A Model-driven and Tool-integration Framework for Whole Vehicle Co-simulation Environments

Jinzhi Lu, Dejiu Chen, Martin Törngren, Frédéric Loiret

A Model-driven and Tool-integration Framework for Whole Vehicle Co-simulation Environments

Jinzhi Lu, Dejiu Chen, Martin Törngren, Frédéric Loiret
School of Industrial Engineering and Management
KTH Royal Institute of Technology, Stockholm 100 44
jinzhl@kth.se

***Abstract*— Throughout the design of automotive vehicle systems, modeling and simulation technologies have been widely used for supporting their conceptualization and evaluation. Due to the increasing complexity of such systems, the overall quality management and design process optimization are becoming more important. This in particular brings the necessity of integrating various domain-specific physical models that are traditionally based on different formalisms and isolated tools. In this paper, we present the initial concepts towards a model-driven tool-integration framework with automated managed simulation services in the system development. We exploit EAST-ADL and some other existing state-of-the-art modeling technologies as the reference frameworks for a formal system description, with the content including requirements, design solutions, extra-functional constraints, and verification and validation cases. Given such a formal specification, dedicated co-simulation services will be developed to provide the support for automated configuration and execution of simulation tools.**

Keywords—Development, Simulation, System Design, Model-driven, Tool-integration

I. INTRODUCTION

Automotive vehicle systems have evolved continuously over the past decades with increasing functional and technical complexity. The development involves complex interactions of designers and stakeholders, [1]. Typically, an automotive vehicle can be divided into different parts with associated engineering tasks allocated to different work groups or companies. These groups often use specific domain-specific simulator tools to assist the specification, analysis and synthesis tasks.

For a whole vehicle, the overall system behaviors and other properties depend on the characteristics of its subsystems and components being composed. It is therefore important for system developers to conduct early system integration and thereby to understand the system behaviors and analyze system performance in the initial design phases.

In current industrial practices, different CAD and CAE tools have been used for the development of vehicle subsystems and components, such as for requirement analysis, high level design, detailed level design, implementation, testing, etc. Since the subsystem designers often use their own tools to build models, it is a challenge for the system developers to integrate those detailed models and to predict the whole system performance. This calls for a complete integration framework for different models and tools while taking the process management perspective into consideration.

This paper describes an initial version of, and work towards establishing, a model-driven tool-integration design framework for whole vehicle systems in regard to the methodology and technical roadmap. In such a framework, a formal system description is provided to share information among various stakeholders, including project managers, system engineers, modeler, system designers and simulation testers. The content of system information being described and integrated includes requirements, function analysis, system design, parameter settings, interface contract of physical system models, and tool specific information for co-simulation. The framework integrates subsystem models in order to predict the whole vehicle performance. Also, parameters related to optimized simulation behaviors can be added.

The paper has five main sections. Section 2 provides a brief introduction of simulation technologies and system modeling methods. Section 3 presents a problem analysis of an auto-breaking system. Section 4 illustrates a model-driven and tool-integration framework we provided. Section 5 demonstrates a co-simulation test case of a vehicle auto-breaking system. Section 6 and 7 discusses how Open Services for Lifecycle Collaboration (OSLC) can be integrated into our framework and future work.

II. STATE OF THE ART

Nowadays, computation design and simulation technology can help to improve design efficiency and decrease the time consumption and R&D cost [2]. In the vehicle conception design phase, model-based design is very important to predict the system performance in advance. Designers have used many types of modeling technologies and tools to finish their design jobs. For example, CAE and CAD tools, such as Solidworks, Catia, AMESim, Simplorer, Matlab\Simulink, are very widely used in the vehicle industries.

However, there are still various challenges in the whole life cycle of vehicle design. Firstly, different CAD and CAE tools can help designers to build models for designing these subsystems or components to satisfy design requirements, however the whole system performance is very different (and more difficult) to predict, because the subsystem designers are often from other groups or companies and use different tools to build models.

