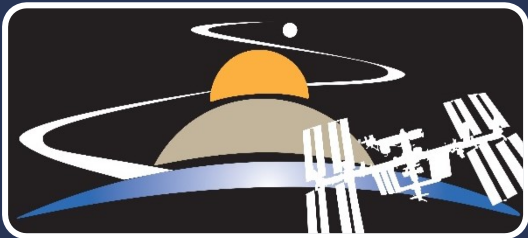


Exploration Research and Technology Programs

Command & Data Handling and Software

8 June 2023

Tyler DeCaussin

Approved for public release; distribution is unlimited. Public Affairs release approval AFRL-2023-2756

Command & Data Handling

Approved for public release; distribution is unlimited. Public Affairs release approval
AFRL-2023-2756

UNP What is it?

Generates and stores all the telemetry you can; doesn't all need to be downlinked.	Maintains team's high level telemetry to downlink during ground contacts for monitoring subsystem health.	Has a time-to-live for state of health telemetry.	Keeps a command execution history.
Generates and stores subsystem metadata. Accept/reject counts, last communication time.	Should not store telemetry from subsystems that are powered off.	Utilizes data reduction techniques when applicable. Averaging, binning, compression	Provides hooks for command execution on mode transitions.
Has a way to verify command receipt with ease for in pass operations.	Keeps a command execution history.	Plans for the unexpected and have a native pass through command for all subsystems.	Has commands to change telemetry generation, storage, and downlink parameters.

Approved for public release; distribution is unlimited. Public Affairs release approval AFRL-2023-2756

UNP

Processor Selection

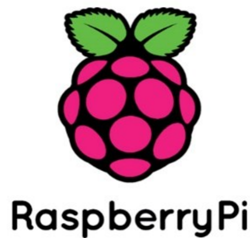
1. **Bare metal or OS?**
   - Bare metal (no OS) means micro-controllers are a viable option
   - An OS on a micro-controller is possible but often not practical
2. **Power budget**
   - Instantaneous and average power draw limitations
   - Is there a sleep mode and does that help?
   - Frequency scaling using a governor may help slightly
3. **Memory (RAM)**
   - The amount of memory required to run the OS and flight software
   - Is there room to store flight software on a RAM disk?
4. **Memory (Non-volatile)**
   - The amount of memory required to store the bootloader, kernel, OS, flight software, telemetry, and duplicates for any redundant copies.
   - Can memory be added? SD card, eMMC, NAND, NOR, FRAM?
5. **Interfaces**
   - Are there enough UARTs, SPI buses, I2C buses, USB, ethernet, etc…?
   - Is the throughput high enough on each interface?

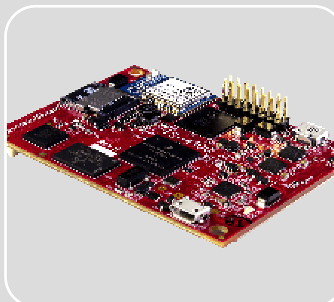
Approved for public release; distribution is unlimited. Public Affairs release approval
AFRL-2023-2756

UNP Hardware

Single Board Computers

Microcontrollers





Beaglebone Black	Raspberry pi	TS-4100	MSP430FRxxx	Atmel AVR (Arduino)

UNP OS Selection

OS	None (Bare Metal)	Linux	RealTime OS
Power Draw	Low (sleep modes)	Higher	Higher
Memory Consumption	Low	Higher	Medium
Scheduling	Deterministic	Less Deterministic	Deterministic
Dependencies	Few	More	More
User Base	Depends on processor	Large	Smaller
Learning Curve	Steep	Shallower	Medium

Approved for public release; distribution is unlimited. Public Affairs release approval
AFRL-2023-2756

UNP

Memory - Radiation

Problem: Memory corruption is a common occurrence in space-based applications, and steps should be taken to mitigate mission ending corruption.

- Generally orbits with high inclinations and altitudes have a harsher radiation environment.
- Memory is particularly vulnerable when being written to and energized.

Memory Types: Some memory types are more resilient to radiation. Electronic memory is the least resilient to single event upsets; SD cards in particular are known to fail on orbit. Below are some more resilient memory types to use for high importance memory.

- Phase change memory (PCM)
- Ferrite RAM (FRAM)
- Single-layer cell (SLC) NAND instead of Multi-layer cell (MLC) NAND

Approved for public release; distribution is unlimited. Public Affairs release approval AFRL-2023-2756
Distribution C. U.S. Government and Contractors

UNP

Memory – Radiation Mitigation

Mitigation Strategies: Shielding, error correction, and .

- Radiation hardened memory is effective, but expensive.
- Error correction
    - Error code correction (ECC), triple modular redundancy (TMR).
- Partitioning, provides some isolation to corruption and regular utilization.
- Hard reset, assuming the startup process is reliable resetting can be a viable response.

Error Correction Implementations:

MD5

- Hash comparison between recent computation and known good hash.
- No ability to correct file.

TMR

- Bit level verification and voting scheme between at least 3 file copies
- All versions scanned and majority vote wins minority bit values are corrected in differing files.

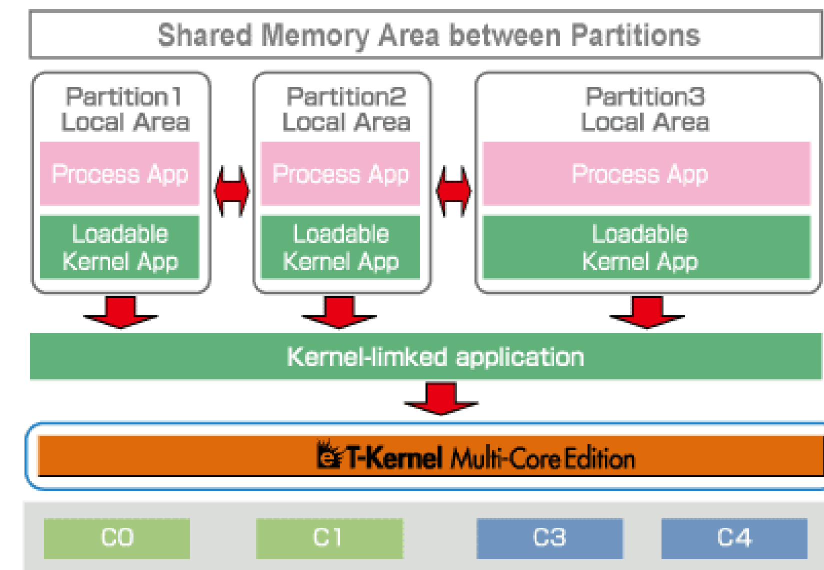
Approved for public release; distribution is unlimited. Public Affairs release approval AFRL-2023-2756
Distribution C. U.S. Government and Contractors

Memory - Partitions

Partitions: Partitions help to isolate memory usage. They are helpful for preserving space in critical areas if logs or telemetry grow to unexpected size and may be used to mitigate the spreading of corruption.

