Alabama Space Grant Consortium
Alabama CubeSat Initiative

ABEX
Alabama Burst Energetics eXplorer

Intro to Systems Engineering

Systems Engineering Primer for New Spacecraft Engineers

University of Alabama in Huntsville – Management Team
Michael Halvorson

Document Classification: *Public*, Distribution: *Unlimited*

Agenda

1. Who is a Systems Engineer?
2. Design Life Cycles
3. Stakeholders
4. ConOps & OpsCon
5. Requirements
6. Technology Assessment
7. Assembly & Integration
8. Verification & Validation
9. Technical Performance Measurement
10. Risk Assessment
11. Summary

Who is a Systems Engineer?

What a Systems Engineer Does

- Systems Engineers architect, design, and manage complex systems
  - Looking at the whole system from a top-down, interdisciplinary approach

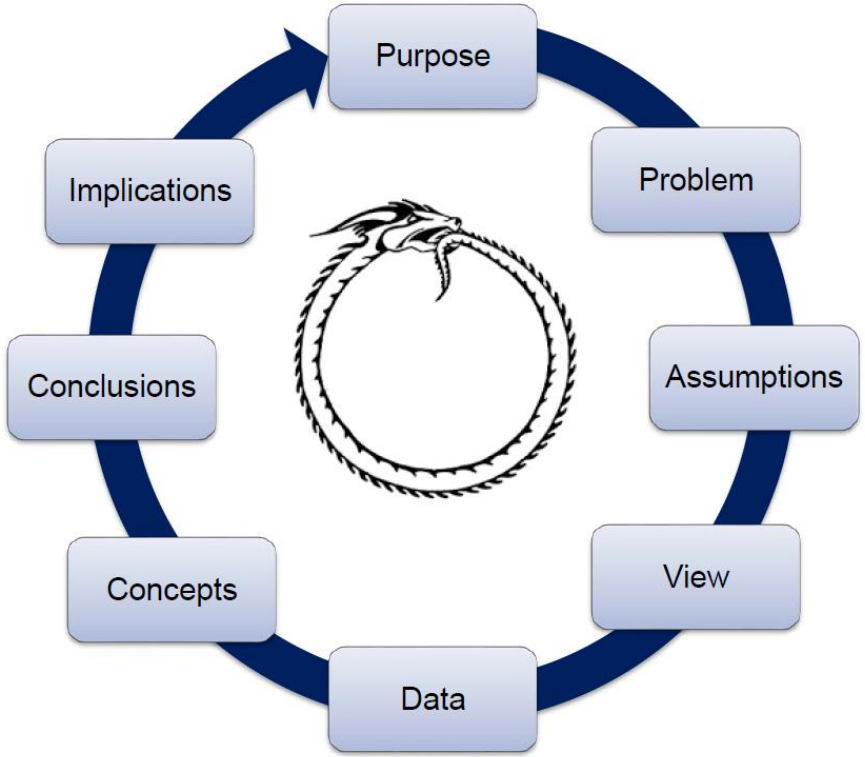
- Every system, especially complex systems, has a life cycle
  - Systems Engineering tasks change throughout that life cycle
  - Life cycles are different for many disciplines, and sometimes they must mix & match within a single program.

- Often, Systems Engineers must define problems before solving them
  - What is important about the system?
  - What is important about the system right now?

What is Systems Engineering?

- Formal Definition:
  - "The multi-disciplined application of analytical, mathematical, and scientific principles for formulating, selecting, developing, and maturing an optimal solution from a set of viable candidates that has acceptable risk, satisfies User operational need(s), and minimizes development and life cycle costs while balancing Stakeholder interests [1]."

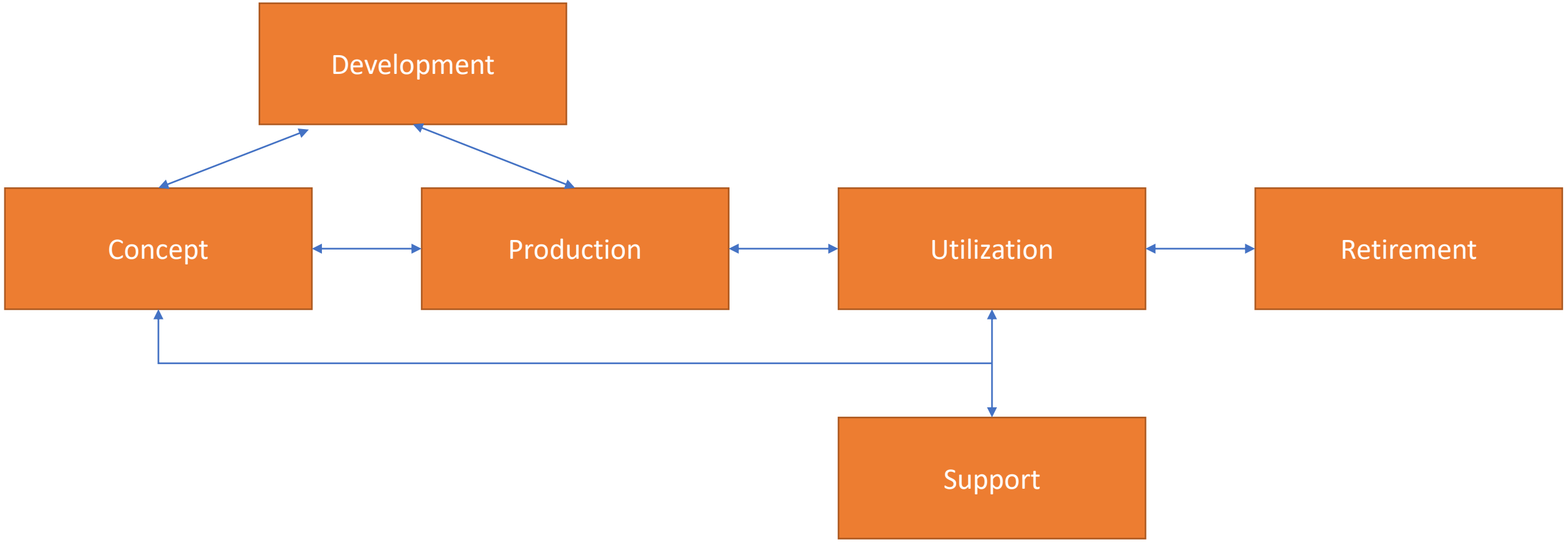
    - Defined principles
    - Solution-oriented
    - Problem and solution space exploration
    - Risk assessment
    - User operational needs met
    - Stakeholder interests represented



[1] INCOSE, 2015

It's Not a Linear Process, and It's not Supposed to Be

- Iteration and recursion possible on all paths



[1] INCOSE, 2015

What is the utility of Systems Engineering?

- SE is not about producing documentation; **it's about decision making**

- SE replaces linear processes with interdependent steps

Grade School

1. State a question
2. Do background research
3. Formulate hypothesis
4. Design experiment
5. Test hypothesis
6. Analyze results/draw conclusions
7. Communicate results

Scientific Method, sort of

Undergrad Engineering

1. Define the problem
2. Do background research
3. Specify requirements
4. Create alternative solutions, select one
5. **Build a prototype**
6. **Test & redesign as necessary**
7. Communicate results

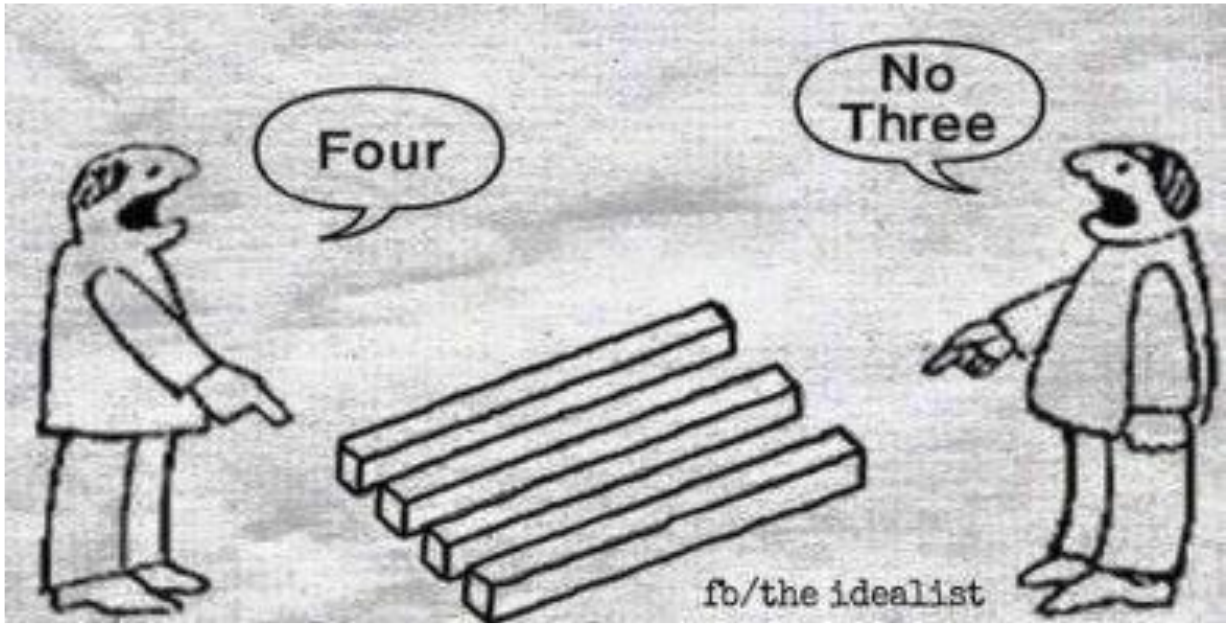
Specify-Design-Build-Test-Fix

Professional Systems Engineering

1. Understand Problem/Opportunity Space
   - Stakeholder needs, Operational Concept
2. Develop Domain Solutions
   - Requirements, Operations, Behavioral, Physical
3. Define System Architecture from Solution Space
4. Define System Design
   - Architecture is what and why. Design is how
   - Design is Verified & Validated against reqs.
5. Assemble, Integrate, Implement
6. Operate, Maintain, Support
7. Retire System

Okay, So What Is a System?