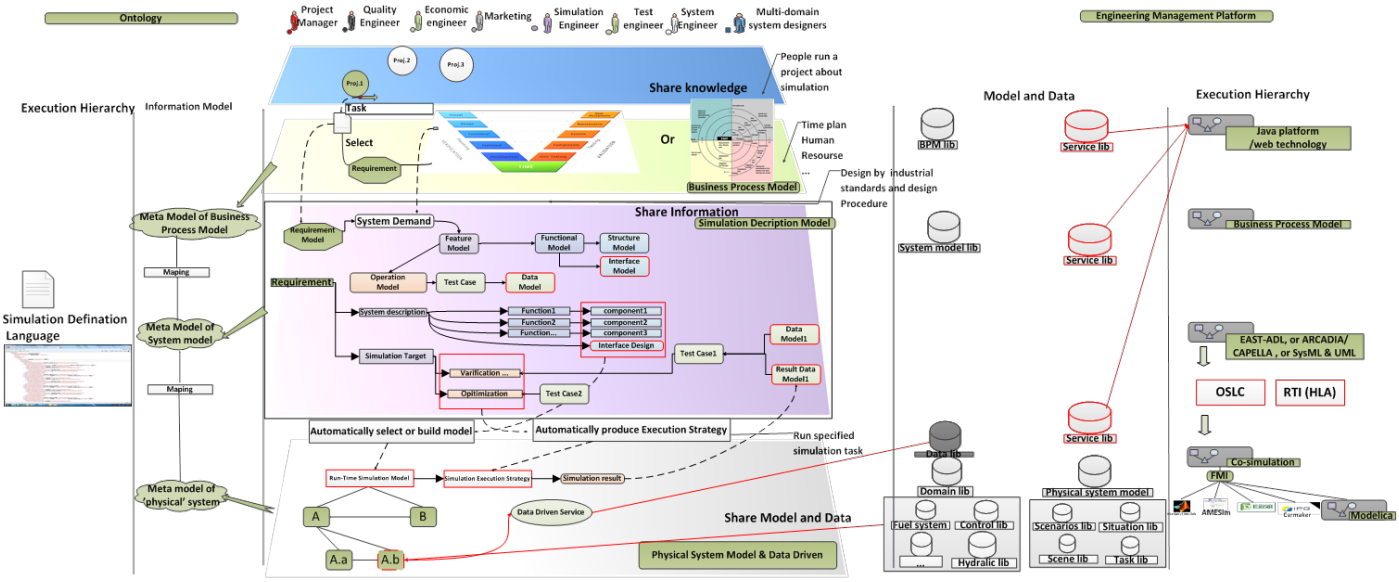

FIGURE 1 MODEL-DRIVEN AND TOOL-INTEGRATION FRAMEWORK FOR WHOLE VEHICLE SIMULATION

Sometimes because of the intellectual property, they even cannot provide the original models. That's why it's hard for integrated system department to integrate the detailed models to predict the whole system performance.

Secondly, even though various PDM and PLM systems can help companies to manage information during vehicle design, such tool-based information management has a restricted scope. For example, if some subsystems are designed by different companies, it is no possible to push all the subsystem suppliers to use the same tool.

Thirdly, nowadays documents still play a very important role in communication and information description, but it's difficult to manage the changes in documents and releases of a huge document may delay the other parts of development. Consequently, a framework with formalized and graphical conception models for model-driven and tool-integration of co-simulation is needed. Several technologies that may be used for such a framework are now briefly surveyed.

A. Multi-domain Simulation Technologies

In addition to commercial simulation platforms, several languages and standards are provided for multi-domain system simulation and analysis.

1) Modelica

Modelica language is a non-proprietary, object-oriented, equation based language to conveniently model complex physical systems [3]. Modelica models are object-oriented and schematics. Nowadays, Modelica language is widely used in automobile industries.

2) Co-simulation

Co-simulation is a technology which can solve the multi-physics model integration. It represents a particular case of simulation scenarios in which there are at least two simulators to solve coupled algebraic equations and exchange the data with each other during simulation [2].The High-Level Architecture (HLA) is a technology for developing distributed simulation developed by the Simulation Interoperability Standards Organization (SISO) [4]. The Functional Mock-up Interface (FMI) standard was initiated through the Modelisar project. FMI is designed for commercial simulators to transform their models to a normative form [5]. The HLA is powerful in mastering the whole co-simulation process and control the data flow. FMI is better for interface design for simulator tools.

