AARHUS UNIVERSITY

# Coversheet

**This is the accepted manuscript (post-print version) of the article.**
Contentwise, the accepted manuscript version is identical to the final published version, but there may be differences in typography and layout.

**How to cite this publication**
Please cite the final published version:

Couto L.D., Basagiannis S., Ridouane E.H., Mady A.ED., Hasanagic M., Larsen P.G. (2018) Injecting Formal Verification in FMI-Based Co-simulations of Cyber-Physical Systems. In: Cerone A., Roveri M. (eds) Software Engineering and Formal Methods. SEFM 2017. Lecture Notes in Computer Science, vol 10729. Springer, Cham

## Publication metadata

| | |
|---|---|
| **Title:** | Injecting Formal Verification in FMI-Based Co-simulations of Cyber-Physical Systems |
| **Author(s):** | Couto L.D., Basagiannis S., Ridouane E.H., Mady A.ED., Hasanagic M., Larsen P.G. |
| **Journal:** | Software Engineering and Formal Methods |
| **DOI/Link:** | 10.1007/978-3-319-74781-1_20 |
| **Document version:** | *Accepted manuscript (post-print)* |

The final authenticated version is available online at https://doi.org/10.1007/978-3-319-74781-1_20

***General Rights***
*Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognize and abide by the legal requirements associated with these rights.*

- *Users may download and print one copy of any publication from the public portal for the purpose of private study or research.*
- *You may not further distribute the material or use it for any profit-making activity or commercial gain*
- *You may freely distribute the URL identifying the publication in the public portal*

*If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.*

*If the document is published under a Creative Commons license, this applies instead of the general rights.*

**This coversheet template is made available by AU Library**
Version 2.0, December 2017

# Injecting Formal Verification in FMI-based Co-Simulations of Cyber-Physical Systems

Luís Diogo Couto[1], Stylianos Basagianis[1], El Hassan Ridouane[1], Alie El-Din Mady[1], Miran Hasanagic[2], and Peter Gorm Larsen[2]

[1] United Technologies Research Center, Cork, Ireland
{CoutoLD,BasagiS,RidouaE,MadyAA}@utrc.utc.com
[2] Aarhus University, Denmark
{miran.hasanagic,pgl}@eng.au.dk

**Abstract.** Model-based design tools supporting the Functional Mockup Interface (FMI) standard, often employ specification languages ideal for modelling specific domain problems without capturing the overall behavior of a Cyber-Physical System (CPS). These tools tend to handle some important CPS characteristics implicitly, such as network communication handshakes. At the same time, formal verification although a powerful approach, is still decoupled to FMI co-simulation processes, as it can easily lead to infeasible explorations due to state space explosion of continuous or discrete representations. In this paper we exploit co-modelling and co-simulation concepts combined with the injection of formal verification results indirectly in a model-based design workflow that will enable verification engineering benefits in a heterogeneous, multi-disciplinary design process for CPSs. We demonstrate the approach using a Heating, Ventilation and Air Conditioning (HVAC) case study where communication delays may affect the CPS system's analysis. We model discrete events based on the Vienna Development Method Real-Time dialect, Continuous Time phenomena using Modelica, and communications using PROMELA. Results are considered and inspected both at the level of constituent models and the overall co-simulation.

## 1 Introduction

One of the approaches to Cyber-Physical System (CPS) analysis is based nowadays on the availability of interoperable, precise, and efficient co-simulation frameworks. Starting from requirements definition of heterogeneous components of a CPS, designers are being challenged to trace and analyse correctly the requirements validity against intermediate system models. Moreover, the complexity of the cyber and physical aspects of the system makes it important to employ approaches based on model abstraction techniques. Those interactions will include shared parameters exchanged between Discrete Event (DE) and Continuous Time (CT) models in a contract based manner that will realise the CPS evolution. A well known approach that standardises co-simulation is the Functional Mockup Interface (FMI) standard [4].

The INTO-CPS project [16] is developing a tool-chain to support model-based engineering of CPSs [9], based on FMI. In the INTO-CPS approach, the system configuration and CPU instruction execution is typically modelled within the DE models; in this

case using the Overture tool [15] and the VDM-RT notation [17]. CT models describing physical phenomena (e.g. thermal, fluid or air flow dynamics) are modelled using notations and tools such as Modelica [8] or 20-Sim [13] in order to engage differential equation solvers to evaluate dynamic behaviour.

The INTO-CPS approach enables CPS developers to take advantage of the powerful domain-specific features and abstraction capabilities of various specialized modelling and simulation tools. This maps well onto the kinds of heterogeneous and multidisciplinary teams that are necessary to carry out CPS design and development. On the other hand, the kinds of simulation tools employed in INTO-CPS and similar approaches often handle CPS characteristics like network communication protocols implicitly. Formal verification and especially model checking is a powerful tool that can complement these approaches but it is typically decoupled from the co-simulations, as it can easily lead to infeasible explorations due to state space explosion challenges.

In this work in progress paper, we seek to address the aforementioned issue by combining co-modelling and co-simulation with formal verification by means of injecting verification results in the co-simulations. The work does not focus on pruning the resulted state space; instead it uses the expresiveness power of PROMELA to model communications priniples in interacting co-simulated objects using the FMI standard that previously could not be realized. In our case, we manually model the communication medium and protocol in Promela, in order to fully exploit synchronous or asynchronous communication packet exchange between our Heating, Ventilation and Air-Conditioning (HVAC) objects, using un-buffered (rendez-vouz) or buffered channels respectively [12]. This enables both verification engineers to be brought into a closer collaboration loop with the multi-disciplinary CPS team and results to be used to steer subsequent validation efforts including e.g. communication delays . We demonstrate our approach by means of a case study drawn from the HVAC domain.

