Software Engineering for Smart Cyber-Physical Systems – Towards a Research Agenda

Report on the First International Workshop on Software Engineering for Smart CPS

Tomas Bures[1], Danny Weyns[2] (contributing workshop chairs)
Christian Berger[3], Stefan Biffl[4], Marian Daun[5], Thomas Gabor[6], David Garlan[7], Ilias Gerostathopoulos[1], Christine Julien[8], Filip Krikava[9], Richard Mordinyi[4], Nikos Pronios[10] (contributing participants)

[1]Charles University Prague, [2]Katholieke Universiteit Leuven & Linnaeus University, [3]Gothenburg University, [4]Technical University Vienna, [5]Universität Duisburg-Essen, [6]LMU München, [7]Carnegie Mellon University, [8]University of Texas, Austin, [8]Czech Technical University, [10]Innovate UK

<bures@d3s.mff.cuni.cz>, <danny.weyns@kuleuven.be>, <christian.berger@gu.se>, <stefan.biffl@tuwien.ac.at>, <marian.daun@paluno.uni-due.de>, <thomas@denkfrei.de>, <garlan@cs.cmu.edu>, <iliasg@d3s.mff.cuni.cz>, <c.julien@mail.utexas.edu>, <filip.krikava@fit.cvut.cz>, <richard.mordinyi@tuwien.ac.at>, <nikos.pronios@innovateuk.gov.uk>

DOI: 10.1145/2830719.2830736
http://doi.acm.org/10.1145/2830719.2830736

Abstract

Cyber-Physical Systems (CPS) are large interconnected software-intensive systems that influence, by sensing and actuating, the physical world. Examples are traffic management and power grids. One of the trends we observe is the need to endow such systems with the "smart" capabilities, typically in the form of self-awareness and self-adaptation, along with the traditional qualities of safety and dependability. These requirements combined with specifics of the domain of smart CPS – such as large scale, the role of end-users, uncertainty, and open-endedness – render traditional software engineering (SE) techniques not directly applicable; making systematic SE of smart CPS a challenging task. This paper reports on the results of the First International Workshop on Software Engineering of Smart Cyber-Physical Systems (SEsCPS 2015), where participants discussed characteristics, challenges and opportunities of SE for smart CPS, with the aim to outline an agenda for future research in this important area.

Keywords: software-engineering, cyber-physical systems

Introduction

Cyber-physical systems have been in focus of the academic community and industry for almost a decade. They are recognized as a high priority for research and innovation by funding bodies all over the world. NSF describes Cyber-Physical Systems (CPS) as "engineered systems that are built from, and depend upon, the seamless integration of computational and physical components" [1]. EU refers to CPS as "the next generation embedded ICT systems that are interconnected and collaborating, providing citizens and businesses with a wide range of innovative applications and services" [2]. In the funding programs, CPS are seen as an overarching solution to bringing the vision of "smart everything" (i.e. smart society, smart energy, smart mobility, etc.).

While CPS have traditionally belonged to the domain of embedded systems and hardware, the recent heavy proliferation of smart embedded and mobile devices moved CPS towards large-scale networked distributed systems, which combine various data sources to control real-world ecosystems (e.g. intelligent traffic control and smart grids) [8][10][11]. As a result, CPS have to deal with environment dynamicity, control their emergent behavior, and be scalable and tolerant to threats. This creates the discernible trend of towards endowing CPS with a good portion of intelligence that allows them to effectively cope with all these issues and deliver the "best possible" service in a given situation. Following the terminology of EU's research agenda (e.g. H2020 ICT 2014 call [2], ECSEL JU [3][4]), we call these systems *smart CPS* (sCPS).

To achieve the necessary level of intelligence, sCPS rely heavily on software – up to the level where software becomes the most complex and critical part of the systems. Software not only controls the operation of sCPS, but also manages the self-awareness of sCPS, the awareness of the environment in which they operate and their goals, and provides the ability to cope with uncertain and emerging situations. The complexity of such software is further increased by the fact that sCPS are heavily distributed and decentralized, yet they need to operate cooperatively. The software also takes an active role in the safety and dependability of sCPS.

This poses significant challenges in software development for sCPS and calls for systematic software engineering (SE) approaches to build sCPS. The problem in particular stems from the fact that sCPS are subject to a number of specifics that render traditional SE methods not directly applicable [10]. These specifics include the blurring boundaries between the system and its environment, large scale and inherent complexity, the role of end-users, multi-level uncertainty, open-endedness, to name a few. What is needed are innovative approaches that jointly reflect and address the specifics of such systems. Although such innovative approaches exist in isolation, their synergy to address the specifics of sCPS in a holistic manner remains an open challenge [9].

