Methodological Approach to Model and Validate CPS

Perla Tannoury, Ahmed Hammad
University of Bourgogne Franche-Comté, Besançon, France, FEMTO-ST Institute, CNRS
{*perla.tannoury, ahammad*}*@femto-st.fr*

Keywords: CPS, Specifications, Validation Techniques, System Engineering, Semi-formal Modeling, Verification.

Abstract: As Cyber-Physical Systems (CPS) become increasingly complex and critical, ensuring high-quality specifications is crucial. However, the process is often challenging due to the need for expertise and lack of guidance. To address this, we propose a methodological framework and tools for CPS specification and validation. Our approach utilizes semi-formal modeling and incorporates SysML and AcTRL technologies. By applying validation strategies early in the development process, we aim to reduce the costs of correcting errors. This comprehensive approach offers vital support to designers navigating CPS complexities.

1 INTRODUCTION

In recent years, Cyber-Physical Systems (CPS) have become increasingly complex, integrating diverse components and technologies (Walia et al., 2024; Yu et al., 2023). These systems exhibit intricate behaviors, challenging conventional analysis and prediction methods. Their complexity stems from the dynamic interplay among components, enabling real-time sensing, processing, and actuation, often resulting in unpredictable behaviors.

Moreover, the criticality of CPS, particularly those involving human safety or significant financial risks (Hossain et al., 2024; Barroso et al., 2023; Tripathi et al., 2021), underscores the necessity for stringent quality standards. To exacerbate the challenge, the competitive global market demands cost-effectiveness and time efficiency from CPS developers.

Addressing the intricacies of CPS development requires a systematic approach to meet diverse requirements and constraints such as cost, time, quality, and operation. System Engineering (SE) has emerged as a framework to formalize and manage the design of complex CPS and products (Wade et al., 2015).

Advocating for a Model-Based Systems Engineering (MBSE) paradigm, INCOSE (Friedenthal et al., 2007) suggests employing semi-formal graphical languages, like a UML profile, to model CPS software and hardware components (Ordinez et al., 2020; Chatterjee and Reza, 2020). +

Model-Driven Engineering (MDE) has significantly advanced CPS development by focusing on abstract concerns rather than immediate practical implementation (Mohamed et al., 2021a; Mohamed et al., 2021b). Specifying a CPS involves defining its requirements, structures, and behaviors, categorized into functional and non-functional aspects. Functional requirements specify the system's functions, while non-functional requirements encompass performance, reliability, and security considerations.

Various methodologies, such as SysReo (Tannoury et al., 2022; Tannoury, 2022; Tannoury et al., 2023), ModelPlex (Mitsch and Platzer, 2016), and INTO-CPS (Thule et al., 2019) have been proposed to validate complex system designs. However, validating heterogeneous complex CPS requires a comprehensive approach that addresses both functional and non-functional requirements. This approach should facilitate automated validation through formal verification and simulation, grounded in requirements modeling and ensuring traceability of the system's structure and behavior.

In this paper, we propose a comprehensive approach to facilitate CPS design, incorporating tools for specifying and validating these systems. Our method emphasizes semi-formal modeling of system requirements and integrates environments tailored for verification and simulation. By applying validation approaches early in the system development process, we aim to mitigate the costs associated with correcting specification and design errors. This integrated approach not only streamlines the design process but also enhances the overall robustness of CPS by identifying and addressing potential issues at an early stage. Additionally, our methodology seeks to foster collab-

oration and knowledge sharing among diverse stakeholders involved in the CPS design process, promoting a more holistic and efficient development lifecycle.

The paper is structured as follows. Section 2 outlines the validation approach, while Section 3 elaborates on validation through simulation. Section 4 delves into the validation method via formal verification. Finally, Section 5 concludes the paper.

2 VALIDATION APPROACH

2.1 Introduction

Ensuring early validation of complex CPS requires a tailored approach, integrating simulation for non-functional and verification for functional requirements. Our methodology initiates validation from project outset, merging audit and simulation into specification. We endorse SysML for specification, employing system engineering techniques to transform SysML into simulation and/or verification environments. This paper motivates language selection before detailing our approach.

