Analysis of Viewpoints for Modeling Cyber-Physical Systems

Anca Daniela Ionita
Automation and Industrial Informatics
University Politehnica of Bucharest
Bucharest, Romania
anca.ionita@upb.ro

Johannes Meier
Software Engineering Group
University of Oldenburg
Oldenburg, Germany
meier@se.uni-oldenburg.de

Christian Schönberg
Software Engineering Group
University of Oldenburg
Oldenburg, Germany
christian.schoenberg@uni-oldenburg.de

Andreas Winter
Software Engineering Group
University of Oldenburg
Oldenburg, Germany
winter@se.uni-oldenburg.de

***Abstract*— For modeling cyber-physical systems (CPS) there is a multitude of languages to choose from, and often one needs more than one of the existing languages to cover all concerns to be considered. The design of CPS needs to go across languages, pertaining to different paradigms, and creating overlapping and combined challenges for reaching coherence. An appropriate selection of languages for creating a new CPS requires an in-depth analysis of the architecture viewpoints relevant to all the stakeholders' concerns. This paper identifies and analyzes a set of viewpoints and their corresponding model kinds for UML and two of its profiles, SysML and MARTE, well-known for their applicability to cyber-physical systems. We discuss how the viewpoint coverage and the relationships between them can lead towards a recommendation system for choosing the right combination of languages, to be used by educators and software architects.**

Keywords—View-Based Modeling, Cyber-Physical Systems, Architecture Viewpoints, UML, SysML, MARTE

I. INTRODUCTION

Cyber-Physical Systems (CPS) represent one of the most prominent trends in computer technology, establishing integrated systems consisting of hardware and software components [1]. They have evolved in conjunction with Internet of Things (IoT) and various smart approaches, in manufacturing, transportation, health care, buildings, emergency response. As the world has become increasingly inter-related - due to services, mobile devices, cloud computing, and wireless networks - CPS have highly proliferated, facing challenges like scalability, composability, and heterogeneity. The state of practice shows that, in the development of various CPS, one uses a large variety of modeling languages, some descriptive, others executable, corresponding to different paradigms. In this context, our research question is why one selects a language or another for designing a given CPS.

Modeling is considered fundamental to software and systems engineering and choosing the right language for the system under development is not a simple task. Describing systems requirements is realized by multiple viewpoints that frame the concerns relevant to all the stakeholders. A *viewpoint* is a perspective from which a system is regarded. It defines, based on concerns of stakeholders, which parts of the system are regarded, and which are disregarded. The ISO/IEC/IEEE standard 42010:2011 [2] considers that each architectural viewpoint has one or more associated *model kinds* that specify a notation for the viewpoint, as well as usage conventions. Using these notations, concrete models can be created that belong to equally concrete *views*. Views are snapshots of the system under development from certain perspectives, i.e., following viewpoints.

An in-depth analysis of viewpoints is essential for cyber-physical systems because they integrate strongly interdependent, but interrelated parts, including "computation, networking, and physical processes" [3]. Since CPS have a growing number of sensors, actuators, communication channels and processing nodes, their development involves lots of different stakeholders, with lots of different concerns, which leads to lots of different viewpoints. To support them all, and to model all aspects of CPS, a systematic, multi-view and multi-paradigm modeling [4] method should be investigated.

First, this paper identifies the most important CPS concerns, framed by a set of broad viewpoints (Section II). However, to be able to differentiate between languages, one needs to get to another level of detail and identify narrower viewpoints, which have very specific types of diagrams that represent different model kinds. Then, Section III analyzes three standard languages - Unified Modeling Language (UML), Systems Modeling Language (SysML) and Modeling and Analysis of Real-Time and Embedded systems (MARTE) - and gives examples of applying them to describe cyber-physical systems, and section IV identifies a set of narrow viewpoints provided by them. Finally, section V discusses the coverage of these viewpoints by the three standard languages and identifies challenges for preserving the overall model consistency.

II. CPS CONCERNS

Cyber-physical systems have three groups of stakeholders: software engineers, application domain engineers and systems engineers. We represent their concerns, at a very high level of granularity, in the conceptual model from Fig. 1. Software engineers are particularly interested in the cyber concern, with the correspondent computational view, where it is very important to represent the software architecture and the diagrams are mainly graph-based. Domain specialists have the physical concern, associated with the hardware viewpoint and the representations of the topology and the network graphs, plus the physical phenomena viewpoint, often characterized based on differential equations. System engineers are concerned by the system as a whole, related to the control viewpoint, which uses mathematical modeling. We thus notice that CPS need multiple architecture viewpoints, for which the models conform to diverse paradigms.

