Model Based Development and Simulation of Cross Domain Physical System

Mohd Azizi Abdul Rahman*, Nor Aziyatul Izni Mohd Rosli, Saiful Amri Mazlan, Hairi Zamzuri
Malaysia Japan International Institute of Technology
Universiti Teknologi Malaysia, Malaysia
Email: * azizi.kl@utm.my

Makoto Mizukawa
Department of Electrical Engineering
College of Engineering
Shibaura Institute of Technology, Japan
Email: mizukawa@sic.shibaura-it.ac.jp

Abstract — The development of cross domain physical systems (CDPS) inevitably requires integration of different viewpoints from different domains. The authors introduce a comprehensive design framework, which brings different modeling languages and simulation tools seamlessly integrated for developing CDPS. The framework is built on existing technologies from each of systems, software, electrical, mechanical and control domains, and based on the extendable modeling platform, where its ability to combine different design and simulation tools has proven meaningful. The contribution of the study is the intended framework produces auto-generate Simscape/Simulink behavior models in the Simulink environment supported from the descriptive models in SysML standards. This is achieved by combining the SysML modeling tool with Simscape/Simulink simulation tools using a customized plug-in in order to keep different models in sync. Consequently, the design information can be exchanged amongst development teams for effective decision making and proper documentation without shifting between numerous design tools.

Keywords - Model based development, systems engineering, system level modeling and simulation

I. INTRODUCTION

A. Background

Cross-domain physical systems (CDPS) such as mechatronic systems are typically made up of various subsystems for performing several different functions within the main system. Usually, the subsystems are independently developed by teams of engineers having different technical background, see Fig. 1. Even worse they work on the same system but using different tools. Engineers are only experts on their own domain and could not technically understand the detailed information of the subsystems developed by engineers from other domain. As a consequence, the information used during conceptual systems design cannot be distributed among different development teams for effective design decision-making, communication and documentation.

The development of CDPS requires different individual views or perspectives. Fig. 2 illustrates the current system design scenario with a problem at hand. From control engineering perspectives, the design of dynamic control systems works well using a model-based design (MBD) approach with a widely used executable simulation tool, for example Matlab/Simulink. Simulink uses block diagram whose syntax is familiar to control engineers [1].

The problem is that they often lack understanding of how to write formal specifications for the system they intend to design, which leads to inconsistent system specifications. In contrast, systems engineers are capable of preparing those specifications in a particularly consistent approach from the conceptual design to the detailed design [2].

Fig. 1. System design scenario where control engineers wet the desired system specifications while system engineers write the specs with proper tools

Consequently, high level system designs and specification that usually expressed in SysML are disjoint with executable simulations employed in Matlab/ Simulink. Thus, an integrated design framework for cross-domain knowledge transfer is needed to facilitate analysis...
and synthesis as well as preventing undocumented design changes.

**Fig. 2** Systems design scenario where control engineers set the desired system specifications while systems engineers write the specs with proper tools.

### B. Purpose

In this paper, we introduce an effective method to facilitate cross-domain (e.g. mechanical, electrical, computer science, etc.) physical systems design from the viewpoints of systems and control engineering in general, applied to the context of mechatronic in particular. Designing such systems requires a comprehensive framework which should take appropriate design and simulation (DS) tools integrated. This is one of significant factors to bring good design results and meet the system’s performance evaluation at the conceptual stage of development.

The objective of this study is to introduce a comprehensive modeling framework for CDPS design and simulation. The purpose of the framework is to collect different DS tools with the idea of letting domain-wise engineers collaborate in a single workspace while speeding up the system development process and testing. Accordingly, design and verification works could be accelerated, and various information transfer can be granted. Note that the work presented in this paper is delimited to the context of systems design and simulation, and taking a model-based approach. As systems development tends to be more and more bigger with complex integrations, a model-based development approach is inevitably important to date.

The proposed design framework is influenced by the fundamental concept of OMG’s model driven architecture (MDA). MDA is an approach to using models in software development and its primary goals are software portability, interoperability and reusability through architectural separation of concerns [3]. As an overview, MDA begins with the idea of separating the specification of the operation of a system of the platform independent model (PIM) from the details of the way that system uses the capabilities of its platform on the platform specific model (PSM). In other words, PIM describes the system but does not show details of its use of platform. As we focus on the system design and development as a whole, the idea of model-based systems engineering (MBSE) which involves a paradigm shift from traditional document-centric approaches to using computerized model-based approaches [4] is adopted. As pointed out by Johnson, MBSE can be leveraged to manage systems design complexity via interactive modeling approach [5].

In support of MBSE, a general purpose modeling language for systems engineering activities, the Object Management Group (OMG) SysML [6] has been extended based on its predecessor OMG’s UML [7]. Given this information, the proposed framework is supposed to be a model-centric platform within the SysML-based environment.