2.2 Presentation of the approach

Our validation approach begins with meticulous modeling of Cyber-Physical Systems (CPS). We start by identifying and classifying requirements into functional and non-functional categories using SysML. This phase establishes clear relationships among requirements, forming a solid foundation.

Moving to the next phase, we model the CPS's structure and behavior using SysML diagrams such as blocks, internal blocks, parametric structures, sequence, and activity diagrams. Critical traceability links and verification connections between requirements and model elements are established, ensuring a comprehensive representation.

These initial phases constitute the modeling stage, focusing on creating accurate representations. We then use model-driven engineering techniques to transform SysML diagrams into simulation or verification environments tailored to functional and non-functional requirements.

Finally, we employ simulation environments like VHDL-AMS (Haase et al., 2003) and Modelica (Otter and Elmqvist, 2001)), alongside verification tools like Model-checker CPN-Tools (Jensen et al., 2007) and SPIN (Holzmann, 1997), to compare design models with requirement models. This ensures alignment, enhancing the overall robustness and reliability of the CPS. Figure 1 outlines the sequential steps of our approach, from requirement identification to simulation and verification.

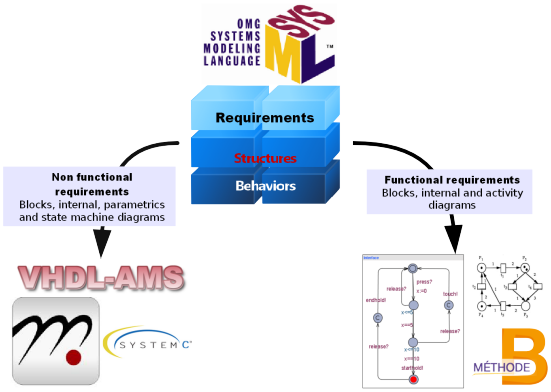

Figure 1: Specification and validation methodology.

3 VALIDATION APPROACH BY SIMULATION

3.1 Context and Motivations

In this section, we explore the integration of SysML with VHDL-AMS and Modelica, as illustrated in Figure 1. Emphasizing simulation's role, we use it to visualize system designs comprehensively and preempt integration challenges. By testing virtual prototypes, developers can validate requirements and identify potential issues, enhancing the validation process's efficacy.

The design and validation of complex CPS pose scientific and industrial challenges due to the integration of heterogeneous components from various domains. Non-functional requirements, such as safety and reliability, are formalized using temporal logic languages, while quantitative constraints, like energy consumption and performance, are expressed through mathematical equations (Wang et al., 2012).

Validating non-functional requirements, especially those expressed by partial differential equations, presents challenges in formal verification. Computer simulation offers a pragmatic alternative, effectively modeling physical constraints and complex phenomena within computer-based environments.

By providing insights into complex physical processes, simulation serves as a cost-effective means to explore CPS development intricacies and understand system behavior in controlled virtual settings. SysML, with its parametric diagram feature, facil-

itates seamless integration with simulation environments (Gauthier, 2013).

Our approach leverages SysML for CPS specification, transforming models into simulation languages. Continuously refining this integration, we explore the potential synergy of SysML with SystemC (Gauthier, 2013), positioning our approach as a robust framework for modeling, analysis, and simulation in CPS development.

3.2 Joint use of SysML and VHDL-AMS

3.2.1 Introduction

Modeling is crucial for managing complexity and ensuring system security and compliance with defined requirements. This challenge is particularly evident in Systems on Chip (SoCs) and embedded systems, where the gap between software and hardware is narrowing (Prevostini and Zamsa, 2007). Collaboration among development teams and clients is essential to define needs, clarify requirements, and ensure specifications meet expectations.

System engineering methods rely on modeling, verification, and simulation to validate requirements and enhance the quality and reliability of critical systems. Graphical modeling languages, such as SysML, serve as powerful tools for modeling complex Cyber-Physical Systems (CPS) (Holt and Perry, 2008). Our proposed methodology, illustrated in Figure 2, combines SysML with VHDL-AMS and PSL (Pêcheux et al., 2005).

