FSW Development Process

Aadil Rizvi
NASA Jet Propulsion Laboratory
October 23, 2024

Jet Propulsion Laboratory
California Institute of Technology

Agenda

- **FSW Development Process Overview**
- **FSW Subsystem Level Process**
  - Requirements Phase
  - Proof of Concept and Prototyping
  - Design Phase
  - Implementation Phase
  - Version Control
  - Verification Phase
  - Delivery Review
  - Change Requests and Maintenance
- **Component Level Process**
  - Component Requirements Development
  - Component Design
  - Component Block Diagram and Ports List
  - Component Implementation
  - Component Unit Testing
  - Integrated Testing
  - Component Closeout
  - Checklists
- **Status Reports**
  - Weekly Progress Summary
  - Issue Tracking

FSW Development Process Overview

Why is it important to have a development process?

Enables a higher chance of success by providing backing for adjusting scope and schedule.

- Scope creep
- Delays and issues with receivables
- Unavailability of items required for development and test
- Communicating risks

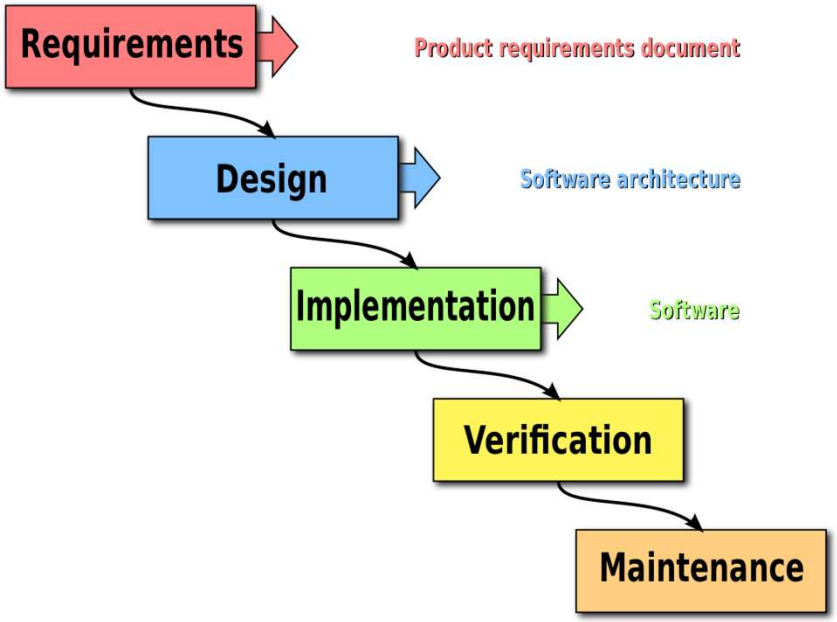
Improves quality: -

- Reliability
- Testability
- Maintainability
- Portability

Waterfall model fits in line with deadline driven development

The right level of process is important

- Too much process can bog you down
- Too little process makes it impossible to manage a complex project
- Either can lead to bad outcomes



Waterfall Model (Peter Kemp / Paul Smith [CC BY 3.0])

FSW Development Process Overview

Development Phases

Requirements	- Provide measurable constraints and characteristics from concept of operations
Design	- Provides blue print for software implementation given a set of requirements
Implementation	- Provides a testable product for verification
Verification	- Ensures implementation functionality and correctness

- Each phase has a review to ensure readiness for the next phase and address any issues

FSW Subsystem Level Process

Requirements Phase

- Why is it important to gather requirements: -
    - Map from concept of operations to specific capabilities that can be designed and implemented
    - Manages assumptions
    - Heads off disagreements/misunderstanding between designers/implementors and their stakeholders
        - "That's not what I wanted you to build!" Or
        - "That's not how I assumed it would work!"
    - Provides the structure for
        - Measuring progress of design and implementation
        - Verifying that we have built/delivered what is needed

FSW Subsystem Level Process

Requirements Derivation

- Understand project level requirements and concept of operations (ConOps) i.e. what is needed for the project
  - Decompose into various software components at high level
  - Functional breakdown rather than design
- Most sub-system requirements are expected to be derived from a parent requirement, but some may be self-derived
- Artifact: Requirements specification document
- Conduct requirements review

