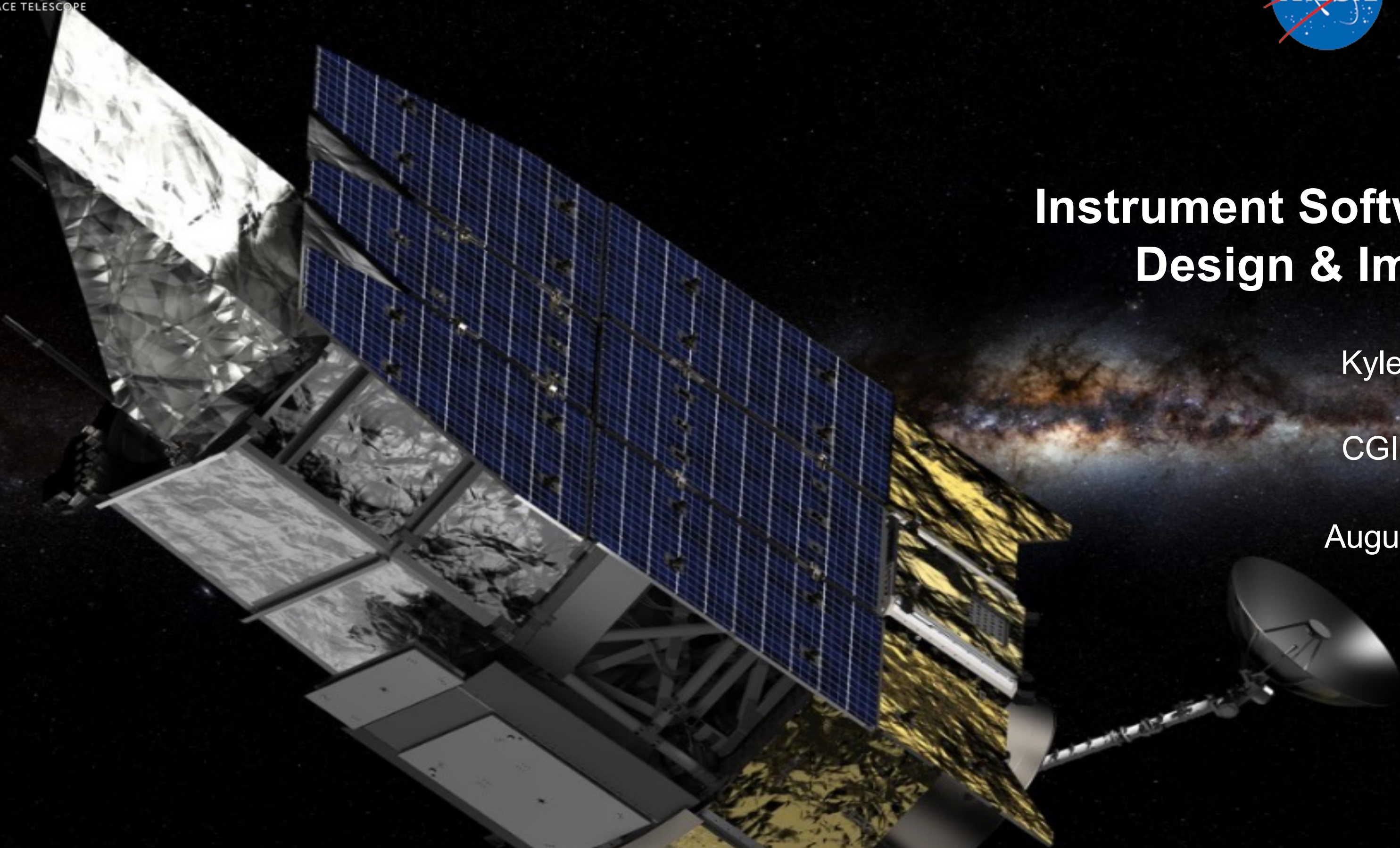
ROMAN SPACE TELESCOPE

Jet Propulsion Laboratory
California Institute of Technology



# Instrument Software Architecture: Design & Implementation

Kylee Fluckiger

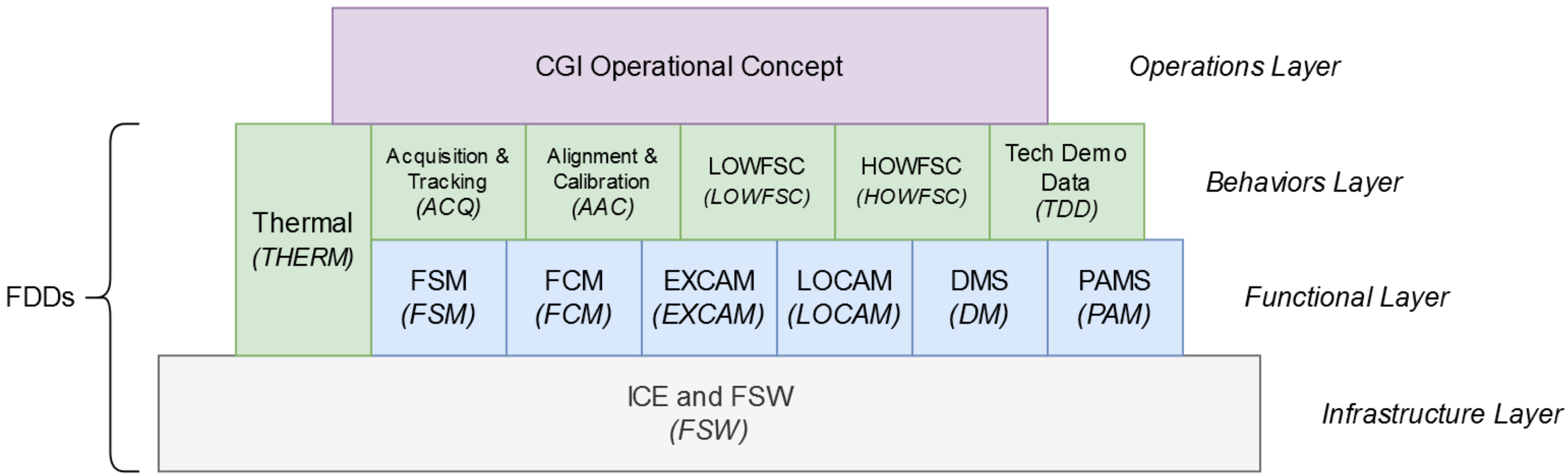
CGI FSW Lead

August 27th, 2024

• NASA GODDARD SPACE FLIGHT CENTER • JET PROPULSION LABORATORY •
• L3HARRIS TECHNOLOGIES • BALL AEROSPACE • TELEDYNE • NASA KENNEDY SPACE CENTER •
• SPACE TELESCOPE SCIENCE INSTITUTE • INFRARED PROCESSING AND ANALYSIS CENTER •
• EUROPEAN SPACE AGENCY • JAPAN AEROSPACE EXPLORATION AGENCY •
• CENTRE NATIONAL d'ÉTUDES SPATIALES • MAX PLANCK INSTITUTE FOR ASTRONOMY •

# CGI System Architecture

- Coronagraph System Architecture is divided into 13 functional domains, the expected behavior of each of which is described in a functional design document (FDD) that is used to guide FSW implementation.



