Fitzgerald J, Gamble C, Payne R, Larsen PG, Basagiannis S, Mady AE-D.
[Collaborative Model-based Systems Engineering for Cyber-Physical Systems,](#)
[with a Building Automation Case Study](#). In: *INCOSE International Symposium*
***(IS2016)*. 2016, Edinburgh, UK: Wiley Online Library.**

Copyright:

This is the peer reviewed version of the following article: Fitzgerald J, Gamble C, Payne R, Larsen PG, Basagiannis S, Mady AE-D. [Collaborative Model-based Systems Engineering for Cyber-Physical Systems, with a Building Automation Case Study](#). In: *INCOSE International Symposium (IS2016)*. 2016, Edinburgh, UK: Wiley Online Library., which has been published in final form at <http://dx.doi.org/10.1002/j.2334-5837.2016.00195.x> This article may be used for non-commercial purposes in accordance with Wiley Terms and Conditions for Self-Archiving.

Date deposited:

27/09/2016

Embargo release date:

18 July 2017

This work is licensed under a [Creative Commons Attribution-NonCommercial 3.0 Unported License](http://creativecommons.org/licenses/by-nc/3.0/)

Collaborative Model-based Systems Engineering for Cyber-Physical Systems, with a Building Automation Case Study

John Fitzgerald
Carl Gamble
Richard Payne
Newcastle University
Newcastle upon Tyne, UK
firstname.lastname@ncl.ac.uk

Peter Gorm Larsen
Aarhus University
Aarhus, Denmark
pgl@eng.au.dk

Stylianos Basagiannis
Alie El-Din Mady
United Technologies
Cork, Ireland
BasagiS@utrc.utrc.com
MadyAA@utrc.utrc.com

Copyright © 2016 by Fitzgerald, Gamble, Payne, Larsen, Basagiannis, Mady. Published and used by INCOSE with permission.

Abstract. We describe an approach to the model-based engineering of cyber-physical systems that permits the coupling of diverse discrete-event and continuous-time models and their simulators. A case study in the building automation domain demonstrates how such co-models and co-simulation can promote early cooperation between disciplines within a systems engineering process before the expensive commitment is made to integration in physical prototypes. We identify areas for future advances in foundations, methods and tools to realise the potential of a co-modelling approach within established systems engineering processes.

Introduction

INCOSE's vision for Systems Engineering in 2025 (INCOSE, 2014) sets out key challenges in transforming systems engineering to meet the 21st Century's demands. It envisages systems of systems enabled by networked, autonomous computing elements (pg. 32), and anticipates collaborative and model-based engineering methods as means of managing risk by supporting integrated cross-disciplinary analyses, design space explorations and optimisations (pgs. 28, 38-39). Our work aims at the realisation of this vision. Specifically, we ask how collaborative model-based methods and tool chains can enable the multidisciplinary engineering of Cyber-Physical Systems (CPSs).

CPSs are smart networked systems that integrate embedded computational elements (sensors, processors, actuators) with physical processes and human users. Examples include agile manufacturing systems, responsive electricity grids, and smart buildings (NIST, 2013). In a CPS, reliance is placed on joint behaviour of the cyber and physical elements, and so delivering confidence in their interaction and emergent properties is a focus for research on systems engineering for CPS. CPSs can interact to deliver new emergent behaviours, and exhibit many of the characteristics of systems of systems (NIST, 2015).

The nature of CPS design places demands that have not yet been met on the methods and tools of systems engineering (Törngren et al., 2014). We focus on three such challenges:

Model-based Tool Chains and Work Flows. The production of CPS prototypes is costly, and there are risks of damage in testing. Model-based systems engineering is intended to improve stakeholder communications. Model-based validation, analysis and simulation from early life-cycle stages have the potential to allow alternative designs to be explored, permitting early identification of bottlenecks and defects, managing prototyping and development risks, and reducing cycle times (Walden et al., 2015). There have been many calls for improved model-

based notations, methods and tools to enable integration with work flows and tool chains, e.g., (Broy, 2013) (Derler et al., 2012) (Horvath and Gerritsen, 2013).

Semantic Gaps. Simultaneous satisfaction of cross-cutting cyber, physical and other design concerns is a challenge. For CPSs this entails integrating models with highly diverse semantics. For example, discrete-event (DE) models of computing elements describe series of discontinuous events, whilst continuous-time (CT) models of mechanics, electrics, etc. describe continuously varying quantities, often as systems of differential equations. Current design flows are clustered in discipline-specific verticals (electrical, mechanical, software, etc.), so there is a need for model and tool integration across traditionally separated disciplines (NIST, 2015). Can we bridge this “semantic gap” to integrate such diverse methods and tools?

Traceability for Assurance. If we are to develop CPSs on which reliance can justifiably be placed, evidence in the form of models, design rationale, analyses and simulation outcomes must support traceability, both through the life-cycle (e.g., from requirements models to functional specifications and on to test cases and results), and between diverse models integrated into co-models, as a CPS evolves.