SEsCPS Workshop

The SEsCPS workshop[1] aims at bringing the required synergy of domains and SE methods in the specific settings of sCPS. It brings together academics and practitioners with as primary objectives: (i) to increase the understanding of problems of SE for sCPS, (ii) to study the underlying foundational principles for engineering sCPS (e.g. reasoning about uncertainty, dealing with emergent behavior, distributed control), and (iii) to identify promising SE

[1] http://d3s.mff.cuni.cz/conferences/sescps/

solutions for sCPS (e.g. architectural principles, innovative languages and models, engineering processes, integrated V&V).

The first edition of the workshop was collocated with the International Conference on Software Engineering, ICSE 2015 in Firenze, Italy. The workshop attracted 17 submissions, out of which 5 were accepted as full papers and 5 as position and future-trends papers. In total, 25 participants attended the workshop. The workshop started with a keynote by David Garlan, CMU, USA on modeling challenges for CPS. The rest of the morning was devoted to presentations of accepted papers, grouped in themes. The whole afternoon of the workshop was devoted to discussion in breakout groups, where participants discussed focused topics of SE for smart CPS. A plenary report session concluded the workshop.

Workshop Themes

The workshop presentations focused on timely and important aspects of SE for sCPS. They were organized in four themes: modeling dimension, faults and conflicts, testing and verification and multi-dimensional collaboration, overviewed below.

Modeling Dimension

The first theme concerned the modeling dimension of sCPS. Since sCPS are inherently multi-disciplinary, multiple views of the same system are produced within each discipline (control engineering, electrical engineering, software engineering, mechanical engineering, etc.). This raises the fundamental question how to obtain a "ground truth" for the system in place, that is, a consistent representation that will unify the different views and will support system-level tradeoff analysis? David Garlan [13] in his keynote talk pointed out that today's model-based CPS approaches have various difficulties, including the integration of different modeling approaches, handling analysis of trade-offs, ensuring consistency and completeness, and integrating humans in the loop. To tackle these difficulties, he proposed to extend traditional software architecture with architectural views to support both cyber and physical elements through a CPS architectural style. Such representation should go hand in hand with tools for dependency analysis and coordination. Finally, the keynote speaker explained how his team employs stochastic multi-player games to handle the difficulty of integration of humans in the loop. The modeling challenges emerged as a recurring theme in several of the following paper presentations. Vasileios Koutsoumpas [17] explained how uncertainty in CPS is multi-faceted and to handle that presented a model-based approach for the specification of a virtual power plant operating in open context, based on focus theory [5]. Franco Raimondi [18] pointed out the discrepancy between high-level application requirements and resource-limited microcontrollers and proposed an approach that exposes "components" as services in resource-limited microcontrollers, enabling the faster development of sCPS.

Faults and Conflicts

The second theme discussed at the workshop was faults and conflicts. Given the variety of complexities of sCPS on the one hand, and the level of criticality of many of these systems, faults and conflicts are a core theme of these systems. Andy Podgurski [14] elaborated on the problem of localization of faults in models. He proposed to use statistical models to model normal behavior and identify and trace variables that cause adverse and anomalous events. Miki Yagita [15] zoomed in on conflicts in cooperation of CPS. He proposed to employ and check metadata to handle conflicts of actuating the same physical entity in different ways. Finally, Richard Mordinyi [16] presented challenges in collaborations across engineering disciplines for the parallel engineering of cyber-physical production systems. He highlighted the conflicts in specifications when CPS evolve and the need for traceable versioning of engineering artifacts at modeling level rather than at file level.

Testing and Verification

An important challenge for smart cyber-physical systems is the systematic evaluation of the entire software system. In contrast to, for instance, Web-based systems, a sCPS interacts with its physical surrounding by perceiving data thereof and derive decisions for actions therein – like driving trajectories for self-driving vehicles. The interaction with the physical world (which is of continuous nature and inherently uncertain) and the dynamicity and open-endedness of sCPS makes their testing and verification a difficult challenge. Approaching this challenge Christian Berger [19] presented an approach for regression testing of software modules that based on resource-isolation; thus, individual test cases are wrapped into isolated process contexts allowing their parallel execution without mutual interference. Christine Julien [20] presented interesting results of an empirical study on the demands on tools and techniques for verification and validation of CPS. The initial conclusions are: (i) trial-and-error testing (state of the practice) does not provide sufficient rigor in error detection, (ii) formal methods provide a desired level of expressiveness but are neither intuitive nor efficient, and (iii) existing simulation tools are limited in their capabilities to jointly model physical and cyber components.

