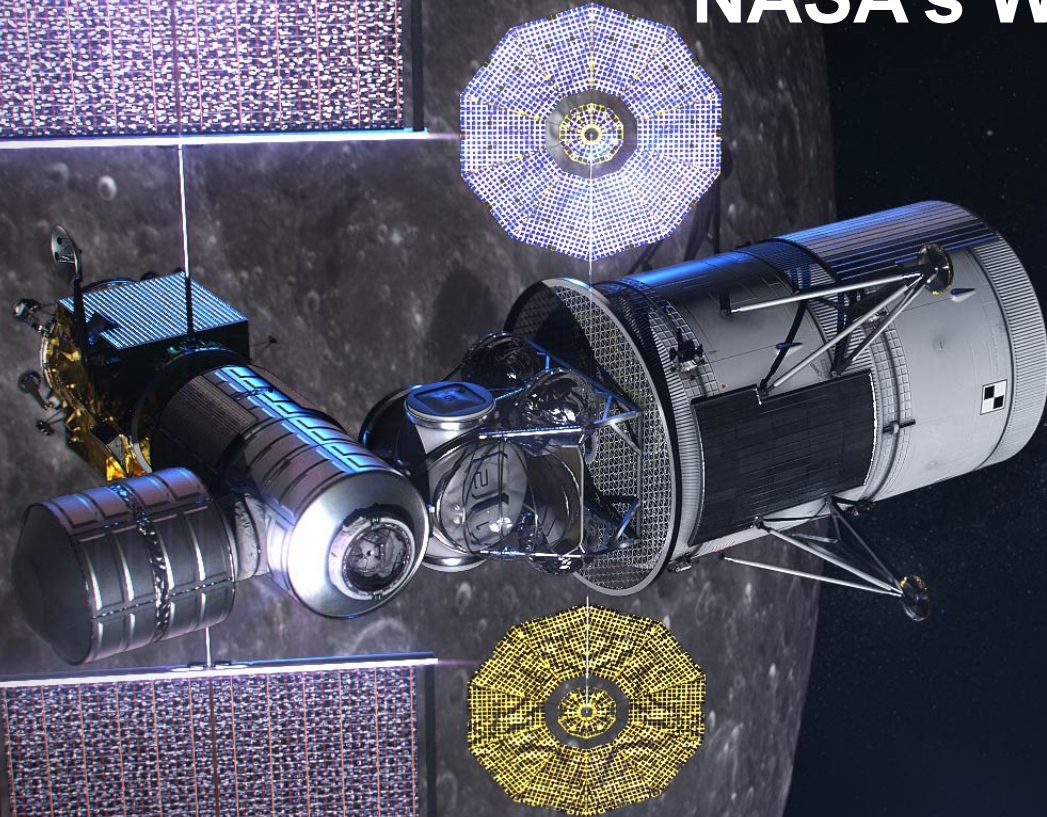


Agile Software Engineering in NASA's Waterfall World



David Swartwout
GW Production Software Lead
20 February 2024





Overview



- **Agenda:**
 - NASA Procedural Requirements
 - Lifecycles
 - Process Rigor
 - JSC Engineering's Agile Process Evolution
 - Toolchain
 - Following the NPR Lifecycle
 - SRR to PDR
 - PDR to CDR
 - CDR to TRR
 - TRR to SAR
 - Closing Thoughts



NASA Procedural Requirements



NASA projects are governed by NASA Procedural Requirements (NPR)

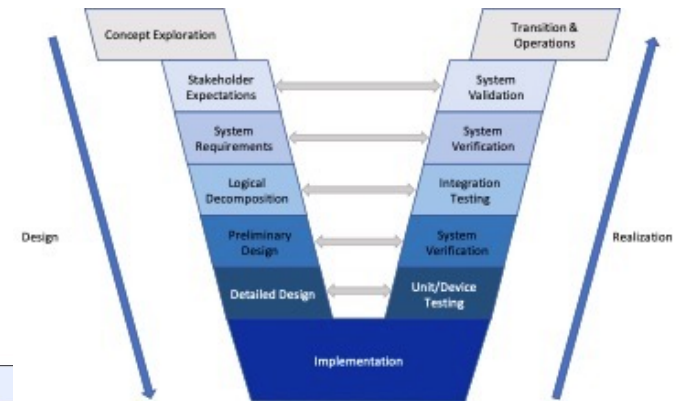
- NPR 7120.5, NASA Space Flight Program and Project Management Requirements
- NPR 7120.7, NASA Information Technology Program and Project Management Requirements
- NPR 7120.8, NASA Research and Technology Program and Project Management Requirements

And the overarching Systems Engineering NPR

- NPR 7123.1 NASA Systems Engineering Processes and Requirements

All of these NPRs describe a reasonably rigid Waterfall lifecycle

- The Systems Engineering “V”
- Milestone Gate Reviews
 - SRR, PDR, CDR, etc.
 - Tailorable Entrance/Exit Criteria
 - Products required at each review



NASA Life-Cycle Phases	Approval for Formulation		Approval for Implementation				
Project Life-Cycle Phases	Pre-Phase A: Concept Studies	Phase A: Concept & Technology Development	Phase B: Preliminary Design & Technology Completion	Phase C: Final Design & Fabrication	Phase D: System Assembly, Integration & Test, Launch & Checkout	Phase E: Operations & Sustainment	Phase F: Operations & Sustainment
Project Life-Cycle Gates, Documents, and Major Events	KDP A	KDP B	KDP C	KDP D	KDP E	KDP F	Final Archival of Data
Agency Reviews	FAD, FA, Preliminary Project Requirements	Preliminary Project Plan	Baseline Project Plan		Launch	End of Mission	
Human Space Flight Project Life-Cycle Reviews ^{1,2}	MCR, ASIR ⁶	SRR, SDR	PDR	CDR / PRR ³ , SIR	ORR, FRR, PLAR	CERR ⁴	DR, DRR
Reflights ⁵						End of Flight, FAR	
Robotic Mission Project Life-Cycle Reviews ^{1,2}	MCR	SRR, MDR ⁵	PDR	CDR / PRR ³ , SIR	ORR, MRR, PLAR	CERR ⁴	DR, DRR
Other Reviews				SAR ⁶	SNLSR, LRR (LV), FRR (LV)		
Supporting Reviews	Peer Reviews, Subsystem PDRs, Subsystem CDRs, and System Reviews						

FOOTNOTES

- Flexibility is allowed as to the timing, number, and content of reviews as long as the equivalent information is provided at each KDP and the approach is fully documented in the Project Plan.
- Life-cycle review objectives and expected maturity states for these reviews and the attendant KDPs are contained in Table 2-5.
- PRR is needed only when there are multiple copies of systems. It does not require an SRB. Timing is notional.
- CERRs are established at the discretion of program.
- For robotic missions, the SRR and the MDR may be combined.
- SAR generally applies to human space flight.
- Timing of the ASM is determined by the MDA or AA, compliant with NPD 1000.5, and between MCR and KDP A.
- Placement of arrows is notional. See Section 2.2.4.3 for more guidance on reflights.

ACRONYMS

ASM - Acquisition Strategy Meeting
 CDR - Critical Design Review
 CERR - Critical Events Readiness Review
 DR - Decommissioning Review
 DRR - Disposal Readiness Review
 FA - Formulation Agreement
 FAD - Formulation Authorization Document
 FRR - Flight Readiness Review
 KDP - Key Decision Point
 LRR - Launch Readiness Review
 LV - Launch Vehicle
 MCR - Mission Concept Review
 MDR - Mission Definition Review
 MRR - Mission Readiness Review
 ORR - Operational Readiness Review
 PDR - Preliminary Design Review
 PLAR - Post-Flight Assessment Review
 PRR - Production Readiness Review
 SAR - System Acceptance Review
 SDR - System Definition Review
 SNLSR - System Integration Review
 SRB - Standing Review Board
 SRR - System Requirements Review

▲ Red triangles represent life-cycle reviews that require SRBs. The Decision Authority, Administrator, MDA, or Center Director may request the SRB to conduct other reviews.



NASA's Software Engineering Requirements



Software Engineering process and product requirements, as a component of the overall project execution are then detailed in

- NPR 7150.2, NASA Software Engineering Requirements

Guidelines are described in

- NASA Software Engineering and Assurance Handbook, NASA-HDBK-2203

- NPR 7150.2 describes the “what” that has to get done

- Document requirements
- Document design
- Perform testing
- Etc.

- But luckily, it doesn't describe the “how”

- Does not dictate format or templates

- Though, NPR 7123.1 does tell you “when”

- Milestone review entrance criteria describe what products are needed for each review
 - Software requirements at PDR
 - Software design at CDR
 - Etc.

- Caution - some Centers may levy additional constraints via Center specific Procedural Requirements

The screenshot shows the NASA Software Engineering and Assurance Handbook (NASA-HDBK-2203) website. The header includes the NASA logo and the text "Software Engineering Handbook Content based on NPR7150.2D" and "NASA ENGINEERING NETWORK". The navigation menu includes: A. Introduction, B. Institutional Requirements, C. Project Software Requirements, D. Topics, E. Tools, References, and Terms, and F. SPAN (NASA Only). A search bar is located to the right of the menu. Below the menu, the page is titled "Book A. Introduction". The main content area has a "New in SWEHB" section with a breadcrumb trail: 1. Welcome, 2. SWEHB Introduction, 3. Title Material, 4. Resources, 5. Accessing Other Versions of SWEHB, 6. NASA-STD-8739.8B Title Material. The main heading is "Welcome to the NASA Software Engineering and Assurance Handbook, NASA-HDBK-2203." followed by the NASA logo. The text states: "Software is a core capability and key enabling technology for NASA's missions and supporting infrastructure." Below this, there is a "What's New in SWEHB!" section with three entries:

- Updates to Topic 7.08: Hagh, Fred Douglas. (HQ-KA000) [PEROT] posted on Jan 26, 2024. Topic 7.08 - Maturity of Life Cycle Products at Milestone Reviews has been updated and is now available for use. Click here to see more ...
- New Topic on Software Assurance Risk: Hagh, Fred Douglas. (HQ-KA000) [PEROT] posted on Dec 07, 2023. A new Topic titled "8.24 - Software Assurance Risk" has been added to the D. Topics page. Click here to see more ...
- New Topic on Space Security: Hagh, Fred Douglas. (HQ-KA000) [PEROT] posted on Oct 18, 2023. A new Topic titled "7.22 - Space Security."

 At the bottom, there is a section for "Documents can be viewed and downloaded in PDF format:" listing:

- The NASA Software Engineering Requirements, NPR 7150.2D 063
- The NASA Software Assurance and Software Safety Standard requirements, NASA-STD-8739.8B 278

 A note at the bottom states: "You can submit any inputs and suggestions regarding SWEHB via 'Feedback' in the NASA Technical Standards System (NTSS)."

