Using CCDD to Automate Software development on AA2
<table>
<thead>
<tr>
<th>Agenda</th>
</tr>
</thead>
<tbody>
<tr>
<td>• AA2 Background</td>
</tr>
<tr>
<td>• cFS Command and Data Dictionary (CCDD) Overview</td>
</tr>
<tr>
<td>• CCDD Products used on AA2</td>
</tr>
<tr>
<td>• Development on AA2</td>
</tr>
<tr>
<td>• Next Steps</td>
</tr>
</tbody>
</table>
• AA-2 is a development flight test for Multi Purpose Crew Vehicle (MPCV)
  • Launch planned for May 2019 from Space Launch Complex 46
  • AA-1 test was dropped, but AA-2 was not renumbered
  • Pad Abort 1 (PA1) demonstrated similar LAS functionality from a launch pad. (May 2010)

• Largely a test of the Launch Abort System (LAS) on Orion
  • Safety system to quickly separate crew capsule from the Booster (during a failure)
  • Verify LAS works under flight-like conditions to help certify system for crewed missions

• Two identical CPU’s running, only 1 is required (redundancy)
  • Each CPU is cFS instance running on vxWorks (on a PPC)
  • Each CPU has separate serial link to the LAS, but only 1 is needed (redundancy)
  • Code is essentially identical on each computer, but each computer used a different value for each MID that it sends on the Software Bus (SB)
AA-2 Background

AA-2 Avionics & Software
- Designed to use COTS avionics wherever possible
- Dual string design using cFE/CFS on VxWorks
- Reuse of ANTARES Trick Simulation
- CFS wrapped GNC Matlab/Simulink Autocode from mainline MPCV
• CCDD stands for cFS Command and Data Dictionary
• Goddard’s Core Flight System (cFS) has been, is, and is intended to be used by many projects
  • Examples: Lunar Reconnaissance Orbiter (LRO), Morpheus, Exploration EMU (xEMU) spacesuit, Orion Backup Flight Software (BFS)
  • Success of the cFS concept is shown by the number cFS projects at FSW-2018
• A command and data dictionary (CDD) defines telemetry/command messages
• Each cFS project must select a way to manage their CDD
  • Frequently involves using a spreadsheet, with custom SW to convert into useful files
• cFS Command and Data Dictionary utility (CCDD) was designed as a generic utility to eliminate duplication of effort in order to make CDD management easier
CCDD Goals

• Create a configurable CDD utility that runs on multiple operating systems
  • Written in Java for maximum portability
• Easy creation/modification of CDD information
  • Graphical user interface (GUI) to interact with the database
• Store all CDD information into a standard database (postgresql)
• Bidirectional transfer of information to/from the CCDD
  • Cut-n-paste to Excel, import/export via XTCE/CSV/JSON
• Easy access to CDD information (via scripting languages and web applications)
  • Allows user to code in various languages (ruby/python/js) and access CDD information
    » Create vehicle and ground software products, data summary, etc
    » Generate complicated CFS products: Schedule or network tables, copy table, etc
Data is accessible to scripting languages (JavaScript, Python, etc.)
  • Example scripts provided for common products

- **Imported/exported via:**
  - CSV (comma-separated values)
  - JSON (JavaScript Object Notation)
  - XML (Extensible Markup Language)
    - EDS (Electronic Data Sheet)
    - XTCE (XML Telemetric and Command Exchange)
  - OS clipboard (“cut & paste”)