Collaboration

The challenges arising from the collaboration of smart cyber-physical systems is multi-dimensional, as different types of collaboration can be distinguished. First, sCPS are commonly developed by collaboration of different developers, units, and companies. Furthermore, there is a need to develop sCPS in collaboration with existing system releases. For example, the use of a sCPS in an unintended context results in new requirements to be elicited and addressed during development. Third, sCPS are commonly designed to be collaborative in nature. This means, several sCPS collaborate at runtime to create a system network, which comprises not only the single systems but generates additional benefits from the collaboration of the single systems. Thomas Gabor [21] addressed the first challenge within his presentation and proposed an approach for continuous collaboration between human programmers in the development phase and automated adaptation agents during runtime, thus blending classical phases of a sCPS life-cycle. Regarding the second problem, Marian Daun [22] discussed in his talk the problem of collaborating multiple system instances of sCPS. While embedded systems are typically designed on a type level, specifications lack the definition of collaboration between multiple instances of the same sCPS. Franck Fleurey proposed an approach to foster heterogeneity and distribution of collaborating sCPS, which allows to combine multiple existing engineering approaches into a common framework.

Open Research Topics

The whole afternoon of the workshop was allocated to breakout groups, which focused on selected topics from the morning presentations. In total there were 4 groups, each focusing on one of the

topics selected for discussion: Aligning different disciplines; Human in the Loop; Pragmatic and Systematic Engineering; Uncertainty. In the rest of the section, we report on the findings of each breakout group in turn.

Aligning different disciplines

One of the grand challenges in SE for sCPS is to devise ways to integrate software engineering principles and practices with disciplines such as mechanical and electrical engineering, control engineering, and physics [6][8]. This alignment is challenging, as different disciplines adopt different views over the sCPS world, and base their models and analysis and design methods in different sets of assumptions. It is also essential, as sCPS is by its own nature a complex multi-disciplinary domain.

One of the questions discussed in the breakout group is on which basis to attempt such an alignment. The conclusion was that a promising direction, advocated also by David Garlan in the keynote talk, is to use *software architecture models* as the common vocabulary across disciplines and as the vehicle to map different views into a representation that is commonly understood and used by software engineers. In this respect, software architectures become "richer", as they integrate information not only about the software elements of a sCPS, but also about other aspects such as electro-mechanical elements, physical constraints and laws.

Regarding the development of sCPS with self-adaptive capabilities, there was a consensus that there is large potential in integrating well-studied and formally proven techniques from control engineering [7]. Control engineering will also benefit from a new, exciting, and highly challenging field of application.

The software engineering community is only beginning to investigate the alignment (or, rather, the lack thereof) between software engineering and other disciplines in the sCPS domain and beyond. In this respect, we identified the need for more empirical research in understanding, quantifying, and bridging this gap.

Human in the Loop

A distinct element of sCPS is the prominent role of humans in the creation and operation of these systems, as also outlined by David Garlan in his keynote. Different "types of humans" can be involved in cyber physical systems, ranging from developers and engineers from different disciplines (including software, mechanical, electrical) who create; operate, and evolve the system up to end users who define (and change) key requirements and interact and use the system. A specific type is "system-external humans" that are influenced by the CPS and may be affected by it (e.g. pedestrians in context of autonomous driving).

"Being in the loop" for sCPS implies being involved in a lifetime loop that spans all phases of the lifecycle of sCPS, from inception to operation and termination. Engineers create the runtime loop and improve the system throughout its lifetime (iterative development). End users interact with the system during the runtime-loop either based on explicit communication between CPS and the human, e.g., humans need to resolve conflicting requirements; or based implicit communication, e.g., assumptions made by the CPS based on the analysis of historical data.

