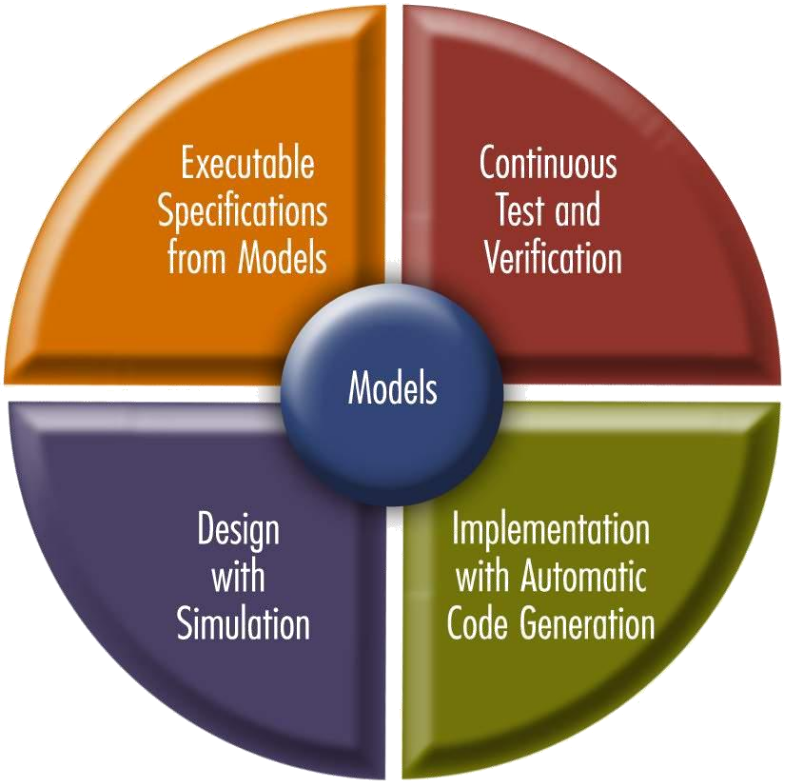
Pragmatic Strategies for Adopting Model-Based Design for Embedded Applications

Larry E. Kendrick, PhD **Senior Principle Technical Consultant**
The MathWorks, Inc.

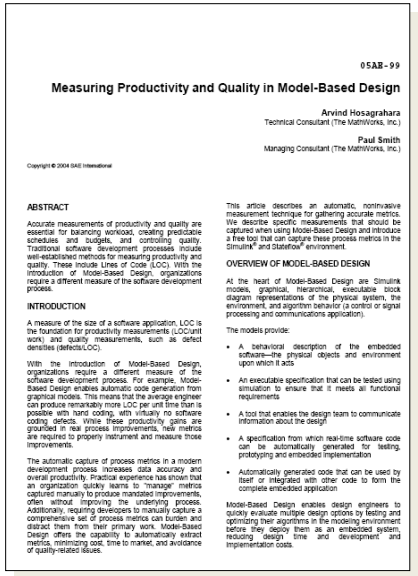
Introduction



- What's MBD?
- Why do it?
  - Make Products Faster
  - Minimize HW prototypes
  - Build it right the first time
- How to do it?

Best Practices for Establishing a Model-Based Design Culture (SAE Paper 2007-01-0777, Smith, Prabhu, Friedman)

1. Identify the problem you are trying to solve
2. Use models for at least two things – “Rule of Two”
3. Use models for production code generation
4. Treat models as the sole source of truth
5. Use migration as a learning opportunity
6. Focus on design, not on coding
7. Integrate the development process
8. Designate champions with influence, expertise, and budgetary control
9. Have a long-term vision
10. Partner with your tool suppliers



2005-01-0777

Measuring Productivity and Quality in Model-Based Design

Arvind Hosagrahara
Technical Consultant (The MathWorks, Inc.)

Paul Smith
Managing Consultant (The MathWorks, Inc.)

