The Core Flight System (cFS) Experiences in Opening an Architecture

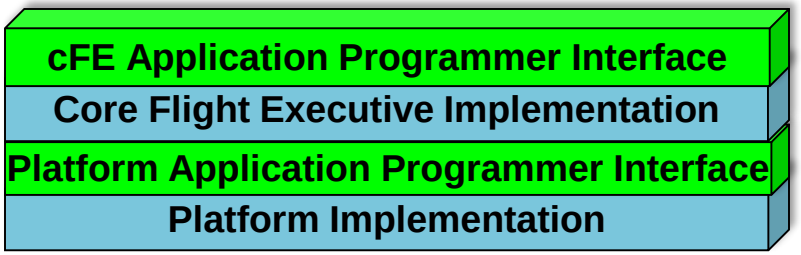
David McComas

NASA Goddard Space Flight Center

October 24, 2018

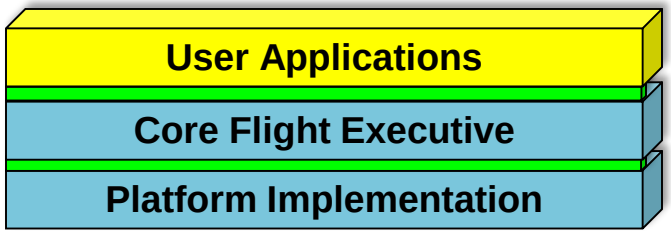
What is the Core Flight System (cFS) ?

- A NASA multi-center configuration controlled open source flight software framework



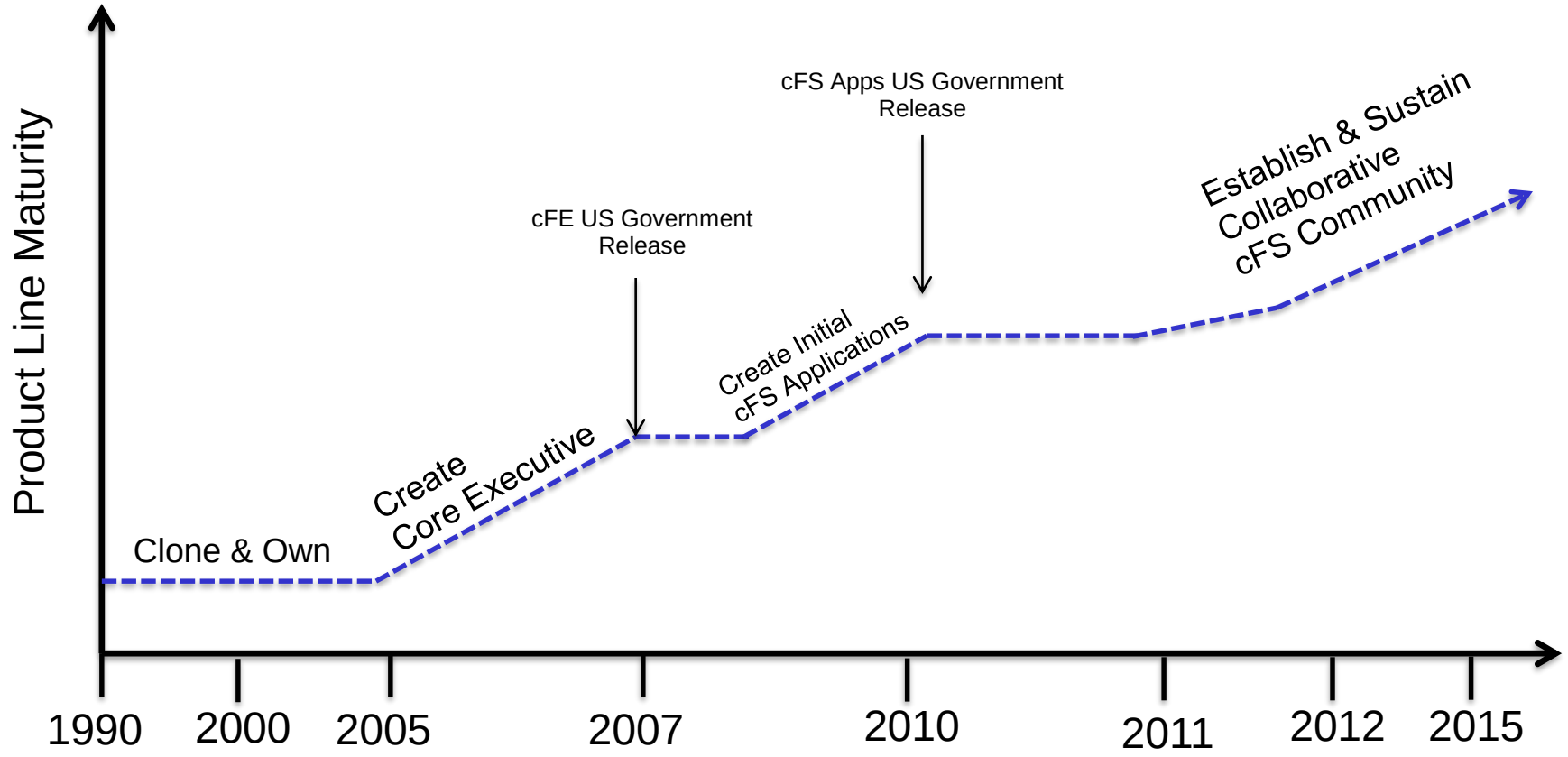
- Layered architecture with international standards-based interfaces
- Provides development tools and runtime environment for user applications
- Reusable Class A lifecycle artifacts: requirements, design, code, tests, and documents