Suggested Practices:

- Critical memory should rarely be written to and have its own partition.
- Low priority, high write frequency files (like logs) should have their own partition.
- Keep redundant partitions for flight software, this allows TMR across multiple partitions.
- Leave an unused partition to fail over to for things like telemetry storage.



Approved for public release; distribution is unlimited. Public Affairs release approval AFRL-2023-2756

Distribution C. U.S. Government and Contractors

UNP

Memory – File System

Exploration Research and Technology Programs

Problem: File systems that work well terrestrially are not always the best option for space based applications. The following are important factors in deciding on a file systems to use.

Memory Retirement: In flash, blocks may be marked as retired if a bad read or write occurs. Single event upsets can potentially wreak havoc on a partition by retiring large blocks of memory. Partitions may help to isolate some of this.

Power Loss: Unexpected power cycles can cause a filesystem to be corrupted. A journaling file system can be used to mitigate this, but it is still crucial to test and understand what the behavior is and potentially account for it with scrubbing or TMR.

RAM Disk: For files that do not need to persist, a RAM disk can be used instead of flash. This is particularly useful for files written to at a higher frequency because it will reduce wear and the likelihood of a single event upset.

Flash File Systems: Raw flash is handled differently than other memory types so JFFS2, YAFFS, and UBIFS are the main options.

Approved for public release; distribution is unlimited. Public Affairs release approval AFRL-2023-2756
Distribution C. U.S. Government and Contractors

UNP System Startup

- Understand your process
  - Bootloader → Kernel → OS → FSW
- Identify and mitigate your risks
  - Redundant bootloader, kernel?
  - Scrub critical OS files?
  - TMR flight software?
- Handle memory failures
  - Scrub, TMR, reformat, fail over?

Approved for public release; distribution is unlimited. Public Affairs release approval
AFRL-2023-2756

Startup Stages

Hardware Stage

- Checkout Memory
  - Mounting
  - I/O tests
- Correct Memory
  - Disk check
  - Reformat
  - Fail over to alternate
  - RAM disk

Software Stage

- Checkout Software
  - Hash checks
- Correct Software
  - Scrubbing
  - TMR
  - Reinstall
  - Fail over

Monitor Stage

- Faults
  - Trigger software restarts
  - Trigger CDH reboot
  - Trigger software reinstalls
- Reverts
  - Only if on orbit software updates

Approved for public release; distribution is unlimited. Public Affairs release approval AFRL-2023-2756

UNP Software Start-up

Exploration Research and Technology Programs

Flight software startup options	Startup Verifications	Failover Options
• Bash script to executes binaries • Use services and Linux package management	• Startup scripts should include fail over logic in case of failure to run a binary. • If on-orbit software updates are available, allow for reverting packages to previous version. • Use radiation mitigation strategies	• A 'gold copy' of flight software should be kept in its own partition. • OS partition backup (requires custom bootloader or other intricate design) • Failover scenarios, such as SD -> NAND -> PCM -> RAM

Approved for public release; distribution is unlimited. Public Affairs release approval AFRL-2023-2756
Distribution C. U.S. Government and Contractors

UNP Serial Protocol

Asynchronous

Advantages: Polling not required
Disadvantages: Single connection
Considerations: GPIO limitations

Synchronous

Advantages: Multi-device bus
Disadvantages: Polling required
Considerations: Board layout

Single ended

- **Advantages**: Fewer data lines
- **Disadvantages**: Speed
- **Considerations**: Harness length

Differential

- **Advantages**: Speed
- **Disadvantages**: Additional data lines
- **Considerations**: Power utilization

	Asynchronous	Synchronous
Single-ended	TTL, RS-232, RS-485	SPI, I2C, OneWire
Differential	LVDS, RS-422	USB

Approved for public release; distribution is unlimited. Public Affairs release approval AFRL-2023-2756

I2C / One Wire Communication Tips

Bus capacitance

- Numerous sensors on comm bus creates additional impedance
- Mitigation 1: Adding a buffer in line can reduce
- Mitigation 2: More separate comm buses
- Mitigation 3: Digital Mux different paths to comm bus.

Bad reads

- Even healthy comms will provide bad reads occasionally.
- Add validity checking on values
- See SSC18-I-01: Dellingr: Reliability lessons learned from on-orbit

Stuck Bus lock out state

- Especially for critical subsystems
- Read more:
  - See Analog AN-686
- Mitigation 1 (I2C): Add a buffer chip like Analog LTC4308 which automatically senses a stuck bus condition and pulses the SCL line
- Mitigation 2 (I2C): Command that can send 16x clock pulses on timeout
- Mitigation 3: Fault condition that power cycles all devices on comm bus

UNP

Fault Handling: Watchdog Timer

Benefits of Watchdogs

- To detect and recover from component or communication malfunctions
- Deterministic timed response to anomalies

Possible malfunctions

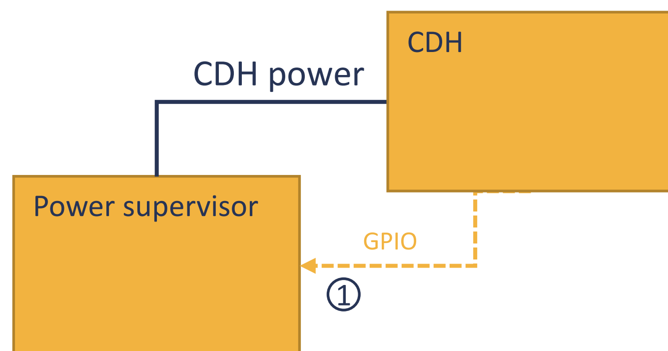
- Clock failure
- Stuck comms bus
- Program crashes

Types of Watchdogs

- Hardware-based
  - GPIO tap reset
  - Bus/component power supervisor
  - Timing often defined by RC-circuits
  - Pros: Reliable
  - Cons: Fixed
- Software-based
  - Software tap reset on Command
  - Through UART, I2C etc
  - Pros: Flexible
  - Cons: Expects higher level of vehicle functionality

Approved for public release; distribution is unlimited. Public Affairs release approval AFRL-2023-2756

CDH Watchdog (Simple)



① CDH taps Power supervisor

Fault – CDH fails to tap Power supervisor

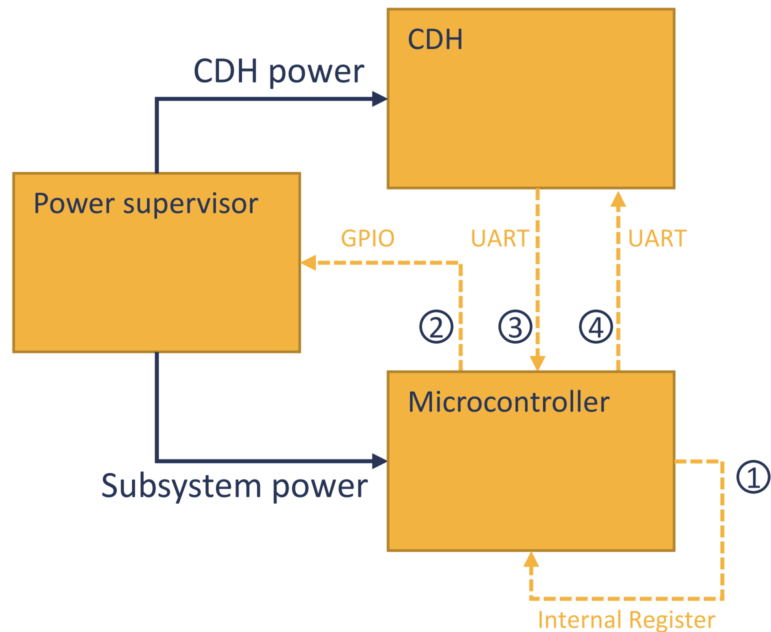
Cause: FSW fault, OS fault, I/O failure, commanded reset

