



Model-Based Design and Beyond: Solutions for Today's Embedded Systems Requirements

Jerry Krasner, Ph.D., MBA

January 2004

EMBEDDED MARKET FORECASTERS

American Technology International

www.embedded-forecast.com

Overview

Complexities and uncertainties surrounding embedded software developments continue to escalate development costs and frequently challenge product “windows of opportunity”. Engineering costs associated with design delays are further exacerbated by the failure of most embedded designs to approximate pre-design expectations.

Annual surveys by Embedded Market Forecasters (EMF) of embedded developers have clearly shown that software development is responsible for more than 80% of design delays and associated design complications. Whether the system is poorly conceived and specified or whether crucial algorithms fail to adequately address systems performance, traditional methods of embedded software development are yielding to a process known as “model-based design”. Model-based design is used to more clearly define design specifications, to test systems concepts and to automatically develop code for rapid prototyping and for software development.

Data is presented to demonstrate that model-based design technologies, including simulation-modeling, rapid prototyping, hardware-in-the-loop testing and automatic code generation offers better design results and considerable savings to OEM developers.

Data is also presented to show how developers – even those using model-based design - can further enhance their design efforts by utilizing the comprehensive set of model-based design tools.

What is Model-Based Design?

Model-based design emerged as a means of addressing the difficulties and complexities inherent in control systems designs. Developers recognized that software design needs to start before physical prototypes and systems are available. Traditional design processes resulted in the discovery of design and requirements errors late in the design cycle resulting in expensive delays and missed windows of opportunities.

In traditional design processes, design information is communicated and managed as text based documentation. Frequently this documentation is difficult to comprehend. Code is created manually from specification and requirements documents that are time consuming and error prone. There is little tracking to ensure that changes are correctly implemented.

Designs, such as those for avionics and automotive systems, have become too complex to develop and coordinate without the creation of a design environment common to all involved developers.

Model-based design, used to its fullest, provides a single design environment that enables developers to use a single model of their entire system for data analysis, model visualization, testing and validation, and ultimately product deployment, with or without automatic code generation.

At a minimum model-based design can be used as a specification that contains greater detail than text-based specifications. In real-time applications, it enables developers to evaluate multiple options, predict systems performance, test systems functionality by imposing I/O conditions that might be operationally expected (before product deployment), and test designs.

Once the model is built and completely tested, accurate real-time software for the production embedded design is automatically generated, thereby saving time and reducing costs compared to traditional manual coding. Model-based design with automatic code generation can also be used in rapid prototyping, enabling sub-system designs to be tested and optimized.

Although it is very important in highly complex design applications (e.g., guidance systems, engine controls, autopilots, anti-lock braking systems) that otherwise might be difficult to realize without it, model-based design can be used effectively and economically for less complex designs.

Furthermore, model-based design creates a structure for software reuse that permits established designs to be effectively and reliably upgraded in a more simplistic and cost effective manner.

Model-based design works in the following manner:

- The entire system model is visualized via block diagrams and state charts to describe knowledge and implementation details
- Design options can be evaluated and systems performance predicted via simulation of the system model
- Algorithm and behavioral models are optimized and refined yielding a fully tested specification
- Production quality software is automatically created for real-time testing and deployment from the fully tested specification

Model-based design saves money by cutting design time and providing final designs that more closely approximate pre-design expectations for performance, systems functionality, and features and schedule. It provides:

- Faster design iterations that produce desired performance, functionality and capabilities.
- Design cycles that are more predictable and result in faster product shipments
- Reduction in design, development and implementation costs.

Model-Based Design Enables OEMs to Retain Legacy Code

An important feature of model-based design tools is the ability to reuse existing pre-tested and deployed code as part of the model. In this way, existing applications can be updated or enhanced without having to model the complete design. Or put another way, it allows the developer the capability of more completely modeling the system without having to represent everything from scratch in the model-based design. A good example is the need to update or replace a user interface for an application while maintaining a majority of the existing design. For example, you can encapsulate the production code that will remain unchanged and execute it as part of the model, easily make the changes, and do less system-level testing before deploying the updated application.

The developer can encapsulate the production or legacy code modules using wrappers, usually written in 'C' that provide the appropriate interface to the model-based design tools. The developer needs only to describe the inputs, outputs, and timing required for the encapsulated code to insure that the timing and execution of the model is precise.

