

An Agile-Like Approach to Hardware Development: The Ejectable Data Recorder (EDR) for Orion's Ascent Abort 2 (AA-2) Test Flight

Kristina Rojdev, NASA-JSC
Antony Williams, Jacobs Engineering
Jenny Devolites, NASA-JSC
Jon Olansen, NASA-JSC

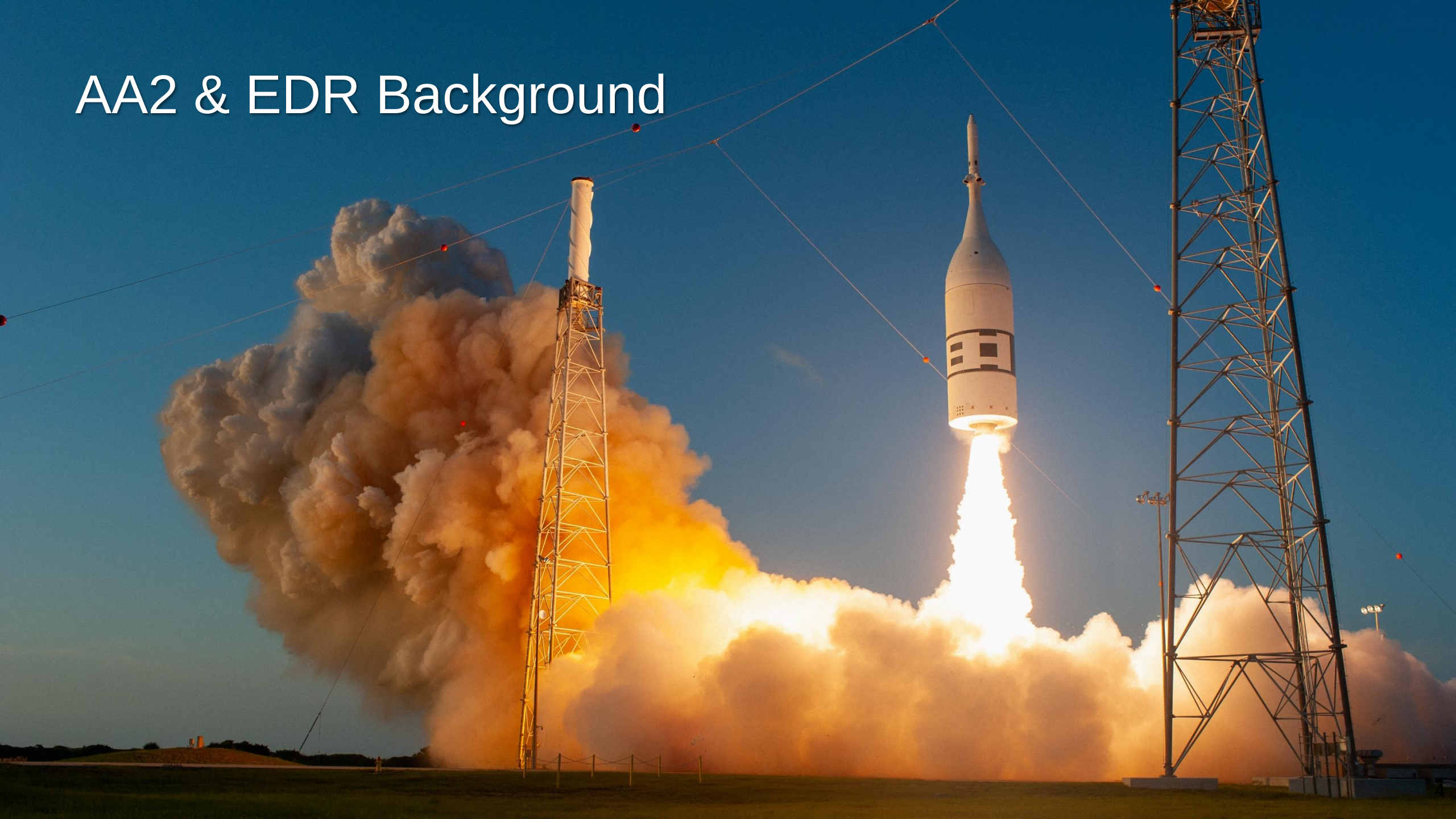
Executive Summary

- Many “Agile” software development approaches emphasize:
 - Prototyping and user testing instead of lengthy requirements analysis and development
 - Short, rapid development cycles with specific objectives instead of lengthy development periods
 - Collaboration and team engagement instead of extensive documentation
- The Ascent Abort 2 Crew Module and Separation Ring Team employed these principles to produce multiple hardware and software end items
 - Flight software
 - Simulation development
 - Design/Analysis cycles
 - Hardware/Software integrated testing
 - Ejectable Data Recorder (EDR) subsystem
- Purpose: Present a case study (EDR) on how to efficiently build flight hardware
 - Did not set up this project with an Agile approach/Scrum Framework
 - Found many similarities to Agile approach in certain areas
 - Compare EDR team execution with Agile approach
 - Discuss lessons learned and recommendations for future approaches

Outline

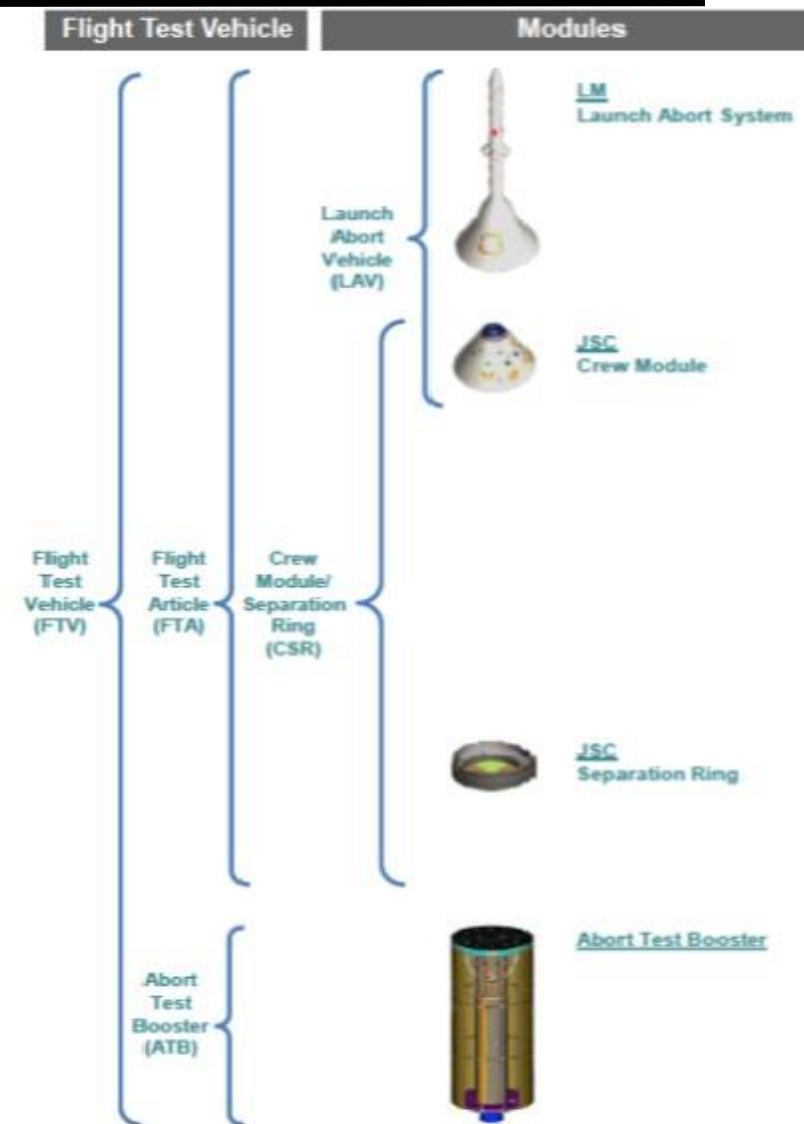
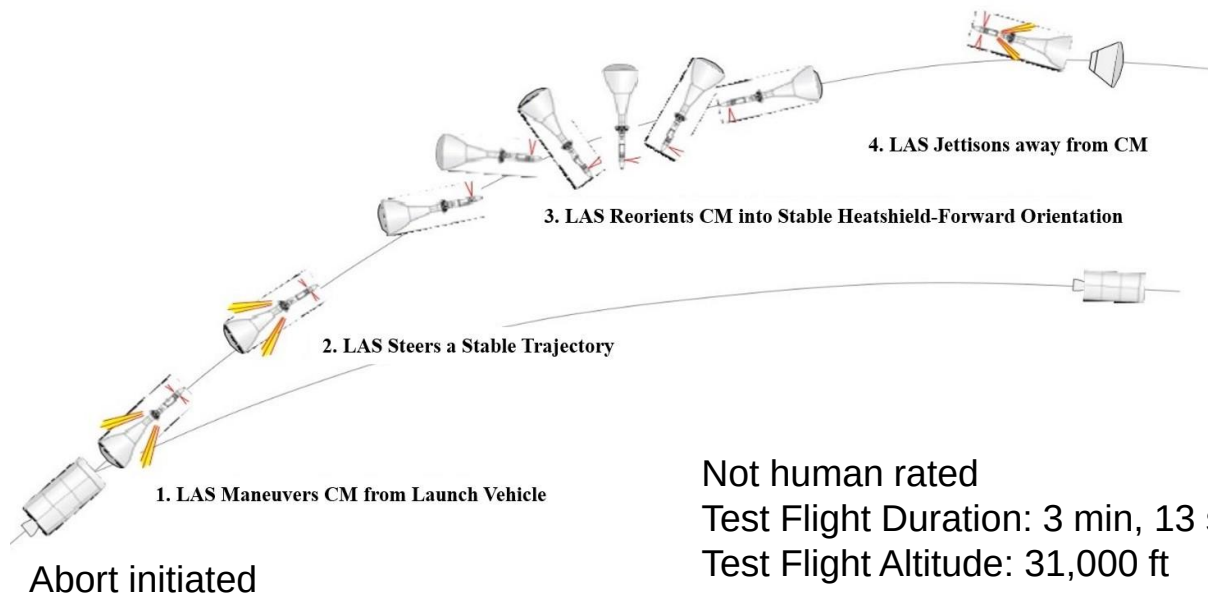
- Background on AA2 and the addition of the EDR Subsystem
- EDR Approach and Experience Juxtaposed with the 12 Agile Principles
- EDR Comparison with Agile and Recommendations for future Agile-like Hardware Development

AA2 & EDR Background



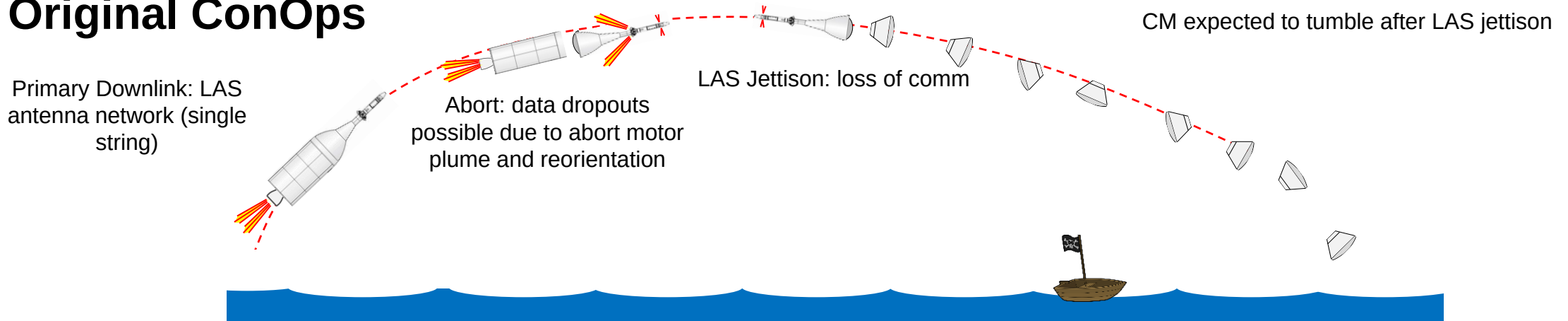
Background: AA2 Flight Test

- Purpose: to perform an ascent abort with the LAS under flight-like, high dynamic pressure conditions
- Objectives:
 - Primary: Demonstrate that the LAS would effectively maneuver the Crew Module (CM) away from the booster and jettison from the CM
 - Secondary: Collect flight test data which will be used in Orion vehicle certification



Background: Why EDR?

