Foundations for Co-simulation

Cláudio Gomes

September 11, 2015

1 Problem Definition

Integration – the interconnection of the components that comprise a system – is identified as a major source of problems [46, 47, 64, 66] in the concurrent development of complex engineered systems. It is usually caused by assumptions about other components of the system having to be made early in the development of each component [2, 65]. For instance, the control unit of an Anti-lock Breaking System (ABS) might be built assuming that the speed sensor's output is in imperial units when in fact it is in metric units.

Modelling and simulation techniques are used to mitigate these issues: models of components are created [23] and simulated before any physical prototype is built. Simulation can also be used to analyse the behavior of interacting models of components: one might be interested in simulating not only the heat conduction model across the brake disc, but also the whole Anti-lock Breaking System (ABS) model, assumptions included. Different languages are used to model different components [27]: fuzzy control rules [3] for the control unit of the ABS [63] and partial differential equations for the heat conduction in brake pads [69]. The simulation of interactions between models specified in different languages is an open challenge [41], mostly done on a case-by-case basis, making it difficult to reuse to other scenarios. To aggravate, specialized suppliers of components are interested in protecting their Intellectual Property (IP) leading to the situation were the simulation needs to be made without access to the full models [12].

Co-simulation is a technique to couple multiple simulators, each simulating a single component, often seen as a black box, in order to perform simulation of the whole system. The lack of information makes wrongly made assumptions harder to detect. For example: mismatch between relative measures, different time scales, different data representations (continuous time signals vs discrete events), different physical units, non-causal dependencies between simulators [5], etc. Despite this, co-simulation shortens development time and improves quality, as reported by the industrial partners of the DESTECS project [17, 30, 60]. Due to its success, many co-simulation frameworks are now available but most of them are hand-coded point-to-point solutions that can couple two simulators. They restrict the languages available for the development of complex systems. In Bosch, for example, there are more than 100 different simulation tools [11]. The FMI [12] is an attempt to standardize the representation of models and simulators but falls short on capturing the interactions between these. In addition, the rapidly expanding [1, 4, 7, 14, 16, 28, 33–35, 45, 53, 56, 59, 62, 67, 70] state of the art knowledge cannot be easily reused in the development of new co-simulation frameworks [39].

There is a need to work on the fundamental techniques to couple simulators, study the kind of information necessary and maximize reuse of the knowledge scattered through the state of the art.

2 Strategic Objective

Co-simulation is a very active research field, with many open challenges and co-simulation frameworks available[1]. This project aims at reusing and contributing to the fundamental techniques that are currently scattered through the state of the art, promote its industrial adoption and enhance further research in this field.

A proven technique to maximize reuse is to study co-simulation scenarios as a product family [55] and explicitly describe the techniques necessary to overcome the challenges of each specific scenario [6].

Goal 1. I will apply domain engineering techniques to identify the commonalities, variabilities and missing features across several co-simulation scenarios and create a software product line for co-simulation scenarios supported by generative techniques [26]. I will use formalisms to describe the fundamental techniques to simulate the interactions between simulators in complex co-simulation scenarios, laying the foundations of co-simulation.

Upon successful implementation of Goal 1, I will have a framework which I can use to develop new analyses and optimizations.

Goal 2. I will extend the analyses for non-determinism [19], stability [61], correctness [37] and performance related optimizations [33], for complex co-simulation scenarios.

The innovative aspects of this project are: (i) the explicit representation of the fundamental behavior of complex co-simulation scenarios; and (ii) the application and development of new sophisticated analyses that make the most of the information available on each scenario.

This project will be developed at the Modelling, Simulation and Design Lab (MSDL), internationally known for its experience in modelling, language and software engineering, generative techniques and co-simulation. MSDL regularly collaborates with industrial partners such as The MathWorks, General Motors Research, IBM Research and Siemens PLM Software (previously known as LMS International).

[1] At the time of writing, I have found 20 tools whose authors claim to perform co-simulation.

3 Project Description

The high level goal of this project is to contribute to and reuse the fundamental techniques used for co-simulation. For the reuse part, I draw on the already proven [26] domain engineering techniques to implement a Software Product Line (SPL) [55] using generative techniques. For the contribution part, I will explore new co-simulation scenarios and the challenges that they pose. I will adapt current analyses and optimization techniques for those scenarios and I will attempt to make them applicable to any co-simulation scenario.

3.1 Background

To better understand how the tasks detailed in this section will help achieve those goals I need to give some background knowledge.

Co-simulation is a simulation performed by coupling multiple simulators. Each simulator is responsible for the simulation of one model. The result is a simulation of the whole system as a set of interacting simulators.

A co-simulation scenario is minimally characterized by a set of simulators along with their models and the data dependencies between them. The models and simulators can be in some inaccessible (but executable) format such as a binary file to protect Intellectual Property (IP). As an example, consider the topology of the Anti-lock Braking System (ABS) illustrated in the top of Figure 1 and the corresponding co-simulation scenario immediately bellow. In the example, for each physical component there is a dedicated simulator and a model of that component specified in some language. The arrows in the co-simulation scenario show the data-flow between the simulators (e.g., the speed sensor simulator feeds data to the control unit simulator, which in turn exchanges data with the hydraulic modulator simulator). Notice that the models only exchange data with the corresponding simulators. This is to illustrate the black-box nature of co-simulation.

The co-simulation scenario illustrated in Figure 1 only shows the simulators and their dependencies. To actually execute the scenario much more information is needed: a) names and types of the variables used by the simulators (e.g., the current speed for the speed sensor simulator); b) time constraints of each simulator (real/scaled time or analytic); c) location of the simulators (available at the local system or at a remote location); d) the coupling mechanism used to control the progress of the simulated time and to move and adapt data from one simulator to the next (the simulators do not know each other); etc.

