Model-Based Design for High-Integrity Software Development

Flight Software Workshop 2016

Matt Rhodes
Why did we miss our deadline?

Reasons for late projects, as reported by Venture Development Corporation.
Source: Embedded Software Strategic Market Intelligence report, Volume 4, December 2007, VDC.
Note: Percentages sum to over 100% due to multiple responses.
Minimize Costs by Detecting Errors Earlier

“…each delay in the detection and correction of a design problem makes it an order of magnitude more expensive to fix…”

Clive Maxfield and Kuhoo Goyal
“EDA: Where Electronics Begins”
TechBites Interactive, October 1, 2001
ISBN: 0971406308
Phase where errors are *found*

- **Requirements phase**
- **Design phase**
- **Coding phase**
- **Testing phase**

Relative number of errors detected

- Errors are typically found late in the process!
- Latent errors may remain in the software!

*Source:* How good is the software: A review of defect prevention techniques; Brad Clark, David Zubrow, Carnegie Mellon Software Engineering Institute; Software Engineering Symposium 2001
62% Cost Savings

- Total Savings: $3,720,000
- Total Investments: $592,000
- ROI: 528%

Bar chart showing costs and savings across different phases of development:

- Requirements: $1,500,000
- Design: $1,000,000
- Coding: $500,000
- Analysis: $500,000
- Testing: $1,000,000

Table:

<table>
<thead>
<tr>
<th>Phase</th>
<th>Costs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirements</td>
<td>$1,500,000</td>
</tr>
<tr>
<td>Design</td>
<td>$1,000,000</td>
</tr>
<tr>
<td>Coding</td>
<td>$500,000</td>
</tr>
<tr>
<td>Analysis</td>
<td>$500,000</td>
</tr>
<tr>
<td>Testing</td>
<td>$1,000,000</td>
</tr>
</tbody>
</table>
## Model-Based Design Maturity

**Requirements-Based V&V** (requirements-based algorithm development and testing, requirements modeling)

**Model Verification & Validation** (simulation-based analysis of requirements)

**System Verification & Validation** (re-use of requirements-based tests and results comparisons between simulation and real-time environments)

**Fully-leveraged Model-Based Design** (requirements-based design, development, and deployment of production hardware/software)

**System Simulation** (algorithm models and plant models)

**Modeling** (closed-loop simulation of algorithms and plant models)

**Prototyping** (real-time closed-loop system simulation with production algorithms on target hardware)

**Algorithm Modeling** (algorithm models, no plant models)

**Design Modeling** (open-loop simulation of algorithms)

**Design Prototyping** (real-time open-loop simulation of algorithms)

**Model-Based Programming** (automatic production code generation from algorithm models)

### Simulation

- **Model Verification & Validation**
- **System Verification & Validation**
- **Model-Based Development**

### Real-Time Testing

- **System Modeling**
- **System Prototyping**
- **Model-Based Programming**

### Production

- **Algorithm Modeling**
- **Design Modeling**
- **Design Prototyping**

### MAXIMIZE ROI

**MathWorks**

---

**Modeling & Simulation Adoption**

**Code Generation Adoption**
The Design V: Let’s Start from the Very Top
A Systems of Systems (SoS) is comprised of Operational Deliverables, e.g.:
- Ground Station
- Aircraft
- Communications Relay
- etc
An Operational Entity (e.g. an Aircraft) is comprised of Systems:
- Vehicle Management System
- Payload
- Electrical Power

A System (e.g. Vehicle Management) is comprised of Items:
- Flight Control Computer
- Sensors
- Actuators
Each Software Module is Embedded Object Code that satisfies the Requirement(s) allocated to that Module.
Model-Based Software Development Workflow (DO-178C)

- Requirements Validation
- Model Conformance
- Source Code Conformance
- Model Verification Compliance
- Source Code Verification Compliance
- Code on Target Compliance
Model-Based Software Development Workflow (DO-178C)

- Simulink Verification and Validation: Requirements Management Interface*
- Simulink
- Stateflow
- Simulink Report Generator*

