Semantic Integration Platform for Cyber-Physical System Design

Qishen Zhang
Institute for Software Integrated Systems
Vanderbilt University
Nashville, TN USA
qishen.zhang@vanderbilt.edu

Tamas Kecskes
Institute for Software Integrated Systems
Vanderbilt University
Nashville, TN USA
tamas.kecskes@vanderbilt.edu

Ted Bapty
Institute for Software Integrated Systems
Vanderbilt University
Nashville, TN USA
ted.bapty@vanderbilt.edu

Janos Sztipanovits
Institute for Software Integrated Systems
Vanderbilt University
Nashville, TN USA
janos.sztipanovits@vanderbilt.edu

***Abstract*—Cyber-Physical Systems (CPS) are establishing heterogeneous engineering domains leading to engineering processes that span multiple design disciplines with separate modeling approaches, design flows and supporting tool suites. One of the challenges of design automation in CPS is the deep integration of models, tools and design flows such that design trade-offs across traditionally isolated design disciplines is facilitated. In this paper we overview experience and results gained along the implementation of an experimental design automation tool suite, OpenMETA, created for a complex CPS design challenge in the ground vehicle domain. The focus of the paper is domain agnostic methods and tools providing infrastructure for the model- and tool- integration platforms in OpenMETA. We present the arguments leading to the creation of the integration platforms instead of pursuing ad-hoc integration of heterogeneous tools and provide details on facilitating semantic integration.**

Keywords—cyber-physical systems, design automation, model integration, tool integration, semantic integration

I. Introduction

Cyber-Physical Systems (CPS) are engineered systems where functionalities and essential properties emerge from the networked interaction of physical and computational processes. Since key system-level functionalities and their properties cannot be assigned to only physical or only computational components, CPS design is inherently a co-design process involving several design disciplines.

CPS design flows are dominantly model- and component-based. Model-based design is characterized by the pervasive use of models throughout the design process, such as application models, platform models, physical system models, environment models, and the interaction models between these modeling aspects. The primary goal in model-based CPS design flows is achieving high level of predictability of system properties "as implemented/manufactured" at the end of the design process. A typical problem of the current systems engineering practice is that limited predictability forces the development process to iterate over lengthy design → build/manufacture → integrate → test → redesign cycles until all essential requirements are achieved. There are three fundamental contributors to radically shortening systems development time: (1) selecting the level and scope of abstractions in the design flow, (2) reusing design knowledge captured in component model libraries and using compositional design methods, and (3) introducing extensive automation in the design flow for executing rapid requirements evaluation and design trade-offs.

In this paper we provide an overview of a design automation tool suite, OpenMETA [2][4], summarize key lessons learned in a large scale application and present new research results in developing a key component, the semantic integration tool suite for models and tools.

II. Integration Platforms

Integration in CPS design flows is a fundamental challenge. We start the discussion with a short summary of integration platforms for CPS design. Detailed discussion of the topic can be found in [2].

A. *Conceptual Framework for Model- and Component- Based Design*

The goal of the model-based design process is to obtain a formal description (model) of the system *S*, using as input the set of requirements *R* that must be satisfied in some environment *E*. If *S*, *R* and *E* can be represented formally, the system design can be framed as a synthesis process **Error! Reference source not found.** of *S* such that when *S* composed with the environment *E*, it satisfies the requirements *R*, or $S \parallel E \models R$ (Fig. 1).

Affordability of model-based design requires extensive reuse of design knowledge manifested in two basic forms: component models for the system to be designed and analysis methods, tools and processes that are integrated into testbenches for evaluating the design models against the requirements. With this consideration, model- and component-based design can be conceptualized as a design space exploration process that – by using some exploration strategy - incrementally narrows the size of the initial design space defined by the component library until (one or more) satisfying design is found. The Multidisciplinary Verification, Testing and Optimization activities (Fig. 1) evaluate designs points for *S* against different requirements in *R* while operating in the targeted environment *E*. Reuse of analysis tools and processes is achieved by composing the analysis testbenches that incorporate specific configurations of tools for implementing analysis flows. Testbenches are linked to requirement categories (e.g. mobility requirements that are tested using a dynamic simulation testbench) and well suited for model-based integration. Testbench models that incorporate the model of an analysis flow and tool interfaces can be templatized and placed in a Testbed Template Repository. A testbed template is instantiated by configuring it with specific requirement models, and with a suite of design models required by the incorporated analysis tools. The Testbench Integration process deploys a configured testbench for execution. As Fig. 1 shows we conceptualize the model- and component- based design process as the interaction of two subprocesses: (1) the design space exploration process that searches architecture variants and composes analysis models for evaluation, and (2) the requirement specific analysis processes that select and instantiates testbench models, composes the testbench models with system models and environment models, integrates the testbenches and executes the analysis.