Original ConOps



Risk 10 - Comm Design: Given that the comm subsystem conceptual design is a single string subsystem that ceases to function at initiation of LAS jettison, there is a possibility that mission critical data will not be captured.

DFI-001: Redundant Comm and near-field separation data trade

Options:

- Redundant comm with antennae on CM
- On-board recorder (hardened and/or ejectable)

DFI-002: Telemetry antenna trade

Options:

- Add heat shield antenna at abort and transition from LAS antenna to CM back shell antenna (with heat shield antenna) at LAS Jettison

DFI-003: Ejectable/Jettisonable DFI on-board recorder trade

Options:

- Ruggedized on-board recorder that can be ejected prior to splash down

Rojdev, K.Hagen, J., Burlone, M., et al., "An Ejectable Data Recorder Subsystem for the Ascent Abort-2 Test Flight of the Orion Launch Abort System," *IEEE Aerospace Conference*, March 2019.

-
- A top-down diagram of the Orion crew module interior. The module is conical in shape. At the top (aft) is a large, grey, cylindrical structure labeled 'Ejectable Data Recorder'. Two red arrows point towards this structure from the side walls. Below this structure are several rows of seats or bays, colored in shades of brown and green. At the bottom (forward) of the module is a control area with various panels and equipment. The entire interior is enclosed within a grey outer shell.



Challenge

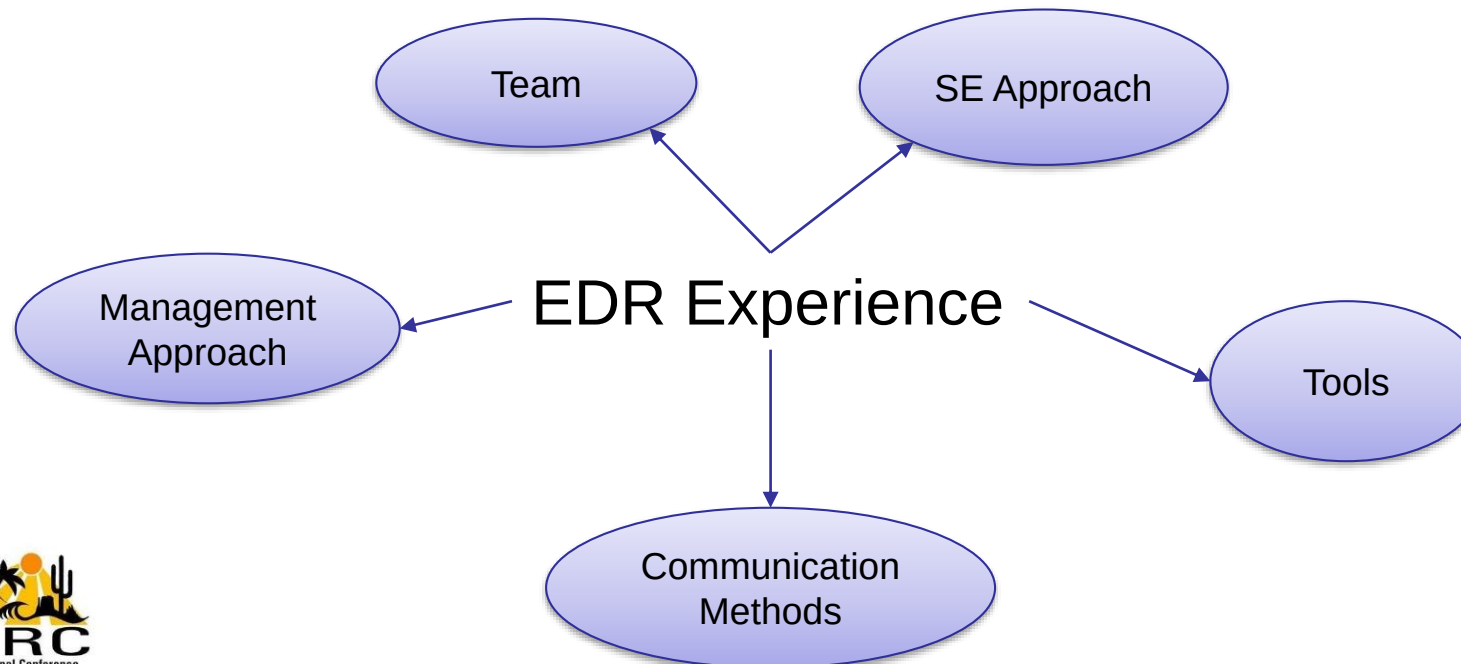
Drivers that led to the approach we used:

- EDR was a backup system
 - Multiple antennae on LAS and CM
 - Buffer rebroadcast system
 - EDR with 2 strings and 12 recorders
- EDR schedule highly compressed compared to other subsystems
 - New build hardware that started later than the rest of the project
 - Launch date was accelerated by 6 months

Challenge: Find an efficient way to develop hardware that is “good enough”

“Good Enough”

- The system performs the functions it needs to perform
- The system survives its environments
- The system provides enough insight to determine whether there is an issue
- We only need to recover one recorder with data on it



EDR Approach and Experience Juxtaposed with the 12 Agile Principles

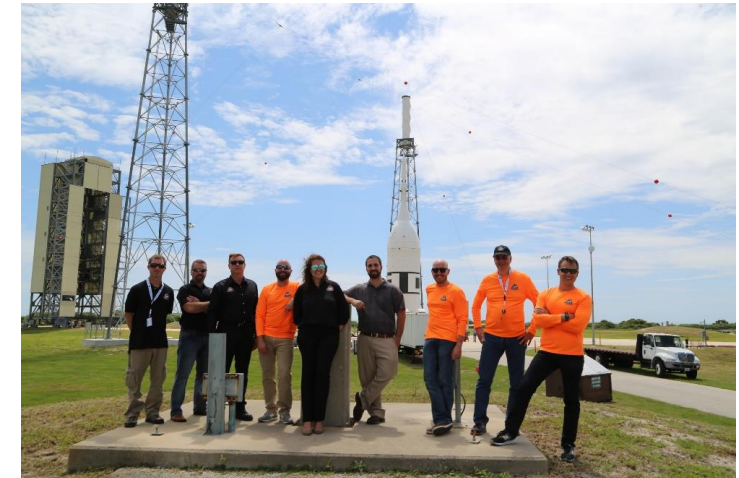


What is Agile?

- Agile came out of software development as a way to more efficiently delivery software
 - *Agile software development refers to software development methodologies centered around the idea of iterative development, where requirements and solutions evolve through collaboration between self-organizing, cross-functional teams.* – cprime.com
 - Gaining popularity and increasingly being used in the software world
- Agile is a mindset on the ability to move quickly
 - The team *becomes* Agile, you don't *do* Agile
- Multiple frameworks that describe how to become Agile (i.e. Scrum, XP, Kanban, etc.)
 - Scrum helps teams become self-emerging and self-managing
- 4 Core Values to becoming Agile (<http://agilemanifesto.org/>)
- 12 principles that follow from these values

Team

- Agile principles:
 - Build projects around **motivated individuals**. Give them the **environment** and **support** they need, and trust them to get the job done.
 - The most efficient and effective method of conveying information to and within a development team is **face-to-face** conversation.
 - The best architectures, requirements, and designs emerge from **self-organizing teams**.
 - **Business people** and developers must work together daily throughout the project
- Core Team Size: 4 – 12
- Team Composition: 60% Contractor, 40% Civil Servant
- Ground Rules
 - Respect and trust one another
 - Participation is mandatory
 - Everyone wears multiple hats
 - Communication is key
 - Individuals expected to take ownership and responsibility for their tasks
- Co-Location
 - Majority of the team co-located in one building, which was the same location of the main lab
 - EDR Subsystem Lead maintained strong communication with the whole team and traveled for major activities
 - Consolidated work to 2 labs
- Funding resided with the subsystem lead



Team Effectiveness

- Agile principle:
 - At regular intervals, the team reflects on how to become **more effective**, then tunes and **adjusts** its behavior accordingly.
- Meeting Approach
 - Avoided status meetings, unless there was lack of communication via other means
 - Attempted to limit distribution to those necessary for the discussion
 - Primary use of meetings was to work issues and have necessary discussions
- Communication tools: Email, Text, IM, and phone calls
 - Primary method was email
- Daily log (only during periods of intense activity) **Scrum-like**
 - Bulleted list of what was accomplished, what's planned for tomorrow, where you need support, and from whom you need support
- Short daily tagups (only during major testing or schedule critical troubleshooting) **Scrum-like**
- Lessons learned list used throughout the lifecycle
- Engineering skill set and expectations adjusted throughout the lifecycle



Development:

- Out of the box thinking
- Experimental
- Diverse experience



Engineering Unit:

