Cyber-Physical Systems Design: Formal Foundations, Methods and Integrated Tool Chains

(Invited Paper)

John Fitzgerald*, Carl Gamble*, Peter Gorm Larsen†, Kenneth Pierce*, Jim Woodcock‡
*School of Computing Science Newcastle University, Newcastle upon Tyne, UK
Email: John.Fitzgerald@ncl.ac.uk, Carl.Gamble@ncl.ac.uk, Kenneth.Pierce@ncl.ac.uk
†Dept. of Engineering, Aarhus University, Denmark
Email: pgl@eng.au.dk
‡Dept. of Computer Science, University of York, UK
Email: Jim.Woodcock@york.ac.uk

***Abstract*—The engineering of dependable cyber-physical systems (CPSs) is inherently collaborative, demanding cooperation between diverse disciplines. A goal of current research is the development of integrated tool chains for model-based CPS design that support co-modelling, analysis, co-simulation, testing and implementation. We discuss the role of formal methods in addressing three key aspects of this goal: providing reasoning support for semantically heterogeneous models, managing the complexity and scale of design space exploration, and supporting traceability and provenance in the CPS design set. We briefly outline an approach to the development of such a tool chain based on existing tools and discuss ongoing challenges and open research questions in this area.**

I. Introduction

Cyber-physical systems (CPSs) consist of interacting computing and physical entities. Examples range from products incorporating embedded systems to large-scale applications such as distributed control of driverless vehicles. To the CPS engineer, the system of interest includes both cyber and physical elements. Consequently the foundations, methods and tools of CPS engineering should incorporate both the discrete models of computing hardware and software, and the continuous-value formalisms of physical (e.g. mechanical, electrical, electronic) engineering.

Since system-level dependability properties involve both cyber and physical aspects, the verification of global behaviours must take account of the heterogeneity of models. If diverse models are brought together only at the end of a development process, emerging failures are likely to be hard to both trace back to faults, and to remedy. Treating disciplines separately within the design process thus has the potential to slow innovation. If dependable CPSs are to be engineered economically, the design process must be collaborative and multi-disciplinary, while also permitting the assurance-raising activities of simulation, testing and verification.

There is growing interest in sound methods and tools for CPS engineering. There have been significant research investments by the National Science Foundation in the US[1], and by Horizon 2020 in the EU [1], as well as by community activities such as a thriving CPS Week and numerous workshops. There have been repeated calls for better notations for model-based CPS engineering [2], [3], [4], [5], [6]. However, the nature of CPS design raises many challenges not currently met [7].

In this paper we motivate the use of formal methods in addressing three challenges that we believe must be met if dependable CPSs are to be designed cost-effectively. First, engineering disciplines have distinct cultures and formalisms. For example, systems engineers work with notations such as SysML [8], whereas control engineers use continuous-time formalisms and software engineers use discrete-event notations; dialogue between disciplines is essential [9]. Second, the design space for a CPS is large; we require firm semantic foundations to allow exploration of trade-offs between physical components, hardware and software, rapidly modifying and reevaluating designs. Third, support is needed to help maintain traceability over the complex collections of artefacts produced in a CPS development, allowing the provenance of all elements to be recorded, and the final system linked to the requirements. In considering the role of formal methods in CPS design, we aim to adhere to principles identified for their successful industrial deployment, including the carefully targeted use of formalism, development of robust tools, and a focus on the priorities of integration into established design flows [10].

In Section II we identify some basic concepts of collaborative model-based design for cyber-physical systems and review baseline technologies on which our current work builds. We then consider the foundations required for heterogeneous modelling and analysis (Section III), the exploration of the design space (Section IV) and the need to support traceability and provenance (Section V). Throughout, we refer to an example based on a simple 2-wheeled personal transportation device (the "ChessWay"). We conclude by identifying open issues in the methods, tools and practice of model-based CPS design (Section VI), and briefly describe the goals of our current work in this area.