Our current work aims to address these challenges. In the project INTO-CPS¹ we aim to create an integrated tool chain that supports multidisciplinary, collaborative modelling (co-modelling) and simulation (co-simulation) from requirements, through design, to realisation in hardware, software and physical elements, enabling traceability at all stages. Rather than produce a single multi-purpose tool, we aim for a pragmatic integration of baseline tools that have relatively high Technology Readiness Levels (TRL 6-9) in their domains. The tool chain will be underpinned by semantic foundations that ensure the results of analysis can be trusted.

In this paper we review the current lively research scene (Section 0) before introducing basic co-modelling (Section 0). We present a case study in building automation design which highlights strengths and current limitations of this basic approach (Section 0). Our new work in INTO-CPS to address these limitations is described in Section 0. We review the extent to which this helps to address the INCOSE vision and conclude in Section 0.

Related Work

There is a rapidly growing body of research on methods and tools for CPS engineering, supported by significant investments by the National Science Foundation and other agencies in the US², and by the European Commission (EC)³. Among efforts to provide a conceptual basis for this emerging field, the most comprehensive to date is the NIST draft framework for CPS (NIST, 2015). It identifies three *facets* of CPSs: conceptualisation (relating to the production of models), realisation (relating to the design and implementation of specific CPSs), and assurance (relating to the claims, argumentation and evidence required to allow reliance justifiably to be placed on CPSs). The challenges identified in Section 0 map to these facets: integration of models into workflows addresses conceptualisation; bridging of semantic gaps is needed to analyse models in realisation; traceability is needed to provide evidence for assurance. We briefly review the state of the art in the challenge areas identified in Section 0.

¹ A European Commission Horizon 2020 Research and Innovation Action. See into-cps.au.dk.

² For example, www.cps-vo.org.

³ CyPhERS (www.cyphers.eu) has produced an EU roadmap and strategy (Schätz et al., 2015). The CPSoS (www.cpsos.eu) brings CPS and SoS communities together. Road2CPS (www.road2cps.eu) advances CPS roadmapping and community building in the EU, while TAMS4CPS (www.tams4cps.eu) and CPSSummit (<http://cps-vo.org/group/cps-summit>) aim to develop a cooperative transatlantic research agenda.

Model-based Tool Chains and Work Flows

There are many overlapping initiatives in the development of tool chains for CPS engineering. The Vanderbilt tool chain provides an integrated framework for embedded system development, providing multiple views including software architecture, hardware modelling, and deployment (Mosterman et al., 2004, Zhang et al., 2014). HybridSim (Wang and Baras, 2013) supports importing existing components from multiple domains into SysML blocks (OMG, 2012), from where Functional Mock-up Units (FMUs) can be generated. These can be co-simulated using the Functional Mockup Interface (FMI) standard⁴ to synchronise their corresponding simulators. (Canedo et al., 2014) have developed a multi-disciplinary design automation tool for automotive CPSs that evaluates the system-level impact of domain-specific design decisions using simulation, reducing design cycle time. There is progress on platforms to support key links in tool chains, e.g. Cosimate⁵ is a backplane co-simulation tool offering interfaces to tools like Simulink, Modelsim, and Modelica, and system description in SysML, as used in (Bouffaron et al). Contractual approaches comparable to those proposed for SoSs (e.g., (Bryans et al., 2014)) have also been proposed for CPSs (Sangiovanni-Vincentelli et al., 2012). In spite of these many activities, the state of the art is still some way from providing life-cycle tool chains, with formal foundations integrating the semantics required by different forms of analysis (e.g., operational semantics for simulation, axiomatic semantics for proof) to yield coherent results.

There appears to be less work on the integration of diverse models into work flows. (Petnga and Austin, 2014) propose a systematic iterative work flow for CPS development that utilises semantic technologies, including the definition of ontologies for concepts in the application and infrastructure levels. This is complementary to the work we describe here, which focuses on the formal semantics of diverse modelling notations.

Bridging Semantic Gaps

There have been calls for a foundation for CPS design that is precise and predictable (Sztipanovits et al., 2012) while supporting integration of semantic bases. Several model-based approaches support heterogeneous modelling and simulation. Much of the research utilises hybrid statecharts and automata as a common modelling basis (Maler et al, 1992), (Alur et al., 1995). Around this, design languages have been proposed supporting simulation and verification (Carloni et al., 2006). (Bauer, 2012) argues that such languages should have precise operational semantics for simulation, and denotational or algebraic semantics for automated analysis. However, it has also been suggested that many languages do not meet these requirements (Guturu and Bhargava, 2011). Another approach is to integrate heterogeneous models. (Karsai and Sztipanovits, 2008) introduce a model-integrated approach to CPS design that covers all aspects of hardware and software components and their interactions. One of the leading heterogeneous frameworks is Ptolemy-II, where computation modes are specified for model elements (Eker et al., 2003). Our own work links DE models of controllers with CT models of controlled plant based on a reconciled operational semantics (Fitzgerald et al., 2015).

