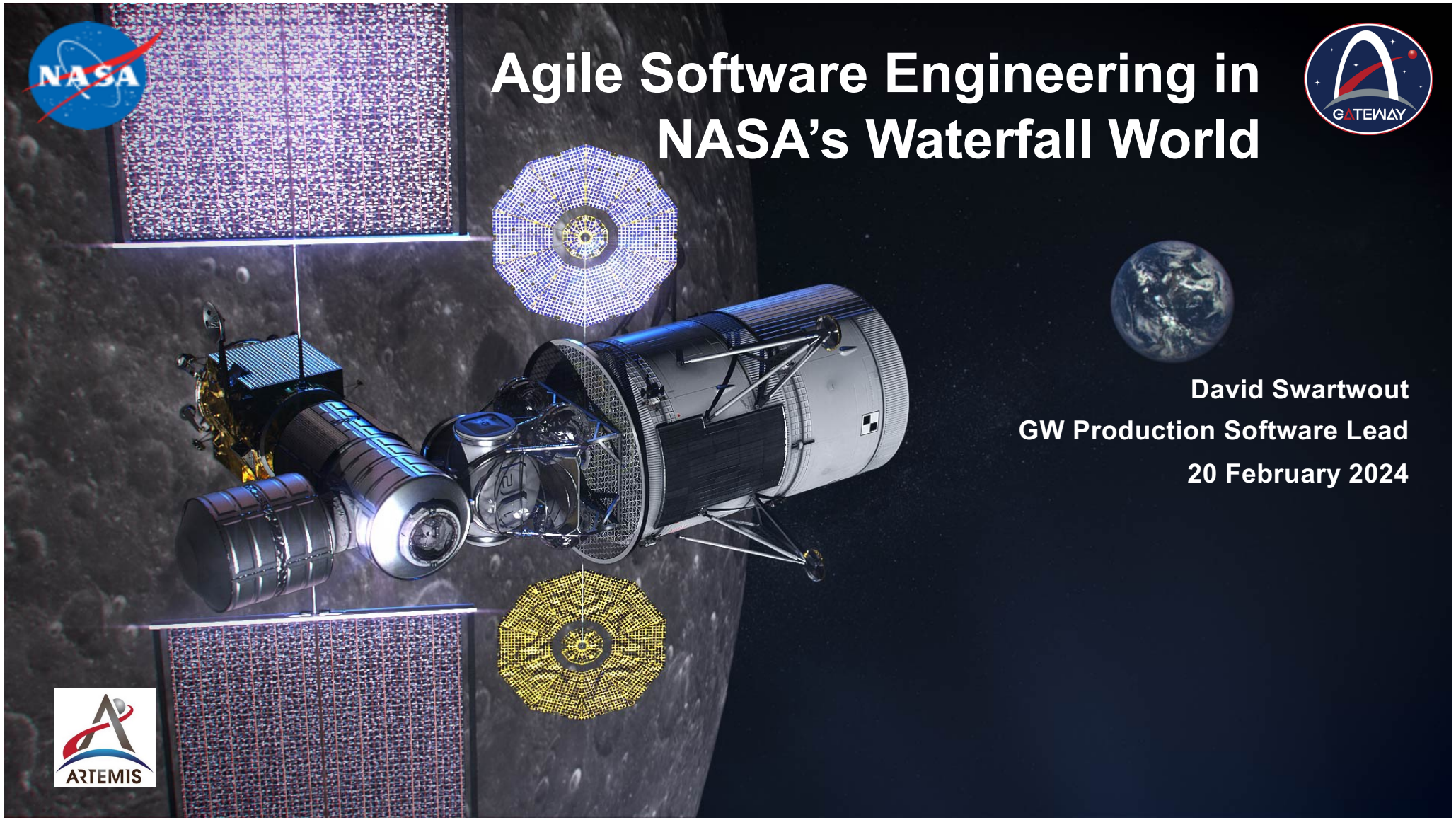


Agile Software Engineering in NASA's Waterfall World

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)

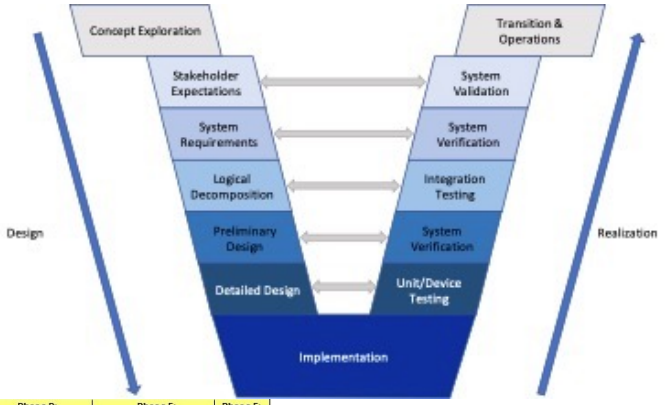
- NPR 7120.5, NASA Space Flight Program and Project Management Requirements
- NPR 7120.7, NASA Information Technology Program and Project Management Requirements
- NPR 7120.8, NASA Research and Technology Program and Project Management Requirements

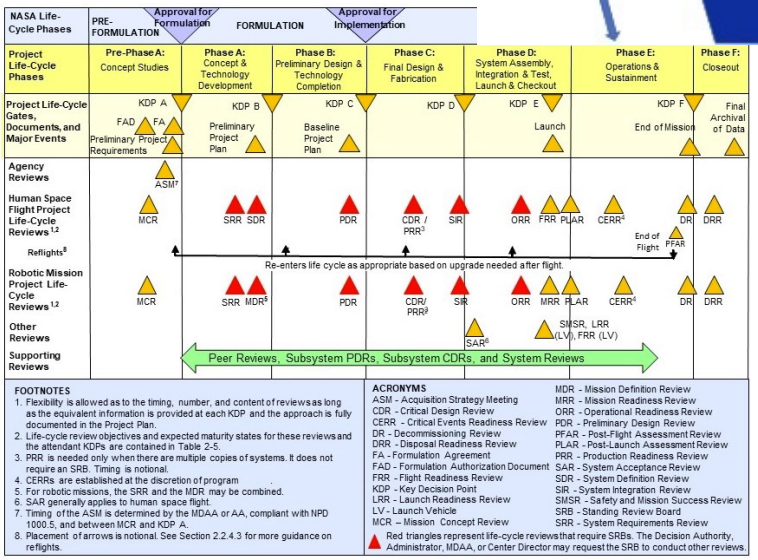
And the overarching Systems Engineering NPR

- 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'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

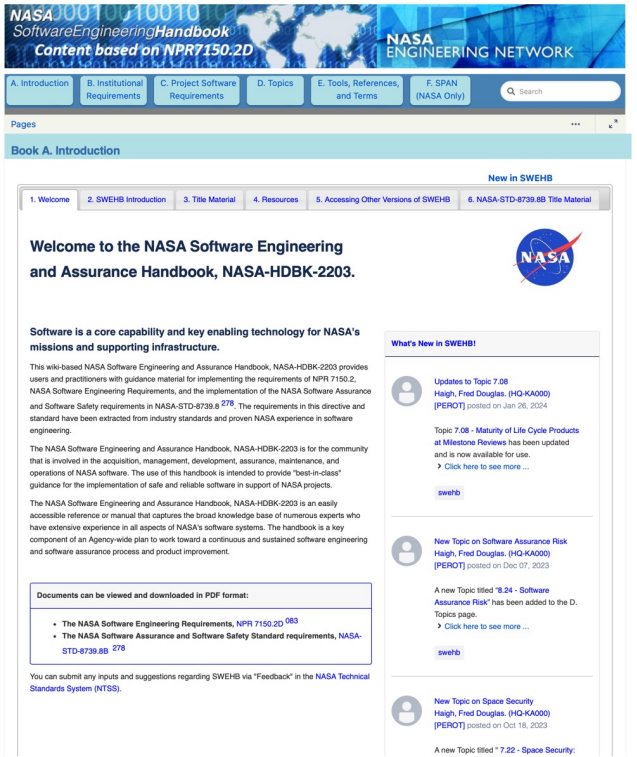
- NASA Software Engineering and Assurance Handbook, NASA-HDBK-2203

- 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.