Example requirement

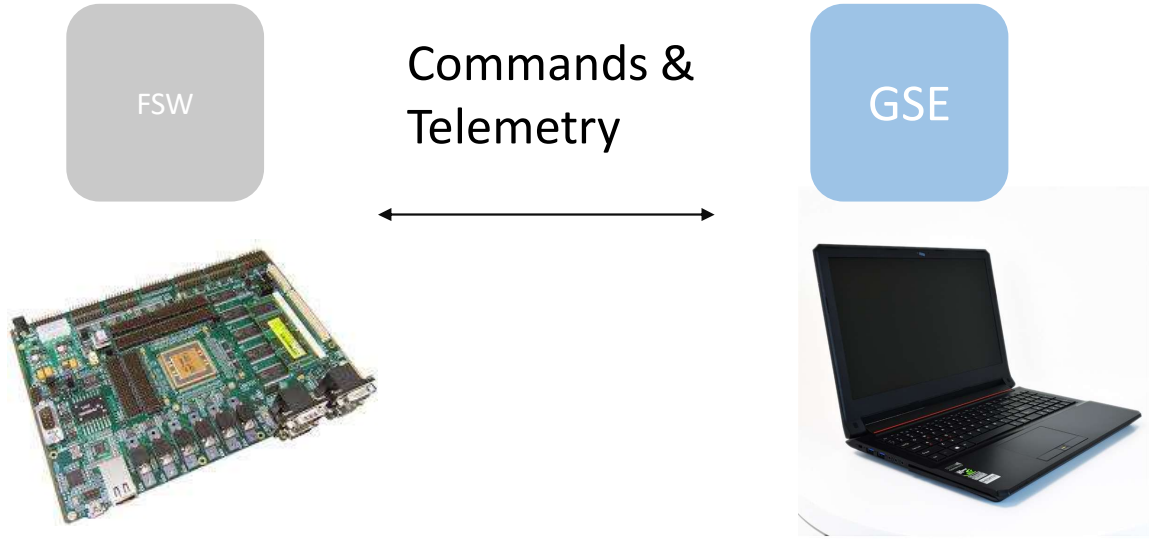
	Category	L4 Parent ID	L4 Parent Body	L5 MCFSW ID	L5 MCFSW Title	L5 MCFSW Requirement Body	Rationale
2	1553 General	L4-LMC-001	The LMC shall communicate with the Compute Element over a dual redundant MIL-STD-1553B bus interface, with a realtime interrupt rate of 64Hz.	L5-MCFSW-001	1553 Interface Commanding	MCFSW shall receive high priority commands, normal priority commands and heartbeat messages via the 1553 Interface on SIPC FPGA from LPCE at 64Hz.	MCFSW to service high priority, normal priority and heartbeat messages received from LPCE over 1553.
3	1553 General	L4-LMC-001	The LMC shall communicate with the Compute Element over a dual redundant MIL-STD-1553B bus interface, with a realtime interrupt rate of 64Hz.	L5-MCFSW-002	1553 Interface Telemetry	MCFSW shall output telemetry packets via the 1553 Interface on SIPC FPGA to LPCE at 64Hz.	MCFSW to output telemetry over 1553 each 64Hz RTI back to LPCE.

What makes a good requirement?

FSW Subsystem Level Process

Proof of concept and Prototyping

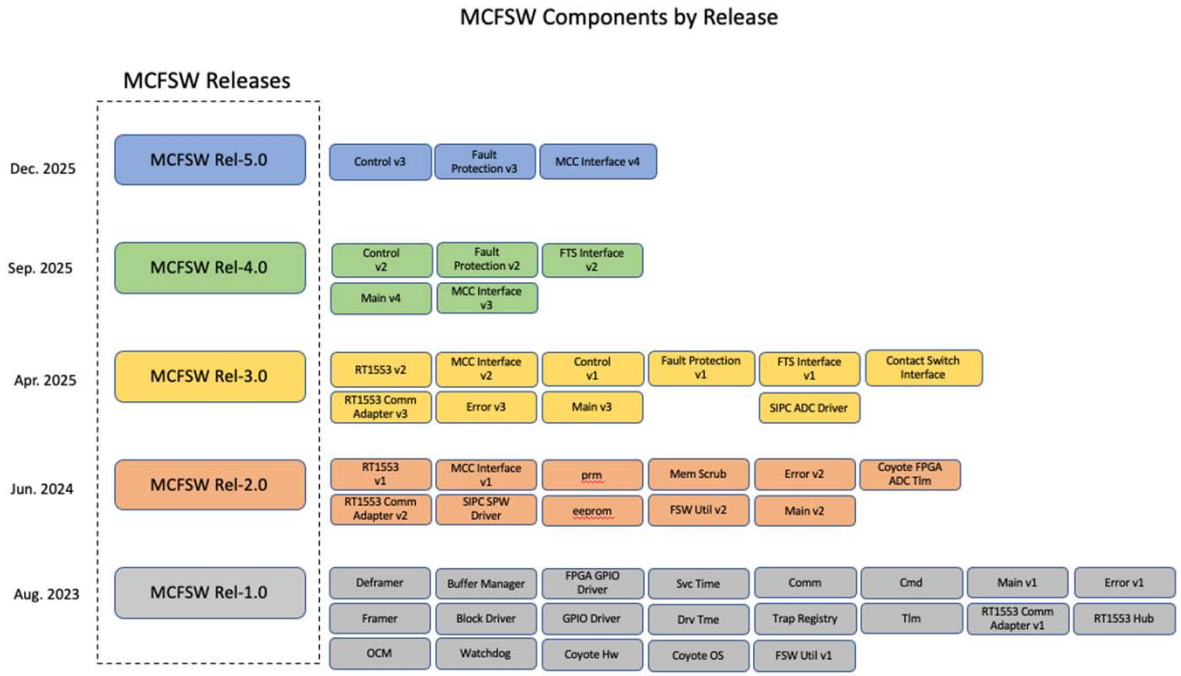
- Target OS and hardware platform
- Compile and execute software on target
- Communicate over planned interfaces
- Data bandwidth and performance analysis



