Pragmatic Strategies for Adopting Model-Based Design for Embedded Applications

Paul Smith – Director Consulting Services
The MathWorks, Inc.
Introduction

- What’s MBD?
- Why do it?
  - Make Models?
    - Make products!
  - Eliminate HW prototypes?
    - Minimize HW prototypes!
  - Build it right the first time?
    - Build it wrong a thousand times!
- 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
Phased Approach Leads to Success

Plan
- Proof of Concept: Develop Migration Plan

Execute & Refine
- Deploy Component
- Deploy Full Application

Optimize & Improve
- Initial MBD Process
- Deploy Enterprise-wide

Capability / Maturity

Research
Adv Engineering
Product Engineering Teams
Supplier Involvement
Time
Pragmatic Adoption of Model-Based Design

Phase 1
(3-6 months)

Phase 2
(5-9 months)

Phase 3
(1-3 years)

Phase 4
(continuous)

Plan & Train

Org

Proof of Concept

Assess

Plan & Train

Org

Process

Component

Assess

Plan & Train

Org

Process

Full Application

Assess

Time
Theme: *Proof of Concept*

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

What does success look like:

- Focus on technology – prove the tools can do the job
- Learn and build support for future changes
- **KEY OUTPUT**: Initial Migration Plan

*This should take 3-6 months, depending on scale and scope*
The Migration Plan

- Objectives
- Metrics
- Organization
- Training
- Process Changes
- Constraints
- Standards
- Automation
- Component Migration Strategy

This plan will change – it is not static!
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*
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
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

<table>
<thead>
<tr>
<th>Company</th>
<th>Application</th>
<th>Strategy</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Astrium</td>
<td></td>
<td>• Modeling, Early Verification, Code Generation, HIL/RPC</td>
<td>• Design iterations reduced from days to hours</td>
</tr>
<tr>
<td>BAE Systems</td>
<td></td>
<td>• Modeling, Early Verification, VHDL • Traditional Effort Comparison</td>
<td>• Project development time reduced by 80%:</td>
</tr>
<tr>
<td>Honeywell</td>
<td></td>
<td>• Modeling Early verification, code generation • Legacy Reuse</td>
<td>• 5:1 improvement in productivity • Highly accurate, reusable code</td>
</tr>
<tr>
<td>Lockheed Martin</td>
<td></td>
<td>• Modeling Early verification, code generation • Large-Scale &amp; Collaborative Devel</td>
<td>• Reduced Software Defects • Overall Reduction in Manhours/SLOC of ~40%</td>
</tr>
</tbody>
</table>
Astrium Creates World’s First Two-Way Laser Optical Link Between an Aircraft and a Communication Satellite

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

LOLA telescope assembly, as fitted to aircraft in Artemis laser link trials.
BAE Systems Achieves 80% Reduction in Software-Defined Radio Development Time with Model-Based Design

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
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