INTO-CPS: An integrated “tool chain” for comprehensive Model-Based Design of Cyber-Physical Systems

Alessandra Bagnato, Étienne Brosse, Imran Quadri, and Andrey Sadovykh

1. Introduction

Cyber-Physical Systems (CPSs) can be considered as the next general of Embedded Systems. In recent years, the growth of connected CPSs and Internet of Things (IoT) devices has increased tremendously due to the availability of high-capacity networks (3G and 4G/LTE networks), advanced sensors (e.g. RFID, NFC, etc.), protocols (e.g. IPv6, MQTT, etc.), mobile Internet and wearable devices. This paradigm shift will accelerate in coming years to drive the next technological revolution for CPSs, where a plethora of light-weight interconnected devices will be able to interact, communicate and share vast amounts of data.

CPSs and especially Model-based CPSs methodologies as an emerging area of increasing relevance require a comprehensive framework for their validation and certification. This includes both, the validation and certification of the

embedded devices (sensors and actuators), as well as of the optional Cloud-based services which can take over the computation of critical aspects of the CPSs operation. This paper aims to provide an overview of the current research activities regarding Model-based methodologies for CPSs and the related challenges involved in developing these complex systems, and has the objective to determine the current and future directions related to CPSs. The paper will also serve as a foundation for future extensions of the SysML standard to support CPSs modeling and to the clear semantics on SysML usage that will be carried out within the INTO-CPS research project.

2. Cyber-Physical Systems

The wide number of application areas of CPSs demand design technologies able to cover various industrial domains like automotive, industrial control, medical, mobile communication, etc. Each domain has different point of views on the underlying technical and physical details. By observing the different challenges that are inherited in the design of CPSs, it is evident that CPSs need improved multi-disciplinary modeling and specification methodologies able to support static analysis, verification, simulation, and performance analysis and implementation technologies [1].

To address these issues, domain specific languages have been developed to cover the design challenges of specific design domains. For example, a famous example can be found in the automotive domain: AUTOSAR (AUTomotive Open System ARchitecture) is the de facto standard for automotive software and E/E (Electrics/Electronics) architectures [2]. It provides a basic infrastructure to assist with developing vehicular software using Atomic Software Components, running on a standardized middleware layer, called Run-Time Environment (RTE).
Furthermore, it includes the standardization of basic systems functions, enables scalability to different vehicles and platform variants and upgrades over the vehicle’s lifetime. Various commercial AUTOSAR system development tools are available on the market: dSpace SystemDesk, ETAS ISOLAR-A, KPIT K-SAR, Mentor Graphics Volcano Vehicle Systems Architect, and Vector Informatik PRE-EVision [3] and DaVinci Developer. Unfortunately, these very automotive domain specific solutions do not support any other application domains yet. Standards such ANSI/ISA-5.1-1984 have been used to specify CPSs by making use of process models to describe measuring and control devices. While these process models are able to describe different properties of the physical environment, they cannot adequately cover the computational architecture details.

### 2.1 Modeling of cyber-physical systems

Model-Based Design (MBD) has been identified as a powerful design technique for CPSs [4]. In MBD, models are at heart of the design process. Specifications of system and its underlying components are defined in the form of models able to reflect the evolution of the system design. These models can be used for early design analysis; can help in separation of concerns, traceability, trace generation, impact analysis, formal verification, simulation and synthesis. By making use of models, it is possible to have earlier identification of design defects instead of during the prototyping phase at a much higher cost. Additionally, automated or semi-automated processes can also help to synthesize implementations from models, such as automatic code generation and software synthesis on heterogeneous platforms [5]. However, the intrinsic heterogeneity and complexity of CPSs stresses all existing modeling languages and frameworks, and, currently, it is not possible for a single modeling language or tool to adequately address all challenges related to CPSs.

For CPSs modeling, a large number of modeling languages have been utilized to address the underlying aspects such as physical processes and requirements management. A good survey has been made covering languages and tools like Stateflow/Simulink, Modelica, Checkmate and Massaccio; by the Columbus project [6] in order to define an interchange format for CPSs. These languages enable CPSs modeling for design phases such as simulation and verification. In [7], the authors introduce a test bed for collaborative control and information acquisition for maritime applications. An abstraction language in the form of a Domain Specific Language (DSL) for implementation of mission-level controllers has been developed, termed as the Collaborative Sensing Language (CSL).

Recently, high level languages such as UML [8], SysML [9] and MARTE [10] have also been utilized for modeling these complex distributed systems. However, as stated before, none of these languages can singly address all the challenges related to CPSs modeling. While UML, traditionally used for modeling of software systems, defines the syntax of model diagrams, it does not offer any underlying semantics. The OMG SysML standard does offer aspects such as requirements management which can be interesting for CPSs, but suffers from having many semantic alternatives, which are usually provided by tool vendors [11] and does not provide manners to define characteristics of real-time embedded systems such as non-functional constraints, and aspects like performance and energy consumption. MARTE, which is the recent OMG standard for real-time embedded systems does enable designers to define non-functional constraints, but suffers from the same pitfall of not having detailed guidelines and semantics, which can be used by system designers. In absence of concrete MARTE usage guidelines, designers can be plagued with the problem of correct utilization of the profile concepts.

