Model Based Software Development for DDG 1000 Advanced Gun System

Dirk Jungquist
Advanced Gun System on the Zumwalt DDG 1000

AGS Key Features
155 mm / 62 caliber vertical load mount
10 rounds per min continuous rate-of-fire
(2) 300 round magazines
Replenishment at-sea or in-port
Fully-automated unmanned operation
AGS Overview

- Advanced Gun System (AGS) is a 155 mm vertical load gun employing a fully automated magazine and gun loading system
  - 84 electric brushless DC drives
  - Largest motor - 250 HP
  - 850 KW Peak power draw per gun mount
  - 4 VME Chassis with 15 Single Board Computers
  - 100 lb cased propellant
  - 220 lb, 14.5 cal (88”) LRLAP
Application Trade Study

- Trade criteria of hand code versus model based design (Matlab/Simulink)
  - Cost / Risk / Performance
- Model Based Design Selected:
  - Control systems
  - Logic diagrams / state machines
  - Sensor filtering
  - Sequencing, BIT, fault handling
- Hand Code Selected:
  - Queuing problems
  - Linked lists
  - Database applications
  - Operating system interface
  - Hardware drivers

<table>
<thead>
<tr>
<th>Costs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Safety critical code compliance</td>
</tr>
<tr>
<td>ODP process compliance</td>
</tr>
<tr>
<td>Design</td>
</tr>
<tr>
<td>Design Development Effort</td>
</tr>
<tr>
<td>Design review / verification</td>
</tr>
<tr>
<td>Requirements traceability</td>
</tr>
<tr>
<td>Configuration management</td>
</tr>
<tr>
<td>SW Test</td>
</tr>
<tr>
<td>Unit level code &amp; test</td>
</tr>
<tr>
<td>SCI level test (host)</td>
</tr>
<tr>
<td>Test vector development</td>
</tr>
<tr>
<td>HW/SW Integraton</td>
</tr>
<tr>
<td>Integration against emulated environment</td>
</tr>
<tr>
<td>Integration test (HW/SW integration)</td>
</tr>
<tr>
<td>Maintenance (LCC)</td>
</tr>
<tr>
<td>Training / Tools (Resource availability)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Risk</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ability to meet schedule</td>
</tr>
<tr>
<td>Ability to meet cost</td>
</tr>
<tr>
<td>Technical performance (Design intent implementation (reqmts interpretation))</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Performance (of end product)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reliability</td>
</tr>
</tbody>
</table>
Implementation Trade Studies

- Mix hand code and auto generated code?
  - Selected a separation where all the application functions on the processor are in the model
  - Simplified team organization and integration

- Interface directly to operating system?
  - Chose to use a OE layer which separates the application from the operating system

- Embed hand code into models?
  - Not required based on application requirements
  - Limit to interfaces

- Interface generated code only through a top level generated subroutine call?
  - Used direct OE calls through s-functions
  - Top level model called by frequency based scheduler
Software Safety Implications

• The AGS is the first weapon system for which the U.S. Navy Weapon System Explosives Safety Review Board (WSESRB) Software Systems Safety Technical Review Panel (SSSTRP) gave concurrence for using a model based software development approach
  – Several possible approaches evaluated including certified code generators
  – To meet safety expectations a software safety process is required

• Qualification approach chosen with native Matlab code generator
  – Code generator treated like a new compiler
  – Limited set of base library blocks used
  – Usage enforced through automated script checking
  – Style guide checks also enforced through automatic scripts
Matlab Autocode Qualification Process Steps

1. **Identify “Core” Building Blocks**
   - Only those building blocks necessary for AGS.

2. **Unit Test**
   - Complete Model Coverage
     - No Dead Code
     - All Logic Tested

3. **Peer Review**
   - Review “Core” Block Construction
     - Verify Safe Construction
     - Eliminate Unnecessary Blocks
   - Confirm Coverage and Results

4. **Code Generation Inspection**
   - Code Generate Models and Test Vectors
   - Inspect Code
     - Tailored AGS Coding Standards
     - Verify Generated Code Correctness
     - No Dead Code Generated

5. **Target Compile And Execute**
   - Compile Generated Code On Target
   - Execute Test Results

6. **Validate Test Results**
   - Validate results with those from graphical model-base testing.
Product Development Flow

- **Concept**
  - Modeling
    - Performance, Concept
    - Sensors, Electronics
    - Pro-E, Dynamics
  - Concept Model
    - Pro-E
    - Dynamics
    - Sensors, Electronics
    - Environment
    - M&S VIL
    - Model Verification
    - Prototype - HIL
    - Production
  - Simulation Timeline, Error Budgets
  - Analysis
    - Performance, Controllability
  - Simulation & Test
    - Non-realtime
  - Automatic Code Generation
  - Computing/Implementation Environment

**Flowchart:**
- **Simulation SIL – Emulation**
- **Prototype - HIL**
- **Production**
Development Artifacts

- Generic model library components
- Unit level tests
  - Model based with execution in simulation and on target processor
  - Coverage analysis
  - Peer review with SME
  - Automatic script checks for block usage and style guide
  - Code review if safety critical
- Independent test team
  - Integration functional testing against requirements
- Configuration management in Clearcase
  - Models, unit tests, coverage analysis
Benefits of Model Based Design

• Subject matter expert direct implementation
  – Less detailed requirement decomposition
  – No algorithm translation issues

• Subject matter expert review of graphical logic
  – Design accessible by system, electrical and mechanical engineers

• Powerful automation tools available
  – Model style checking, unit test development, coverage analysis, regression testing and code generation

• Cost and schedule savings
  – Prototype applications five times less costly than traditional hand code
  – AGS objective production code actual costs were two times less than traditional hand code estimates
    • Improved performance expected as processes mature
Advantages of Graphical Logic

Off
CommStatus = COMMS_OFF;
Initialized = FALSE;

[PowerOn == TRUE] [PowerOn ~= TRUE]

Operating

[Initialized == TRUE]

Initializing
CommStatus = COMMS_INITIALIZING;

[after(SerialOnTicks, tick)]
{Initialized = TRUE;}

[SerialDataValid == FALSE]

Up
CommStatus = COMMS_UP;

[SerialDataValid == FALSE]

[SerialDataValid == TRUE]

Failed
CommStatus = COMMS_FAILED;

BIT
CommStatus = COMMS_STATUS_UNKNOWN;

[BIT_InProgress == TRUE] [BIT_InProgress ~= TRUE]
Challenges to a Model Based Design

- Estimating project cost
  - Functional based estimates
  - Calibrating SEER-SEM
  - Scaling for project size
- Reporting development progress
  - SLOC is meaningless metric
- Artifacts for independent test teams
  - Design can be performed with higher level requirements
  - Specific documents only for testing may be required
Contact Info

- Dirk Jungquist
- BAE Systems – Land and Armaments
- 763-572-5312
- Dirk.Jungquist@baesystems.com