Effect: Supervisor removes power for 30 seconds

Approved for public release; distribution is unlimited. Public Affairs release approval AFRL-2023-2756

UNP

CDH Watchdog (Use Case)



Fault 1 – Microcontroller fails to itself

Cause: Microcontroller fault
Effect: Microcontroller soft resets itself

Fault 2 – Microcontroller fails to tap Power supervisor

Cause: Microcontroller fault, I/O failure, commanded
Effect: Supervisor removes power for 30 seconds

Fault 3 – CDH fails to tap Microcontroller

Cause: CDH fault, I/O failure
Effect: Microcontroller starves supervisor, prompting power reset

Fault 4 – Microcontroller fails to tap CDH

Cause: Microcontroller fault, I/O failure
Effect: CDH starves Microcontroller (1), (hopefully) prompting power reset

Approved for public release; distribution is unlimited. Public Affairs release approval
AFRL-2023-2756

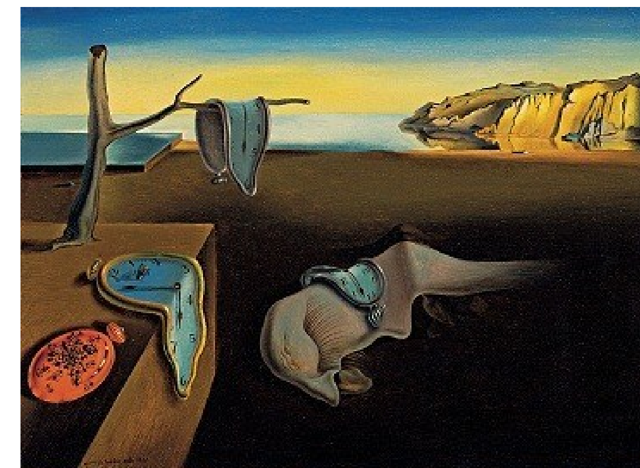
Watchdog Timer Tips

- Startup pin states (high impedance, pull-down etc.)
  - Avoid tripping watchdog during nominal startup
  - Avoid floating pin states with pull-up or pull-down
- Consider AIT states of flat sat population

Approved for public release; distribution is unlimited. Public Affairs release approval
AFRL-2023-2756

UNP Synchronizing and Managing Time

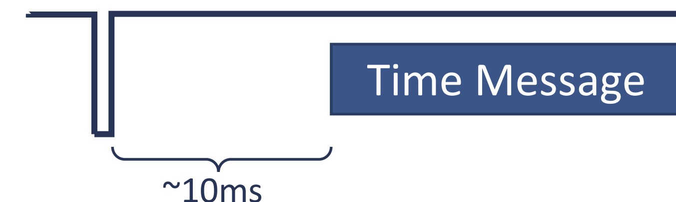
- Time requirements affect many subsystems and satellite operations
  - Command execution accuracy
  - Telemetry timestamp accuracy
  - ADCS pointing accuracy based on knowledge of position, velocity, and time
  - Synchronizing telemetry collects for payload(s)
- Time management on board a satellite needs to handle varying time knowledge circumstances to account for various CONOPS, fault handling, and GPSR lock conditions
  - Satellite reboots
  - Powering off GPSR due to entering low power modes
  - Often a GPSR will be restarted as a fault response
  - Attitude maneuvers may cause degradation or loss of GPS time knowledge
- Time sources, standards, and leap seconds may need to be accounted for
  - UTC (GMT): -18s from GPS *(currently)* , -37s from TAI *(currently)*
  - GPS: +18s from UTC *(currently)*, -19s from GPS *(always)*
  - TAI: +19s from GPS *(always)*, +37s from UTC *(currently)*
  - Linux epoch: Jan 1st 1970 00:00 UTC
  - GPS epoch: Jan 6th 1980 00:00 UTC

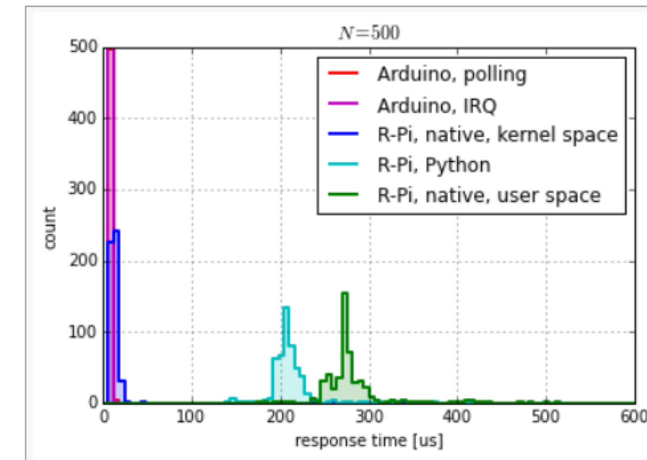


local	2022-04-06 10:15:26	Wednesday	day 096	timezone UTC-6
UTC	2022-04-06 16:15:26	Wednesday	day 096	MJD 59675.67738
GPS	2022-04-06 16:15:44	week 2204	317744 s	cycle 2 week 0156 day 3
Loran	2022-04-06 16:15:53	GRI 9940	380 s until	next TOC 16:21:46 UTC
TAI	2022-04-06 16:16:03	Wednesday	day 096	10 + 27 leap seconds = 37

Synchronizing and Managing Time

- Synchronizing to GPSR time with PPS
  - GPS position, velocity, and time are accurate at a given PPS edge
  - GPS time messages are sent after the PPS edge they correlate with
  - GPS disciplined services may exist could be used with your GPS (e.g. chrony)
- Hardware and firmware/software implementation drives accuracy of time knowledge when relying on PPS
  - Linux OSs are not real-time and have more limitations in PPS based time synchronization
    - Interrupt driven events can have large latencies and jitter in some implementations
    - Linux user space and kernel space implementations vary drastically
  - FPGAs and microcontrollers are much more accurate in responding to GPIO interrupts like PPS
- Synchronizing time-based between CDH and data collects
  - Starting collects on PPS edges can be a vary accurate way to start a data collect. This method can reduce the timestamping accuracy requirement.
  - If disturbing the PPS signal to multiple subsystems, a signal buffer may be needed
- Maintaining time without GPS and across reboots
  - Large jumps in time should be avoided, so time should be propagated accurately in cases of loss GPSR time
  - RTCs can be used to keep moving time forward which is important for telemetry timestamping





UNP Testing Interfaces

Integrated Functional Testing Considerations

Radiating over an RF path isn't always allowed or possible.

