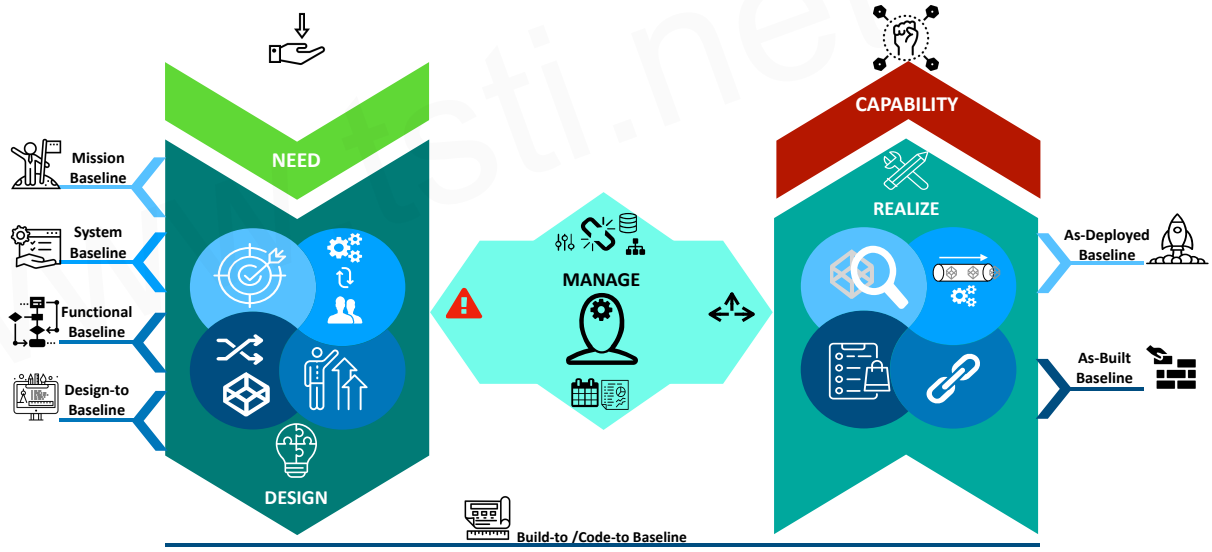
Overview of Applied Systems Engineering

With Focus on Solution Architectures



Presentation Introduction

Copyright ©2020 Teaching Science and Technology Inc.

Copyright ©2020 Teaching Science and Technology Inc.

Copyright Notice

- TSTI owns the copyright on the course as presented in its entirety
  - –If you would like to use the course notes for republication or commercial purposes, please request permission from:

Jerry Sellers
Teaching Science and Technology, Inc.
14 Via Piedras
Manitou Springs, CO 80829
Phone: (719) 237-4455
E-Mail: courses@tsti.net

If you have any suggestions about the course, we'd like to hear from you!

Copyright ©2020 Teaching Science and Technology Inc.

About TSTI

A respected name in systems engineering workforce development for over 25 years.

- **Teaching Science and Technology, Inc.** (TSTI) is a veteran-owned small business that assists companies and organizations who wish to improve their systems engineering skills as they relate to the space and other domains.
  - We present quality short courses and workshops in space systems engineering and program management to customers around the world. Our teaching style is highly interactive with emphasis on learning-by-doing and practical real-world examples and applications.
  - Our systems engineering courses provide the processes, tools, and information necessary to "jump-start" new-hires to the space business or improve the performance of mid- to senior-level engineers, systems engineers, and technical program managers.
  - In addition, we provide focused systems engineering support to organizations as they approach critical events in the development process and work as red team support.
  - We wrote the books that describe SE in the space domain, and we've all had practical SE and PM experience in the development world.

- Since 2010, TSTI has taught more than 500 courses to more than 20,000 students for NASA, ESA, DoD and industry around the world
- What Sets Our Courses Apart:
  - CREDIBILITY: We wrote the books. We've been there, done that. We bring extensive systems engineering experience.
  - COMPETENCE: We're professional teachers. This is our day job. We take the complex and make it simple. We're explainers and motivators. Hands-on exercises ensure that you learn space by doing space. No death by PowerPoint.
  - CURRICULUM: We take you from new hire to seasoned space professional. Our on-line learning portal enables continuity through your increasing levels of expertise.
  - COST: We're small, with low overhead, and come to you. Your cost is 50% less – you can train twice the people for the same price.
  - CONTRACTING: You can get on contract in minutes! We're only a credit card purchase away.

Learn more at: www.tsti.net

Copyright ©2020 Teaching Science and Technology Inc.

Copyright ©2020 Teaching Science and Technology Inc.

About TSTI

ONSITE TRAINING

TSTI is a premier provider of onsite systems engineering training to NASA, DoD and Industry.

VIRTUAL TRAINING

Since Covid, TSTI has pivoted to providing 100% virtual training. Self-paced online courses also available.

PROJECT COACHING AND INTEGRATED PROGRAMS

Starting a new project? Struggling with an existing one? Need integrated work force development tailored to your project?

SYSTEMS ENGINEERING CONSULTING SERVICES

Project start ups. Red Team proposal review. Expertise in Model-Based Systems Engineering (MBSE).

Learn more at: www.tsti.net

Copyright ©2020 Teaching Science and Technology Inc.

Instructors

✦Dr. Jerry Sellers, Ph.D., University of Surrey, UK (jsellers@tsti.net)

- Partner and President, TSTI with over 30 years of aerospace engineering experience, including Space Shuttle Mission Controller, NASA/JSC, and Chief of Astronautics for the European Office of Aerospace Research and Development, London, UK
- Former Director of the Space Systems Research Center, USAF Academy. Responsible for designing, building, testing and operating small, scientific satellites.
- Author of the text *Understanding Space: An Introduction to Astronautics*, Editor and contributing author of the text *Applied Space Systems Engineering*
- Adjunct Professor, Stevens Institute of Technology. Adjunct Professor, Georgetown University
- Certified Agile/Scrum Master, Certified Scaled Agile Framework (SAFe) Program Consultant (SPC5)
- Corresponding Member - International Academy of Astronautics

✦Dr. Peter Van Wirt, Ph.D., Utah State University, (pvanwirt@tsti.net)

- Former deputy head of the Astronautics Dept at the U.S. Air Force Academy, responsible for labs and research
- Program manager and systems engineer for numerous aircraft, missile and space systems
- Over 30 years of aerospace engineering experience, contributing author to Applied Space Systems Engineering
- Adjunct Professor at Stevens Institute of Technology

Connect with us on LinkedIn!

✦Dr. Wiley Larson, Ph.D., Texas A & M University (wlarson@tsti.net)

- Program Director and Distinguished Service Professor at Stevens Institute of Technology
- Managing editor of the Space Technology Series (including *Space Mission Analysis and Design, Understanding Space, and Cost-Effective Space Mission Operations*) sponsored by NASA and DoD
- Over 30 years of experience in space mission and spacecraft system design

Copyright ©2020 Teaching Science and Technology Inc.

Copyright ©2020 Teaching Science and Technology Inc.

Course Objectives and Agenda

✦Learning Objectives

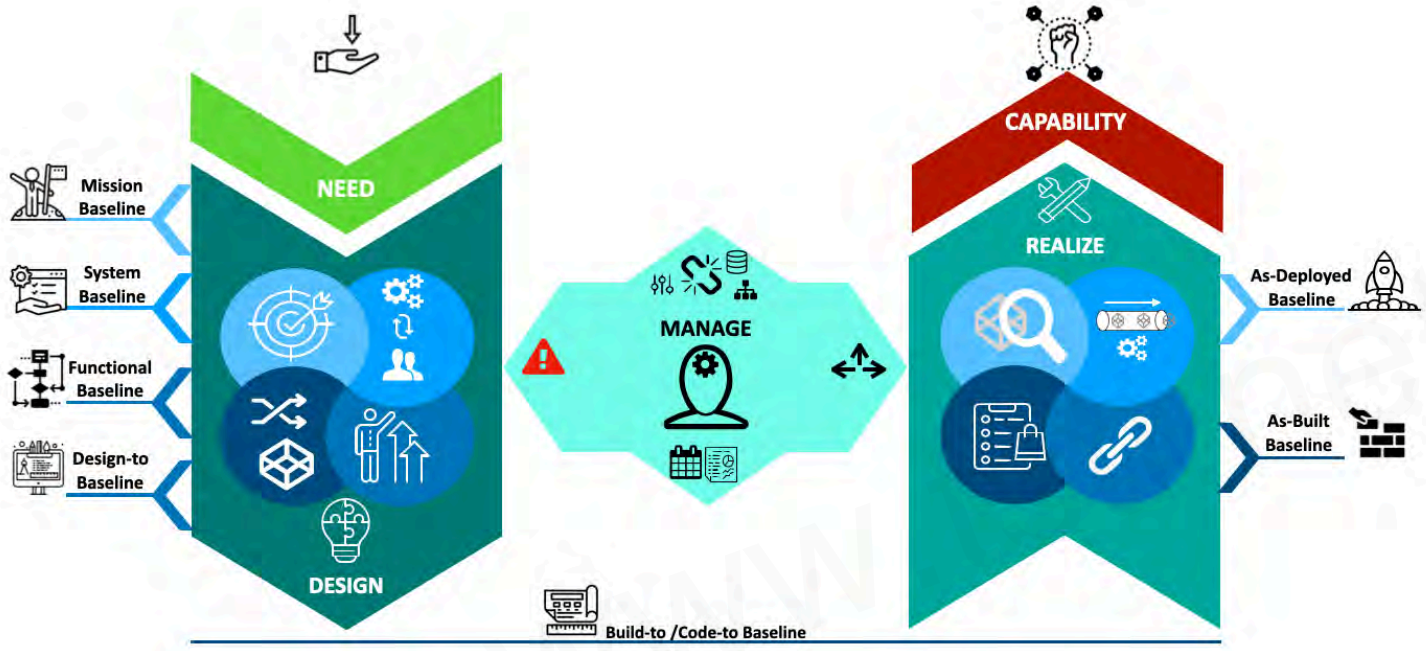
- At the end of this course you should be able to…
  - Define key systems engineering terms
  - Explain fundamental systems engineering principles
  - Apply systems engineering tools and techniques to solve specific design, manage and realization challenges
  - *Develop relevant systems engineering artifacts for a given scenario that captures and communicates design, systems management and system realization decisions*

✦Agenda

- **Introduction to Systems Engineering**
- **DESIGN**
  - Bounding Scope
  - Engineering Requirements
  - **Developing the Solution Architecture**
- MANAGE
  - Managing Interfaces, Configuration and Data
  - Making Decisions
  - Managing Risk
  - Technical Planning
- REALIZE
  - Implementing and Integrating
  - Verification and Validation
  - Transition and DevOps
- Capstone Exercise
- FireSAT Case Study (Reference)

Copyright ©2020 Teaching Science and Technology Inc.