- A system is an integrated set of elements that accomplish a defined objective
  - Elements:
    - Products, processes, people, information, techniques, facilities, services, support
  - Four Domains in any System:
    - Requirements
    - Operations
    - Behavioral
    - Physical



- Any domain can be seen differently from differing perspectives
- If you want to be a manager or executive, you need *Systems Thinking*

Elements of Systems Thinking

- Focus on the System itself, starting with output:
  - What should the system do?
  - What is desired by customers and stakeholders?

- …then work backwards to identify system design
  - What actions transform system inputs to desired outputs?
  - What system configuration maximizes value?

- Remember, it's not a linear process, and it's not supposed to be

Who is a Systems Engineer?

- **Every engineer is a Systems Engineer**
  - There is rigorous theory behind Systems Engineering, and ignoring it will cost you time & money

- The optimal amount of budget allocation for SE in a project is 14.4%
  - Spending $10,000,000 on a project? Put $1,440,000 towards SE
  - Any lower or higher than that regularly results in cost or schedule overruns

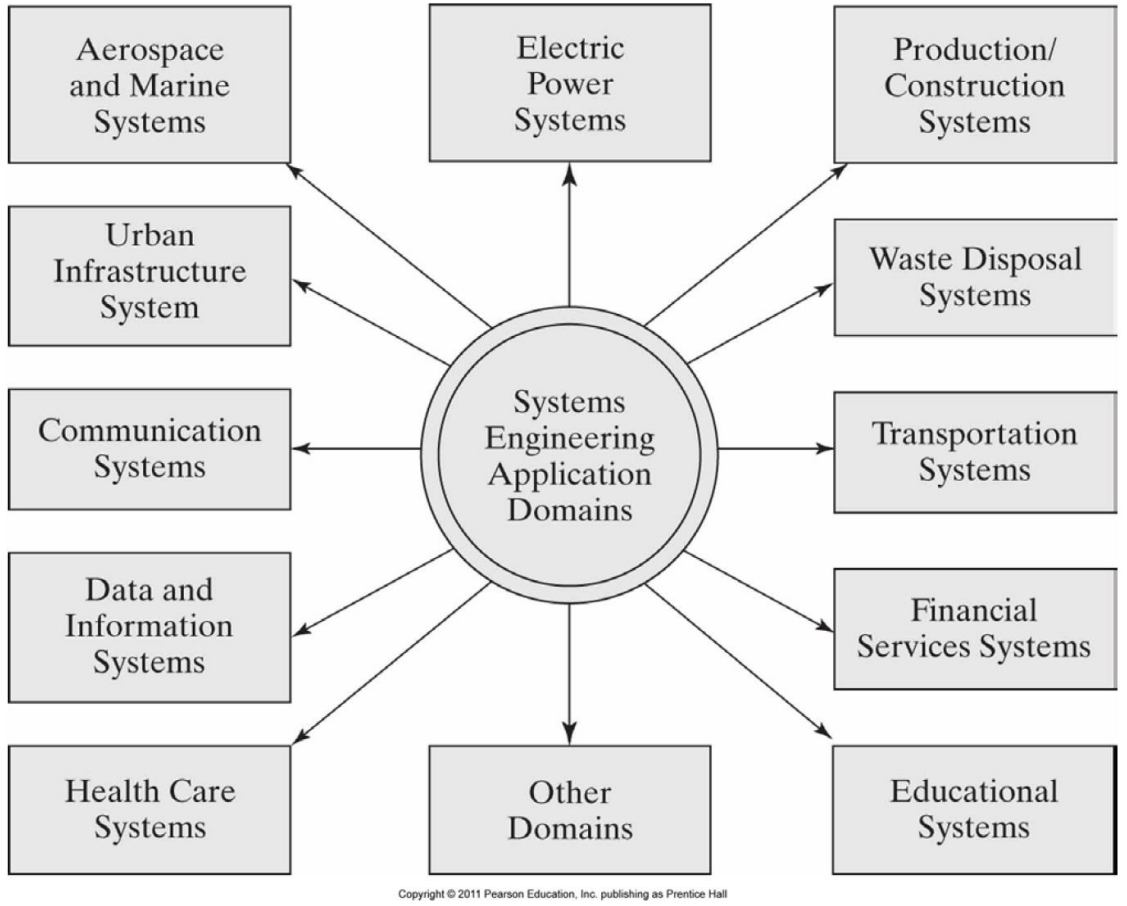
[2] Honour, 2013

Design Life Cycles

Document Classification: *Public,* Distribution: *Unlimited*

Design Life Cycles

- There's not just one correct life cycle
- Depends on the system under development
  - Usually defined by the Domain
- Life cycles have a Stage-Gate organization
  - Example shown in two slides
- There might be multiple Life Cycles within a single project
  - Software doesn't have a production stage, but many systems have software



[3] Blanchard & Fabrycky, 2011

Life Cycle Segmentation by Domain

- Phases can exist within these
- Software doesn't have a production segment

System	Concept	Development	Production	Utilization	Support	Retirement
Software	Concept	Development		Operation & maintenance		Retirement
Hardware	Concept	Design	Fabrication	Operation & maintenance		Retirement
Services	Service strategy	Service design	Service transition	Service operation	Continual service improvement	
Human	Skill needs definition	Acquisition	Training	Skills use & maturation		Retirement
Facility	Rendering	Structure & site design	Permitting	Construction	Operation & maint.	Retirement
Process	Output definition	Flow charting	Write-up	Pilot use	Use & improvement	Retirement
Natural entity	Acquisition	Development		Exploitation		Retirement

[4] ISO/IEC TR 24748-1 2010

Common Life Cycles

- DoD 5000
- NASA 7120.5F
- IEEE 1220
- ISO/IEC 24748-1
- Wasson Life Cycle

Concept → Development → Production → Utilization → Support → Retirement

System	Concept	Development	Production	Utilization	Support	Retirement

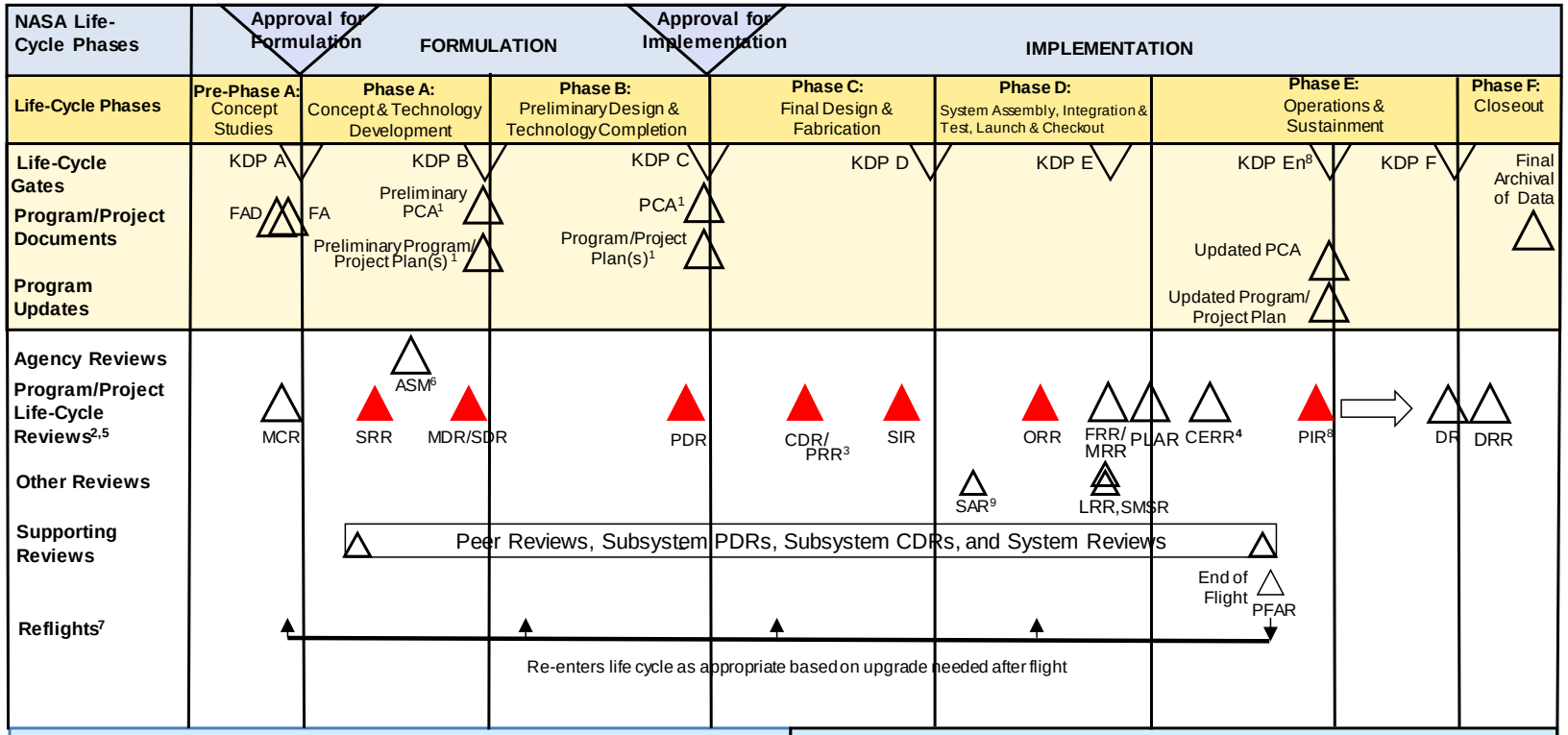
Software	Concept	Development	Operation & maintenance	Retirement

[5] Wasson, 2015, [6] Thomas, 2021

NASA Single Project Life Cycle

- Development divided into predetermined set of stages
- Entrance to each stage is a gate
- Gates are guarded by senior managers with authority
  - Results:
    - Go
    - Hold
    - Recycle
    - Kill

[7] NASA NPR 7123.1, 2020



FOOTNOTES

1. Program Plans and PCAs are baselined at KDP C. These are reviewed and updated, as required, to ensure program content, cost, and budget remain consistent. Program and Project Plans may be combined if approved by the MDAA.
2. Flexibility is allowed to the timing, number, and content of reviews as long as the equivalent information is provided at each KDP and the approach is fully documented in the Program/Project Plan(s).
3. PRR needed for multiple system copies. Timing is notional. PRR is not an SRB review.
4. CERRs are established at the discretion of Program Offices.
5. Life-cycle review objectives and expected maturity states for these reviews and the attendant KDPs are contained in Table 2-5.
6. Timing of the ASM is determined by the MDAA or AA, as compliant with NPD 1000.5 and may take place at any time during Phase A.
7. Placement of arrows is notional. See Section 2.2.4.3 for more guidance on reflights..
8. Once the program is in operations, PIRs are conducted as required by the NASA AA or the MDAA. KDP En follows the PIRs, i.e., KDP E2 would follow the first PIR, etc.
9. SAR generally applies to human space flight.

