Multi-Paradigm Discrete-Event Modelling and Co-simulation of Cyber-Physical Systems

Mihai NEGHINA[1]*, Constantin-Bala ZAMFIRESCU[1], Peter Gorm LARSEN[2], Kenneth LAUSDAHL[2], Ken PIERCE[3]

[1] Lucian Blaga University of Sibiu, Faculty of Engineering, Department of Computer Science and Automatic Control, 4 Emil Cioran Street, Sibiu, 550025, Romania
{*mihai.neghina,zbcnanu*}*@gmail.com* (**Corresponding author*)

[2] Aarhus University, Department of Engineering, Inge Lehmanns Gade 10, 8000 Aarhus C, Denmark
{pgl,lausdahl}@eng.au.dk

[3] School of Computing Science, Newcastle University, 1 Science Square, Science Central, Newcastle upon Tyne, NE4 5TG, UK
kenneth.pierce@newcastle.ac.uk

Abstract: In the modelling of Cyber-Physical Systems (**CPS**s), there are different possible routes that can be followed to gradually achieve a collection of constituent models that can be co-simulated with a high level of accuracy. This paper demonstrates a methodology which initially develops all constituent models at a high level of abstraction with discrete-event models expressed using the Vienna Development Method (**VDM**). Subsequently, a number of these are refined (without changing the interfaces) by more detailed models expressed in different formalisms, and using tools that can export Functional Mock-up Units (**FMU**s) for co-simulation through the Functional Mock-up Interface (**FMI**) standard. The development team of each of these more detailed models can then experiment with the interactions with all the other constituent models, using the high-level discrete-event versions until higher-fidelity alternatives are ready. The results reported in this paper were obtained in an innovation experiment within the EU CPSE Labs research project, part of Smart Anything Everywhere initiative.

Keywords: Co-Simulation, Cyber-physical Production Systems, Heterogeneous modelling.

1. Introduction

In the development of CPSs, a model-based approach can be an efficient way to master system complexity through an iterative development. In this paper we illustrate how a co-simulation technology [6] can be used to gradually increase the detail in a collaborative model (co-model) following a "discrete event first" methodology. In this approach, initial abstract models are produced using a discrete event (**DE**) formalism (in this case VDM) to identify the proper communication interfaces and interaction protocols among different models. These are gradually replaced by more detailed models using appropriate technologies, for example continuous time (**CT**) models of physical phenomena.

The case study deals with the virtual design and validation of a CPS-based manufacturing system for assembling USB sticks that was developed in the iPP4CPPS project [7]. It is a representative example of a part of a distributed and heterogeneous system in which products, manufacturing resources, orders and infrastructure are all cyber-physical [17]. In this setting, several features (such as asynchronous communication, messages flow, autonomy, self-adaptation, etc.) should be investigated at design time, for example using a collaborative modelling approach. Consequently, the case study offers a balance between being sufficiently simple to be easily followed as a production line example, including generating a tangible output, and at the same time being sufficiently general to allow the study of the co-simulation complexity. Furthermore, by choosing a USB stick, the example opens the possibility of extending the purpose of the study to interactions between the generated hardware and the generated software solutions in the production line.

The paper demonstrates the effectiveness of the DE-first approach in a setting with many different constituent models. To enable any number of DE and CT constituent models to be co-simulated, the iPP4CPPS project employed the INTO-CPS technology [2, 13] that generalizes previous results in combining different models, such as one DE and one CT model [5]. The remaining part of this paper starts by introducing the INTO-CPS technology. This is followed by Section 3 describing the CPS development approach for starting out with DE models for each constituent model and then gradually exchanging some of these with more realistic models. Section 4 introduces the industrial case study and presents the results of using co-simulation technology

https://doi.org/10.24846/v27i1y201804

both in a homogeneous and heterogeneous setting on that case study. Finally, Section 5 provides some concluding remarks and future uses of this approach. The paper is an extended version of the presentation given in the 15th Overture Worksop [16].

2. The INTO-CPS Technology

In the INTO-CPS project we start from the view that disciplines such as software, mechatronics and control engineering have evolved with notations and theories that are tailored to their engineering needs, and that it is undesirable to suppress this diversity by enforcing uniform general-purpose models [3,4,10,11,12]. Our goal is to achieve a practical integration of diverse formalisms at the semantic level, and to realise the benefits in integrated tool chains. The overall workflow and services from the tool chain used in this project are illustrated in Figure 1.