Systems Engineering—Overview



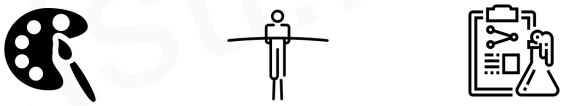
Copyright ©2020 Teaching Science and Technology Inc.

Copyright ©2020 Teaching Science and Technology Inc.

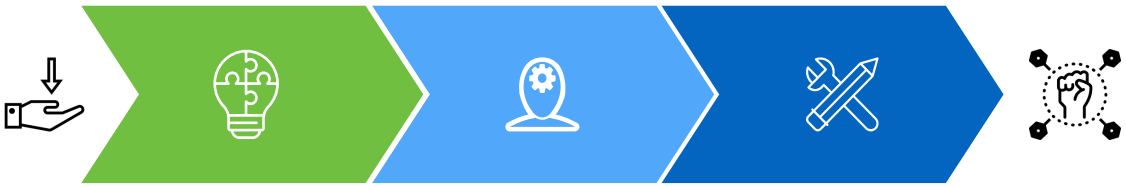
What is Systems Engineering?

Systems Engineering is the

Art and **Science**



Value Stream that turns a *NEED* into a *CAPABILITY*

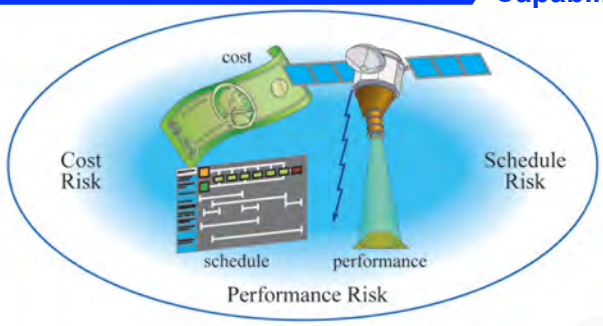


Copyright ©2020 Teaching Science and Technology Inc.

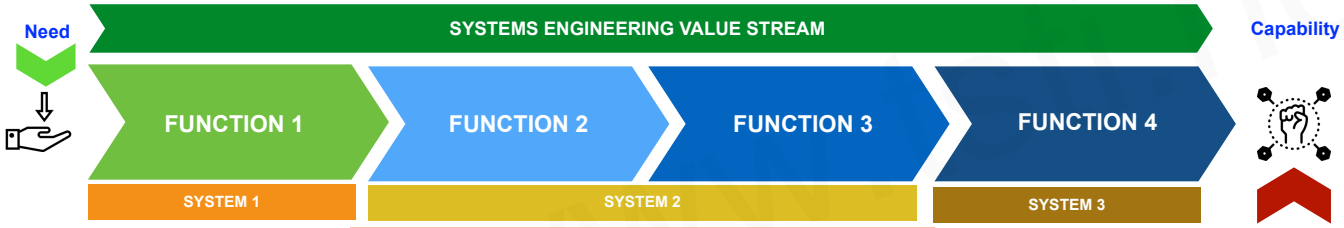
Systems Engineering Value Stream

Needs → **Capabilities**

- ✦Sources of Customer Needs—may be in response to:
  - A current functional deficiency
  - An existing operational deficiency
  - A desire to leverage new technology breakthroughs to enhance mission capability or market positioning
  - An evolving threat or competition
- ✦Needs don't come with a blank check, so along with the need are some constraints on cost, schedule, performance and acceptable risk
- ✦The needed Capabilities are delivered by a values stream that designs, manages and realizes an integrated set of systems that perform a collection of functions



The Systems Engineering "Universe"



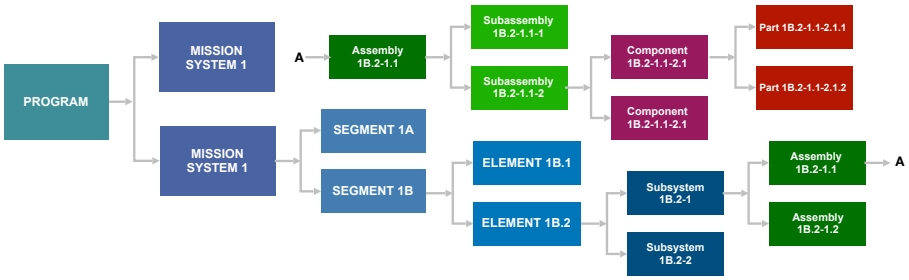
A system (or system of systems) *only* exists to deliver value to stakeholders

Copyright ©2020 Teaching Science and Technology Inc.

Copyright ©2020 Teaching Science and Technology Inc.

Systems

- Useful capability to stakeholders is the value delivered by systems engineering. That capability is provided by a system or an interdependent system of systems.
- It is useful to think of these systems in a hierarchical arrangement (product breakdown structure)



Copyright ©2020 Teaching Science and Technology Inc.

Systems Engineering Process Framework

We can think of the Systems Engineering Value Stream as a *Process Framework* for turning needs into capabilities. This framework has 3 interdependent elements:

DESIGN **MANAGE** **REALIZE**

Copyright ©2020 Teaching Science and Technology Inc.

Copyright ©2020 Teaching Science and Technology Inc.

DESIGN — MANAGE — REALIZE

- First and foremost, systems engineering is about getting the DESIGN right
  - A poor design can't be build
  - A poor design will be impossible (or too expensive) to operate
- The **DESIGN** process must empathize with the stakeholder's need, translate it into technical requirements and envision a design solution that balances cost, schedule, performance and risk
  - The DESIGN solution is captured in models, drawings, diagrams, documents and communicated to other engineers charged with building and operating it

- To herd the technical and programmatic "cats" of any program there are a myriad of people, processes and things to **MANAGE**
  - From risk to requirements, from data to decisions, management processes keep the solution train on the tracks

- To deliver the capability to stakeholders, we have to **REALIZE** the detailed DESIGN solution, turning an idea on paper into hardware and software that works

Copyright ©2020 Teaching Science and Technology Inc.

DESIGN Processes

There are four interdependent DESIGN processes

We iterate through these four processes until we converge on a design solution that balances cost, schedule, performance and risk

✦Stakeholder Expectations

- What are the measurable objectives and key results (OKRs)?
- OKRs and other parameters help to define "definition of done" for the systems engineering effort

✦Concept of Operations

- How do stakeholders plan to operate the design solution?
- What solution context are we constrained by?

✦Requirements Engineering

- What are the form, fit and functional requirements the solution must meet to satisfy all passive and active stakeholders?

✦Solution Architectures

- What does the solution do?
- What does the solution look like?

Copyright ©2020 Teaching Science and Technology Inc.

Copyright ©2020 Teaching Science and Technology Inc.

REALIZE Processes

✦**Implementation**
- How will we buy, build or re-use parts, components or whole systems to get everything we need to integrate?

✦**Integration**
- How will we put all the pieces to together to make the whole system (or systems)?

✦**Verification and Validation**
- Did we build the system *right*?
- Did we build the *right* system?

✦**Transition and DevOps**
- How will we get the solution into the hands of the user?
- How will we support it?

We iterate through these four processes until we deliver an effective solution to stakeholders on time, on budget with acceptable risk

There are four interdependent REALIZE processes

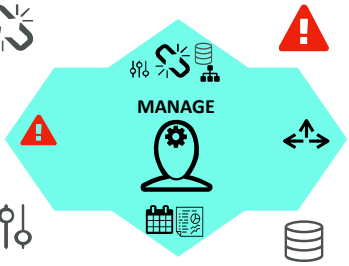
Copyright ©2020 Teaching Science and Technology Inc.

MANAGE Processes

✦Technical Planning

✦Requirements Management

✦Interface Management



✦Risk Management

✦Configuration Management

✦Technical Data Management

✦Technical Assessment

✦Decision Analysis

Copyright ©2020 Teaching Science and Technology Inc.

Copyright ©2020 Teaching Science and Technology Inc.

Space Systems Engineering Road Map—DESIGN

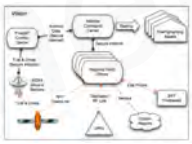
NEED:

The U.S. Forest Service needs....

GOALS/ OBJECTIVES:

1. Detect wildfires...
2. Notify USFS...

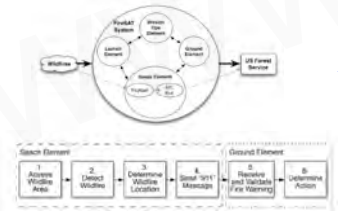
CONOPS:



Mission Scope

Defining Stakeholder Requirements

Functional Analysis



Requirements Engineering

System Requirements Document

3.1 Form
3.2 Fit
3.3 Function

Architecture Trade-offs

Defining the Technical Solution

System Context Element

Environments

Orbits & Trajectories

Derived & Allocated Requirements

Spacecraft Design

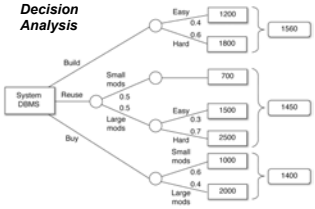
Derived & Allocated Requirements

Launch Element Selection

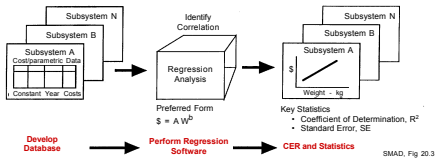
Copyright ©2020 Teaching Science and Technology Inc.

Space Systems Engineering Road Map—MANAGE

Decision Analysis



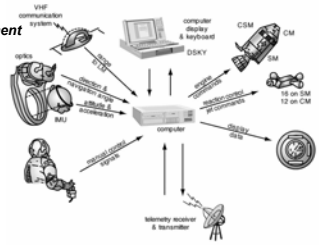
Cost Estimation



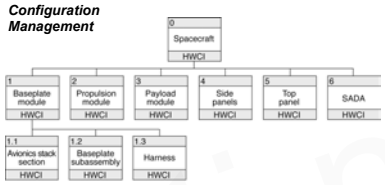
Deciding On and Assessing Solutions

Technical Risk Management

Interface Management



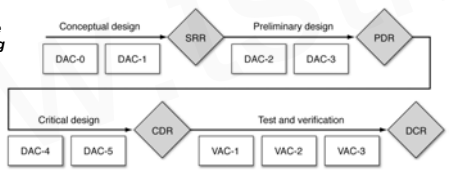
Configuration Management



Controlling Processes

Work Breakdown Structure

Design & Analysis Cycle (DAC) Planning



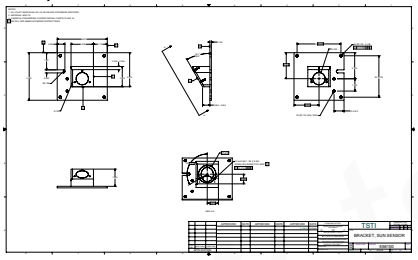
Technical Planning

Copyright ©2020 Teaching Science and Technology Inc.