There are endless ways of implementing the concrete co-simulation scenario and each approach requires certain capabilities of the participating simulators. For instance, if the simulators cannot be moved to a single computer to be executed (e.g., they are provided as a web service), then a distributed coupling mechanism

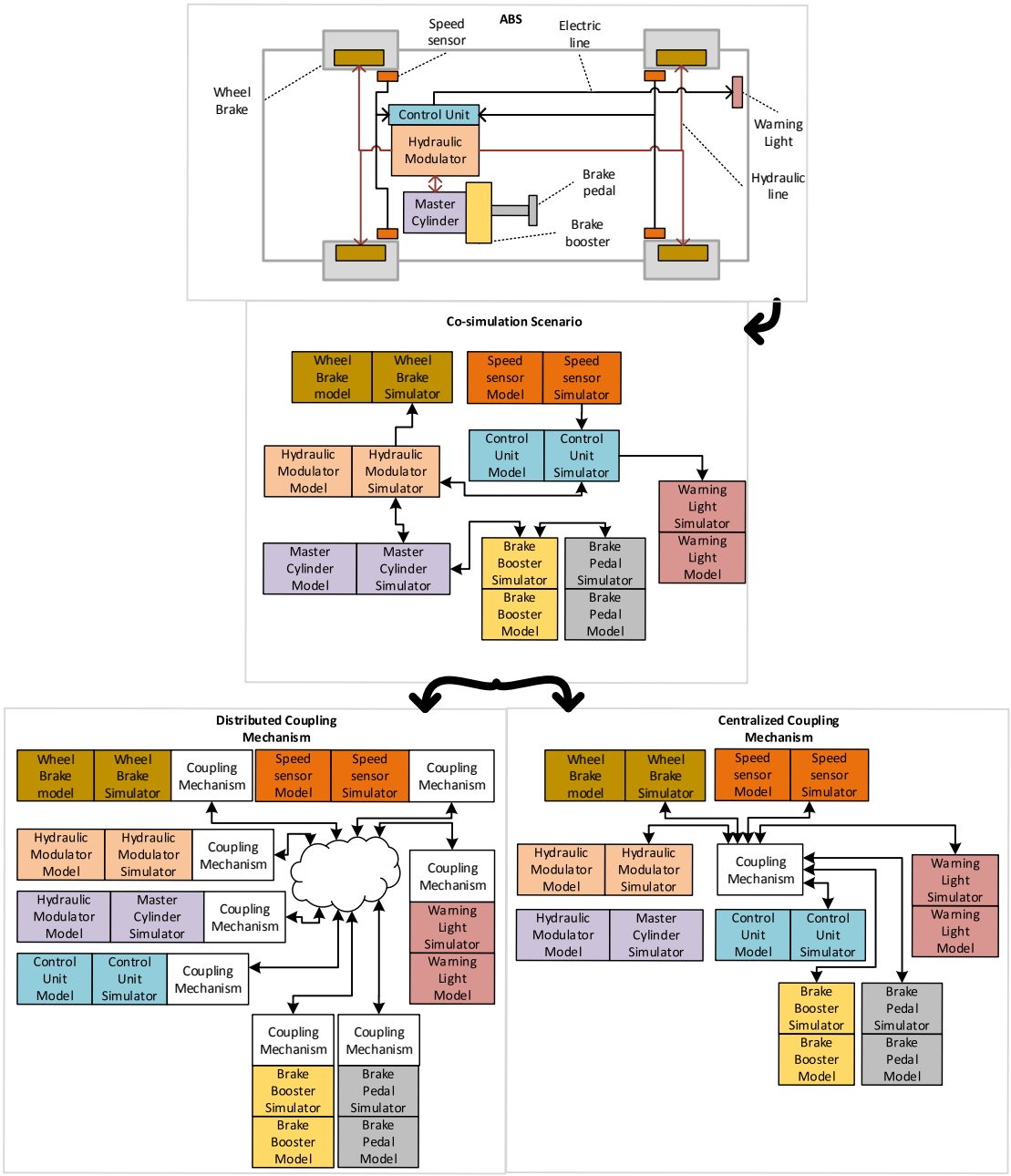

Figure 1: High-level topological view of an Anti-lock Braking System (ABS) at the top, corresponding co-simulation scenario and two possible implementations of the coupling mechanism bellow.

(see left branch of Figure 1) might be the only option. Otherwise, a centralized coupling mechanism can be used (see right branch of Figure 1). Orthogonally, if the simulators do not provide a way to subscribe to events that can occur during the simulation, a client/server architecture such as the one advocated in the Functional Mockup Interface (FMI) standard [12] might be the best solution.

Even with information of the capabilities of each simulator, the co-simulation execution might still yield wrong results: suppose that the values given by the speed sensor are in m/s but the control unit model assumes they are in km/h. Not only does the names and types of variables matter, but also the physical units in which they are expressed. With this information, the coupling mechanism can adapt the values given by the simulated speed sensor to the values expected by the simulated control unit or at least signal the incompatibility between them. This is a simple instance of the semantic adaptation problem [16, 28, 51].

Another problem occurs when there is a circular dependency among the simulators, i.e., at a specific time instant, one simulator's outputs depends on other simulator's outputs, which is turn depends, at the same instant, on the first simulator's outputs. These circular dependencies are called algebraic loops [23] and they emerge when multiple models, and their simulators, are coupled. The nature and amount of information exposed about each model and simulator plays a key role in deterministically and accurately executing a co-simulation scenario that has algebraic loops [19].

An important part of that information is the nature of the dependency between inputs and outputs (I/O) of the models. For instance, if a change in the input of the hydraulic modulator model will immediately (at the same instant) affect its output, then there is a direct feedthrough dependency between its output and its input. If, on the other hand, there is a slight delay – e.g., a natural hydraulic delay – between the change in the input and the reaction of the output, then there is a delayed feedthrough dependency. This example also shows that these dependencies change according to the level of abstraction of the model and hence, regardless of the intellectual property protection of the model, they need to be exposed. The dependency information was recognized as crucial and incorporated as extra data in the version 2.0 of the FMI standard [13], allowing for algebraic loops to be detected – not solved – across multiple coupled simulators.

When algebraic loops are present in a scenario, co-simulation frameworks can reject it or they can try to solve the loops. It makes sense to reject because attempting to solve algebraic loops without extra information leads to non-determinism and stability problems [19]. However, in the simulation of differential equations, it is possible to solve algebraic loops with extra information about the fixed point techniques to be used and how the initial values are computed, unless they are unstable (i.e., no fixed point can be found) [22]. For the co-simulation domain,

there is no study of what information is necessary to ensure that, in the face of algebraic loops across simulators, a solution – a fixed point – can be found.

In summary, co-simulation inherits the challenges of hybrid and distributed simulation [15, 22, 32, 41, 48, 71] and the challenges created by the limited information exposure of the models and simulators to protect the IP of the suppliers.