- Experimental
- Collaborative
- Organized



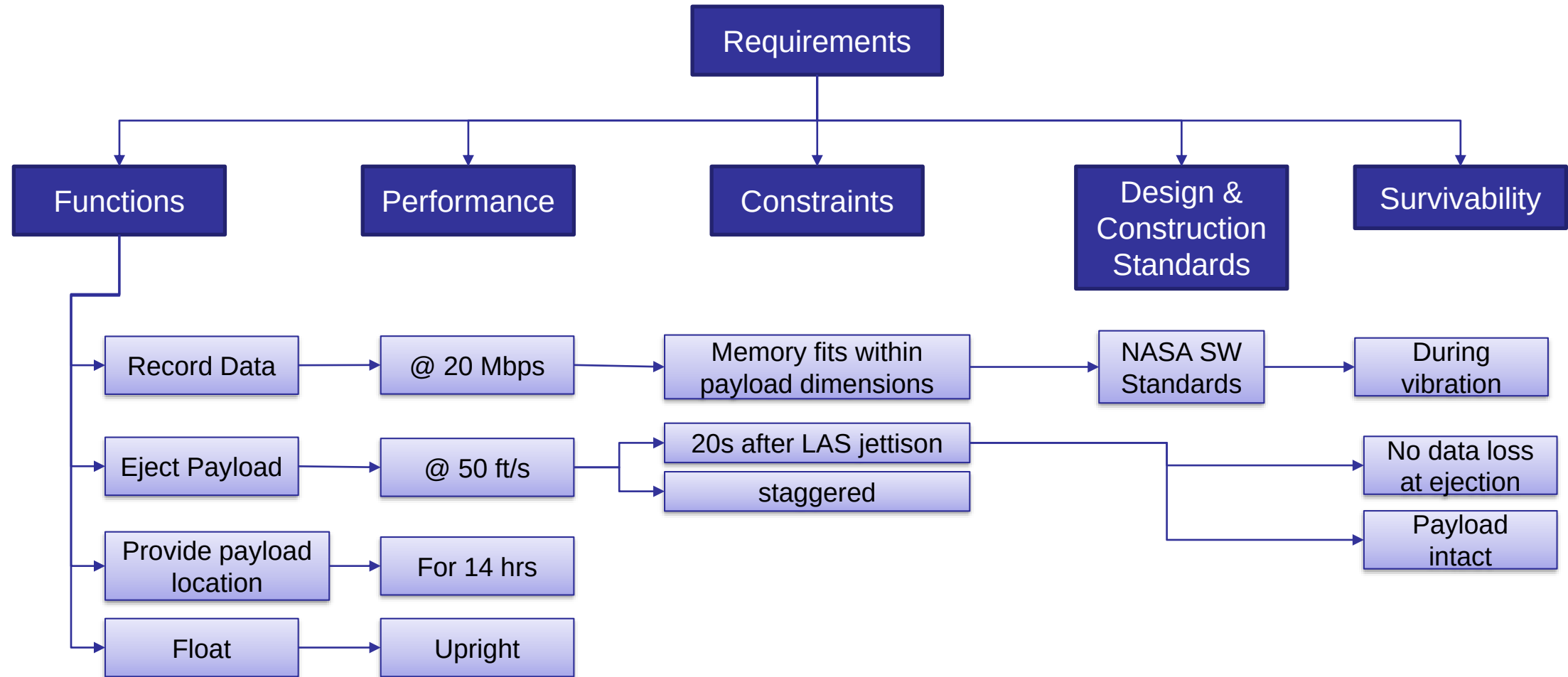
Flight:

- Collaborative
- Very organized
- Attention to detail
- Patient

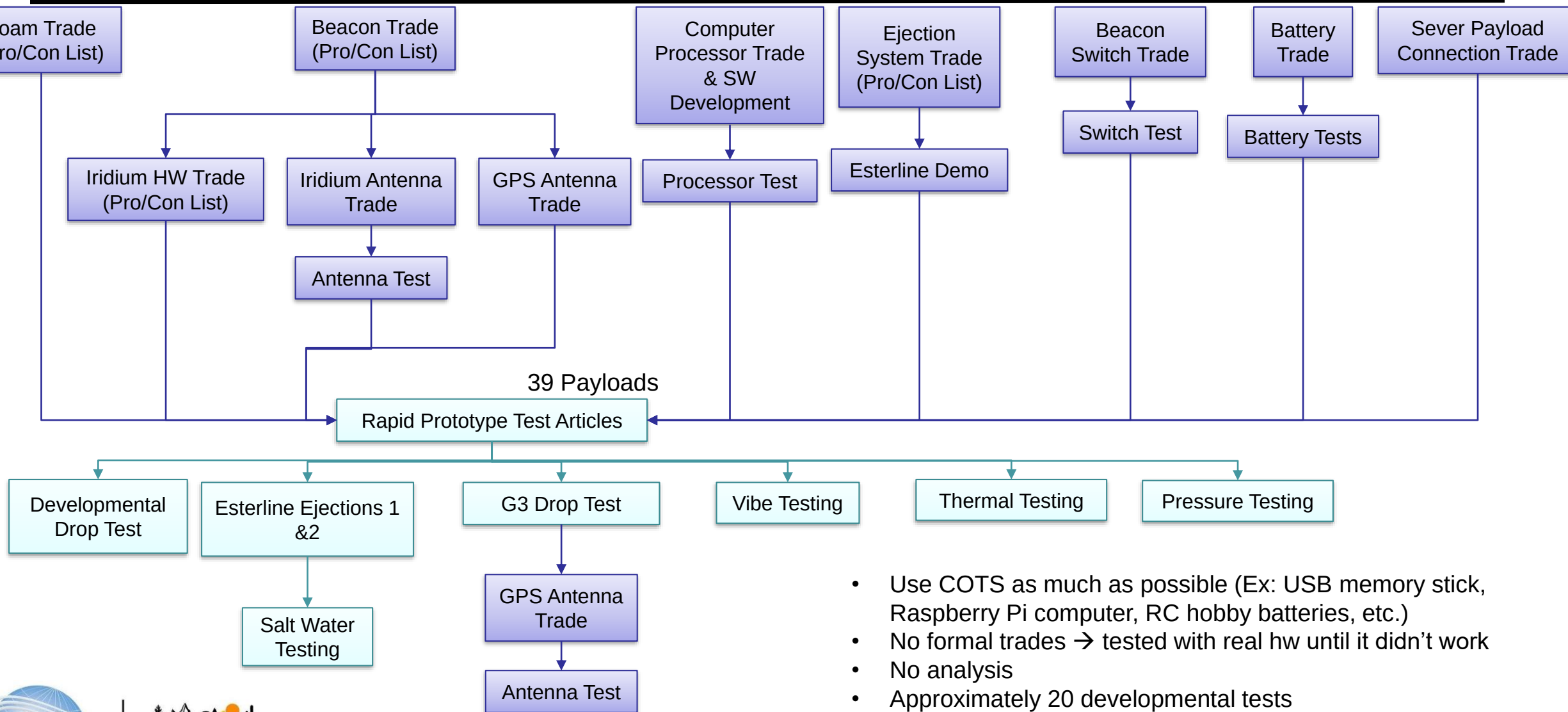
Technical

- Agile Principles:
 - Working components are the primary measure of progress.
 - Continuous attention to technical excellence and good design enhances agility.
 - Simplicity--the art of maximizing the amount of work not done--is essential.
 - Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
- Used the Distributed, Data-centric Collaborative Information Management (DDCIM) Approach
 - Devolites, J., Williams, A., Rojdev, K., “Distributed, Data-centric Collaborative Information Management (DDCIM) for Project Management and SE&I,” *INCOSE WSRC*, Sept. 17, 2020.
 - Subsystem lead maintained ownership of systems engineering related products (i.e. requirements)
 - EDR Lead developed constructs
 - EDR Team collaboratively added/edited information that fed into the products
 - Enabled by SharePoint Tool
- Iterative approach to design and development

SE Approach: Incremental Demonstrations

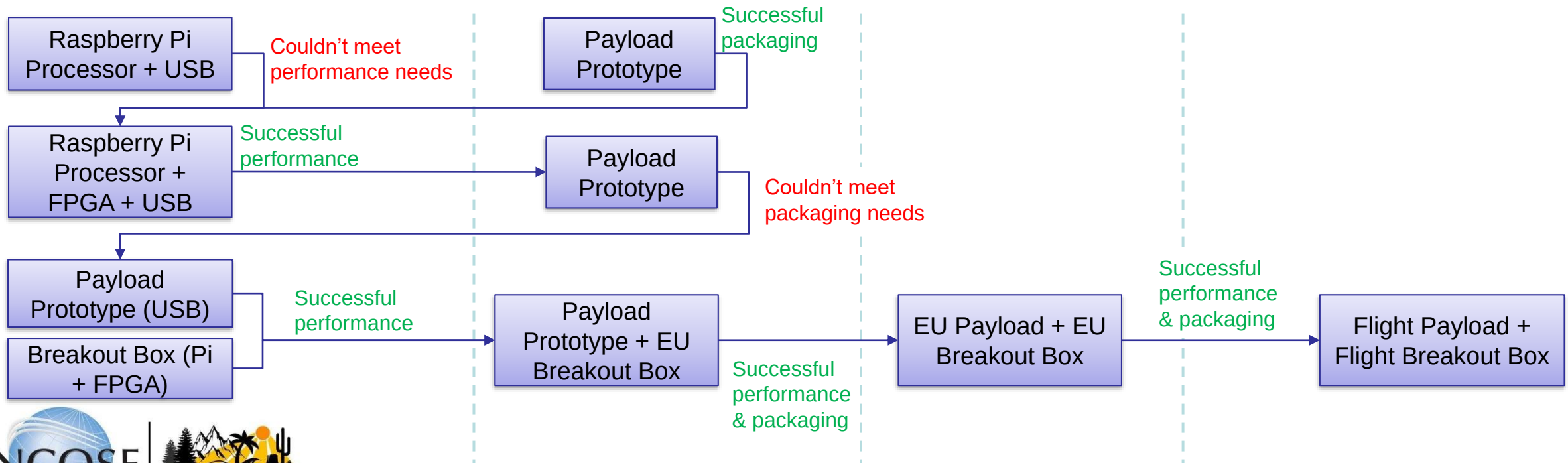
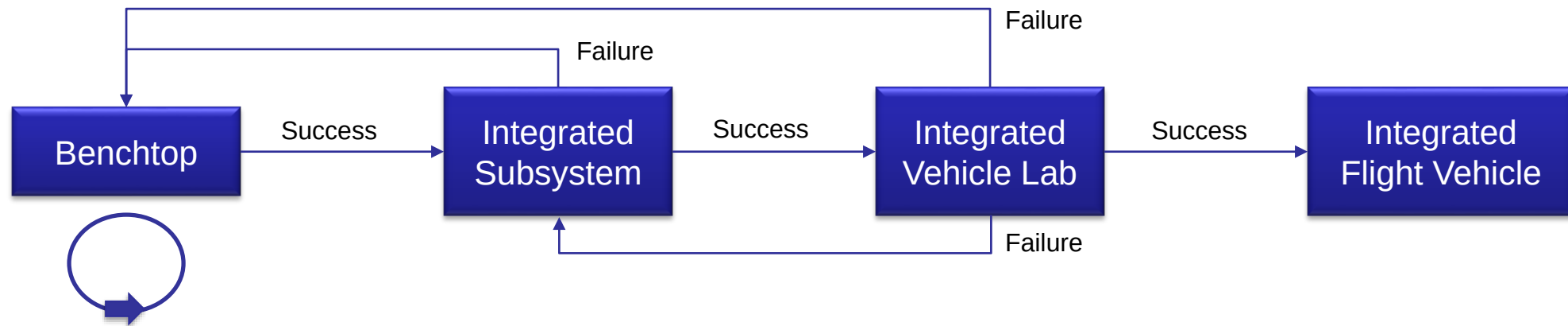


SE Approach: Prototyping Phase



- Use COTS as much as possible (Ex: USB memory stick, Raspberry Pi computer, RC hobby batteries, etc.)
- No formal trades → tested with real hw until it didn't work
- No analysis
- Approximately 20 developmental tests
 - Napkin to hw (Payload prototype – 1 week)
 - Rapid Prototype → Test → Learn → Repeat