Traceability for Assurance

Traceable documentation is crucial to model-based systems engineering to record design rationale, and to help estimate the impact of change and evolution (Fisher et al., 2014). In a traditional document-centric approach, traceability matrices may be used to map external

⁴ See www.fmi-standard.org.

⁵ See www.chiastek.com/products/cosimate.html

sources to system requirements, system requirements to software/physical requirements, and so on to detailed design, implementations, and test documentation. These links allow traceability forwards (from external sources through requirements to code) and back. However, the maintenance of these matrices can be labour-intensive, and is often dropped under pressure (Juristo et al., 2002). While many tools support the basic functionality of creating traceability links, none yet do this automatically (Mäder, 2010). Further, there is a lack of support for the retrieval and re-simulation of the multiple models in diverse formalisms that arise in model-based systems engineering for CPSs, as part of regression testing.

Co-modelling and Co-simulation

Using a single modelling technique to design a whole system can limit the system representation because the lack of native abstractions needed to describe the full range multi-domain requirements, effectively limiting the precision and accuracy of the resultant models. We argue that cyber-physical models require the integration of approaches that can capture physical phenomena as well as abstractions describing the computation, data and communication capabilities of modern systems.

The state of the art in model-based CPS engineering suggests that the design chain spanning multiple model types is not well supported. A first approach to this problem might be to deploy unified notations and tools. However, this may mean abandoning notations and legacy models that are very effective in their application domains. For example, much CPS software is concerned with managing mode changes, communications, concurrency and supervisory control functions including safety logic. The description of such software requires abstractions of complex data, software structure and concurrency that are not readily available in continuous-time (CT) notations that have evolved principally to describe physical systems, e.g. as systems of differential equations. Likewise, discrete-event (DE) formalisms are not always competent to model physical phenomena to sufficient fidelity. We would argue for the integration of established but heterogeneous modelling tools and notations, linking them as *co-models* that may be analysed and run in *co-simulations*.

A binary co-model contains two *constituent models*: a DE model that typically describes computational processes, and a CT model of physical elements such as controlled plant. A *co-simulation* of a co-model entails the coordinated running of simulations in the two constituent models and requires a unified/tool-independent platform, where analysis of the whole system can examine the models interaction in a collaborative manner. Co-simulation based on heterogeneous models can be an ideal solution to inspect and validate CPS requirements. However, the complexity of the models raises the challenge of efficiently executing system-wide co-simulations. Analysis of the models is required to sweep over a set of simulations and explore the design-space. This functionality is currently not present in similar model-based co-simulation design tools, such as SimCoupler (PSIM-Simulink co-simulation)⁶.

We have developed and demonstrated methods for binary co-modelling and co-simulation, using VDM-RT as the DE formalism, and 20-sim⁷ as the CT framework (Fitzgerald et al., 2014). VDM-RT extends the ISO Standard formal specification language VDM (ISO, 1996). It supports the modelling of structured data constrained by logical invariants. Functionality may be described explicitly, or contractually via preconditions and postconditions (Fitzgerald and Larsen, 2009). It has features to describe sophisticated control software, including object-orientation, concurrency, and bounded computation times. Model construction and simulation

⁶ See www.powersimtech.com/products/psim-modules/simcoupler/

⁷ See www.20-sim.com.

is supported by the open-source Overture tool⁸. On the CT side, 20-sim permits the specification of mechatronic systems by means of bond graphs or as blocks of ordinary differential equations. It supports simulation by means of a powerful solver.

The co-modelling approach for VDM-RT and 20-sim is implemented in the open source Crescendo platform⁹. Crescendo implements a reconciled operational semantics that yokes the Overture simulation tool with the 20-sim solver so that shared data (such as sensor values and actuator settings) and simulation time are passed between the two simulation engines, keeping them sufficiently in step to allow the coupled simulations to perform as a single simulation.

We have developed guidelines and patterns for co-model construction, particularly focusing on the description of potential faults, error detection, isolation and recovery as this is a source of complexity in the design of embedded systems. These methods, and the Crescendo tools, have been validated in a series of industry case studies in embedded systems design for paper processing, heavy machinery operation, vehicle control and other sectors (Fitzgerald et al., 2014). The results demonstrated a reduction in iterations over physical prototypes, the ability to develop of logically complex subsystems such as safety control, and the ability to address a range of failures, such as noisy communications.

Figure 1: Co-simulation in Crescendo: overview

A Case Study in Co-modelling for Building Automation

This section provides a first illustration of the technical capability of co-modelling and co-simulation. We motivate a case study in building automation (Section 0), describe the co-model and constituent models (Section 0), discuss co-simulation and design space exploration (Section 0), and assess the strengths and weaknesses of the co-model against required capabilities (Section 0).

Motivation and Outline

Buildings are responsible for 40% of energy consumption and 36% of EU CO₂ emissions¹⁰, so high energy performance of building infrastructures is key to achieving climate and energy objectives. Significant savings can be achieved using strategies such as user profiling based on data gathered using Internet of Things technologies to control equipment distributed in a building. For example, peak shaving strategies can yield savings in infrastructure required to provide peak power, and improved reliability of energy supply (Mady, 2011). In order to gain

⁸ See www.overturetool.org.

