An Empirical Evolution of Frameworks Supporting Co-simulation Tool-chain Development

Jinzhi Lu[1], Didem Gürdür[1], De-Jiu Chen[1], Jian Wang[2], and Martin Törngren[1]

[1] KTH Royal Institute of Technology,
Brinellvgen 83, SE- 100 44 Stockholm, Sweden
{jinzhl,dgurdur,martint}@kth.se
chen@md.kth.se
[2] University of Electronic Science and Technology of China,
Xiyuan Ave, West Hi-Tech Zone,611731, Chengdu, China.
wangjian3630@uestc.edu.cn

Abstract. Co-simulation has been found as an effective method for facilitating integrated simulation of multi-domain models of Cyber-physical Systems (CPS). To ensure that co-simulations are well-managed, more concerns about related tool-chains need to be addressed before development. In this paper, we first introduce an evolution process of two frameworks supporting co-simulation tool-chain development. In detail, a '***SPIT***' framework is proposed to support tool-chain development based on model-driven techniques. Then, in order to implement co-simulation more effectively and flexibly, we develop a service-oriented framework, '***SPIRIT***' framework based on model-driven and tool-integration techniques. Second, we propose a 3D viewpoint based method to formalize concept models of co-simulation tool-chain. Finally, in order to evaluate the evolution process, we compare tool-chains developed based on these two frameworks by using visualizations of related concept models.

Keywords: Model-driven, Tool-integration, Co-simulation, Process Management, Framework Design

1 Introduction

Co-simulation is a proposed solution to deal with incorporating from multi-domain models of Cyber-Physical Systems (CPS), such as automotive and aerospace equipments [3]. As CPS complexity is increasing, management and integration of development processes and information related to co-simulation are challenged by growing concerns of tool-chain developers, in particular during CPS co-design. For example, traceability and dependency of related information and technical resources, e.g. models, data and tools, are important concerns to the information management related to co-simulation [15]. Moreover, although a set of tools, techniques, methods and methodologies have been developed to support co-simulation, it's impossible to realize entire simulation automation in a unified and seamless platform, specifically integration and configuration between isolated tools [7].

The motivation of our work is to overcome the challenges identified in the above paragraph and provide a framework to support co-simulation tool-chain[3] development as follows. First, CPS development can be supported in a systematic way. Integrated information infrastructure should facilitate formalism and description of CPS development by a model-based way in order to promote dependency and traceability of related information during co-simulation. Second, flexible process management supporting automated co-simulation can promote traceability among development processes, information artifacts and the related technical resources (data, model and tool operations). Third, tool-integration based on open standards can improve the interoperability of co-simulation tool-chains. During tool-integration, data, models and APIs in each tool are expressed into formal descriptions which can be accessed by other tools. In closing, a service-oriented approach enables the coordination and integration of technical resources at a high-level of abstraction to support co-simulation, e.g. particular services applied to deploy technical resources and realize change and configuration management.

In this paper, we first introduce an evolution process of frameworks supporting co-simulation tool-chain development. Based on an initial framework called '***SPIT***' framework capturing concerns from ***S***ocial, ***P***rocess, ***I***nformation and ***T***echnical aspects [18], *d*omain *s*pecific *m*odeling (DSM) and model-driven techniques are used to realize management and implementation of co-simulations. After summarizing empirical findings from the

[3] In [15], a co-simulation tool-chain is defined as a toolset supporting implementation and management of co-simulations.

tool-chains developed based on '***SPIT***' framework, we proposed a new framework called '***SPIRIT***', replacing ***I***nformation layer as ***I***nformation and se***R***vice ***I***nfrastructure layer. A service-oriented approach is adopted for standard based tool-integration, process management, linked data of technical resources and simulation automation. Then a concept model is proposed to formalize co-simulation tool-chains based on a 3D viewpoint-based method. Finally, the evolution process of tool-chains developed based on these two frameworks is evaluated by related visualizations of such concept models.

Our contribution is to propose a concept model to formalize co-simulation tool-chains. This concept model is developed based on a 3D viewpoint-based method which viewpoints of process, system and cognition are considered to frame related concerns of tool-chain developers. These viewpoints govern views of a model flow, referring to a specific architectural description of tool-chains. Each view includes a set of models integrated with stakeholders, model connections and tool adapters to construct a model flow representing how stakeholders implement models during co-simulations. Based on such model flows, tools implementing models and tool adapters, tool channels and stakeholders can construct tool-chain concepts. Finally, developers can evaluate tool-chain effectiveness by comparing tool-chain concept models through visualizations.

The rest of the paper is organized as follows. We discuss the related work in Section 2 and introduce the evolution process of frameworks in Section 3. Then, we propose a 3D viewpoint-based method to define concept models of co-simulation tool-chain in Section 4. In Section 5, two co-simulation tool-chains developed based on '***SPIT***' and '***SPIRIT***' frameworks are evaluated based on the related visualizations. Finally, we conclude the study with a summary in Section 6.

2 Related Work

Several existing studies proposed concept models to capture co-simulation features, such as [23]. The author provided a methodology for formalizing the co-simulation networks. Specifically, we contribute to formalize co-simulation tool-chain based on solutions integrated with model-driven techniques and to find a better means for developing tool-chains. Tool-integration techniques which may support co-simulation are also included to explore how an effective co-simulation tool-chain can be constructed based on related concepts.

