From Design to Production

An integrated approach

Paolo Fabbri
Senior Engineer
Do you know what it is?

- Requirements
- Functional Spec
- Detailed Design
- Implementation
- System Test
- Integration Test
- Unit Test
Is the industry still using this approach for Embedded Software Development?
For Sure! But it’s more about Modeling and Simulation

- Requirements
- Model Design
- Model V&V
- Code Generation
- Code V&V
- HIL Simulation
- System Test
Case Study: Vehicle Speed Limiter

![Image of speed limiter graphic]

- **Actual speed**
- **Setpoint**

### Graph Details
- **X**: 10.9
- **Y**: 71.1

### Diagram
- **Requirements**
- **System Test**
- **HIL Simulation**
- **Model Design**
- **Model V&V**
- **Code V&V**
- **Code Generation**

---

**MATLAB EXPO 2014**
Project Management in Simulink
Simulink Project

Collaborative development

Source Control Integration

- Support for \texttt{svn} and \texttt{git}
- Model compare/merge
- Data compare/merge

Files Management

Dependencies Analysis

Impact Analysis
Source Control Integration
Data Dictionary Comparison
Dependency and Impact Analysis
Manage Complexity in Simulink
Typical challenges with large projects

Reuse models among different teams and projects

Manage multiple design options in a single-model workflow

Share models outside the team protecting IP

Ensure proper modeling rules and configuration

Ensure data consistency across models

...What else?
Component-Based Modeling

Reuse models among different teams and projects

- Modular Architecture
- Reusable Models
- Variants Management
- IP Protection
Model and Subsystem Variants
Protected Models for IP Protection

Read-Only View Support
Simulation Support
Code Generation Support
Model Advisor Technology

Ensure proper modeling rules and configuration

Modeling Standards

- MISRA-C:2004
- MAAB Style Guidelines
- IEC-61508/ISO-26262
- EN-50128
- DO-178C/DO-331

Specialized Advisor

- Upgrade Advisor
- Performance Advisor
- Code Generation Advisor
Simulink Data Dictionary

Ensure data consistency across models

Design Data

Configuration Sets

Change tracking

Data partitioning

Scalability and performance

Simulink Projects Integration
Simulink Data Management System

- Simulink Project
- Model Explorer
- Simulink Data Dictionary
- Simulink Data Objects
Model Verification and Validation
Early Model Verification and Validation

*Simulation, simulation, simulation...*

Simulation Data Inspection

Simulation Comparison

Coverage Analysis
Analyze Simulation Data with Simulation Data Inspector
Simulation Comparison
Model Coverage Analysis

Details:

1. Model "sf_car"

- Child Systems: Engine, shift_logic, transmission

<table>
<thead>
<tr>
<th>Metric</th>
<th>Coverage (this object)</th>
<th>Coverage (inc. descendants)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cyclomatic Complexity</td>
<td>1</td>
<td>36</td>
</tr>
<tr>
<td>Condition (C1)</td>
<td>NA</td>
<td>67% (8/12) condition outcomes</td>
</tr>
<tr>
<td>Decision (D1)</td>
<td>NA</td>
<td>75% (33/44) decision outcomes</td>
</tr>
<tr>
<td>MCDC (C1)</td>
<td>NA</td>
<td>33% (2/6) conditions reversed the outcome</td>
</tr>
<tr>
<td>Look-up Table</td>
<td>NA</td>
<td>36% (8/24) interpolation/extrapolation intervals</td>
</tr>
</tbody>
</table>

Decision 82% (18/22)  Condition 67% (8/12)  MCDC 33% (2/6)
Embedded Code

Generation and Verification
Embedded Code Generation

Unified Code Generation

Objective-Based Workflow

Code Report and Code Metrics
Unified Code Generation Capabilities

function [symbols, weights] = gaintrlsigng(spec, tools);
% L-tap adaptive equalizer using LM or HDL algorithms.

% equalizer settings
L = 3; % number of taps
Delta = 0.1; % equalizer step size

for a = 1:length(spec);
    % received sample
    y = spec{a};
    % weights = 0;
    if y
        Delta = 0.1;
        a = 0;
    elseif
        d = detect_corrected(y) + L * detect_corrected(y);
        a = 1;
        Delta = Delta + u_corrected(y);
        weights = weights + Delta * u_corrected(y);
        symbol(a) = y;
end

MATLAB

Unified Code Generation

C Code
C++ Code
HDL Code
PLC Code
Algorithm Export

Communication Interfaces

Sensors

Controller Model

Generated Algorithm Code

Input Drivers

Input Device Drivers

Included Target Optimized Code

Included Legacy Code

Output Drivers

Special Device Drivers

Scheduler/Operating System and Support Utilities

Actuators

Special Interfaces

M

RGBSplit-4

V RCS
Full Executable
## Code Integration Approaches

<table>
<thead>
<tr>
<th>Purpose</th>
<th>Algorithm Export</th>
<th>Full Executable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Purpose</td>
<td>Mass Production</td>
<td>On-Target Rapid Prototyping Low Volume Production</td>
</tr>
<tr>
<td>Usage</td>
<td>Systems and software engineers</td>
<td>Systems engineers</td>
</tr>
<tr>
<td>Hardware</td>
<td>Any processor</td>
<td>Specific Hardware Kits or Boards</td>
</tr>
<tr>
<td>Flexibility</td>
<td>Higher</td>
<td>Lower</td>
</tr>
<tr>
<td>Ease of Use (Turnkey)</td>
<td>Works after one-time customization</td>
<td>Works out-of-the-box</td>
</tr>
</tbody>
</table>
Objective-Based Code Generation
Code Report and Code Metrics

2. Global Variables [hide]

Global variables defined in the generated code.

