2005 American Control Conference
June 8-10, 2005. Portland, OR, USA

Towards a Concurrent Engine System Design Methodology

Akira Ohata, *Toyota Motor Corporation*, Kenneth R. Butts, *Toyota Technical Center, U.S.A.*

***Abstract*— Today's hyper-competitive automotive marketplace places very stringent quality and productivity demands on the industry's manufacturers and thus dictates an investment in new engineering processes and methods. In this paper, an engine design and development process based on Toyota's process improvement vision will be presented. The process requires the development and deployment of a Model-based Concurrent Engine System Design Methodology. Model based engineering has the potential to enable engineers to investigate design alternatives, system sensitivities, and component integration compatibility prior to building and testing physical prototypes. These benefits could be further magnified by engineering concurrency between the various teams involved in engine system development. Tools and methods needed to implement the vision are discussed.**

I. Introduction

Today's hyper-competitive automotive marketplace places very stringent quality and productivity demands on the industry's manufacturers. In this context, Toyota Motor Corporation (TMC) continuously improves its business operation by driving for innovation and evolution [1]. As argued in [2], the exponential complexity growth of emerging engine systems dictates an investment in new engineering processes and methods. The authors state that Toyota plans to develop improved gasoline, electric-gasoline-hybrid, and electric-fuel-cell-hybrid powerplants to address exhaust emission and fuel resource challenges in the global marketplace. They argue that successful application of these technologies requires robust and high-performance control systems. Thus it becomes crucial to employ efficient development processes that are more inter-disciplinary and concurrent than today's methods. Figure 1 [2] shows a representation of such a process.

The purpose of this paper is to provide a starting point for evaluating and developing Toyota's vision for improved engine system design and development. Essentially, this vision is to develop and deploy a Model-based Concurrent Engine System Design Methodology. Model-based engineering has the potential to enable engineers to investigate design alternatives, system sensitivities, and component integration compatibility prior to building and testing physical prototypes. We believe these benefits are further magnified by engineering concurrency between the various teams involved in engine system development.

The engine system under consideration includes the base-engine, intake, exhaust, emissions after-treatment, and electronic control (including sensor, actuator, and communications) systems. Thus, as shown in Figure 2, we wish to promote concurrency between the engine hardware, control logic, embedded computing architecture, and tuning / optimization design domains. Figure 2 also shows the information dependencies between design domains. It is important to observe that each of the domains is dependent upon information generated by the others, thus there is a strong need for concurrency in the over-all design methodology.

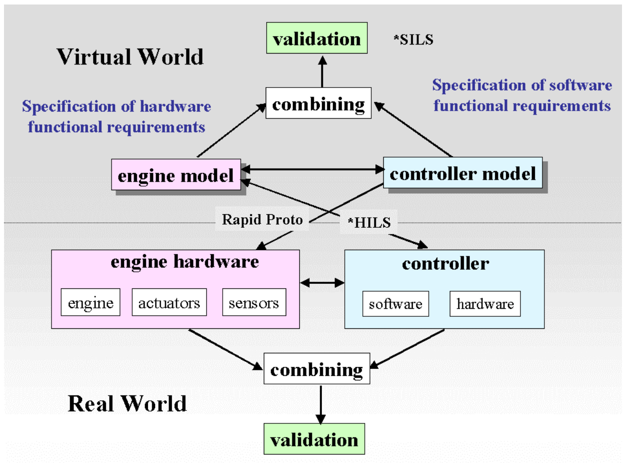

Fig. 1. Abstract Model-based Engine System Development Process

The remainder of the paper includes a more detailed presentation of the concurrency needs for each design domain (Section II – Design Domains) and an elaboration of usage scenarios for the new design methodology (Section III – Use Cases for the Concurrent Engine System Design Methodology.) Given that we are still in the planning phase for this work, Section II focuses on related work that should be considered during methodology development.