⁹ See www.crescendotool.org.

¹⁰ See http://ec.europa.eu/research/industrial_technologies/eeb-challenges-ahead_en.html

confidence in overall operation, it is necessary to integrate discipline-specific models (e.g., software, thermal, electrical).

Our case study is inspired by heating, ventilation and air conditioning (HVAC) systems. We consider a single room control unit in order to assess the potential for co-modelling in this setting. In Section 11 we consider how this would be integrated into a larger building model. We model a Fan Coil Unit (FCU) which controls air temperature in a room through the use of physical components and software controllers (Figure 2). Water is heated or cooled in a heat pump and flows into a *Coil*. A *Fan* blows air through the coil. The air is heated or cooled by the coil, and flows into the room. A *Controller* can alter the fan speed and the rate of the water flow from the heat pump to the coil. In addition, the room temperature is affected by the walls. The aim of the system is to maintain a set temperature in the room in which the FCU is located.

Figure 2: Fan Coil Unit Example

A co-model of the FCU's control functionality has been developed and co-simulated using the Crescendo technology described in Section 0. Potential users (in UTRC) identified the following capabilities as a basis for assessing the study as an illustration of the potential of co-modelling in the building automation CPS domain:

1. **Co-modelling.** Create co-models of the FCU, room and external environment. Run co-simulations over a range of values of design parameters with user-set control inputs.
2. **Co-simulation Performance.** Demonstrate co-simulations of FCU operation in excess of one week's operation in an acceptable time frame.
3. **Co-simulation Accuracy and Precision.** Demonstrate user selection of simulation precision.

We discuss the traceability aspect of the study in Section 0.

A Co-model

The co-model's architecture is shown in SysML in Figure 3. A *HeatingSystem* co-model has two constituent models: *RoomHeating* representing the room and heating plant, and *Controller* representing the digital control system. The *Environment* block represents the external environment which in this case will consist of the ambient temperature outside the room, and a user who may choose and change the desired temperature (set point). The *RoomHeating* subsystem is best modelled as a continuous subsystem, and has components representing the *Room* and *Wall*, both of which are contained in a CT model. The *Controller* subsystem is a cyber element and is thus in the DE model. The *Environment* is modelled as tables of time-indexed values for the external temperature and the set point.

Figure 3: Architectural Structure Diagram for the Fan Coil Unit in SysML

Figure 4 shows the connections between the components, reflecting the hierarchy shown in Figure 3 with the CT (*RoomHeating*) and the DE (*Controller*) models. The connections between CT and DE models show the interface that is managed during the co-simulation. The room air temperature (*RAT*) from the CT system is communicated to the controller, which sets the fan speed *fanSpeed* and the valve state *valveOpen* used by the *Room* component *r*, with the aim of achieving the set point *RATSP* provided by the user in the *Environment*.

Figure 4: Connection Diagram for the Fan Coil Unit co-model in SysML

Within the CT side, the *Room* model determines the room air temperature (*RAT*) on the basis of *fanSpeed*, *valveOpen*, and the temperature of the internal surface of the wall (delivered by the *Wall* model). The thermal model of the wall determines the internal wall surface temperature on the basis of the *RAT* and the outside air temperature (*OAT*) supplied by the *Environment*. The *Controller* block is to be described as a DE model that in this case performs

a basic PID control algorithm (but which, because it is modelled using a rich DE notation) can readily be extended to model concurrent, distributed and moded computation.

We now consider the realisations of the functionality in the CT and DE models. The CT model of the *RoomHeating* system was developed in 20-sim. Figure 5 gives an overview of the model. The *Wall* and *Room* elements are defined as systems of ordinary differential equations. Delving one level deeper into the CT model, Figure 6 shows the equations describing the thermal performance of the *Wall*. The design parameters include characteristic properties of the wall material such as *lambdaWall*. The equations are solved for given times by the 20-sim simulation engine. Recall that the value *Tisurf* is shared with the *Room* block.

Figure 5: Overview of the Continuous Time model of the Fan Coil Unit (in 20-Sim)

```

parameters
    real rhoWall = 1312.0;    -- wall density
    real cWall = 1360.71;    -- specific heat capacity
    real lambdaWall = 0.1192; -- wall thermal conductivity
    real lWall = 0.001;      -- wall thickness
    real aWall = 60.0;       -- wall area
    real hi = 8.33;          -- indoor heat transfer coefficient
    real ho = 33.33;         -- outdoor heat transfer coefficient
    real TisurfInit = 16.0;   -- initial internal surface temperature
    real TosurfInit = 10.0;   -- initial external surface temperature

variables
    real Tosurf; -- external surface temperature
    real R;      -- wall resistance
    real C;      -- wall thermal capacity

equations
    R = lWall / (lambdaWall * aWall);
    C = 0.5 * rhoWall * cWall * lWall * aWall;
    Tisurf = int((hi * aWall * (RAT - Tisurf) + (Tosurf - Tisurf) / R) /
    C, TisurfInit);
    Tosurf = int((ho * aWall * (OAT - Tosurf) + (Tisurf - Tosurf) / R) /
    C, TosurfInit);