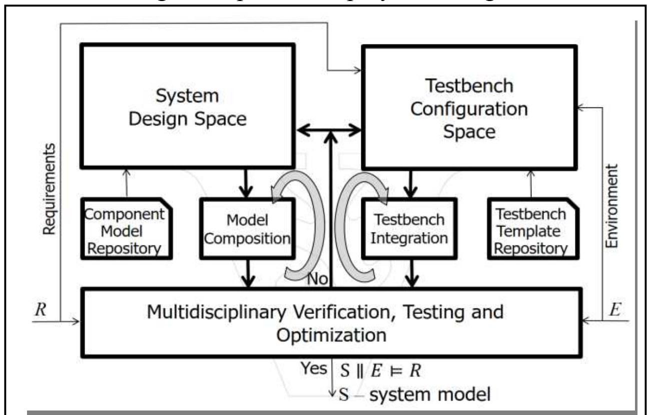
System Design Space
Testbench Configuration Space
Requirements
Environment
Component Model Repository
Model Composition
Testbench Integration
Testbench Template Repository
No
R
E
Multidisciplinary Verification, Testing and Optimization
Yes
S ∥ E ⊨ R
S – system model

Figure 1: Model- and component- based design process

B. Horizontal Integration Platforms

The design space exploration process usually proceeds from early conceptual design towards detailed design using models and simulation-based virtual prototyping. This progressive refinement process starts with the composition of abstract system models from component models that capture essential aspects of the system behavior. The system models are evaluated against requirements using simulation and verification tools. The promising designs are refined using higher fidelity component models and more detailed modeling abstractions. The design process is completed by optimizing relatively few, high fidelity models. The automation of this design process has been a fundamental goal of the OpenMETA tool suite [5][6].

To facilitate the seamless integration of heterogeneous models and tools, OpenMETA complemented the traditional, vertically structured and isolated model-based tool suites with horizontal integration platforms for models, tools, and executions as shown in Fig. 2. The function of the integration platforms is summarized below.

1) Model Integration Platform

The modeling functions of the OpenMETA design flow are built on the introduction of the following model types:

1. Component Models that include a range of domain models representing various aspects of component properties and behaviors, a set of standard interfaces through which the components can interact, and the mapping between the component interfaces and the domain models.
2. Design Models that describe system architectures using components and their interconnections.
3. Design Space Models that define architectural and parametric variabilities of design models.
4. Environment Models that capture the external effects, entities that interact with the designed system.
5. Testbench Models that specify analysis models and analyzes flows for computing key performance parameters linked to specific requirements.

The first three model types focus on the designed system, while the last two represent models used in Multidisciplinary Verification, Testing and Optimization processes. Since these different model types are heterogeneous and still need to be composed for specific analysis processes, CPS face a fundamental model-integration challenge.

The Model Integration Platform supports the integration of domain-specific modeling languages used in discipline specific design verticals, such as architecture, real-time software, CAD, fluid dynamics and others. Being an integration platform, it adopts the philosophy and key characteristics of platform-based design [7]. The key challenge to address is the design of domain and discipline independent tools that support model integration.

2) Tool Integration Platform

The role of the Tool Integration Platform is to integrate testbenches that evaluate designs against various requirements. A testbench usually executes an analysis flow incorporating already existing simulation, verification, data analysis and visualization tools. As Fig. 2 shows the Tool Integration Platform includes the following elements:

1. Component Exporters for extracting relevant component models from the Component Model Repository
2. Model Composers for synthesizing analysis models for the testbenches from component and design models,
3. A composed suite of analysis and simulation models and workflow models for testbenches,
4. Testbench Integrator that configures the testbenches and instantiates them for execution.