3.2 Domain Engineering Tasks

This section details the tasks that will help accomplish Goal 1 introduced in Section 2.

Due to the advantages of co-simulation [17, 30, 60], a plethora of co-simulation frameworks are now available. To the best of our knowledge, there are at least 20 tools which claim to perform co-simulation. However, most of these tools, even the ones taking advantage of the FMI standard, are point-to-point solutions coupling two simulators [11]. While these solutions work well for those scenarios, they restrict the available languages for the development of a complex system, such as a modern car, with more than 40 subsystems. Furthermore, the knowledge of how to overcome the challenges of coupling different simulators cannot be easily reused for the support of other scenarios, making this task cumbersome and redundant.

An example of this difficulty is reported in [39], where two new formalisms – 20-Sim [18] and gCSP [29] – were integrated into the co-simulation framework CosiMate [2]. CoSiMate supports a wide range of formalisms – ModelSim [44], C, SystemC [40], Simulink®, StateMate [42] and SABER [25]. The authors had to write two interfaces to allow the co-simulation of two specific models. Furthermore, based on that experience, the authors suggest that a generative approach to generate the interfaces from any models conforming to one of the two formalisms would make co-simulation less cumbersome [17].

I propose to apply domain engineering techniques to capture and reuse the knowledge accumulated in the co-simulation domain by creating a Software Product Line (SPL) [26] for co-simulation scenarios. To realize this, I will provide an instrument to describe co-simulation scenarios. These descriptions, along with other constraints, will contain enough information to automatically synthesize the code that ultimately will execute the co-simulation. The method I will use is the Feature Oriented Domain Analysis (FODA) [26], which can be broken down into domain analysis, design and implementation tasks.

3.2.1 Domain Analysis

Task 1. Identify missing features in the state of the art. For this, I will study the currently supported features in the state of the art, including industrial case

[2] `http://site.cosimate.com/`

studies on co-simulation. The results of this study will be documented using a feature model. A feature model intuitively groups the features required to execute co-simulation scenarios, their relations, and allows for the identification of "gaps" in the groups of features. Figure 2 shows a preliminary feature model of co-simulation scenarios. The features can be grouped into three categories: number of different formalisms, number of simulators being coupled and the capabilities of each of those simulators. The capabilities of each simulator are the information exposed to the outside, rollback support, interaction mechanism, availability, IP protection, time constraints imposed by the simulator, and step related features. The missing features identified so far – Linearized model and Lumped model information exposure – are highlighted. These represent the capability of a simulator to expose a linearized or a lumped version of the model, both useful abstractions to perform analysis and model checking. The capabilities of the simulators, along with the formalisms and number of simulators dictate the kind and complexity of a co-simulation scenario to be executed. The co-simulation scenario illustrated in the top of Figure 1 is characterized by multiple simulators, multiple formalisms (assuming fuzzy rules are being used for the control unit, partial differential equations for the brake pads and differential equations for the remaining components), each simulator exposing only the I/O variable information, with no rollback support, passive interaction, local availability, full IP protection (provided as a binary executable), unconstrained time and supporting fixed step simulation. Most of these features are not explicitly depicted in Figure 1 for simplicity reasons, but they must be part of the co-simulation scenario. The utility of missing features will be measured by pitching them to companies.

Task 2. Study the state of the art and identify the non-functional requirements for the execution of co-simulation scenarios. The non-functional requirements are desired qualities of a co-simulation execution. Through my preliminary studies, I have found that the main non-functional requirements are efficiency, accuracy, parallelism and fault tolerance.

Task 3. Study the interactions among the features and non-functional requirements identified in the previous tasks. These interactions can be depicted directly in the feature model, as prescribed in [49], but for improved readability I show them in Table 1. In the table, the green color (happy face) represents cooperation, the red color (sad face) represents a tradeoff or competition. As an example, if a simulator supports rollback, then its accuracy is higher, hence the accuracy of the overall co-simulation will be higher. This is because rollback capabilities can be used for accurate zero crossing detection (state events) through iterative methods such as Newton's or bisection methods [21]. This preliminary table is symmetric but that is not necessarily the case. By observing how the information

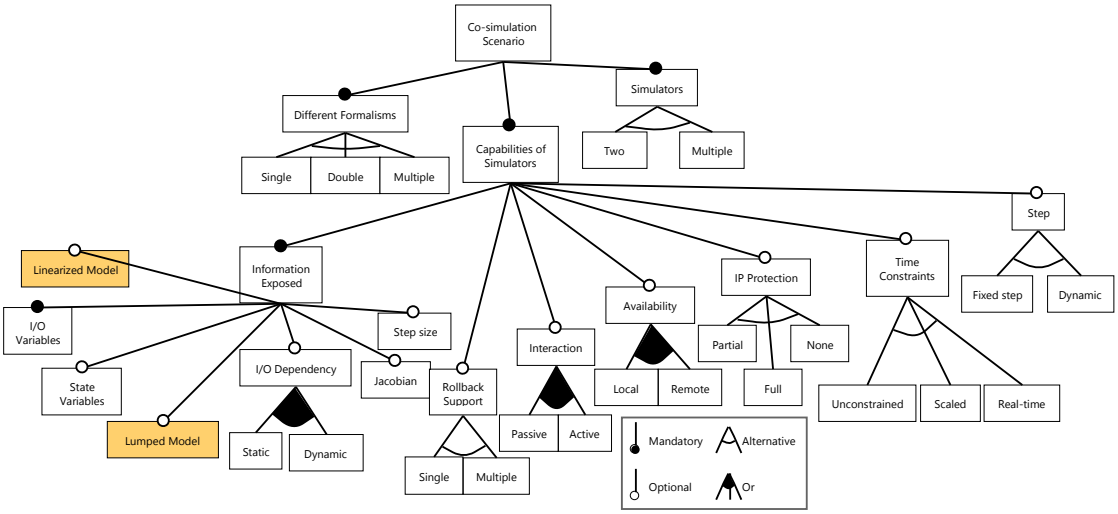

Figure 2: Preliminary feature model of co-simulation scenarios in the state of the art.

exposure affects the other features and non-functional requirements, one can easily see why co-simulation is a difficult challenge: the less information is available, the more difficult it becomes to perform accurate, efficient co-simulation of scenarios containing many simulators in different formalisms. Information is key.

