Study of Tools for Command and Telemetry Dictionaries

Craig Pires
NASA Ames Research Center
Spacecraft and Ground FSW needs

• **Spacecraft Defined by data**
  • Commands - Formatted to the Spacecraft
  • Telemetry – From the Spacecraft
  • Ground Software
  • Subsystem Teams
    • Command and Data Handling (C&DH)
    • Electrical Power System (EPS)
    • Guidance, Navigation and Control (GN&C)
  • Third Party Vendors
  • Science Payloads.............

• **Flight Software needs to figure out how to talk to all of it.**
  • And not do it multiple times for each use (Simulink, C Code, ITOS, …)
Our early approach – C&T Database

- **LADEE – Integrated Command and Telemetry Database (Mostly…)**
  - Shared between C&DH Flight Software, Simulink Models and ITOS
  - Other data analysis tools leveraged

- **Not for all data uses:**
  - Legacy source code. (cFS was hand inserted into C&T DB)
  - Created “Pass-Thru” Commands
  - Mostly just Commands and Telemetry (Packets)
  - Other systems for: Parameter Tables, Temperature Calibration Curves
    - Monstrous Excel Spreadsheets
    - Spreadsheet difficult to maintain
Next Step - SCIMI

- SCIMI – System Configuration Information & Mission Interfaces
- Relational Database based off of Django
- Address limitations of previous “LADEE” approach
  - Full Command & Telemetry plus…
  - Produces Simulink tables products
  - Handle other cFS tables
  - Calibration Curves
  - Consumes YAML and uses Python for product generation
  - Infrastructure entirely Python
  - Built-in Embedded web GUI and command-line interfaces
  - Light on documentation
  - Extremely customizable per mission (almost too much, meaning not out-of-the-box)
SCIMI Logical Flow

Modify & view data in the database via a GUI.

Webpage serving as GUI

View data in the database via a GUI

Tells us what is to be called when a user is interested in navigating to a certain page

Urls

Templates

Set up dynamic user interfaces with embedded code

Admin

YAML Dump

Model/Relational DB

Create necessary tables & ORMS

Data source

GUI

Webserver
SCIMI In’s & Out’s

- Ground Products (itos, assist, etc)
- Docs (C&T Dict, ICDs)
- Spread Sheets (xls, csv)
- ??

Django – Relational DB

- Web-based interaction

Text Archival (YAML, JSON, XML)
- MATLAB (I/F, tunables)
- C Headers
- Engr. Drawings
- Web-based interaction
New Development - CCDD

- CCDD – Core Flight System (cFS) Command and Data Dictionary
- JSC Developed, officially NASA Open-Sourced
  - These are two tools similar GIT vs Subversion
- Central SQL Database - (PostgreSQL)
- Infrastructure implemented in Java
- Multiple sources and products
  - Consumes CSV, JSON, EDS, and XTCE and uses JavaScript, Python, Ruby, Groovy, etc… for product generation
  - Highly customizable due to variety of input and output formats
- Embedded web GUI and command-line interfaces
- Extremely well documented
CCDD Logical

- **Script Access**: Javascript, python, ruby, groovy, etc.
- **Output Files**: C headers, ITOS record and display, etc.

- **Project Database**
- **CCDD**
- **JSR-223 Scripts**
- **CSV Files**
- **JSON Files**
- **XML Files** (EDS, XTCE)
- **Clipboard**
- **Embedded Web Server**
- **Web Applications**
What to Do? (SCIMI vs CCDD)

- Struggled to keep consistency with tools
- Problems with “Clone and Own”
- Looking hard at transitioning to CCDD – Test Implementation
- Creating needed CCDD Additions for us.
  - Simulink
    - Strength of DB tools is – adapter/translator
- Todo for Us
  - Interfaces to additional data analysis tools
  - Data Marshalling - Translation
  - Possible – Database flexibility
What did we need to do to implement CCDD test for RP?

• Every mission has slightly different approach
  • Command and Data Dictionary tools need flexibility
  • CCDD provides such flexibility through table customization
  • In addition, provides full API for interaction with database

• Slight modifications to infrastructure
  • Facilitated by tool creators
  • Particularly relevant for Model Based Design code auto-generation
  • Not too much needed because of high customizability of tables
    • Namely the ability to add “data fields” to tables that uniquely identify information contained therein
    • For example, “Produce REC” and “Simulink App” boolean data fields for structures

• Most modification done to scripts that generate products
  • Again, primarily to facilitate specific products for MBD auto-generation
Summary

- Space Missions are defined by data
- Tools for Managing make it much simpler and reduce errors
- CCDD and SCIMI are two tools for managing that data
  - Both have strengths and weaknesses
  - Tools themselves need support
- Evaluation of tools based on project/mission needs
Backup
Implementation Details

Common issues for all tools

• **Maintenance of the tools**
• **Which group “Owns” the Database through the Project?**
  • Start with FSW
  • Transition to Mission Operations
  • Science Operation Requirements?
• **What Tools?**
  • FSW Development.
  • Integration and Test
  • Mission Operations
  • Science Operations
• **Procedures for modification?**
• **Backup and CM issues?**
SCIMI vs CCDD

- **SCIMI**
  - Pros:
    - Django
      - Powerful/Flexible Database Tool
      - Database agnostic
    - Met mission needs
  - Cons:
    - Need for internal mission support
    - Lack of documentation
    - Mission specific implementation

- **CCDD**
  - Pros:
    - Complete implementation – with API
    - Mission/project agnostic
    - Well documented
  - Cons:
    - Customization needed for missions
    - Needed script development (Simulink)