MathWorks Vision for Systematic Verification and Validation

Bill Aldrich
Senior Development Manager
Simulink Verification and Validation, Simulink Design Verifier
Growing Complexity of Automotive Controls

- Emergency Braking
- Adaptive Cruise Control
- Stability Control
- Automatic Parking
- Battery Management
- Instrument Panel
- Infotainment
- Engine Management
- Transmission Control
- Propulsion Motor Control
- Airbag
- DC/DC Converter
- Navigation
- Forward Camera
- Adaptive Front Lighting
- HVAC Control
- Vehicle-to-Vehicle Communication
- E-Call
- Keyless Entry
- Short-Range Radar

- Body Control Module
- Voice Recognition
- Power Window
- Vehicle-to-Infrastructure
- Power Liftgate
- Power Seat
- Back-up Camera
- Long-Range Radar
- All-Wheel Drive
- Ultrasonic Sensor
- Tire Pressure Monitor
- 4-Wheel Steer
- Active Damping
- Smart Junction Box
- Stability Control

Growing Complexity of Automotive Controls

- Engine Management
- Transmission Control
- Forward Camera
- Electric Power Steering
- Smart Junction Box
- Battery Management
- Propulsion Motor Control
- DC/DC Converter
- Stability Control
- Infotainment
- Navigation
- Vehicle-to-Vehicle Communication
- HVAC Control
- Navigation
- Instrument Panel
- Airbag
- Forward Camera
- Adaptive Front Lighting
- Keyless Entry
- E-Call
- Short-Range Radar
- Adaptive Damping
- 4-Wheel Steer
- Active Damping
- Smart Junction Box
- 6 M Lines of Code
- 2-3 M Lines of Code
- 16 M Lines of Code

McKendrick, J. “Cars become ‘datacenters on wheels’, carmakers become software companies,” ZDNet, 2013
Growing Complexity of Automotive Controls

Growing Complexity of Automotive Controls

- Emergency Braking
- Body Control Module
- Voice Recognition
- Power Window
- Vehicle-to-Infrastructure
- Power Liftgate
- Power Seat
- Back-up Camera
- Long-Range Radar
- All-Wheel Drive
- Tire Pressure Monitor
- Ultrasonic Sensor
- Stability Control
- Adaptive Cruise Control
- Automatic Parking
- Smart Junction Box
- Battery Management
- Instrument Panel
- Airbag
- Electric Power Steering
- Engine Management
- DC/DC Converter
- Propulsion Motor Control
- Navigation
- Transmission Control
- Forward Camera
- Adaptive Front Lighting
- HVAC Control
- Vehicle-to-Vehicle
- E-Call
- Keyless Entry
- Short-Range Radar

Development Challenges
Development Challenges

- Representing complex systems
Development Challenges

- Representing complex systems
- Coordinating work across teams
Development Challenges

- Representing complex systems
- Coordinating work across teams
- Working efficiently
Development Challenges

- Representing complex systems
- Coordinating work across teams
- Working efficiently
- Ensuring quality
Traditional Development Process
Models for Specification

Textual Requirements → Executable Specification

C/C++ Hand code → Object code

Manual Coding → Compilation and Linking

Compilation and Linking
Model Abstraction – Work at an appropriate level of detail

- Simscape Fluids
- Simscape Multibody
- Simulink
- Stateflow
- MATLAB

% Predicted state and covariance
\[ \hat{x}_{\text{pred}} = A \cdot \hat{x}_{\text{est}} \]
\[ \Sigma_{\text{pred}} = A \cdot \Sigma_{\text{est}} \cdot A^T + Q \]

% Estimation
\[ \hat{x} = B \cdot \hat{x}_{\text{pred}} + \hat{y} \]
\[ \hat{p} = p_{\text{pred}} \]
\[ \Sigma_{\text{est}} = (I - B) \cdot \Sigma_{\text{pred}} \cdot (I - B)^T \]
\[ \text{kim_gain} = (I - B) \cdot \Sigma_{\text{pred}} \cdot (I - B)^T \]

% Estimated state and covariance
\[ \hat{x}_{\text{est}} = \hat{x}_{\text{pred}} + \text{kim_gain} \cdot (z - H \cdot \hat{x}_{\text{pred}}) \]
\[ \Sigma_{\text{est}} = \Sigma_{\text{pred}} - \text{kim_gain} \cdot H \cdot \Sigma_{\text{pred}} \]

% Compute the estimated measurements
\[ y = H \cdot \hat{x}_{\text{est}} \]
Complete Model Based Design Workflow, Concept to Code

- Textual Requirements
- Executable Specification
- Model used for production code generation
- Generated C/C++ code
- Object code
- Code Generation
- Compilation and Linking

Modelling
How do you ensure correctness?
Model-Based Design Maturity, Automotive Industry

- Modeling
- Simulation and Analysis
- Implementation
- Verification and Validation
- Process, Tools and Infrastructure
- Enterprise Management

Graph showing the maturity levels in various aspects of model-based design.
Model-Based Design Maturity, Automotive and Aerospace

Diagram showing maturity levels in modeling, simulation and analysis, implementation, verification and validation, enterprise management, and process, tools, and infrastructure.
Model Based Design Verification Workflow

- Textual Requirements
- Executable Specification
- Model used for production code generation
- Generated C/C++ code
- Object code

Modelling

Review and static analysis

Component and system testing

Equivalence checking

Equivalence testing

Code Generation

Compilation and Linking
Model Based Design Verification Workflow

- Perform simulation
  - Link and review requirements
  - Isolate and test components
  - Measure model coverage
  - Address missing coverage
  - Property proving