ABSTRACT

Accurate measurements of productivity and quality are essential for balancing workload, creating predictable schedules and budgets, and controlling quality. Traditional software development processes include well-established methods 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 foundation for productivity measurements (LOC/unit work) and quality measurements, such as defect densities (defects/LOC).

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 remarkably more LOC per unit time than is possible with hand coding, with virtually no software coding defects. While these productivity gains are grounded in real process improvements, new metrics are required to properly instrument and measure those improvements.

The automatic capture of process metrics in a modern development process increases data accuracy and overall productivity. Practical experience has shown that an organization quickly learns to 'manage' metrics captured manually to produce mandated improvements, often without improving the underlying process. Additionally, requiring developers to manually capture a comprehensive set of process metrics can burden and distract them from their primary work. Model-Based Design offers the capability to automatically extract metrics, minimizing cost, time to market, and avoidance of quality-related issues.

This article describes an automatic, noninvasive measurement technique for gathering accurate metrics. We describe specific measurements that should be captured when using Model-Based Design and introduce a free tool that can capture these process metrics in the Simulink® and Stateflow® environment.

OVERVIEW OF MODEL-BASED DESIGN

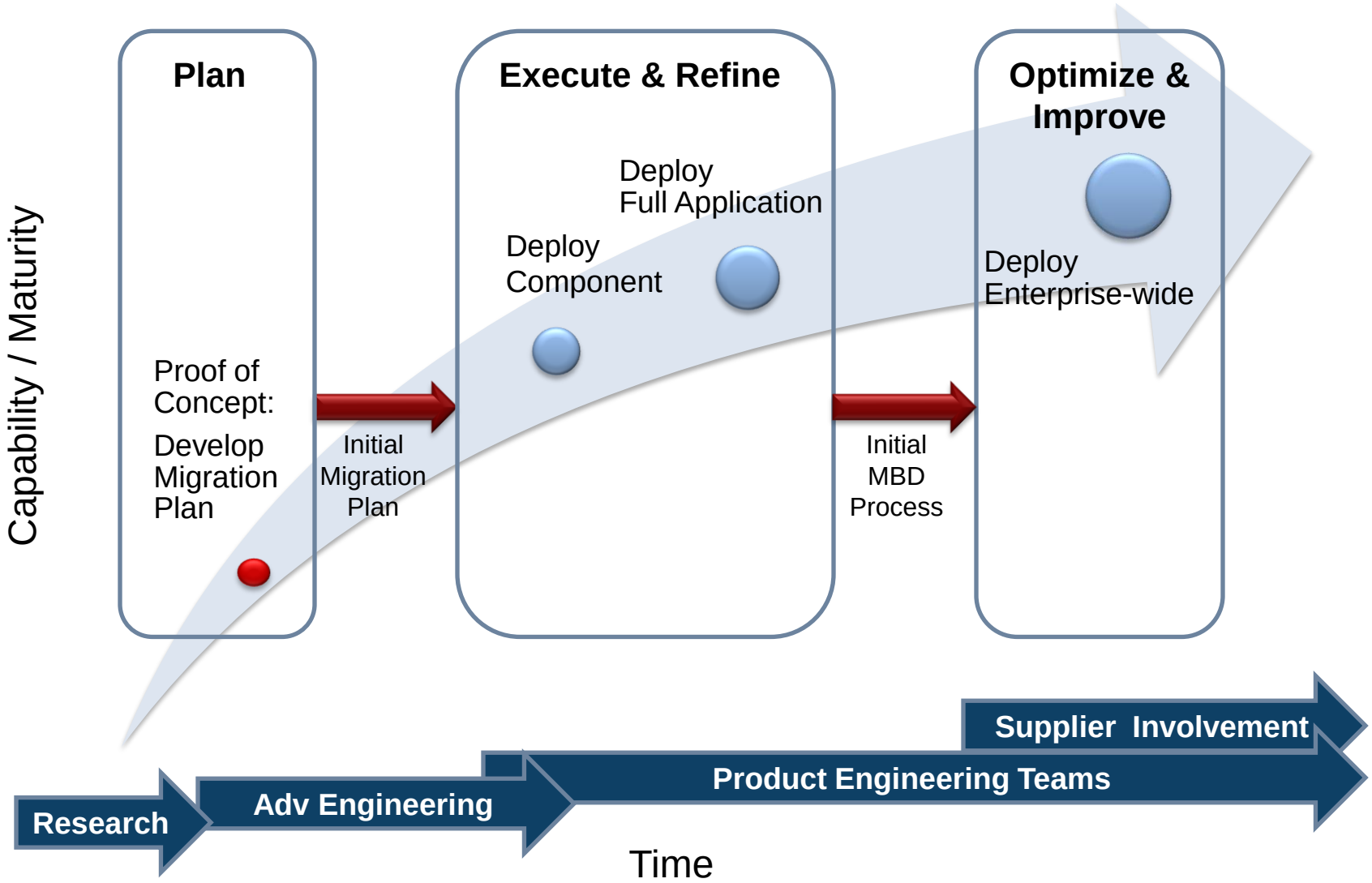
At the heart of Model-Based Design are Simulink models, graphical, hierarchical, executable block diagram representations of the physical system, the environment, and algorithm behavior (a control or signal processing and communications application).