# CGI FSW Functional Capabilities

- Channel
  - Telemetry data point
  - Mapped to single source
  - Ex: temperature, parameter value, mode state, calculated offset
- Monitor
  - Condition for fault protection
  - Ex: trigger if voltage exceeds some value for 2 or more seconds
- Parameter
  - Nonvolatile value stored onboard
  - Modifiable via command
  - Ex: pixel threshold, gain factor, min/max values
- Command
  - Input to FSW to initiate some behavior
  - Ex: power hardware, perform move, close Zernike loop, halt image taking

# CGI FSW Complexity

| Functional Description Document Domain | # Req's | | # Cmds | # Params | # Channels | | # Monitors |
|---|---|---|---|---|---|---|---|
| | FSW | GND | | | FSW | GND | |
| ICE and FSW Infrastructure | 104 | -- | 30 | -- | 371 | 156 | 3 |
| FSM | 45 | -- | 21 | 22 | 54 | 24 | 12 |
| FCM | 26 | -- | 11 | 21 | 46 | 38 | 19 |
| EXCAM | 49 | -- | 17 | 28 | 174 | 93 | 27 |
| LOCAM | 38 | -- | 10 | 11 | 153 | 67 | 14 |
| DM | 58 | -- | 18 | 14 | 90 | 56 | 28 |
| PAM | 45 | -- | 14 | 232 | 443 | 158 | 4 |
| Thermal | 28 | -- | 1 | 295 | 450 | 142 | 71 |
| Acquisition & Tracking | 38 | -- | 6 | 71 | 130 | 3 | -- |
| Alignment & Calibration | 24 | 100 | 7 | 65 | 65 | 3 | -- |
| LOWFSC | 75 | 2 | 20 | 20 | 88 | 40 | 20 |
| HOWFSC | 18 | 19 | 1 | 83 | 92 | -- | -- |
| Tech Demo Data | 5 | 14 | 0 | 8 | 8 | -- | -- |
| **TOTALS** | **553** | **135** | **165** | **883** | **2,179** | **810** | **198** |
| | **688** | | | | **2,989** | | |

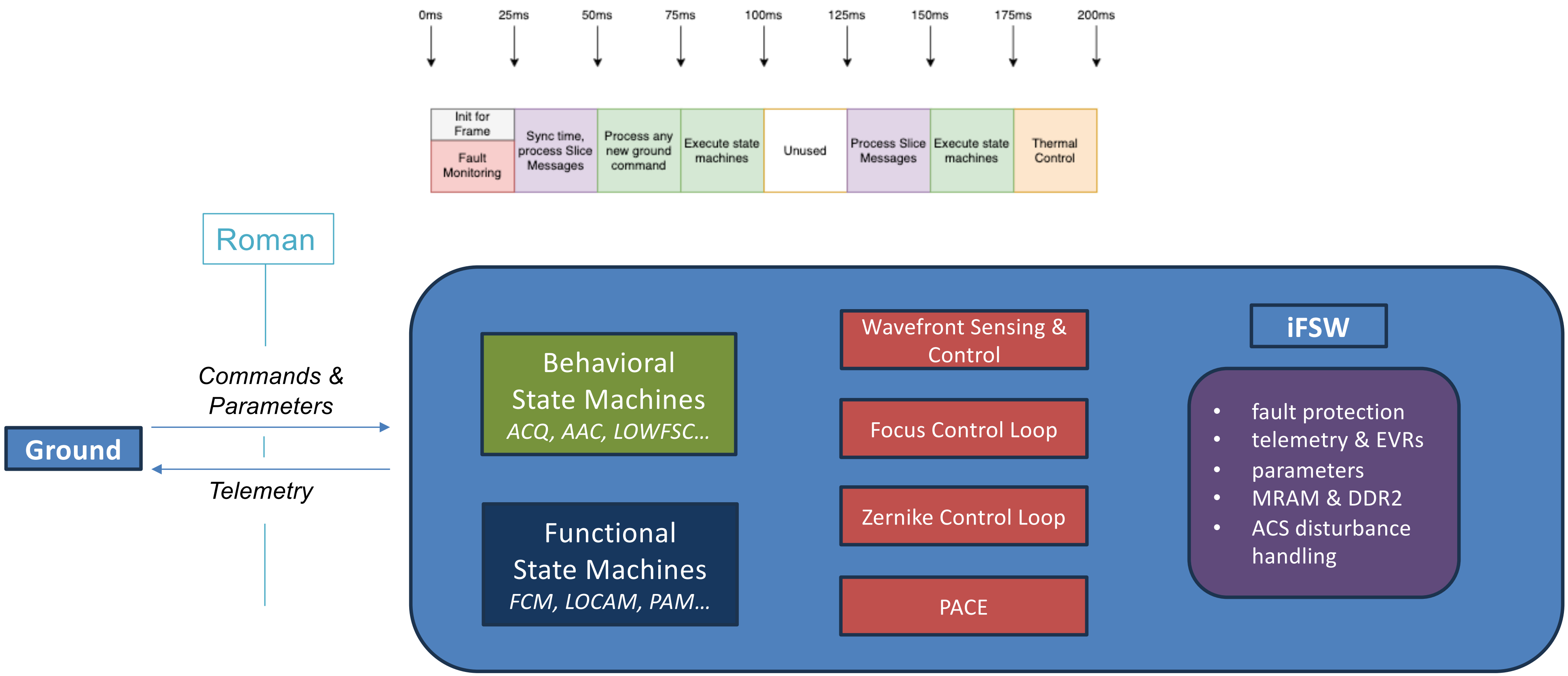
- High instrument code complexity
  - **CGI FSW: 104,557 SLOC**
  - typical: 30,000 – 70,000 SLOC
  - *GSW: 107,000 SLOC*
  - *FPGA: 60,000 SLOC*
- Embedded linear algebra required
  - modified Eigen template library [1]
- Code reuse
  - FP, telemetry, slice interfaces, parameter table
- Command and telemetry dictionary management
- Sequence engine autocoding for PAM
- Camera class inheritance for EXCAM and LOCAM