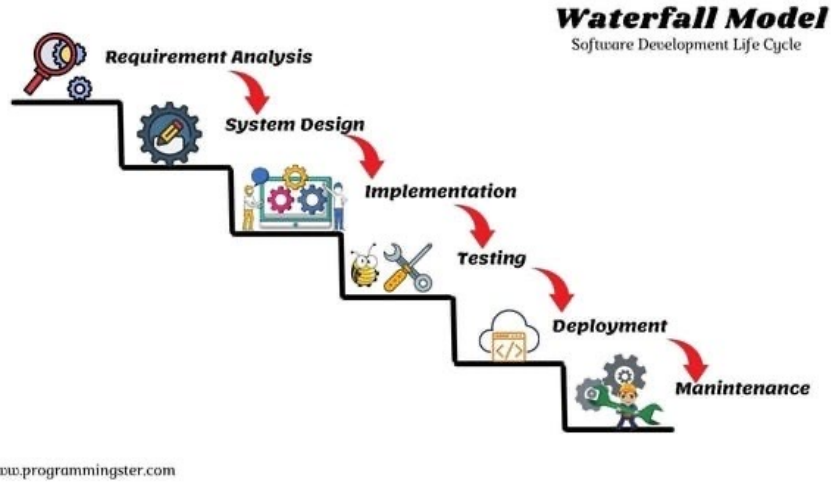
- Caution - some Centers may levy additional constraints via Center specific Procedural Requirements



The Waterfall Lifecycle

Some challenges and pitfalls of Waterfall

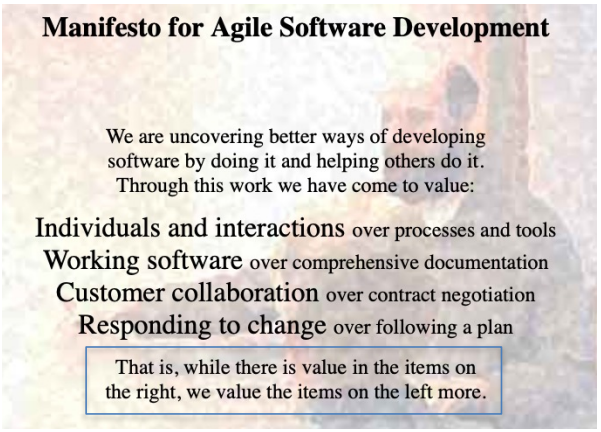
- It assumes you got it right the first time
- Going back to earlier cycles is nearly impossible
- It takes a very long time to learn what you don't know
  - Or what you didn't get right in the previous cycles
- Course corrections are hard
- Feedback is slow

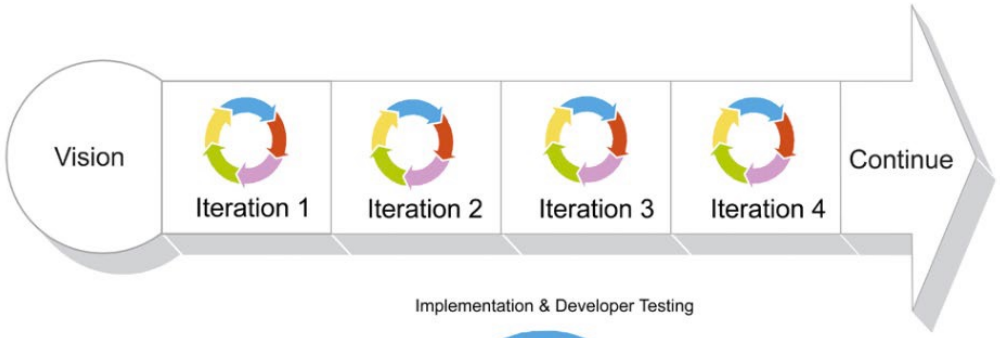


Agile

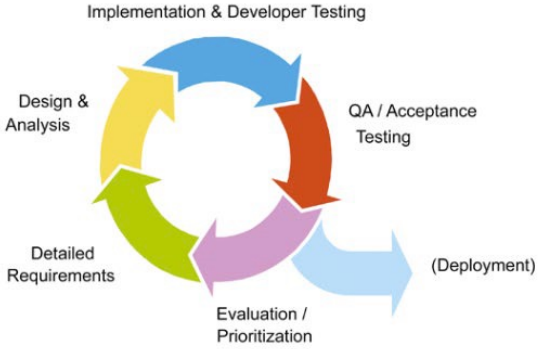
Industry experience has shown that Agile project management can be highly successful

- 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



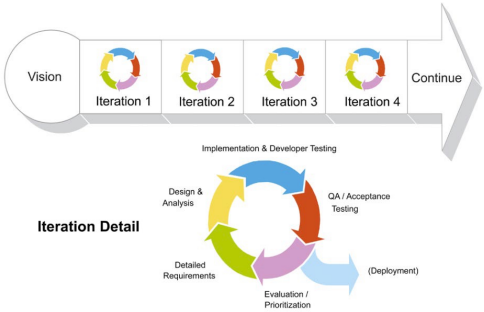


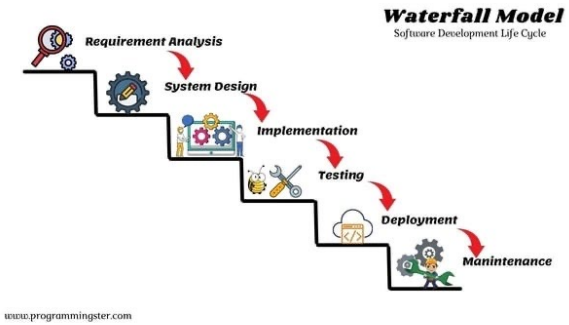
Iteration Detail



How did we bridge the gap?

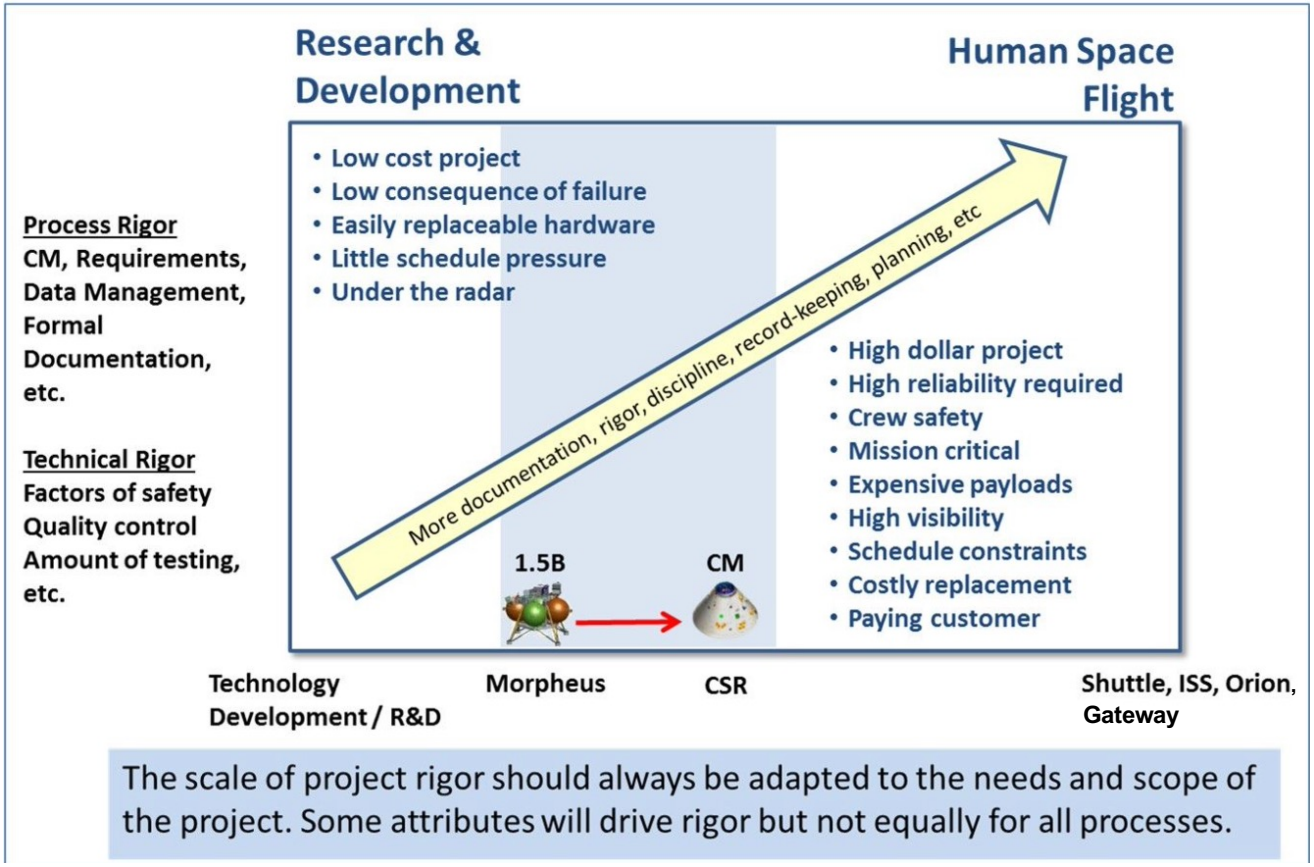
So, how did we achieve Agile software engineering within NASA's NPR dictated waterfall lifecycle?