Additionally, high level languages have also been used for modeling aspects of Cloud computing, software tests or services resulting in development of CloudML [12], UTP [13] and SoaML respectively [14]. UTP can be efficiently applied to foster early testing [15] and to establish test automation by generation of executable test scripts [16]. UTP has been applied to various industrial and research case studies to increase automation in test execution and test design in various domains, such as telecommunications, enterprise services choreographies and eHealth [17].

In short, many modeling languages have been used to describe CPSs or aspects which can be used in CPSs development. However, modeling techniques that address only the software aspects are not able to accurately specify CPSs. The complexity of CPSs design demands the usage of new system models/analytical tools, and software simulation tools, along with modeling languages and appropriate learning mechanisms [18, 19] that are able to take into account aspects related to the physical processes of CPSs. In the modeling process, systems are usually considered as static entities where current or general system characteristics are used to emulate system’s behavior.

Thus for the modeling of CPSs, effective semantics are still needed able to integrate any language to reap the benefit offered by MBD. Extending both SysML for CPSs modeling and MARTE for the underlying embedded devices, while integrating both modeling languages under a common, homogeneous framework able to support a holistic modeling of complex heterogeneous CPSs, is still an open problem that needs to be addressed.

### 2.2 Simulating cyber-physical systems and cyber-physical systems of systems

Co-Simulation (Co-operative Simulation) is a simulation method that permits simulating individual components using different simulation tools simultaneously and collaboratively.

Individual simulation tools exchange information such as system variables and their values, time steps for synchroni-
zation, and control signals for orchestrating the co-operative simulation. Thus, engineers can use different effective simulation tools together to create virtual prototypes of entire CPSs. In practice, however, significant challenges remain with regard to the syntax and semantics of model and system integration.

Recent effort by the MODELISAR ITEA2 project that developed a tool independent standard called the Functional Mock-up Interface (FMI) [FMI] [74] has gained significant influence, more prominently in the automotive industry. Many vendors have agreed to use FMI and now provide the facility of exporting simulations as reusable shared components. The FMI standard provides a well-defined specification and API to integrate simulation components. All simulation tools participating in the FMI co-simulation follow the defined standard and provide standardized access to model equations. This permits coupling of Continuous-Time, Discrete-Time, and Discrete-Event that are part of a CPS. Another key element for co-simulation via FMI is the Master Algorithm (MA) that orchestrates the steps of the co-simulation: (1) control the data exchange and (2) control time advancement among individual simulations according to the requirements of the integrated simulation of the overall CPS.

The foundations of FMI-based co-simulation for simple models are well established [27]. Simulation tool vendors are rapidly integrating FMI export and/or import functionality as an answer to the growing demand for flexible multi-domain solutions. However, integrative solutions often suffer from restrictions and tool-specific workarounds because the tools were not designed as dedicated co-simulation frameworks.

Particular attention needs to be given to FMI’s co-simulation variant due to its optimization potential in multi-domain applications (domain specific solver and integration settings, optimized concurrency) [28]. Since the FMI standard does not describe or limit the implementation of the MAs, it leaves out the two fundamental challenges of data exchange and time management. Solution for integrated data and time management in distributed simulations is technically complex and errors can easily lead to performance bottleneck and failures. This complexity pushes designers to adhere to the simplest solutions – losing much of the potential advantages of co-simulation.

3. Related projects carrying out CPSS research

We now look at some of the research related to CPSs in recent years.

The CHESS project [20] focuses on improving Model-Based Design (MBD) practices and technologies to better address issues such as safety, reliability, performance, robustness and other extra-functional concerns for real-time and dependable embedded systems. The project addresses the challenges related to compositional structure, interactions and behavior of system components while guaranteeing their correctness and the level of service at run time. The tool set developed in the project supports verification of extra-functional properties of different system components. The project also proposes a multi-concern component modeling language and editor to fits multiple industrial domains. The language proposed in the project extends UML, SysML and MARTE languages.

In the CONCERTO project [22], a UML/MARTE methodology for distributed, mixed-critical embedded systems has been proposed. This modeling effort is focused on extending the standards to integrate aspects related to distributed networks and mixed-criticality systems, which are not fully addressed in the standards. While CPSs modeling is not undertaken in the project, the outputs of this project can serve as a foundation for new research activities related to CPSs oriented Model-Based methodologies in the near future.

The DESTECS project [24] proposes a methodology for enabling correct-by-construct component assembly for multicore systems. The automatic generation of virtual prototypes has been made possible, along with introduction of support for separation of concerns using a meta-model based approach. New run-time monitoring mechanisms have been developed to analyse extra-functional properties such as energy consumption. Finally, the project enables iterative development by enabling back propagation from platform-specific to platform-independent models.

