Verification and Validation of Models and Code

Presenter: Goran Begic
Technical Marketing
Goran.Begic@mathworks.com
Agenda

- Introductions
- Workflows for verification and validation
Introductions

- I spend most of my time:
  
  A. Creating specifications and requirements (systems and software)
  
  B. Implementation based on specification and requirements created by somebody else (generating / writing / deploying / debugging code)
  
  C. Other (including both, or none of the above)
Great Demo

- How much time do we need to get 100% MC/DC coverage?
Methods for Early Verification and Validation

- **Traceability**
  - Requirements to model and code
  - Model to code

- **Modeling and Coding Standards**
  - Modeling standards checking
  - Coding standards checking

- **Testing**
  - Model testing in simulation
  - Processor In the loop

- **Proving**
  - Proving design properties
  - Proving code correctness
Increasing Confidence In Your Designs

Verification Method

- Traceability
- Modeling and Coding Standards Checking
- Model and Code Testing
- Proving Design Properties and Code Correctness

Confidence
Traceability

- **Tracing Requirements ↔ Model**
  Simulink® Verification and Validation™

- **Tracing Model ↔ Source Code**
  Real-Time Workshop® Embedded Coder™

- **Tracing Requirements ↔ Source Code**
  Simulink Verification and Validation
Modeling and Coding Standards

- **Modeling Standards Checking**
  Simulink Verification and Validation

- **Coding Standards Checking**
  PolySpace™ Client™ for C/C++

Design

- Environment
- Physical Components
- Algorithms

Implement

- Hardware
  - Digital Electronics
    - VHDL, Verilog
  - Embedded Software
    - C, C++
  - FPGA, ASIC, MCU, DSP

Integration

- Hand-Generate
  - Generate

Modeling Standards

Coding Standards
Early Validation and Robustness Testing

Requirements

System V&V
Requirements Validation
Robustness Testing
Modeling Standards Checking

Design

- Environment
- Physical Components
- Algorithms

MathWorks Symposium
Adopting Model-Based Design within Aerospace and Defense
Component Testing

- Functional Requirements
- Design
- Verification
- Code
- Verification
- Environment
- Physical Components
- Algorithms
- Design
- Verification
- Integration
- Implement
- VHDL, Verilog
- C, C++
- FPGA, ASIC, MCU, DSP

MathWorks Symposium  Adopting Model-Based Design within Aerospace and Defense
Test Generation Workflow

Functional Requirements

Design
- Physical Components
- Environment
- Algorithms

Hand-Generate

Generate

Digital Electronics
- VHDL, Verilog
- FPGA, ASIC

Embedded Software
- C, C++

Generate

Code Generation

Analysis Model

Detailed models

Test Application

Component Source Code

Design Verification

Code Verification

Implement

Integration

Functional Requirements

MathWorks Symposium
Adopting Model-Based Design within Aerospace and Defense
Code Testing with Generated Signals

Simulink

- Software-in-the-loop
  - On the host
- Processor-in-the-loop
  - On the target processor

- Independent code testing environment
  - Generated signals and model outputs are saved as a .mat data file
  - Exported input signals drive code tests
  - Exported model outputs become expectation values for code testing
Proving

- Proving Design Properties
  Simulink Design Verifier
  Prove that design meets the key functional requirements

- Proving Code Correctness
  PolySpace™ Server for C/C++
  Prove that code meets non-functional runtime requirements
Code Correctness

Formal method: Abstract Interpretation

Results are proven for all possible executions of the code!!
Summary

- Model-Based Design enables early verification and validation!

- Early verification and validation methods improve and optimize your existing development process.

- Early problem detection significantly reduces time spent debugging – shorter time to resolution
Master Class Invitation

- Methods for Early Verification and Validation
  - Robustness Testing
  - Automatic Test Generation
  - Property Proving