OpenMETA: A Model- and Component-Based Design Tool Chain for Cyber-Physical Systems

Janos Sztipanovits, Ted Bapty, Sandeep Neema, Larry Howard
Institute for Software Integrated Systems (ISIS), Vanderbilt University
1025 16th Ave S, Suite 102, Nashville, TN 37212 USA

Ethan Jackson
Microsoft Research
One Microsoft Way, Redmond, WA 98052 USA

{sztipaj,sandeep,bapty,howardlp}@isis.vanderbilt.edu
ejackson@microsoft.com

Abstract. Model- and component-based design have yielded dramatic increase in design productivity in several narrowly focused homogeneous domains, such as signal processing, control and aspects of electronic design. However, significant impact on the design and manufacturing of complex cyber-physical systems (CPS) such as vehicles has not yet been achieved. This paper describes challenges of and solution approaches to building a comprehensive design tool suite for complex CPS. The primary driver for the OpenMETA tool chain was to push the boundaries of the "correct-by-construction" principle to decrease significantly the costly design-build-test-redesign cycles in design flows. In the discussions we will focus on the impact of heterogeneity in modeling CPS. This challenge is compounded by the need for rapidly evolving the design flow by changing/updating the selection of modeling languages, analysis and verification tools and synthesis methods. Based on our experience with the development of OpenMETA and with the evaluation of its performance in a complex CPS design challenge we argue that the current vertically integrated, discipline-specific tool chains for CPS design need to be complemented with horizontal integration layers that support model integration, tool integration and design process integration. This paper will examine the OpenMETA technical approach to construct the new integration layers, provides and overview of the technical framework we established for their implementation and summarize our experience with their application.

Keywords: Model-Based Design, Component-Based Design, Cyber Physical Systems, Design Automation, Model-Integrated Computing, Domain-Specific Modeling Language, Model Integration Language

1 Introduction

Model- and component-based design have been recognized as key technologies for radically changing productivity of CPS design [1]. Model-based design uses formal

and sufficiently complete models of physical and computational processes, their environment and their interactions. These models are mathematically and physically accurate for verifying and testing the behavior of the designed system against established requirements. The main promise of model-based design is a significant decrease or elimination of costly design-build-test-redesign iterations. The ultimate goal of model-based design is "correct-by-construction", where properties of the synthesized models of the designed system predict the properties of the implemented/manufactured system with sufficient accuracy.

Component-based design constructs systems from reusable components. A component is the superposition of two models: a behavior model and an interaction model [2]. In a model-based design flow, models of components are used for constructing valid system models. The promise of component-based design is the potentially massive productivity increase due to the reuse of design knowledge captured by the component models.

While model- and component-based design methods and tools have demonstrated significant success in several engineering domains, such as VLSI design, electronics design and specific segments of software design [6], success has been elusive for CPS. Among the reasons are the following technical barriers.

1. *Heterogeneity of CPS models*. Heterogeneity in CPS design has several dimensions such as physical phenomena, levels of abstraction used in modeling physical and computational structures and processes, and engineering disciplines involved in CPS design.
2. *Heterogeneity of design tools*. Tool chains that are used in traditional CPS design flows are discipline oriented, vertically integrated and cover "islands" in the overall design space. Integration across the tool suites is hard and usually not supported.
3. *Life-cycle heterogeneity*. A unique aspect of CPS design is the significant impact of manufacturing on system performance. In fact, design of the physical part of the system needs to be integrated with the design of manufacturing processes that will make those. Manufacturability constraints and properties of the system "as manufactured" require tradeoffs with the design even in the early conceptual design phase.

Separation of concerns is a widely used strategy to deal with heterogeneity in the design process. Its goal is to decrease design complexity by decomposing the overall design problem according to physical phenomena (electrical, mechanical, thermal, structural, etc…), level of abstraction (static, lumped parameter dynamics, distributed parameter dynamics, etc…) or engineering discipline (performance, systems engineering, software engineering, manufacturing, etc…). Consequences of this design strategy are quite significant both in terms of weakening the opportunity for correct-by-construction design, as well as performing cross-domain optimizations in CPS design flows. The chief reason is that discipline oriented design flows usually miss modeling interactions/interdependences among the various design views. The approach would work if the design concerns were orthogonal, but in tightly coupled CPS this is not the case. The price of the simplification is decreased predictability of properties of the implemented CPS and costly re-design cycles.