ACRONYMS

ASM—Acquisition Strategy Meeting
CDR—Critical Design Review
CERR—Critical Events Readiness Review
DR—Decommissioning Review
DRR—Disposal Readiness Review
FA—Formulation Agreement
FAD—Formulation Authorization Document
FRR—Flight Readiness Review
KDP—Key Decision Point
LRR—Launch Readiness Review
MDAA—Mission Directorate Associate Administrator
MCR – Mission Concept Review
MDR—Mission Definition Review
MRR—Mission Readiness Review
ORR—Operational Readiness Review
PCA—Program Commitment Agreement
PDR—Preliminary Design Review
PFAR—Post-Flight Assessment Review
PIR—Program Implementation Review
PLAR—Post-Launch Assessment Review
PRR—Production Readiness Review
SAR—System Acceptance Review
SDR—System Definition Review
SIR—System Integration Review
SMSR—Safety and Mission Success Review
SRB—Standing Review Board
SRR—System Requirements Review

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

Historical Method

- Specify-Design-Build-Test-Fix is historical
  - "If my university thought this was important, they would have taught me!"
  - SDBTF is an endless loop erroneously employed as a problem-solving and solution development methodology
  - No identifiable outcomes other than
    - Design components
    - Integrate them
    - Deliver on time
    - Deliver within budget
    - No regard for User needs, only basic solutions

[5] Wasson, 2015

Stakeholders

What or Who is a Stakeholder?

- Stakeholder: Any entity (individual or organization) with a legitimate interest in a system.

- They can be internal or external. NASA likes this delineation.
  - Internal Stakeholders (NASA calls these “participants”)
    - Users
    - Operators
    - Developers
  - External Stakeholders
    - Science Mission Directorates
    - Regulatory Agencies
    - Scientific Communities

[1] INCOSE, 2015

Understanding Your Stakeholders

- They're not all the same
  - NASA Mission Directorates might require document submissions
  - Investors might require quarterly reports
  - Participants might require engineering requirements
- Entities your system will interact with are also stakeholders

- ABEX Stakeholders:
  - NASA Mission Directorate
  - Astrophysics Science Community
  - Alabama Space Grant Consortium
  - Alabama CubeSat Initiative
  - Mission Operators (users)
  - Undergrad & Grad Students
  - Faculty Members
  - NASA Near Space Network
  - Component Vendors

Stakeholder Requirements

- Figure out who they are, figure out what they need in their words
  - Identifying all relevant Stakeholders can be challenging
  - Who are your Stakeholders?

- Turn those needs into formal Requirements, usually at the top level

- Knowing who you're beholden to and what they want allows,
  - Life Cycle Definition
  - Starting your formal Requirements list
  - Concept of Operations Definition (ConOps)
  - Operational Concept Definition (OpsCon)

ConOps & OpsCon

Engineers are Bad at Naming Things

- The Concept of Operations (ConOps) and Operational Concept (OpsCon) are not the same thing
  - ANSI/AIAA G-043A-2012 straight up says people confuse these because of the name, but they're not going to change it
- When thinking about Systems Engineering, draw a line between "what" and "how"
- Architecture vs. Design
  - Architecture = What
  - Design = How
- OpsCon vs. ConOps
  - OpsCon = What + Why
  - ConOps = How + To What End?

Your Voice Matters

- There is a difference between a formal engineering voice and the “Voice of the User” or “Voice of the Customer”
  - A Customer might tell you they want to be comfortable in a car
  - The formal requirement would say there “shall be seat heaters capable of providing a maximum of 10 W”

- The OpsCon is written in the Voice of the User
  - What are we doing here? What characteristics does the system have?

- The ConOps is in a formal engineering voice
  - What does the future look like when operating the system?

OpsCon

- The OpsCon describes what the system will do and the rationale behind what the system will do.
  - User-oriented document (Voice of the User)
  - Describes system characteristics from the user's viewpoint
  - Communicates the system to acquirer, user, supplier, etc.

- I find it easier to write an OpsCon before a ConOps
  - Easier to say what a system will do (OpsCon) before saying how it will do it (ConOps)
  - Writing the "what" can give you ideas for "how"

[1] INCOSE, 2015

ConOps

- The ConOps addresses the leadership's intended way of operating the organization.
  - Requires an understanding of How the system is doing its job, not just what the system is
  - Describes assumptions or intent to the operation or series of operations
  - Describes how operations work when using the system or possible future systems.
  - Usually a long-range strategic plan or annual operational plan publication

- Generally a big word document, which sucks

[1] INCOSE, 2015

ConOps Writing Dangers

- Undergraduate engineering programs have a common problem
  1. This is my problem
  2. This is my solution
  3. This is why my solution is good
- That's a bad way to do engineering because it doesn't characterize the Solution Space
- Problem Space, Opportunity Space, Solution Space. Instead,
  1. Here is our Problem Space and all the aspects of it from different viewpoints
  2. Here is our Solution Space and all the possible solutions from the various viewpoints
  3. Here is our Solution in comparison to others in the Solution space

What ABEX Is Doing

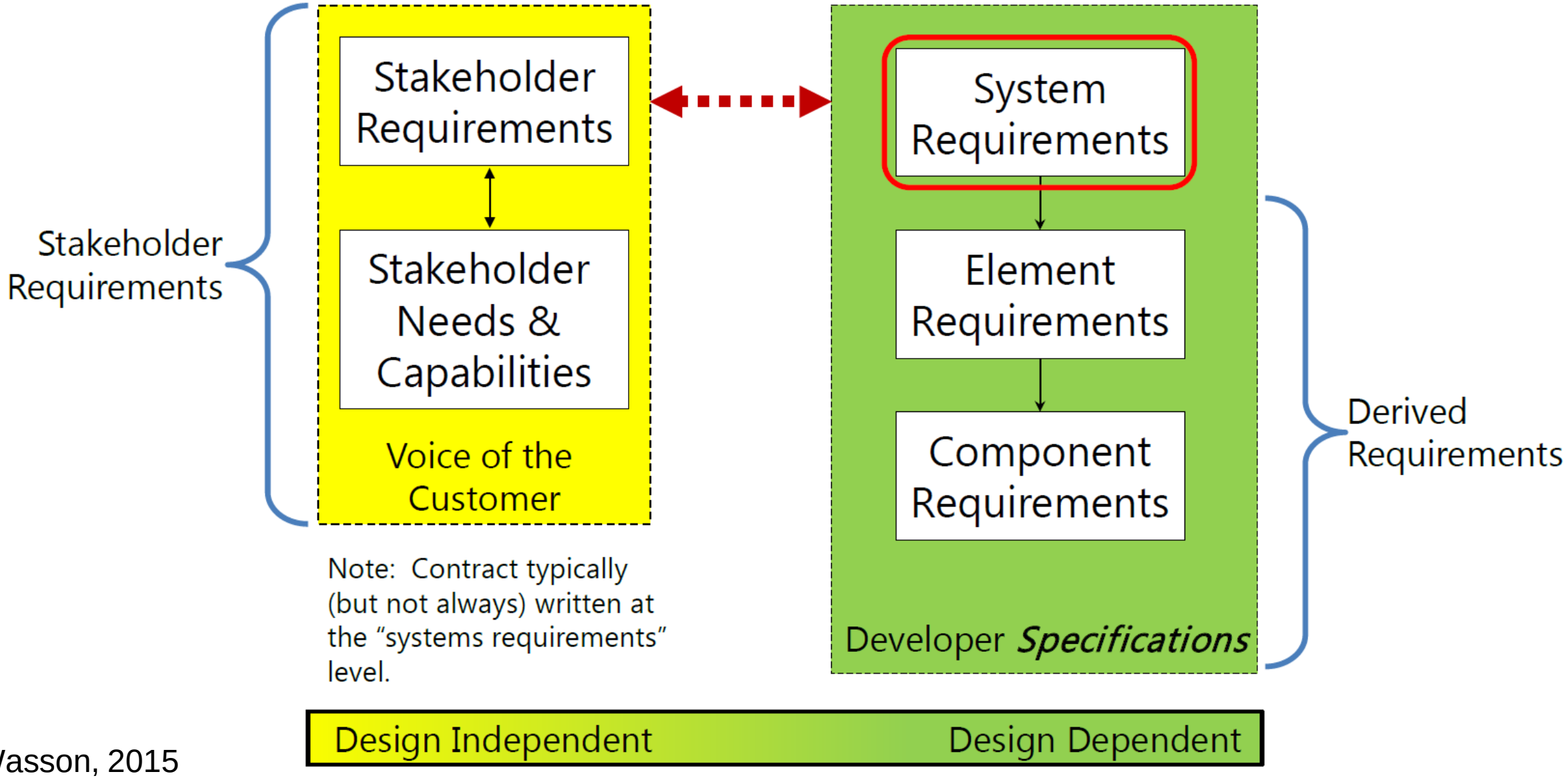
- Asking students to read 5+ long documents doesn't work
  - There's subsystem-specific documents you do need to read
  - You're not going to read 500 pages
  - That's all. This was fun.
- To help you understand the ABEX what, why, and how, you'll get an onboarding package
  - Project Summary. Lots of ConOps/OpsCon material in here.
  - Your Subsystem Qualification or Management Plan
    - This is a long document and you do have to read this one
  - Work Task Templates
  - Architecture Descriptions (structural, electrical, technical)
  - Requirements

Requirements

Where do Requirements Come From?