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

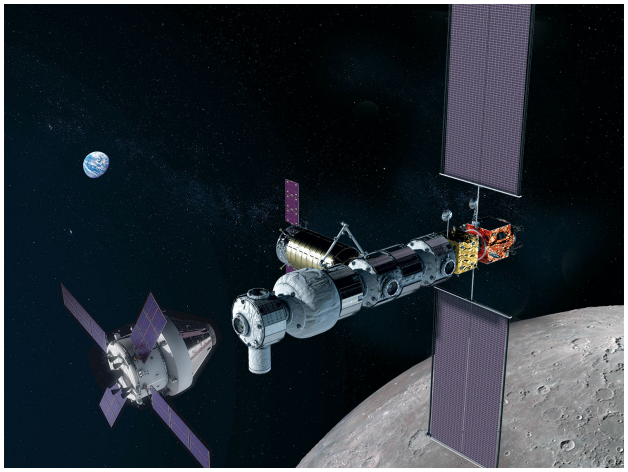
- 1st focused attempt with Agile
- Highly successful R&D project
- Class C software project
- CMMI ML 2

Orion Ascent Abort – 2 ~ 2015 - 2019

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

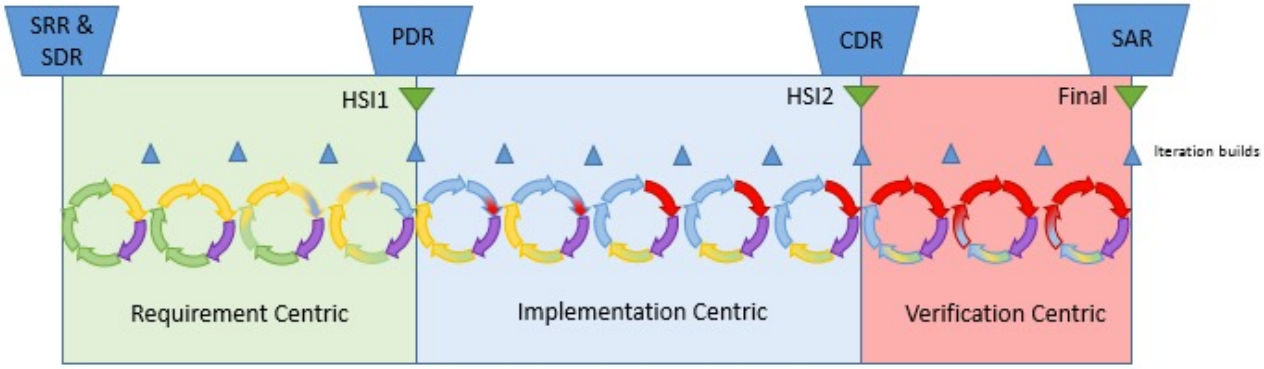
Gateway Lunar Outpost ~ 2019 - today

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



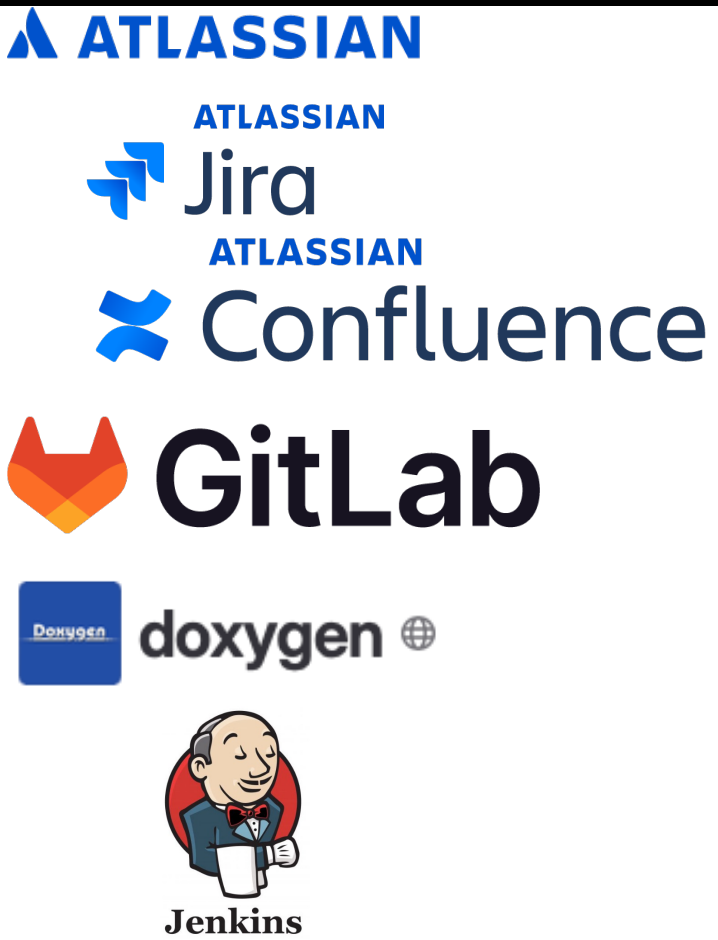
Our lifecycle focused Agile flow

- 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



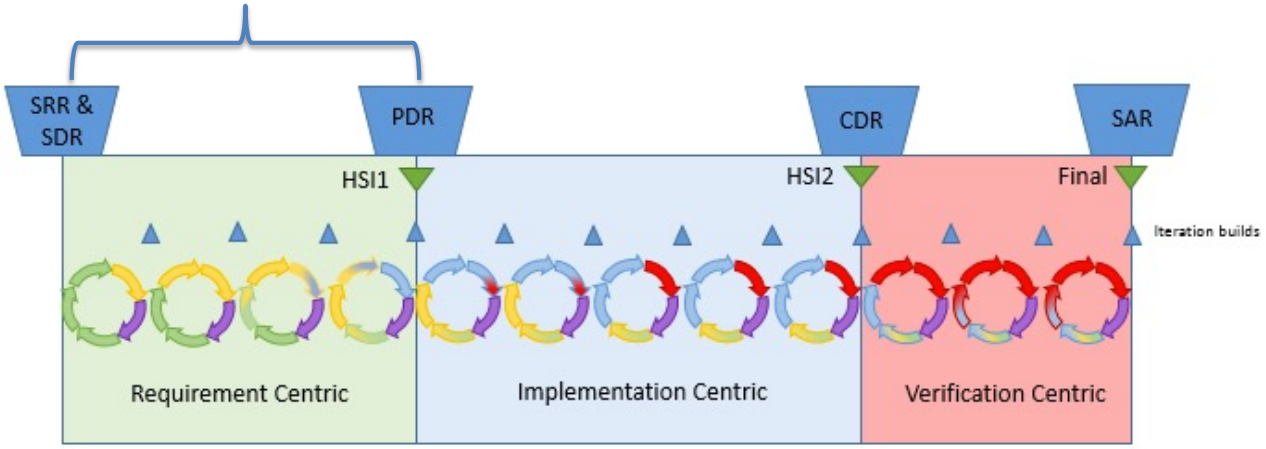
Our Solution Leverages Several Tool Ecosystems

- 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



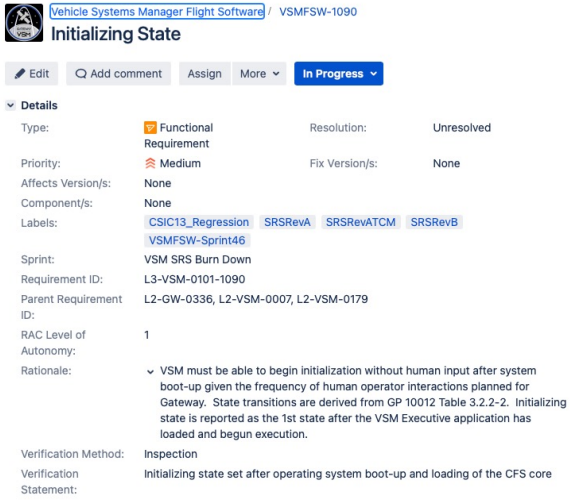
Team Focus: SRR to PDR

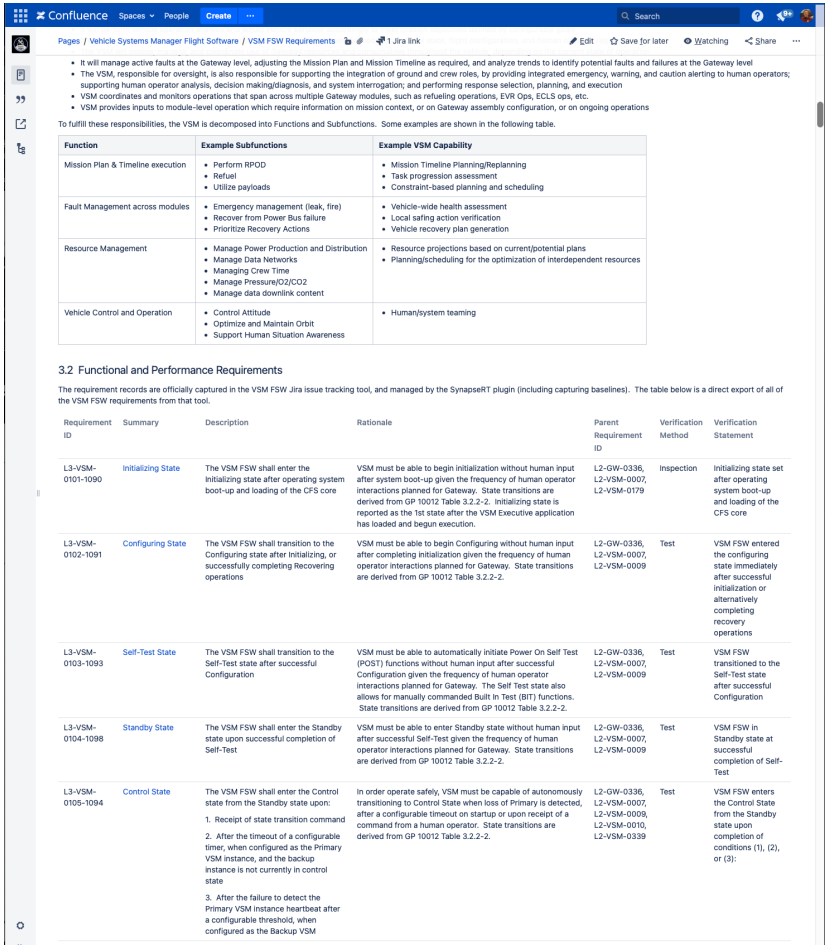
- Requirements
- Verification Planning
- Architecture
- Tool Chain
- **Use Case Based Advancement of System Capability**