- Redirect flight software to a test interface and FEP
- Use an RF switch and attenuators to directly connect radio to ground station
- Turn file logging up and use ssh to verify functionality.

During thermal vacuum testing, the satellite will be powered down during temperature transitions.

- Provide hooks to EGSE to read temperature sensors

Software

Approved for public release; distribution is unlimited. Public Affairs release approval
AFRL-2023-2756

UNP

What is Software?

- Software is defined as the programs and other operating information used by a computer
- Software defined satellites:
  - Decouple network control and routing functions (directly programmable)
  - Can be written to quickly manage, secure, and optimize network resources
  - Modify beams, capacity, and power distribution dynamically
  - Allows the operator the ability to reconfigure the satellite, if/when needed

Approved for public release; distribution is unlimited. Public Affairs release approval
AFRL-2023-2756

UNP Workflow

- Define a process
  - How and when will things occur
    - Formal software reviews
    - Peer reviews
  - How are priorities decided
  - How is tasking assigned
    - Plan out tasks in advance and track progress
    - Use trello, bitbucket tasks, Jira, etc…
  - Have frequent tag-up meetings
    - Share progress
    - Ask for help, talk about blockers
  - Continuous integration
    - Doxygen
    - Style check
    - Valgrind

Organization Method – Waterfall Methodology

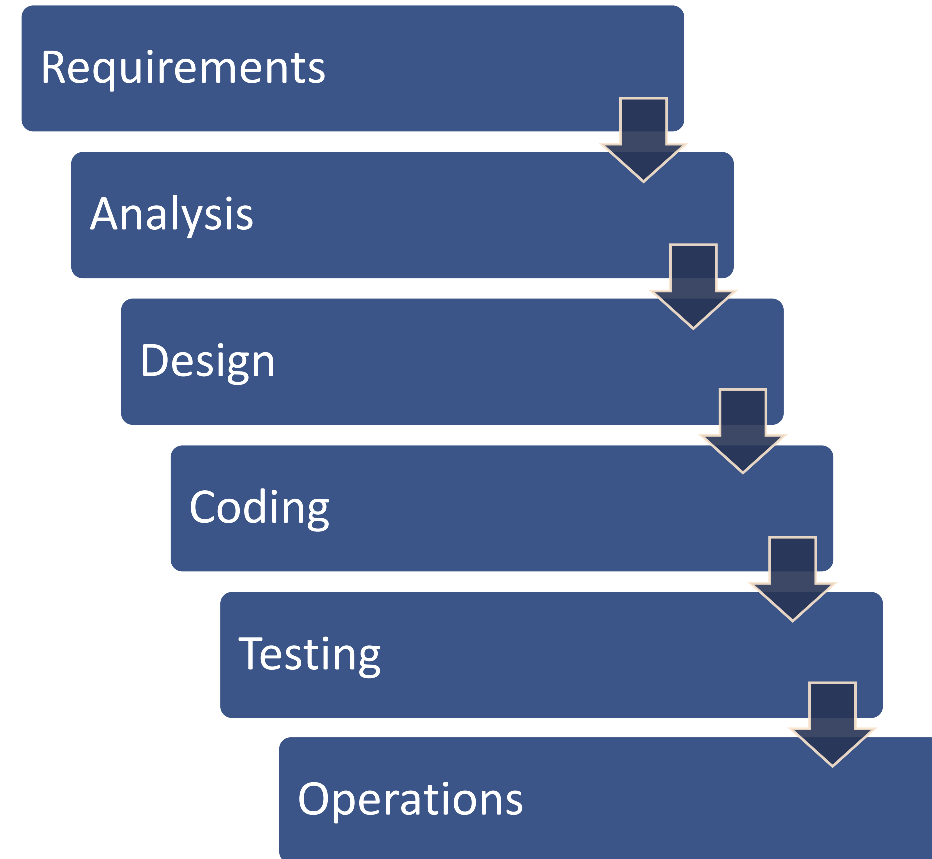
- Originates with manufacturing and construction processes
- Project mindset
- Slow phase transitions
  - Each phase depends on the deliverables of the previous ones

Pros

- Clearly links requirements to production
- Useful tool for defining milestones

Cons

- “make it up before you start, live with the consequences”
- Deadline creep – every phase delay delays the next



en.wikipedia.org/wiki/Waterfall_model#Model

Organization Method – Agile Methodology

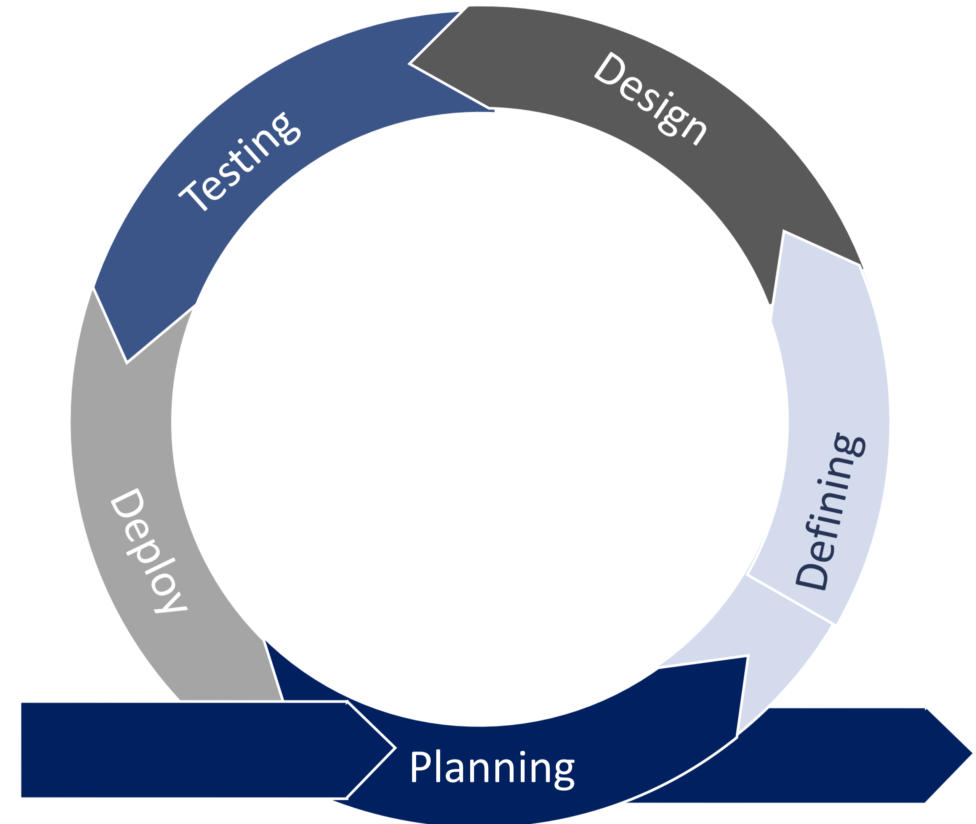
- Originates with software development
- Adaptive mindset
- Frequent phase transitions
- Work divided into iterative cycles
  - Ex. Time constrained, or feature constrained

