sciendo RIGA TECHNICAL UNIVERSITY

Applied Computer Systems

ISSN 2255-8691 (online)
ISSN 2255-8683 (print)
May 2019, vol. 24, no. 1, pp. 1–8
https://doi.org/10.2478/acss-2019-0001
https://content.sciendo.com

Brief Overview of Modelling Methods, Life-Cycle and Application Domains of Cyber-Physical Systems

Kristaps Babris[1*], Oksana Nikiforova[2], Uldis Sukovskis[3]
[1–3]*Riga Technical University, Riga, Latvia*

***Abstract* – Cyber-Physical Systems (CPSs) are systems that connect the physical world with the virtual world of information processing. They consist of various components that work together to create some global behaviour. These components include software systems, communication technologies and sensors, executive mechanisms that interact with the real world, often including embedded technologies. One CPS may include a variety of components from different manufacturers or service providers, often without even knowing that their products and services are integrated with others as a result of CPS. This paper systematises information about CPS modelling methods and domains and presents the CPS modelling cycle – from abstraction to architecture and from concept to realisation.**

***Keywords* – Cyber-physical systems, modelling methods.**

I. Introduction

Cyber-Physical System (CPS) is a new software-controlled distributed system class with complex interactions with the physical world. CPS is becoming more important as it offers socio-economic benefits that go beyond the contribution of classic embedded systems. For example, fully automated cars run by a computer, not driven by a man, promise to increase the efficiency of traffic, which outperforms the efficiency of a man-driven car [1].

As a discipline, CPS is a technology discipline focused on engineering with a strong mathematical abstraction. The main technical task is to combine abstractions that have evolved over the centuries by modelling physical processes (differential equations, stochastic processes, etc.) with abstractions that have developed over the years in computer science (algorithms and programs that provide "procedural epistemology").

Quality design CPSs are relatively complicated because engineers need to find a balance between continuous and discrete structures using sophisticated system representations, interacting with physical and digital processes [2]. These representations should include a wide range of behaviours such as software execution, hardware operations, and mechanical dynamics. An additional factor that makes CPS system design complicated is timing setup – calculations, networking, and physical changes must be properly synchronised to enable CPS to work as intended [3].

CPS modelling techniques and their origins can be found in a variety of disciplines, such as engineering, software engineering, and control theory. Unfortunately, in practice, this diversity leads to heterogeneous engineering processes, which are often difficult to combine and customise when planning a system. The reason can be the fact that different parts of the system may depend on each other and their execution must take place in a certain sequence and under certain circumstances.

Despite the fact that CPS systems are a relatively new phenomenon, there are already different studies in this field – both in terms of CPS definitions, their analysis in modelling and others. For example, a source of information that includes an overview of concepts, applications, and challenges in the CPS area [4]. This source examines the origin, terminology, relatively similar concepts and current challenges in the CPS field; the authors present an overview of related literature discussing practical CPS applications and key research areas. As CPS is a very broad field of research, CPS covers a wide range of applications on different scales. The authors conclude that the existing systems have limited attention to the requirements of the CPS systems and that revolutionary planning and modelling approaches are required to achieve the CPS companion goals [5].

Another source associated with CPS security [5] aims to identify, classify and analyse existing research on CPS security to better understand how security is managed by CPS [6]. The authors empirically define a comparison framework for CPS security techniques and classification techniques. From the data collected, the authors conclude that despite the recent emergence of CPS security solutions, scientific interest in this topic has been growing rapidly in recent years in various publications [6].

In article [7], from a computer science point of view it is claimed that today's computing and networking methods are not suitable for CPS modelling, as they do not take due account of the time and concurrency of the physical system and have set promising research directions to better use computing and networking systems in the CPS codec. Recently, researchers have introduced a programming model that reflects the physical concept of time to simulate a distributed real-time embedded system called Programming Temporally Integrated Distributed Embedded Systems (PTIDES). Giotto – programming language for real time CPS, and Ptolemay 2 – the modelling and simulation environment for heterogeneous systems are also a significant work in this research direction. Other model-based design and development approaches are Model-driven Architecture (MDA) and Model-Integration Computing (MIC).

[*] Corresponding author's e-mail: kristaps.babris@edu.rtu.lv

The goal of the paper is to summarise wide information about modelling methods used for CPS, to describe the approach used for CPS modelling and to overview application domains.

The paper is structured as follows. The first section contains short introduction of the paper. The second section introduces modelling methods of CPS. The next section is about model-based development lifecycle. The fourth section contains CPS application domain information and paper is finished with the fifth section of conclusions and future work.

II. Modelling Methods

When designing a CPS, it is important to develop a formal description [8]. Structural modelling approaches that are based on the representation of complex system model graphs are quite widely used. This section uses results from several authors [9].