Development of Software System Requirements (SRS)

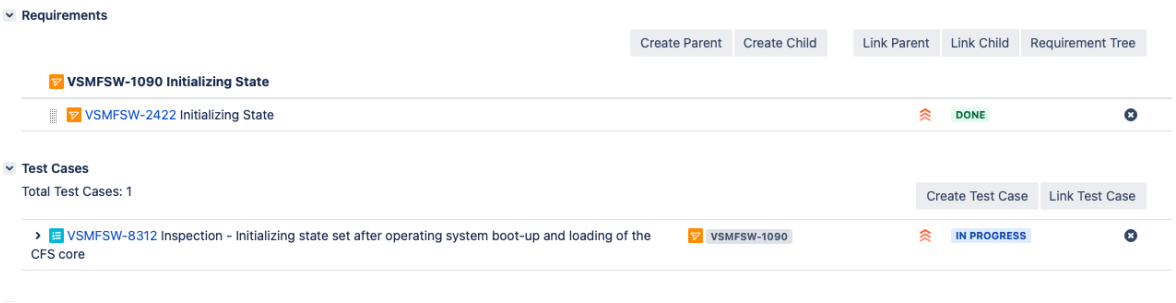
- Use con ops and operational use cases to drive requirements
  - Higher level requirements
- Computer Software Configuration Item (CSCI) requirements are defined in Jira
  - The box shall …
  - Utilizes new Jira ticket types from TestRay plugin
- Stakeholders review and comment right in the tools
  - 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

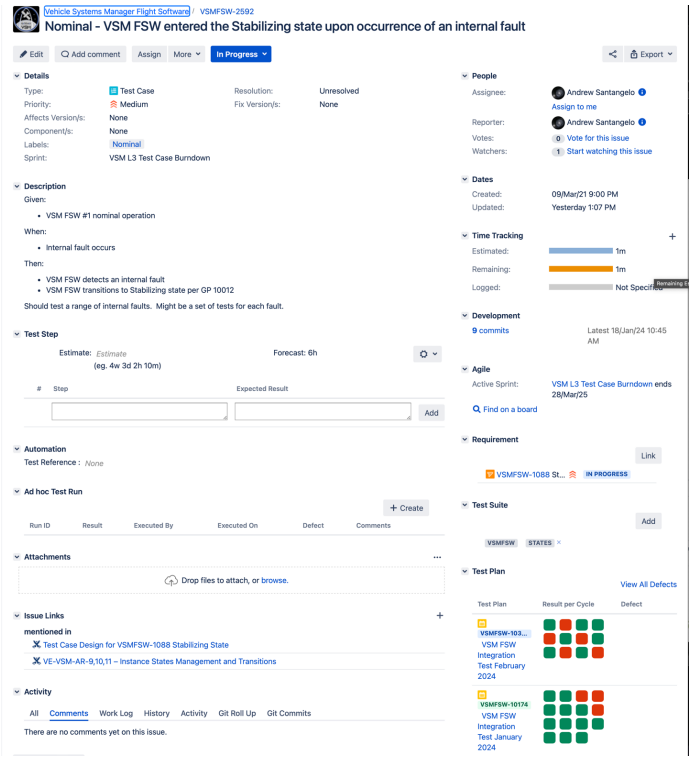




Test Planning and Development

- 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

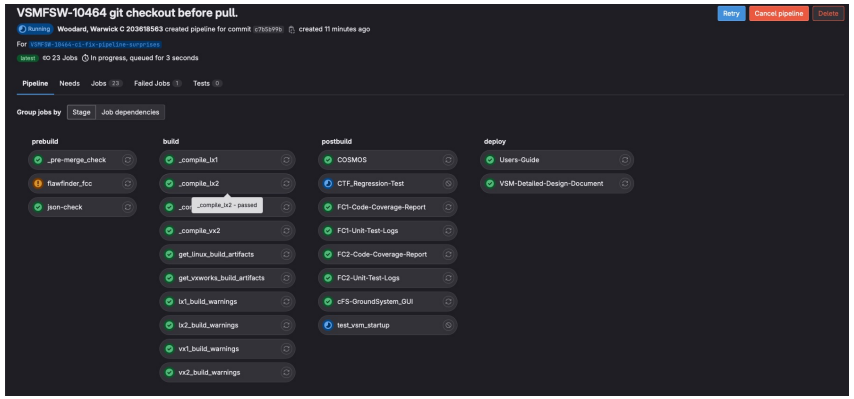


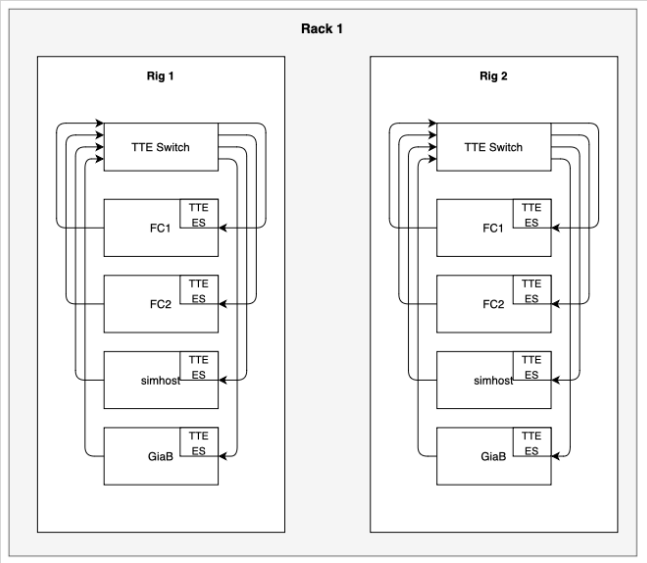


Software Architecture & Implementation

- 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)

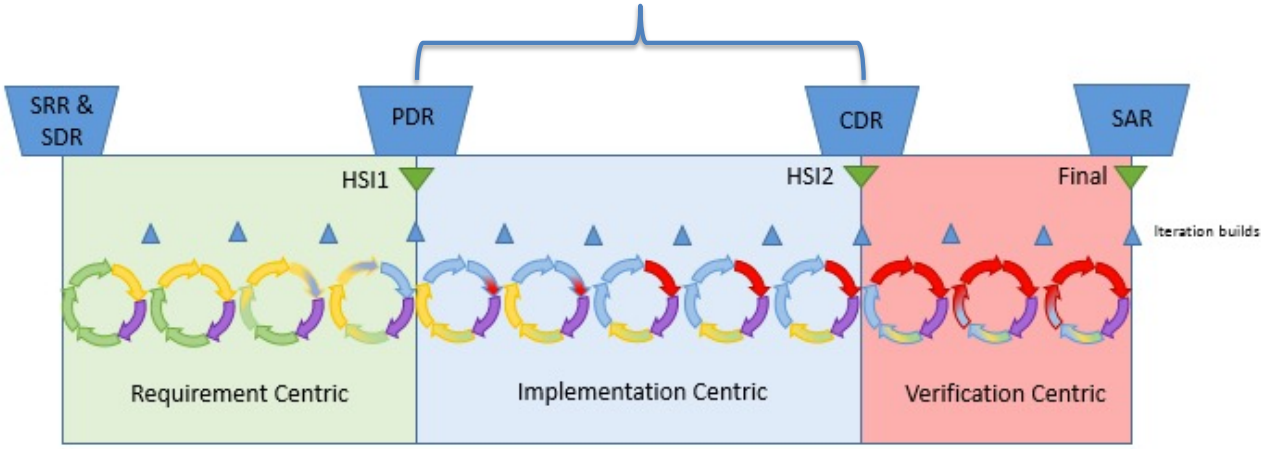
Created by Jia Lin, last modified by Eric Vineyard on Nov 08, 2023