In 2010, the Defense Advanced Research Project Agency (DARPA) initiated the Adaptive Vehicle Make (AVM) program[1] to construct a fully integrated model- and component-based design flow for the "make" process of complex cyber-physical systems (CPS) [1]. The resulting integrated tool suite, OpenMETA, provides a manufacturing-aware design flow, which covers both cyber and physical design aspects. In order to test and demonstrate the capabilities of the new design flow and the integrated tool suite in a real-life system, the AVM program also includes the Fast, Adaptable, Next-Generation Ground Vehicle (FANG) design challenge sequence[2]. FANG is a combined design and manufacturing effort constructing and building a new amphibious vehicle (IFV) in three competitions: (1) Drive-train design challenge – FANG 1, (2) Hull design challenge – FANG 2, and (3) Full vehicle design challenge – FANG 3. While the target system for the AVM program is ground vehicle, the created infrastructure for model- and component-based design is generic and targets radical changes in the overall "make" process of large CPS systems. Through our work in leading the research on the open-source design tool suite, OpenMETA, the open-source model exchange and web-based collaborative design environment, VehicleForge, and the curation effort for the FANG component model library, we had the opportunity to gain experience with the challenges of using model- and component-based design methods in large-scale CPS.

In this paper we focus on the impact of heterogeneity on the OpenMETA tool architecture. We argue that the primary barriers to apply model- and component-based design flows for CPS are the lack of the following three integration frameworks:

1. *Model Integration Framework*. Model integration is required for expressing interactions across modeling domains - creating the need for multi-modeling. Semantic heterogeneity of domain specific modeling languages (DSMLs) used in different modeling views and the fact that DSMLs in CPS subdomains evolve more or less independently, further add to the modeling language and model integration challenge. The overall semantic complexity of CPS modeling domains and the differences among CPS product categories make the development and standardization of some form of unified CPS multi-modeling languages impractical. Instead, a different solution is needed that enables the semantically sound integration of modeling domains.

2. *Tool Integration Framework*. End-to-end tooling for complex CPS product lines such as automotive and aerospace systems is too heterogeneous and extends to too many technical areas for a single tool vendors to fully cover. In addition, significant part of the companies' design flow is supported by in-house tools that are proprietary and capture high value design IP. Integration of end-to-end tool chains for highly automated execution of design flows is such a complex task that successful examples are hard to find – even after massive investment by OEMs. Change demands robust tool integration frameworks that go well beyond the semantically weak and necessarily fragile ad-hoc connection among tools.

---

[1] http://www.darpa.mil/Our_Work/TTO/Programs/Adaptive_Vehicle_Make__(AVM).aspx

[2] http://www.darpa.mil/Our_Work/TTO/Programs/AVM/AVM_Design_Competitions_(FANG).aspx

3. *Execution Integration Platform*. The dominant approach in current tool suites is desktop integration using platforms such as Microsoft's Visual Studio[3], or Eclipse[4]. However, overall complexity and heterogeneity of CPS tool chains increasingly demand the use of software as a service (SaaS) models, web-based tool integration platforms, high performance cloud-based back-ends for model repositories and web-based distributed collaboration services.

The rest of the paper has the following structure. First, we provide an overview of the three layers and their relationships. Next, we analyze the challenges and lessons learned in the design and implementation of the Model Integration Framework and show the significance of Model Integration Languages in CPS design flows. Finally we discuss application experience and summarize the ongoing research efforts.

2 OpenMETA Integration Layers

Achieving the goal of "correct-by-construction" design requires that models and analysis methods in the design phase predict with the required accuracy the behavior of the designed system. Our approach to improve the predictability of design has been the explicit modeling of multi-physics, multi-abstraction and multi-fidelity interactions and providing methods for composing heterogeneous component models.

The OpenMETA design flow is implemented as a multi-model composition/synthesis process that incrementally shapes and refines the design space using formal, manipulable models [3][19]. The model composition and refinement process