CPS modelling techniques vary depending on the scientific field from which they are derived. As CPS engineering is located around the boundaries between discrete digital and continuous physical worlds, one of the most important features of modelling techniques is the processing of their possible continuous phenomena such as time and space. At one end of this spectrum, there are classic software engineering models, such as Statecharts and process algebra [10]. They support connectivity and automated verification. However, managing their continuous phenomena is often too limited for the CPS.

At the end of the second spectrum, there are models that include continuity, such as differential equations and difference equations [11], [12] and engineering tools such as Simulink [13]. Although these models are well suited to traditional control settings, it is increasingly difficult to apply such models to complex autonomous systems. For example, it is difficult to analyse behavioural planning in signal flow control models. The purpose of hybrid systems is to harmonise discrete and continuous system dynamics. A common model is a hybrid model. A common model is a hybrid authomata [14], which combines continuous evolution through differential equations with discrete state jumps. Although this area has been successful in symbolic and numerical calculations to analyse hybrid models, these models are very complex, have limited scalability and lack typical modularity mechanisms [15], which make them complex with common software and system engineering techniques.

A. Discrete Modelling Methods

There are several categories of discrete modelling methods. One category of discrete models focuses on describing complex states and their relationships, often represented as data schemas and object models. Popular formalisms of this category include [9]:

1) Alloy [16];
2) TLA+ [17];
3) Object-Z [18];
4) VDM-SL [19];
5) B [20].

In these models, formalism can be seen as a set of declarative constraints applied to an abstract structure or position. Such modelling techniques usually support expansion through refinement and composition, thus allowing for a logical analysis – contradictory testing or the possibility of reverse model generation [9].

Other discrete modelling methods focus on describing processes, where the primary focus is placed on the system state transitions and changes [9]. These modelling methods are based on algorithmic notations and different state machine forms, such as:

1) Process algebra, such as CSP and FSP [9], [10];
2) Transitions systems – statecharts and Promela/Spin [9], [10];
3) PlusCal [9];
4) Petri networks [9];
5) Dynamic logic (based on JML specification) [9], [12];
6) Reactive models [9], [21].

B. Continuous Modelling Methods

In addition to discrete modelling techniques, continuous and hybrid modelling techniques based on, for example, differential equations can be used. These equations are traditionally used to describe various physical processes, such as mechanics, thermodynamics, and electromagnetism [22].

Differential equations describe the system using state variables such as location or temperature, triggering conditions, and their development rules that can judge the states in which systems may be located. Differential equations of partial derivatives can be simplified by lump element models mapped into simple differential equations – a combination of discrete units with shared variables – such an approach is used, for example, in Modelica language [23]. These models allow simulating, performing a theoretical analysis, such as stability and security, and in some cases predicting the future state of the system.

Continuous modelling techniques are often used in control. For example, formalism of signal flows is used to design control and environmental models, allowing for the analysis of simulation of values, such as rise time, overshoot time, setup time etc. Simulink (Matlab) and SCADE can be used to create such models [13], [24]. Signal flow models are expanded with these discrete state descriptions to form prototypes for algorithmic decision-making (for example, by creating modes or reactions). For example, the StateFlow included in Matlab allows combining graphical and table images, including state transition diagrams, flow diagrams, state transition tables, and truth tables to simulate how the system responds to events, time-dependent conditions, and external input signals [25].

C. Hybrid Modelling Methods

Discrete and uninterrupted modelling approaches provide an opportunity to model individual elements, but in complex systems such as CPS, it is necessary to combine both of the above approaches to create one holistic model. Thus, hybrid modelling approaches allow filling the void between discrete jumps (state changes) and discrete evolution (continuous trajectories based on differential equations). These hybrid models are the hybrid automaton [14] and the hybrid program [26]. Typically, the analysis of hybrid systems is based on direct

reach (taking into account the initial set of states, looks at whether the system achieves a safe or unsafe set of states) and reversibility (for which initial states to give preference, avoid or not at all). To calculate the flowpipe, automated tools use different geometric approximations of states, such as rectangular bodies, polygons, and ellipsoids. Another way to analyse hybrid systems is to identify their invariant and to demonstrate how it is done in differential dynamic logic and hybrid programs [11].

An important subset of hybrid machines is Timed Automatic, where constant development is limited to real-time variables. Timed Automat's accessibility problem is crucial, and computational tracking can be performed without any approximate values. These characteristics check the logical security and the ability of the machine to operate using tools such as UPPAAL [27]. Timed automata are also used as a semantic basis for component-based models such as BIP [28] and EAST-ADL [21], which enable design and synchronization analysis at higher abstraction levels.