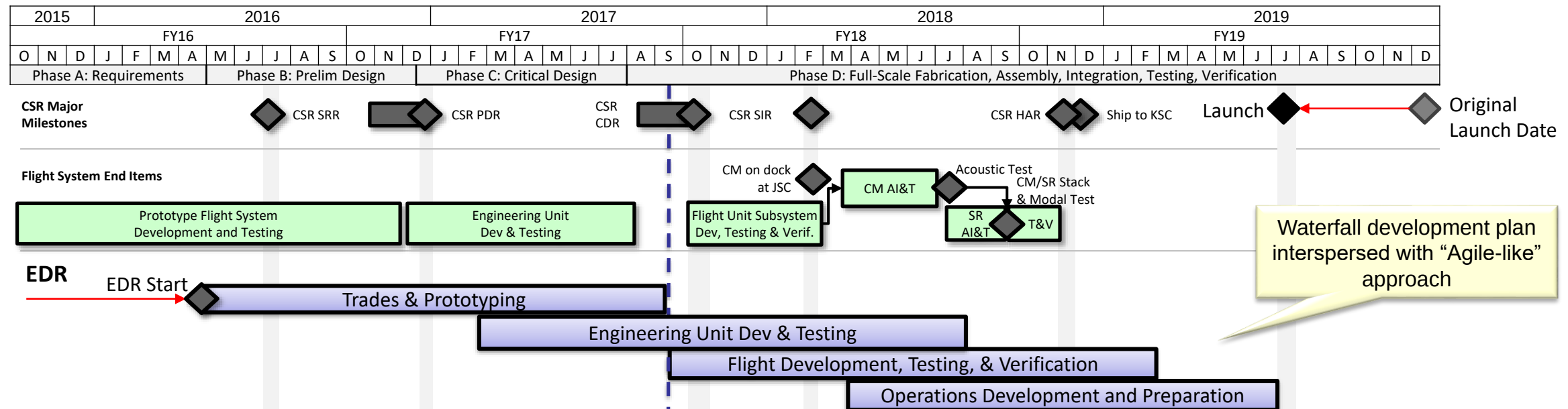
SE Approach: Iterative Development - Computer Packaging Example



Schedule

Agile principles:

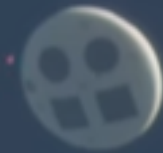
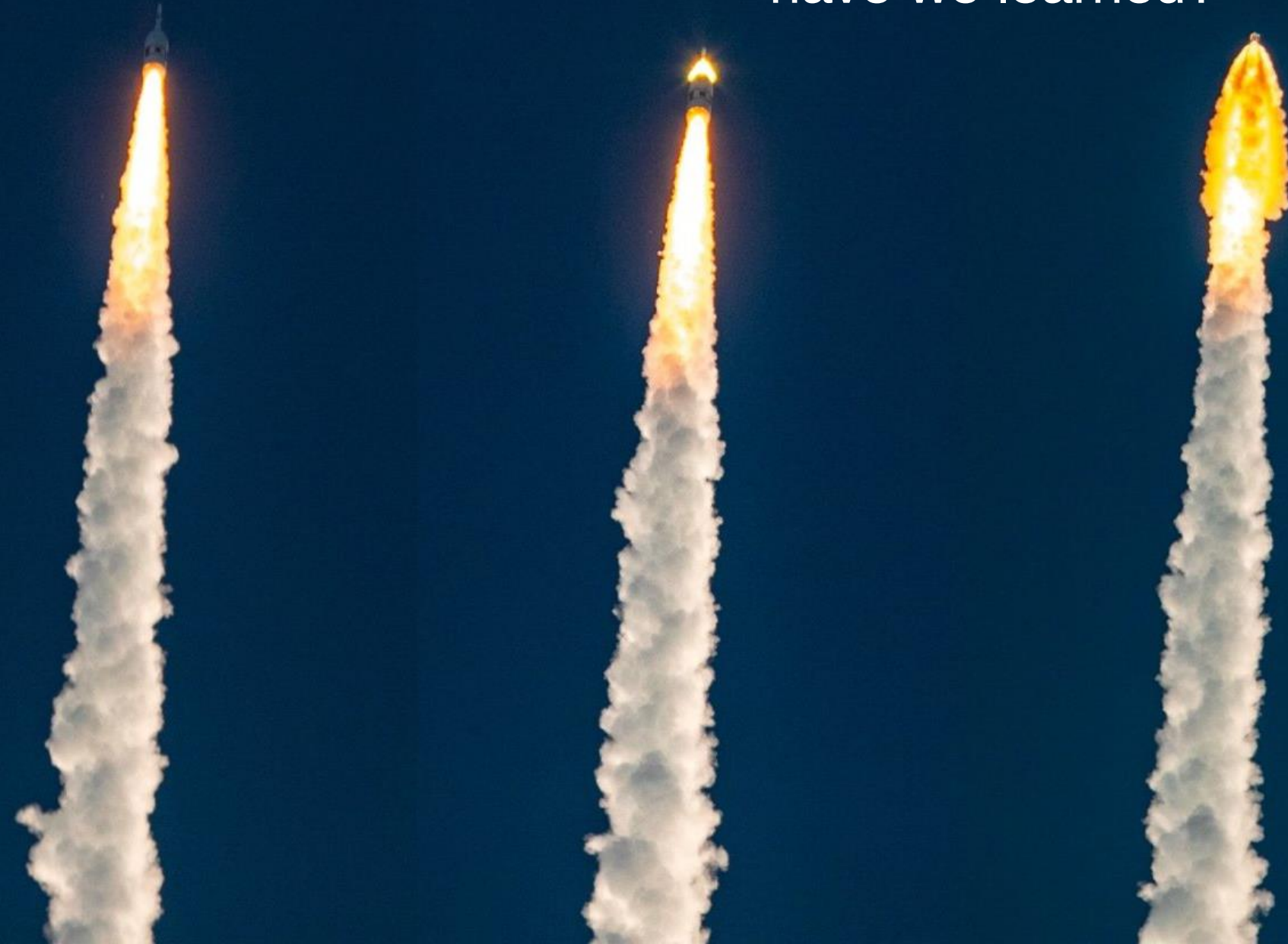
- Highest priority is to satisfy the customer through **early and continuous delivery** of valuable components.
- **Deliver working components frequently**, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
- Agile processes promote sustainable development. The sponsors, developers, and users should be able to **maintain a constant pace indefinitely**.



HW	Development	EU	Flight/Spare
Payload	30	24	32
Breakout Box	1	1	3
Ejection System	0	1	2

- Payload switchboard complexity led to issues and late re-design
- Challenge to finish payload flight hw build and prep for flight ops at the same time

How does the EDR experience compare with Agile and what have we learned?



Agile Manifesto Core Values – EDR Self Assessment

Agile Values	EDR Self Assessment	Discussion
Individuals and interactions over processes and tools	Meet	• Focused on having the right skill set and right team mix • Some challenges with this, but overall had a great team • Co-located team to facilitate team interaction • Provided multiple communication methods to foster communication • New processes and tools were developed/used to efficiently support the team
Working hardware over comprehensive documentation	Meet	• Team focus was hardware development and testing • Example: Assembly details folded into drawings, procedures referenced the drawings • Example: Status/testing details frequently provided via email/IM/text -> cleaned up and copied into SharePoint lists for various documentation needs (activity reports, discrepancy reports, testing log, etc.). Wikis became “living” documents that pulled directly from the authoritative source (database).
Customer collaboration over contract negotiation	Meet	• EDR ground rules included fostering a “badgeless” team that was fully collaborative • No blaming/finger pointing
Responding to change over following a plan	Meet	• General plan existed • Worked with CSR PM to find flexibilities as the EDR plan changed

Challenges Experienced with the EDR Approach

- External interfaces to the EDR team
 - Were uncomfortable with the new approach
 - Were not used to the speed at which we were moving to get tasks accomplished
- Range of skill sets needed for the EDR team
 - Need a larger pool of personnel with an array of skills and experience levels to pull from
 - Distributed labs and facilities
 - Lack of flight experience
- Balancing “good enough” with “making it better”

Lessons Learned: Agile-Like HW Development

Category	Principle	Future Improvements
Team	Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.	- Establish a multi-functional lab and office environment with dedicated technicians and engineers who are co-located and trained in an Agile-Like hardware development - Increase the use of video conferencing when not able to be in-person - Consistently evaluate the team to ensure the right mix of skills and personalities Budget personnel and PMs should be more closely integrated - More streamlined process for understanding testing and manufacturing costs from various labs
	The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.	
	The best architectures, requirements, and designs emerge from self-organizing teams.	
	Business people and developers must work together daily throughout the project.	
Team Effectiveness	At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.	

Lessons Learned: Agile-Like HW Development

Category	Principle	Future Improvements
Technical	Working components are the primary measure of progress.	- Up front focus on ensuring the right personnel/skills mix is available for the job - More efficient test and verification process - Continue to build upon the DDCIM approach and making the Systems Engineering process more efficient - Unclear whether continuous changing requirements is truly feasible for hardware development
	Continuous attention to technical excellence and good design enhances agility.	
	Simplicity--the art of maximizing the amount of work not done--is essential.	
	Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.	
Schedule	Highest priority is to satisfy the customer through early and continuous delivery of valuable components	- Spend more planning time breaking the hardware development into modular components that flow from functions - Develop schedule based upon the functions and continuously building upon demonstrated functionality (i.e. Scrum Product Backlog) - Ensuring the right personnel/skills mix is available for the job → if not, schedule should be adjusted accordingly to account for training and learning curve of personnel - Continuous improvement of processes through lessons learned evaluations
	Deliver working components frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.	
	Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.	

Other Recommendation for Future Agile-like Hardware Development

- Understand the level of rigor required for the project and whether an Agile-like approach is appropriate
- Up front coordination on the level of rigor required at various stages and the expectations associated with that from all entities involved (management, engineers, technicians, quality, safety, etc.)
 - Consider formalization of a new classification for projects similar to EDR (not human spaceflight, not safety critical, but still flight hardware)
- Review of support services and processes to understand where efficiencies can occur (i.e. shipping, procurement, etc.)
- Provide teams with autonomy and the ability to take ownership of their work
- Management puts responsibility on their people and trusts their people, rather than the process
 - Requires up front, careful consideration of the team and mix of skills

Questions



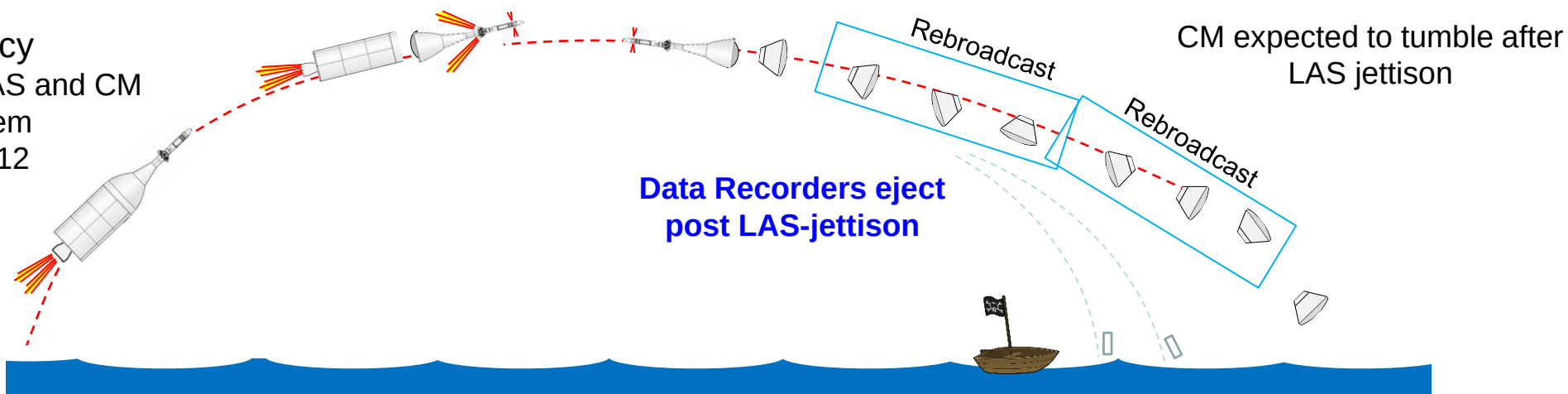
Kristina Rojdev
Kristina.Rojdev-1@nasa.gov

