National Aeronautics and Space Administration

Core Flight System (cFS) Training

Introduction

Objectives and Intended Audience

- **Introduce flight software (FSW) concepts and core Flight System (cFS) architectural features**
  - Spacecraft technical managers, systems engineers, discipline engineers, and software engineers
  - No programming skills required

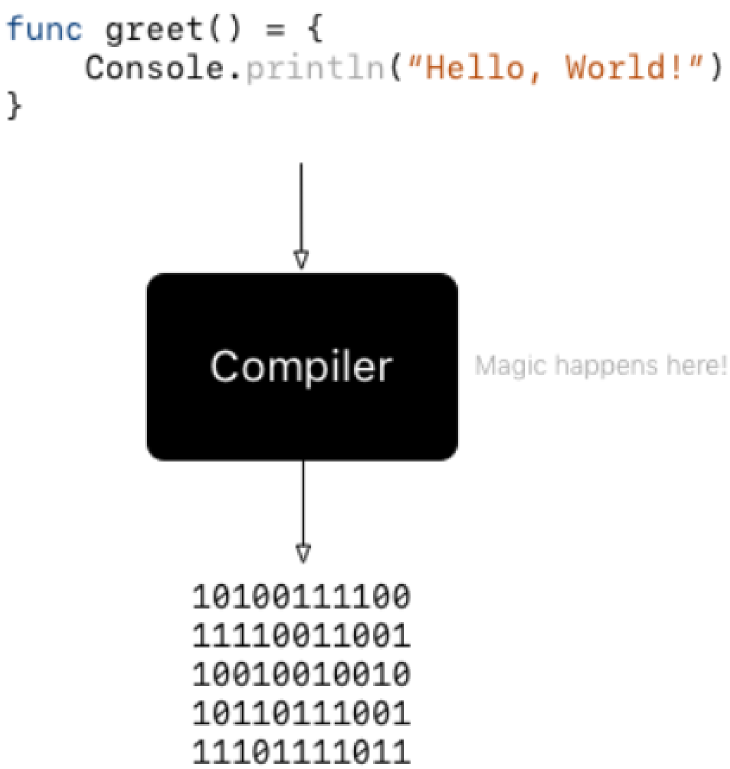
Introduction Agenda

- Introduction to Flight Software
- cFS Business Context
- cFS Architecture Overview
- FSW Engineering with the cFS

Introduction to Flight Software

What is Software?

- Microcode are instructions and data processed by a microprocessor
- Programming languages are formal languages (i.e. grammatical rules) that are translated into microcode by a compiler



- Interpreted languages such as Python, Ruby, and JavaScript run on a virtual machines (VMs) and the VM is ported to different processors

What is an Embedded Computer?

- An *Embedded Computer* is a computer that is integrated in a product
- Embedded Computers do not usually have a keyboard, mouse, or monitor interface

Embedded Software is software that that runs on an Embedded Computer in order to make a device or product work

"Embedded software is the opposing thumb of hardware"

Embedded Software Characteristics

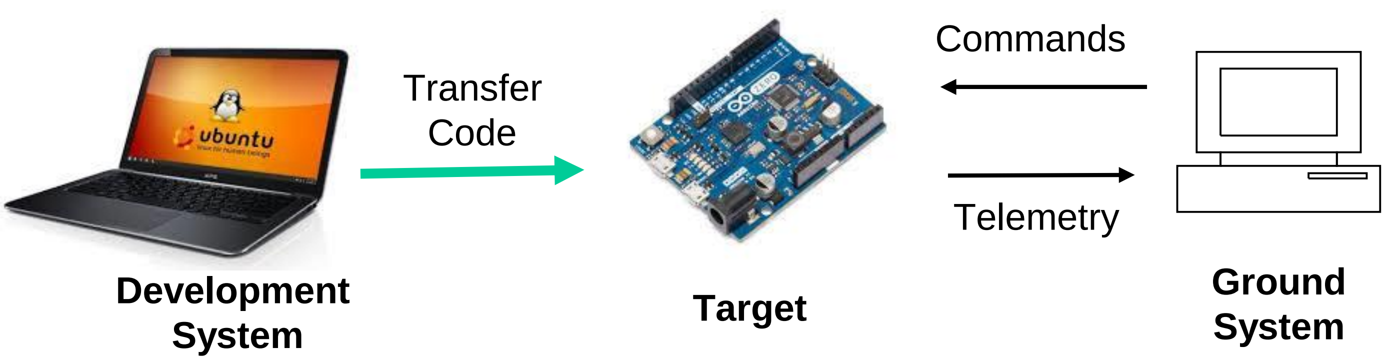
- **Microcontroller**
  – Typically limited resources: low power, small memory, and slower clock speeds
  – Typically no operating system
  – For example, the processor embedded with reaction wheel assembly

- **Embedded Operating System**
  – Real-time preemptive multi-tasking
  – Typically much smaller than Windows or OS X
  – Examples include VxWorks, RTEMS, and FreeRTOS

- ***Embedded System***
  – The combination of an *Embedded Computer* and *Embedded Software*

How do we Develop Embedded Software?

- **Use a Windows, Mac, or Linux computer to write the code**
  - Programming Languages used
    - For Flight Software: C, C++, Assembly Language, sometimes Ada
    - For Ground and test software: C, C++, Python, Java, Ruby, etc.
- **Write code, cross-compile for target processor, transfer object code to embedded computer, control embedded system with a ground system**
- **This is known as Cross Development**



Spacecraft Definitions

- **A few terms:**
  - **Spacecraft Bus** – usually refers to the fundamental systems of a spacecraft, i.e.
    - Mechanical Structure
    - Electrical System
    - Power System
    - Command and Data Handling System (C&DH)
    - Attitude Control System/ Propulsion System
    - RF System
    - Thermal System
  - **Payloads** – refers to the instruments on board, i.e.
    - Cameras, Telescopes, Radars, etc
  - **Observatory** – Usually refers to the entire system, i.e. the combination of the Spacecraft Bus and the Payloads

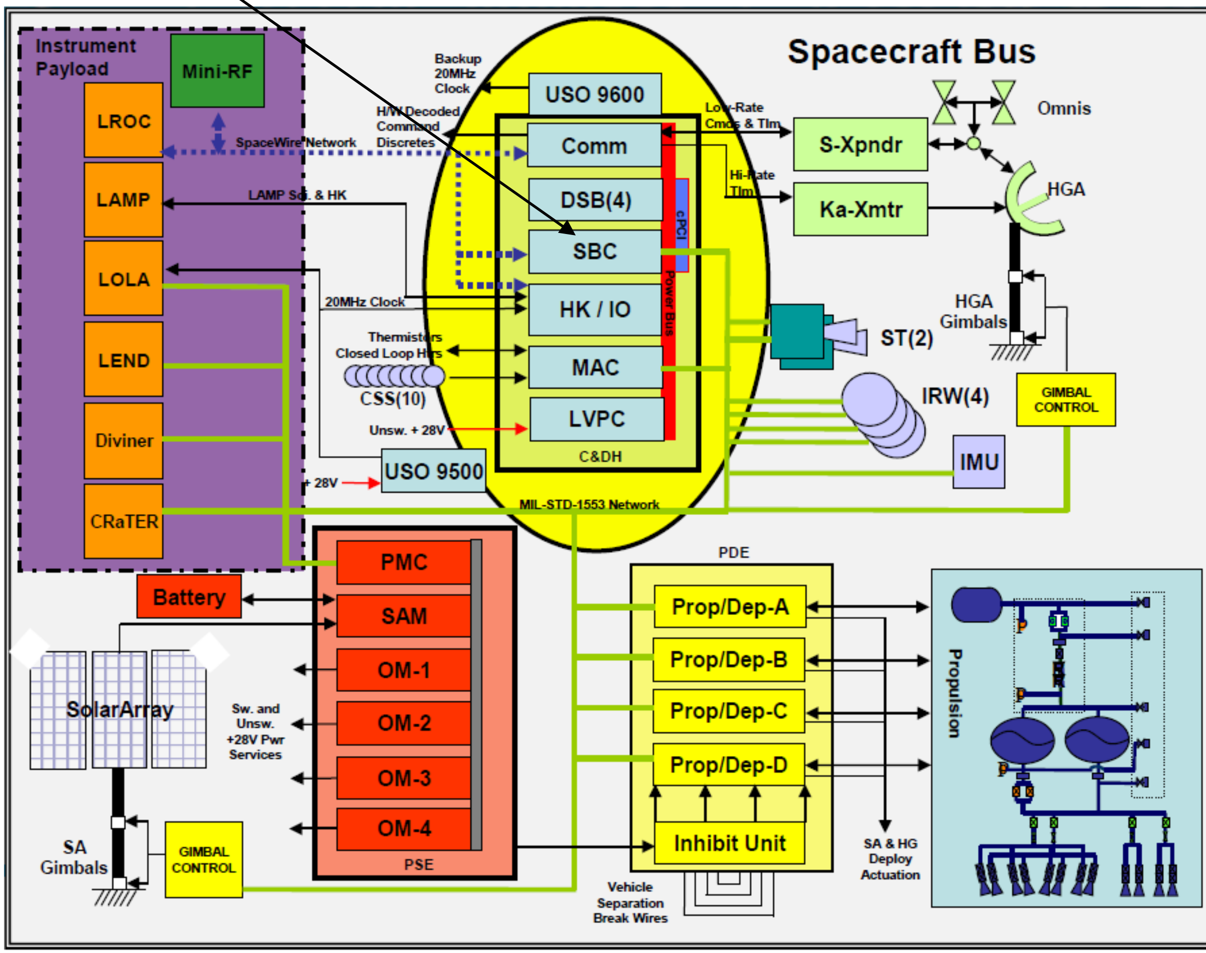
FSW Challenges – Limited Resources

- **Flight processor clock speeds are in the MHz range while our laptops are in the GHz range**
  - NICER (6/12/17) 83Mhz Broad Reach 440 Power PC
- **Non-volatile memory**
  - GPM (2/27/14) 2MB banks of EEPROM for each FSW image
  - Compressed operating system and apps
- **Flight RAM in MegaByte range while our laptops are in the GigaByte range**
  - GPM (2/27/14) 36MB RAM

FSW Challenges – Large number of Custom Interfaces

More than 20 Interfaces controlled by FSW on this single string example

FSW



C&DH	Command & Data Handling
Comm	Communication
CRaTER	Cosmic Ray Telescope for the Effects of Radiation
CSS	Coarse Sun Sensor
DSB	Data Storage Board
EVD	Engine Valve Driver
HGA	High-Gain Antenna
H/W	Hardware
HK	Housekeeping
IO	Input/Output
IMU	Inertial Measurement Unit
IRW	Integrated Reaction Wheel
LAMP	Lyman-Alpha Mapping Project
LEND	Lunar Exploration Neutron Detector
LOLA	Lunar Orbiter Laser Altimeter
LROC	Lunar Reconnaissance Orbiter Camera
LVPC	Low Voltage Power Converter
MAC	Multi-Function Analog Card
OM	Output Module
PMC	Power Management Controller
PSE	Power System Electronics
SA	Solar Array
SAM	Solar Array Module
ST	Star Tracker
SBC	Single Board Computer
SSR	Solid State Recorder
Xmtr	Transmitter
Xpndr	Transponder

FSW Challenges – Robustness & Remote Diagnostics

- **Curiosity Probe's Computer Reset (2/25/19)**
  - “Last week, its computer rebooted without warning. Now, NASA engineers are trying to figure out what caused the unprompted restart.”

- **Israeli Lunar Lander Suffers Glitch on Way to the Moon (2/27/19)**

  - "During the pre-maneuver phase the spacecraft computer reset unexpectedly, causing the maneuver to be automatically cancelled,“
  - "The engineering teams … are examining the data and analyzing the situation. At this time, the spacecraft's systems are working well, except for the known problem in the star tracker.“
  - https://www.space.com/israel-moon-lander-suffers-glitch.html

FSW Challenges – Manage & Protect Hardware

- Detect flight hardware anomalies and failures
  - Sensors, Actuators, Clocks, CPU, Memory, Power, Thermal, Communications

- Respond to onboard anomalies/failures – pre-planned actions, e.g.,
  - Capture Diagnostic Data
  - Use “numerically safe” data for current control law cycle
  - Reconfigure onboard hardware and/or software – minimize the problem
  - ‘Safe’ all Flight Hardware (Science Instruments, Spacecraft Hardware)
  - Ensure Sun is on the Solar Array Panels -- Maintain Power Positive State
  - Notify Ground of problems

- FSW can often compensate for hardware problems found during spacecraft I&T or post-launch

FSW Challenges – Serving Multiple Customers

- **FSW "presents" the spacecraft to the end-user and there are multiple end users therefore a FSW expert should be involved from early formulation stages to**
  - Participate in trades: ground-/flight, hardware/software, budget options, etc.
  - Champion requirements, design, and interface features with knowledge of all of the customers and spacecraft lifecycle needs
- **Who are FSW Customers?**
  - Principle Investigator
    - "I want all my data and I want it now!"
  - Spacecraft Operators (work under a management plan)
    - "Make my life simple and cheap"
  - Other spacecraft subsystems
    - "It would be really nice if I could see X"
  - FSW Test Team
    - "I want to see all of the data all of the time"
  - Spacecraft Integration and Test team
    - "I want to see or at least log all of the data all of the time"
    - "I also need to test as we will fly"
  - FSW Maintenance Team
    - "Can I access/view X and change Y?"