SysML, an extension of UML2, offers a simple and effective vocabulary for system engineering (Holt and Perry, 2008). It allows for concrete system representations and specification implementation in various development environments, ensuring coherence between modeling and implementation.

For software, commonly used languages include Java or C++, while Hardware Description Languages (HDLs) are used for material description (Arnold et al., 2005; Harris and Harris, 2022). VHDL, SystemC, and Verilog are among the most commonly used HDLs (Christen and Bakalar, 1999; Palnitkar, 2003).

Simulation is crucial for systems integrating analog and digital technology. VHDL-AMS addresses this by allowing modeling of abstracted objects and treatment of values in continuous time (Christen and Bakalar, 1999).

The Property Specification Language (PSL) enables writing complex temporal logic properties for circuit verification, including structured language with formally defined semantics, model-checking, and parallel analysis of HDL functional simulation properties.

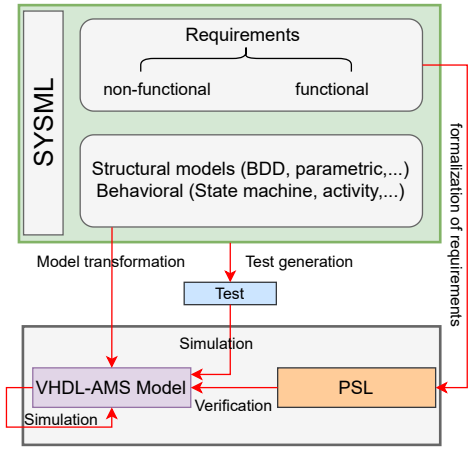

Figure 2: SysML, VHDL-AMS and PSL.

3.2.2 Case Studies

In the ANR Smart Surface project, dedicated to developing a Smart Surface based on autonomous distributed micro-robotic systems for robust and adaptive micromanipulation, an effective synergy between SysML and VHDL-AMS was discovered. This was evident in the study conducted by Giorgetti, showcasing a convergence of concepts between SysML and VHDL-AMS (Giorgetti et al., 2010).

In another context, within the SyVAD project situated in the Franche-Comté region in France, the primary objective is to control the boundary layer using distributed generators of inclined jets for a variable-speed automobile under normal conditions. The research focused on analyzing the airflow around the Ahmed body, a simplified yet extensively studied car geometry (Ahmed et al., 1984). This body serves as a fundamental model for comprehending fluid flow around cars, and it has been widely utilized in numerical simulations aimed at minimizing drag (Bouquet et al., 2012). These projects exemplify the versatile application of SysML in conjunction with VHDL-AMS and the significance of their integration in diverse research initiatives, illustrating their adaptability across different domains within the field of Cyber-Physical Systems.

4 VALIDATION METHOD BY FORMAL VERIFICATION

4.1 Introduction

This section focuses on formally verifying complex Cyber-Physical Systems (CPS) through the combined use of SysML and Hierarchical Colored Petri Net models (HCPN) (Huber et al., 1991).

In the development cycle of complex CPS, verification is crucial for ensuring adherence to specified requirements. While SysML is effective for describing complex systems, it lacks built-in verification mechanisms. To address this, integrating SysML with formal languages enhances the verification process, leveraging SysML's robust specification capabilities while ensuring alignment with essential requirements. This integrated approach enables a comprehensive assessment of complex CPS, upholding the highest standards of compliance and reliability.

4.2 Verification process

We present our proposed approach to specify and verify complex systems. The initial step involves specifying the system and its requirements. This phase encompasses describing the requirements, structure, and behavior of the system while accounting for the verification process. Requirements specification relies on the SysML requirements diagram. Functional requirements, particularly those intended for verification on activity diagrams, need to be formalized using AcTRL (Activity Temporal Requirement Language) (Rahim et al., 2014). The formalization of requirements aims to eliminate any ambiguity regarding their semantics to enable formal verification. Additionally, the specification of system behavior and structure is SysML-based. Specifically, the block definition diagram is used to depict the system's structure, while the activity diagram is employed to delineate its behaviors.

