A Model-Based Design & Testing Approach for Orion GN&C Flight Software Development

Joel Henry NASA-JSC
This presentation is based on the following conference papers:

- A Model-Based Design & Testing Approach for Orion GN&C Flight Software Development
  - 2010 IEEE Aerospace Conference, paper #1491
- Orion GN&C Model Based Development: Experience and Lessons Learned
  - 2012 AIAA GN&C Conference, paper #2012-5036
Overview

- Introduction
- GN&C FSW Development Process
- GN&C Architecture
- Development Tools
- Inspection Process and CSU Memo
- Unit Testing for Models and Autocode
- Rhapsody Architecture
- Process Benefits, Lessons Learned & Challenges
Introduction: Orion GN&C Subsystem

Orion GN&C flight software developed using Model Based Design (MBD)

- Complex FSW executes GN&C for multiple mission phases (Ascent, Orbit, Entry)
  - Interface with multiple sensors, effectors, and crew
  - GN&C FSW executes in an ARINC 653 partition

Original design was for LEO and Lunar Operations (Scaled Back for EFT-1)

Early in the Orion program Matlab/Simulink were selected as GN&C development Tool
Orion will be the first human spacecraft built by NASA in 3 decades

New flight-test based approach is now being used, so first mission for GN&C software on Orion avionics will be Exploration Flight Test One (EFT-1)

- Previous Pad Abort One flight test also used MBD but a somewhat different process
- The EFT-1 mission includes an elliptical orbit designed to increase entry velocities to test thermal components – commercial booster used for launch system
- FSW modes for EFT-1 include: Pad align, ascent navigation, orbit coast, CM translation burn, guided direct entry, drogue rate damping, touchdown roll control
Development Process: Traditional vs Model Based Development

“Traditional” GN&C FSW Development

Orion GN&C FSW Development

Modeling in Matlab/Simulink is the “executable specification”, and essentially replaces the “FSSR”
Development Process: GN&C/FSW Team Interface

- Algorithms developed as Computer Software Units (CSU) using Simulink
- Integrated and matured using RAMSES-M and tested using RAMSES-M and RAMSES-A

After iterating and maturing algorithms in the “design loop” the autocoded CSU’s were delivered to GN&C FSW, where they were integrated into the ARINC 653 GN&C partition

- The GN&C partition may be executed either on target hardware (real-time only) or by using software emulators. Partition development and test is referred to as the “production loop”
- Problems encountered during hardware integration that affect the Simulink models are fed back to the GN&C team for rapid fixes. No manual modification of the autocode is allowed.
• CSU’s are collected into rate groups (1 Hz, 5 Hz, 20 Hz and 40 Hz) and then into domains (guidance, navigation, control) – this simplifies the emulation of rate group interaction within Simulink
• Figure shows the top level RAMSES-M diagram with each rate group
GN&C Architecture

- GNC design is a hierarchical decomposition of flight software by flight phase & function
- Centralized GNC Executive coordinates via GNC Activities
- Data Driven Lists, Modes, Configs

<table>
<thead>
<tr>
<th>GN&amp;C Domain</th>
<th>GN&amp;C Domain Functionality</th>
</tr>
</thead>
<tbody>
<tr>
<td>GCI</td>
<td>GN&amp;C Command Interface</td>
</tr>
<tr>
<td>NVA</td>
<td>Absolute Navigation</td>
</tr>
<tr>
<td>NVR</td>
<td>Relative Navigation (Rendezvous)</td>
</tr>
<tr>
<td>NVE</td>
<td>Ephemeris Processing</td>
</tr>
<tr>
<td>NHM</td>
<td>Navigation Health Manager</td>
</tr>
<tr>
<td>GMP</td>
<td>Vehicle Mass Properties</td>
</tr>
<tr>
<td>GDA</td>
<td>Ascent Guidance</td>
</tr>
<tr>
<td>GDE</td>
<td>Entry Guidance</td>
</tr>
<tr>
<td>GDO</td>
<td>On-Orbit Guidance</td>
</tr>
<tr>
<td>GHM</td>
<td>Guidance Health Manager</td>
</tr>
<tr>
<td>CNC</td>
<td>Command-Module (CM) Control (Entry)</td>
</tr>
<tr>
<td>CNS</td>
<td>Service-Module (SM) Control (Ascent, Orbit)</td>
</tr>
<tr>
<td>CNL</td>
<td>Launch Abort System (LAS) Control (Ascent)</td>
</tr>
<tr>
<td>CNE</td>
<td>Propulsion Engine Control</td>
</tr>
<tr>
<td>CNP</td>
<td>Propulsion Systems Control</td>
</tr>
<tr>
<td>CHM</td>
<td>Control Health Manager</td>
</tr>
</tbody>
</table>
Working, well understood legacy simulations, together with the desire/requirement for autocode led to a hybrid tool set.