```

Figure 6: CT model – thermal equations for *Wall*

The DE model of the controller is constructed and executed in the Overture tool. Figure 7 shows the structure of the DE model as a class diagram. Within each class, interface specifications

indicate the data and functionality available from the control, sensor and two actuators. Note the use of a common actuator class that can be particularised to the needs of the specific model: The *ActuatorLimited* class overrides the *setState* functionality to limit outputs from the PID control algorithm to suit the physical valve and fan in the FCU. Figure 8 shows a small extract of the functional description from the *Controller* class. This example shows a very simple PID controller, but – crucially – could contain significantly more complex computational elements. The model extracts shows the body of the control loop, and the “thread periodic” declaration, which models the deployment of the thread to a CPU with specified performance characteristics (in this case the period of the control loop: other parameters allow the modeller to experiment with temporal jitter, delay and offset).

Figure 7: DE model of *Controller*: class structure

```

private PIDcalculate: () ==> ()
PIDcalculate() ==
(
    syncSensorsAndActuators();

    MV:=RAT.getLevel();
    err:=RATSP-MV;
    factor:=Td/(sampletime+(Td/N));
    uP:=K*(b*RATSP-RAT.getLevel());
    uI:=previousuI+sampletime*(K*err/Ti);
    previousuI:=uI;

    uDin:=c*RATSP-MV;
    previousuDin:=uDin;
    uD:=factor*(uD/N+K*(uDin-previousuDin));
    control:=uP+uI+uD;

    valveOpen.setState(control);
    fanSpeed.setState(control);
);

thread
-- execute the control loop every 80 ms
periodic(80E6/*ms*/,0,0,0)(PIDcalculate);

```

Figure 8: Extract from DE model of *Controller*: VDM-RT description of control thread

Co-simulation and Design Space Exploration

Figure 9 shows the outputs of the co-simulation. Traces show the variation in RAT (top left) as the OAT (bottom left) changes. The upper right shows (rather extreme) actions of the control algorithm. The bottom right shows FCU's cumulative energy output through the co-simulation.

Figure 9: Co-simulation output from FCU co-model

Figure 10: DSE Result: cost against thermal conductivity for a range of operating periods.

Design Space Exploration (DSE) can be performed by running sets of co-simulations that sweep over design parameters. For example, a sweep over the wall's thermal conductivity calculating energy consumption, combined with energy and construction costs, permits a review of overall cost of ownership with the controller for a given period. Figure 10 shows one such analysis, illustrating how the cost of low conductivity walling comes to be outweighed by energy savings over a longer period. (Note: this is merely an illustrative example; the conductivity figures would normally be much lower.) We can equally sweep over DE parameters such as control loop frequencies.

Limitations of the Co-model: the need for Multi-modelling

The case study is deliberately small, but does suggest that it is feasible to support multidisciplinary co-simulation and DSE for a system with a replaceable components such as the FCU, the wall materials, etc. However, there are some limitations. We review the case study against the capabilities identified in Section 0.

Co-modelling. The study has demonstrated the ability to create co-models and run co-simulations of the FCU, room and external environment with alternative values of design parameters and control inputs. Our co-simulation is restricted, however, to two constituent models of fixed types, and this dictates the model architecture. For example, we are obliged to package the *Wall*, *Room* and *FCU* into a single CT model in 20-sim. In practice, for a large CPS composed of elements from multiple suppliers with diverse models, we might for example have to integrate an OpenModelica model of the FCU with a 20-sim model of the thermal properties of the room, with a supplier's different model of the *Wall*. We would need to step from binary co-models to *multi-models* composed of many multiple diverse constituent models (enabling the flatter structure of Figure 11). This would better equip the systems engineer to substitute a wider range of supplier-provided constituent models.

Figure 11: Architecture Structure Diagram for a Fan Coil Unit multi-model in SysML

Co-simulation Performance. A simulation of one week's operation can be run in approximately 7.5 minutes on a quad core i5 processor at 3.3GHz with 8Gb RAM running Windows 10. This is considered acceptable performance.

Co-simulation accuracy and precision. The DE modelling notation for the controller enables the engineer to set features including control loop frequencies, resource consumption for specific instruction sets, and other factors. There is a good basis for adjusting precision in the DE side, though this is not currently explicitly supported.

Towards Integrated Tool Chains

Although we have demonstrated the potential of co-modelling, it still has limitations. We need to move from binary co-models to multi-models, integrate a wider range of model types, and develop support for traceability. These needs have inspired the INTO-CPS project, which began in 2015, and aims to deliver these features in a SE toolchain for CPSs to a greater level of maturity than hitherto. To appreciate the need for such a project, consider a sophisticated HVAC system enabled by Internet of Things technologies. Consider a controller that uses data from connected devices such as mobile phones, and building security data, to determine who is on the premises. The system may use this data to control FCUs, bringing rooms to a set point only "as needed". Model-based design of such a CPS would benefit from mixed formalisms

describing the cyber and physical sides. For example, in order to analyse communications faults, a DE model able to describe message loss, reordering or corruption would be required. We may also wish to integrate diverse CT thermal models from existing libraries.

