

# Advanced Testing Methods for Automotive Software

Madison Turner, Technology Analyst

Accelerated Technology, a Mentor Graphics Division

Recent history attests to the need for improved software testing methods in the automotive industry. Take, for example, Toyota's October 2005 recall of 75,000 Prius hybrids in North America. A logic error in embedded software exacted a huge, if undisclosed, financial cost for the company and an inestimable cost in consumer confidence. And while we may never know the exact series of events that led to flawed code being shipped in the Prius, we do know the key to delivering bug-free software: test early and test often.

However, the distributed nature of automotive control systems, as well as the nature of the embedded software development process itself, makes testing a challenge. In the first place, simply gaining access to the necessary hardware environment for early and frequent testing may be expensive or impracticable. Moreover, effective validation and testing of distributed applications requires a sophisticated software infrastructure that is unlikely to be available off the shelf for the wide variety of microprocessors and peripheral hardware used in automotive systems and inordinately time-consuming and expensive to develop in-house. These factors speak to the need for testing environments that are abstracted from the underlying hardware architecture, capable of advanced regression testing and debugging of distributed applications, and aware of the operating system and protocol standards used in the automotive industry.

## ***System Simulation Offers a Solution***

When the term simulation is mentioned, most embedded software developers tend to think of instruction set simulators. And while instruction set simulators certainly have a place in the developer's toolbox, the problem domain they address is a narrow one, namely prototyping embedded software on a development host in lieu of running on target hardware. But today's embedded systems increasingly function in the context of larger systems and networks, where simulating a particular processor architecture is mostly irrelevant to the difficult problems of testing and validation. This leaves the bulk of testing for correctness, reliability and ease of use to the phase of development where hardware prototypes are available – that is to say the end of the development cycle, when scheduling pressures are at their peak.

Automotive systems are a perfect example of the type of application where last minute testing is simply unacceptable. Obviously, vehicle control systems must perform flawlessly, and the dozens of processors communicating over a CAN network in a modern automobile show the futility of simulating a single processor as a method of validating the entire system. But these days telematics and infotainment systems are some of the key differentiators among an ever-expanding choice of vehicles, and the early testing of these systems, with their difficult-to-quantify ergonomic and ease of use dimensions, could be a major boon to carmakers as well.

System simulation and behavioral hardware modeling are approaches that are emerging to solve these problems. System simulation refers to the practice of simulating networks, human-machine interfaces and peripheral hardware in order to test an embedded software

application in its full system context. With a full system simulation, network communications can be tested amongst multiple applications and application instances, virtual models of human-machine interfaces can be tested with the code that will drive them, and indeed any source of stimulus to the application can be simulated and tested with production code. Behavioral hardware modeling is an approach to simulating peripherals wherein the hardware can be modeled at any level of abstraction from the register transaction level (RTL) to the hardware driver API level. Selecting the interface between the model and application is a tradeoff between the model complexity and the amount of hardware driver code that can be kept from the simulation environment.

Figure 1 shows a complete set of simulation models in the Nucleus SIMdx environment. It includes a virtual steering wheel with cruise control, a dashboard display, brake and gas pedals, and a gearshift. Waveform and event windows show system events and CAN protocol events, and a state window displays a snapshot of selected system data.



Figure 1: A system simulation environment for automotive software

The benefits of these simulation and testing methods for automotive software developers include increased code quality and easier integration due to early testing; improved ergonomics and ease of use for human-machine interfaces; and streamlined workflows and savings on development hardware.

## ***System Simulation for Automotive Control Systems***

Automotive control systems are a natural application for system simulation techniques. First of all, these safety-critical and mission-critical systems require the kind of advanced and early testing that a system simulation solution provides. In addition, the networked configurations of controllers in these systems exceed the limitations of instruction set simulation. Moreover, the great expense and complexity of testing these systems in a hardware environment indicates the need for a simulation environment with extensive modeling and regression testing facilities.

For control systems on which an automobile's safety and reliability depend, software testing can't wait until the car is built. Early and rigorous testing can mean the difference between a reputation for dependability and a disastrous recall. The ability to test networked systems over a simulated CAN bus, with all of the attendant software protocols in place, constitutes a tremendous advantage for automakers that adopt a system simulation methodology.

Multiple different applications or instances of the same application can be executed on a single workstation, or they can run on multiple workstations and communicate with one another over the PC network. The system simulation environment provides facilities not only for creating and running behavioral hardware models, but also for developing and automating test scripts that provide various combinations of stimuli to the target applications. Sophisticated regression testing suites can be created in this way.

The system simulation environment also provides monitoring facilities that log and visualize selected aspects of system execution and communication, for example,

messages sent and received over the bus protocol. In addition, the simulation environment can provide kernel-aware monitoring for controllers running an OSEK-compliant operating system and display execution data in terms of OSEK kernel objects.

Used together, these system simulation capabilities add up to unprecedented power and flexibility in testing automotive control systems. And not only does software-based simulation introduce full system debugging earlier in the development cycle, before hardware is available, it also reduces the expense of development hardware throughout the project.

### ***Interface Modeling for Telematics and Infotainment***

In-car telematics and infotainment devices present a different set of testing requirements. Here, ergonomics and ease of use in large part determine the success of the system. The system simulation approach to developing and testing these systems relies on human-machine interface (HMI) modeling to solve difficult problems of interaction between people and devices.

By creating models of human-machine interfaces within the simulation environment, developers can test their software with virtual representations of the controls and displays that it will drive in the real world. An HMI model is essentially bitmapped artwork depicting the physical device – such as a dashboard LED display, a stereo face panel or a GPS receiver – that is attached to the real embedded code that used to control the device. Developers and testers can interact with the HMI model in the same ways they would with the real device, albeit using a mouse pointer instead of their hands.

Testing embedded code with an HMI model can often reveal problems with the interface that would otherwise only show up in the final phases of development, when they are expensive or impossible to fix. As part of a system simulation environment, it can also help sort out complex interactions between subsystems. For example, when a phone call is received in the cab of a car, the stereo should be muted and caller ID information displayed. Moreover, having a virtual representation of the physical device allows feedback from non-technical users of the system, informs software decisions that make the system easier to use, and exposes the system's operation to review by interface designers and managers that would otherwise have to wait for a physical prototype.

Infotainment and telematics systems are among the leading factors influencing purchase decisions in both the consumer and commercial automotive sectors today. System simulation with HMI modeling offers a path to competitive advantage by facilitating the development of in-car electronics that are well thought out and easy to use.

## ***Conclusions***

The system simulation approach to early testing and validation of embedded software provides automotive software developers with the tools they need to build quality into control systems and make in-cab gadgets and interfaces intuitive. By taking advantage of system simulation methodologies, automakers can improve the safety and reliability of their products, differentiate their offerings with winning telematics and infotainment devices, and save money on software development to boot.