Backup

Drivers: Data Retrieval ConOps Overview

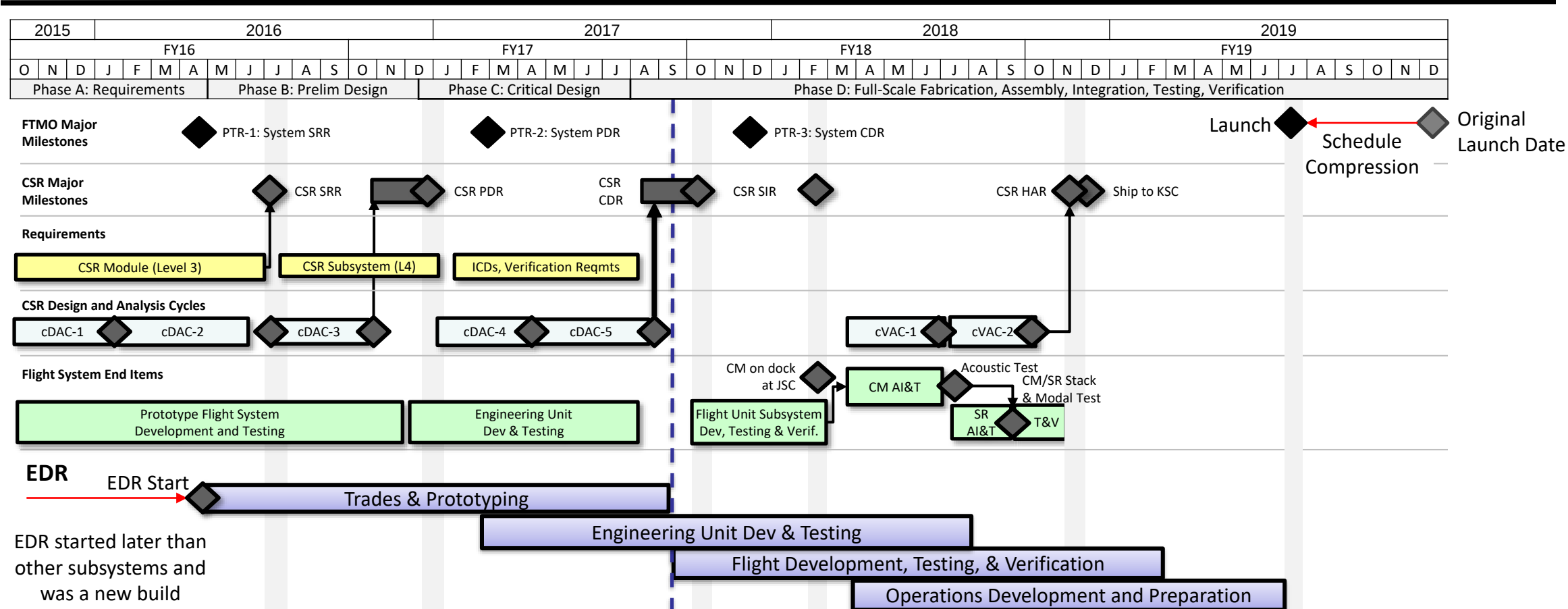
System Data Redundancy

- Multiple antennae on LAS and CM
- Buffer rebroadcast system
- EDR with 2 strings and 12 recorders



	Ascent	Abort and Reorientation	LAS Jettison-Freefall	Post-Splashdown
Activities	Launch and ascent	Abort condition met, AM burnout, LAV reorientation	LAV settling complete, LAS jettison initiated, free fall to splashdown	n/a
Data Broadcast	All DFI/OFI	All DFI/OFI	Rebroadcast of critical data until splashdown	n/a
Data recording	All DFI/OFI recorded	All DFI/OFI recorded	All DFI/OFI recorded until ejection at staggered time. 12 EDRs eject in 10s sequence and land in the water. Beacon receives GPS location and Iridium broadcasts position.	Ship recovers EDRs floating in the water.

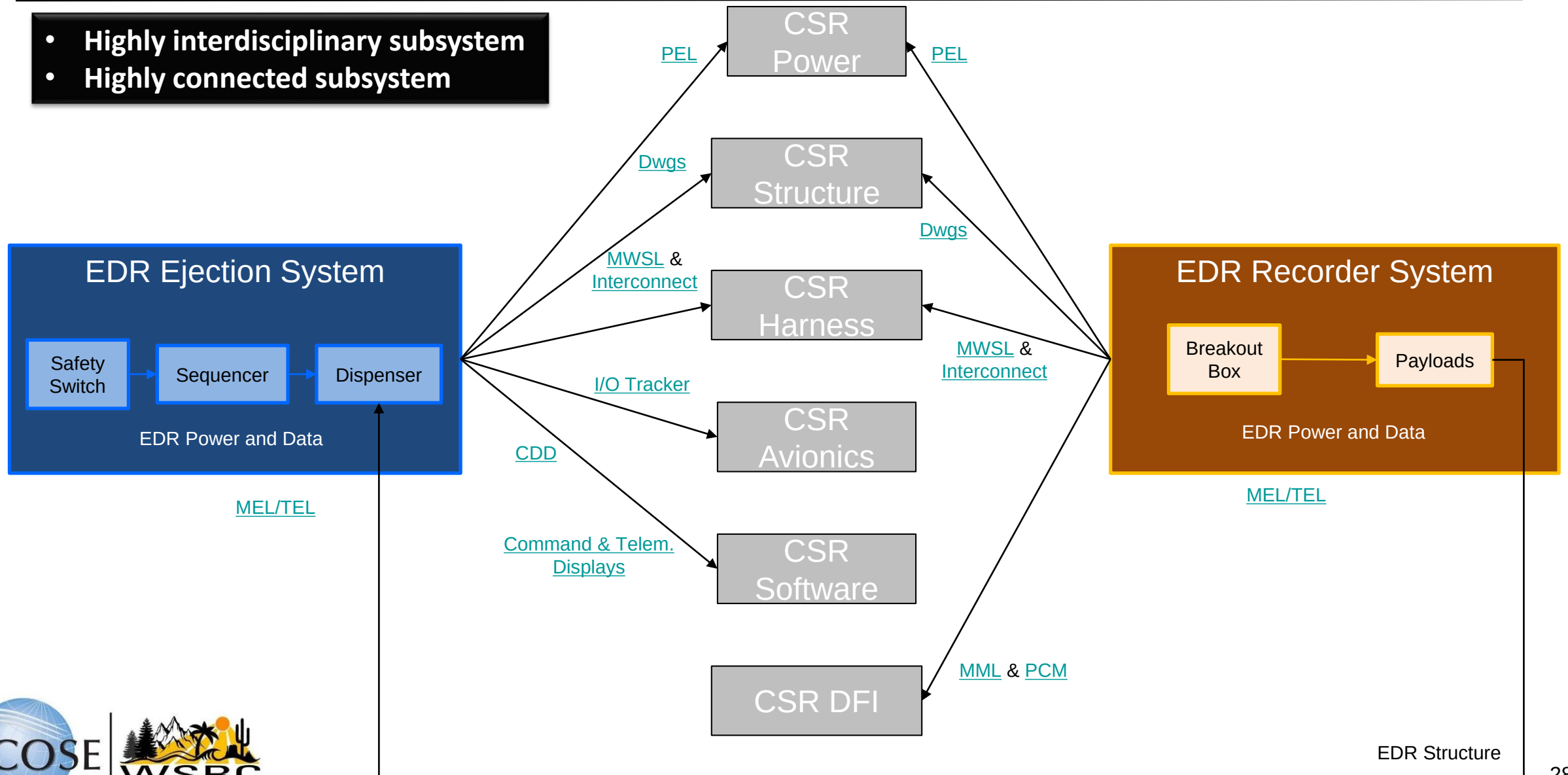
Drivers: Schedule



Driver: EDR development schedule highly compressed compared to other CSR subsystems.

Background: Internal and External Interfaces

- Highly interdisciplinary subsystem
- Highly connected subsystem



Management Approach and Team Expectations

- Ground Rules
 - Communication is key
 - Participation is mandatory
 - Everyone wears multiple hats
 - Respect and trust one another
- Challenges with this approach
 - Members with other assignments were not always in sync with fast moving development
 - Coaching required to span the gap for multiple hat approach vs classical stove-pipe development approach
 - Members outside of EDR with which we needed to engage (i.e. shipping, procurement, quality, etc.)



Management Approach: Co-Location

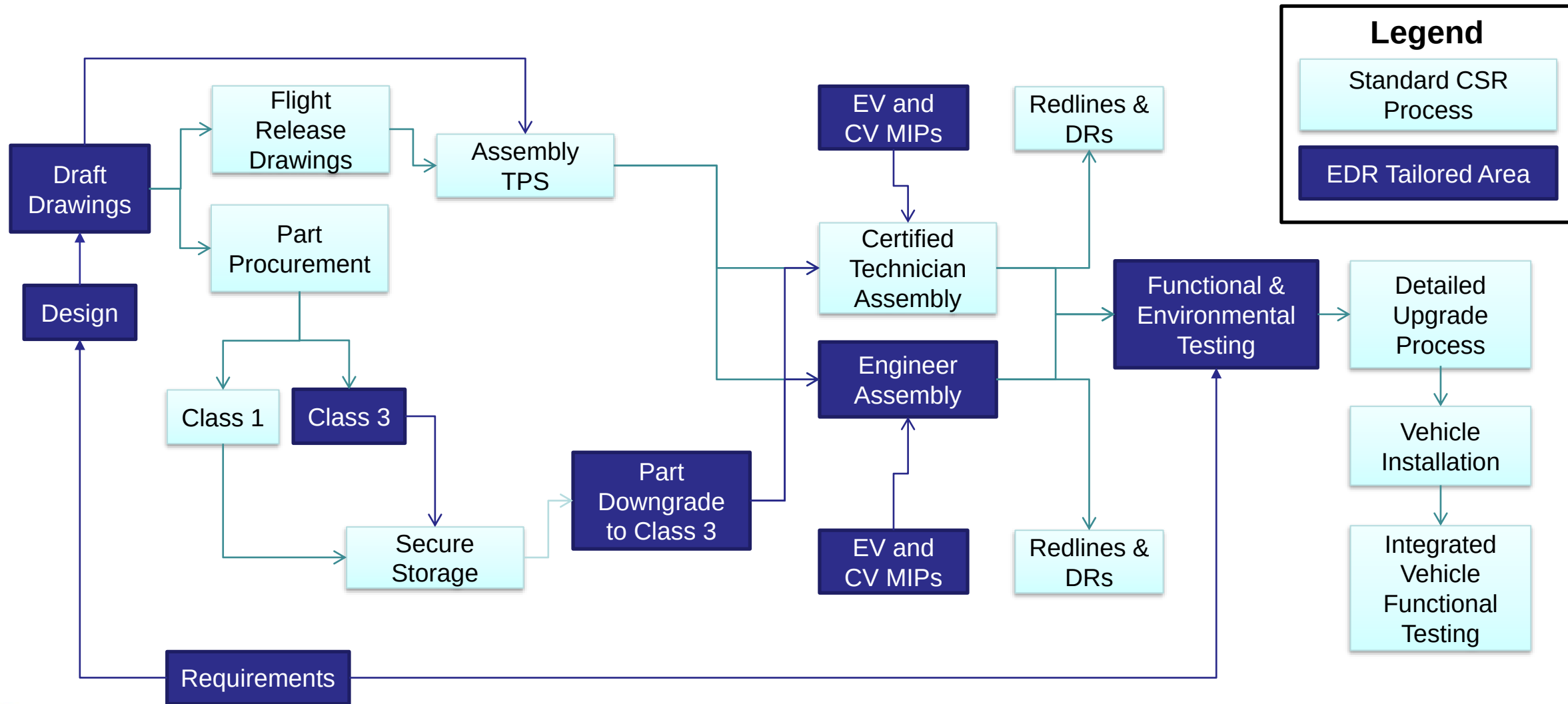
■ Challenges:

- EDR was a multi-faceted project
 - Required skills from different organizations
 - Required different types of labs
- Personnel were distributed across the center
 - Not having the team collocated decreased their effectiveness
 - Difficult to get some team members collocated because they were working multiple projects and their management didn't want to move them
- Work had to be performed in multiple labs
 - Hardware was distributed in multiple buildings, making it difficult to track
- EDR's main lab was shared with another project
 - Was not coordinated at the outset of the project
- With a majority of the team being contractors, moving people or hardware between the JSC buildings and the contractor buildings was onerous
- EDR Subsystem Lead was remote in California

■ Implementation:

- Majority of the team collocated in one building, which was the same location of the main lab
- Worked to get approval to do battery work in the secondary lab we used rather than having to use the battery lab
- SharePoint list for inventory tracking was created, but wasn't very effective
- EDR Subsystem Lead maintained strong communication with the whole team and traveled for major activities

SE Approach: EDR Process Tailoring



Scale of Rigor & Personnel Skill Set

Development

- Focus on key requirements
- Little to no documentation
- Freedom to experiment, build, and test
- Core engineers only (no technicians or process)
- Work quickly
- Demonstrate it works

Engineer Skills:

- Out of the box thinking
- Experimental
- Diverse experience

Engineering Unit

- Add more ancillary requirements to the design
- Investigate edge cases
- Engineer works with technician to build and document as they go to build procedures
 - Start with a napkin sketch and draft drawings
- Minimal documentation of discrepancies and changes (paragraph in SharePoint)
 - No management oversight/signatories required
- Medium Speed

Engineer Skills:

- Experimental
- Collaborative
- Organized

Flight

- Full verification of requirements
- Technicians assembly of hw with eTPS and drawings
 - eTPS only required subsystem lead signature
 - Engineers could support hw assembly
- Documentation of any discrepancies
 - Signature of Func. Int. Lead
- Review board required for design changes
- Slow speed

Engineer Skills:

- Collaborative
- Very organized
- Attention to detail
- Patient

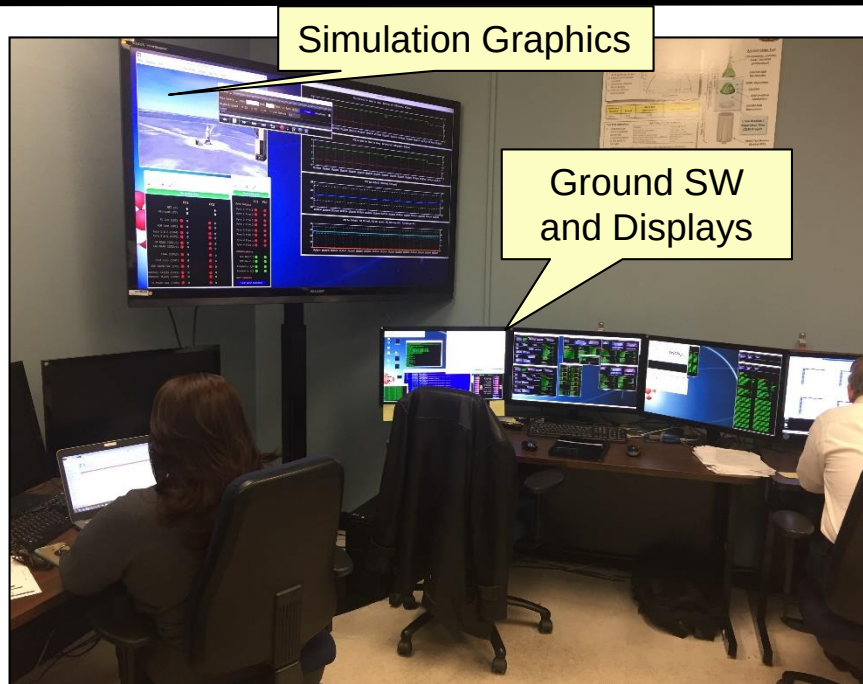
SE Approach: Engineering Unit & Flight Unit Phases

Test	EU BOB	EU Dispenser	EU Seq/SS	EU/Qual Payload	FI/Sp BOB	FI/Sp Dispenser	FI/Sp Seq/SS	FI/Sp Payload
EMI Testing	Yes	Yes	Yes	No	No	No	No	No
Ejection Testing	Yes	Yes	Yes	Yes	No	No	No	No
Drop Testing	No	No	No	Yes	No	No	No	No
Pre-Launch Ops Test	Yes	No	No	No	Yes	No	No	No
Piece Part Burn-In Testing	No	No	No	No	No	No	No	Yes
Functional Testing	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Thermal Testing	No	No	No	No	Yes	No	No	Yes*
Vibe Testing	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Stress Analysis	No	No	No	No	Yes	Yes	Yes	No

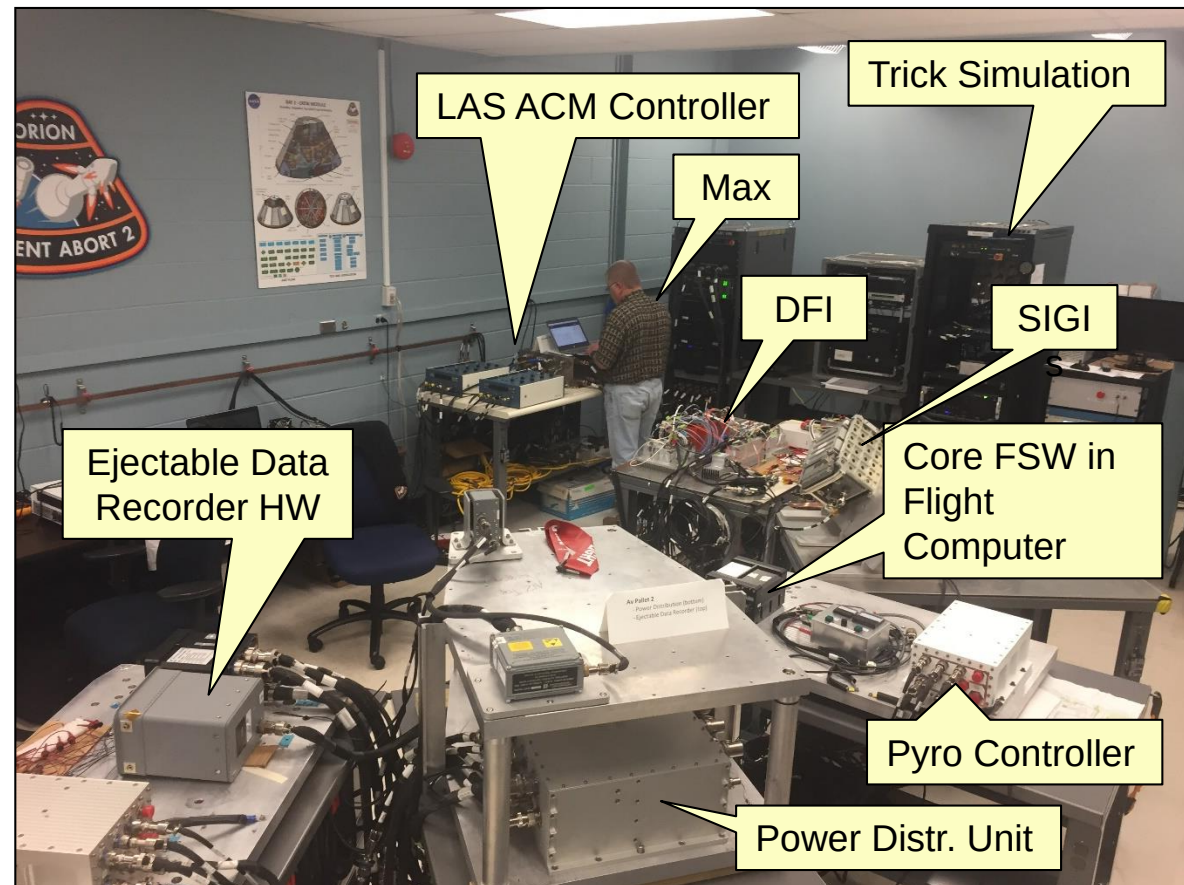
*only some components thermal tested

- Heavy testing on the EU to remove areas of concern
- Focus testing on those environments of most concern
- Heavy screening of COTS components that we could not verify quality (i.e. CofCs)

SE Approach: Integrated Vehicle Testing

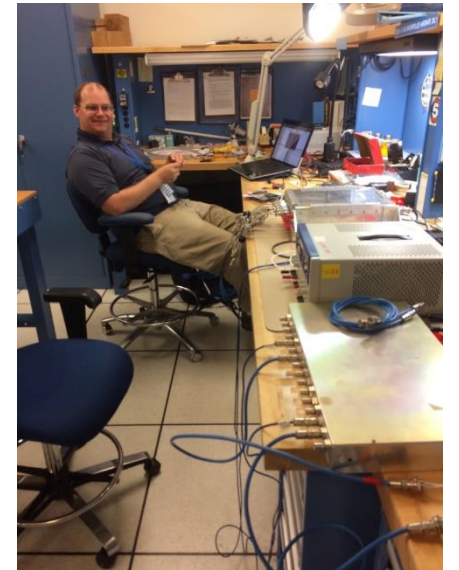


- The CSR Avionics and Software Testing Laboratory Environment (CASTLE) provided a high-fidelity, closed-loop hardware-in-the-loop test facility / mission training capability using JSC MCC
- CASTLE provided risk reduction through early and frequent hardware-software integrated testing