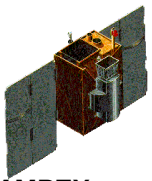
- The framework is ported to a platform and augmented with applications to create Core Flight System (cFS) distributions



- A worldwide community from government, industry, and academia

cFS Timeline

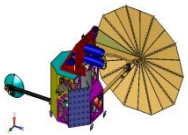




SAMPEX (1992 launch)

GSFC Consolidates FSW Orgs

SDO (2010 launch)



LRO (2009 launch) (Co-develop cFE)

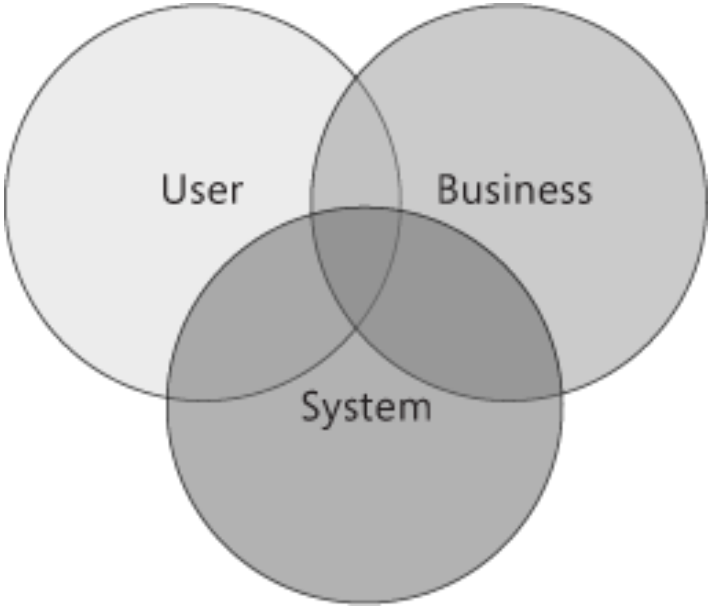
GPM (2014 launch) (Co-develop cFS)

cFE Open Source

Multiple NASA centers using cFS

cFS Open Source

What is Software Architecture?

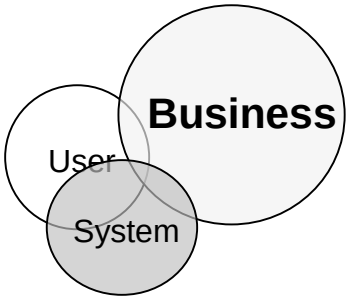


- “Software application architecture is the process of defining a structured solution that meets all of the technical and operational requirements, while optimizing common quality attributes “
  - https://msdn.microsoft.com/en-us/library/ee658098.aspx

cFS Architectural Timeline

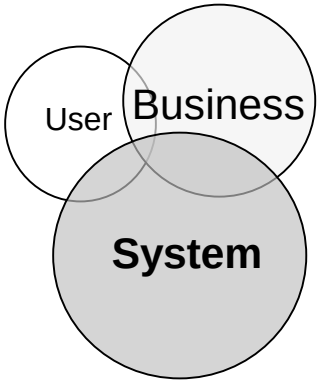
How has the cFS User-Business-System intersection changed?

