Features of Integrated Model-based Co-modelling and Co-simulation Technology

Peter Gorm Larsen[1], John Fitzgerald[2], Jim Woodcock[3], Carl Gamble[2], Richard Payne[2], and Kenneth Pierce[2]

[1] Department of Engineering, Aarhus University, Denmark , pgl@eng.au.dk
[2] School of Computing Science, Newcastle University, UK
john.fitzgerald@ncl.ac.uk
[3] Department of Computer Science, University of York, jim.woodcock@york.ac.uk

Abstract. Given the considerable ongoing research interest in collaborative multidisciplinary modelling and co-simulation, it is worth considering the features of model-based techniques and tools that deliver benefits to cyber-physical systems developers. The European project "Integrated Tool Chain for Model-based Design of Cyber-Physical Systems" (INTO-CPS) has developed a well-founded tool chain for CPS design, based on the Functional Mock-up Interface standard, and supported by methodological guidance. The focus of the project has been on the delivery of a sound foundation, an open chain of compatible and usable tools, and a set of accessible guidelines that help users adapt the technology to their development needs.

Keywords: Co-Simulation, CPS Engineering, Tool chain, methodology, foundations

1 Introduction

In Cyber-Physical Systems (CPSs), computing and physical processes interact closely. Their effective design therefore requires methods and tools that bring together the products of diverse engineering disciplines. Without such tools it would be difficult to gain confidence in the system-level consequences of design decisions made in any one domain, and it would be challenging to manage trade-offs between them. How, then, can we support such multidisciplinary design with semantically well-founded approaches in a cost-effective manner?

In the INTO-CPS project we start from the view that disciplines such as software, mechatronic and control engineering have evolved notations and theories that are tailored to their needs, and that it is undesirable to suppress this diversity by enforcing uniform general-purpose models [16,31] Our goal is to achieve a practical integration of diverse formalisms at the semantic level, and to realise the benefits in integrated tool chains. In order to demonstrate that the technology works industrially it has been applied in very different application domains (e.g.,[17,30,19,12,34,38]).

To the CPS engineer, the system of interest includes both computational and physical elements, so the foundations, methods and tools of CPS engineering should incorporate both the Discrete-Event (DE) models of computational processes, and the continuous-value and Continuous-Time (CT) formalisms of physical dynamics engineering. Our approach is to support the development of collaborative models (*co-models*) containing DE and CT elements expressed in diverse notations, and to support their analysis by means of co-simulation based on a reconciled operational semantics of the individual notations' simulators [18]. This enables exploration of the design space and allows relatively straightforward adoption in businesses already exposed to some of these tools and techniques. The idea is to enable co-simulation of extensible groups of semantically diverse models, and at the same time the semantic foundations are extended using Unifying Theories of Programming (UTP) to permit analysis using advanced meta-level tools that are primarily targeted towards academics and thus not considered as a part of the industrial INTO-CPS tool chain.

Given the considerable interest in model-based CPS engineering, we believe that it is useful to consider what the *Unique Selling Points* (USPs) are for integrated tool chains. In this paper, we first provide an overview of what we consider the main USPs of the INTO-CPS technology from the perspective of industry use (Section 2). We then describe the open INTO-CPS tool chain (Section 3). In order to realise the benefits of the tools it is important to develop guidance for their use in collaborative modelling, and this is described in Section 4. We discuss our approach to providing integrated semantic foundations needed to underpin such co-modelling (Section 5) before looking forward (Section 6).

2 The Unique Selling Points

In our work on INTO-CPS, we have sought to deliver the following distinctive features, relevant to the industrial use of co-modelling and co-simulation technology. We see the main USPs as:

1. **Faster route to market for engineering CPSs:** In a highly active CPS marketplace, getting the right solution first time is essential. We believe that the interoperability of tools in the INTO-CPS tool suite enables a more agile close collaboration between stakeholders with diverse disciplinary backgrounds.
2. **Avoiding vendor lock-in by open tool chain:** Some commercial solutions provide at least a part of the functionality provided by the INTO-CPS tool chain with a high level of interoperability. However, in particular for Small

and Medium-sized Enterprises (SMEs), there is a risk of being restricted in the choice of specialist tools.

3. **Exploring large design spaces efficiently:** CPS design involves making design decisions in both the cyber and physical domains. Trade-off analysis can be challenging. Co-simulation enables the systematic exploration of large design spaces in the search for optimal solutions.
4. **Limiting expensive physical tests:** CPS development often relies on the expensive production and evaluation of a series of physical prototypes. Co-simulation enables users to focus on testing different models of CPS elements in a virtual setting, gaining early assessment of CPS-level consequences of design decisions.
5. **Enabling traceability for all project artefacts:** In both documenting the coverage and quality of analysis and in managing the consequences of design change, there is a need to support the maintenance of traceable links between the many diverse artifacts produced during CPS development. We have sought to provide a basis for delivering levels of design traceability.