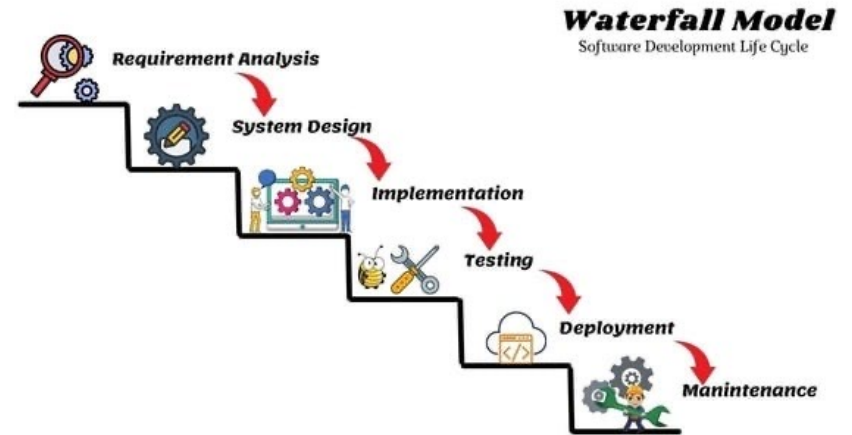


The Waterfall Lifecycle

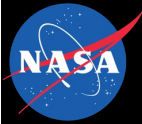


Some challenges and pitfalls of Waterfall

- It assumes you got it right the first time
- Going back to earlier cycles is nearly impossible
- It takes a very long time to learn what you don't know
 - Or what you didn't get right in the previous cycles
- Course corrections are hard
- Feedback is slow



www.programmingster.com

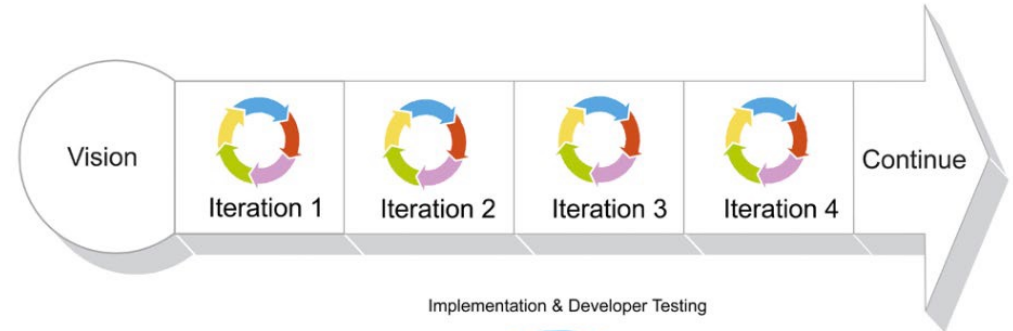


Agile



Industry experience has shown that Agile project management can be highly successful

- Cyclic
 - Looking at all aspects each iteration
- Faster to uncover issues and gaps
- Burn down risk earlier
- Quicker Feedback
 - Drive metrics for faster decisions
- Fosters collaboration
- Incremental growth of knowledge



Iteration Detail

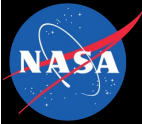


Manifesto for Agile Software Development

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

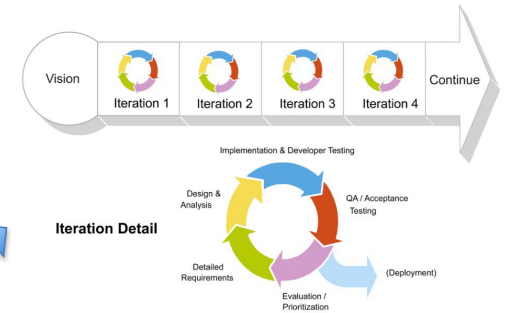
That is, while there is value in the items on the right, we value the items on the left more.



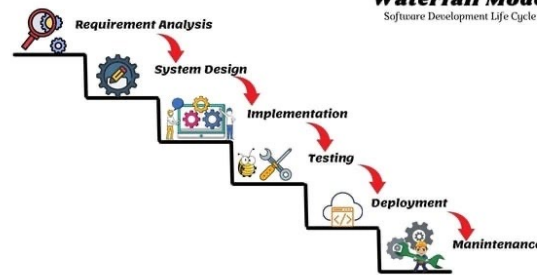
How did we bridge the gap?



So, how did we achieve Agile software engineering within NASA's NPR dictated waterfall lifecycle?



Waterfall Model
Software Development Life Cycle



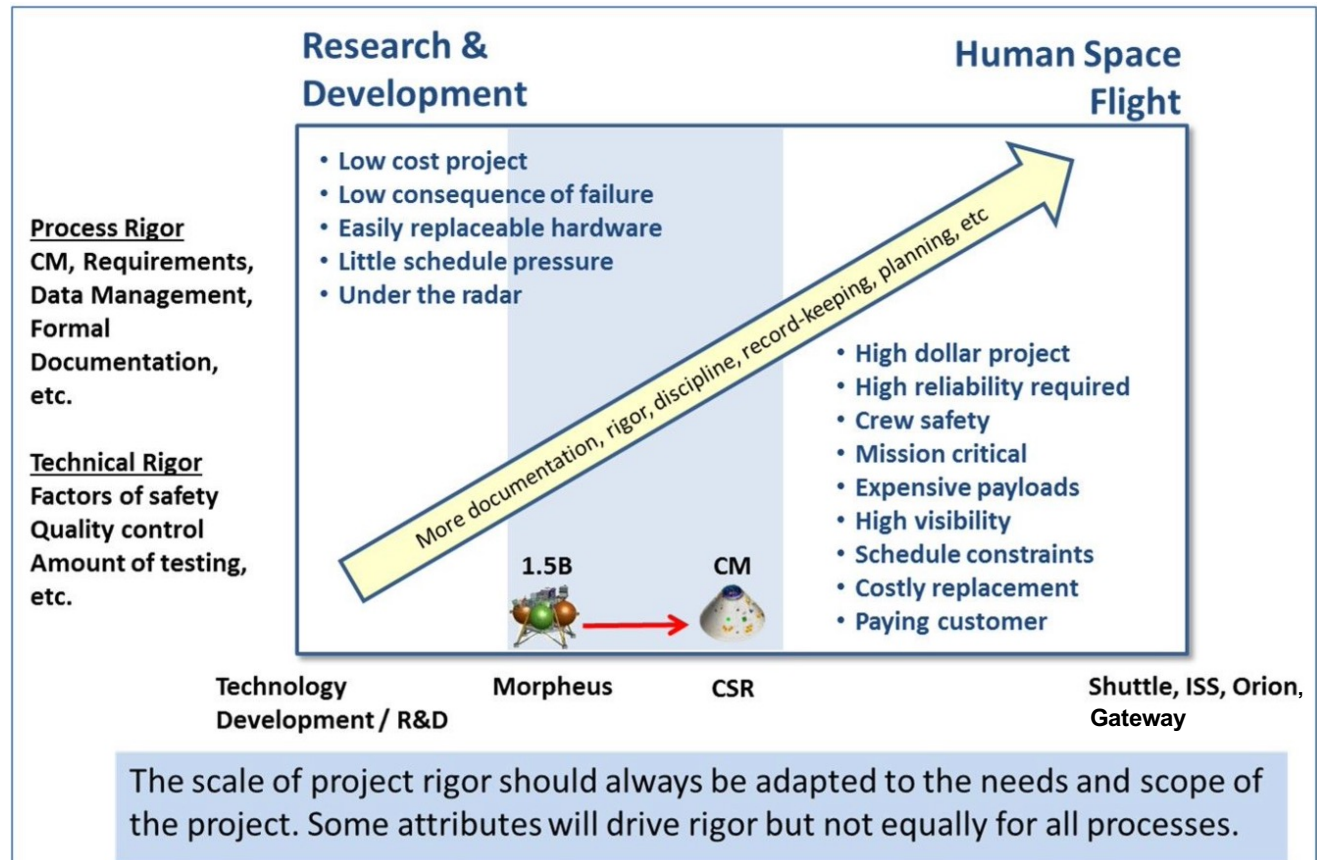
www.programmingster.com



Apply process rigor when it is important



- Process rigor is important
 - But only at the right time
- The content of the artifacts detailed in the NPRs are important
 - Requirements
 - Design documentation
 - Test plans and procedures
- Agile software engineering is not the “wild west” of project management
 - In my experience with Agile, there are more day to day process requirements than other non-Agile projects I have worked
 - Daily feedback loops
 - Metrics





Evolution of JSC Engineering's Agile Software Engineering Process



Morpheus Lander ~ 2010 - 2014

- 1st focused attempt with Agile
- Highly successful R&D project
- Class C software project
- CMMI ML 2

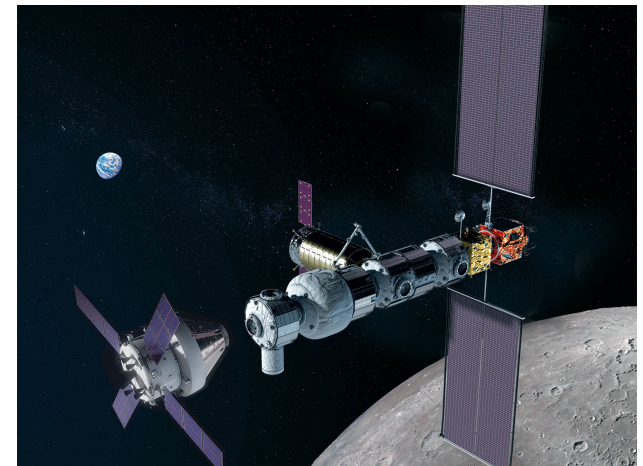


Orion Ascent Abort – 2 ~ 2015 - 2019

- Orion flight test of Launch Abort System
- Class B safety critical software project
- CMMI ML 3

Gateway Lunar Outpost ~ 2019 - today

- Multiple contractors, multiple teams
- Class A safety critical software projects
- CMMI ML 3

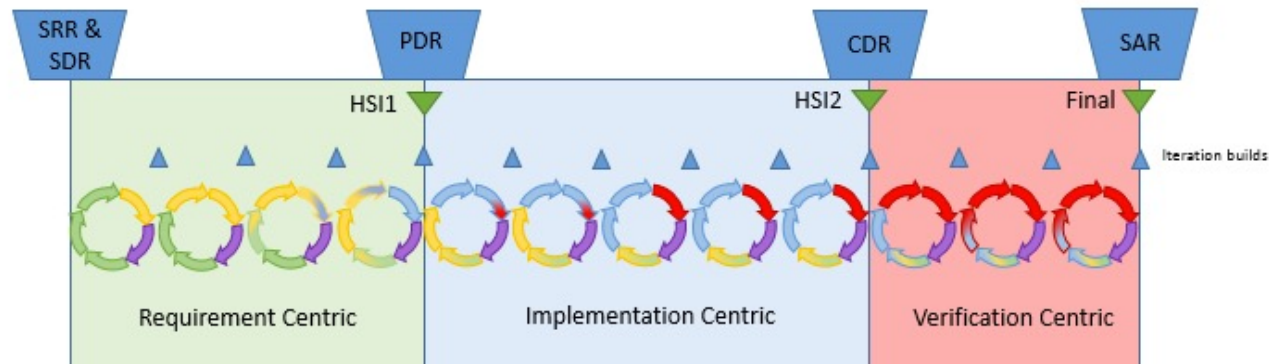




Our lifecycle focused Agile flow



- Changing software team focus as project moves through 7123.1 lifecycle
- Focus on meeting the intent of the milestone without spending wasted time writing documents
- Make NPR required artifact development part of the standard development flow
 - Updating the design documentation is part of closing each code implementation story
 - Evolving definition of done for different story types as we move through development gates