<table>
<thead>
<tr>
<th>Global Variable</th>
<th>Size (bytes)</th>
<th>Reads / Writes</th>
<th>Reads / Writes in a Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>tracking_MdlrefDWork</td>
<td>14</td>
<td>22</td>
<td>22</td>
</tr>
<tr>
<td>Total</td>
<td>14</td>
<td>22</td>
<td></td>
</tr>
</tbody>
</table>
Embedded C Code Verification

“In-the-Loop” Simulations
Code Execution Profiling
Source Level Debugging
Code Coverage Analysis
Run-Time Error Detection
«In-the-Loop» Simulations
MIL/SIL/PIL Simulation Comparison
Code Execution Profiling

The code execution profiling report provides metrics based on data collected from a SIL or PIL execution. Execution times are calculated from data recorded by instrumentation probes added to the SIL or PIL test harness or inside the code generated for each component. See Code Execution Profiling for more information.

1. Summary
   - Total time (seconds = 1e-09) 1988.115
   - Measured time display options ('Units', 'Seconds', 'ScaleFactor', '1e-09', 'NumericFormat', '%.0.0f')
   - Timer frequency (ticks per second) 2.000e+00
   - Profiling data created 19-Oct-2014 16:48:00

2. Profiled Sections of Code

<table>
<thead>
<tr>
<th>Section</th>
<th>Maximum Execution Time</th>
<th>Average Execution Time</th>
<th>Maximum Self Time</th>
<th>Average Self Time</th>
<th>Calls</th>
</tr>
</thead>
<tbody>
<tr>
<td>speedcontrol</td>
<td>6895.74</td>
<td>6895.74</td>
<td>6895.74</td>
<td>6895.74</td>
<td>1</td>
</tr>
<tr>
<td>speedcontrol</td>
<td>57604</td>
<td>637</td>
<td>57604</td>
<td>637</td>
<td>30003</td>
</tr>
</tbody>
</table>
Source Level Debugging in SIL
Code Coverage Analysis

Bullseye Coverage code coverage enabled

Function: 100%  
Condition/decision: 70%

1 /* File: speedcontrol.c */
2
3 switch (pCtrlC1Mode)
4 {
5     case 0:
6         case 1:
7             case 2:
8             break;
9 
10     case 3:
11         /* Switch: '<S107>/Switch' incorporates: */
12         /* Constant: '<S85>/Parameter' */
13         /* Relational Operator: '<S107>/Relational Operator' */
14         /* Sum: '<S97>/Add' */
15         /
16         if (20.0F >= (FLOAT_Switch - 1.0F))
17         {
18             FLOAT_Switch = 20.0F;
19         }
20         else
21         {
22             FLOAT_Switch--; 
23         }
24         /* End of Switch: '<S107>/Switch' */
25         break;
26 
27     case 4:
28         FLOAT_Switch = FLOAT_Add_nrwn;
29         break;
30 
31     case 5:
32         /* Switch: '<S105>/Switch' incorporates: */
33         /* Constant: '<S87>/Parameter' */
34         /* Relational Operator: '<S105>/Relational Operator' */
35         /* Sum: '<S97>/Add' */
Polyspace Technology

- Finds bugs
- Checks coding rule conformance (MISRA/JSF/Custom)
- Provides metrics (Cyclomatic complexity etc)
- Proves the existence and absence of run-time errors
- Certification help for Functional Safety standards
Run-time checks proven by Code Prover

C run-time checks

- Unreachable Code
- Out of Bounds Array Index
- Division by Zero
- Non-Initialized Variable
- Scalar and Float Overflow (left shift on signed variables, float underflow versus values near zero)
- Initialized Return Value
- Shift Operations (shift amount in 0..31/0..63, left operand of left shift is negative)
- Illegal Dereferenced Pointer (illegal pointer access to variable of structure field, pointer within bounds)
- Correctness Condition (array conversion must not extend range, function pointer does not point to a valid function)
- Non-Initialized Pointer
- User Assertion
- Non-Termination of Call (non-termination of calls and loops, arithmetic expressions)
- Known Non-Termination of Call
- Non-Termination of Loop
- Standard Library Function Call
- Absolute Address
- Inspection Points
That’s Model-Based Design

Requirements

Model Design

Model V&V

Code V&V

HIL Simulation

Code Generation

System Test
What is the current industry maturity in the adoption of Model-Based Design?
Model-Based Design Maturity Framework
Profile of the industries in the benchmark

- **OEMs and Suppliers from all regions of the world**

- **Automotive:**
  - Passenger, Commercial, Off-Highway

- **Aerospace:**
  - Commercial, Military, Space

- **Other:**
  - Industrial Automation, Medical, Transportation, Electronics
Automotive vs. Aerospace: Leaders
Thank you!