2379-0482/23/$31.00 ©2023 IEEE
DOI 10.1109/CSCS59211.2023.00057

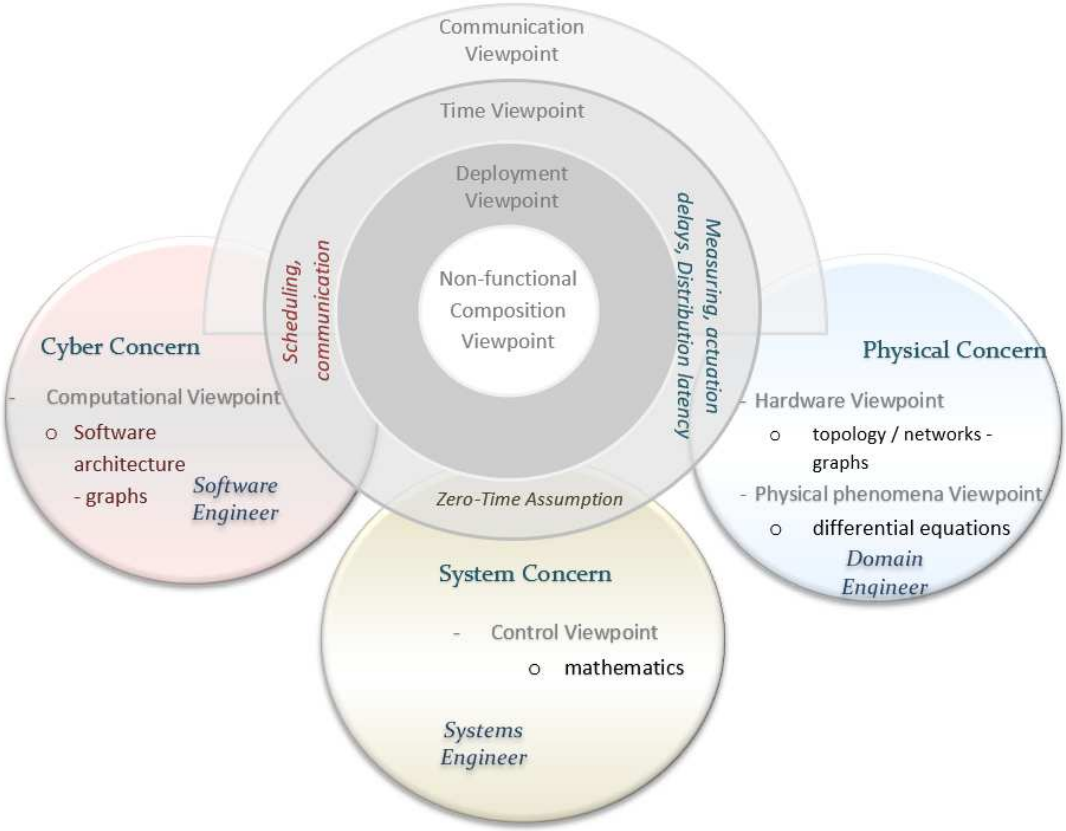

Fig. 1. CPS Conceptual Model

Moreover, apart from the three concerns mentioned above, there is a cross-cutting concern particularly related to the non-functional properties, time management, deployment, and communication. We call this the integration concern.

Fig. 2 represents the abstract syntax for the CPS concerns identified above, using the concrete syntax of the UML class diagrams. The two concepts that are the basis of the hierarchy conform to the ISO/IEC/IEEE 42010:2011(E) standard, together with the association between them. In addition, the diagram includes the four concerns of the cyber-physical systems (cyber, physical, system and integration). The class hierarchy from the right side represents the CPS stakeholders (software engineer, domain engineer and systems engineer). The representation also includes associations drawn as lines and showing which are the concerns for each stakeholder. Let us notice that all of them are connected to the Integration Concern.