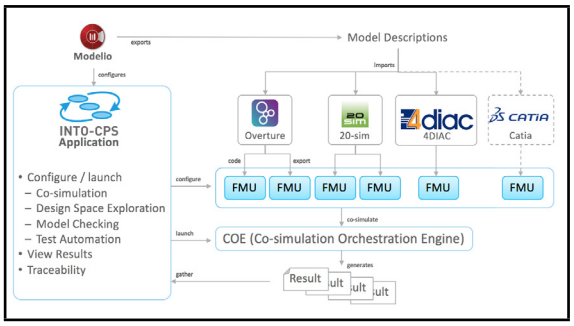

Figure 1. The INTO-CPS tool chain used in this project

At the top level, the tool chain will allow requirements to be described using SysML with a new CPS profile made inside the Modelio tool [15]. This SysML profile allows the architecture of a CPS to be described, including both software and physical elements and based on this it is possible to automatically generate FMI model descriptions for each constituent model [18]. It is also possible to automatically generate the overall connection between different FMUs for a co-simulation. Note that although SysML also have diagrams to express behaviour the CPS profile in Modelio has not been extended to enable the generation of FMUs from such diagrams, so SysML in this work is primarily used at the high-level architecture describing how different constituent models are combined. It is also worth noting that the types that can be used in the interfaces for FMUs are quite basic so elaborate VDM values can only be exchanged using strings and in that case one need to have the same understanding of such structured values in the constituent models exchanging such values.

This type of FMI model description can subsequently be imported into all the baseline modelling and simulation tools included in the INTO-CPS tool suite. All of these can produce detailed models that can be exported as FMUs, each acting as independent simulation units that can be incorporated into an overall co-simulation. The constituent models can either be in the form of DE models or in the form of CT models combined in different ways. Thus, heterogeneous constituent models can then be built around this FMI interface, using the initial model descriptions as a starting point. A Co-simulation Orchestration Engine (**COE**) then allows these constituent models of a CPS to be evaluated through co-simulation. The COE also allows real software and physical elements to participate in co-simulation alongside models, enabling both Hardware-in-the-Loop and Software-in-the-Loop simulation.

The different modelling and simulation tools used in this experiment included Overture [8], 20-sim [8] and 4DIAC [19] technologies. The original intention was also to include Catia to model the robotic arm in the case study, but unfortunately a license was not available for the version of this tool capable of generating FMUs. The benefit of the co-simulation approach is that Catia could be replaced by an equivalent 20-sim model.

In order to have an user-friendly interface to manage this process, a web-based INTO-CPS Application has been produced. This can be used to launch the COE, enabling multiple co-simulations to be defined and executed, and the results to be collated and presented automatically. The tool chain allows multiple co-simulations to be defined and executed using its design space exploration capabilities.

3. Discrete-Event First Modelling with INTO-CPS

3.1 Initial Models

Given a set of FMI model descriptions generated from a SysML model using Modelio, initial creation of the constituent models can begin. In general, this means importing the model descriptions into modelling tools (e.g. Overture,

20-sim), modelling behaviour in each, generating FMUs and then bringing them together in the INTO-CPS Application to run a co-simulation.

This direct approach can however be prone to failure. It requires that FMUs are available for all constituent models before co-simulations can be made. If each model is produced by a different team, then delays in one team may well delay all other teams. Similarly, if a single team produces constituent models in turn, then co-simulation can only begin at the end of modelling.

One way to overcome this challenge is to produce quick, first version FMUs as early as possible to perform initial integration testing. These can then subsequently be replaced iteratively by increasingly detailed models, with integration testing performed each time a model is available, falling back to older versions as required. This however may be more difficult to do in some modelling paradigms.

An alternative is to take a DE-first approach. Here, a simple, abstract model of the entire system is created in the DE modelling environment, e.g. VDM and Overture, in order to sketch out the behaviour of all constituent models. This approach in a two-model setting is described in Fitzgerald et al. [5], including guidelines for simplifying continuous behaviour in DE models. It is worth noting that this type of approach is conceptually similar to a traditional component-based development approach, where stub-modules are commonly used in the initial phases.

3.2 Discrete-Event First with VDM/Overture

Given a SysML model and model descriptions for each constituent model, the suggested approach is to begin by building a single VDM-Real-Time (**VDM-RT**) [20] project in Overture with the following elements:

- A class for each constituent representing an FMU. Each class should define *porttype* instance variables (i.e. of type *IntPort*, *RealPort*, *BoolPort* or *StringPort*) corresponding to the model description and a constructor to take these ports as parameters. Each FMU class should also define a thread that calls a Step operation, which should implement some basic, abstract behaviour for the FMU.

- A System class that instantiates port and FMU objects based on the connections diagram. Ports should be passed to constructor of each FMU object. Each FMU object should be deployed on its own CPU.

- A *World* class that starts the thread of each FMU objects.

Class and object diagrams giving an example of the above is shown in Figure 2.