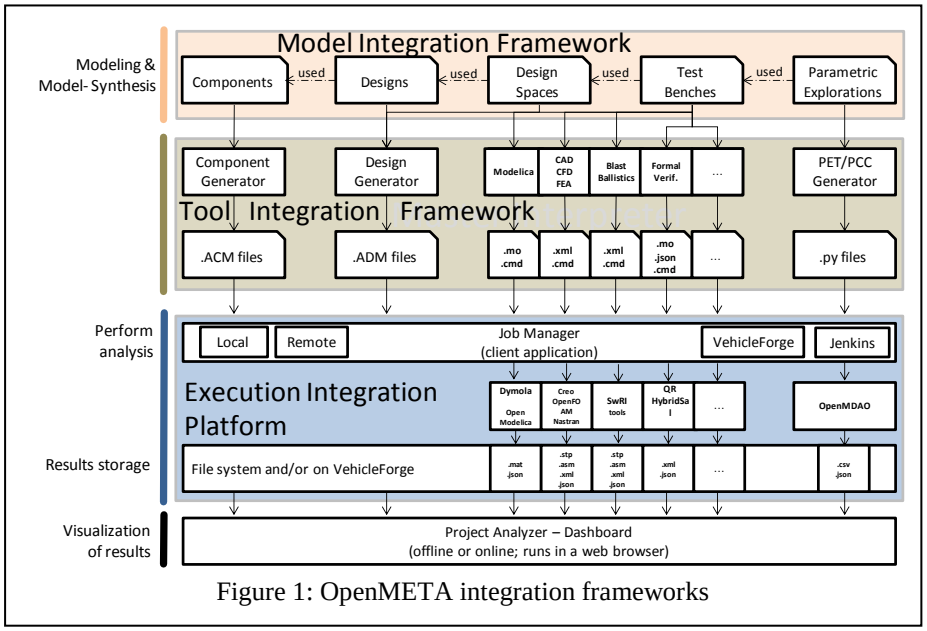

Figure 1: OpenMETA integration frameworks

[3] http://www.visualstudio.com/

[4] http://www.eclipse.org/

is intertwined with testing and analysis steps to validate and verify requirements and to guide the design process toward the least complex, therefore the least risky and least expensive solutions. The design flow follows a progressive refinement strategy, starting with early design-space exploration covering very large design spaces using abstract, lower fidelity models and progressing toward increasingly complex, higher fidelity models and focusing on rapidly decreasing number of candidate designs.

The META design flow proceeds in the following main phases:

1. Combinatorial design space exploration using static finite domain constraints and architecture evaluation.
2. Behavioral design space exploration by progressively deepening from qualitative discrete behaviors to precisely formulated relational abstractions and to quantitative multi-physics, lumped parameter hybrid dynamic models using both deterministic and probabilistic approaches.
3. Geometric/Structural Design Space Exploration coupled with physics-based nonlinear finite element analysis of thermal, mechanical and mobility properties.
4. Cyber design space exploration (both HW and SW) integrated with system dynamics.

As discussed before, automation of the design flow leads to complex integration challenges that we decomposed into three integration layers shown in Figure 1. Elements of the framework reflect primarily the FANG 1 drive-train challenge, but the basic structure of the integration architecture remains the same for the FANG 2 hull design challenge as well, with larger emphasis on 3-D/CAD tools and a range of finite element analysis for verifying blast protection and hydrodynamic requirements.

In the following sections we describe each integration framework with more emphasis on model integration.

3 Model Integration Framework – Semantic Integration

The modeling and model-synthesis functions of the OpenMETA design flow is built on the introduction of the following model types:

1. AVM Component Models (ACM) with standard, composable interfaces
2. Design Models (DM) that describe component architectures and related constraints
3. Design Space Models (DSM) that define structural and architectural variabilities
4. Test Bench Models (TBM) representing environment inputs, composed system models connected to a range of testing and verification tools for key performance parameters, and
5. Parametric Exploration Models (PEM) for specifying regions in the design space to be used for optimization and models for complex analysis flows producing results such as Probabilistic Certification of Correctness (PCC).

In META, as well as in all other approaches to model-based design, modeling languages and their underlying semantics play a fundamental role in achieving compositionality. Heterogeneity of the multi-physics, multi-abstraction and multi-fidelity design space, and the need for rapidly evolving/updating design flows require the use

of a rich set of modeling languages usually influenced/determined 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 a multi-physics and cyber domains. Instead, we placed emphasis on the development of a model integration language – CyPhyML – with constructs limited to modeling the interactions among different modeling views (see Figure 2).

3.1 Model Integration Language and Semantic Interfaces

CyPhyML targets multi-modeling – it advances multi-modeling from a mere "ensemble" of models to a formally and precisely integrated, mathematically sound suite of models. Integration of the modeling language suite by CyPhyML is minimal in a sense that only those abstractions that are imported from the individual languages to CyPhyML are those required those for modeling cross-domain interactions. Since the suite of engineering tools is changing and the modeling languages of the individual tools (such e.g. Modelica) evolve independently from the model integration framework, CyPhyML is constructed as a light-weight, evolvable, composable integration language that is frequently updated and morphed. While these DSMLs may be individually quite complex (Modelica, Simulink, SystemC, etc.) ChyPhyML is relatively