Task 4. Classify the state of the art co-simulation frameworks using the feature model and non-functional requirements found. For instance, a co-simulation framework which relies on the FMI 2.0 standard for the representation of the models and corresponding simulators (e.g., [1]) will not be able to support dynamic I/O dependency information, remote availability, linearized version of the model or the lumped model information exposure. There is a need to enable exposition of this information to increase accuracy and efficiency in some co-simulation scenarios.

Task 5. Adapt the feature model so that it can be used to describe co-simulation scenarios. The feature model developed in Task 1 has limited expressivity because it does not allow for the description of the capabilities of each individual simulator in a co-simulation scenario. A possibility to implement this task is to develop a Domain Specific Language that is derived from the feature model [68]. The concrete syntax of this language might resemble the co-simulation scenario shown in Figure 1 and will be based in the language developed in [54], with the extra information about each simulator in separate textual or graphical forms. Regardless of how the scenario will be described, it must be possible to express the missing feature such as varying levels of information exposure of each simulator.

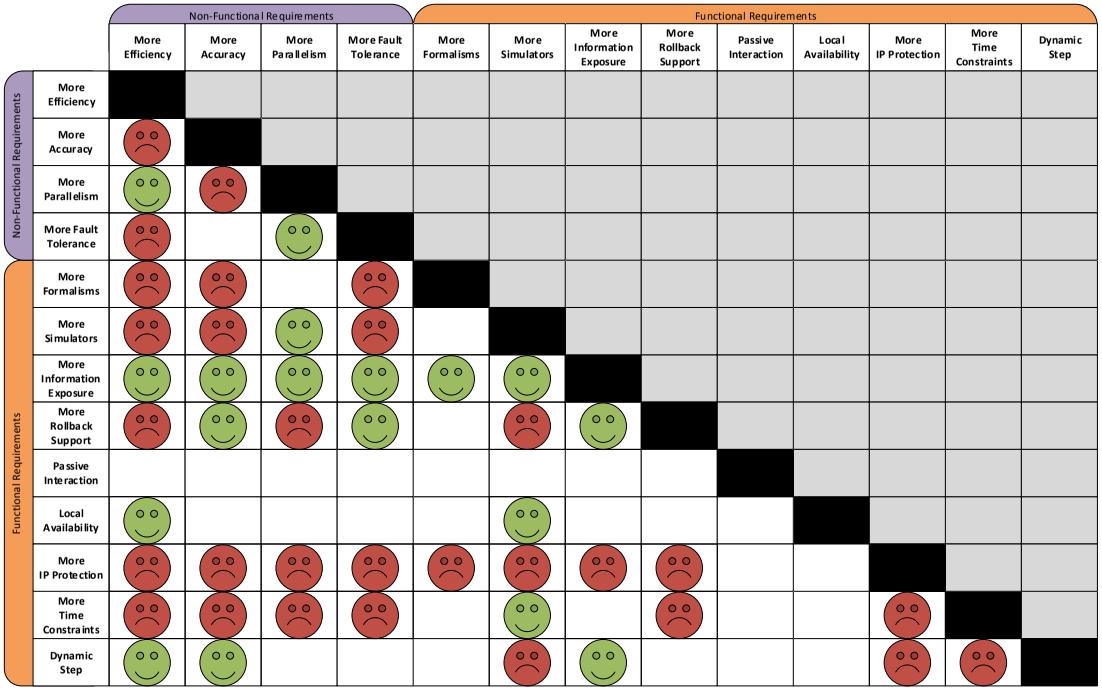

Table 1: Requirements of Co-simulation scenarios and their relationships.

This will enable the validation of scenarios as well the detection of contradicting non-functional requirements based on the experience gained from Task 3. This task will be validated by describing a set of co-simulation scenarios typically used in the industry [33], the power window system [28] and those provided by Siemens PLM Software.

3.2.2 Domain Design

Task 6. Define the generic architecture used to execute all co-simulation scenarios. This work will be based on what was already defined in the FMI 2.0 standard [13] for the representation of models and simulators but will allow for the description of the extra information about the simulators and models. This will leverage all the tools that already support FMI and might provide input for future versions of the standard.

Task 7. Describe explicitly the fundamental techniques to execute the interactions between the simulators, synchronize time, semantic adaptations, etc... To make these descriptions I will need to define a formal language. This language should allow the following items to be described for each specific co-simulation scenario: (i) concrete coupling mechanism to be used (centralized or distributed); (ii) which time advance policy will be used (Gauss–Seidel as is done in [1], or Jacobi type as done in [59, 61]); (iii) which synchronization algorithm will be used (e.g., the classical or canonical algorithm as done in [36] or a more optimistic rollback based one, described in [38]); (iv) and the semantic adaptations needed (zero-order or first-order hold, unit conversion, ...) To deal with the rapidly expanding number of techniques, this language should include extensibility mechanisms such as abstract and modular concepts, as recommended in [58]. The starting point to build this language is to use the concepts introduced in [51] to describe the semantic adaptations and the ones used in [61] to describe the synchronization. The validation of this task will be done by describing the techniques required to execute the scenarios described in Task 5.

3.2.3 Domain Implementation

Task 8. Implement the automatic translation that takes co-simulation scenario descriptions as done in Task 5 to descriptions specified in the language developed in Task 7. The challenge of this task is to make the most of the information available about each simulator to create multiple possible alternatives that can execute it, exploring different techniques. A cost model will be developed in the second part of the project that allows for invalid – or non-optimal – alternatives to be detected, informed decisions to be made and tradeoffs solved. The goal is to end up with an optimal behavioural model described in the language developed