The remainder of this paper is structured as follows: in section 2, we present information about FMI co-simulation and the HVAC domain, necessary to follow the rest of the paper; we describe our approach to combining verification with co-simulation in section 3 and instantiate it in section 4; in section 5, we present and discuss co-simulation results; finally, we conclude in section 6 with remarks and next steps for the proposed approach.

## 2 Background

A CPS can be defined as an integration of computation with physical processes. According to literature, CPSs are considered to be a composition of networks of embedded devices that control physical processes, based on open or closed feedback loops where computations are being affected by physical phenomena [6]. Nowadays, CPS complexity has considerably increased the cyber-physical points of interaction to a state where system correctness and functional safety are a great challenge for validation. In addition to that, cyber-physical devices tend to be developed by multidisciplinary engineers, thus increasing the need for a common engineering framework where different engineers contribute common CPS models and concepts.

In this paper, we present a CPS example derived from the HVAC domain. HVAC systems are in general cyber-physical systems composed by HVAC equipment controlling the temperature of areas through heating or cooling of air circulated in rooms by mastering a series of actuation devices (e.g. fans, water valves) and external physical phenomena (i.e. external temperature, air flow). Embedded devices on the HVAC equipment are typically executing control loop feedback mechanisms (e.g. PI controllers [22]), prognostics, health management and decision support functions that allow the device to handle mechanical components of the equipment. Currently the equipment considered for the HVAC case study consists of: 4 Fan Coil Units (FCUs) responsible for controlling temperature in a given area; 1 supervisory controller coordinating the FCUs; and 1 air handling unit heating or cooling the circulated air to the FCUs. In

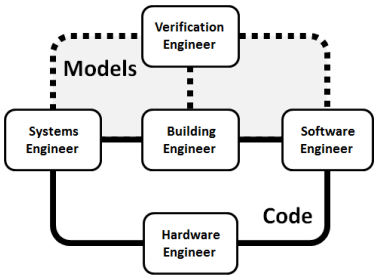


**Fig. 1.** Multidisciplinary Engineer Scheme for Model-based Design of CPS

terms of necessary competencies for CPS development, as seen from Figure 1, there are various engineers involved in the HVAC development cycle. For our example, we have identified the following: a) A *system engineer* will be responsible for CPS system requirements and validation, b) A *building engineer* for modelling the physical effects of the CPS, c) A *software engineer* for modeling and realisation of the CPS software, d) a *verification engineer* for extracting model verification and validation guaranties for high or low level requirements and e) a *hardware engineer* for system realization on a target platform. Towards the CPS analysis, focus is given to:

- Artifacts (models or code) generated from each of the engineer
- The common CPS framework that would allow the multi-model interaction towards validation and verification of the system

An effective way to involve multiple engineers through a common framework is by using co-simulation approaches such as the FMI standard [4]. FMI is a tool independent industry standard that supports the collaborative simulation (co-simulation) of models developed in different tools and notations. This makes it well suited for supporting model-based development of CPSs, whose nature is highly heterogeneous. In the FMI, the models to be co-simulated are exported by their tools as Functional Mockup Units (FMUs) – compiled C code and XML-based model descriptions that specify the

inputs and outputs of the model, thus abstracting away the internal complexity and domain-specificity of the model. A master algorithm is responsible for executing the co-simulation by coordinating the exchange of data between FMUs and progressing overall co-simulation time. In the INTO-CPS project, the implementation of the master algorithm and execution of co-simulations are handled by a tool called Co-simulation Orchestration Engine (COE) from the INTO-CPS project [16].

## 3 Approach

The method proposed in this paper indirectly captures the communication delays due to communication handling mechanisms that are imposed by the used protocol, including them in the FMI-based CPS co-simulation. Since the current engineer development process does not include the means (e.g. modelling tools) to capture these communication mechanism and reason about packet delays within the FMI context, we argue that the proposed approach could be of benefit especially where communication delays can affect the overall CPS correctness. Efforts have been recorded in the current bibliography that try to tackle network verification problems by using either executable formal models in network simulation [2], [3] or extracting formal verification results with respect to communication delays [7], [19]. The main difference highlighted in this work is the incorporation of verification results within the FMI-based co-simulation processes that currently handle communication delays implicitly.

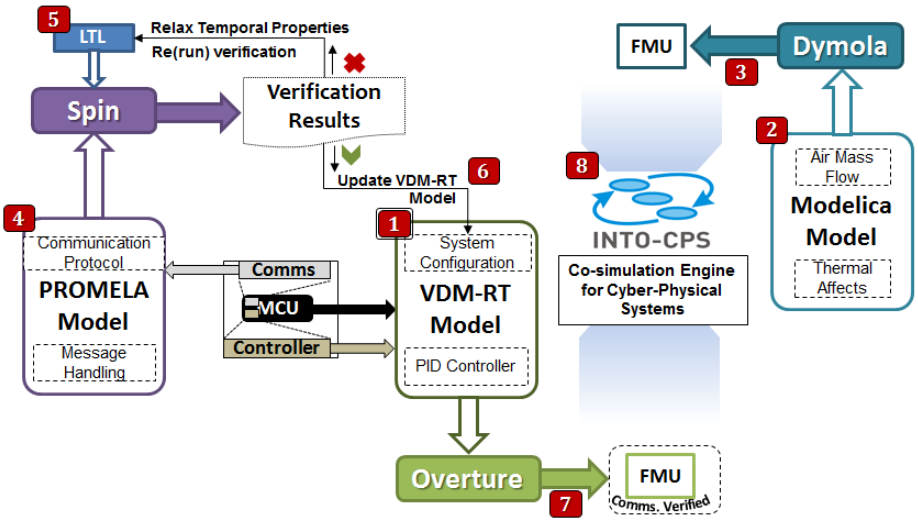


**Fig. 2.** Proposed Experimental Work flow

We propose an experimental work flow that introduce the use of formal verification techniques to evaluate network communication requirements in an FMI-based co-