Typical Spacecraft Flight Software Requirements

FSW

C&DH

- **Operational Environment**
  - App Runtime Env
  - Command Ingest
  - Telemetry Output
  - Inter-processor Comm
- **Onboard Data Management**
  - File Management
  - File Transfer
  - Recorder Management
- **Autonomy, Failure Detection & Correction**
  - Stored Commanding
  - Hardware & Software Monitoring
  - Memory Integrity Support & Checks

ACS-GNC

- **Attitude Determination and Control**
- **Orbit Determination and Control**
- **Models**
  - Solar
  - Lunar
  - Magnetic Field

Mechanisms

- **Deployments**
- **Solar Array Control**
- **Antenna Control**

Thermal

- **Auton Control**
- **Tlm & Cmd I/F**
- **Health & Safety**

Power

- **Auton Control**
- **Tlm & Cmd I/F**
- **Health & Safety**

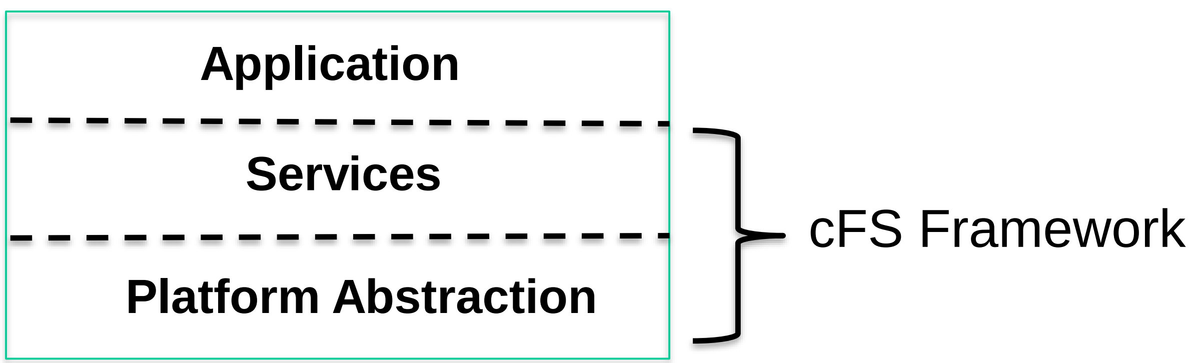
Payload

- **Auton Control**
- **Tlm & Cmd I/F**
- **Health & Safety**

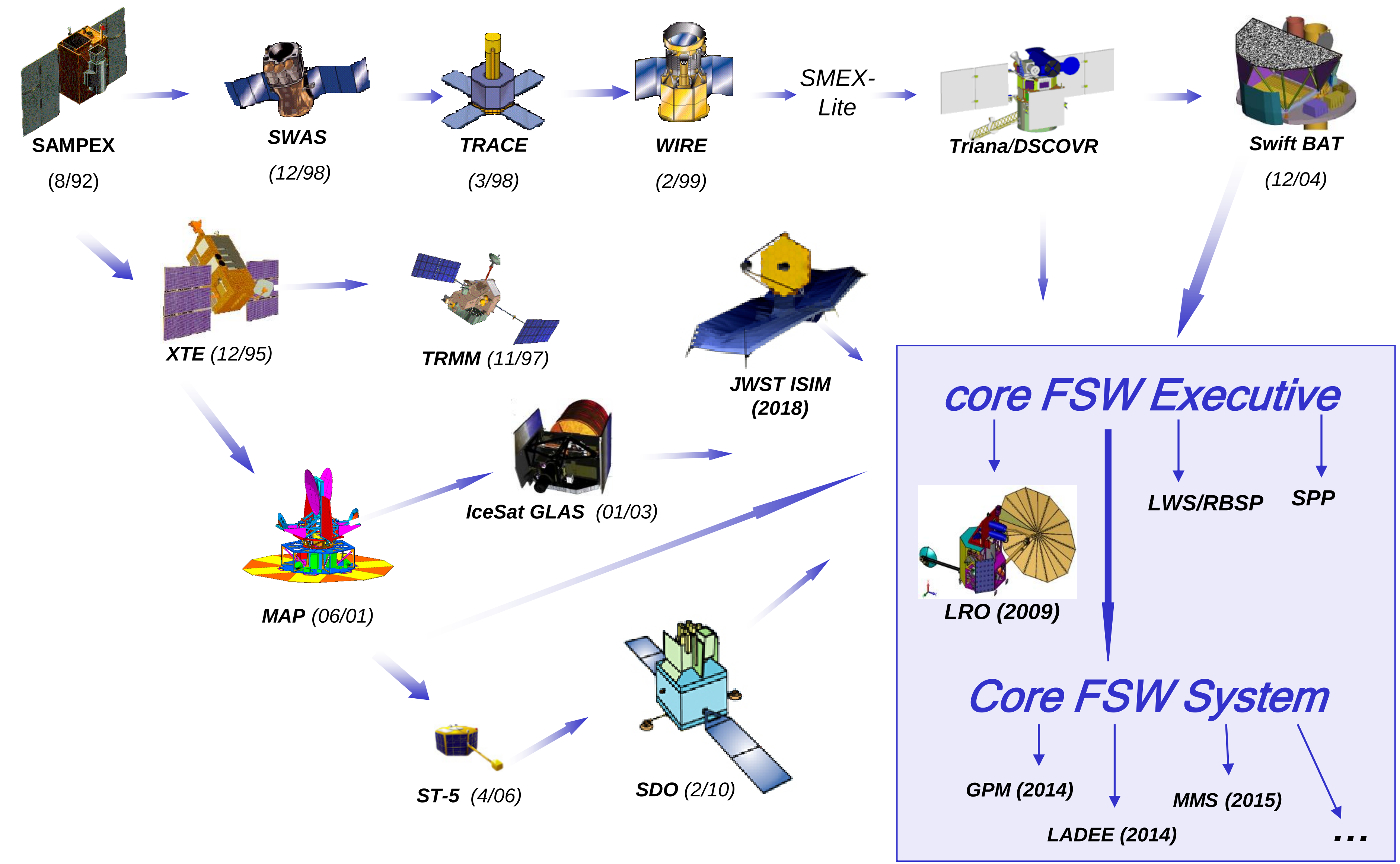
cFS
Business Context

Core Flight System Overview

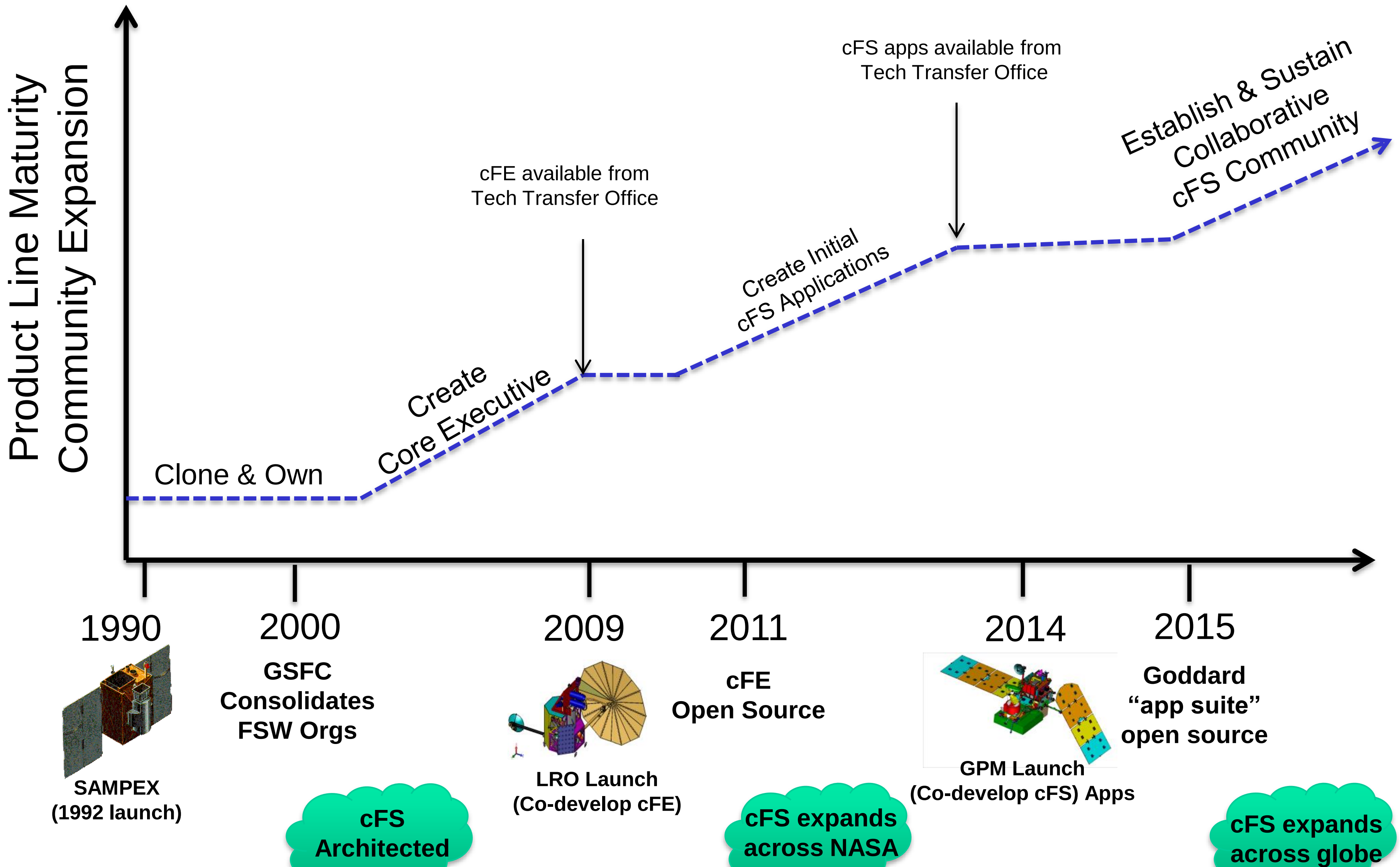
- **The cFS uses a three-tiered software architecture that provides a <u>portable</u> and <u>extendable</u> framework with a <u>product line deployment model</u>**
  - Platform Abstraction Layer supports portability
  - Applications provide mission functionality
  - Compile-time configuration parameters and run-time command/table parameters flexibility and scalability



cFS Architecture Heritage



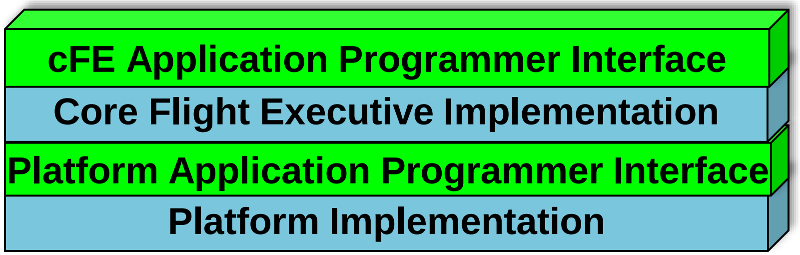
cFS History



Product Line Maturity
Community Expansion
Clone & Own
Create
Core Executive
cFE available from
Tech Transfer Office
Create Initial
cFS Applications
cFS apps available from
Tech Transfer Office
Establish & Sustain
Collaborative
cFS Community
1990
SAMPEX
(1992 launch)
2000
GSFC
Consolidates
FSW Orgs
cFS
Architected
2009
LRO Launch
(Co-develop cFE)
2011
cFE
Open Source
cFS expands
across NASA
2014
GPM Launch
(Co-develop cFS) Apps
2015
Goddard
“app suite”
open source
cFS expands
across globe

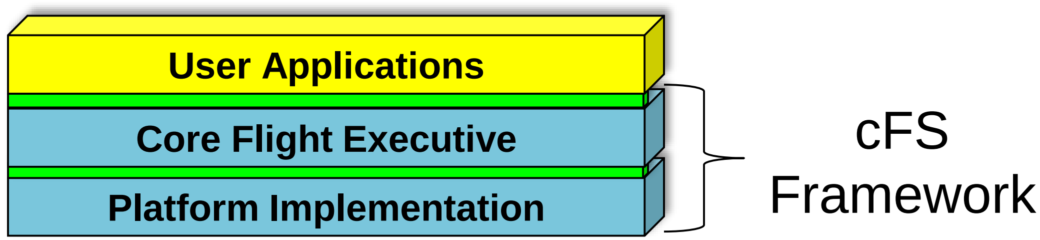
What is the Core Flight System (cFS) ?