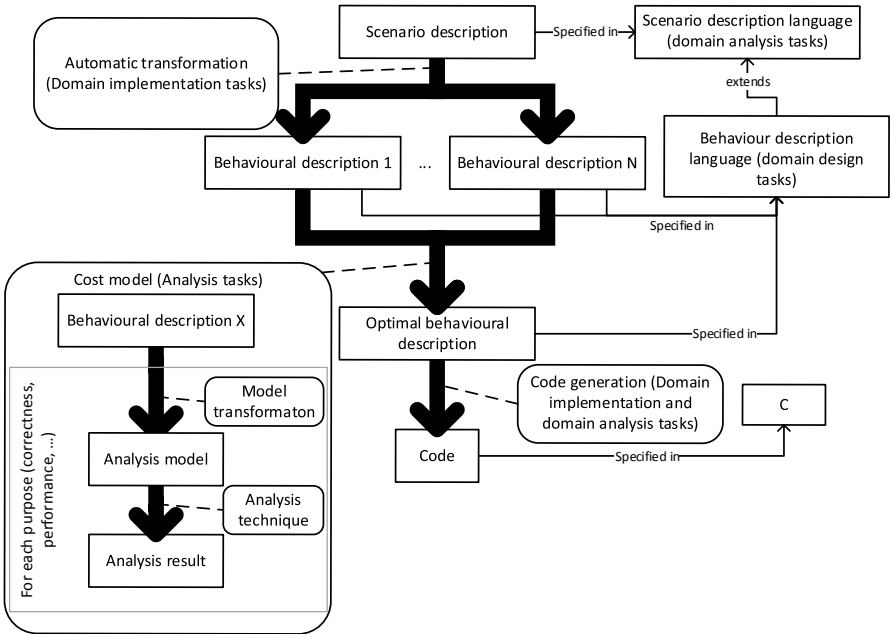

Figure 3: Co-simulation scenario execution process.

in Task 7 with respect to the cost model. As a starting point for this task, I will generalize the descriptions manually done in Task 7 to any co-simulation scenario with the most commonly used simulators and languages. The validation of this transformation will be done by comparing the generated behavioural descriptions against the manually made ones in Task 7.

Task 9. Implement the automatic generation of code from the behavioural refinement described in the language developed in Task 7. The resulting code will execute the co-simulation scenario. The validation of this task will be made by generating the code, executing the scenarios described previously in Task 5 and comparing the execution results with the state of the art. Furthermore, the test scenarios described in [20] provide a good evaluation of the fundamental techniques used.

The result of Tasks 1 to 9 is a tool that allows co-simulation scenarios to be described and executed, by the process illustrated in Figure 3. The second part of this project is concerned with the evaluation of multiple alternative behavioural descriptions and optimizations of its execution.

3.3 Analysis Tasks

This section details the second part of the project: the extension of state of the art analyses and optimization techniques to be applicable to – and take advantage of – any co-simulation scenario. This part is important as it will provide a means of comparing different behavioural descriptions that can be derived from the same co-simulation scenario, as illustrated in Figure 3.

Task 10. Extend the formal verification techniques described in [37] and in [19] to be applicable to any co-simulation behavioural description. This will allow for detecting problems such as deadlocks, non-determinism, causality and other – scenario specific – properties, which I plan to get from industrial use cases. I plan to use tool UPPAAL [8] – based on the timed-automata formalism – to check for time related properties and tool StrataGEM [50] – based on rewriting systems – for other properties. Most of the difficulties in implementing these analyses are either already well known in distributed [31] and/or hybrid simulation [16], or they arise due to the lack of information about the coupled models and simulators. Hence, I will first start by assuming that I have a fixed set of formalisms (State charts [43] and Causal Block Diagrams [57]) and that I have full access to the models. This will enable me to explore the essential information that needs to be exposed in order to check certain properties. Then, I will tackle the general scenario, assuming the essential information is provided. Obfuscation techniques can be used to expose the essential information while protecting the Intellectual Property (IP).

Task 11. Develop new analyses to allow for more efficient execution of the co-simulation scenario. Similarly to the previous task, I will first assume full information exposure and then I will tackle the general case. The purpose of these analyses is to not only give a more realistic cost evaluation of each behavioural description, but also allow for more efficient code generation, making the most of the simulators' capabilities and information available. An example of a good optimization technique is what is done with hierarchical DEVS simulation by establishing point-to-point communication schemes among components that communicate directly [24, 52]. The same can be explored for co-simulation, especially since scenarios of complex hierarchical systems can be described, i.e., the language created in Task 7 is closed under composition. This approach is motivated by noting that information exposure in Table 1 is a key contributor to more efficient coupling mechanisms.

Tasks 10 and 11 complete the process illustrated in Figure 3 by providing means automatically to evaluate and compare different behavioural descriptions for the same co-simulation scenario according to non-functional requirements identified in Task 2.

4 Planning

I am driven by two goals: to define and reuse the existing techniques to execute co-simulation scenarios, and to develop new optimizations and analyses to check for properties in complex co-simulation scenarios. The tasks to achieve these goals will not be executed sequentially, but instead concurrently and iteratively, allowing for maximum flexibility. The scientific contributions are underlined.

Goal	Task	Duration
Define and reuse the fundamental techniques (see Goal 1)	Missing features (Task 1)	3 months
	Non-functional requirements (Task 2)	2 months
	Interactions between features (Task 3)	1 month
	State of the art classification (Task 4)	3 month
	Conference (SpringSim) paper	1 month
	DSL for co-simulation scenarios (Task 5)	3 month
	Runtime architecture (Task 6)	5 months
	Language for behaviour (Task 7)	3 months
	Transformation to behaviour (Task 8)	4 months
	Conference (Modelica) paper	1 month
	Code generation (Task 9)	4 month
	Journal (Simulation) paper	2 months
Extend and develop new analyses (see Goal 2)	Extend analyses (Task 10)	6 months
	Conference (SpringSim or Modelica) paper	1 month
	Optimizations (Task 11)	6 months
	Journal (SoSyM) paper	1 month
PhD Thesis	Write Thesis	2 months
	Total	48 months

Risks Critical tasks include a validation phase with provided industrial cases (Tasks 5, 7, 8 and Task 9). Formal analyses might not scale for complex scenarios. If that is the case, I intend to explore how optimizations from the distributed simulation domain [32] can be applied to co-simulation, as well as develop new ones (Task 11). One of the goals is to contribute to newer versions of the FMI standard. However I am not part of the FMI committee but I will collaborate with Siemens PLM Software to make these contributions part of the standard.

5 Applications

Virtual test environments have impacted every industrial sector. For instance, they have been proven successful for recent highly complex cyber-physical systems like self-driving cars [9, 10]. This project represents an improvement in the ability to create and configure such test environments by allowing multiple independent and readily available simulators to be coupled by non-technical experts. I accomplish this by providing an instrument to describe co-simulation scenarios. The main contributions are:

1. By identifying the essential information required to perform certain safety analyses and allowing suppliers to disclose that information without exposing their Intellectual Property, system integrators will be able to detect problems in their design much earlier in the development process.
2. The analyses developed will not only give insight about the overall system being co-simulated but also help avoid incorrect execution of the simulation.
3. Easy execution of co-simulation scenarios enables the application of agile methods for the development of more sophisticated software in complex systems [30].
4. By explicitly representing the fundamental techniques, this project maximizes knowledge reuse and allows for easy extension as new challenges are overcome. A good example of this fundamental knowledge is the semantic adaptation techniques between simulators of different formalisms (e.g., the communication between a state chart simulator and a Causal Block Diagram simulator) [14, 16, 28].

