Design Tool Chain for Cyber-Physical Systems: Lessons Learned

Janos Sztipanovits
Vanderbilt University
Nashville, TN
+1-615-322-3455
janos.sztipanovits@vanderbilt.edu

Ted Bapty
Vanderbilt University
Nashville, TN
+1-615-322-3455
ted.bapty@vanderbilt.edu

Sandeep Neema
Vanderbilt University
Nashville, TN
+1-615-322-3455
sandeep.neema@vanderbilt.edu

Xenofon Koutsoukos
Vanderbilt University
Nashville, TN
+1-615-322-3455
xenofon.koutsoukos@vanderbilt.edu

Ethan Jackson
Microsoft Research
One Microsoft Way, Redmond, WA
98052 USA
jackson@msr.com

Invited

ABSTRACT

Design automation tools evolved to support the principle of "separation of concerns" to manage engineering complexity. Accordingly, we find tool suites that are vertically integrated with limited support (even intention) for horizontal integratability (i.e. integration across disciplinary boundaries). CPS challenges these established boundaries and with this - market conditions. The question is how to facilitate reorganization and create the foundation and technologies for composable CPS design tool chains that enables reuse of existing commercial and open source tools? In this paper we describe some of the lessons learned in the design and implementation of a design automation tool suite for complex cyber-physical systems (CPS) in the vehicle domain. The tool suite followed a model- and component-based design approach to match the significant increase in design productivity experienced in several narrowly focused homogeneous domains, such as signal processing, control and aspects of electronic design. The primary challenge in the undertaking was the tremendous heterogeneity of complex cyber-physical systems (CPS), where such as vehicles has not yet been achieved. This paper describes some of the challenges addressed and solution approaches to building a comprehensive design tool suite for complex CPS.

Categories and Subject Descriptors

Design, Verification, Cyber Physical Systems, Design Automation, System Description Langauages

Keywords

Cyber-physical systems, Model-based design. Component-based design

1. INTRODUCTION

In 2010, the Defense Advanced Research Project Agency (DARPA) initiated the Adaptive Vehicle Make (AVM) program to construct a fully integrated model- and component-based design flow for the "make" process of complex cyber-physical systems (CPS) [1]. Our team has lead the development of the integrated design automation tool suite, OpenMETA, with the goal of making significant progress in the following key areas:

1. Shortening development times for complex systems: This goal was translated into the following technical challenges: (a) composing designs from component model libraries, (b) raising the level of abstraction in the design of CPS, (c) enabling correct-by-construction design methods, and (d) executing rapid requirements trade-offs.
2. Shifting product value chain toward high-value design activities: Adoption of component- and model-based design laid the foundation for establishing a well-defined interface between design and manufacturing. Moving this interface toward design by incorporating manufacturing awareness into the design flows has been expected to have profound impact on the product value chain by enabling new business models and new capabilities, such as (a) foundry-like manufacturing capability for defense systems, (b) rapid switch-over between designs with minimal learning curve, and "mass customization" across product variants and families.
3. Democratizing design: The appearance of open platforms and open source tools has the potential for dramatically expanding the scope of players in the CPS innovation processes and disrupting old business models. We approached thi schallenge by (a) building OpenMETA on a dominantly open-sourced tool infrastructure to enable open-source development of cyber-electromechanical systems, (b) testing the OpenMETA tools for experimenting with prize-based Adaptive Make Challenges to involve non-traditional players in the make process, and (c) motivating a new generation of designers and manufacturing innovators by initiating student competition challenges.

The fundamental barriers in the OpenMETA project design flow and its supporting tool suite have been (a) heterogeneity and semantic ambiguity of models and tools that span the design and manufacturing space, (b) the lack of integration technology for models and tools available in isolated stove pipes, and (c) the lack of availability of delivery platforms that break down the cost and complexity of using a complex, integrated tool suite. In this paper we discuss some of the key findings and related accomplishments of the project focusing primarily on key architectural aspects of the design tool chain.