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. **Deferred to CSIC-15 due to missing NG support.**	@Devon Debalsi @Minh Luong @Alex Lotze	~~VSMFSW-7600~~ - 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 ACSSL. Will develop CTF script to verify the use case and then call it done.	☐ CTF Tested in ACSSL ☑ Delivered to GDS ☐ 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-8013~~ - CSIC-15 SHM Exceedance Event DONE	DELIVERED One implementation story to finish in sprint 110. Estimated delivery in sprint 111. (Oct. 11-25, 2023)	☑ CTF Tested in ACSSL ☑ Delivered to GDS ☐ Integrated and Tested in GSVL
CSIC-14 UC: Payloads and STRM Management	This page captures the use case of CSIC-14 for GPFS management. **Deferred 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-8320~~ - CSIC-14 Payloads and STRM GPFS Management DONE ~~VSMFSW-8178~~ - CSIC-14 Payloads and STRM GPFS Management UC DONE ~~VSMFSW-8321~~ - 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 e7572d5bfb9f4f0f6fce255f88a844281c44502a. 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 TREX task instead of lightweight tasks. Waiting for L2 confirmation and delivery of the payload TREX tasks.	☑ CTF Tested in ACSSL ☑ Delivered to GDS ☐ 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_Rsc_Data_t" of "PSM_Payload_RsrcTlmMsg_t" tele feeder of "Payload_HERMES_SORL constraint checking for lightweight 1664 • The LSB byte of the "usFreshCntr" #1) in Payload Health State and Stat as the feeder of "Payload_HERMES_Science_Instru constraint checking for lightweight 1670

Team Focus: PDR to CDR

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



Application Requirements Decomposition

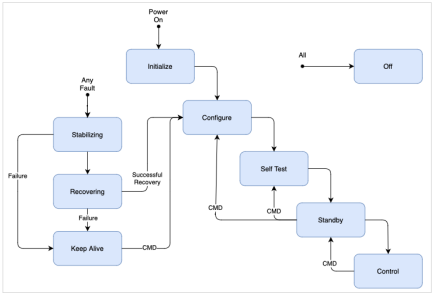
- Structured design process to decompose box level CSCI requirements into multiple application-level computer software component (CSC) requirements
  - Requirements based use cases drive system capabilities
  - 2nd tier requirements are also captured in Jira
  - Designs are documented via organized confluence 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:

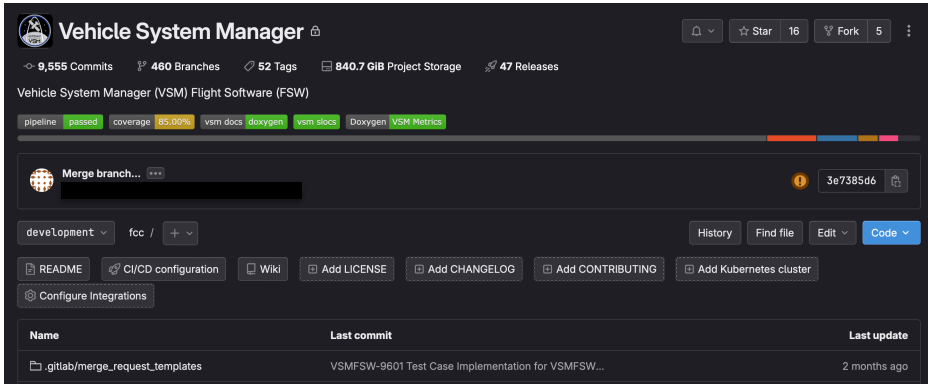


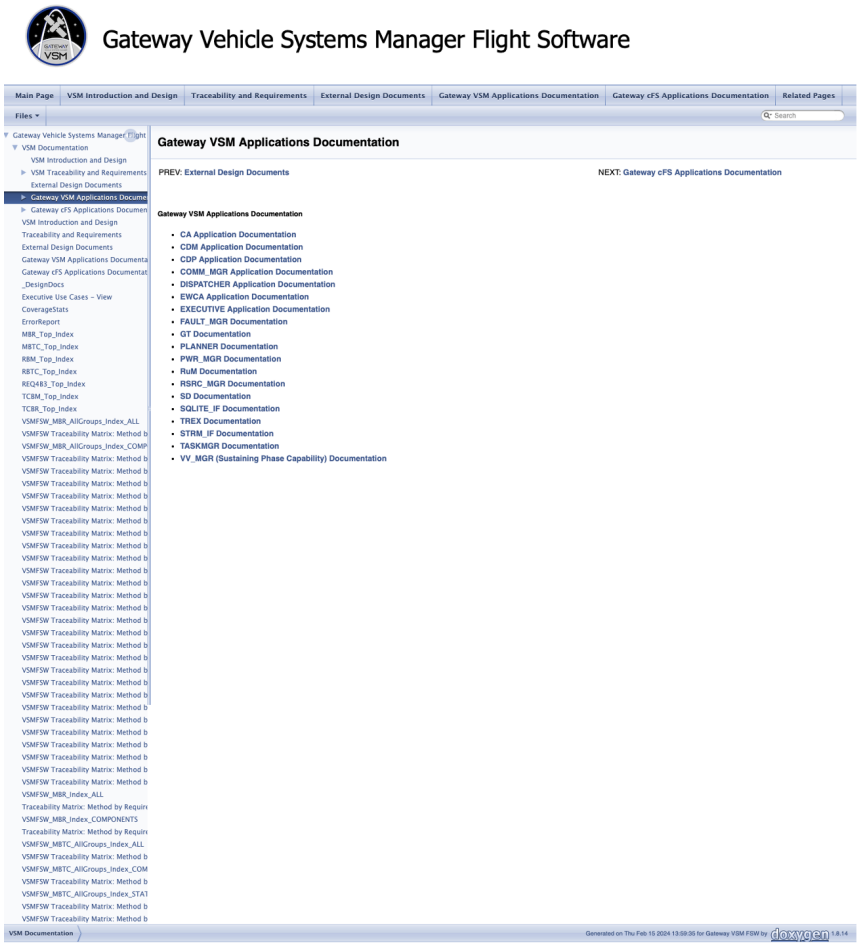
1.2 Power On to Initiailizing Use Case

Use Case Name	Initializing	
Participants	Executive	
Assumptions	None	
Pre-conditions	None	
Scenario	Description	Requirement Tracing
1	sp0 is powered on	
2	VxWorks 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

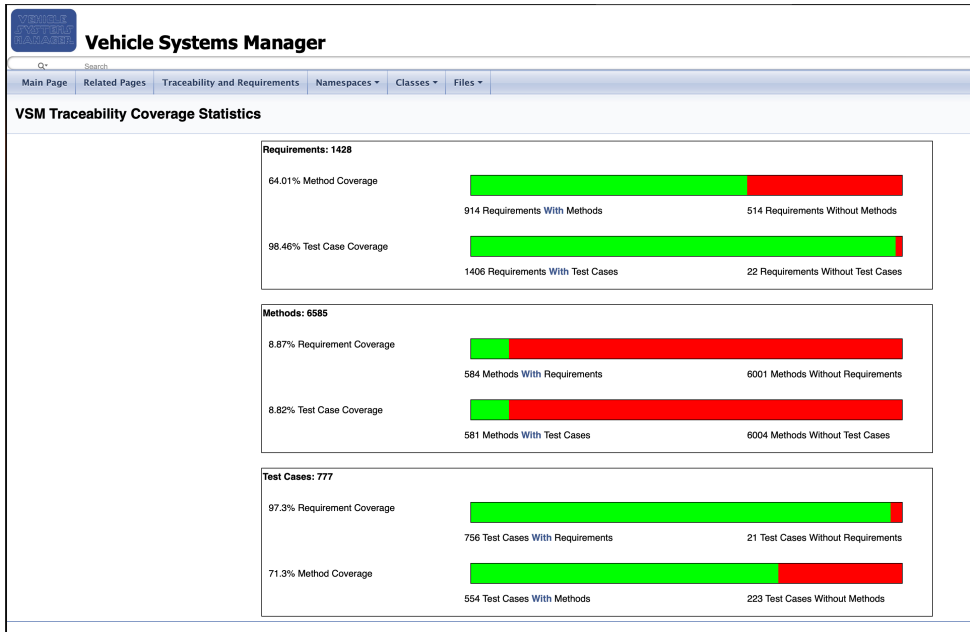
- All design artifacts are “organic” products that are created inline as part of the development process
  - We don’t stop what we are doing to make a big Word document and a bunch of PowerPoint slides
- Utilize Doxygen, Confluence exports, and Jira scripting to create the design package
  - HTML based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

