cFE/CFS

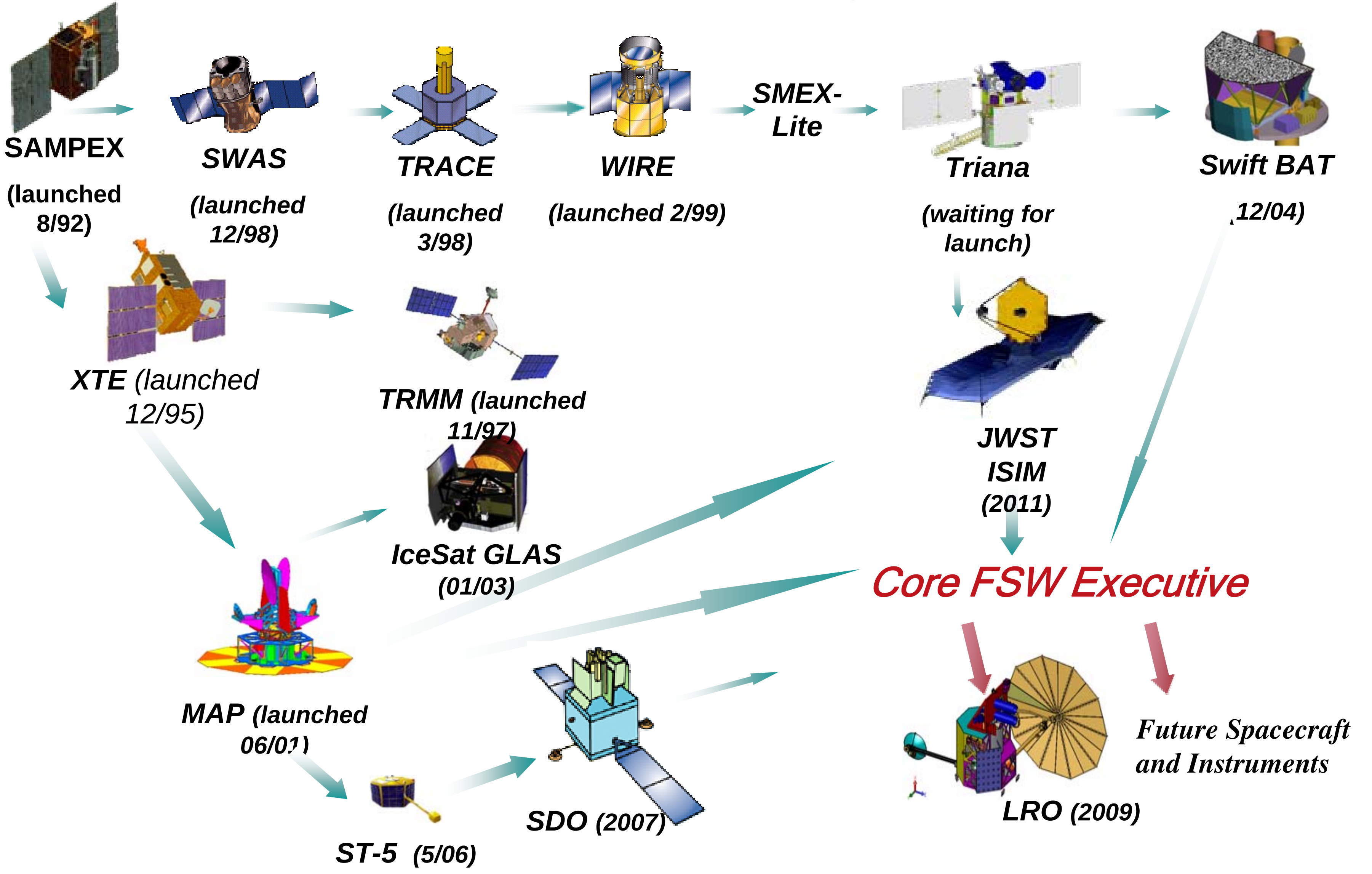
Charlie Wildermann/FSW GSFC

November 13, 2008

Why cFE/CFS

- Requirements
  - The Requirements for Command and Data Handling (C&DH) Flight Software are very similar from Flight Project to Fight Project
  - The Requirements for Guidance Navigation and Control (GNC) Flight Software can also be quite similar from Flight Project to Fight Project
- So, let’s not “re-invent the wheel” each project
  - cFE/CFS responds to this by allowing FSW developers and testers to concentrate on the uniqueness of a project

cFE Heritage



SAMPEX (launched 8/92)

SWAS (launched 12/98)

TRACE (launched 3/98)

WIRE (launched 2/99)

SMEX-Lite

Triana (waiting for launch)

Swift BAT 12/04)

XTE (launched 12/95)

TRMM (launched 11/97)

JWST ISIM (2011)

IceSat GLAS (01/03)

MAP (launched 06/01)