INTO-CPS extends the approach from co-models to multi-models (Fitzgerald et al., 2015) by integrating existing tools into a well-founded chain supporting the life-cycle from requirements over different types of models of constituent elements down to their realisation (Figure 12 provides an overview). This will be prototyped using the baseline tools listed in Figure 13.

Figure 12: Conceptual overview of the INTO-CPS tool chain

In order to bridge the semantic gap between constituent models the tool-independent FMI standard is used. Each of the baseline tools with simulation capabilities is extended to produce stand-alone FMUs which enable simulation of a constituent model without any external tool. To ensure coordination between FMUs a Co-simulation Orchestration Engine (COE) keeps the simulations of the constituent FMUs in harness, managing data transfer and the progression of time, effectively delivering a single simulation. As with Crescendo, the COE executes a reconciled operational semantics of the individual simulators realized by their FMUs.

The approach also extends the capability up and down the development work flow by linking the multi-models to CPS requirements and architectural modelling, and downstream to software and hardware realizations. Developing a CPS will produce a large number of artefacts, including requirements, models, analysis results, and generated code. The tool chain will allow these artefacts to be stored, organised, and easily retrieved at a later date. It will allow the provenance of all artefacts to be recorded and traced back to the requirements. This data can be used at a later stage as evidence in documenting the adequacy of a design to meet the requirements. This results in a complete engineering approach to manage, track and monitor model artefacts used in collaborative heterogeneous modelling.

Figure 13: The baseline tools used as proof of concept in the INTO-CPS project

Building on our experience in SoS (Holt et al., 2015), we are developing guidelines to support the rigorous analysis of requirements using SysML, which includes features for description of both discrete event computing processes and continuous phenomena. Given an architectural model describing computing, physical and networking elements, an FMI interface can be generated, with stub models to reduce the effort in producing initial constituent models. We have implemented such export of model descriptions for each of the constituent models in the baseline Modelio tool. These are imported by the different baseline simulation tools, indicating the interfaces that are needed for the corresponding FMUs. Heterogeneous models can then be built around this FMI interface, using the stub models starting points. Aside from co-simulation, we intend to develop a tool chain that will permit static analysis of FMI interfaces and co-models, including model checking.

The COE will allow real software and physical elements to participate in co-simulation alongside models (Model-in-the-Loop, MiL), enabling both Hardware-in-the-Loop (HiL) and Software-in-the-Loop (SiL) simulation. Code generation from some of the baseline tools will help support automated HiL simulation. The tool chain will allow co-simulations to be defined via DSE or through Test Automation (TA) based on test cases generated from the SysML requirement diagrams. Both DSE and TA will also be used in the FMI co-simulation framework produced by the COE explained above.

Traceability between requirements, model elements and results of diverse analyses is established using the principles from Open Services for Lifecycle Collaboration (OSLC)¹¹. This gives users control over the relationships between artefacts such that assurance of CPS developments can use the models as well as the final realisations. Key to traceability in practice is the ability to assess the provenance of evidence generated in support of claims. The Prov-O ontology and Prov-N notation¹² allow provenance information generated in different systems and under different contexts to be represented and exchanged.

We have identified the need for a formal semantic framework to link diverse modelling notations and tools, so confidence can be placed in analysis. Our approach, which uses Unifying Theories of Programming (Hoare and Jifeng, 1998), is to select language features, giving them a denotational semantics; algebraic, axiomatic and operational semantics can then be proved sound against this. These features can be assembled to form the semantics for modelling languages, facilitating compositional analysis, as has been done in the systems of systems context (Woodcock, 2014).

Conclusions and Future Work

We began from the INCOSE SE vision, asking how collaborative model-based methods and tool chains can enable multidisciplinary engineering for CPSs. We highlighted the challenges of integrating heterogeneous models into tool chains and work flows, providing semantic links between them, and supporting assurance through traceability. Our review of the state of the art suggested that, while some links in tool chains have been established, challenges remain in: the integration of co-modelling into life-cycle SE tool chains and work flows, semantic integration of diverse models, and assisted creation and maintenance of traceability structures.

We demonstrated a co-model approach linking heterogeneous models of controller and plant in a building automation example. It is feasible – at least on the small scale – to perform co-

¹¹ See <http://open-services.net/>.

¹² <http://www.w3.org/TR/prov-o/> and <http://www.w3.org/TR/prov-n/> respectively.

simulation and design space exploration on binary co-models in this way. However, it does suggest targets for our ongoing work to scale up to co-simulations of multiple diverse models. It is clear that much work is required to integrate co-modelling technology into practical SE processes for CPSs. Priorities include increasing the number and semantic diversity of constituent models. For example, there is a need to integrate stochastic and other models of human behaviour, economic models (e.g. of energy pricing in our building automation example), etc. Perhaps most importantly, there is an urgent need for practical patterns and guidelines that will enable practitioners to construct and better integrate models being developed in the rich range of formalisms used today.