Component and system testing

Textual Requirements → Executable Specification → Model used for production code generation → Generated C/C++ code → Object code

- Modelling
- Code Generation
- Compilation and Linking
Ad-Hoc Simulation: Explore Behavior Virtually
Model Based Design Verification Workflow

- Perform simulation
- Link and review requirements
- **Isolate and test components**
- Measure model coverage
- Generate tests for missing coverage
- Manage and organize tests
- Property proving

Simulink Models

- Textual Requirements
- Executable Specification
- Model used for production code generation

Component and system testing

- Perform simulation
- Link and review requirements
- Isolate and test components
- Measure model coverage
- Generate tests for missing coverage
- Manage and organize tests
- Property proving

Modelling

Code Generation

Compilation and Linking

Generated C/C++ code

Object code
Test Harnesses

From any subsystem ...
Test Harnesses

From any subsystem ...

Isolate it with content it to drive inputs and analyze outputs

Simulate independently

Can be embedded in design model file.
Test Sequence Block

A test sequence block can drive inputs
A test sequence block can drive inputs and assess outputs
Test Sequence Block Syntax

Start speed = ramp(10); throttle = 25; verify(gear == 1);
Define Inputs

Start
speed = ramp(10);
throttle = 25;
verify(gear == 1);
Defining Pass/Fail Criteria

Start
speed = ramp(10);
throttle = 25;
verify(gear == 1);

Test Sequence

Pass

Fail

Untested

0 1
Model Coverage

Identify testing gaps:
- Untested switch positions
- Subsystems not executed
- Transitions not taken
- Many more …
Addressing Missing Coverage

Test Cases

Partial Coverage
Addressing Missing Coverage

Test Cases

Test Generator

Simulink Design Verifier

Partial Coverage
Addressing Missing Coverage

New Test Cases

Test Cases

Test Generator

Simulink Design Verifier

Partial Coverage

New Test Cases

Test Cases

Simulink Design Verifier
Addressing Missing Coverage

New Test Cases

Test Cases

Full Coverage
Model Based Design Verification Workflow

- Perform simulation
- Link and review requirements
- **Isolate and test components**
- Measure model coverage
- Generate tests for missing coverage
- Manage and organize tests
- Property proving

Component and system testing

Simulink Models

- Textual Requirements
- Executable Specification
- Model used for production code generation

Modelling → Code Generation → Compilation and Linking → Object code

Generated C/C++ code
Model Based Design Verification Workflow

- Manual review
- Standards compliance checking
- **Design error detection**
- Complexity analysis

Simulink Models

- Executable Specification
- Model used for production code generation
- Generated C/C++ code
- Object code

Textual Requirements → Modelling → Code Generation → Compilation and Linking
Detecting Hidden Run-Time Design Errors

- Integer overflow
- Division by zero
- Array out-of-bounds
- Range violations
- Dead Logic
Detecting Hidden Run-Time Design Errors
Detecting Hidden Run-Time Design Errors
Model Based Design Verification Workflow

- Perform SIL Testing
- **Measure code coverage**
- Verify code with Polyspace
- Verify consistency with Simulink Code Inspector

Textual Requirements → Executable Specification → Model used for production code generation → Generated C/C++ code → Object code

Modelling → Model used for production code generation

Equivalence checking

Compilation and Linking → Object code
Coverage for Generated Code (R2016a)

`cruise_control` (SIL)
Coverage for Generated Code (R2016a)
Coverage for Generated Code (R2016a)

```matlab
77 if (rtb_ActiveControl) {
78     /* Sum: '<S/2>/Sum' incorporates:
79         * DiscreteIntegrator: '<S>/Discrete-Time Integrator'
80         * Gain: '<S>/Gain'
81         * Gain: '<S>/Gain1'
82         */
83     rty_throt = 0.02 * rtb_Switch2 + 0.01 *
84     localDW->DiscreteTimeIntegrator_DSTATE;
85 
86     /* Update for DiscreteIntegrator: '<S>/Discrete-Time Integrator' */
87     localDW->DiscreteTimeIntegrator_DSTATE += 0.01 * rtb_Switch2;
88     if (localDW->DiscreteTimeIntegrator_DSTATE >= 5.0) {
89         localDW->DiscreteTimeIntegrator_DSTATE = 5.0;
90     } else {
91         if (localDW->DiscreteTimeIntegrator_DSTATE <= -5.0) {
92             localDW->DiscreteTimeIntegrator_DSTATE = -5.0;
93         }
94     }
```
Coverage for Generated Code (R2016a)

Press Play

cruise_control (SIL)

Can also be highlighted on model
Model Based Design Verification Workflow

- Perform PIL Testing
- Perform HIL Testing

Equivalence testing

Textual Requirements

Executable Specification

Model used for production code generation

Generated C/C++ code

Object code

Modelling

Code Generation

Compilation and Linking
Model Based Design Verification Workflow

- Textual Requirements
- Executable Specification
- Model used for production code generation
- Generated C/C++ code
- Object code

**Simulink Models**

- Modelling
- Review and static analysis
- Equivalence testing
- Equivalence checking

**Process Steps:**
1. Component and system testing
2. Modelling
3. Review and static analysis
4. Equivalence testing
5. Equivalence checking
6. Code Generation
7. Compilation and Linking
Model Based Design Verification Workflow
Systematic Verification

- Ensure that verification is systematically performed across:
  - All requirements
  - Complete model structure
  - Complete code structure
  - All design behaviors
Test and Verification

- Essential
- Expensive
- Complex

Pain Points
Test and Verification

- Essential → More Complete
- Expensive → Faster
- Complex → Simpler
Thank You!