< 2005



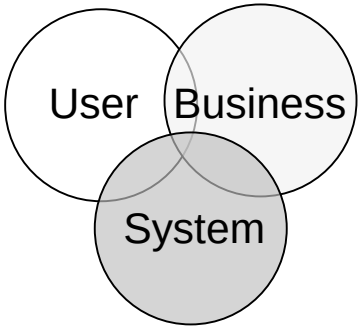
- NASA Goddard, project centric

2010 - 2015



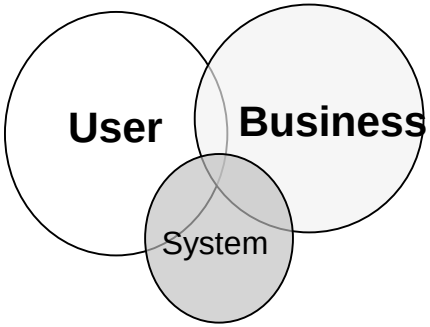
- Architecting Product Line

2010 - 2015



- NASA Community

> 2015



- Open Source Community

Project Centric Reuse

- “Clone & Own” - Used similar heritage missions as starting point
- Changes made to the heritage software for the new mission were not systematically managed
  - New flight hardware and/or operating system required changes throughout FSW
  - Small FSW requirement changes could have significant testing impacts
  - FSW changes were made at the discretion of developer
    - FSW test procedure changes were made at the discretion of the tester
    - Extensive documentation changes were made for style
  - Not all artifacts from heritage missions were available
- Inconsistent and minimal flight software (FSW) reuse cost savings

Breaking Tradition

- Management changed mindset to institutionalize reuse
- Formed a team of senior flight software engineers
  - Management isolated engineers from short term mission schedules
  - Diverse mission experience helped identify the commonality across missions

- Essential team activities:
  - Determine impediments to good flight software reuse
  - Perform heritage analysis
    - Utilize best concepts from missions ranging from Small Explorer class to the Great Observatories
    - Identify and utilize commonality across missions
  - Design with reusability, extendibility, and adaptability
  - Explicitly define architecture goals
  - Leverage increase in onboard processing and advancements in software engineering

Heritage - What Worked Well

- Message bus
  - All software applications use message passing (internal and external)
  - Consultative Committee for Space Data Systems (CCSDS) standards for messages (commands and telemetry)
  - Applications were processor agnostic (distributed processing)
- Layering
- Packet based stored commanding (i.e. Mission Manager)
- Vehicle Fault Detection Isolation and Recovery (FDIR) based on commands and telemetry packets
- Table-driven applications
- Critical subsystems synchronized to the external device communication (i.e. network) schedule
- Well-defined application interfaces
  - Component based architecture

Heritage - What Worked Well

- Innovative culture
  - Constant pipeline of new and varied missions
  - Teams keep trying different approaches
    - Rich heritage to draw from
- Teams worked the entire FSW life cycle
  - Requirements through launch + 60days
  - In-house maintenance teams that participated in the FSW development and test
- Keep the little “c” in the architecture
  - A little core framework, as in low footprint, optimized for flight systems
    - Can we fit in a cubesat with 800KB flash and 2MB RAM?