The models provide:

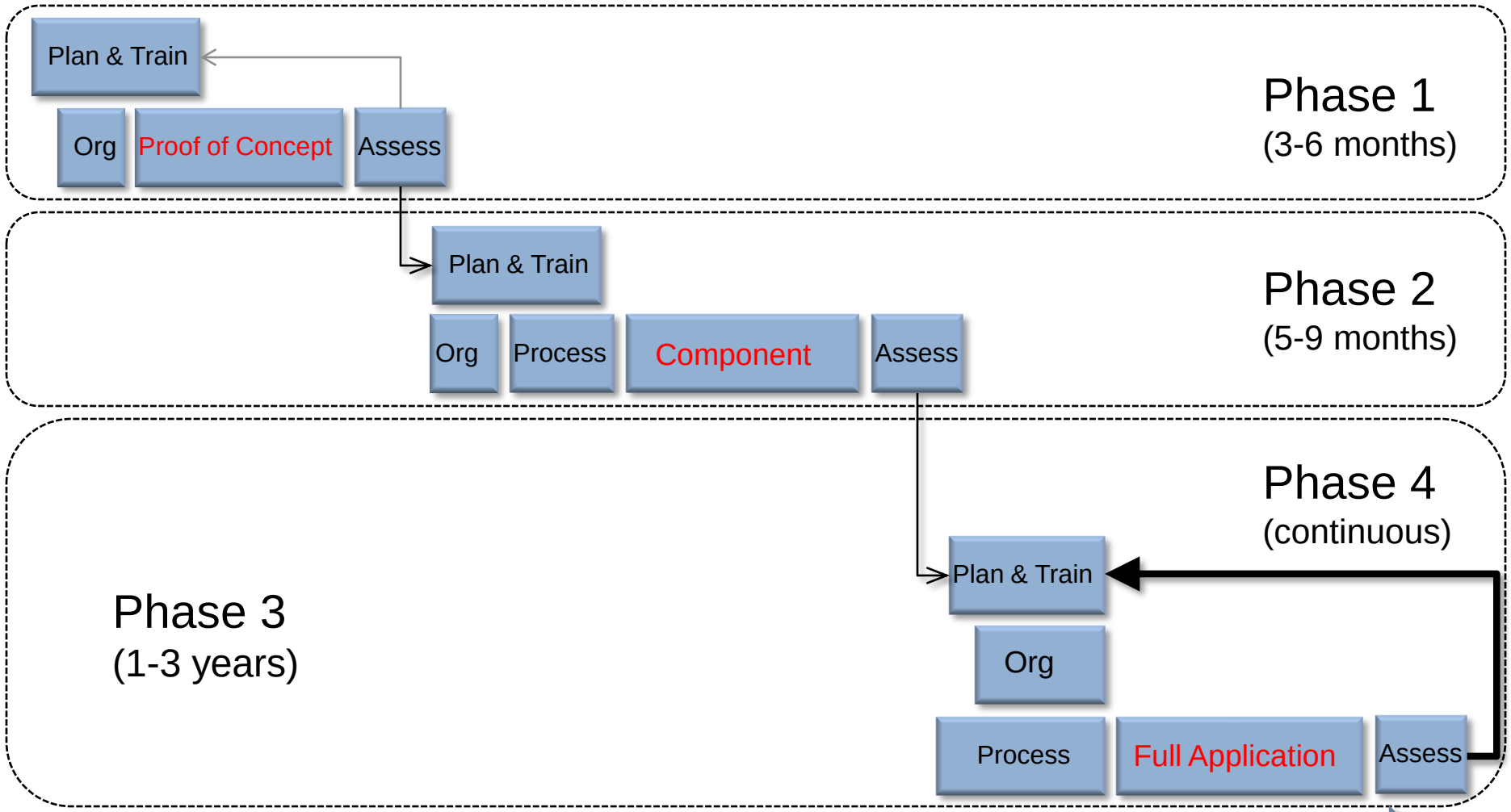
- A behavioral description of the embedded software—the physical objects and environment upon which it acts
- An executable specification that can be tested using simulation to ensure that it meets all functional requirements
- A tool that enables the design team to communicate information about the design
- A specification from which real-time software code can be automatically generated for testing, prototyping and embedded implementation
- Automatically generated code that can be used by itself or integrated with other code to form the complete embedded application