# CGI FSW Overview

- C++ language
- VxWorks real-time operating system (RTOS)
- LEON4 processor on GR740 board
  - radiation-hardened
- Single-core software design
  - additional cores bring additional complexity
- Developed over 5+ years

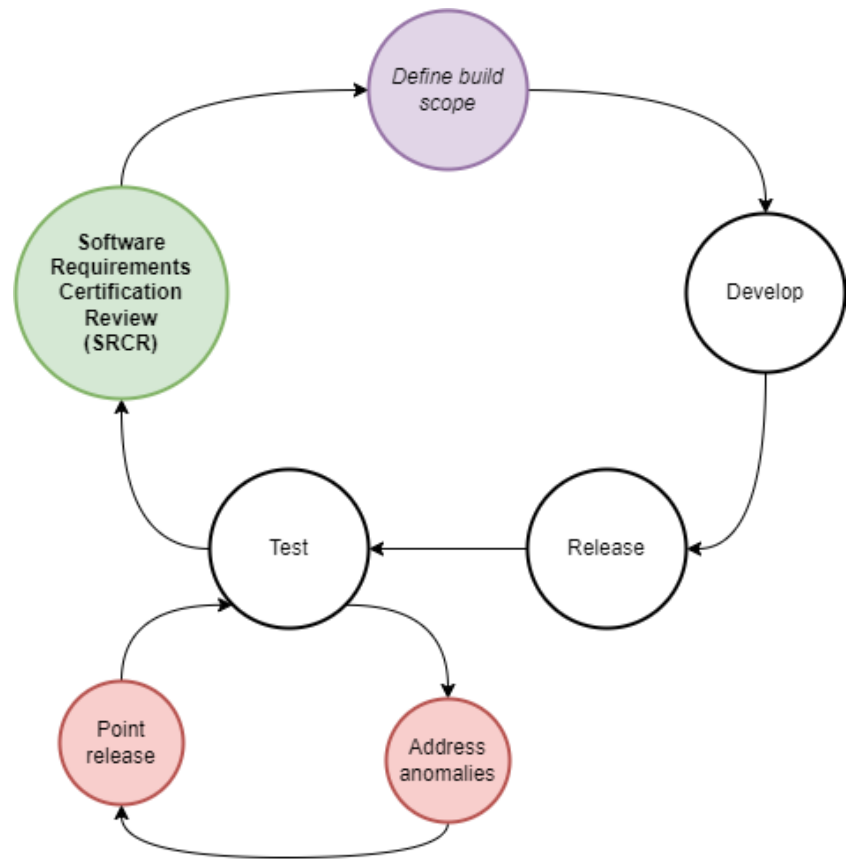
- NASA class C software
  - "necessary for the science return from a single (non-primary) instrument"
- Complies with strict JPL FSW coding standards
  - based on automotive MISRA C
  - increases safety and reliability of code
  - prohibits use of dynamic memory allocation and recursive functions, among others