2. STATUS QUO

Large system companies face immense pressures to deliver safe and complex systems at low cost. Tools are at the heart of their engineering process covering the full spectrum of requirements, design, manufacturing and operations support. The internal tool landscapes of large aerospace and automotive companies contain ~5000 distinct tools totaling several hundreds of millions of dollars in internal investments. End-to-end tooling for these complex CPS product lines is very heterogeneous and spans too many technical areas for single tool vendors to fully cover. In addition, a significant part of the companies' design flow is supported by in-house tools that are proprietary and capture high value design IP. In many areas, such as powertrain electronics in the automotive industry, production tool suites include a combination of in-house and COTS tools in the approximate ratio of 70% and 30%, respectively. The development and use of in-house tools is not necessarily the result of deficient COTS offerings, but, rather, it is an essential part of the innovation process that yields competitive advantage via improved product quality and productivity. The primary technology barrier that slows down this process and makes integration of in-house tools with 3rd party tools extremely expensive and error prone is the lack of modern tool integration and deployment platforms.

Seamless integration of end-to-end tool chains for highly automated execution of design flows is a complex task of which successful examples are rare – even after massive investment by companies. Vendors provide limited integration, primarily of their own tools, with a few cross-vendor integrations for particularly dominant tools (e.g., integration with DOORS, Word or Excel). This limitation results in design flows that consist of islands of integrated tool sub-chains, bridged by various ad-hoc, semi-automated, or manual stopgaps. These stopgaps impose a variety of costs: additional work in performing manual transformations, additional work in guarding against divergence between multiple representations, and forgone analysis opportunities, to name just a few.

Truly transformational impact requires an approach for composing an end-to-end integrated tool chain from a heterogeneous collection of COTS, open source, and proprietary tools. The ideal solution would support tools from multiple vendors, and allow companies themselves to include the most closely guarded of proprietary tools. Such a truly integrated toolset would yield significant improvements in productivity and decreases in design time by eliminating the unnecessary work associated with the existing integration mechanisms and shortening the learning curves associated with diverse, un-integrated tool suites.

3. NEED FOR INTEGRATION PLATFORMS

While model-based design has a proven track record and strong acceptance in many focused areas of engineering (such as VLSI design, control system design) the heterogeneity of CPS technologies and application domains combined with the need for achieving correct-by-construction design create new technical barriers for its wider use. The most widely used strategy to deal with heterogeneity in the design process is separation of concerns. 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 enabling 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 well if the design concerns were orthogonal, but in tightly coupled CPS this is rarely the case. The price of the simplification is decreased predictability of properties of the implemented CPS and costly re-design cycles.

We believe that the single most important change to achieve correct-by-construction design is the introduction and systematic use of cross-domain modeling. However, creating design tool chains that cover all potentially relevant CPS modeling abstractions and satisfy the needs of all application domains is unrealistic. In addition, tool chains that are highly configurable to specific application domains are not available. Consequently, our objective was to develop horizontal integration platforms that allow the rapid construction of domain-specific, end-to-end tool suites for CPS design.

3.1 OpenMETA Horizontal Integration Platforms

OpenMETA complemented the traditional, vertically structured and isolated model-based tool suites with horizontal integration platforms for models, tools, and executions. The horizontal integration platforms allow combining the separation of concerns strategy with cross-domain modeling whenever cross-domain interdependences cannot be neglected. Our focus in the project was not restricted to OpenMETA as configured for a specific ground platform design, but was extended to the OpenMETA Integration Platforms for models, tools and executions such that they can be used both for experimenting with different design flows and for creating highly domain specific design tool chains.

These integration platforms are the following [5]:

1. *Model integration platform* supported by generic OpenMETA tools for creating and using semantically rigorous model integration languages, metaprogrammable modeling tools, metamodel repositories and the OpenMETA Semantic Backplane including formal specification of the model integration language ChyPhyML and all model transformations.
2. Tool integration platform with generic tools for the precise specification, verification and generation of model transformations – a widely used technology in the OpenMETA tool chain. The tool integration

platform also includes the specification of design flows (Experiment Specifications) composed from predefined design threads and vignettes.