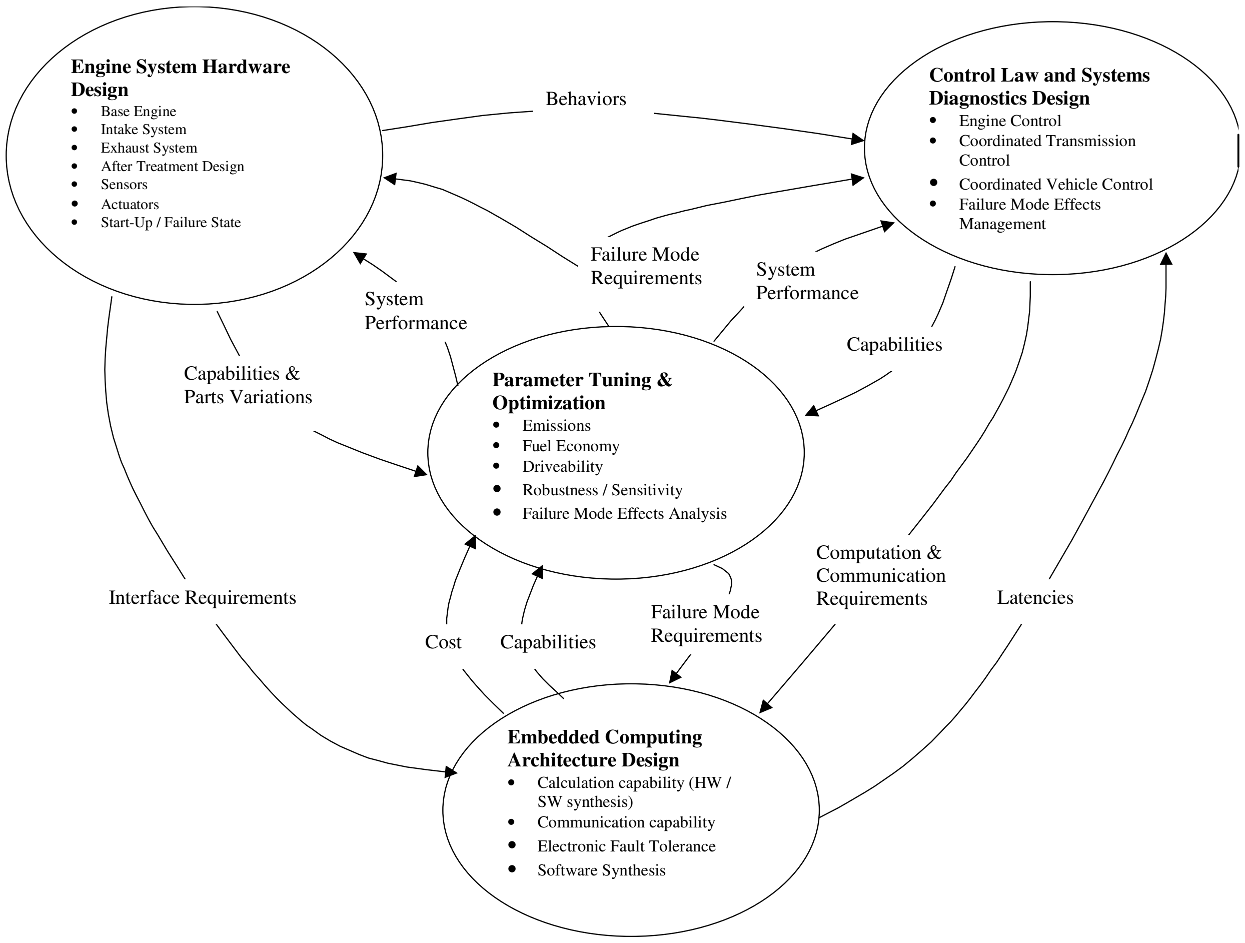
Engine System Hardware Design
Base Engine
Intake System
Exhaust System
After Treatment Design
Sensors
Actuators
Start-Up / Failure State
Behaviors
Control Law and Systems Diagnostics Design
Engine Control
Coordinated Transmission Control
Coordinated Vehicle Control
Failure Mode Effects Management
Failure Mode Requirements
System Performance
System Performance
Capabilities & Parts Variations
Capabilities
Parameter Tuning & Optimization
Emissions
Fuel Economy
Driveability
Robustness / Sensitivity
Failure Mode Effects Analysis
Interface Requirements
Cost
Capabilities
Failure Mode Requirements
Computation & Communication Requirements
Latencies
Embedded Computing Architecture Design
Calculation capability (HW / SW synthesis)
Communication capability
Electronic Fault Tolerance
Software Synthesis