Acknowledgements

The work reported here is supported by the EU Horizon 2020 Project “Integrated Tool Chain for Model-based Design of Cyber-Physical Systems” (INTO-CPS, Grant Agreement 644047).

References

- Alur, R., Courcoubetis, C., Halbwachs, N., Henzinger, T. A., Ho P-H., Nicollin, X. Olivero, A., Sifakis, J., Yovine, S. 1995. “The algorithmic analysis of hybrid systems.” *Theor. Comput. Sci.*, 138(1):3–34.
- K. Bauer 2012. “A New Modelling Language for Cyber-Physical Systems”. PhD dissertation, Kaiserslautern University.
- Bouffaron F., Marangé P., Morel G. 2014 “Checking models based on an iterative co-specification process of a critical system”, In *Proc. Intl Conf on Industrial Informatics*.
- Broy, M., 2013. “Engineering Cyber-Physical Systems: Challenges and Foundations,” in *Complex Systems Design & Management*, M. Aiguier, et al., pp. 1-13, Eds. Springer.
- Bryans, J., Fitzgerald, J., Payne, R. and Kristensen, K. 2014. “Maintaining Emergence in Systems of Systems Integration: a Contractual Approach using SysML”, In *Proc. 24th INCOSE International Symposium*. International Council on Systems Engineering.
- Canedo, A., Faruque, M. A. A., Richter, J. 2014. “Multi-disciplinary integrated design automation tool for automotive cyber-physical systems.” In *IEEE/ACM Design Automation and Test in Europe (DATE’14)*, Dresden, Germany, pp. 1–2.
- Carloni, L. P., Passerone, R., Pinto, A., Sangiovanni-Vincentelli, A. L. 2006 “Languages and tools for hybrid systems design”. *Foundations and Trends in Electronic Design Automation*, 1(1/2).
- Derler, P., Lee, E. A., Sangiovanni-Vincentelli, A. 2012. “Modeling Cyber-Physical Systems,” *Proceedings of the IEEE* 100(1):13 – 28
- Eker, J., Janneck, J., Lee, E. A., Liu, J., Liu, X., Ludvig, J. Sachs, S., Xiong, Y. 2003. “Taming heterogeneity - the Ptolemy approach”, *Proceedings of the IEEE*, 91(1):127-144.
- Fisher, A., Friedenthal, S., Sampson M. et al. 2014. “Model Lifecycle management for MBSE”, In *Proc. 25th Intl. Symp. on Systems Engineering, INCOSE*.
- Fitzgerald, J. Gamble, C., Larsen, P. G., Pierce, K., Woodcock, J. 2015 “Cyber-Physical Systems design: Formal Foundations, Methods and Integrated Tool Chains. In *Proc. FormaliSE: Formal Methods in Software Engineering*, Florence, Italy. ICSE 2015.
- Fitzgerald, J., Larsen, P.G. 2009. “Modelling Systems - Practical Tools and Techniques in Software Development”, 2nd edn, Cambridge University Press.

- Fitzgerald, J., Larsen, P.G., Verhoef, M. Eds. 2014. "Collaborative Design for Embedded Systems – Co-modelling and Co-simulation". Springer.
- Guturu P., Bhargava, B. "Cyber-Physical Systems: a confluence of cutting edge technology streams." In Intl. Conf. on Advances in Computing and Communication ICACC-11.
- Hoare, C. A. R., Jifeng, H. 1998, "Unifying Theories of Programming". Prentice Hall.
- Holt, J., Perry, S., Payne, R., et al. 2015. "A model-based approach for requirements engineering for systems of systems". IEEE Systems Journal, 9(1):252–262.
- Horvath, I., Gerritsen, B. H. 2013. "Outlining Nine Major Design Challenges of Open, Decentralized, Adaptive Cyber-Physical Systems," in ASME 2013 Intl. Design Eng. Technical Conf. / Computers & Information in Engineering Conf (IDETC/CIE 2013).
- INCOSE 2014. "A World in Motion: Systems Engineering Vision 2025". International Council on Systems Engineering.
- ISO, 1996. "ISO/IEC 13817-1 Vienna Development Method Specification Language Part 1: Base language", International Organisation for Standardisation.
- Juristo, B., Moreno, A. M., Silva, A. 2002. "Is the European industry moving toward solving requirements engineering problems?" 19(6):70–77.
- Karsai G., Sztipanovits, J. 2008. "Model-integrated development of cyber-physical systems". In U. Brinkschulte, et al., Eds. "Software Technologies for Embedded and Ubiquitous Systems", Lecture Notes in Computer Science 5287:46–54. Springer.
- Mder. P. 2010. "Rule-based Maintenance of Post-requirements Traceability". MV Wissenschaft. MV-Verlag, 2010.
- Mady, A. E-D., Provan G, Ryan, C., Brown, K. N. 2011 "Stochastic Model Predictive Controller for the Integration of Building Use and Temperature Regulation", Association for the Advancement of Artificial Intelligence.
- Maler, O., Manna, Z., Pnueli, A. 1992. "From timed to hybrid systems". In J. W. de Bakker, et al, Eds, "Real-Time: Theory in Practice, REX Workshop", Lecture Notes in Computer Science 600:447–484. Springer.
- Mosterman, P. J., Sztipanovits, J., Engell. S. 2004. "Computer-Automated Multiparadigm Modeling in Control Systems Technology". IEEE Trans. Contr. Sys. Techn. 12(2).
- NIST 2013. "Strategic R&D Opportunities for 21st Century Cyber-Physical Systems", National Institute for Standards and Technology Steering Committee for Foundations in Innovation for CPSs, January 2013.
- NIST 2015. Draft Framework for Cyber-Physical Systems, National Institute for Standards and Technology, Cyber-Physical Systems Working Group, Release 0.8 (September, 2015).
- OMG 2015, "Systems Modeling Language (OMG SysML)", version, 1.4, Object Management Group, September 2015. www.omg.org.
- L. Petnga and M. Austin, 2014, "Semantic Platforms for Cyber-physical Systems", in Proc. 25th International Symposium, INCOSE.
- Sangiovanni-Vincentelli, A., Damm, W., Passerone, R. 2012. "Taming Dr. Frankenstein: Contract-based Design for Cyber-physical Systems". European Journal of Control, 18(3):217 – 238.

