changes are required to be done on the metamodel when there is a change on the target system. As an abstraction of the models, it is not always desirable to change the metamodel (mostly not). Correlations between the different aspects of the metamodel such as abstraction level and its maintainability should be also researched. Furthermore, extending or including additional aspects of the system view with regards to the quality should be considered since system quality will be more

emphasised than ever before due to the increasing vulnerabilities in cyber-physical systems. A security issue, a cyber invasion as an example, might affect the product quality. Thus, our further research will also focus on whether a metamodel allows such an unobserved (yet) aspect to be integrated with which level of effort entailed.

## 8 Acknowledgment

This work was supported by the German Research Foundation (DFG) under the Priority Programme SPP 1593: Design For Future – Managed Software Evolution (grant number: VO937/29-1).

## 9 References

[1] Vogel-Heuser, B., Hess, D.: 'Guest editorial industry 4.0-prerequisites and visions', *IEEE Trans. Autom. Sci. Eng.*, 2016, **13**, (2), pp. 411–413
[2] Vogel-Heuser, B., Fay, A., Schaefer, I., *et al.*: 'Evolution of software in automated production systems: challenges and research directions', *J. Syst. Softw.*, 2015, **110**, pp. 54–84
[3] McFarlane, D.C., Bussmann, S.: 'Developments in holonic production planning and control', *Prod. Plan. Control*, 2000, **11**, (6), pp. 522–536
[4] Durdik, Z., Klatt, B., Koziolek, H., *et al.*: 'Sustainability guidelines for long-living software systems'. IEEE Int. Conf. Software Maintenance, ICSM, Trento, Italy, 2012, pp. 517–526
[5] Feldmann, S., Kernschmidt, K., Wimmer, M., *et al.*: 'Managing inter-model inconsistencies in model-based systems engineering: application in automated production systems engineering', *J. Syst. Softw.*, 2019, **153**, pp. 105–134
[6] Vogel-Heuser, B., Schuetz, D., Frank, T., *et al.*: 'Model-driven engineering of manufacturing automation software projects – A SysML-based approach', *Mechatronics. (Oxf)*, 2014, **24**, (7), pp. 883–897
[7] Witte, M.E., Diedrich, C., Figalist, H.: 'Model-based development in automation', *At-Automatisierungstechnik*, 2018, **66**, (5), pp. 360–371
[8] International Organization for Standardization (ISO): 'ISO/IEC 42010:2011 systems and software engineering – architecture description', 2011
[9] Object Management Group (OMG): 'Object management group terms and acronyms'. Available at https://www.omg.org/gettingstarted/terms_and_acronyms.htm, accessed January 2020
[10] Stachowiak, H.: '*Allgemeine modelltheorie*' (Springer-Verlag, Wien, 1973)
[11] Online Etymology Dictionary: 'Meta-origin and meaning of prefix meta- by online etymology dictionary'. Available at https://www.etymonline.com/word/meta, accessed January 2020
[12] Object Management Group (OMG): 'MDA guide rev.2.0', 2014, (June), pp. 1–15
[13] Mellor, S.J., Scott, K., Uhl, A., *et al.*: 'MDA distilled: principles of model-driven architecture', in '*MDA distilled: principles of model-driven architecture*' (Addison-Wesley Professional, Boston, USA., 2004), p. 14
[14] Clark, T., Sammut, P., Willans, J.: 'Applied metamodelling: a foundation for language driven development (3rd edn)'. ArXiv, 2015
[15] Object Management Group (OMG): 'OMG unified modeling language – version 2.5.1', 2017
[16] Vyatkin, V.: 'Software engineering in industrial automation: state-of-the-art review', *IEEE Trans. Ind. Informat.*, 2013, **9**, (3), pp. 1234–1249
[17] Fay, A., Vogel-Heuser, B., Frank, T., *et al.*: 'Enhancing a model-based engineering approach for distributed manufacturing automation systems with characteristics and design patterns', *J. Syst. Softw.*, 2015, **101**, pp. 221–235
[18] Shi, J., Wan, J., Yan, H., *et al.*: 'A survey of cyber-physical systems'. 2011 Int. Conf. on Wireless Communications and Signal Processing, WCSP 2011, Nanjing, China, 2011, pp. 1–6
[19] Reichwein, A., Paredis, C.: 'Overview of architecture frameworks and modeling languages for model-based systems engineering'. Proc. ASME Design Engineering Technical Conf., Washington, DC, USA, 2011
[20] Pietrusewicz, K.: 'Metamodelling for design of mechatronic and cyber-physical systems', *Appl. Sci.*, 2019, **9**, (3), p. 376
[21] Lazarova-Molnar, S., Mohamed, N., Al-Jaroodi, J.: 'Data analytics framework for industry 4.0: enabling collaboration for added benefits', *IET Collab. Intell. Manuf.*, 2019, **1**, (4), pp. 117–125
[22] Bonfè, M., Fantuzzi, C., Secchi, C.: 'Design patterns for model-based automation software design and implementation', *Control Eng. Pract.*, 2013, **21**, (11), pp. 1608–1619
[23] Estévez, E., Marcos, M., Orive, D.: 'Automatic generation of PLC automation projects from component-based models', *Int. J. Adv. Manuf. Technol.*, 2007, **35**, (5), pp. 527–540
[24] Yang, C., Vyatkin, V.: 'Transformation of Simulink models to IEC 61499 function blocks for verification of distributed control systems', *Control Eng. Pract.*, 2012, **20**, (12), pp. 1259–1269
[25] Krause, J., Herrmann, A., Diedrich, C.: 'Test case generation from formal system specifications based on UML state machine', *Atp-International*, 2008, **1**, pp. 47–54
[26] Kormann, B., Tikhonov, D., Vogel-Heuser, B.: 'Automated PLC software testing using adapted UML sequence diagrams'. 14th IFAC Symp. Information Control Problems in Manufacturing, Bucharest, Romania, 2012, vol. 45, no. 6, pp. 1615–1621
[27] Hundt, L., Drath, R., Lüder, A., *et al.*: 'Seamless automation engineering with AutomationML®'. 2008 IEEE Int. Technology Management Conf. (ICE), Lisbon, Portugal, 2008, pp. 1–8
[28] Berardinelli, L., Biffl, S., Lüder, A., *et al.*: 'Cross-disciplinary engineering with AutomationML and SysML', *At-Automatisierungstechnik*, 2016, **64**, (4), pp. 253–269