Pros

- Flexibility for scheduling work
  - Ability to re-vector based on priority (hardware availability, etc.)
- Ability to rapidly correct issues
- Providing releases early (Minimum Viable Product)

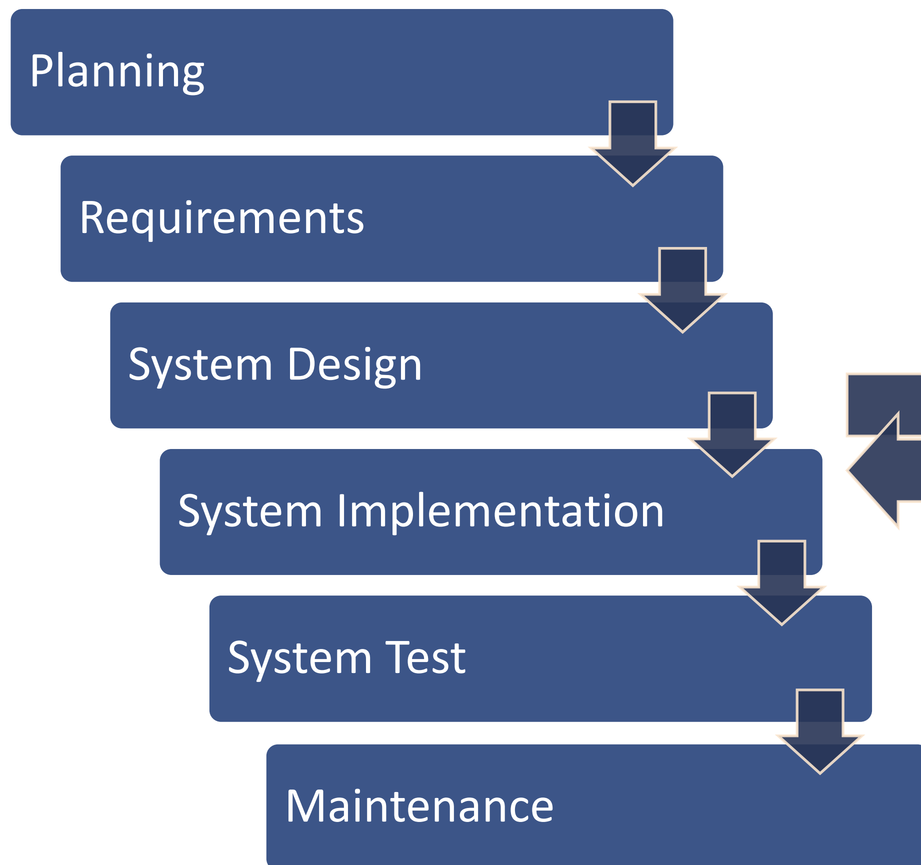
Cons

- "make it up as you go along"
- Ability to rapidly create issues
- Does not inherently guarantee requirements are met

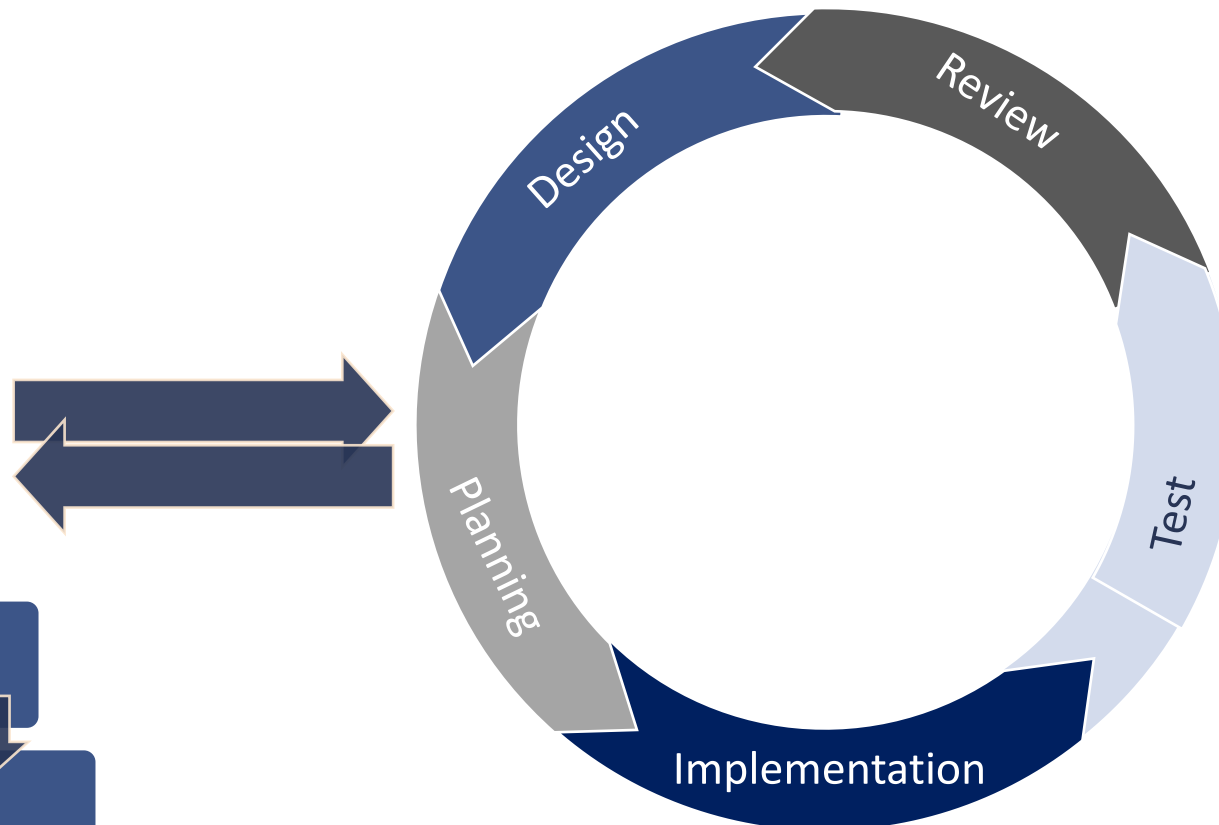


en.wikipedia.org/wiki/Agile_software_development#Agile_software_development_principles