Tools as described in Section 3 will not, on their own, deliver these features. Methods guidance is needed to ensure that users get the greatest benefit from integrating co-modelling in their own development contexts. Firm semantic foundations are required in order to build confidence in the analyses that they deliver. To these ends, we have worked on methodological and semantic integration, discussed in Sections 4 and Section 5, respectively.

3 The INTO-CPS Tool Chain

We have developed an open *integrated tool chain* to allow n-ary co-simulation of a wider range of model types. In order to facilitate this, we have developed an extensible semantic foundation using UTP. Figure 1 gives a graphic overview of the tool chain, which has been developed in the INTO-CPS project.

In the INTO-CPS tool chain, requirements and CPS architectures may be expressed using SysML. We have defined a special CPS SysML profile that allows cyber and physical system elements to be identified such that each of these elements corresponds to a constituent model [3,1]. From each element, we generate an interface following the Functional Mockup Interface (FMI) standard[4]. In our approach the tools in which the constituent models are developed can then import these interfaces and export conformant executable *Functional Mockup Units (FMUs)* following version 2.0 of the FMI standard for co-simulation.

[4] FMI essentially defines a standardised interface to be used in computer simulators to develop complex CPSs.

Heterogeneous system models can be built around the FMI interfaces, permitting these heterogeneous *multi-models* to be co-simulated, and to allow static analysis, including model checking (of appropriate abstractions). A Co-simulation Orchestration Engine (COE) manages the co-simulation of multi-models and is built by combining existing co-simulation solutions. The COE has also been used with FMUs produced with other tools including Modelon[5], Dymola[6], 4diac[7] and SimulationX[8]. In addition a special 3D FMU capability has been enabled using the Unity game engine[9]. This also enables incorporation of 3D glasses such as Oculus Rift enabling a special experience with new CPSs in a virtual setting, before the CPSs are implemented[10]. The COE permits hardware-in-the-loop and software-in-the-loop analysis [19] and it is possible to use it in a distributed fashion. Thus, interoperability in relation to simulation of complex models of CPSs divided up in constituent models expressed using different formalisms and different tools is ensured. This is an important part of USP 1.

Results of multiple co-simulations can be collated, permitting systematic Design Space Exploration (DSE), and allowing test automation based on test cases generated from the SysML requirement diagrams [39]. The ability of both carrying out exploratory experiments as well as systematic testing a combination of constituent models leads to USP 3 since these features enable exploration of very large candidate design spaces. In particular the ability of visualising the results of co-simulations using the 3D FMU described above with the DSE capability (described further in Section 4.3) leads to USP 4 limiting the number of physical tests that needs to be carried out, which in particular is important whenever these are expensive to carry out or difficult to monitor the results of.

CPS SysML profile has been demonstrated in Modelio[11]. FMI-conformant constituent models have been produced in Overture from VDM-RT, and the Continuous-Time (CT) formalisms 20-sim and OpenModelica [23][12]. A graphical front-end for the entire INTO-CPS tool chain called the INTO-CPS Application has been developed based on the cross-platform Electron technology [14]. This is developed as a desktop application, but using web technologies to enable a smoother transition to delivering this as an on-line web service should this be desirable in the future.

[5] `http://www.modelon.com/`.
[6] `https://www.3ds.com/products-services/catia/products/dymola`.
[7] `https://eclipse.org/4diac/`.
[8] `https://www.simulationx.com/`.
[9] `https://unity3d.com/`.
[10] `https://www.oculus.com/rift/`.
[11] `http://www.modelio.org/`
[12] `https://www.openmodelica.org/`

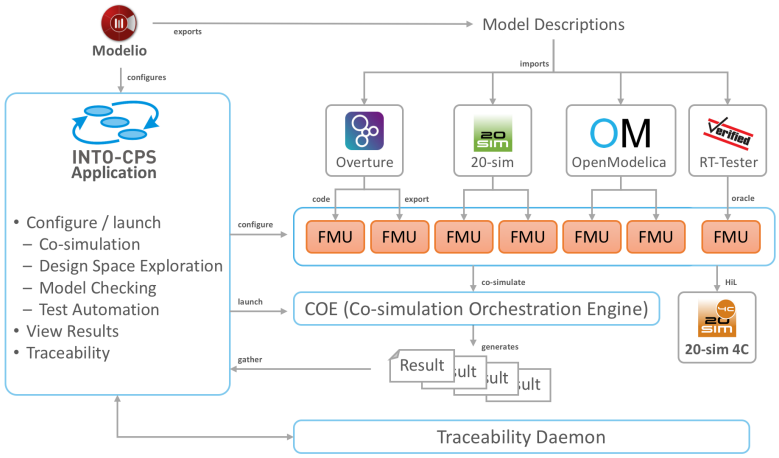

Fig. 1. The INTO-CPS Tool Chain

Multidisciplinary co-modelling and co-simulation naturally generate many design artifacts, including co-models, co-simulation inputs and outputs, requirements, code, etc. Such large design sets are expected to evolve as smart systems are developed by gradual integration of existing elements, and as elements change. The interrelationships between them are vital for allowing validation and third-party assurance. We have used the PROV[13] model to record the temporal relations between activities, entities and agents within a process (which we term *provenance*), and traceability has been supplied based on the Open Services for Lifecycle Collaboration (OSLC) standard[14]. In INTO-CPS, we have regarded it as a priority to lay foundations for provenance and traceability support in the tool chain; all the baseline tools have been extended with such OSLC support [32]. The openness resulting from the combination of FMI and OSLC contributes significantly to USP 2, giving freedom to choose the tools that best fit the purpose of each individual aspect of a CPS.