Heritage - What Didn't Work So Well

- Statically configured message bus and tables
  - Scenario: Guidance Navigation and Control (GN&C) needs a new diagnostic packet
- Monolithic load (The "Amorphous Blob")
  - Raw memory loads and byte patching needed to keep bandwidth needs down
  - Modeling tools did not support loadable objects
- Reinventing the wheel
  - Mission-specific "common" services
  - Desire vs. require to "optimize" for each mission
- Application rewrites for different operating systems
- Changes rippled through development, test, and documentation artifacts
  - Most artifacts were "clone and own"
- Claims of high reuse, but it still took the same effort on each mission

Key Trades

Architecture Trade: Inter-application Communication

- Evaluated CCSDS Asynchronous Message Service (AMS) and COTS Network Data Distribution Service (NDDS)
- Decided on custom implementation of a publish-subscribe message bus using a CCSDS international standard packet definition
  - Compatible with existing ground systems
- Low system coupling
  - Publishers send messages without knowledge of subscribers (Destination agnostic)
  - Any application can receive/listen to any packet
  - Stateless peer-to-peer network simplifies dynamic reconfiguration and resource management
  - Robust/Fault tolerant (no master)
- Ground systems, and simulation applications look like any other component/node
  - External interfaces can be “gatewayed” and firewalled
- Adaptable and extendible
  - Extendible attributes: Identifier, time, sequence number, and length
  - Components can be configured to limit command sources

Architecture Trade: File Systems

- **Challenges and considerations**
  - No GSFC missions had flown a file system
  - File systems are a well supported abstraction for data storage
  - Standard file transfer mechanisms (TFTP, FTP, CFDP)
  - Operating system support across most vendors, but inconsistent performance
  - Lots of resistance to added complexity
- **Trade Result**
  - Use files for code, table data, and recorder
  - Use vendor-supplied file system
- **Consequences**
  - First NASA Goddard cFS mission used VxWorks file system with custom enhancements
  - Funded Real-Time Executive for Multiprocessor System (RTEMS) file system enhancements
  - Evaluated JPL's volatile memory file system RAMFS but haven't flown it

Architecture Trade: Linking

- **Dynamic linking**
  - Requires symbols tables on board
  - Code files, Executable and Linkable Format (ELF) about double in size
  - More efficient use of memory
  - Can map around bad memory blocks (Memory Management Unit required)
- **Static linking**
  - No on board symbols
  - Small code files (stripped ELF)
  - Absolute location for each software component
  - Need to add margin around component memory space
- **Trade result:**
  - The architecture will support both
  - Open source RTEMS now has support for both (GSFC funded)