The advantage of hybrid models is that an engineer can choose which dynamics to display on a continuous basis and which are discreet. Continuous dynamics do not depend on the fixed sampling scheme (for example, in some models). However, the price of this flexibility is the complexity of syntax and semantics, as well as the difficulty of analysing and linking hybrid models with other models.

D. Component Modelling Methods

CPSs are distributed interactive systems that combine computing and physical processes. CPS software development is a complex job that causes many problems. Systems are logical and physically divided; they need to run on different platforms, meet specific execution times and address communication issues.

In order to address these complexity issues, component-based software engineering is deployed by platform-dependent systems, and therefore domain experts are also software developers. Random complexity created by this gap between the problem domain and the deployment domain can be reduced by modelling techniques.

For example, the CPS in the automotive or robotics industry includes many different features, such as trajectory planning, lane correction, battery management, or engine management that require smooth interaction with the environment for sensors and actuators. Compiling all these distinctive features is one of the key challenges in developing such complex systems. Component and Connector (C&C) models are widely used to design and develop CPS to display features and their logical interactions. The advantage of C&C models is that complex functions can be subdivided hierarchically into sub-functions developed and managed by different domain experts [29].

The C&C modelling force in the logical layer includes the ability to describe architectures with components that perform calculations and information flows modelled using connections between their interfaces. The paradigm focuses on software functions and their logical communication. Due to the hierarchical division of components, different stakeholders can build extensive and complex systems by dividing them. Simulink and LabView are excellent examples of C&C languages used both in the academic world and in the industry [29].

Table I presents a comparison of C&C tools and standards in CPS modelling languages [29].

III. Model-Based Development Life Cycle

Model-based development is a powerful CPS development technique that is often used, mainly focusing on individual steps such as simulation, software synthesis, or verification. There is often a lack of reference to a clear methodology for using this method, so this section describes the known CPS development steps based on [30].

The authors of this article propose combining the existing knowledge and dividing ten steps to create an open and holistic methodology for model-based CPS development – from abstraction to architecture and from concept to implementation.

A. Problem Statement

To describe the problem simply, intuitively and generally, it is possible to use a simple language to describe the situation to be solved without mathematics or technical terminology. Developers of large and dangerous critical systems will also need to develop a project plan consisting of requirements management, metrics, formal test process descriptions, and other professional review processes. Given the dependence of cyber-physical systems on different industries, this step is necessary to effectively communicate the design requirements at all levels.

B. Physical Process Modelling

At the first iteration phase of physical processes, it is necessary to create an insight into the observations and environment, in which CPS or physically controlled processes will be located. Physical process models are simplified representations of real systems and are usually in the form of differential equations or Laplace transformations. Simple mathematical models in the next steps can be redefined as the development of control algorithm, hardware specifications, components and sub-systems testing.

C. Characterisation of the Problem

At this stage, the separation of fixed parameters, variable parameters and controlled values takes place. Factors that identify physical processes, such as configuration space, security constraints, input and output sets, saturation points, and modal behaviour are identified. This stage defines how physical processes can interact with calculations, including end-to-end latency requirements, error cases and condition, and noise and quantization. Quantification is an operation that replaces the discrete signal (pulse sequence) with numbers – the results of the countdown. This sequence of numbers creates a digital signal.

TABLE I
COMPARISON OF C&C TOOLS AND STANDARDS
Criteria: 1. Unit Support; 2. Unit Conversion; 3. Component Arrays; 4. Domains; 5. Resolutions; 6. Static Analysis; 7. Configuration Parameters; 8. Generics; 9. Matrix Support; 10. Differential Equation; 11. Atypical Modelling; 12. Operating Environment; + Yes, − No, P – Partly

Language	Unit Support	Unit Conversions	Component Arrays	Domains	Resolutions	Static Analysis	Configuration Parameters	Generics	Matrix Support	Differential Equation	Atypical Modelling	Operating Environment
Simulink	−	−	−	+	−	P	+	−	+	−	−	−
Modelica	+	+	P	+	+	P	+	+	+	+	+	−
SysML	+	+	−	P	P	−	+	+	−	−	−	−
Marte	+	+	−	P	P	−	+	+	+	+	−	−
AutoFocus3	−	−	−	+	−	+	+	−	−	−	−	−
xADL	P	−	−	P	−	P	P	−	−	−	−	−
AutoSAR	+	+	−	+	+	P	+	−	−	−	−	−
LabView	+	+	−	+	+	+	+	−	+	P	−	−
MontiArc	−	−	−	+	+	P	+	+	−	−	−	−
MontiCar	+	+	+	+	+	+	+	+	+	−	−	+
Ptolemy	−	−	+	−	−	+	+	+	−	−	−	−
Verilog	+	P	+	+	+	+	+	+	−	P	+	−
VHDL	+	P	+	+	+	+	+	+	−	P	−	−
Rapide	−	−	−	−	−	+	+	−	−	−	−	−
SADL	−	−	−	−	−	+	+	−	−	−	−	−
SCADE	−	−	−	−	−	+	−	−	−	−	−	−
SystemC	−	−	+	−	−	+	+	+	−	−	−	−
TECS	−	−	−	−	−	+	+	−	−	−	−	−
UniCon	+	−	+	+	+	+	+	−	−	−	−	−
WRIGHT	−	−	+	−	−	+	+	−	−	−	−	−