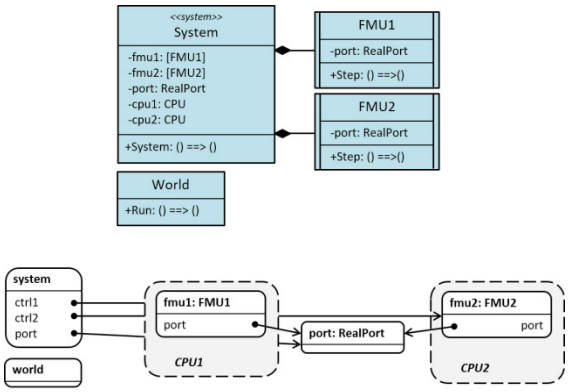

Figure 2. Class diagram showing two simplified FMU classes created within a single VDM-RT project, and an object diagram showing them being instantiated as a test

In this example, there are two constituent models (called FMU1 and FMU2) joined by a single connection of type real. Such a model can be simulated within Overture to test the behaviour of the FMUs. Once the behaviour of the FMU classes has been tested, FMUs can be produced for each and integrated into a first co-model. To generate FMUs, a project must be created for each constituent model, comprising:

- One of the FMU classes from the main project.

- A *HardwareInterface* class that defines the ports and annotations required by the Overture FMU export plug-in, reflecting those defined in the model description.

- A system class that instantiates the FMU class and passes the port objects from the *HardwareInterface* class to its constructor.

- A *World* class that starts the thread of the FMU class.

The above structure is shown in Figure 3. A skeleton project with a correctly annotated *HardwareInterface* class can be generated using the model description import feature of the Overture FMU plug-in. The FMU classes can be linked into the projects (rather than hard

copies being made) from the main project, so that any changes made are reflected in both the main project and in the individual FMU projects. Note that if the FMU classes need to share type definitions, these can be created in a class (e.g. called Types) in the main project, and then this class can be linked into each of the FMU projects in the same way.

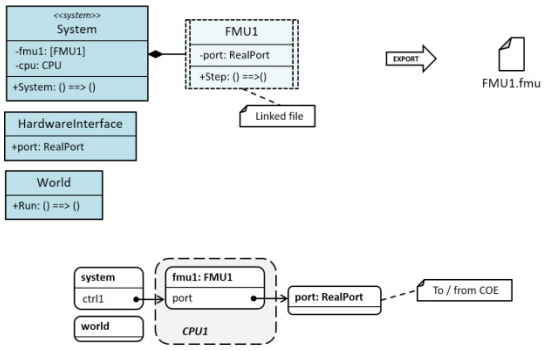

Figure 3. Class and object diagrams showing a linked class within its own project for FMU creation

From these individual projects, FMUs can be exported and co-simulated within the INTO-CPS tool chain. These FMUs can then be replaced as higher-fidelity versions become available. They can also be retained and used for regression and integration testing through different co-model configurations for each combination.

4. Case Study: Manufacturing USB Sticks

The case study below concerns the manufacturing of USB sticks. The case study was chosen as a representative plant that might potentially be expanded into a full production line.

The USB sticks considered have two lids and the main body. The production line received orders from virtual users, containing the requested characteristics of the item, as well as additional information, such as urgency of the order (the requested transportation speed between the warehouse and the test station), change requests (indicating a new set of stick characteristics to be assembled and tested) or cancellation requests (for dropping orders). The assembly takes place in the warehouse unit, and the generated item is moved on wagons to the test station for validating the components of the item. If the item is rejected (the test fails to confirm the requested characteristics of the stick), then the process is automatically re-started, and a new request is made to the warehouse unit.

4.1 The Constituent Systems

Figure 4 shows the plant layout as realized during the project, with emphasis on the physical entities where the control units will be embedded. These are:

W: Warehouse - buffer unit;

R: Robotic Arm - processing station;

C: Wagons – transportation units;

T: Test Station - processing station.

In addition, to capture the value adding processes in Industry 4.0 [14,18,20], the case study includes distinct units to reflect the users who place orders for assembling the USB sticks, and the required infrastructure that make possible for the others CPSs to exist. These are:

H: HMI (Human-Machine Interface),

P: Part Tracker - reflecting the infrastructure

U: Unity – dynamic 3D graphics of the simulation, providing the image in Figure 4.