2.1 Co-simulation approaches based on model-driven techniques

Currently, there are two main purposes of model-driven techniques to support co-simulation implementations. First, they aim to generate components of co-simulation model to realize model integration. For example, SysML and Simulink models were generated into formal codes used for co-simulation implementations [24]. Other model-driven techniques transformed standard based system models to a formal description in order to implement co-simulations, such as FMI [21]. The second goal is to realize automated testing using co-simulation, e.g. [28]. Several researchers adopted model-driven techniques for both purposes, e.g. INTO-CPS project [17].

These approaches can promote the efficiency and effectiveness by automating tests and generating components of co-simulation models. However, tool-integration and process management are not covered. These approaches limit to the capability of process management and further co-simulation automation.

2.2 Tool-integration

Literature reviews on tool-integration are provided by Wicks [27] and Brown [6]. Matthias proposed the dimensions about tool integration [4]. Data-integration refers to data sharing and the relevant relationship management between data objects of different tools. Control-integration provides tool operations to notify and activate other tools. Presentation integration provides a unified and formal representation of user interface. Process integration refers to integration of process management and development data. Platform integration provides a virtual operation environment for integrating tools.

Currently, frameworks and platforms have provided tool-integration solutions for co-simulation environment. In ([25] [16], [11]), the researchers developed platforms to support tool-integration during co-simulations. The Open Service for Lifecycle Collaboration (OSLC) aims to standardize tool integration solutions [10]. It is an industrial effort to support requirement and change management. In ([20], [1], [14]), they proposed solutions to enable tool-integration based on OSLC in order to support co-simulations.

Our proposed framework is somewhat different from all of the above. First, the proposed framework is mainly focus on concepts of co-simulation tool-chain rather than a real tool-chain. In addition, through an empirical evolution, a new framework concept makes tool-chain development more effective and comprehensive. Moreover, the new framework proposes a vision of service-oriented co-simulation tool-chains to support process management, simulation automation, traceability management and tool integration.

3 A Framework Evolution to Support Co-simulation Tool-chain Development

In this section, we first introduce an overview of a framework evolution process and then two frameworks.

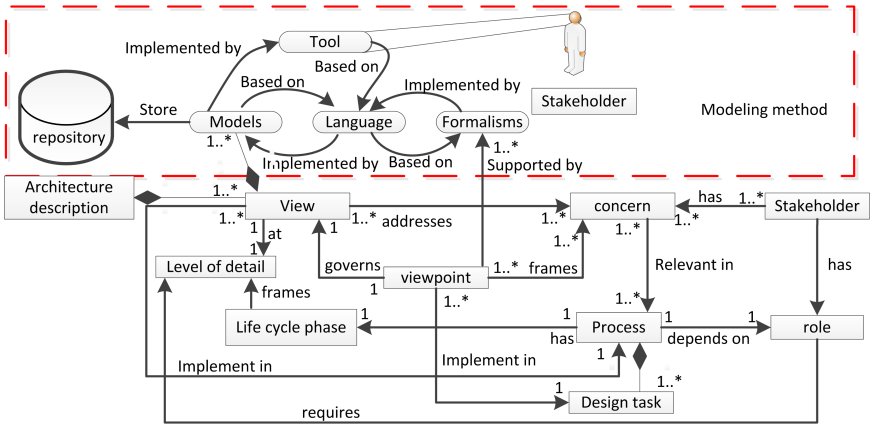

Fig. 1. Basic principles of coupling process, system and modeling method

3.1 Overview of the Evolution Process

We adopt a systems engineering standard [13] to formalize architectures of co-simulation tool-chains aiming to integrate development processes, related product systems and modeling methods, especially co-simulation.

As shown in Fig. 1, based on [26] and [5], basic principles of coupling process, system and modeling method are proposed. The basic principles provide clues to integrations of co-simulation, development and CPS products. An architecture description of CPS includes views to address stakeholders' concerns. Each stakeholder has a role requiring detail level of views framed by life cycle phases. Processes related to concerns include design tasks which viewpoints can be implemented in. Viewpoints frame concerns and govern views. Each view includes various models. Model stored in repository can be implemented by tools based on language. Such languages are implemented based on formalism determined by viewpoints.

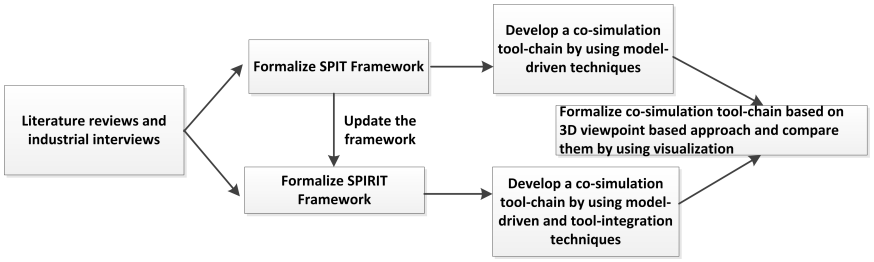

Fig. 2. Evolution of framework concepts

In order to realize the coupling principles, two frameworks are developed in order to support co-simulation tool-chain development as shown in Fig. 2. We formalize a ***SPIT***

framework and develop a co-simulation tool-chain based on model-driven techniques. After summarizing empirical findings from such tool-chains, we propose a new framework concept, ***SPIRIT*** framework. Based on this framework, a new tool-chain is developed in order to realize tool-integration and further co-simulation automation. In order to assess the evolution, we propose a concept model based on a 3D viewpoint based approach to formalize tool-chains and compare them by using visualizations.