D. Expression of Control Algorithm

This step determines the conditions under which physical processes will be controlled and obtains an appropriate management algorithm that can be executed by the embedded computer. Based on the previous step, the requirements for latency and retention, sampling rate, jitter and quotation can be specified to such an extent that the physical dynamics of interest can be accurately measured and appropriately controlled. These attributes must meet the requirements of the computing platform used. In highly distributed applications or systems that are globally asynchronous but locally synchronous, it may be particularly important to choose calculation models before retrieving control algorithms. This step needs to be reviewed after the calculation models and hardware are selected, which determines the latency jitter or the variable sampling frequency, which is determined by the asinhronic calculation model, or saturation, or any other hardware-induced non-linear artifacts.

E. Choice of Computing Models

The calculation model is a set of permitted instructions in the calculations, as well as the rules governing the components of interaction, communication and computational control flows [31]. The formal calculation model defines semantics, which often leads to a higher level of analysis and gives the potential to simulate CPS using heterogeneous modelling tools. Models that are defined as formal calculation models can be used to simplify the analysis of execution time, state reach, memory usage and delay [32], [33]. The dynamics of this software changes the development of CPS and can be generalised and used in the MBD workflow. The complexity of many CPSs often requires the composition of multiple calculation models and their interaction. The advantages of using a particular computing model depend on its semantics, timing constructs, and whether it can be calculated as a Turing complete.

F. Determination of the Hardware to be Used

In this step, hardware that can withstand the environment interacts with the simulated physical systems, and the implemented control algorithm is selected. For each component, it is necessary to consider its input and output bandwidth, delays from input to output, power consumption, measurement resolution and frequency, as well as mechanical parameters such as form factor, rejection of electrical interference, durability and service life. Mechanical actuators must be able to create forces and torques that exceed the minimum values obtained from earlier problem characteristics. It can be emphasised that it is important to consider and model the effects of replacing the ideal parts with, for example, cheaper ones, given that the manufacturer's specifications are not always accurate and that the hardware components should be independently tested. The choice of embedded computer may depend on a deeper understanding of the latency and execution time requirements of the control algorithms, the measurement of the worst run time of the synthesised software, and the rationale for the software to interact with a particular hardware architecture. This step may require several iterations with software development and simulation before the embedded computer can be chosen with confidence.

G. Simulation

In this step, the problem is solved using simulation software. If multiple computing models are used, then this simulation software should support composition and interaction between these multiple computing models. Depending on the robustness

of the development environment, models of sensors, actuators and physical processes can be included. Platform-based design to separate application logic and architecture-specific software modules or modular components can improve code portability, reduce hardware component change effects, and allow components to be reused in other contexts [34].

Models of individual components and subsystems are as important as the end-to-end model. Component models enable the creation of automated testing frameworks for the verification and testing of synthesised software. If no modelling tool can fully describe the system, a modelling tool that best reflects its dynamics can be used for each subsystem. Although unmatched simulations cannot represent the relationship between the signals crossing the boundaries of the subsystem or the behaviour of the composition of these subsystems, this step facilitates the harmonisation of physical modelling, simulation and testing. For example, the Ptolemy 2 modelling tool [31] developed at the University of California, Berkeley, is a versatile tool for studying heterogeneity. This allows developers to easily create new computing models and simulate their behaviour.

H. Design

In this step, the design of the device can be started according to the specifications, taking into account where exceptions are made, which may affect the earlier modelling (in this case, it is necessary to return to the previous steps). The design must be designed to allow for testing of individual components and sub-systems against theoretical models, facilitating the harmonisation of simulation and testing.

I. Software Synthesis

This step uses code synthesizers that can work with software simulation environments, for example, LabVIEW and Ptolemay mentioned above 2. Software of this type can provide direct support to an embedded computer or a generic code that can be manually adapted to the architecture specific code. Assuming that the code synthesizer creates a code that truly executes the semantics of the calculation models, it can be assumed that the logic of the synthesized code is correct by structure. However, the timing should be checked because code generators and compilers can implement software timing artifacts, and hardware features such as pipelines and caches that can trigger trembling errors. Other restrictions, such as Memory Footprint or Processor Usage, may also require independent verification. Time and other constraints should be checked against existing models.