Our Solution Leverages Several Tool Ecosystems



- Jira
 - TestRay plugin
 - Functional Requirements
 - Non-Functional Requirements
 - Test Cases
 - Test Plans
 - Test Automation
 - Stories – Work code
 - Epics – Use cases
 - Tasks – Design work
 - Scripting API back end for trace tables
 - Release management
- Confluence
 - Design documentation
 - Use case management
- Gitlab
 - Source code
 - Continuous integration pipeline
- Doxygen
 - Design documentation
- Jenkins
 - Automated testing
- Home grown scripts to tie it all together

 **ATLASSIAN**

 **ATLASSIAN**
Jira
 **ATLASSIAN**
Confluence

 **GitLab**

 **doxygen** 



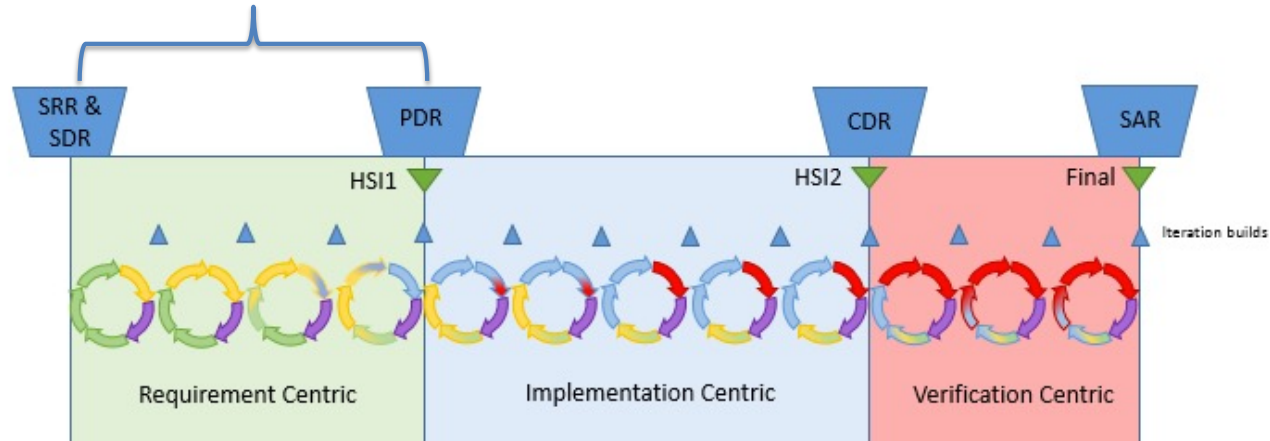
Jenkins



Team Focus: SRR to PDR



- Requirements
- Verification Planning
- Architecture
- Tool Chain
- **Use Case Based Advancement of System Capability**





Development of Software System Requirements (SRS)



- Use con ops and operational use cases to drive requirements
 - Higher level requirements
- Computer Software Configuration Item (CSCI) requirements are defined in Jira
 - The box shall ...
 - Utilizes new Jira ticket types from TestRay plugin
- Stakeholders review and comment right in the tools
 - Enhances collaboration and feedback
- The actual SRS is a confluence page with the embedded Jira tickets
 - Jira/TestRay allows management of requirements baselines
 - Export to a PDF to support a “baseline” for lifecycle reviews
 - Manage change traffic between versions using labels on the Jira requirements

Vehicle Systems Manager Flight Software / VSMFSW-1090

Initializing State

Edit Add comment Assign More In Progress

Details

Type: **Functional Requirement** Resolution: **Unresolved**

Priority: **Medium** Fix Version/s: **None**

Affects Version/s: **None**

Component/s: **None**

Labels: [CSIC13_Regression](#) [SRSRevA](#) [SRSRevATCM](#) [SRSRevB](#)

Sprint: **VSM SRS Burn Down**

Requirement ID: **L3-VSM-0101-1090**

Parent Requirement ID: **L2-GW-0336, L2-VSM-0007, L2-VSM-0179**

RAC Level of Autonomy: **1**

Rationale:

- VSM must be able to begin initialization without human input after system boot-up given the frequency of human operator interactions planned for Gateway. State transitions are derived from GP 10012 Table 3.2.2-2. Initializing state is reported as the 1st state after the VSM Executive application has loaded and begun execution.

Verification Method: **Inspection**

Verification Statement: **Initializing state set after operating system boot-up and loading of the CFS core**

Confluence Vehicle Systems Manager Flight Software / VSM FSW Requirements 1 Jira link

Pages / Vehicle Systems Manager Flight Software / VSM FSW Requirements 1 Jira link

• It will manage active faults at the Gateway level, adjusting the Mission Plan and Mission Timeline as required, and analyze trends to identify potential faults and failures at the Gateway level

• The VSM, responsible for supporting the integration of ground and crew roles, by providing integrated emergency, warning, and caution alerting to human operators; supporting human operator analysis, decision making/diagnosis, and system interrogation; and performing response selection, planning, and execution

• VSM coordinates and monitors operations that span across multiple Gateway modules, such as refueling operations, EVR Ops, ECLS ops, etc.

• VSM provides inputs to module-level operation which require information on mission context, or on Gateway assembly configuration, or on ongoing operations

To fulfill these responsibilities, the VSM is decomposed into Functions and Subfunctions. Some examples are shown in the following table.

Function	Example Subfunctions	Example VSM Capability
Mission Plan & Timeline execution	<ul style="list-style-type: none"> Perform RPOD Refuel Utilize payloads 	<ul style="list-style-type: none"> Mission Timeline Planning/Replanning Task progression assessment Constraint-based planning and scheduling
Fault Management across modules	<ul style="list-style-type: none"> Emergency management (leak, fire) Recover from Power Bus failure Prioritize Recovery Actions 	<ul style="list-style-type: none"> Vehicle-wide health assessment Local safing action verification Vehicle recovery plan generation
Resource Management	<ul style="list-style-type: none"> Manage Power Production and Distribution Manage Data Networks Managing Crew Time Manage Pressure/O2/CO2 Manage data downlink content 	<ul style="list-style-type: none"> Resource projections based on current/potential plans Planning/scheduling for the optimization of interdependent resources
Vehicle Control and Operation	<ul style="list-style-type: none"> Control Attitude Optimize and Maintain Orbit Support Human Situation Awareness 	<ul style="list-style-type: none"> Human/system teaming

3.2 Functional and Performance Requirements

The requirement records are officially captured in the VSM FSW Jira issue tracking tool, and managed by the SynapseRT plugin (including capturing baselines). The table below is a direct export of all of the VSM FSW requirements from that tool.

Requirement ID	Summary	Description	Rationale	Parent Requirement ID	Verification Method	Verification Statement
L3-VSM-0101-1090	Initializing State	The VSM FSW shall enter the Initializing state after operating system boot-up and loading of the CFS core	VSM must be able to begin initialization without human input after system boot-up given the frequency of human operator interactions planned for Gateway. State transitions are derived from GP 10012 Table 3.2.2-2. Initializing state is reported as the 1st state after the VSM Executive application has loaded and begun execution.	L2-GW-0336, L2-VSM-0007, L2-VSM-0179	Inspection	Initializing state set after operating system boot-up and loading of the CFS core
L3-VSM-0102-1091	Configuring State	The VSM FSW shall transition to the Configuring state after initializing, or successfully completing Recovering operations	VSM must be able to begin Configuring without human input after completing initialization given the frequency of human operator interactions planned for Gateway. State transitions are derived from GP 10012 Table 3.2.2-2.	L2-GW-0336, L2-VSM-0007, L2-VSM-0009	Test	VSM FSW entered the configuring state immediately after successful initialization or alternatively completing recovery operations
L3-VSM-0103-1093	Self-Test State	The VSM FSW shall transition to the Self-Test state after successful Configuration	VSM must be able to automatically initiate Power On Self Test (POST) functions without human input after successful Configuration given the frequency of human operator interactions planned for Gateway. The Self-Test state also allows for manually commanded Built In Test (BIT) functions. State transitions are derived from GP 10012 Table 3.2.2-2.	L2-GW-0336, L2-VSM-0007, L2-VSM-0009	Test	VSM FSW transitioned to the Self-Test state after successful Configuration
L3-VSM-0104-1098	Standby State	The VSM FSW shall enter the Standby state upon successful completion of Self-Test	VSM must be able to enter Standby state without human input after successful Self-Test given the frequency of human operator interactions planned for Gateway. State transitions are derived from GP 10012 Table 3.2.2-2.	L2-GW-0336, L2-VSM-0007, L2-VSM-0009	Test	VSM FSW in Standby state at successful completion of Self-Test
L3-VSM-0105-1094	Control State	The VSM FSW shall enter the Control state from the Standby state upon: <ol style="list-style-type: none"> Receipt of state transition command After the timeout of a configurable timer, when configured as the Primary VSM instance, and the backup instance is not currently in control state After the failure to detect the Primary VSM instance heartbeat after a configurable threshold, when configured as the Backup VSM 	In order operate safely, VSM must be capable of autonomously transitioning to Control State when loss of Primary is detected, after a configurable timeout on startup or upon receipt of a command from a human operator. State transitions are derived from GP 10012 Table 3.2.2-2.	L2-GW-0336, L2-VSM-0007, L2-VSM-0009, L2-VSM-0010, L2-VSM-0039	Test	VSM FSW enters the Control State from the Standby state upon completion of conditions (1), (2), or (3):



Test Planning and Development



- Verification Statements are part of the Jira requirement ticket
- Each requirement is broken down into one or more test cases
 - Nominal
 - Erroneous
 - Boundary
 - Etc.
- Test cases are another Jira issue type from the TestRay tool set
- Review and collaboration continue in Jira
- PDF exports can be made as snapshot baselines for reviews
- Plan automated testing architecture

Requirements

Create Parent Create Child Link Parent Link Child Requirement Tree

VSMFSW-1090 Initializing State

VSMFSW-2422 Initializing State DONE

Test Cases

Total Test Cases: 1

Create Test Case Link Test Case

VSMFSW-8312 Inspection - Initializing state set after operating system boot-up and loading of the CFS core IN PROGRESS

Vehicle Systems Manager Flight Software VSMFSW-2092

Nominal - VSM FSW entered the Stabilizing state upon occurrence of an internal fault

Edit Q Add comment Assign More In Progress Export

Details

Type: Test Case Resolution: Unresolved
 Priority: Medium Fix Version/s: None
 Affects Version/s: None
 Component/s: None
 Labels: Nominal
 Sprint: VSM L3 Test Case Burndown

Description

Given:

- VSM FSW #1 nominal operation

When:

- Internal fault occurs

Then:

- VSM FSW detects an internal fault
- VSM FSW transitions to Stabilizing state per GP 10012

Should test a range of internal faults. Might be a set of tests for each fault.