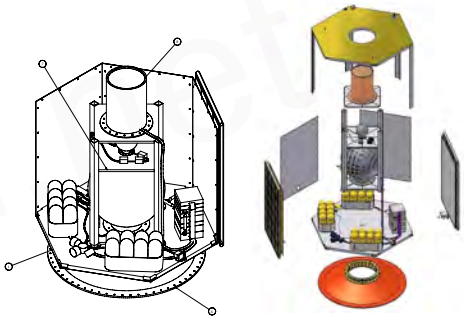
Copyright ©2020 Teaching Science and Technology Inc.

Space Systems Engineering Road Map—REALIZE

Implementation

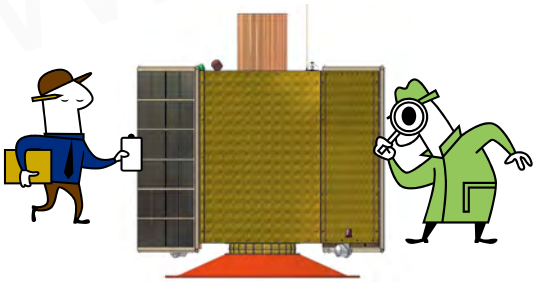


Integration



Implementing and Integrating the System

Verification & Validation



Transition

Verifying, Validating and Transitioning the System

Copyright ©2020 Teaching Science and Technology Inc.

The Role of the Systems Engineer

Copyright ©2020 Teaching Science and Technology Inc.

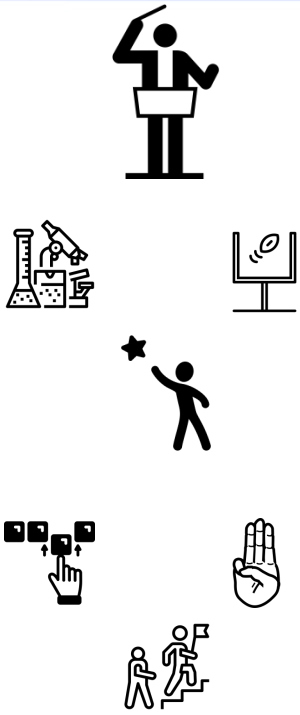
Copyright ©2020 Teaching Science and Technology Inc.

What Systems Engineers Do

✦The systems engineer is like the maestro, who knows what the music should sound like (the look and function of a design) and has the skills to lead a team in achieving the desired sound (meeting the system requirements).

✦Systems engineers:

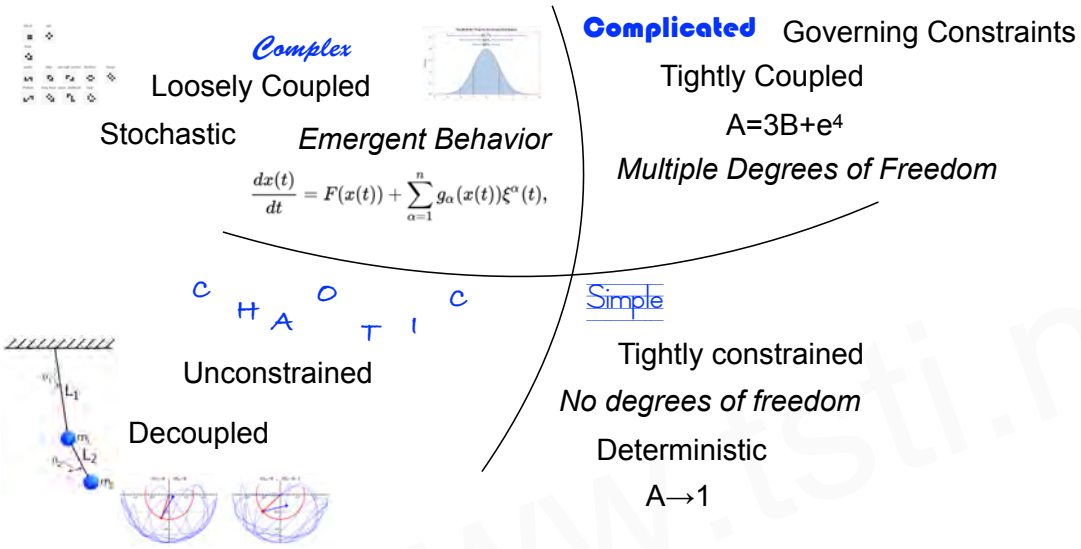
- Understand the fundamentals of mathematics, physics, and other pertinent sciences as well as the capabilities of various people and disciplines
- Have mastered a technical discipline and are familiar with multiple disciplines
- Must understand the end game and overall objectives of the endeavor
- Create a vision and approach for attaining the objectives
- May be architects or designers
- Select and shape the technical issues to be addressed by multidisciplinary teams
- Must often interpret and communicate objectives, requirements, system architecture, and design
- Are **responsible for the project's technical integrity**
- Organize and lead multidisciplinary teams
- Are responsible for the successful delivery of a complex product or service



Copyright ©2020 Teaching Science and Technology Inc.

Types of Systems

✦The Cynefin (kuh-NEV-in) framework describes systems in four domains

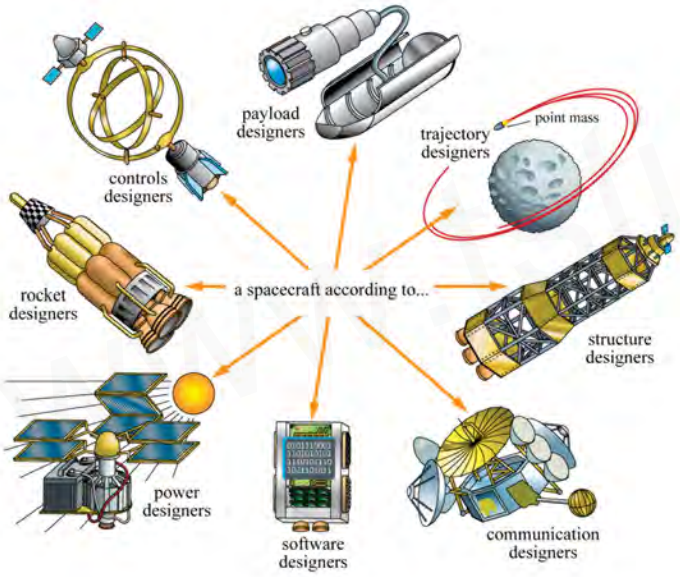


Systems are increasingly more *complex*

Copyright ©2020 Teaching Science and Technology Inc.

Copyright ©2020 Teaching Science and Technology Inc.

Be a "Systems Thinker"

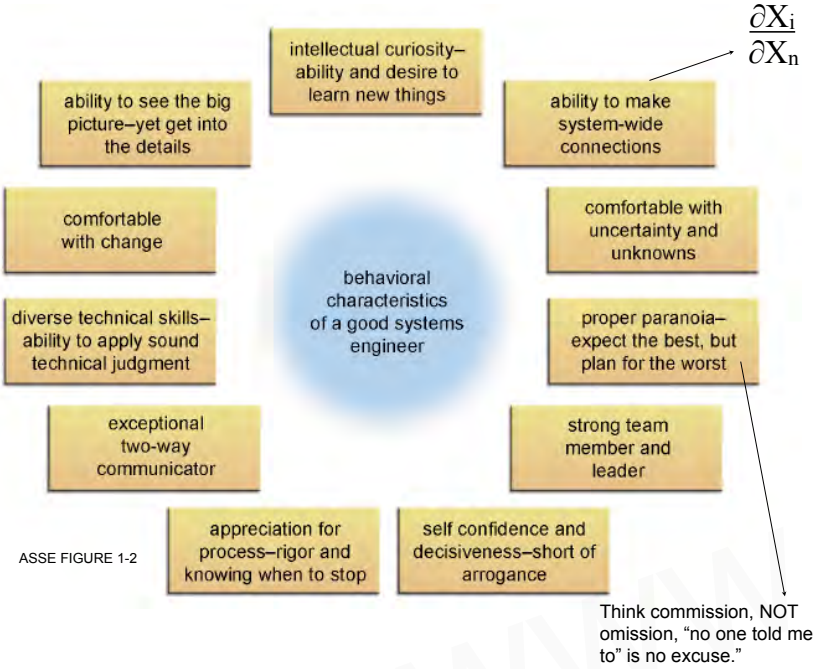


US: Fig. 11-34

US: Fig. 11-34

Copyright ©2020 Teaching Science and Technology Inc.

Characteristics of a Systems Engineer



ASSE FIGURE 1-2

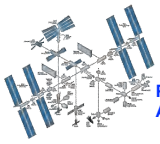
- **Architecting** encompasses planning, designing, and overseeing the development of a product or system—often buildings, bridges, and infrastructure as well as software.
- **Design** is creating a product or system, as well as a plan to develop and use it. For our purpose, architects provide the rules; designers create implementations using those rules.
- **Systems engineers do both**—help create the design and maintain its integrity throughout its lifecycle.

Copyright ©2020 Teaching Science and Technology Inc.

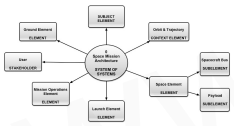
Copyright ©2020 Teaching Science and Technology Inc.

Architectures

Architectural Vision



Physical Architecture

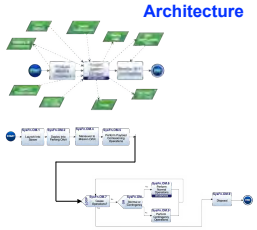


Verification Architecture

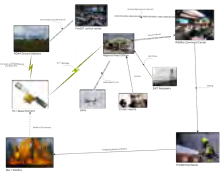
Requirements Architecture

- Systems only exist to deliver value to stakeholders
- Systems design begins with an ***architectural vision***
  - **Architecture** encompasses the fundamental organization of a system, embodied in its components, their relationships to each other and the environment, and the principles governing its design and evolution
  - **Design** is the process of creating products or systems to meet that vision, as well as a plan to develop it (Dev) and use it (Ops) [DevOps]
- *Architects* provide the "rules" for what the system(s) will do, how they'll look and how they'll be used
- *Designers* create solutions using those rules
- *Systems engineers often do both*
- Connecting architectural vision to design is done through a variety of a ***architectures***
- The information captured by these architectures can be organized into ***architectural frameworks***

Functional Architecture



Operational Architecture



Architectural Framework

UAF OMG UNIFIED ARCHITECTURE FRAMEWORK

UPDM

Copyright ©2020 Teaching Science and Technology Inc.