[1]See http://www.cps-vo.org

978-1-4673-7043-1/15 $31.00 © 2015 IEEE
DOI 10.1109/FormaliSE.2015.14

II. Basic Concepts

We regard a *system* as a combination of interacting elements organised to achieve a stated purpose [11], and a *dependable* system as one on which reliance may justifiably be placed [12]. A CPS is a system in which some of the elements are computational and some are physical [3]. The verification of CPS dependability therefore entails analysis of computational and physical processes, and their interactions. The goal of our research is to free CPS engineers to explore design alternatives, allocating and reallocating responsibilities to cyber and physical elements in an effort to deliver the functional and extra-functional behaviours required at the CPS system level. We use the term *design space exploration (DSE)* to refer to the construction and evaluation of a range of designs in order to identify a preferred solution.

In our work, designs are represented as models. A *model* is an abstract representation of a putative CPS. The abstractions in the model should be guided by its declared purpose, but the model should be *competent* in that it should contain features sufficient to allow confidence to be placed in the outcome of analyses conducted on it. The challenge that we address in our work is that of creating methods and tools to support multi-disciplinary model-based development. A *collaborative model (co-model)* contains discrete-event (DE) models of CPS cyber elements with continuous-time (CT) models of physical elements (typically the environment and/or controlled plant).

We have developed and demonstrated methods and tools for the construction of co-models, and their analysis through *co-simulation*, using VDM [13] as the DE formalism and 20-sim[2] [14] as the CT framework. The approach, which has been implemented in the Crescendo[3] open tools platform [15], has been applied in case studies that have demonstrated the value of early co-modelling in reducing the number of physical prototypes required in design [16]. However, the technology is limited to co-models of single controllers and plant, rather than networked controllers and multiple physical elements. The approach has so far also been restricted to verification by means of co-simulation.

The CPS concept is often used to refer to networked "smart" devices interacting with their physical environment. In many such applications, notably in areas such as transport and infrastructure, the elements of the system of interest are themselves operationally and managerially independent systems, often pre-existing, but brought together to deliver an emerging collaborative behaviour. Such *systems of systems (SoSs)* share CPS characteristics [17], and also merit holistic and collaborative design [18]. In our previous work on model-based approaches to SoS engineering [19] we have sought to provide common semantic foundations for heterogeneous SoS models at the DE level only; there is potential to extend this to the demands of CPS engineering.

[2]http://www.20sim.com

[3]http://www.crescendotool.org

III. Heterogeneous Modelling and Analysis

A. Current Capabilities

Semantic heterogeneity arises in two contexts. First, at a given level of abstraction, we would expect co-models to integrate diverse engineering formalisms. Second, while much research in formal methods for model-based design focuses on the delivery of individually effective tools, multi-disciplinary CPS design requires coherent *tool chains* formed from diverse tools, each optimised for a given purpose.

There have been calls for a science and technology foundation for CPS design that is model-based, precise, and predictable [20] while supporting integration of a range of semantic bases [21]. The state of the art has been characterised as almost exclusively involving discrete abstractions of continuous behaviour [22], although several model-based approaches, including that of Ptolemy [23], support heterogeneous modelling and simulation.

The state of the art is still some way from providing generic life-cycle tool chains from requirements to maintenance, especially with sound formal foundations. Such an integrated tool chain for CPS requires that evidence supplied by the different tools can be reconciled to produce coherent analysis results. Different analysis tools are based on different notations, for example a simulator may work at the level of a transition relation described using Structural Operational Semantics (SOS) rules (as is the case for the Crescendo co-simulation tool), whilst a program verifier may use an axiomatic Hoare calculus.

Although comprehensive formal foundations are still required, there is progress on platforms to support key links in tool chains, e.g. Cosimate[4] is a backplane co-simulation tool offering interfaces to tools like Simulink, Modelsim, and Modelica [24], test automation, DSE and system description in SysML. Canedo [25] has developed a multi-disciplinary, integrated, design automation tool for automotive CPS that evaluates system-level impact of domain-specific design decisions using simulation.

B. Example: a Personal Transporter

We introduce a small example to illustrate the need for heterogenous modelling and analysis: a 2-wheeled self-balancing personal transporter called the "ChessWay" (described in detail elsewhere [16]). We refer to this example in describing further challenges in Sections IV and Section V below.

