

Ford Motor Company

General Motors Corporation

Motorola Automotive and Industrial Electronics Group

SmartVehicle Challenge Problems

In support of the University California Open
Experimental Platform for DARPA – MoBIES,
Contract F33615-00-C-1698.

SmartVehicle Challenge Problems

Overview

This report presents challenge problems that are designed to assess the effectiveness of new tools and methods for embedded software development. These problems are consistent with the SmartVehicle Open Experimental Platform (OEP) provided by the University of California Berkeley. The current efforts to use model-based methods in the automotive industry motivate the definition of these problems.

Various aspects of improved model-based embedded software development are to be demonstrated via these challenge problems. Figure 1 shows where MoBIES technologies (as shown in the green ovals) might be integrated with the typical automotive development process.

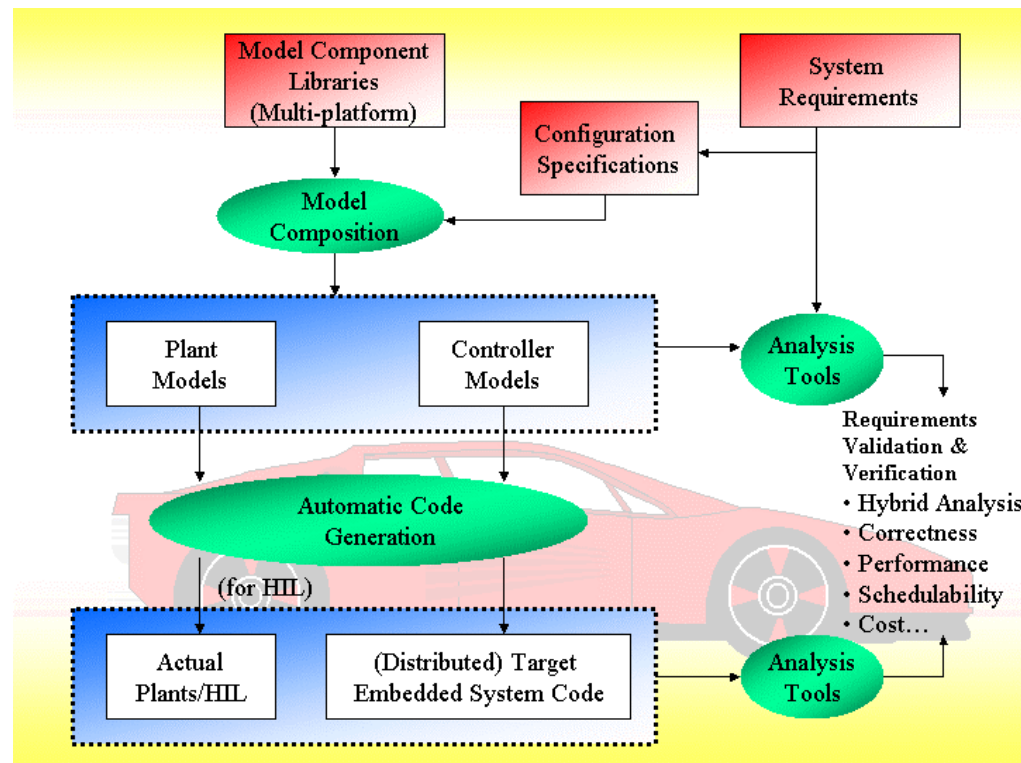


Figure 1 - Automotive Development

Typical automotive development processes are further described in [Partitioned Control](#).

The MoBIES challenge problems are presented in The MathWorks¹ tool environment due to the tool suite's popularity in the automotive industry. This environment provides hybrid

¹ The MathWorks, Inc., 3 Apple Hill Drive, Natick, MA 01760-2098, Release 12, Version 6

systems modeling and simulation, control system analysis and synthesis, controller prototyping, automatic code generation, mode-to-code verification via simulation, and Hardware-in-the-Loop electronic control module verification. MoBIES tool technologies will be evaluated relative to The MathWorks tools. Specific challenge problems are defined in the following areas:

- 1) Multiple-view modeling
- 2) Composition of component models into consistent system models
- 3) Dynamic analysis of hybrid system models for design validation
- 4) Embedded software analysis and generation
- 5) Allocation of system function and performance to distributed computing environments

In the following, we provide a general motivation for each challenge problem and couple them with specific demonstration requirements to be applied to the SmartVehicle OEP. We use references to additional documents to mitigate copyright and contractual issues. Table 1 is a tabulation of these references that is intended to help the reader manage the complexity of this structure.