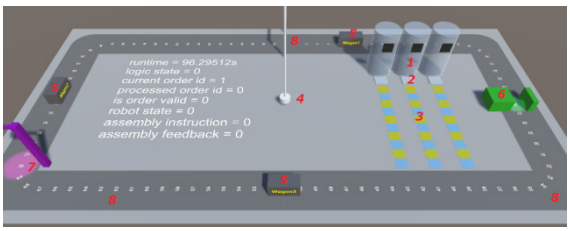

Figure 4. Layout of the production line as depicted in the 3D rendering of the simulation: 1) the warehouse stacks; 2) the assembly box at the base of the warehouse stacks; 3) the memory boxes of the warehouse unit; 4) the robotic arm for moving parts around the warehouse; 5) wagons on different locations of the track; 6) the loading station; 7) the test station; 8) the circular track for the wagons

The HMI unit handles the user interface; it communicates only with the Part Tracker. On the deployment of the solution, the HMI allows for real-time placing of orders, change requests and cancellations through an app running on smart devices. The Part Tracker is the central logical unit that handles orders; it communicates with all other units except the Robotic Arm.

The Warehouse assembles the USB from the component parts; it consists of stacks for each type of component, an assembly box for the actual assembly of the items and memory

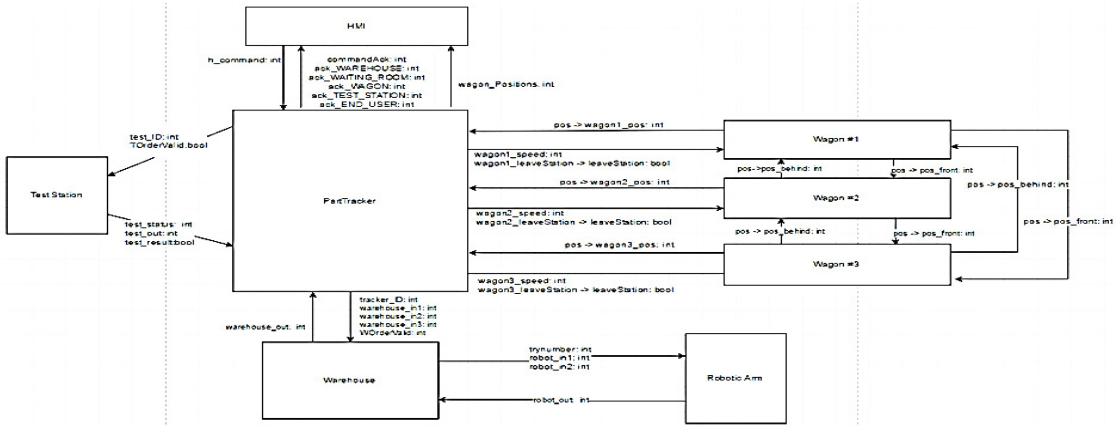

Figure 5. Connections between VDM models

boxes for storing components that do not fit the current order. The memory boxes may also be a source of components for new orders, if the requested colour is available. It communicates only with the Part Tracker and the Robotic Arm, used for moving USB components from one location to another, including the waiting line at the loading station.

The Robotic Arm moves parts or pieces from one location to another; it communicates only with the Warehouse. As with other models, the Overture/VDM-RT model of the Robotic Arm is on purpose incomplete, since the time required to move a part or piece from one location to another is implemented simply as a delay through a countdown timer.

The purpose of the Wagons is to transport the items from the waiting room (at the loading station) to the test station. The wagons communicate with the Part Tracker and to each other. Each wagon can be certain of its location only at the loading or test station, while in between the position is estimated periodically from the previous known position, time and own speed.

The Test Station reads the item characteristics and reports on their conformity to requirement. It communicates only with the Part Tracker, from which it receives the requested set of colours for the item and to which it reports the test result. From the test station the item is then pushed either to the pile of rejected items or to the end user, while the empty wagon returns to the loading station for transporting further items.

Besides these units, the co-simulation includes a 3D rendering unit that dynamically displays what is happening in the simulation.

4.2 Homogeneous Co-Simulations

The first step in the modelling of the case study was generating a functional, albeit not fully featured, model of the components (units) that would make up the USB stick production line. Each component was first modelled abstractly in VDM by using the Overture tool, following the approach described in Section 3. The goal of this homogeneous co-simulation was to identify the right interaction protocols (signals) among the various components (stations) of the prototype, and not on the model's accuracy. Therefore, the VDM simulation model includes distinct models for each component of the system (see Figure 5).

Having an early (working) co-simulation also provides other advantages, such as validating the interaction protocols, decomposing the project into units which could then be worked on separately, and the possibility of gradually increasing the complexity of the simulation and replacing units of the model one by one with more accurate FMUs generated from dedicated platforms.