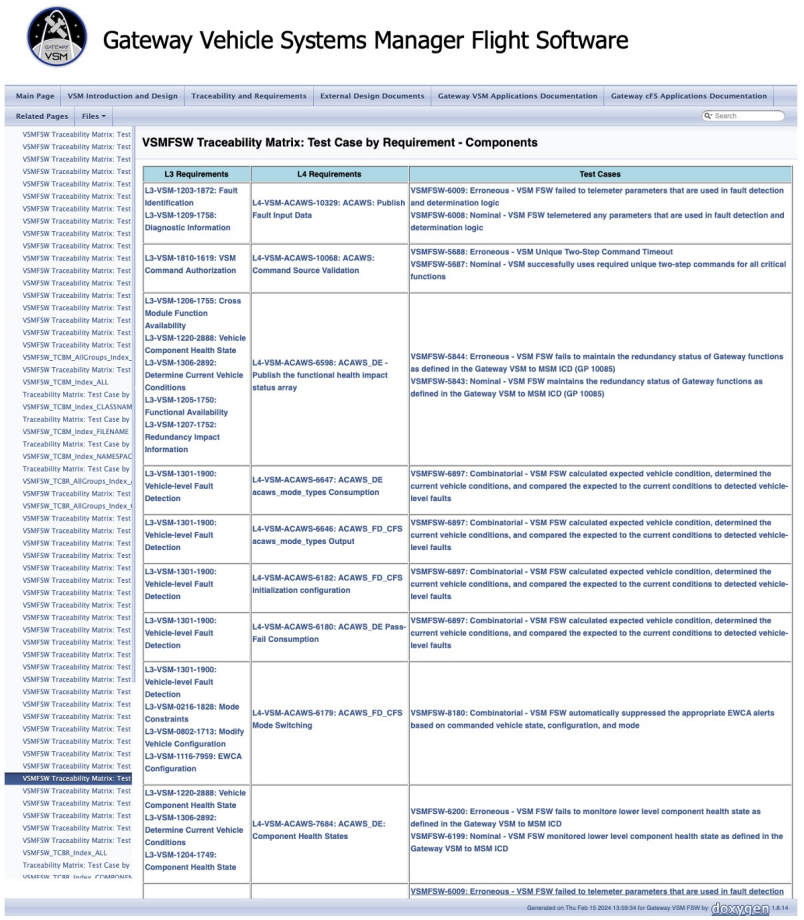




Trace Tables

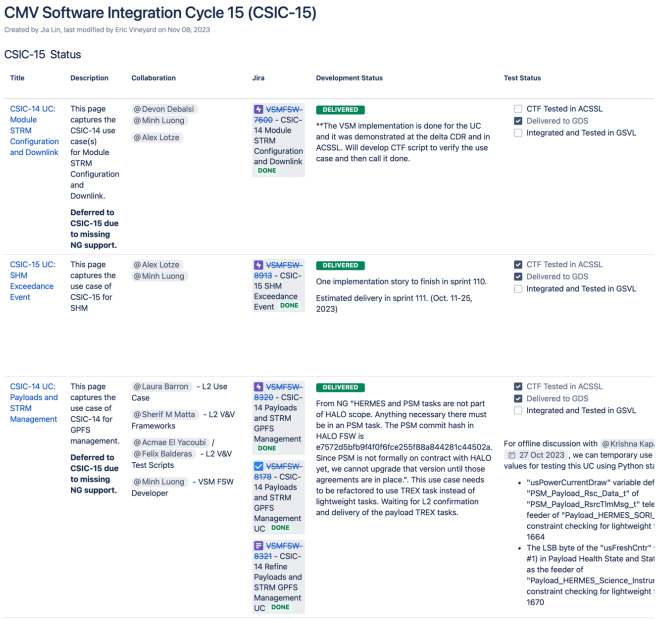
- NPR 7150.2 requires requirements and verifications traceability
  - Depending on software classification
- Utilizing custom Doxygen tags in the source code, along with Jira scripting we can generate tables for bi-directional traceability
  - Requirement <–> Design <–> Code
  - Requirement <–> Code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





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)

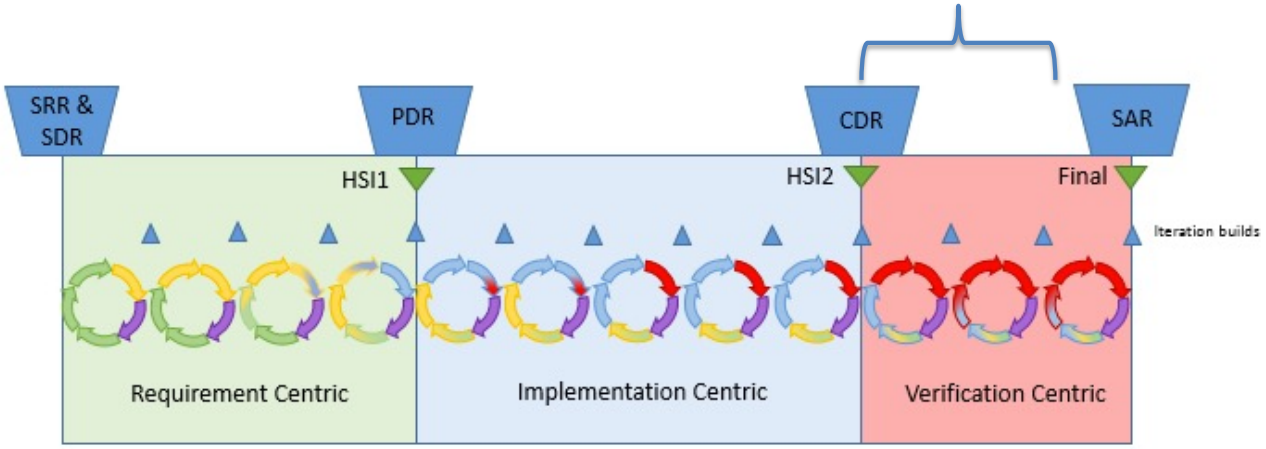
Created by Jia Lin, last modified by Eric Vineyard on Nov 08, 2023

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. **Deferred to CSIC-15 due to missing NG support.**	@Devon Debalsi @Minh Luong @Alex Lotze	~~VSMFSW-7600~~ - 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 ACSSL. Will develop CTF script to verify the use case and then call it done.	☐ CTF Tested in ACSSL ☑ Delivered to GDS ☐ 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-8913~~ - CSIC-15 SHM Exceedance Event DONE	DELIVERED One implementation story to finish in sprint 110. Estimated delivery in sprint 111. (Oct. 11-25, 2023)	☑ CTF Tested in ACSSL ☑ Delivered to GDS ☐ Integrated and Tested in GSVL
CSIC-14 UC: Payloads and STRM Management	This page captures the use case of CSIC-14 for GPFS management. **Deferred 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-8320~~ - CSIC-14 Payloads and STRM GPFS Management DONE ☑ ~~VSMFSW-8178~~ - CSIC-14 Payloads and STRM GPFS Management UC DONE ~~VSMFSW-8321~~ - 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 e7572d5bfb9f4f0f6fce255f88a844281c44502a. 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 TREX task instead of lightweight tasks. Waiting for L2 confirmation and delivery of the payload TREX tasks.	☑ CTF Tested in ACSSL ☑ Delivered to GDS ☐ 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_Rsc_Data_t" of "PSM_Payload_RsrcTlmMsg_t" tele feeder of "Payload_HERMES_SORL constraint checking for lightweight 1664 • The LSB byte of the "usFreshCntr" #1) in Payload Health State and Stat as the feeder of "Payload_HERMES_Science_Instru constraint checking for lightweight 1670

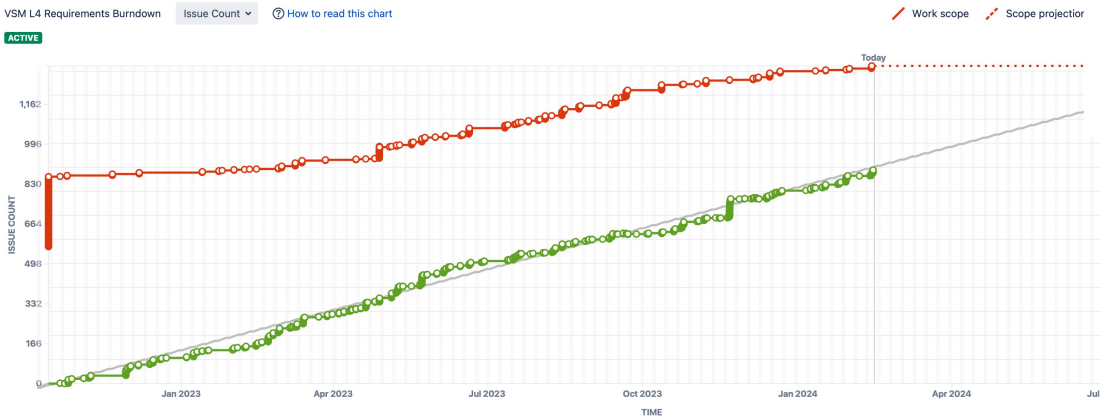
Team Focus: CDR to TRR

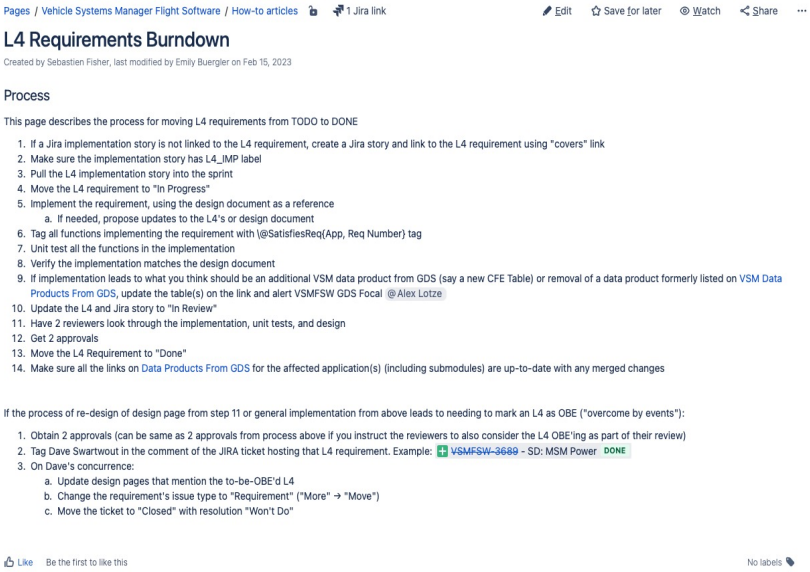
- Final Implementation of Design
- Test Script Development
- Continuous Integration Pipeline
- **Use Case Based Advancement of System Capability**