A variety of domain and discipline independent tool integration solutions exist both open source and proprietary variants. Notable open source examples are OpenMDAO and the High Level Architecture (HLA). OpenMDAO (Multidisciplinary Design Analysis and Optimization) is an optimization framework [8] that controls the execution of optimization workflows incorporating different tools. Besides being an execution platform for tool invocation, OpenMDAO supports gradient-based optimization with analytic derivatives to allow the exploration of large design spaces with hundreds or thousands of design variables. It can also work with gradient-free optimization, mixed-integer nonlinear programming, and traditional design space exploration as well. HLA is an IEEE standard for integrating simulators on distributed computing platforms [9]. The standard goes way beyond just invoking simulators in some workflow or wrapping a simulator code in a standard interface. It provides elaborate solution for time management and distributed object management – the real hard problems in integrating distributed simulation tools. Using open standards as a backbone for tool integration has many advantages in implementing heterogeneous design flows mainly in terms of reusability of tools and methods.

3) Execution Integration Platform

Improving the coverage and efficiency of design space exploration requires the highly automated execution of analysis processes using a variety of deployed tools that need to receive and produce models, and need to be scheduled as the analysis flow requires. The Execution Integration Platform incorporates the services and mechanisms required for running the analyses processes and generating the analysis artifacts. In this paper we do not discuss implementation details of execution platforms for design automation. Example for a cloud-based execution platform can be found in [10]..

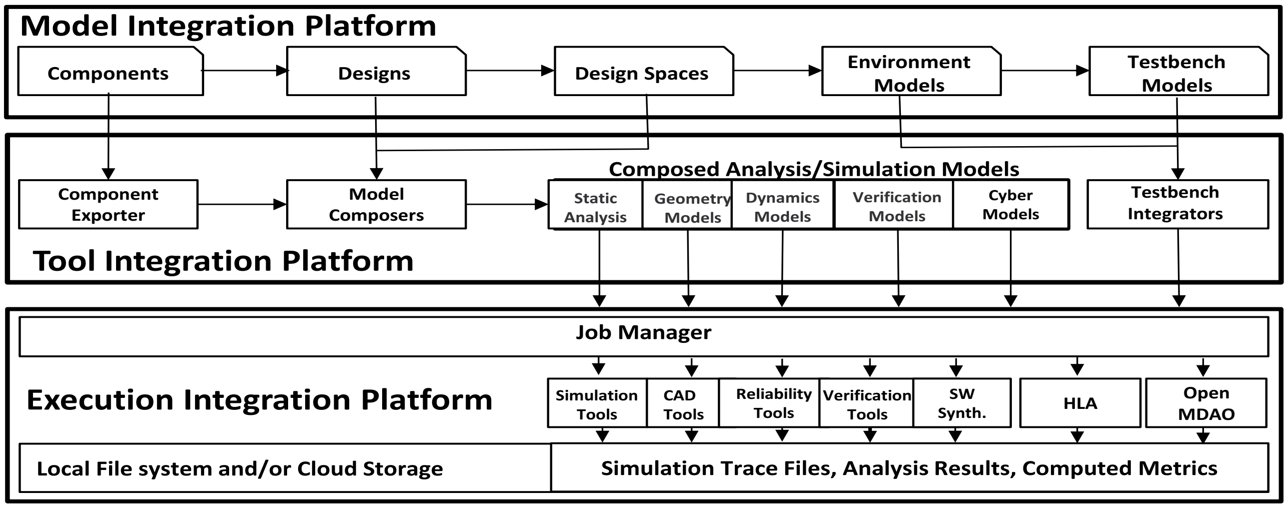

Figure 2: Horizontal Integration Platforms

III. NEED FOR SEMANTIC INTEGRATION

The integration platforms summarized above emerged along the development of a fully integrated model-based design tool chain prototype for complex CPS, as part of the Defense Advanced Research Project Agency (DARPA) Adaptive Vehicle Make (AVM) program [11]. The resulting design automation tool chain, OpenMETA [4] was first tested in the FANG-1 challenge prize competition [12][13] focusing on the drivetrain and mobility subsystems of a ground infantry vehicle. Since the competition was also a crowdsourcing experiment with over 1,000 individuals grouped in over 200 design teams [13], we have learned many lessons on integrating large design automation tool suites for CPS and providing web-based access to distributed design teams. In the rest of this paper we will focus on the need and role for semantic integration in the Model Integration Platform and Tool Integration Platform because of the far reaching implications on the overall tool architecture.