# CGI FSW Execution Architecture



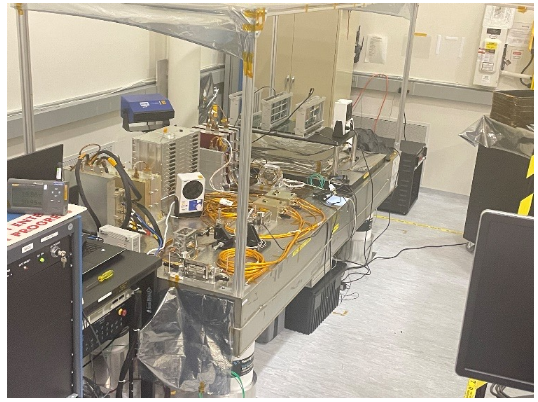
# CGI FSW Development Process

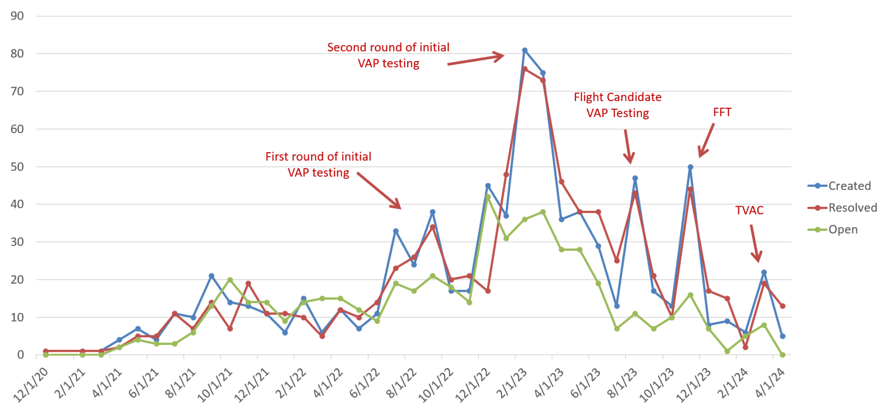
- GitHub
  - Feature branches, merge into dev branch, release on master
  - Pull requests with reviewers enforced
- Jenkins
  - Continuous integration tool
  - Simics used for hardware emulation in testing
- Static code analysis
  - Coverity
  - CodeSonar



# CGI FSW Testing

- Functional testbed (FTB) venues
  - enable both FSW development and V&V
  - unit testing
- Written test procedures (VAPs and PBATs)
  - regression testing
- Telemetry data system (TDS) and Grafana
  - efficient anomaly investigation