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 Sebastien Fisher, last modified by Emily Buergler 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

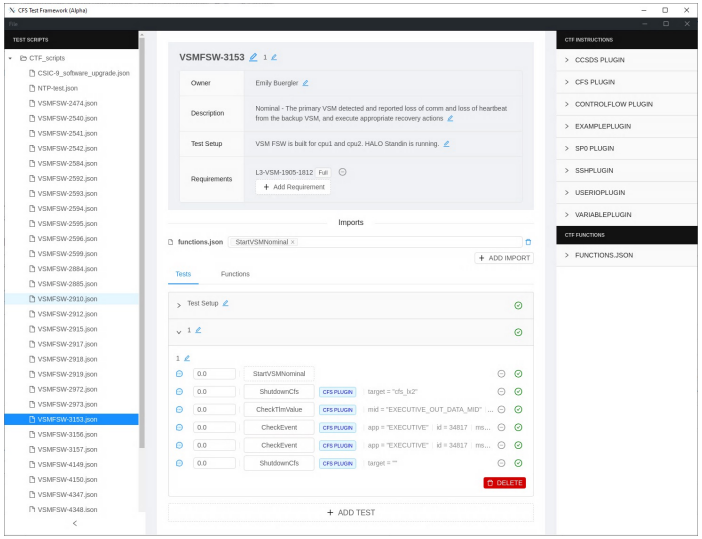
If the process of re-design of design page from step 11 or general implementation from above leads to needing to mark an L4 as OBE ("overcome by events"):

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

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 Designs — 10 Jira links — Edit — Save for later — Watch — Share

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

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

Overview

L3 Requirement	Test Plan	Test Cases	Test Development Stories
Transitioning from Control to Standby VSMFSW-1118 - Transitioning from Control to Standby IN PROGRESS	Preliminary-I6, Set 6 ~~VSMFSW-4330~~ - Development-I6, 1118 DONE ~~VSMFSW-10293~~ - Regression Test Review - 1118 DONE	VSMFSW-2596 - Nominal - Primary VSM transitioned from Control to Standby upon receipt of a state transition command IN PROGRESS VSMFSW-2599 - Erroneous - Backup VSM does not transition from Standby to Control after receiving a state transition command IN PROGRESS 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-4348 - Erroneous - Primary VSM does not transition from Control to Standby due Backup VSM unresponsive upon receipt of a state transition command IN PROGRESS	~~VSMFSW-4331~~ - Develop CTF Test Plan for Transitioning from Control to Standby DONE ~~VSMFSW-7813~~ - Investigate regression test failure for VSMFSW-1118 Requirement DONE ~~VSMFSW-10213~~ - Debug VSMFSW-1118 CTF possible regression failure DONE

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

Table of Contents

Acronyms

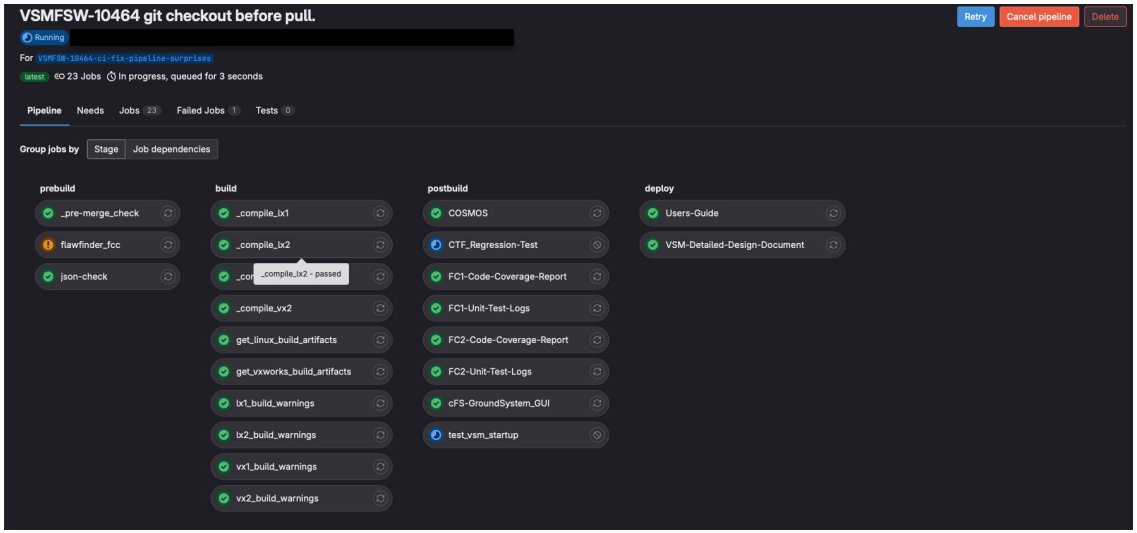
ASM = Autonomous System Management

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)



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

CMV Software Integration Cycle 15 (CSIC-15)

Created by Jia Lin, last modified by Eric Vineyard on Nov 08, 2023

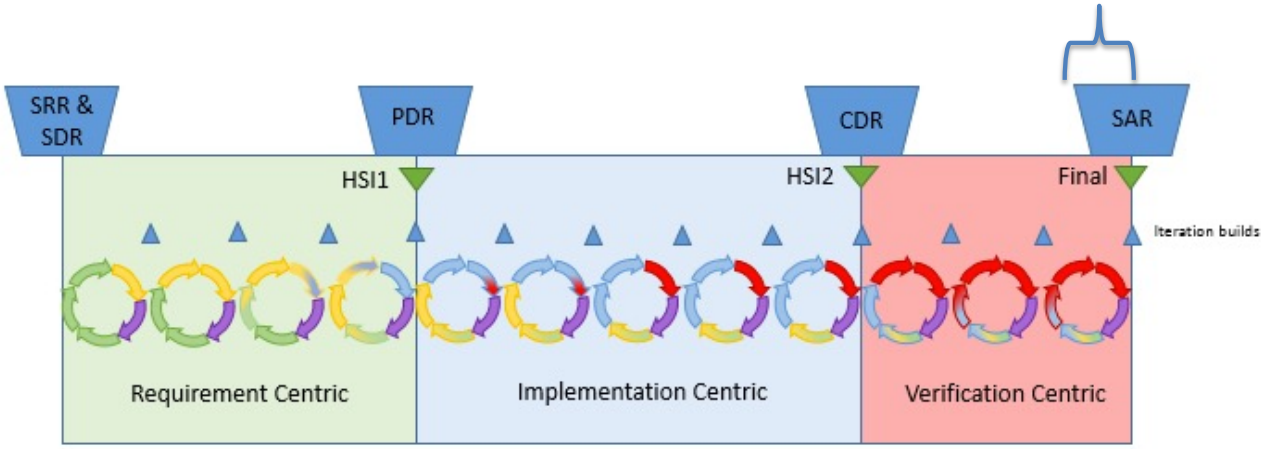
CSIC-15 Status

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. **Deferred to CSIC-15 due to missing NG support.**	@Devon Debalsi @Minh Luong @Alex Lotze	VSMFSW-7600 - 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 ACSSL. Will develop CTF script to verify the use case and then call it done.	☐ CTF Tested in ACSSL ☑ Delivered to GDS ☐ 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-8913 - CSIC-15 SHM Exceedance Event DONE	DELIVERED One implementation story to finish in sprint 110. Estimated delivery in sprint 111. (Oct. 11-25, 2023)	☑ CTF Tested in ACSSL ☑ Delivered to GDS ☐ Integrated and Tested in GSVL
CSIC-14 UC: Payloads and STRM Management	This page captures the use case of CSIC-14 for GPFS management. **Deferred 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-8320 - CSIC-14 Payloads and STRM GPFS Management DONE VSMFSW-8178 - CSIC-14 Payloads and STRM GPFS Management UC DONE VSMFSW-8321 - 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 e7572d5bfb9f4f0f6fce255f88a844281c44502a. 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 TREX task instead of lightweight tasks. Waiting for L2 confirmation and delivery of the payload TREX tasks.	☑ CTF Tested in ACSSL ☑ Delivered to GDS ☐ 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_Rsc_Data_t" of "PSM_Payload_RsrcTlmMsg_t" tele feeder of "Payload_HERMES_SORL constraint checking for lightweight 1664 • The LSB byte of the "usFreshCntr" #1) in Payload Health State and Stat as the feeder of "Payload_HERMES_Science_Instru constraint checking for lightweight 1670