3.2 Frameworks in the Empirical Evolution

In this section, we introduce the evolution process in detail: ***SPIT*** framework, the empirical findings for framework update and ***SPIRIT*** framework.

SPIT Framework: The ***SPIT*** framework covers social, process, information and technical layers to support developers to capture concerns of co-simulation tool-chains. In Fig. 3, a tool-chain workflow based on ***SPIT*** framework is described. System developers make use of DSM tools to build DSM models to formalize, describe, design and manage the integration of processes and related information for co-simulation. By model-driven technique, technical resources (data, codes, model and tools) can be implemented and managed automatically.

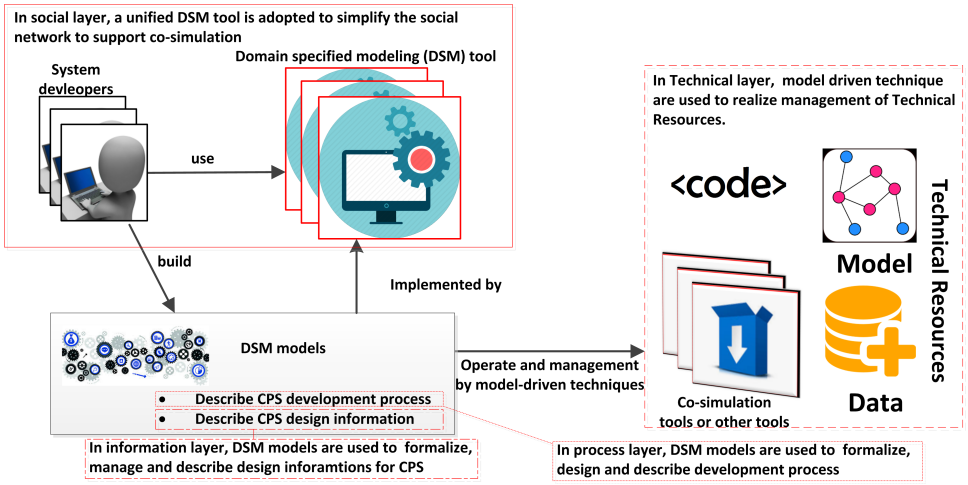

Fig. 3. Co-simulation tool-chain Workflow based on SPIT framework

Empirical findings of *SPIT* framework: Though the tool-chains developed based on ***SPIT*** can support co-simulations and satisfy the specific purposes, there are still several limitations:

1. ***Process control and management***: Though system developers can use DSM models to formalize, design and describe their co-simulation processes, such processes are difficult to be controlled and managed as their complexity is increasing. For example, in some co-design scenarios, various stakeholders participate in modeling, such tool-chains are limited to manage the technical resources during co-simulations across stakeholders.
2. ***Tool-integration***: Tool-chains based on ***SPIT*** can realize control-integration and data integration in partial scenarios. For example, DSM model can generate Matlab language to control simulations in Simulink [14]. But no open standard support makes this type of end-to-end integration inefficiently. Integrations across design tools are difficult to realize.
3. ***Management of the "lities"***:With the help of DSM tools, "lities", e.g. dependency and traceability of elements in DSM models can be managed. However, such "lities" are difficult to be managed among technical resources. For example, relationships between DSM models and technical resources can be traced, however, traceability among technical resources is difficult to manage.

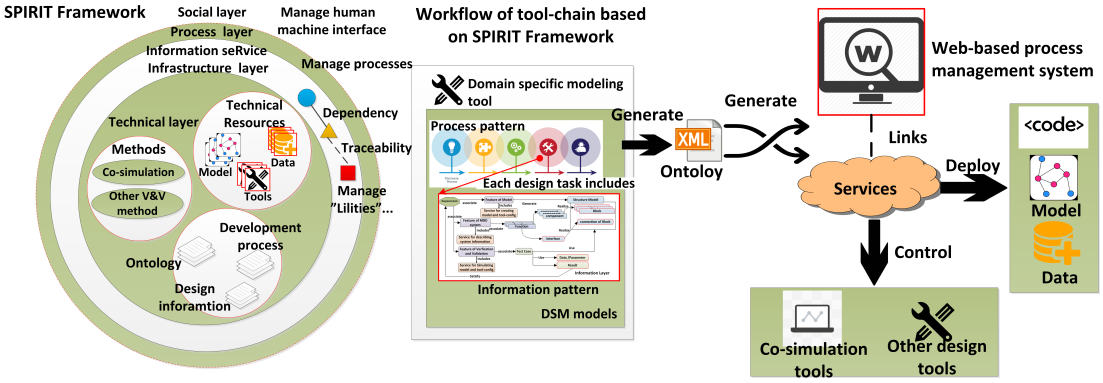

Fig. 4. SPIRIT framework and Co-simulation Tool-chain Workflow based on SPIRIT framework

SPIRIT Framework Based on the previous empirical findings, we propose a ***SPIRIT*** framework to support co-simulations by using a service-oriented approach in order to enhance process management, tool-integration and "lities".

