


**E**mbedded systems traditionally have limited processing and memory resources to minimize costs, yet consumer expectations are escalating to include more functions along with auxiliary features, such as high-speed communications and interactive multimedia. As the software portions of these embedded projects increase in complexity while schedules shrink, designers are considering graphical-design tools and automatic code generators to augment the limited pool of skilled programmers. With an up-front investment to precisely define the system, design tools can automate some of or the entire software-development task. In many cases, these

tools transform embedded-software development from a line-by-line-programming task to a fully automatic process that directly generates operational source code from an abstract model of the system.

Although the thought of replacing a portion of the embedded-software staff with a desktop computer is intriguing, it is not a simple task to sufficiently specify the system for automatic code generators. Current development tools require a significant learning period and possibly a complete change of programming viewpoint before designers are ready to prepare the data necessary for automatic code generation. It may also require a change in mindset among your current software staff to overcome prejudice and accept a different embedded-development technique. Designers have strong opinions when it comes to automatic code generation. Based on early product experience, some developers feel that code generators require substantial overhead and produce inefficient code. Yet, modern tool vendors claim that their code generators produce error-free code that precisely matches the specification and is more compact than handwritten code.

AS EMBEDDED-SYSTEM TEAMS RACE TO BRING NEW PRODUCTS TO MARKET, DESIGNERS ARE EVALUATING MODEL-BASED DEVELOPMENT TOOLS TO DEAL WITH ESCALATING SOFTWARE COMPLEXITY.



# MODEL BEHAVIOR: CREATING EMBEDDED- SOFTWARE SHORT CUTS

BY WARREN WEBB • TECHNICAL EDITOR

Automatic code generation is an element or subset of model-based design in which the designer defines the system with a formal graphical representation instead of a traditional written specification. The model must unambiguously define the intended functions and behavior of the proposed system. In general, most system-modeling programs allow the user to construct a system representation by selecting prebuilt blocks from a library and connecting them to form the graphical model. Becoming adept at system modeling requires the same devotion and training period as learning a new programming language.

## PRE-PROTOTYPE CODE

One of the main benefits of system modeling is the ability to simulate the operation of the embedded system before committing the design to hardware. A simulator allows you to test and optimize performance to verify that the system meets requirements. A PC-based simulation does not duplicate the deterministic performance of a real-time system yet allows you to test many functions, such as the user interface. Most modeling systems also have built-in error-checking features to automatically check for inconsistencies and ambiguities to eliminate

potential mistakes early in the development cycle. Coupling a graphical model with a simulator and code generator can produce operational system software even before the prototype hardware is available. This approach is especially effective in troubleshooting and debugging the first hardware prototypes because it exercises all functional paths (**Reference 1**).

Some modeling systems offer reverse-engineering to automatically synchronize and update the model when you change object code in the lab. This feature connects the design and coding activities and guarantees that the model always reflects the latest product status. Reverse-engineering also implies that you can directly create a model from any embedded source code, but success depends on the way the developer created the code. Most modeling programs also allow the user to extract all system data, in both graphical and textual formats, to create standard or custom system documentation. Like reverse-engineering, automatic documentation ensures that the paperwork tracks the hardware.

The UML (Unified Modeling Language) is one of the popular standards developers use to prepare models for automatic code generation. The industry in 1997 first adopted UML, a



consolidation of as many as 50 modeling approaches developers advanced in the early 1990s. The OMG (Object Management Group) maintains UML, and you can download the latest version of the specification plus tutorials without charge from its Web site at [www.uml.org](http://www.uml.org). UML is a nonproprietary graphical technique for capturing the analysis and design of object-oriented software. As with any other object-oriented method, UML is based on classes and objects. A class is a set of objects that have the same data and behavior. Class attributes define the encapsulated data, and class services describe the functions that act on attributes. The latest UML specification defines 13 diagram types to document the software system from the behavior, interaction, and structure points of view.

IBM offers one of the first and most widely used commercial UML modeling tools: Rational Rose. IBM has versions for development of standard business and real-time embedded-software systems. For example, the Rational Rose Technical Developer software is a UML-model compiler that generates complete C, C++, and Java code for Unix, Linux, Microsoft Windows, and RTOS targets. In addition to runtime-model execution, visual-model debugging, and model-based testing, the package includes a porting wizard that provides

### AT A GLANCE

■ Embedded-system designers are turning to model-based development to shorten software schedules and simplify functional changes or hardware upgrades.

■ Although complex, the Unified Modeling Language provides development teams an open standard to model the architecture of embedded systems.

■ With built-in error checking, automatic documentation, and reverse-engineering, modeling software eliminates many common software mistakes.

■ System models and simulators allow designers to test system performance and prepare final target code before delivery of prototype hardware.