The ChessWay (Fig. 1, left) consists of a platform, a handlebar and two wheels. It has two powerful motors to drive the wheels and keep the device upright— without active control, the device will fall over. The rider stands on the platform and holds the handlebar. By leaning forwards and backwards, the rider can command the device to move forwards and backwards. The device has various sensors to allow its position and movement to be detected, as well as on/off, safety cut out and direction switches, shown in the simplified schematic (Fig. 1, left). Active control is performed by two

[4]http://www.chiastek.com/products/cosimate.html

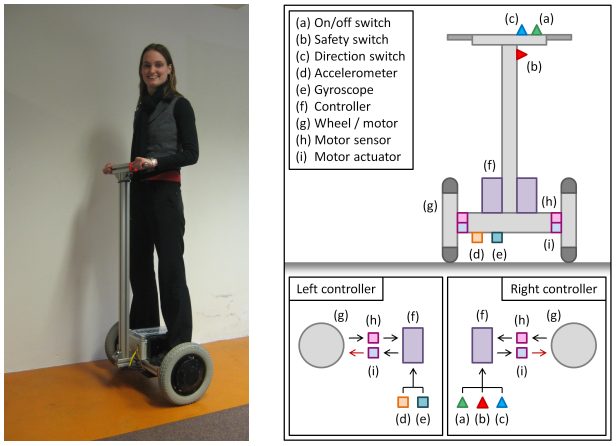

Fig. 1. Photograph of the ChessWay personal transporter (left) and a schematic of its important components (right).

networked controllers that each control one wheel and have access to different sets of sensors. Therefore the controllers must communicate regularly to maintain safe operation.

The ChessWay was initially studied with a high-fidelity physics model created in 20-sim using bond graphs. It was also possible to design the low-level, discrete-time (DT) loop controllers that balance the device. However, the ChessWay requires much more than this loop controller can deliver, including modal behaviours for safe startup and shutdown, fault tolerance mechanisms such as a safety monitor, and synchronisation between distributed controllers. These are much better studied in a DE model, such as the one produced for the ChessWay study in VDM. These two models were combined using the Crescendo technologies [16] to produce a co-model that better captures the ChessWay as a whole cyber-physical system.

Other elements of the ChessWay system could be better modelled in other formalisms, supported by other tools, to create better system models. For example, the network connection between the two controllers receives significant electro-magnetic interference from the large motors, therefore a realistic model of network loss and corruption is necessary to truly demonstrate the control software's suitability. On the CT side, contact modelling and collision response are currently not well supported in 20-sim or the bond graph formalism. Other modelling tools could provide more realistic models to enhance the physical model of the ChessWay.

In other cases it is not always clear where to model certain behaviours. For example, the DT loop controller was originally designed in the CT formalism, however in order to analyse the potential system performance and distribution across multiple controllers, it was necessary to move this behaviour to the DE model in VDM. Consider also that the rider's behaviour has a profound effect on the system's behaviour. They are responsible for moving their centre of gravity around to command the ChessWay where to go, and by acting erratically —such as swinging back-and-forth at a resonant frequency— can cause serious control problems. The question of where and how to model human reactions is unresolved in the Crescendo approach and has received limited attention in the context of co-modelling.

C. New Challenges: well-founded integration via UTP

Even a dependable CPS as simple as the ChessWay illustrates the need for multi-paradigm modelling. In developing dependable CPSs, we have therefore to achieve semantically well-founded integration of models and tools across disciplines and paradigms. We have also identified (and will illustrate in Section V) the need for well-founded integration across whole tool chains. In our current work we are exploring the use of Unifying Theories of Programming (UTP) as a source for the foundations of this integration.

UTP is originally the work of Hoare & He [26], who set out an open-ended research programme to unify the different paradigms for modelling and implementing computer systems. Their motivation was to bridge the gap between the academic and industrial cultures, where academic researchers propose sound theories to underpin system development and industrial practitioners propose pragmatic techniques to develop real systems. Hoare & He's programme was to study the links between all paradigms, both academic and industrial, and in UTP they developed a framework with three orthogonal axes:

