A Reusable and Adaptable Software Architecture for Embedded Space Flight Systems (CFS)

The Core Flight Software System

Jonathan Wilmot
NASA GSFC Code 582
Jonathan.J.Wilmot@nasa.gov
Some Challenges of the Space Software Domain

- High availability
  - Self Safing
  - Must run with minimal interruption for a few to tens of years
- Hardware is in a harsh environment
  - Bits can change due to radiation, blocks of memory can degrade and become unusable.
- Flight Processor (constraints) vary widely due to power and weight constraints
  - Processors are several generations behind terrestrial components
  - High end - 166 MHz PowerPC with 8meg non-volatile program store and 128MB RAM
  - Low end - 12MHz Coldfire embedded in gate array with 512k non-volatile and 4MB RAM
- Software must be remotely modifiable and still operate while changes are being made
- Many custom one of a kind interfaces for one of a kind missions
- Sustaining Engineering
  - Systems must be maintained long after the platform and systems are obsolete.
- Price of failure is high, tens to hundreds of millions of dollars
An Approach to the Challenges

The Core Flight System is a platform-independent, reusable Flight Software (FSW) environment integrating a core flight executive, software component library, and Integrated Development Environment (IDE). It addresses some of the Space domain software challenges using key concepts of the GMSEC reference architecture.

- Layered Architecture
- Standard Middleware/Bus
- Application Programmer Interface
- Plug and Play
- Reusable Components

Core Flight Executive
Component Library
Key Concept #1
A Layered Architecture

- CFS internals are carefully layered.
- Each layer "hides" its implementation and technology details.
- Internals of a layer can be changed -- without affecting other layers' internals and components.
- Small-footprint, light-weight architecture and implementation minimizes overhead.

**Benefits:**
- Eases technology infusion and evolution.
- Eases modification at all stages of development and on-orbit.
- Provides Middleware, OS and HW platform-independence.

*CODES+ISSS'05, Sept. 19–21, 2005*
Key Concept #2
Standard Middleware Bus

Publish/Subscribe
- Components communicate over a standards-based Message-oriented Middleware/Software Bus.
- The Middleware/Software Bus uses a Publish/Subscribe model, so cooperating components don’t need to know the details of inter-communication (location of others, protocols used, etc.).

Benefits:
- Simplifies component SW
- Minimizes interdependencies
- Supports HW and SW runtime “plug and play”
- Speeds development and integration.
- Enables dynamic component distribution and interconnection.

Legacy: Tightly-coupled, custom interfaces- data formats - protocols, internal knowledge, component interdependence.

Publish/Subscribe: loosely-coupled, standard interface, data formats, protocols, & component independence.
Key Concept #3
Standardized API for Software and Hardware Components

Application Programmer Interfaces
- CFS services and middleware communication bus has a **standardized, well-documented API**
- An abstracted HW component API enables standardized interaction between SW and HW components.

Benefits:
- Eases development and testing when using **distributed teams**
- With the framework already in place, applications can be started earlier in the development process
- Don’t need to wait for the target hardware to be completed.
- **Simplifies integration, reduces development time, shortens schedules.**

API supplies all functions and data components developers need.
Packaging the first 3 Concepts
The
Core Flight Software Executive (cFE)

- Strategic Software Layering
  - Software of a layer can be changed without affecting the software of other layers

- Advanced Message Handling
  - Eliminates manual configuration of FSW
  - Automates integration of FSW with applications and hardware components (Publish/Subscribe model)

- Standardized, Abstracted Interfaces
  - Minimizes software impacts from flight hardware, RTOS(\textsuperscript{*}), and application changes

\textit{CODES+ISSS'05}, Sept. 19–21, 2005
CFS Key Concept #4
SW Components and HW Devices Plug and Play

Plug and Play
- cFE API’s support add and remove functions
- SW components can be switched in and out at runtime, without rebooting or rebuilding the system SW.
- Qualified Hardware and CFS-compatible software both “plug and play.”

Benefits:
- Changes can be made dynamically during development, test and on-orbit even as part of contingency management
- Technology infusion and upgrades can be taken advantage of later in the development cycle.
- Testing flexibility

This powerful paradigm allows SW components to be switched in and out at runtime, without rebooting or rebuilding the system SW.
Reuschable Components

- Common FSW functionality has been abstracted into a library of reusable components and services.
- Tested, Certified, Documented
- A system is built from:
  - Core services
  - Reusable components
  - Custom mission specific components
  - Adapted legacy components
  - Associated HW

Benefits:

- Reuse of tested, certified components supplies savings in each phase of the software development cycle:
- Reduces risk
- Teams focus on the custom aspects of their project.
Packaging the last 2 Concepts

CFS Component Library

Flight Software Reuse Libraries
- Select pre-validated FSW components from FSW reuse libraries
- Validated common SW components include Requirements, Test Plan, Documentation

Mission-Unique Components
- Mission-unique FSW Components can be adapted for use
- Science applications developed on Scientist’s desktop can plug into flight systems without change when developed with the API standards

Certified Reusable, Custom and Adapted Legacy Components - facilitate rapid assembly of a working Flight SW System as well as supplying a path to migrate custom and legacy components into the CFS reuse library.

Component Standard API and Data Communication: components are loosely-coupled with a standard API and standard publish/subscribe intercommunications protocol and data formats. These features enable plug & play connectivity and dynamic integration / reconfiguration.
Multiprocessor/System of Systems
Rings of Functionality

- Interchangeable avionics boxes have a core ring of standard functions.
  - Actual code and implementation can be different for each vendor, but need to meet the API's and ICD's
  - Event messages to alert operators or other components of asynchronous data
  - Common software load/dump interfaces
  - Common set of file service interfaces
  - Core ring can use memory management unit (MMU) for self protection
  - Boot loader is in hardened ROM

- Benefits
  - Speed prototyping and integration times
  - Compress schedule and reduce schedule risk
  - Common training for maintenance teams.
  - Reduced cost for FSW development and maintenance
  - Reduce training for on board trouble shooting for spacecraft personnel
  - Boxes can take on "personalities" as needed
  - Fewer flight spares needed

CODES+ISSS '05, Sept. 19–21, 2005
Next Steps
End-to-End
Status

Targeted Missions

- Original – Global Precipitation Measurement (GPM)
- Current - Lunar Reconnaissance Orbiter (LRO)

2004 multi-CPU/Box prototype demonstrated

- Core, generic FSW services and Software components
- Dynamic application load and startup
- Dynamic message bus reconfiguration after simulated "Box faults"

Version 2 cFE, delivered to LRO July 15 2005

- Process and Memory protection integration to start Q1 FY 2006
  - No API changes anticipated.

- Started work on an Integrated Development Environment (IDE) based on the open source Eclipse.

CODES+ISSS'05, Sept. 19–21, 2005