B. System Modeling Approach

From the perspective of system engineering, physical system models are not the unique concern. Though a lot of complex and completed model libraries and model management platforms already exist, it's also different for system designers in each layer to understand the whole physical system model and share the model information with each other except for documents or reports. This leads to a need for an information model for information exchange and function descriptions. In this part, several system modeling languages, (e.g., SysML, EAST-ADL, etc.) are investigated.

1) SysML

SysML is a general-purpose modeling language for systems engineering with a subset of UML2 and additional extensions to satisfy the demands of the language description. SysML provides constructs for modeling systems engineering problems including requirements, structure, behavior, allocations, and constraints [6].

2) EAST-ADL

EAST-ADL is a language to describe automotive electrical and electronic systems; it relates closely to SysML and can be seen as a domain specific tailoring for automotive systems. It enables to capture detailed information for documentation, design, analysis and synthesis from the top level characteristics to tasks and interface &communication framework [7].

3) ARCADIA

Arcadia / Capella is a field-proven domain specific modeling solution for system engineering. It has an open source framework to define operation, system, logic and physical system [8].

III. PROBLEM ANALYSIS

We provide a test case to show how co-simulation can be used in autobreaking system design. The verification purpose is to test if the new autobreaking controller can satisfy the '3-second rule', which is a measurement on the time interval for the vehicles to pass the same fixed point on road. If a vehicle reaches such a point within 3 seconds after its foregoing vehicle, then the following distance is too short. For icy or snow-covered road, the corresponding interval can be 7~8 sec.

As Figure 2 shows, we define the Green zones, Red zones and Black zones for the autobreaking controller. X1 and X2 is the position in X direction of vehicle1's and vehicle2's geometric center. If the distance between V1 and V2 is not less than v1/1.2, vehicle2 need accelerate. When it's in red zones, vehicle 2 need decelerate. When the distance between V1 and V2 is less than 1.2m, these two vehicles crashed.

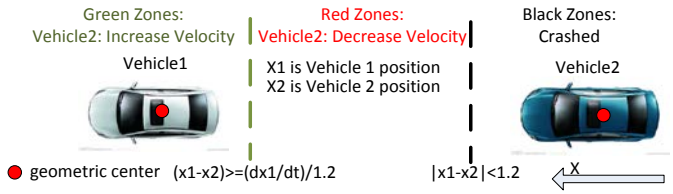

FIGURE 2 CHECKING ZONES FOR AUTOBREAKING STRATEGY

In this case, Carmaker, MWorks and Simulink were used for subsystem modeling and the whole system model is integrated in Simulink which represents the co-simulation process. As shown in Figure 3, the controller model was built by Modelica language in MWorks and was exported to a FMU. Then S-function for co-simulation with FMU and interface library for Carmaker have been used to connect with these two subsystem models with Simulink.

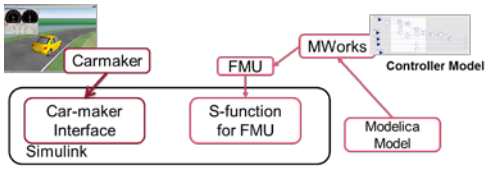

FIGURE 3 TOOL-CHAIN DURING CO-SIMULATION

When the co-simulation is running, Simulink Model, S-function for FMU interface, co-simulation environment setting should be set by hand. And when the version of FMU or Carmaker model is changed, the simulation result should be dependence to the changing. That's really difficult to manage the models and hard to configure the tool setting each time by hand. And simulation technology is used in whole life cycle. In [8], for example, simulation was used in different phases of 'V' model. So simulation information integration with different phase is also a big challenge for current environment without any tool or platform support.

