2017 International Conference on Engineering of Complex Computer Systems

Multi-modeling Approach to Performance Engineering of Cyber-Physical Systems Design

Lorenzo Pagliari
Gran Sasso Science Institute
L'Aquila, Italy
Email: lorenzo.pagliari@gssi.it

Raffaela Mirandola
Politecnico di Milano
Milano, Italy
Email: raffaela.mirandola@polimi.it

Catia Trubiani
Gran Sasso Science Institute
L'Aquila, Italy
Email: catia.trubiani@gssi.it

***Abstract*—The modeling and analysis of Cyber Physical Systems (CPS) is inevitably challenging due to the intrinsic problem of merging the specification of different ensembles that indicate hardware, software and physical aspects of such systems. This intrinsic complexity is exacerbated in performance engineering since multiple models need to co-exist in order to get meaningful performance indicators. In this paper we introduce a guided process called IMPACt (multI Modelling PerformAnce Cps),which helps architects during the design phase, to better understand the behavior of the system under development and the design choices they made, through performance analysis on results obtained by runnable models derived from system high-level specification.**

***Keywords*-Cyber-Physical Systems; Software Performance Engineering.**

I. Introduction

Cyber-Physical Systems (CPS), Systems of Systems, Internet of Things, Industry 4.0, are all sectors that recently resulted to attract massive attention from research and major investments from industry [17]. All these systems share the trend of having a swarm of inter-connected devices that may represent the integration of computation with physical processes [6], or an orchestration of computers and physical systems [12]. The increasing heterogeneity of these systems paves the way to the development of modeling techniques suitable to sense-and-control their non-functional characteristics, such as performance, reliability, etc. Indeed, high system responsiveness coupled with low failures and costs becomes fundamental for the development of high-quality systems. Such demanding quality is also due to the wide range of application areas like health-care, smart grids and renewal energy, automotive with intelligent vehicles and aircraft avionics systems, e.g., [2], [4], [12].

With the evolution of computational and communication technology, CPS find applications in an increasing number of fields in industry, leading to the new factory generation called *smart factory*, where the core is represented by communicating and collaborating intelligent systems [13], that have the goal to optimize the products production and management while dealing with performance requirements [4]. For example, advanced robots, also known as smart machines operate autonomously and can communicate directly with manufacturing systems to accomplish tasks within a certain time frame [6]. Designing a CPS is always subject to many type of non-functional requirements, such as safety, performance and cost. These requirements are usually connected together, e.g., performance may affect both safety and budget.

In literature several approaches recently emerged for the modelling of CPS (e.g., [4], [16], [3], [9], [10]), however most of them focuses on functional analysis ([4], [9], [16]) or formal verification ([3], [10]), while performance-related characteristics result to be almost neglected. In this paper we investigate, at design time, if a CPS will be adequate to execute a particular task subject to performance requirements. The performance evaluation of systems under development before their implementation is fundamental, in fact it is intended to anticipate performance flaws and avoid late fixes during the testing and after the implementation. This early study allows engineers to evaluate the alternative system configurations and understand what are the ones fulfilling the stated requirements.

In our previous work [15], we presented a case study to elicit the challenges for the performance engineering of CPS, and we identified several challenges requiring a deeper investigation. In this paper we concentrate on the *modeling and evaluation of performance metrics* challenge, i.e., the difficulty to have software and hardware aspects coexisting in the same model still maintaining, when possible, a separation of concerns and allowing an approximate evaluation of systems characteristics. To deal with these issues, here we introduce a guided process called **IMPACt** (mult**I** **M**odelling **P**erform**A**nce **C**ps), which helps architects during the design phase, to better understand the behavior of the system under development and the design choices they made, through performance analysis on results obtained by runnable models derived from system high-level specification.