- Schätz, B. et al. 2015. “Cyber-Physical European Roadmap and Strategy: Research Agenda and Recommendations for Action”, www.cyphers.eu.
- Sztipanovits, J., Koutsoukos, X., Karsai, G., et al. 2012. “Toward a Science of Cyber-Physical System Integration,” *Proceedings of the IEEE*, 100(1):29–44.
- Törngren, M., Bensalem, S., Cengarle, M. V., et al. 2014 “CPS State of the Art,” <http://www.cyphers.eu/>
- Walden, D. D., Roedler, G. J., Forsberg, K. J. Hamelin, R. D., Shortell, T. M. (Eds.) 2015. “Systems Engineering Handbook”, 4th Edn., INCOSE, Wiley.
- Wang, B., Baras, J. S. “HybridSim: A Modeling and Co-simulation Toolchain for Cyber-physical Systems”. In *Proc. 17th IEEE/ACM Intl. Symp. on Distributed Simulation and Real Time Applications*, pp. 33-40. IEEE Computer Society.
- Woodcock, J., “Engineering UToPiA – Formal Semantics for CML” in Jones, C. B., et al., Eds., *Proc. 19th Intl. Symp. on Formal Methods Springer, Lecture Notes in Computer Science* 8442: 2-41, 2014.
- Zhang, Z., Eyisi, E., Koutsoukos, X. D., Porter, J., Karsai, G., Sztipanovits, J., 2014. “A co-simulation framework for design of time-triggered automotive cyber physical systems”. *Simulation Modelling Practice and Theory*, 43:16-33, 2014.

Biographies

Dr John Fitzgerald, full Professor in Computing at Newcastle University, UK, is a researcher and practitioner in formal model-based methods. He led the EC COMPASS project on SoS engineering and now leads methods work in INTO-CPS. He co-leads Newcastle’s \$90m centre for digitally enabled urban sustainability. He is a member of INCOSE and a Fellow of the BCS.

Dr Carl Gamble is a Research Associate in Computing at Newcastle University, UK. He worked as a manufacturing engineer before completing a PhD in service-oriented architecture. He has worked on metadata-driven reconfiguration and is now working on the INTO-CPS project developing methods for design space exploration and traceability in CPS engineering.

Dr Richard Payne is a Research Associate in Computing at Newcastle University, UK. His interests include architectural modelling, dynamic architectural reconfiguration, formal modelling, and the engineering of systems of systems and Cyber-Physical Systems (CPS). He is working on the INTO-CPS project developing integrated tool chains for CPS engineering.

Dr Peter Gorm Larsen is a full Professor in Engineering at Aarhus University, where he leads the software engineering group. After receiving his M.Sc. in Electronic Engineering at the Technical University of Denmark, he worked in industry before completing an industrial Ph.D. in 1995. He leads the INTO-CPS project. He is a member of the board of INCOSE Denmark.

Dr Stylianos Basagiannis is a Senior Research Scientist at United Technologies Research Centre (UTRC) in Cork, Ireland. His PhD is in formal verification (2009). He is Principal Investigator for UTRC in several H2020/ARTEMIS projects, leading application of model-based design technology to building information infrastructures and aerospace systems.

Dr Alie El-Din Mady is a Senior Research Scientist at United Technologies Research Centre. He holds a Ph.D. in Computer Science from University College Cork, and his M.Sc. in embedded systems from Università della Svizzera Italiana. His research interests include model-based design, embedded systems, system-on-chip and networked control systems.