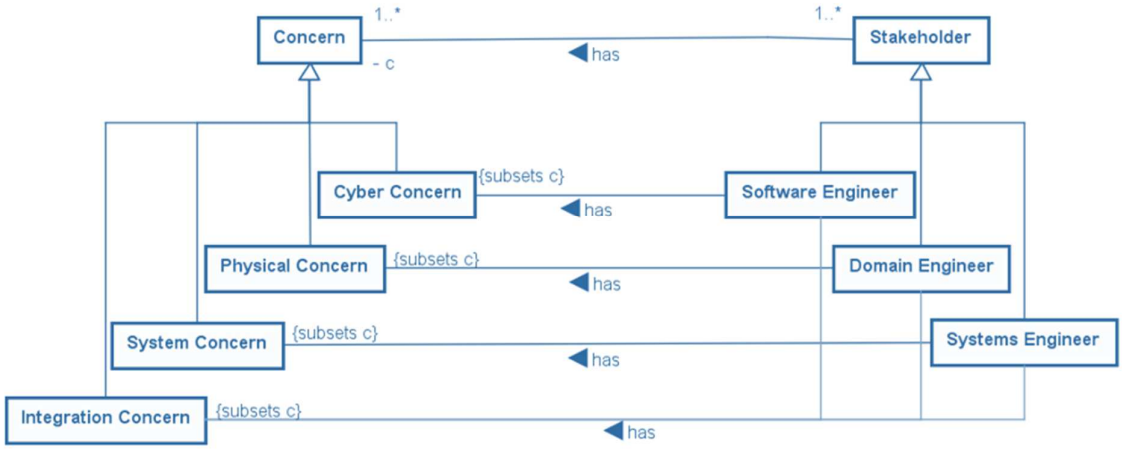

Fig. 2. Abstract Syntax for the CPS Concerns

III. Method and Related Work

In a systematic literature review, Barisic et al. identified 16 different formalisms adopted for the design of cyber-physical systems [5], like Petri Nets, discrete events, temporal logic, data flow, differential equations, bond graphs etc. Moreover, according to their study, the most used modeling languages in CPS development are UML and SysML, followed by MARTE and Architecture Analysis & Design Language (AADL) language.

Our research objective has been to investigate a large set of languages used for modeling cyber-physical systems and identify similarities and differences between them. The analysis proved that, although there are many commonalities of these languages, even within the same formalism the languages differentiate from each other by some essential details that correspond to narrow architecture viewpoints, which may be of great importance for some applications and not for others. Hence, the challenge is to properly differentiate between these languages by identifying these narrow viewpoints, as a starting point to defining selection rules and creating a recommendation system. Their granularity was chosen to allow a comparison between languages in order to:

- show the types of diagrams from different languages (i.e., model kinds) that represent the same viewpoint
- outline whether a language offers or not a representation for a given viewpoint
- make the difference between similar types of diagrams that cover a narrower or broader scope (i.e., one or several viewpoints).

The part of the research presented in this paper concerns UML, which is a general modeling language and a standard in object-oriented modeling, and two of its profiles – SysML and MARTE, also adopted as standards by the same organization - Object Management Group. These profiles were selected because they include extensions with metaclasses that are specific to systems' modeling, real-time and embedded systems, and because they are widely used. Therefore, they are very important for CPS and have multiple applications in real-life; several examples are given below.

As a general modeling language, UML (*Unified Modeling Language*) can also be employed for modeling cyber-physical systems [6]. As an example, UML activity diagrams are used to model the behavior of a robot, whose executability was realized with a Moka engine, based on fUML (The Foundational Subset for Executable UML Models) [7]. UML was also extended in profiles like SysML and MARTE, introducing concepts for a more advanced modeling of CPS.

SysML *(Systems Modeling Language)* reuses a part of UML, including sequence, state machine, use case and package diagrams. UML class and composite structure diagrams are extended to blocks, internal block diagrams, and allocations of behavior/structure to blocks, plus other modeling elements that keep the correspondence between similar models with more and less details. There is also the possibility to introduce constraints that define mathematical formulas, applied on a large variety of parameters - like mass, response time, availability, security, cost - and may be used for trade-off analysis [8]. SysML allows for the extension by customized viewpoints and views, as well as profiling. The standard was extended in a framework for designing time-sensitive CPS, applied in [3] for simulating a rotorcraft Unmanned Aerial Vehicle (UAV). An academic example of SysML models based on four different resources - a machine, a robot, a conveyor and a pump - is presented in [9].