IMPACt enables the performance engineering of CPS by exploiting multiple models and properly combining them to support the integration of computational algorithms and physical components. In particular, in this first instance, we specify the following two model abstractions: (i) design *scenario* models, i.e., an explicit representation of system

978-1-5386-2431-9/17 $31.00 © 2017 IEEE
DOI 10.1109/ICECCS.2017.22

entities and their interactions regulated by input/output ports, along with a finite set of behaviors (or modes) and the rules that govern transitions between them; (ii) performance *meter* models, i.e., an explicit representation of performance indicators (such as utilization, throughput, and response time) and their calculations regulated by system events along with statistical information that report aggregated system observations.

The remainder of the paper is organized as follows. Section II provides an overview of state-of-art approaches. Section III details the IMPACt approach. Section IV concludes the paper and provides future research directions.

II. Related Work

A. Modeling of CPS

The problem of integrating multiple formalism and supporting their heterogeneity in CPS is indeed not trivial. In [4] the heterogeneity of CPS is highlighted, and it is discussed the need of computational cooperating models. Authors explain the pros and cons of an actor-based model-integrated development approach, commonly used for embedded systems, and present it as a good start point to design CPS. In [16] a domain-specific language (DSL) for the design of CPS is proposed and it is able to capture the computation, the communication and the control aspects of CPS. Furthermore, there is the *MechatronicUML* method [3]. based on the well-known UML standard and is designed to enable model-driven design of discrete software for self-adaptive mechatronic systems. Another modeling approach called Ptides is presented in [12] and is a programming model that aims to be a coordinator language for model-based design approaches of distributed real-time systems. Another approach to model CPS is the co-simulation [1] that takes as input models from different specific domains and a co-simulation engine manages the simultaneous execution of domain-specific tools while guaranteeing the synchronization and communication between them. A recent variation of co-simulation is presented in [10] where Functional Mockup Units (FMU) are used in the simulation process.

B. Performance evaluation of CPS

In literature there are not many works dealing with the performance evaluation of CPS, and usually they are specific to certain types of analysis. In [6] the problem of evaluating the performance of CPS is investigated by describing such systems as hybrid and dynamic distributed ones that are made of arbitrary compositions of timely (where execution time has known bounds) and untimely components. However, in [6] the simulation framework is sketched only, no case study is shown to demonstrate how the combination of multiple CPS models enables the performance analysis. There are related works dealing with non-functional characteristics but they are not primarily focused on performance. In [14] the main focus is security, and the performance is a

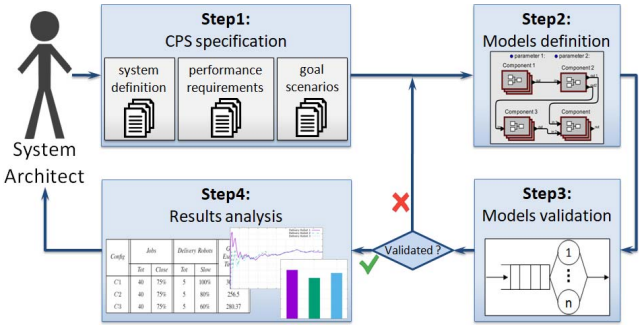


Figure 1. IMPACt high level view.

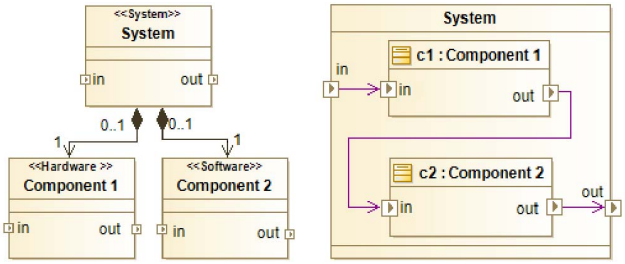
mathematical analysis on the maximum perturbation by an attacker. In [18] various privacy techniques are analyzed by quantifying their impact on communication and transmission performance, thus to evaluate the trade-off between privacy and performance. An alternative methodology to get measurements is represented by co-simulation, e.g., there are tools for modeling physical phenomena and deriving their measurements (OpenModelica [8]). The combined use of different tools (e.g, for software and physic dynamics) may lead to detailed performance measurements, but this still implies to have a deep knowledge of the system, and the performance evaluation is delayed until all the information is available. On the contrary, IMPACt allows the performance engineering of the system design and it is not mandatory to have equations of the physical phenomena.