### II. RELATED WORK

The overall objective of this study is to integrate seamlessly SysML and Simulink/Simscape in order to support executable simulations for CDPS design and development.

#### A. Issue and Challenge

The idea is to achieve executable simulations in the Matlab’s Simulink extension based on stereotypes and specializations of SysML standards. However, as far as the modeling languages and simulation tools are concerned, the following issues and their challenges need to be highlighted:

1) SysML is a general-purpose modeling language that aims to describe and record systems rather than simulate them. In other words, it’s only capable of descriptive semantics and cannot be executed. Thus, how SysML can support executable simulations is still remaining an open issue. Therefore, it is crucial for SysML’s descriptive semantics to be extended with respect to the executable simulation semantics. This issue emphasizes on the extension of SysML’s standards semantics [8].

2) Simulink, on the other hand, is commonly used for model-based dynamic control systems design and simulation. For system model abstraction, Simulink models are signal-flow oriented (i.e. the effect of causality) and somewhat inappropriate for use in physical systems modeling where the energy-flow elements are considered amongst the subsystems. Apart from that, an algebraic loops issue in Simulink models can cause bad
effects on the simulation potential unless they can be avoided.

B. Practical Approaches

This study examines several related approaches to cope with the stated issues. According to Cao et al., the challenge of extending the SysML’s model semantic can be solved through using the UML stereotypes mechanism [9]. Based on the suggestion, a new set of stereotypes can be created to provide distinctive models in support of executable models. This approach ensures the descripive system model is equally consistent with the simulation model so that executable simulations can be supported from the SysML standards. Model compatibility means the concept of model’s semantic translation is adopted [10].

When considering the design of continuous dynamical systems with physical modeling approach, the algebraic loops issue is strictly emphasized in Simulink causal models. Algebraic loops typically occur when dynamics are unaccounted for, or when connecting component models to each other. In Simulink, algebraic loops can be avoided by inserting additional component (i.e. a tearing concept) as suggested by Elmqvist and Otter in [11]. However, the tearing concept introduces another additional dynamics element, which makes the system even more complex.

With respect to that problem, one solution is to combine the signal-flow model with a physical network approach (i.e. acausal modeling or equation-oriented modeling) using Simulink’s extension product, Simscape. Be noted that physical systems modeling is not trivial because if the physical model is too concrete, then the simulation and design space exploration become expensive and it can even be difficult to solve problems [2]. In that sense, a simplified version by suppressing some selected details on a system is more preferable at the conceptual stage of development; thus a model referencing technique [12] could be adopted accordingly.

Another problem in developing the proposed framework is how to choose the right base tool in order to implement solutions on the earlier challenges. It is time-consuming to develop such a framework from scratch. To that end, UML-based Rational Rhapsody [13] software of IBM is adopted as a core modeling platform. The advantage of using Rhapsody is that both the SysML and Simulink profiles are readily available in the tool directory. These profiles are developed by IBM engineers, and can be used to assist modelers to develop any engineered system. For example, Sakairi et al. [14] used Rhapsody to design a car transmission control system with its simulated output generated automatically in the Simulink environment, while Liu et al. [15] developed a virtual prototype of UAV flight control system with the same tool and profiles.

C. Concrete Objectives

To the extent of the author’s knowledge, not many contributions to the development of integrated modeling frameworks that exploit the potential of UML/SysML languages as modeling environments are presented. Motivated by the works done the previous researchers, the authors aim at attempting to develop a modeling framework for CDPS design and simulation. Therefore, the concrete objectives of this study are set:

1) To present a seamless integrated modeling framework with the intention of filling the gap between high-level system design specifications with low-level executable simulations as depicted in Fig. 2. The advantage is that no shifting between different DS tools will be required to transfer the design knowledge and information amongst cross-domain development teams.

2) To improve the existing approaches (i.e. a synergistic of Simulink and UML [16], and a co-simulation of Simulink and SysML [1]) by introducing an additional SysML-based Simscape profile to create a components library in support of CDPS modeling and simulation.

For this study, practical solutions to form the proposed framework are based on two approaches by:

1) Creating an additional SysML-based model profile to assist CDPS modeling (i.e. a new SysML profile for Simscape that uses the UML stereotypes mechanism [17] and specialization of SysML standard models). To realize that, the authors implement a lightweight model extension as suggested in [18], in order to map between SysML and Simscape semantics.

2) Combining the Simulink/Simscape simulation models with the SysML system models using a simple plugin mechanism. In this work, a Java-based plugin (e.g., a tool adapter) for Rhapsody and simple Matlab APIs are used as part of the combining process. The plug-in interface has two functions;

a) for multiple model synchronization
b) for building a new CDPS component library

The latter function is developed by the authors while the former is already available in Rhapsody developed by IBM software engineers. The newly-added component library is then used for designing simulation models for CDPS in the context of mechatronics.