FSW Subsystem Level Process

Design Phase

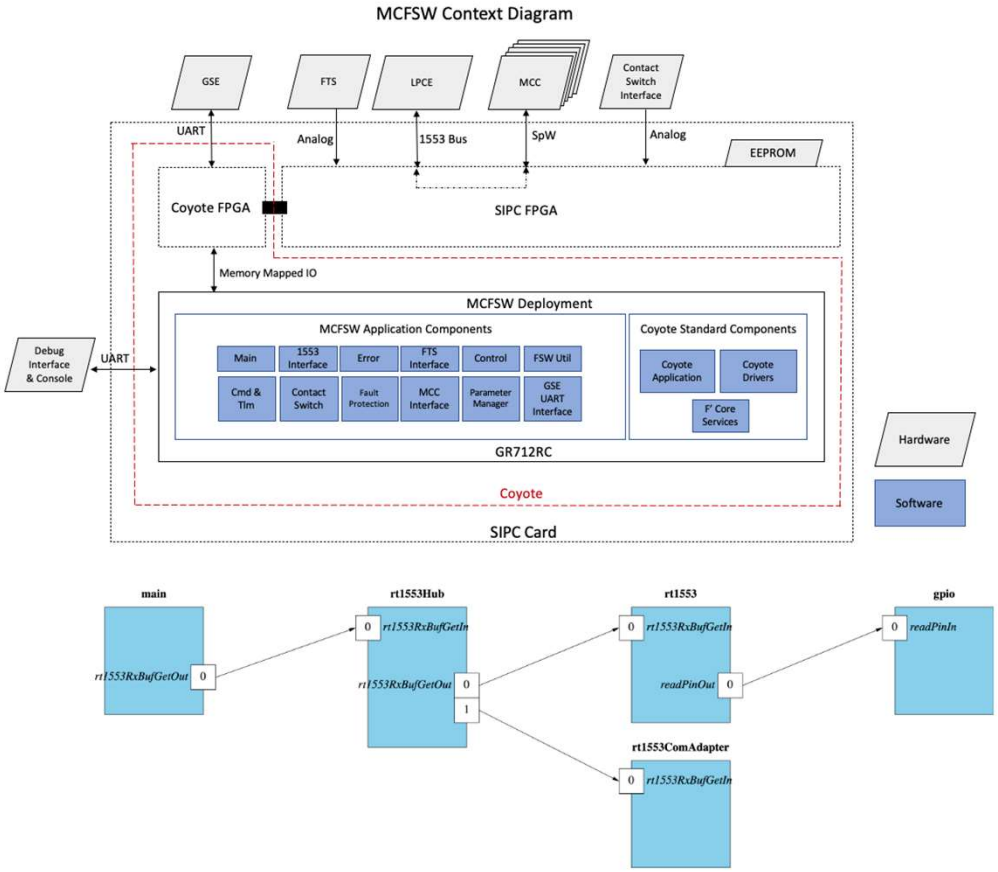
- Trade studies and prototyping
- Develop list of components with functionality description
  - Services
  - Communication
  - Hardware managers
  - Hardware drivers
  - Guidance and control
  - Science
  - Fault protection and mode management
- List of planned releases by components



FSW Subsystem Level Process

Design Phase

- Design Artifacts
  - Context diagrams
  - Interconnect block diagrams (topologies)
  - Sequence diagrams
  - Data flow diagrams
  - Component block diagrams
- Define ports and types to be used across components
- Analyze resource utilization and performance
  - Memory, CPU, I/O
- Address any concurrency issues
- Artifacts:
  - Software architecture and design documentation
  - Trade study results
  - Resource utilization and performance analysis
  - Receivables and deliverables list
  - FSW release delivery schedule
  - Budget and staffing plan
- Conduct design review



FSW Subsystem Level Process

Implementation Phase

- Conduct component level reviews – requirements, design, implementation and unit-test, integrated test results, closeout
- With good design, this should not be too complicated
- May require some design updates, but majority of design expected to be completed in design phase
- Deployment
  - Functional integration of software components
- Development test venues
  - Simulation
  - Prototype/development hardware
  - Testbed
  - EM

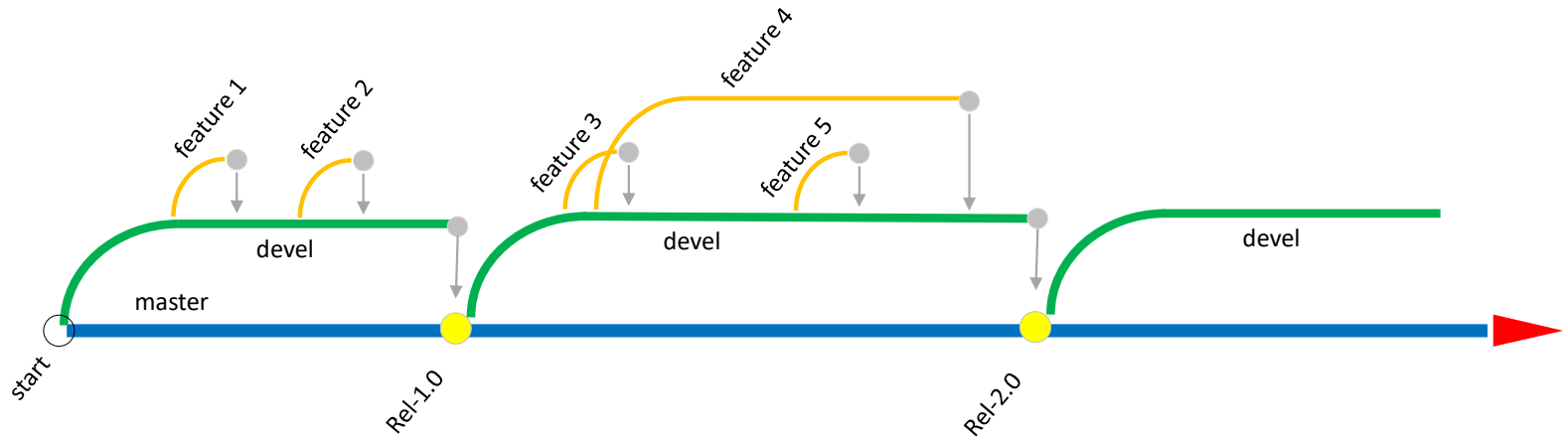
FSW Subsystem Level Process

Implementation Phase

- FSW builds
  - Prototype
  - Internal releases
  - External releases to support sub-system proof of concept
- Artifacts:
  - FSW release package
    - FSW binaries, non-volatile parameter or config files
    - Documentation, build environment, config etc.
    - Dictionaries
  - Test reports
  - Preliminary requirements verification and validation matrix
- Conduct delivery review