Test Step

Estimate: Estimate Forecast: 6h
 (eg. 4w 3d 2h 10m)

#	Step	Expected Result

Automation

Test Reference: None

Ad hoc Test Run

Run ID	Result	Executed By	Executed On	Defect	Comments

Attachments

Drop files to attach, or browse.

Issue Links

mentioned in

- Test Case Design for VSMFSW-1088 Stabilizing State
- VE-VSM-AR-9,10,11 - Instance States Management and Transitions

Activity

All Comments Work Log History Activity Git Roll Up Git Commits

There are no comments yet on this issue.

People

Assignee: Andrew Santangelo
 Reporter: Andrew Santangelo
 Votes: Vote for this issue
 Watchers: Start watching this issue

Dates

Created: 09/Mar/21 9:00 PM
 Updated: Yesterday 1:07 PM

Time Tracking

Estimated: 1m
 Remaining: 1m
 Logged: Not Specified

Development

Latest 18/Jan/24 10:45 AM

Agile

Active Sprint: VSM L3 Test Case Burndown ends 28/May/25
 Find on a board

Requirement

Link

VSMFSW-1088 St... IN PROGRESS

Test Suite

Add

VSMFSW STATES

Test Plan

View All Defects

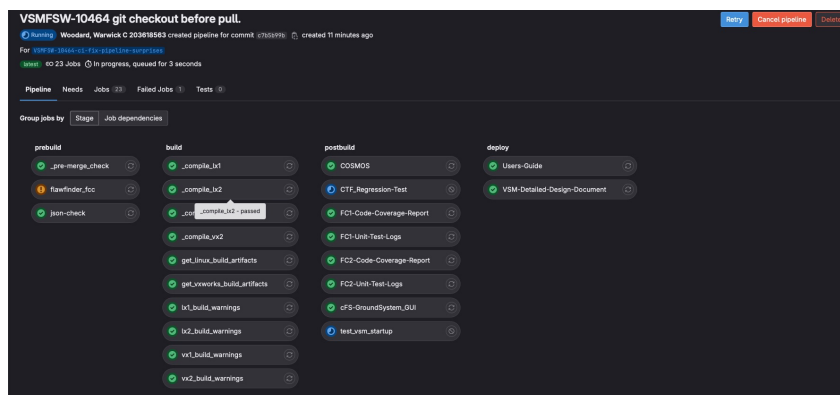
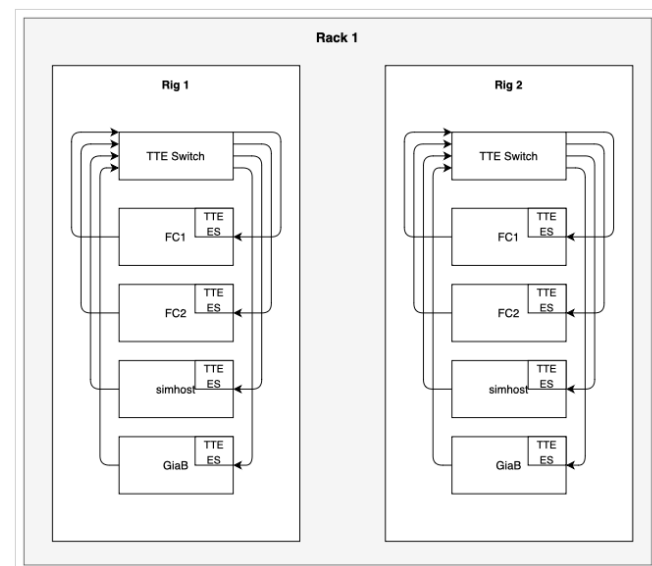
Test Plan	Result per Cycle	Defect
VSMFSW-103... VSM FSW Integration Test February 2024	■ ■ ■ ■	■ ■ ■ ■
VSMFSW-10374 VSM FSW Integration Test January 2024	■ ■ ■ ■	■ ■ ■ ■



Software Architecture & Implementation



- Even though there is a concentrated effort on requirements, we are still implementing capability
- Stand up development environment
 - Toolchain
 - CI pipeline
 - Lab
 - Hardware in the loop – the earlier the better!
 - Emulators
 - Simulators
 - Desktop development
 - Virtual machines
 - Docker images
- Implement preliminary architecture
 - All pieces of the architecture cycling in early development sprints





Use Case Based Advancement of System Capability



- Working with product owners and relevant stakeholders
 - Determine list of operational use cases that can be reasonably developed and demonstrated at this phase of the lifecycle
 - Use cases are peer reviewed
 - Used for training and familiarization
 - Helps to flush out requirements
 - Helps to define interfaces
- Advance hardware in the loop capabilities
 - Understand simulation and emulation requirements
 - Understand fidelity of hardware environment
- Integrated demonstration of capabilities and regular cadence throughout the phase
 - We chose a three-month integration cycle
 - Based on a two-week team sprint cycle
- Continuous Improvement of Tools and Processes
 - Enhancing CI pipeline
 - Development of emulators and simulators
 - Lessons learned from prior phases
 - Documenting process enhancements
 - Updating templates

CMV Software Integration Cycle 15 (CSIC-15)

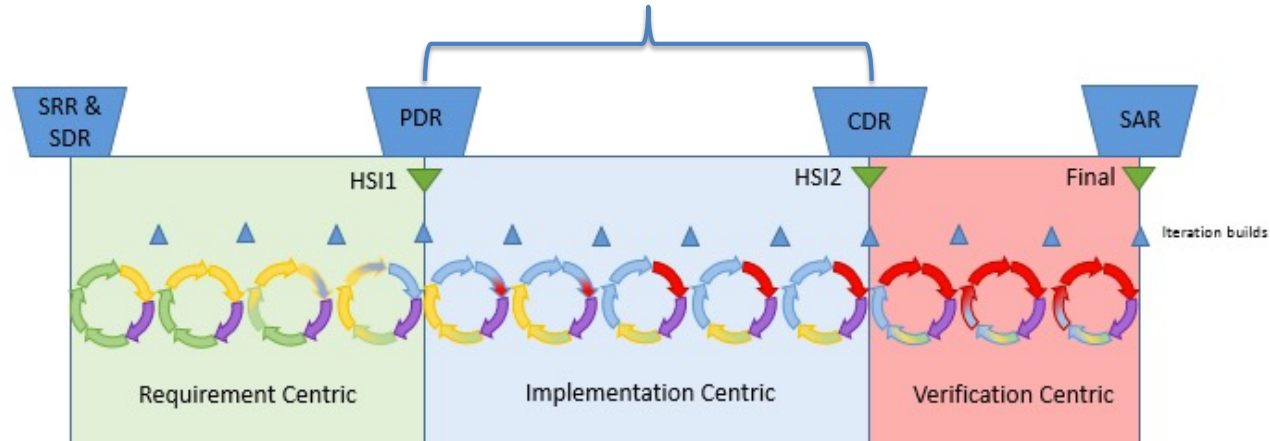
Created by Jia Liu, last modified by Eric Vieward on Nov 08, 2023

CSIC-15 Status

Title	Description	Collaboration	Jira	Development Status	Test Status
CSIC-14 UC: Module STRM Configuration and Downlink	This page captures the CSIC-14 use case(s) for Module STRM Configuration and Downlink. Defered to CSIC-15 due to missing NG support.	@Devon DeBakal @Minh Luong @Alex Lotze	VSMFWSW-7690 - CSIC-14 Module STRM Configuration and Downlink DONE	DELIVERED **The VSM implementation is done for the UC and it was demonstrated at the delta CDR and in ACSL. Will develop CTF script to verify the use case and then call it done.	<input type="checkbox"/> CTF Tested in ACSL <input checked="" type="checkbox"/> Delivered to GDS <input checked="" type="checkbox"/> Integrated and Tested in GSVL
CSIC-15 UC: SHM Exceedance Event	This page captures the use case of CSIC-15 for SHM	@Alex Lotze @Minh Luong	VSMFWSW-8916 - CSIC-15 SHM Exceedance Event DONE	DELIVERED One implementation story to finish in sprint 110. Estimated delivery in sprint 111. (Oct. 11-25, 2023)	<input checked="" type="checkbox"/> CTF Tested in ACSL <input checked="" type="checkbox"/> Delivered to GDS <input type="checkbox"/> Integrated and Tested in GSVL
CSIC-14 UC: Payloads and STRM Management	This page captures the use case of CSIC-14 for GPS Management. Defered to CSIC-15 due to missing NG support.	@Laura Barron - L2 Use Case @Sheif M Matta - L2 V&V Frameworks @Acmae El Yacoubi / @Felix Balderas - L2 V&V Test Scripts @Minh Luong - VSM FSW Developer	VSMFWSW-9390 - CSIC-14 Payloads and STRM GPS Management DONE VSMFWSW-9318 - CSIC-14 Payloads and STRM GPS Management UC DONE VSMFWSW-9351 - CSIC-14 Refine Payloads and STRM GPS Management UC DONE	DELIVERED From NG "HERMES" and PSM tasks are not part of HALO scope. Anything necessary there must be in a PSM task. The PSM commit hash in HALO FSW is 47572a5b09440f6ea255f88a84281c44503a. Since PSM is not formally in contact with HALO yet, we cannot upgrade that version until those agreements are in place. This use case needs to be refactored to use TRES task instead of lightweight tasks. Waiting for L2 confirmation and delivery of the payload TRES tasks.	<input checked="" type="checkbox"/> CTF Tested in ACSL <input checked="" type="checkbox"/> Delivered to GDS <input checked="" type="checkbox"/> Integrated and Tested in GSVL For offline discussion with @Krishna Kap 27 Oct 2023... we can temporary use values for testing this UC using Python st • "usPowerCurrentDraw" variable def "PSM_Payload_Exec_Data_1" of "PSM_Payload_RscTimMsg_1" tele feeder of "Payload_HERMES_SORP", constraint checking for lightweight 1664 • The LSB byte of the "usFreshCtrl" #1) in Payload Health State and Stat as the feeder of "Payload_HERMES_Science_Instru constraint checking for lightweight 1670



- Requirements Decomposition
- Design Elaboration
- **Use Case Based Advancement of System Capability**





Application Requirements Decomposition



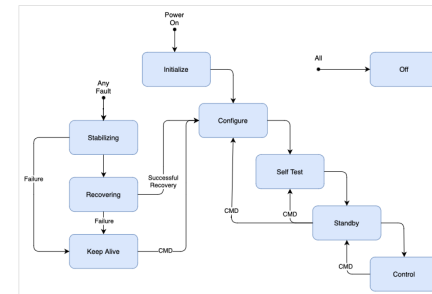
- Structured design process to decompose box level CSCI requirements into multiple application-level computer software component (CSC) requirements
 - Requirements based use cases drive system capabilities
 - 2nd tier requirements are also captured in Jira
 - Designs are documented via organized confluence pages
 - Structured templates to drive commonality
 - Jira requirements are linked into confluence design for traceability
 - Confluence allows for
 - Version control
 - Review / comments / collaboration
 - Flushing out interface design to both external and internal components

1. Instance State Management Use Cases

1.1 Instance State Management Overview

The instance state machine is the "prime mover" in the VSM after cFE initializes all the applications. It autonomously initiates actions across the VSM on startup and on internal fault. Because Executive does a lot of specific commanding to other VSM applications, it is less of a general purpose application and works mainly as a piece of VSM, not an independent application.