In Fig. 4, a concept of the ***SPIRIT*** framework is introduced. In the updated framework, we make use of **I**nformation se**R**vice **I**nfrastructure layer instead of information layer. This layer is mainly used to manage "*lilities*" among technical resources by a service-oriented infrastructure. In the technical layer, three main aspects are concerned. *Methods* refer to the used technical solutions or steps taken in setting about a task or a problem. *Ontology* refers a formal way to describe related development processes and information. *Technical resources* refers to data, models and tools used to support co-simulations. In process layer, a IT system is used for managing co-simulation processes in order to implement social networks of stakeholders.

The workflow of tool-chains developed based on ***SPIRIT*** framework is introduced in Fig. 4. In domain specific modeling tools, stakeholders build DSM models including process pattern and information pattern. The process pattern is used to formalize development processes of co-simulation. In the process pattern, each work task includes an information pattern to formalize related design information. Then by code generator of DSM tools, ontology is generated from DSM models. Afterwards, ontology is used to generate a web-based process management system linked with services. Through these services, stakeholders can implement tool operations to control tools and deploy data, models and codes by using the process management system.

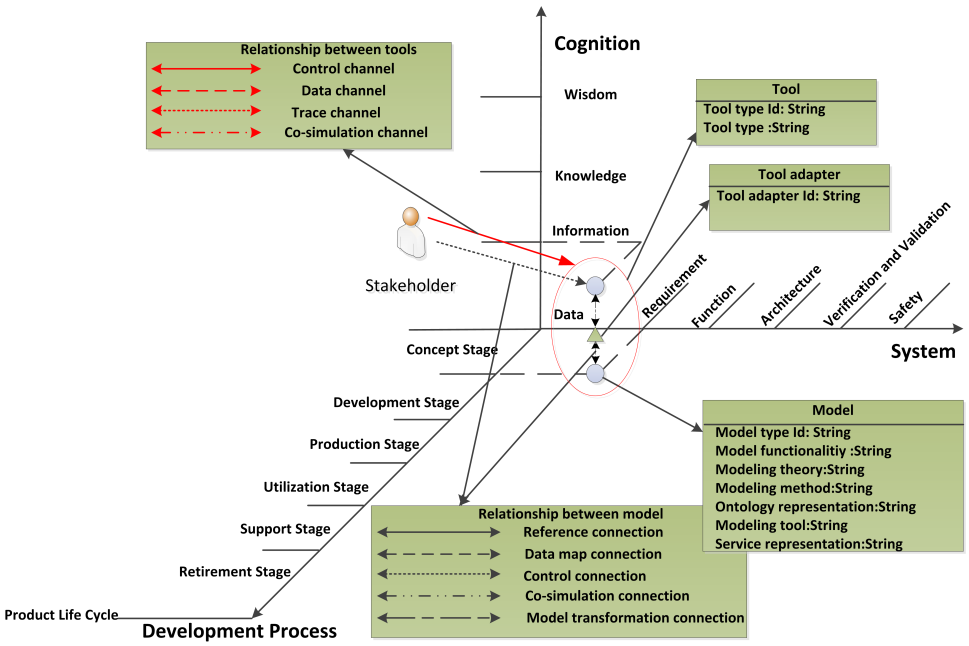

Fig. 5. 3D method to formalize co-simulation tool-chain

4 A 3D Viewpoint-based Method for Formalizing Co-simulation Tool-chain

In order to assess tool-chains, we proposed a three-dimension viewpoint-based method to formalize concept models and to address functional requirements of tool-chains. Based on integration of Hall's model [12] and DIKW model [22], the 3D viewpoint method is shown in Fig. 5. In this method, development processes, CPS products and cognition levels of technical resources are used to formalize concerns about tool-chain development shown as follows:

1. **Process viewpoint**, P refers to development life cycles of CPS including various stages and phases.
2. **System viewpoint**, S refers to system viewpoints, e.g. requirement, function and architecture.
3. **Cognition viewpoint**, C refers to a cognition level of technical resources including data, information, knowledge and wisdom.
   - ***Technical resources in Data layer*** The original technical resources used in co-simulation.
   - ***Technical resources in Information layer*** Technical resources used to describe 'information' of co-simulation.
   - ***Technical resources in Knowledge layer***: Technical resources considered as 'right' which has been verified or validated.
   - ***Technical resources in Wisdon layer***: Technical resources supporting decision-making during co-simulation.

Each point in the coordination of 3D viewpoint-based method refers to a view addressing developers' concerns, $V(P,S,C)$, including a set of models, $M_{V(P,S,C)}$.

$$M_{V(P,S,C)} = \xi(M_0(P,S,C), ..., M_i(P,S,C), ..., M_z(P,S,C)) \tag{1}$$

where M_i refers to one model, i is the i-th model; z is total number of models related to the view *V(P, S, C)*, the $\xi()$ refers to a function collecting a model set including $M_0(P,S,C), ..., M_i(P,S,C), ..., M_z(P,S,C)$ at *V(P,S,C)*. Each model, M_i includes properties:

- ***Id***:Identification of used model.
- ***Model functionality***: the purpose of modeling.
- ***Modeling theory***: the required modeling theory.
- ***Modeling method***: the required modeling method.
- ***Modeling tool***: the tools implementing models.
- ***Ontology representation***: formal descriptions of meta models in the DSM models. For example, concrete syntax in XML represents ontology, "$< Model >< /Model >$" representing a model structure in DSM models.
- ***Service representation***: service descriptions of meta models in the DSM models, e.g. a service representing a model structure in DSM models.