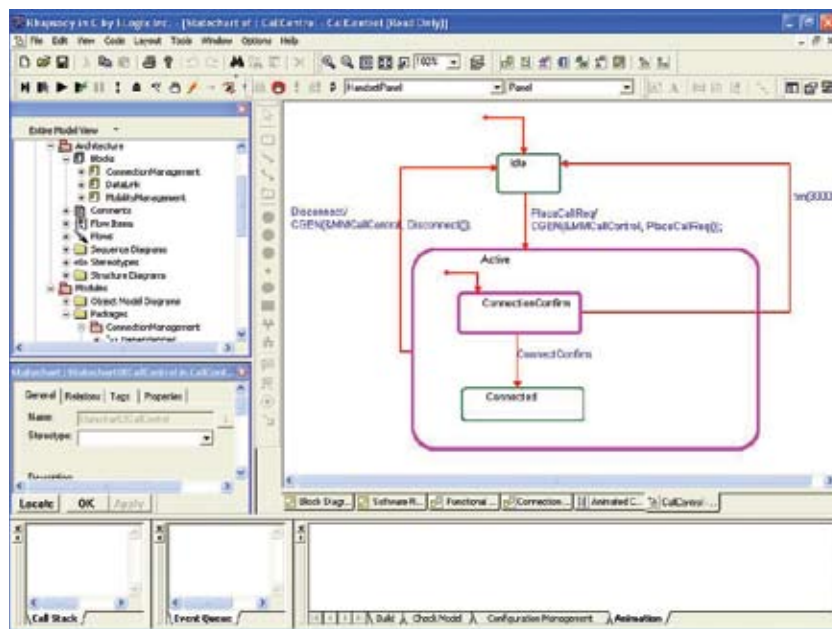
support for virtually any 8-bit or larger target platform. The software also offers forward- and reverse-engineering support for some of the most common Java constructs. Rational Rose automatically updates configuration management in the background to support remote team development. Prices for IBM's Rational Rose Technical Developer software plus 12 months of support start at \$5995.

Specializing in real-time embedded software, the Rhapsody Visual Program-

ming Environment from Telelogic integrates development into a single process in which design, code, and documentation continuously synchronize. You can generate C, C++, or Java directly from a UML model, merge your design with legacy code, or add your own custom code. You can also extract software components from a model and use them in new designs without the need for support from the original developer. **Figure 1** shows a representative user interface for the Rhapsody in C programming environment. The recently introduced Version 7.0 of Rhapsody features an enhanced user interface, reverse-engineering of legacy code, and seamless integration with the open-source Eclipse development platform. Telelogic offers unique software versions depending on the type of code produced and the target operating system. Prices start at slightly more than \$1000 for Rhapsody, and typical bundles sell for \$10,000 to \$20,000.

You can try UML without the expense of an off-the-shelf commercial program using one of several free alternatives. Both IBM and Telelogic offer free, although huge, time-limited trial downloads of their modeling software for buyer evaluation. Another possibility is ArgoUML, a free UML-modeling tool and active open-source-development project. ArgoUML is coded in Java and uses the Java foundation classes, allowing it to run on virtually any platform. Although not yet compatible with UML Version 2.0, ArgoUML supports most class, state-machine, activity, use-case, collaboration, and sequence diagrams. In addition to code-generation features, ArgoUML can reverse-engineer Java code and generate the corresponding UML diagrams. Similarly, the Umbrello UML editor, available with the Linux KDE (K Desktop Environment) desktop, is a Sourceforge open-source project that supports most of the latest UML standard (**Figure 2**). Umbrello delivers code generation for C++, Java, JavaScript, Python, Perl, and several other languages, along with reverse-engineering for C++. Unlike many other open-source projects, the Umbrello developer's Web site provides a complete user handbook that describes UML basics and each diagram type.

Many automatic-code-generation systems are based entirely on the UML



**Figure 1** The recently introduced Version 7.0 Rhapsody from Telelogic generates C, C++, or Java code directly from a standard UML model.