The instance state machine as implemented in Executive can be seen here:



1.2 Power On to Initializing Use Case

Use Case Name	Initializing	
Participants	Executive	
Assumptions	None	
Pre-conditions	None	
Scenario	Description	Requirement Tracing
1	sp0 is powered on	
2	VvWorks boots	
3	CFS/VSM boots	
4	Executive initializes Hk TLM and Outdata TLM with ASM instance state = Initializing State	VSMFSW-2422 - Initializing State DONE
5	Executive checks if stabilizing state flag is set to 1	
5.1	Stabilizing state flag is not set to 1, Executive begins transition to Configuring State Scenario options in configuring state: Default Configuration, Safe State Configuration, Synchronization Request	VSMFSW-3474 - Initialization Criteria DONE
Post-conditions	Executive starts autonomous transition to configuring state	
Implementation/Design Tracing		



Design Products



- All design artifacts are “organic” products that are created inline as part of the development process
 - We don’t stop what we are doing to make a big Word document and a bunch of PowerPoint slides
- Utilize Doxygen, Confluence exports, and Jira scripting to create the design package
 - HTML based “clickable” package



Gateway Vehicle Systems Manager Flight Software

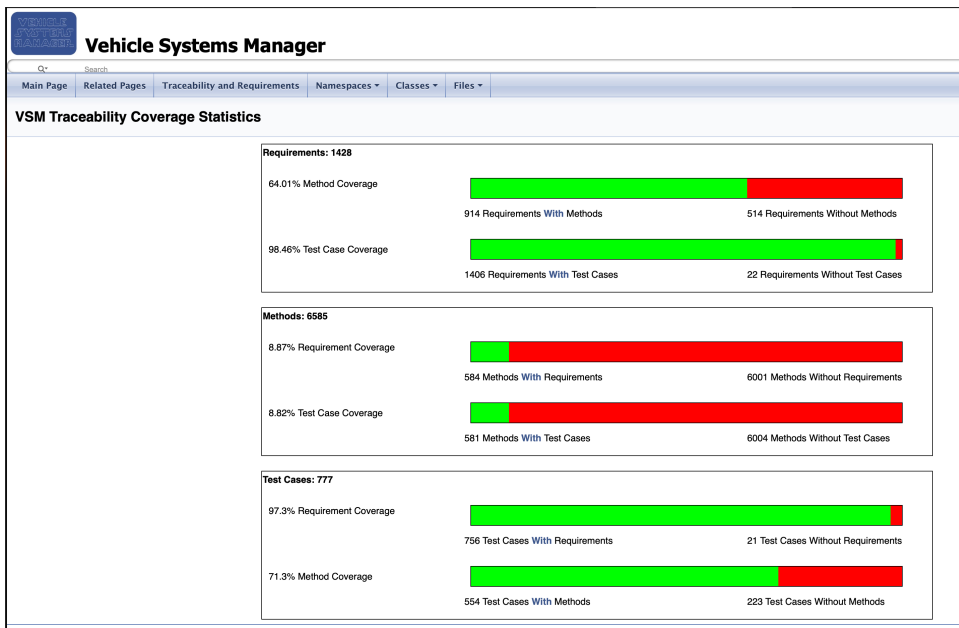
The screenshot shows the 'Gateway VSM Applications Documentation' website. The navigation menu includes: Main Page, VSM Introduction and Design, Traceability and Requirements, External Design Documents, Gateway VSM Applications Documentation, Gateway cFS Applications Documentation, and Related Pages. The main content area is titled 'Gateway VSM Applications Documentation' and lists various documentation items under the heading 'PREV: External Design Documents'. The list includes: CA Application Documentation, CDM Application Documentation, CDP Application Documentation, COMM_MGR Application Documentation, DISPATCHER Application Documentation, EWCA Application Documentation, EXECUTIVE Application Documentation, FAULT_MGR Documentation, GT Documentation, PLANNER Documentation, PWR_MGR Documentation, RuM Documentation, RSRM_MGR Documentation, SD Documentation, SOLITE_IF Documentation, TREX Documentation, STRM_IF Documentation, TASKMGR Documentation, and VV_MGR (Sustaining Phase Capability) Documentation. The footer indicates the documentation was generated on Thu Feb 15 2024 13:39:35 for Gateway VSM FSW by doxygen 1.8.14.

The screenshot shows the GitHub repository page for 'Vehicle System Manager'. The repository has 9,555 Commits, 460 Branches, 52 Tags, 840.7 GiB Project Storage, and 47 Releases. The repository description is 'Vehicle System Manager (VSM) Flight Software (FSW)'. The page shows a merge request for branch 'development' from 'fcc' with commit hash '3e7385d6'. Below the merge request, there are buttons for 'README', 'CI/CD configuration', 'Wiki', 'Add LICENSE', 'Add CHANGELOG', 'Add CONTRIBUTING', and 'Add Kubernetes cluster'. A table of recent commits is visible at the bottom, with the most recent commit being 'VSMFSW-9601 Test Case Implementation for VSMFSW...' by 'gitlab/merge_request_templates' 2 months ago.

- NPR 7150.2 requires requirements and verifications traceability
 - Depending on software classification
- Utilizing custom Doxygen tags in the source code, along with Jira scripting we can generate tables for bi-directional traceability
 - Requirement \leftrightarrow Design \leftrightarrow Code
 - Requirement \leftrightarrow Code \leftrightarrow Test



Gateway Vehicle Systems Manager Flight Software



Related Pages | Files

VSM Introduction and Design | Traceability and Requirements | External Design Documents | Gateway VSM Applications Documentation | Gateway CFS Applications Documentation

Search

VSMFSW Traceability Matrix: Test Case by Requirement - Components

L3 Requirements	L4 Requirements	Test Cases
L3-VSM-1203-1872: Fault Identification L3-VSM-1209-1758: Diagnostic information	L4-VSM-ACAWS-10329: ACAWS: Publish Fault Input Data	VSMSFSW-6009: Erroneous - VSM FSW failed to telemeter parameters that are used in fault detection and determination logic VSMSFSW-6008: Nominal - VSM FSW telemetered any parameters that are used in fault detection and determination logic
L3-VSM-1810-1619: VSM Command Authorization	L4-VSM-ACAWS-10068: ACAWS: Command Source Validation	VSMSFSW-5688: Erroneous - VSM Unique Two-Step Command Timeout VSMSFSW-5687: Nominal - VSM successfully uses required unique two-step commands for all critical functions
L3-VSM-1206-1785: Cross Module Function Availability L3-VSM-1220-2888: Vehicle Component Health State L3-VSM-1306-2892: Determine Current Vehicle Conditions L3-VSM-1205-1750: Functional Availability L3-VSM-1207-1752: Redundancy Impact Information	L4-VSM-ACAWS-6598: ACAWS_DE - Publish the functional health impact status array	VSMSFSW-5844: Erroneous - VSM FSW fails to maintain the redundancy status of Gateway functions as defined in the Gateway VSM to MSM ICD (GP 10088) VSMSFSW-5843: Nominal - VSM FSW maintains the redundancy status of Gateway functions as defined in the Gateway VSM to MSM ICD (GP 10085)
L3-VSM-1301-1900: Vehicle-level Fault Detection	L4-VSM-ACAWS-6647: ACAWS_DE acaws_mode_types Consumption	VSMSFSW-6897: Combinatorial - VSM FSW calculated expected vehicle condition, determined the current vehicle conditions, and compared the expected to the current conditions to detected vehicle-level faults
L3-VSM-1301-1900: Vehicle-level Fault Detection	L4-VSM-ACAWS-6648: ACAWS_FD_CFS acaws_mode_types Output	VSMSFSW-6897: Combinatorial - VSM FSW calculated expected vehicle condition, determined the current vehicle conditions, and compared the expected to the current conditions to detected vehicle-level faults
L3-VSM-1301-1900: Vehicle-level Fault Detection	L4-VSM-ACAWS-6182: ACAWS_FD_CFS initialization configuration	VSMSFSW-6897: Combinatorial - VSM FSW calculated expected vehicle condition, determined the current vehicle conditions, and compared the expected to the current conditions to detected vehicle-level faults
L3-VSM-1301-1900: Vehicle-level Fault Detection	L4-VSM-ACAWS-6180: ACAWS_DE Pass-Fail Consumption	VSMSFSW-6897: Combinatorial - VSM FSW calculated expected vehicle condition, determined the current vehicle conditions, and compared the expected to the current conditions to detected vehicle-level faults
L3-VSM-1301-1900: Vehicle-level Fault Detection L3-VSM-0216-1828: Mode Constraints L3-VSM-0802-1713: Modify Vehicle Configuration L3-VSM-1116-7859: EWCA Configuration	L4-VSM-ACAWS-6179: ACAWS_FD_CFS Mode Switching	VSMSFSW-8180: Combinatorial - VSM FSW automatically suppressed the appropriate EWCA alerts based on commanded vehicle state, configuration, and mode
L3-VSM-1220-2888: Vehicle Component Health State L3-VSM-1306-2892: Determine Current Vehicle Conditions L3-VSM-1204-1749: Component Health State	L4-VSM-ACAWS-7684: ACAWS_DE: Component Health States	VSMSFSW-6200: Erroneous - VSM FSW fails to monitor lower level component health state as defined in the Gateway VSM to MSM ICD VSMSFSW-6199: Nominal - VSM FSW monitored lower level component health state as defined in the Gateway VSM to MSM ICD
		VSMSFSW-6009: Erroneous - VSM FSW failed to telemeter parameters that are used in fault detection