Typical Project Lifecycle

NASA Life Cycle Phases	Pre-Systems Acquisition	FORMULATION	Approval for Implementation	IMPLEMENTATION	Systems Acquisition	Operations	Decommissioning
Project Life Cycle Phases	Pre-Phase A: Concept Studies	Phase A: Concept & Technology Development	Phase B: Preliminary Design & Technology Completion	Phase C: Final Design & Fabrication	Phase D: System Assembly, Int & Test, Launch	Phase E: Operations & Sustainment	Phase F: Closeout
Project Life Cycle Gates & Major Events	KDP A; FAD; Draft Project Requirements	KDP B; Preliminary Project Plan	KDP C; Baseline Project Plan	KDP D	KDP E; Launch	KDP F; End of Mission	Final Archival of Data
Human Space Flight Project Reviews[1]	ASP MCR	ASM SRR SDR (PNAR)	PDR (NAR)	CDR / PRR[2] SIR	SAR ORR; Inspections and Refurbishment	FRR PLAR CERR[3]; End of Flight; PFAR	DR
Re-flights			Re-enters appropriate life cycle phase if modifications are needed between flights				
Robotic Mission Project Reviews[1]	MCR	ASM SRR MDR[4] (PNAR)	PDR (NAR)	CDR / PRR[2] SIR	ORR	FRR PLAR CERR[3]	DR
Launch Readiness Reviews						SMSR, LRR (LV), FRR (LV)	
Supporting Reviews	Peer Reviews, Sub-system PDRs, Sub-system CDRs, and System Reviews						

FOOTNOTES

1. 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 Project Plan
2. PRR needed for multiple system copies. Timing is notional.
3. CERRs are established at the discretion of Program Offices.
4. For robotic missions the SRR and the MDR may be combined.

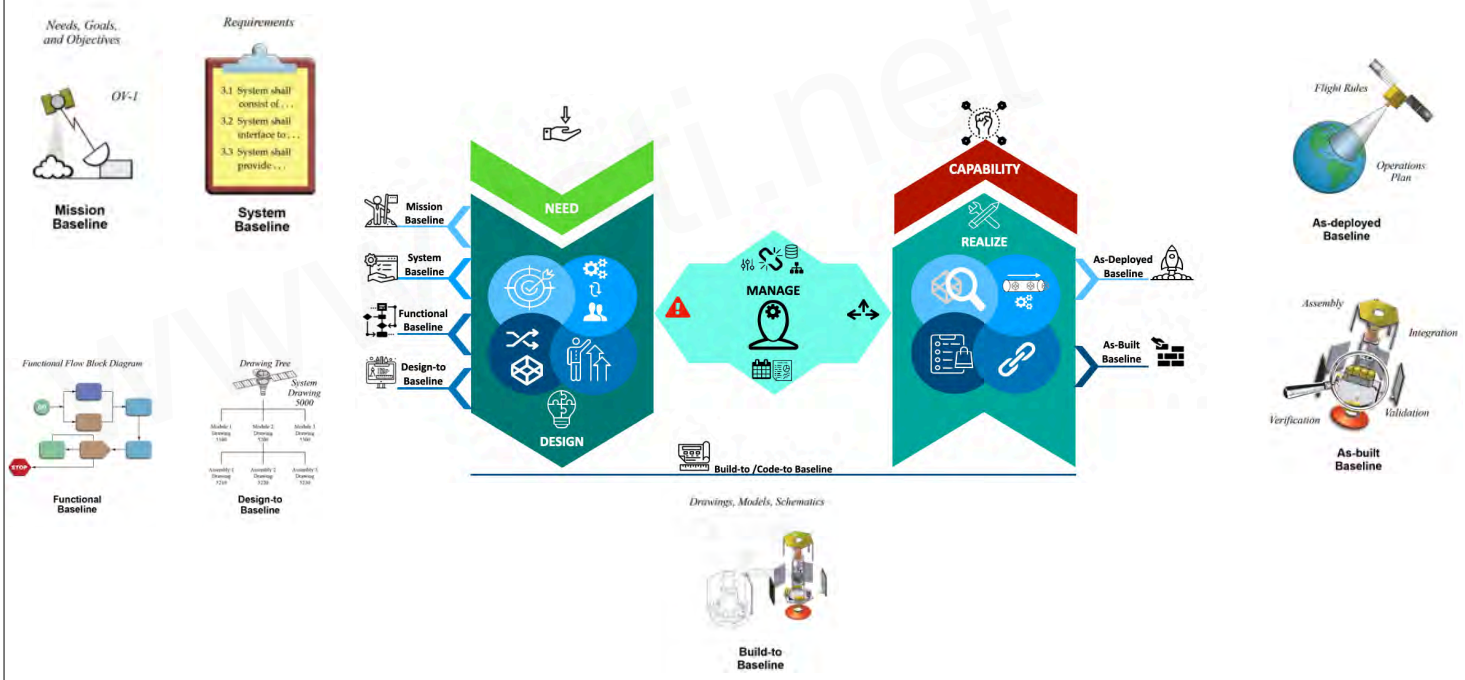
ACRONYMS

ASP—Acquisition Strategy Planning Meeting
ASM—Acquisition Strategy Meeting
CDR—Critical Design Review
CERR—Critical Events Readiness Review
DR—Decommissioning Review
FAD—Formulation Authorization Document
FRR—Flight Readiness Review
KDP—Key Decision Point
LRR—Launch Readiness Review
MCR—Mission Concept Review
MDR—Mission Definition Review
NAR—Non-Advocate Review
ORR—Operational Readiness Review
PDR—Preliminary Design Review
PFAR – Post-Flight Assessment Review
PLAR—Post-Launch Assessment Review
PNAR—Preliminary Non-Advocate Review
PRR—Production Readiness Review
SAR – System Acceptance Review
SDR—System Definition Review
SIR—System Integration Review
SMSR—Safety and Mission Success Review
SRR—System Requirements Review

Copyright ©2020 Teaching Science and Technology Inc.

Copyright ©2020 Teaching Science and Technology Inc.

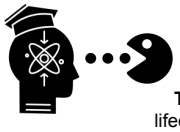
Evolving Technical Baselines



Copyright ©2020 Teaching Science and Technology Inc.

Design Maturity—Technical Baselines

Systems Engineering is like a "*knowledge game*." The object of the game is to get "*smarter*" over time. How smart we are is defined by maturity metrics.



Baseline	Representative Technical Artifacts	Established at
Mission	Need, goals, and objectives; Mission requirements	MCR
System	Concept of operations; System requirements; SEMP; IMP; Natural environment definition; Safety and mission assurance plan	SRR
Functional	Baseline configuration; Interface requirements; Allocated requirements; Test and evaluation master plan; Software management plan; Technology development plan	SDR
Design-to	Component specifications; Software specifications; Ground support equipment specifications; Interface control documents; Verification plans; Manufacturing plan	PDR
Build-to	Engineering drawings; Operational limits and constraints; Manufacturing process requirements; Acceptance plans and criteria; Verification procedures; Integration and assembly plan; Test plans; Training plan	CDR
As-built	End products; Enabling products; Acceptance data package; Operations procedures; In-flight checkout plan	SAR
As-deployed	Integration and test anomaly resolutions; Mission support training and simulation results; Scientific instrument calibration results; System activation results	ORR FRR PLAR

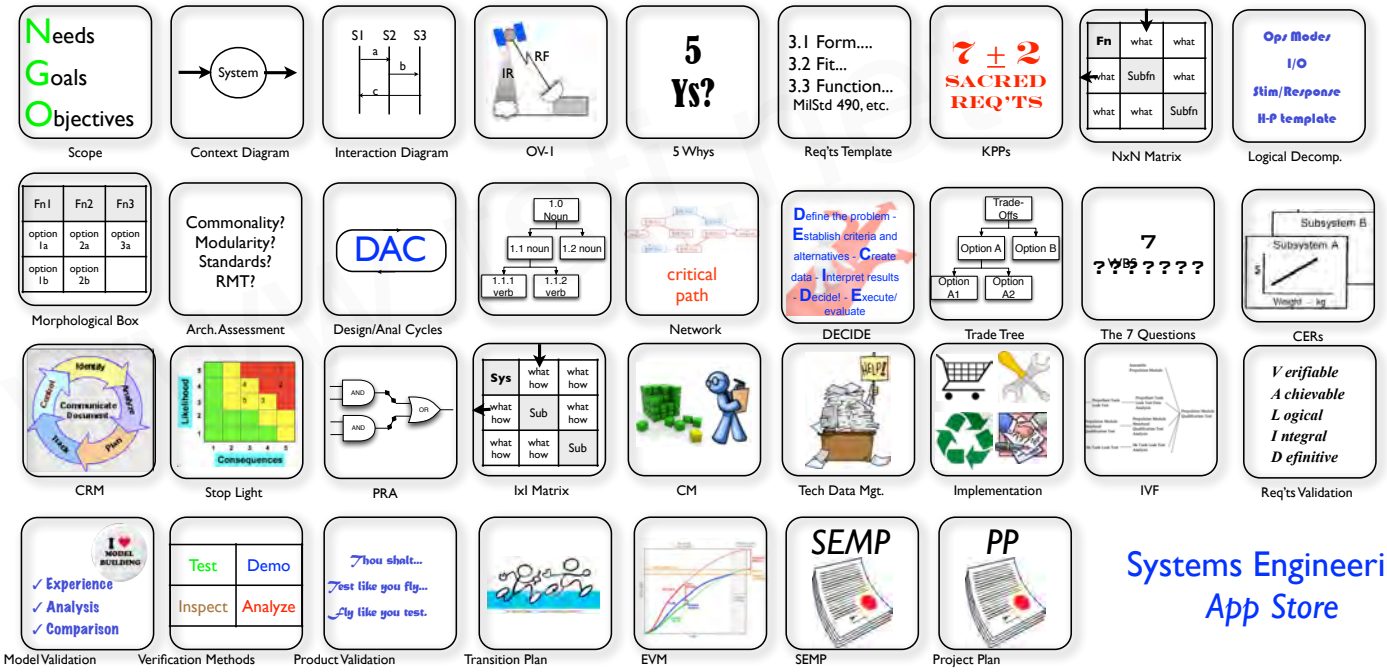
ASSE FIGURE 1-7

Typical Technical Baselines for Space Systems and Missions. Our discussion centers on seven baselines in the project lifecycle. Each has a set of artifacts or documents that represent its status. These baselines are established at certain milestone reviews. (MCR is mission concept review; SRR is system requirements review; SDR is system design review; PDR is preliminary design review; CDR is critical design review; SAR is system acceptance review; ORR is operational readiness review; FRR is flight readiness review; PLAR is post-launch assessment review.)

Copyright ©2020 Teaching Science and Technology Inc.

Copyright ©2020 Teaching Science and Technology Inc.