4 The INTO-CPS Methodology

Our work on methods aims to develop concrete guidelines, frameworks, and patterns for co-modelling and co-simulation that can be adapted to existing development processes, rather than defining a single workflow. Specifically, we focus on model-based CPS requirements engineering, architectural modelling in SysML, traceability and provenance, and DSE.

[13] http://www.w3.org/TR/prov-overview/
[14] http://open-services.net/

We support model-based systems engineering approaches because of their potential to enable early detection of potential bottlenecks and defects before commitment is made to physical prototypes (USPs 1 and 4). We advocate the use of *architectural models* that define the major elements of a system, their relationships and interactions. The bulk of our architectural modelling work uses SysML, which allows us to describe both digital interfaces between components in terms of the properties or functionality *provided* or *required*, and physical interfaces in terms of physical flows (e.g. material) between components.

An *architecture diagram* is a symbolic representation of part of an architectural model. An *architectural view* is typically an architecture diagram that includes specific system facets. An *architectural framework* is a set of architectural views defined to support a task, role, or industry [26]. A SysML *profile* is a collection of extensions to SysML that support a particular domain. We have developed a SysML profile consisting of diagrams for defining *cyber*, *physical* components, as well as components representing *environment* and *visualisation* elements (the delivering graphical presentation of co-simulation outputs).

4.1 Requirements Engineering

CPSs share important characteristics with Systems-of-Systems (SoSs) [35]. Cyber and physical elements can be independently owned and managed, evolve over time, and are distributed [40]. CPSs add the challenge of differing domain contexts [47]. In developing model-based requirements engineering approaches for CPSs, we have therefore extended a systematic approach to SoS requirements engineering, the *SoS Approach to Context-based Requirements Engineering(SoS-ACRE)* [27]. SoS-ACRE provides several views that encourage the systematic consideration of requirement context, sources and stakeholders.

A survey of our industry collaborators showed that a wide range of tools were used for requirements management and modelling, ranging from Microsoft Word to IBM Rational DOORS. It was therefore important to develop an approach that, while it could be supported by specialist notations like SysML, could also be adopted using document-based tools. This is a key facet of providing an open tool chain (USP 2). For example, SoS-ACRE can be adapted to these CPS needs as follows:

Source Element View (SEV) which identifies the sources from which requirements are derived. This could be represented as a SysML block definition diagram, an Excel table or a Word document, or by simply referring to source documents using OSLC traces.

Requirement Description View (RDV) defines the requirements. This could be a SysML requirements diagram, or in tabular form, or in DOORS.

Context Definition View (CDV) identifies interested stakeholders and points of context, including customers, suppliers and system engineers themselves. These could be SysML block definition diagrams, Excel tables or Word documents, and can be used to identify the CT/DE elements of a system.

Requirement Context View (RCV) defined for each constituent system context identified in CDVs. A *Context Interaction View (CIV)* is then defined to understand the overlap of contexts and any common/conflicted views on requirements. In SoS-ACRE, RCVs and CIVs are both defined with SysML use case diagrams. Excel could be used if unique identifiers are defined for contexts and requirements as described earlier.

Given that we aim to support the integration of co-simulation into established development processes, we realise the SoS-ACRE views using a range of combinations of SysML with other tools. For example, a single SysML model for both requirements engineering and architectural modelling will contain all SoS-ACRE views (SEV, RDV, CDV, RCV and CIV), in addition to diagrams defined using the INTO-CPS profile for the CPS composition and connections. Modelling in this way enables trace links to be defined inside a single SysML model, using `<<trace>>` relationships (e.g., Figure 2). By contrast, one might combine URIs for the source elements with an Excel document for the RDV, CDV, RCV and CIV. As above, SysML can be used to define the architecture in a single model. Trace links using OSLC may then be used to link source elements, rows of Excel documents (with internal tracing using unique identifiers referenced between sheets), and architectural elements of the SysML architectural model. Figure 3 presents an example with URI, Excel and SysML models and OSLC links between the artifacts.

4.2 Traceability and Provenance

USP 5 deals with the need to support engineers navigating the complexities of CPS design sets. INTO-CPS considers two tool-supported methods for recording design rationale. *Traceability* associates one model element (e.g. requirements, design artifacts, activities, software code or hardware) to another. *Requirements traceability* "refers to the ability to describe and follow the life of a requirement, in both a forwards and backwards direction" [24]. *Provenance* "is information about entities, activities, and people involved in producing a piece of data or thing, which can be used to form assessments about its quality, reliability or trustworthiness" [33].