FSW Subsystem Level Process

Version Control



= Master Branch
= Devel Branch
= Feature Branch
= Branch Merge
= Release Tag

Git Workflow

FSW Subsystem Level Process

Verification Phase

- Critical to overall software functionality and mission success
- Catching bugs early is cheaper and easier to fix
- Driven by requirements verification
    - Performed using test scripts executed against a release deployment
- FSW builds
    - External releases
- Development test venues
    - Simulation
    - Testbed
    - EM
    - FM

FSW Subsystem Level Process

Verification Phase

- Artifacts
  - FSW release package
    - FSW binaries, non-volatile parameter or config files
    - Documentation, build environment, config etc.
    - Dictionaries
  - Test reports
  - Requirements verification and validation matrix

- Conduct delivery review / SRCR

Requirements verification and validation matrix

REQ ID	Short Title	Level 4 Requirement	Rationale	L3 Parent	V&V Strategy	Status	V&V IDs
NEASC-L4-FSW-3	Data Storage Interface	The FSW shall interface with the non-volatile data storage memory on board the flight CDH unit for read/write access for atleast 4 GB] bytes.	FSW needs access to this memory to manage science and engineering data	NEAS-FS-L3-32	Test	PASS	jpl_ffs-VI-1 thru jpl_ffs-VI-7, prmDb-VI-1

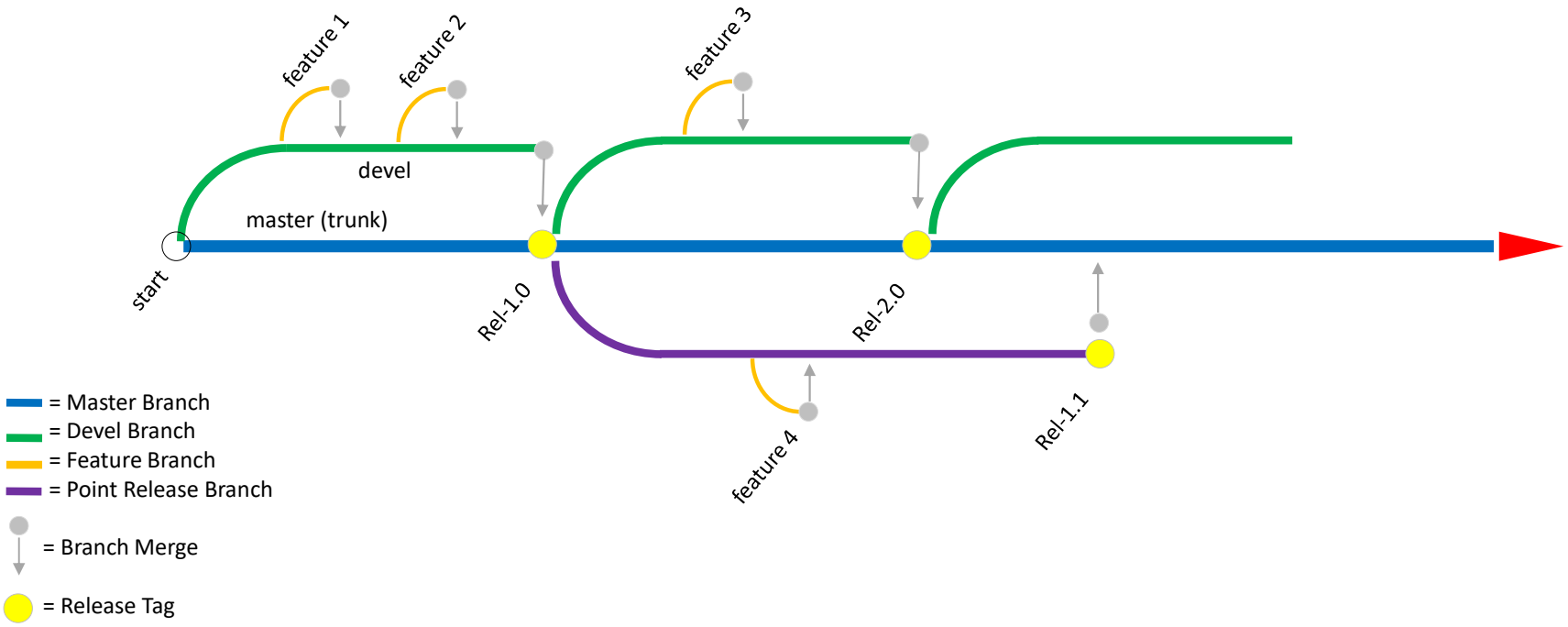
FSW Subsystem Level Process

Delivery Review

- Release description document (RDD)
  - Change log
  - Version Identification
  - Project Overview and Release Description
  - Controlling Documents
  - Test Reports
  - Requirements Verification Summary
  - Idiosyncrasies and Known Issues
  - Problem Disposition
  - Detailed Contents
- Users guide
  - Operational constraints
  - Usage guidelines
- Software design documents

FSW Subsystem Level Process

Change Requests and Maintenance



Component Level Process

Component Requirements Development

- List assumptions relevant to component design
- Develop component requirements table
  - Requirement ID for traceability in unit testing
  - Requirement description and rationale
  - Indicate verification method: unit-test, inspection or analysis
  - Link to parent FSW sub-system level requirement
- Conduct component requirements review

1.1. Assumptions

- This component's cycle input port is invoked in interrupt context
- This component's cycle input port handler is ISR safe
- This component's cycle input port is driven by a hardware triggered ISR at 512Hz per TBD requirement.
- A main loop runs forever and watches for events such as 512Hz cycle count increment and receive of new 1553 high priority command and heartbeat messages as captured in TBD requirement.
- There are 8 512Hz cycles within a 64Hz RTI, numbered 0 through 7, that are sync'd with the 64Hz RTI in hardware to guarantee occurrence of exactly 8 512Hz cycles within a 64Hz time slice as captured in TBD requirement.
- The 512Hz cycle number (0 through 7) is maintained in a SIPC FPGA register and available for software to read each 512Hz cycle within a 64Hz RTI per TBD requirement.
- The 512Hz interrupt is guaranteed to be generated by hardware at the expected rate throughout MCFSW execution per TBD requirement.
- The 1553 64Hz RTI MCFSW heartbeat messages are guaranteed to be received at the expected rate throughout MCFSW execution per TBD requirement.