III. The IMPACT approach

In this section we present the main steps involved in IMPACt. Starting from the specification of the system and performance requirements, multiple models are derived to support the architects in the evaluation of its performance characteristics. Figure 1 illustrates the main operational steps described hereafter.

Step 1. CPS specification

As first step the architect defines all the aspects of the target system needed for the design through specific documentation. The *system definition* is specified by some specific formalism, i.e., diagrams or high level models that allow describing the structural characteristics of the system, the connection between elements, their behavior, etc. (e.g., SysML [7]). In this documentation, the architect specifies for each element its type (e.g., sytem, hardware/physical, software). Figure 2a depicts an example of system definition, where on the left there is a diagram showing the system composition (i.e., components, cardinality, types, etc.), and on the right there is a diagram regulating the connections between components. Moreover, after the system definition, the architect has to specify its *performance requirements*. One possible way to define requirements is through tables, as shown in the example of Figure 2b, where the



(a) System definition.

RQ1	System.Throughput > 5 j/s
RQ2	System.ResponseTime < 90ms
RQ3	c1.Utilization < 50%

(b) Performance requirements.

G1	Each robot has to deliver objects without fully consuming its battery

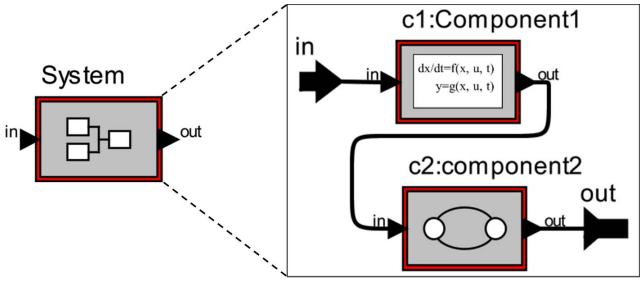
(c) Goal scenarios.

Figure 2. Example of CPS specification.

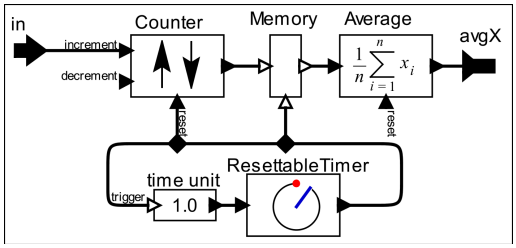
system throughput, system response time, and component utilization are required to fulfill some specific values. This specification represents the performance characteristics the architect is interested in monitoring for the system under design. Furthermore, the architect can define *goal scenarios* that represent the objectives required by the system to be considered successful, e.g., in Figure 2c we report an example where the completion of a task requires to avoid a full consumption of resources. Summarizing, at the end of this first step of IMPACt, the system architect has a set of documents, tables and abstract models that define the system under design, its performance requirements and its goal scenarios, if any.

Step 2. Models definition

The intrinsic heterogeneity and complexity of CPS requires a modeling approach that adopts a separation of concerns in terms of multiple view points and that comprises models of software, computational platforms, networks and system environment. Besides, as for any significative complex design, creating the right solution in a single step and/or using only one type of model is quite impossible. Furthermore, mixing high level concepts with lower level ones and environmental constraints, clearly reduces the solution reusability. To this end, we propose to apply a hierarchical component-based multi-modeling approach that includes the definition of different types of models. Then, the combination of these models in a hierarchical global one is aimed to embed the different components of the system and their behavior at the same time, even if they are subject to different execution rules (e.g., a controller's software is a discrete event object while a physic phenomenon is a continuous event). We distinguish the models in *design scenario* models, which describe the system structure and behavior, and the *performance meter* models that have to gather and elaborate performance data. Design scenario models are intended to represent system structure, components, their interactions and behavior. The skeleton



(a) Design scenario model.