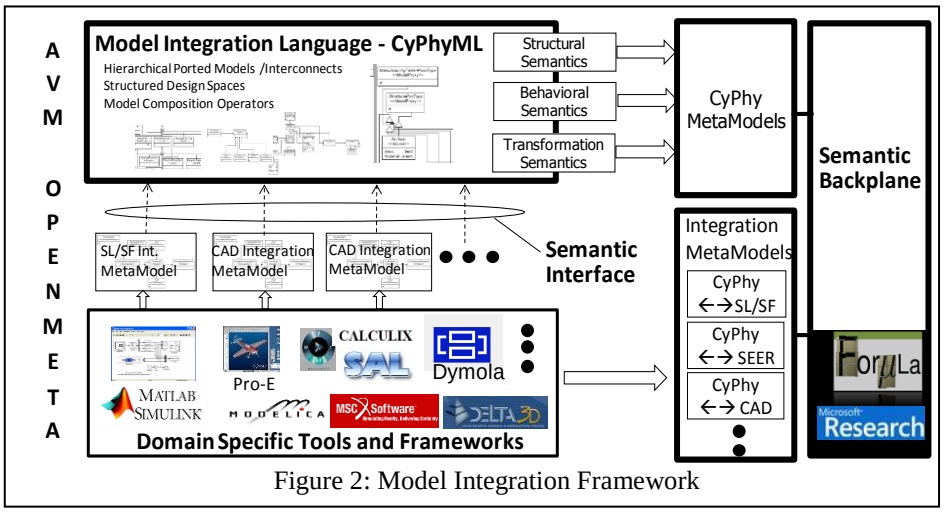

Figure 2: Model Integration Framework

simple and easily evolvable. This "semantic interface" between CyPhyML and the domain specific modeling languages (DSML) (Figure 2) is formally defined, evolved as needed, and verified for essential properties (such as well-formedness and consistency) using the methods and tools of formal metamodeling [4][7]. By design, CyPhyML is moving in the opposite direction to unified system design languages, such

as SysML or AADL. Its goal is specificity as opposed to generality, and heavy weight standardization is replaced by layered language architecture and specification of explicit semantics.

3.2 Semantic Backplane of Open META

The "cost" of introducing a dynamic model integration language is that a mathematically precise formal semantics for model integration had to be developed. The OpenMETA Semantic Backplane [4][22][23] is at the center of our semantic integration concept. The key idea is to define the structural [5] and behavioral semantics [8] of the CyPhy model integration language using formal metamodeling, and use a tool supported formal framework for updating the CyPhy metamodels and verifying its overall consistency and completeness as the modeling languages are evolving. The selected tool for formal metamodeling is FORMULA[5] from Microsoft Research [10]. FORMULA's algebraic data types (ADTs) and constraint logic programming (CLP) based semantics is rich enough for defining mathematically modeling domains, transformations across domains, as well as constraints over domains and transformations. In the followings we provide a brief summary of the basic elements of the mathematical framework used in the Semantic Backplane.

Structural semantics of modeling languages represents the domain of well-formed models [5]. Domains are modeled as a term algebra whose function symbols characterize the key sets and relations through uninterpreted functions. A syntactic instance of some structural semantics is a finite set of terms over its term algebra $TY(\Sigma)$, where Σ is an infinite alphabet of constants, Y is a finite set of n-ary function symbols (signature) standing for uninterpreted functions, and the algebra is inductively defined as the set of all terms that can be constructed from Σ and Y. A syntactic instance of some structural semantics is a finite set of terms over its term algebra $TY(\Sigma)$.The set of all syntactic instances is then the power set of its term algebra: $P(TY(\Sigma))$.

To avoid the many unintended instances of the syntax of a modeling language, FORMULA enriches term algebra semantics with types by reconstructing Σ as the union of smaller alphabets and alphabets are ordered by set inclusion. Structural semantics often contain complex conformance rules; these rules cannot be captured by simple-type systems. One common solution to this problem is to provide an additional constraint language for expressing syntactic rules such as the Object Constraint Language (OCL). Unlike other approaches, FORMULA choose Constraint Logic Programming (CLP) to represent syntactic constraints because it can extend term algebra semantics while supporting declarative rules and unlike purely algebraic specifications it provides a clear execution semantics for logic programs making it possible to specify model transformations in the same framework. FORMULA supports a class of logic programs with the following properties: (1) expressions may contain uninterpreted function symbols, (2) the semantics for negation is negation as finite failure, (3) all logic programs must be stratified and (4) supports fixpoint logic over theories [13][14].

