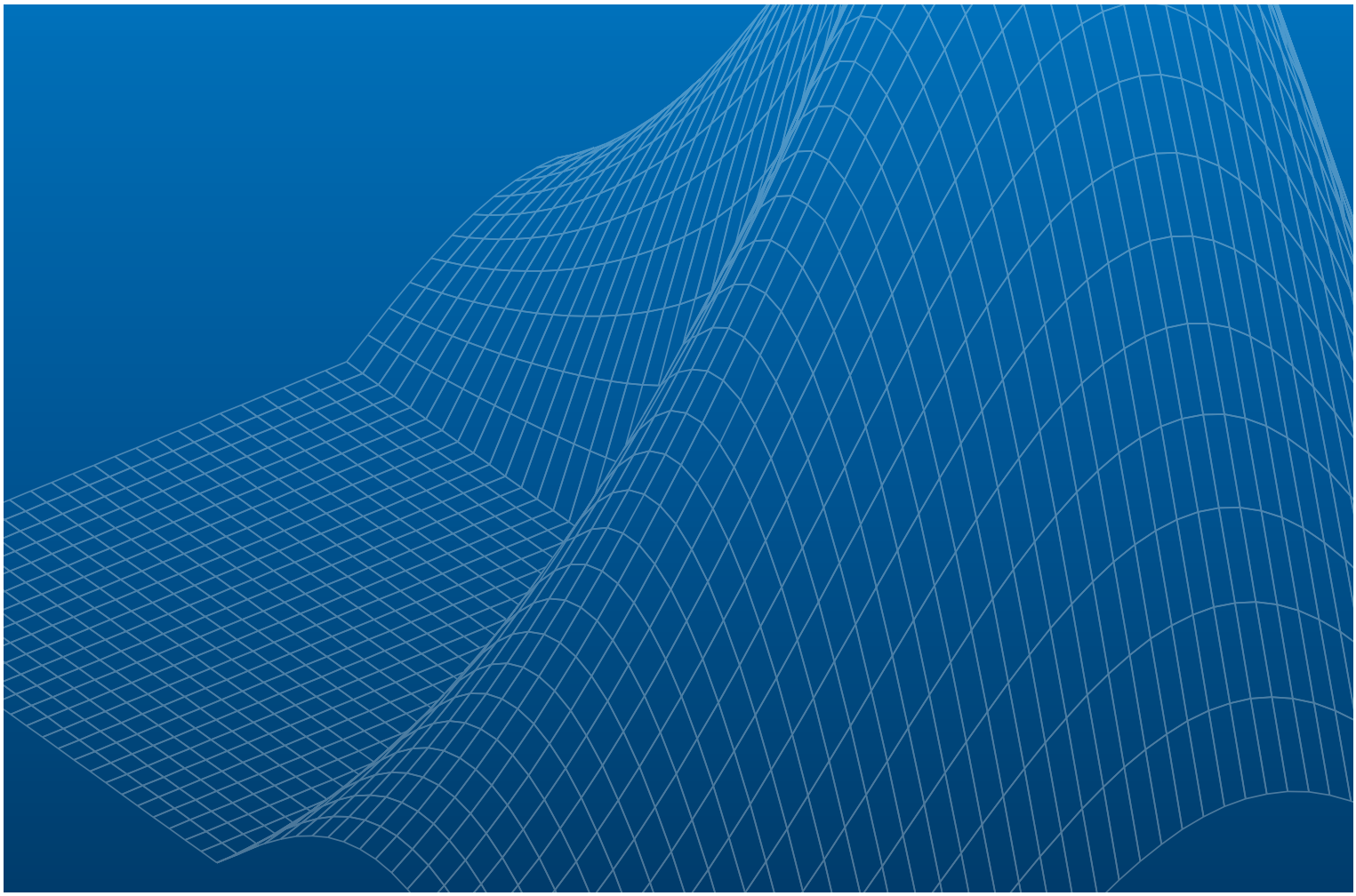

Measuring Return on Investment of Model-Based Design

By Joy Lin, Aerospace Industry Marketing Manager, MathWorks

As embedded systems become more complex, it is becoming more difficult to maintain quality and control costs with traditional approaches to software development. To meet this challenge and improve their competitive position, companies are shifting to Model-Based Design for embedded software development. However, the benefits of adopting Model-Based Design along with the supporting processes to fully leverage its benefits needs to be justified before the investment can be made. The Model-Based Design Return on Investment (ROI) framework described in this paper provides an analytical tool to justify investment in Model-Based Design by quantifying the expected savings of Model-Based Design over a traditional development approach.