Based on the analysis, we identified three important research questions: (1) Who should a sCPS interact with and how to make humans aware of changes? When to interact with engineers; when with end users? How should a notification model look like so that the human is efficiently supported in decision making (e.g., resolving conflicts regarding requirements) and can provide high-quality feedback to CPS. (2) What type of changes may end-users introduce to CPS? Can the CPS „calculate" the costs of adaptation to new requirements? In what way does the need for human input limit the capabilities of CPS? How are CPS capable of refining/optimizing „bad" user input to obtain the most effective/efficient outcome? Finally: (3) Which responsibilities/actions do humans (always) need to have/execute? What are the dependencies of the capabilities of CPS on human input? When does the CPS have to involve the human? How does the system make the user aware of the impacts of the user's decision? Clearly, the prominent role of humans in the creation and operation of sCPS poses severe questions and challenges to software engineers.

Pragmatic and Systematic Engineering

From a "purist" software engineering perspective, developing a software system requires a clear and complete high-level description of the system and a "compiler" that can generate correct code. Domain specific languages and models offer a first step towards bringing this idealistic perspective closer to reality. However, engineering CPS is not a "purist" software engineering problem, it is an inherently interdisciplinary problem. This leads to a fundamental problem that needs to be solved, i.e., the "mismatch" in the languages spoken by software engineers and other domain experts. In the current practice the domain experts typically build a system. When they face (inherent software engineering) problems and become desperate, they call in a software engineer to try to find a solution to the problem. The software engineer has then to become immersed in the domain. Some recent anecdotal efforts hint at interdisciplinary teams that engineer CPS with domain experts and software engineers from the start.

Another orthogonal engineering aspect related to the degree of engineering rigor is that there is a continuum of sCPS, ranging from mission critical systems that require a rigorous engineering approach from the outset, to more "user" level systems (e.g., in the IoT, consumer electronics, "maker" systems, etc.) that may allow more engineering pragmatics because of less demanding requirements or the pressure of the market.

Based on the analysis, we identified several interesting directions to move engineering practice of sCPS forward. We formulate them here as questions for further study. Can engineers leverage on system's "smartness" to aid in high quality systems? Maybe it's acceptable for the system to have flaws, as long as it is equipped with the necessary intelligence to recover or repair itself when needed. Can we derive models that are inherently composable? This would enable transferring quality properties of model elements based on design time checking. Furthermore, the analysis of individual models may be more tractable. In this vision, model "connectors" could account for consistency, timing, security, etc. In addition, there will be a need to add runtime checking to resolve context issues and uncertainties.

In any case, software engineers should not try to replace the domain experts. The domain expert needs to be able to rely on tools to introspect correctness (even at the component level), test in the deployment environment (relationship to test suites and "certifiable" components), and introspect the level of smartness and self-adaptation. This raises important questions such as: How do we make tools accessible and trustworthy to the domain experts? Do

we need mixed teams? Key will be to provide evidence for the effectiveness and efficiency of tools.

Uncertainty

Another fundamental challenge in SE for sCPS is how to cope with uncertainty. While uncertainty has garnered much attention in SE research for the last years, a concrete definition of uncertainty in the context of sCPS is difficult to give. Hence, an important question is to identify what uncertainty for sCPS contains. The term of uncertainty is elusive as it subsumes uncertainty regarding the requirements, the environment and context, behavior, and infrastructure. Furthermore, regarding the behavior it is important to distinguish between the uncertainty in the (expected) behavior of human users, uncertainty in the participation of human users, uncertainty in the emergent behavior of systems of systems, and uncertainty in the adaptation of the system. Furthermore, current approaches differ w.r.t. uncertainty that is known (i.e. the developers know at design time that they are uncertain regarding specific aspects of system development) and unknown uncertainty (i.e. situations in which developers lack the knowledge about certain aspects either aware or unaware; in the latter case they may assume to have complete knowledge of a development aspect but it reveals later on that they were wrong). In addition, particularly in model-based engineering, proper documentation of uncertainty is a current research topic.

Handling uncertainty usually requires monitoring, as well-known from self-adaptive systems. Monitoring can be used to collect data from the context as well as the system itself. In doing so, the collected data is commonly used to validate the assumptions made during engineering but can, furthermore, be used to help revealing unknown uncertainty. Therefore, it is necessary to collect a sufficient amount of data, even if some of the data turns out to be irrelevant. Given the multifaceted nature, monitoring to handle uncertainty in sCPS needs to go beyond monitoring well known variables, but about collecting data of multiple variables and analyzing combinations of the values.

Another important element to mitigating uncertainty is involving humans in the operation of sCPS in situations that have been unforeseeable. Hence, human in the loop engineering can significantly aid in coping with unknown uncertainty to modify the system's behavior at runtime with binding decisions of either the user in the loop, the developer in the loop, or an operator in the loop.