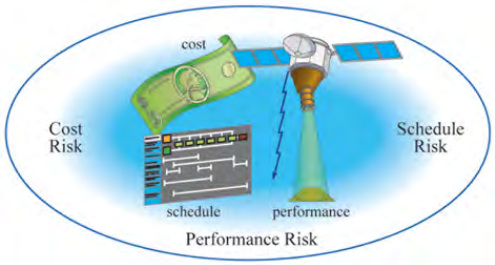
Systems Engineering "App Store"



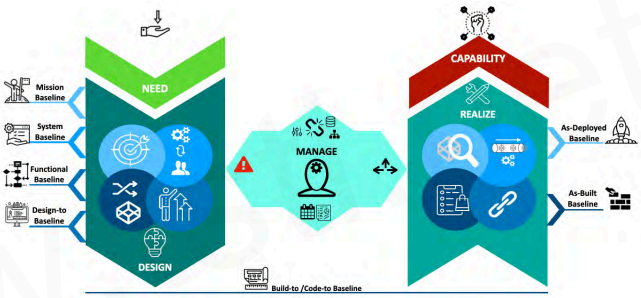
Copyright ©2020 Teaching Science and Technology Inc.

Key Take-Aways

- A *system* is a set of interrelated components working together to accomplish a common purpose
- The systems engineer is like the maestro in a orchestra
  - Among their varied roles they are responsible for the project's technical integrity
  - Among their varied characteristics is an ability to make system-wide connections
- Systems Engineering is both an art and a science
  - Technical Leadership is the art of systems engineering, Systems Management is the science



The Systems Engineering Universe



For this course, we'll view Systems Engineering as a *Framework* of highly inter-related activities with the goal of creating real products delivering real capabilities (not just churning out documents)

Copyright ©2020 Teaching Science and Technology Inc.

Copyright ©2020 Teaching Science and Technology Inc.

Notes

www.tsti.net

Copyright ©2020 Teaching Science and Technology Inc.

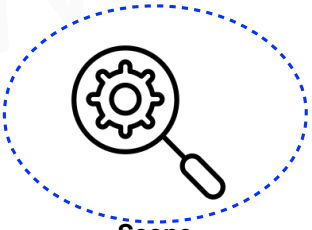
Notes

www.tsti.net

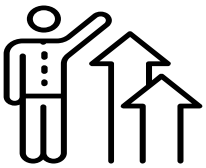
Copyright ©2020 Teaching Science and Technology Inc.

Copyright ©2020 Teaching Science and Technology Inc.

DESIGN



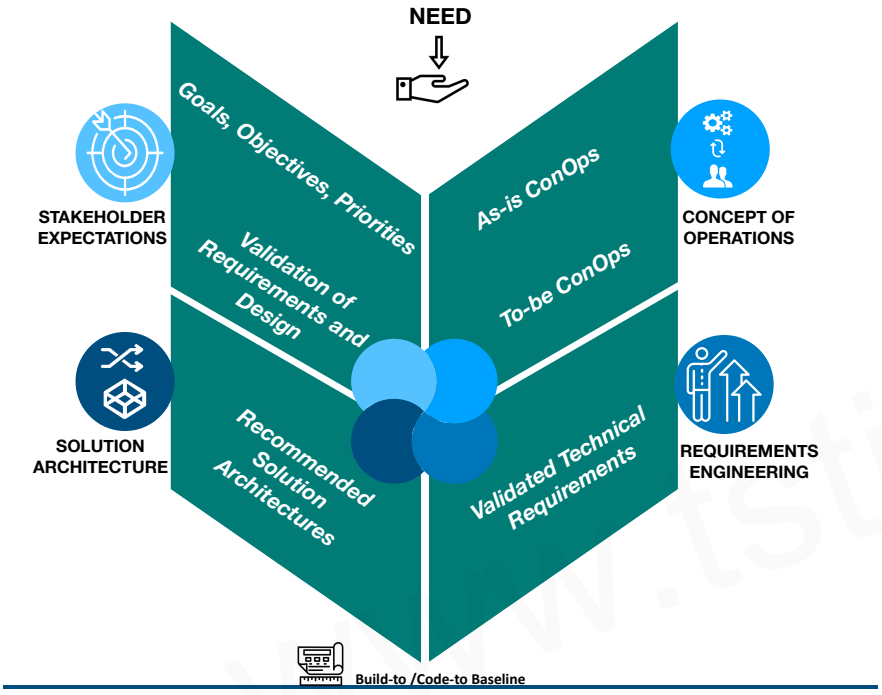
Scope



Requirements

Solution Architecture

DESIGN—Value Stream

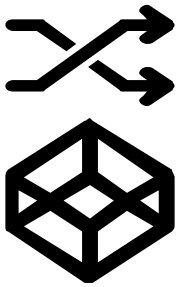


DESIGN PROCESSES

VALUE STREAM

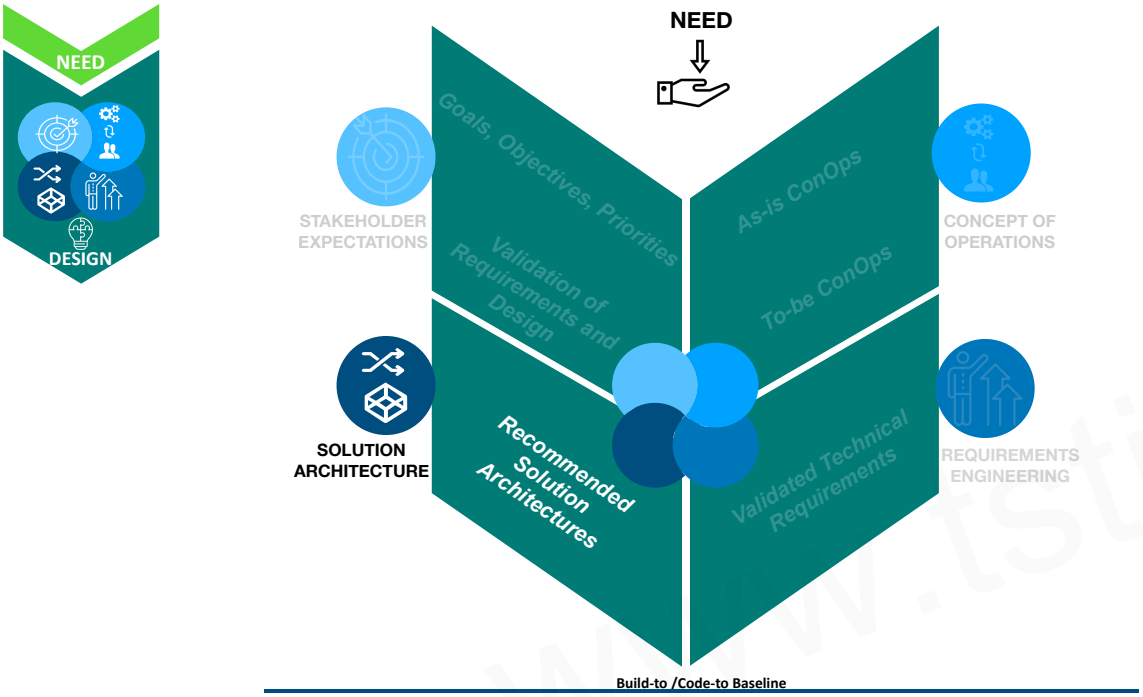
Copyright ©2020 Teaching Science and Technology Inc. 34

Solution Architecture



Copyright ©2020 Teaching Science and Technology Inc.

We are Here



Copyright ©2020 Teaching Science and Technology Inc.

Copyright ©2020 Teaching Science and Technology Inc.

Solution Architecture—Objectives and Agenda

✦Learning Objectives

- At the end of this section you should be able to…
  - Describe the elements of functional and physical architectures and how they relate to each other (e.g. functions *performed by* physical)
  - Explain various approaches to functional architecture development
  - Apply systems engineering tools and techniques to capture and communicate functional and physical architectures

✦Agenda

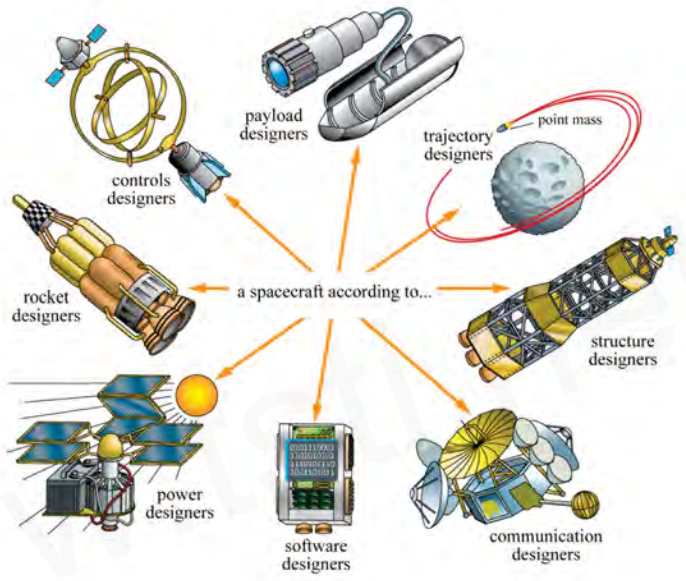
- Creating Solution Architectures
- Defining Functional Partitions
  - Static Approaches
  - Dynamic Approaches
- Design Solution—Physical Architecture
- Allocation
  - Functional to Physical Allocation
  - Requirements Allocation
- Assessing Architectures

Copyright ©2020 Teaching Science and Technology Inc.

Creating Solution Architectures

✦A general process to create the functional and physical architecture for a system

Step	Description
1	Specify the system context
2	Define the functional partitions
3	Create the physical partitions and allocate functions to the physical components
4	Assess the architecture
5	Allocate requirements to the components and generate detailed specifications
6	Trace requirements



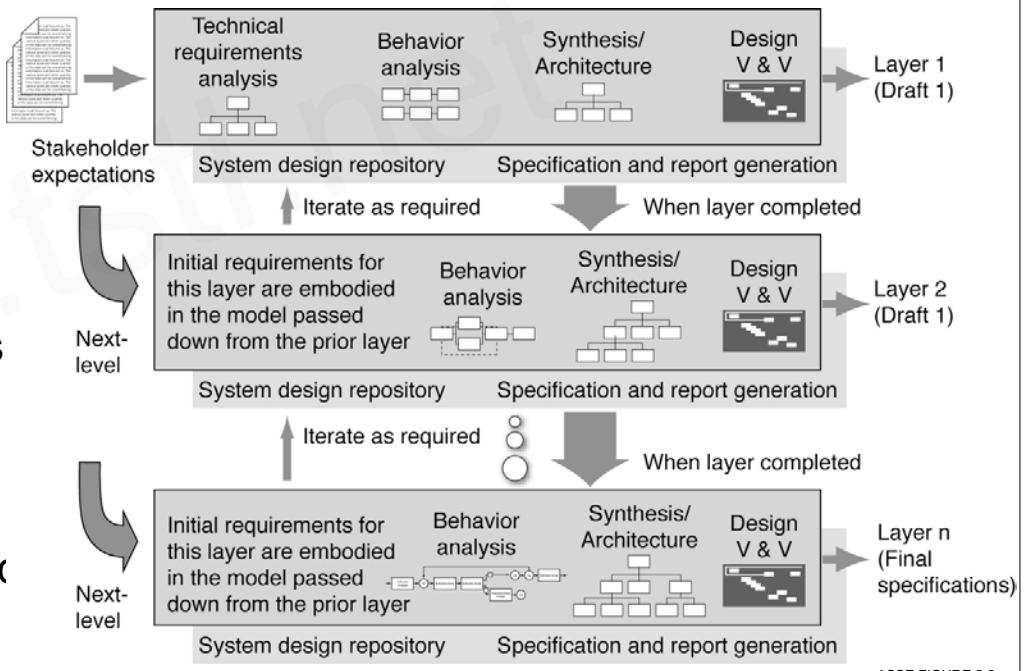
US: Fig. 11-34

Copyright ©2020 Teaching Science and Technology Inc.