J. Verification, Validation and Testing

In this step, changeable parameters are configured to create as simple a testing environment as possible and to check each component and sub-system independently. Computing systems can be isolated from physical systems using hardware-in-the-loop testing where programmable hardware, such as embedded computers or FPGA (Field-programmable Gate Array), simulates feedback from physical or other computing processes. Performance time and latency measurements can be used to improve previous models, and unexpected test results may indicate errors in modelling or deployment. Formal verification and validation provide insight into the behaviour of the algorithm for all or certain combinations of its inputs or throughout the cycle. Accurately set requirements transform them into a physical verification and validation specification. List Invariant – constant values to be tested during testing. Verification and validation are probably the most complex aspects of CPS development.

IV. Application Domains

Examples of CPS include Smart Grid, Autonomous Car Systems, Medical Monitoring Equipment, Process Process Control Systems, Robotic Equipment, Autonomous Avionics Systems, and many other industries [35]. CPS uses transdisciplinary approaches combining cybernetics, mechatronics, design, and science. Process management is often referred to as embedded systems [36]. In these systems, the greatest emphasis is placed on computing elements and less on the intensive link between computing and physical elements. In this context, CPS is more like the Internet of Things (IoT) because it uses the same basic architecture. However, CPS provides a higher level of combination and coordination between computing and physical elements [37]. Thus, the distinctive features of CPS are:

1) Digital integration;
2) Digital communication sharing and distribution;
3) Parallel processes;
4) Upward, downward and regulatory process flows;
5) Managing synchronous and asynchronous processes, etc.

Typical CPSs are found in manufacturing, automotive, smart energy, avionics and distributed robots. Modern software systems are becoming more widespread, autonomous and incorporated into the physical world. Such systems are important in science and technology because they offer socio-economic benefits in addition to classical embedded systems. For example, self-driving cars promise a dramatic reduction in the number of accidents.

The most popular CPS application domains can be listed [38]:

1) Healthcare (National Health Information Network, Electronic Patient Record Initiative, Remote Monitoring, Home Care, Operating Room, etc.);
2) Aviation (Transportation Systems, Aircraft Systems);
3) Smart Grid (Smart-Grid Metering and Control Systems);
4) Emergency Management Systems (e.g., Resource Tracking, Personnel Management, Developing and Implementing Response Contingency Plans);
5) Other Domains (e.g., Distributed Physical Games, Traffic Control and Safety, Financial Networks and Systems, Assisted Living, Advanced Automotive Systems, Environmental Control, Distributed Robotics, Military Systems, Smart Structures, etc.).

A. CPS and Embedded Systems

An embedded system is an independent system that includes elements of control logic and real world interaction. Unlike

CPS, an embedded system is usually just one device, and CPS can include many systems and devices [39].

Embedded systems typically have a limited number of tasks to complete with software and hardware components specifically designed to accomplish these tasks, usually with very limited resources [39].

In contrast, while embedded systems are very important elements for many CPS, CPS itself operates on a much larger scale, possibly including many embedded systems or other devices and systems, including human and social systems [39]. If an organisation works with embedded systems, especially connected or network-based systems, this project or product may be classified as CPS [39].

B. CPS and Internet of Things

The Internet of Things (IoT) overlapped significantly. The IoT's vision for the future is millions of devices connected to the Internet, allowing them to remotely collect information about the real world and share it with other systems and devices. IoT and CPS have many common problems, but there are also some significant differences [39].

IoT places great emphasis on uniquely identifiable and Internet-related devices and embedded systems, while CPS engineering science plays an important role in the relationship between computing and the physical world (for example, between complex software and hardware aspects of the system) [39].

For example, if an organisation is working with IoT, especially if it involves interaction with the physical world using sensors and/or actuators, then this project or product can be classified as CPS [39].

C. CPS and Systems of Systems

System of Systems (SoS) is a system consisting of components that are independent systems in themselves [40]. These systems are leading and functionally independent, physically divided, constantly evolving and collaborating to create new global behaviours that they cannot produce separately [39]. Or it is the integration of a limited number of independent and functioning systems, which are linked together for a fixed period of time to achieve a certain higher target [41]. SoS is a relatively new concept and defines the five key features [41], [42]:

1) Independence of the components of the general system;
2) Independence of control management of the component system;
3) Geographical distribution;
4) Emerging behaviour;
5) Evolutionary development processes.

CPS and SoS have a lot in common – many CPSs are made up of independent components and, like SoS, CPS also addresses the problems of addiction, development and distribution [39].

Although CPS component systems are often independent, it is not the key to describing CPS. Similarly, with SoS, although SoS includes computing elements as a real world interaction, it is not the main characteristic of SoS [39].