[29] Derler, P., Lee, E.A., Sangiovanni-vincentelli, A.L.: 'Addressing modeling challenges in cyber-physical systems', 2011
[30] Jensen, J.C., Chang, D.H., Lee, E.A.: 'A model-based design methodology for cyber-physical systems'. IWCMC 2011 – Seventh Int. Wireless Communications and Mobile Computing Conf., Istanbul, Turkey, 2011, pp. 1666–1671
[31] Vogel-Heuser, B., Böhm, M., Brodbeck, F., *et al.*: 'Interdisciplinary engineering of cyber physical production systems: highlighting the benefits of a combined interdisciplinary modelling approach on the basis of an industrial case', *Des. Sci.*, 2020, **6**, pp. 1–36
[32] Vogel-Heuser, B., Zou, M.: 'Leveraging inconsistency management in the multi-view collaborative modelling of cyber-physical production systems', *IET Collab. Intell. Manuf.*, 2019, **1**, (4), pp. 126–129
[33] Farias, K., De Oliveira, T.C., Gonçales, L.J., *et al.*: 'UML2Merge: a UML extension for model merging', *IET Softw.*., 2019, **13**, (6), pp. 575–586
[34] Wolfenstetter, T., Basirati, M.R., Böhm, M., *et al.*: 'Introducing TRAILS: A tool supporting traceability, integration and visualisation of engineering knowledge for product service systems development', *J. Syst. Softw.*, 2018, **144**, pp. 342–355
[35] Kramer, M.E., Burger, E., Langhammer, M.: 'View-centric engineering with synchronized heterogeneous models'. Proc. First Workshop on View-Based, Aspect-Oriented and Orthographic Software Modelling, Montpellier France, 2013, pp. 5:1–5:6
[36] Egyed, A.: 'Automatically detecting and tracking inconsistencies in software design models', *IEEE Trans. Softw. Eng.*, 2011, **37**, (2), pp. 188–203
[37] Hardebolle, C., Boulanger, F.: 'Exploring multi-paradigm modeling techniques', *Simulation*, 2009, **85**, (11–12), pp. 688–708
[38] International Electrotechnical Commission (IEC): 'IEC 61512-1:1997 – batch control – part 1: models and terminology', 1997
[39] Verein Deutscher Ingenieure (VDI) e.V.: 'VDI/VDE 2193 – sprache für I4.0-komponenten – struktur von nachrichten (en: language for I4.0 components – structure of messages)', 2019
[40] Gîrbea, A., Nechifor, S., Sisak, F., *et al.*: 'Design and implementation of an OLE for process control unified architecture aggregating server for a group of flexible manufacturing systems', *IET Softw.*., 2011, **5**, (4), pp. 406–414
[41] International Organization for Standardization (ISO): 'IEC 62264-1:2013 – enterprise-control system integration – part 1: models and terminology', 2013
[42] International Electrotechnical Commission (IEC): 'IEC TR 62390:2005 – common automation device – profile guideline', 2005
[43] Verein Deutscher Ingenieure (VDI) e.V.: 'VDI/VDMA 5100 Blatt 1 – Systemarchitektur für die intralogistik (SAIL) – grundlagen (en: system architecture for intralogistics (SAIL) – fundamentals)', 2016
[44] International Council on Systems Engineering (INCOSE): 'Systems engineering vision 2020', 2007
[45] International Council on Systems Engineering (INCOSE): 'System engineering vision 2025', 2014
[46] Vogel-Heuser, B., Diedrich, C., Fay, A., *et al.*: 'Challenges for software engineering in automation', *J. Softw. Eng. Appl.*, 2014, **07**, pp. 440–451
[47] International Electrotechnical Commission (IEC): 'IEC 60050-351:2013 – International electrotechnical vocabulary (IEV) – part 351: control technology', 2013
[48] Vogel-Heuser, B.: 'Automation in the wood and paper industry', in Nof, S.Y. (Ed.): '*Springer handbook of automation*' (Springer Berlin Heidelberg, Berlin, Germany, 2009), pp. 1015–1026
[49] International Electrotechnical Commission (IEC): 'IEC 61131-3 programmable logic controllers – part 3: programming languages', 2009
[50] Thramboulidis, K.: 'The 3 + 1 SysML view-model in model integrated mechatronics', *J. Softw. Eng. Appl.*, 2010, **03**, (2), pp. 109–118
[51] Regulin, D., Schuetz, D., Aicher, T., *et al.*: 'Model based design of knowledge bases in multi agent systems for enabling automatic reconfiguration capabilities of material flow modules'. 2016 IEEE Int. Conf. on Automation Science and Engineering (CASE), Fort Worth, TX, USA., 2016, pp. 133–140
[52] Mayer, S.H.: '*Development of a completely decentralized control system for modular continuous conveyors*' (Universitätsverlag Karlsruhe, Karlsruhe, 2009)
[53] Witsch, D.: '*Modellgetriebene entwicklung von steuerungssoftware auf basis der UML unter berücksichtigung der domänenspezifischen anforderungen des maschinen- und anlagenbaus (en: model-driven development of control software based on UML considering domain-specific requirements of machine and plant engineering)*' (Technische Universität München, Munich, Germany, 2013)
[54] Schlie, A., Rosiak, K., Urbaniak, O., *et al.*: 'Analyzing variability in automation software with the variability analysis toolkit'. Proc. 23rd Int. Systems and Software Product Line Conf. – Volume B', Paris France, 2019
[55] Hildebrandt, C., Scholz, A., Fay, A., *et al.*: 'Semantic modeling for collaboration and cooperation of systems in the production domain', 2017 22nd IEEE Int. Conf. on Emerging Technologies and Factory Automation (ETFA), Limassol, Cyprus, 2017, pp. 1–8
[56] Priego, R., Armentia, A., Estévez, E., *et al.*: 'On applying MDE for generating reconfigurable automation systems'. 2015 IEEE 13th Int. Conf. on Industrial Informatics (INDIN), Cambridge, UK., 2015, pp. 1233–1238
[57] Heinrich, R., Koch, S., Cha, S., *et al.*: 'Architecture-based change impact analysis in cross-disciplinary automated production systems', *J. Syst. Softw.*, 2018, **146**, pp. 167–185
[58] Fischer, J., Marcos, M., Vogel-Heuser, B.: 'Model-based development of a multi-agent system for controlling material flow systems', 2018, **66**, p. 438
[59] Aicher, T., Regulin, D., Schuetz, D., *et al.*: 'Increasing flexibility of modular automated material flow systems: A meta model architecture', *IFAC-PapersOnLine*, 2016, **49**, (12), pp. 1543–1548