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



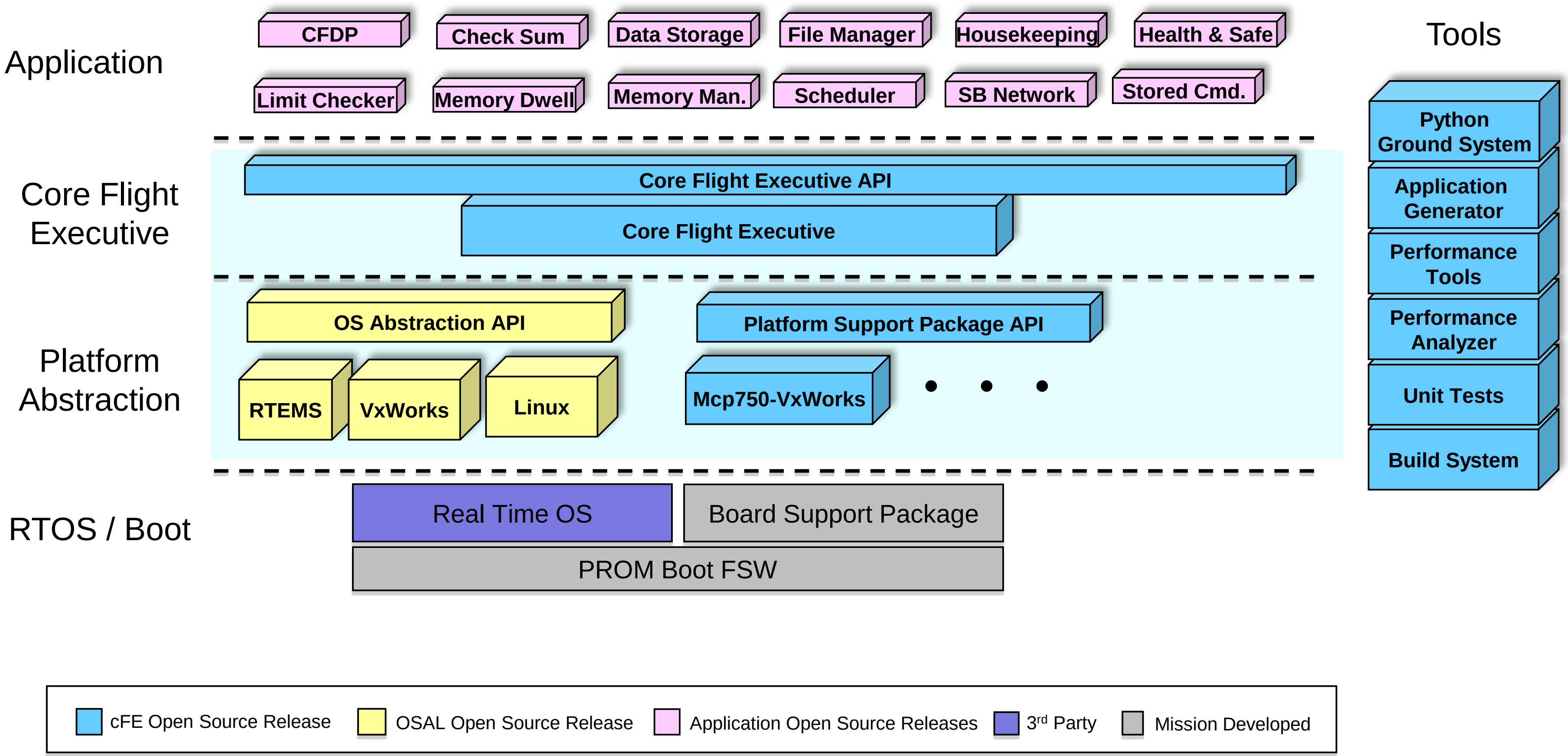
- Layered architecture with international standards-based interfaces
- Provides development tools and runtime environment for user applications
- Reusable NASA Class A/B lifecycle artifacts: requirements, design, code, tests, and documents

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

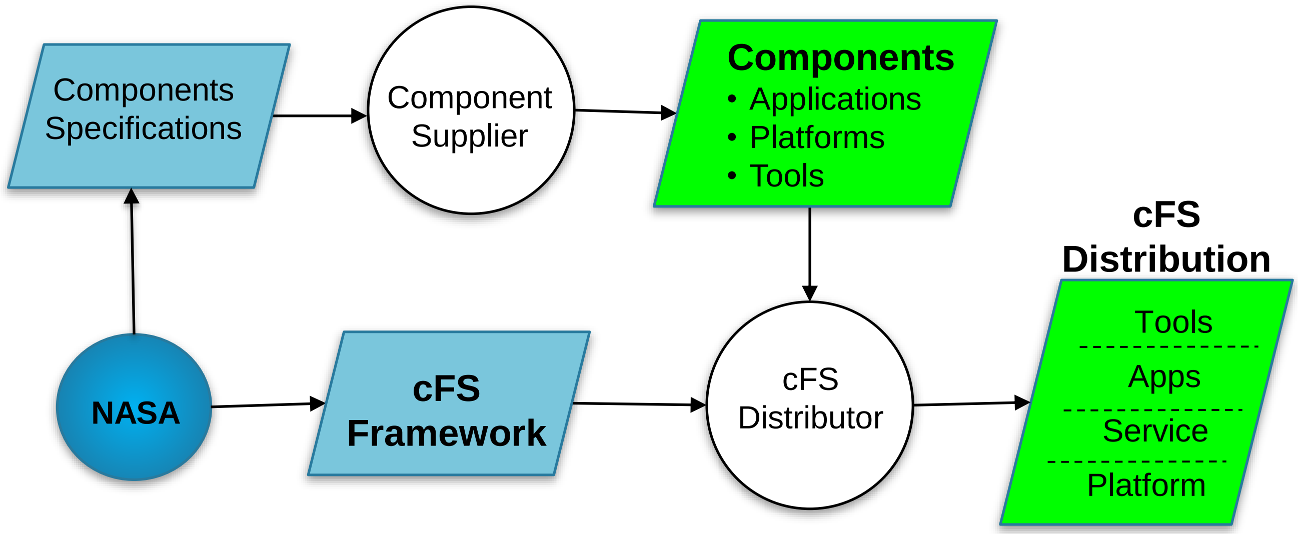


- **A worldwide community from government, industry, and academia**

cFS Architecture Layers



Community-based Product Model



- A NASA multi-center configuration control board (CCB) manages releases of the open source cFS Framework and component specifications

- Community members (regardless of affiliation)
  - Supply applications, platforms, and tools
  - Create cFS distributions

Community-based Product Model

- **Community component supplier value proposition**
  - As the number of supported platforms increases then apps become more valuable
  - As the number of apps increases then supporting a cFS platform become more valuable
- **In 2019 vendors starting to offer processor boards integrated with the cFS**
  - AI Tech partnered with Embedded Flight Systems to offer the cFS integrated on the SP0-S Single Board Computer
  - Genesis Engineering developing an integrated GEN6000 (SpaceCube 2.0) cFS product
  - Genesis pursuing a Space Act Agreement (SAA) that would include the creation of a platform certification test suite
- **Community members release, maintain, and distribute (typically via git) their apps**
  - No one has established an “app store”

User Responsibilities

- **The cFS Framework is has a NASA NPR-7150.2C Class E classification**
  - **Class states Design Concept, Research, Technology and General Purpose Software**
  - The cFS Framework provides artifacts to support Class B missions and a subset of artifacts to support Class A missions
  - End-users are responsible for classifying the software system that uses the cFS Framework
  - This is consistent with the Apache license's "Disclaimer of Warranty" that states, *"…provides the Work…on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of … FITNESS FOR A PARTICULAR PURPOSE."*
- **End-users are responsible for complying with International Traffic in arms Regulations (ITAR)**

Obtaining cFS "Products"

- **cFS Bundle**
  - Contains the cFS Framework packaged with additional components to create a system that can easily be built, executed, and unit tested on a Linux platform
  - http://github.com/nasa/cFS
- **User Components**
  - Search https://github.com/nasa/ or do a general google on the cFS
- **Distributions**
  - Listed on the next slide
  - Some distributions contain many of the common apps which give you a good starting point for apps
- **Engage with the Community**
  - Ask the community mailing list (See backup slides)
    - Especially useful when porting to a new platform
  - Contact a cFS team member (See backup slides)

cFS Distributions

Name/Link	Intended Audience	Overview
cFS Framework-101	cFS Framework training package	This is a training tool for individuals to learn how to develop software with NASA-developed Core Flight software (CFS) framework. No agreement is necessary through this catalog. Software is available at open source site.
cFS Build	Initial cFS build for a developer or a project	This repository contains the core Flight Executive and a number of submodules including OSAL, example "lab" applications, and mission ready applications. This distribution has been compiled/linked but has not been verified as an operational system.
NASA Operational Simulator for Small Satellites (NOS3)	Initial cFS platform for a project	NOS3 provides a complete cFS system designed to support satellite flight software development throughout the project life cycle. It includes •42 Spacecraft dynamics and visualization, NASA GSFC • cFS – core Flight System, NASA GSFC • COSMOS – Ball Aerospace • ITC Common – Loggers and developer tools, NASA IV&V ITC • NOS Engine – Middleware bus simulator, NASA IV&V ITC
OpenSatKit (OSK)	cFS training platform for new cFS developers Initial cFS platform for a project	OSK provides a complete cFS system to simplify the cFS learning curve, cFS deployment, and application development. The kit combines three open source tools to achieve these goals: • cFS – core Flight System, NASA GSFC • COSMOS – command and control platform for embedded systems, Ball Aerospace • 42 dynamic simulator, NASA GSFC

Community Operational Procedures

- **Version Control**
  - Master Branch
  - Integration Candidates
  - Release Candidates

- **User Contributions**
  - Community Contribution process and Contributor License Agreement (CLA)

- **Feature Deprecation**
  - Mark feature as deprecated on any release
  - Provide tools/process that will warning applications when a feature is marked as deprecated
  - Only deprecate on major versions

Consequences of cFS Evolution

- **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**
  - Project-centric funding
  - Cumbersome software licensing and release processes

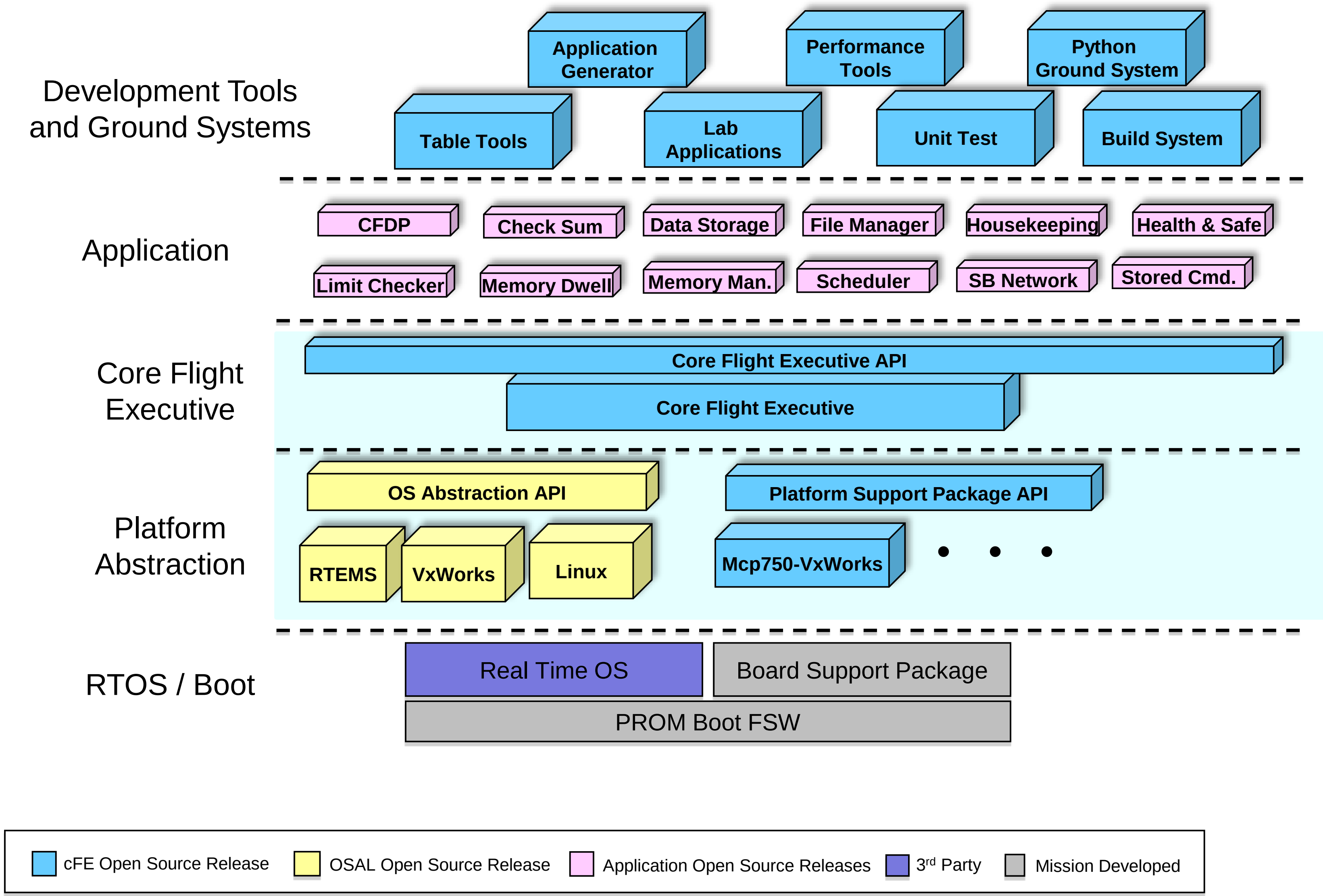
- **The product model and community infrastructure is immature**
  - Lack of online component and distribution catalogs
  - cFS mailing list used as primary Q&A forum