	<i>Composition of component models</i>	<i>Dynamic analysis of hybrid system models</i>	<i>Embedded software analysis and generation</i>	<i>Allocation of system function and performance to distributed computing environments</i>	<i>Multiple-view modeling</i>
<i>General Motivation</i>	Model Composition Simple Compiler	[Butts] Transmission model	Embedded Software	Partitioned Control Vehicle-to-Vehicle Control	Ford Style Guide [Magner]
<i>Specific Challenge</i>	Model Composition Challenge	In this document			

Table 1 - Document Map

Multiple-view modeling

General Motivation

The engineering resources required to realize a coordinated ground vehicle control system are vast and inter-disciplinary. While model-based development processes have realized improvements in time-to-market and quality, modeling style [\[Ford Style Guide\]](#)

compromises [\[Magner\]](#) have been made to achieve a "single-model" development process. Unfortunately, each engineering discipline has had to modify their natural view of the system to accommodate the single-view of this single-model approach. It is desirable to have a modeling environment that retains the single-model philosophy (to provide efficiency and consistency) while providing multiple, domain-specific, views of the single system model.

The MoBIES community has defined a three level model taxonomy. Our challenge problems use Level 2 and Level 3 models.

Level 1 – continuous time, no regard for controller sampling or signal naming. Anything goes.

Level 2 – conforming to style guides with rigorous signal naming and characterization. Model building blocks are restricted to a subset to admit analysis. Controller models are restricted to discrete-time/discrete-event building blocks and have timing information included.

Level 3– Level 2 model augmented with software design information to admit automatic code synthesis.

We believe multiple-view modeling is closely related to meta-modeling for the purposes of model-information exchange between tools because many engineers prefer to view the system via their domain-specific and familiar tools. Of course, such inter-tool exchange also facilitates domain-specific model analysis.

Application to the SmartVehicle OEP

Challenges related to multiple view modeling are expressed in [Partitioned Control](#) and [Vehicle-to-Vehicle Control](#)

Composition of component models into consistent system models

General Motivation

A great deal of effort has been directed towards creating simulation models of both physical automotive components and embedded controllers. Yet, these models are rarely reused beyond the original applications for which they were developed. This is usually because one has to make manual modifications to interface the old model with the new application. For example, one may have to surgically remove the needed part of an old model, adjust the wirings, create or delete some inputs and outputs, and perform some unit conversions. This could be especially difficult if one is not familiar with the original component model. We envision a modeling environment that facilitates the archiving of reusable components and that facilitates the automatic composition of these components. The development of such an environment will greatly reduce the duplication of modeling effort and further enhance productivity when new applications are being developed. It will also enhance the engineers' productivity as they can focus on system level design while taking advantage of the existing model base.

Refer to [Model Composition](#) and [Simple Compiler](#) to see a prototype of a model composition environment.

Application to the SmartVehicle OEP

Some initial thoughts on composability rules for automotive systems modeling are presented in [Model Composition Challenge](#).

Dynamic analysis of hybrid system models for design validation

General Motivation

A *hybrid system* is a system that contains both continuous- and discrete-valued states. Such systems are ubiquitous in automotive applications. Embedded controllers often contain discrete-valued states describing the mode of operation and the automotive systems being controlled often contain physical states that are continuous-valued. Today the validation of such systems entails extensive simulation and testing. The ability to analyze models of hybrid systems would greatly enhance the automotive industry's embedded systems design and development processes.

Refer to [Butts](#) to see how hybrid systems analysis could be applied to a typical automatic transmission shift control problem.