Fig. 2. Scope of Concurrent Engine System Design

II. Design Domains

Because our vision for concurrent engine system design is so broad, there is much to be considered and the related work is bountiful. A relevant overview paper on modeling and simulation [3] focuses on technologies that support the design of engineering systems. The paper identifies several on-going research challenges:

- Modeling at the component level.
  - Component interaction modeling.
  - Selecting an adequate level of detail.
  - Improvement of numerical solvers.
- Integration with design tools.
  - CAD.
  - Finite-element modeling.
  - Optimization and synthesis tools.
- Collaborative modeling.
  - Unified model representation.
  - Ontologies for modeling and simulation.

An emerging and closely related field of study is called "Computer Automated Multi-Paradigm Modeling." (CAMPaM) [4]. It introduces collaborative modeling concepts in three dimensions:

- Levels of abstraction
- Multi-formalisms (transformation between modeling formalisms)
- Model meta-modeling (modeling to describe model formalisms)

The following discussion introduces activities that expound on the items listed above. The presentation is far from exhaustive and we plan to refine it as the design methodology matures. Our immediate task is to weave the salient aspects of these activities into a coherent plan for the concurrent engine system design environment and promote technology development where necessary.

A. *Systems Design Domain*

The goal in the system design domain is to provide the infrastructure and architectures that transcend and integrate the engine system's hardware, control systems, and electronics and software architecture design domains.

Product-Line Development Methods: The Software Engineering Institute has documented effective development and business methods that are critical to organizations employing software product-line principles. "A software product line is a set of software-intensive systems that share a common, managed set of features satisfying the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way." [5] We believe these architecture-based practices are largely applicable to model-based engine system development as described in this paper.

Systems Architecture Analysis: One technique to promote rapid and efficient modeling is to formulate systems architectures. The system architecture serves as the reference specification for component and assembly modeling that is reusable across the product line. Ford Motor Company (FMC) has published system-modeling architectures for control systems and hardware systems development at the vehicle level [6], [7]. Similar concepts could be applied to promote rapid modeling of engine systems.

Systems Modeling Standards (SysML): The systems engineering community, in association with INCOSE, is developing a modeling standard called Systems Modeling Language. This standard operates at an abstract structural level that may be useful for systems architecture modeling and analysis. This work [8] is set in the context of the OMG's Unified Modeling Language [9].

Model Management: FMC has piloted a commercial product-life-cycle-management tool as the foundation for its model management system. This system adheres to the principles of product-line development because it uses an architecture centric data model that allows a reference design to be the basis for tailored product instances. The data model is also designed so that the reference and associated instances are easily maintained as they evolve over time. Please see [10] for more detail. [11] and [12] discuss model management in a more general setting.

Systems Optimization: Dynamic programming can be an effective method to leverage system models for parameter optimization and feasibility assessment. FMC has used the dynamic programming technique to assess and optimize the performance of engine after-treatment systems. Please see, for example, [13].

Alternatively, rules based and direct approaches have been used successfully at TMC [14] and FMC [15] respectively.

B. Hardware Design Domains

As presented in [11], the strong relationship between design and validation can be illustrated as in Figure 3. A design's form, behavior, and function typically progress iteratively from concept to detailed realization. Thus, it is critical that we improve the knowledge transfer from computer-aided design tools to behavioral modeling and simulation environments to improve development productivity. In [3], the authors note that this capability exits for single-domain design environments. While the trend is to expand this capability towards more general, multi-domain settings, we believe there remains a rich opportunity to exploit the idea in our engine system design context.