A. *Model Integration Challenge*

Existing approaches for tool integration by and large neglect the fundamental importance of model integration for heterogeneous tool suites. Fig. 3 illustrates the problem.

As discussed before, the complexity of the design space exploration process requires progressive refinement where the process starts using high level abstractions spanning large number of variants and progressively narrows the number of candidates using increasingly complex models. In CPS the models used in the process are inherently heterogeneous. Sources of heterogeneity can be structured along three dimensions (see left side of Fig. 3):

5. Hierarchical component abstractions that represent CPS designs on different levels of details and fidelity.
6. Modeling abstractions that span a wide range of mathematical models such as static models, discrete event models, lumped parameter dynamic models, hybrid dynamics, geometry and partial differential equations,
7. Analysis domains including software, mechanical, electrical, thermal, hydraulic and other.

While CPS design requires the exploration of the integrated design space, the widely used "separation of concerns" principle establishes "slices" in this complex space such as physical dynamics domain, computer aided design (CAD) domain, electronic CAD (E-CAD) domain, or finite element analysis (FEA) domain. These individual design domains are relatively isolated, linked to different engineering disciplines and supported by domain-specific tool suites (right side of Fig. 3). Since the existing tool suites represent enormous value in terms of design knowledge, established modeling languages and model libraries, the only reasonable approach to providing support for CPS design flows is to reuse existing assets. This approach works well if the design concerns are independent, which is usually not the case -

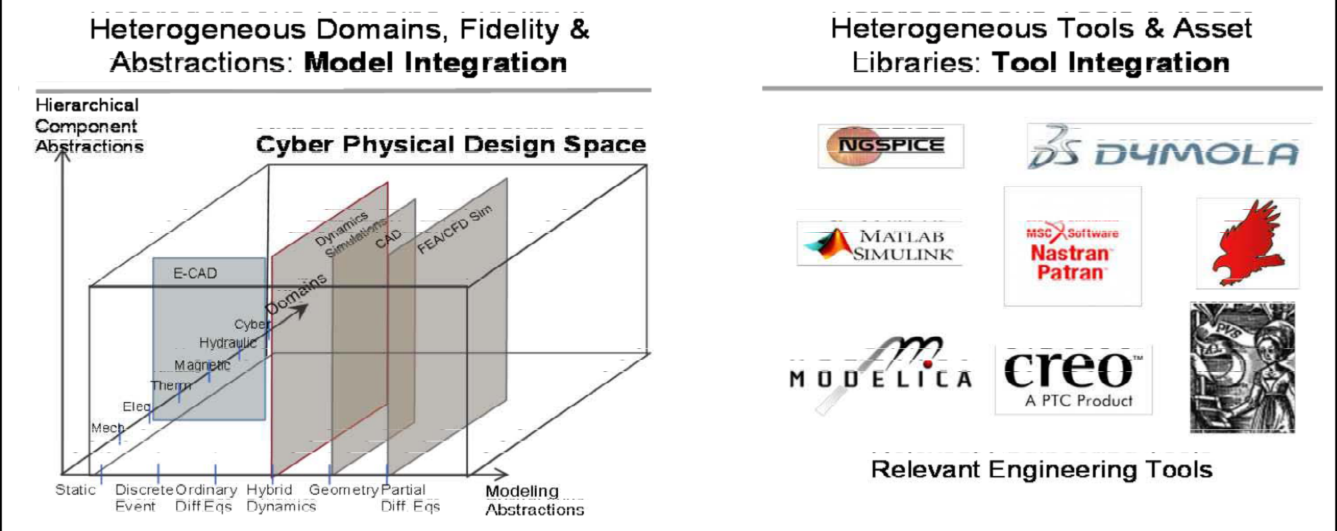
Heterogeneous Domains, Fidelity & Abstractions: Model Integration
Hierarchical Component Abstractions
Cyber Physical Design Space
E-CAD
Domains
Dynamics Simulations
CAD
FEA/CFD Sim
Cyber
Hydraulic
Magnetic
Therm
Elec
Mech
Static
Discrete Event
Ordinary Diff Eqs
Hybrid Dynamics
Geometry
Partial Diff Eqs
Modeling Abstractions
Heterogeneous Tools & Asset Libraries: Tool Integration
NGSPICE
DYMOLA
MATLAB SIMULINK
MSC Software Nastran Patran
MODELICA
creo A PTC Product
Relevant Engineering Tools