Copyright ©2020 Teaching Science and Technology Inc.

Creating Solution Architectures (cont'd)

Recursive Partitioning of Functions and Physical Components. This process repeats and may advance in layers—top-down, bottom-up, or some combination of both. (V&V is verification and validation.)



ASSE FIGURE 5-2

Copyright ©2020 Teaching Science and Technology Inc.

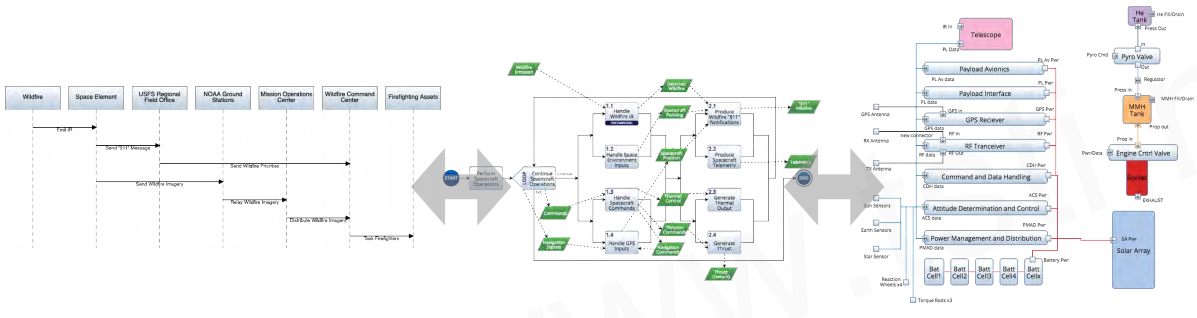
Creating Solution Architectures (cont'd)

- Architecture Viewpoints and Work Products
  - By developing these three architectures at the same time, we gain insights and produce a more complete solution. (IDEFO is integrated definition for function modeling.)

Operational architecture
- Shows how the operator will use the system
- Shows inputs and outputs to users and other systems
- Usually described by:
  - Operational concept
  - Context diagram
  - Use cases and scenarios
  - Sequence diagrams

Functional architecture
- Defines the capabilities, the services, or the functions provided by the system
- Shows the messages and data between functions
- Usually described by:
  - IDEFO diagrams
  - Data flow, functional flow, and N2 diagrams
  - Classes, methods, use cases, etc.

Physical architecture
- Allocates resources (hardware and software)
- Shows the interconnections between the resources
- Usually described by:
  - Physical block diagrams
  - Physical interface definitions



Iteratively define all three views and allocate the requirements to the elements within these views to 'completely' define the architecture.

Updated ASSE FIGURE 5-4

Copyright ©2020 Teaching Science and Technology Inc.

Copyright ©2020 Teaching Science and Technology Inc.

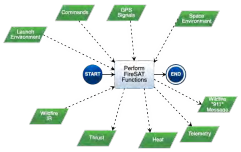
Defining Functional Partitions

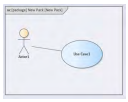
- Developing the system's functional architecture is the first step in framing how a system will meet its requirements and operational scenarios
  - To do so, we must understand each function as a process that transforms inputs into outputs
  - A function describes an action from the system or one of its elements, so it should begin with a verb
  - We must not confuse the thing that performs the function with the function itself
    - For example, a database is not a function, but we can use one to perform the function of Store and retrieve data. Accordingly, the following are not functions:
      - Computers or software applications—we assign them the functions of Process data or Compute current navigation state
      - Cold-gas thrusters—they provide the function of Generate thrust
      - Batteries—they have the function Provide electrical power
- The functional architecture describes how the system meets its input and output requirements, and more specifically "what the system does." So it fills two roles:
  - A partition of the system into more detailed functions
  - A model indicating how the system transforms its inputs into outputs and achieves its ability to realize required behaviors

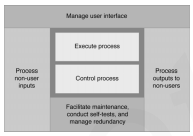
Copyright ©2020 Teaching Science and Technology Inc.

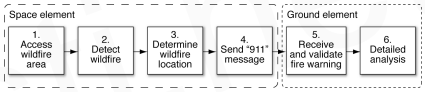
Approaches to Functional Partitioning

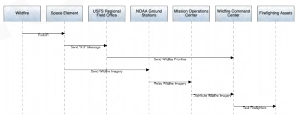
- There are numerous techniques for performing functional analysis (which can lead to confusion and 'analysis paralysis')
- These techniques, and ways to model them, include…
  - "Static" Methods
    - **Inputs / Output Analysis—Context Diagram**
    - **Use Case Diagrams**
    - **Fault Tree Analysis**
    - **Hatley-Pirbhai template**
    - **Using functional requirements**
    - **Matching the physical architecture**
    - Using organizational structure
    - Defining stimulus-response threads (cases or operational scenarios)
  - "Dynamic" Methods
    - **Operational modes/scenarios using Action / Activity / FFBD Diagrams / Timelines**
    - **Operational modes/scenarios using Sequence Diagrams**
    - **Operational modes/scenarios using IDEF0 Diagrams**
    - **Integrated Functional Architecture using EFFBD**
- We'll look at examples of techniques in bold









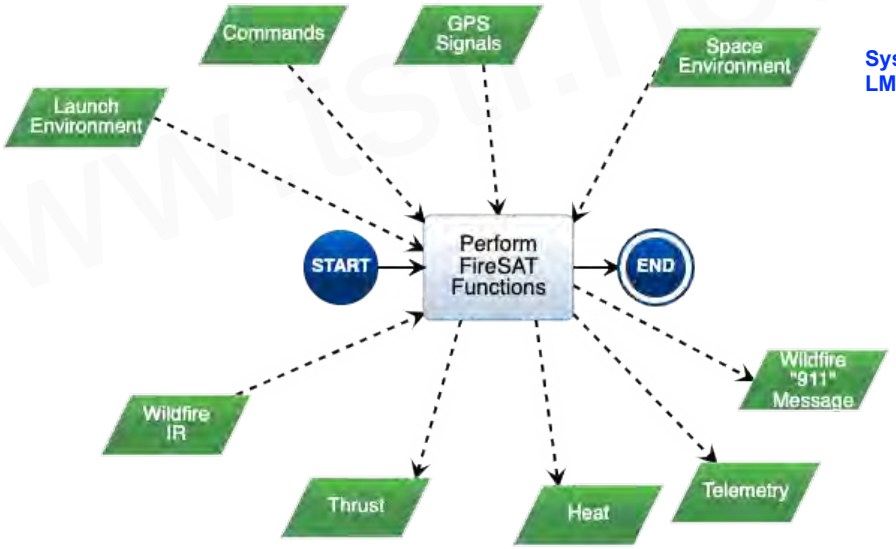


Copyright ©2020 Teaching Science and Technology Inc.

Copyright ©2020 Teaching Science and Technology Inc.

Static Functional Partitioning: Input/Output Analysis

- A Context Diagram is a powerful tool to capture inputs/outputs and define the scope (boundary) of the system



System Context Diagram using LML Action Diagram

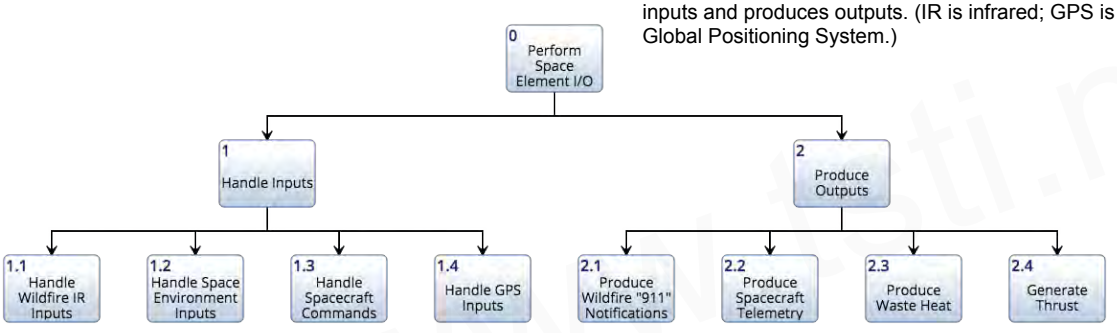
Modified ASSE FIGURE 3-3

Copyright ©2020 Teaching Science and Technology Inc.

Static Functional Partitioning: Input/Output Analysis (cont'd)

- From the context diagram view, we can then organize the inputs / outputs using a hierarchy chart and assign internal sub-functions to "handle" or "produce" them.
  - These sub-functions can then be further decomposed into lower level functions.
  - You could also add an additional sub-function "Perform Internal Processes" for functions that stay inside the system.

This partitioning method emphasizes a system's external interfaces by focusing on how it handles inputs and produces outputs. (IR is infrared; GPS is Global Positioning System.)



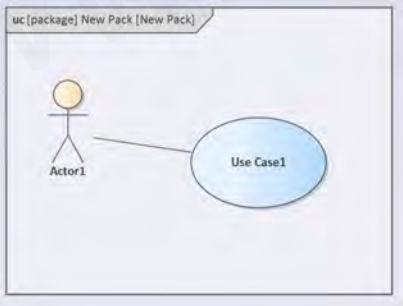
Revised ASSE FIGURE 5-14

Copyright ©2020 Teaching Science and Technology Inc.