Termed The Rapid Algorithm MATLAB/Simulink Engineering Simulation (RAMSES), the hybrid environment included two variants:

- RAMSES-M executed the GN&C algorithms in the native MATLAB process
- RAMSES-A executed the autocoded algorithms in a hand coded wrapper for higher speed execution and Monte Carlo Analysis

Legacy 6 DOF simulations were attached to MATLAB process for Simulink algorithm development, debugging and test.
Development Tools: GN&C CSU’s and Domains

- GNC CSU’s are expressed as Simulink diagrams housed within model reference blocks (MRBs)
  - MRBs allow CSU’s to be housed in separate files for configuration management – simple subsystems would mean that all changes are made to a single file.

- Each CSU has 4 interfaces which are expressed as Simulink bus types:
  - Inputs – time varying signals from upstream CSU’s or sensors
  - Parameters – static values that are initialized upon SW load. Some parameters may be changed on events by the automation and sequencing software.
  - Outputs – signals produced for consumption by downstream CSU’s or effectors
  - Telemetry – items needed for analysis of internal functioning

- Junction boxes pick off output signals from upstream CSUs or parameter buses

- CSU’s are unit tested with drivers that populate inputs and parameters and compare outputs.
• **Modeling library and template**
  – All Simulink atomic-level blocks were reproduced in an Orion library to provide control over autocode configurations, standards on settings, etc. This is highly recommended for serious MBD projects
  – Orion developed a template for all CSU’s to provide uniformity, limit diagram size in a layer, and provide printable artifacts.
  – A standard configuration set was used by all CSUs to ensure compatibility and autocode efficiency
Development Tools:
Modeling Standards and Guidelines

- When the program started, there were no Aerospace Specific Modeling standards
  - Needed a Standard for modeling the GN&C algorithms in Simulink, Stateflow, and embedded Matlab (eML).
- Started with Automotive Industry’s published “MAAB” (MathWorks Automotive Advisory Board) Standard
  - This document was tailored (via GNC & FSW splinter team) based on previous experiences and known architectural drivers for the Orion GN&C FSW.
- Standards are available from the Mathworks website for the aerospace community.
- Three major drivers behind the standards
  - Compatibility
  - Autocode Quality
  - Readability
  - Efficiency
Inspections and CSU Memo

- Detailed inspections were performed on the models, not autocode – CSU Development Checklist were used to aid CSU preparation for reviews

- CSU memo’s where generated to further document and clarify design, derived requirements, and testing
• CSU design requirements and model development is iterative
• Re-use of model component test suite by FSW is a significant cost/schedule reduction opportunity
Three types of Model tests were developed:
- Non-Conforming Confidence tests
- Conforming Confidence tests
- Structural Tests

5 Types of Test Criteria:
- Design requirements
- MCDC
- Error Handling
- Limits/Boundaries
- Threshold
Unit Testing: Processor-in-the-loop Testing

- Tests can be developed in the Matlab environment and run on the emulated target environment for increased confidence early in the development process.
Process Benefits

1. **GN&C Design & Analysis environment is merged with the FSW Development & Test environment**
   - GN&C designers are directly involved in the flight implementation of the algorithms.
   - Eliminates traditional “translation” phase of having FSW interpret GN&C’s written-word FSSRs, thus eliminating potential source of error.