Figure 3: Model Integration Challenge

unless the system is specifically architected for decoupling selected design concerns [14]. Neglecting interdependences across design concerns is one of the primary sources of anomalies and unexpected behaviors detected during system integration. In conclusion, finding solution for the model integration and tool integration challenges are the only practical approach for creating CPS design tool suites.

The role of the Model Integration Platform is to allow the development of a reusable infrastructure that is agnostic to the individual tools needed in product specific design flows. A core concept in OpenMETA was the introduction of a Model Integration Language (CyPhyML) for representing component models, system architectures (designs) in terms of component interactions, design spaces, composition constraints and cross-domain interactions. The Model Integration Language is influenced by existing and emerging model-based design, verification and simulation technologies and tools. Consequently, the language suite and the related infrastructure cannot be static; it will continuously evolve. To address both heterogeneity and evolvability simultaneously, we departed from the most frequently used approach to address heterogeneity: the development or adoption of a very broad and necessarily hugely complex language standard designed for covering all relevant views of multi-physics and cyber domains. In our philosophy, the Model Integration Languages are kept as simple as possible reflecting the specific needs of model integration and not the aggregate complexity of the domains.

B. *Model Composition Challenge*

The Model Composers (see Fig 2.) are the bridge between the tool agnostic Model Integration Platform and the analysis tools incorporated in the Tool Integration Platforms. Their roles are: (1) composing testbench- (and incorporated tool-) specific analysis models from architecture and multi-fidelity component models, (2) integrate those with the testbench models to obtain executable testbench specification and (3) prepare the fully configured testbench models for deployment on the Execution Integration Platform.

Model Composers are implemented as networks of model transformations. They are crucial in ensuring the semantic integrity of the design flow in the context of heterogeneous modeling languages and analysis tools. Details about designing and implementing Model Composers can be found in [2].

IV. Towards a Tool suite for semantic integration

In a naïve approach, model and tool integration is considered to be an interoperability issue that can be taken care of with appropriate standards and conversions. In complex design problems these approaches inevitably fail due to the rapid loss of control over the semantic integrity of design flows. The problem is that core constituents of designs flows have very different lifecycles and evolution trajectories: the product is changing with market needs and technology innovations (e.g. increased use of composite materials in structures or incorporation of learning enabled components in control), which rapidly change the design flows. In addition, established tool suites (e.g. for system-level verification and software design) are also changing driven by progress in CPS foundations and changes on computing platforms. Keeping integrity of design flows in a continuously shifting environment with multiple stakeholders and vendors is hard. Model Integration Languages (like CyPhyML in OpenMETA) play essential role in decoupling these differing evolution trajectories via enabling the explicit modeling their interdependencies.

