cFS Development at Ames

Primarily SIL,
Some Testing, and
a Little Knowledge DB

J. Patrick Castle
Themes

- Model Based Design
- Automation of common task
  - Continuous integration
  - Automated testing (Continuous Verification)
- Iterate early and often – Allow experimentation
- Clear knowledge of state
  - Test Framework
  - Knowledge Database (SCIMI)
- Remain Agile and still adhere to NPR7150.2B
Flight Software Overview

• **Scope**
  - Onboard Flight Software (Class B)
  - Support Software and Simulators (Class C)
  - Integration of FSW with avionics

• **Guiding Documents**
  - NPR7150.2 Software Engineering Requirements
    - CMMI Level 2 or Equivalent
  - NASA-STD-8739.8 NASA Software Assurance Standard

• **Development Approach**
  - Model Based Development Paradigm (prototyped process using a “Hover Test Vehicle”)
    - 5 Incremental Software Builds, 2 Major Releases before launch
      • Final Release during mission.

• **Leverage Heritage Software**
  - GSFC OSAL, cFE, cFS, ITOS
  - Broad Reach Drivers, VxWorks
  - Mathworks Matlab/Simulink & associated toolboxes
- Develop Models of FSW, Vehicle, and Environment
- Automatically generate High-Level Control Software
- Integrate with hand-written and heritage software.
- Iterate while increasing fidelity of tests – Workstation Sim (WSIM), Processor-In-The-Loop (PIL), Hardware-in-the-Loop (HIL)
- Automated self-documenting tests providing traceability to requirements
Model Based Development

Requirements

Design/Algorithm Development

Flight Software Modeling

Vehicle & Environment Modeling

Workstation Simulations (eg. Simulink)

Hand Developed Apps

Code Generation

Unit Tests

Integrated Tests
Processor-in-the-Loop
Hardware-in-the-Loop

Automated Reporting

Analysis

Heritage Models

Heritage Software

*L4/L5

*Rough Mapping
Model Based Development

Requirements → Design/Algorithm Development → Heritage Models

Design

Virtual Test

Virtual Integration

System Analysis

Flight Software Modeling → Vehicle & Environment Modeling

Workstation Simulations (eg. Simulink)

Code Generation

Hand Developed Apps

Unit Tests

Automated Reporting

Integrated Tests
Processor-in-the-Loop
Hardware-in-the-Loop
Future of SIL

• Open Source (get it out there)
• Ease of use
  – Documentation
  – Tutorials and examples
  – Install script
  – cFS Integration – Scimi, EDS, XML, Other tools?
• New Features
  – Asynchronous
  – Multi-Rate messages
• Feedback from community
  – What would you like to see?
Testing Framework

• Developed for LADEE
• Manages Oracles and tests in single location
• Maps related components, requirements, and tests
• Multiple data sources
  – WSIM
  – Telemetry
  – CnT dictionary
• Automatic report generation
Knowledge DB Discussion

• **System Level Tool**
  – Collects and manages project information
  – Tracks and propagates changes
  – Verifies content (interfaces, msg_ids, tables, calibration curves etc.)
  – Interfaces with tests (in the works)
  – Autogenerates products (headers, rec, pages, xml, tables, etc.)

• **Now Django based**

• **Trying to make project customization easy**
Knowledge DB Discussion

• Compliments/Overlaps with tools from other centers and companies
• Interest in formalizing community development?

How standards proliferate:
(See: A/C chargers, character encodings, instant messaging, etc.)

Situation: There are 14 competing standards.

14?! Ridiculous! We need to develop one universal standard that covers everyone’s use cases. Yeah!

Soon:

Situation: There are 15 competing standards.
Some Projects

• BioSentinel
  – Deep Space CubeSat
• Resource Prospector (RP)
  – Lunar rover w/ drill
• ASO
  – Autonomy for ISS
• Autonomy Operating System
  – UAS and Rovers
• Safe50
  – UAS flight software