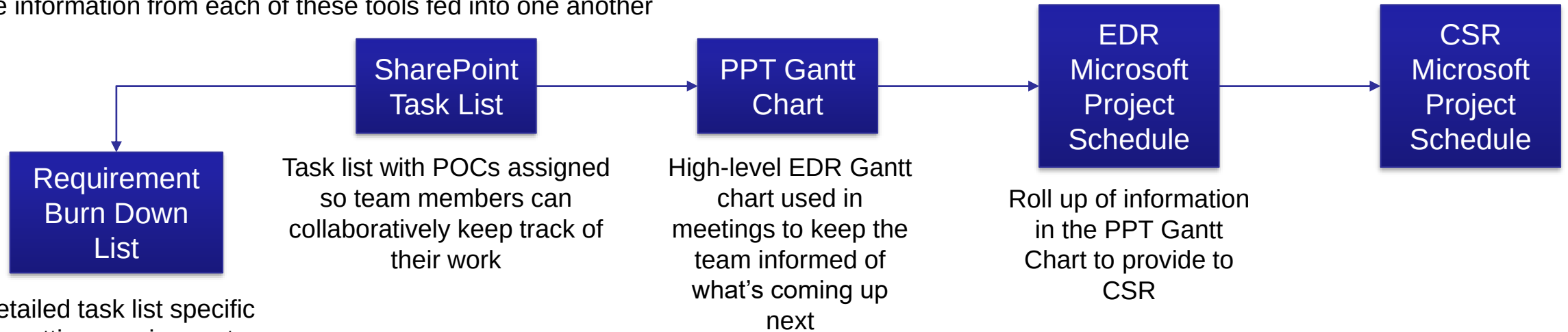
Tools

- Custom Designed SW Tools
 - EDR Data Comparison Scripts
 - Pre-Launch Ops Test Script
 - USB Burn in Test
 - Squib Test Set SW
 - Beacon Tracking SW
- SharePoint: EDR Custom Lists
 - Task List
 - Requirements
 - Procedure Tracking
 - Drawing Tracking
 - Test Log
 - Hardware Tracking
 - Inventory Tracking
 - Tracking of Piece Part Screening
 - EDR Upgrades

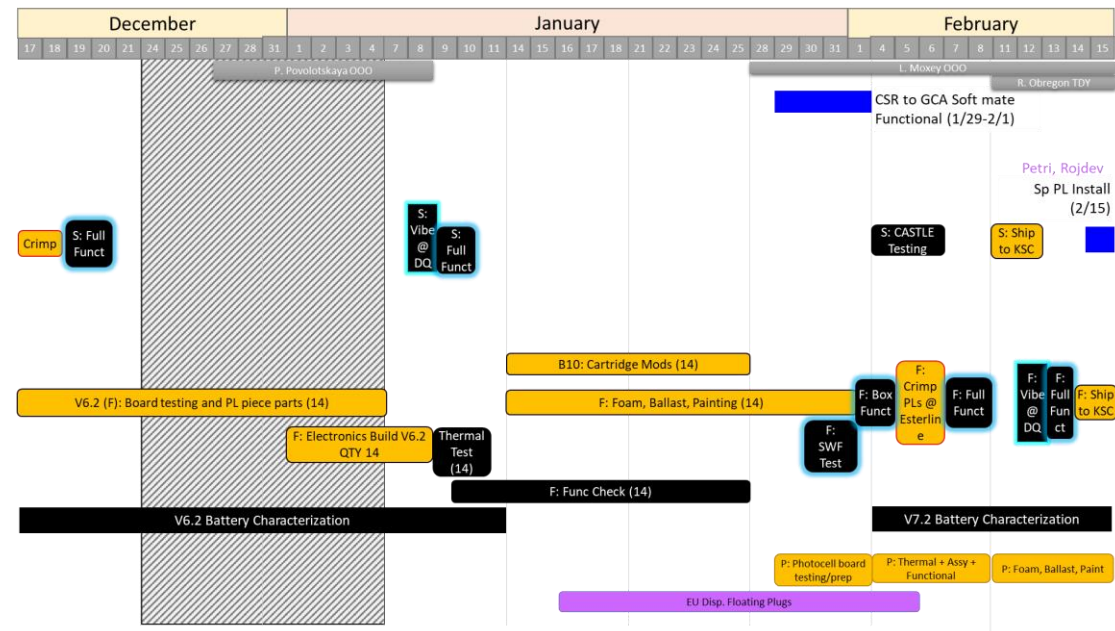


Tools: Task and Schedule Tracking (Example)

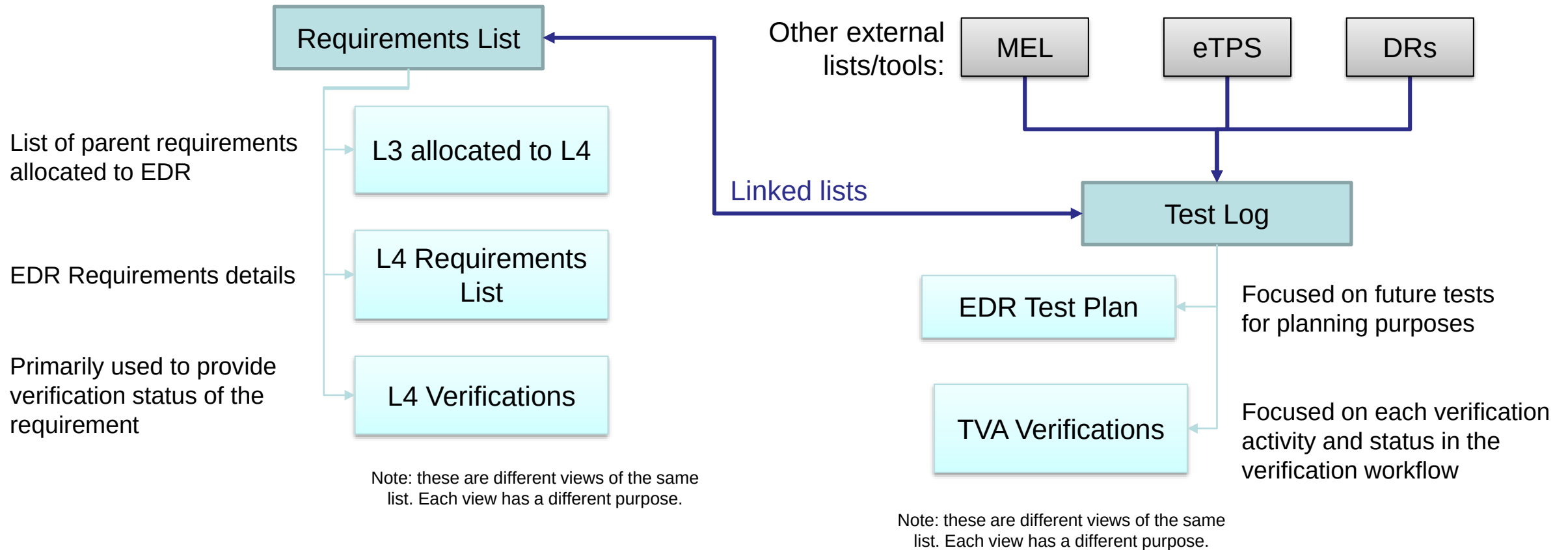
The information from each of these tools fed into one another



Title	Description	Assigned To	POC	Status
Battery Risks	Put together a list of battery risks, their likelihood, and potential mitigations. Have this ready for discussion at the EDR tagup on 2/28/2017.	Deoja	Deoja	Completed
Payload Parts List	Pull together the current plan for the payload parts list. Include vendors and component model numbers.	Deoja; Rojdev	Deoja; Rojdev	Completed
EMI/EMC for the power and data distribution box	3/7/2017: Tom working on getting Christina the part #s. Work with Bob Scully to review the EDR power/data distribution design and provide recommendations before we begin building the EU.	Barbour	Barbour	Completed
WOW Switch Removal - ConOps for flight and ground testing	3/7/2017: Bob Scully didn't like the Vicos because they can be noisy. Garrett redid the power board and currently getting quotes on them. Garrett will discuss with Bob as to whether we need to do separate testing or if the design is good enough to proceed with integrated EMI testing. 3/8/2017: The path forward for EDR is to not do any component level EMI testing and only be a part of the integrated EMI test. Work through understanding the ConOps for flight and ground testing without the WOW switch. Walk through this with EDR, PM, SEI, SW, and Safety to make sure we are all on the same page and there aren't any safety risks with removing the WOW switch. 3/7/2017: Kristina working on finding a time that everyone is available to discuss this in more detail. 3/8/2017: the countdown/countup test was discussed at SEIWG and recommendation was put forward. We need to answer how we will verify that the squib power and signals are good without having the squibs in place. See slides. 3/9/2017: The plan is to use fuses to simulate a squib for the firing simulator. The idea is that when we send a firing command to the sequencer, the firing current will go through the	Rojdev	Rojdev	Completed



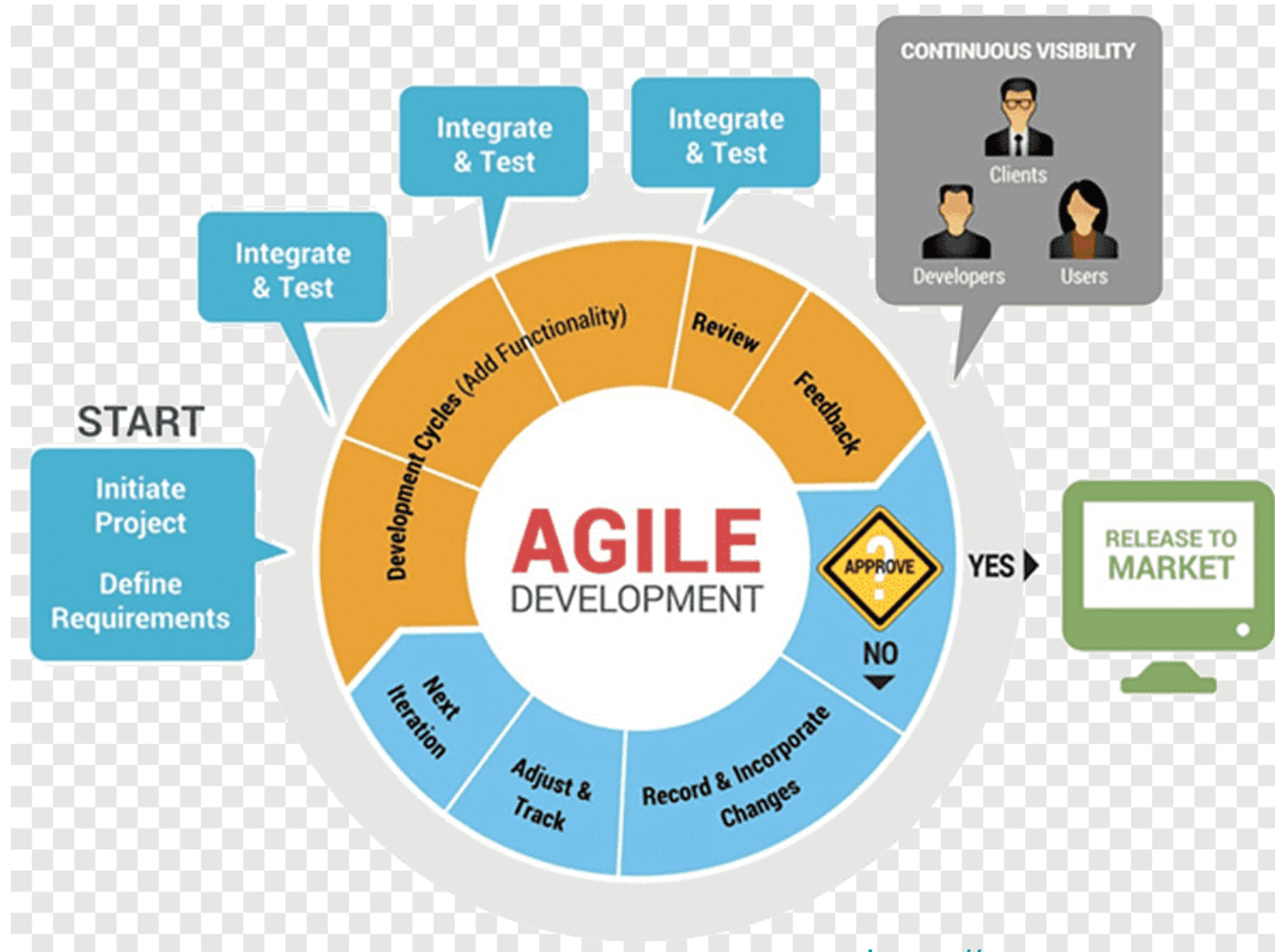
Tools: SharePoint Requirements Management (Example)



Why are we comparing with Agile?