simulation experiment. Figure 2 illustrates the work flow, instantiated in the HVAC case study with the INTO-CPS tool chain. The work flow consists of the following steps:

1. model the discrete aspects of the systems, including system deployment configuration and behaviour of the cyber components in VDM-RT
2. model physical phenomena of the system in Modelica
3. extract physical plant FMU using Dymola tool
4. model communication aspects of distributed components in Promela
5. verify desired properties using the Spin for VDM-RT time parameters
6. if verification is not successful, update the DE models (e.g. VDM-RT) by constraining the RT timing parameters compared to the initial ones verified in Spin, for communicating objects in VDM; rerun the verification till no errors found with the refined Linear Temporal Logic (LTL) properties
7. generate FMU for DE model with injected verification results using Overture tool. The new FMU will contain updated RT timing parameters that have been previously formally verified by Spin
8. launch a co-simulation experiment with the produced FMUs using the INTO-CPS COE and generate co-simulation results

Spin model checking verification results are glued to the co-simulation by confirming that our VDM-RT timing parameters do take into account certain communication delays contributed to the communication overhead of the modelled protocol. For example if communication specifications for a UART protocol are being defined in Promela for the communicating objects i.e. number of data packets expected to be sent to a common bus in $t$ msec reaching an object, we verify total time delays and packet loses for a network of objects mapped in Promela, equally as in our co-simulation. Total time delays and packet conflicts are defined in our customer requirements, depending on the use-case. Parameter $t$ is being revised (e.g. decreased) in our VDM specification when model checking results can be produced. If verification is successful for the modelled communication protocol, timing parameters are acting as a guard to the VDM model enforcing the delays of the protocol in its generated FMU. Otherwise, the engineer will have to check its hardware characteristics to validate the constrained $t$ value -whether it is supported- and re-run the verification, or to consider changing its hardware to support higher communication bandwidth.

In VDM, models consist of representations of the data on which a system operates and the functionality that is to be performed. In our case this will be the discrete aspects of the HVAC system which will include behavior of the control functionality residing in the FCU devices. Data in the VDM model includes the externally visible input or output and internal state data, which in our case will be the states of the FCUs. PROMELA and its expressiveness for modelling synchronous and asynchronous (buffered) communication channels, will help us 'quantify' communication conflicts and delays in the shared bus of the VDM-RT objects, represented on the discrete model. Since, those objects are co-simulated with the physical part (e.g. sensing the continuous-time environment temperature), each FCU controller expected behavior has to take into account communication conflicts such as lost messages, sychronization issues between master and slave as well as, jamming cases due to message overflow in the common bus due to malfunctions.

## 4 CPS Modelling

In this section, we describe the CPS modelling tasks necessary to combine co-simulation and verification. Our approach is instantantiated in a case study from the HVAC domain: temperature control in multiple rooms with supervision [20, 10]. Based on the INTO-CPS project [11], we carry out *Cyber* modelling using DE formalisms and *Physical* modelling using CT formalisms, leading to two kinds of constituent models. The verification model is created separately, and its results are subsequently injected in the co-simulation. For our case study, we have one DE model, and one CT model, mapped at an architecture level using SysML, and handling communication handshakes using a PROMELA model.

SysML is used by the *system engineer* to describe the components of the CPS (which map onto the constituent models) and the connections between them, which are necessary to carry out co-simulations. The combination of the multiple constituent models and their connections is called *multi-model*. The components of the case study are shown in the INTO-CPS Architecture Diagram [5, 1] of 3. Broadly speaking, the