- Stakeholders
  - Captured in the Voice of the Customer
  - Provided/Documented in the Concept of Operations
- Use Cases
  - How do we interact with the System? What does it need to do for that?
- Physical Constraints
  - ABEX has to fit in a box with an upper limit for mass and volume
- Interfaces to other Systems
  - Complicated Systems have lots of inputs/outputs
  - Complex systems have lots of interfaces

Generalized Requirements Hierarchy



[5] Wasson, 2015

Specifications are Low Level Requirements

- Specifications are low level requirements, for ABEX that level is L4.
  - This will show up on the requirements spreadsheet
  - We might say that every interface specification must be verified by test
- Specification Categories:
  - Operational Requirements: Achieve objectives or behavioral interactions
  - **Capability Requirements:** Action and associated level or performance
  - Non-Functional Requirements: Same as before
  - Interface Requirements: Specify connectivity or interaction requirements
  - Constraint Requirements: Constraints for design and constructuion. This could be manufacturing techniques, safety factors, etc.
  - Verification Requirements: Dictating a verification method
  - Validation Requirements: Dictating a validation method

[5] Wasson, 2015

ABEX Requirement Hierarchy

- ABEX has 5 levels. Low levels (L1) decompose to high levels (L5)
  - Level 1: Mission Requirements.
    - These stem directly from Stakeholders like the Astrophysics Research & Analysis solicitation and the Alabama CubeSat Initiative
  - Level 2: Project Requirements
  - Level 3: System Requirements
  - Level 4: Subsystem Requirements
  - Level 5: Component Requirements

  } Specifications

- We might go to L6: Assembly or L7: Subassembly if we need it
  - Everything in SE should be tailored to your program

Types of Requirements

- Functional Requirements:
  - Define what functions need to be performed to accomplished the objectives. Typically higher level requirements defined early
- Performance Requirements:
  - Define how well the system needs to perform the functions. Typically result from refined functional requirements
  - Vector = direction + magnitude. Capability = function + level of performance.
- Non-Functional Requirements:
  - Safety, system security, human integration, environmental, maintenance, disposal. These generally com from guidelines or regulations.

[1] INCOSE, 2015

Requirement Examples

- Functional Requirement Statement
  - A.L2.PRO.2: The ABEX spacecraft shall downlink organized science data to the Mission Operations Center as a radio frequency signal
- Performance Requirement Statement
  - A.L4.GNC.3: The GN&C subsystem shall have attitude sensors capable of determining the spacecraft's attitude within <1 degree
- Non-Functional Requirement Statement
  - A.L1.PRGM.1: The ABEX project shall submit an End of Mission Plan to NASA consistent with NPR 8715.6 and NASA-STD-8719.14

Requirement Structure

- There's good ways and bad ways to write requirements
- Every specification requirement should consist of,
  - Source of the action (actor)
  - Verb ("shall")
  - Outcome to be accomplished
  - Relational Operator(s)
  - Level of performance
  - If needed, Condition based action or action recipients

Req. ID	Actor	Action	Outcome	Relational Operator	Level of Performance	Condition
A.L4.GNC.3	The GN&C Subsystem	Shall	Have attitude sensors capable of determining the spacecraft's attitude	Within	1 degree	[none]

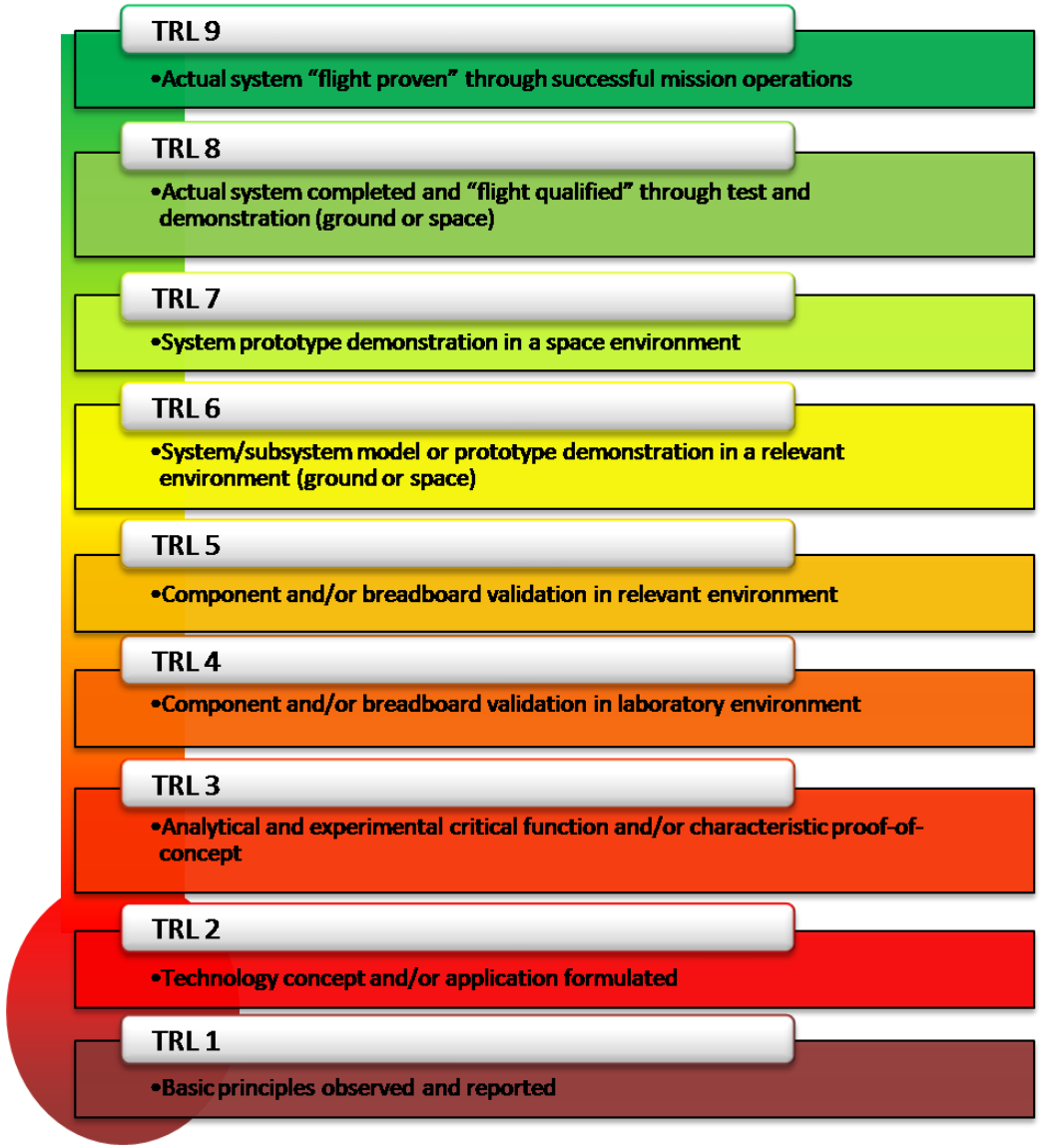
[5] Wasson, 2015

Technology Assessment

Document Classification: *Public,* Distribution: *Unlimited*

Technology Readiness Levels

- NASA and the DoD use Technology Readiness Levels (TRL)
- They're ranked 1-9
  - 1 is least ready
  - 9 is most ready
- This chart is what you'll encounter most. It's okay.



[7] NASA NPR 7123.1, 2020

Where to Find the Best TRL Definitions

- NASA Procedural Requirement (NPR) 7123.1C is the *Systems Engineering Processes and Requirements* document
- https://nodis3.gsfc.nasa.gov/displayDir.cfm?Internal_ID=N_PR_7123_001C_&page_name=AppendixE
- Appendix E has the best description you'll find for TRLs
- TRLs can be defined at varying levels of complexity
  - Total Systems (full spacecraft)
  - Subsystems (Thermal Control)
  - Components (Patch Heater)

Which TRLs Do We Care About Most?

- TRL 6 is the goal
- Colloquial breakdown:
  - Have an idea for a concept? TRL 1
  - Have you modeled it at all? TRL 2
  - Is there a physical Proof-of-Concept? TRL 3
  - Have you tested a prototype in a lab? TRL 4
  - Have you tested a prototype in a “relevant environment”? TRL 5
  - Have you tested the flight model in a “relevant environment?” TRL 6
- Everything we’re doing for in-house components is a long march to TRL 6

How ABEX Advances TRLs

- Two Plans, Two Reports, Four Total Documents per Subsystem
  - All plans/reports are design-specific
  - (1) Subsystem Development & Integration Plan
    - Design Overview, Technical Performance Measure (TPM) calculation methodology, integration chain definitions, development test plans, test entry & success criteria
  - (2) Subsystem Development & Integration Report
    - TPM calculations for the current design, sections for each functional or integration test, record of test results as compared to expectations, approval for continuance
  - These two get us to TRL 4

How ABEX Advances TRLs

- Two Plans, Two Reports, Four Total Documents per Subsystem
  - All plans/reports are design-specific
  - (3) Subsystem Qualification Plan
    - Design Overview, environmental test plans, test entry & success criteria
  - (4) Subsystem Development & Integration Report
    - TPM calculations for the current design at environmental extremes, sections for each environmental test, record of test results as compared to expectations, approval for continuance
  - These two get us to TRL 6

Assembly & Integration

Document Classification: *Public,* Distribution: *Unlimited*

Assembly vs. Integration

- Assembly emphasizes the mechanical mating of components to form a system
  - Usually just physical/electrical. Subsystems can be assembled individually

- Integration encompasses the entire process of combining less complex function to achieve a system satisfying its requirements
  - Problems with requirements/specifications are revealed
  - More difficult with more complex systems
  - Functional integration of subsystems (electrical, structural, power)
  - Hardware/Software integration
  - Launch vehicle integration

[6] Thomas, 2021

Challenges with Assembly

- Material certifications
- Component lead times from vendors
- Erroneous confirmation of structural drafts
- Clean Rooms
- Electrostatic Discharge

Challenges with Integration

- Individual disciplines tend to see the entire spacecraft from a single point of view
- Software is generally modular from the bottom-up without knowledge of what the full spacecraft looks like
  - Change in data values (Mars Climate Orbiter)
  - One software module can monopolize resources (CPU, memory, bandwidth, peripherals)
  - Software modules can accidentally overwrite memory of other modules