There are five types of model connection that exist among different models implemented by related tool adapters, *MR* [4]:

- ***Reference connection***: track/compare/synchronize versions of models.
- ***Data map connection***: track/compare/synchronize attributions of models.
- ***Function Wrap connection***: track/control models.
- ***Co-simulation connection***: execute co-simulations between models.
- ***Model transformation***: track/compare/synchronize model information.

We define a modeling flow referring to a workflow how models are implemented to support co-simulations. Through tool adapters, models can be connected to each others. Therefore, the modeling flow is defined, *modelFlow* as follow:

$$modelFlow = \xi(M_0, ..., M_i, ..., M_z, MC_0, ..., MC_\alpha, ..., MC_\zeta, ST_0, ..., ST_a, ..., ST_b, TA_0, ..., TA_A, ..., TA_B) \tag{2}$$

where $\xi()$ refers to a collection of models, stakeholders, tool adapters and their relationships; ST refers to stakeholders; TA refers to tool adapters; i, α, a and A refer to i-th model, α-th

[4] In [2], the author proposed a method formalizing model relationships by four types, we added another type: co-simulation connection to represent co-simulations between models.

model relationships, a-th stakeholder and A-th tool adapter, separately; z, ζ, b and B are the total number of models, model relationships, stakeholders and tool adapters, separately.

We define a *tool* implementing models and model connections, T as follows:

$$T = \xi(M_0, ...M_i, ..., M_z, MR_0, ..., MR_\alpha, ..., MR_\zeta, TA_0, ..., TA_A, ..., TA_B) \tag{3}$$

where $\xi()$ is a collection of models , tool adapters and their connections implemented in the tool; i, α and A refer to i-th model, α-th model relationships and A-th tool adapter; z, ζ and B refer to the total number of models, model connections and tool adapters in a tool [5].

Based on [15], four types of Tool Channel (short for TC) are used to represent connections between tools:① **Control channel**; ② **Data channel**; ③ **Trace channel**; ④ **Co-simulation channel**. Therefore, a whole tool-chain, *Toolchain* is formalized as follows:

$$Toolchain = \xi(T_0, ..., T_i, ..., T_z, TC_0, ..., TC_\alpha, ..., TC_\zeta, ST_0, ..., ST_a, ..., ST_b) \tag{4}$$

where $\xi()$ is a collection of tools, stakeholders and their relationships; ST refers to stakeholders; i, α and a refer to i-th tool, α-th tool channels and a-th stakeholder, separately; z, ζ and b are the total number of tools, tool channels and stakeholders.

5 Case Study and Evaluation

We make use of visualizations to compare the effectiveness of two tool-chains developed based on SPIT framework and SPIRIT framework. These two tool-chains are introduced in [14] and [19] [6].

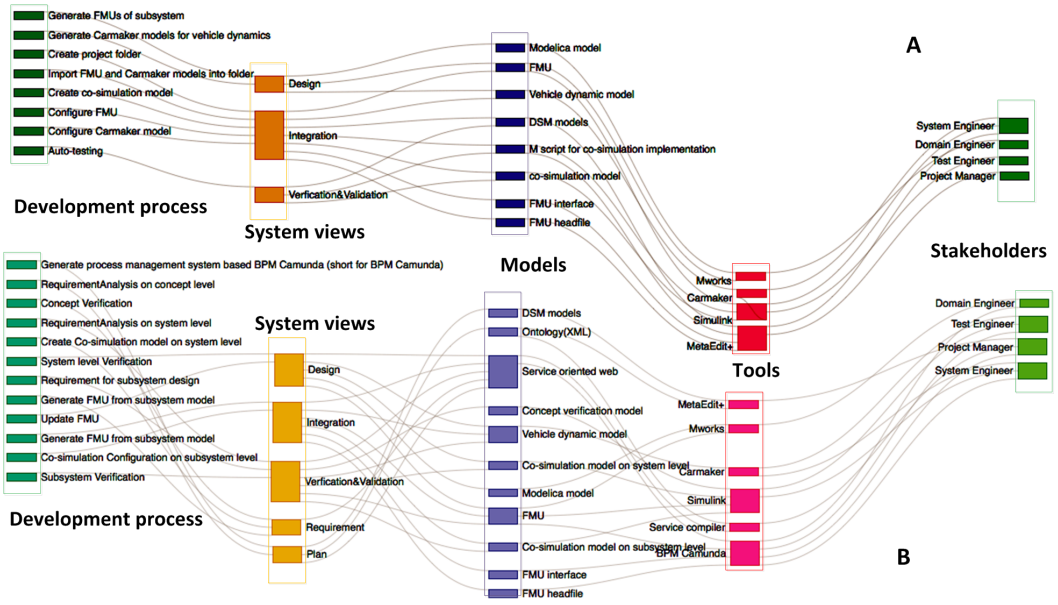

Fig. 6. Visulizations of tool-chain complexities. *A (Tool-chain A)*-The complexity of tool-chain based on *SPIT*; *B (Tool-chain B)*-The complexity of tool-chain based on *SPIRIT*.