Application to the SmartVehicle OEP

1. Given Nominal powertrain model configuration, constant road grade [0 – 10%], time-varying throttle [0 – 100%], and time-varying brake [on – off], show that `shift_schedule` never reaches a terminal state. (A liveness query)
2. Given Nominal powertrain model configuration, constant throttle [0 – 100%], and constant brake [off], show that there does not exist a constant road grade [0 – 10%] that causes a shift cycle (e.g. 3-2-3, or 2-1-2) within five seconds. (A mode transition integrity query)
3. Given Nominal powertrain model configuration, show that `Pc2_command` always satisfies some smoothness property. (A mode transition integrity query)
4. Given Nominal powertrain model configuration, `gear = [1 2]`, show that `Tc1 >= Tt`. (A design requirement validation; Butts presented as region of operation query)
5. Given Nominal powertrain model configuration, show that `| gear – to_gear |` is always `<= 1`. (A design requirement validation)
6. Given Nominal powertrain model configuration, show that at $t=0^+$, `gear = 1` and `to_gear = 1`. (An initialization design requirement validation)
7. Given Nominal powertrain model configuration, show that is possible to achieve the following state trajectories: **1)** `gear = 1; gear = 2; gear = 3; gear = 4; gear = 3; gear = 2; gear = 1` **2)** `gear = 1; gear = 2; gear = 1; gear = 2; gear = 3; gear = 4; gear = 3; gear = 2; gear = 1` **3)** `gear = 1; gear = 2; gear = 3; gear = 2; gear = 3; gear = 4; gear = 3; gear = 2; gear = 1` **4)** `gear = 1; gear = 2; gear = 3; gear = 4; gear = 3; gear = 2; gear = 1`. (A state trajectory query)
8. Given Nominal powertrain model configuration, constant road grade [0], constant throttle [10%], constant brake [off], and 12-inertia phase mode (PID active),

determine the gain and phase margin of the closed-loop system. (A dynamic systems analysis query.)

9. Given Nominal powertrain model configuration, show that 12-inertia phase mode states in the transmission model and the controller model are consistent within 50 ms. Note the transmission mode should lead in while the controller mode should lead out. (A design integrity question). Show that this property holds when the C2 coefficient of friction is constant within a range $[.5 * \text{nominal } 1.5 * \text{nominal}]$. (A parameter sensitivity query)
10. Given Nominal powertrain model configuration, show that W_e is always < 6000 RPM. (A design requirement validation)
11. Given Nominal powertrain model configuration, check the shift_scheduler and shift_control subsystems for unreachable states. (A software efficiency question via dead code elimination.)
12. Given Nominal powertrain model configuration, while ignoring the Stateflow graphical "clockwise rule" semantics check the shift_schedule and pressure_control state-machines for non-determinisms in the transition logic. (A specification clarity query to eliminate the potential for hidden behaviors.)

Embedded software analysis and generation

General Motivation

Computer Aided Control System Design (CACSD) tools help the control engineer to design the functional behavior of today's automotive embedded systems. However, there remains a significant amount of potential automation in the embedded software area that would greatly increase the productivity of embedded software development. Three sub-categories will be emphasized in the challenge problems. The first category, automatic code generation, has received much attention for several years, and commercial solutions are either available or nearly available. The main challenges left in this technology are to apply it to specific CACSD tools, increase the code efficiency, and increase the flexibility of the generated code. The second category is scheduling of the code. Tools exist today to help developers, but additional tools are needed. In particular a tool to automatically generate test inputs to stimulate the minimum, maximum, and "nominal" execution-time-paths to provide timing data for scheduling analysis. The third category is unit testing/model checking. A tool is needed that will automatically check the design model for inconsistencies, and to automatically generate test vectors for software verification testing.

Refer to [Embedded Software](#) for a detailed enumeration of embedded software related challenges.

Allocation of system function and performance to distributed computing environments

General Motivation

Ground vehicle electrical architectures are trending toward distributed controller realizations due to 1) constraints such as complexity management, thermal packaging, wire-harness packaging, and supplier business relationships and 2) increasing amounts of coordinated system control. Tools and methods are needed to allocate model-based functional design requirements to the distributed controller architecture. The capability of the electronic architecture (e.g. computational power, memory space, communication protocols, network bandwidth) must be budgeted relative to the functional requirements while preserving functional behavior and performance. Automatic messaging protocol synthesis of the resulting inter-function communication is essential.

Two levels of distributed control challenges are further described in [Partitioned Control](#) and [Vehicle-to-Vehicle Control](#).

References

[Butts] Butts, Kenneth, " Analysis Needs for Automotive Powertrain Control ," Proceedings of The 7th Mechatronics Forum International Conference, September 6 – 8 2000, Atlanta, GA

[Magner] Magner, Steve, Butts, Kenneth, Toeppe, Steve, "Multiple Model Views in Mechatronic Control and Strategy Simulations," Proceedings of The 7th Mechatronics Forum International Conference, September 6 – 8 2000, Atlanta, GA