Conversely, if an organisation or project works with SoS, especially if it involves interaction with the real world, using sensors and/or actuators, we can talk about classification as CPS [39].

V. Conclusion and Future Work

Modelling and simulation are essential to improve CPS design, performance and continuous development. CPS design is increasingly based on models, but reliable and accurate system models often require a significant investment. Modelling challenges include high costs associated with model creation and maintenance, as well as difficulties associated with model reuse, modelling, simulation and stochastic behaviour analysis, towing tools of various strengths without the need for re-modelling, detailed and abstract models, and effort necessary to create models that include failure states and response to unusual situations for validation and verification purposes. Different stages and types of models need to be provided for the various stages of the design process as well as the support for defect detection and operation. Updating and consistency of these models (model management) is an important issue, as well as modelling effort and cost reduction by reusing the model (object or module modelling) and pre-defined and customisable standard models. Changes in suppliers' technical or software elements as well as changes in the business model may lead to situations where simulation models for potential system components are delivered in such a way that the whole system can be designed for these models.

One of the solution for such kind of composition of CPS is offered in [43], where a two-hemisphere model is used to integrate all components together. The system in that case has one conceptual model of general system structure and different scenarios for behaviour of different system components.

The simulation challenges are large-scale heterogeneous system simulation, efficient hybrid (continuous discrete) simulation, system simulation with many different time scales, and system physical part and management strategy integrated modelling performance analysis, including emergency situations. Effective modelling of complex systems with reconfiguration and behaviour change requires dynamic modelling by modelling model, including transition between different levels of detail according to required precision and operating conditions, as well as high performance numerical algorithms.

Model-based CPS modelling requires an elementary collaborative environment and integration of the legacy system simulation [44], as well as open and efficient integration and consolidation of data, models, engineering tools and other information across different platforms. An important topic for further research is the behavioural modelling of users or operators of CPS or CPS components and the formation of structures among users.

References

[1] P. Gao, R. Hensley, and A. Zielke, "A road map to the future for the auto industry," *McKinsey Quarterly*, October 2014.