Looking at existing solution approaches, the question arose whether uncertainty in sCPS poses some really new challenges, and whether this requires changes the way software for sCPS is developed. From the analysis, we concluded that, first of all, there is a lack of methods to systematically analyze the collected data, which must be addressed in the future. Second, as sCPS are inherently evolutionary, this may deeply affect the way these systems are built. System design must already allow for flexibility to cope with arising uncertainty later on, but at the same time, system design should not strive to cope with every possible change at runtime, as this imposes unmanageable development processes.

Conclusions

The demanding requirements of cyber physical systems combined with their specifics require that we equip them with "smart" capabilities. Such smartness crosscuts the whole system stack, but heavily relies on software. We have argued from different angles – the modeling dimension, faults and conflicts, testing and verification, and multi-dimensional collaboration – that traditional software engineering techniques no not render to be directly applicable to deal with such smart cyber physical systems. This raises fundamental challenges to software engineers. In this paper, we have elaborated on four of them.

Engineering sCPS requires the alignment of software engineering principles and practices with other disciplines such as mechanical and electrical engineering, control engineering, and physics. This alignment needs to be grounded in open-mindness and respect for disciplines and a shared vocabulary for communication. An interesting approach for further study is to extend software architecture to incorporate cyber and physical elements through a CPS architectural style.

Humans are involved in all phases of the lifecycle of sCPS. As such, engineers need to treat humans as first-class citizens in the design, operation, and usage of sCPS. This raises fundamental challenges about with whom, when, and how the system should interact with humans. Underneath, this requires engineers to identify and distribute responsibilities across machines and humans. Evidently, this raises ethical issues for critical domains such as autonomous driving cars and smart e-health systems.

Engineers of sCPS need to find the right balance between rigor with pragmatism. Engineering sCPS is inherently an interdisciplinary effort that spans a continuum of types of systems, from mission critical ones to user level systems as in the IoT. Opportunities to provide the necessary assurances for the domain at hand lay in exploiting the smartness of the sCPS and the potential of composable models to make analysis tractable and consolidate qualities.

Last but not least, a fundamental challenge to engineers of sCPS is handling uncertainty, resulting from aspects such as inherent decentralization, the role of end-users, and open-endedness. One promising idea is continuous verification, where evidence is provided at design time under uncertainty, and the machine adds complementary evidence at runtime to handle uncertainty, when the necessary information becomes available. In [12] the authors go a step further and coin the term "perpetual assurances," referring to the enduring process that continuously provides new evidence by combining system-driven and human-driven activities.

Acknowledgements

The SEsCPS workshop is a collective endeavor, as such, the authors would like to express their appreciation to all the participants of the workshop. We also thank all who have participated in the organization of this workshop. In particular, we thank Mark Klein and Rodolfo E. Haber for their support in preparing the workshop proposal and planning the event. Furthermore, we are grateful to the ICSE 2015 organizing team that provided excellent support ot host the workshop, the ICSE 2015 Workshops Co-Chairs, in particular Raffaela Mirandola and Zhendong Su, and the SEsCPS Program Committee comprised of Paris Avgeriou, Steffen Becker, Johann Bourcier, Herman Bruyninckx, Radu Calinescu, Sagar Chaki, Ivica Crnkovic, Rogerio De Lemos, Dionisio de Niz, Antonio Filieri, Carlo Ghezzi, Holger Giese, Matthias Hölzl, Gabor Karsai, Steffen Zschaler, Filip Krikava, Martina Maggio, Henry Muccini, Maurizio Murroni, Bernhard Schätz, Ina Schieferdecker, Lionel Seinturier, Vitor E. Silva Souza, and Petr Tuma.

References

[1] National Science Foundation, Cyber Physical Systems, NSF 14-542. Online: http://www.nsf.gov/pubs/2014/nsf14542/nsf14542.htm

[2] EU Horizon 2020, Smart Cyber-Physical Systems ICT-01-2014. Online: http://ec.europa.eu/research/participants/portal/desktop/en/opportunities/h2020/topics/78-ict-01-2014.html

[3] ECSEL Joint Undertaking 2014 call. Online: http://www.ecsel-ju.eu/Call2014.html

[4] 2014 ECSEL MultiAnnual Strategic Research and Innovation Agenda. Online: http://www.artemis-ia.eu/sra

[5] M. Broy and K. Stølen. Specification and Development of Interactive Systems – Focus on Streams, Interfaces, and Refinement. Monographs in Computer Science, Springer, 2001.