- Continuous Improvement of Tools and Processes
  - Enhancing CI pipeline
  - Development of emulators and simulators
  - Lessons learned from prior phases
  - Documenting process enhancements
  - Updating templates

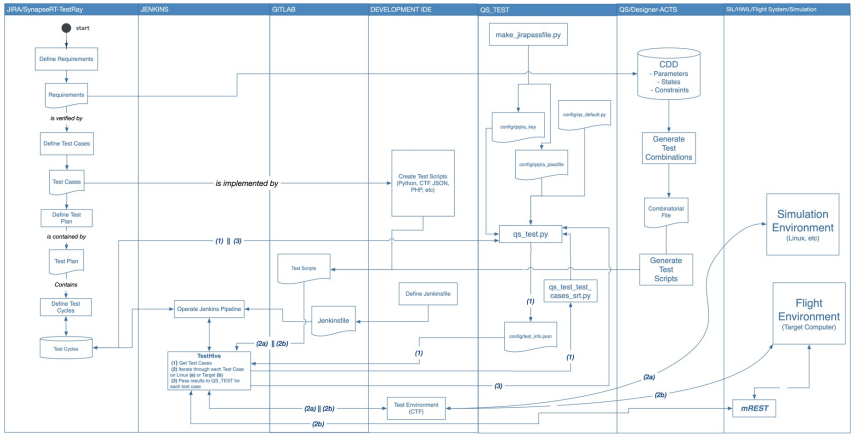
Team Focus: TRR to SAR

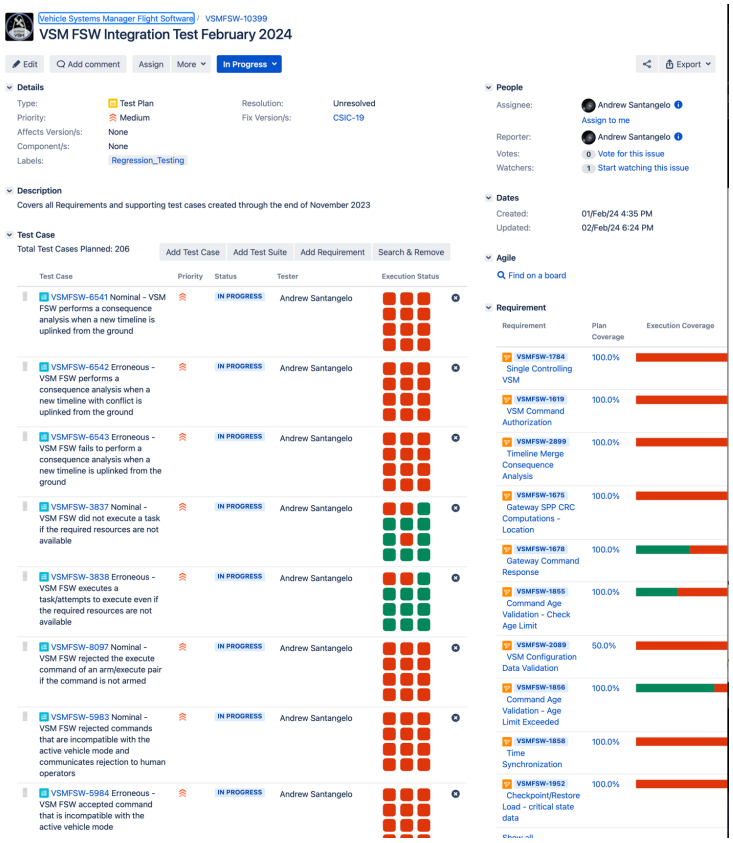
- Formal Test and Verification



Automated Testing

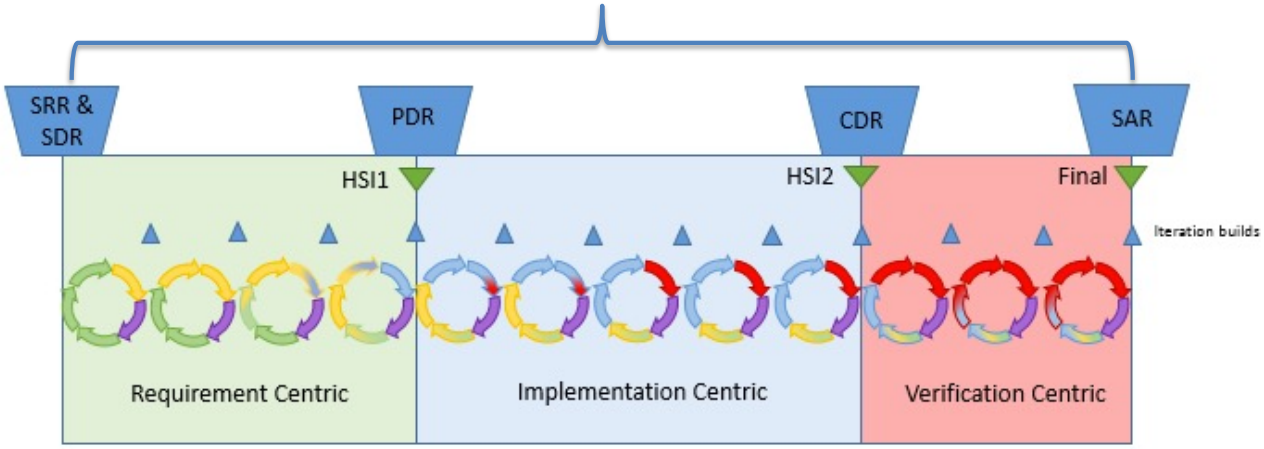
- Created around Jira/TestRay and Jira API scripting
- TestRay allows creation of Test Plans
  - Grouping of test cases into test activities
  - Plans can be executed in batch
- Jira back end communicates with Jenkins server for automated test execution
  - Test cycles executed
    - Development servers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





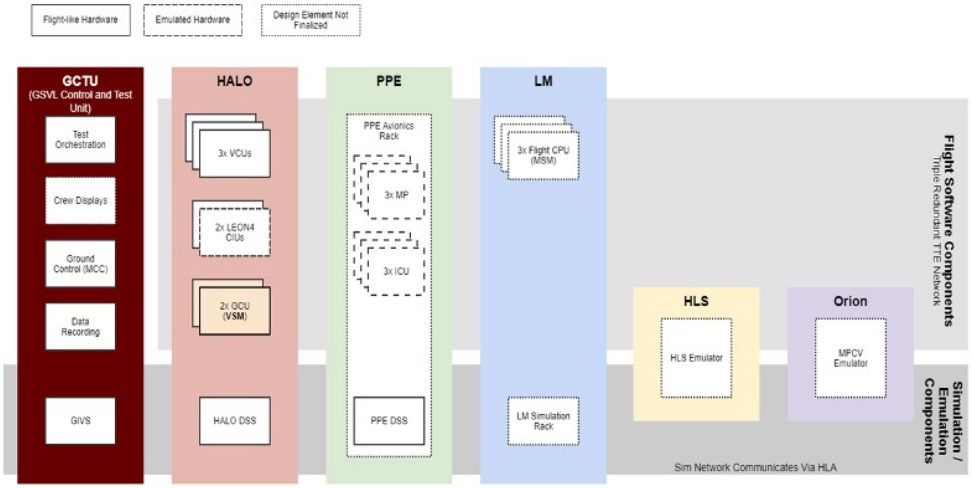
Across the Lifecycle

- Cross Team Integration

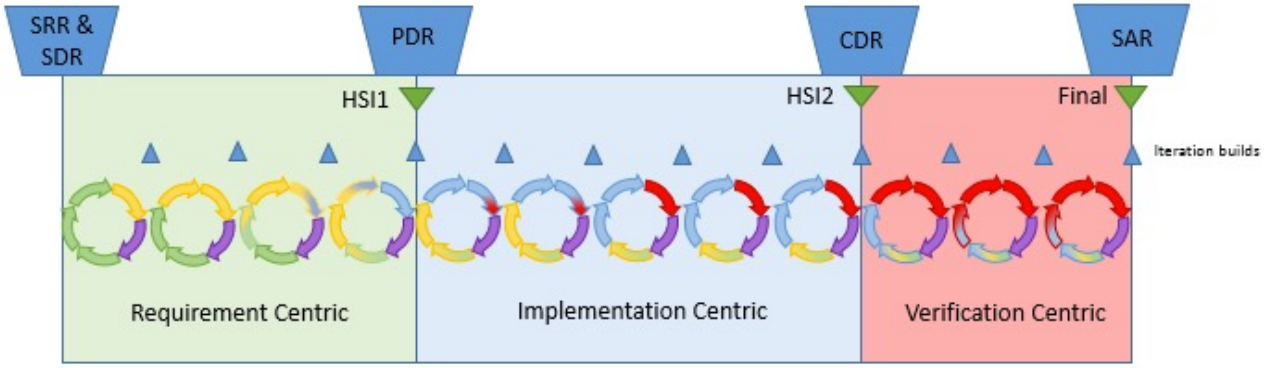


Program Wide Integration Cycles

- 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


SRR &
SDR
PDR
CDR
SAR
HSI1
HSI2
Final
Iteration builds
Requirement Centric
Implementation Centric
Verification Centric

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