MARTE (*Modeling and Analysis of Real-Time and Embedded systems*) adds specific and detailed modeling capabilities concerning non-functional properties, time, resource modeling, allocation, and model annotations useful for the analysis of performance. This language, used in conjunction with UML, is applicable for modeling systems that are time-critical, resource-critical, reactive to the environment, characterized by intensive data computations, or controlling physical objects or processes [10]. A case study for a quadcopter presented in [11] adds MARTE-specific elements to its UML model, e.g., the spatial distribution of a controller, the time scheduling for the actions that have to be performed on the controller, and the clock constraints for synchronization.

IV. Viewpoints Analysis

This section presents a set of narrow viewpoints identified for UML, SysML and MARTE and their correspondent model kinds / types of diagrams, which are relevant for CPS modeling. Section A describes viewpoints supported by UML, some of which are also covered by SysML and MARTE. These viewpoints were derived by deeply analyzing the model kinds of these modeling languages to allow a comparison between them, as presented in Section III. Section B presents a set of viewpoints that can only be specified in SysML and MARTE and are not supported by UML, or not in sufficient detail. The necessity for expressing the CPS architecture from such a point of view would indicate that UML is not appropriate enough and the designer should choose SysML or MARTE, instead or in addition. A summary of all these viewpoints and the correspondent model kinds is presented in Table I.

A. Viewpoints Supported by UML

This section mainly presents the viewpoints covered by UML, but it also identifies the situations where SysML or MARTE provide more modeling elements for the same model kind, or even additional types of diagrams, thus supporting a more detailed design.

The *Data* viewpoint focuses on static systems aspects, i.e., the types of entities - described together with their properties, the operations that may change them - and the relationships between these entities. The model kinds used in this viewpoint govern information models (for entities represented in an abstract way) or data models (for entities at a low level of abstraction). For instance, the description of the value range, the precision, the mean deviation, and the update interval of a sensor are described in the *Data* viewpoint.

The *Data Flow* viewpoint focuses on the movement and the storage of data, within a process made of activities that transform input data and produce output data, without considering their sequencing or parallelism. An example for this viewpoint is the data transfer from a sensor (as described above) to an actor, through a series of processing nodes.

The *Control Flow* viewpoint focuses on the sequencing of activities within a process, also including decisions, conditions, iteration, timing, parallelism, and handling events. The description of the logic governing the processing nodes mentioned above is an example for the *Control Flow* viewpoint.

The *State Flow* viewpoint focuses on the states and the transitions between them, driven by discrete external events,

or automatically triggered after the finalization of activities, and guarded by various conditions. An example is the definition of states in which a sensor or a processing node can be in, and how to transition between these states.

The *Interaction* viewpoint focuses on the way the entities inside and outside the system (at various levels of granularities) communicate (e.g., based on synchronous or asynchronous messages) and how they influence each other.

The *Communication* viewpoint, related to *Interaction*, shows the structure of information interchanged by entities that communicate and the rules or protocols for transmitting this information. Apart from the UML communication diagrams, MARTE adds message and flow-oriented communication modeling to its general component models.

TABLE I. SUMMARY OF MODEL KINDS FOR EACH VIEWPOINT

Viewpoint	**Language**	**Model Kinds / Descriptions**
Data	UML	Object Diagram
		Class Diagram
Data Flow	UML	Activity Diagram
		Information Flows Package
	SysML	Activity Diagram
Control Flow	UML	Activity Diagram
	SysML	Activity Diagram
State Flow	UML	State Machine Diagram
	SysML	State Machine Diagram
Interaction	UML	Sequence Diagram
		Communication Diagram
		Interaction Tables
	SysML	Sequence Diagram
Communication	UML	Communication Diagram
	MARTE	Generic Component Model (GCM) Package
Timing	UML	Timing Diagram
	MARTE	Time Modeling packages, CCSL and CVSL
Software Assembly Structure	UML	Component Diagram
	SysML	Block Definition Diagram
		Internal Block Diagram
	MARTE	Generic Component Model (GCM) package
Internal Structure	UML	Composite Structure Diagram
	SysML	Internal Block Diagram
Deployment	UML	Deployment Diagram
Functional Requirements	UML	Use Case Diagram
	SysML	Use Case Diagram
		Requirement Diagram
Non-functional Requirements	SysML	Requirement Diagram
	SysML	Parametric Diagrams
Non-functional Properties	MARTE	packages for Non-functional Properties Modeling (NFPs) and Value Specification Language (VSL)
		Generic Quantitative Analysis Modeling (GQAM)
		Schedulability Analysis Modeling (SAM)
		Performance Analysis Modeling (PAM)
Resources	MARTE	Generic Resource Modeling (GRM)
		Detailed Resource Modeling (DRM)
		Software Resource Modeling (SRM)
		Hardware Resource Modeling (HRM)
		High-Level Application Modeling (HLAM)
Allocation	SysML	Allocation extensions applied to Block Definition Diagram, Internal Block Diagram and Activity Diagram
	MARTE	Allocation Package
Scheduling	MARTE	Scheduling Package
		Allocation Package