Web-based dataserver (JSON)
CCDD Demo

Project: SampleProject

<table>
<thead>
<tr>
<th>Index</th>
<th>Server</th>
<th>Project</th>
<th>Date/Time</th>
<th>Type</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>6248</td>
<td>jsc-er-cfs01.jsc.nasa.gov 5432</td>
<td>SampleProject</td>
<td>12:42:42.376</td>
<td>Success</td>
<td>Project 'SampleProject' locked</td>
</tr>
<tr>
<td>6266</td>
<td>jsc-er-cfs01.jsc.nasa.gov 5432</td>
<td>SampleProject</td>
<td>11/27/2018 12:42:42.584</td>
<td>Success</td>
<td>Project 'SampleProject' unlocked</td>
</tr>
<tr>
<td>6269</td>
<td>jsc-er-cfs01.jsc.nasa.gov 5432</td>
<td>SampleProject</td>
<td>11/27/2018 12:43:42.680</td>
<td>Success</td>
<td>Project database 'sampleproject' closed</td>
</tr>
<tr>
<td>6270</td>
<td>jsc-er-cfs01.jsc.nasa.gov 5432</td>
<td>SampleProject</td>
<td>11/27/2018 12:43:42.480</td>
<td>Success</td>
<td>Connected to server as user</td>
</tr>
<tr>
<td>6271</td>
<td>jsc-er-cfs01.jsc.nasa.gov 5432</td>
<td><em>server</em></td>
<td>11/27/2018 12:43:42.494</td>
<td>Status</td>
<td>PostgreSQL: 8.4 *** JDBC: PostgreSQL 9.4.1207 jre7 (type 4)</td>
</tr>
<tr>
<td>6272</td>
<td>jsc-er-cfs01.jsc.nasa.gov 5432</td>
<td><em>server</em></td>
<td>11/27/2018 12:43:42.494</td>
<td>Status</td>
<td>Server connection closed</td>
</tr>
<tr>
<td>6273</td>
<td>jsc-er-cfs01.jsc.nasa.gov 5432</td>
<td><em>server</em></td>
<td>11/27/2018 12:43:45.622</td>
<td>Success</td>
<td>Connected to project 'SampleProject' as user</td>
</tr>
<tr>
<td>6276</td>
<td>jsc-er-cfs01.jsc.nasa.gov 5432</td>
<td>SampleProject</td>
<td>11/27/2018 12:43:45.060</td>
<td>Status</td>
<td>PostgreSQL: 8.4 *** JDBC: PostgreSQL 9.4.1207 jre7 (type 4)</td>
</tr>
<tr>
<td>6279</td>
<td>jsc-er-cfs01.jsc.nasa.gov 5432</td>
<td>SampleProject</td>
<td>11/27/2018 12:43:45.061</td>
<td>Status</td>
<td>Password: 8.4 *** JDBC: PostgreSQL 9.4.1207 jre7 (type 4)</td>
</tr>
</tbody>
</table>
• C header files defining SB command/telemetry messages
  • Define the structure for all software bus (SB) messages.

• MID file generation (Same file used by both CPUs)
  • Defines all the MIDs for each cFS message sent/received on each CPU
    » CPU2 adds 0x100 to all the MID values sent out by CPU1
    » Allows Ground SW to know which computer sent a message

• HK copy table generation
  • Telemetry link is constrained. Select various parts of messages to go at different rates
  • 2 separate telemetry paths (per CPU), so 4 separate messages are sent

• ITOS “rec” files (ground control system)
  • Used to define commands/messages in ITOS

• Data decom config files (post-flight data processing)
  • Provides CSV files with desired parameters to be analyzed
  • Custom built utilities to decode data from “raw” recorded telemetry files
    » Significantly faster than data replay and seqprt utility in ITOS
Major AA-2
Activity in 2018

• Worked to develop the CDD before the SW development was complete
  • CDD not treated as an “as built” post-development documentation effort
  • Required iterations on data structures and MIDs, but minimized interface issues
• Added ability to track ~900 DFI system parameters (in addition to OFI)
  • Allows additional insight into vehicle for all ground controllers
• Automated data processing and “quicklook” of key parameters after tests
  • Allows rapid verification of how the system performs during simulation runs
• Automated regression testing to perform SW verification activities
  • Test framework consumed CCDD-generated files to define CCSDS messages
  • Can verify any parameter (of any messages) meets expected values (at specific times)
• Automated remote control/monitoring of ground power supplies
  • Quite useful since people need to stay miles away during launch window
### Future Work

- Launch (with a successful abort!) in May 2019
- Working with Education/Outreach on student outreach contest
  - Build a real-time application to show the position of the vehicle during the test
  - Winning team to view KSC launch (and see app in use)
- The CCDD tool has successfully been used to automate/autocode a large amount of software used on AA-2.
  - Pass on the lessons learned so they can be leveraged on other programs