2. Requirements

Requirement	Description	Rationale	Verification Method	Parent Requirement
MCFSW-SVC-MAIN-001	The `MCFSW::Svc::Main` component shall implement a continuous loop which forever watches for events including updates in the 512Hz cycle counter value and receive of a 1553 high priority command.	Implementation for main event loop driving each 512Hz cycle behavior and servicing high priority commands.	Unit Test	L4-MCFSW-1010
MCFSW-SVC-MAIN-002	The `MCFSW::Svc::Main` component shall implement an ISR handler function that increments the 512Hz cycle count by 1 for each consecutive execution of the ISR.	Track number of 512Hz cycles that occur in a U32 integer that keeps incrementing by 1 for each 512Hz interrupt and rolls back to 0 automatically upon integer overflow.	Unit Test	L4-MCFSW-1010
MCFSW-SVC-MAIN-003	On each iteration of its main loop, the `MCFSW::Svc::Main` component shall, at the start of each loop iteration, poll for a new 1553 high priority command.	Check for 1553 high priority commands in each iteration of the main loop.	Unit Test	L4-MCFSW-1100

Component Level Process

Component Design

- Develop component software design document (SDD)
  - Component overview
  - Assumptions
  - Component level requirements
  - Design
    - Component block diagram
    - Sequence, dataflow, state transition, class diagrams
    - Port List
    - Custom data types
    - State
    - Port Behaviors
  - Commands , telemetry, events and parameters
- Reference datasheets and other technical documents as applicable
- Component and port models (FPP)
- Conduct component design review
- Design checklist walkthrough

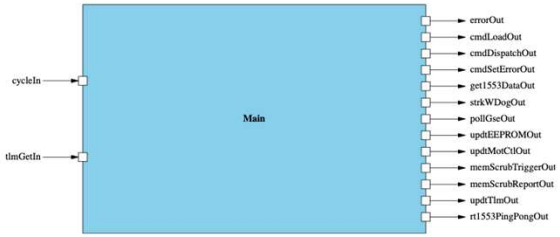
Component Level Process

Component Block Diagram and Ports List

Illustrates component model (FPP)

3.1. Component Diagram

The `MCFSW::Svc::Main` component has the following component block description diagram (BDD) diagram:



3.2. Ports

The `MCFSW::Svc::Main` component uses the following port types:

Kind	Name	Port Type	Usage
`output`	`errorOut`	`Error`	Port to report errors.
`output`	`cmdLoadOut`	`CmdLoad`	Port to load and validate commands.
`output`	`cmdDispatchOut`	`CmdDispatch`	Port to dispatch a command.
`output`	`cmdSetErrorOut`	`CmdSetError`	Port to disable normal priority command dispatching.
`sync input`	`cycleIn`	`Svc.Cycle`	Port to increment cycle counter in ISR context
`sync input`	`tlmGetIn`	`TlmGet`	Port to get periodic telemetry items
`output`	`get1553DataOut`	`MCFSW_Drv.RT1553RxBufGet`	Port to get new 1553 data if available
`output`	`strkWDogOut`	`Svc.Sched`	Port to stroke GR712 and Coyote watchdog timers
`output`	`pollGseOut`	`Svc.Sched`	Port to poll for GSE uplink commands
`output`	`updtEEPROMOut`	`Svc.Sched`	Port to trigger EEPROM update
`output`	`updtMotCtlOut`	`Svc.Sched`	Port to trigger motor control update
`output`	`memScrubTriggerOut`	`Svc.Sched`	Port to trigger memory scrubbing cycle
`output`	`memScrubReportOut`	`MemScrubReport`	Port to trigger reporting of memory scrubbing results (CMEs)
`output`	`updtTlmOut`	`UpdtTlm`	Port to trigger telemetry update
`output`	`rt1553PingPongOut`	`MCFSW_Drv.RT1553PingPong`	Port to perform 1553 ping-pong

Component Level Process

Component Implementation

- Use auto-coded component implementation template files as starting point
- Review JPL C-Coding Standard (JPL Rules DocID 78115) and code checklist being used by the project for coding guidelines
- Reference other mature F’ components for C++ coding style
- Code development
  - Port handler behaviors
  - State management
  - Command handlers
  - Telemetry & events
  - Parameters
- Component compilation for all targets
- Static analysis (SCRUB tool – GCC, Coverity, Code Sonar)
- Conduct code review
- Code checklist walkthrough