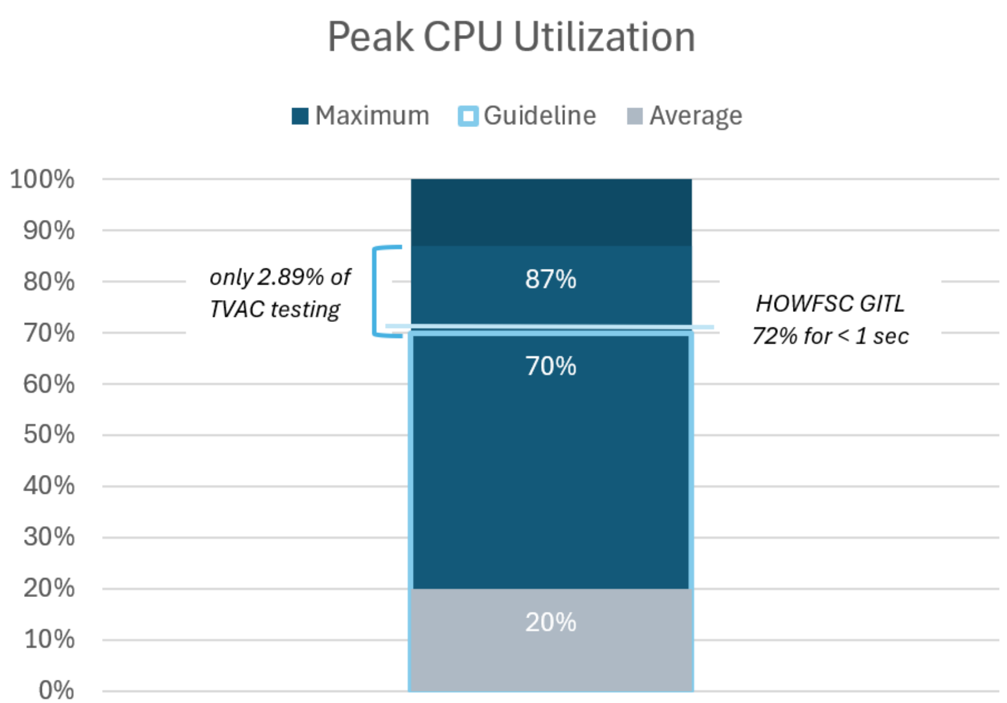





Jira Anomaly Report Burndown

*VAP = verification activity plan*
*FFT = full functional test*
*TVAC = thermal vacuum test*

# CGI FSW Resource Utilization




*Margin on utilization helps prevent starvation of critical processes such as fault protection and software watchdog reset.*

- Memory
  - SDRAM: 119 MB / 256 MB
  - DDR2: 854 MB / 4 GB
  - MRAM: 26 MB / 32 MB
- HK Packet Throughput (bps)
  - Max: 100 K
  - Avg: 27.2 K
  - Allotted: 1 M
- Sci Packet Throughput (bps)
  - Max: 1.76 M
  - Avg: 125 K
  - Allotted: 60 M

# CGI FSW Issues Encountered

- Dictionary inconsistencies
  - Autocoding FSW encouraged
- Resource contention
  - Shared SpaceWire ports
  - WFSC task demand
    - High and low priorities
  - Priority inversions & deadlock
- Software bugs
  - State machine flow errors
  - Misuse of types and casting
    - Floating point issues
- Incorrect or missing asynchronous behavior
  - Base classes lacked abort mechanism
    - State machines can be halted at any time by FP or ACS disturbance handling
  - Semaphores for resource locking
    - Edge cases across all code paths
  - Undesired hardware responses to asynchronous commanding required software solutions

# CGI FSW Lessons Learned

- Asynchronous behavior defined clearly and early
  - Hardware and software can incorporate in original design
  - All requirements, really!
- Solid build and test process
  - Releases traced to requirements and V&V
- COTS hardware can cost more
  - Increased number of interfaces
  - Complexity and risk introduced further down the pipeline
- Software documentation
  - Code commenting, diagrams
  - Notes on design, unit tests, idiosyncrasies, danger zones
- HOWFSC: GITL or onboard?
  - Computational gap between requirements and capability
    - Single vs multi-core processing
    - Need margin on CPU utilization
  - Onboard HOWFSC required second processor board
  - Partition software requirements between FSW and GSW

# Appendix

- [1] Malloc-Free Pseudoinverse Solver with Eigen C++ Template Library: https://github.com/silverSapphire/Eigen-Malloc-Free-Pseudoinverse