Requirements
Validation

Model
Conformance

Model Verification
Compliance

Source Code
Verification
Compliance

Source Code
Conformance

Object Code

Trace
Trace
Trace
Confirm requirements map to design

Produce requirements trace reports

- **Identify missing requirements!**
- Design components that do not map to a requirement
- Indicative of incorrect design, or misinterpreting requirements
- Avoid finding unintended behaviors late in project phases

**Tool:** Simulink Verification and Validation
Modeling and simulation with Model-Based Design

Reference: http://www.flightgear.org/
Modeling and simulation with Model-Based Design

Reference: http://www.flightgear.org/
Model-Based Software Development Workflow (DO-178C)

- **Model Conformance**
- **Source Code Conformance**
- **Code on Target**

**Requirements Validation**

**Model Verification Compliance**

**Source Code Verification Compliance**

Object Code Verification

- Simulink: Model Advisor
- Simulink Verification and Validation: 
  *DO-178C/DO-331 Checks*
Check model complies with standards

Tool: Simulink Verification and Validation
Model-Based Software Development Workflow (DO-178C)

- Requirements Validation
- Model Conformance
- Source Code Conformance
- Simulink Design Verifier: Design Error Detection
Confirm design is free of implementation errors
Prove absence of dead logic, unreachable states

Tool: Simulink Design Verifier
Confirm design is free of implementation errors

Prove absence of overflow, divide by zero, and other design errors

Overflow proven *not* to occur (colored GREEN)

Overflow and divide by zero proven *not* to occur (colored GREEN)

Overflow proven to occur (colored RED)

Tool: Simulink Design Verifier
Model-Based Software Development Workflow (DO-178C)

- Requirements
- Model
- Source Code
- Code on Target

- Validation
- Conformance
- Compliance
- Trace

- Model Verification
- Source Code Verification

- Compliance
- Object Code

- Simulink Test*
- Simulink Verification and Validation: Model Coverage*
- Simulink Design Verifier: Property Proving
Execute functional tests in simulation

Time series test vectors

Pass / fail results

Tool: Simulink Test
Execute functional tests in simulation

Sequence based tests and assessments

Requirement → In the event of an failure condition of the position sensor or low hydraulic pressure, the Fault Detection Isolation and Recovery (FDIR) application shall select the primary actuator to isolated mode
Complement testing with formal methods

Graphical representation of safety requirements for thrust reverser

If average airspeed > 150 knots, deploy cannot be true

If two WOW sensors are false, deploy cannot be true

If either wheelspeed sensor < 10 knots, deploy cannot be true

Tool: Simulink Design Verifier
Counter example test generation

Tool: Simulink Design Verifier
Generate additional tests with formal methods

- Top off functional tests with generated tests
  - Use model checking to explore state space
  - Generate missing test-cases to execute all paths
  - Tests generated for decision, condition, MC/DC, table

Tool: Simulink Design Verifier, Simulink Test
Confirm design is fully tested

Tool: Simulink Verification and Validation, Simulink Test
Model-Based Software Development Workflow (DO-178C)

- **Requirements Validation**
- **Model Conformance**
- **Source Code Verification**
- **Object Code Verification**

- **Embedded Coder**
- **Simulink Code Inspector**

- Model
  - Requirements Validation
  - Model Conformance

- Source Code
  - Source Code Conformance
  - Source Code Verification
  - Embedded Coder

- Object Code
  - Object Code Verification
  - Compliance

- Code on Target
Model-Based Software Development Workflow (DO-178C)

- Requirements Validation
- Model Conformance
- Source Code Conformance
- Model Verification Compliance
- Source Code Verification Compliance
- Object Code Verification

• Polyspace Bug Finder
Model-Based Software Development Workflow (DO-178C)

- Requirements
  - Validation
  - Trace
- Model
  - Conformance
  - Trace
- Source Code
  - Conformance
  - Verification
  - Compliance
  - Trace
  - Object Code
    - Compliance
    - Object Code Verification
    - Simulink Code Inspector*
Model-Based Software Development Workflow (DO-178C)

- Requirements Validation
- Model Conformance
- Model Verification Compliance
- Source Code Conformance
- Source Code Verification Compliance
- Object Code Verification