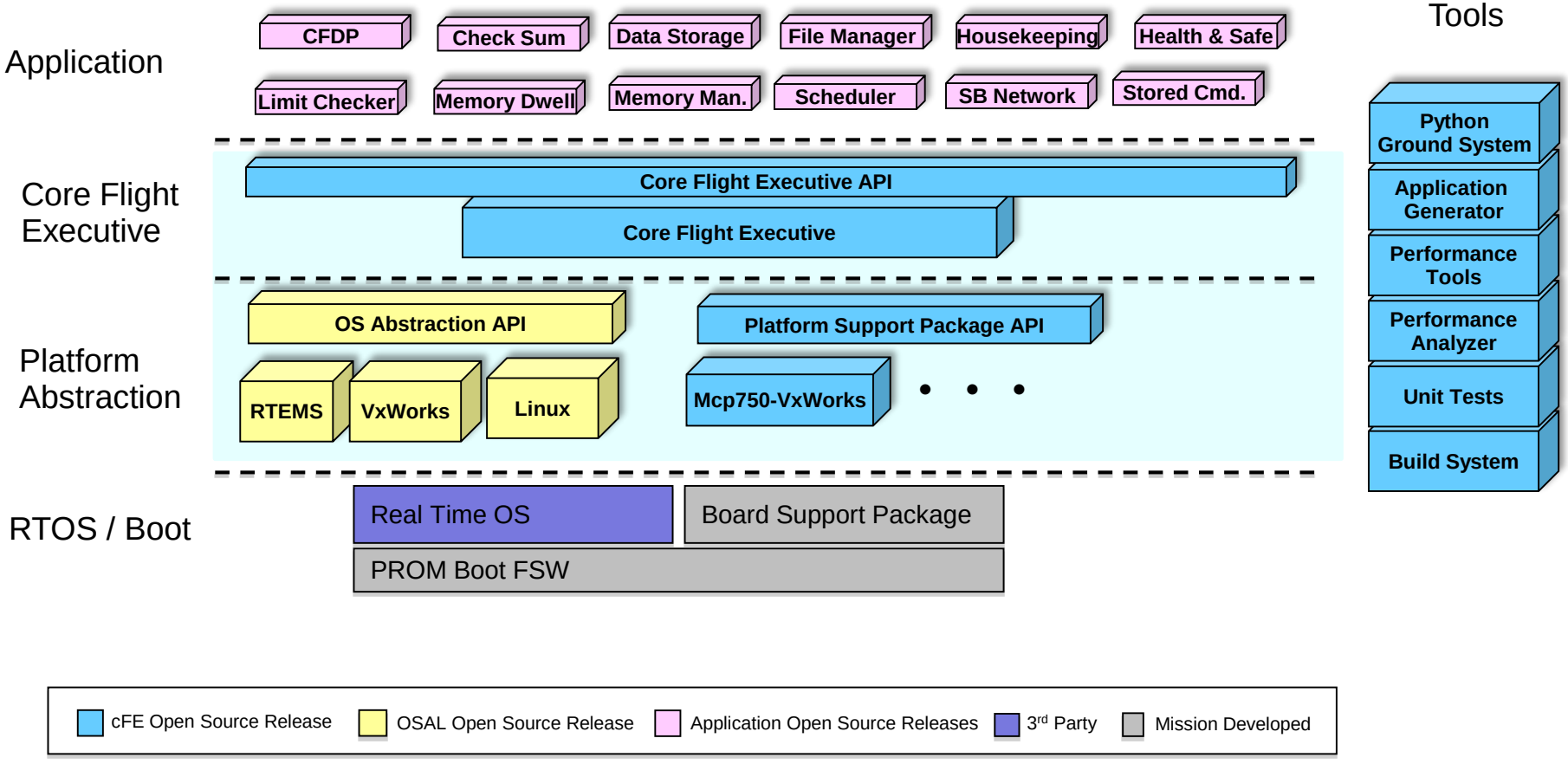
Final Architecture

Architecture Goals

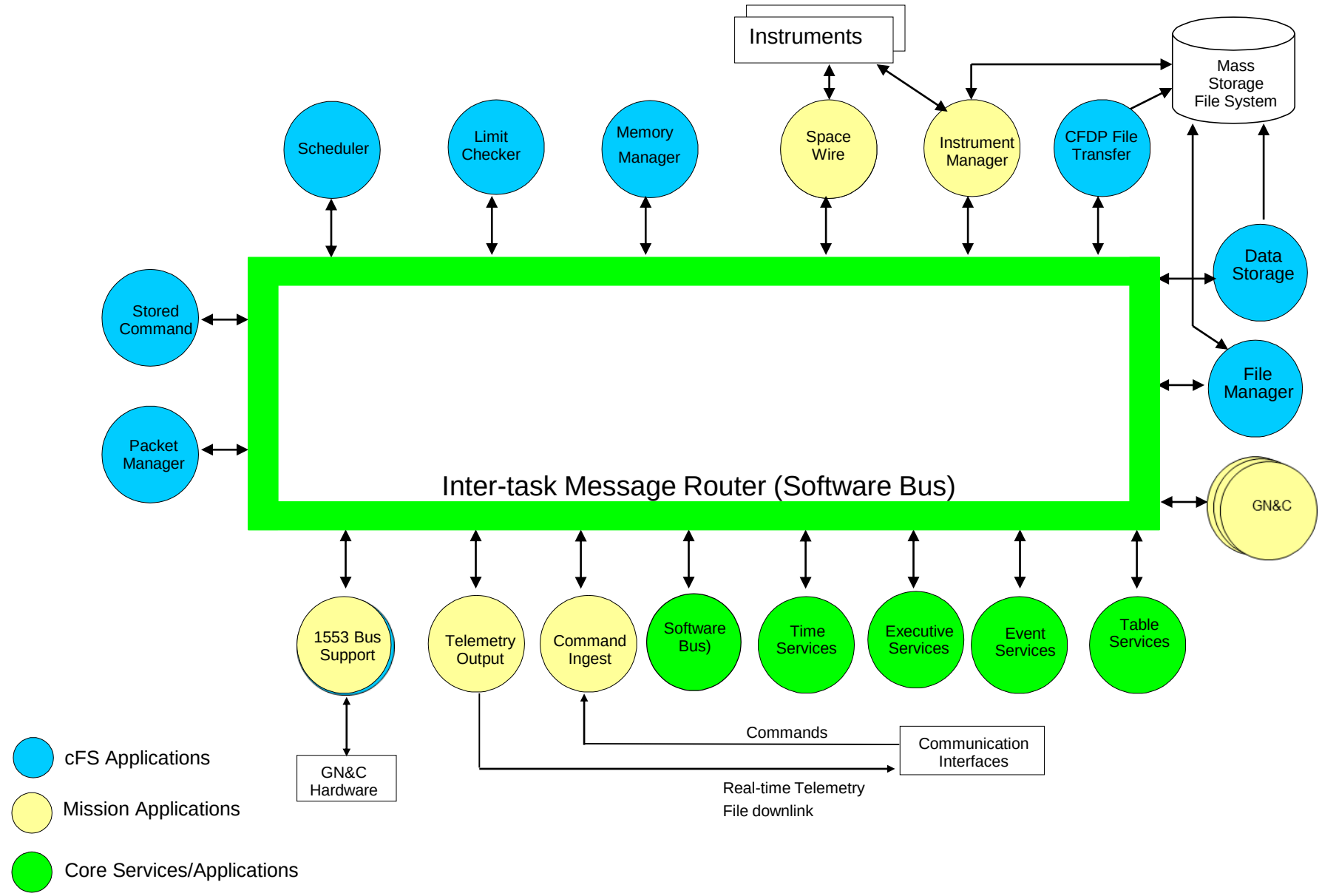
1. Reduce time to deploy high quality flight software
2. Reduce project schedule and cost uncertainty
3. Directly facilitate formalized software reuse
4. Enable collaboration across organizations
5. Simplify sustaining engineering (i.e. On Orbit FSW maintenance)
   Missions last 10 years or more
6. Scale from small instruments to Hubble class missions
7. Build a platform for advanced concepts and prototyping