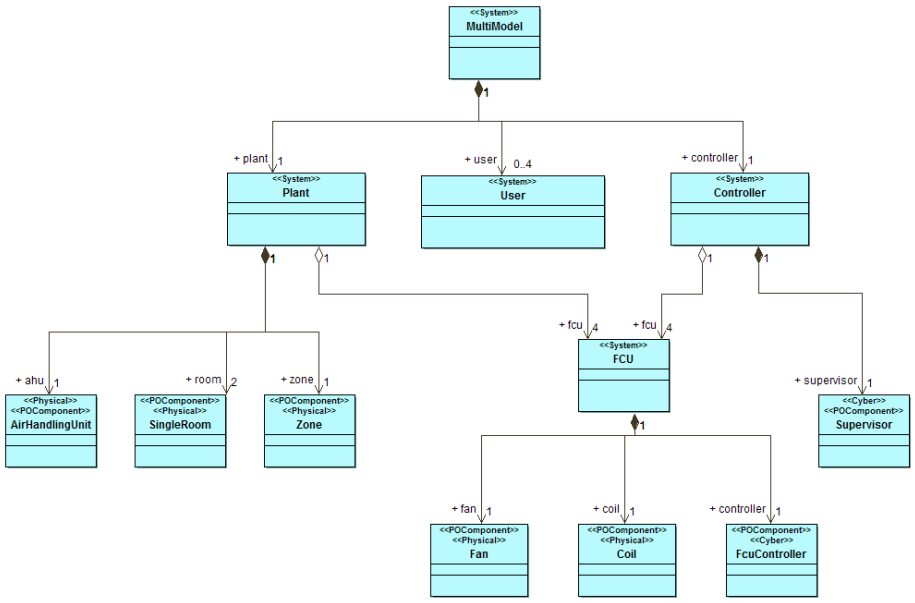


**Fig. 3.** Multi-model components for the HVAC case study.

most relevant components are the controllers (which are parts of the DE model) and the various physical components such as rooms, encapsulated in a single CT plant model. The FCU component itself is abstract and represents the boundary between the CT and DE worlds, thus our system is inherently a hybrid system. The controllers and hardware

of the FCU communicate via exchange of FMI signals across various ports, as shown in the INTO-CPS Connections Diagram of fig. 4.[3]

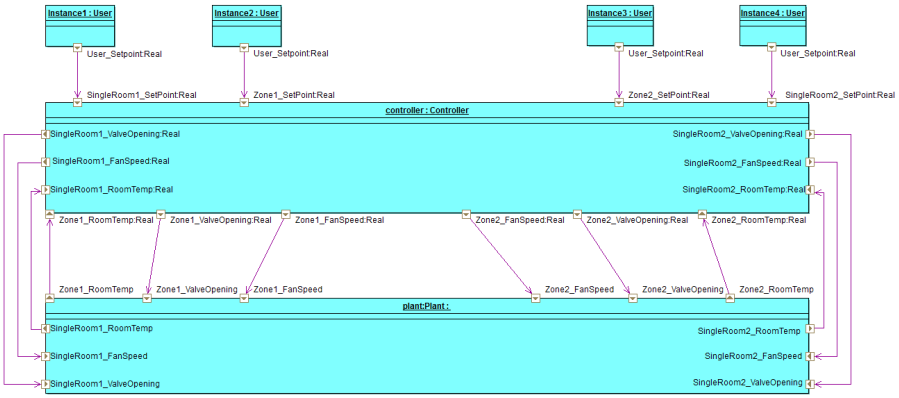


**Fig. 4.** Connections between DE and CT components.

### 4.1 Continuous Time Modelling in Modelica

The continuous behaviour is modelled by the *building engineer* using a CT formalism – in this case, the Modelica language [8] using the Dymola tool. We develop a single plant model that contains all relevant CT components. We use a single CT representation to model all the components we are interested in the Modelica language. Thus, having a single CT model is convenient for model development and debugging since the CT engineer can run stand-alone simulations of his model.

Our physical model consists of rooms and a zone, as well other HVAC components such as the air handling unit. The temperature in the rooms is controlled by the FCUs through the water coils and fans. Air flows from the air handling unit to the FCUs, whereas the heat pump supplies the FCUs with hot or cold water. A PI controller regulates the fan speed and the rate of the water flow from the heat pump to the coil to maintain a constant temperature in the area in which the FCU is located. For space reasons, the Air Handling Unit descriptions modelled in DYMOLA is omitted from this paper.

The PI controller controls the actuation signals based on the sensor signals. This controller is present in both the CT and DE models. At the co-simulation level, the DE controller is used. The CT controller is primarily used to enable standalone simulation of the CT model.

[3] The User component is a basic abstraction used only to enable requests to the system during co-simulation.

The model is based on mass and energy balances in a given room/zone. Two major assumptions were used to simplify the model for a given zone: a) zone air is uniformly distributed, and b) long wave radiation exchange between surfaces is ignored. The energy balance equation of a zone can be described by:

$$m_{air\_zone}(C_{pa} + \omega_{zone}C_{pv})\frac{dT_{zone}}{dt} = \dot{m}_{air\_sa}C_{pa}(T_{sa} - T_{zone}) + \dot{m}_{air\_sa}C_{pv}(\omega_{sa}T_{sa} - \omega_{zone}T_{zone}) + Q_{int} + \sum_i^{N_{surface}} Q_{structure\_i} + \dot{m}_{inf}C_{pa}(T_{oa} - T_{zone}) + \sum_i^{N_{zone}} \dot{m}_i C_{pa}(T_{zone_i} - T_{zone}) + \dot{m}_v C_{pv} T_{zone}$$

The internal water vapour generation rate $m_v^{\cdot}$ is neglected since it is typically small in office buildings. The small heat transfer due to water vapor temperature difference between supply air flow and room air $m_a^{\cdot} ir_{sa} C_{pv}(\omega_{sa}T_{sa} - \omega_{zone}T_{zone})$ is also neglected. The time rate of change of the zone air temperature is given by:

$$m_{air\_zone}C_{pa}\frac{dT_{zone}}{dt} = \dot{m}_{air\_zone}C_{pa}(T_{sa} - T_{zone}) + Q_{int} + \sum_i^{N_{surface}} Q_{structure\_i} + \dot{m}_{inf}C_{pa}(T_{oa} - T_{zone}) + \sum_i^{N_{zone}} \dot{m}_i C_{pa}(T_{zone_i} - T_{zone}) + \dot{m}_v C_{pv} T_{zone}$$