Generated on Thu, Feb 15 2024 13:09:34 for Gateway VSM FSW by doxygen 1.8.14



Use Case Based Advancement of System Capability



- Working with product owners and relevant stakeholders
 - Determine list of operational use cases that can be reasonably developed and demonstrated at this phase of the lifecycle
 - Use cases are peer reviewed
 - Used for training and familiarization
 - Helps to **clarify** requirements
 - Helps to **understand** interfaces
- Advance hardware in the loop capabilities
 - **Develop** simulation and emulation **capabilities**
 - **Enhance** fidelity of hardware environment
- Integrated demonstration of capabilities and regular cadence throughout the phase
 - We chose a three-month integration cycle
 - Based on a two-week team sprint cycle
- Continuous Improvement of Tools and Processes
 - Enhancing CI pipeline
 - Development of emulators and simulators
 - Lessons learned from prior phases
 - Documenting process enhancements
 - Updating templates

CMV Software Integration Cycle 15 (CSIC-15)

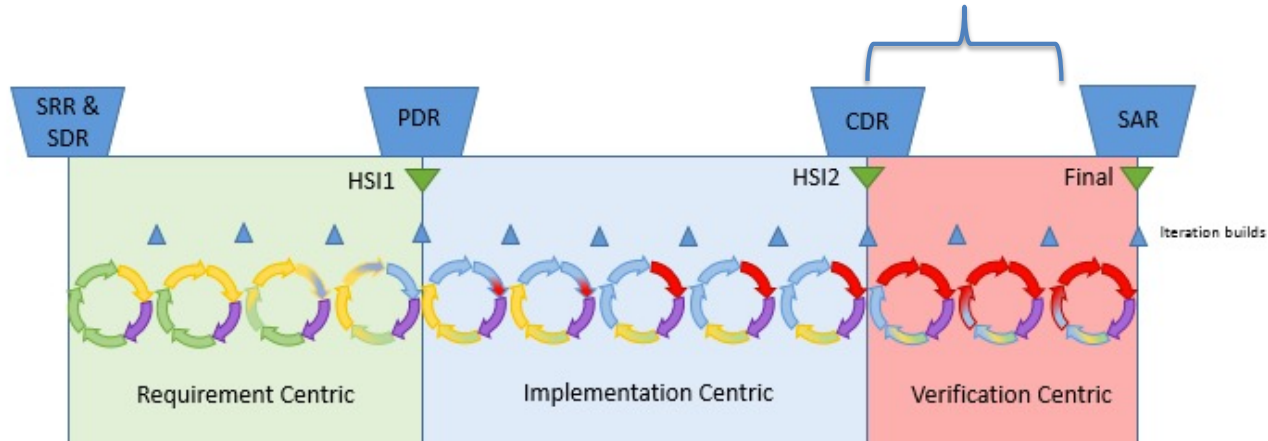
Created by Jia Liu, last modified by Eric Vineyard on Nov 05, 2023

CSIC-15 Status

Title	Description	Collaboration	Jira	Development Status	Test Status
CSIC-14 UC: Module STRM Configuration and Downlink	This page captures the CSIC-14 use case(s) for Module STRM Configuration and Downlink. Defered to CSIC-15 due to missing NG support.	@Devon DeBassi @Minh Luong @Alex Lotze	VSMFSW-7690 - CSIC-14 Module STRM Configuration and Downlink. Done	DELIVERED **The VSM implementation is done for the UC and it was demonstrated at the delta CDR and in ACSL. Will develop CTF script to verify the use case and then call it done.	<input type="checkbox"/> CTF Tested in ACSL <input checked="" type="checkbox"/> Delivered to GDS <input checked="" type="checkbox"/> Integrated and Tested in GSVL
CSIC-15 UC: SHM Exceedance Event	This page captures the use case of CSIC-15 for SHM	@Alex Lotze @Minh Luong	VSMFSW-8919 - CSIC-15 SHM Exceedance Event. Done	DELIVERED One implementation story to finish in sprint 110. Estimated delivery in sprint 111. (Oct. 11-25, 2023)	<input checked="" type="checkbox"/> CTF Tested in ACSL <input checked="" type="checkbox"/> Delivered to GDS <input type="checkbox"/> Integrated and Tested in GSVL
CSIC-14 UC: Payloads and STRM Management	This page captures the use case of CSIC-14 for GPFS management. Defered to CSIC-15 due to missing NG support.	@Laura Barron - L2 Use Case @Sherif M Matta - L2 V&V Frameworks @Acmae El Yacoubi / @Felix Balderas - L2 V&V Test Scripts @Minh Luong - VSM FSW Developer	VSMFSW-8930 - CSIC-14 Payloads and STRM GPFS Management. Done VSMFSW-8919 - CSIC-14 Payloads and STRM GPFS Management UC. Done VSMFSW-8821 - CSIC-14 Refine Payloads and STRM GPFS Management UC. Done	DELIVERED From NG "HERMES" and PSM tasks are not part of HALO scope. Anything necessary there must be in an PSM task. The PSM commit hash in HALO FSW is 6757205b0940f66ca255f8ba84281c44502a. Since PSM is not formally on contract with HALO yet, we cannot upgrade that version until those agreements are in place. This use case needs to be refactored to use TREQ task instead of lightweight tasks. Waiting for L2 confirmation and delivery of the payload TREQ tasks.	<input checked="" type="checkbox"/> CTF Tested in ACSL <input checked="" type="checkbox"/> Delivered to GDS <input type="checkbox"/> Integrated and Tested in GSVL For offline discussion with @Krishna Kap 27 Oct 2023, we can temporary use values for testing this UC using Python stz • "usPowerCurrentDraw" variable def "PSM_Payload_RscData_1" of "PSM_Payload_RscData_1" tele feeder of "Payload_HERMES_SORP", constraint checking for lightweight 1664 • The LSB byte of the "usFreshCtrl" #13 in Payload Health State and Stat as the feeder of "Payload_HERMES_Science_Instru", constraint checking for lightweight 1670



- Final Implementation of Design
- Test Script Development
- Continuous Integration Pipeline
- **Use Case Based Advancement of System Capability**





Closing Application Requirements



- Implement the requirement, using the design document as a reference
 - Update the design page in Confluence if necessary
- Tag all source code functions implementing the requirement with custom Doxygen tag
- Unit test all the functions in the implementation
- Verify the implementation matches the design document
- Update associated data products
 - Table definitions
 - Commands
 - Telemetry
- Peer Reviews

Pages / Vehicle Systems Manager Flight Software / How-to articles 1 Jira link

Edit Save for later Watch Share

L4 Requirements Burndown

Created by Sebastian Fisher, last modified by Emily Buergerler on Feb 15, 2023

Process

This page describes the process for moving L4 requirements from TODO to DONE

1. If a Jira implementation story is not linked to the L4 requirement, create a Jira story and link to the L4 requirement using "covers" link
2. Make sure the implementation story has L4_IMP label
3. Pull the L4 implementation story into the sprint
4. Move the L4 requirement to "In Progress"
5. Implement the requirement, using the design document as a reference
 - a. If needed, propose updates to the L4's or design document
6. Tag all functions implementing the requirement with @SatisfiesReq(App, Req Number) tag
7. Unit test all the functions in the implementation
8. Verify the implementation matches the design document
9. If implementation leads to what you think should be an additional VSM data product from GDS (say a new CFE Table) or removal of a data product formerly listed on VSM Data Products From GDS, update the table(s) on the link and alert VSMFSW GDS Focal @Alex Lotze
10. Update the L4 and Jira story to "In Review"
11. Have 2 reviewers look through the implementation, unit tests, and design
12. Get 2 approvals
13. Move the L4 Requirement to "Done"
14. Make sure all the links on Data Products From GDS for the affected application(s) (including submodules) are up-to-date with any merged changes

If the process of re-design of design page from step 11 or general implementation from above leads to needing to mark an L4 as OBE ("overcome by events"):

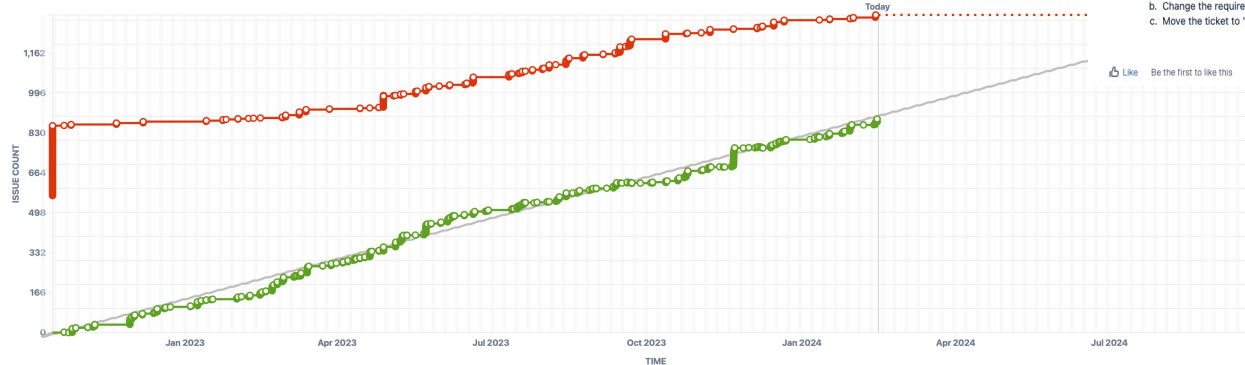
1. Obtain 2 approvals (can be same as 2 approvals from process above if you instruct the reviewers to also consider the L4 OBE'ing as part of their review)
2. Tag Dave Swartwout in the comment of the JIRA ticket hosting that L4 requirement. Example: [VSMFSW-3689 - SD: MSM Power](#) **DONE**
3. On Dave's concurrence:
 - a. Update design pages that mention the to-be-OBE'd L4
 - b. Change the requirement's issue type to "Requirement" ("More" -> "Move")
 - c. Move the ticket to "Closed" with resolution "Won't Do"

Like Be the first to like this

No labels

VSM L4 Requirements Burndown Issue Count How to read this chart

ACTIVE





Test Script Development



- Using test case statements defined in Jira/TestRay
- Structured test case design process
 - Captured in Confluence
 - Test team collaborates with design team
- Implement automated test script for each test case
- Test development in parallel with system development
 - Flushes out requirements questions, inconsistencies, and confusions
 - Clarifies design
 - Uncovers foundational issues early

Pages / ... / L3 Test Case Design     ...