I. Introduction

As customer requirements increase in scope and complexity, the logic and control software for systems has also grown in scope and complexity. As they develop the millions of lines of code required for airplanes and automobiles under ever tightening schedules, organizations have found that traditional development processes are no longer sufficient to meet quality and schedule targets. Model-Based Design for embedded software development lowers costs by identifying defects early in the development process and reducing the total number of latent defects. By helping companies deliver higher-quality systems at lower cost and in less time, Model-Based Design provides a competitive advantage.

II. Traditional Development vs. Model-Based Design

In a traditional development process, requirement, design, implementation, and test tasks are performed sequentially in different tool environments, with many manual steps (Figure 1). Requirements are captured textually, using tools such as Microsoft Word or IBM DOORS. Designs are created using domain-specific tools, which precludes system-level testing until after implementation in software or hardware. The designs are then manually translated into code, which is a time consuming and defect-prone process. At each phase, some defects are introduced, leaving the test phase to be the catch-all for all the defects that have accumulated throughout the previous phases. As a result, the test phase constitutes the bulk of development time and cost. Lack of a common tool environment, multiple manual steps, and late-stage defect discovery all drive up development time and cost.

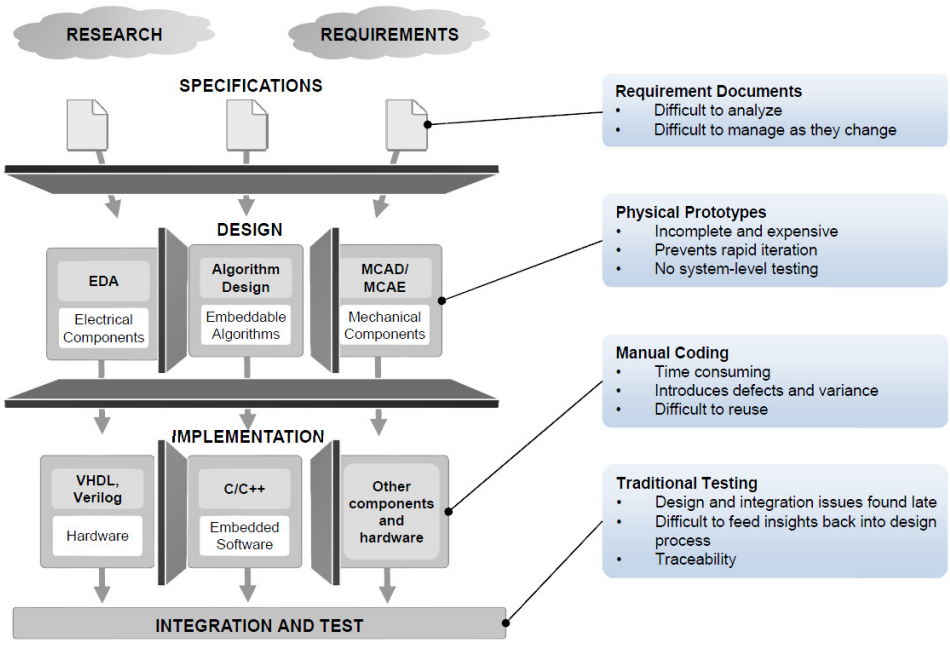

Figure 1. Traditional development methods require many unnecessary manual steps that can introduce defects.

Model-Based Design (Figure 2) starts with the same set of requirements as a traditional process. Rather than serving as a basis for textual specifications, however, the requirements are used to develop an executable specification in the form of models. Engineers use these models to clarify requirements and specifications. The models are then elaborated to develop a detailed design. Using the tools for Model-Based Design, engineers can simulate the design at the system level, uncovering interface defects before implementation. Once the design is finalized, the engineers automatically generate production code and test cases from the models. This workflow enables engineers to stay in the same environment from requirement to test, minimizing the amount of manual work. In addition, testing can begin at the requirement phase when engineers simulate their executable specifications in models to verify that the requirements are met. As a result, defects are caught and removed earlier, lowering the total cost of development.