IV. MODEL-DRIVEN AND TOOL-INTEGRATION FRAMEWORK

To solve the former challenge, we design a framework as shown in Figure 1. This framework is inspired by SPIT (Social layer, Process layer, Information layer and Technical layer) [9]. This framework is designed for tool-integration, data-integration, MBSE technical integration and web deployment for the whole life cycle in product design.

A. Social layer

In the social layer, the social models of people's behaviors, organizations, management views, social views and culture for MBSE (Model-based System Engineering) technology are created and describe relative engineering activity in the whole life cycle. From our research, we take an example of MBSE transitioning model as a social model in this layer to describe MBSE maturity level.

This MBSE maturity level model describes enterprises or research teams from the traditional document-based system engineering approach to MBSE approach. In the technical aspect, we separately define transitioning of software and multi-domain system design. As Table 1, we define transitioning level of software and multi-domain system. And we will have an investigation of different effect factors including cultural aspect, stakeholder aspect and transitioning aspect.

TABLE 1 MATURITY LEVEL OF MBSE

Document-based Design	Software	Multi-domain system
Transitioning	Component Design	CAD-based Design
	Model-based Design	Simulation-based design and analysis
		Model-integrated Engineering and analysis
MBSE	Model-driven	Model-based development

B. Business process layer (Process layer)

In this layer, based on different stakeholders' requirements, domain or industrial standards and simulation targets, project managers or team leaders can build different kinds of Business process model (BPM) to describe the process of projects as Figure 6. For the distributed simulation as an example, the IEEE Recommended Practice for Distributed Simulation Engineering and Execution Process (DSEEP) is a standard to define engineering activities for the Process of simulation. We can see, in Figure 8,the BPMN can be used to describe the process model in DESSP.

FIGURE 4 PROCESS LAYER

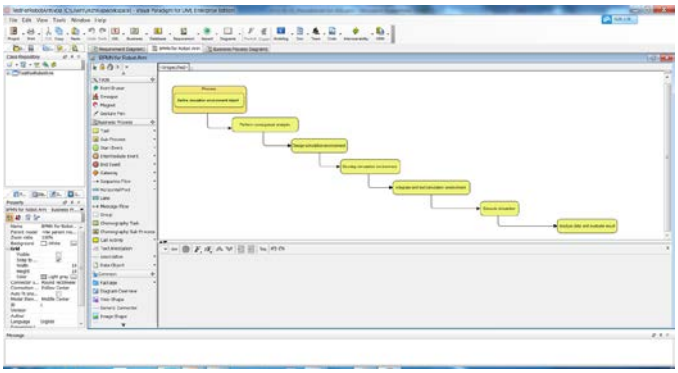

FIGURE 5 BPMN DESCRIBING DSEEP

C. Simulation description model layer (Information layer)

System designers can use system models to describe simulation requirement, physical system feature, model function, model structure, data model, simulation behavior model including verification model, optimization model, and so on as shown in Figure 7.

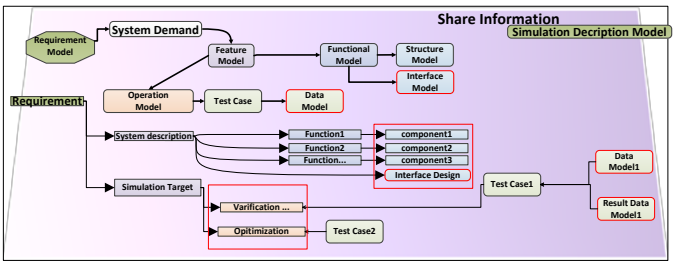

FIGURE 6 SIMULATION DESCRIPTION MODEL

In Figure 7, System model can describe simulation includes requirement diagrams, feature diagrams, functional diagrams, structure diagrams, data diagrams, and behavior diagrams. Requirement diagrams can describe different stakeholders' need including modeling purposes, simulation technologies, constrains of simulation target and etc. Then based on requirement we can select different feature model to describe design targets, simulation targets, optimization targets, system descriptions and simulation settings. System description diagrams can connect to different function model then extend to component diagrams of structure models. Each component block maps to the specified physical system model on the forth layer. Different simulation target and optimization target block can be achieved corresponding verification model and optimization model which are constructed by behaviors model. Each behavior model connects to the corresponding set of services to run the simulation process automatically. Simulation setting models in feature diagram can describe all the information during simulation including simulation type, tool selection, communicational time step point, solver type and so on. Each data model represents sets of parameter for the corresponding physical system model.