[5] http://research.microsoft.com/formula

Expressiveness of the formal framework discussed up to this point is sufficient for formalizing structural semantics, but does not support yet the specification of behavioral semantics. Since multi-physical modeling of systems requires modeling languages for continuous (DAE, PDE), discrete and hybrid dynamics, semantics need to be defined both denotationally and operationally [4]. Fortunately, the key to formalization in both cases is the development of precise specification of model transformations. In addition, formal modeling of model transformation are fundamental in all design automation frameworks, because they are used pervasively in integrating tool chains. In the OpenMETA Semantic Backplane, model transformations are encoded as logic programs where data types distinguish the inputs and outputs of the transformation [10]. For example:

```
Filter = out.MetaNode(x) :- in.MetaNode(x).
```

The constructor in.MetaNode() stands for primitives at the input of the transformation. Similarly, out.MetaNode() stands for primitives on the output of the transformation. A transformation is executed by providing an interpretation for the input primitives, and then computing the output primitives according to the CLP semantics. Specifying model transformations in the same CLP framework has fundamental advantages in allowing reasoning over the fully integrated representation of the input and output domains (e.g. proving that selected invariants will hold before and after the transformation).

While ADTs and CLP are sufficient for defining complex modeling domains and transformations, consistency checking and constructive modeling (model finding) [10] require the generation of automatic proofs from formal specifications by solving CLP satisability problems. Satisfiability is different from checking satisfaction of goals that be solved by simply running a logic program. It is to determine if a CLP program can be extended by a finite set of facts so that a goal is satisfied [9][16]. It requires searching through (infinitely) many possible extensions, which we achieve by efficient forward symbolic execution. FORMULA achieves this by efficient forward symbolic execution of logic program into the state-of-the-art satisfiability modulo theories (SMT) solver Z3 [15]. As a result, specifications can include variables ranging over infinite domains and rich data types (partial models). The method is constructive; it returns extensions of the CLP program witnessing goal satisfaction. An interesting application of this capability is design-space exploration [12].

The Model Integration Framework of OpenMETA (Figure 1) currently includes a large suite of modeling languages and tools for multi-physics, multi-abstraction and multi-fidelity modeling such as OpenModelica, Dymola, Bond Graphs, Simulink/Stateflow, STEP, ESMOL and many others. The CyPhyML model integration language provides the integration across this heterogeneous modeling space and the FORMULA - based Semantic Backplane provides the semantic integration for all OpenMETA composition tools.

4 Tool Integration Framework

The OpenMETA Tool Integration Framework (see Figure 1) comprises a network of model transformations that compose models for individual tools (e.g. Modelica models from ChyPhyML design models and component models) and integrate model-based design flows. Model-transformations are used in the following roles:

1. *Packaging.* Models are translated into a different syntactic form without changing their semantics. For example, AVM Component Models and AVM Design Models are translated into standard Design Data Packages (Figure 1, .ACM and .ADM files) for consumption by a variety of design analysis, manufacturability analysis and repository tools.
2. *Composition.* Model- and component-based technologies are based on composing different design artifacts (such as DAE-s for representing lumped parameter dynamics as Modelica equations [23], input models for verification tools [25], CAD models of component assemblies [19], design space models [25], and many others) from appropriate models of components and component architectures.
3. *Virtual prototyping.* Several test and verification methods (such as Probabilistic Certificate of Correctness – PCC) require test benches that embed a virtual prototype of the designed system executing a mission scenario in some environment (as defined in the requirement documents). We found distributed, multi-model simulation platforms the most scalable solution for these tests. We selected the High Level Architecture (HLA) as the distributed simulation platform and integrated FMI Co-Simulation components with HLA [26].
4. *Analysis flow.* Parametric explorations of designs (PET), such as analyzing effects of structural parameters (e.g. length of vehicle) on vehicle performance, or deriving PCC for performance properties frequently require complex analysis flows that include a number of intermediate stages. Automating design space explorations require that Python files controlling the execution of these flows on the Multidisciplinary Design Analysis and Optimization (OpenMDAO[6]) platform (that we currently use in OpenMETA) are autogenerated from the test bench and parametric exploration models (Figure 1).

Continuous evolution of the OpenMETA design flow makes it essential that the modeling tool suite for ChyPhyML is metaprogramable [4][17][20][18] and all model transformations used in the Composition Framework are formally specified as part of the Semantic Backplane. We believe that the lack of these capabilities are significant contributors to the failure of numerous large-scale model and tool integration efforts, due to the fact that semantic errors are all but impossible to detect without formal models.