Core Flight System Architectural Overview

cFS Overview

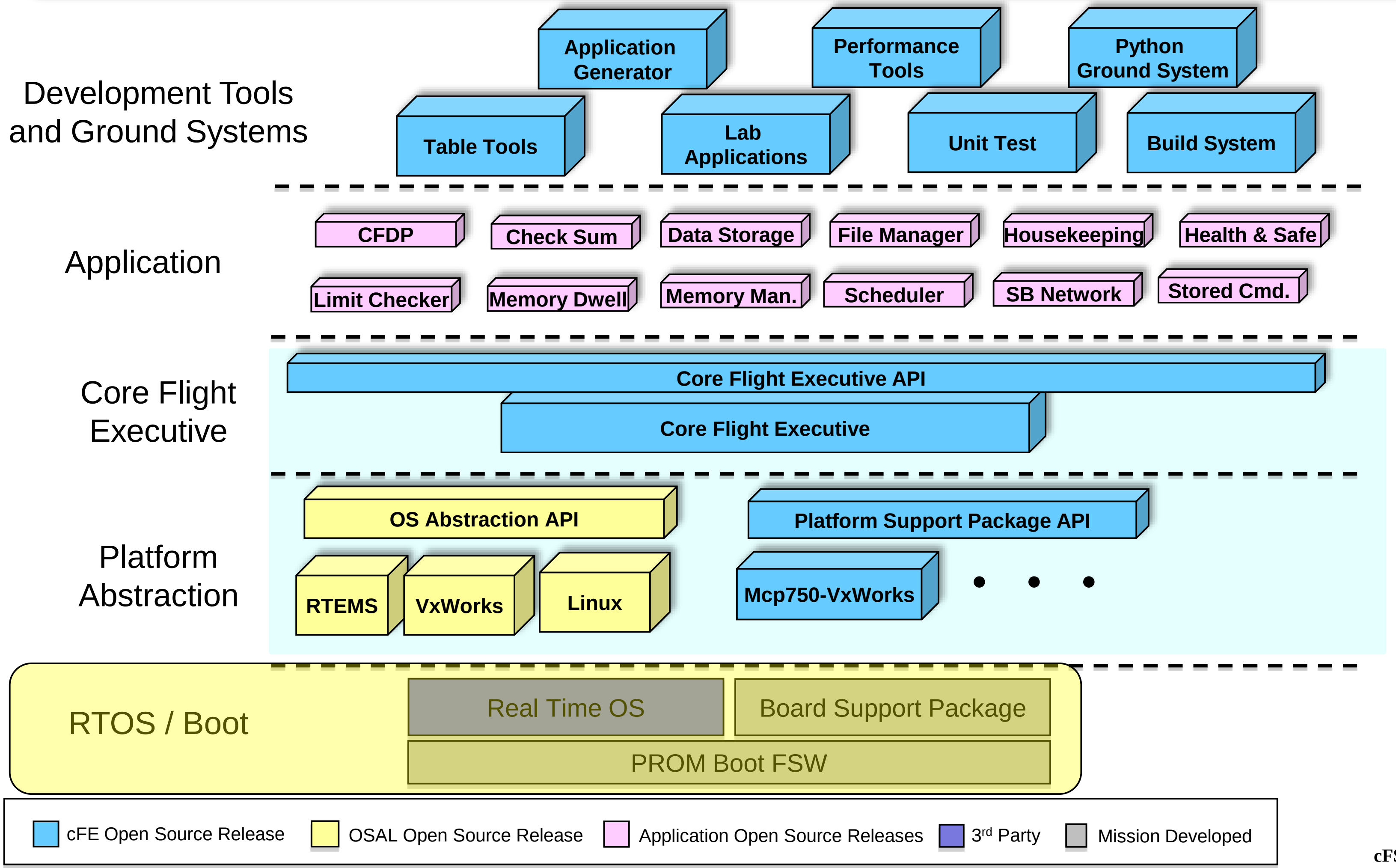
- **A set of** *mission independent, re-usable, core* **flight software services, applications, and operating environment**
  - Layered architecture
    - Supports a variety of hardware platforms
  - Provides standardized Application Programmer Interfaces (API)
  - Supports and hosts flight software applications
    - *Applications can be added and removed at run-time (eases system integration and FSW maintenance*)
  - Supports software development for on-board FSW, desktop FSW development and simulators
  - Contains platform and mission configuration parameters that are used to tailor to a specific platform and mission.

cFS Architecture Layers



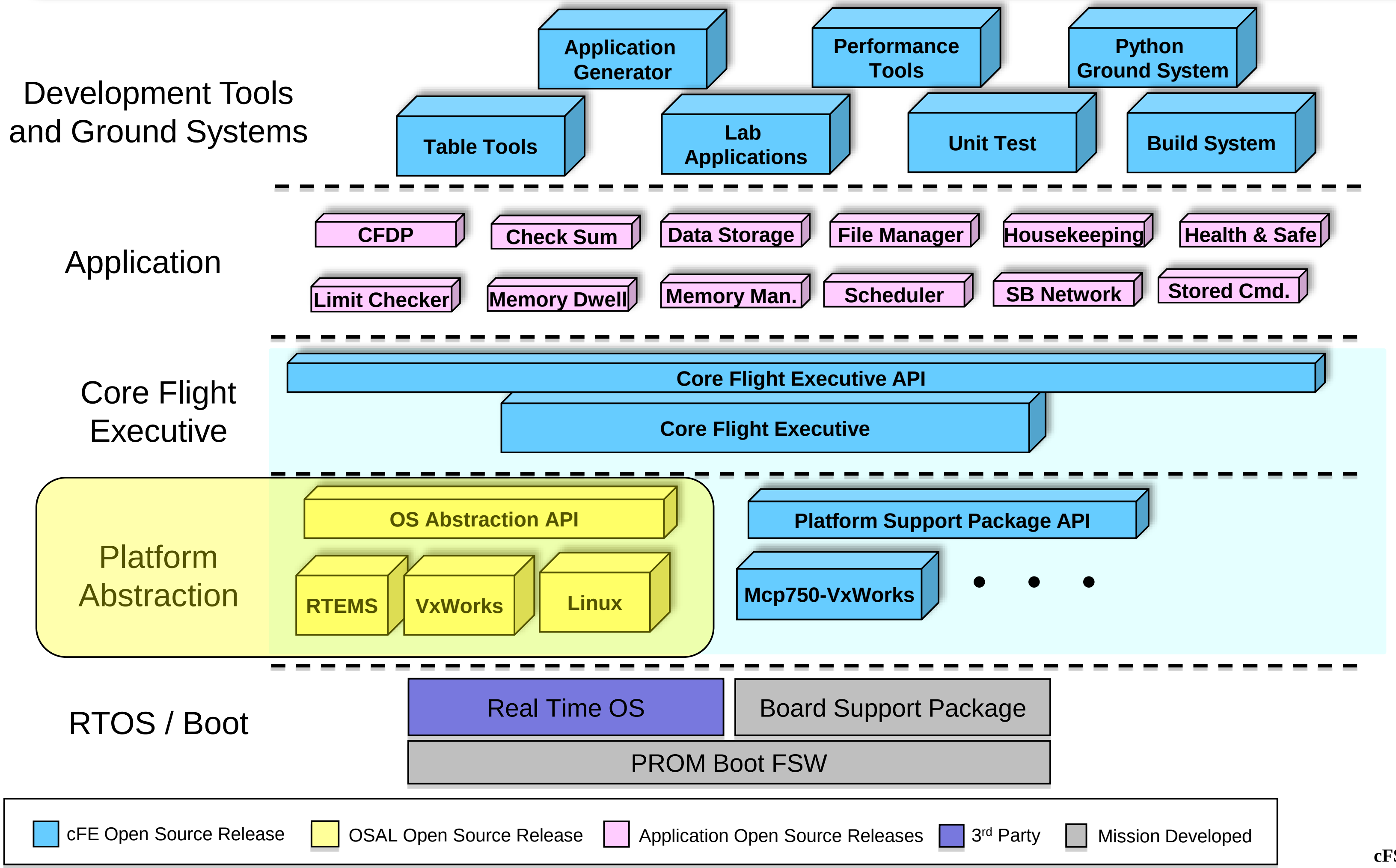
Operating System / Boot Layer

Provides the commercial, open-source, or custom software interface between the processor and the FSW. Real-time multi-tasking preemptive scheduling operating systems used for flight applications.



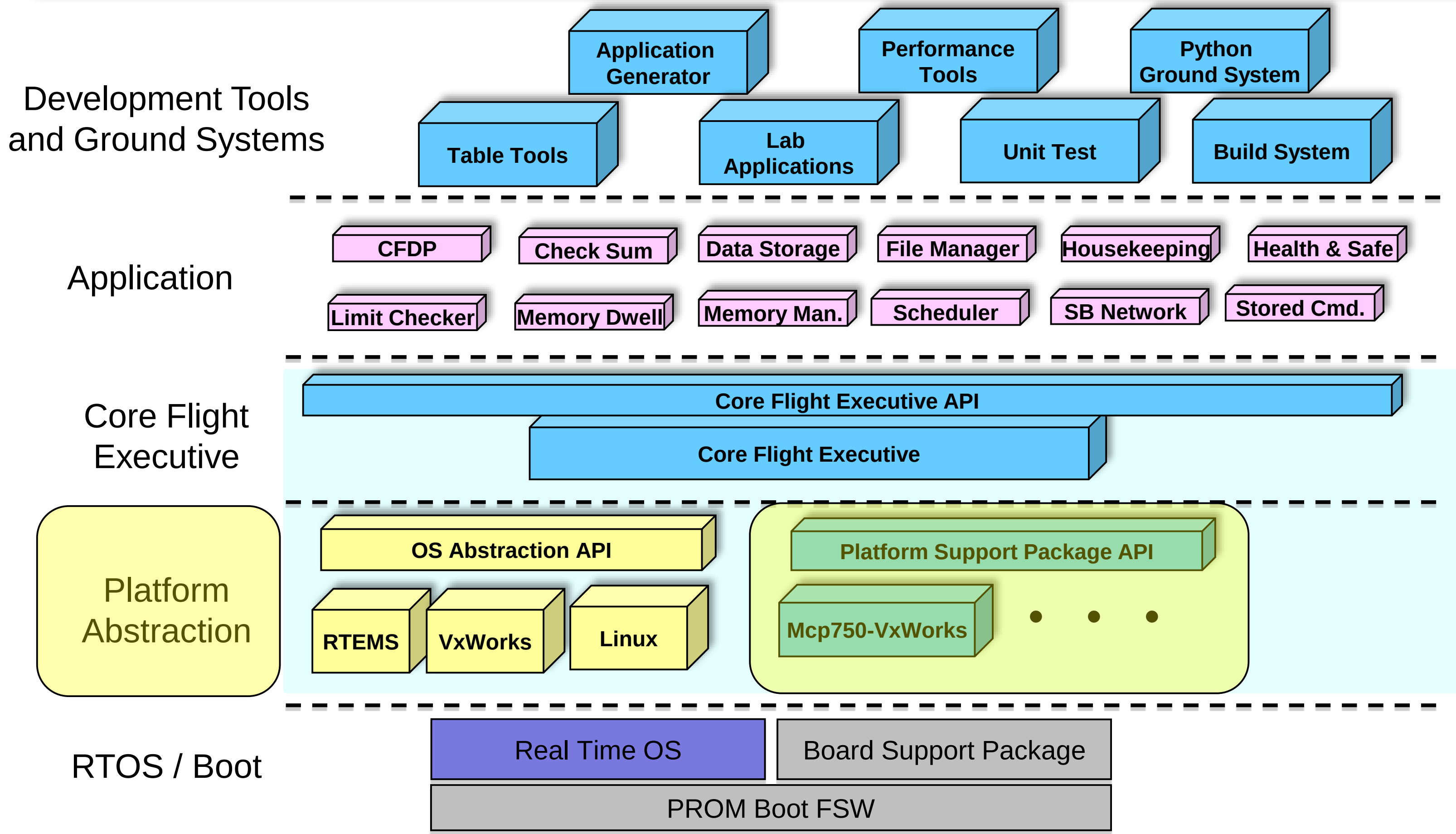
Platform Abstraction - OSAL

The OS Abstraction Layer (OSAL) is a software library that provides a single Application Program Interface (API) to the core Flight Executive (cFE) regardless of the underlying real-time operating system.