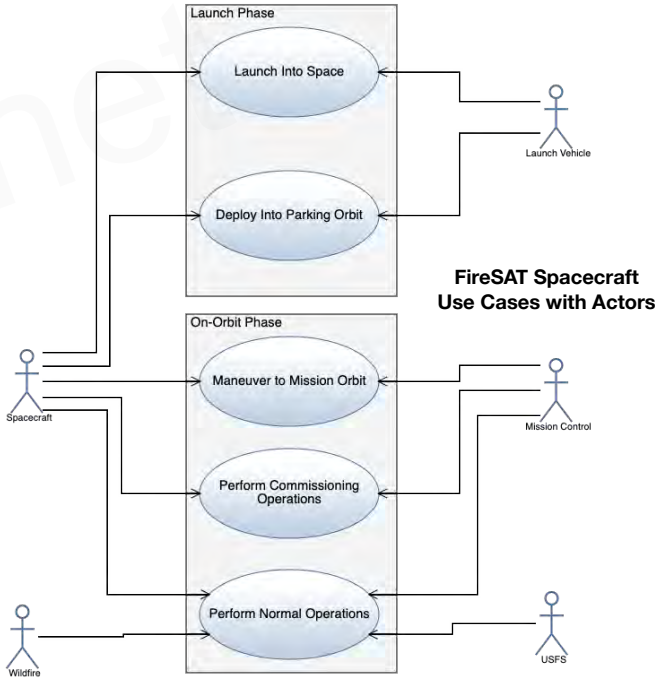
Copyright ©2020 Teaching Science and Technology Inc.

Static Functional Partitioning: Use Case Analysis

- Use Cases capture unique hardware/software system configurations along with unique interactions with actors (users or other systems)



Generic Use Case Diagram



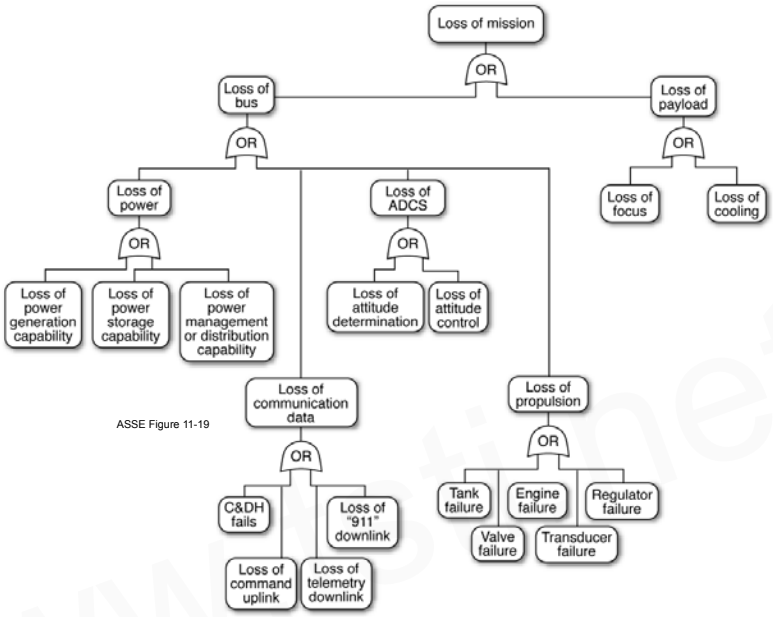
FireSAT Spacecraft Use Cases with Actors

Copyright ©2020 Teaching Science and Technology Inc.

Static Functional Partitioning: Fault Tree Analysis (FTA)

- Fault tree analysis (FTA) is a graphical tool to explore the causes of system level failures.
- It uses boolean logic to combine a series of lower level events and it is basically a top-down approach to identify the component level failures (basic event) that cause the system level failure (top event) to occur.
- Fault tree analysis consists of two elements “events” and “logic gates” which connect the events to identify the cause of the top undesired event.

Adapted from: https://sixsigmastudyguide.com/fault-tree-analysis/



ASSE Figure 11-19

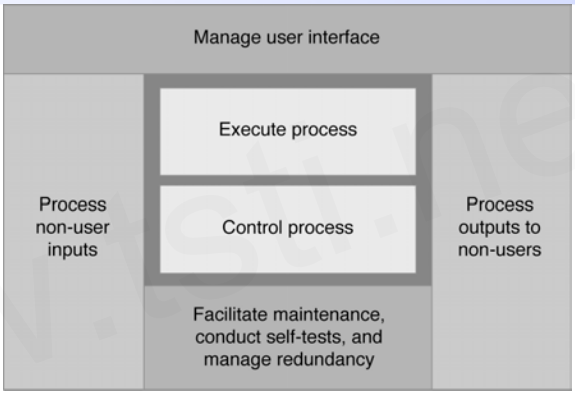
FireSAT Fault Tree. We derive a fault tree from an understanding of the mission concept of operations and the underlying system functionality. The concept of operations, together with the fault tree, guides overall validation planning. (ADCS is attitude determination and control subsystem; C&DH is command and data handling subsystem.)

Copyright ©2020 Teaching Science and Technology Inc.

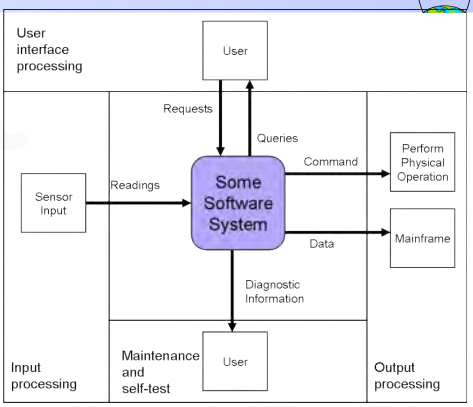
Copyright ©2020 Teaching Science and Technology Inc.

Static Functional Partitioning: Hatley-Pirbhai Template

The Hatley−Pirbhai template is a "generic" functional architecture, or design pattern, that states that any system can be functionally partitioned into the sub-functions shown here (though we may combine or eliminate some of them depending on the application).

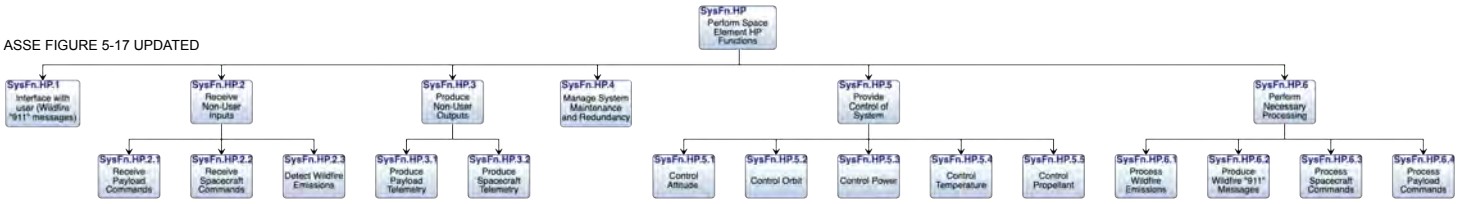


ASSE FIGURE 5-16



https://en.wikipedia.org/wiki/Hatley–Pirbhai_modeling

ASSE FIGURE 5-17 UPDATED



Copyright ©2020 Teaching Science and Technology Inc.

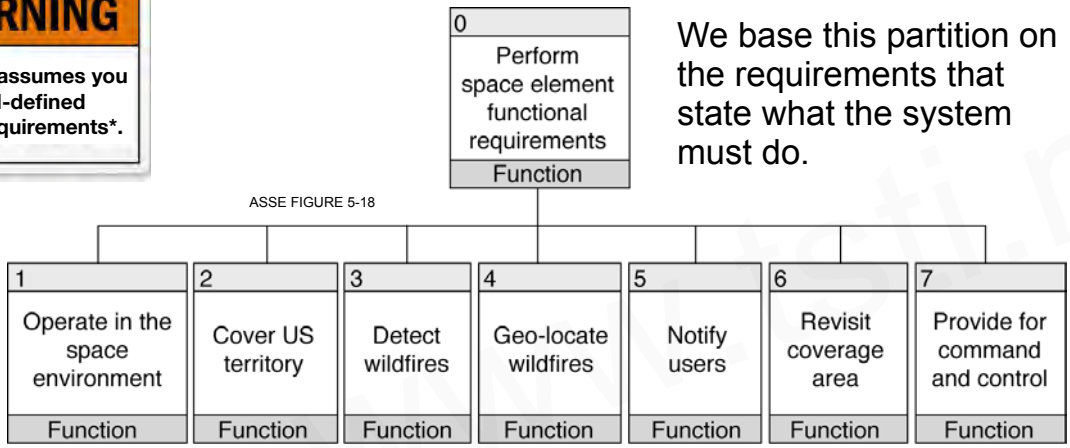
Static Functional Partitioning: Based on Functional Requirements

- Because a system's functional requirements describe what it must do, we can use them to determine its sub-functions.
  - Requirements and architectures develop iteratively, so we use the same method for lower−level functions.
  - Each subsystem's functional requirements help us define its sub-functions.

⚠ WARNING

This method assumes you have well-defined functional requirements*.

***It is strongly suggested that you first develop your functional architecture using one of the other methods before writing functional requirements!**



ASSE FIGURE 5-18

We base this partition on the requirements that state what the system must do.

Copyright ©2020 Teaching Science and Technology Inc.

Copyright ©2020 Teaching Science and Technology Inc.

Static Functional Partitioning: Based on Physical Architecture

If we know a system's physical architecture, we can allocate functions to physical resources—hardware, software, and people.

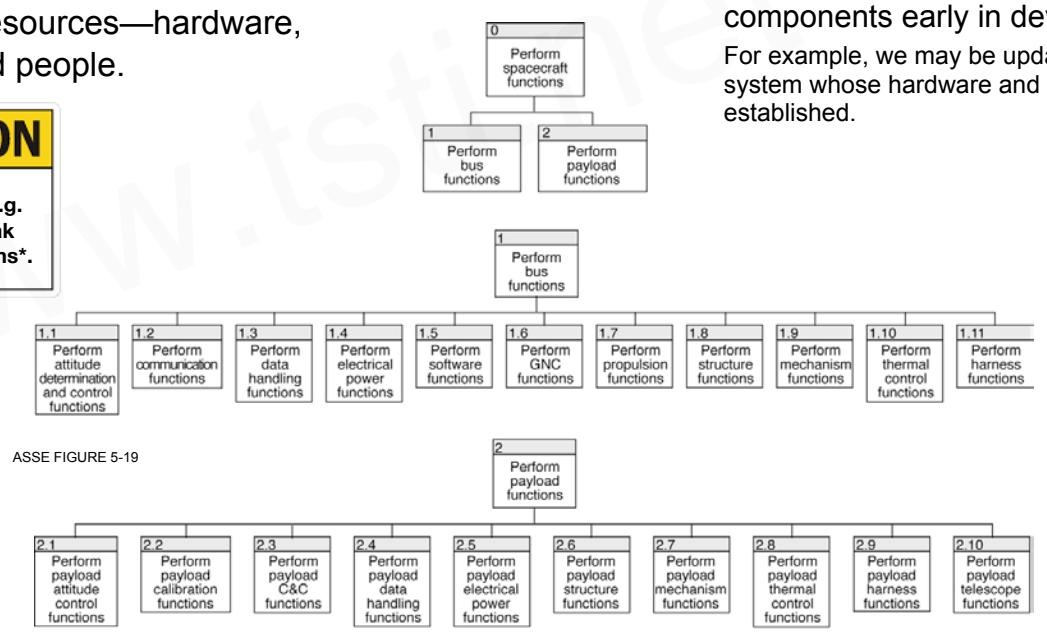
⚠CAUTION

Some functions (e.g. software) may leak between subsystems*.