Model-Based Design enables design engineers 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 design time and development and implementation costs.

Phased Approach Leads to Success



Pragmatic Adoption of Model-Based Design



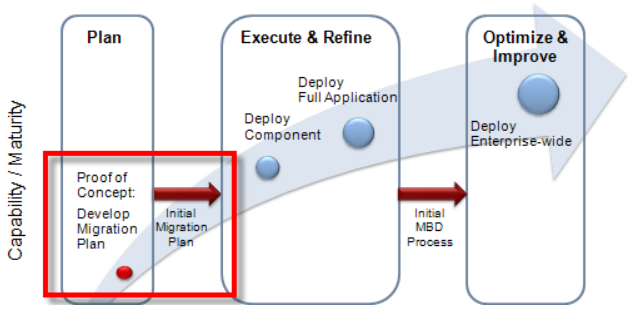
Plan & Train

Org | Proof of Concept | Assess

Phase 1

Theme: *Proof of Concept*

- Define objectives
- Get trained
- Develop the P.O.C. control algorithm
- Execute on the target
- Migration Plan

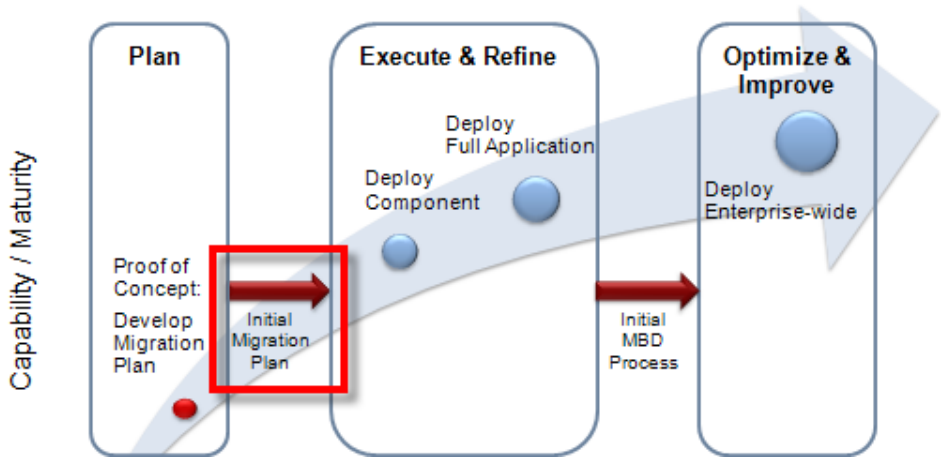


What does success look like:

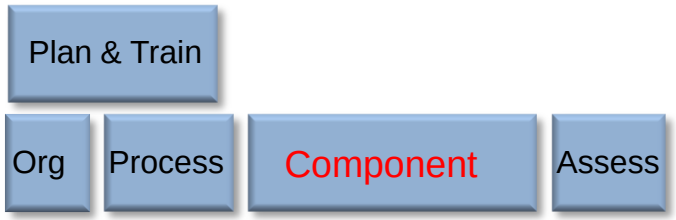
- Focus on technology – prove the tools can do the job
- Develop understanding of MBD Tools and Processes
- Build support for future changes
- KEY OUTPUT: Initial Migration Plan

The Migration Plan

- Objectives
- Metrics
- Organization
- Training
- Process Changes
- Constraints
- Standards



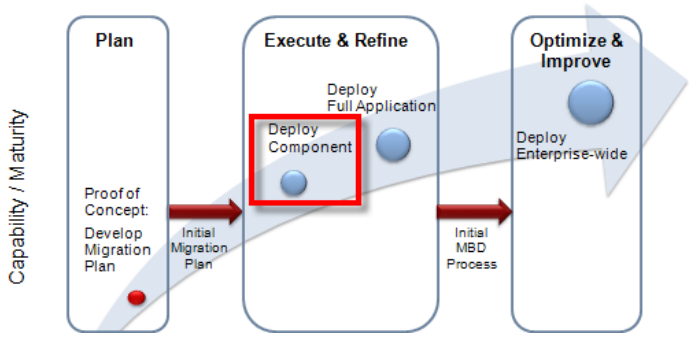
This plan will change – it is not static!



Phase 2

Theme: "*Component" Design*

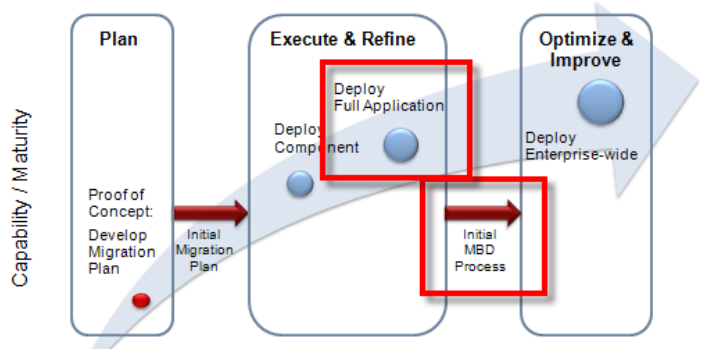
- Test and refine new capabilities
- Control risk



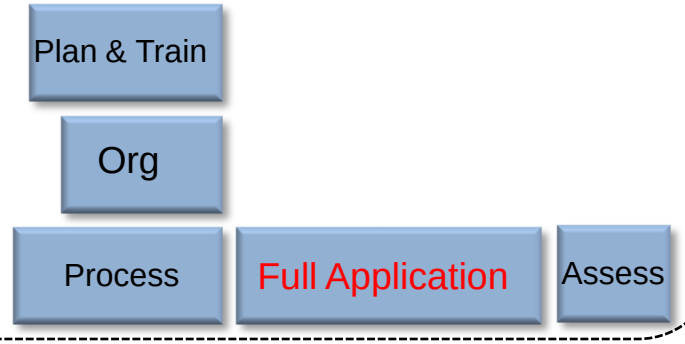
What does success look like:

- Larger number of people engaged in Model-Based Design
- Bigger model representing more functionality
- More than just modeling and code generation
- Increased automation
- Model-Based metrics and process definition
- KEY OUTPUTs:
  1. Production "component" delivered
  2. V1.0 Model-Based Process Definition