Component Level Process

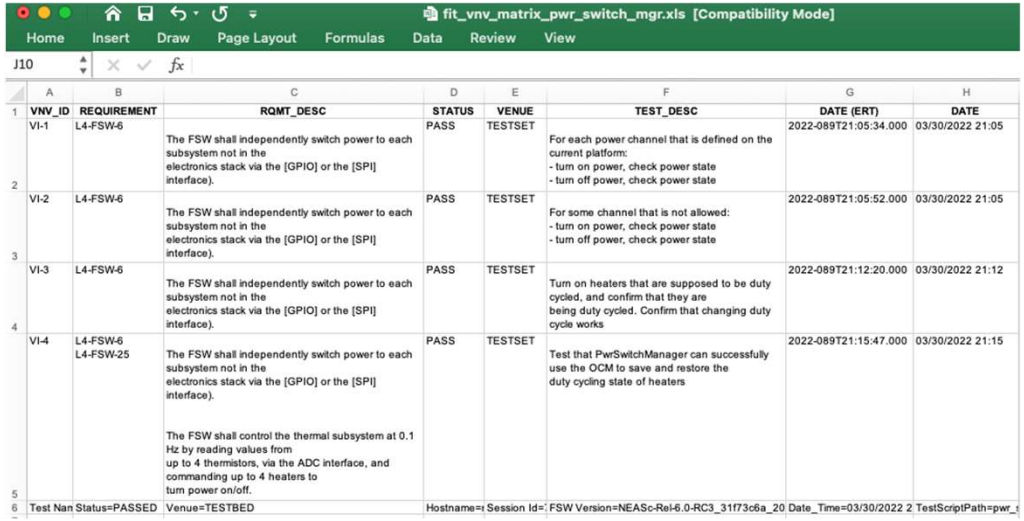
Component Unit Testing

- Component unit tests are developed using F’ unit test harness
  - Provides interfaces for invoking component ports and commands
  - Provides macros for verifying expected behavior, telemetry and events
  - Can be executed as part of an automated regression test-suite
- Traceability of each test-case to component level requirements
- Code coverage analysis
- Unit test output and coverage results
- Conduct unit test review
- Unit test checklist walkthrough

Component Level Process

Integrated Testing

- Test component functionality with the integrated FSW build
- Test venues
  - Simulation
  - Hardware
- Test scripts
  - Send commands and verify telemetry/events
- Test reports
  - Test as-runs
  - Telemetry and event logs
- Requirements V&V
  - FSW sub-system level requirements V&V
  - Traceability for each requirement to all test-cases verifying that requirement
  - Traceability for each test case to all requirements being tested in that test case



VNV_ID	REQUIREMENT	RQMT_DESC	STATUS	VENUE	TEST_DESC	DATE (ERT)	DATE
VI-1	L4-FSW-6	The FSW shall independently switch power to each subsystem not in the electronics stack via the [GPIO] or the [SPI] interface).	PASS	TESTSET	For each power channel that is defined on the current platform: - turn on power, check power state - turn off power, check power state	2022-089T21:05:34.000	03/30/2022 21:05
VI-2	L4-FSW-6	The FSW shall independently switch power to each subsystem not in the electronics stack via the [GPIO] or the [SPI] interface).	PASS	TESTSET	For some channel that is not allowed: - turn on power, check power state - turn off power, check power state	2022-089T21:05:52.000	03/30/2022 21:05
VI-3	L4-FSW-6	The FSW shall independently switch power to each subsystem not in the electronics stack via the [GPIO] or the [SPI] interface).	PASS	TESTSET	Turn on heaters that are supposed to be duty cycled, and confirm that they are being duty cycled. Confirm that changing duty cycle works	2022-089T21:12:20.000	03/30/2022 21:12
VI-4	L4-FSW-6 L4-FSW-25	The FSW shall independently switch power to each subsystem not in the electronics stack via the [GPIO] or the [SPI] interface). The FSW shall control the thermal subsystem at 0.1 Hz by reading values from up to 4 thermistors, via the ADC interface, and commanding up to 4 heaters to turn power on/off.	PASS	TESTSET	Test that PwrSwitchManager can successfully use the OCM to save and restore the duty cycling state of heaters	2022-089T21:15:47.000	03/30/2022 21:15
Test Nan	Status=PASSED	Venue=TESTBED	Hostname=r	Session Id=7	FSW Version=NEASc-Rel-6.0-RC3_31f73c6a_20	Date_Time=03/30/2022 2	TestScriptPath=pwr_s

REQ ID	Short Title	Level 4 Requirement	Rationale	L3 Parent	V&V Strategy	Status	V&V IDs
NEASC-L4-FSW-3	Data Storage Interface	The FSW shall interface with the non-volatile data storage memory on board the flight CDH unit for read/write access for at least 4 GB] bytes.	FSW needs access to this memory to manage science and engineering data	NEAS-FS-L3-32	Test	PASS	jpl_ffs-VI-1 thru jpl_ffs-VI-8, prmDb-VI-1
NEASC-L4-FSW-6	Power Switch Control Interface	The FSW shall independently switch power to each subsystem not in the electronics stack via the [GPIO] or the [SPI] interface)	FSW must provide independent power switch control for each subsystem	NEAS-L3-FS-23	Test	PASS	pwr_eps-VI-7, pwr_eps-VI-8, pwr_switch_mgr-VI-1 thru pwr_switch_mgr-VI-4

Component Level Process

Component Closeout

- Verify all component requirements have been verified
- Verify any additional design and code updates have been reviewed with checklist updates
- Verify all other component checklists have been completed
    - Design
    - Code
    - Unit Test
- Generate component metrics
    - Source lines of code (SLOC)
    - Number of commands, telemetry, events, parameters
- Conduct component closeout review
- Closeout checklist walkthrough

Component Level Process

Checklists

- Can be tailored per the project's risk posture
  - Class D / CubeSats
    - May use a single simple checklist for the entire component development process
  - Type 1 Missions
    - Separate detailed checklist for each component phase including design, code, unit-test and close-out