The COMPASS project provides tools and techniques to support a model-based approach to developing systems of CPSs, also called Systems of Systems (SoSs) by introducing the COMPASS Modeling Language (CML) [23]. They extend SysML by the addition of formal CML notations. COMPASS augments CPSs modeling by means of additional tools and techniques to enable informal SoS development to be undertaken under the guidance of CML analysis techniques, some of which can be presented at the SysML level.

The DESTRECS project [24] proposes a methodology for defining co-models allowing discrete event (DE) and continuous time (CT) models to be co-simulated. The DE and CT models are linked through a common interface specification that identifies shared (monitored/controlled) variables, design parameters and events. While the project supports co-simulation, verification is not supported.

In the Scyphys project [25], research is being carried out to utilize an architectural approach for the design and analysis of CPSs. The project uses a software architectural model as a skeletal frame in order to associate different views of the architectural model, which represents different physical/software models of the system. The research aims to provide architectural views and plug-ins into a common framework that can be easily integrated with existing models and tools, such as Simulink and Modelica, etc.
4. NEW ADVANCES IN INTO-CPSS

Many of the challenges mentioned in the paper are deemed to be addressed in the upcoming HORIZON 2020 INTO-CPS project, INTO-CPS (Integrated Tool Chain for Model-based Design of CPSs, http://into-cps.au.dk/).

The aim of the INTO-CPS project is to create an integrated tool chain for comprehensive model-based design of CPSs. The tool chain will support the multidisciplinary, collaborative modeling of CPSs from requirements, down to actual implementation both in hardware and software.

INTO-CPS is a currently running project will support the Model-Based Design, enabling modeling of CPSs that will permit building and analysis of high level system models that would otherwise not be possible using standalone currently available tools. The solution will be advancing the current State-of-the-Art and will revolve centrally around FMI-compatible co-simulation. In this context, in order to support diagrammatic multi-modeling, we will identify and extend a subset of SysML to produce a profile for FMI. The continuous aspects can be modeled in SysML by means of blocks, parametric diagrams, ports and flows. However, the expression language used in SysML parametric diagrams must be fixed prior to formalization.

Once the SysML profile for FMI co-modeling will be identified and extended, a semantic mapping to FMI will be defined, and this will require formal foundations that, for instance, allow the primitive types of SysML to be mapped to suitable types in FMI, and handles mismatches between concepts like time and synchronization. This will result in model transformations to automatically map SysML diagrams to FMI contracts.

The INTO-CPS project tool chain will provide powerful analysis techniques for CPSs, including generation and static checking of FMI interfaces; model checking; Hardware-in-the-Loop (HiL) and Software-in-the-Loop (SiL) simulation, supported by automatic code generation. The tool chain will allow for both Test Automation (TA) and Design Space Exploration (DSE) of CPSs. The INTO-CPS technologies will be accompanied by a comprehensive set of method guidelines in order to describe how to adopt the INTO-CPS approach, lowering entry barriers for CPSs development. The tool chain will be tested with four case studies in railways, agriculture, building and automotive domains.

5. CONCLUSION

The paper presents the current State-of-the-Art related to CPSs and provides an overview of some of the challenges involved in the design and development of CPSs that will be tackled by INTO-CPS. The paper is also to be used as a basis for the future research activities carried out in the INTO-CPS project where we would aim to extending the SysML standard to support CPS modeling and providing clear semantics on SysML usage.

ACKNOWLEDGMENT

This research presented in this paper is funded by the European Community’s H2020 framework under grant agreement no 644047 (INTO-CPS).

REFERENCES


Biographies

Alessandra Bagnato, Étienne Brosse, and Andrey Sadovykh, see page 19

Imran Rafiq Quadri. Consultant senior R&D pour les projets de l’UE à Softeam, France, il possède plus de neuf années d’expérience, dont plus de six ans d’expérience en tant que consultant et gestionnaire de projets, dans le contexte du programme FP7/ ARTEMIS de l’Union Européenne, liés aux systèmes embarqués et à l’ingénierie dirigée par les modèles. Depuis 2010, il travaille à Softeam dans plusieurs projets de l’UE tels que MADES, ENOSYS et PRESTO. Son expertise et ses centres d’intérêt portent sur le temps réel, les systèmes embarqués, l’ingénierie de systèmes, les technologies et normes relatives aux modèles, telles qu’UML et les profils connexes (SysML, MARTE) ainsi que les plates-formes d’exécution telles que les FPGA reconfigurables dynamiquement basés sur des puces.

Systems engineering principles for software engineers
Ray J. Madachy, Chapman & Hall, 2016, 300 pages

In a world dominated by complex software-intensive systems, it is important for software engineers to take on broader and more informed roles. This book addresses the increasing importance of systems engineering in professional software engineering education and practice. Complex systems bring many disciplines together so software engineers should understand the larger system context and trade space where their critical software functions. The book’s holistic and interdisciplinary approach helps educate software engineers with proven quantitative and qualitative systems engineering principles. It shows how to use systems engineering methods based on the technical fundamentals of probability/statistics, decision analysis, modeling and simulation, quantitative methods, and heuristic approaches as well as non-technical considerations of customers and other stakeholders and project and organizational management.