3. Execution integration platform to provide an affordable, web-based delivery platform of integrated design tools, enabling their cloud-based deployment through a software-as-a service delivery model. The platform includes job manager for distributing simulation and verification jobs across cloud resources.

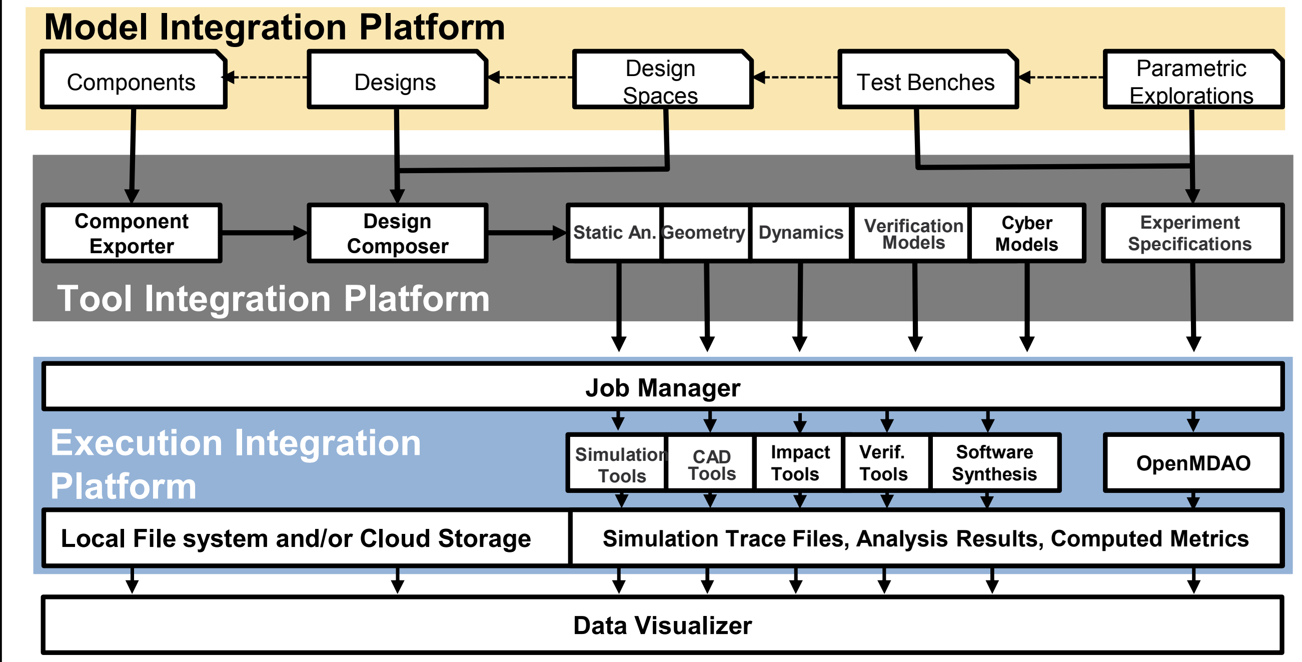

Figure 1: OpenMETA Horizontal Integration Platforms

The OpenMETA design tool chain has been delivered as a fully configured integrated tool suite with model libraries and has been tested in DARPA's Fast Adaptable Next-Generation Ground Vehicle (FANG1) Mobility/Drivetrain Challenge (2013) and Hull Design Challenge (2014) (called Gamma Test). During the FANG1 challenge, the tool suite was stress tested in a national design competition for the power train of an amphibious vehicle that included over 1000 participants in more than 200 design teams in 2013 [11].

3.2 Semantic Integration

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 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.

In a naïve approach, model and tool integration is considered to be an interoperability issue that can be taken care of with appropriate syntactic 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 "cost" of introducing a dynamic model integration language is that mathematically precise formal semantics for model integration had to be developed.