The second step in the approach involves considering the relationships between requirements and model elements to identify the diagrams relevant for the verification phase. This traceability serves as a guide for the verification of requirements on the design templates. Subsequently, the SysML diagrams under consideration are transformed into formal specifications to articulate their execution semantics and facilitate the use of formal verification tools. However, formulating such a transformation is challenging and necessitates a comprehensive understanding of both the source and target domains to identify structural and semantic equivalences between the elements

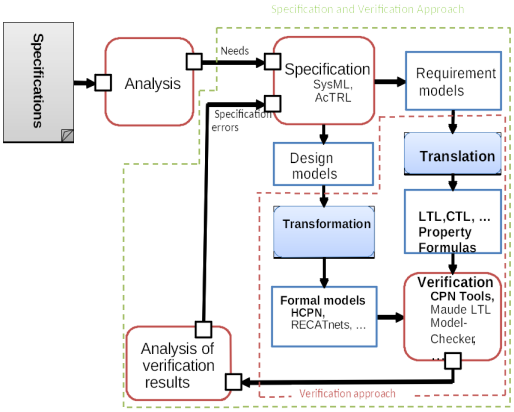

Figure 3: Specification and verification approach.

in each domain.

Given their resemblance to activity diagrams, Hierarchical Colored Petri Nets (HCPN) were chosen. Temporal logic languages were used to articulate the requirements.

Once the translation into formal specifications is done, the SysML requirements verification is started based on the Model-Checking technique. The verification based on the Model-Checking is completely automatic and allows to check the satisfaction of the properties written in temporal logic. It also provides counterexamples when the formula tested is not satisfied. This is the scenario (execution trace) that led to the error. These counter-examples are very useful because they provide important indications for model correction under verification.

The interpretation of the results of the verification on the high level specification (SysML specification) is the last step in this specification and verification process. This step of interpreting audit results relies on human expertise. It must be able to detect the origin of the error in the SysML specification (error in the model requirements or in the design models). This approach is described in Figure 3.

4.3 Formalization of the requirements

Our proposal for the formalization of the requirements concerning behaviors modeled by activity diagrams consists in the definition of the language AcTRL (Rahim et al., 2014). This language was designed for use by SysML practitioners. It allows to formalize, while remaining at a level of abstraction close to SysML, the requirements described by natural texts, in expressing them as temporal properties concerning the elements of the activity diagrams. In the SysML requirements diagram, the designer can bind a requirement **R** to the activity A, by a **"Verify"**

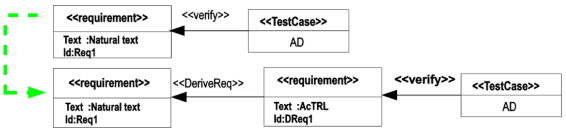

Figure 4: Representation of the requirements expressed in AcTRL.

relationship to express that the activity **A** satisfy the requirement **R**. If this requirement is functional and concerns a behavior of the system modeled by an activity diagram, it can surely be expressed according to the states of the elements of SysML activities. So, a possible formalization (at the SysML level) of the requirements related to the activities consists in expressing them according to the states of the elements of the activity diagram. It's starting from this idea, the proposed AcTRL language. This language is therefore intended for SysML designers to describe the requirements written in natural language in a language more formal but close to SysML. It only concerns the requirements related to SysML activities. It makes it possible to express a requirement as a property concerning the elements of the activity on which this requirement will be verified. To define this language, the approach is rooted in establishing a high-level representation of the operational semantics of a SysML activity. This representation conceptualizes all potential states resulting from the execution of the activity, resembling a system of states/transitions. Following this definition, predicates are established to characterize the states of elements within the activity diagram and articulate their properties. OCL (Object Constraint Language) (OMG, 2012) is used to express object and object node properties within this framework. Subsequently, the language is enriched with time operators leveraging the system of specification patterns outlined in prior literature (Dwyer et al., 1999).