Platform Abstraction - PSP

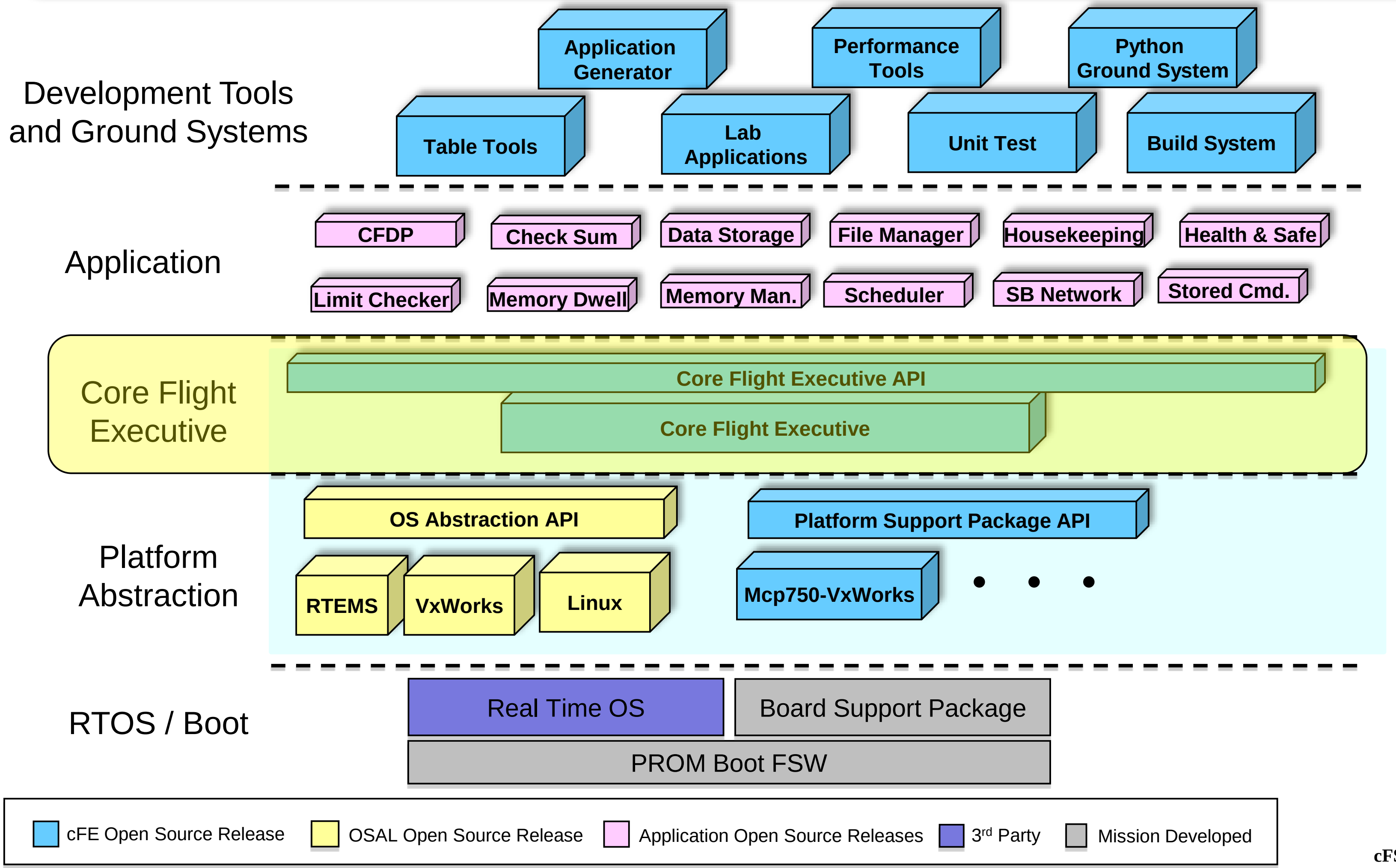
The Platform Support Package (PSP) is a software library that provides a single Application Program Interface (API) to underlying avionics hardware and board support package.



cFE Open Source Release | OSAL Open Source Release | Application Open Source Releases | 3rd Party | Mission Developed

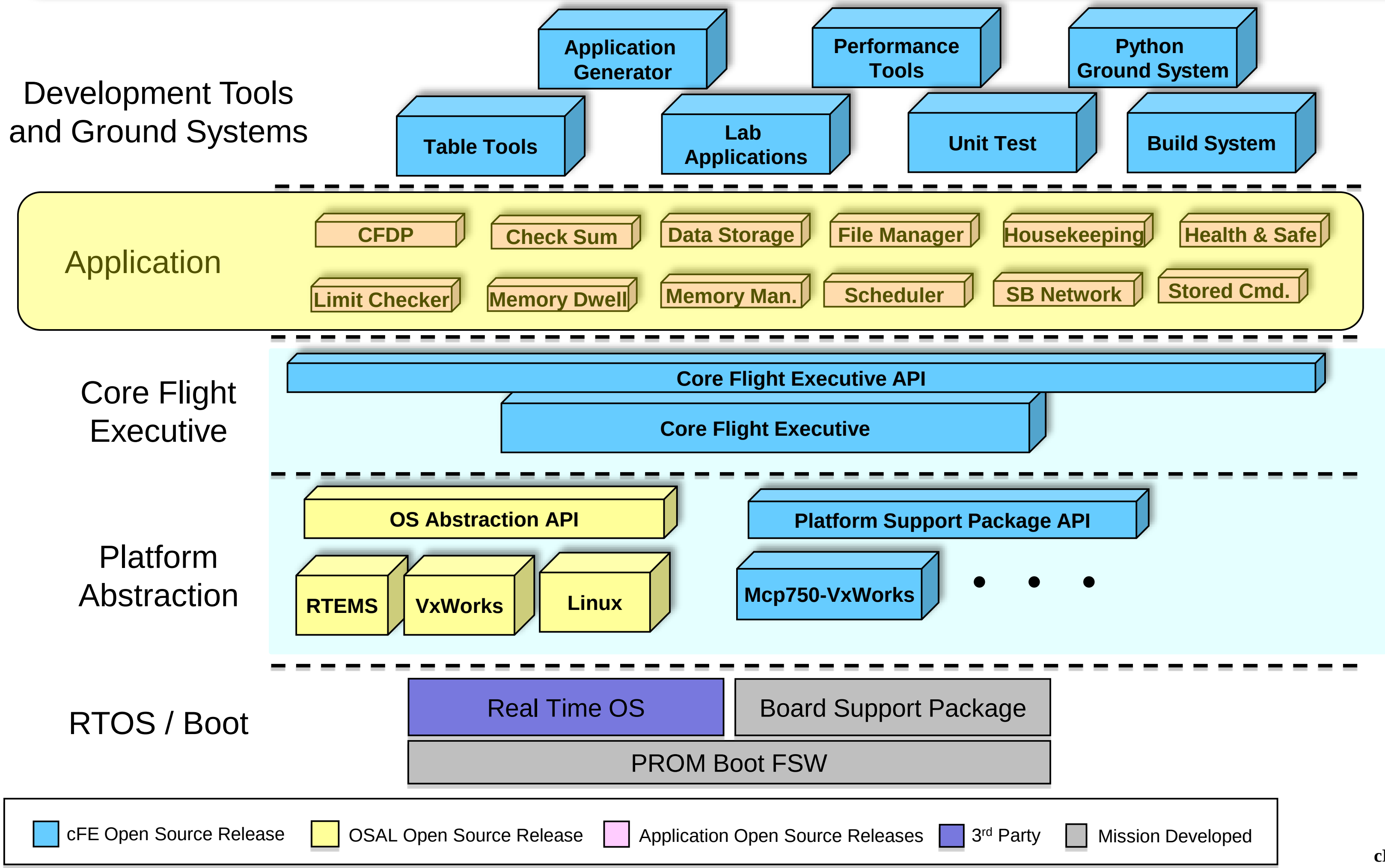
Core Flight Executive

The cFE is a portable, platform-independent framework that creates an application runtime environment by providing services that are common to most flight applications.

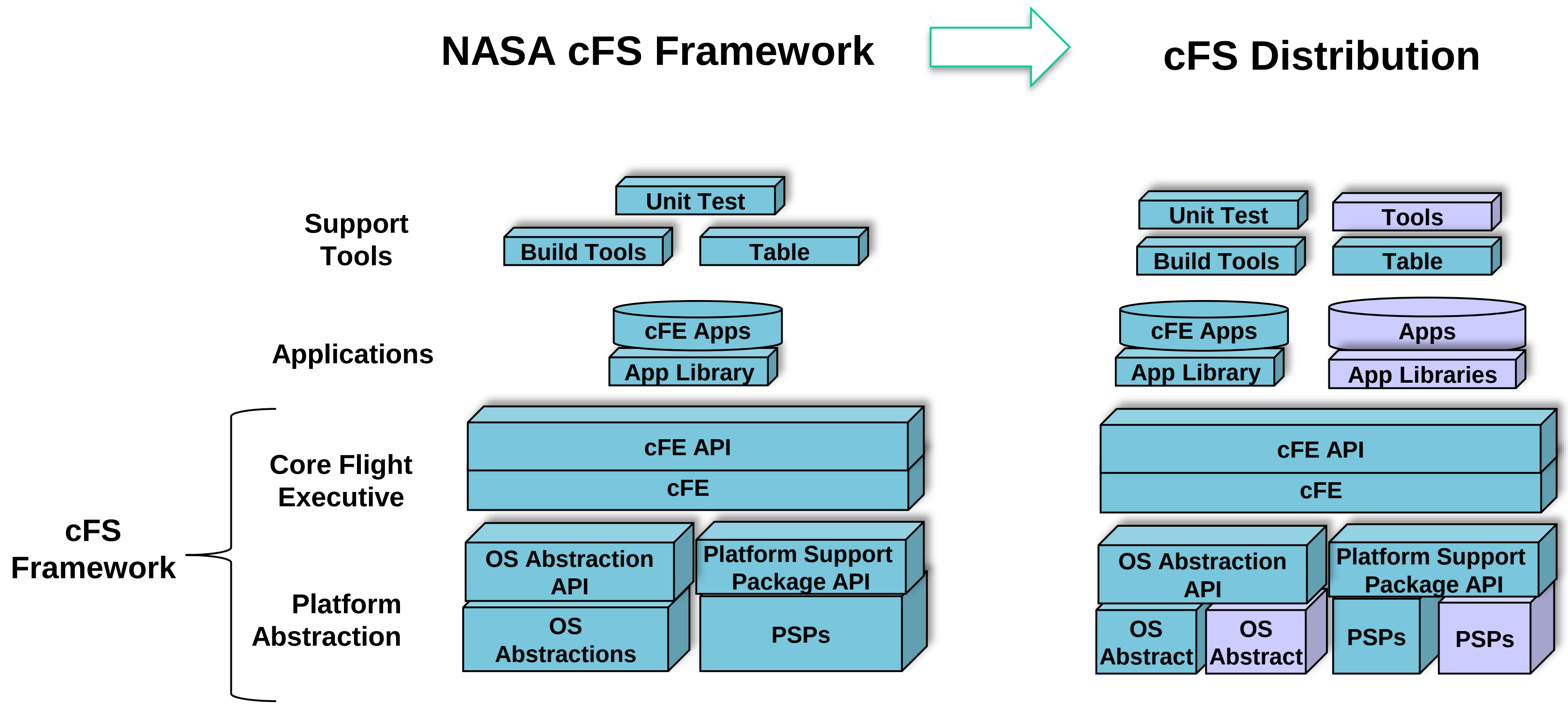


Applications

Applications provide mission functionality using a combination of cFS community apps and mission-specific apps.



cFS Product Model



- The NASA Configuration Control Board (CCB) manages the “cFS Framework”
- “cFS Distribution” created by augmenting the NASA cFS Framework with components (platforms, apps, and tools) to create an operational system