D. Physical model and data model layer (Technology layer)

In this paper, each sub-system component model can be uploaded into the library as a FMU or a component of co-simulation model. A master engine should be designed to master the co-simulation procedure between different physical system models. Now the solver of Matlab\Simulink is used to control the co-simulation and S-function is designed as an interface. Each physical system model has its own connections to the component diagrams in the third layer [10].

E. Comparion with other modelling frameworks or projects

1) C2WT Framework

C2WT (Command and Control Wind Tunnel) is a model-based multi-model integration platform which is from a project of American Air Force Research Laboratory. Actually, it's a framework based on Run-Time Infrastructure (RTI) of HLA [11]. In [12], this framework can support RTI to control FMU to run the co-simulation. They have their domain specified language (DSL) to configure and control the simulation. However, the DSL can only describe the model structures and configuration which cannot satisfy the demand for information integration and MBSE approach.

2) DESTECS Project

DESTECS (Design Support and Tooling for Embedded Control Software) is an EU project developed for fault-tolerant embedded systems design. They have a co-simulation engine to control co-simulation process between 20-sim for continuous system model and Overture tools based on Vienna Development Method for discrete-event models. An integrated development environment has been developed to integrate such physical system models [13]. However, this project has limitation for embedded system and is short for information integration during the whole life cycle.

3) WSAF Project

WSAF (Whole-of-System Analytical Framework) is a project running by Australia and New Zealand. In [14], the author introduces a model-based way to share contractual contents between acquirers and suppliers and provides several requirements for such technology. In this paper, only architectural model including requirements and functions has been used for information exchange. The physical system model has not been used for information exchange which means integrated system performance cannot be analyzed in the initial phase of the whole life cycle.

4) ModelicaML

ModelicaML is a UML profile and a language extension for Modelica. It's a good way to connect requirements with physical system simulation by Modelica language to verify if the system can satisfy the requirements which are built by UML[15]. In general, based on several industrial investigations, many industrial firms like to choose the co-simulation way for their model-integration rather than Modelica language. This is because time and financial

consumption for training programs and model-rebuild. So this method has practice restrictions which will influence on its future application.

5) CERTI

CERTI is an HLA RTI developed since 1996 by ONERA, the French Aerospace Lab. It's an open source project for HLA usage. There is no DSL support, but an open source RTI is provided in this project.

6) CRYSTAL

The ARTEMIS Joint Undertaking project CRYSTAL (CRitical sYSTem engineering AcceLeration) takes up the challenge to establish and push forward an Interoperability Specification (IOS) and a Reference Technology Platform (RTP) as a European standard for safety-critical systems[18].

This project has some investigation to show tool-integration's consistency, the methodology of co-simulation was also covered to be tested and applied within the frame.

From such projects, we can find such framework which can cover models and tool-integration is really needed for current industries. As we see, SPIT framework has a more clearly structures for MBSE technologies, data-integration, tool-integration and service-oriented web deployments.

V. User Case For Physical System Model Description

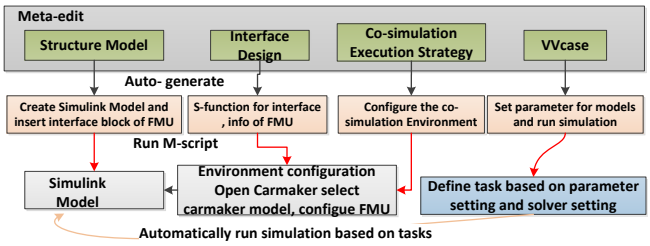

Figure 7 Simulation Description Model and co-simulation execution

We want to use a test case to understand the clear requirements for SPIT framework. So we use MetaEdit to build the SDL and use the code-generator in MetaEdit to produce the M-script in Matlab to run the co-simulation execution automatically.

