ABSTRACT

Model-based control system design improves quality, shortens development time, lowers engineering cost, and reduces rework. Evaluating a control system's performance, functionality, and robustness in a simulation environment avoids the time and expense of developing hardware and software for each design iteration. Simulating the performance of a design can be straightforward (though sometimes tedious, depending on the complexity of the system being developed) with mathematical models for the hardware components of the system (plant models) and control algorithms for embedded controllers. This paper describes a software tool and a methodology that not only allows a complete system simulation to be performed early in the product design cycle, but also greatly facilitates the construction of the model by automatically connecting the components and subsystems that comprise it. A key element of this technique is the software-in-the-loop (SIL) capability, which permits compiled production controller code to be incorporated into the simulation environment, thus allowing the inclusion of algorithm functionality for which no simulation models exist. With this approach, the control system can be developed early in the vehicle or powertrain design cycle, incorporating plant models, algorithm models, existing controller code, and architectural constructs that greatly expedite the creation of a system simulation that can be used for algorithm development, testing, and validation. An application of this methodology at General Motors Powertrain is described in detail.

INTRODUCTION

MODEL-BASED DESIGN

Model-based control system design has been widely accepted in the automotive industry. Designing and studying the performance of a control system in the virtual environment of a computer system simulation:

• Provides a math-based environment for thorough multidisciplinary integration and testing prior to hardware builds.

• Permits sorting technologies quickly to reduce hardware build iterations.

• Promotes parallel and integrated virtual development of control systems and hardware.

• Delivers better-integrated, initial designs that balance fuel economy, emissions, and drivability (FEED) requirements.

• Provides common methods and tools for comparing/evaluating technologies.

• Facilitates efficient, seamless linking from research to production to maximize reuse of work products and eliminate waste.

The control system development process is represented in Figure 1.

System Simulation

In the system design phase, plant models are created that represent the dynamic behavior of the system, to evaluate its
performance under a proposed control strategy. The plant models are then integrated with models of the controller hardware and algorithm software to provide a complete system simulation environment in which to perform the control design study. Traditional methods of developing control algorithms do not usually include interactions with other subsystems and components of the vehicle or powertrain.

A systems engineering approach calls for the entire system to be modeled, including plant, sensor/actuators, controller hardware, and algorithm/application software, as shown in Figure 2.

In many cases, although plant models are already available or can be readily developed, the control algorithm models are not all available, and the only representation of the algorithm functionality is contained in existing software. In these cases a full system simulation can be performed only when a complete software build has been created and loaded into the target embedded controller. Then the control algorithm software can be tested in a hardware-in-the-loop (HIL) system, where the controller and its application software is connected to a real-time computer system that runs a simulation of the plant model(s) representing the physical system with which the controller will eventually be used.

This paper describes a methodology and set of tools that permit the above restriction to be circumvented. A system
Simulation can be constructed even when models of all the control algorithms are not available, thus yielding what may be referred to as a virtual HIL simulation.

Challenges
Developing a complete system simulation of an automotive application can be a challenging task. Complex plant models, models of sensors and actuators, algorithm models, and production software modules must be connected. Some of the subsystem blocks have hundreds of inputs and outputs, which makes manually connecting them a time-consuming and error-prone process. To obtain a complete system model the functionalities of all the control algorithms must be included—but in many cases models of all the algorithms are not available.

For a system simulation to be effective and structured for the task at hand, users must be able to readily select the level of fidelity for each of the subsystem models. There also must be a mechanism to manage all of the artifacts of the system simulation, including the models themselves, initialization and calibration files to be used with the models, test cases and drive cycles used to exercise the system simulation, and results files. These artifacts also need to be configuration managed and version controlled.

SOLUTION
The challenges listed above were solved with a combination of the Autonomie software developed at Argonne National Laboratory and a unique software-in-the-loop (SIL) capability created at GM Powertrain Engineering.

Autonomie
Autonomie was developed under a Cooperative Research and Development Agreement (CRADA) between the U.S. Department of Energy, General Motors (GM), and Argonne National Laboratory. It is available free of charge to all members of the FreedomCAR and Fuel partnership, which includes Chrysler, Ford, GM, five energy companies, and two electric utilities.

Autonomie supports a plug-and-play modeling architecture that allows the user to select the appropriate level of fidelity for each model in the system. Autonomie also automatically interconnects the subsystem models that comprise the total system, and then builds the Simulink model needed to run the simulation. For the system simulation required to support control system design, an SIL block is included that incorporates a compiled version of the production software used in existing embedded controllers. The example given in this paper demonstrates the method for an engine control module. Figure 3 shows the SIL-based system simulation in the Autonomie environment.

Figure 4 shows the controller model structure and includes:
- A block for the real-time operating system (RTOS) that schedules the execution of the tasks in the software and algorithm models. This RTOS block consists of a behavioral
model of the actual RTOS used in the embedded controller, and is implemented as a Simulink model.

- A Controller Area Network (CAN) communication block to supply serial data signals to the controller software
- An algorithm model (AM) block that contains the AMs under development
- An hardware I/O (HWIO) block that contains behavioral models of the HWIO functions in the controller
- A software-in-the-loop (SIL) block that permits the inclusion of compiled production code into the simulation model. An SIL block is required for each controller included in the system, thus allowing for multiple ECUs to be represented.

Once the user has configured the desired system model, a process is selected (Figure 5), which in this case builds the Simulink model, establishes the solver settings, initializes the model parameters, and sets up the model for the user to perform the algorithm development tasks.

The entire model is then automatically built in Simulink, as shown in Figure 6.
The control algorithm template subsystem in this model is initially a placeholder for the algorithm models to be developed by the user, as shown in Figure 7.