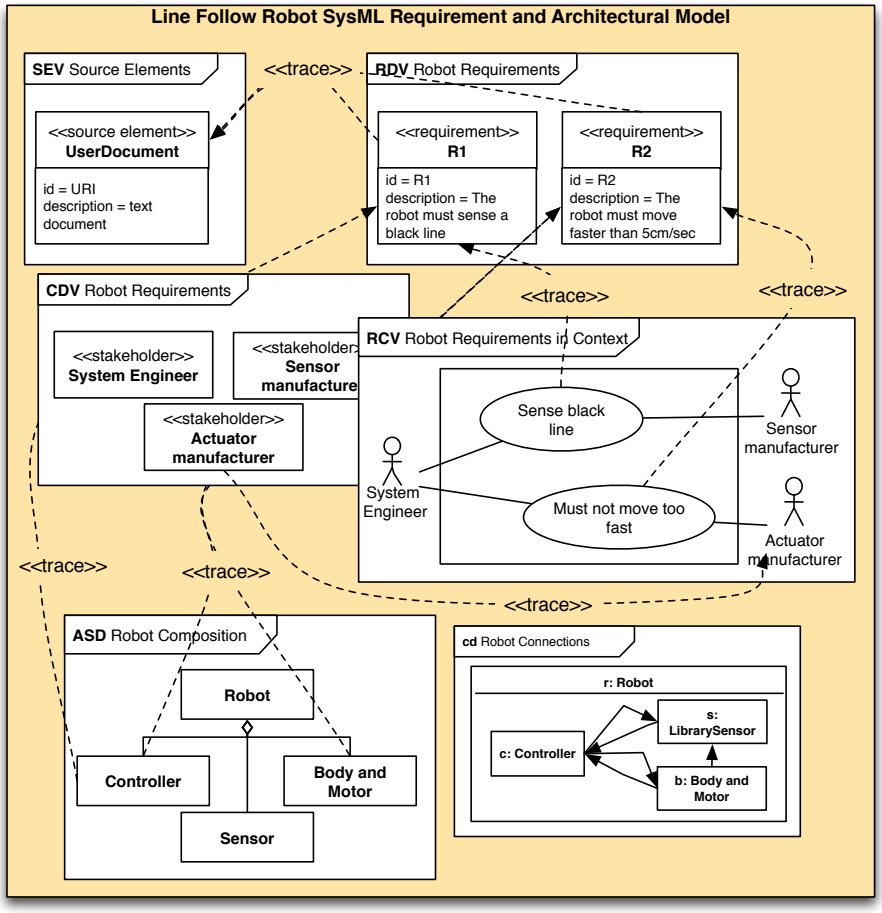

Fig. 2. Single SysML model – model overview

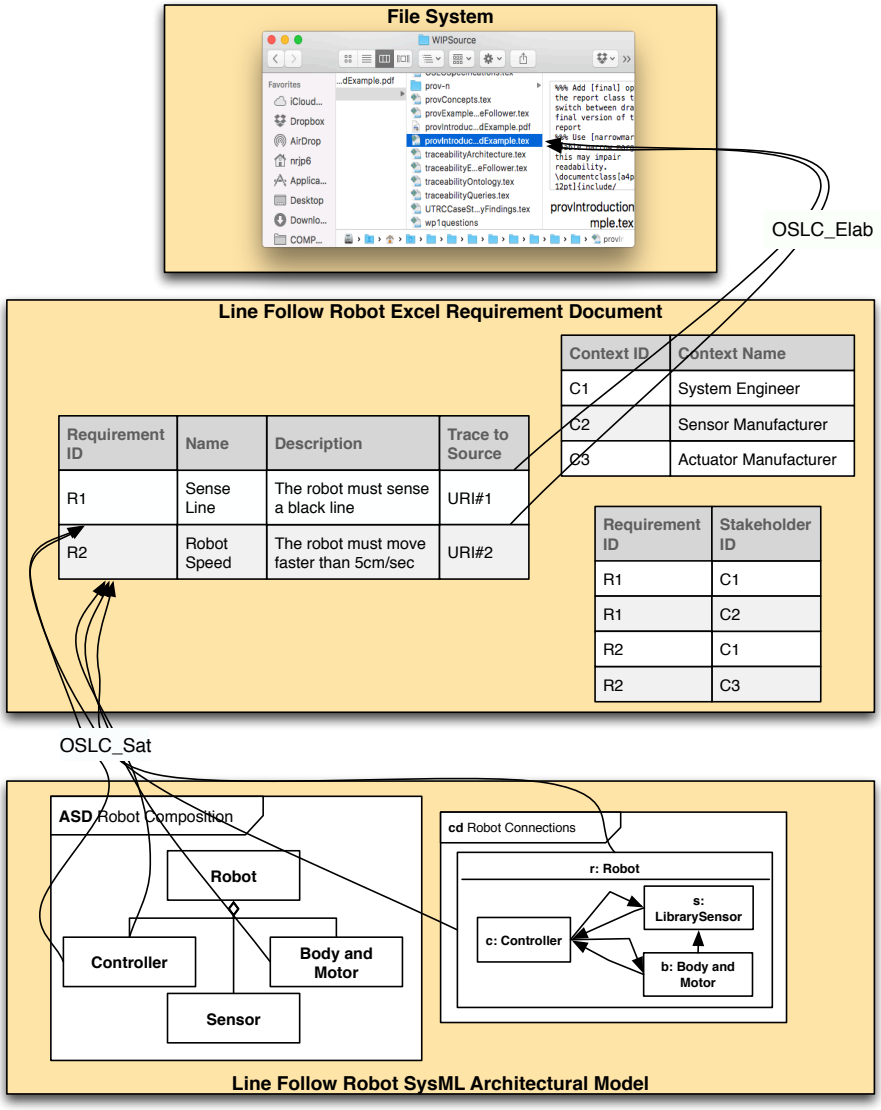

Fig. 3. URI, Excel and SysML – model overview

4.3 Design Space Exploration

USP 3 is the ability to sweep over the design space to identify optimal combinations of parameters with respect to evaluation criteria. A *design parameter* is a property of a model that varies the behaviour described, but remains constant during a simulation; a *variable* is a property that may change during a simulation. Where two or more models represent different solutions to the same problem, these are considered *design alternatives*. In INTO-CPS, design alternatives are defined using either a range of parameter values or different multi-models.

Designing DSE experiments can be complex and depends closely on the multi-model being analysed. Engineers need, therefore, to be able to model at an early stage of design how the experiments relate to the model architecture, and where possible trace from requirements to the analysis experiments. We have defined a SysML profile for modelling DSE experiments in a consistent and traceable way. The profile comprises five diagrams for defining *parameters*, *objectives* and *rankings*. Based on a requirements analysis (e.g. an RDV coming out of the processes described in Section 4.1), we identify objectives, and use the SysML profile for DSE to define the parameters, objectives and ranking function, traced to the requirements.

We use the INTO-CPS tool chain to simulate each design alternative. Whilst exhaustively seaching the design space, evaluating and ranking each design is acceptable on small-scale studies, it quickly becomes infeasible as the design space grows. For example, varying n parameters with m alternative values produces a design space of m^n alternatives. We have therefore implemented genetic approaches [15].

5 The Underlying Unified Semantic Approach

Since CPSs are networks of computational devices interacting with the world through their sensors and actuators, CPS models must combine discrete computational models with continuous physical environmental models. CPS engineering necessarily involves a wide range of modelling and programming paradigms [13], including concurrency, real-time, mobility, continuous variables, differential equations, object orientation, and diagrammatic languages. Practical CPS engineering uses a variety of domain-specific and general-purpose languages, such as Simulink, Modelica, SysML, Java, and C, and engineering trustworthy CPS requires that semantic models for these languages are integrated in a consistent way, which then enables reasoning about an entire CPS[15].