For traceability reasons, the requirements expressed in AcTRL language are represented in the SysML requirements diagram and linked to the requirements expressed in natural language. This is illustrated in Figure 4. It is observed that following the formalization of the **Req1** requirement, which is linked by a "Verify" relation to a SysML element stereotyped as "test case" representing a SysML activity, another requirement **DReq1** expressed in AcTRL is generated. The latter is linked to the requirement **Req1** by the "DeriveReq" relation, while the "Verify" relation is established between the requirement **DReq1** expressed in AcTRL and the element "Test Case".

4.4 Transformation in formal specifications

Our approach is based on translating the activity diagrams into Hierarchical Colored Petri Nets (HCPN) (An et al., 2018). Before detailing this transformation, we present the HCPN.

4.4.1 Hierarchical Colored Petri Nets

The HCPN are a class of high-level Petri nets proposed for compositional modeling of systems. An HCPN model is constructed from several colored Petri nets connected to each other. In HCPNs, the substitution transition concept makes reference to a colored Petri net called subnet. The latter is described in a separate page and gives a more detailed description of the part of the system modeled by the substitution transition. The notion of substitution transition is used to structure the HCPN in several pages (several subnets). A substitution transition has places at the entrance and places at the exit called places sockets. The relationship between a substitution transition and its subnet is given by the specification that links the subnetwork places with the sockets of the substitution transition.

4.4.2 PNML (Petri Net Markup Language)

PNML (Petri Net Markup Language) is a standardized format for Petri nets, specified in ISO/IEC 15909 (Billington et al., 2003). It aims to enable interoperability among Petri net-based tools. In our approach, we propose using PNML as an intermediate format for translating activity diagrams into Petri nets. This choice leverages PNML's tool-independence and the availability of meta-models within the PNML framework (Hillah et al., 2010).

4.4.3 CPN and Tools

The CPN tool, developed by the CPN Group at the University of Aarhus, is widely used for analyzing and verifying Petri nets (Jensen et al., 2007). It supports editing, simulation, and verification of ordinary and colored Petri nets, as well as HCPN. CPN Tools includes a simulator, state space analysis, and integrates a model-checker for ASK-CTL (Billington et al., 2003).

4.4.4 Transformation of activity diagrams in HCPN

Various translations of activity diagrams to Petri nets have been proposed (Foures et al., 2011; Vladimirovich et al., 2015), each with specific goals. In this

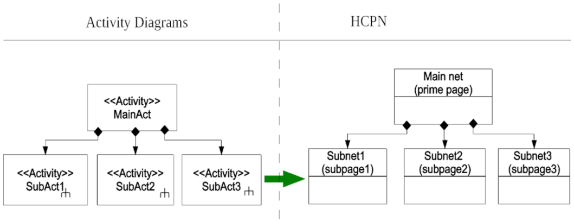

Figure 5: Translation of the structure.

work, we aim to check requirements on activity diagrams by automatically translating them into a formal specification based on HCPNs.

Our approach considers the composition of activities, including activity call actions, to preserve the structure and semantics of the activity diagrams in the resulting HCPN models. The translation involves two steps: an M2M transformation using ATL to convert the activity diagram into a PNML Petri net, followed by an M2T transformation generating an HCPN in CPN Tools format from the PNML representation.

To maintain the composite structure of activity diagrams, each sub-activity is translated into a subnetwork of the HCPN, represented on separate pages. The main activity of the diagram corresponds to the primary page of the HCPN, as illustrated in Figure 5.

4.4.5 Translation of requirements into temporal logic formulas

The AcTRL language enables high-level specification of requirements for verification by different approaches, depending on the verification tools used. To verify SysML requirements expressed in AcTRL on the HCPN translated from an activity diagram, the requirements are translated into temporal logic language. As the CPN Tools tool is chosen for verification, the requirements expressed in AcTRL must be translated into the ASK-CTL temporal logic. The ASK-CTL logic is a variant of the CTL logic, supported by the CPN Tools tool.

The passage from AcTRL to ASK-CTL is carried out in two stages. The first step is based on the library proposed in (Dwyer et al., 1999) to switch from AcTRL to CTL. The second step is based on the equivalences between CTL and ASK-CTL to switch from CTL to ASK-CTL. The interpretation of the AcTRL predicate expressions on the RdPHC is defined according to the translation of the HCPN activity diagrams.