The OpenMETA Semantic Backplane [5][7] is at the center of our semantic integration concept. The key idea is to define the semantics of the CyPhyML model integration language using formal metamodeling, and to use a tool supported formal framework for updating the CyPhyML metamodels [8] and verifying its overall consistency and completeness as the modeling languages are evolving. The selected tool for formal metamodeling is FORMULA from Microsoft Research [9][12]. FORMULA's algebraic data types (ADTs) and constraint logic programming (CLP) based semantics are rich enough for mathematically defining modeling domains, transformations across domains, as well as constraints over domains and transformations. At the conclusion of the project, the META Semantic Backplane includes the formal specification of CyPhyML, the semantic interfaces to all constituent modeling languages, and all model transformations used in the tool integration framework.

The Semantic Backplane is a new approach to construct complex component- and model-based design tool chains. It is an essential tool for those who design and evolve domain specific tool chains and responsible for the overall integrity of the model and tool configurations used in the design process. Its importance was proven in the following use cases:

1. As in all areas of engineering, mathematical modeling helped designing and evolving modeling languages,

composition semantics and model transformations. It was invaluable in finding and correcting inconsistencies, identifying incompleteness problems, and fixing errors in the semantic foundations of the tool chain.
2. The Formula-based executable specifications were used for generating reference traces and served as abstract prototypes for constraint checkers and transformations used throughout the tool chain.
3. The CyPhyML Reference Manual was auto-generated from the formal specifications.

While most of the activities in the use cases above are manual at this point, creating tighter link between the specification and the production tools and increased automation such as autogeneration of transformation code from formal specification is feasible.

4. COMPONENT MODELING TECHNOLOGY

The appeal of component-based design is the potentially massive productivity increase due to the reuse of design knowledge captured by the component models. Their significance was recognized early in the AVM program and was noted as the main contributor to achieving the 5X decrease in design time. In a component- and model-based design flow, system models are composed of component models guided by architecture specifications. To achieve correct-by-construction design, the system models are expected to be heterogeneous multi-physics, multi-abstraction and multi-fidelity models that also capture cross-domain interactions. Accordingly, the component models, in order to be useful, need to satisfy the following generic requirements:

1. Elaborating and adopting established, mathematically sound principles for compositionality. Composition frameworks are strongly different in physical dynamics, structure and computing that needs to precisely defined and integrated.

2. Inclusion of a suite of domain models (e.g., structural, multi-physics lumped parameter dynamics, distributed parameter dynamics, manufacturability), on an established number of fidelity levels with explicitly represented cross-domain interactions.

3. Precisely defined component interfaces required for heterogeneous composition. The interfaces need to be decoupled from the modeling languages used for capturing domain models. This decoupling ensures independence from the modeling tools selected by the component model developers.

4. Established bounds for composability expressed in terms of operating regimes where the component model remains valid.

These requirements are widely accepted in all engineering design where component-based approaches are used. A common misconception in physical system modeling is that useful models need to be hand-crafted for specific phenomena. One explanation for this is the quite frequent use of modeling approaches that do not support compositionality. The AVM Component Model places strong emphasis on compositional semantics that can resolve this problem.

A harder problem is that automated composition from multi-phenomenon component models can easily produce very large model sizes, if incorrectly used. A promising solution is to adapt the selected level of abstraction, level of component fidelity and the suite of physical phenomena to the examined system property. While we applied this approach in OpenMETA, there are still open challenges to be addressed in the future, such as goal directed composition making the composition process adaptive to the property computed by test benches. In addition, a well-known method for controlling design complexity is to adjust the level of granularity for components, and use more abstract models for larger, more complex components such as engines, transmissions or microprocessors. There are excellent examples for existing and emerging component libraries, both in crowdsourced or COTS form: DOE's EnergyPlus is an open-source model and simulation library for building energy analysis, the Modelica Standard Library (MSL) is a crowdsourced , multi-physics lumped parameter dynamics library developed and maintained by the OpenModelica Consortium, Modelon's Vehicle Dynamics Library is a COTS component library on the top of the Modelica Standard Library, and many others. We believe that domain specific model libraries will continue emerging both in open-source and COTS form and will become one of the engines in the progress of component and model-based design.