In Fig. 6, a hierarchy graph illustrates two-chains (*Tool-chain A* and *Tool-chain B*) to address the complexity of tool-chains. The hierarchy graph can easily detect relationships between development process, system views, tools, models and stakeholders. The rectangles in different colors represent different entities, e.g. the green rectangles representing different phases of co-simulation process. The links between different rectangles are showing the dependencies between entities, e.g. different system views include different models. Compared with *Tool-chain A*, we can observe that there are more development processes, system views

[5] In [15], five types of tools are introduced: *Tool*, *Hardware*, *Repository*, *Sequencer* and *CosimControl*

[6] These tool-chains have been introduced in these two papers. Both of them aimed to support co-simulations for verification of auto-braking system.

and links between stakeholders and tools supported by *Tool-chain B* which means *Tool-chain B* has higher complexity than *Tool-chain A* on these aspects.

Moreover, complexity of cognition views refers to how technical resources at different cognition levels link to each other. For example, linkings between models at information level and at data levels can illustrate automation levels of given tool-chains - the more linkings, the more automated. As shown in Fig. 7, a directed graph is used to illustrate how DSM models at information level support co-simulation automation. The orange and green cycles represent DSM models and models used in co-simulations. The green triangles represent model connections. In the case study, DSM models, referring to models at information level, are adopted both in *Tool-chain A* and *Tool-chain B* to describe development process and information of co-simulations. Models used in co-simulations refer to models at data level to support co-simulation implementation. Both of these tool-chains adopt DSM models to refer (Reference connection) to/control (Function wrap connection)/transform to (Model transformation connection)/map (Data map connection) models implementing co-simulations. From the result, we can find that DSM models in *Tool-chain B* can links more models than the ones in *Tool-chain A*. This can be inferred that *Tool-chain B* has more effectiveness of co-simulation automation.

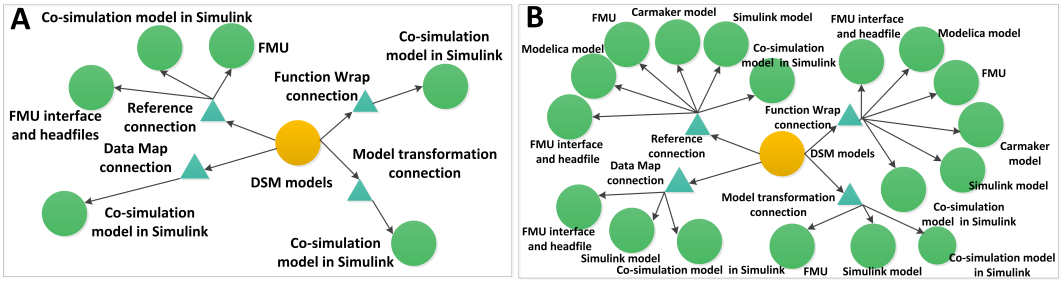

Fig. 7. Automation Effectiveness of co-simulation operations. A-Automation Effectiveness of *Tool-chain A*; B-Automation Effectiveness of *Tool-chain B*.

In Fig.8, another directed graph is used to describe model flows and tool-chain concepts of *Tool-chain A* and *Tool-chain B*. The green, orange and blue cycles represent models, tools and stakeholders. The connections between such elements refer to model and tool connections. In the model flows (Fig.8-A and Fig.8-B), we can find M scripts generated by DSM models in *Tool-chain A* can support partial operations of technical resources, however, in *Tool-chain B*, service-oriented webs, generated from DSM models, can support more operations, e.g. related operations for Modelica models and FMUs. This can be explained that the service-oriented webs can control MWorks (a Modelica tool [8]) to load Modelica models and generate FMUs. Moreover, when we compared tool-chain concepts (Fig.8-C and Fig.8-D), we can find that stakeholders' social networks are centralized in *Tool-chain B*, though more tools are used. This is because of the fact that, a process management system based on BPM Camunda is used to implement co-simulation. Stakeholders only need to implement tool-operations on the web-based process management system without manual operations of other tools.

The comparison of *Tool-chain A* and *Tool-chain B* can provide us with clues about their effectiveness and flexibility to support co-simulations. First, through the visualizations in Fig. 6, we find the *Tool-chain B* can support more complex co-simulation scenarios. Second, we compared effectiveness of co-simulation automation in Fig. 7. The number of models implementing co-simulations which DSM models can link to through model connections in *Tool-chain B* are more than the one in *Tool-chain A*. This indicates that *Tool-chain B* can support automation effectively than *Tool-chain A*. Third, compared with the model flows of *Tool-chain A* in Fig. 8, service-oriented webs in *Tool-chain B* can link to more technical resources than *Tool-chain A*. This is because the service-oriented approach used in *Tool-chain B* enhances the tool interoperability. In closing, stakeholders' social networks are more centralized in *Tool-chain B*. They can make use a web-based process management system based on BPM Camunda to implement co-simulations automatically without manual operations in other tools.