I justify the application of Model Driven Development (MDD) techniques for the implementation of the co-simulation scenarios by arguing, both from my experience in MDD and model transformations, the extensive experience of the Modelling, Simulation and Design Lab (MSDL), and from the reports of the state of the art [6], that modelling every aspect of the system at an appropriate level of abstraction, using the right formalism, allows for easier understanding of co-simulation in general. This project reinforces the expertise of the MSDL group in modelling and simulation and in promoting Model-based systems engineering (MBSE) as an engineering discipline.

Among the companies which will benefit from the results of this project, I highlight the following: 1. MathWorks already provides support for co-simulation between MATLAB®, Simulink® and HDL models. 2. Siemens PLM Software can make use of this project to support co-simulation of not only white-box models, but also black-box models. 3. Companies such as Punch Powertrain, Luperco, Bombardier, Dana and VERHAERT, that traditionally make use of modelling and simulation tools gain easy access to more modelling languages.

References

[1] Bert Van Acker, Joachim Denil, Paul De Meulenaere, Hans Vangheluwe, Bert Vanacker, and Paul Demeulenaere. Generation of an Optimised Master Algorithm for FMI Co-simulation. In *Proceedings of the Symposium on Theory of Modeling & Simulation-DEVS Integrative*, pages 946–953. Society for Computer Simulation International, 2015.

[2] O Albayrak, H Kurtoglu, and M Biaki. Incomplete Software Requirements and Assumptions Made by Software Engineers. In *Software Engineering Conference, 2009. APSEC '09.*, pages 333–339, Asia-Pacific, 2009.

[3] Ayman A Aly, El-Shafei Zeidan, Ahmed Hamed, and Farhan Salem. An antilock-braking systems (ABS) control: A technical review. *Intelligent Control and Automation*, 2(03):186, 2011.

[4] Muhammad Usman Awais, Peter Palensky, Wolfgang Mueller, Edmund Widl, and Atiyah Elsheikh. Distributed hybrid simulation using the HLA and the Functional Mock-up Interface. In *39th Annual Conference of the IEEE Industrial ElectronicsSociety (IECON 2013)*, 2013.

[5] Jens Bastian, Christoph Clauß, Susann Wolf, and Peter Schneider. Master for Co-Simulation Using FMI. *8th International Modelica Conference 2011*, 2011.

[6] Don Batory, Clay Johnson, Bob MacDonald, and Dale von Heeder. Achieving Extensibility Through Product-lines and Domain-specific Languages: A Case Study. *ACM Trans. Softw. Eng. Methodol.*, 11(2):191–214, April 2002.

[7] Lionel Belmon, Yujung Geng, and Huaqiang He. Virtual Integration for hybrid powertrain development, using FMI and Modelica models. *10th International Modelica Conference*, 2014.

[8] Johan Bengtsson, Kim Larsen, Fredrik Larsson, Paul Pettersson, and Wang Yi. UPPAAL — a tool suite for automatic verification of real-time systems. In Rajeev Alur, ThomasA. Henzinger, and EduardoD. Sontag, editors, *Hybrid Systems III SE - 18*, volume 1066 of *Lecture Notes in Computer Science*, pages 232–243. Springer Berlin Heidelberg, 1996.

[9] Christian Berger. *Automating acceptance tests for sensor-and actuator-based systems on the example of autonomous vehicles*. Citeseer, 2010.

[10] Christian Berger and Bernhard Rumpe. Engineering autonomous driving software. *Experience from the DARPA Urban Challenge*, pages 243–271, 2012.

[11] Christian Bertsch, Elmar Ahle, and Ulrich Schulmeister. The Functional Mockup Interface-seen from an industrial perspective. In *10th International Modelica Conference*, 2014.

[12] T Blochwitz, M Otter, M Arnold, C Bausch, C Clauß, H Elmqvist, A Junghanns, J Mauss, M Monteiro, T Neidhold, D Neumerkel, H Olsson, J V Peetz, and S Wolf. The Functional Mockup Interface for Tool independent Exchange of Simulation Models. *8th International Modelica Conference 2011*, 2011.

[13] Torsten Blochwitz, Martin Otter, Johan Å kesson, Martin Arnold, Christoph Clauss, Hilding Elmqvist, Markus Friedrich, Andreas Junghanns, Jakob Mauss, Dietmar Neumerkel, Hans Olsson, and Antoine Viel. Functional Mockup Interface 2.0: The Standard for Tool independent Exchange of Simulation Models. In *Proceedings of the 9th International MODELICA Conference*, pages 173 – 184, Munich, Germany, 2012. The Modelica Association.

[14] Faouzi Bouchhima, Gabriela Nicolescu, El Mostapha Aboulhamid, and Mohamed Abid. Generic discrete–continuous simulation model for accurate validation in heterogeneous systems design. *Microelectronics Journal*, 38(6–7):805–815, 2007.

[15] F Boulanger and C Hardebolle. Simulation of Multi-Formalism Models with ModHel'X. In *Software Testing, Verification, and Validation, 2008 1st International Conference on*, pages 318–327, 2008.

[16] F Boulanger, C Hardebolle, C Jacquet, and D Marcadet. Semantic Adaptation for Models of Computation. In *Application of Concurrency to System Design (ACSD), 2011 11th International Conference on*, pages 153–162, 2011.

[17] J F Broenink and Yunyun Ni. Model-driven robot-software design using integrated models and co-simulation. In *Embedded Computer Systems (SAMOS), 2012 International Conference on*, pages 339–344, 2012.

[18] Jan F Broenink. Modelling, simulation and analysis with 20-sim. *Journal A Special Issue CACSD*, 38(3):22–25, 1997.

[19] David Broman, Christopher Brooks, Lev Greenberg, Edward A Lee, Michael Masin, Stavros Tripakis, and Michael Wetter. Determinate Composition of FMUs for Co-simulation. In *Proceedings of the Eleventh ACM International Conference on Embedded Software*, EMSOFT '13, pages 2:1—-2:12, Piscataway, NJ, USA, 2013. IEEE Press.