(b) Performance meter model - system throughput.

Figure 3. Example of model definition.

of each component is defined from the specifications but how it is made internally is an architect choice. An example is a component describing a physical phenomena that could be modeled in detail with Ordinary Differential Equations (ODEs) or approximated with a discrete set of equations, but still conform to its interfaces specification. The approach gives this degree of freedom, the architect can model the system based on her/his knowledge and refine the model later in the design process. Figure 3a shows an example of a design scenario model derived from the diagrams reported in Figure 2a, where the hierarchical structure, the cardinality of the composition, the interfaces/ports, the connections, etc., are preserved. In the system definition the *Component1* and *Component2* elements have the stereotype of *Hardware* and *Software*, and such elements become respectively a component that models physic hardware phenomena by ODEs and a finite state automaton component to model discrete software behaviors. Performance meter models are specified as ad-hoc components for gathering performance indicators of interest. These models allow the derivation of the performance analysis results in order to understand the impact of each component on the overall system performance. Due to the enormous set of CPS applications, it is impossible to define general enough performance indices that hold for any application domain. Our approach allows the derivation of the canonical performance indices (such as utilization, response time, and throughput). Figure 3b shows an example of a performance meter model that measures the average system throughput over time, as required by RQ1 in Figure 2b. Note that the approach is extendible and it is also possible to implement specific ad-hoc model elements that measure other performance

characteristics relevant to the specific domain and system environment. Besides, the models definition step includes the specification of critical parameters (e.g., initial battery level) related to the goal scenarios, and it is possible to make use of a probabilistic model checking tool (PRISM[1]) to deduce the best values that can be assigned to achieve the specified goals with a high probability.

Step 3. Model validation

Once the models have been specified at Step 2, before using them for performance analysis, it is necessary to verify their accuracy in the prediction results. There are different ways to investigate this aspect. First of all, if the system implementation is available, then it is possible to conduct an actual validation of the model by comparing prediction results with monitored measurements. However, at design time, typically there is no actual implementation of the system, hence it is recommended the adoption of a virtual validation approach (similarly to [5]). The basic idea underlying the virtual validation is to compare the model results with other well-known and established theoretical models (e.g., Queueing Theory [11]), and this means that the most suitable theoretical model needs to be identified. If the system model obtained at Step 2 is too complex, then it is necessary to adapt or simplify it until such model becomes comparable to a corresponding one in the theoretical background. Model parametrization supports this step, in fact the architect can customize the models by introducing parameters facilitating the comparison with theoretical performance models (e.g., parametrize the cardinality of components, the mix of job types characterizing the workload, etc). When the design models approximate a theoretical one, then the results are compared and the relative gap is calculated. If the obtained gap is within a given error threshold, then the model can be considered as validated, otherwise the system architect has to go back to the modeling phase, adjust the model and repeat the validation step.

Step 4. Results analysis

Once the system model has been validated, it can be used to analyze complex scenarios to obtain the performance analysis results of interest. These results are then used by the system architects to investigate the different design alternatives and select the ones fulfilling the performance requirements and the goal scenarios defined at Step 1. In this way, architects are supported in the task of analyzing the performance characteristics of CPS design before the actual implementation is available. Besides, in case of unsatisfactory performance analysis results, some design changes in the model can be performed by the architect and the whole approach can be re-executed again.

[1]http://www.prismmodelchecker.org/

IV. Conclusion

In this paper we presented IMPACt, a multi-modeling methodology for enabling the performance engineering of CPS at design time. The model-based performance analysis allows an early understanding of the system behavior since it empowers the system architects with the possibility of evaluating alternative system configurations and understanding what are the ones fulfilling the requirements. As future work, we plan to to exploit model-driven engineering techniques to automatically generate performance models starting from abstract CPS design specifications. We also plan to validate the proposed approach on real-world case studies, possibly from different domains and industrial contexts.