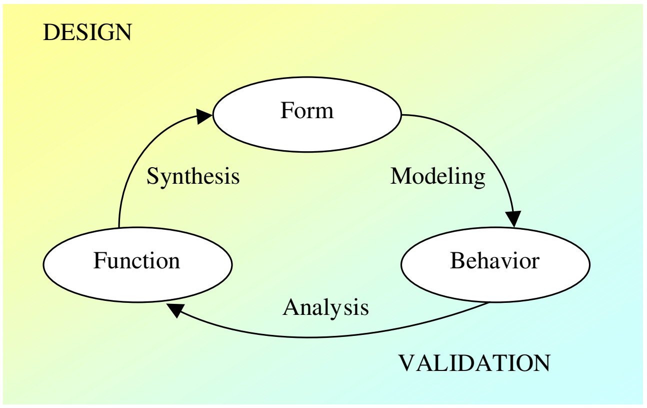

Fig. 3. Relationship between Form, Behavior, and Function

Physical Modeling: The choice of modeling language is important here. Modern declarative languages admit a modeling method based on component composition where the model components have a strong mapping to the physical system components. Model components constructed in this way are more re-usable (i.e. high quality product-line assets) because their causality is defined by their system usage, not by their component definition. [3] TMC has had a long interest in declarative modeling in support of engine hardware and systems development [16]. Much of this work has been done in the Modelica [17] modeling language context. Modelica is declarative and object-oriented and thus well suited for component-based modeling of complex systems.

C. Control Systems Domain

Computer-Aided Control System Design (CACSD) tools are generally well accepted within the automotive control design community and the mapping of these tools onto the prototypical systems engineering 'V' process model is well established [18]. Specific design methodologies include Model Based Control Design, Rapid Controller Prototyping, and Hardware-In-the-Loop Verification. In the following, we present additional control system design domain topics that are particularly relevant related to concurrent engine system design.

MAAB Controller Guidelines: The MathWorks' tool suite (Matlab®[1], Simulink®, Stateflow® …) is one of the primary tool suites used in automotive control systems development. Consequently, the MathWorks Automotive Advisory Board (MAAB) was formed to serve as a discussion forum for automotive control system design tool needs. An important product of MAAB is an industry guideline [19] that helps to ensure controller models are prepared in a common way. This common preparation results in efficiencies for the automobile manufacturer and the control system electronic control unit suppliers.

Automatic Code Generation: Controller models serve as executable specifications to the embedded software engineers. In fact, it is now possible to use automatic code generation technology to rapidly develop a large portion of the engine control application software and engine control system suppliers are reporting promising automatic code generation results. For example, Visteon say that they can use data dictionary overlays to target multiple computing platforms from a common engine control specification [20]. This work demonstrates how model-based development can be used in a product-line context. Moreover, Visteon's metrics show that automatically generated software can be as efficient (in terms of RAM and ROM size) as manually generated software.

Automated Testing: The CACSD domain now has automated tools that help validate a control system specification and verify its software realization from either a requirements or a development-regression perspective. Moreover, it is possible to use automate data collection experiments on engine test-benches to improve control system calibration productivity. A discussion of infrastructure for integrated model-based control system development and automated testing can be found in [21].

Model-based Calibration: The automotive industry has developed efficient statistical modeling techniques to characterize static engine input-output relationships. These relationships can be analyzed to set optimal control settings for each engine operating condition. A commercial analysis suite called the Model-Based Calibration Toolbox codifies many of these techniques [22]. Some representative application references include [23], [24], [25], and [26].

[1] Matlab, Simulink, and Stateflow are Registered Trademarks of The MathWorks, Inc, Natick, MA.

D. Electronics Architecture and Embedded Software Domains

Architecture Analysis & Design Language (AADL): The architecture theme is also important for electronic computing platform design and development. For example, the aerospace partition of the Society of Automotive Engineers is working with the Software Engineering Institute to standardize an architecture description language called Architecture Analysis & Design Language. In contrast to SysML, AADL is designed to describe and analyze the embedded computing aspects of complex systems. More specifically, "it is used to design and analyze the software and hardware architecture of embedded real-time systems and properties that are critical to the operation of such a system such as timing, throughput, reliability. … The vision of the SAE AADL is for avionics, aerospace, automotive, and robotics application developers in DoD and in industry to use predictive model-based software system engineering practices, resulting in a major reduction in systemic errors leading to performance and dependability failures." [27]

AUTOSAR: The Automotive Open System Architecture (AUTOSAR) [28] is an industry partnership to establish an open standard for automotive electronic control unit software architecture. In particular, the partnership aims to standardize 1) functional interfaces across manufacturers and suppliers and 2) the interfaces between the different software layers to promote modularity, scalability, transferability and re-usability of software-based functionality. Modularity of automotive software elements will enable tailoring of software according to the individual requirements of electronic control units and their tasks. Scalability of functions will ensure the adaptability of common software modules to different vehicle platforms to prohibit proliferation of software with similar functionality. Transferability of functions will optimize the use of resources available throughout a vehicle's electronic architecture. Re-usability of functions will help to improve product quality and reliability and to reinforce corporate brand image across product lines.