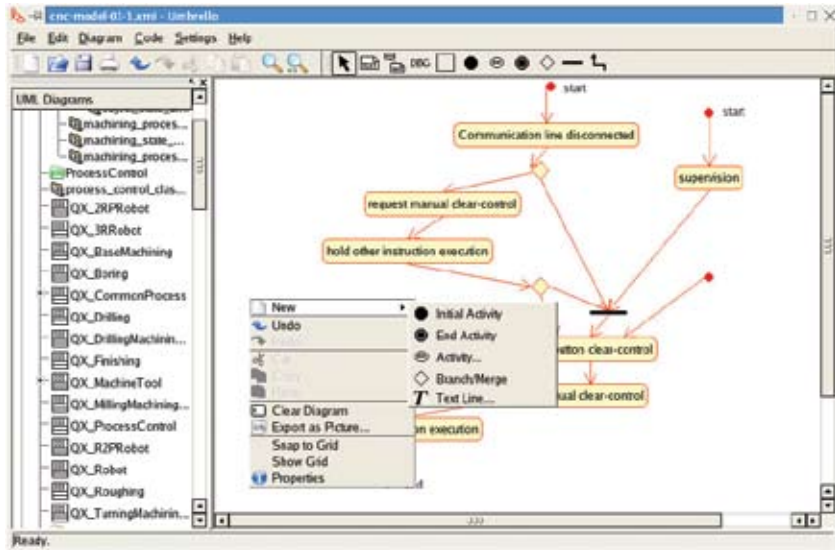


Figure 2 The Umbrello open-source UML editor supports the latest UML standard and generates code for C++, Java, JavaScript, Python, Perl, and several other languages.

standard; however, you can also find several successful software-development tools built on proprietary graphical-modeling systems. For example, The MathWorks offers Simulink, an interactive tool for modeling, simulating, and analyzing dynamic systems. You can build a model of your system from a library of predefined but customizable blocks. You can also divide model elements into subsystems to simplify construction and to enable reuse on multiple projects. The interactive simulator allows you to evaluate system performance and refine your designs. Simulink integrates with The MathWorks' Stateflow for modeling event-driven behavior and Real-Time Workshop Embedded Coder to automatically generate ANSI/ISO-C-compliant code (Figure 3). All tools require Matlab, which sells for \$1900, and extension tools sell for \$2800 for Simulink or Stateflow and \$5000 for the Real-Time Workshop Embedded Coder.

**TARGET OPTIMIZATION**

Gedae, another proprietary, model-based development tool, finds use in intensive signal-processing applications, such as radar, sonar, imaging, and audio. The Gedae tool set includes a workstation-development environment and target-specific runtime kernels for embedded targets, such as PowerPC processors, FPGAs, and DSPs. The Gedae core language specifies the functions of the model without regard to the target or its programming language (Figure 4). With built-in or user-supplied implementation parameters, Gedae transforms the model to generate optimized performance for the target. You then export target code to implement the application. Gedae also includes a variety of tools to map applications to multiple processors and visualize the execution activity. Support packages exist for multiple board vendors, and the Gedae BSP (board-support-package) development kit, which allows users to target code to their own custom hardware. Gedae comes in a variety of configurations for algorithm, workstation, and target development with prices starting at \$7500.

Recent upgrades to National Instruments' 20-year-old graphical programming language, LabView, provide embedded-software designers with another

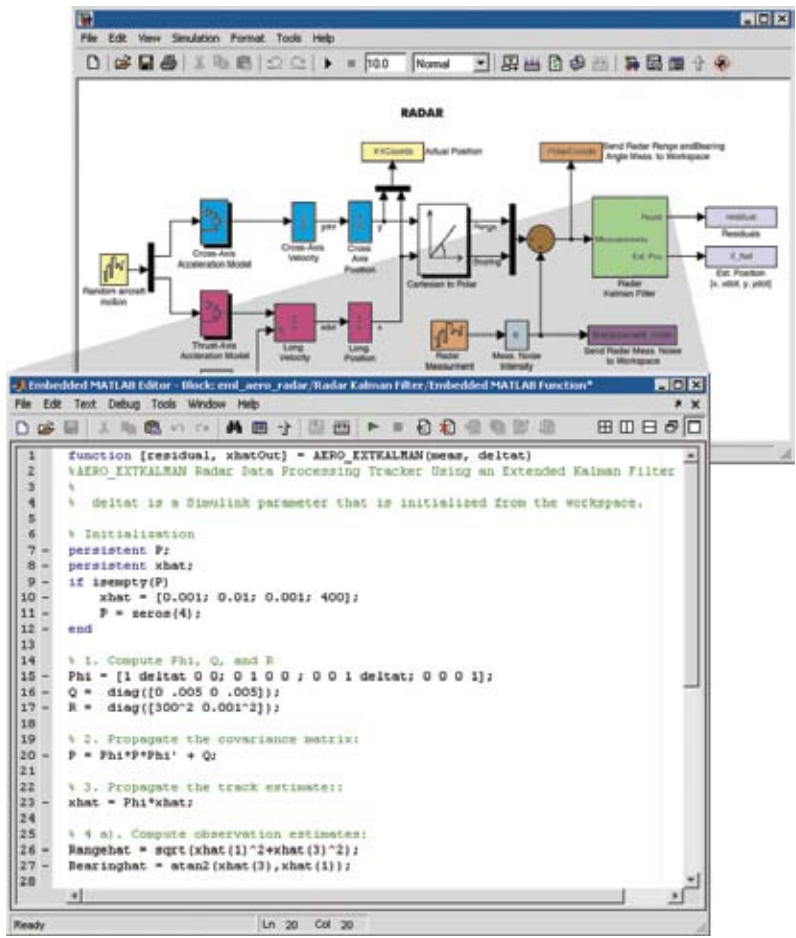


Figure 3 Simulink from The MathWorks allows you to build a model of your system from a library of predefined blocks and evaluate system performance.