[6] P. Derler, E. a. Lee, and a. S. Vincentelli. Modeling Cyber-Physical Systems. Proceedings of the IEEE, 100(1):13–28, January 2012.

[7] A. Filieri, M. Maggio, K. Angelopoulos, N. D'Ippolito, I. Gerostathopoulos, A. B. Hempel, H. Hoffmann, P. Jamshidi, E. Kalyvianaki, C. Klein, F. Krikava, S. Misailovic, A. V. Papadopoulos, S. Ray, A. M. Sharifloo, S. Shevtsov, M. Ujma, and T. Vogel. Software Engineering Meets Control Theory, 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems. IEEE, 2015.

[8] B. K. Kim and P. R. Kumar, "Cyber–Physical Systems: A Perspective at the Centennial," Proceedings of the IEEE, vol. 100, no. Special Centennial, pp. 1287–1308, 2012.

[9] K. H. (Kane) Kim, "Desirable Advances in Cyber-Physical System Software Engineering," in 2010 IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing, 2010, no. 978, pp. 2–4.

[10] E. A. Lee, "Cyber Physical Systems: Design Challenges," 11th IEEE International Symposium on Object Oriented Real-Time Distributed Computing, 2008, pp. 363–369.

[11] L. Sha, S. Gopalakrishnan, X. Liu, and Q. Wang, "Cyber-Physical Systems: A New Frontier," in Proceedings of the 2008 IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing, 2008, pp. 1–9

[12] D. Weyns, N. Bencomo, R. Calinescu, J. Camara, C. Ghezzi, V. Grassi, L. Grunske, P. Inverardi, J.M. Jezequel, S. Malek, R. Mirandola, M. Mori, and G. Tamburrelli, Perpetual assurances in self-adaptive systems, Assurances for Self-Adaptive Systems, Dagstuhl Seminar 13511, 2014

[13] D. Garlan, Modeling Challenges for CPS Systems, Software Engineering of smart Cyber Physical Systems, SEsCPS 2015 (keynote)

[14] K. Liang, Z. Bai, M. C. Cavosoglu, A. Podgurski, S. Ray, Fault Localization in Embedded Control System Software, Software Engineering of smart Cyber Physical Systems, SEsCPS 2015

[15] M. Yagita, F. Ishikawa, S. Honiden, An Application Conflict Detection and Resolution System for Smart Homes, Software Engineering of smart Cyber Physical Systems, SEsCPS 2015

[16] R. Mordinyi, S. Biffl, Versioning in Cyber-Physical Production System Engineering? Best-Practice and Research Agenda, Software Engineering of smart Cyber Physical Systems, SEsCPS 2015

[17] V. Koutsoumpas, A Model-based Approach for the Specification of a Virtual Power Plant Operating in Open Context, Software Engineering of smart Cyber Physical Systems, SEsCPS 2015

[18] M. Bordoni, M. Bottone, B. Fields, N. Gorogiannis, M. Margolis, G. Primiero, F. Raimondi, Towards Cyber-Physical Systems as Services: the ASIP Protocol, Software Engineering of smart Cyber Physical Systems, SEsCPS 2015

[19] C. Berger, Accelerating Regression Testing for Scaled Self-Driving Cars with Lightweight Virtualization – A Case Study, Software Engineering of smart Cyber Physical Systems, SEsCPS 2015

[20] X. Zheng, C. Julien, Verification and Validation in Cyber Physical Systems: Research Challenges and a Way Forward, Software Engineering of smart Cyber Physical Systems, SEsCPS 2015

[21] M. Hölzl, T. Gabor, Continuous Collaboration: A Case Study on the Development of an Adaptive Cyber-Physical System, Software Engineering of smart Cyber Physical Systems, SEsCPS 2015

[22] M. Daun, J. Brings, T. Bandyszak, P. Bohn, T. Weyer, Collaborating Multiple System Instances of Smart Cyber-Physical Systems: A Problem Situation, Solution Idea, and Remaining Research Challenges, Software Engineering of smart Cyber Physical Systems, SEsCPS 2015

[23] B. Morin, F. Fleurey, O. Barais, Taming Heterogeneity and Distribution in sCPS, Software Engineering of smart Cyber Physical Systems, SEsCPS 2015