III. METHOD OVERVIEW

In Fig. 3, the numbers (1-3) represent the steps taken primarily to develop the framework as a whole. Each of the key steps has another subtask(s) to do. In the first step,
a new set of Simscape profile is developed in addition to existing Simulink and SysML profiles developed by IBM researchers [19] and [20]. The Simscape profile is essential to enable CDPS modeling since the inappropriateness of Simulink at that. In the second step, a software-based tool adapter (e.g. a Simscape plug-in for Rhapsody) is designed to enhance the current tool’s functionality in Rhapsody. Finally, in the third step, a model referencing technique (e.g. a black-box modeling approach) is employed to wrap both Simulink and Simscape behavior models, and keep those models in sync with the SysML model for simulation.

A. Step 1: Creating the Simscape profile

This step aims at defining a new profile to assist physical systems modeling based on languages extensions between Simscape and SysML. However, only one-to-one structural language map is introduced rather than a conventional approach. Note that the Simscape language implements a model file structure (.mdl). The main tasks in this step include:

1) Defining languages extensions between Simscape and SysML. For example, in Fig. 3, the Simscape module named *Dynamics* can be semantically extended from the standard SysML block. According to the abstract syntax stated in [6], the semantic extension can be read as `SysML::Blocks::Block` because blocks are the basic model elements in SysML.

2) Implementing a lightweight semantic extension based on the UML stereotypes and specialization of SysML standards. The Simscape profile can be recognized through its package’s extension name (e.g., .sbs) in the tool directory.

3) The Simscape profile, *Simscape.sbs* is temporarily stored in the SysML model package for use in the later stage of development.

B. Step 2: Plug-in Prototyping

The objective of this step is to develop a software plug-in (e.g., a plug-in for Rhapsody) as a function to create the Simscape component(s) library based on the designed Simscape profile. Executing the plug-in will allow the Simscape component library be generated in the Matlab directory automatically. Thus, it can assist modelers to design CDPS using acausal modeling approaches. Apart from that, an existing plug-in (e.g., the Simulink plug-in) is used to facilitate different models synchronization process. Basically, a plug-in for Rhapsody is a user Java application that Rhapsody loads into its process. With plugins, users can extend Rhapsody’s capabilities with user-defined functions by adding user-defined context menus to the Rhapsody’s integrated development environment (IDE). The key points in this step are to:

1) Code the plug-in using Java callback methods, Rhapsody’s APIs and Matlab’s commands.
2) Write a helper file (.hep) with the plugin’s definition. Each plugin needs to establish at least one helper file that will be used to load the plug-in when it is activated.
3) Attach the plug-in along with the Simscape profile. Rename the helper file according to the Simscape profile, and place those under the same package, for instance, the Simscape profile (Simscape.sbs) must be deposited along with the helper file (Simscape.hep). Based on steps 1 and 2, a full feature of models synchronization can be created.
C. Step 3: Implementing model reference method

This step focuses on the implementation of the newly added Simscape profile (i.e., Simscape.sbs) to assist in designing CDPS. Proof-of-concept on combining Simscape, Simulink with SysML is presented in [2] and [21]. A model referencing method is utilized to wrap the Simulink’s signal-flow subsystems and/or Simscape’s energy-flow components along with their interfaces into a black-box so that they are compatible to the SysML design model. The wrapped models should be imported into the main design model. This step is particularly crucial to ensure executable simulations are well-supported in SysML.

IV. DISCUSSION AND CONCLUSION

The work presented throughout the paper provides evidence in the effort to integrate different viewpoints from SysML-based system models with Simulink/Simscape behavior models in the development of CDPS. Some technical challenges on modeling and simulation have been highlighted and the presented approaches in this paper traverse to resolve the challenges with practical solutions. These solutions have proven to be meaningful for achieving the objectives of this paper.

To conclude the paper, Fig. 4 shows a resulting toolchain where the Simscape, Simulink and SysML are seamlessly combined. This platform enables cross-domain engineers to explore wider design space at systems level rather than shifting different simulation models in each subsystem. Thus, integration of general-purpose modeling language (SysML) with simulation languages (Simulink/Simscape) for physical system design is promising.

ACKNOWLEDGMENT

The research work in this paper is supported by the IBM Academic Initiative program, Ministry of Education Malaysia and Universiti Teknologi Malaysia under Research University Grant, Vote No:01K16 and 09H90. The authors also wish to express gratitude to Ishikawa and Sakairi of IBM Research Tokyo for their constructive comments and fruitful discussion.
REFERENCES