where: $m_{airZone}$ is air mass of room air [kg], $C_{pa}$ is specific heat capacity of air [J/kg.$^o$C], $C_{pv}$ is specific heat capacity of water [J/kg.$^o$C], $T_{zone}$ is room air temperature (RAT) [$^o$C], $\dot{m}_{air_{sa}}$ is the supply air mass flow rate [kg/s], $\dot{m}_{inf}$ is the infiltration mass flow rate [kg/s], $\dot{m}_v$ is the internal water vapor generation rate [kg/s], $Q_{int}$ is the sum of the convective internal loads [W] (assumed to be constant), $T_{isurf}$ is inside surface temperature [$^o$C], $T_{osurf}$ is outside surface temperature [$^o$C], $A$ is wall surface area [$m^2$], $R_{wall}$ is thermal resistance of the wall [$^o$C/W], $C$ is capacity of the wall [J/$^o$C] and $T_{amb}$ is outside air temperature (OAT) [$^o$C].

Profiling information on the complexity of the Modelica model, will report: 2087 components, 21278 variables, 979 constants, 11083 parameters, 9216 unknowns, 412 differentiated variables, 7028 equations, 5775 nontrivial. Modelica was also used to develop a simple model of the user behaviour. This model merely outputs different set point requests at predefined instances in time. Its primary purpose was to enable us to assess the performance of the models when users request changes in temperature.

### 4.2 Discrete Event Modelling Using VDM-RT

The discrete behaviour is modelled using a DE formalism by the *software engineer*. In the case of INTO-CPS, the real-time dialect of Vienna Development Method (VDM) [23] is used. In our case study, we employ a single VDM-RT model that includes behavior of the PI controllers of the FCUs and the supervisor. This modelling choice provide us access to the rich set of VDM-RT features to specify distribution and to minimise the amount of DE-to-DE communication done through the FMI signals.

The model consists of four instances of PI controllers and a single supervisory controller. Each controller is allocated to an individual VDM-RT CPU and connected via a single VDM-RT bus. The role of the PI controllers is to execute a PI loop that regulates the temperature in the FCU's area. The role of the supervisor is to monitor the behavior of the PI controllers to ensure that desired higher level properties are exhibited. As an example, we show the supervisor operation that enforces that FCU set point stays within a given range by overriding the controller set points if they fall outside the range.

A VDM post-condition ensures that the property must hold after the operation executes.

```
private setPointAdjust: Controller ==> ()
setPointAdjust (fcu) == (
  let sp = fcu.getSPValue(),
      target = if minTemp >= 0 and sp < minTemp
               then minTemp
               elseif maxTemp >= 0 and sp > maxTemp
               then maxTemp
               else sp
  in
    fcu.setSuperSetPoint(target)
)
post fcu.acquireSetPoint() <= maxTemp and
  fcu.acquireSetPoint() >= minTemp;
```

**Listing 1.1.** Supervisor property enforcing through set point adjustment.

Using the interpreter of the Overture tool, the model can be executed, enabling independent analysis of the behaviour of the discrete parts of the CPS. In this analysis, communication concerns are implicitly handled by VDM-RT by means of remote operation calls behind the scenes [23], as illustrated in fig. 5. This figure shows the `setPointAdjust` operation where the `getSPValue` invocation is being sent across the bus that connects the CPUs of the supervisor and PI controller. In VDM-RT,

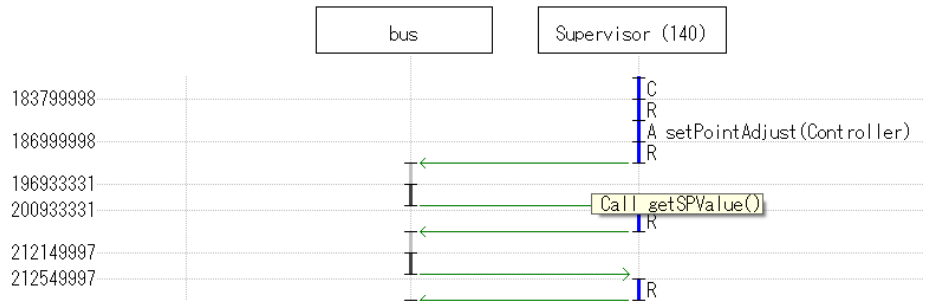


**Fig. 5.** Excerpt of Overture VDM-RT log viewer showing remote operation call.

communication times across the bus are implicitly calculated from the speed of the bus and size of arguments of the remote operation calls. We extend the model to allow for explicit control of communication duration, by making the bus connections instantaneous and prefixing them with parametrized delays. This is only an approximation – the delay occurs before executing the remote operation where the actual message passing takes place – but it allows us to inject timing result estimates extracted from the

verification analysis. These results can them be further experimented with and analysed further in the DE model itself and, eventually, in a co-simulation.

It is worth noting that an important aspect of communication time to consider is the availability of the communication channel since this could affect the communication time by introducing delays if the channel is not immediately available. Since we use an instant bus as channel, there are no communication delays in the DE model. Therefore, we have to account for them in the Promela model and the estimates extracted from it.

### 4.3 Communications Modelling in PROMELA

PROMELA (the input language of the Spin model checker [12]) is used by the *verification engineer* to model communication handshakes between the HVAC CPS entities in our use case. The PROMELA channel definition is ideal for describing distributed communication message exchange between multiple entities accessing a common bus. We model an arbitrary serial bus (UART), conflict resolution mechanism and 4 FCU entities as agents that will exchange messages with the supervisor, which controls the overall HVAC functionality.