1) **Computational Paradigms:** UTP groups modelling notations according to their classification by computational model; for example, this might be object-oriented, concurrent, synchronous, real-time, discrete, continuous or hybrid. The technique used to give semantics to each computational model is to identify common concepts and deal separately with additions and variations. In doing this, UTP exploits uses two of the most important scientific principles: simplicity of presentation and separation of concerns.
2) **Abstraction:** An orthogonal concern involves studying different levels of abstraction in the development process. The highest level is a statement of requirements, whilst the lowest level is the platform-specific technology of an implementation. An idealised development process runs through these levels of abstraction, bridging various semantic gaps to show how the requirements are correctly implemented. Interfaces are specified using contracts to guarantee the correctness of moving a model from one level to another. This mapping between levels is based on a formal notion of refinement that provides guarantees of correctness all the way from requirements to code.
3) **Presentation:** The third classification is by the method chosen to present a language definition. These are the following:
   a) *Denotational*, given by a function from syntax to a single mathematical meaning: its *denotation*. A specification is then just a set of denotations: the permitted behaviours of a system. Refinement is simply inclusion: every behaviour of the program

must also be a behaviour permitted by the specification.

b) *Algebraic*, given by a collection of equations relating descriptions in the language.

c) *Axiomatic*, where the meaning of a command in a program is described by its effect on assertions about the program state. Axiomatic semantics underpins the assertional technique, the most widely used formal method in industry.

d) *Operational*, given by a set of rules describing how the language is executed on an idealised abstract mathematical machine.

As Hoare & He point out, a comprehensive account of constructing systems in any theory needs all four kinds of presentation. The UTP technique allows studies differences and mutual embeddings, and derives each semantics from the others by mathematical definition, calculation, and proof.

A practical and large-scale application of UTP is in the definition of the COMPASS Modelling Language, CML [19], which has been used to develop and verify systems of systems [27]. The approach is to create models of the constituent systems being used, whether they are new systems under development or existing systems. Naturally, a system of systems tends to be composed of semantically heterogeneous constituent systems. CPSs, for example, will have both discrete-time controllers and continuous-time plant; there may may be synchronous hardware and asynchronous message passing over the internet between software components; some hardware components may be deterministic, whilst some software may be stochastic. Within these different paradigms, there may be different levels of abstraction. For example, a socio-technical cyber-physical system may have components that operate at different granularities of time: patients may have courses of treatment that last for several months, while their personally prescribed medicines must be taken on a daily basis and their adaptive pacemaker be accurate to 100ms. Each of these different paradigms and levels of abstraction can be formalised in UTP and the relationships between them can be expressed.

Currently, CML contains a few paradigms relevant to CPS modelling. These are largely "cyber-side" paradigms, but nevertheless they demonstrate the compositional approach:

1) **State-based description.** The theory of *designs* in UTP provides a nondeterministic programming language with pre- and postcondition specifications as contracts. The concrete realisation of this theory is the VDM language with its type system and structuring mechanisms.
2) **Concurrency and communication.** The theory of reactive processes in UTP provides a way of constructing networks of processes that communicate by passing messages. The concrete realisation is the CSP_M language with its rich collection of process combinators.
3) **Object orientation.** The theory of object orientation in UTP is build on top of the theory of designs and provides a way of structuring state-based descriptions through sub-typing, inheritance, and dynamic binding, with mechanisms for object creation, type testing, type casting, and state-component access.
4) **Pointers.** The theory of pointers in UTP provides a way of modelling heap storage and its manipulations, as found in implementations of object orientation. Crucially, it supports modular reasoning about the heap.
5) **Time.** The theory of timed traces in UTP supports the observation of events in discrete time. It is used in a theory of Timed CSP.

Theories of continuous time, probability, and dynamic reconfiguration are all under development.

IV. Exploring the CPS Design Space

A. Support for DSE

A key role of collaborative modelling is to permit systematic exploration of the space of solutions to a given design problem. DSE is the process of building and evaluating co-models in order to reach a design from a set of requirements. in DSE, there are important selection points when design alternatives are selected on the basis of criteria that are important to the developer (e.g. cost, performance). The alternative selected at each point constrains the range of designs that may be viable next steps forward from the current position. Support for DSE permits the selection of a single design from a (possibly large) set of alternatives. Ranges of values for co-model settings and design parameters can be defined before co-simulations are run for each combination of these settings. Results are stored for each simulation and can be analysed. In the Crescendo tools, we call this feature *Automated Co-model Analysis (ACA)*. The simulation results typically report upon multiple objectives such as speed, accuracy and energy consumed and a method for selecting the best designs must be employed. One way is to define a ranking function on which to evaluate designs; another is to compute a non-dominated set of designs to determine the Pareto Optimal front [28].