[2] E. A. Lee, "CPS Foundations," *Proceedings of the 47th Design Automation Conference*, DAC '10, New York, pp. 737–742, 2010.
[3] E. A. Lee, "Cyber Physical Systems: Design Challenges," *In Proceedings of the 11th Symposium on Object Oriented Real-Time Distributed Computing*, IEEE Computer Society, Washington, pp. 363–369, 2008.
[4] V. Gunes, S. Peter, T. Givargis, and F. Vahid, "A Survey on Concepts, Applications, and Challenges in Cyber-Physical Systems," *KSII Transactions on Internet and Information Systems*, vol. 8, no. 12, pp. 4242–4268, 2014. https://doi.org/10.3837/tiis.2014.12.001
[5] Y. Z. Lun, A. D'Innocenzo, I. Malavolta, and M. D. Di Benedetto, "Cyber-Physical Systems Security: a Systematic Mapping Study. pp. 1–32, 2016. Available from: https://www.researchgate.net/publication/303698739_Cyber-Physical_Systems_Security_a_Systematic_Mapping_Study
[6] A. Barišić, STSM Report: Systematic literature review on multi-paradigm modeling for CPS Systems, Visiting University of Belgrade, Faculty of Organizational Sciences, Belgrade (RS), The STSM report, MPMCPS 2018.
[7] K-D. Kim, and P. R. Kumar, "An Overview and Some Challenges in Cyber-Physical Systems," *Journal of the Indian Institute of Science*, vol. 93, no. 3, pp. 341–352, 2013.
[8] V. Ya. Tsvetkov, "Information Constructions," *European Journal of Technology and Design*, vol. 5, no. 3, pp. 147–152, 2014. https://doi.org/10.13187/ejtd.2014.5.147
[9] I. Ruchkin, "Integration of Modeling Methods for Cyber-Physical Systems," PhD thesis, Institute for Software Research School of Computer Science, Carnegie Mellon University, Pittsburgh, 2018.
[10] J. Magee and J. Kramer, *Concurrency: State Models & Java Programs*, Wiley, 1999.
[11] A. Platzer, *Logical foundations of cyber-physical systems*. Springer, Berlin Heidelberg, New York, NY, 2018.
[12] A. Platzer, "Differential Dynamic Logic for Hybrid Systems," *Journal of Automated Reasoning*, vol. 41, no. 2, pp. 143–189, 2008. https://doi.org/10.1007/s10817-008-9103-8
[13] J. Dabney and T. L. Harman, *Mastering SIMULINK 2*. Prentice Hall, Upper Saddle River, NJ, 1998.
[14] R. Alur, T. A. Henzinger, H. Wong-toi, "Symbolic Analysis of Hybrid Systems," *In Proceedings of the IEEE CDC*, 1997.
[15] I. Ruchkin, B. Schmerl, and D. Garlan, "Architectural Abstractions for Hybrid Programs," in *Proceedings of the 18th International ACM SIGSOFT Symposium on Component-Based Software Engineering*, CBSE '15, Montréal, QC, Canada, pp. 65–74, 2015. https://doi.org/10.1145/2737166.2737167
[16] D. Jackson, *Software abstractions: logic, language, and analysis*. MIT Press, Cambridge, Mass., 2012.
[17] L. Lamport, *Specifying Systems: The TLA+ Language and Tools for Hardware and Software Engineers*. Addison-Wesley Professional, Boston, 1st edition edition, July 2002.
[18] G. Smith, Introduction. In: *The Object-Z Specification Language. Advances in Formal Methods*, vol 1. Springer, Boston, MA, 2000. https://doi.org/10.1007/978-1-4615-5265-9_1
[19] P. G. Larsen, K. Lausdahl, N. Battle, J. Fitzgerald, S. Wolff, S. Sahara, M. Verhoef, P. W. V. Tran-Jorgensen, T. Oda, and P. Chisholm, "VDM-10 Language Manual. Overture Technical Report Series TR-001", pp. 39–234, 2018. Available from: www.overturetool.org.
[20] K. Lano, *The B Language and Method: A Guide to Practical Formal Development (Formal Approaches to Computing and Information Technology (FACIT))*. Springer-Verlag, London, 1996.
[21] R. Marinescu, Model-driven Analysis and Verification of Automotive Embedded Systems, PhD thesis, Maladaren University, 2016.
[22] E. Hairer, S. P. Nørsett, and G. Wanner, *Solving Ordinary Differential Equations I Nonstiff Problems*, Springer Series in Computational Mathematics, 2nd edition, 1993.
[23] P. Fritzson, *Principles of Object-Oriented Modeling and Simulation with Modelica 2.1*. John Wiley-IEEE Press, 2010.
[24] G. Walde, and R. Luckner, "Bridging the tool gap for model-based design from flight control function design in Simulink to software design in SCADE", *IEEE/AIAA 35th Digital Avionics Systems Conference (DASC)*, 2016. https://doi.org/10.1109/DASC.2016.7778044
[25] M. Wermelinger, and T. Margaria-Steffen, "Fundamental Approaches to Software Engineering", *Proceedings 7th International Conference, FASE 2004. Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS*, Barcelona, Spain, 2004. https://doi.org/10.1007/b95935
[26] A. Benveniste, T. Bourke, B. Caillaud, J. Colao, C. Pasteur, and M. Pouzet, "Building a Hybrid Systems Modeler on Synchronous Languages Principles," *Proceedings of the IEEE*, vol. 106, no. 9, 2018. https://doi.org/10.1109/JPROC.2018.2858016
[27] K. G. Larsen, P. Pettersson, and W. Yi, "Model-Checking for Real-Time Systems," In H. Reichel (eds). *Fundamentals of Computation Theory, Lecture Notes in Computer Science*, vol. 965, pp. 62–88, 1995. https://doi.org/10.1007/3-540-60249-6_41
[28] A. Basu, M. Bozga, and J. Sifakis, "Modeling Heterogeneous Real-time Components in BIP", *Proceedings of the Fourth IEEE International Conference on Software Engineering and Formal Methods*, Washington, DC, USA, 2006.
[29] E. Kusmenko, A. Roth, B. Rumpe, and M. von Wenckstern, "Modeling Architectures of Cyber-Physical Systems," in A. Anjorin, H. Espinoza (Eds) *Modelling Foundations and Applications. ECMFA 2017. Lecture Notes in Computer Science*, vol 10376, pp. 34–50, 2017. https://doi.org/10.1007/978-3-319-61482-3_3
[30] J. C. Jensen, D. H. Chang, and E. A. Lee, "A Model-Based Design Methodology for Cyber-Physical Systems," *7th International Wireless Communications and Mobile Computing Conference*, Istanbul, Turkey, 2011. https://doi.org/10.1109/IWCMC.2011.5982785
[31] J. Eker, J. Janneck, E. A. Lee, J. Liu, X. Liu, J. Ludvig, S. Neuendorffer, S. Sachs, and Y. Xiong, "Timing Heterogeneity - The Ptolemy Approach," *Proceedings of the IEEE*, vol. 91, no. 1. 2003. https://doi.org/10.1109/JPROC.2002.805829
[32] E. A. Lee, "Computing needs time, ACM Communications," vol. 52, no. 5, pp. 70–79, May 2009. https://doi.org/10.1145/1506409.1506426
[33] J. Eidson, E. A. Lee, S. Matic, S. A. Seshia, and J. Zou, "Time-centric Models for Designing Embedded Cyber-Physical Systems," California Univ. Berkeley Dept. of Electrical Engineering and Computer Science, Technical report. UCB/EECS-2009-135, October 2009. https://doi.org/10.21236/ADA538747
[34] K. Keutzer, A. R. Newton, J. Rabaey, and A. Sangiovanni-Vincentelli, "System-Level Design: Orthogonalization of Concerns and Platform-Based Design," *IEEE Transactions*, vol. 19, no. 12. December 2000. https://doi.org/10.1109/43.898830
[35] Khaitan et al., "Design Techniques and Applications of Cyber Physical Systems: A Survey," *IEEE Systems Journal*, 2014.
[36] E. A. Lee, and S. A. Seshia, "Introduction to Embedded Systems - A Cyber-Physical Systems Approach," LeeSeshia.org, 2011.
[37] C-R. Rad, O. Hancu, I-A. Takacs, G. Olteanu, "Smart Monitoring of Potato Crop: A Cyber-Physical System Architecture Model in the Field of Precision Agriculture," Agriculture *and Agricultural Science Procedia*, vol. 6, pp. 73 – 79, 2015. https://doi.org/10.1016/j.aaspro.2015.08.041
[38] M. N. Al-Mhiqani, R. Ahmad, K. H. Abdulkareem, and N. S. Ali, "Investigation study of Cyber-Physical Systems: Characteristics, application domains, and security challenges," *Journal of Engineering and Applied Sciences*, vol. 12, no. 22, 2017.
[39] CPSE Labs, 2018. [Online] Available from: http://www.cpse-labs.eu/cps.php [Accessed: 27th October 2018].
[40] J. O. Clark, "System of Systems Engineering and Family of Systems Engineering from a standards, V-Model, and Dual-V Model perspective," *3rd Annual IEEE Systems Conference*, IEEE, 2009. https://doi.org/10.1109/SYSTEMS.2009.4815831
[41] S. Engell, "Cyber-physical Systems of Systems – Definition and core research and innovation areas," Working Paper of the Support Action CPSoS, pp.1–11, 2014.
[42] M.W. Maier, "Architecting Principles for System of Systems," *Systems Engineering*, vol. 1, no. 4, 1998. https://doi.org/10.1002/(SICI)1520-6858(1998)1:4<267::AID-SYS3>3.0.CO;2-D
O. Nikiforova, N. El Marzouki, K. Gusarovs, H. Vangheluwe, T. Bures, R. Al-Ali, M. Iacono, P. O. Esquivel, and F. Leon "The Two-Hemisphere Modelling Approach to the Composition of Cyber-Physical Systems" Proceedings *of International Conference on Software Technologies (ICSOFT 2017)*, 24–26 July, 2017, Madrid, Spain. SCITEPRESS Digital Library, pp. 286–293. https://doi.org/10.5220/0006424902860293
[43] P. Derler, E. A. Lee, S. Tripakis, and M. Torngren, "Cyber-physical System Design Contracts," in *Proceedings of the ACM/IEEE 4th International Conference on Cyber-Physical Systems, ICCPS '13*, pp. 109–118, 2013. https://doi.org/10.1145/2502524.2502540