There are some aspects of system design that may be difficult to accurately and precisely model using model-based design tools. These include hardware device drivers and interrupt service routines, which are typically hand coded by software designers who have intimate knowledge of the hardware and system requirements in order to meet timing and performance constraints. These routines can also be encapsulated using the same interface and be incorporated into the model.

By using this method, code reuse is realized and design iterations can be accomplished more easily and with less testing required.

Looking at Embedded Design Processes

The following information was developed from the Embedded Market Forecasters 2003 Survey of Embedded Developers (www.embedded-forecast.com). The results presented are consistent with the results derived from surveys conducted over the past 24 months.

Table 1 presents a summary of design results according to schedule. The results are also cross tabbed according to architecture and vertical markets. Approximately 60% of respondent use FPGAs in their designs.

Table 1: Percentage of design completions according to schedule				
	Ahead	Behind	Cancelled	Outsourced
32-bit	15.4%	53.5%	13.1%	11.4%
64-bit	17.7%	47.3%	11.9%	16.8%
DSP	17.2%	54.8%	12.9%	11.3%
FPGA	15.8%	54.3%	12.6%	11.0%
Auto-Transport	14.8%	55.4%	15.1%	13.1%
Avionics	15.1%	52.0%	11.8%	15.2%
Bus Mach & Peripherals	15.1%	52.9%	14.8%	11.6%
Consumer Electronics	17.2%	52.3%	14.8%	11.9%
Datacom	11.8%	57.2%	16.5%	13.2%
Telecom	10.6%	60.2%	18.3%	7.5%
Electronic Instrumentation	18.3%	57.3%	13.3%	11.1%
Industrial Automation	19.1%	51.3%	13.0%	8.1%
Medical	18.1%	56.2%	11.6%	11.3%
Military	17.6%	52.1%	8.3%	14.3%

**Table 1
Percentage of Design Completions According to Schedule**

The cost of delays and cancellations were established in an EMF report “2003: *Embedded Hardware/Software Design Preferences*”, which took into account the number of developers per project, the average cost per developer and the period of delay (be it a delay in design completion or the period between design start and cancellation). By assigning a cost per developer, the reader can calculate the costs of delays and cancellations.

Of further concern to an OEM is the relationship between final design results and pre-design expectations. Failure to approximate pre-design expectations can result in design delays, removal of product features and other non-productive actions.

In a survey of 947 embedded developers, each was asked to reply to the question “how close to your pre-design expectation was your final design?” Three categories were presented (Performance, Systems Functionality, and Features and Schedule). Developers were given the following choices: within 10%; within 20%; within 30%; within 40%; within 50%; and, “not within 50%”.

In such, a developer responding, say, to performance with “within 30%” was indicating that the design was within 30% but not within 20%.

Table 2 presents the results from the 2003 EMF embedded developer survey.

Comparing Final Design Result to Pre-Design Expectations						
	Withi n	Withi n	Withi n	Withi n	Withi n	Not Within
	10%	20.0 %	30.0 %	40%	50%	50%
Performance	31.7 %	19.3 %	8.7%	2.3%	6.7%	31.3%
Systems Functionality	39.0 %	14.0 %	6.6%	1.9%	5.8%	32.8%
Features & Scheduling	19.8 %	18.6 %	13.8 %	5.4%	15.8 %	26.6%

Table 2
Comparing Final Design Results to Pre-Design Expectations

Cross tabulations were run according to 10 vertical markets, 7 microprocessor families, bus architecture, microprocessor architecture, and 5 types of engineers.

Over the preceding two year's of surveys, the comparison continues to worsen. It is clear that with over 30% of designs failing to approximate 50% of pre-design expectations (and 40% not being within 40% of expectations) that as designs become more complex OEM design expenditures will increase while windows of opportunity and product features will be challenged.

Cross tabulations were conducted in order to compare final design results/pre-design expectations between all respondents and those that use simulation-modeling, rapid prototyping and automatic code generation in their designs.

There are important aspects of this data need to be emphasized:

- Most developers use simulation-modeling as a systems development tool rather than as a software development tool
- Final design/pre-design expectations are software development related
- 35% of embedded developers use simulation-modeling in their design practices, but only 11.5% also use automatic code generation
- Of the developers that use automatic code generation in their design practices, 58% use it in conjunction with simulation-modeling (34% use it in conjunction with rapid prototyping)

And equally important, developers who use these model-based design technologies in their design practices indicate that they have a *higher* pre-design expectation than when they are not using these technologies.