4.5 Verification of the SysML requirements

Once the SysML specifications (SysML activity diagram and requirements) are translated into formal specifications (CPN Tools CPN and ASK-CTL Formulas), the verification step can be initiated using the tool CPN Tools. As part of our approach, Figure 6 illustrates the verification of SysML requirements on activity diagrams using the CPN Tools tool. The verification process for ASK-CTL formulas with the CPN Tools tool involves the following steps:

- Generation of the state space,
- Invoking the ASK-CTL library with the following command: use (ogpathˆ ”ASKCTL / ASKCTL-loader.sml”)
- Opening the file containing the ASK-CTL formulas,
- Evaluation of ASK-CTL formulas.

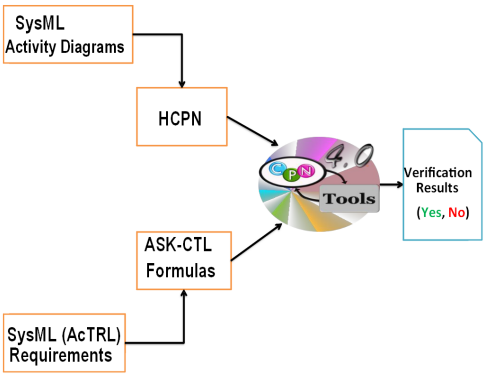

Figure 6: Verification of the SysML requirements via CPN Tools.

5 CONCLUSION

In this paper, we introduce a methodology to specify and validate complex CPS, leveraging SysML and emphasizing precise requirement descriptions and the utilization of relationships between SysML model elements. SysML, well-suited for heterogeneous complex CPS, facilitates hierarchical block modeling of hardware and software, and enables graphical representation of mathematical equations defining physical behaviors.

However, despite the various advantages of SysML, it remains a semi-formal language lacking direct execution or simulation capabilities, making val-

idation challenging. This approach aims to integrate requirement traceability and Model Driven Engineering (MDE) techniques to transform SysML models into simulation (e.g., Modelica, VHDL-AMS, SystemC) and verification (e.g., SPIN (Holzmann, 1997)) formats.

Functional and non-functional requirements in complex CPS systems vary in nature. Non-functional requirements are often validated through simulation, achieved by converting SysML diagrams, such as the parametric diagram, into simulation environments. On the other hand, functional requirements undergo validation through formal verification. We propose a temporal language, closely aligned with the SysML environment to specify functional requirements, which can be expressed on SysML diagrams and subsequently transformed along with SysML models into verification environments.

When developing a new methodological approach, the construction of software tools to support it is crucial. The aim is to provide researchers with a methodological guide for the specification, design, and development of their systems.

The first step in our methodology is the requirement specification step. The role of this step is to define and model the requirements after the needs analysis described in the loads. In engineering, the validation of the requirements and their evolution over the whole life cycle of a hardware project, software, or mixed software allows the development of a common understanding of all stakeholders of the system to develop. Requirements accurately measure system or software compliance in relation to needs or constraints. To enrich our approach, SysML requirements will be briefly analyzed. The first step involves formalizing them to express them precisely and remove any ambiguity. Formalizing requirements is essential for verification. The formal language AcTRL is proposed to formalize requirements related by the **"Verify"** relationship with activity diagrams. Extending this language to incorporate other categories of requirements would be beneficial. This step can also be used to validate and/or formally verify operations on requirements, including refinement. The SysML **"Refine"** relationship details a requirement by another or by a modeling element, enabling traceability between requirements and model elements. The formal verification of refinement of SysML requirements is not addressed in the literature. Investing in this area will propose a process to verify the correctness of the **"Refine"** relationship. Generalizing this approach to other relationships like **Derive**, **Contains**, **Satisfy** is necessary.

REFERENCES

Ahmed, S. R., Ramm, G., and Faltin, G. (1984). Some salient features of the time-averaged ground vehicle wake. *SAE transactions*, pages 473–503.