Advantages of the Semantic Backplane and the logic-based formal framework is particularly important in the specification of composition semantics for mixed, multi-physical and computation modeling. For physical interactions, we chose acausal, power flow oriented modeling (e.g. Modelica, Simscape or Bond Graph modeling languages). In this approach, safe modeling of multi-physics interactions require rich

[6] http://openmdao.org/

typing for expressing and enforcing connectivity constraints. Beyond these static constraints, the semantics of acausal physical interconnections are expressed using algebraic constraints over the effort and flow variables [3]. This leads to a formal composition semantics that is simply the merging of the DAE equations representing component behaviors with the interconnection constraints [4]. Since our logic-based formal framework is expressive enough for describing typing and variables ranging over infinite domains and rich data types), description of composition semantics denotationally is quite straightforward in FORMULA.

5 Execution Integration Platform

The OpenMETA model and tool integration technology needs an infrastructure for creating and executing complex analysis flows including heterogeneous tool components. Our Analysis and Execution Framework (Figure 1) includes a wide range of cloud-deployed services such as component model repositories, ontology driven search engine, collaboration mechanisms, and cloud-deployed tools for design space exploration and data analytics. Our VehicleForge platform developed for DARPA's AVM program is essentially a gateway to shared resources and integrated services, not all of which are collocated. A central aim was to provide users secure and centrally managed access to these resources without the responsibility to individually respond to their evolution. An essential aspect of VehicleForge is its "software-as-a-

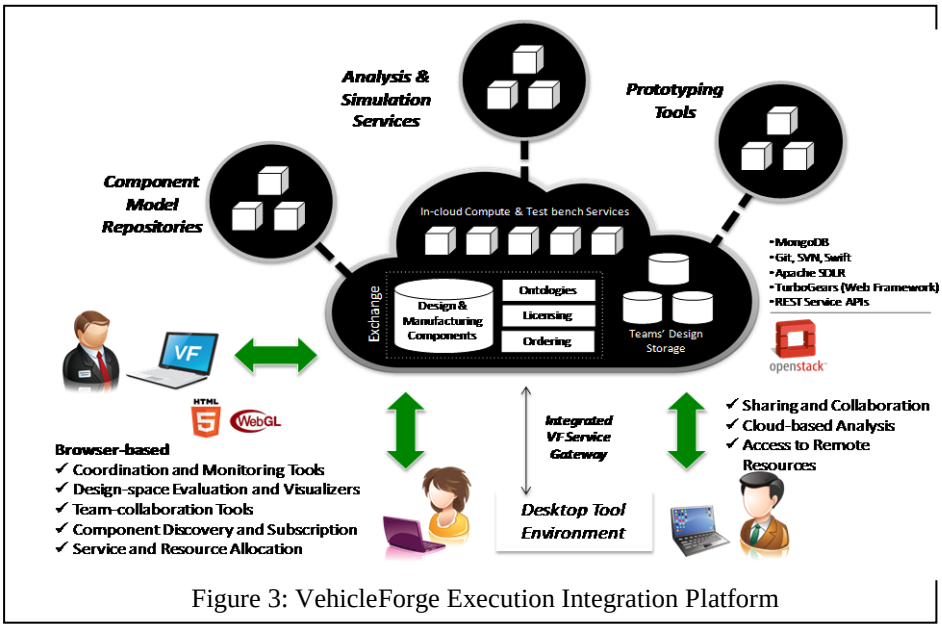

Figure 3: VehicleForge Execution Integration Platform

service" delivery model. It allows the low-cost access of end users (individuals, research groups, and larger companies) to repositories, analytic services and design tools, without the very high cost of acquiring and maintaining desktop engineering tools.

Two key aspects of sustainability are addressed by this platform. The first is resource elasticity and service staging to address scalability and system-wide optimization. This is a fundamental cloud computing rationale, which is predicated on the dynamism of demand for various forms of infrastructure over time. The second, perhaps more vital aspect, is managing the evolution of data, data representations, and their use by services and service integrations over time.

6 Lessons Learned

The first release of the OpenMETA tool suite capable of model-based compositional design, design-space exploration, multi-physics analysis and virtual performance testing was used during the FANG 1 Mobility/Drivetrain Challenge from January15 to April 15, 2013. During this competitive design event, over 1000 competitors organized into over 250 teams worked to design the drivetrain, suspension, propulsion elements and associated subsystems for FANG 1. The FANG component of the AVM program is currently building the winning design using the capabilities of the AVM program foundry, iFAB. Our team is currently expanding the OpenMETA tool suite with modeling and analysis capabilities required for hull design with strong focus on blast protection, structure, and fluid dynamics. This is preparation for the upcoming FANG 2 design challenge in February 2014.