The META project developed a standard AVM Component Model that provides a framework for integrating multi-domain and multi-language structural, behavioral and manufacturing models and provides the composition interfaces for the OpenMETA tools. In constructing an AVM Component Model from domain models, (such as from Modelica models representing lumped parameter dynamics) the interfaces, connectors, and parameters must be extracted from the domain models, and mapped to the interface abstractions used in the AVM Component Model. This process can be time-consuming and error prone. In order to improve productivity, the META program has developed a full tool suite for importing domain models (such as Modelica dynamic models), integrating them with standard AVM Component Model Interfaces, automatically checking compliance with the standard, and automatically checking model properties, such as restrictions on the types of domain models, wellformedness rules, executability, and others. Based on our direct experience, the automated model curation process resulted in orders-of-magnitude reduction in required user effort for building AVM Component Model libraries.

5. AUTOMATION IN EXECUTING DESIGN FLOWS

As shown in Figure 2, CPS design in META is divided into the following main phases:

1. Architecture design using combinatorial design space exploration using static, finite-domain constraints and architecture evaluation.

2. Integrated multi-physics/cyber design using quantitative, lumped parameter hybrid dynamic models, and incorporating both deterministic and probabilistic approaches.

3. Detailed design including geometric/structural design space exploration using physics based, nonlinear PDE analysis of thermal, mechanical and mobility properties

The design space exploration phases require the composition of system models using model libraries, the analysis of the models against design requirements and the performance of a multi-objective optimization process combined with probabilistic and deterministic verification methods (Accomplishment 6). The META design flow needs to manage heterogeneity in multiple dimensions, such as physical phenomena, levels of abstraction used in modeling physical and computational structures and processes, and engineering disciplines involved in CPS design. If we combine this challenge with the need for exploring large design spaces, it is clear that without full automation of the exploration process, the overall META vision would not be achievable.

5.1 Design-Space Exploration Using Progressive Refinement

Automated exploration of a heterogeneous CPS design space is not only semantically complex, but is also computationally expensive. Quality of the resulting design depends on the size of the explored space, which is determined by the number of architectural variants, the number of parameters, and the parametric ranges. However, executing the exploration process with the highest fidelity detailed design models is computationally prohibitive.

One of the key enablers for automating the design space exploration process is the automated adjustment of the level of abstraction of the composed system models starting with static models and combinatorial exploration of very large design spaces, progressing for lumped parameter dynamic models using different levels of fidelity, and finally performing first principle-based deep analysis for only a few candidate designs.

There are several exploration strategies that can be built in the OpenMETA design flow. We currently implemented the *progressive refinement strategy* that starts with a seed design (a single design point), around which the design space can be carefully defined by designers using architectural alternatives and parametrization.

The automated design space exploration process was stress tested in the FANG 1 Powertrain Challenge [6]. During the 3 month competition period, design teams submitted for remote evaluation 51,424 candidate models and each received evaluation scores against a set of system requirements.