References

[1] A. T. Al-Hammouri, "A comprehensive co-simulation platform for cyber-physical systems," *Computer Communications*, vol. 36, no. 1, pp. 8–19, 2012.

[2] R. Baheti and H. Gill, "Cyber-physical systems," *The impact of control technology*, vol. 12, pp. 161–166, 2011.

[3] S. Becker, C. Brenner, C. Brink, S. Dziwok, C. Heinzemann, U. Pohlmann, W. Schäfer, J. Suck, O. Sudmann, and R. Löffler, "The mechatronicuml design method–process, syntax, and semantics," *Software Engineering Group, Heinz Nixdorf Institute, University of Paderborn, Tech. Rep. tr-ri-12-326*, 2012.

[4] P. Derler, E. A. Lee, and A. S. Vincentelli, "Modeling cyber–physical systems," *IEEE*, vol. 100, no. 1, pp. 13–28, 2012.

[5] P. Feth, T. Bauer, and T. Kuhn, "Virtual validation of cyber physical systems." in *Software Engineering & Management*, 2015, pp. 201–206.

[6] A. E. S. Freitas and R. M. da Silva Bezerra, "Performance evaluation of cyber-physical systems," *ICIC Express Letters*, vol. 10, 2016.

[7] S. Friedenthal, A. Moore, and R. Steiner, *A practical guide to SysML: the systems modeling language*. Morgan Kaufmann, 2014.

[8] P. Fritzson, P. Aronsson, A. Pop, H. Lundvall, K. Nystrom, L. Saldamli, D. Broman, and A. Sandholm, "OpenModelica-A free open-source environment for system modeling, simulation, and teaching," in *IEEE International Conference on Control Applications*, 2006, pp. 1588–1595.

[9] G. Karsai and J. Sztipanovits, "Model-integrated development of cyber-physical systems," in *IFIP International Workshop on Software Technolgies for Embedded and Ubiquitous Systems*. Springer, 2008.

[10] P. G. Larsen, J. Fitzgerald, J. Woodcock, R. Nilsson, C. Gamble, and S. Foster, "Towards semantically integrated models and tools for cyber-physical systems design," in *International Symposium on Leveraging Applications of Formal Methods*. Springer, 2016, pp. 171–186.

[11] E. D. Lazowska, J. Zahorjan, G. S. Graham, and K. C. Sevcik, *Quantitative System Performance: Computer System Analysis Using Queueing Network Models*. NJ, USA: Prentice-Hall, Inc., 1984.

[12] E. A. Lee, "The past, present and future of cyber-physical systems: A focus on models," *Sensors*, vol. 15, no. 3, pp. 4837–4869, 2015.

[13] J. Lee, B. Bagheri, and H.-A. Kao, "A cyber-physical systems architecture for industry 4.0-based manufacturing systems," *Manufacturing Letters*, vol. 3, pp. 18–23, 2015.

[14] Y. Mo and B. Sinopoli, "On the performance degradation of cyber-physical systems under stealthy integrity attacks," *IEEE Trans. Automat. Contr.*, vol. 61, no. 9, pp. 2618–2624, 2016.

[15] L. Pagliari, R. Mirandola, and C. Trubiani, "A Case Study to Elicit Challenges for Performance Engineering of Cyber Physical Systems," in *International Workshop on Challenges in Performance Methods for Software Development (WOSP-C)*, 2017.

[16] M. U. Tariq, J. Florence, and M. Wolf, "Design Specification of Cyber-Physical Systems: Towards a Domain-Specific Modeling Language based on Simulink, Eclipse Modeling Framework, and Giotto." in *ACESMB@ MoDELS*, 2014, pp. 6–15.

[17] A. S. Vincentelli, "Let's get physical: Adding physical dimensions to cyber systems," in *IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, 2015, pp. 1–2.

[18] H. Zhang, Y. Shu, P. Cheng, and J. Chen, "Privacy and performance trade-off in cyber-physical systems," *IEEE Network*, vol. 30, no. 2, 2016.