Advanced Exploration Systems (AES) Core Flight Software (CFS) Project

- CFS Product Highlights
- CFS Projects @ JSC
- Upshots of SW development with CFS

Tam Ngo/Software Manager
NASA/Johnson Space Center (JSC)
Software, Robotics & Simulation Division
Spacecraft Software Engineering Branch (ER6)
December 2018
CFS Product Highlights

- **Generic Command Ingest (CI) & Telemetry Output (TO) CFS applications**
  - Can be easily expanded and customized
  - Supports multiple channels
  - Supports multiple communication protocols: UDP, TCP & RS-422
  - Supports CCSDS Space-Data Link Protocols: TM-SDLP, TC-SDLP, COP-1
  - Works with CFDP (CF) application to do CFDP file transfers

- **Time-triggered scheduler (SCH_TT) CFS application**
  - Uses TTE network scheduling for scheduling messages
• **Software Bus Network (SBN) CFS application**
  - A collaborative work with Ames Research Center (ARC)
  - Serves as a data bridge between cFS systems
  - Makes distributed network of cFS systems possible
  - Supports multiple communication protocols: UDP, TCP, serial

• **Protobetter tool**
  - Generates code for serializing/de-serializing CCSDS messages for transmission across the network
  - Generated code is compiled into its own CFS library
  - Works with CDD databases to access message definitions
  - Can run stand-alone if given message definitions
  - Is used by the SBN application, and eventually CI & TO applications
• **Command & Data Dictionary (CDD) Tool**
  – Is used to define system command & data definitions
  – Information is stored in a relational database, PostgreSQL
  – Provides a set of APIs for users to access the data via scripts
  – Provides a set of standard scripts that can be customized to generate output files like C headers, XTCE files, copy tables for HK application, and eventually, scheduling tables for SCH application.
  – Available at [https://github.com/nasa/CCDD](https://github.com/nasa/CCDD)

• **CFS-101 Training**
  – A self-guided training package
  – Comes with a virtual machine as the tutorial workspace & a step-by-step tutorial guide
  – The virtual machine is equipped with the latest CFE, OSAL & PSP from [https://sourceforge.net](https://sourceforge.net) and all the necessary CentOS libraries.
  – Available at [https://github.com/nasa/CFS-101](https://github.com/nasa/CFS-101)
CFS Projects @ JSC

- **Orion Multi-Purpose Crewed Vehicle (MPCV)**
  - Class A, safety-critical FSW
  - Vision Processing Unit (VPU) Flight Computer
    - VxWorks on Leon3 SPARC processor
    - Backup Flight System (BFS) apps
  - Camera Controller Units
    - Ubuntu Linux on I5 Intel processor
    - Optical Navigation (OpNav) app

- **Orion Ascent & Abort 2 (AA2) Flight Test (May 2019)**
  - Class B, safety-critical FSW
  - VxWorks on SP0 PPC processors
CFS Projects @ JSC (cont.)

- Advanced Extra-vehicular Mobility Unit (xEMU) Flight Experiment – Caution & Warning System (CWS)
  - Class A, safety-critical FSW
  - VxWorks on Leon3 SPARC processor
  - An advanced space suit

- Seeker Flight Experiment on ISS (July 2019)
  - Class C, non-safety-critical FSW
  - Wumbo GNU/Linux on CHREC space processor
  - An external, free-flying robotic inspector

- Certification of cFE on VxWorks ARINC-653
  - Class A, safety-critical FSW
  - VxWorks ARINC-653 on SP0 processor

- Avionics & Software Platform for Exploration Capabilities & Technologies (ASPECT) as ISS payload
  - Class C, non-safety-critical FSW
  - VxWorks on SP0-S processors
CFS Projects @ JSC (cont.)

• AES integrated testing project
  – A&S platforms for multi-center collaborations to integrate autonomous systems & operations in a distributed run-time environment

• Next Space Technologies for Exploration Partnerships (NextSTEP) program
  – Seeks commercial development of deep space exploration capabilities
  – Assists private companies in applying NASA-invested technologies to develop their capabilities
Upshots of SW Development with CFS

From SW development perspective,

- **Having a common SW framework increases SW productivity**
  - Modular application code with the same “look-n-feel”
    - Same code template
    - Same naming conventions
      - File names, function names, message ID names, command code names
    - Same build template
    - Same source directories
  - Minimal time to stand up an initial running system from scratch before any mission-specific capabilities are implemented
    - SW re-use actually works and with minimal effort!
  - Quicker to get up to speed with the new project for CFS-experienced developers, even when joining at a later phase of the development life cycle
    - Already familiar with the code setup
    - Can detect & fix application bugs in shorter time
    - Know where to look
  - Easier to pre-train newbies to become CFS-proficient before they join the project
From SW development perspective (cont.),

• **Collaborations within the CFS user community**
  – Enable tool & code sharing
    • Increase the level of SW re-use at minimal cost
    • Open source SW makes this possible
  – Dissemination of lessons learned & best practices
    • The TODOs and the NOT-TODOs
    • Minimize development risks
  – Provide leverage for future projects with current & past projects
    • Emulate the existing system HW/SW configurations or tweak it to suit the new missions' needs
From project management perspective,

• **Making use of high quality flight software**
  – CFS has its pedigree in flight software!
    • CFS has been tested, certified & flown in space in numerous HW/SW configurations.

• **Better grip on SW development schedule & risk mitigation**
  – Using metrics, lessons learned & best practices from other CFS projects to help planning for new CFS projects

• **Increase in management buy-ins on new SW projects that use CFS framework**
  – Costs & schedules of past & current CFS projects show the productivity, benefits & effectiveness of the paradigm
  – Increasing number of CFS projects that are at various level of certifications