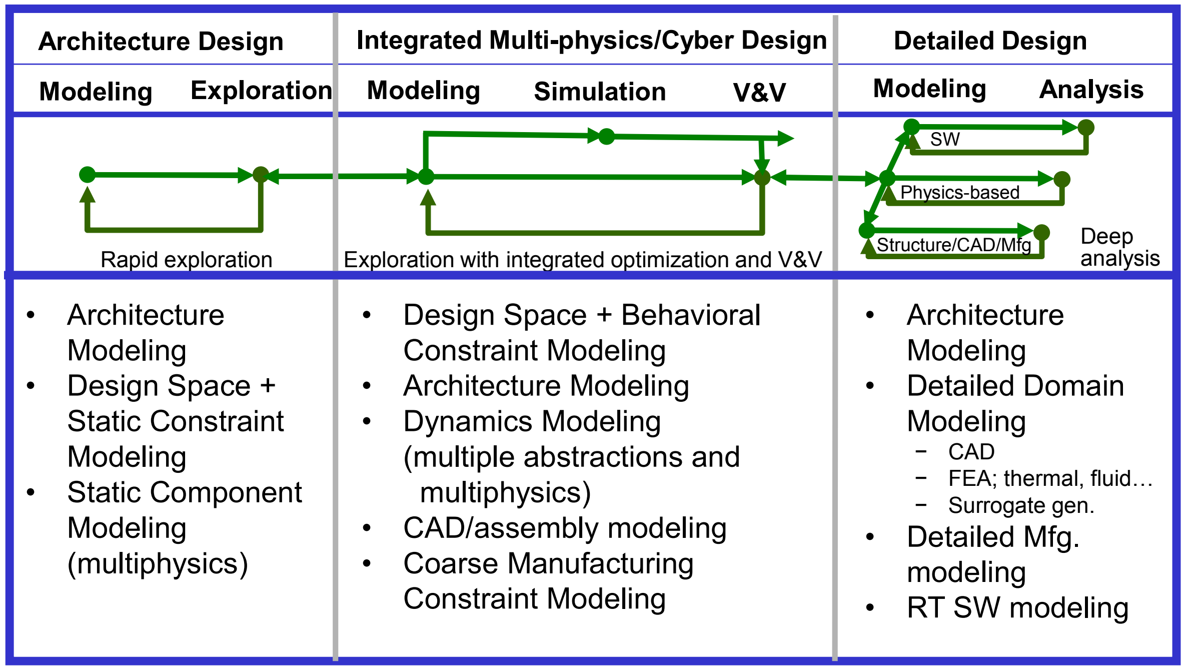
Architecture Design
Modeling
Exploration
Integrated Multi-physics/Cyber Design
Modeling
Simulation
V&V
Detailed Design
Modeling
Analysis
Rapid exploration
Exploration with integrated optimization and V&V
SW
Physics-based
Structure/CAD/Mfg
Deep analysis
• Architecture Modeling
• Design Space + Static Constraint Modeling
• Static Component Modeling (multiphysics)
• Design Space + Behavioral Constraint Modeling
• Architecture Modeling
• Dynamics Modeling (multiple abstractions and multiphysics)
• CAD/assembly modeling
• Coarse Manufacturing Constraint Modeling
• Architecture Modeling
• Detailed Domain Modeling
− CAD
− FEA; thermal, fluid…
− Surrogate gen.
• Detailed Mfg. modeling
• RT SW modeling

Figure 2: Notional Design Flow in Open META

5.2 Automated Analysis Using Virtual Test Benches

Another enabler for automating the design space exploration process is the fully automated evaluation of points in the design space against the full set of system requirements. The key OpenMETA innovation for this is the introduction of Virtual Test Benches that are the executable versions of the requirements. Each Test Bench is linked to the specification of a design space and used for evaluating all encoded requirements across all generated design point samples. While executing Test Benches during the exploration process, the design space continually evolves to include only satisfying designs. Test Benches are also modeled using a modeling language that defines analysis tool setup, parameters, context models, metrics, and post processing scripts.

5.3 Collaboration Platforms and Access to Integrated Open- Source Tools

The accepted model of open source software development is peer production by collaboration, with the end-products – source code, "blueprints", and documentation; available at no cost to the public. Key enablers for the tremendous success and economic impact of open-source software are the collaboration platform technologies emerging from the tradition of software forges, such as SourceForge, GoogleCode, and GitHub. These platforms combine project hosting, web-based collaboration, and centralized version control system repositories. The open source software communities were transformed by services that support collaboration within teams by coordinating the work of geographically dispersed developers, and between the teams and their user communities, by providing web-based tools including documentation wikis, issue tracking systems and discussion forums and shared code repositories. Open- and crowd-sourced tools have long tradition due to their long development and maturation time, and significant public investment in their creation. Today, open source repositories include 666,998 projects, 674,380 source control repositories, 30,879, 289,910 lines of code produced by 3,627,589 contributors worldwide.

The open source movement is now spreading across different fields. While open-source frameworks, platforms and tools were isolated and highly specialized in the past, appearance of new, domain-specific open platforms (such as ROS, Eclipse, Open Source Ecology) and the related viable business models extended from software industry to robotics and to physical equipment design and manufacturing are becoming more popular and form well established part of the innovation infrastructure .