The “cost” of introducing a dynamic Model Integration Language in a design automation context is that its semantics cannot be handled loosely or informally; there is a need for mathematically precise *semantic integration*. This recognition led to the development of the Semantic Backplane concept in OpenMETA [15][16]. The Semantic Backplane is a collection of formal specification of all CyPhyML constructs (component models, ontologies, designs (system architectures), design spaces, cross-domain interactions, composition constraints, data model interfaces for tools) and all model transformations in Model Composers in a formal language, FORMULA-2 provides a convenient formalism for representing, validating, and transforming models. Unified model validation and transformation using open-world logic programming (OLP), which allows both validation and transformation operations to be formalized as an extension of first-order logic with fix-point operations [17]. Axioms written in FORMULA-2 can be dually understood as executable programs, providing engineers with an additional mental-model for comprehending their specifications. Automated reasoning is enabled by efficient symbolic execution of FORMULA-2 logic programs into quantifier-free sub-problems, which are dispatched to the state-of-the-art SMT solver Z3 [18]. FORMULA-2 has been developed at Microsoft Research and applied within Microsoft to develop modeling languages for verifiable device drivers and protocols [19]. FORMULA-2 is released under an open-source license and can be found at https://github.com/Microsoft/formula.

A. *Semantic Backplane: Lessons Learned*

The overall size of the FANG-1 challenge problem [13] focusing on the drivetrain and mobility aspects of the vehicle and the second challenge on hull design [2] allowed us to gain experience with the practical usability of the Semantic Backplane concept. (Other lessons learned regarding OpenMETA can be found also in [2].) In this section we summarize the lessons learned related to the use of the Semantic Backplane and discuss ongoing research related to a tool suite for semantic integration.

Divergence. The Semantic Backplane was close to 20,000 lines of FORMULA-2 specification of CyPhyML and Model Composers. However, the fact that the "production tools" in OpenMETA (the metaprogrammable Generic Modeling Environment (GME), its metamodels, and the Javascript code of the Model Composers) were separate from their FORMULA-2 based formal specifications created the risk of divergence between the implemented integration components and their formal models.

Scalability. While FORMULA-2 allows checking the well-formedness of integration models, translating very large models to FORMULA-2 is a slow process.

Efficiency. FORMULA-2 is able to execute the specified model transformations, therefore can support their validation and verification. However, using FORMULA-2 as a production tool for transforming very large models proved to be highly impractical due to performance limitations.

Model Engineering. Being a formal framework, FORMULA-2 was not designed for providing model engineering services that are crucial for large-scale modeling efforts, such as version control, split and merge, scalability to very large models, persistency mechanisms and multi-user/concurrent modeling.

These challenges made it clear that semantic integration is at the intersection of highly different requirements, and the related use cases can be satisfied only by creating a tool suite that seamlessly integrates these widely different capabilities.

B. *Tool Suite for Semantic Integration*

Up to this point, the two fundamental tools supporting semantic integration are WebGME (http://webgme.org) and FORMULA-2. WebGME is the metaprogrammable modeling tool, the latest element of over 2.5 decades of development of Model Integrated Computing tools [21] at the Institute of Software Integrated Systems of Vanderbilt. WebGME supports the design of domain-specific modeling languages via metamodeling [20] and domain specific modeling via graphical and textual interface. The metamodels and system models are tightly integrated and stored in a version controlled database deployed on cloud resources. Concurrent modeling is transparently supported. Clients are web browser-based, resulting in platform independence and seamless client updates. The user interface supports multiple built-in visualization techniques as well as tools to create highly domain-specific views/editors. Multiple APIs are provided to interface with existing external tools, such as other modeling environments, databases, simulators, and analysis tools, as well as to enable the development of code generators. Evolution of WebGME was driven by the needs of design environments like OpenMETA and multi-model, distributed simulations [22].

To address the challenges described in the previous section, we followed a deep integration approach between WebGME and FORMULA-2. As the architecture of the system show on Fig. 4, there are several separate processes communicating to give an integrated experience to the user. On the client side, different WebGME components ensure access to the different interpretations of the model. The WebGME metamodels are manipulated by the Metamodel Editor while the models are created with the Model Editor that enforces metamodel constraints. An additional visualizer incorporated in the WebGME client is the FORMULA-2 "viewer" and code editor of the metamodels and models. Through this view, users can see an integrated FORMULA-2 document containing static and editable portions of the metamodels and models. This "formal view" of the engineering models and metamodels that are built in WebGME contains the auto-generated constraints which are derived from structural semantics of the domain alongside with any additional constraints the user might add [23]. Finally, the document shows the FORMULA-2 representation of the model that is currently under review.