Test Case Design for VSMFSW-1118 Transitioning from Control to Standby

Created by Emily Buegler, last modified by David Bernal on Feb 08, 2024

Overview

L3 Requirement	Test Plan	Test Cases	Test Development Stories
Transitioning from Control to Standby	Prelim-8, Set 6	VSMFSW-2596 - Nominal - Primary VSM transitioned from Control to Standby upon receipt of a state transition command. IN PROGRESS	VSMFSW-4341 - Develop CTF Test Plan for Transitioning from Control to Standby. DONE
VSMFSW-1118 - Transitioning from Control to Standby. IN PROGRESS	VSMFSW-4330 - Development-8, 1118. DONE	VSMFSW-2599 - Erroneous - Backup VSM does not transition from Standby to Control after receiving a state transition command. IN PROGRESS	VSMFSW-7813 - Investigate regression test failure for VSMFSW-1118 Requirement. DONE
	VSMFSW-10284 - Regression Test Review - 1118. DONE	VSMFSW-4347 - Erroneous - Primary VSM does not transition from Control to Standby due to Primary VSM unresponsive upon receipt of a state transition command. IN PROGRESS	VSMFSW-10216 - Debug VSMFSW-1118 CTF possible regression failure. DONE
		VSMFSW-4348 - Erroneous - Primary VSM does not transition from Control to Standby due Backup VSM unresponsive upon receipt of a state transition command. IN PROGRESS	

L3 Requirement Verification Statement: Gateway VSM transitioned from Control to Standby upon receipt of a state transition command

Table of Contents

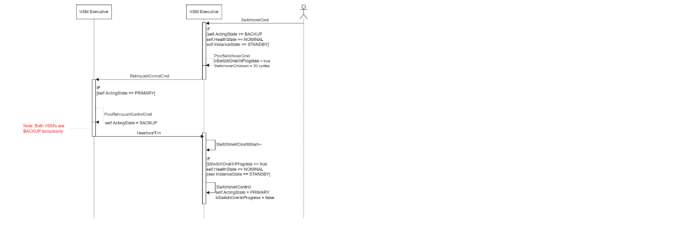
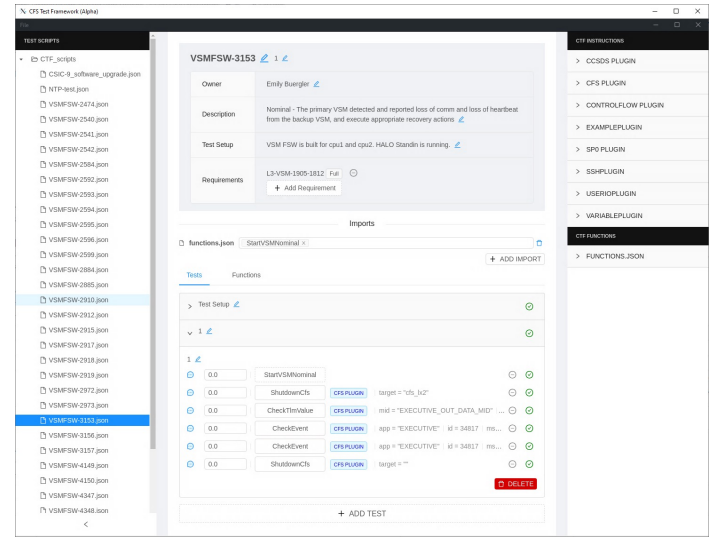
- Overview
 - Table of Contents
 - Acronyms
 - VSM Nominal Switchover Design
- Test Case Design
 - VSMFSW-2596 Nominal - Primary VSM transitioned from Control to Standby upon receipt of a state transition command
 - CTF Test Procedure
 - VSMFSW-2599 Erroneous - Backup VSM does not transition from Standby to Control after receiving a state transition command
 - CTF Test Procedure
 - VSMFSW-4347 Erroneous - Primary VSM does not transition from Control to Standby due to Primary VSM unresponsive upon receipt of a state transition command
 - CTF Test Procedure
 - Additional Notes
 - VSMFSW-4348 Erroneous - Primary VSM does not transition from Control to Standby due to Backup VSM unresponsive upon receipt of a state transition command
 - CTF Test Procedure
 - Additional Notes
- Implementation & Development
 - Test Results
 - Run_10_06_2021_21_13_53 PASSED
 - Run_10_06_2021_22_06_47 PASSED
 - Run_10_08_2021_20_10_30 FAILED
 - Run_10_08_2021_20_44_31 FAILED
 - Run_10_08_2021_22_20_32 BLOCKED
 - Run_10_13_2021_18_27_27 PASSED
 - Run_03_04_2022_22_06_36 PASSED
 - Run_03_04_2022_22_06_06 PASSED
 - Run_03_05_2022_00_03_11 PASSED

Acronyms

ASM = Autonomous System Management

VSM Nominal Switchover Design

The sequence diagram shown below gives a high level overview of the switch over logic handled by EXECUTIVE. Note that this scenario is a nominal switch-over, which is initiated by sending a "SwitchoverCmd" to VSM. This design was reviewed and discussed with the team on 11-18-2020.



The Pipeline



- Continuous Integration Pipeline for everything that can be automated
 - Static code analysis tools
 - Compilation warnings
 - Build errors
 - “Hello World” integrated tests
 - Generation of design package
 - Generation of release products
- Required before code can be integrated (merged)

VSMFSW-10464 git checkout before pull. Retry Cancel pipeline Delete

Running

For VSMFSW-10464-ci-fsa-pipeline-surprises

latest 23 Jobs 3 In progress, queued for 3 seconds

Pipeline Needs Jobs 23 Failed Jobs 0 Tests 0

Group jobs by Stage Job dependencies

prebuild	build	postbuild	deploy
✓ _pre-merge_check	✓ _compile_bx1	✓ COSMOS	✓ Users-Guide
⚠ flawfinder_fcc	✓ _compile_bx2	⚡ CTF_Regression-Test	✓ VSM-Detailed-Design-Document
✓ json-check	✓ _cor_compile_bx2 - passed	✓ FC1-Code-Coverage-Report	
	✓ _compile_vx2	✓ FC1-Unit-Test-Logs	
	✓ get_linux_build_artifacts	✓ FC2-Code-Coverage-Report	
	✓ get_vxworks_build_artifacts	✓ FC2-Unit-Test-Logs	
	✓ bx1_build_warnings	✓ cFS-GroundSystem_GUI	
	✓ bx2_build_warnings	⚡ test_vsm_startup	
	✓ vx1_build_warnings		
	✓ vx2_build_warnings		



Use Case Based Advancement of System Capability



- Working with product owners and relevant stakeholders
 - Determine list of operational use cases that can be reasonably developed and demonstrated at this phase of the lifecycle
 - Use cases are peer reviewed
 - Used for training and familiarization
 - Helps to **finalize** requirements
 - Helps to **finalize** interfaces
- Advance hardware in the loop capabilities
 - **Finalize** simulation and emulation capabilities
 - **Finalize** fidelity of hardware environment
- Integrated demonstration of capabilities and regular cadence throughout the phase
 - We chose a three-month integration cycle
 - Based on a two-week team sprint cycle
- Continuous Improvement of Tools and Processes
 - Enhancing CI pipeline
 - Development of emulators and simulators
 - Lessons learned from prior phases
 - Documenting process enhancements
 - Updating templates

CMV Software Integration Cycle 15 (CSIC-15)

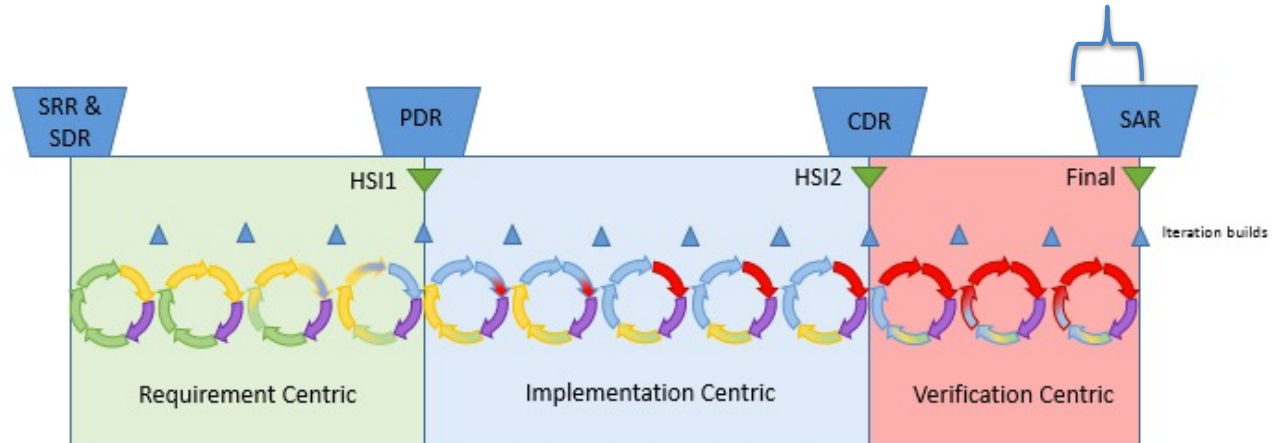
Created by Jia Liu, last modified by Eric Vineyard on Nov 05, 2023

CSIC-15 Status

Title	Description	Collaboration	Jira	Development Status	Test Status
CSIC-14 UC: Module STRM Configuration and Downlink	This page captures the CSIC-14 use case(s) for Module STRM Configuration and Downlink. Defered to CSIC-15 due to missing NG support.	@Devon DeBassi @Minh Luong @Alex Lotze	VSMFSW-7690 - CSIC-14 Module STRM Configuration and Downlink. Done	DELIVERED **The VSM implementation is done for the UC and it was demonstrated at the delta CDR and in ACSL. Will develop CTF script to verify the use case and then call it done.	<input type="checkbox"/> CTF Tested in ACSL <input checked="" type="checkbox"/> Delivered to GDS <input checked="" type="checkbox"/> Integrated and Tested in GSVL
CSIC-15 UC: SHM Exceedance Event	This page captures the use case of CSIC-15 for SHM	@Alex Lotze @Minh Luong	VSMFSW-8919 - CSIC-15 SHM Exceedance Event. Done	DELIVERED One implementation story to finish in sprint 110. Estimated delivery in sprint 111. (Oct. 11-25, 2023)	<input checked="" type="checkbox"/> CTF Tested in ACSL <input checked="" type="checkbox"/> Delivered to GDS <input type="checkbox"/> Integrated and Tested in GSVL
CSIC-14 UC: Payloads and STRM Management	This page captures the use case of CSIC-14 for GPFS management. Defered to CSIC-15 due to missing NG support.	@Laura Barron - L2 Use Case @Sherif M Matta - L2 V&V Frameworks @Acmae El Yacoubi / @Felix Balderas - L2 V&V Test Scripts @Minh Luong - VSM FSW Developer	VSMFSW-8930 - CSIC-14 Payloads and STRM GPFS Management. Done VSMFSW-8919 - CSIC-14 Payloads and STRM GPFS Management UC. Done VSMFSW-8821 - CSIC-14 Refine Payloads and STRM GPFS Management UC. Done	DELIVERED From NG "HERMES" and PSM tasks are not part of HALO scope. Anything necessary there must be in an PSM task. The PSM commit hash in HALO FSW is 67572d5b94940f6e255f8ba84281c44502a. Since PSM is not formally on contract with HALO yet, we cannot upgrade that version until those agreements are in place. This use case needs to be refactored to use TRES task instead of lightweight tasks. Waiting for L2 confirmation and delivery of the payload TRES tasks.	<input checked="" type="checkbox"/> CTF Tested in ACSL <input checked="" type="checkbox"/> Delivered to GDS <input type="checkbox"/> Integrated and Tested in GSVL For offline discussion with @Krishna Kap 27 Oct 2023, we can temporary use values for testing this UC using Python st • "usPowerCurrentDraw" variable def "PSM_Payload_RscData_1" of "PSM_Payload_RscData_1" tele feeder of "Payload_HERMES_SOP", constraint checking for lightweight 1664 • The LSB byte of the "usFreshCtrl" #13 in Payload Health State and Star as the feeder of "Payload_HERMES_Science_Instru constraint checking for lightweight 1670