This should take 5-9 months depending on scale and scope



Phase 3



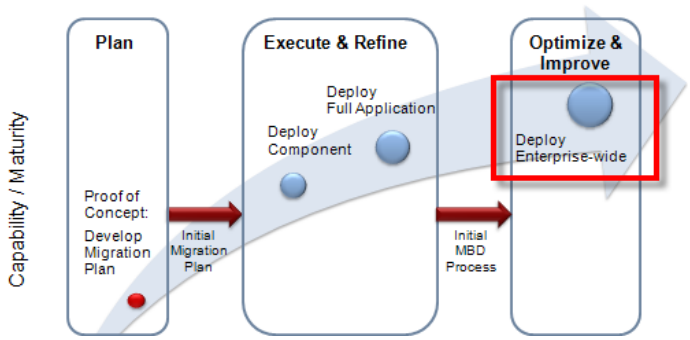
Theme: ***Full Application Design***

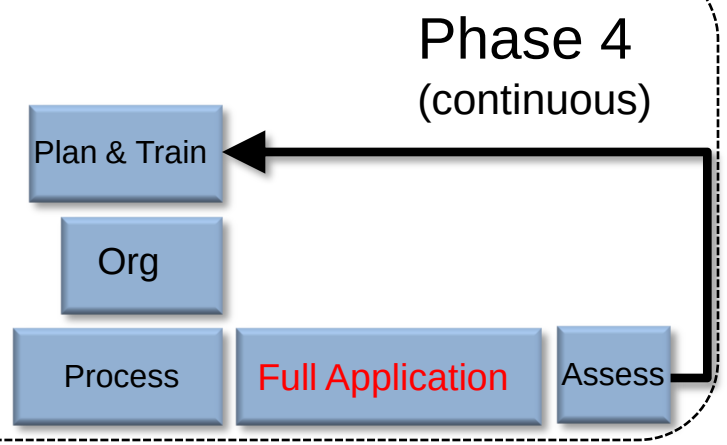
- Apply what was learned and model and automate code production for a full application – Scale up!
- Platform Software is not automated, but build process is.

What does success look like:

- Industrial grade process, tools and high quality product
- Significant return on investment
- KEY OUTPUTs:
  1. Production application delivered
  2. V2.0 Model-Based Process Definition – full spectrum

This should take 1-3 years depending on scale and scope





Improve & Replicate the Success

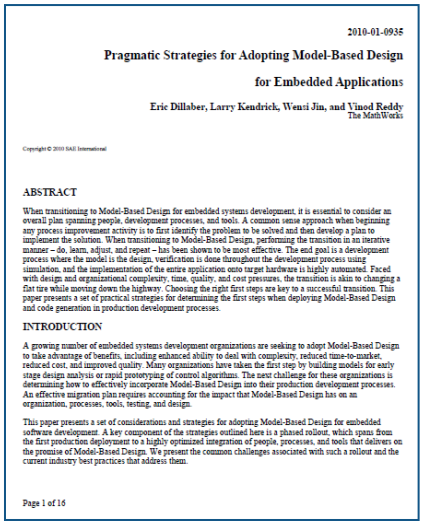
Theme: *Continuous Improvement*

- Adapt & Deploy Enterprise Wide
- Optimization

What does success look like:

- Replicated success at multiple sites
- Dramatic productivity improvement
- Increased capacity for complexity

Pragmatic Strategies for Adopting Model-Based Design (SAE Paper 2010-01-0935, Dillaber, Kendrick, Jin, Reddy)



2010-01-0935

Pragmatic Strategies for Adopting Model-Based Design for Embedded Applications

Eric Dillaber, Larry Kendrick, Wensi Jin, and Vinod Reddy
The MathWorks

ABSTRACT

When transitioning to Model-Based Design for embedded systems development, it is essential to consider an overall plan spanning people, development processes, and tools. A common sense approach when beginning any process improvement activity is to first identify the problem to be solved and then develop a plan to implement the solution. When transitioning to Model-Based Design, performing the transition in an iterative manner – do, learn, adjust, and repeat – has been shown to be most effective. The end goal is a development process where the model is the design, verification is done throughout the development process using simulation, and the implementation of the entire application onto target hardware is highly automated. Faced with design and organizational complexity, time, quality, and cost pressures, the transition is akin to changing a flat tire while moving down the highway. Choosing the right first steps are key to a successful transition. This paper presents a set of practical strategies for determining the first steps when deploying Model-Based Design and code generation in production development processes.