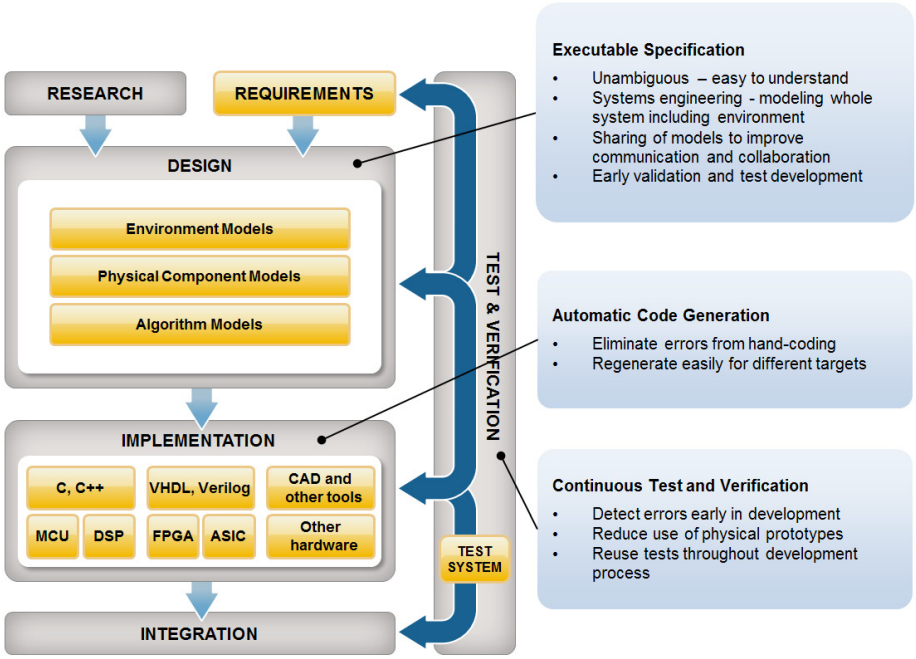

Figure 2. Model-Based Design uses a system-level model as an executable specification throughout development. This design approach supports system- and component-level design and simulation, automatic code generation, and continuous test and verification.

III. Advantages of Model-Based Design

Organizations that adopt Model-Based Design realize savings ranging from 20-60%, when compared to traditional methods. [1, 2] The bulk of these savings come from better requirements analysis combined with early and continuous testing and verification. As requirements and designs are simulated using models, defects are uncovered much earlier in the development process, when they are orders of magnitude less costly to fix (Figure 3).

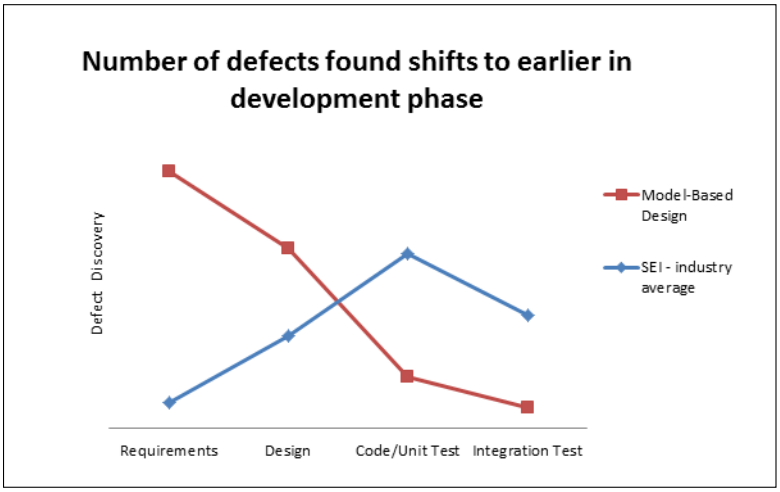

Figure 3. Model-Based Design shifts defect discovery to early development phases.

IV. Quantifying the Savings Using the Model-Based Design ROI Framework

The Model-Based Design ROI framework is designed to estimate the ROI from adopting Model-Based Design on specific projects. Based on project size, team size, and other factors, the framework calculates a cost for traditional development using the Basic Constructive Cost Model (COCOMO) model, and then subtracts the savings from Model-Based Design to obtain the Model-Based Design cost of development. (The Basic COCOMO model was chosen for the framework because it is a general parametric cost estimation tool widely used in the aerospace and defense industry, in which procurement cost accountability demands rigorous models for software cost estimation.) ROI is then calculated by accounting for the costs of software and training for the project team. The framework uses metrics from Software Engineering Institute (SEI), Institute of Electrical and Electronics Engineers (IEEE), and industry studies. Because project scope, existing processes, and team expertise using Model-Based Design vary across industries and companies, the Model-Based Design ROI framework can be customized for specific projects and teams.

Consider a baseline case of a software project with 500,000 LOC. Using the Basic COCOMO model, the cost of development using traditional methods would be approximately USD 6 million. To calculate the savings of Model-Based Design over traditional methods, each development phase—requirements, design, implementation, and test—is analyzed based on industry metrics. The savings are then summarized and subtracted from the traditional cost of development. In this case, the Model-Based Design cost is USD 3million, a 50% savings compared to the traditional method.