Conformance checking of the model is executed on the server side, since FORMULA-2 cannot be executed on the

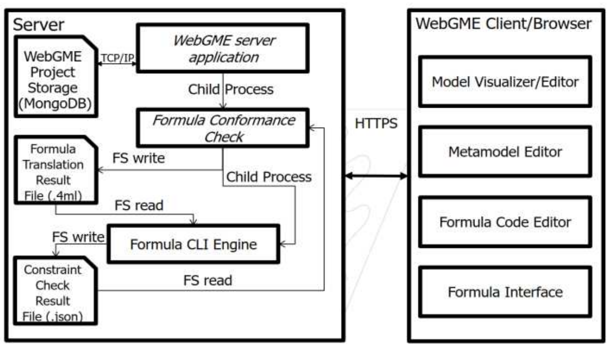

Figure 4: WebGME – FORMULA Integration

client side. The user initiates the check, which starts up the server side checking process that interacts with the FORMULA-2 Engine. When the checking finished, the result is read from an output file and sent back to the client that visualizes it to the user.

While the results for conformance check require a server roundtrip, the FORMULA-2 code editor can follow any domain changes initiated in the WebGME Editors in a synchronous manner. This means that the translation of the metamodel information and meta elements of the model occurs in real time so the FORMULA-2 view of the models and metamodels is kept fully synchronized. It also allows multiple users to interact with the same model, though editing the constraints and checking conformance at the same time might end up in outdated results.

V. Conclusion

The focus of the paper was the complex relationship between product-specific engineering processes and vendor specific tool suites in heterogeneous CPS design flows. Based on our experience with model- and component- based design automation tool suites, we believe that establishing decoupling between the two is feasible by introducing domain independent model- and tool- integration platforms. A key component of these platforms is a tool suite for semantic integration that provides effective tool support for specifying, generating and evolving Model Integration Languages and Model Composers.

Acknowledgment

This work was supported in part by the Defense Advanced Research Project Agency under awards #FA-8650-10-C-7075, #FA8650-10-C-7082 #HR0011-12-C-0008, and by the National Science Foundation under award #CNS-1521617.

References

[1] Sanjit A. Seshia, Shiyan Hu, Wenchao Li, Qi Zhu, "Design Automation of Cyber-Physical Systems: Challenges Advances and Opportunities", Computer-Aided Design of Integrated Circuits and Systems IEEE Transactions on, vol. 36, pp. 1421-1434, 2017, ISSN 0278-0070

[2] Janos Sztipanovits, Ted Bapty, Ethan Jackson, Xenofon Koutsoukos, Zsolt Lattman, Sandeep Neema, : "Model and Tool Integration Platform for Cyber-Physical System Design", Proceedings of the IEEE, 106(9), 1501-1526 (2018).

[3] Karsai, G., Sztipanovits, J., Ledeczi, A., Bapty, T.: Model-integrated development of embedded software. Proceedings of the IEEE 91(1), 145-164 (2003)

[4] Sztipanovits, J., Bapty, T., Neema, S., Koutsoukos, X., Jackson, E. "Design Tool Chain for Cyber Physical Systems: Lessons Learned," In Proceedings of DAC'15, pp. 106, June 07 - 11, 2015, San Francisco, CA, USA

[5] Lattmann, Zs., Nagel, A., Scott, J., Smyth, K., vanBuskirk, C., Porter, J., Neema, S., Bapty, T., Sztipanovits, J.: "Towards Automated Evaluation of Vehicle Dynamics in System-Level Design," Proceedings of the ASME 2012, IDETC/CIE 2012 August 12-15, 2012, Chicago, IL

[6] Wrenn, R., Nagel, A., Owens, R., Yao, D., Neema, H., Shi, F., Smyth, K., vanBuskirk, C., Porter, J., Bapty, T., Neema, S., Sztipanovits, J., Ceisel, J., Mavris, D.: "Towards Automated Exploration and Assembly of Vehicle Design Models," Proceedings of the ASME 2012 IDETC/CIE 2012 August 12-15, 2012, Chicago, IL

[7] Sangiovanni-Vincentell, Alberto: Quo Vadis, SLD? Reasoning About the Trends and Challenges of System Level Design, Proceedings of the IEEE, vol. 95, No. 3, pp 467-506, March 2007.