INTRODUCTION

A growing number of embedded systems development organizations are seeking to adopt Model-Based Design to take advantage of benefits, including enhanced ability to deal with complexity, reduced time-to-market, reduced cost, and improved quality. Many organizations have taken the first step by building models for early stage design analysis or rapid prototyping of control algorithms. The next challenge for these organizations is determining how to effectively incorporate Model-Based Design into their production development processes. An effective migration plan requires accounting for the impact that Model-Based Design has on an organization, processes, tools, testing, and design.

This paper presents a set of considerations and strategies for adopting Model-Based Design for embedded software development. A key component of the strategies outlined here is a phased rollout, which spans from the first production deployment to a highly optimized integration of people, processes, and tools that delivers on the promise of Model-Based Design. We present the common challenges associated with such a rollout and the current industry best practices that address them.

Assess organizational challenges and impact

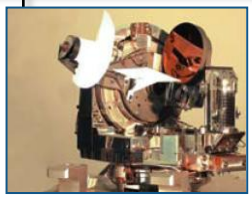
Plan for change

1. Identify the problem you are trying to solve
2. Choose a project with proper complexity and technology
3. Mitigate risk with a phased approach
4. Choose the appropriate legacy components for migration

Create a process and tool migration plan (key items below)

1. Use executable spec development as an opportunity to solidify requirements
2. Make the model a source for documentation
3. Choose architecture and component technology early
4. Establish and enforce design standards
5. Develop a plant model with “trend-correct” behavior
6. Verify what you need, not what you want
7. Migrate key supporting processes such as CM

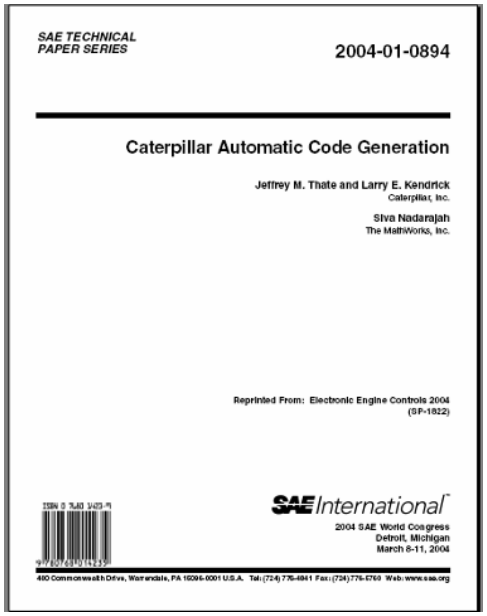
User Stories

Company	Application	Strategy	Result
Astrium	 First of its Kind Laser Link	• Modeling, Early Verification, Code Generation, HIL/RPC	• **Design iterations reduced from days to hours** • **Overall development time reduced by six months**
BAE Systems	 SDR	• Modeling, Early Verification, VHDL • Traditional Effort Comparison	• **Project development time reduced by 80%:** • **SDR SP Devel 10:1** • **Overall time 4:1**
Honeywell	 Flight Control System	• Modeling Early verification, code generation • Legacy Reuse	• **5:1 improvement in productivity** • **Highly accurate, reusable code** • **A superior product**
Lockheed Martin	 JSF - Flight Control System	• Modeling Early verification, code generation • Large-Scale & Collaborative Devel	• **Reduced Software Defects** • **Overall Reduction in Manhours/SLOC of ~40%**

Caterpillar

Phased Adoption of Model-Based Design and Code Generation

- Background
  - Needed to satisfy demands for increased software feature content, added complexity, and short turn-around time
- Results
  - Caterpillar uses MathWorks simulation, rapid prototyping, and code generation products as part of their production development capability
  - The data collected indicated a **reduction in person hours by a factor of 2 to 4** depending on the project and a **reduction of calendar time by a factor of greater than 2**