Systems Design: Metropolis [29] is a design and development tool initiative from the Center for Hybrid and Embedded Software Systems (CHESS has four generally relevant research thrusts: Hybrid Systems Theory, Model-based design, Advanced tool architectures, and Experimental applications including vehicle electronics [30].) Metropolis contains simulation and analysis environments aimed at supporting embedded system level design from conception to implementation. One automotive top-to-bottom design methodology in Metropolis is called fault tolerant data flow.

Another CHESS initiative is called CHIC – Checker for Interface Compatibility. CHIC [31] is a formal verifier that checks compatibility between software and hardware component models. The fundamental goal is to ensure that components satisfy the assumptions they make about each other and therefore can be reliably composed to create larger systems.

MoBIES: The United States Defense Advance Research Program Agency (DARPA) recently completed the Model-based Integration of Embedded Software project [32]. This project was focused on tools and methods for improving embedded software development. As such, much of the MoBIES work is relevant to engine system development. In particular, the DESERT model compiler tools [33] from Vanderbilt University's ISIS lab use constraint and signal connectivity analysis to assess whether a set of model components is compatible from a systems perspective. Given a compatible component set, the tool can automatically construct a large-scale and operational model of the system assembly.

Another important focus for MoBIES was model-based system verification. Thus Carnegie-Mellon University, The Software Engineering Institute, Emmeskay, Inc., and The MathWorks, Inc. collaborated to develop the System Verification Manager (SVM) [34]. SVM provides requirements to model architecture traceability, change-driven verification activity management, and verification result management and access.

Formal Methods: In [2] the authors remark that with current methods for engine system validation and verification "a vast number of data and tests are necessary to guarantee sufficient reliability." Thus they call for continued validation and verification methods research and development to improve productivity.

Fortunately, some practical, semi-formal tools targeted at control-logic and software validation and verification are now commercially available. For example, [21] shows how tools can dramatically improve the productivity of model-based development. Validation tools can be used to automatically check that design models satisfy high-level system performance assertions (i.e. system requirements). And the improved productivity enabled by automatic test case generation allows aerospace-quality test suites to be used in automotive software verification. Early results of formal methods application to automotive control logic specifications are presented in [35].

III. Use Cases for the Concurrent Engine System Design Methodology

Model-based work products have utility throughout the product-line life cycle. They help systems architects evaluate structural alternatives and cascade requirements from system to subsystem. They can document product-line reference architectures for re-use and optimization. Models often serve as rigorous specification and validation criteria for component designers. Moreover, validation test results can be re-used for component and system level verification.

In this section we describe several specific engineering processes and tasks that could be improved with the application of the model-based Concurrent Engine System Design Methodology.

Early assessment of new engine technologies and actuation systems: Engine hardware designers have traditionally used static operating point analysis to assess the benefit of new engine technologies and actuation systems. The results of this analysis are often optimistic because systems must be detuned to satisfy transient and dynamic behavioral requirements. Concurrent hardware design, control design, and optimization, facilitated by rapid hardware modeling, will allow technology assessment based on dynamic operating scenarios.

Virtual engines and vehicles for desktop calibration: Engine control calibration is a very time consuming process that relies on the availability of physical hardware. These characteristics are becoming more critical as additional control dimensions are introduced by new technology. Predictive dynamic models that exhibit the appropriate levels of accuracy and computational cost would reduce the reliance on costly physical prototypes and allow the calibration process to start at an earlier development stage [36]. As mentioned above, additional benefits will be realized because system performance feedback can be provided to the hardware design teams at design time.

Early vehicle system integration studies: While the focus of this design environment is to improve the engine system design, the work products could be used to facilitate improved systems interaction at the vehicle level. The interaction of the engine system with transmission, braking, vehicle dynamics, active safety, and energy management control systems is critical to the delivery of today's sophisticated vehicle features that include vehicle stability control, adaptive cruise control, fuel usage minimization, and regenerative braking.