Organization Method – AgileFalling Methodology



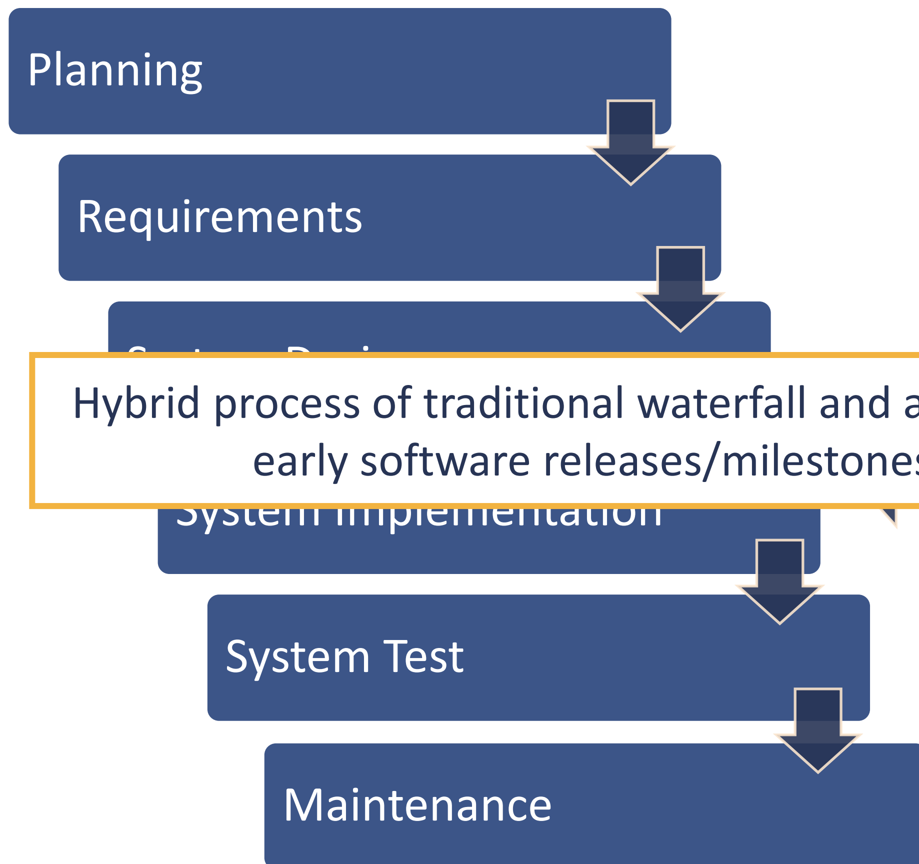
en.wikipedia.org/wiki/Waterfall_model#Model

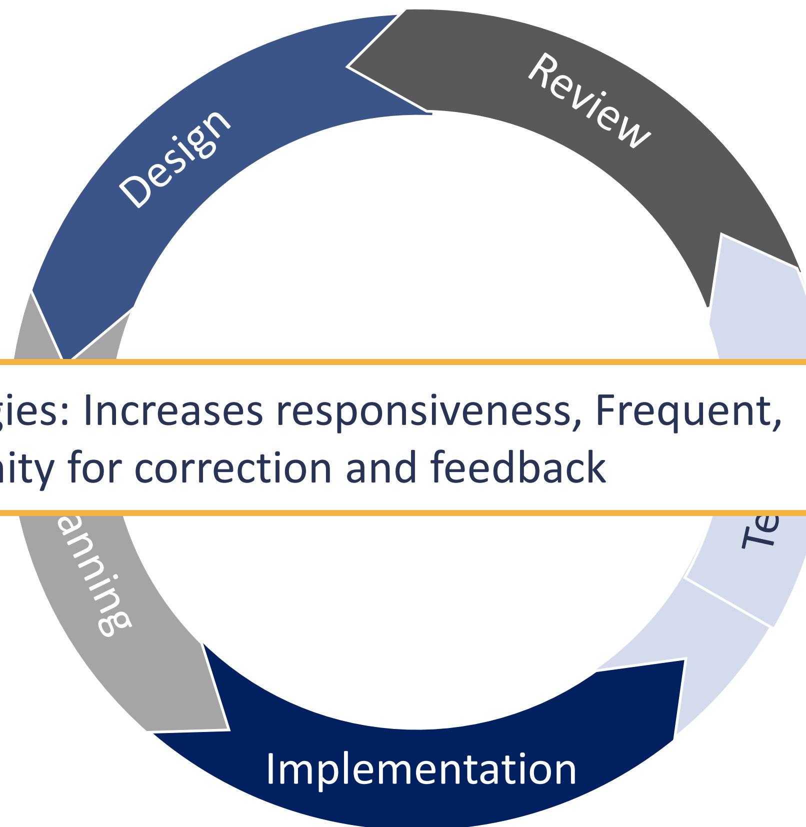


en.wikipedia.org/wiki/Agile_software_development#Agile_software_development_principles

Approved for public release; distribution is unlimited. Public Affairs release approval AFRL-2023-2756

Organization Method – AgileFalling Methodology





Hybrid process of traditional waterfall and agile methodologies: Increases responsiveness, Frequent, early software releases/milestones, Early opportunity for correction and feedback

Version Control Systems (VCS)

- Version Control Systems is the practice of tracking and managing changes to digital files
- Keeps a complete history of file changes
  - Tracking every modification in a special database
  - Allows roll-backs and version comparisons
- Allows code branching for parallel feature development
- Using a repository provides a home for code that accessible by entire team
  - Is not just one programmer's laptop
- Complements the team's workflow with deliverables and testing support

Version Control System Tools
Git
Mercurial
SVN (Subversion)

Approved for public release; distribution is unlimited. Public Affairs release approval AFRL-2023-2756