An, Y., Wu, N., Zhao, X., Li, X., and Chen, P. (2018). Hierarchical colored petri nets for modeling and analysis of transit signal priority control systems. *Applied Sciences*, 8(1):141.

Arnold, K., Gosling, J., and Holmes, D. (2005). *The Java programming language*. Addison Wesley Professional.

Barroso, S., Bustos, P., and Nunez, P. (2023). Towards a cyber-physical system for sustainable and smart building: a use case for optimising water consumption on a smartcampus. *Journal of Ambient Intelligence and Humanized Computing*, 14(5):6379–6399.

Billington, J., Christensen, S., Van Hee, K., Kindler, E., Kummer, O., Petrucci, L., Post, R., Stehno, C., and Weber, M. (2003). The petri net markup language: Concepts, technology, and tools. In *Applications and Theory of Petri Nets 2003: 24th International Conference, ICATPN 2003 Eindhoven, The Netherlands, June 23–27, 2003 Proceedings 24*, pages 483–505. Springer.

Bouquet, F., Gauthier, J.-M., Hammad, A., and Peureux, F. (2012). Transformation of sysml structure diagrams to vhdl-ams. In *2012 Second Workshop on Design, Control and Software Implementation for Distributed MEMS*, pages 74–81. IEEE.

Chatterjee, A. and Reza, H. (2020). Toward modeling and verification of uncertainty in cyber-physical systems. In *2020 IEEE International Conference on Electro Information Technology (EIT)*, pages 568–576. IEEE.

Christen, E. and Bakalar, K. (1999). Vhdl-ams: a hardware description language for analog processing. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 46(10):67.

Dwyer, M. B., Avrunin, G. S., and Corbett, J. C. (1999). Patterns in property specifications for finite-state verification. In *Proceedings of the 21st international conference on Software engineering*, pages 411–420.

Foures, D., Albert, V., and Pascal, J.-C. (2011). Activitydiagram2petrinet: transformation-based model in accordance with the omg sysml specifications. In *Eurosis, The 2011 European Simulation and Modelling Conference*, pages p429–p433.

Friedenthal, S., Griego, R., and Sampson, M. (2007). Incose model based systems engineering (mbse) initiative. In *INCOSE 2007 symposium*, volume 11. sn.

Gauthier, J.-M. (2013). Test generation for rtes from sysml models: Context, motivations and research proposal. In *2013 IEEE Sixth International Conference on Software Testing, Verification and Validation*, pages 503–504. IEEE.

Giorgetti, A., Hammad, A., and Tatibouët, B. (2010). Using sysml for smart surface modeling. In *2010 First Workshop on Hardware and Software Implementation and Control of Distributed MEMS*, pages 100–107. IEEE.

Haase, J., Bastian, J., and Reitz, S. (2003). Vhdl-ams in mems design flow. *System Specification & Design Languages: Best of FDL'02*, pages 51–60.

Harris, D. M. and Harris, S. L. (2022). Hardware description languages. *Digital Design and Computer Architecture*, pages 170–235.

Hillah, L.-M., Kordon, F., Petrucci, L., and Treves, N. (2010). Pnml framework: an extendable reference implementation of the petri net markup language. In *Applications and Theory of Petri Nets: 31st International Conference, PETRI NETS 2010, Braga, Portugal, June 21-25, 2010. Proceedings 31*, pages 318–327. Springer.

Holt, J. and Perry, S. (2008). *SysML for systems engineering*, volume 7. IET.

Holzmann, G. J. (1997). The model checker spin. *IEEE Transactions on software engineering*, 23(5):279–295.

Hossain, M. M., Kashem, M. A., Nayan, N. M., and Chowdhury, M. A. (2024). A medical cyber-physical system for predicting maternal health in developing countries using machine learning. *Healthcare Analytics*, 5:100285.

Huber, P., Jensen, K., and Shapiro, R. M. (1991). Hierarchies in coloured petri nets. In *Advances in Petri Nets 1990 10*, pages 313–341. Springer.

Jensen, K., Kristensen, L. M., and Wells, L. (2007). Coloured petri nets and cpn tools for modelling and validation of concurrent systems. *International Journal on Software Tools for Technology Transfer*, 9:213–254.