[15] For SysML we have only formalised the subset we need in a co-simulation setting [1].

In practice, semantic integration is often achieved using the FMI standard [6], mentioned above, which describes a CPS using a network of FMUs to simulate components and their solvers and simulators. Each FMU has observable discrete and continuous variables that can be observed and modified, as well as an interface to drive the simulation engine in various ways. In a co-simulation a master algorithm manages stepping the individual FMUs forward, and distributing information in between time steps. In this way, FMI describes heterogeneous multi-models in different notations with different underlying semantics integrated through a common operational interface[16].

FMI provides a way of experimenting with the combined operational interfaces to heterogeneous models, and so it is useful for validation; but it does not provide the basis for verifying CPS properties. To do that, we need to be able to verify properties, both at the model level and at the multi-model level. One way of doing this is to explore the links between the different semantics. To do this, we use Hoare and He's *Unifying Theories of Programming* (UTP) [25,48,10] to describe different computing paradigms and their formal connections. We treat the various semantic aspects of a heterogeneous multi-model as individual theories that characterise a particular abstract modelling paradigm. Hoare & He [25] show how the mathematics of the alphabetised relational calculus can be used to construct a hierarchy of such theories, including an assertional approach to hybrid imperative parallel programming and control of continuous physical phenomena. Within this hierarchy, there are theories of real-time programming [45], object-oriented programming [43], security and confidentiality [4], mobile processes [44], probabilistic modelling [7], and hybrid systems [20]. The FMI API itself has been given a UTP-based semantics [11,49] that can be used as an interface to the semantic model of individual FMUs.

Our approach to practical CPS verification in the meta-tools is based on the theorem prover for UTP built on Isabelle/HOL [36], which we call *Isabelle/UTP* [22,21]. Isabelle is a powerful automated proof assistant that was used, for example, in the seL4 microkernel verification project [29]. Isabelle include recent work on formalising the integral and differential calculus, real analysis, and Ordinary Differential Equations (ODEs) [28], work that we are applying to verification of hybrid systems[17].