[6] Thomas, 2021

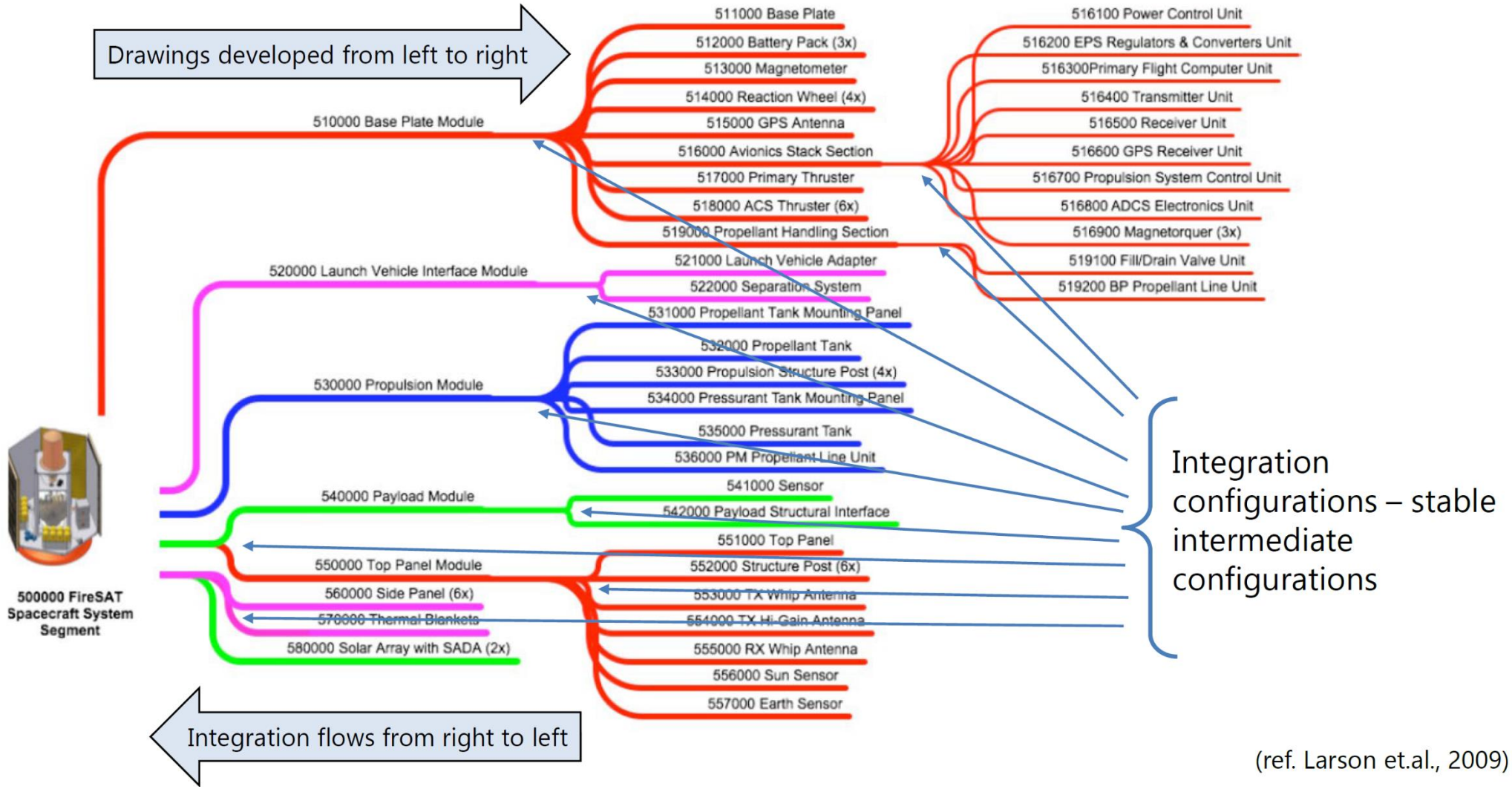
Hora & Tempus

- High-end watchmakers making watches of 1,000 pieces
- Tempus assembles watches from scratch
- Hora assembles watches from 10 subassemblies of 10 components of 10 pieces
- If there is a 1% chance they're interrupted and forced to start over, then **it will take Tempus on average 4,000 times longer to assemble a watch**.

Lesson: Complex systems will evolve from simple systems much more rapidly if there are stable, intermediate forms

[8] Simon, 1965

A Physical View of Integration Configuration Items



(ref. Larson et.al., 2009)

[9] Larson et al., 2009/2018

Integration Chains & Integration Points

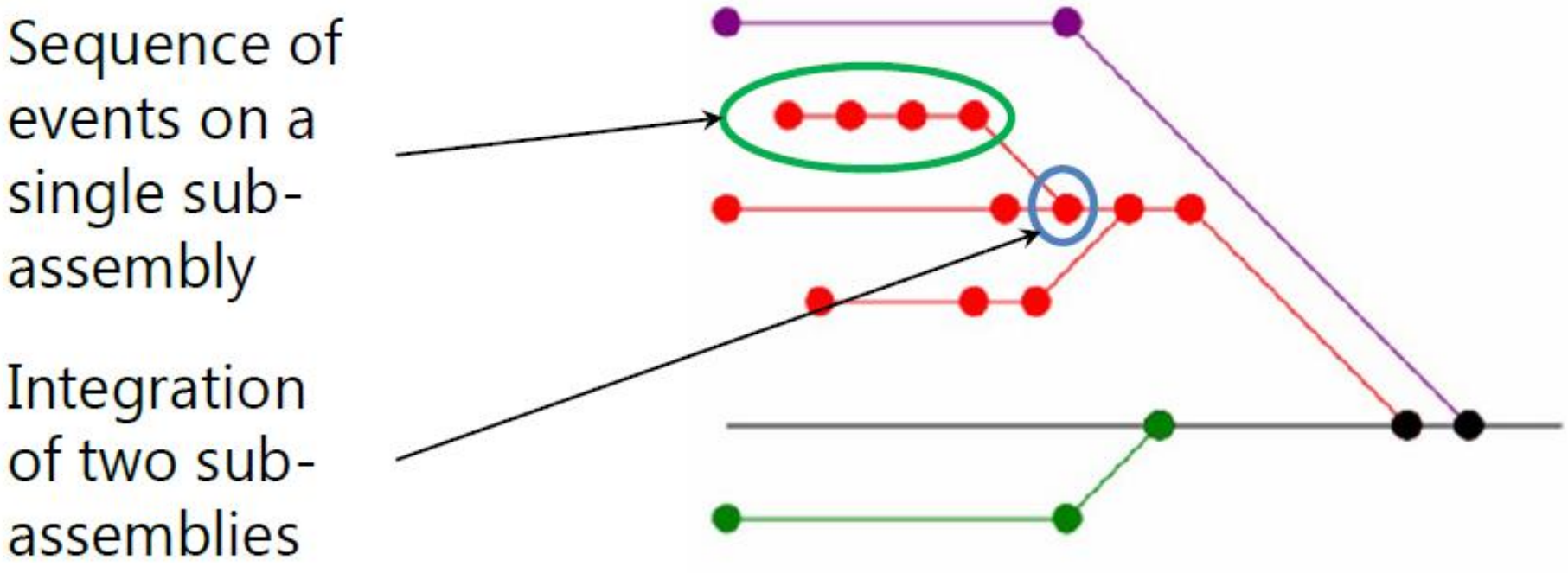
1. Determine a limited set of the most critical system performance parameters.
   - Requirements that are difficult to allocate to a subsystem generally indicate system components that will be difficult to integrate

2. Identify chains that result in critical parameters
   - The set of functions and subsystems resulting in the critical parameter is called a chain

3. Integrate Configuration Items (CI) along those chains of subsystems and functions
   - Define the CIs for the chain and you'll find your Integration Points (IP)

[6] Thomas, 2021

Integration Chains & Integration Points



- Creating these integration diagrams for your subsystem is the task
- A single Integration Point can be the end result of multiple Integration Chains

[6] Thomas, 2021

Integration Chains for Software

- In-The-Loop Nomenclaure (XIL)
  - Model-In-The-Loop
    - Computer model of both controller and environment
  - Software-In-The-Loop
    - Computer model of the environment and sensors/actuators running on a hardware controller
  - Processor-In-The-Loop
    - Real-time controller running production code connected to a computer model of the environment and sensors/actuators
  - Hardware-In-The-Loop
    - Real-time controller running production code connected to actual sensors/actuators with a computer model of the environment
    - Environmental testing would come next with simulated environment instead of a modeled environment

[10] Thomas & Mesmer, 2016

Integration Strategy

- Should a CI exist for every combination of two parts?
  - CIs undergo integration tests, and tests cost time & money
  - How much is enough? How much is too much?

- Failure to plan for integration tests has resulted in mission failures
  - Ariane 5 (https://www.youtube.com/watch?v=gp_D8r-2hwk)
  - Mars Climate Orbiter (https://www.youtube.com/watch?v=urcQAKKAAl0)
    - This one is particularly dumb. All Imperial/Freedom units are just metric units x conversion.
    - I cannot express enough how worthless Imperial units are.

- Runaway costs lead to project cancellations
  - Have to balance between cost & risk when considering integration strategy

[6] Thomas, 2021

Verification & Validation

Verification & Validation: The Most Useful SE Tools

- Verification & Validation (V&V) are the lifeblood of SE that fuels the heart of champions

- Verification: Did we build the thing correctly?
  - Provides proof of the compliance of the produce with the requirements
  - Must be planned for early in design process
  - Required for a system of any complexity
  - Dependent on the requirement you're verifying compliance to

[6] Thomas, 2021

Verification & Validation: The Most Useful SE Tools

- Validation: Did we build the correct thing?
  - Assessing both activities and work products
  - Is this what the user/customer wanted?
  - Technical Reviews, Requirements Development, Analysis Outputs

- V&V methods are the same for both; the purpose is different
  - Validation usually happens earlier than Verification

[6] Thomas, 2021

Verification (and Validation) Methods

- Test, Demonstration, Analysis, Inspection
  - Verification by Test is a ***measured*** compliance
  - Verification by Demonstration is an ***observed*** compliance
  - Verification by Analysis is a ***predicted*** compliance
  - Verification by Inspection is a ***documented*** compliance

- Verification needs to know,
  - Method of the requirement compliance verification
  - Level of verification (System, Subsystem, Component, etc.)
  - Life cycle phase it is happening in (Development vs. Production)
- Verification activities often lead to requirements changes
  - There's a big V&V description section in the requirements spreadsheet

[6] Thomas, 2021

Verification Method: Inspection

Description

- Conformance to requirements by reviewing drawings & data.