Our work in the AVM program yields to two different kinds of results. First, we have created an end-to-end integrated tool suite, OpenMETA, that is now slated for transitioning to both the industry and the academic research communities. To ease transitioning and enable continued community-based development of OpenMETA, most of the tools integrated into the tool suite are open source with liberal BSD or MIT licensing. The second result is the insight we gained regarding promising directions and open problems in model- and component-based design. Below we list three essential points we have identified.

1. *Horizontal integration layers.* CPS companies face immense pressures to deliver safe and complex systems at low cost. End-to-end tooling for CPS industries is too heterogeneous and spans too many technical areas for any single tool vendor to fully support. In addition, CPS companies must develop, maintain and integrate in-house, proprietary tools to remain competitive. Consequently, integration of models and modeling languages in design flows, integration of tools into tool chains capable for the highly automated execution of design processes have emerged as a major challenge. Ad-hoc integration of models, tools and automated analysis threads is fragile, intractable, error prone and extremely costly. An essential insight of OpenMETA is that horizontal integration layers are fundamentally important in end-to-end CPS design flows. Their complexity requires the establishment of integration frameworks that provide the foundations and reusable, high-complexity components for rapid integration and evolution of CPS design tool chains.
2. *Component model repositories*. Component models capture reusable design knowledge, therefore model repositories play crucial role in improving design

productivity. AVM components contain a suite of modeling views and their interactions: static properties, structural, behavioral, geometric, cyber on multiple levels of fidelity. Selected abstractions of these modeling views are exposed via the components' semantic interfaces for composition. Building reusable, composable model libraries requires deep domain understanding and semantic rigor. Open research topics include hard problems such as formal relationship among modeling abstractions, establishing multiple fidelity levels, and understanding methods for representing and managing uncertainties.

3. *Goal-driven model composition.* In real-life CPS, model composition methods frequently lead to extremely large models. For example, composition of lumped parameter physical dynamics easily produces models including tens of thousands of equations with algebraic loops and non-linearities. A common problem is that simulation and verification tools do not scale to this complexity. The most promising research direction we identified is goal directed composition, that composes models according to the system property under study.

7 Acknowledgements

Authors are grateful for the advice and directions they received from Prof. Joseph Sifakis and Prof. Alberto Sangiovanni-Vincentelli, members of the Senior Strategy Group of the program. The OpenMETA and VehicleForge projects involve a large group at ISIS/Vanderbilt. Authors recognize the exceptional contributions of Zsolt Lattman, Adam Nagel, Jason Scott to OpenMETA. This research is supported by the Defense Advanced Research Project Agency (DARPA) under award # HR0011-12-C-0008 and the National Science Foundation under award # CNS-1035655.

References

1. Eremenko, Paul: "Philosophical Underpinnings of Adaptive Vehicle Make," DARPA-BAA-12-15. Appendix 1, December 5, 2011
2. Gossler, G., Sifakis, J.: "Composition for component-based modeling," Science of Computer Programming - Formal methods for components and objects pragmatic aspects and applications, Pages 161 – 183, Volume 55 Issue 1-3, March 2005
3. 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 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference IDETC/CIE 2012 August 12-15, 2012, Chicago, IL
4. 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 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference IDETC/CIE 2012 August 12-15, 2012, Chicago, IL-Sztipanovits J., Karsai G.: Model-Integrated Computing. In: IEEE Computer 30, 1997, pp. 110-112.