Core FSW Executive

ST-5 (5/06)

SDO (2007)

LRO (2009)

Future Spacecraft and Instruments

Past vs. Future Comparison

Past

- FSW lead for Mission X would obtain FSW and artifacts from heritage mission that they knew
  - Branch had several different “heritage architectures” to choose from
- Changes were made to heritage FSW artifacts for new mission
  - New flight hardware or Operating System required changes throughout FSW
  - FSW changes were made at the discretion of developer
  - FSW test procedure changes were made at the discretion of the tester
  - Extensive documentation updates were made
- Integrating new FSW components required manual coordination
  - Manually defined flight tables

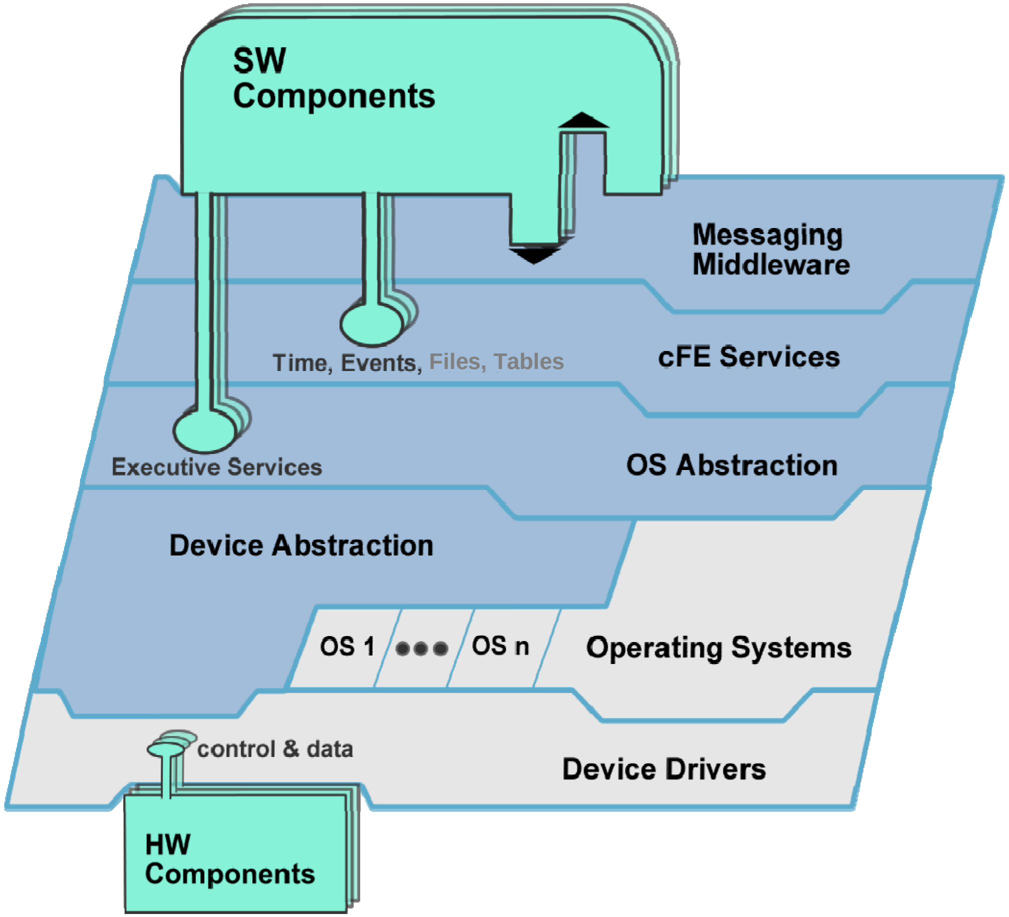
Future (with CFS)

- FSW lead for Mission X will obtain FSW and artifacts from the CFS *Re-use Library*
  - One CFS “product line” architecture to choose from
  - All artifacts are contained in the re-use library
- CFS Changes required for a mission are *controlled* and *localized*
  - New hardware and Operating System changes are localized to Operating System Abstraction Layer (OSAL) – other FSW *not* affected.
  - FSW Requirements, source code and test procedures are *controlled* by Re-use Library CCB
- Integrating new FSW components requires *little* manual effort
  - Run-time registration

Layered Architecture

- 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.

- Enables technology infusion and evolution.
- Doesn’t dictate a product or vendor.
- Provides Middleware, OS and HW platform-independence.