Kristaps Babris received the Master degree in Computer Systems from Riga Technical University, Latvia, in 2018. He is at present the first-year Ph. D. student at the Department of Applied Computer Science, Riga Technical University.
In parallel, he is a CTO at company that develops Business Intelligence solutions.
His current research interests include design and modelling.
E-mail: kristaps.babris@edu.rtu.lv

Oksana Nikiforova received the Doctoral degree in Information Technologies (system analysis, modelling and design) from Riga Technical University, Latvia, in 2001.
At present, she is a Professor at the Department of Applied Computer Science, Riga Technical University, where she has been working since 1997. Her current research interests include object-oriented system analysis, design and modelling, especially the issues in model driven software development.
E-mail: oksana.nikiforova@rtu.lv
ORCID iD: https://orcid.org/0000-0001-7983-3088

Uldis Sukovskis. Professor Uldis Sukovskis holds doctor degree in information technology since 1992, he is corresponding member of Latvian Academy of Science since 2008. His scientific research interests are model-driven system development, IT security and improvement of educational systems.
He is Vice-Rector for Academic Affairs and Head of the Department of Applied Computer Science at Riga Technical University, Latvia. He has work experience in company Exigen Services Latvia in the position of Director for IT Consulting and Audit.
Professor Sukovskis is member of Information Systems Audit and Control Association (ISACA) and Certified Information Systems Auditor (CISA).
E-mail: uldis.sukovskis@rtu.lv
ORCID iD: http://orcid.org/0000-0002-7960-1049