B. Example: a Wireless ChessWay?

The ChessWay pilot study offered several opportunities for DSE. For example, it was necessary to determine how fast the controllers could run to maintain safe balancing. Additionally, the (higher) frequency for the safety monitor needed to be determined. The monitor intervenes to cut power to the motors in unsafe situations (for example, when the ChessWay leans over too far) and therefore needs to react with sufficient speed to minimise danger to the rider. Also, since the ChessWay has two controllers with diverse sensor inputs, the distribution of functionality was also considered. The design space was explored by sweeping through various controller and monitor frequencies, combined with functionality distributions, to determine the optimal setup.

As mentioned above, the communication between controllers is affected by electromagnetic interference from the motors, causing data to be corrupted. This could be solved in physics by adding more shielding, but this takes space and increases weight. Alternatively, a software solution might

deal with corrupted data by sending it more often between the controllers, or including some form of dead reckoning until updated data comes in. To test the software solution, the ChessWay engineers determined how much lost data the controllers could tolerate and still function correctly. To do this, lossy communications were added to the co-model in Crescendo, with the percentage of lost messages as a parameter. Then a design sweep was performed with increasing amounts of data loss, to find the safe threshold. At the selected controller frequency, the ChessWay controller could handle about 15% message loss (above the limit offered by the existing shielding solution). In fact, the engineers were able to demonstrate that a *wireless* ChessWay would be possible if data loss (due to message collisions, for example) remained under this threshold.

C. New Challenges

DSE is a systematic process, and tool performance remains critical. For example, in the automotive domain, one can imagine having tens or hundreds of thousands of parameters that can be varied in a DSE campaign. The tacit knowledge of engineers and their "gut instinct" is clearly vital in these areas, but there are – at least on the surface – grounds to suppose that formal analysis could help in taming the scale and complexity of the design space to be explored. The properties of interest in DSE may well be extra-functional, such as power consumption or performance measures. Approaches have been proposed to exploit constraint solving in support of static DSE encompassing functional and extra-functional properties [29], while control performance analysis has been explored using co-simulation and DSE [30], and the results combined with other trade-off factors such as monetary cost and energy consumption.

The Crescendo tool enables the user to carry out DSE by sweeping over model parameter values on both DE and CT models [16], and this has been used to explore the design space from an energy perspective [31]. Techniques from test automation, while typically restricted to the discrete-event domain, might also prove valuable in managing DSE.

V. Traceability and Provenance in CPS Design

A. The Role of Traceability and Provenance

The artefacts produced in CPS development will be diverse, covering cyber and physical elements, including requirements statements, models, records of DSE and analysis results, and generated code. Further, they will change over time. In order to understand the ramifications of change, and to obtain the rationale for design decisions, it is necessary to record the semantic relationships between elements of the design set. The goal is to provide traceability, both up and down the development chain, and through time during design evolution. The maintenance of traceability documentation can be labour-intensive and is often dropped under pressure [32]. While many tools support basic traceability links, none of them yet do this automatically [33], and there is limited semantic support. The model management required to support the retrieval of the multiple models and their respective parameters involved in co-simulations is lacking in automated support and is therefore performed manually.

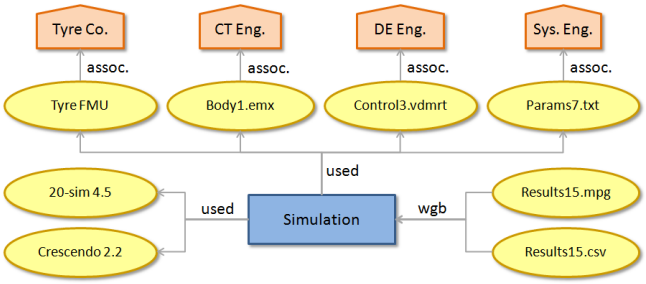

Fig. 2. A fragment of the PROV-N provenance graph for the ChessWay example. It relates results generated by simulation (*wgb*) to the simulation tools used (*used*), input models (*used*) and associated engineers (*assoc.*)

Further motivation for structuring the design set comes from the need for CPS technology in domains in which certification is necessary. Hence, there is a need to record the provenance of designs, and claims about system elements.