Our objective is to identify the maximum time delays caused by UART conflicts by repetitive verification experiments for LTL formulae that will define serial channel properties. Particularly for our case, we are interested in verifying a certain degree of channel reliability with respect to errors found in the technical specifications of message transmission or receipt (i.e. Underrun or Framing errors [21]). As we would like to reason about all paths in the generated state space, we define LTL-P1 and LTL-P2 properties as follows:

**LTL-P1:** *Always if there is traffic in the bus, communication conflicts for all exchanged messages will be less than $\gamma = 20\%$ of total exchanges for total time $t$.*

$$[\ ]((Exc_{msg} > 0 \;\&\&\; Exch_{msg} \leq Total_{msg}) \;\&\&\; Total_{time} < Time_{bound} \rightarrow (Total_{conflicts}) < Total_{msg} * (\gamma))$$

**LTL-P2:** *An ACK message will be sent to the FCU before a PAYLOAD message has been received from the Supervisor*

$$FCU_{(i)}[ACK_{rec} = TRUE] \rightarrow <> (FCU_{(i)}[PAYLOAD_{Sd} = TRUE] \;\&\&\; Time < 100)$$

We developed 3 different versions of PROMELA models where the Spin model checker produced verification results, both for assertion statements within the models, as well as LTL properties defined as LTL-P1 and LTL-P2. Version 1 consists of simple message exchange of the HVAC scenario without a conflict resolution mechanism, for which verification identified message loss within the bus. Version 2 includes a non-deterministic conflict resolution mechanism (protocol) where verification is successful for the resulting state space. The final version for which we evaluated our LTL properties includes asynchronous clocks in order to infer to message delays due to UART common errors. Verification results are shown in Table table 1. Successful verification of certain serial communication characteristics (e.g. $\gamma$ parameters in LTL-P1 for UART) with respect to time, will provide answers as to whether there are delays due to message exchange problems that cause overhead in the communications. If verification fails, we relax the

**Table 1.** PROMELA model verification results using Spin Model Checker

| Model Version | States | Transitions | Memory | Exp. Time | Result |
|---|---|---|---|---|---|
| Version 1 | 3031 | 6137 | 612MB | 0.05sec | Fail |
| Version 2 | 44285 | 304015 | 611MB | 1.65sec | Success |
| Final Version | 45706 | 866430 | 622MB | 4.4sec | Success |

time constraints of the LTL property and rerun the experiment in order to produce a successful verification of the overall communications. Total relaxed time will create an additional delay overhead to the communication instructions executed, and thus force an update to our VDM-RT instruction execution duration parameters. For similar real-time model checking studies the reader can refer to [14].

## 5 Co-Simulation Using the INTO-CPS Platform

In this section, we present the combination of co-simulation and formal verification. Co-simulation allows us to combine the capabilities of various domain-specific tools to validate more complex scenarios. As described in section 3 and modelled in section 4, our co-simulations results are based on a HVAC proof of concept use-case with FMUs extracted from Dymola and Overture tools (fig. 6).

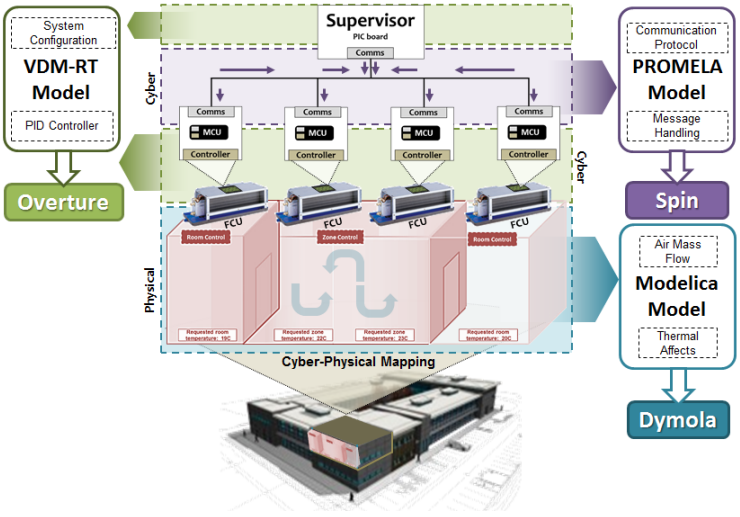


**Fig. 6.** HVAC proof of concept use case for proposed approach

The timing estimates verified with Spin were taken and injected into the VDM-RT model. This model is then co-simulated with the Dymola model, using the INTO-CPS

COE, a standalone tool for running FMI co-simulations with both fixed and variable step-size algorithms. Figure 7 shows co-simulation results for a scenario where the user requests set point changes outside the range allowed by the supervisor. We can see that, unlike the results shown from the standalone simulation, the supervisor steps in and ensures that the temperature is kept within the defined range – this is an example of the kinds of scenarios that co-simulation approaches can enable.

The results from fig. 7 are not particularly sensitive to the timing estimates of the communication between supervisor and controllers. This is because temperature variations takes minutes to assert themselves, as described by the CT model, whereas the communications occur within milliseconds. Thus, even though the communication estimates from SPIN are significantly slower than the implicit times from VDM-RT – a supervisor loop takes 26 times longer to complete – the overall effect on the temperature is minimal.