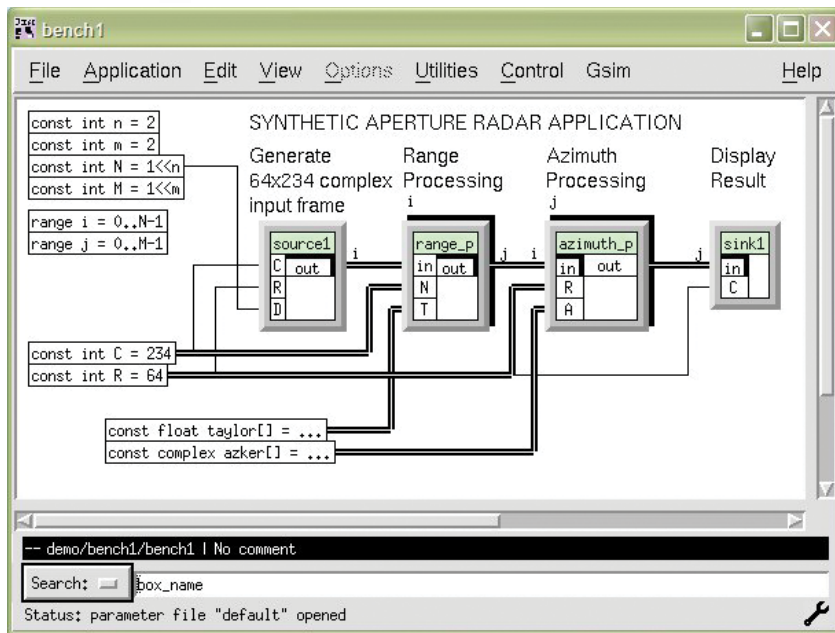


Figure 4 With the Geda workstation-development environment, users can specify model functions without regard to the target or its programming language.



Figure 5 The CompactRIO system from National Instruments defines custom hardware circuitry using a reconfigurable FPGA and LabView graphical-development tools.

avenue for automatic embedded-code generation. In addition to its system-modeling and -simulation features, the latest LabView release supports object-oriented programming with classes; objects; and data-encapsulation, or data-hiding—the mechanism whereby the implementation details of a class are hidden from the user—object-oriented-programming methods. A new embedded-development module provides C-code generation for most real-time operating

systems and any target 32-bit processor or DSP. Prices for LabView start at \$1199, and the embedded-development module costs \$10,995. National Instruments also offers the CompactRIO system, which allows developers to define custom hardware circuitry using a reconfigurable FPGA and LabView graphical-development tools (Figure 5). CompactRIO features a real-time embedded processor, a four- or eight-slot chassis containing a user-programmable FPGA, and 15 hot-

## FOR MORE INFORMATION

- ArgoUML  
argouml.tigris.org
- CMP Media  
www.cmp.com
- Gedae  
www.gedae.com
- IBM  
www.ibm.com
- The MathWorks  
www.mathworks.com
- National Instruments  
www.ni.com
- Object Management Group  
www.omg.org
- Telelogic  
www.telelogic.com
- Umbrello UML Modeller  
uml.sourceforge.net
- Venture Development Corp  
www.vdc-corp.com

swappable industrial-I/O modules. The FPGA circuitry at the heart of the CompactRIO system is a parallel-processing computing engine that executes embedded LabView applications at rates as much as 100 times faster than previously possible. Designers program CompactRIO hardware using LabView, the LabView Real-Time Module, and the LabView FPGA Module. Prices for CompactRIO embedded systems start at \$2495.

With software the largest embedded-project-budget item, development teams must consider software-automation tools to reduce the programming burden. CMP Media reports that more than half of all embedded-system projects reach completion at least three months behind schedule, and Venture Development Corp estimates that the number of lines of code per embedded project is growing at an average rate of 46% per year. These statistics point to a growing problem that traditional software-development practices fail to address. Although there are many initial hurdles to cross before adopting a model-based design approach, the savings in code generation, documentation, and product updates will eventually pay the transition costs.EDN

## REFERENCE

- De Niz, Dionisio, and Raj Rajkumar, "Model-Based Embedded Real-Time Software Development," Carnegie Mellon University, [www.cse.wustl.edu/~cdgill/RTAS03/published/TimeWeaverPosition.pdf](http://www.cse.wustl.edu/~cdgill/RTAS03/published/TimeWeaverPosition.pdf).

You can reach  
Technical Editor  
**Warren Webb**  
at 1-858-513-3713  
and [wwebb@edn.com](mailto:wwebb@edn.com).