2. **Orders of magnitude MORE run-time testing on the FSW source (compared to Traditional process)**
   - FSW autocode is being used in all the analysis runs (not proto-code)

3. **Reduces schedule risk.**
   - FSW implementation is largely complete and tested by CDR (vs. just starting)

4. **Single, common algorithmic development environment with Matlab/Simulink.**
   - No mix of prototyping languages (like C, Fortran, Ada, and other analysis tools)
   - Commonality fosters sharing, algorithm/utility reuse (i.e., Orion Std Lib) and consistency.

5. **Use of RTW/Code-Gen by GN&C Team gives them “eyes-on” the flight code**
   - GN&C developers will gain working familiarity with autocode through the practice of generating it themselves for closed-loop testing with the external simulations for analysis and debugging.
   - Is value-added when needing to understand real-time performance or in-flight issues.
6. “Code Inspections” supplemented, if not supplanted, by Model Inspections
   • Can walk-through the source design graphically (don’t need PowerPoint facsimiles)

7. During Sustaining Engineering, modifications to the “Design Spec” (i.e., the MW Models) can be directly autocomd.
   • Continuous sync between design, documentation and FSW.

8. MathWorks tools are fast-becoming the “industry standard”
   • Modern, prevalent toolset.
   • Matlab programming has become the latest “language” being instituted in many university aerospace curricula today (vs. C or Fortran).
Process Lessons Learned

1. Mandate Team-wide Use of a Common Matlab/Simulink version baseline
   • Coordinating & baselining upgrades at mgmt level is needed for large teams w/ multiple companies

2. Prohibit Dependencies on MW Toolboxes *(aside from RTW, V&V)*
   • Alleviates cost impacts across a large team.

3. Use centralized, customized libraries for “one-stop shopping” *(in lieu of MW toolboxes)*
   • Ensures the entire team is “on the same page” using only the corralled, approved blocks, which adhere to the standards and are “autocodable”.
   • Customization and masks ensures library blocks are used in the intended fashion.

4. Use a single, secure, collaborative, web-based sharing repository
   • Minimum requirement for sharing models and releasing baselines across company lines, firewalls.

5. Every Domain and CSU should have a designated owner/Point of Contact (POC)
   • The CSU POC is the single, acknowledged “hands-on owner” of the model (aids serial development)

6. Each CSU should be a Model Reference
   • Allows CSU to be developed, maintained & config-managed as its own .mdl file (owned by 1 POC).

7. Each Domain sub-team should have a Code Gen POC
   • Since RTW access and skill-base may be limited, 1 POC should be identified to help the others.
   • Important for each Domain sub-team to ensure their CSUs integrate and gen-code (as quality check, at a minimum) before submitting updates to the FSW team for the next baseline.

8. Modeling and algorithm nomenclature standards must be clearly documented, trained, and maintained.
• **Scalability**
  – Time required to update and process (“Mex”) the Simulink diagram, generate autocode and execute time domain simulation grew disproportionately with project size with existing MBD tools
  – Recommendation: See paper for multiple technical solutions to reduce build and execute time, and consider splitting development environment into mission phases – especially during early development

• **Configuration Management**
  – Use model reference blocks to break MBD application into separate CM artifacts, each with an assigned “owner”
  – Avoid parallel development when possible
  – Familiarize team with graphical merge tools and cost the training and licenses

• **Mixed Tool Development Environment**
  – Mixed C simulation and Simulink FSW was workable, but required a broad range of skills for developers. Should probably be avoided for projects that do not have the particular legacy of Orion
Conclusion

- Orion paid some upfront costs for transitioning to an MBD process:
  - A steep learning curve for engineers not familiar with MBD tools
  - Initially slow and complex development tools and processes
  - Configuration management issues
- These issues were mitigated by many of the lessons learned, improved Mathworks products and custom tools that are described in the paper
- Some of the benefits that GN&C is now observing include:
  - Detailed requirements review was replaced by review of MBD artifacts which had proven functionality
  - Automated test framework and report generation has simplified testing and production of test artifacts
  - Automated standards checking tools (e.g. Model Advisor) and graphical artifacts have facilitated the inspection process
  - No schedule time was needed for hand coding GN&C algorithms (40,000+ SLOC were autocoded by CDR)