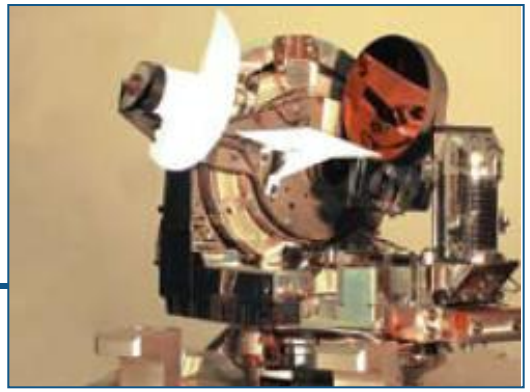
SAE Technical Paper 2004-01-0894

Thank You for Your Attention

Are there any questions?

Larry E. Kendrick, PhD **Senior Principle Technical Consultant**
The MathWorks, Inc.

Astrium Creates World's First Two-Way Laser Optical Link Between an Aircraft and a Communication Satellite



LOLA telescope assembly, as fitted to aircraft in Artemis laser link trials.

Challenge

To develop controls to ensure the precision of a laser optical link between an aircraft and a communication satellite

Solution

Use MathWorks tools to model control algorithms and pointing hardware, conduct hardware-in-the-loop tests, and deploy a real-time system for flight tests

Results

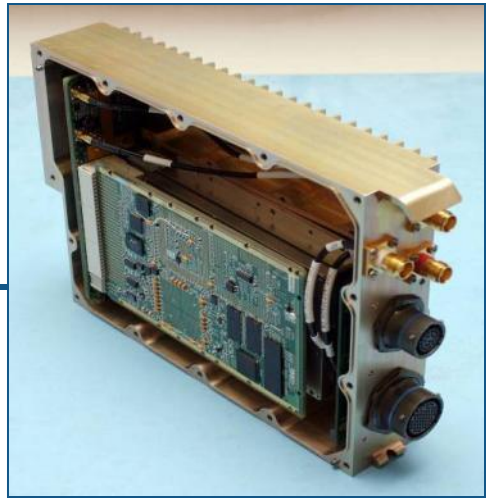
- First of its kind optical link demonstrated
- Design iterations reduced from days to hours
- Overall development time reduced by six months

"Using MathWorks tools for Model-Based Design, we simulated not only our control algorithms but also the physical hardware. By automatically generating code for the control software and the test bench, we reduced development time and implemented changes quickly. We visualized simulation and test results, which gave us confidence in the design we ultimately deployed."

David Gendre
Astrium

Link to user story

BAE Systems Achieves 80% Reduction in Software-Defined Radio Development Time with Model-Based Design



Custom board used in the traditional design workflow.

Challenge

To develop a military standard SDR waveform for satellite communications

Solution

Use Simulink and Xilinx System Generator to rapidly design, debug, and automatically generate code for an SDR signal processing chain

Results

- Project development time reduced by 80%
- Problems found and eliminated faster
- Clocking and interfacing simplified

"Using Simulink and Xilinx System Generator™ we designed and developed the signal processing chain of the SDR and achieved a 10-to-1 reduction in development time."

Dr. David Haessig
BAE Systems

Link to user story

Design Times at Honeywell Cut by 60%

Challenge

To update a flight control system while reducing development time and costs

Solution

Use design tools from The MathWorks to enable one team to design, model, and simulate the flight-control laws and automatically generate flight-ready code

Results

- A five-to-one improvement in productivity
- Highly accurate, reusable code
- A superior product

"[Using Simulink and Real-Time Workshop] we found we could do in half a day what previously took a week or more… It is pretty easy to see at least five-to-one improvement over the way we used to work."

Wayne King
Honeywell Commercial Aviation Systems

Link to user story

Flight Control Law Development for F-35 JSF