These goals were written in 2006 and have remained essentially unchanged over the years!

cFS Architecture



Example Mission Applications



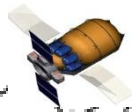
Applications

- **Write once run anywhere the cFS framework has been deployed**
- **15 Goddard applications released as open source that provide common command and data handling functionality such as**
  - Stored command management and execution
  - Onboard data storage file management
- **Reduce project cost and schedule risks**
  - High quality flight heritage applications
  - Focus resources on mission-specific functionality
- **Framework provides seamless application transition from technology efforts to flight projects**

User and Business Communities

Worldwide cFS Community

DOD and US industry
- Potential for standardization though the CCSDS and the Space Universal MOdular Architecture (SUMO) team sponsored by Office of the Director of National Intelligence



GRC – CPST and Advanced suit

APL - RBSP. Proposing use on Solar Probe, DoD programs.

ARC- LADEE

Kirtland AFB – Onboard Autonomous Planning System

LRO, MMS, GPM, NICER, OPIS and many others.

Commercial - Moon Express (Lunar X-Prize)

JPL – Evaluating architecture for robotic missions and ESTO missions

JSC-Used Successfully on Morpheus. Using on AES projects , Habitats, Waypoint, Certified for Class A (human rated).

MSFC- Mighty Eagle Lander, AES RESOLVE

KSC-Evaluating for AES, sounding rockets and UAV's

JAXA's Engineering Digital Innovation Center
Next generation software architecture research