The VDM models do not need to have complete functionality for the homogeneous co-simulation, only the bare minimum from which the communication lines between units can be determined. The incompleteness of the VDM models is related to details of the inner workings of the components, not necessarily respecting all the constraints of reality, such as randomly generating colours for USB parts and ignoring the physical capacity limit of the memory boxes in the warehouse, randomly choosing a duration for piece relocation for the robotic arm, or randomly

considering the test successful or unsuccessful in the test station. Another example is breaking continuity: the warehouse model generates random colours for USB parts and can react to order cancellations. But if an order comes and the warehouse was able to select the correct colour for two of the three parts of the USB before the order was cancelled, when a new order arrives the warehouse just starts generating coloured parts anew and those leftover coloured parts are not considered (the reality of already being a part of a certain colour in a certain place is ignored and continuity is broken). Such aspects of the functionality are minor details with respect to the DE modelling used for the abstract validation. The internal states however are all well-established, along with the communication patterns and lines between modules, such that the behaviour of the refined modules does not diverge substantially from the behaviour of the abstract models. Once established, the communication lines and the types of data they carry become hard constraints of the simulation that cannot be easily changed, but new lines of communication could be added if necessary. For instance, new communication lines have been added later in the development of the project, for transmitting (to the Part Tracker) the level of perturbation recorded by each unit.

The Comma-Separated Variables (**CSV**) library from the Overture tool is used in the HMI model to provide test data for orders in the production line. This approach is both flexible and powerful. The flexibility stems from the possibility of creating various scenarios with various amounts of orders and order/cancellation time delays for covering statistical scenarios, while the power comes from the repeatability of experiments. Having the same input CSV file, the co-simulation can be run with various parameters, but having exactly the same input orders at exactly the same time, thus generating a detailed picture of the behaviour of the system in controlled, repeatable experiments. The influence of certain system parameters may also be analysed.

The VDM model for the Warehouse is on purpose incomplete, since it randomly generates parts of various colours until the necessary (requested) colour is available. There is no stack of parts with pre-defined colours (therefore there is no possibility of ever running out of a colour) and there is no memory box where unused parts are deposited temporarily until a stick request with those characteristics comes around.

The abstract VDM model for the Wagons is idealised in the sense that the wagons can change speeds (accelerate or decelerate) immediately after receiving the command from the Part Tracker, (un)loading is instantaneous and that their estimated positions always correspond to the real positions on the track, which probably will not necessarily be true in the real implementation. Also, the wagons have no possibility of stopping unexpectedly, losing the load or falling outside the track.

The abstract VDM model of the Test Station is also incomplete. It simply waits for a time and randomly outputs true or false as the test result. The only parameter of controlling the output is the frequency of rejecting an item.

4.3 Communication between Units

The communication between units contains both simple (straightforward) messages, requesting the setting of a certain value or indicating the current value or state of a component, as well as composed messages that need to be decoded and the information extracted from them before that information can become useful. The purpose of the composed messages is twofold: to ensure that certain bits of information arrive simultaneously (as opposed to them coming on different message lines that may become unsynchronised or for which further synchronisation logic might have been needed) and to account for the possibility of coded messages (that might be interesting in applications with significant noise, where error correcting codes might become useful).

For instance, request from the Part Tracker to the Wagons to assume a certain speed or the feedback of the Wagon positions to the Part Tracker is done with straightforward messages (the value requested) on dedicated lines (that only carry these types of messages and nothing else). On the other hand, the order requests from the HMI to the Part Tracker or their acknowledgement (feedback) contain multiple pieces of information each. Finally, the perturbation levels of all the units are gathered by the Part Tracker, merged into one composed message and sent together to the HMI for display purposes.

4.4 Heterogeneous Co-Simulations

The heterogeneous co-simulation covers the successful attempts to replace the Overture-generated FMUs with more detailed and complete FMUs generated by specialized tools using appropriate formalisms. Table 1 shows the correspondence between the use case units and the adequately suited program for implementing a complete simulation for each. As may be noted the complete simulation covers all the required types of CPSs included in the reference architecture described in [21].

Having already established a homogeneous co-simulation (with all FMUs generated from Overture), the improvements to various parts of the project can be achieved independently from each other, because any newly generated FMU would simply replace the corresponding Overture/VDM-RT FMU if the interfaces between units remained unchanged. Thus, the order of integration for completely-functional FMUs is determined by the progress of individual teams rather than pre-defined dependencies. Teams were able at any time to rely on a working DE-first co-simulation and only replace their unit. Furthermore, the co-simulation could at any time be assembled from any combination of FMUs (generated by Overture or other tools).