It is clear from EMF surveys that:

- Embedded developers that use simulation-modeling in their designs have a higher pre-design expectation than non-model-based design developers
- Developers that use automatic code generation for their designs experience final designs much closer to pre-design expectations than those that don't incorporate automatic code generation into their designs
- Developers that use rapid prototyping in their designs experience the greatest improvement between pre-design and final design results

Most simulation-modeling users do not incorporate rapid prototyping or automatic code generation into their designs – they may not be aware of design enhancements that accrue from such use.

There may be several reasons for this.

- Simulation-modeling is (according to EMF data) used largely as a systems-based tool and is less frequently used as a software development tool. Enhanced systems design is an essential component of good design practices
- From the 2003 EMF embedded developer survey: Systems Engineers - 44.7% use simulation modeling (25.4% for software engineers), 15.3% use rapid prototyping (7.9% for software engineers) and 12.0% (11.6% for software engineers) use automatic code generation in their designs
- The EMF 2003 survey is responded to by a broad cross-section of embedded developers and applications. Hence the numbers reported herein are significant
- The use of automatic code generation for software development is growing rapidly in certain industries, such as aerospace and automotive

How Different Developers Use Model-Based Design Tools

The 2003 Survey of Embedded Developers tracked the broad use of different tools that are used by developers. This data was examined by cross tabbing each of the respondent's answers to each question posed according to engineering type, 10 vertical market design applications, processor architecture utilized in the primary design (including FPGAs and DSPs), and by seven microprocessor families (e.g., Motorola, Intel, ARM, etc.).

Table 3 presents the breakout use of model-based design tools (and software verification tools as a comparison) according to developer specialty.

	Percent of Total Respondents that Use a Model-Based Design Tool					
	Type of Developer/Engineer					
Tool	Hardware	Software	Systems	Firmware	Manager	All
Simulation Modeling	36.9%	28.0%	44.7%	36.3%	57.0%	35.0%
Automatic Code Gen	7.8%	11.6%	12.0%	8.7%	17.0%	11.5%
Rapid Prototyping Tool	3.9%	7.9%	15.3%	2.6%	14.1%	9.3%
Hdw in Loop Simulation	8.7%	8.2%	20.0%	5.2%	8.9%	10.1%
Software Verification Tools	9.7%	13.3%	17.3%	10.4%	26.7%	15.5%

Table 3
Percent of Responding Developers that Use a Model-Based Design Tool

In Table 3 it is evident that 44.7% of systems engineers use some form of simulation modeling in their designs, while only 12% use automatic code generation and 15.3% use a rapid prototyping tool. The use of software verification tools is used as a comparison.

Recalling that the use of rapid prototyping and automatic code generation resulted in superior design results that more closely approximated pre-design expectations, it is nonetheless clear from Figure 2 that only a fraction of developers using simulation modeling tools in their designs also use automatic code generation and rapid prototyping, notwithstanding the improvements in design results that are shown to be evident.

EMF believes that this data makes a clear case for developers to realize immediate payback from using the panoply of available Model-Based Design tools.

Examples of How Model-Based Design Tools Function in Developmental Applications

Model-based design tools can be invaluable in providing early design verification and a true executable specification. This is very important if a design has one or more of the following characteristics:

- Expensive or impractical to build early prototypes
- Complex system with multivariable and/or multi-rate subsystems
- Subject to governmental or other regulatory constraints
- Proof of concept must be demonstrated prior to full project funding

- Failure of the system or subsystems in testing could cause injury or liability
- Product design cycles are compressed and testing and validation cannot be

8

A typical example of the type of application well suited for Model-Based Design tools are automotive applications: power plant (engine), drive train, transmission, electrical, engine control, braking systems, and telematics (communications, A/C, heating, ventilation, etc.). There are many other applications that have the same types of systems: aerospace, trains, light rail, and ships. By no means is transportation the only application well suited for model-based design. Almost any embedded application can be modeled accurately using model-based design. The systems modeled can include both physical and electrical components.

The following table shows how model-based design is used in developing real applications.