[60] Legat, C., Mund, J., Campetelli, A., *et al.*: 'Interface behavior modeling for automatic verification of industrial automation systems' functional conformance', *At-Automatisierungstechnik*, 2014, **62**, (11), pp. 815–825
[61] International Electrotechnical Commission (IEC): 'IEC 62769–5 field device integration (FDI) – part 5: FDI information model', 2015
[62] Kernschmidt, K., Feldmann, S., Vogel-Heuser, B.: 'A model-based framework for increasing the interdisciplinary design of mechatronic production systems', *J. Eng. Des.*, 2018, **29**, (11), pp. 617–643
[63] Ulewicz, S., Vogel-Heuser, B., Member, S.: 'Industrially applicable system regression test prioritization in production automation', *IEEE Trans. Autom. Sci. Eng.*, 2018, **15**, (4), pp. 1839–1851
[64] Leitão, P., Karnouskos, S., Ribeiro, L., *et al.*: 'Smart agents in industrial cyber–physical systems', *Proc. IEEE*, 2016, **104**, (5), pp. 1086–1101
[65] Straube, F., Pfohl, H.-C., Günthner, W.A., *et al.*: 'Trends und strategien in der logistik, ein blick auf die agenda des logistik-managements 2010', 2005
[66] DIN: 'DIN SPEC 91345: reference architecture model industrie 4.0 (RAMI4.0)', 2016
[67] Verein Deutscher Ingenieure (VDI) e.V.: 'VDI/VDE 3695: engineering of industrial plants – evaluation and optimization – subject processes', 2010
[68] Greeff, G., Ghoshal, R.: 'Business process design models and concepts used in operations systems', in '*Practical E-manufacturing and supply chain management*' (Newnes, Boston, MA, USA, 2004), pp. 66–111