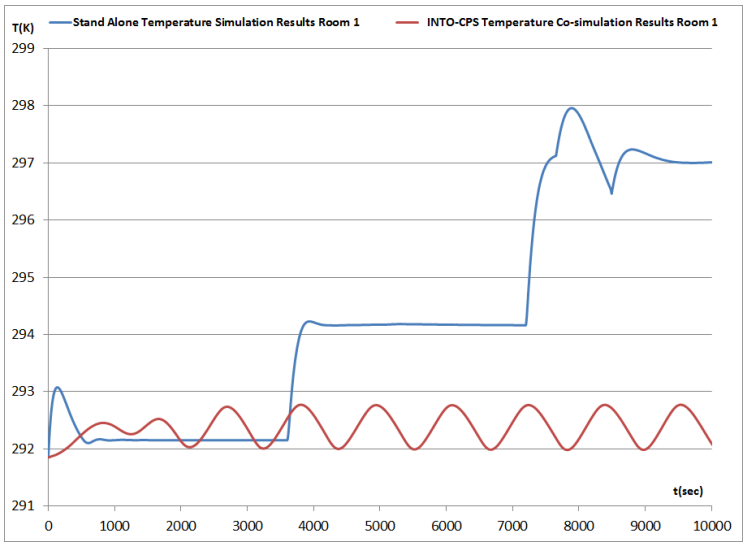


**Fig. 7.** Comparison between standalone simulation and INTO-CPS co-simulation results for Room 1 Temperature.

Nonetheless, the timing estimates are affecting the co-simulation results, as can be seen from a close inspection of the co-simulation output logs, as shown in fig. 8. Co-simulation for Room 1 temperature, indicates different control signals (56.55) compared to the Spin verified model (56.15) version, where timing constraints have been successfully verified. VDM-RT model timing parameters of the FCU controllers, were altered (decreased) in order to pass the verification occurred by Spin model checker. Such a result, although indicative for the purpose of this work, could reveal important outcomes for large-scale system co-simulations, with distributed objects sharing common com-

munication mediums. Finally, it is worth noting that in fig. 7 the room temperature in the co-simulation exhibits much greater oscillation at the co-simulation level than in the standalone simulation (compared to the standalone Dymola simulation result).

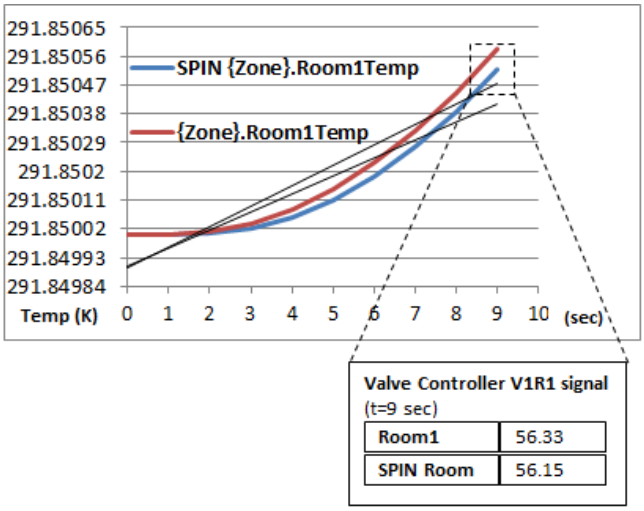


**Fig. 8.** Co-simulation results detail for valve opening times based on verification results injection.

PI controllers are very sensitive to sampling time and the PI controller in the co-simulation is modelled in VDM-RT (an imperative, DE notation) where sampling time is harder to control, particularly at the co-simulation level. In general, we have observed that this kind of DE formalism and tool is not ideal for modelling PI controllers. While this decision yielded significant benefits in terms of facilitating distributed discrete communication, the PI performance is a significant drawback.

## 6 Conclusions and Future Work

We have presented an approach for combining formal verification with co-simulation by means of results injection. This approach enables verification engineers to be brought into a closer collaboration loop with the multi-disciplinary teams that use co-simulation platforms to carry out CPS design and development. We have applied the approach to a case study from the HVAC domain, though we argue the approach is generic enough to be applied to other CPS domains – higher criticality domains may particularly benefit from it. This work will enable additional opportunities to extend it further based on evolving co-simulation results and INTO-CPS project features. We have identified two particularly interesting domains that include a) code generation of the verified solution as well as, b) model based testing of co-models. INTO-CPS code generators

currently are being developed with a focus on distributed aspects, which will give us additional motivation to explore concurrency requirements. Capability of generating C code for a simplified version of the VDM-RT model is currently enabled, though future improvements to the generators will extend them to support the complete model for experimental hardware platforms. In the future, we will validate the code generators by deploying the code onto hardware platforms and performing Hardware-in-the-Loop co-simulations. Furthermore, model-based testing as an INTO-CPS plugin could be combined with our approach as an alternative to state space explosion cases.

Finally, we plan to address the performance verification of PI controllers in the co-simulation by increase our co-model heterogeneity. The PI controllers will be modelled in Simulink [18] and connected to the co-simulation as FMUs, while a supervisor model handling master-slave FCU policies will be still modelled in VDM-RT. This will bring additional challenges in terms of verifying communications between the supervisor and the controllers at the model level, but we expect again the PROMELA model to guide our model parameterization.