Table 1. Technologies used for different system units

Type	System Units	Technology
Planning and control	HMI	4DIAC + MQTT Overture (VDM)
Infrastructure	Part Tracker	Overture (VDM)
Production	Warehouse	20-sim
Production	Robotic Arm	Catia v6
Production	Wagons	4DIAC
Production	Test Station	4DIAC
General overview	Unity	20-sim / Unity animation

While for most units the FMU generated by the specialised program is simply an improvement (a more realistic or more detailed simulation) over the Overture model, the test case captures two exceptions: the HMI and the Part Tracker. The HMI is different from the rest of the units in the sense that the two FMUs are meant to complement each other, although they cannot be used both at the same time. The Overture/VDM-RT FMU reads the orders from a CSV file, and can be used for performing benchmark like tests with completely controlled and repeatable sequences of orders. The 4DIAC-MQTT FMU implements user heuristics, allowing for real-time placement of orders and real user interaction with the simulation. Furthermore, because orders can be requested from a smartphone or tablet, the 4DIAC-MQTT FMU also implements a graphical interface. On the other hand, since the Part Tracker handles the flow of the process, it is a logical unit rather than a physical one. Therefore, the Part Tracker is the only unit whose FMU is generated in Overture throughout the heterogeneous co-simulation phase, also being the only complete VDM model.

For one of the other units, the flexibility of the co-simulation allows for a back-up plan to be used. As described in Table 1, the generation of an FMU for the Robotic Arm was intended to be produced in Catia v6, of which unfortunately only the academic version of 2013 was available to the team. This presented a major problem, because the libraries necessary for generating an FMU were theoretically available only since Catia V6 2015 FD01 (Dymola) and the support received from the producer of the tool was rather limited. Because of the modularity of the co-simulation, it was possible to use 20-sim to produce an alternative FMU and replace the envisaged Catia FMU model.

The co-simulation in the heterogeneous phase allows for the analysis of interactions between the units that have been simulated using specialised programs. All messages exchanged between VDM FMUs (or their more rigorous versions) are available for display in the co-simulation engine and INTO-CPS Application. Figure 6 shows the communication between the HMI unit and the Part Tracker for two orders.

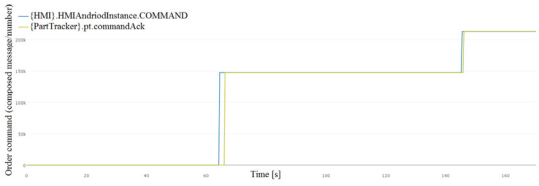

Figure 6. Orders and acknowledgements exchanged between HMI and Part Tracker

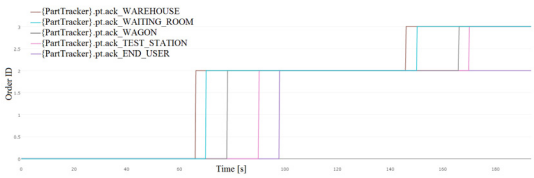

Figure 7. Acknowledgement messages for stages of assembly sent by the Part Tracker

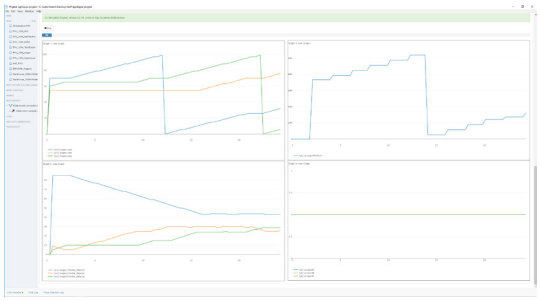

Figure 8. Wagon positions on a circular track of length 100

Two orders are sent from the HMI and acknowledged by the Part Tracker. The progress of assembly on the orders is clearly noticeable in Figure 7 because acknowledgement messages are sent back to the HMI whenever the item reaches the next stage. Furthermore, the acknowledgement messages contain the ID of the order, which would make the process easy to follow even in case of items being assembled in parallel, as if in a pipeline. Figure 8 shows the capability of the co-simulation tool to display multiple graphs simultaneously, especially useful when trying to visualize signals having different scales. The positions of the wagons on the circular track of length 100 (upper left graph) and the frontal distance as reported by each wagon (lower left graph) have the range 0 to 99, but composed message containing the positions of all wagons, sent by the Part Tracker to the HMI (upper right graph) goes into the millions, and the intended speed of the wagons (lower right graph) rarely goes above 10.

4.5 Deployment

The units modelled and tested by the heterogeneous co-simulation have been deployed in a demo stand for fine tuning under real-life conditions. Figure 9 shows the actual demo stand layout, with all physical units present.

The HMI is a virtual unit, which was the first to be implemented on smartphones and tablets as an app, communicating with the other units through MQTT even during the heterogeneous phase of the project. The Part Tracker is a logical unit for which the FMU was generated by the Overture tool in both the homogeneous and heterogeneous phases. For deployment, the C code was generated directly from Overture and deployed on a Raspberry Pi 3, also employing MQTT as communication protocol [1].