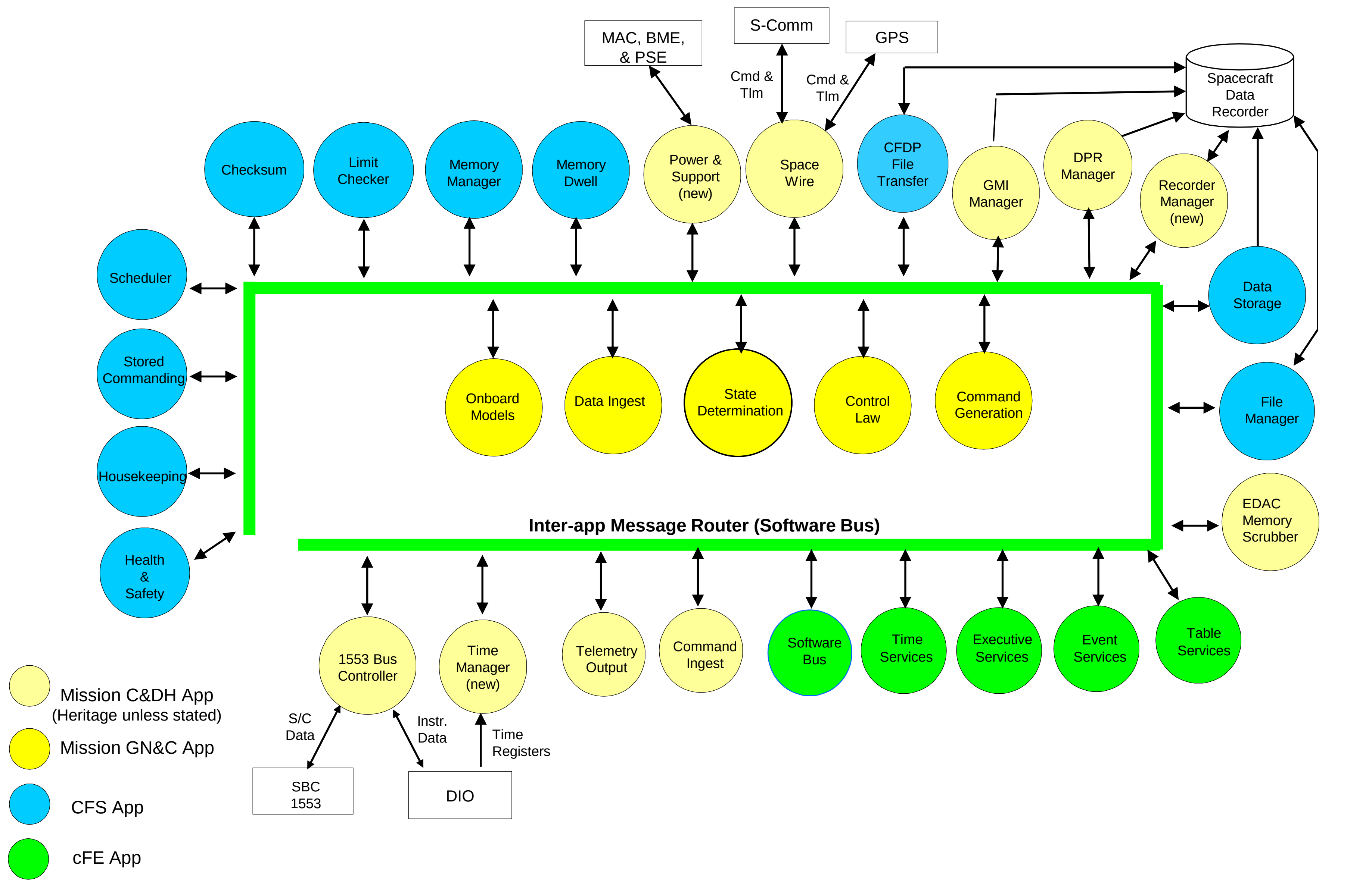
Applications

- **Write once run anywhere the cFS framework has been deployed**
- **Framework has been ported to many popular hardware platform/operating system platforms including MUSTANG and Spacecube**
- **Goddard has released 15 applications 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**

Initial GSFC Open Source Apps

Application	Function
CFDP	Transfers/receives file data to/from the ground
Checksum	Performs data integrity checking of memory, tables and files
Command Ingest Lab	Accepts CCSDS telecommand packets over a UDP/IP port
Data Storage	Records housekeeping, engineering and science data onboard for downlink
File Manager	Interfaces to the ground for managing files
Housekeeping	Collects and re-packages telemetry from other applications.
Health and Safety	Ensures that critical tasks check-in, services watchdog, detects CPU hogging, and calculates CPU utilization
Limit Checker	Provides the capability to monitor values and take action when exceed threshold
Memory Dwell	Allows ground to telemeter the contents of memory locations. Useful for debugging
Memory Manager	Provides the ability to load and dump memory
Software Bus Network	Passes Software Bus messages over various “plug-in” network protocols
Scheduler	Schedules onboard activities via (e.g. HK requests)
Scheduler Lab	Simple activity scheduler with a one second resolution
Stored Command	Onboard Commands Sequencer (absolute and relative)
Telemetry Output Lab	Sends CCSDS telemetry packets over a UDP/IP port

Mission Application Example



Cubesat/SmallSat Distribution (1)

The Distribution Starts with the **Platform Layer** (bottom), which is specific to a processor card and operating system. The Platform layer is everything needed to allow the cFS to run on a Cubesat/Smallsat

Mission Specific Layer

Cubesat/Smallsat Specific Apps and Components

Reusable cFS Layer

Core Flight Executive and Reusable cFS Applications

Platform Layer

Operating System Abstraction Layer

Platform Support Package

Real Time Operating System (Linux, VxWorks, etc)

Real Time OS Board Support Package

Boot Software – Usually Vendor Provided (u-boot, etc)

Cubesat/SmallSat Hardware

Cubesat/SmallSat Distribution (2)

Next, the **Reusable cFS Layer** provides the core functionality that is common to every mission

Mission Specific Layer

Cubesat/Smallsat Specific Apps and Components

Reusable cFS Layer

Reusable cFS Apps

- App Scheduling, Cmd Ingest, and Tlm Output
- Data and File Management and Transfer
- Failure Detection, Isolation and Recovery
- Spacecraft Health, Safety and Maintenance

Core Flight Executive

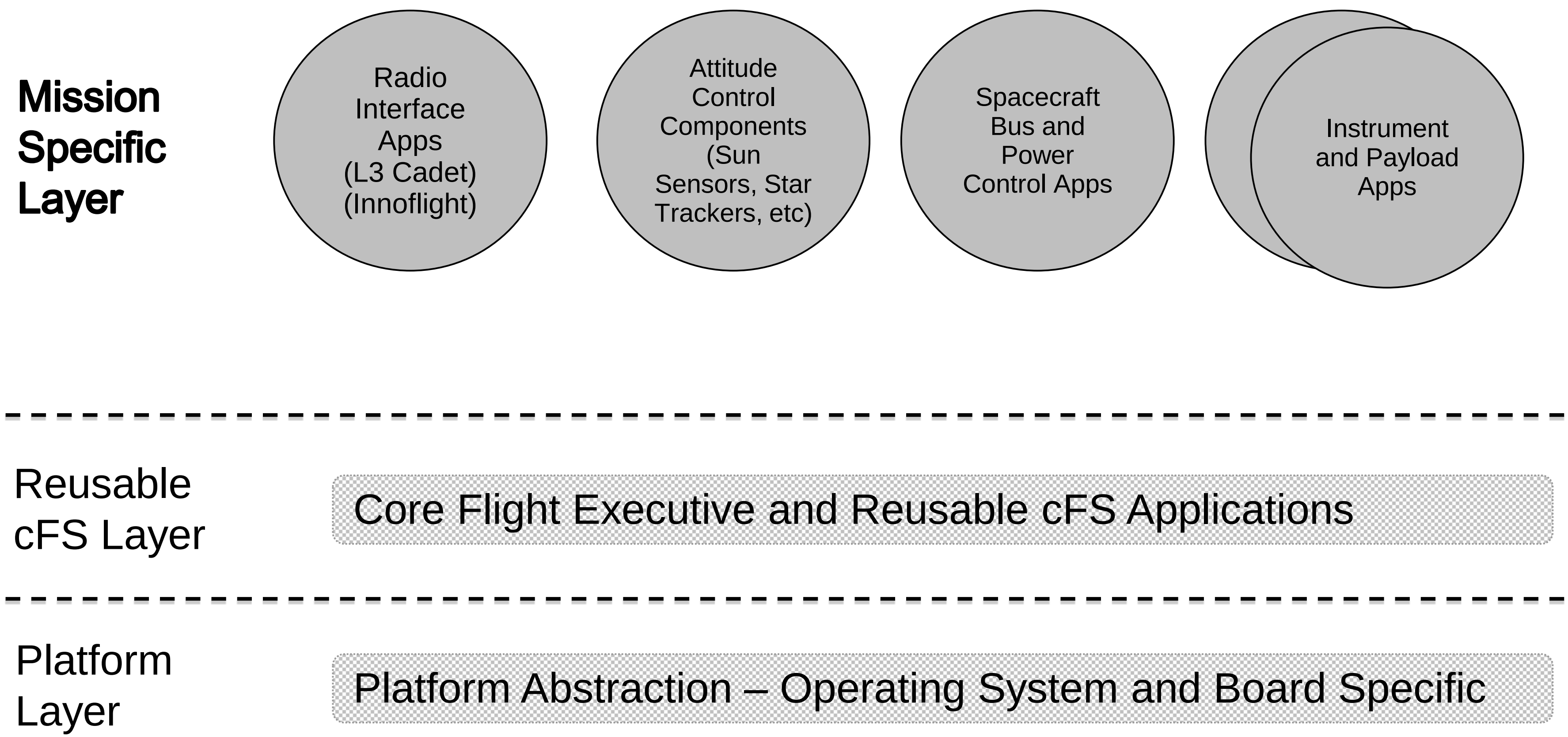
- System Startup Exception, and Restart Services
- Table Services For Loadable Flight Parameters
- Publish and Subscribe Software Bus
- Time Services and Event Services

Platform Layer

Platform Abstraction – Operating System and Board Specific

Cubesat/SmallSat Distribution (3)

Finally, the **Mission Specific Layer** adds support for commonly used components, and mission specific functionality



cFS Metrics

cFE/ App	Logical Lines of Code (non-table)	Config. Parameters	EEPROM (bytes)
cFE 6.4.0	12,930	General: 17 Executive Service: 46 Event Service: 5 Software Bus: 29 Table Service: 10 Time Service: 32	341,561
CFDP	8,559	33	85,812
Checksum	2,873	15	35,242
Data Storage	2,429	27	40,523
File Manager	1,853	22	16,272
Health & Safety	1,531	45	15071
House-Keeping	575	8	8.059
Limit Checker	2,074	13	31,026
Memory Dwell	1,035	8	8,617
Memory Manager	1,958	25	15,840
Scheduler	1,164	19	35,809
Stored Command (124 command sequences)	2,314	26	104,960

Example Mission Code Metrics
Goddard Class B Mission

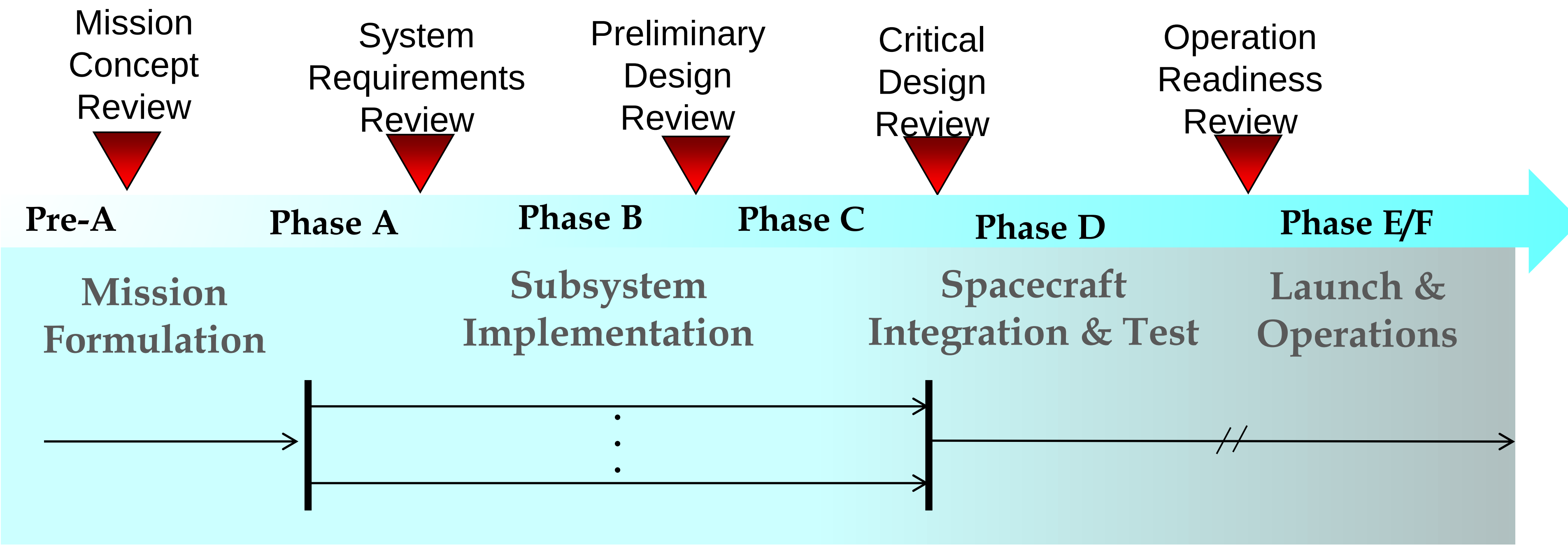
- Noteworthy items
  - + cFE was very reliable and stable
  - + Easy rapid prototyping with heritage code that was cFE compliant
  - + Layered architecture has allowed COTS lab to be maintained through all builds
  - - Addition of PSP changed build infrastructure midstream
- Lines of Code Percentages:

Source	Percentage
BAE	0.3
EEFS	1.7
OSAL	2.1
PSP	1.0
cFE	12.4
GNC Library	1.6
CFS Applications	23.5
Heritage Clone & Own	38.9
New Source	18.5