By using visualization of tool-chain concepts based on the 3D viewpoint-based method, we can deduct the tool-chains based on ***SPIRIT*** framework can promote effectiveness of co-simulations. However, the assessment approach of the evolution processes has several limitations:

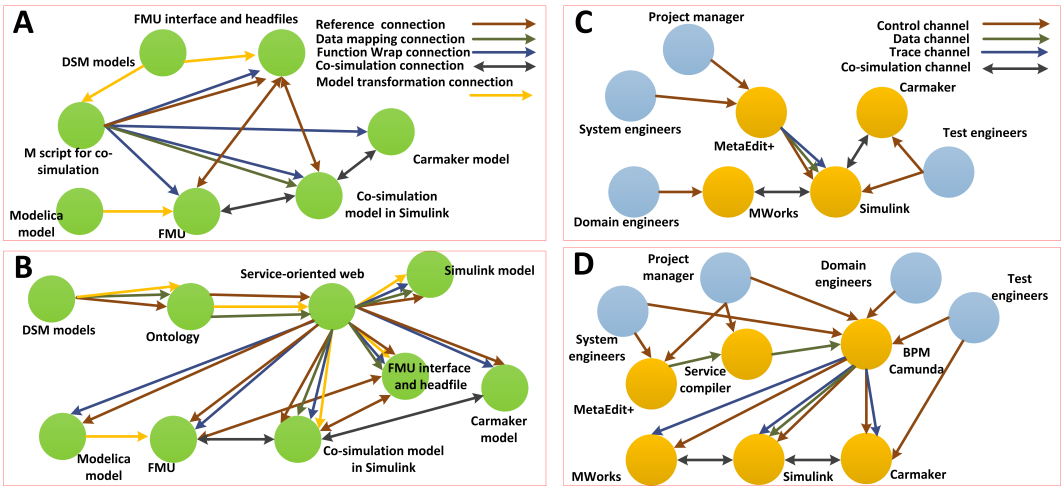

Fig. 8. Visualizations of model flows and tool-chains. A-Model flow of *Tool-chain A*; B-Model flow of *Tool-chain B*; C-Tool-chain concept of *Tool-chain A*; B-Tool-chain concept of *Tool-chain B*

- **Lack of quantitative measurements supporting tool-chain assessment**: Quantitative measurements include readiness levels, capability levels, e.g. *People Capability Maturity* [9] and other metrics have not been considered in this paper.
- **Lack of model-based approach to support tool-chain assessment**: Currently, the concept models of related tool-chains are formalized by a set of data including basic elements of model flow and tool-chain, e.g. models, model connections and tools. Current data is analyzed by an inefficient and manual analysis process. Therefore a model-based approach is needed to formalize the tool-chains and generate related data in order to decrease the time-consumption used for data analysis.
- **The need of more detailed concerns of tool-chain developers**: The existing viewpoints of development processes, system and cognition views are not enough to capture more concerns about complex tool-chain development and assessment. More detailed and accurate information about such concerns is needed to formalize tool-chains and make a more convincing assessment.
- **The need of automated process supporting visualization**:Currently, visualizations cannot be implemented by an automated way which also influences the efficiency of assessment. After a model-based approach is used for formalizing tool-chain concepts, related visualizations can be generated automatically.

6 Conclusion and Future Work

This paper presents an empirical evolution of frameworks supporting co-simulation tool-chain development. The evolution indicates an updated process from a ***SPIT*** framework to a ***SPIRIT*** framework. Then this paper introduce a 3D viewpoint-based method to formalize tool-chain concepts and evaluate them by visualizations. After knowing more measurements, a further research based on a DSM approach will be used to formalize, describe, analyze and evaluate effectiveness of co-simulation tool-chains' concepts by quantitative approaches.

References

1. Allen, J.: Managing Data and the Testing Process in the MBD Environment. Tech. rep., SAE Technical Paper (2014)
2. Bajaj, M., Backhaus, J., Walden, T., Waikar, M., Zwemer, D., Schreiber, C., Issa, G., Martin, L.: Graph-Based Digital Blueprint for Model Based Engineering of Complex Systems. In: INCOSE International Symposium. vol. 27, pp. 151–169. Wiley Online Library (2017)
3. Becker, D., Singh, R.K., Tell, S.G.: An engineering environment for hardware/software co-simulation. In: Proceedings of 29th Design Automation Conference, ACM/IEEE. pp. 129–134. IEEE (1992)
4. Biehl, M., El-Khoury, J., Loiret, F., Törngren, M.: On the modeling and generation of service-oriented tool chains. Software & Systems Modeling 13(2), 461–480 (2014)
5. Broman, David and Lee, Edward A. and Tripakis, Stavros and Törngren, Martin: Viewpoints, formalisms, languages, and tools for cyber-physical systems. Proceedings of the 6th International Workshop on Multi-Paradigm Modeling - MPM '12 1(212), 49–54 (2012)