The Warehouse, mainly consisting of stacks and memory locations, colour sensors and pneumatic pistons, uses a Raspberry Pi 3 with UniPi Expansion Board for deploying the 20-sim generated C code of the heterogeneous co-simulation. The Robotic Arm is a Stäubli robot with no internal logic, just following the lead of the Warehouse. The Wagons, being the moving parts of the stand, contain DC motors (with PWM drivers), position sensors for detecting the loading and testing station, anti-collision ultrasonic sensors to avoid collision with other wagons and embedded Raspberry Pi boards for the internal logic generated from the 4DIAC FMU. Finally, the testing station uses a camera for image processing and also includes actuators to push the USB sticks from the wagons into the pile of rejected items or towards the end user.

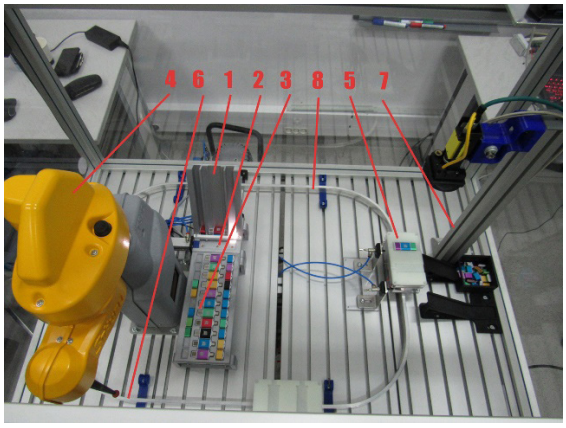

Figure 9. Demo stand for deployment of the co-simulated units, containing: 1) the warehouse stacks; 2) the assembly box at the base of the warehouse stacks; 3) the memory boxes of the warehouse unit; 4) the robotic arm for moving parts around the warehouse; 5) wagons on different locations of the track; 6) the loading station; 7) the test station; 8) the circular track for the wagons

5. Concluding Remarks

This paper presented a methodology for modelling CPSs, as well as the steps for achieving a working heterogeneous co-simulation (with units modelled in various dedicated tools) for the development of an assembly line of USB sticks. The test case

chosen was sufficiently simple to be easily followed, but still with enough complexity to allow the exploration of the capabilities and the limitations of both the methodology and the tools. The production line was first modelled completely in Overture/ VDM-RT and the models were co-simulated using the INTO-CPS technology (a homogeneous co-simulation). The flexibility of the co-simulation engine allowed for the gradual integration of closer-to-reality and more detailed FMUs, generated in dedicated tools (4DIAC, 20-sim).

The test case also emphasised key possibilities of the methodology, such as:

- The initial development of the homogeneous co-simulation in VDM was particularly useful in driving cooperation and making clear the assumptions of the distributed teams involved in modelling the specific components; this phase proved to be the most difficult and time-consuming phase in building the co-simulation, requiring a very intensive communication for a shared understating of the requirements;

- Once the VDM co-simulation is running, the independent development of units may be integrated and validated in the co-simulation in any order, and using any formalism, e.g. some units to remain modelled in Overture, while the others in their own formalisms;

- An improved capability to handle some unpredictable requirements; the employment of co-simulations when designing an automated production system avoids the build-up inertia of subsequent design constraints, facilitating the low and late commitment for these decisions, i.e. the specific controllers or PLCs, the plant layout, the number of storage stacks from the warehouse, etc.

Further investigations in the co-simulation capabilities on the current test case include:

- The improvement of visualisation and debugging features for co-simulations;

- The inclusion of perturbations in the production line for a more realistic simulation;

- The optimisation of parameters for one or multiple units, depending on the perturbations, the amount and distribution of input orders and the physical constraints of the units (using the Design Space Exploration capabilities);

- The integration of FMUs generated by tools different from the ones mentioned in this paper (e.g. Catia for the robot arm);

- The possibility of extending the purpose of the study to interactions between generated hardware and generated software solutions, in the case of the production line, writing data onto the USB sticks and verifying it as part of the production process.

Acknowledgements

This work has been supported through the iPP4CPPS project (Horizon 2020, grant agreement no. 644400, experiment no. 16-UK-GERS-01) and DiFiCIL project (contract no. 69/08.09.2016, ID P_37_771, web: http://dificil.grants.ulbsibiu.ro), co-funded by ERDF through the Competitiveness Operational Programme 2014-2020.

REFERENCES

1. Bandur, V., Tran-Jørgensen, W. V. P., Hasanagić, M. & Lausdahl, K. (September 2017). Code-generating VDM for Embedded Devices. In *The 15th Overture Workshop: New Capabilities and Applications for Model-based Systems Engineering*, *CS-TR-1512-2017*.

2. Blochwitz, T. (2014). *Functional Mock-up Interface for Model Exchange and Co-Simulation*. <https://www.fmi-standard.org/downloads>.