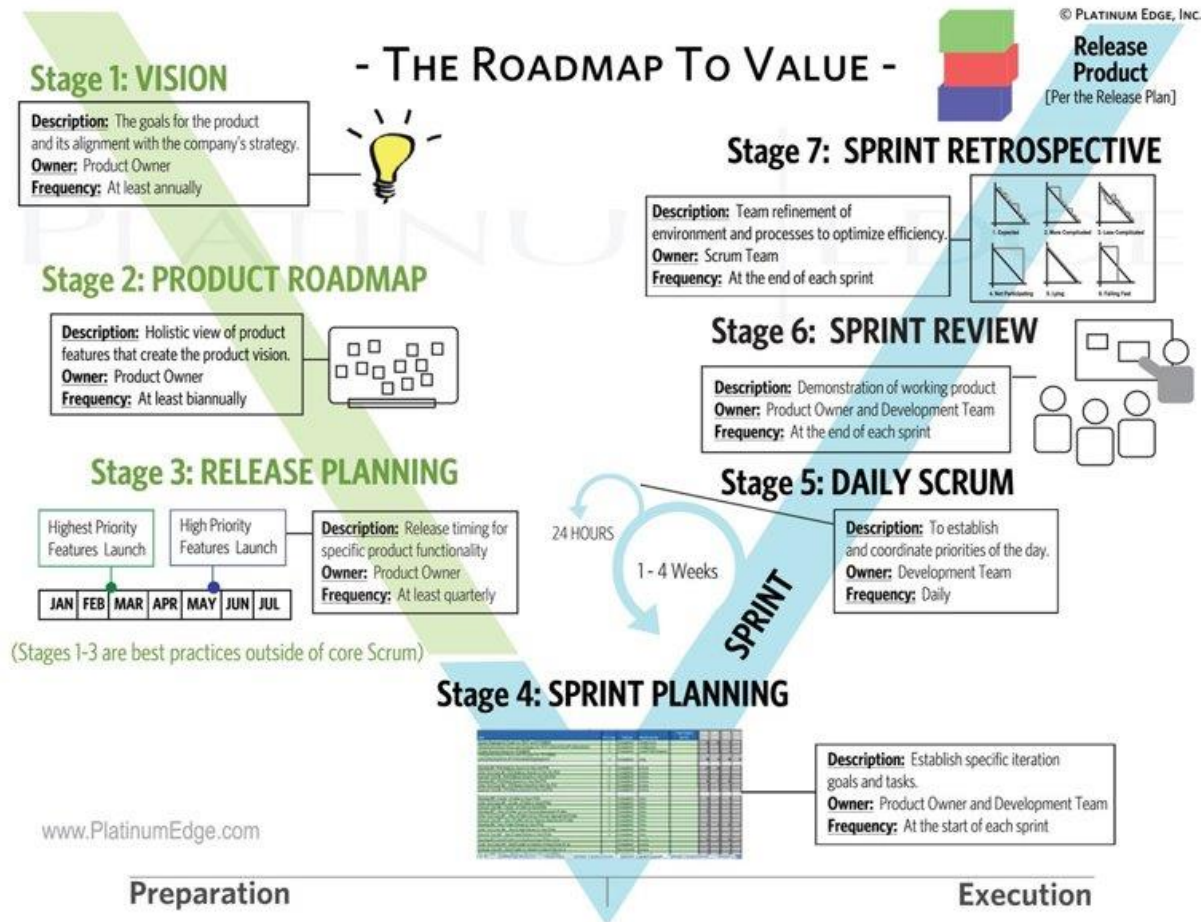
- SE community has recognized need for more efficient systems engineering as systems complexity increases and requirements stability decreases (Dove, et al, 2019)
- INCOSE's Agile for Systems Engineering efforts have produced a number of papers but not (we think) a framework that could be readily applied
- Finding a way to more efficiently build hardware is needed
- Found many similarities with our EDR experience and the Agile approach/Scrum Framework

Agile Software Development Methodology



<https://www.pngwave.com/png-clip-art-xjasp>

The Platinum Edge Agile Roadmap to Value



- In **Stage 1**, the [product owner](#) identifies the [product vision](#).
- In **Stage 2**, the product owner creates a [product roadmap](#).
- In **Stage 3**, the product owner creates a [release plan](#).
- In **Stage 4**, the product owner, the [scrum master](#), and the [development team](#) plan sprints, also called iterations, and start creating the product within those sprints.
- In **Stage 5**, during each sprint, the development team has [daily scrum meetings](#).
- In **Stage 6**, the team holds a [sprint review](#).
- In **Stage 7**, the team holds a [sprint retrospective](#).
- (See backup for details)

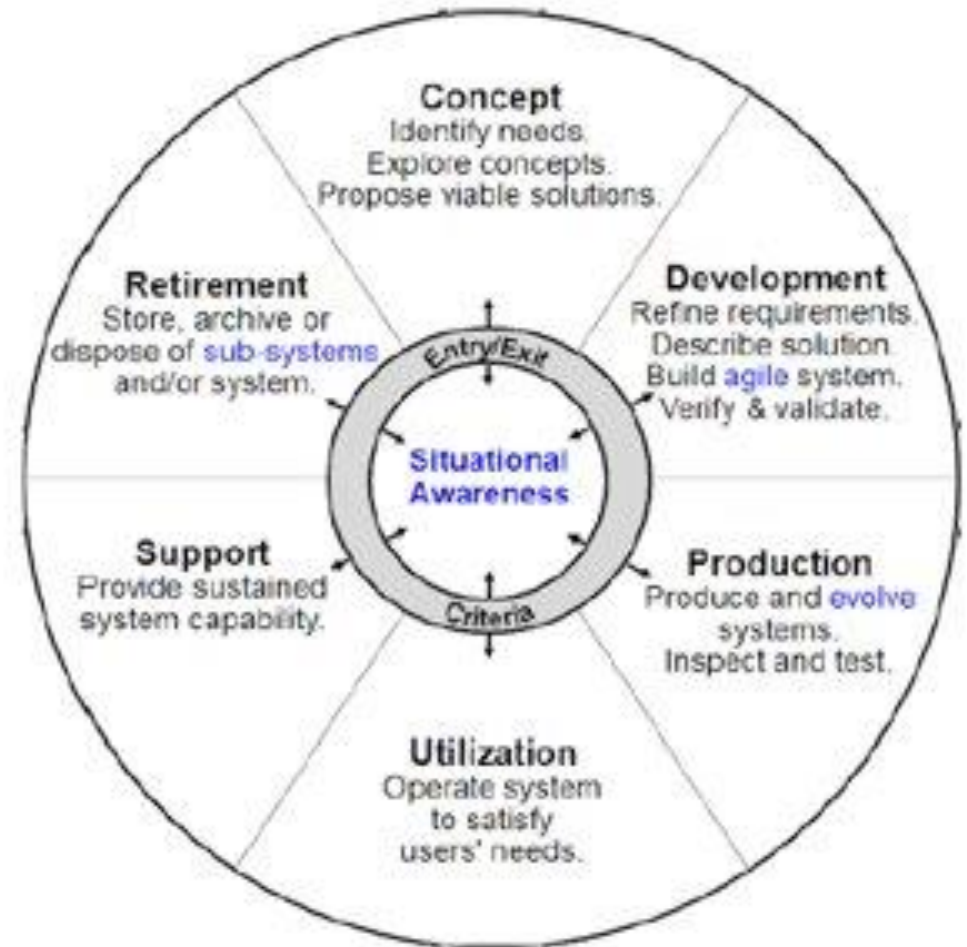
CURVE – Agile Systems Engineering

- Caprice: Unknowable situations. Unanticipated system-environment change.
- Uncertainty: Randomness with unknowable probabilities. Kinetic and potential forces present in the system.
- • Risk: Randomness with knowable probabilities. Relevance of current system-dynamics un-derstandings.
- Variation: Knowable variables and associated variance ranges. Temporal excursions on ex-isting behavior attractors (a reference to complex system behavior trajectories).
- Evolution: Gradual successive developments. Experimentation and natural selection at work.

Agile Systems Engineering Life Cycle Model for Mixed
Discipline Engineering, Rick Dove & Bill Schindel
Copyright © 2019 by Rick Dove and Bill Schindel.
Permission granted to INCOSE to publish and use.

Agile Systems Engineering Life Cycle Model Framework

- Rather than the sequential Waterfall Model, the ASELCM is asynchronous with activities occurring in some or all processes at the same time



Agile Principles – EDR Self Assessment

Principle	EDR Self Assessment	Discussion
Highest priority is to satisfy the customer through early and continuous delivery of valuable components	Meet	- Demonstrated capabilities and improvements throughout - Continuously delivered payloads with upgraded capability - Delivered support test equipment and software to increase fidelity of testing results
Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.	Meet	Requirements were evaluated continuously, based on customer needs and hardware capabilities
Deliver working components frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.	Meet	Distributed EDR architecture allowed for swapping out components on a frequent basis
Business people and developers must work together daily throughout the project.	Meet	- Budgetary function resided with the subsystem lead - Lead ensured that funding was available when needed to get the work completed quickly
Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.	Meet	- Focus on having the right team mix (highly motivated, willing to wear multiple hats, willing to learn new skills, etc.) - Did have a few cases where the skills mix and personality was not a fit for this team - Each individual was expected to take ownership and responsibility of their tasks, and any products assigned to him/her
The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.	Meet	- Push for co-location of the team - Consolidation of required labs to support colocation and minimize travel time

Agile Principles – EDR Self Assessment (Cont.)

Principle	EDR Self Assessment	Discussion
Working components are the primary measure of progress.	Meet	- Success was focused on components meeting the intended function - With respect to recovery, we only needed to recover 1 of 12 payloads
Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.	Needs Work	For the schedule constraints on EDR, we needed very experienced people who could come up with high-quality, working solutions quickly. This was not always feasible and led to instances where we had to do a lot of troubleshooting. Thus, the team struggled with meeting all of their responsibilities and we came close to burnout in some instances.
Continuous attention to technical excellence and good design enhances agility.	Meet	- Overall, there was good design and technical excellence, which allowed us to be agile in overcoming some of the challenges we had - Needed to balance “good enough” with “making it better” in some cases - In other cases, the level of quality/technical excellence needed to be increased
Simplicity--the art of maximizing the amount of work not done--is essential.	Meet	- Constantly worked to come up with the simplest design to meet the functions - In one case where an incorrect assumption was made that increased design complexity, there were numerous problems with the design
The best architectures, requirements, and designs emerge from self-organizing teams.	Meet	Team started with the main functions and let that drive the design
At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.	Meet	- Throughout the lifecycle, new approaches were tried and vetted - Lessons learned list that was used throughout the project to document