When to Use

- Whenever drawings, documents, or data can be visually checked to verify the item's physical characteristics requirements have been designed into the product.

[9] Larson et al., 2018

Verification Method: Analysis

Description

- Modeling & Simulation, statistical analysis, and quantitative engineering.

When to Use

1. When rigorous and accurate analysis is possible.
2. Test is not cost effective
3. Verification by inspection is not adequate

- Advantageous if,
  - Expected precision is known
  - Consequences of undiscovered problems can be tolerated
  - Test can't be done credibly

[9] Larson et al., 2018

Analysis Composition

1. **Problem definition**
   - Why is this analysis wanted? What are we trying to learn? What is the scope of the analysis?
2. **Parameters**
   - These are the selected inputs that we believe are pertinent or that were given to us
3. **Assumptions**
   - These are the assumptions we made, why we made them, and the basis for those assumptions. Discussion on the impact of those assumptions
4. **Methodology**
   - This is how our analysis process and logic work. Domain Knowledge Maps live here.
5. **Results**
   - These are the outputs of our analysis.
6. **Conclusion**
   - This is what we learned from the analysis and the answer to our initial question/problem

Verification Method: Test

- Three types of tests:
  - Type 1: Development tests to validate new design concepts
    - Confirms performance margins, manufacturability, failure modes, requirements compliance
    - Purpose: **Inform the system design**
  - Type 2: Qualification tests to validate the soundness of the design
    - Confirms hardware is built to specification, tooling is correct, personnel competency
    - Purpose: **Show the system design meets the requirements/specifications**
  - Type 3: Acceptance tests to assure conformance to specification requirements
    - Workmanship, not design. This happens after a build.
    - Derived from but less extreme than qualification tests
    - Purpose: **Prove a product was manufactured, assembled, and integrated correctly**
- Testing is more complicated and costly than other verification methods

[9] Larson et al., 2018

Qualification vs. Verification

Verification:

- The formal process (test, analysis, inspection, or demonstration) to confirm a system and its associated hardware and software components satisfy all specified requirements.
- It's performed once on the final design.

Qualification:

- Ensures the flight unit design will meet functional and performance requirements in anticipated environmental conditions. Qualification tests, a type of verification method, is performed at the extremes of the environmental envelope.
- It's performed once on the final design.
- Verification by analysis doesn't mean a system is qualified

Verification Method: Test

- Three categories of tests:
  - Functional tests, “Does it work?”
    - Specific to functional requirements. May or may not include a level of performance.
    - Function is not the same as capability. Capability = function + level of performance
  - Integration tests, “Do the pieces work together as intended?”
    - Mechanical, electrical, software as Configuration Items along Integration Points
  - Environmental tests, “Does it work in the operational environment?”
    - Thermal Vacuum Chamber
    - Vibration Table
    - Electromagnetic Compatibility
    - Radio Frequency Compatibility

[9] Larson et al., 2018

Verification Method: Test

Description

- Conformance to requirements through technical means such as special equipment, instrumentation, and simulation.

When to Use

1. Analytical techniques do not produce adequate results
2. Failure modes exist which could compromise personnel safety, flight systems, payload operation, or mission objectives
3. Any components directly associated with critical system interfaces

[9] Larson et al., 2018

Verification Method: Demonstration

Description

- Conformance to requirements through operation, adjustment, or reconfiguration of a test article.
- A characteristic observable by humans.
- Usually implemented during a test procedure for a different requirement

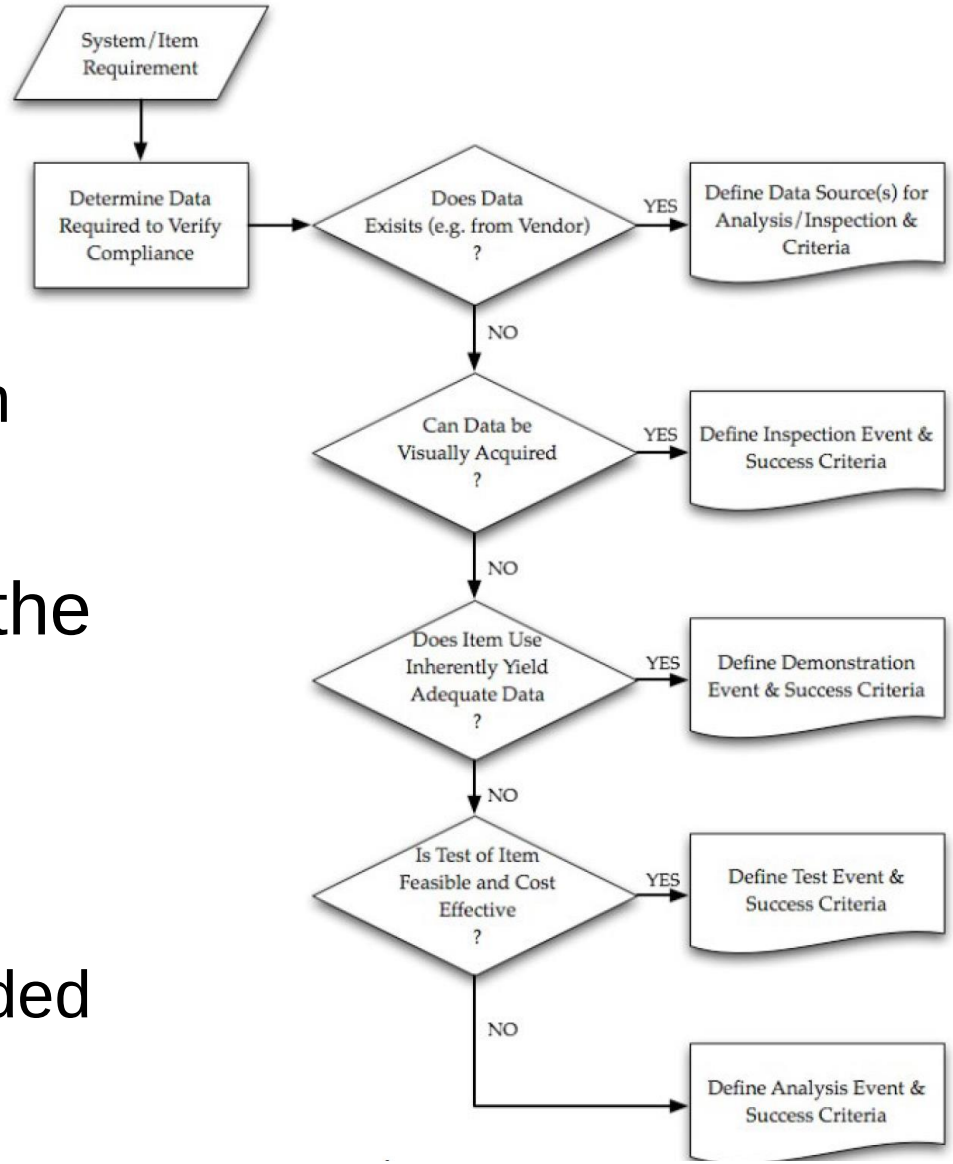
When to Use

- Whenever designed functions under specific scenarios can be performed and observed.
- Generally not specific to a level of performance. Just a function.

[9] Larson et al., 2018

How to Select a Verification Method

1. What's the requirement?
2. What data is needed?
3. Does a vendor give it?
   – Vendor certifications common
4. Can we see it?
5. Does using it answer the question?
6. Is testing feasible?
7. Okay we'll analyze it.
   - Verification by analysis provided here for educational purposes



(ref: Larson, et.al., 2009)

[9] Larson et al., 2009/2018

Modeling & Simulation V&V

- It's not just products that are verified an validated.
  - Models for analysis need V&V
    - Are we modeling the right thing and are we analyzing it correctly?
  - Simulations for tests need V&V
    - Are we simulating the right thing and are we testing it correctly?

- NASA-STD-7009A: Standard for Models & Simulations
  - When to use standards rigorously, M&S Criticality Assessment
    - Can't test something but it's critical to the mission? Be sure you're analyzing correctly
  - How credible is your model?
    - Data pedigree, uncertainty characterization, results robustness
    - Graded on a scale from 1-4 for both Verification & Validation
      - Your model can be verified but not validated and vice versa

Technical Performance Measurement

Document Classification: *Public,* Distribution: *Unlimited*

TPMs Are Tricky

- Here's how we're going to tackle this
  - How does NASA look at TPM definitions?
  - What are TPMs?
  - What does it mean to define a good TPM?
  - How are we going to represent TPMs?

The NASA Systems Engineering Approach

- Technical performance measurement decomposes from qualitative Measures of Effectiveness (MOE) to quantitative Measures of Performance (MOP)

- MOP: A quantitative measure that, when met by the design solution, helps ensure that a MOE for a product or system will be satisfied.

[11] NASA-SP-2016-6105, 2016

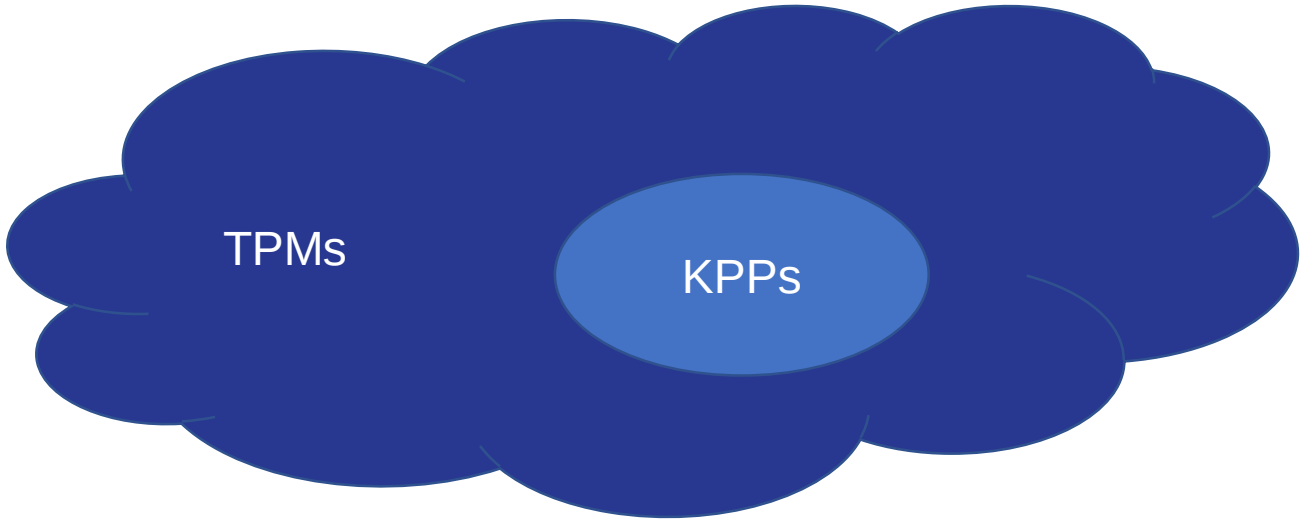
What are TPMs to NASA?