There are two essential infrastructure need for democratizing design for complex CPS. The first is the adoption of the collaboration platform concept that facilitates the formation of geographically dispersed designer teams, providing easy access to shared computation resources and design repositories. The second is availability of open source (or integratable COTS) tools that can be deployed in a software-as-a-service (SaaS) distribution model.

The first need is addressed by the VehicleForge project of DARPA's AVM program executed by another research team at the Institute for Software Integrated Systems at Vanderbilt. While VehicleForge was not part of the META project, the Vanderbilt teams worked closely together and integrated OpenMETA with the VehicleForge collaboration platform – a key technical condition for the FANG1 design competition.

The second need led us to explore the availability and quality of open source tools essential for model-based CPS design. Many of the open source tools, such as NASA's OpenMDAO, Sandia National Lab's DAKOTA or OpenModelica are directly relevant to the goals of META and of high quality. We have found that 70-80% of the OpenMETA tool suite functionalities can be covered using high quality open source tools.

6. SUMMARY

The OpenMETA project of the AVM program addressed very hard problems, which are at the epicenter of component- and model-based design: (a) composing designs from reusable component model libraries, (b) extending the limits of correct-by-construction design, (c) raising the level of abstraction in design of CPS, (d) executing rapid requirements trade-offs, (e) restructuring the interface between design and manufacturing for CPS, and (f) creating an open framework for reusing open-source tool assets. The project gave OpenMETA developers unique opportunity not only to understand the limits of the current state-of-the-art in the context of a real-life DoD challenge problem.

7. ACKNOWLEDGMENTS

This work was supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. HR0011-13-C-0041. DARPA's AVM program provided exceptional opportunity for testing the current state-of-the-art and pushing the limits of component- and model-based design in a real-life DoD challenge problem.

8. REFERENCES

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

[2] Sztipanovits, J., T. Bapty, S. Neema, X. Koutsoukos, and J. Scott, The META Toolchain: Accomplishments and Open Challenges, no. ISIS-15-102, 2015. (download: http://www.isis.vanderbilt.edu/publications)

[3] Lattmann, Z., J. Klingler, P. Meijer, T. Bapty, S. Neema, and J. Scott, Integration Platform Technology Components in the META Toolchain, , no. ISIS-15-110, Nashville, Institute for Software Integrated Systems, 01/2015

[4] Sztipanovits, J., Bapty, T., Neema, S., Howard, L., Jackson, E. (2014). OpenMETA: A Model and Component-Based Design Tool Chain for Cyber-Physical Systems. In Bensalem, S., Lakhneck, Y., Legay, A. (Eds.) From Programs to Systems – The Systems Perspective in Computing, LNCS 8415 (pp.235-249), 2014

[5] 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

[6] Neema, H., Z. Lattmann, P. Meijer, J. Klingler, S. Neema, T. Bapty, J. Sztipanovits, and G. Karsai: "Design Space Exploration and Manipulation for Cyber Physical Systems," IFIP First International Workshop on Design Space Exploration of Cyber-Physical Systems (IDEAL' 2014), Springer-Verlag Berlin Heidelberg 2014.

[7] Simko, G., Lindecker, D., Levendovszky, T., Neema, S., & Sztipanovits, J. (2013). Specification of Cyber-Physical Components with Formal Semantics–Integration and Composition. In Model-Driven Engineering Languages and Systems (pp. 471-487). Springer Berlin Heidelberg.

[8] 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

[9] 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

[10] Jackson, E., Sztipanovits, J.: 'Formalizing the Structural Semantics of Domain-Specific Modeling Languages," Journal of Software and Systems Modeling pp. 451-478, September 2009

[11] O. L. de Weck, "Feasibility of a 5× speedup in system development due to meta design," in *32nd ASME Computers and Information in Engineering Conference*, Aug. 2012, pp. 1105–1110

[12] 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. Springer Berlin Heidelberg, pp. 1–2