B. Example: Changing Independent Suppliers

Traceability is important to maintain the record of CPS development, both internally for mapping requirements to functionality, and externally for certification. Traceability allows arguments of correctness to be rebuilt and determination of how evidence of correctness was produced. For example, imagine that the ChessWay was certified for use on public roads in some jurisdiction. Then a few years later, a ChessWay is involved in an accident. The investigation may look back for the models to see the results that generated the evidence for certification. In which case the correct versions of models have to be retained and found, along with their parameters and also the version of the tools that were used (Fig. 2).

Provenance data can be necessary along with traceability data, when external manufacturers are used as suppliers. For example, consider that the ChessWay tyres would likely be sourced from an external manufacturer, and that that manufacturer provided models for simulation purposes during development. Component suppliers may wish to keep certain trade secrets and thus only be willing to supply models in "black box" form, for instance as a compiled functional mock-up unit (FMU) for use in a functional mock-up interface (FMI) co-simulation. These models can then be used in co-simulation, however sensitive data cannot be gathered. In the case of an investigation into a ChessWay accident, it would be necessary to record the provenance of the data on the tyres, in order to determine on what authority their behaviour is trusted (Fig. 2, top left).

C. New Challenges

Richer models of the design set could capture the relationship between co-models, co-simulations, DSE outcomes,

test information, etc. Features supporting this exist to some extent already in the W3C provenance notation (PROV-N) model [34]. However, we have not yet seen this integrated with a model-based tool chain. While developed for a different domain, PROV-N has been successfully mapped to a graph database that could support large scale developments. From a tool support perspective, graph queries and graph abstractions are necessary to reduce the potentially very large provenance graphs to smaller but still semantically correct versions [35]. Such graph reduction may be required to support information hiding in the context of CPSs with elements procured externally.

VI. Conclusions and Future Work

In this paper, we have argued that there is a significant role for formal methods in the model-based engineering of cyber-physical systems. First, we have emphasised the need for a rigorous approach to semantic heterogeneity. Second, formal techniques have potential to help manage the complexity of exploring the CPS design space, particularly if formal theories can be sufficiently diverse to permit the analysis of extra-functional properties. Third, effective traceability of the complex design set that arises in model-based CPS engineering, and the management of provenance data, benefit from semantically rich descriptions of the relationships between the artefacts produced in design, DSE, test and maintenance.

The goal of our current work[5] is to develop a well-founded tool chain rather than a single "factotum" tool. Using foundations defined using UTP, we aim to create a family of interlinked tools, supporting CPS development from requirements and architectural modelling formalised using SysML, via FMI interface definitions to co-models. The tool chain is intended to permit static analysis of co-models, as well as co-simulation, including co-simulation of models with implementations of cyber and/or physical elements. We aim to allow these co-simulations to be exploited in DSE and test automation. The baseline technologies are Modelio[6] for SysML, co-modelling and co-simulation using VDM Overture, 20-sim[7], and OpenModelica[8]. Co-simulation will build on Crescendo and the TWT co-simulation engine[9], and test automation builds on RT-Tester[10]. We plan to evaluate the framework using applications in railways, agriculture, automotive systems and building automation.

There are ample opportunities for formal methods to play a key role in enabling the cost-effective design of cyber-physical systems. These arise not only because of the need for dependability, but because they enable exploration and management of design spaces. The targeted application of formal techniques, integrated with sound but established development practices, has the potential to deliver significant improvements in this emerging and exciting engineering discipline.

[5]http://into-cps.au.dk/

[6]http://www.modelio.org/

[6]http://overturetool.org/

[7]http://www.20sim.com/

[8]https://www.openmodelica.org/

[9]http://www.twt-gmbh.de/produkte/co-simulationen/co-simulation-framework.html/

[10]http://www.verified.de/products/rt-tester/

Acknowledgment

The work presented here is partially supported by the INTO-CPS project funded by the European Commission's Horizon 2020 programme under grant agreement number 664047.

References

[1] H. Thompson, Ed., *Cyber-Physical Systems: Uplifting Europe's Innovation Capacity*. European Commission Unit A3 - DG CONNECT, December 2013.

[2] M. Broy, "Engineering Cyber-Physical Systems: Challenges and Foundations," in *Complex Systems Design & Management*, M. Aiguier, Y. Caseau, D. Krob, and A. Rauzy, Eds. Springer Berlin Heidelberg, 2013, pp. 1–13. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-34404-6_1

[3] E. A. Lee, "CPS foundations," in *Proceedings of the 47th Design Automation Conference*, ser. DAC '10. New York, NY, USA: ACM, 2010, pp. 737–742.