The *Timing* viewpoint focuses on the time scale in a more precise way than the simple ordering on the timeline, including information about time durations and constraints that characterize interactions or state changes. It thus considers:

- logical time, showing causality
- synchronous time based on the simultaneity with clock instants
- real-time duration values.

Several model kinds are UML timing diagrams (also describing the *Interaction*) and the more complex time modeling extensions from MARTE, like multiple time bases, clocks, time values, duration values; for a more precise specification, MARTE is also accompanied by CCSL (Clock Constraint Specification Language) and Clocked Value Specification Language (CVSL).

The *Software Assembly Structure* viewpoint only considers the software parts. One assumes that a software entity (be it a class, a component, or a subsystem) may be composed of several parts but, at the same time, it can hide its parts and expose interfaces for becoming a part in a larger assembly. Multiple examples of model kinds are available in UML, SysML and MARTE, as shown in Table I.

The *Internal Structure* viewpoint focuses on the properties that are inside an entity and how they are connected to each other; it may also show the structure of the collaboration between the parts for fulfilling a goal. Two model kinds that express it are the UML composite structure diagram and the SysML internal block diagram.

The *Deployment* viewpoint shows the assignment of software parts to the hardware that supports the execution, sometimes by using several intermediate conceptual or physical elements, like environments or platforms. A model kind is the deployment diagram in UML / SysML.

The *Functional Requirements* viewpoint covers the functionality expected by external entities - like humans, hardware, or other systems - from the system to be developed; the entities are only considered in terms of their roles played in

respect with the system; the viewpoint is also interested of the internal functionality that realizes the externally accessible functionality. The use case diagram in UML / SysML represents a well-known model kind.

B. Additional Viewpoints Supported by SysML and MARTE

This section identifies viewpoints (essential for some CPS) that are not covered by UML, leading to the clear need to use its specialized profiles like SysML or MARTE.

The *Non-functional Requirements* viewpoint focuses on non-functional properties that are prescribed for the cyber-physical system. SysML supports a model kind for it with its requirements diagrams, where one does not make the functional / non-functional difference.

The *Non-functional Properties* viewpoint covers what conditions, constraints or criteria are characteristic or designed for an entity, including their precise specification for quantitative analysis. Some model kinds are the parametric diagram from SysML and MARTE package for non-functional properties modeling, together with the language for specifying values. MARTE provides a more detailed support for quantitative analysis modeling of workload, observers, and resources, with attributes relevant for real-time systems, like host demand, delay, throughput; even more specific quantitative annotations are available for the schedulability, and for the performance of best-effort and soft-real-time embedded systems.

The *Resources* viewpoint focuses on the system entities that offer services, like memory, timers, clocks, processing devices, communication media, concurrent execution arbiters etc. They may be software or hardware, physical or logical, and they are often modeled for CPS, where resource availability may be limited. MARTE supports several model kinds for this viewpoint with generic and detailed resource modeling, and the differentiation between modeling elements for software (including multi-tasking) and hardware; it also adds specific resources for managing concurrency and real-time, within the high-level application modeling.

The *Allocation* viewpoint focusses on the cross-relationships between elements that may pertain to different models, for allocating behavior, structure, or flows, generally by mapping logical to physical parts from the execution platform. It may be relevant in various cases: allocating computations to processing elements, data to memories, control dependencies to communication resources. MARTE introduces a dedicated model kind for *Allocation* that represents spatial distribution to resources (see Table I).

The *Scheduling* viewpoint focuses on the temporal arrangement of units of execution (e.g., activities, tasks, processes, threads) at run-time, under concurrency conditions over the same resources. It also considers policies, algorithms, and synchronization resources. MARTE covers this viewpoint in two model kinds: the Scheduling package and the Allocation package, which also considers the scheduling of the allocated algorithmic parts (e.g., threads, tasks etc.).