Component: FSW/Components/FSWImageManager Component Owner: John Component Contributors: Mike, Peter		
	Status	Notes
Modeling		
Model generated in MagicDraw with interfaces defined	YES	
Component auto-coded using component autocoder	YES	
Auto-coded component builds successfully for SPHINX platform	YES	
Implementation		
Behaviors, states, commands, telemetry, and events implemented	YES	
Component builds with the topology	YES	
Deployment		
Static analysis of code performed using SCRUB	YES	
Component Unit Tested	YES	
Executes with topology on SPHINX platform without any issues	YES	
Close-Out		
SDD generated	YES	
Component reviewed by peer(s)	YES	Peers: David
All open issues and action items related to the component have been addressed and closed out	YES	

Design Review - Software Identification

Software Component Name	
Component Developer	
Component Peer Reviewers	
Software Type	
Review Date	

Question Number	Subject	Developer Assessment	Peer Assessment	Comments
Requirements Phase Completion				
1	Are the component requirements complete, clear and unambiguous?	Yes	Yes	
2	Have the component requirements been associated with parent requirements?	Yes	Yes	
3	Have initialization requirements been specified?	Yes	Yes	
4	Have exception handling requirements been specified?	Yes	Yes	
5	Have performance requirements been specified?	Yes	Yes	

Component Unit-Test Peer Review

Software Component Name	
Component Developer	
Component Peer Reviewers	
Software Type	
Review Date	

Question Number	Subject	Developer Assessment	Peer Assessment	Co
8	Verify all requirements have been tested in unit test	Yes	Yes	
Unit-Test Regressionability				
9	Has the peer re-run the unit-test suite?	Yes	Yes	
10	Are all output results repeatable?	Yes	Yes	
11	Are the output results understandable?	Yes	Yes	
12	Is the unit test an automated regression style test?	Yes	Yes	

Component Code Peer Review

Software Component Name	
Component Developer	
Component Peer Reviewers	
Software Type	
Review Date	

Question Number	Subject	Developer Assessment	Peer Assessment	C
Prior Phase Completion				
1	Have all action items from the design review been closed?	Yes	Yes	
2	Do all non-closed actions items have NO code impact?	Yes	Yes	
3	Are all open SPRs for this module incorporated into the code?	Yes	Yes	
Package Completeness				
4	Has the entire component been committed to the remote delivery branch of the Git repository.	Yes	Yes	
5	Does the reviewed software have a git hash code of the last commit?	Yes	Yes	
6	Does the unit compile without warnings for all required targets?	Yes	Yes	

Component Closeout

Software Component Name	
Component Developer	
Component Peer Reviewers	
Software Type	
Review Date	

Question Number	Subject	Developer Assessment	Peer Assessment	C
Coding Closeout				
1	Verify no TBDs, TODOs or printfs in the code (no incomplete code).	Yes	Yes	
2	Verify all requirements allocated to this module in the current development plan are implemented.	Yes	Yes	
3	Verify that all requirements allocated to this module in the current development plan are unit tested and/or have verification analysis or waiver statements in the SDD.	Yes	Yes	
4	Verify component complies for all applicable targets with the latest implementation checklist.	Yes	Yes	

Status Reports

Weekly Progress Summary

- Highlight accomplishments and progress
- Indicate delays in receivables
- Describe pending items
- Report estimated upcoming release delivery date
- Describe current progress against development plan schedule
- Communicate problems to stakeholders early on to facilitate timely action

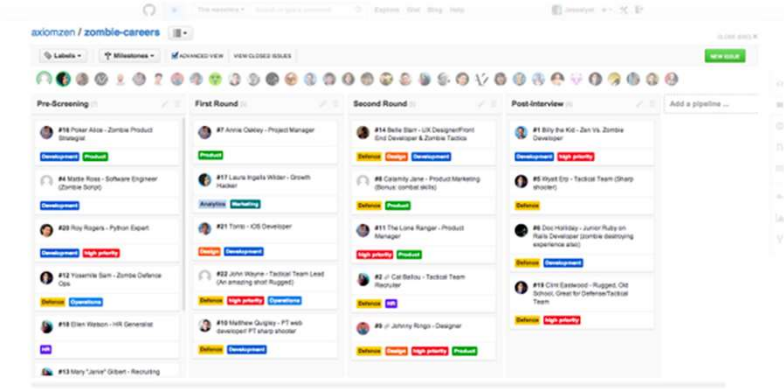
Status Reports

Issue Tracking

- Track current progress using percent complete metric for each task
  - Compute sum of all earned completion points
- Estimate delivery dates and forecast delivery slips early on