Aerospace	Automotive	Industrial Controls & Automation	Consumer Goods & Office Equipment
DO 178B Certification	Engine emission control	Robotics & automation	Printers
Satellites	Anti-lock brakes	Motor and machine control	Copying machines
Missiles	Climate control, windows	Theme park rides	Mass storage
Autopilots	Automatic transmissions	Upgrades to unsupported legacy systems	Home appliances, refrigerators, washing machines
Guidance & navigation	Hybrid electric vehicles		
	Wire-to-wheel controls		

Other Application Examples

Military COTS: Current DoD primes and subs have not established methods for COTS evolution in order to address obsolescence and IC replacements. Model-based design can contribute significantly to establishing an evolutionary process.

Model-Based Design in the Product Lifecycle

The product lifecycle utilizing model-based design is presented in Figure 1.

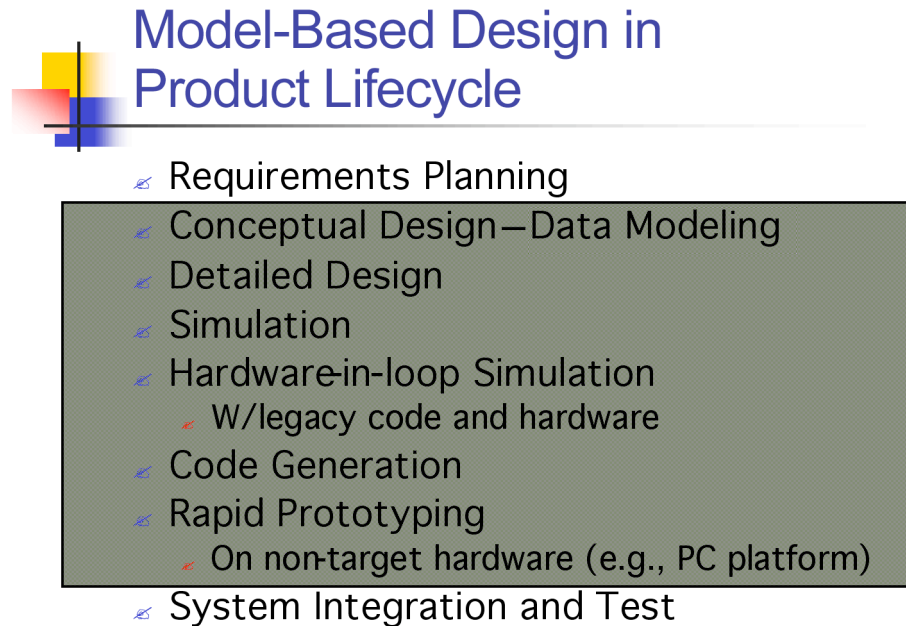


Figure 1
Model-Based Design in Product Lifecycle

The shaded area represents those aspects of a product's lifecycle that can be successfully addressed using model-based design. Different vendors offer tools that address specific aspects of the model-based design process, but very few can offer an array of model-based design tools that comprehensively address the product lifecycle.

Tools that Support the Model-Based Design Process – The MathWorks

In the EMF 2003 Survey of Embedded Developers (947 respondents representing a broad cross-section of the embedded marketplace) 414 developers responded to the question:

Which Simulation Modeling Tool Vendors Have You Used In The Last 12 Months?

- 69.8% indicated that they used MATLAB from The MathWorks
- 29.5% indicated that they used Simulink and Stateflow – both MathWorks model-based design products
- 49.4% of respondents indicated that they will use Simulink and Stateflow in the ensuing 12-month period

The MathWorks ranked ahead of all competitors in this market segment.

To compare the entire model-based design process from the viewpoint of the comprehensive model-based design product line from The MathWorks, Figure 2 illustrates MathWorks model-based design product line, and Figure 3 illustrates MathWorks model-based design product line superimposed on the product lifecycle.