To arrive at the 50% savings, the framework examines inefficiencies in the traditional development process that Model-Based Design eliminates, and calculates the savings based on industry metrics and averages. Savings for each development phase are calculated separately, so that the framework is adaptable for step-by-step adoption of Model-Based Design.

The following section discusses one of the requirement inefficiencies to give a sense of how the framework works. In the requirements phase, using models to uncover, vague, inconsistent, or un-testable requirements enables engineers to uncover a higher percentage of defects. The baseline case assumes 9%. Uncovering these defects at the requirements phase means avoid costly rework later on in the development phase. Part of the requirement savings is arrived at by multiplying this 9% of additional uncovered defects by the average length of time for problem report resolution for a defect with root cause in incorrect requirement. In the baseline case, the average number of hours per requirement defect is 4.5 hours [3]. By this calculation, Model-Based Design saves 2,025 engineering hours. Figure 4 demonstrates a section of the framework that deals with the requirement analysis pain point. The framework contains seven additional sections that deal with different inefficiencies.

Requirements	
Percent un-testable	3%
Percent conflicting	3%
Percent vague	3%
Number of incorrect requirements	450
Hours to rework one requirement	4.5
Time spent on incorrect requirements (in hours)	**2,025**

Figure 4. ROI Framework calculates number of engineering hours saved by fixing incorrect requirements early.

Summarizing the savings from the entire development process, the overwhelming majority of the savings in this example came from requirement and test phases (Figure 5). This is due to more thorough requirements analysis, which results in fewer defects being passed on to the subsequent phases. Put simply, better requirements lead to better designs. Early and continuous testing results in more defects being identified and addressed in the same phase that they are introduced, which leaves fewer latent defects in the software and lowers overall development costs.

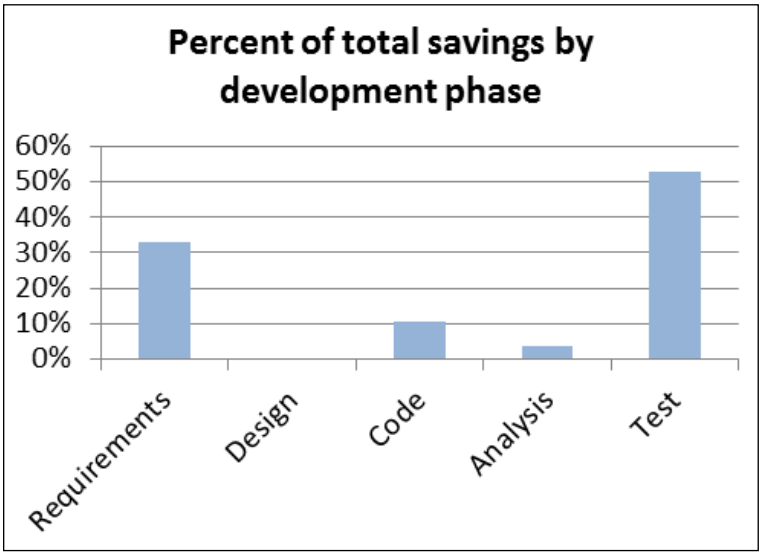

Figure 5. Savings in the requirements and testing phases accounted for most of the total savings.

When MathWorks collaborates with aerospace and automotive companies that are adopting Model-Based Design, the ROI framework helps guide the adoption process, enabling the companies to identify areas that will benefit immediately and significantly from transitioning to Model-Based Design.

Summary

For most companies, investing in new technology and processes is a risky endeavor. The return on investment calculation described in this paper aims to provide analytical backup for investment in Model-Based Design. In addition to justifying the investment, the ROI framework enables teams to identify areas in which Model-Based Design provides the most savings, as well as areas in which further investigation could lead to substantial additional cost reductions.

References

1. MathWorks, Swedish Space Corporation Develops Satellite Guidance, Navigation, and Control Software for Autonomous Formation Flying
www.mathworks.com/company/user_stories/userstory51480.html?by=industry

2. MathWorks, BAE Systems Achieves 80% Reduction in Software-Defined Radio Development Time with Model-Based Design
www.mathworks.com/company/user_stories/BAE-Systems-Achieves-80-Reduction-in-Software-Defined-Radio-Development-Time.html?by=industry

3. Tom King, Joe Marasco, What Is the Cost of a Requirement Error? StickyMinds
www.stickyminds.com/sitewide.asp?ObjectId=12529&Function=edetail&ObjectType=ARTCOL