National Aeronautics and Space Administration

Flight Software Engineering with the cFS

FSW Perspective of Mission Lifecycele



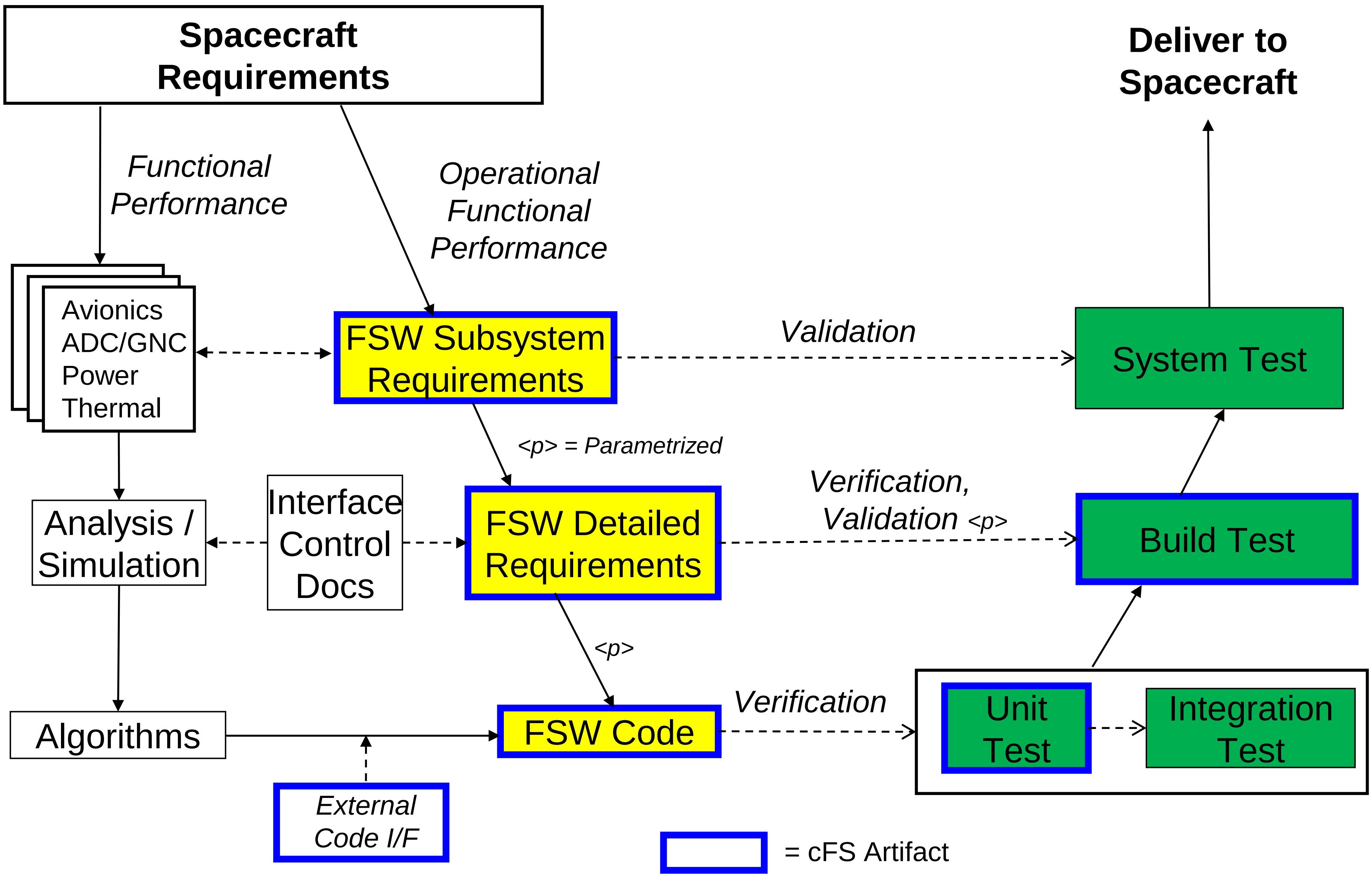
- Estimate Costs
- Define Ops Concept
- Define System Requirements
- Perform Trades
- Identify Risks

- Establish Development & Test Facility
- Refine FSW Requirements
- Develop & Test FSW in builds

- Support I&T test development, runs, and results verification
- FSW consultation and tool support

- Support initial checkout
- Support Ops
  - Troubleshoot
  - Update FSW

FSW Engineering "V"



Flight Software Engineering

- **Describe spacecraft mission lifecycle from a FSW perspective**

- **Identify NASA standards and procedures**

- **So I was thinking we could frame the compliance as:**
-
- **The open source cFS Framework provides the framework to support certification for use by a project. The verification, software classification and quality assurance activities for a specific project use are all the responsibility of the project.**
-
- **Then I could mark up the verification matrix/traceability with all the artifacts we supply, what is a project responsibility, etc. So I wouldn’t say “we don’t comply with” or require any tailoring, but mark the things we don’t do or don’t quite meet as “Flight project responsibility”?**
-
- **With 7150.2C on the verge of coming out it seems like a good time to refresh…**
-

I’ll do a gap analysis and try to prioritize my TODO list.

- **Provide guidelines for different mission classes**

Requirements

- Describe parameters, requirements, and test approach
- Combine next 2 slides and make a good cFS systems top-down story and what cFS does and doesn’t provide
- Add IVV chart and relationship to V

Concept of Operations

- **Key mission events and their timelines**
- **Tracking and navigation**
- **Detailed operational performance requirements**
- **Mission data management**
  - What needs to be stored

A Complete Engineering Solution

Includes reusable:

- Requirements
- Source Code
- Design Documentation
- Development Standards
- Test Artifacts
- Tools
  - Unit Test Framework
  - Software Timing Analyzer
- User's Guides
  - Application Developers Guide
  - API Reference Guides
  - Deployment Guides
  - Flight Operations Guides
- Simple Ground system

The cFS architecture reduces Non-Recurring Engineering (NRE) up to 90%

NASA Standards

- **NASA Procedural Requirements (NPRs)**
  - A set of documents that define procedural and process requirements
  - Compliance is mandatory
  - Defined in NASA Online Directive Information System

Ref https://nodis3.gsfc.nasa.gov/main_lib.html

- **Lifecycle phases defined in**
  - NPR 7120.5E NASA Space Flight Program and Project Management
  - Also referenced in the NASA Systems Engineering Handbook

Ref https://www.nasa.gov/feature/release-of-revision-to-the-nasa-systems-engineering-handbook-sp-2016-6105-rev-2

FSW Process Requirements (1 of 2)

- **NPR 7150.2C – NASA Software Engineering Requirements**
  - Defines software "classes"
    - A – Human Rated Space Software Systems
    - B – Non-human Space Rated Software Systems or Large Scale Aeronautics Vehicles
    - C – Mission Support Software or Aeronautic Vehicles, or Major Engineering/research Facility Software
    - D – Basic Science/Engineering Design and Research and Technology Software
    - E – Design Concept, Research, Technology and General Purpose Software
- **Capability Maturity Model CMMI**
  - Process level improvement training and appraisal program
  - NASA Goddard FSW appraised at CMMI Level 2 ("Managed")

FSW Process Requirements (2 of 2)

- **Goddard Open Learning Design (GOLD) Rules**
  - Specify sound engineering principles and practices, which have evolved in the Goddard community
  - Intended to describe foundational principles that “work,” without being overly prescriptive of an implementation “philosophy

<u>Global Precipitation Measurement (GPM) Critical Design Review</u>

For this chart, **Margin** = (Available – Estimate) / Available

Resource	Amount Available	Current Estimate	Current Margin		GOLD Rule Margin Guidance @ CDR
CPU	100%	40.00%	60%		40%
EEPROM(kB)	2048	1311	36%		40%
uP RAM(kB)	24576	8424	66%		40%
PCI Bus	100%	2.00%	98%		60%
1553 S/C Bus (kbps)	300000	110890	63%		20%
1553 Instr. Bus (kbps)	300000	227000	24%		20%

Backup Slides

cFS References

Contacts

- **cFS Program Manager**

  David McComas - NASA GSFC/Code 580 Software Engineering Division

  david.c.mccomas@nasa.gov, 301.286.9038

- **cFS Framework Development Lead**

  Jake Hageman - NASA GSFC/Code 582 Flight Software Branch

  jacob.j.hageman@nasa.gov, 301.286.1803

- **cFS Lead Architect**

  Jonathan Wilmot - NASA GSFC/Code 582 Flight Software Branch

  jonathan.j.wilmot@nasa.gov, 301.286.2623

- **cFS Platform Development Lead**

  Alan Cudmore - NASA GSFC/Code 582 Flight Software Branch

  alan.p.cudmore@nasa.gov, 301.286.5809

- **Goddard cFS Component Development Lead**

  Beth Timmons - NASA GSFC/Code 582 Flight Software Branch

  elizabeth.timmons@nasa.gov, 301.286.0903

Where is the cFS?

- **cFS Framework, http://github.com/nasa/cFS**
  - Source code
  - Requirements and user guides

- **OSAL, http://sourceforge.net/projects/osal/**
  - Source code
  - Requirements and user guides
  - Tools

- **Links to GSFC applications, https://cfs.gsfc.nasa.gov**

cFS Community Projects on Babelfish

- **Babelfish provides two services for each project:**
  - Git repository
  - Trac system
    - Provides issue tracking and Wiki services
- **Babelfish hosts six separate cFS projects/repos:**
  - cfs_cfe
  - cfs_osal
  - cfs_psp
  - cfs_tools
  - cfs_apps
  - cfs_test
- **Anyone with an NDC account can acquire access**
  - Contact Chris Knight (christopher.d.knight@nasa.gov) for an account

References

- **Open Source**
  - OSAL 4.2.0: http://sourceforge.net/projects/osal/
  - cFE 6.5.0: http://sourceforge.net/projects/coreflightexec
  - Goddard: http://opensource.gsfc.nasa.gov
  - NASA: http://code.nasa.gov

- **Goddard's Strategic Partnership Office**
  - https://partnerships.gsfc.nasa.gov/index.html
  - cFS POC: TBD
- **cFS Websites and Publications**
  - https://cfs.gsfc.nasa.gov
  - Publications
    - Software Architecture Review Board (SARB) Review and Assessment of Goddard Space Flight Center's (GSFC's) core Flight Executive/Core Flight System (cFE/cFS), https://nen.nasa.gov/web/software/sarb
    - Verifying Architectural Design Rules of the Flight Software Product Line, Dharmalingam Ganesan, Mikael Lindvall, Chris Ackermann *Fraunhofer CESE,* http://www.fc-md.umd.edu/save
    - LINUX JOURNAL
    - Ask Magazine
    - AETD Monthly Message

Application user's guide, HTML, System doc, Show desired complete set and goals

UNDER CONSTRUCTION

Backup Slides

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 (AKA. 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
8. Create common standards and tools across the center

Quality Analysis - 1

- **Operability**
  - The architecture must enable the flight system to operate in an efficient and understandable way
- **Reliability**
  - The architecture implementation must be known to behave correctly in nominal and expected off-nominal situations
- **Robustness**
  - The architecture implementation must be predictable and safe in the presence of unexpected conditions
- **Performance**
  - The architecture implementation must efficient in runtime resources given the targeted processing environments
- **Testability**
  - The architecture implementation must be easily and comprehensively testable in situ in flight like scenarios
- **Maintainability**
  - The architecture implementation must be maintainable in the operational environment

Quality Analysis - 2

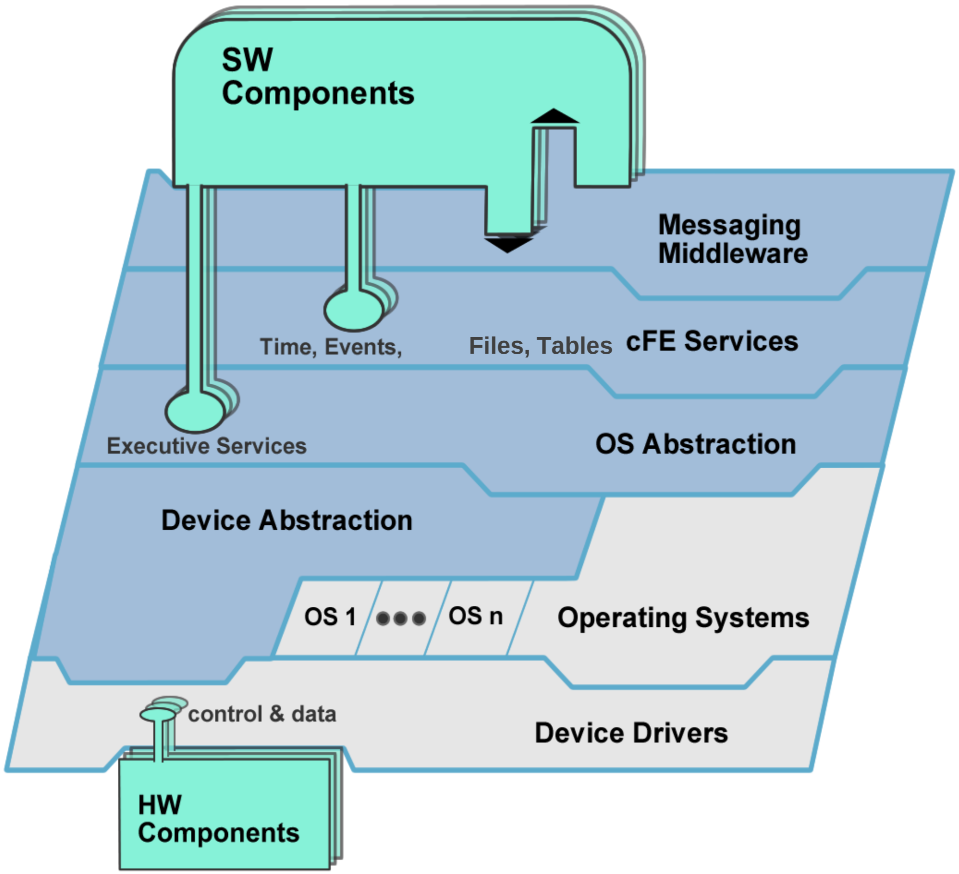
- **Effective Reuse**
  - The architecture must support an effective reuse approach. This includes the software and artifacts. Requirements, design, code, review presentations, test, operations guides, command and telemetry databases. The goal is to achieve 100% reuse of a software component with no code changes
- **Composability**
  - Properties established at the component level, such as interfaces, timeliness or testability, also hold at the system level. For an application or node to be composable the architecture and process must support:
    - Independent development of nodes
    - Integration of the node into a system should not invalidate services in the value and temporal domains
    - Integration of an additional node into a functioning system should not disturb the correct operation of the existing nodes
    - Replica determinism – identical copies of nodes must produce identical results in an identical order, within a specified time interval
- **Predicable Development Schedule**
  - Development estimates provided by the FSW team should be reliable