[4] K. Wan, D. Hughes, K. L. Man, and T. Krilavicius, "Composition Challenges and Approaches for Cyber Physical Systems," in *Networked Embedded Systems for Enterprise Applications (NESEA), 2010 IEEE International Conference on*, 2010, pp. 1–7.

[5] P. Derler, E. A. Lee, and A. Sangiovanni-Vincentelli, "Modeling Cyber-Physical Systems," *Proceedings of the IEEE (special issue on CPS)*, vol. 100, no. 1, pp. 13 – 28, January 2012. [Online]. Available: http://chess.eecs.berkeley.edu/pubs/843.html

[6] I. Horvath and B. H. Gerritsen, "Outlining nine Major Design Challenges of Open, Decentralized, Adaptive Cyber-Physical Systems," in *Proceedings of the ASME 2013 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference IDETC/CIE 2013*, Portland, Oregon, USA, August 2013.

[7] M. Törngren, S. Bensalem, M. V. Cengarle, D.-J. Chen, J. McDermid, R. Passerone, A. Sangiovanni-Vincentelli, and T. Runkler, "CPS State of the Art," EC FP7 Project 611430 CyPhERS, Tech. Rep. Deliverable D5.1, 2014.

[8] "OMG Systems Modeling Language (OMG SysML$^{\text{TM}}$)," SysML Modelling team, Tech. Rep. Version 1.3, June 2012, http://www.omg.org/spec/SysML/1.3/.

[9] Thomas A. Henzinger and Joseph Sifakis, "The Embedded Systems Design Challenge," in *FM 2006: Formal Methods, 14th International Symposium on Formal Methods, Hamilton, Canada, August 21-27, 2006, Proceedings*, 2006, pp. 1–15.

[10] J. Woodcock, P. G. Larsen, J. Bicarregui, and J. Fitzgerald, "Formal Methods: Practice and Experience," *ACM Computing Surveys*, vol. 41, no. 4, pp. 1–36, October 2009.

[11] INCOSE, "Systems Engineering Handbook. A Guide for System Life Cycle Processes and Activities, Version 3.2.2." International Council on Systems Engineering (INCOSE), Tech. Rep. INCOSE-TP-2003-002-03.2.2, October 2011.

[12] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr, "Basic Concepts and Taxonomy of Dependable and Secure Computing," *IEEE Transactions on Dependable and Secure Computing*, vol. 1, pp. 11–33, 2004.

[13] J. Fitzgerald, P. G. Larsen, P. Mukherjee, N. Plat, and M. Verhoef, *Validated Designs for Object–oriented Systems*. Springer, New York, 2005. [Online]. Available: http://www.vdmbook.com

[14] J. van Amerongen, *Dynamical Systems for Creative Technology*. Controllab Products, Enschede, Netherlands, 2010.

[15] J. Fitzgerald, K. Pierce, and P. G. Larsen, *Industry and Research Perspectives on Embedded System Design*. IGI Global, 2014, ch. Collaborative Development of Dependable Cyber-Physical Systems by Co-modelling and Co-simulation.

[16] J. Fitzgerald, P. G. Larsen, and M. Verhoef, Eds., *Collaborative Design for Embedded Systems – Co-modelling and Co-simulation*. Springer, 2014. [Online]. Available: http://link.springer.com/book/10.1007/978-3-642-54118-6

[17] H. Thompson, R. Paulen, M. Reniers, C. Sonntag, and S. Engell, "Analysis of the State-of-the-Art and Future Challenges in Cyber-physical Systems of Systems," EC FP7 project 611115 CPSoS, Tech. Rep. D2.4, February 2015. [Online]. Available: http://www.cpsos.eu

[18] J. Fitzgerald, J. Bryans, P. G. Larsen, and H. Salim, "Collaborative systems of systems need collaborative design," in *PRO-VE 2014 – 15th Working Conference on Virtual Enterprises*, October 2014.

[19] J. Fitzgerald, P. G. Larsen, and J. Woodcock, "Foundations for Model-based Engineering of Systems of Systems," in *Complex Systems Design and Management*, M. A. et al., Ed. Springer, January 2014, pp. 1–19.