**Acknowledgments:** This work is supported by the INTO-CPS H2020 project: *Integrated Tool Chain for Model-based Design of Cyber-Physical Systems*. Funded by the European Commission-H2020, Project Number:664047

## References

1. Amalio, N., Cavalcanti, A., Miyazawa, A., Payne, R., Woodcock, J.: Foundations of the SysML for CPS modelling. Tech. rep., INTO-CPS Deliverable, D2.2a (December 2016)
2. Bernardeschi, C., Masci, P., Pfeifer, H.: Early prototyping of wireless sensor network algorithms in pvs. In: Harrison, M.D., Sujan, M.A. (eds.) Computer Safety, Reliability, and Security. pp. 346–359. Springer Berlin Heidelberg, Berlin, Heidelberg (2008)
3. Bernardeschi, C., Masci, P., Pfeifer, H.: Analysis of wireless sensor network protocols in dynamic scenarios. In: Proceedings of the 11th International Symposium on Stabilization, Safety, and Security of Distributed Systems. pp. 105–119. SSS '09, Springer-Verlag, Berlin, Heidelberg (2009)
4. Blochwitz, T.: Functional Mock-up Interface for Model Exchange and Co-Simulation. https://www.fmi-standard.org/downloads (July 2014)
5. Brosse, E., Quadri, I.: SysML and FMI in INTO-CPS. Tech. rep., INTO-CPS Deliverable, D4.2c (December 2016)
6. Derler, P., Lee, E.A., Sangiovanni-Vincentelli, A.L.: Addressing modeling challenges in cyber-physical systems. Tech. Rep. UCB/EECS-2011-17, EECS Department, University of California, Berkeley (Mar 2011), http://www.eecs.berkeley.edu/Pubs/TechRpts/2011/EECS-2011-17.html
7. Duflot, M., Kwiatkowska, M., Norman, G., Parker, D.: A formal analysis of Bluetooth device discovery. Int. Journal on Software Tools for Technology Transfer 8(6), 621–632 (2006)
8. Elmqvist, H., Mattsson, S.E.: An introduction to the physical modelling of language modelica. Tech. rep., Proceedings of the 9th European Simulation Symposium (October 1997)
9. 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)

10. Fitzgerald, J., Gamble, C., Payne, R., Larsen, P.G., Basagiannis, S., Mady, A.E.D.: Collaborative Model-based Systems Engineering for Cyber-Physical Systems – a Case Study in Building Automation. In: Proc. INCOSE Intl. Symp. on Systems Engineering. Edinburgh, Scotland (July 2016)
11. Fitzgerald, J., Gamble, C., Payne, R., Pierce, K.: Method Guidelines 2. Tech. rep., INTO-CPS Deliverable, D3.2a (December 2016)
12. Holzmann, G.J.: The model checker spin. IEEE Transactions on Software Engineering 23(5) (May 1997)
13. Kleijn, C.: Modelling and Simulation of Fluid Power Systems with 20-sim. Intl. Journal of Fluid Power 7(3) (November 2006)
14. Lamport, L.: Real-time model checking is really simple. In: Proc. of the 13 Int. Conf. on Correct Hardware Design and Verification Methods. pp. 162–175. CHARME, Berlin (2005)
15. Larsen, P.G., Battle, N., Ferreira, M., Fitzgerald, J., Lausdahl, K., Verhoef, M.: The Overture Initiative – Integrating Tools for VDM. SIGSOFT Softw. Eng. Notes 35(1), 1–6 (January 2010), http://doi.acm.org/10.1145/1668862.1668864
16. Larsen, P.G., Fitzgerald, J., Woodcock, J., Fritzson, P., Brauer, J., Kleijn, C., Lecomte, T., Pfeil, M., Green, O., Basagiannis, S., Sadovykh, A.: Integrated Tool Chain for Model-based Design of Cyber-Physical Systems: The INTO-CPS Project. In: CPS Data Workshop. Vienna, Austria (April 2016)
17. Larsen, P.G., Lausdahl, K., Battle, N., Fitzgerald, J., Wolff, S., Sahara, S., Verhoef, M., Tran-Jørgensen, P.W.V., Oda, T.: The VDM-10 Language Manual. Tech. Rep. TR-2010-06, The Overture Open Source Initiative (April 2010)
18. MathWorks: http://www.mathworks.com/ (October 2011), simulink official website
19. Olveczky, P.C., Thorvaldsen, S.: Formal modeling, performance estimation, and model checking of wireless sensor network algorithms in real-time maude. Theoretical Computer Science 410(2), 254 – 280 (2009), http://www.sciencedirect.com/science/article/pii/S0304397508006683, distributed Computing Techniques
20. Ouy, J., Lecomte, T., Christiansen, M.P., Henriksen, A.V., Green, O., Hallerstede, S., Larsen, P.G., ger, C.J., Basagiannis, S., Couto, L.D., din Mady, A.E., Ridouanne, H., Poy, H.M., Alcala, J.V., König, C., Balcu, N.: Case Studies 2, Public Version. Tech. rep., INTO-CPS Public Deliverable, D1.2a (December 2016)
21. Philips Semiconductors Corp.: SCC2691 UART Data Sheet. Tech. rep. (May 2006)
22. Timothy, S.: A survey of control technologies in the building automation industry. In: Proceedings of the IFAC 16th world Congress; Prague; Czech Republic. pp. 13–96 (2005)
23. Verhoef, M., Larsen, P.G., Hooman, J.: Modeling and Validating Distributed Embedded Real-Time Systems with VDM++. In: Misra, J., Nipkow, T., Sekerinski, E. (eds.) FM 2006: Formal Methods. pp. 147–162. Lecture Notes in Computer Science 4085, Springer-Verlag (2006)