6. Brown, A.W., Penedo, M.H.: An annotated bibliography on integration in software engineering environments. ACM SIGSOFT Software Engineering Notes 17(3), 47–55 (1992)
7. Broy, M., Feilkas, M., Herrmannsdoerfer, M., Merenda, S., Ratiu, D.: Seamless model-based development: From isolated tools to integrated model engineering environments. Proceedings of the IEEE 98(4), 526–545 (2010)
8. Chen, X., Wei, Z.: A new modeling and simulation platform-MWorks for electrical machine based on Modelica. In: International Conference on Electrical Machines and Systems 2008 (ICEMS 2008). pp. 4065–4067. IEEE (2008)
9. Curtis, B., Hefley, B., Miller, S.: People Capability Maturity Model (P-CMM) Version 2 . 0 , Second Edition (July) (2009)
10. Elaasar, M., Neal, A.: Integrating modeling tools in the development lifecycle with oslc: A case study. In: International Conference on Model Driven Engineering Languages and Systems. pp. 154–169. Springer (2013)
11. Fitzgerald, J., Gamble, C., Payne, R., Larsen, P.G., Basagiannis, S., Mady, A.E.D.: Collaborative Model-based Systems Engineering for Cyber-Physical Systems, with a Building Automation Case Study. In: INCOSE International Symposium. vol. 26, pp. 817–832. Wiley Online Library (2016)
12. Hall, A.D.: Three-dimensional morphology of systems engineering. IEEE Transactions on Systems Science and Cybernetics 5(2), 156–160 (1969)
13. ISO/IEC, .: Systems and software engineering - Recommended practice for architectural description of software-intensive systems, vol. 2007 (2007)
14. Jinzhi Lu, Dejiu Chen, F.L., Törngren, M.: A Model-driven and Tool-integration Framework for Whole Vehicle Co-simulation Environments. In: 8th European Congress on Embedded Real Time Software and Systems (ERTS 2016), Jan 2016, TOULOUSE, France. No (2016)
15. Jinzhi Lu, Dejiu Chen, J.W., Torngren, M.: A Tool Integration Language to Formalize Co-simulation Tool-chains for Cyber-physical System (CPS). In: 1st Workshop on Formal Co-Simulation of Cyber-Physical Systems of SEFM2017. Springer (2017)
16. Krammer, M., Marko, N., Benedikt, M.: Interfacing Real-Time Systems for Advanced Co-Simulation-The ACOSAR Approach. In: STAF Doctoral Symposium/Showcase. pp. 32–39 (2016)
17. Larsen, P.G., Fitzgerald, J., Woodcock, J., Fritzson, P., Brauer, J., Kleijn, C., Lecomte, T., Pfeil, M., Green, O., Basagiannis, S., et al.: Integrated tool chain for model-based design of Cyber-Physical Systems: The INTO-CPS project. In: Modelling, Analysis, and Control of Complex CPS (CPS Data), 2016 2nd International Workshop on. pp. 1–6. IEEE (2016)
18. Lu, J., Chen, D.J., Gürdür, D., Törngren, M.: An Investigation of Functionalities of Future Tool-chain for Aerospace Industry. In: INCOSE International Symposium. vol. 27, pp. 1408–1422. Wiley Online Library (2017)
19. Lu, J., Chen, D., Tao, X., Wang, J., Torngren, M.: Towards a tool-chain supporting automated parameter estimation of autonomous driving system. under review. In: Design Automation Conference (DAC), 2018. pp. 1–6. IEEE (2018)
20. Mengist, A., Pop, A., Asghar, A., Fritzson, P.: Traceability support in openmodelica using open services for lifecycle collaboration (OSLC). In: Proceedings of the 12th International Modelica Conference, Prague, Czech Republic, May 15-17, 2017. pp. 823–830. No. 132, Linköping University Electronic Press (2017)
21. Modelica Association Project FMI: Functional Mock-up Interface for Model Exchange and Co-Simulation (07006), 1–120 (2013)
22. Rowley, J.: The wisdom hierarchy: representations of the DIKW hierarchy. Journal of information science 33(2), 163–180 (2007)
23. Sánchez, B.B., Alcarria, R., Sánchez-Picot, Á., Sánchez-de Rivera, D.: A Methodology for the Design of Application-Specific Cyber-Physical Social Sensing Co-Simulators. Sensors 17(10), 2177 (sep 2017), http://www.mdpi.com/1424-8220/17/10/2177
24. Schamai, W., Fritzson, P., Paredis, C.J.J., Helle, P.: ModelicaML value bindings for automated model composition. Proceedings of the 2012 Symposium on Theory of Modeling and Simulation - DEVS Integrative M&S Symposium pp. 31:1–31:8 (2012)
25. Schopfer, G., Yang, A., von Wedel, L., Marquardt, W.: CHEOPS: A tool-integration platform for chemical process modelling and simulation. International Journal on Software Tools for Technology Transfer 6(3), 186–202 (2004)
26. Tschirner, Christian and Bretz, Lukas and Dumitrescu, Roman: Applying Model-Based Systems Engineering for Product Engineering Management Concepts for Industrial Application pp. 42–49 (2015)
27. Wicks, M.: Tool integration within software engineering environments: An annotated bibliography. Heriot-Watt University, Tech. Rep (2006)
28. Zhang, H., Wang, H., Chen, D., Zacharewicz, G.: A model-driven approach to multidisciplinary collaborative simulation for virtual product development. Advanced Engineering Informatics 24(2), 167–179 (2010)