Crucial to all of these developments is the ability to integrate external tools into Isabelle that can provide decision procedures for specific classes of problems. Isabelle is well suited to such integrations due to its architecture based on the ML and Scala programming languages, both of which can be used to

[16] FMI-based co-simulation with a black-box approach does have limitations [8,9] and we do not claim to repair those issues in any way in this work.

[17] This library can be viewed at `github.com/isabelle-utp/utp-main`.

implement plugins. Isabelle is sometimes referred to as the *Eclipse* of theorem provers [46]. The `sledgehammer` tool [5], for example, integrates a host of first-order automated theorem provers and SMT solvers, which often shoulder the burden of proof effort. `Sledgehammer` has been used both at the theory engineering level, for constructing an algebraic hierarchy of verification logics (Kleene algebras), and also at the verification level, where it is used to discharge first-order proof obligations [2]. For verification of hybrid systems, it is necessary to integrate Isabelle with computer algebra systems like Mathematica, MATLAB, and SageMath, to provide solutions to differential equations, an approach that has been previously well used by the KeYmaera tool [41,42].

Our vision is the use of Isabelle and UTP to provide the basis for CPS verification through formalisation of the fundamental building-block theories of CPS multi-modelling, and the integration of tools that implement these theories for coordinated verification.

6 Concluding Remarks

Integrated modelling such as that presented in this paper is essential to efficient engineering of CPSs. We believe that openness of the tool chain using different standards, the methodology supporting it and the underlying unified semantic approach jointly enable stakeholders with different disciplinary backgrounds to collaborate in the development of CPSs. This is by no means the only scientific approach by which such systems can be developed, but we think that it is a promising candidate that has future extension possibilities as well.

In the current version of the FMI standard there are a number of limitations. This includes that it is not able to cope with the modelling of the network communication in a natural manner and it is incapable of modelling dynamic reconfigurations. Thus, in a future extension it would be ideal if it would be possible to enable such capabilities for example to be able to appropriately model and develop constituents with their own independent behaviour. This could mean integration of machine learning capabilities as well as software agents including potentially incorporation of human-in-the-loop. Initial work with humans included have already started [37] but it is easy to imagine that it would be advantageous to combine this further in the future with human centered design ideas. We envisage great capabilities for the future that would bring additional USPs to the table.

Acknowledgments. 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. We would like to thank all the participants of those projects for their efforts making this a reality.

References

1. Amálio, N., Payne, R., Cavalcanti, A., Woodcock, J.: Checking SysML Models for Co-Simulation. Lecture Notes in Computer Science, Springer (2016)
2. Armstrong, A., Gomes, V., Struth, G.: Building program construction and verification tools from algebraic principles. Formal Aspects of Computing 28(2), 265–293 (2015)
3. Bagnato, A., Brosse, E., Quadri, I., Sadovykh, A.: The INTO-CPS Cyber-Physical System Profile. In: DeCPS Workshop on Challenges and New Approaches for Dependable and Cyber-Physical System Engineering Focus on Transportation of the Future. Vienna, Austria (June 2017)
4. Banks, M., Jacobs, J.: Unifying theories of confidentiality. In: UTP. LNCS, vol. 6445, pp. 120–136. Springer (2010)
5. Blanchette, J.C., Bulwahn, L., Nipkow, T.: Automatic proof and disproof in Isabelle/HOL. In: FroCoS. LNCS, vol. 6989, pp. 12–27. Springer (2011)
6. Blochwitz, T., Otter, M., Arnold, M., Bausch, C., Elmqvist, H., Junghanns, A., Mauss, J., Monteiro, M., Neidhold, T., Neumerkel, D., Olsson, H., Peetz, J.V., Wolf, S., Clau, C.: The Functional Mockup Interface for tool independent exchange of simulation models. In: Proceedings of the 8th International Modelica Conference. pp. 105–114 (2011)
7. Bresciani, R., Butterfield, A.: A UTP semantics of pGCL as a homogeneous relation. In: Proc. 9th Intl. Conf. on Integrated Formal Methods (iFM). LNCS, vol. 7321, pp. 191–205. Springer (2012)
8. Broman, D., Brooks, C., Greenberg, L., Lee, E.A., Masin, M., Tripakis, S., Wetter, M.: Determinate Composition of FMUs for Co-Simulation. In: 13th International Conference on Embedded Software (EMSOFT), Montreal (September 2013), `http://chess.eecs.berkeley.edu/pubs/1002.html`
9. Broman, D., Greenberg, L., Lee, E.A., Masin, M., Tripakis, S., Wetter, M.: Requirements for hybrid cosimulation standards. In: Proceedings of 18th ACM International Conference on Hybrid Systems: Computation and Control (HSCC). pp. 179–188. ACM (2015)
10. Cavalcanti, A., Woodcock, J.: A tutorial introduction to CSP in *Unifying Theories of Programming*. In: Cavalcanti, A., Sampaio, A., Woodcock, J. (eds.) Refinement Techniques in Software Engineering, First Pernambuco Summer School on Software Engineering, PSSE 2004, Recife, Brazil, November 23–December 5, 2004, Revised Lectures. Lecture Notes in Computer Science, vol. 3167, pp. 220–268. Springer (2006)
11. Cavalcanti, A., Woodcock, J., Amálio, N.: Behavioural models for FMI co-simulations. In: ICTAC (2016), in press
12. Couto, L.D., Basagianis, S., Mady, A.E.D., Ridouane, E.H., Larsen, P.G., Hasanagic, M.: Injecting Formal Verification in FMI-based Co-Simulation of Cyber-Physical Systems. In: The 1st Workshop on Formal Co-Simulation of Cyber-Physical Systems (CoSim-CPS). Trento, Italy (September 2017)
13. Derler, P., Lee, E.A., Sangiovanni-Vincentelli, A.: Modeling cyber-physical systems. Proceedings of the IEEE (special issue on CPS) 100(1), 13–28 (January 2012)
14. The Electron website. `http://electron.atom.io/` (2016)
15. Fitzgerald, J., Gamble, C., Lam, B.: Exploring the Cyber-Physical Design Space. In: Proc. INCOSE Intl. Symp. on Systems Engineering. Adelaide, Australia (July 2017)
16. Fitzgerald, J., Gamble, C., Larsen, P.G., Pierce, K., Woodcock, J.: Cyber-Physical Systems design: Formal Foundations, Methods and Integrated Tool Chains. In: FormaliSE: FME Workshop on Formal Methods in Software Engineering. ICSE 2015, Florence, Italy (May 2015)
17. Fitzgerald, J., Gamble, C., Payne, R., Larsen, P.G., Basagiannis, S., Mady, A.E.D.: Collaborative model-based systems engineering for cyber-physical systems, with a building automation case study. INCOSE International Symposium 26(1), 817–832 (2016)