- Formal Test and Verification

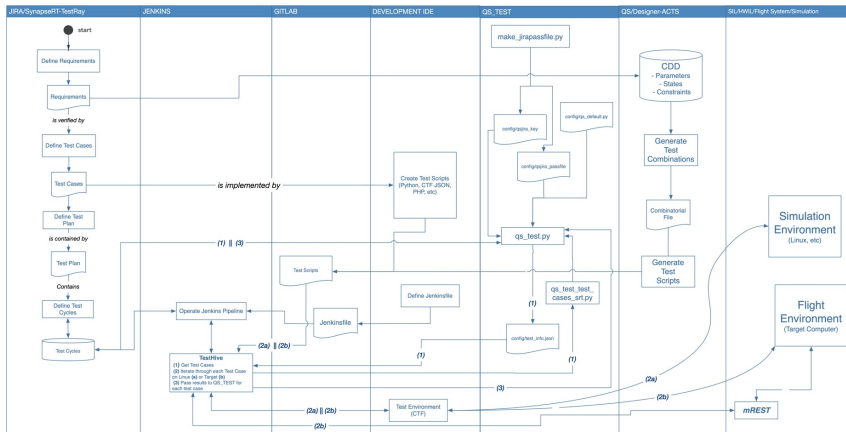




Automated Testing



- Created around Jira/TestRay and Jira API scripting
- TestRay allows creation of Test Plans
 - Grouping of test cases into test activities
 - Plans can be executed in batch
- Jira back end communicates with Jenkins server for automated test execution
 - Test cycles executed
 - Development servers
 - Hardware in the Loop Rigs
 - Test results pushed back into Jira/TestRay
 - Test execution status tracked to completion
 - Defects documented



Vehicle Systems Manager Flight Software / VSMFSW-10399
VSM FSW Integration Test February 2024

Details
 Type: Test Plan
 Priority: Medium
 Affects Version/s: None
 Component/s: None
 Labels: Regression_Testing

Description
 Covers all Requirements and supporting test cases created through the end of November 2023

Test Case
 Total Test Cases Planned: 206

Test Case	Priority	Status	Tester	Execution Status
VSMFSW-6541 Nominal - VSM FSW performs a consequence analysis when a new timeline is uplinked from the ground	High	IN PROGRESS	Andrew Santangelo	100%
VSMFSW-6542 Erroneous - VSM FSW performs a consequence analysis when a new timeline is uplinked from the ground	High	IN PROGRESS	Andrew Santangelo	100%
VSMFSW-6543 Erroneous - VSM FSW fails to perform a consequence analysis when a new timeline is uplinked from the ground	High	IN PROGRESS	Andrew Santangelo	100%
VSMFSW-3837 Nominal - VSM FSW did not execute a task if the required resources are not available	High	IN PROGRESS	Andrew Santangelo	100%
VSMFSW-3838 Erroneous - VSM FSW executes a task/attempts to execute even if the required resources are not available	High	IN PROGRESS	Andrew Santangelo	100%
VSMFSW-8097 Nominal - VSM FSW rejected the execute command of an arm/execute pair if the command is not armed	High	IN PROGRESS	Andrew Santangelo	100%
VSMFSW-5983 Nominal - VSM FSW rejected commands that are incompatible with the active vehicle mode and communicates rejection to human operators	High	IN PROGRESS	Andrew Santangelo	100%
VSMFSW-5984 Erroneous - VSM FSW accepted command that is incompatible with the active vehicle mode	High	IN PROGRESS	Andrew Santangelo	100%

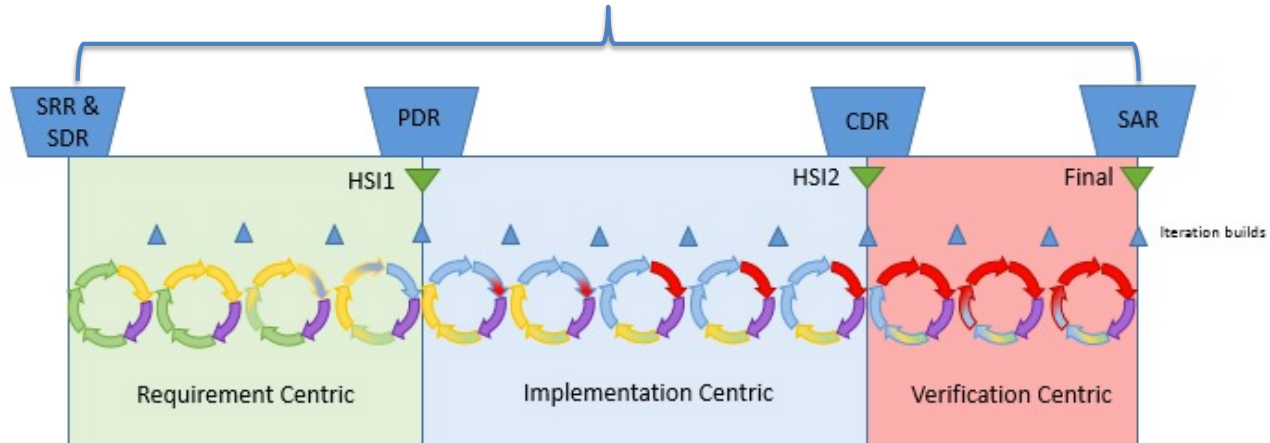
Agile
 Find on a board

Requirement

Requirement	Plan Coverage	Execution Coverage
VSMFSW-1784 Single Controlling VSM	100.0%	100%
VSMFSW-1619 VSM Command Authorization	100.0%	100%
VSMFSW-2899 Timeline Merge Consequence Analysis	100.0%	100%
VSMFSW-1675 Gateway SPP CRC Computations - Location	100.0%	100%
VSMFSW-1678 Gateway Command Response	100.0%	100%
VSMFSW-1665 Command Age Validation - Check Age Limit	100.0%	100%
VSMFSW-2089 VSM Configuration Data Validation	50.0%	100%
VSMFSW-1668 Command Age Validation - Age Limit Exceeded	100.0%	100%
VSMFSW-1858 Time Synchronization	100.0%	100%
VSMFSW-1662 Checkpoint/Restore Load - critical state data	100.0%	100%



- Cross Team Integration

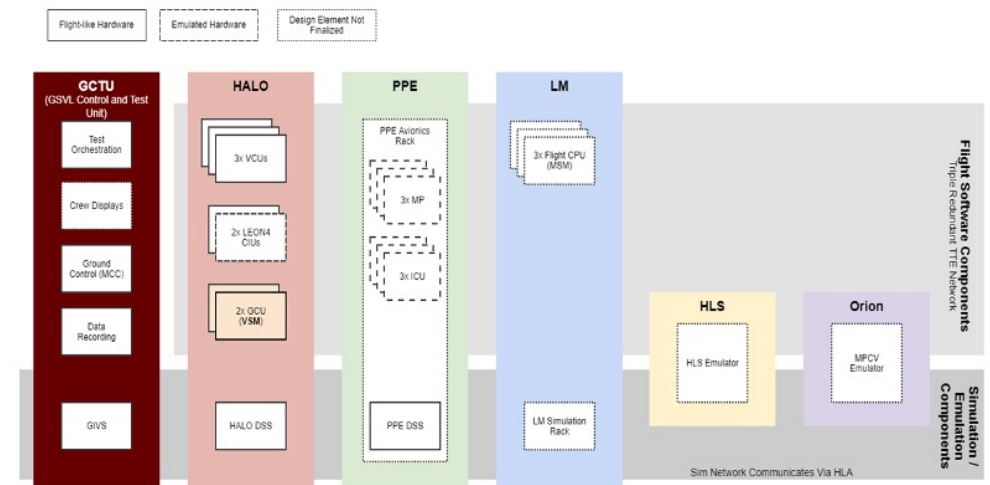




Program Wide Integration Cycles

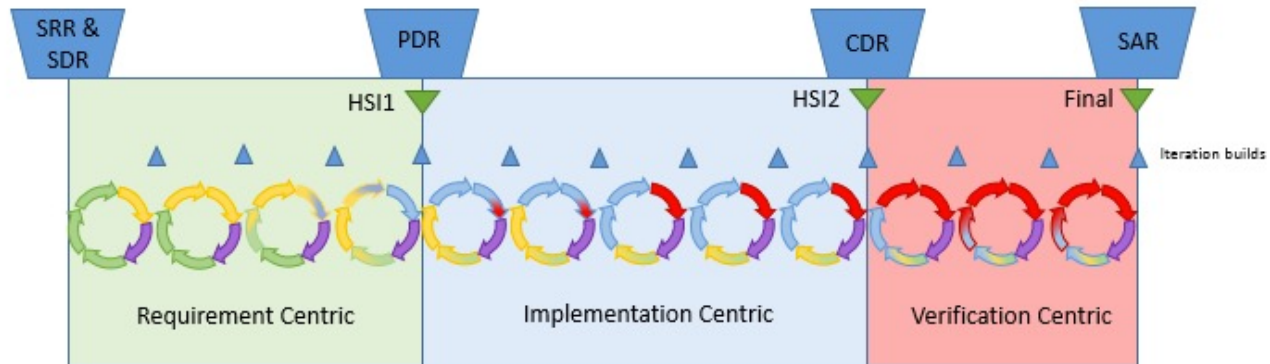


- Gateway is a large program – 11 software development teams (plus sub teams) working initial element launch
 - Several larger contractors
 - International Partners
 - Numerous NASA internal teams
 - Test & Verification Labs
 - Countless stakeholders
- We had to find a way to bring these teams together early and often
 - Gateway Software Integration Cycles
 - Three month cadence
 - Quarterly planning
- Essential for driving out interfaces and integrated system operations





Takeaways





Closing Thoughts



- On any evolving program where change is ongoing, agility is key
 - Continual learning is essential
 - Continual feedback and process improvement
- We have to live within the NPR requirements
 - Don't fight it, try to understand it
 - Tailor to what makes sense
 - Look for the intent
 - Why do we have this artifact or milestone review
 - Focus on the "what", not the "how"
 - I have to have requirements, but I don't need a 300 page Word document