[8] http://openmdao.org

[9] IEEE 1516.1–2010 – Standard for Modeling and Simulation High Level Architecture – Federate Interface Specification

[10] Juracz, L., Z. Lattmann, T. Levendovszky, G. Hemingway, W. Gaggioli, T. Netterville, G. Pap, K. Smyth, and L. Howard. "VehicleFORGE: A Cloud-Based Infrastructure for Collaborative Model-Based Design," Vol-1118. Chapter 25. MODELS 2013,

[11] Eremenko, Paul: "Philosophical Underpinnings of Adaptive Vehicle Make," DARPA-BAA-12-15. Appendix 1, December 5, 2011.

[12] Ackerman, Spencer: "This Is the Million-Dollar Design for Darpa's Crowdsourced Swimming Tank". Wired. (22 April 2013) (Retrieved 24 April 2013, https://www.wired.com/2013/04/darpa-fang-winner).

[13] Olivier L de Weck, Eun Suk Suh. "Complex System Design through Crowdsourcing : DARPA FANG - 1 as a Case Study," Industrial Engineering Magazine 21(4), 2014.12, 32-37, url: http://www.dbpia.co.kr/Article/NODE06085424.

[14] Sztipanovits, J., Koutsoukos, X., Karsai, G., Kottenstette, G., Antsaklis, P., Gupta, V., Goodwine, B., Baras, J., Wang, S.: "Toward a Science of Cyber-Physical System Integration," Proceedings of the IEEE, Vol. 100 No. 1, pp. 29-44, 2012

[15] Simko, G., Levendovszky, T., Neema, S., Jackson, E., Bapty, T., Porter, J., Sztipanovits, J.: "Foundation for Model Integration: Semantic Backplane" Proceedings of the ASME 2012, IDETC/CIE 2012 August 12-15, 2012, Chicago, IL

[16] Simko, G., Sztipanovits, J.: "Model Integration in CPS," in Raj Rajkumar, Dioniso de Niz, Mark Klein (Ed.) Cyber Physical Systems, pp 331-360, Addison-Wesley (2017) ISBN-13:978-0-321-92696-8

[17] Jackson, E.: Engineering of domain-speciifc languages with FORMULA 2. HILT '13 Proceedings of the 2013 ACM SIGADA annual conference on High Integrity Language Technology.

[18] Bjørner, N., Ken McMillan and Rybalchenko. A., Higher-order Program Verification as Satisfiability Modulo Theories with Algebraic Data-types. In informal proceedings of HOPA 2013 (Workshop on Higher-Order Program Analysis.

[19] Desai, A., Vivek G., Jackson, E., Qadeer, E. Rajamani, S., and Zufferey, D.:. P: Safe asynchronous event-driven programming. In Proceedings of ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI 2013)

[20] Maroti, M., R. Kereskenyi, T. Kecskes, P. Volgyesi, and A. Ledeczi: "Online Collaborative Environment for Designing Complex Computational Systems," ICCS 2014, Volume 29, 2014, Pages 2432–2441. Cairns, Australia, 06/2014, Elsevier Procedia.

[21] Janos Sztipanovits and Gabor Karsai. Model-integrated computing. Computer, 30(4):110–111, 1997.

[22] Neema, H., Gohl, J., Lattmann, Zs., Sztipanovits, J., Karsai, G., Neema, S., Bapty, T.,, Batteh, J., Tummescheit, H., Sureshkumar: C.: "Model-Based Integration Platform for FMI Co-Simulation and Heterogeneous Simulations of Cyber-Physical Systems," Proceedings of the 10th International Modelica Conference, pp. 235-245, March 10-12, 2014, Lund, Sweden

[23] Anastasia Mavridou, Tamas Kecskes, Qishen Zhang and Janos Sztipanovits: "A Common Integrated Framework for Heterogeneous Modeling Services." GEMOC 2018, co-located with MODELS 2018. Kopenhagen, Denmark, 10/2018