Ticket Name	Percent Complete (%)	Milestone	Estimate	Actual Completion Date	FY19 Replan V3 Date	FY18 Replan V2 Date	Original Plan
FSW Rel-1.0 Build, Execution and Integrated Testing	100	FSW Rel-1.0 Delivery	2	8/31/17			8/15/17
SPI Interface Testing	100	FSW Rel-2.0 Delivery	1	1/16/18			1/2/18
Power Switch Testing	100	FSW Rel-2.0 Delivery	1	12/4/17			12/4/17
Propulsion Interface	100	FSW Rel-2.0 Delivery	4	1/29/18			1/15/18
Power EPS testing with FSW	100	FSW Rel-2.0 Delivery	3	1/29/18			1/15/18
Temp sensor testing with ADCs	100	FSW Rel-2.0 Delivery	1	11/9/17			10/13/17
Update CmdSequencer with AMPCS SCMF format	100	FSW Rel-2.0 Delivery	3	1/16/18			1/7/18
Update GenMonitor and EuConverter Database Files (Rel 2.0)	100	FSW Rel-2.0 Delivery	1	1/29/18			1/19/18
Update bad block map for Sphinx (EM)	100	FSW Rel-2.0 Delivery	1	10/31/17			10/31/17
Update and test file system with full 8GB NAND flash	100	FSW Rel-2.0 Delivery	1	11/3/17			11/3/17
Test XACT+Prop Component with EDU	100	FSW Rel-2.0 Delivery	3	1/8/18			1/8/18
FSW Rel-2.0 Build, Execution and Integrated Testing	100	FSW Rel-2.0 Delivery	4	2/1/18			2/1/18
FP Manager	100	FSW Rel-3.0 Delivery	2	3/6/18		2/23/18	
Mode Manager	100	FSW Rel-3.0 Delivery	2	3/13/18		3/9/18	
FP State Manager	100	FSW Rel-3.0 Delivery	2	4/12/18		4/4/18	
FSW Rel-3.0 Build, Execution and Integrated Testing	100	FSW Rel-3.0 Delivery	4	4/17/18		4/17/18	
Instrument Electronics Interface	100	FSW Rel-4.0 Delivery	4	8/7/18		5/23/18	
FP State Manager Updates	100	FSW Rel-4.0 Delivery	2	8/7/18		6/13/18	
Update GenMonitor and EuConverter Database Files for flight	100	FSW Rel-4.0 Delivery	1	6/29/18		6/13/18	
Re-map power switches and ADC Channels	100	FSW Rel-4.0 Delivery	1	7/3/18		6/29/18	
FSW Rel-4.0 Build, Execution and Integrated Testing	100	FSW Rel-4.0 Delivery	3	8/10/18		7/6/18	
Payload Interface Updates	100	FSW Rel-4.1 Delivery	1			9/12/18	
Bootloader for VxWorks+FSW	100	FSW Rel-4.1 Delivery	1			9/19/18	
FSW Rel-4.1 Build, Execution and Integrated Testing (Payload interface only)	100	FSW Rel-4.1 Delivery	2	11/15/18		9/26/18	
XACT interface updates for fault protection	100	FSW Rel-4.2 Delivery	4	4/8/19	4/8/19	1/11/19	
PWM Duty Cycling	100	FSW Rel-4.2 Delivery	2	1/15/19		1/17/18	
Iris Temperature Conversions	100	FSW Rel-4.2 Delivery	1	4/10/19	4/11/19	1/17/18	
Iris V2.1 RevC updates in FSW	100	FSW Rel-4.2 Delivery	2	3/12/19	4/11/19		
Update Image Burn Process	50	FSW Rel-4.2 Delivery	3	4/18/19	4/18/19		
FSW Rel-4.2 Build, Execution and Integrated Testing (XACT interfaces only)	30	FSW Rel-4.2 Delivery	2		5/23/19	1/25/19	
Prop interface updates for fault protection	10	FSW Rel-4.3 Delivery	4		6/28/19		
FSW Rel-4.3 Build, Execution and Integrated Testing (Prop interfaces only)	0	FSW Rel-4.3 Delivery	2		7/26/19		
Bug fixes, updates and change requests post FSW Rel-4.0	0	FSW Rel-5.0 Delivery	1		9/12/19	2/8/19	
FSW Rel-5.0 Build, Execution and Integrated Testing	0	FSW Rel-5.0 Delivery	1		10/4/19	2/22/19	

Completion Points



ZenHub Pipelines

Status Reports

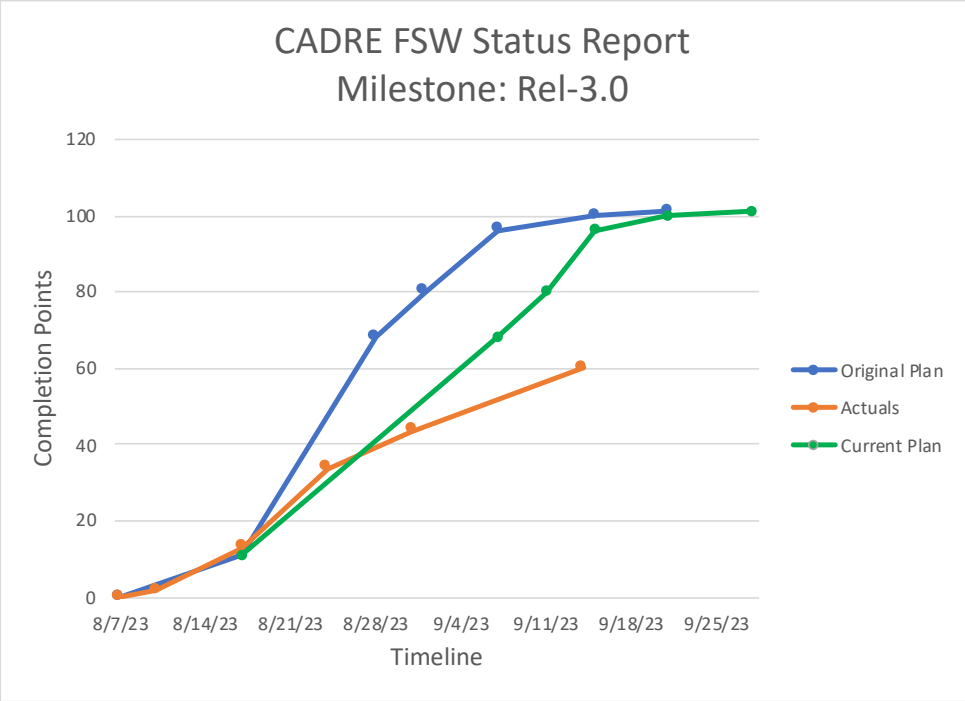
Issue Tracking

Total Planned Completion Points for Rel-1.0 = 17

Status Reports

Earned Value Management

- Measuring planned work against actual work completed
- Schedule variance
- Cost variance



Questions