Using the Autonomie Utility functions, the user can populate the Control Algorithm Template subsystem with the algorithm model(s) to be developed, and interface the algorithm model to the rest of the simulation. The result of this operation appears in Figure 8.

Now the user can work on the algorithm model in the context of a complete system simulation, and perform the appropriate development and testing tasks.
Autonomie Utility

The Autonomie software also includes utility functions to help manage the work products of the simulation. A picture of the Autonomie utility user interface is shown in Figure 9.

The left-hand pane of the utility interface provides an explorer view of the projects, folders, and files in the configuration management (CM) Synergy database used to provide CM of the models and associated files, as well as the folders and files on the user's hard drive.

The center pane provides functions to permit the user to download/upload models, systems, and vehicle files from/to the database; select a library to be used with the Simulink model(s); validate a model for use in the Autonomie environment, which includes generation of an XML file describing the inputs, outputs, and parameters of the model; and manage the models saved on the user's hard drive.

The right-hand pane of the utility provides the detailed functionality to perform the task that was selected in the center pane.

Software-in-the-Loop

GM has developed a proprietary process by which production source code can be brought into Simulink simply, robustly, and in a way that is easily repeatable. This process enables a control algorithm engineer to use existing production code in a model-based control design environment. As indicated earlier in this paper, development of control algorithms in the context of a complete system simulation enables an engineer to consider interactions among components, subsystems, and other algorithm functions, as well as the effects of part variation and implementation details such as data typing and operating system scheduling. Including existing production code in a Simulink model is not a simple task, considering the complexity of the code and the number of interconnections needed. The process developed by GM, and the model management and automated connection capabilities of Autonomie, eliminate the difficulties associated with this task.

SIL Process

The SIL process, which brings production source code into Simulink, consists of two subprocesses - a build process and
a usage process. An overview of the SIL process is shown in Figure 10.

The build process consists of processing source code through a set of utility programs that produce components consumed by the usage process. The build process is tightly integrated with GM Powertrain's software build environment. The software build environment is supported on a Linux platform.

The usage process consists of using the components produced by the build process to bring compiled code into Simulink. The usage process is tightly integrated with GM Powertrain's model-based algorithm development environment. The modeling environment is supported on a Windows platform.

SIL Block
The SIL block, a Simulink block developed by the GM Algorithm Modeling team, creates an S-function that allows the model to interface to objects such as global data or functions within the SIL build (.dll).

SIL User Interface
The SIL block includes a user interface that:

• allows the user to select the software build to be included in the model
• permits the user to select the function(s) to be executed from the software build
• displays the global variables in the compiled software and allows the user to assign them to inputs or outputs
• supports scheduling of the functions within the software build, and allows the user to interleave execution of those functions with execution of the functions of the included algorithm model.

SUMMARY/CONCLUSIONS
The tools and methods described in this paper enable a control system design capability that was heretofore very difficult to achieve. Control algorithm development in the context of a complete system simulation provides many advantages that result in high-quality robust control systems. Engineers are able to account for interactions among the many functions, subsystems, and controllers that comprise modern automotive controls, and to study the effects of variation on the system's robustness. The approach developed cooperatively between GM and Argonne National Laboratory overcomes the challenges and difficulties of constructing a complete system simulation.

The Autonomie software developed at Argonne facilitates management of the models, including automatic interconnection of the subsystems. Autonomie provides a complete environment for assembling, building, running, and analyzing complete vehicle (or vehicle subsystem) models.

The SIL capability developed at GM allows the user to include a complete representation of production control algorithm software, thus permitting the simulation to include the functionality of algorithms for which no simulation

Figure 9. Autonomie Utility User Interface
models exist. The SIL facility provides complete flexibility in scheduling and executing the production code tasks, as well as mixing and matching them with algorithm models represented in Simulink. This permits the engineer to run the same tests and experiments that are normally performed on a hardware-in-the-loop simulator, and provides the capability afforded by a rapid controller prototyping system running in bypass mode.

Autonomie is based on MATLAB/Simulink, which may encounter computer performance limitations for very large models. The SIL feature avoids many of these performance issues by including compiled C code for all production control algorithms except the one under development. Even so, there may be situations where simulation speed is adversely impacted by the model size. In such cases co-simulation technology may be employed to distribute the computing workload among multiple processors or computers.

This simulation environment is being deployed at GM for all control algorithm development work. Future development will provide mechanisms for deployment of Autonomie in a production environment, and integration of additional modeling tools into the framework.

Organizations that embrace model-based control system development can benefit greatly from adoption of Autonomie by virtue of its ability to facilitate algorithm development in a complete system simulation environment.

ACKNOWLEDGMENTS

This work was supported by DOE’s Vehicle Technology Office under the direction of Lee Slezak and David Anderson. The submitted manuscript has been created by UChicago Argonne, LLC, Operator of Argonne National Laboratory (“Argonne”). Argonne, a U.S. Department of Energy Office of Science laboratory, is operated under Contract No. DE-AC02-06CH11357. The U.S. Government retains for itself, and others acting on its behalf, a paid-up nonexclusive, irrevocable worldwide license in said article to reproduce, prepare derivative works, distribute copies to the public, and perform publicly and display publicly, by or on behalf of the Government.

REFERENCES


CONTACT INFORMATION

ARGONNE NATIONAL LABORATORY
Aymeric Rousseau
(630) 252-7261
arousseau@anl.gov

GENERAL MOTORS COMPANY
Michael A. Kropinski
(586) 634-9854
michael.kropinski@gm.com