So we use the test case mentioned in Chapter 3 to build a Simulation Description Model to capture the information for co-simulation process as shown in Figure 9. Function design architecture diagram demostrated the structure of co-simulation model. Interface design diagram showed the interface for the assigned FMU. Co-simulation Strategy diagram was created to configure co-simulation environment and VVcase diagram presented the parameter setting and solver setting for each task in this scenarios. Then Meta-edit uses such diagram model to automatically produce the M-script to control the Co-simulation. Function design architecture diagram produced the M-script to create Simulink models and insert the FMU block in Simulink. Interface design diagram produced the S-function for the assigned FMU and set the FMU block in Simulink. The M-Script which produced by Co-simulation strategy diagram was used to execute co-simulation environment configuration. VVcase diagram produced the M-Script for parameter setting in Simulink Model.

A. *Testcase for SDL in MetaEdit*

We have four models in MetaEdit as Figure 9 shows. In Figure 10, the model describes the co-simulation execution strategy as operation model. The information of FMUs, subsystem models and tools which will be used in this co-simulation will be described and now we can fill such information by hand.

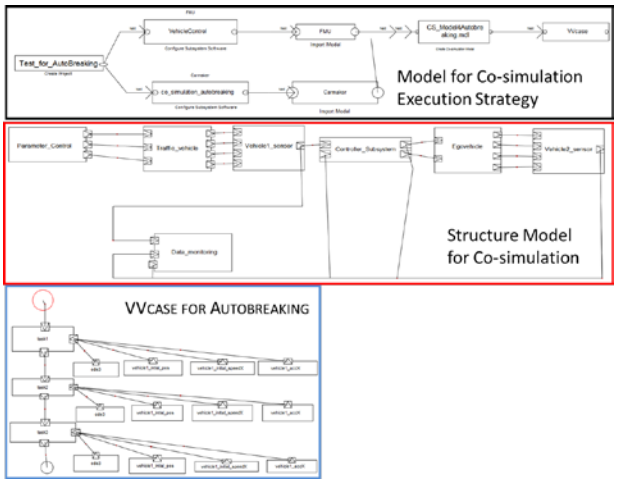

Figure 8 Models in Meta-Edit

As Figure 10 shows, the top structure model in Simulink for this autobreaking user case is built in MetaEdit. Each block describes a subsystem block in Simulink which has been put in the Simulink library.

The VVcase model in Figure 10 describes different solver settings and parameters for the same co-simulation model. In each task, different solver and parameter configurations can be finished by the parameter blocks.

In Figure 10, the interface model is used to describe to S-function for FMUs which will be used for the autobreaking test case. It includes the methods for the S-functions and all the function codes in each method. We need to fill some information for the interface by hand to produce the s-function code automatically.

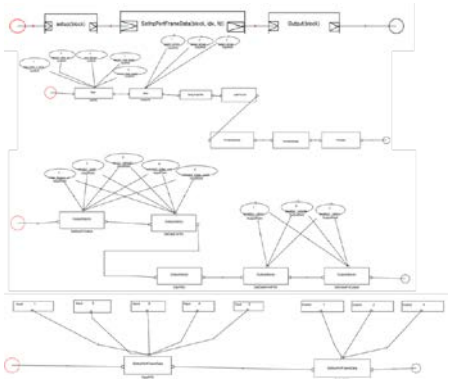

Figure 9 Interface Model for FMUs

B. *Automatic Execution for co-simulation*

We design the Code-generator in the MetaEdit. Based on the models we built in the Meta-Edit and the information we fill by hand, the M-script in Figure 9 will be produced.

C. *Physical system model in Simulink*

In Figure 11, a Simulink model has been automatically produced by M-script and block library which we built.

Then we run the M-script produced by VVcase and co-simulation task is running automatically. During co-simulation process, Carmaker model is running as Figure 12. A front-end window has been designed for verifying the simulation result online and simulation result data has been stored in .mat files as Figure 13.

