Best Practices for Establishing a Culture of Model-Based Design

Paul Smith,
Director – North American Consulting Services
The MathWorks
Review: Benefits of Model-Based Design

- Simulation enables validation and documentation of requirements
- Requirements models are reused to generate code and documentation
- Test cases are reused to verify deployed application
- Results:
  - Improved quality
  - Reduced time to market
  - Capability to develop more complex systems
Adoption of Model-Based Design

What happens in the “real world”? Why do organizations adopt Model-Based Design?

- Corporate mandate
- 6 Sigma / quality initiatives
- Young engineers used it in college
- It’s more fun than writing C
Transformational Levels for the Adoption of Model-Based Design

From Phil Martens, Ford Motor Co., 2003 DARATECH Conference
Best Practice # 1: Identify the problem you are trying to solve

- Have metrics that identify the weak points in your current process
- Attack your greatest weaknesses first
- Monitor your Return on Investment (ROI)

**Example 1:** Can’t hit release dates
**Example 2:** Excessive software defects
**Example 3:** Availability of prototype hardware
Best Practice #2:

*Use models for at least two things – “Rule of Two”*

- Overcome startup costs and resistance to change
- ROI increases with multi-use models

**Example 1:** Validate requirements through simulation and add new functionality through rapid prototyping

**Example 2:** System specification and automatic code generation
Best Practice # 3: Use models for production code generation

- To ensure success you must connect models to real system
- Enable a culture of modeling by removing temptation and option to write code
- Executable code is what makes machines move and generates profits
Best Practice # 4: Treat models as the sole source of truth

- Remove the temptation to hack code by hand late in a program when under time pressure
- Prevent divergence of code and model
**Best Practice # 5:**

*Use migration as a learning opportunity*

- Learn what really happens in the current system
- Solicit help on process and tools, not on translation
- Focus on value-added features first
- Conversion is a tremendous learning and quality improvement opportunity
  - True even in small code footprints and efficient organizations
Best Practice # 6:  
*Focus on design, not on coding*

- Software design is still taking place

- Software engineers establish and manage the code generation infrastructure

- Model refinement continues after the controls engineers finish their work and before model is ready to generate code, especially in a fixed-point implementation

- Legacy code must be integrated and maintained
Best Practice # 7: 
**Integrate the development process**

- Develop a comprehensive plan:
  - Training
  - Modeling Style
  - Enforcement.
  - Supporting Tools
  - Configuration Management
  - Requirements Management
  - Process

- Develop new metrics

---

**Measuring Productivity and Quality in Model-Based Design**

Arvind Hosepahara
Technical Consultant (The MathWorks, Inc.)

Paul Smith
Managing Consultant (The MathWorks, Inc.)

Copyright © 2004 The MathWorks, Inc.

**ABSTRACT**

Accurate measurements of productivity and quality are essential for benchmarking workloads, ensuring predictable outcomes, and controlling quality. Traditional software development processes utilize well-accepted metrics for measuring productivity and quality. These include Lines of Code (LOC). With the introduction of Model-Based Design, organizations require a different measure of the software development process.

**INTRODUCTION**

A measure of the size of a software application, LOC is the translation for productivity measurements (LOC/MTM) and quality measurements, such as defects/deliverable.

With the introduction of Model-Based Design, organizations require a different measure of the software development process. For example, Model-Based Design enables automatic code generation from graphical models. This means that the average engineer can produce more than one LOC per unit time than is possible with hand coding, with virtually no software coding defects. Since these productivity gains are guaranteed in the process improvements, new metrics are required to properly measure and assess these improvements.

The automatic capture of process metrics in a modern management process increases data accuracy and output productivity. Practical experience has shown that an organization's quality metrics are likely to increase, thus making it possible to directly measure improvement, often without changing the underlying process. Additionally, requiring developers to manually capture a comprehensive set of process metrics can introduce bias and dilute them from their primary work. Model-Based Design offers the capability to automatically extract metrics, eliminating dead time to market, and avoiding the accuracy of quality-related issues.

---

**Model-Based Design**

1. Strategy
2. Organization
3. Process
4. Tools
5. Culture

This article describes an automatic, continuous measurement technique for gathering accurate metrics. The specific metrics that should be captured when using Model-Based Design and introduce a new long that can leverage these process metrics in the Simulink® and MATLAB® environment.

**OVERVIEW OF MODEL-BASED DESIGN**

At the heart of Model-Based Design are simulation models, graphical, test vector, executable model diagram representations of the physical system, the environment, and algorithm selection or control of signal processing and communications applications.

The models provide:
- A behavioral description of the embedded software—this physical object and environment upon which it acts.
- An executable specification that can be tested in simulation to ensure that it meets all functional requirements.
- A tool that allows the design team to convey information about the design.
- A specification from which code can be automatically generated, prototyping, and embedded into the model.
- Automatically generated test of test of integrated will contain embedded.

Model-Based Design helps to quickly evaluate multiple design options by testing and optimizing their algorithms in the modeling environment before they deploy them as an embedded system, reducing cycle time and design, and development and implementation costs.
Best Practice # 8: Designate champions with influence, expertise, and budgetary control

Business champion:
- Assigns overall priorities
- Assigns people
- Acquires tools, equipment, and services
- Sometimes act as a consensus builder
- Sometimes act as a benevolent dictator
- Handles issue escalation

Technical champion:
- Assigns technical priorities
- Is point of contact for Model-Based Design issues
- Attends MathWorks Advisory Boards
- May also be business champion in some organizations
Best Practice # 9: **Have a long-term vision**

- Good things come to those who have a vision and work hard to achieve it
- The full transition from hand-coded, textual languages takes 2-3 years to fully implement in a production organization
- Research organizations often have fewer constraints and less legacy code, and can move faster
- Be flexible
  - Don’t get bogged down with needs derived from traditional approaches (*paving the cow paths*)
  - Be receptive to workarounds
  - Plan for migration
Best Practice # 10: *Partner with your tool suppliers*

Suppliers bring the experience of working with entire industries and can help you avoid common pitfalls, accelerate your ROI breakeven point, and quickly achieve productivity and quality goals.

![Graph showing effort over time with Do it yourself and Leverage the supplier's experience compared.](image-url)