Mitsch, S. and Platzer, A. (2016). Modelplex: Verified runtime validation of verified cyber-physical system models. *Formal Methods in System Design*, 49:33–74.

Mohamed, M., Kardas, G., and Challenger, M. (2021a). Model-driven engineering tools and languages for cyber-physical systems–a systematic literature review. *IEEE Access*, pages 1–1.

Mohamed, M. A., Kardas, G., and Challenger, M. (2021b). A systematic literature review on model-driven engineering for cyber-physical systems. *arXiv preprint arXiv:2103.08644*.

OMG, O. (2012). Object constraint language (ocl), version 2.3. 1.

Ordinez, L., Eggly, G., Micheletto, M., and Santos, R. (2020). Using uml for learning how to design and model cyber-physical systems. *IEEE Revista Iberoamericana de Tecnologias del Aprendizaje*, 15(1):50–60.

Otter, M. and Elmqvist, H. (2001). Modelica overview. *URl: https://www. modelica. org/education/educationalmaterial/lecture-material/english/ModelicaOverview. pdf*.

Palnitkar, S. (2003). *Verilog HDL: a guide to digital design and synthesis*, volume 1. Prentice Hall Professional.

Pêcheux, F., Lallement, C., and Vachoux, A. (2005). Vhdl-ams and verilog-ams as alternative hardware description languages for efficient modeling of multidiscipline systems. *IEEE transactions on Computer-Aided design of integrated Circuits and Systems*, 24(2):204–225.

Prevostini, M. and Zamsa, E. (2007). Sysml profile for soc design and systemc transformation. *ALaRI, Faculty of Informatics University of Lugano via G. Buffi*, 13(5).

Rahim, M., Boukala-Ioualalen, M., and Hammad, A. (2014). Petri nets based approach for modular verification of sysml requirements on activity diagrams. In *PNSE@ Petri Nets*, pages 233–248.

Tannoury, P. (2022). An incremental model-based design methodology to develop cps with sysml/ocl/reo. In *Journées du GDR GPL*.

Tannoury, P., Chouali, S., and Hammad, A. (2022). Model driven approach to design an automotive cps with sysreo language. In *Proceedings of the 20th ACM International Symposium on Mobility Management and Wireless Access*, pages 97–104.

Tannoury, P., Chouali, S., and Hammad, A. (2023). Joint use of sysml and reo to specify and verify the compatibility of cps components. In *International Conference on Formal Aspects of Component Software*, pages 84–102. Springer.

Thule, C., Lausdahl, K., Gomes, C., Meisl, G., and Larsen, P. G. (2019). Maestro: the into-cps co-simulation framework. *Simulation Modelling Practice and Theory*, 92:45–61.

Tripathi, D., Singh, L. K., Tripathi, A. K., and Chaturvedi, A. (2021). Model based security verification of cyber-physical system based on petrinet: A case study of nuclear power plant. *Annals of Nuclear Energy*, 159:108306.

Vladimiriovich, M. A., Alexandrovich, V. A., and Olegovich, R. D. (2015). Automatic translation uml activity diagrams to petri net. In *2015 International Siberian Conference on Control and Communications (SIBCON)*, pages 1–4. IEEE.

Wade, J., Cohen, R., Blackburn, M., Hole, E., and Bowen, N. (2015). Systems engineering of cyber-physical systems education program. In *Proceedings of the WESE'15: Workshop on Embedded and Cyber-Physical Systems Education*, pages 1–8.

Walia, G. S., Kumar, D., Sanger, J., and Sahana, S. (2024). Cyber physical system and enabling technologies: Opportunity, challenges, and applications. *Security and Risk Analysis for Intelligent Cloud Computing*, pages 128–143.

Wang, S., Yan, C., and Xiao, F. (2012). Quantitative energy performance assessment methods for existing buildings. *Energy and buildings*, 55:873–888.

Yu, Z., Gao, H., Cong, X., Wu, N., and Song, H. H. (2023). A survey on cyber-physical systems security. *IEEE Internet of Things Journal*.