• Polyspace Code Prover*
Verify robustness of integrated code
Polyspace formal methods static static analysis

Tool: Polyspace Code Prover

```c
static void pointer_arithmetic (void) {
    int array[100];
    int *p = array;
    int i;

    for (i = 0; i < 100; i++) {
        *p = 0;
        p++;
    }

    if (get_bus_status() > 0) {
        if (get_oil_pressure() > 0) {
            *p = 5;
        } else {
            i++;
        }
    }

    i = get_bus_status();

    if (i >= 0) {
        *(p - i) = 10;
    }
}
```
Model-Based Software Development Workflow (DO-178C)

- Embedded Coder: SIL/PIL
- Simulink Test*
- Simulink Verification and Validation: SIL/PIL Code Coverage*
Confirm design and code consistency

Execute all tests to check equivalence of model to code

Normal mode simulation model

Code generated from model, compiled on host

Test Vectors

Compare

Tool: Simulink Test
Examples of Successful Certifications with MathWorks Tools
Septentrio Streamlines DO-178B Certification with MATLAB and Simulink

Challenge
Obtain DO-178B certification for a GNSS-based landing system for precision aviation applications

Solution
Use Model-Based Design with MATLAB and Simulink to trace requirements, architect system components, simulate the design, and generate and verify source code

Results
- Design test cases reused on generated C source code
- Models verified via simulation, ensuring virtually bug-free code
- Key SOI-1 certification milestone achieved

“Model-Based Design enabled us to streamline the certification process by tracking requirements, verifying the design using simulation, and maintaining the system model as the single source of truth throughout development.”

Jan D'Espallier
Septentrio

Link to article
Eurocopter Accelerates Development of DO-178B Certified Software with Model-Based Design

**Challenge**
Speed the development, validation, and verification of DO-178B certified helicopter flight software

**Solution**
Use Model-Based Design to model the system design and software design, and to generate flight code

**Results**
- Software testing time cut by two-thirds
- Requirements stabilized earlier
- Certified flight software automatically generated

“We use our system design model in Simulink for ARP4754 to establish stable, objective requirements. We save time by using the model as the basis for our software design model for DO-178—from which we generate flight code—and reusing validation tests for software verification.”

Ronald Blanrue
Eurocopter
Airbus Develops Fuel Management System for the A380 Using Model-Based Design

Challenge
Develop a controller for the Airbus A380 fuel management system

Solution
Use MATLAB, Simulink, and Stateflow for Model-Based Design to model and simulate the control logic, communicate the functional specification, and accelerate the development of simulators

Results
- Months of development time eliminated
- Models reused throughout development
- Additional complexity handled without staff increases

“Model-Based Design gave us advanced visibility into the functional design of the system. We also completed requirements validation earlier than was previously possible and simulated multiple simultaneous component failures, so we know what will happen and have confidence that the control logic will manage it.”

Christopher Slack
Airbus

Airbus A380, the world’s largest commercial aircraft.
MathWorks can Help with Certification
The Certification Advisory Service will help you to:
- Leverage Model-Based Design and supporting tools to their fullest extent to maximize ROI
- Reduce duplicated and manual effort
- Avoid common pitfalls by providing proven best practices

Actual customer ROI:
- Up to 30% time savings of requirements review and verification
- Reduction of time-to-flight by 20%
- Automation of certification activities

MathWorks Consulting
Certification Advisory Service

Phase 1: Coaching and Instruction

Phase 2: Processes Gap Analysis and Improvement Plan

Phase 3: Hands-on deployment support

Service Details

- Present the best practices for each verification and traceability activity in the MBD workflow
- Each activity is covered via a PowerPoint slide deck
- Industry-specific demos supplement the slides

- Scope and objectives of Analysis are **jointly defined**
- Onsite interviews scheduled, **pre-analysis survey completed by participants**
- Gap Analysis: **onsite interviews** with participants
- Develop & **deliver gap closure recommendations**

- Help the customer apply the concepts, best practices and recommendations to **their application**
- Examples: modeling, simulation, code generation, establishing a model architecture, developing automation, integration with 3rd party tools, PIL