[20] J. Sztipanovits, X. Koutsoukos, G. Karsai, N. Kottenstette, P. Antsaklis, V. Gupta, B. Goodwine, J. Baras, and S. Wang, "Toward a Science of Cyber-Physical System Integration," *Proceedings of the IEEE*, vol. 100, no. 1, pp. 29–44, 2012.

[21] X. Zheng, C. Julien, M. Kim, and S. Khurshid, "On the State of the Art in Verification and Validation in Cyber Physical Systems," The University of Texas at Austin, The Center for Advanced Research in Software Engineering, Tech. Rep. TR-ARiSE-2014-001, 2014.

[22] M. Sanwal and O. Hasan, "Formal Verification of Cyber-Physical Systems: Coping with Continuous Elements," in *Computational Science and Its Applications – ICCSA 2013*, ser. Lecture Notes in Computer Science, B. Murgante, S. Misra, M. Carlini, C. Torre, H.-Q. Nguyen, D. Taniar, B. Apduhan, and O. Gervasi, Eds. Springer Berlin Heidelberg, 2013, vol. 7971, pp. 358–371.

[23] J. Davis, R. Galicia, M. Goel, C. Hylands, E. Lee, J. Liu, X. Liu, L. Muliadi, S. Neuendorffer, J. Reekie, N. Smyth, J. Tsay, and Y. Xiong, "Ptolemy-II: Heterogeneous concurrent modeling and design in Java," University of California at Berkeley, Technical Memorandum UCB/ERL No. M99/40, July 1999.

[24] S. S. Phatak, D. McCune, and G. Saikalis, "Cyber Physical System: A Virtual CPU Based Mechatronic Simulation," in *5th IFAC Symposium on Mechatronic Systems*, G. T.-C. Chiu and K. Youcef-Toumi, Eds., IFAC. Elsevier, 2010, pp. 405–410.

[25] A. Canedo, M. A. A. Faruque, and J. Richter, "Multi-disciplinary integrated design automation tool for automotive cyber-physical systems," in *IEEE/ACM Design Automation and Test in Europe (DATE'14), Dresden, Germany*, 2014, pp. 1–2.

[26] T. Hoare and H. Jifeng, *Unifying Theories of Programming*. Prentice Hall, April 1998.

[27] J. Woodcock, "Engineering UToPiA - Formal Semantics for CML," in *FM 2014: Formal Methods*, ser. Lecture Notes in Computer Science, C. Jones, P. Pihlajasaari, and J. Sun, Eds., vol. 8442. Springer, 2014, pp. 22–41.

[28] K. Deb, *Multi-objective optimization using evolutionary algorithms*. John Wiley & Sons, 2001, vol. 16.

[29] B. Hockner, P. Hofstedt, S. Kaltschmidt, P. Sauer, and T. Vörtler, "Design space exploration for cyber physical system design using constraint solving," in *Proceedings of the 2013 Forum on specification and Design Languages, FDL 2013, Paris, France, September 24–26, 2013*. IEEE, 2013, pp. 1–4.

[30] N. Mühleis, M. Glaß, L. Zhang, and J. Teich, "A co-simulation approach for control performance analysis during design space exploration of cyber-physical systems," *SIGBED Review*, vol. 8, no. 2, pp. 23–26, 2011.

[31] J. A. E. Isasa, P. W. Jørgensen, and P. G. Larsen, "Hardware In the Loop for VDM-Real Time Modelling of Embedded Systems," in *MODELSWARD 2014, Second International Conference on Model-Driven Engineering and Software Development*, January 2014.

[32] M. Jarke, "Requirements tracing," vol. 41, no. 12, pp. 32–36, December 1998.

[33] P. Mäder, *Rule-based Maintenance of Post-requirements Traceability*, ser. MV Wissenschaft. MV-Verlag, 2010.

[34] "PROV-DM: The PROV Data Model," World Wide Web Consortium, Tech. Rep., 2012. [Online]. Available: http://www.w3.org/TR/prov-dm/

[35] P. Missier, J. Bryans, C. Gamble, V. Curcin, and R. Danger, "ProvAbs: model, policy, and tooling for abstracting PROV graphs," in *Procs. IPAW 2014 (Provenance and Annotations)*. Springer, 2014.