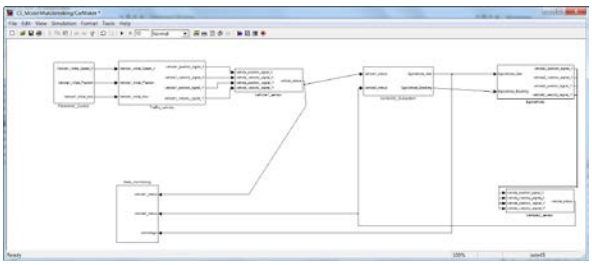

FIGURE 10 SIMULINK MODE FOR AUTOBREAKING TESTCASE

FIGURE 11 CARMAKER MODELS

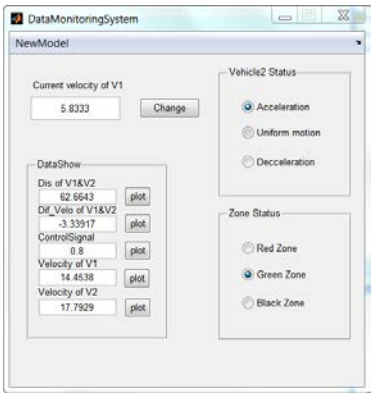

FIGURE 12 FRONT-END FOR VERIFICATION FOR AUTOBREAKING CASE

D. *Result*

As Table 2 shows, we have three different tasks for 3 tests. Each test has different parameters for each models based on various situations. After co-simulation executions, results are shown as Figure 17.

TABLE 2 TEST CASE FOR AUTOBREAKING CASE

Test1: traffic vehicle has a constant speed	Test 2: traffic vehicle has deceleration	Test3: traffic vehicle has a sudden acceleration and then keep constant speed
•EgoVehicle •Initial position:[40 0.0 0.4] •Initial Velocity:20 •Acceleraion:0 •TrafficVehicle •Initial position:[0 0.0 0.4] •Initial Velocity:20 •Initial Acceleraion:0	•EgoVehicle •Initial position:[40 0.0 0.4] •Initial Velocity:20 •Acceleraion:-1 •TrafficVehicle •Initial position:[0 0.0 0.4] •Initial Velocity:20 •Initial Acceleraion:0	•EgoVehicle •Initial position:[40 0.0 0.4] •Initial Velocity:20 •Acceleraion:0(0-30s),1(30-60s),0(60-100s) •TrafficVehicle •Initial position:[0 0.0 0.4] •Initial Velocity:20 •Initial Acceleraion:0

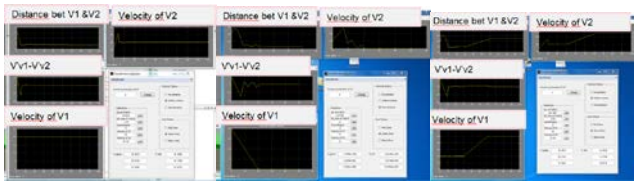

FIGURE 13 SIMULATION RESULT FOR 3 TESTS

VI. DISCUSION

In the whole lifecycle collaboration of the SPIT framework for MBSE, several sections, such as design space exploration, system validation, searching for models, will need model management, information exchange and data management. We take an example of scenarios which OSLC can be used for the whole SPIT framework. As Figure 18 as an example, if we use the web deployments to execute the design process for SOS by co-simulation, the information exchange, co-simulation trigger, and simulation result verification can be achieved by OSLC technology.

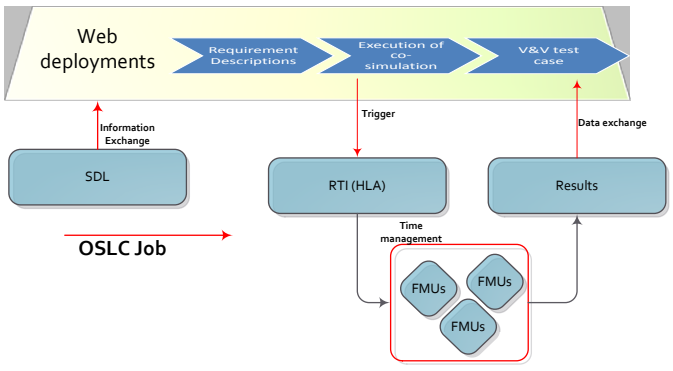

FIGURE 14 SCENARIOS FOR OSLC JOBS

VII. CONCLUSION AND FUTURE WORK

From the test case, we can see Simulation Description Modeling Language can be used to describe the basis