3. Fitzgerald, J., Gamble, C., Larsen, P. G., Pierce, K. & Woodcock, J. (May 2015). 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.

4. Fitzgerald, J., Gamble, C., Payne, R., Larsen, P. G., Basagiannis, S. & Mady, A.E.D. (July 2016). Collaborative Model-based Systems Engineering for Cyber-Physical Systems – a Case Study in Building Automation. In *INCOSE 2016*, Edinburgh, Scotland.

5. Fitzgerald, J., Larsen, P. G. & Verhoef, M. (eds.) (2014). *Collaborative Design for*

Embedded Systems – Co-modelling and Co-simulation. Springer.

6. Gomes, C., Thule, C., Broman, D., Larsen, P. G. & Vangheluwe, H. (Feb 2017). *Co-simulation: State of the art. Tech. rep*, *arXiv.org 157*.

7. Hermann, M., Pentek, T. & Otto, B. Design Principles for Industrie 4.0 Scenarios. In: *2016 49th Hawaii International Conference on System Sciences* (*HICSS*) (pp. 3928-3937).

8. *Integrated product-production co-simulation for cyber-physical production system*, *iPP4CPS*, accessed on November 2017. <http://centers.ulbsibiu.ro/incon/index.php/ipp4cpps/ >.

9. Kleijn, C. (November 2006). Modelling and Simulation of Fluid Power Systems with 20-sim, *Intl. Journal of Fluid Power*, *7*(3).

10. Larsen, P. G., Battle, N., Ferreira, M., Fitzgerald, J., Lausdahl, K. & Verhoef, M. (January 2010). The Overture Initiative – Integrating Tools for VDM*, SIGSOFT Softw. Eng. Notes*, *35*(1), 1-6.

11. Larsen, P. G., Fitzgerald, J., Woodcock, J., Fritzson, P., Brauer, J., Kleijn, C., Lecomte, T., Pfeil, M., Green, O., Basagiannis, S. & Sadovykh, A. (April 2016). Integrated Tool Chain for Model-based Design of Cyber-Physical Systems: The INTO-CPS Project. In *CPS Data Workshop*, Vienna, Austria.

12. Larsen, P. G., Fitzgerald, J., Woodcock, J. & Lecomte, T. (September 2016). Chapter 8: Collaborative Modelling and Simulation for Cyber-Physical Systems, *Trustworthy Cyber-Physical Systems Engineering*. Chapman and Hall/CRC, iSBN 9781498742450.

13. Larsen, P. G., Fitzgerald, J., Woodcock, J., Nilsson, R., Gamble, C. & Foster, S. (2016). *Towards Semantically Integrated Models and Tools for Cyber-Physical Systems Design*, 171-186. Springer International Publishing, Cham.

14. Larsen, P. G., Fitzgerald, J. Woodcock, J., Gamble, C., Payne, R. & Pierce K. (September 2017). In *Features of Integrated Model-based Co-modelling and Co-simulation Technology*, *CoSim-CPS workshop organised in connection with the SEFM*, Trento, Italy.

15. *Modelio*, accessed on November 2017. <https://www.modelio.org/>.

16. Neghina, M., Zamfirescu, C. B., Larsen, P. G., Lausdahl, K. & Pierce, K. A (2017). Discrete Event-first Approach to Collaborative Modelling of Cyber-Physical Systems. In *The 15th Overture Workshop New Capabilities and Applications for Model-based Systems Engineering*, Newcastle University.

17. Nof, S. Y. (ed.) (2009). *Springer Handbook of Automation*. Springer-Verlag.

18. Quadri, I., Bagnato, A., Brosse, E. & Sadovykh, A. (December 2015). Modeling Methodologies for CyberPhysical Systems: Research Field Study on Inherent and Future Challenges, *Ada User Journal*, *36*(4), 246-253.

19. Strasser, T., Rooker, M., Ebenhofer, G., Zoitl, A., Sunder, C., Valentini, A. & Martel, A. (July 2008). Framework for distributed industrial automation and control (4diac). In *2008 6th IEEE International Conference on Industrial Informatics* (pp. 283-288).

20. Verhoef, M., Larsen, P. G. & Hooman, J. (2006). Modeling and Validating Distributed Embedded RealTime Systems with VDM++. In: Misra, J., Nipkow, T., & Sekerinski, E. (eds.), *FM 2006: Formal Methods*. *Lecture Notes in Computer Science*, *4085*, 147-162. Springer-Verlag.

21. Zamfirescu, C. B., Parvu, B. C., Schlick, J. & Zühlke, D. (2013). Preliminary Insides for an Anthropocentric Cyber-physical Reference Architecture of the Smart Factory, *Studies in Informatics and Control*, *22*(3), 269-278.