V. Discussion

After describing and exemplifying the set of viewpoints from Section III, we discuss below whether they are covered by only one or by several of the modeling languages under study (see Section A) and how difficult it is to distinguish these viewpoints from each other (see Section B). The resulting challenge, to automatically keep the overlapping viewpoints consistent to each other, is motivated in Section C.

A. Comparison of CPS Viewpoints Coverage for UML, SysML, and MARTE

The narrow viewpoints identified for UML, SysML and MARTE and presented in Section III are not orthogonal. Some of them are overlapping and some are included in other viewpoints, but we considered them important for differentiating between the modeling languages under study. Table II shows the viewpoints coverage for UML, SysML and MARTE. The main observation for CPS modeling is that all the three languages may be required, since each of them provides at least one viewpoint that is not covered by the others. This comparison may be a starting point for deciding on the choice of the most appropriate modeling language, or of a combination of them, in respect with the set of viewpoints framed by the stakeholders' concerns.

TABLE II. Analysis of Viewpoints Coverage

Viewpoint	Modeling Language		
	UML	***SysML***	***MARTE***
Data	x		
Data Flow	x	x	
Control Flow	x	x	
State Flow	x	x	
Interaction	x	x	
Communication	x		x
Timing	x		x
Software Assembly Structure	x	x	x
Internal Structure	x	x	
Deployment	x	x	
Functional Requirements	x	x	
Non-functional Requirements		x	
Non-functional Properties		x	x
Resources			x
Allocation		x	x
Scheduling			x

B. Distinction between Viewpoints

Within the same viewpoint, some model kinds introduce more details than others, either within the same modeling language or within different ones. Within the same language, one possibility is to have two types of diagrams, allowing one to define simpler models with one type of diagram and then refine them with another type of diagram. Another possibility is to have a single type of diagram, like the component diagram in UML, and to specify more or less details at modeling time. Moreover, some of the UML diagrams (model kinds) cover more than one of the narrow viewpoints identified above (see Table I). Since viewpoints with different degrees of abstraction are required to support the needs of stakeholders with different technical backgrounds, the criteria to distinguish between viewpoints target not only the scope described, but also the level of abstraction.

The *Deployment* viewpoint is included in the *Allocation* viewpoint, as the types of elements that are bound are less general than for allocation; it considers the distribution of concrete software elements from the physical world to hardware or execution environments, whereas the allocation may also be done for various logical parts of structural or behavioral nature. André et al. give a comparative presentation of allocation models, including UML deployments, as well as the approaches of allocation from SysML and MARTE [12].

The *Allocation*, *Resources* and *Scheduling* viewpoints are overlapping, like in concurrent systems, where processes are allocated to resources and their run-time execution is scheduled for the resource they were allocated to. Here, vague distinctions between viewpoints result from overlaps of the viewpoints regarding the described concepts.

The *Non-functional Properties* viewpoint is strongly related to the *Non-functional Requirements* one, but it is concerned of elements necessary for quantitative analysis and design. For *Non-functional Properties* one can also identify many narrower viewpoints that are strictly focused on one specific non-functional property (e.g., adaptability, cost, reliability, energy, security, safety, stability etc.), but specific model kinds for such narrow viewpoints are not consecrated. This viewpoint is generally linked to crosscutting concerns, as the non-functional properties may characterize a large variety of modeling elements describing functional aspects. Again, several viewpoints have overlaps and describe connected concepts of the same CPS together.

C. Combination and Consistency of Overlapping Viewpoints

As discussed above, multiple viewpoints are required to describe different aspects of cyber-physical systems and to support different stakeholders. Since the viewpoints are often not orthogonal to each other regarding the modeling scope and the level of abstraction, the same concepts can be described by several viewpoints in the same or a similar way. This leads to the challenge of selecting an appropriate set of languages and viewpoints, as discussed in Section A.

The next challenge is to keep the corresponding views of the selected viewpoints consistent to each other, when modeling a concrete CPS. If one developer changes the CPS using the first view(point), the views presented to other developers have to be changed accordingly, preferable in an automatic way. Since different views represent different aspects of a single architecture description of the CPS, changes to a single viewpoint must also be made to the underlying description, which in turn must affect all other related viewpoints.

Additionally, since lots of different view(point)s are required to describe different aspects of a system, as shown for cyber-physical systems in Table II, the relation(ship)s between view(point)s have to be made clear and explicit. For example, the *Functional Requirements* regarding the data exchanged between different sensors have to correspond to what is realized in the *Data* view. If the modeled data or the requirements change, the traceability would help to change the dependent information accordingly, and to understand the system and its design decisions in general.