information for co-simulation. The structure model captures the components in top system model which connected with all the subsystem models. Integrated behavior model describes the procedure of co-simulation. VVcase shows a scenario for the autobreaking test case with parameter setting and simulation setting. Interface Design Model demonstrates the approach of interface design. That means graphical model can be used to describe the co-simulation process and control the simulation running automatically. In the future, we will improve our project from four aspects. Firstly, OSLC will be adapted in this framework in order to strength the capabilities for information exchange between different models and layers. Secondly, a special master (RTI) will be designed to control different FMUs and commercial software for co-simulation [11]. Thirdly, a domain specified modeling language will be designed to capture all the information for co-simulation. The last, requirements from business process model should be dependent to ones in simulation description model.

References

[1] Martin Törgren, Dejiu Chen , Diana Malvius , and Jakob Axelsson, Model-Based Development of Automotive Embedded Systems

[2] Jinzhi. Lu "Co-simulation for heterogeneous simulation system and application for aerospace", Master. dissertation, Univ. Huazhong University of Science and Technology, Wuhan, China, 2011.

[3] Modelica® - A Unified Object-Oriented Language for Systems Modeling, https://www.modelica.org/, 2012

[4] HLA Tutorial, http://www.pitch.se/hlatutorial

[5] FMI specification, MODELISAR, https://www.fmi-standard.org

[6] OMG Systems Modeling Language (OMG SysML™), http://www.omg.org.

[7] EAST-ADL-Specification_V2.1.12,EAST-ADL Association, http://www.east-adl.info/

[8] https://www.eclipsecon.org/france2014/session/arcadia-capella-field-proven-modeling-solution-system-and-software-architecture-engineering

[9] Dong Zhang, Jin-zhi Lu , Lin Wang , and Jun Ii, Research of Model-based Aeroengine Control System Design Structure and Workflow, Asia-Pacific International Symposium on Aerospace Technology, 50(4): 511–515, 2014.

[10] Hillary Sillitto. Design Principles for Ultra-Large Scale (ULS) Systems. In INCOSE, International Symposium (INCOSE'2010, Proceedings, pages 63–82, 2010.

[11] Jinzhi Lu, Jian-wan Ding, Fanli-Zhou, Research of Tool-Coupling Based Electro-Hydraulic System Development Method, IEEE International Conference on Industrial Engineering and Information Technology, 2014.

[12] Roth K E, Barrett S K. Command & control wind tunnel integration and overview[C]//Proceedings of the 2009 SISO European Simulation Interoperability Workshop. Society for Modeling & Simulation International, 2009: 45-51.

[13] Neema H, Gohl J, Lattmann Z, et al. Model-based integration platform for FMI co-simulation and heterogeneous simulations of cyber-physical systems[C]//10th International Modelica Conference. 2014: 10-12.

[14] Broenink J F, Kleijn C, Larsen P G, et al. Design support and tooling for dependable embedded control software[C]//Proceedings of the 2nd International Workshop on Software Engineering for Resilient Systems. ACM, 2010: 77-82.

[15] Do Q, Cook S, Lay M. An Investigation of MBSE Practices across the Contractual Boundary[J]. Procedia Computer Science, 2014, 28: 692-701.

[16] Schamai W, Fritzson P, Paredis C J J, et al. ModelicaML value bindings for automated model composition[C]//Proceedings of the 2012 Symposium on Theory of Modeling and Simulation-DEVS Integrative M&S Symposium. Society for Computer Simulation International, 2012: 31.

[17] Noulard E, Rousselot J Y, Siron P. CERTI, an Open Source RTI, why and how[C]//Spring Simulation Interoperability Workshop. 2009: 23-27.

[18] http://www.crystal-artemis.eu/

[19] OSLC specification, http://open-services.net/resources/