18. Fitzgerald, J., Larsen, P.G., Verhoef, M. (eds.): Collaborative Design for Embedded Systems – Co-modelling and Co-simulation. Springer (2014), http://link.springer.com/book/10.1007/978-3-642-54118-6
19. Foldager, F., Larsen, P.G., Green, O.: Development and Verification of Controller Design for Autonomous Robots using Co-Simulation of CPS. In: 1st Workshop on Formal Co-Simulation of Cyber-Physical Systems. Trento, Italy (September 2017)
20. Foster, S., Thiele, B., Cavalcanti, A., Woodcock, J.: Towards a UTP semantics for Modelica. In: Proc. 6th Intl. Symp. on Unifying Theories of Programming (June 2016), to appear
21. Foster, S., Zeyda, F., Woodcock, J.: Isabelle/UTP: A mechanised theory engineering framework. In: Proc. 5th Intl. Symp. on Unifying Theories of Programming. LNCS, vol. 8963, pp. 21–41. Springer (2014)
22. Foster, S., Zeyda, F., Woodcock, J.: Unifying heterogeneous state-spaces with lenses. In: Proc. 13th Intl. Conf. on Theoretical Aspects of Computing (ICTAC). LNCS, vol. 9965. Springer (2016)
23. Fritzson, P.: Principles of Object-Oriented Modeling and Simulation with Modelica 2.1. Wiley-IEEE Press (January 2004)
24. Gotel, O.C., Finkelstein, A.C.: An analysis of the requirements traceability problem. In: Proc. 1st Intl. Conf. on Requirements Engineering. pp. 94–101 (April 1994)
25. Hoare, T., Jifeng, H.: Unifying Theories of Programming. Prentice Hall (April 1998)
26. Holt, J., Ingram, C., Larkham, A., Stevens, R.L., Riddle, S., Romanovsky, A.: Convergence report 3. Tech. rep., COMPASS Deliverable, D11.3 (September 2014)
27. Holt, J., Perry, S., Payne, R., Bryans, J., Hallerstede, S., Hansen, F.O.: A model-based approach for requirements engineering for systems of systems. IEEE Systems Journal 9(1), 252–262 (2015), http://dx.doi.org/10.1109/JSYST.2014.2312051
28. Immler, F.: Formally verified computation of enclosures of solutions of ordinary differential equations. In: Proc. 6th NASA Formal Methods Symposium (NFM). LNCS, vol. 8430. Springer (2014)
29. Klein, G., et al.: seL4: Formal verification of an OS kernel. In: Proc. 22nd Symp. on Operating Systems Principles (SOSP). pp. 207–220. ACM (2009)
30. Larsen, P.G., Fitzgerald, J., Woodcock, J., Lecomte, T.: Trustworthy Cyber-Physical Systems Engineering, chap. Chapter 8: Collaborative Modelling and Simulation for Cyber-Physical Systems. Chapman and Hall/CRC (September 2016), iSBN 9781498742450
31. Larsen, P.G., Fitzgerald, J., Woodcock, J., Nilsson, R., Gamble, C., Foster, S.: Towards semantically integrated models and tools for cyber-physical systems design. In: Margaria, T., Steffen, B. (eds.) Leveraging Applications of Formal Methods, Verification and Validation, Proc 7th Intl. Symp. Lecture Notes in Computer Science, vol. 9953, pp. 171–186. Springer International Publishing (2016)
32. Mengist, A., Pop, A., Asghar, A., Fritzson, P.: Traceability support in openmodelica using open services for lifecycle collaboration (oslc). In: Kofránek, J., Casella, F. (eds.) Proceedings of the 12th International Modelica Conference. pp. 823–830. Modelica Association and Linköping University Electronic Press (May 15-17 2017)
33. Moreau, L., Groth, P.: PROV-Overview. Tech. rep., World Wide Web Consortium (2013), http://www.w3.org/TR/2013/NOTE-prov-overview-20130430/
34. Neghina, M., Zamrescu, C.B., Larsen, P.G., Lausdahl, K., Pierce, K.: A Discrete Event-First Approach to Collaborative Modelling of Cyber-Physical Systems. In: Fitzgerald, Tran-Jørgensen, Oda (ed.) Submitted to the 15th Overture Workshop: New Capabilities and Applications for Model-based Systems Engineering. Newcastle, UK (September 2017)
35. Nielsen, C.B., Larsen, P.G., Fitzgerald, J., Woodcock, J., Peleska, J.: Model-based engineering of systems of systems. ACM Computing Surveys 48(2) (September 2015), http://dl.acm.org/citation.cfm?id=2794381