Targeted (Optimized) Electronics Architecture Design: The trend in the embedded electronics industry is to provide "systems-on-chips" [37] that serve as highly optimized computational platforms for the task at hand. This optimization is achieved even though the design cycles are relatively short: typically less than one year. In contrast, the automotive industry tends to re-use standard computational platforms from vehicle project to vehicle project with a long electronics platform design cycle. Thus the computational platform acts as a major constraint on the control logic design that often results in compromised system performance or late engineering changes. The concurrent engine system design methodology, used in conjunction with rapid-electronics systems synthesis from the embedded electronics industry, should result in more optimized engine systems.

Automated Software Synthesis: Given control law specification models, it is now (or soon will be) possible to use automatic code generation to synthesize production quality software realizations of the control law [2]. In addition to the inherit benefits of high quality and improved productivity, this rapidly developed software 1) facilitates early verification of the real-time aspects computational platform and 2) can be used as a computationally efficient control logic model for large-scale engine system simulation and optimization.

Knowledge Capture: Model-based development processes produce rigorous and reusable development assets in the form of models, executable specifications, and verification procedures. These assets serve as the base design reference for the corporation's vehicle product line. Given proper care in the implementation of the environment's data repository, these design references can be re-used by a globally distributed development team and customized as needed for a particular product application. This re-use represents a *kaizen* mechanism for on-going productivity and quality improvements. Two specific examples are the capability to rapidly perform regression verification on minor changes and to train new engineers in the model-based concurrent engine system design methodology.

Use Case Considerations: The model-based concurrent engine system design methods should account for the following considerations:

- Engine systems must be developed to meet the needs of the application marketplace. There may be reason to perform local market place optimizations to meet those needs.
- Engine systems are developed in global and cross-organizational teams.
- Engine systems development spans research, advanced development, and series development or any subset thereof. In other words, changes may be large and systematic or they may be minor and incremental.
- Engine systems quality and verification are of critical importance.
- Engine systems must be designed for manufacturability.

IV. Conclusion

This paper presents a formative vision of a model-based and concurrent engineering method that is intended to improve automotive engine system development. The realization of this vision will require a rational synthesis of the elements described. Hopefully, this paper will spur discussion and effort in that regard.

References

[1] J. Liker, The Toyota Way: 14 Management Principles From The World's Greatest Manufacturer, McGraw-Hill, New York, 2004.

[2] T. Ueda, A. Ohata, "Trends of Future Powertrain Development and the Evolution of Powertrain Control Systems," in *Proc. 2004 SAE – Convergence Conference,* October, 2004.

[3] R. Sinha, V-C Liang, C. J. J. Paredis, P. K. Khosla, "Modeling and Simulation Methods for Design of Engineering Systems," JCISE, 2001.

[4] P. J. Mosterman, H. Vangheluwe, "Computer Automated Multi-Paradigm Modeling: An Introduction," *Simulation: Transactions of The Society for Modeling and Simulation International,* vol. 80, no. 9, pp. 433 – 450, September, 2004.

[5] The Software Engineering Institute, The US Department of Defense, and Carnegie-Mellon University, http://www.sei.cmu.edu/plp/plp_init.html.

[6] M. Jennings, et al, "A Vehicle Model Architecture for Vehicle System Control Design," 2003-01-0092, *Proc. SAE 2003 World Congress*, 2003.

[7] M. Tiller et al, "Development of a Vehicle Model Architecture in Modelica," *Proc. Modelica Workshop 2003,* http://www.modelica.org/Conference2003/papers/h32_vehicle_Tiller.pdf, 2003.

[8] Systems Modeling Language, www.sysml.org/.

[9] Object Management Group, http://www.uml.org/.

[10] K. Butts, et al, "A Model Repository for Automotive Embedded Control Systems Design," DETC/CIE-48237, *Proc. DETC'03, ASME 2003 Design Engineering Technical Conferences and Computers and Information in Engineering Conference, 2003.*

[11] C. J. J. Paredis, A. Diaz-Calderon, R. Sinha, P. K. Khosla, "Composable Models for Simulation-Based Design," Technical Report ICES-04-21-00, Institute for Complex Engineered Systems, Carnegie Mellon University, Pittsburgh, PA, 2000.