5. Jackson, E., Sztipanovits, J.: 'Formalizing the Structural Semantics of Domain-Specific Modeling Languages," Journal of Software and Systems Modeling pp. 451-478, September 2009
6. Sangiovanni-Vincentelli, A.: "Quo Vadis, SLD? Reasoning about the Trends and Challenges of System Level Design" Proc. of the IEEE, Vol. 95, No. 3. pp. 467-506, 2007
7. Jackson, E., Porter, J., Sztipanovits, J.: "Semantics of Domain Specific Modeling Languages" in P. Mosterman, G. Nicolescu: Model-Based Design of Heterogeneous Embedded Systems. pp. 437-486, CRC Press, November 24, 2009
8. Kai Chen, Janos Sztipanovits, Sandeep Neema: "Compositional Specification of Behavioral Semantics," in Design, Automation, and Test in Europe: The Most Influential Papers of 10 Years DATE, Rudy Lauwereins and Jan Madsen (Eds), Springer 2008.
9. E. K. Jackson and J. Sztipanovits, "Constructive Techniques for Meta- and Model-Level Reasoning," in Model Driven Engineering Languages and Systems, vol. 4735, G. Engels, B. Opdyke, D. C. Schmidt, and F. Weil, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 405–419
10. E. K. Jackson, T. Levendovszky, and D. Balasubramanian, "Reasoning about Metamodeling with Formal Specifications and Automatic Proofs," in Model Driven Engineering Languages and Systems, vol. 6981, J. Whittle, T. Clark, and T. Kühne, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 653–667
11. Alberto Sangiovanni-Vincentelli, Sandeep Shukla, Janos Sztipanovits, Guang Yang. "Metamodeling: An Emerging representation Paradigm for System-Level Design," IEEE Design and Test of Computers May/June 2009
12. Jackson, E., Simko, G., Sztipanovits, J.: "Diversely Enumerating System-Level Architectures," Proceedings of EMSOFT 2013, Embedded Systems Week, September 29-October 4, 2013 Montreal, CA
13. Dantsin, E., Eiter, T., Gottlob, G., Voronkov, A.: Complexity and expressive power of logic programming. ACM Comput. Surv. 33(3) (2001) 374-425
14. E. K. Jackson and W. Schulte, "Model Generation for Horn Logic with Stratified Negation," in Formal Techniques for Networked and Distributed Systems – FORTE 2008, vol. 5048, K. Suzuki, T. Higashino, K. Yasumoto, and K. El-Fakih, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 1–20.
15. de Moura, L.M., Bjoerner, N.: Z3: An efficient SMT solver. In: TACAS. (2008) 337-340
16. E. K. Jackson and J. Sztipanovits, "Constructive Techniques for Meta- and Model-Level Reasoning," in Model Driven Engineering Languages and Systems, vol. 4735, G. Engels, B. Opdyke, D. C. Schmidt, and F. Weil, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 405–419
17. G. Karsai, M. Maroti, A. Ledeczi, J. Gray, and J. Sztipanovits, "Composition and cloning in modeling and meta-modeling," IEEE Transactions on Control Systems Technology, vol. 12, no. 2, pp. 263– 278, Mar. 2004
18. Á. Lédeczi, Á. Bakay, M. Maróti, P. Völgyesi, G. Nordstrom, J. Sprinkle, and G. Karsai, "Composing Domain-Specific Design Environments," IEEE Computer, vol. 34, no. 11, pp. 44–51, 2001.
19. 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 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference IDETC/CIE 2012 August 12-15, 2012, Chicago, IL

20. G. Karsai, A. Ledeczi, S. Neema, and J. Sztipanovits. The model integrated computing tool suite: Metaprogrammable tools for embedded control system design. In Proceedings of the IEEE Joint Conference CCA, ISIC and CACSD, Munich, Germany, 2006
21. Janos Sztipanovits: "Cyber Physical Systems: Convergence of Physical and Information Sciences" Information Technology, pp. 257-265, 6/2012, Oldenbourg Wissenschaftsverlag GmbH
22. Simko, G., Levendovszky, T., Maroti, M., & Sztipanovits, J.: "Towards a Theory for Cyber-Physical Systems Modeling". Proc. 3rd Workshop on Design, Modeling and Evaluation of Cyber Physical Systems (CyPhy'13), April 08-11, 2013, Philadelphia, USA, pp. 1-6
23. Simko, G., Lindecker, D., Levendovszky, T., Neema, S., & Sztipanovits, J. Specification of Cyber-Physical Components with Formal Semantics–Integration and Composition. In Model-Driven Engineering Languages and Systems (pp. 471-487). Springer Berlin Heidelberg.
24. Emeka Eyisi, Zhenkai Zhang, Xenofon Koutsoukos, Joseph Porter, Gabor Karsai, and Janos Sztipanovits. "Model-Based Design and Integration of Cyber-Physical Systems: An Adaptive Cruise Control Case Studies", Journal of Control Science and Engineering, Special Issue on Embedded Model-Based Control. Volume 2013, Article ID 678016, 15 pages, 2013
25. Peter Fritzson, Zsolt Lattmann, Adrian Pop, Johan de Kleer, Bill Janssen, Sandeep Neema, Ted Bapty, Xenofon Koutsoukos, Matthew Klenk, Daniel Bobrow, Bhaskar Saha and Tolga Kurtoglu. "Verification and Design Exploration through Meta Tool Integration with OpenModelica" 10th International Modelica Conference 2014, Lund, Sweden March 10-12, 2014
26. Himanshu Neema, Jesse Gohl, Zsolt Lattmann, Janos Sztipanovits, Gabor Karsai, Sandeep Neema, Ted Bapty, John Batteh and Hubertus Tummescheit. "Model-Based Integration Platform for FMI Co-Simulation and Heterogeneous Simulations of Cyber-Physical Systems". 10th International Modelica Conference 2014, Lund, Sweden March 10-12, 2014