[20] David Broman, Lev Greenberg, Edward A Lee, Michael Masin, Stavros Tripakis, and Michael Wetter. Requirements for Hybrid Cosimulation. Technical report, DTIC Document, 2014.

[21] Richard L. Burden and John Douglas Faires. *Numerical Analysis*. Cengage Learning, 9 edition, 2010.

[22] François E Cellier and Ernesto Kofman. *Continuous System Simulation*. Springer Science & Business Media, 2006.

[23] Francois E Cellier. *Continuous system modeling*. Springer Science & Business Media, 1991.

[24] Bin Chen and Hans Vangheluwe. Symbolic Flattening of DEVS Models. In *Proceedings of the 2010 Summer Computer Simulation Conference*, SCSC '10, pages 209–218, San Diego, CA, USA, 2010. Society for Computer Simulation International.

[25] S Chwirka. Using the powerful SABER simulator for simulation, modeling, and analysis of power systems, circuits, and devices. In *Computers in Power Electronics, 2000. COMPEL 2000. The 7th Workshop on*, pages 172–176, 2000.

[26] Krzysztof Czarnecki and Ulrich W Eisenecker. *Generative Programming: Methods, Tools, and Applications*. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 2000.

[27] Clarence W De Silva. *Mechatronics: an integrated approach*. CRC press, 2004.

[28] Joachim Denil, Bart Meyers, Joachim Denil, Bart Meyers, Paul De Meulenaere, Paul Demeulenaere, and Hans Vangheluwe. Explicit Semantic Adaptation of Hybrid Formalisms for FMI Co-Simulation. In Society for Computer Simulation International, editor, *TMS-DEVS SpringSim*, 2015.

[29] Ian East, Jeremy Martin, Peter Welch, David Duce, and Mark Green. gCSP: a graphical tool for designing CSP systems. *Communicating Process Architectures*, 27:233, 2004.

[30] Ulf Eliasson, Rogardt Heldal, Jonn Lantz, and Christian Berger. Agile Model-Driven Engineering in Mechatronic Systems - An Industrial Case Study. In Juergen Dingel, Wolfram Schulte, Isidro Ramos, Silvia Abrahão, and Emilio Insfran, editors, *Model-Driven Engineering Languages and Systems SE - 27*, volume 8767 of *Lecture Notes in Computer Science*, pages 433–449. Springer International Publishing, 2014.

[31] Richard M Fujimoto. *Parallel and distributed simulation systems*, volume 300. Wiley New York, 2000.

[32] R.M. Fujimoto. *Parallel and distributed simulation systems*. John Wiley & Sons, Inc., New York, USA, 1 edition, 1999.

[33] Virginie Galtier, Gilles Plessis, and Les Renardi. FMI-Based Distributed Multi-Simulation with DACCOSIM. In *Spring Simulation Multi-Conference*, pages 804–811. Society for Computer Simulation International, 2015.

[34] Priya Gandhi, Gail Brager, and Spencer Dutton. A Comparative Study of Mixed Mode Simulation Methods : Approaches in Research and Practice. In *Spring Simulation Multi-Conference*, pages 1239–1246. Society for Computer Simulation International, 2015.

[35] Alfredo Garro and Alberto Falcone. On the integration of HLA and FMI for supporting interoperability and reusability in distributed simulation. In *Spring Simulation Multi-Conference*, pages 774–781. Society for Computer Simulation International, 2015.

[36] Hamid Ghasemi and Zainalabedin Navabi. An Effective VHDL-AMS SimulationAlgorithm with Event Partitioning. *18th International Conference on VLSI Design, 2005.*, pages 762 – 767, January 2005.

[37] L Gheorghe, F Bouchhima, G Nicolescu, and H Boucheneb. A Formalization of Global Simulation Models for Continuous/Discrete Systems. In *Proceedings of the 2007 Summer Computer Simulation Conference*, SCSC '07, pages 559–566, San Diego, CA, USA, 2007. Society for Computer Simulation International.

[38] L Gheorghe, G Nicolescu, and H Boucheneb. Semantics for Rollback-Based Continuous/Discrete Simulation. In *Behavioral Modeling and Simulation Workshop, 2008. BMAS 2008. IEEE International*, pages 106–111, 2008.

[39] M A Groothuis, A S Damstra, and J F Broenink. Virtual prototyping through co-simulation of a Cartesian plotter. In *Emerging Technologies and Factory Automation, 2008. ETFA 2008. IEEE International Conference on*, pages 697–700, 2008.

[40] Thorsten Grötker, Stan Liao, Grant Martin, and Stuart Swan. *System Design with SystemC^{TM}*. Springer Science & Business Media, 2002.

[41] Cécile Hardebolle and Frédéric Boulanger. Exploring Multi-Paradigm Modeling Techniques. *SIMULATION*, 85(11-12):688–708, November 2009.

[42] D Harel, H Lachover, A Naamad, Amir Pnueli, M Politi, R Sherman, A Shtull-Trauring, and M Trakhtenbrot. STATEMATE: a working environment for the development of complex reactive systems, 1990.

[43] David Harel. Statecharts: a visual formalism for complex systems. *Science of Computer Programming*, 8(3):231–274, June 1987.

[44] U Hatnik and S Altmann. Using ModelSim, Matlab/Simulink and NS for Simulation of Distributed Systems. In *Parallel Computing in Electrical Engineering, 2004. PARELEC 2004. International Conference on*, pages 114–119, 2004.

[45] Abir Ben Khaled, Laurent Duval, Mohamed El Mongi Ben Gaïd, and Daniel Simon. Context-based polynomial extrapolation and slackened synchronization for fast multi-core simulation using FMI. In *10th International Modelica Conference 2014*, pages 225–234. Link{ö}ping University Electronic Press, 2014.

[46] E A Lee. Cyber Physical Systems: Design Challenges. In *Object Oriented Real-Time Distributed Computing (ISORC), 2008 11th IEEE International Symposium on*, pages 363–369, 2008.

[47] Edward A Lee. Cyber-physical systems-are computing foundations adequate. In *Position Paper for NSF Workshop On Cyber-Physical Systems: Research Motivation, Techniques and Roadmap*, volume 2. Citeseer, 2006.