Past vs. Future Comparison (con't)

Past

- Cost advantages of using heritage products was not realized
- Little to no collaboration within GSFC, NASA or outside entities was feasible
- On-orbit FSW maintenance team needed to understand *each* heritage architecture

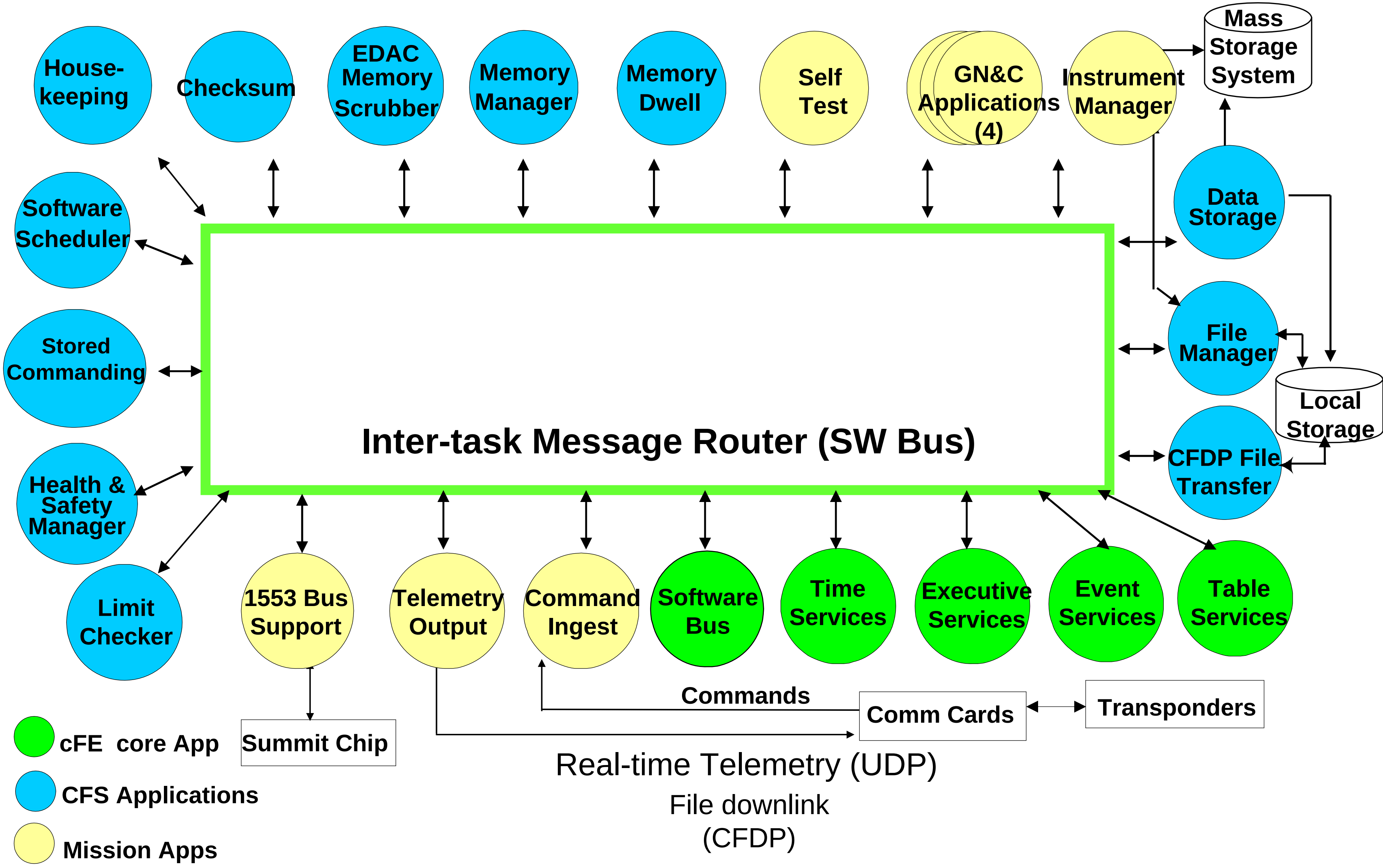
Future (with CFS)

- Effort focused on new and unique FSW applications
- Standard FSW interfaces (APIs) facilitates collaboration across NASA
- On-orbit FSW maintenance team needs to understand *one* product line

What is the CFS?

The Core Flight Software System is a mission-independent, platform-independent, Flight Software (FSW) environment integrating a reusable core flight executive (cFE).

Example FSW Context Diagram



CFS Goals

- Reduce time to deploy high quality flight software
- Reduce project schedule and cost uncertainty
- Directly facilitate formalized software reuse
- Enable collaboration across organizations
- Simplify sustaining engineering (AKA. FSW maintenance)
- Scale from small instruments to System of Systems
- Platform for advanced concepts and prototyping
- Common standards and tools across the branch and NASA wide

Build on the many successful FSW experiences and ideas of FSW staff who worked previous Goddard missions

Supporting the Goals

- Layered Architecture
- Standard Middleware/Bus
- Standard Application Programmer Interface for a set of core services

} Core Flight Executive

- Plug and Play
- Reusable Components

} Component Library

- Configuration Management
- Requirements Tracking
- Development Standards
- Development Tools

} Integrated Development Environment (IDE)

All of the above to be managed in a FSW Re-use Library