36. Nipkow, T., Wenzel, M., Paulson, L.C.: Isabelle/HOL: A Proof Assistant for Higher-Order Logic, LNCS, vol. 2283. Springer (2002)
37. Palmieri, M., Bernardeschi, C., Masci, P.: Co-simulation of semi-autonomous systems: the Line Follower Robot case study. In: 1st Workshop on Formal Co-Simulation of Cyber-Physical Systems. Trento, Italy (September 2017)
38. Pedersen, N., Lausdahl, K., Sanchez, E.V., Larsen, P.G., Madsen, J.: Distributed Co-Simulation of Embedded Control Software with Exhaust Gas Recirculation Water Handling System using INTO-CPS. In: Proceedings of the 7th International Conference on Simulation and Modeling Methodologies, Technologies and Applications (SIMULTECH 2017). pp. 73–82. Madrid, Spain (July 2017), iSBN: 978-989-758-265-3
39. Peleska, J.: Industrial-Strength Model-Based Testing - State of the Art and Current Challenges. Electronic Proceedings in Theoretical Computer Science abs/1303.1006, 3–28 (2013)
40. Penzenstadler, B., Eckhardt, J.: A Requirements Engineering content model for Cyber-Physical Systems. In: RESS. pp. 20–29 (2012)
41. Platzer, A.: Differential-algebraic dynamic logic for differential-algebraic programs. Journal of Logic and Computation 20(1), 309–352 (2010)
42. Platzer, A.: Logical Analysis of Hybrid Systems. Springer (2010)
43. Santos, T., Cavalcanti, A., Sampaio, A.: Object-Orientation in the UTP. In: Dunne, S., Stoddart, B. (eds.) UTP 2006: First International Symposium on Unifying Theories of Programming. LNCS, vol. 4010, pp. 20–38. Springer-Verlag (2006)
44. Tang, X., Woodcock, J.: Travelling processes. In: Kozen, D. (ed.) 7th Intl. Conf. on Mathematics of Program Construction (MPC). Springer, vol. 3125, pp. 381–399. Springer (2004)
45. Wei, K.: Reactive designs of interrupts in Circus Time. In: 10th Intl. Colloquium on Theoretical Aspects of Computing (ICTAC). LNCS, vol. 8049, pp. 373–390. Springer (2013)
46. Wenzel, M., Wolff, B.: Building formal method tools in the Isabelle/Isar framework. In: 20th Intl. Conf. on Theorem Proving in Higher Order Logics. LNCS, vol. 4732, pp. 352–367. Springer (2007)
47. Wiesner, S., Gorldt, C., Soeken, M., Thoben, K.D., Drechsler, R.: Requirements engineering for cyber-physical systems - challenges in the context of "industrie 4.0". IFIP Advances in Information and Communication Technology, vol. 438, pp. 281–288. Springer (2014)
48. Woodcock, J., Cavalcanti, A.: A tutorial introduction to designs in Unifying Theories of Programming. In: Boiten, E.A., Derrick, J., Smith, G. (eds.) Integrated Formal Methods, 4th International Conference, IFM 2004, Canterbury, UK, April 4–7, 2004, Proceedings. Lecture Notes in Computer Science, vol. 2999, pp. 40–66. Springer (2004)
49. Zeyda, F., Ouy, J., Foster, S., Cavalcanti, A.: Formalised Cosimulation Models. In: The 1st Workshop on Formal Co-Simulation of Cyber-Physical Systems (CoSim-CPS). Trento, Italy (September 2017)