Quality Analysis - 3

- **Scalability**
  - The FSW must scale with mission requirements. (Example: instruments or subsystem processor may only need a small amount of message buffer space. This should be configurable to avoid wasting memory resources)
- **Adaptability**
  - The FSW must be capable of supporting a range of platforms and missions
- **Minimized Development Cost**
  - Costs for mission functions should be as low as possible. The teams must consider the difference between NRE and costs for a given mission
- **Technology infusion**
  - The FSW should support the infusion of new hardware and software technologies with minimal side effects

Layered Service Architecture

- Each layer and service has a standard API
- Each layer "hides" its implementation and technology details.
- Internals of a layer can be changed -- without affecting other layers' internals and components.
- Provides Middleware, OS and HW platform-independence.

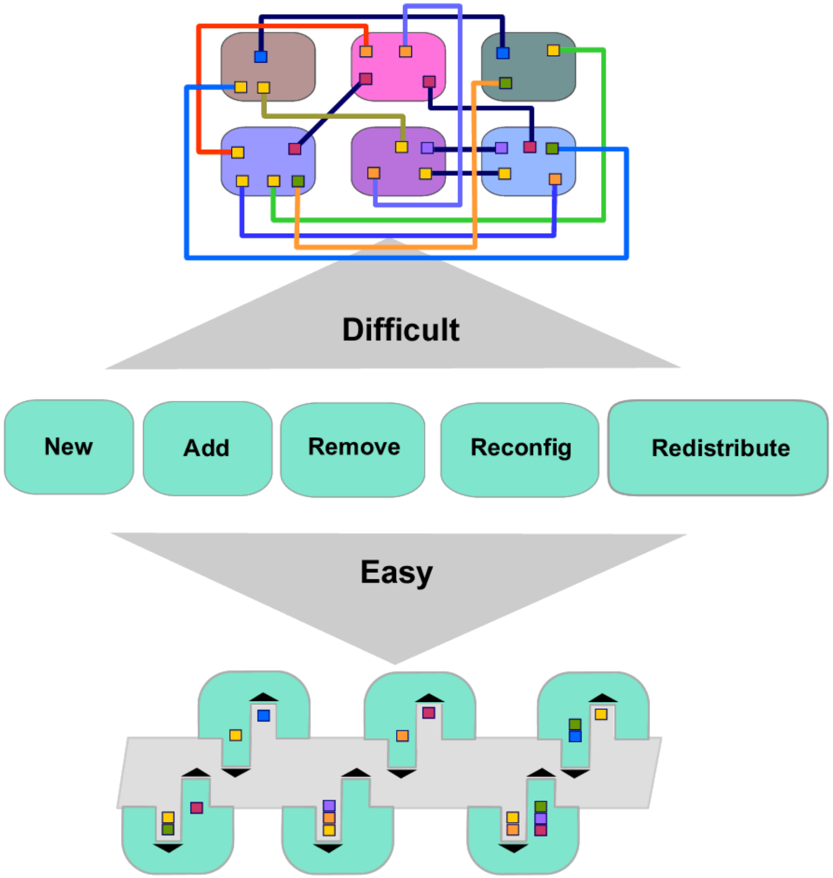


Plug and Play

- 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."

Impact:

- Changes can be made dynamically during development, test and on-orbit even as part of contingency management
- Technology evolution/change can be taken advantage of later in the development cycle.
- Testing flexibility (GSE, test apps, simulators)

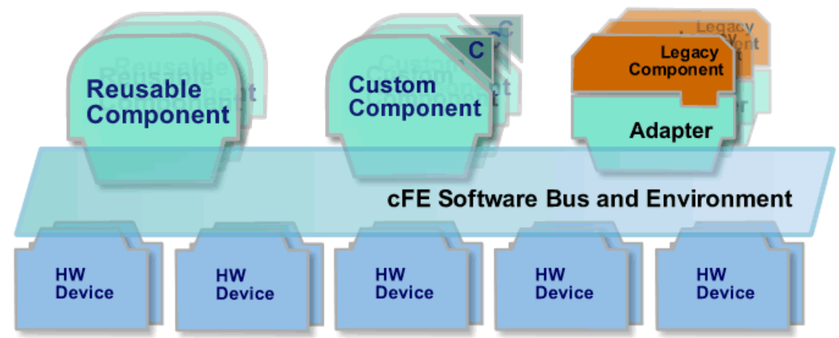


This powerful paradigm allows SW components to be switched in and out at runtime, without rebooting or rebuilding the system SW.

Reusable Components

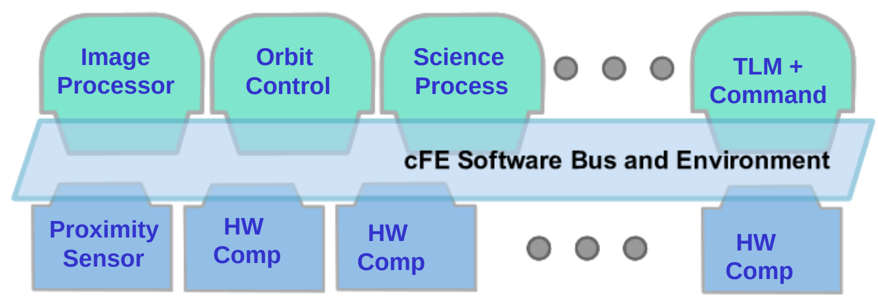
Reusable Components

- 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



Impact:

- 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 and don't "reinvent the wheel."



Component Example

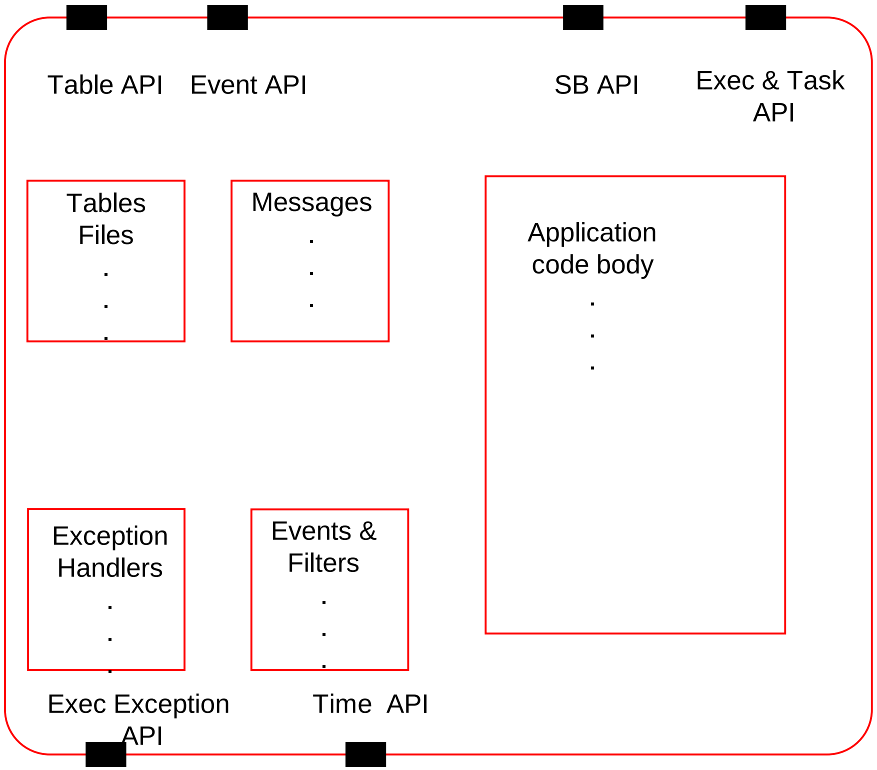
- Interface only through core API's.

- A components contains all data needed to define it's operation.

- Components register for services
  - Register exception handlers
  - Register Event counters and filter
  - Register Tables
  - Publish messages
  - Subscribe to messages

- Component may be added and removed at runtime. (Allows rapid prototyping during development)

- Configuration Parameters



Backup Slides

Acronyms

Acronym List - 1

API	Application Programmer Interface
APL	Applied Physics Lab
ASIST	Advanced Spacecraft Integration and System Testing
ATS	Absolute Time Sequence
BC	Bus Controller
BT	Build Test
bps	bits-per seconds
Bps	Bytes-per seconds
BSP	Board Support Package
C&DH	Command and Data Handling
CCSDS	Consultative Committee for Space Data Systems
CDS	Critical Data Store
CESE	Center for Experimental Software Engineering
CFDP	CCSDS File Delivery Protocol
cFE	Core Flight Executive
cFS	Core Flight Software System
CM	Configuration Management
CMD	Command
COTS	Commercial Off The Shelf
cPCI	Compact PCI
CRC	Cyclic Redundancy Check
CS	Checksum
DMA	Direct Memory Access
DS	Data Storage
EEPROM	Electrically Erasable Programmable Read-Only Memory
EOF	End of File
ES	Executive Services
EVS	Event Services
FDC	Failure Detection and Correction
FDIR	Failure Detection, Isolation, and Recovery
FM	File Management, Fault Management
FSW	Flight Software

Acronym List - 2

GNC	Guidance Navigation and Control
GSFC	Goddard Space Flight Center
GOTS	Government Off The Shelf
GPM	Global Precipitation Measurement
GPS	Global Positioning System
Hi-Fi	High-Fidelity Simulation
HK	Housekeeping
HS	Health & Safety
HW	Hardware
Hz	Hertz
I&T	Integration and Test
ICD	Interface Control Document
IPP	Innovative Partnership Program Office
IRAD	Internal Research and Development
ITAR	International Traffic in Arms Regulations
ISR	Interrupt Service Routine
ITOS	Integration Test and Operations System
IV&V	Independent Verification and Validation
JHU	Johns Hopkins University
KORI	Korean Aerospace Research Institute
LADEE	Lunar Atmosphere and Dust Environment Explorer
LC	Limit Checker
LDS	Local Data Storage
LRO	Lunar Reconnaissance Orbiter
Mbps	Megabits-per seconds
MD	Memory Dwell
MET	Mission Elapsed Timer
MM	Memory Manager
MS	Memory Scrub
NACK	Negative-acknowledgement
NASA	National Aeronautics Space Agency

Acronym List - 3

NESC	NASA Engineering and Safety Center
NOOP	No Operation
OS	Operating System
OSAL	Operating System Abstraction Layer
PCI	Peripheral Component Interconnect
PSP	Platform Support Package
RAM	Random-Access Memory
RM	Recorder Manager
ROM	Read-Only Memory
RT	Remote Terminal
R/T	Real-time
RTOS	Real-Time Operating System
RTS	Relative Time Sequence
SARB	Software Architecture Review Board
S/C	Spacecraft
SB	Software Bus
SBC	Single-Board Computer
SC	Stored Command
SCH	Scheduler
S-COMM	S-Band Communication Card
SDO	Solar Dynamic Observatory
SDR	Spacecraft Data Recorder
SIL	Simulink Interface Layer
SpW	Spacewire
SRAM	Static RAM
SSR	Solid State Recorder
STCF	Spacecraft Time Correlation Factor
SUROM	Start-Up Read-Only Memory
SW	Software, Spacewire
TAI	International Atomic Time
TBD	To be determined

Acronym List - 4

TBL	Table Services
TLM	Telemetry
TDRS	Tracking Data Relay Satellite
TM	Time Manager
TO	Telemetry Output
TRMM	Tropical Rainfall Measuring System
UART	Universal Asynchronous Receiver/Transmitter
UDP	User Datagram Protocol
UMD	University of Maryland
UT	Unit Test
UTC	Coordinated Universal Time
VCDU	Virtual Channel Data Unit
XB	External Bus
XBI	Instrument 1553 External Bus
XBS	Spacecraft 1553 External Bus