Summarizing, new approaches and supporting frameworks are required for a unified and consistent modeling of the entire CPS with many different viewpoints. This also corresponds to solutions from [13].

VI. Conclusion

Based on the analysis of viewpoints that may be modeled with UML, SysML and MARTE, this paper identified two important challenges for modeling CPS. The first challenge is to select the modeling languages and types of diagrams (model kinds) that cover all the stakeholders' needs; the analysis and results of this paper simplify this selection and confirm that a combination of UML, MARTE and SysML may be required to cover all the concerns (see Table II). Future work will also consider combinations of languages that go across different paradigms. As such a selection may become more complex, it leads towards the necessity of a recommendation system to be available for educators and CPS designers. The second challenge is to keep the consistency between overlapping or related view(point)s, which has to be automated and supported by new approaches and frameworks.

References

[1] Platzer, A. (2018). Cyber-Physical Systems: Overview. In: Logical Foundations of Cyber-Physical Systems. Springer, Cham. https://doi.org/10.1007/978-3-319-63588-0_1

[2] IEEE Computer Society, Architecture Working Group, "Systems and software engineering — Architecture description ISO/IEC/IEEE 42010", 2011.

[3] M. Morelli, "A System-Level Framework for the Design of Complex Cyber-Physical Systems from Synchronous-Reactive Models", Corso di perfezionamento in Tecnologie Innovative, Scuola Superiore Sant'Anna di Studi Universitari e di Perfezionamento, 2014-2015.

[4] Carreira, P., Amaral, V., Vangheluwe, H. (2020). Multi-Paradigm Modelling for Cyber-Physical Systems: Foundations. In: Carreira, P., Amaral, V., Vangheluwe, H. (eds) Foundations of Multi-Paradigm Modelling for Cyber-Physical Systems. Springer, Cham. https://doi.org/10.1007/978-3-030-43946-0_1

[5] Ankica Barišić, Ivan Ruchkin, Dušan Savić, Mustafa Abshir Mohamed, Rima Al-Ali, Letitia W. Li, Hana Mkaouar, Raheleh Eslampanah, Moharram Challenger, Dominique Blouin, Oksana Nikiforova, Antonio Cicchetti, Multi-paradigm modeling for cyber–physical systems: A systematic mapping review, Journal of Systems and Software, Volume 183, 2022, 111081.

[6] "OMG® Unified Modeling Language® (OMG UML®)", Version 2.5.1, December 2017.

[7] K. Suri, A. Cuccuru, J. Cadavid, S. Gerard, W. Gaaloul, and S. Tata, "Model-based Development of Modular Complex Systems for Accomplishing System Integration for Industry 4.0.", in Proc. of the 5th Int. Conf. on Model-Driven Engineering and Software Development (MODELSWARD), pp. 487-495, 2017.

[8] "OMG Systems Modeling LanguageTM", Version 1.7, August 2022.

[9] S. Kanthabhabhajeya, P. Falkman, and B. Lennartson, "System Modeling Specification in SysML and Sequence Planner Language - Comparison Study", in Proc. of the 14th IFAC Symposium on Information Control Problems in Manufacturing, Vol. 65, issue 6, pp. 1543-1550, May 23-25, 2012.

[10] "UML Profile for MARTE: Modeling and Analysis of Real-Time Embedded Systems", Version 1.2, formal/19-04-01, April 2019.

[11] F. Mallet, E. Villar, and F. Herrera, "MARTE for CPS and CPSoS: Present and Future, Methodology and Tools", in S. Nakajima et al. (eds.), Cyber-Physical System Design from an Architecture Analysis Viewpoint, Springer, pp. 81-108, 2017.

[12] C. André, F. Mallet, and R. de Simone, "Modeling Time(s)", in G. Engels, B. Opdyke, D.C. Schmidt, F. Weil (eds.), Int. Conf on Model Driven Engineering Languages and Systems (MODELS), 2007. Lecture Notes in Computer Science, vol 4735. Springer, 2007.

[13] J. Meier, H. Klare, C. Tunjic, C. Atkinson, E. Burger, R. Reussner, A. Winter, "Single Underlying Models for Projectional, Multi-View Environments", in: Proc. of the 7th Int. Conf. on Model-Driven Engineering and Software Development, pp. 119-130, Prague, SCITEPRESS, February 2019.