Korea Aerospace Research Institute
Lunar program

European Space Research and Technology Centre

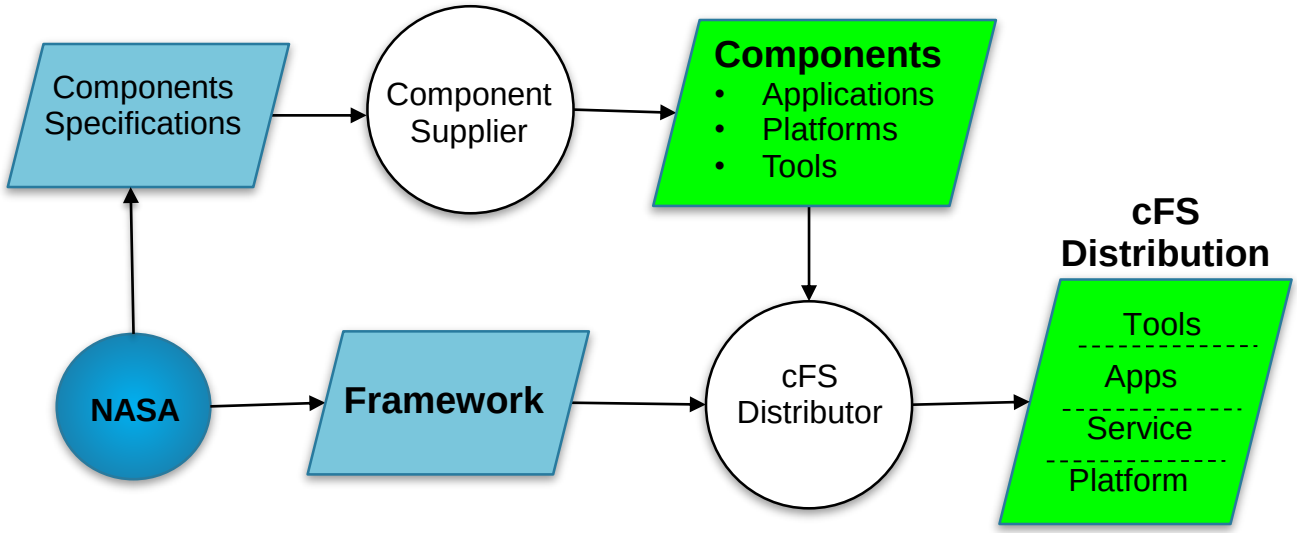
The Power of Community

1993 - Microsoft releases digital encyclopedia called Encarta

2001 - Wikipedia launched

2009 - Microsoft terminates Encarta

Community-based Product Model



- NASA configuration control board releases the open source cFS framework and publishes component specifications
- Community members
  - Supply applications, platforms, and tools
  - Create cFS distributions

Global Community Challenges

- **Some artifacts were developed for Goddard-specific environments and have not been transformed to a general purpose solution**
  - Table Tools
  - Build test scripts

- **Challenges with government run open source programs**
  - Funding
  - Software release processes and licensing

- **The product model and community infrastructure is immature**
  - Online component and distribution catalogs do not exist
  - cFS mailing list used as primary Q&A forum
  - Without clear “rules of engagement” component suppliers and distributors will not commit resources

Conclusion

- Special thanks to the Korean Astronomy and Space Science Institute

- Questions?

Backup Slides

- cFS NASA home page
  - https://cfs.gsfc.nasa.gov
- cFS Community
  - http://coreflightsystem.org
- cFS Program Manager, David McComas
  - david.c.mccomas@nasa.gov

Acronyms

API	Application Programmer Interface
ARC	Ames Research Center
BAT	Burst Alert Telescope
CCSDS	Consultative Committee for Space Data Systems
CDH	Command Data Handling
CFDP	CCSDS File Delivery Protocol
cFE	core Flight Executive
cFS	Core Flight System
CMMI	Capability Maturity Model Integrated
ELOC	Estimated Lines of Code
FSW	Flight Software
GLAS	Geoscience Laser Altimeter System
GN&C	Guidance, Navigation, and Control
GPM	Global Precipitation Measurement
GSFC	Goddard Space Flight Center
JSC	Johnson Space Center
LADEE	Lunar Atmosphere and Dust Environment Explorer
LRD	Launch Readiness Date
LRO	Lunar Robotic Orbiter
MAP	Microwave Anisotropy Probe
MMS	Magnetic Multiscale Mission
OSAL	Operating System Abstraction Layer
RBSP	Radiation Belt Storm Probe
RTEMS	Real-Time Executive for Multiprocessor Systems
SAMPEX	Solar Anomalous and Magnetospheric Particle Explorer
SARB	Software Architecture Review Board (NASA Engineering and Safety Center)
SDO	Solar Dynamics Observatory
SMEX	Small Explorer
ST-5	Space Technology 5
SWAS	Submillimeter Wave Astronomy Satellite
TRACE	Transition Region and Coronal Explorer
TRL	Technology Readiness Level
TRMM	Tropical Rainfall Measuring Mission
WIRE	Widearea Infrared Explorer
XTE	X-Ray Timing Explorer