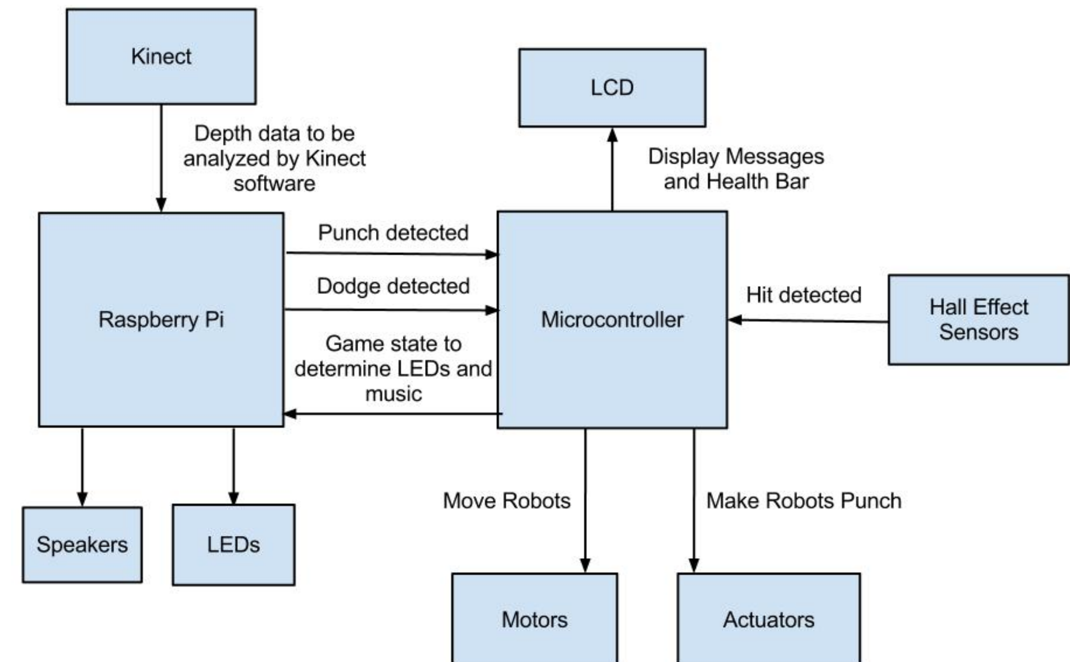
UNP Requirements

- Driven by mission objectives: Define your Minimum Viable Product (MVP)
- Hybrid of traditional requirements used to derive user stories
- User stories: try to stay away from how and focus on what and why
  - WHO “wants to” DO A SOMETHING “so that” THERE IS SOME BENEFIT/OUTCOME
  - Define “Definition of Done” – testable outcome/product
  - Make sure to time box, e.g. if you cannot finish the story in one sprint then it should probably be broken up (could be steps in a larger process)
  - Incorporate feedback
  - Benefits
    - Keep focus on user/operator/consumer
    - Drive creative solutions
    - Enable collaboration
- Other user story considerations
  - Epics – groups of user stories and associated features to realize a larger goal
  - Spikes – information gathering or decision needed to move forward on another story or feature

Approved for public release; distribution is unlimited. Public Affairs release approval
AFRL-2023-2756

Design: Diagrams

- Software Block Diagrams
  - Capture process and data flow
  - Interaction between components (application or modules)
  - Define interface between blocks (parallel development, robust)
  - Expand blocks to design/develop functionality
- Sequence diagrams
  - Processing/handling constraints
  - Illustrates where things can go wrong and how issue will be handled
- Design Concepts
  - Modularity – promotes reuse and parallel dev
  - Low coupling
  - Hardware abstraction – reusability and testing
  - Simplicity – ease of use and maintainability
  - Target configuration versus hardcoded changes
  - Timing considerations ☺ (synchronization, persistence)
  - Define faults and fault handling early
- Timing Diagrams
  - Processing time, Timeouts, Timing Dependencies



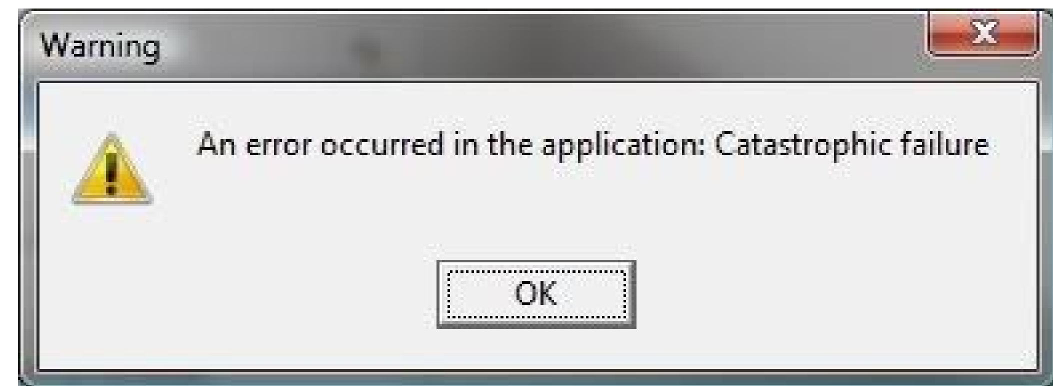
Design: Common Code

- Develop generic reusable code, modules, plugins
  - Depending on your library how you implement this may vary
  - BufferUtils
  - Common keepalive, watchdog handling
- Utilize configuration files
  - Have a common style: Xml, environment files, command line arguments
  - What strategy you use depends on how you want to make changes on orbit
- Shared memory
  - Common values can be shared time, reboot count, etc.
  - Synchronize timing

Approved for public release; distribution is unlimited. Public Affairs release approval AFRL-2023-2756

Design: Faults

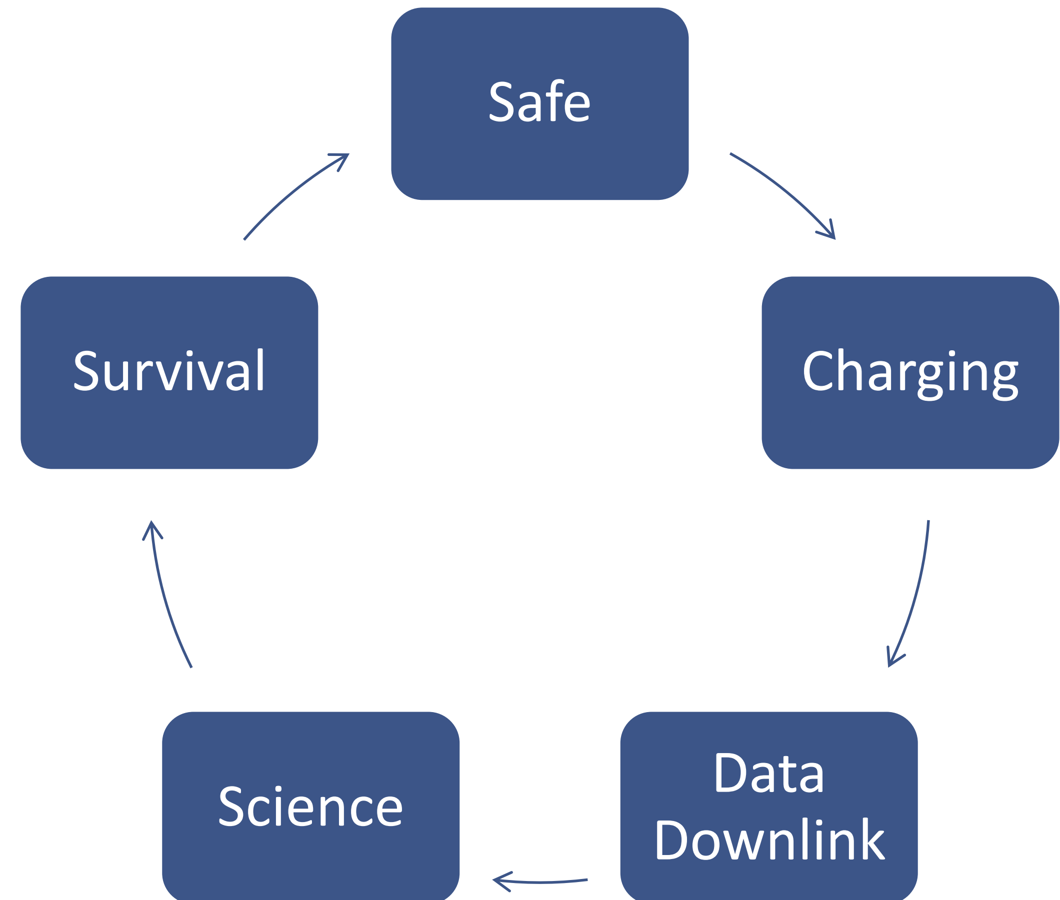
- Basic concept: detect a condition and respond
- Responses:
  - Hard & soft resets
  - Change mode
  - Restart application
  - Log an error
  - Other
- Common Faults:
  - Last successful uplink (days)
  - Time since comms with subsystem
  - Low battery
  - ADCS momentum too high
- Watchdogs