- A set of performance measures that are monitored by comparing the current actual achievement of the parameters with that anticipated at the current time and on future dates
- Typically selected from the defined set of MOPs
- Used to confirm progress and identify deficiencies that might jeopardize meeting a system requirement

[11] NASA-SP-2016-6105, 2016

How are TPMs applied?

- TPMs exist for components, subsystems, and systems alike.
- System-level TPMs are Key Performance Parameters (KPP).
  - KPP: Those capabilities or characteristics (typically engineering-based or related to health and safety or operational performance) considered most essential for successful mission accomplishment. They characterize the major drivers of operational performance, supportability, and interoperability.



[11] NASA-SP-2016-6105, 2016

How are TPMs defined?

- Hallmarks of a well-defined TPM:
  - Should be important and relevant
  - Should be relatively easy to measure
  - Performance or knowledge of performance should be expected to improve with time
  - A target, threshold, or expectation of uncertainty should be known
  - If the measure crosses its threshold, corrective action should be known
  - The measured parameter should be controllable
  - Management should be able to tradeoff cost, schedule and performance
  - Should be documented
  - **Should be tailored for the project**

[6] Thomas, 2021

“Should be tailored for the project”

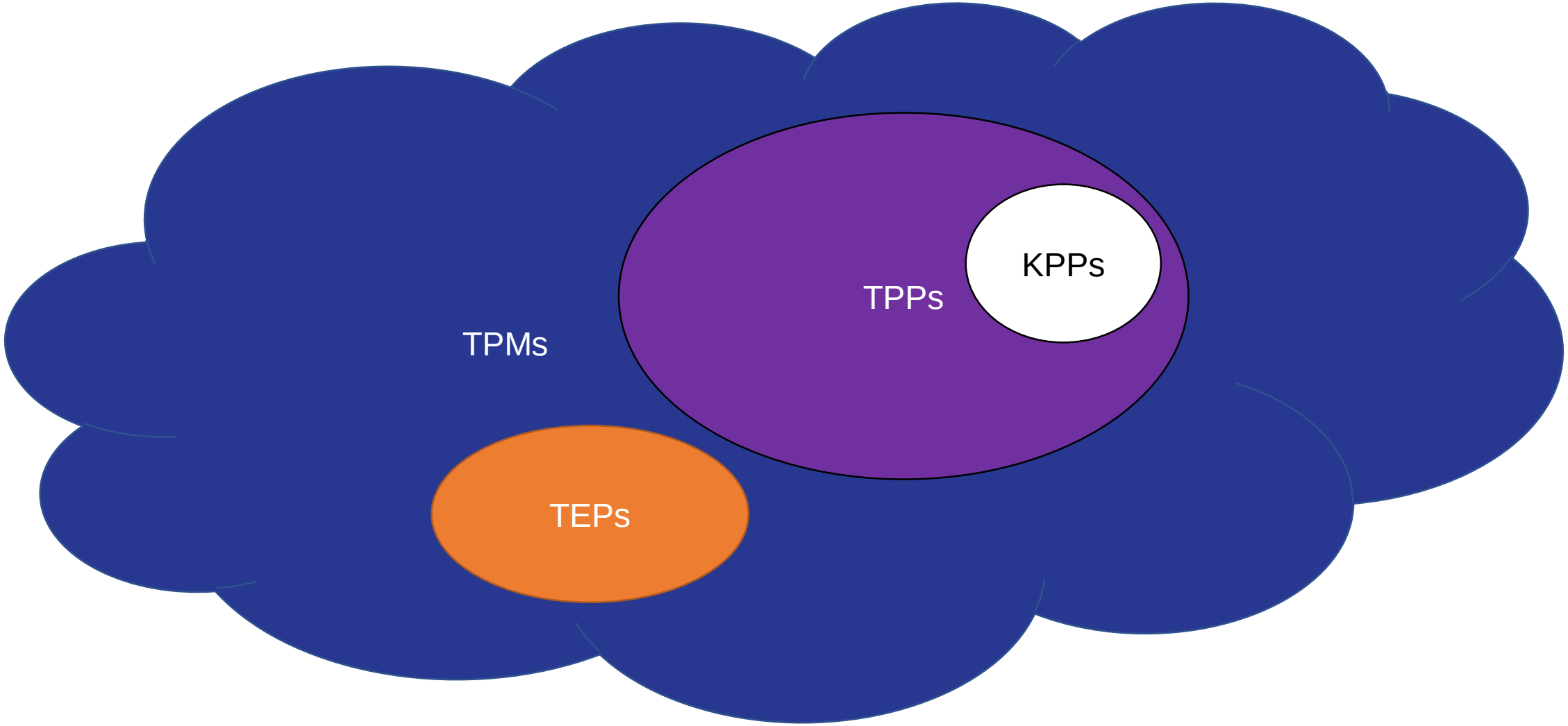
- ABEX is not a $100,000,000 mission, but we do have TPMs
- MOPs and MOEs are associated with Value Functions, which are graduate-level concepts
  - MOEs are not the only way to represent stakeholder concerns
  - MOPs decomposing to TPMs are not the only way to track metrics
- ABEX wants to represent these concepts in Model-Based Systems Engineering environments
  - SysML Parametric Diagrams for calculations
  - SysML Activity Diagrams for methodologies
  - Calculations used for verification procedures
  - Methodologies used as teaching and simulation tools

The ABEX TPM Strategy

- ABEX is not defining MOEs or MOPs, only TPMs
  - Stakeholder concern representation is occurring in requirements Verification & Validation processes

- Three kinds of TPMs:
  - **Key Performance Parameters (KPP):** Same as NASA definition
  - **Technical Performance Parameters (TPP):** Those capabilities or characteristics (typically engineering-based or related to health and safety or operational performance) considered relevant to operational performance, supportability, and interoperability at any level.
  - **Technical Environmental Parameters (TEP):** Those capabilities or characteristics relevant to the definition of system interactions with the Operational Environment.

TPM Strategy Visualization



Where are TPMs Defined?

- ABEX subsystem teams exist within Domains
  - Technical Analysis Domain
    - Structures, Thermal, C&DH, GN&C, EPS, TT&C, Orbit, Payload,
  - Software Development Domain
    - FSW
  - Project & Mission Domain
    - Project Management, Systems Engineering

- Each subsystem defines subsystem-specific TPMs using their domain's relevant concepts
  - All TPMs in Technical Analysis Domain or Software Development Domain
  - No need to blur the lines between domains

Diagrams for TPMs

- Each Domain has an Ontology
  - Ontology: a set of concepts and categories in a subject area or domain that shows their properties and the relations between them.
- The Ontology describes what categories and relationships we can use to describe TPMs
- Domain Knowledge Maps (DKM) are applications of an ontology's categories and relationships to describe a TPM methodology
  - Will talk more about this in MBSE Seminar

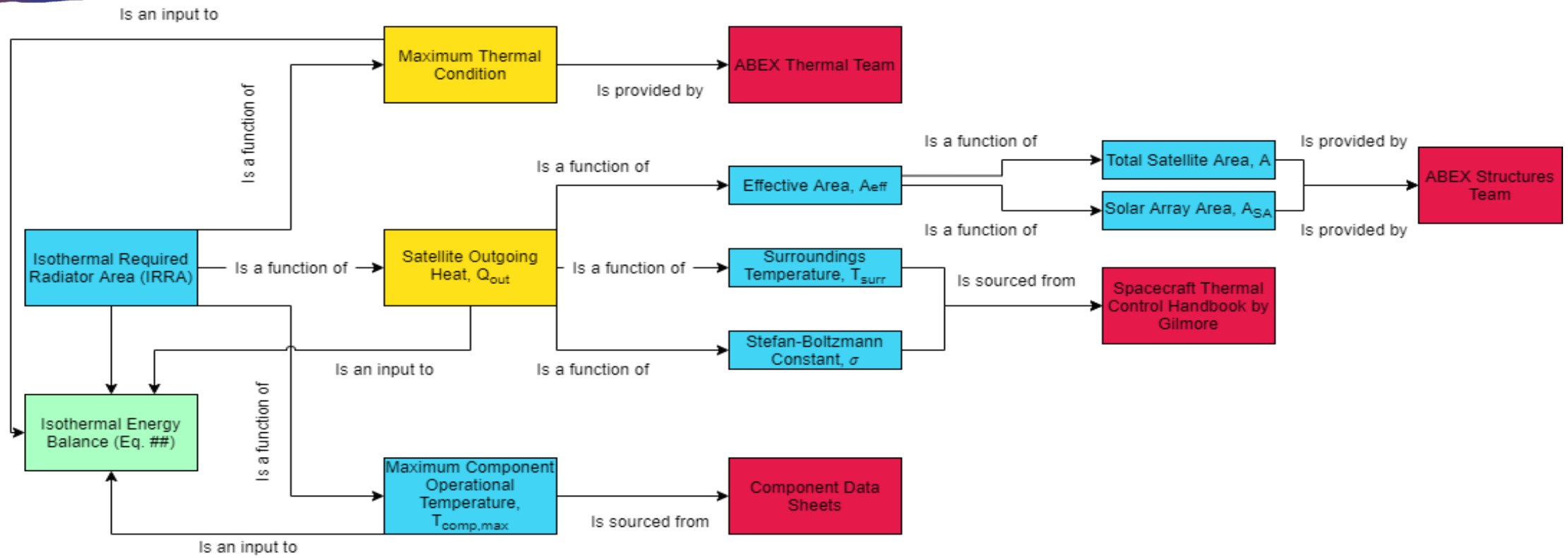
[12] Merriam Webster, 2022

Example: Technical Analysis Domain Ontology

Scalar Parameter	• is a function of • is compared to	• is a function of	• is a function of	• is varied by	• is provided by • is sourced from	• is an input to • is calculated by		• is an input to • is modeled by	
Array Parameter	• is a function of	• is a function of	• is a function of	• is varied by	• is provided by • is sourced from	• is an input to • is calculated by		• is an input to • is modeled by	
Matrix Parameter	• is a function of	• is a function of	• is a function of	• is varied by	• is provided by • is sourced from	• is an input to • is calculated by		• is an input to • is modeled by	
State									
Source									
Equation					• is sourced from				
Modeling Environment Parameter				• is varied by				• is an input to	
Modeling Environment Module									
Modeling Environment								• is comprised of	
	Scalar Parameter	Array Parameter	Matrix Parameter	State	Source	Equation	Modeling Environment Parameter	Modeling Environment Module	Modeling Environment