[48] EdwardA. a Lee and Haiyang Zheng. Operational semantics of hybrid systems. *Hybrid Systems: Computation and Control*, 3414:25–53, 2005.

[49] Sun Lianshan and Wang Jinyu. Modeling Nonfunctional Requirements in Software Product Line. In Min Zhu, editor, *Business, Economics, Financial Sciences, and Management SE - 100*, volume 143 of *Advances in Intelligent and Soft Computing*, pages 745–753. Springer Berlin Heidelberg, 2012.

[50] Edmundo López Bóbeda, Maximilien Colange, and Didier Buchs. StrataGEM: A Generic Petri Net Verification Framework. In Gianfranco Ciardo and Ekkart Kindler, editors, *Application and Theory of Petri Nets and Concurrency SE - 20*, volume 8489 of *Lecture Notes in Computer Science*, pages 364–373. Springer International Publishing, 2014.

[51] Bart Meyers, Joachim Denil, Frédéric Boulanger, Cécile Hardebolle, Christophe Jacquet, and Hans Vangheluwe. A DSL for Explicit Semantic Adaptation. In Edward Jones Tamás Mészáros Christophe Jacquet Daniel

Balasubramanian, editor, *MPM 2013*, number 1112 in CEUR Workshop Proceedings, pages 47–56, Miami, United States, September 2013.

[52] Alexander Muzy and James J Nutaro. Algorithms for efficient implementations of the DEVS & DSDEVS abstract simulators. In *1st Open International Conference on Modeling & Simulation (OICMS)*, pages 273–279, 2005.

[53] Himanshu Neema, Jesse Gohl, Zsolt Lattmann, Janos Sztipanovits, Gabor Karsai, Sandeep Neema, Ted Bapty, John Batteh, Hubertus Tummescheit, and Chandrasekar Sureshkumar. Model-based integration platform for FMI co-simulation and heterogeneous simulations of cyber-physical systems. In *10th International Modelica Conference*, pages 10–12, 2014.

[54] Gabriela Nicolescu, Sungjoo Yoo, Aimen Bouchhima, and Ahmed Amine Jerraya. Validation in a Component-based Design Flow for Multicore SoCs. In *Proceedings of the 15th International Symposium on System Synthesis*, ISSS '02, pages 162–167, New York, NY, USA, 2002. ACM.

[55] D L Parnas. On the Design and Development of Program Families. In *Software Engineering, IEEE Transactions*, volume SE-2, pages 1–9, 1976.

[56] Kosmas Petridis, Andreas Klein, and Michael Beitelschmidt. Asynchronous method for the coupled simulation of mechatronic systems. *PAMM*, 8(1):10521–10522, December 2008.

[57] Ernesto Posse, Juan De Lara, Hans Vangheluwe, and Others. Processing causal block diagrams with graphgrammars in atom3. In *European Joint Conference on Theory and Practice of Software (ETAPS), Workshop on Applied Graph Transformation (AGT)*, pages 23–34, 2002.

[58] Daniel Ratiu, Markus Voelter, Zaur Molotnikov, and Bernhard Schaetz. Implementing Modular Domain Specific Languages and Analyses. In *Proceedings of the Workshop on Model-Driven Engineering, Verification and Validation*, MoDeVVa '12, pages 35–40, New York, NY, USA, 2012. ACM.

[59] Tom Schierz, Martin Arnold, and Christoph Clauß. Co-simulation with communication step size control in an FMI compatible master algorithm. In *9th Int. Modelica Conf., Munich, Germany*, pages 205–214, 2012.

[60] Stefan-Alexander Schneider, Johannes Frimberger, and Michael Folie. Significant Reduction of Validation Efforts for Dynamic Light Functions with FMI for Multi-Domain Integration and Test Platforms. In *10th International Modelica Conference*, 2014.

[61] S Sicklinger, V Belsky, B Engelmann, H Elmqvist, H Olsson, R Wüchner, and K.-U. Bletzinger. Interface Jacobian-based Co-Simulation. *International Journal for Numerical Methods in Engineering*, 98(6):418–444, May 2014.

[62] Julien Siebert, Laurent Ciarletta, and Vincent Chevrier. Agents and Artefacts for Multiple Models Co-evolution: Building Complex System Simulation As a Set of Interacting Models. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: Volume 1 - Volume 1*, AAMAS '10, pages 509–516, Richland, SC, 2010. International Foundation for Autonomous Agents and Multiagent Systems.

[63] Han-Shue Tan and M Tomizuka. Discrete-time controller design for robust vehicle traction. In *Control Systems Magazine, IEEE*, volume 10, pages 107–113, 1990.

[64] T Tomiyama, V D'Amelio, J Urbanic, and W ElMaraghy. Complexity of Multi-Disciplinary Design. *CIRP Annals - Manufacturing Technology*, 56(1):185–188, 2007.

[65] S Uchitel and D Yankelevich. Enhancing architectural mismatch detection with assumptions. In *Engineering of Computer Based Systems, 2000. (ECBS 2000) Proceedings. Seventh IEEE International Conference and Workshopon the*, pages 138–146, 2000.

[66] Herman Van der Auweraer, Jan Anthonis, Stijn De Bruyne, and Jan Leuridan. Virtual engineering at work: the challenges for designing mechatronic products. *Engineering with Computers*, 29(3):389–408, 2013.

[67] Antoine Viel and L M S Imagine. Implementing stabilized co-simulation of strongly coupled systems using the Functional Mock-up Interface 2.0. *10th International Modelica Conference*, 2014.

[68] M Voelter and E Visser. Product Line Engineering Using Domain-Specific Languages. In *Software Product Line Conference (SPLC), 2011 15th International*, pages 70–79, 2011.

[69] Ming-chin Wu and Ming-chang Shih. Simulated and experimental study of hydraulic anti-lock braking system using sliding-mode PWM control. *Mechatronics*, 13(4):331–351, May 2003.

[70] Faruk Yılmaz, Umut Durak, Koray Taylan, and Halit Oğuztüzün. Adapting Functional Mockup Units for HLA-compliant Distributed Simulation. In *10th International Modelica Conference*, 2014.

[71] Bernard P Zeigler, Herbert Praehofer, and Tag Gon Kim. *Theory of modeling and simulation: integrating discrete event and continuous complex dynamic systems*. Academic press, 2 edition, 2000.