Approved for public release; distribution is unlimited. Public Affairs release approval AFRL-2023-2756

Design: Modes

- Modes should be known good states
  - Useful for recovery from fault conditions, set everything to known states
- Things to define
  - Telemetry gathering
  - Subsystem power states
  - Fault conditions



Approved for public release; distribution is unlimited. Public Affairs release approval AFRL-2023-2756

Implementation

- Try to document as you go
  - Language supported or independent (Doxygen, xml)
- Follow Language Coding Standards (C/C++ as an example)
  - Easier to review/understand, easier to maintain, more consistent
  - References
    - C - https://users.ece.cmu.edu/~eno/coding/CCodingStandard.html
    - C++
      - Hits the high points: https://users.ece.cmu.edu/~eno/coding/CppCodingStandard.html
      - Comprehensive: http://isocpp.github.io/CppCoreGuidelines/CppCoreGuidelines
        - More than just style
  - Enforcement
    - Linters, clang-tidy
- Issue tracking (use tools)
  - Capture notes, related resources, and progress (continuity between developers)
  - Relay stage/status (ticket workflow, e.g. in Design, Implementation, Review, Testing)
  - Tie issues/features to SCM system (e.g. Jira-> Bitbucket)

Approved for public release; distribution is unlimited. Public Affairs release approval
AFRL-2023-2756

Review Structure

Peer Review

- FSW code review process followed with minimum of 2 reviewers
- Benefits
  - More in-depth review than is possible in a sit-down meeting (e.g. component ratings)
  - Tight collaboration within the team to keep all pieces progressing on a consistent design
  - Catch bugs early
  - Improved team awareness of code

UNP Testing – Unit Testing

- Invest in Continuous Integration testing tool
  - Validates all of the unit tests
  - Compile source and documentation
  - Static analysis (cppcheck, clang-tidy, cpplint, vera++, CHAP, LLVM/Clang Sanitizers)
  - Dynamic analysis (valgrind and subtools, LDRA)
  - Generate reports
  - Know what broke and when
  - Example Tools: Jenkins, Travis CI, Bamboo, GitLab CI
- Tie-in for testing/test results against user stores/requirements (e.g. Jira Test tickets linked to User Story tickets)
- Develop simulators/emulators to support testing
  - Develop them only to the level needed for testing
  - Use mocks/stubs when possible (easily control inputs to interface)
    - Target abstracted hardware interfaces
- Feature/Unit Testing
  - Define coverage goals
  - Write meaningful tests
  - Leverage existing frameworks, e.g. gtest, nunit
  - Confirm core functionality

Approved for public release; distribution is unlimited. Public Affairs release approval AFRL-2023-2756

Testing – System/ Integration Testing

- Exercise each piece thoroughly and then incrementally build up (easier to track what caused an issue)
  - Confirm the full system interoperates as expected
- Define your test flow
  - How you build up your testing and system
  - Gates that must be passed to move to next phase
- Test Hardware-In-The-Loop (HIL) ASAP
  - flat-sat, segmented test hardware
- Typical types/categories:
  - Acceptance Testing Procedure (ATP) - releases/deliveries
  - Abbreviated Functional Tests (AFT) - minor/limited updates, hardware verification after environmental testing
  - Full Functional Tests (FFT) - hardware checkouts, larger code updates, hardware verification after environmental testing
  - Day In The Life (DiTL)
    - Target as much real hardware as possible in ops environment
    - Verify system meets requirements/mission objectives
    - Train “operators”
    - Find pain points, learn how to resolve anomalies on the ground
    - Tune configuration
  - Fault testing
    - Test against hardware (when safe and possible)
    - Use simulators remaining for any faults
- Automated where/when possible (trade between time to automate versus repetition)
  - Reduce error
  - Automatically capture results
  - Test early and often
  - Test-as-you-fly – ideally automated test environment, scripts, tools, etc can be leveraged for operations

Approved for public release; distribution is unlimited. Public Affairs release approval
AFRL-2023-2756

UNP Maintenance

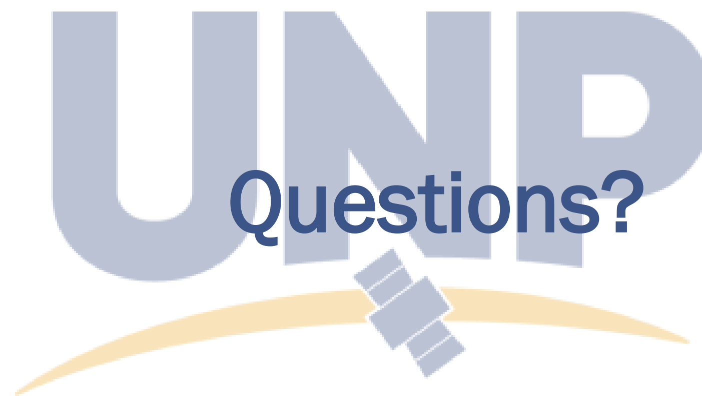
- Types
  - Corrective – fix bugs
  - Adaptive – account for hardware changes/degradation or new stretch goals
  - Perfective – address things like ease of operations
  - Preventative – forward looking to avoid serious problems in the future
- Mission lifespan support
  - On-orbit updates: functional changes, bug fixes
    - Define release process, testing process, update process (upload procedure and verification)
  - Plans for handling degradation of hardware: solar array efficiency, battery capacity
    - Ideally configuration changes
- Reuse in other missions/programs
  - Plan for LTS or ROI
  - Bug fixes/tracking
  - Improved design
  - Implement enhancements
- Documentation
- Retire/EOL – source, build artifact preservation

Approved for public release; distribution is unlimited. Public Affairs release approval AFRL-2023-2756

Tips for Collaboration

- Encourage aptitude within the team
  - Fosters innovation
  - Promotes ownership
- Prioritize readability and reusability in code
  - Modular architecture
    - Avoid monolithic architectures that are complex and not reusable
  - Write code that reviewers can review meaningfully
    - If it is hard to read, it is hard to review
      - Ex. Nested for-loops versus broken out functions
- Provide programmers with convenient access to hardware for rapidly testing code
  - Having an minimum viable test setup for software task
  - Look for a COTS development board, or a cheap COTS equivalent

Approved for public release; distribution is unlimited. Public Affairs release approval AFRL-2023-2756



Questions?

Approved for public release; distribution is unlimited. Public Affairs release approval
AFRL-2023-2756