- Read
  - Left-to-Right
  - Go to relationship
  - Then Down
- Ontology Ex:
  - “Scalar Parameter is varied by State”
- DKM Ex:
  - “Direct Solar Heat Flux is varied by Thermal Environment State”

Example: Isothermal Required Radiator Area DKM



- This DKM isn't perfect
- We'll be providing ample guidance on the creation of DKMs
- Task: Define TPMs, create DKMs to show your methodology

Risk Assessment

Risk Guidance

- Risks are due to design decisions
  - What decision in the known design is causing the risk?
- Risks should include **design uncertainty**, but not general uncertainty
  - What about the design are we unsure of that impacts its performance and final implementation?
- Consequence is a property of the risk, not the risk itself
  - What is the source that results in the risk's consequence?

Risk Guidance

- Risks must have known, quantifiable impact and source
  - Do we understand what causes the risk's consequence?
- Risks exist in both development and management
  - What in the development or leadership approach could be a risk?
- Risks do not exist for a design that hasn't been defined
  - What is our current design that we can determine the risk consequence from?

Risk List

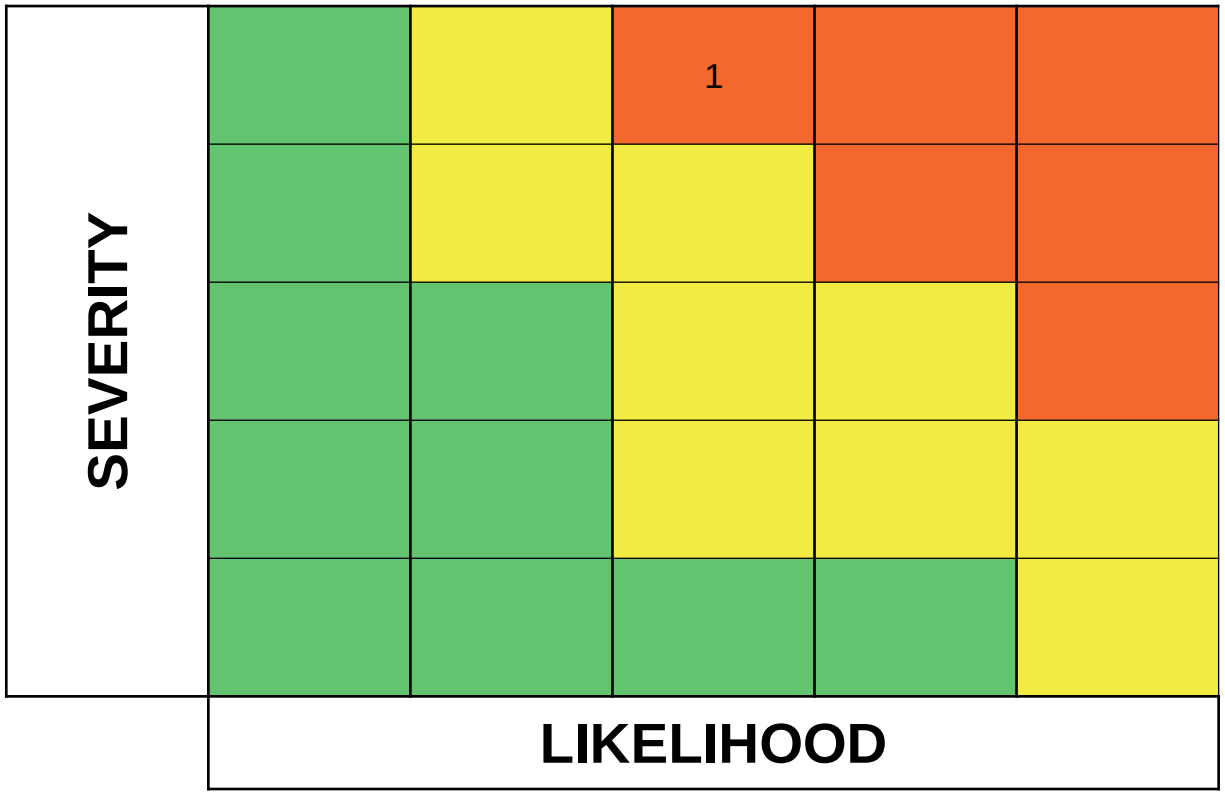
			Pre-handling						Handling Method		Post-handling								
			Severity								Severity								
ID	Title	Statement	Safety	Performance	Cost	Schedule	Likelihood	Criticality	Handling	Handling Strategy	Safety	Performance	Cost	Schedule	Likelihood	Criticality	Status	Revision Date [mm/dd/yyyy]	Revision Notes
1	Risk Name	Given that [Condition], there is a possibility of [Departure] adversely impacting [Asset], which can result in [Consequence].	1	5	1	1	3	15	Research	TBD or current description of handling strategy.	1	3	1	1	3	9	Open		TBD or current description of handling strategy.

- Copy the “Risk_Report.xlsx” table form the “Risk List” sheet and remove the condition, departure, asset, and consequence columns. Make sure to right click and select “Keep Source Formatting (K)” to maintain the style
- Once pasted here, Increase font size, table size, and row/column size as deemed fit

Risk Summary and Criticality Matrix

ID	Title	Handling	Current Handling Strategy
	Risk Name	Research	TBD or current description of handling strategy.

Show the summary of the ID, risk title, handling and handling strategy as from the prior slide



Show the criticality matrix, copy from the "Risk_Report.xlsx" and keep the same formatting style

Summary

In Summary, Systems Engineering is Hard

- Every engineering is a systems engineer
- All systems have a life cycle, and some systems have multiple
- Requirements are derived from stakeholders
  - Specifications are low-level requirements
- All ABEX subsystems must be TRL 6 by PDR/CDR
  - TRL 1-4 advancement achieved by development tests
  - TRL 5-6 advancement achieved by qualification tests

In Summary, Systems Engineering is Hard

- Integration is probably the hardest part of Systems Engineering
  1. Identify subsystem-specific TPMs, even if it is a few
  2. Identify the components that realize those TPMs
  3. Work backwards along the integration chain to realize the TPMs
     a. Identify Configuration Items
     b. Identify Integration Points
  4. Develop the Integration Chain schedule

- Verification is did I do the thing right, Validation is did I do the right thing?
  - Analysis, Inspection, Demonstration, Test.

In Summary, Systems Engineering is Hard

- TPMs help us compare current performance with desired or expected performance

- ABEX defines three kinds of TPMs
  - Key Performance Parameter (system-wide, 1 if any per subsystem)
  - Technical Performance Parameter (most of them)
  - Technical Environmental Parameter

- TPM content & methodology is represented by Domain Knowledge Maps
  - Verification by Analysis for educational purposes
  - Covered much more in MBSE Seminar

Citations

[1] Systems Engineering Handbook, INCOSE, 4th Ed, 2015

[2] Honour, Eric C. Systems engineering return on investment. Adelaide: University of South Australia, 2013.

[3] Blanchard, Benjamin S. & Fabrycky, Systems Engineering and Analysis, 5thed., Prentice-Hall, 2011.

[4] International Organization for Standardization. (2010). Systems and software engineering – Life cycle management – Part 1: Guide for life cycle management (ISO Standard No. 24748-1:2010). https://www.iso.org/standard/50502.html

[5] Wasson, Charles S. System engineering analysis, design, and development: Concepts, principles, and practices. John Wiley & Sons, 2015.

[6] Thomas, L. D. (n.d.). Engineering Systems. University of Alabama in Huntsville; University of Alabama in Huntsville. 2021.

[7] NASA Systems Engineering Processes and Requirements (w/Change 1), NPR 7123.1C, Feb 2020.

[8] Simon, Herbert, “The Architecture of Complexity,” Proceedings of the American Philosophical Society, Vol. 106, No. 6. (Dec. 12, 1962), pp. 467-482.

[9] Larson, W. J., Kirkpatrick, D., Sellers, J. J., Thomas, L. D., & Verma, D. (2018). Applied Space Systems Engineering. CEI.

[10] Thomas, Dale, and Bryan Mesmer. "Virtual Systems Integration using Model Based Systems Engineering."AIAA SPACE 2016. 2016. p. 5311.

[11] NASA Office of the Chief Engineer, NASA Technical Handbook. NASA-SP-2016-6105 Rev2, “NASA Systems Engineering Handbook”. https://www.nasa.gov/sites/default/files/atoms/files/nasa_sy stems_engineering_handbook_0.pdf.

[12] “Ontology.” Merriam-Webster, Merriam-Webster, https://www.merriam-webster.com/dictionary/ontology.


ABEX
Alabama Burst Energetics eXplorer
Alabama Space Grant Consortium
Industry
University
Community
est. 1989
THE UNIVERSITY OF
ALABAMA IN HUNTSVILLE
AUBURN
UNIVERSITY
USA
UNIVERSITY OF
SOUTH ALABAMA
THE UNIVERSITY OF
ALABAMA