[12] B. Johansson, **Model Management for Computational System Design**, Dissertation No. 857, Linköping Studies in Science and Technology, Institute of Technology, Linköping University, Sweden, 2003.

[13] Y. Kim et al, "Optimization of Lean NOx Trap Control for Fuel Economy and Exhaust Emissions," *Proc. 2004 American Control Conference,* 2004.

[14] T. Fukuma, et al, "Challenging the Vision of Calibration-Free Diesel Engine Development by Means of Model-based Control and Automatic Optimization," *25th Internationales Wiener Motorensymposium*, Vienna, Austria, April 2004.

[15] J. Sun, et al, "Issues in Cold Start Emission Control for Automotive IC Engines," *Proc. American Control Conference*, Philadelphia PA, June 1998.

[16] S. Soejima, "Examples of usage and spread of Dymola within Toyota," *Proc. Modelica Workshop 2000,* http://www.modelica.org/workshop2000/proceedings/Soejima.pdf, 2000.

[17] Modelica Association, http://www.modelica.org/.

[18] dSpace, Inc., http://www.dspaceinc.com/ww/en/inc/systems.html.

[19] The MathWorks, Inc., “Controller Style Guidelines for Production Intent Development Using MATLAB, Simulink, and Stateflow,” Version 1.00, April 2001.
[20] G. Hodge et al, “Multi-Target Modelling for Embedded Software Development for Automotive Applications,” 2004-01-0269, *Proc. 2004 SAE World Congress*, 2004.
[21] Reactive Systems, “Model-Based Testing and Validation of Control Software with Reactis®,” November, 2003, http://www.reactive-systems.com/.
[22] The MathWorks, Inc., Model-Based Calibration Toolbox, http://www.mathworks.com/products/mbc/.
[23] T. Holliday, et al, “Engine-Mapping Experiments: A Two-Stage Regression Approach,” TECHNOMETRICS, May 1998, vol. 40. no 2, American Statistical Association and the American Society for Quality, 1998.
[24] D. W. Rose, et al, “An engine mapping case study – a two-stage regression approach,” ImechE Paper C606/025/2002, Statistics & Analytical Methods in Automotive Engineering, London, September, 2002.
[25] K. Ropke, editor, Design of Experiments (DoE) in der Motorenentwicklung, ISBN 3-8169-2271-6, Expert Verlag, 2003.
[26] K. Plischke, “Bringing MATLAB® Into the Test Cell, SAE 2003-01-1025, *Proc. 2003 SAE World Congress*, 2003.
[27] Society for Automotive Engineers, Architecture Analysis & Design Language, http://aadl.info/.
[28] Automotive Open System Architecture, http://www.autosar.org/.
[29] Gigascale Systems Research Center, http://www.gigascale.org/metropolis/.
[30] Center for Hybrid and Embedded Software Systems, http://chess.eecs.berkeley.edu/.
[31] Checker for Interface Compatibility, CHIC, http://www-cad.eecs.berkeley.edu/~tah/chic/.
[32] DARPA Model-based Integration of Embedded Software project, http://dtsn.darpa.mil/ixo/programdetail.asp?progid=38.
[33] S. Neema, et al, “Constraint-Based Design Space Exploration and Model Synthesis, EMSOFT 2003, Lecture Notes in Computer Science 2855, Springer-Verlag, 2003.
[34] System Verification Manager, http://www.ece.cmu.edu/~webk/svm/.
[35] S. Sims, R. Cleaveland, K. Butts, and S. Ranville, “Automated validation of software models,” *Proc. 16th IEEE International Conference on Automated Software Engineering, ASE'01*. IEEE, 2001.
[36] T. M. Morton, et al, “Model-based Optimal Calibration of a Dual Independent Variable Valve-Timing Engine,” Kartsen Ropke, editor, Design of Experiments (DoE) in der Motorenentwicklung, ISBN 3-8169-2271-6, expert verlag, 2003.
[37] F. Winters, et al, “Design Process Changes Enabling Rapid Development,” *Proc. 2004 SAE Convergence Conference*, October, 2004.