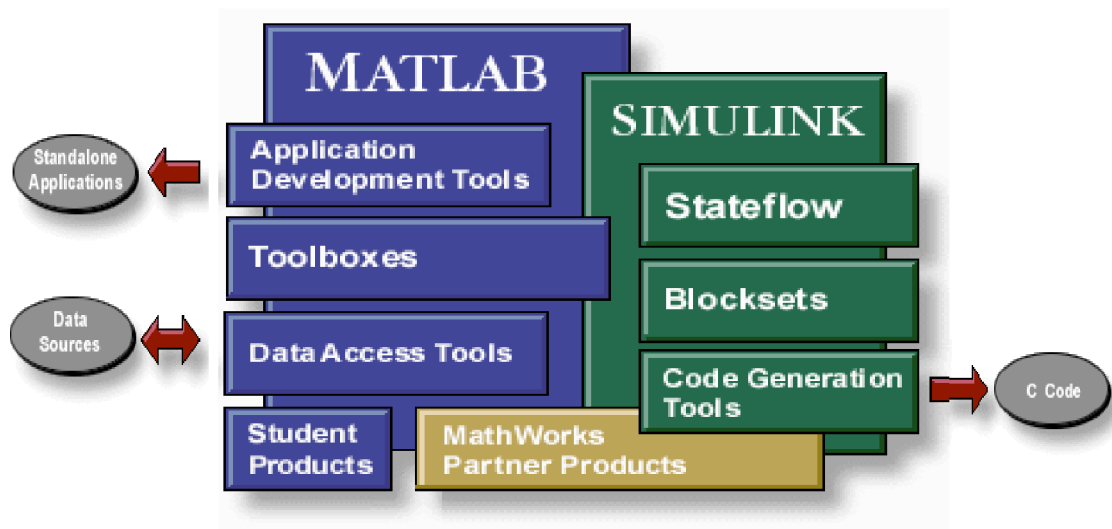


Figure 2
MathWorks Model-Based Design Product Line

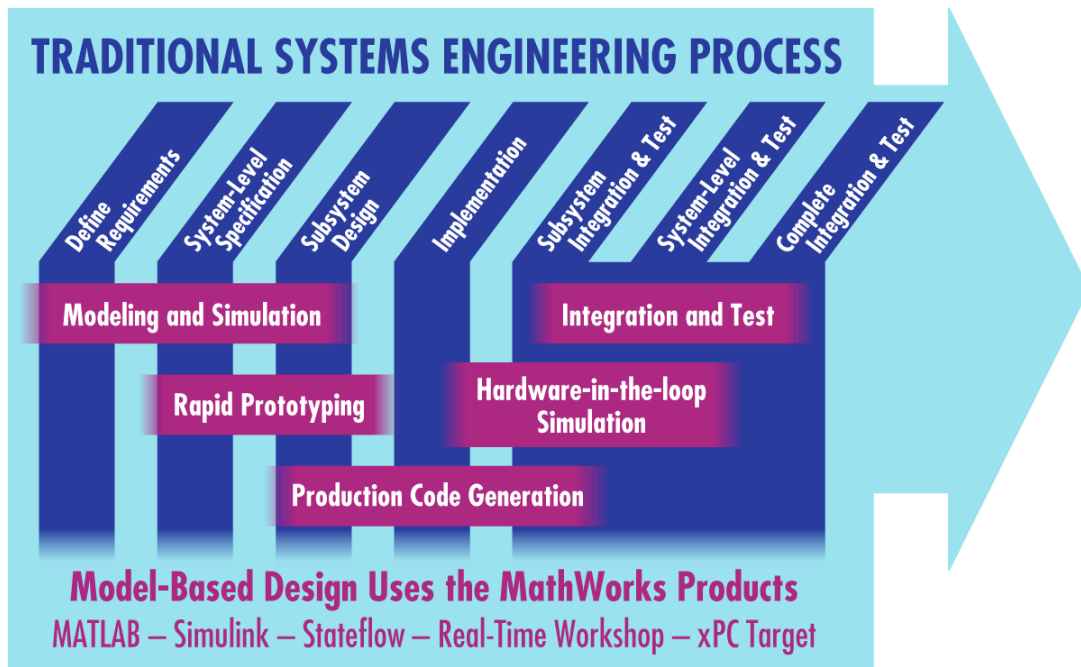


Figure 3
MathWorks Tools in Product Lifecycle

Summary

Model-based design offers developers a distinct advantage over traditional product development techniques. The ability to perform design validation and verification at the onset of a project as well as the ability to test system segments using rapid prototyping and automatic code generation (production code) provides a distinct advantage to the developer.

From a design/project management viewpoint, the ability to validate at different points during the design cycle enables management to more accurately forecast duration and costs. Model-based design also brings the concept of “design reuse” into the design cycle creating a framework for future savings.

A standard metric in product development is the number of lines of debugged code per developer per day. Assume a typical application generates a million lines of code. If model-based design tools are used to generate a portion of production code –say 30%, this relieves programmers from having to write 330,000 lines of code.