***It is strongly suggested that you develop additional functional architectures using one of the other methods as a check against this method.**

Several conditions could enable architects to know these physical components early in development.

For example, we may be updating a legacy system whose hardware and software are established.

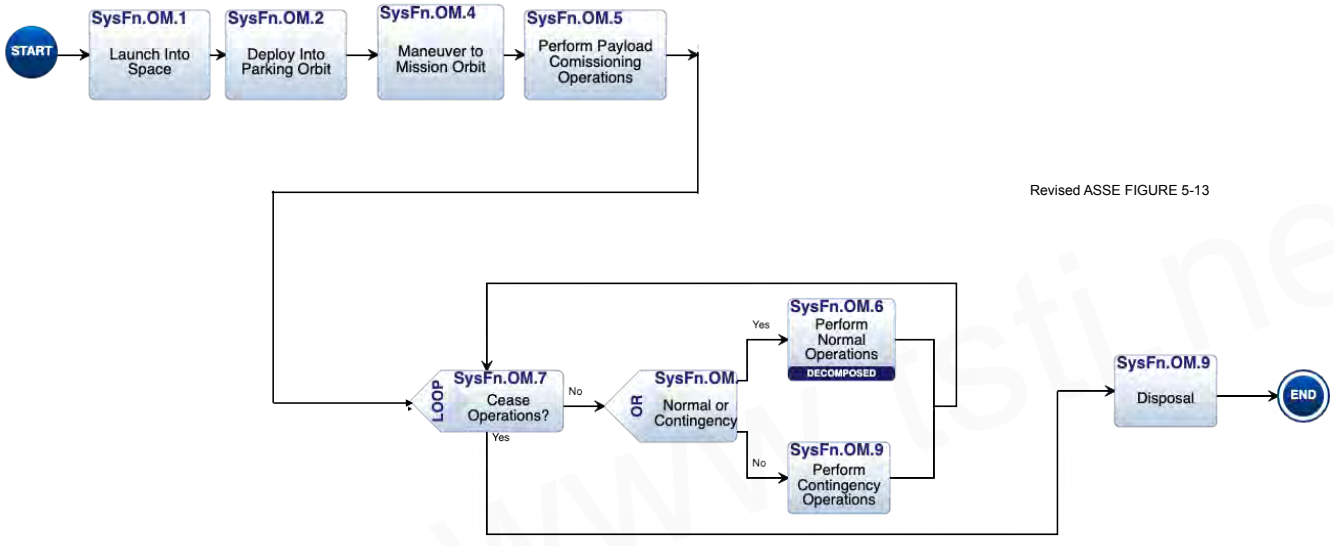


ASSE FIGURE 5-19

Copyright ©2020 Teaching Science and Technology Inc.

Dynamic Functional Partitioning: Operational Modes using Activity Diagrams

Functional Flow Block Diagram Based on Operational Modes. This diagram helps us visualize the logical sequencing of system functions. (LP is loop; OR is an "or" decision point; LE is loop exit.)



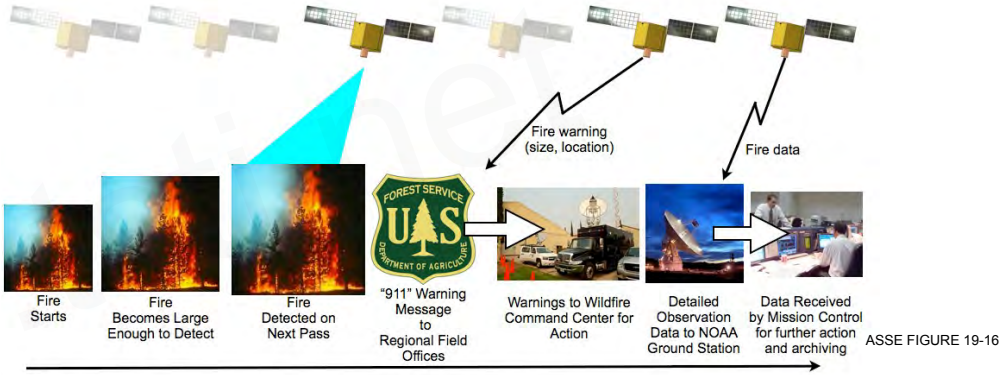
Revised ASSE FIGURE 5-13

Copyright ©2020 Teaching Science and Technology Inc.

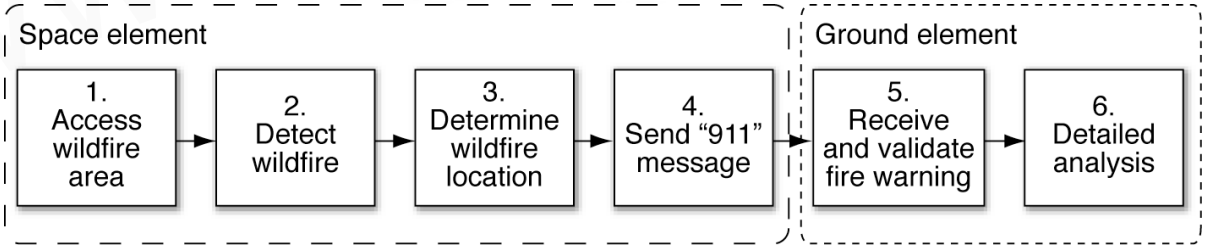
Copyright ©2020 Teaching Science and Technology Inc.

Dynamic Functional Partitioning: Operational Modes using Activity Diagrams

Simple Storyboard of Timeline for "Detect Wildfire" Capability. Pictures quickly capture a simple scenario that shows how a fire starts, becomes larger, and is detected by the FireSAT system, which then alerts NOAA and the firefighters.



ASSE FIGURE 19-16

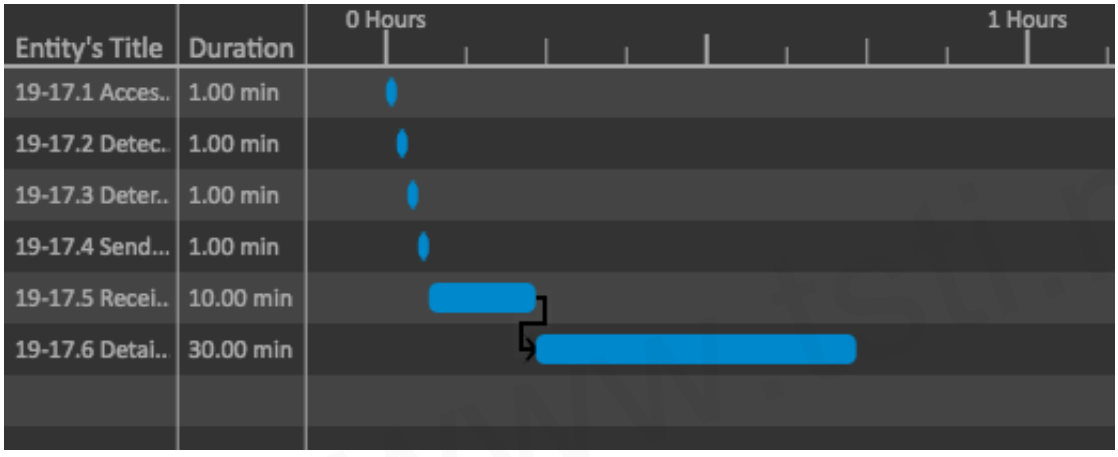


ASSE FIGURE 19-17

Block Diagram for an Operational Scenario That Responds to the "Detect Wildfire" Capability.

Copyright ©2020 Teaching Science and Technology Inc.

Dynamic Functional Partitioning: Timeline Analysis



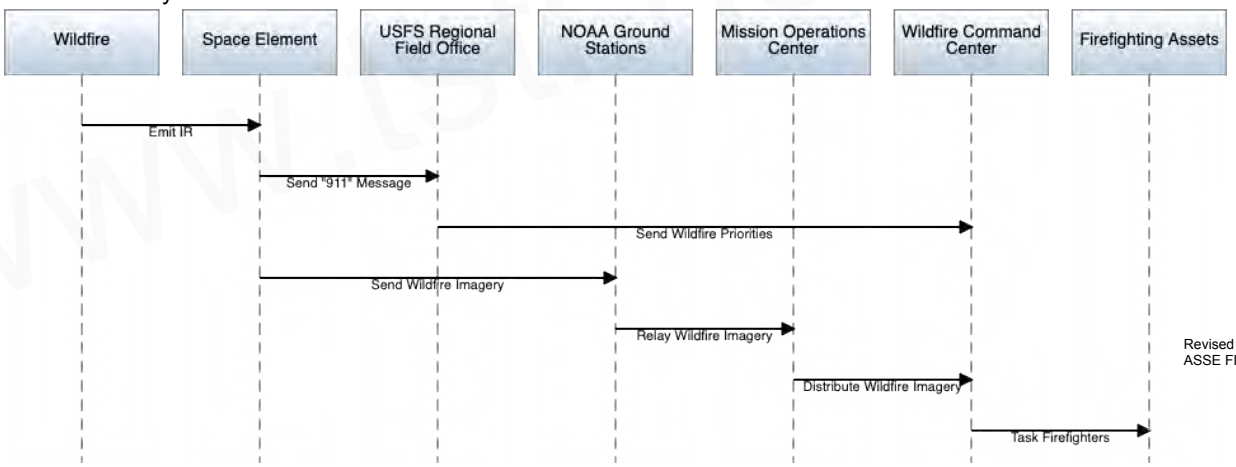
Copyright ©2020 Teaching Science and Technology Inc.

Copyright ©2020 Teaching Science and Technology Inc.

Dynamic Functional Partitioning: Operational Scenarios using Sequence Diagrams

✦We represent each stakeholder expectation by one or more use cases that describe a session (or a partial session) that a set of active stakeholders, such as the users, has with the system

- The use-case scenarios are developed from the concept of operations—sometimes they're even defined within it—and they document how the system will be used

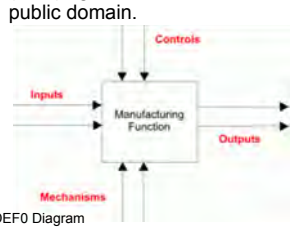


The Wildfire "911" Scenario. At the System Level we show the inputs and outputs of the scenario for the system in this interaction diagram.

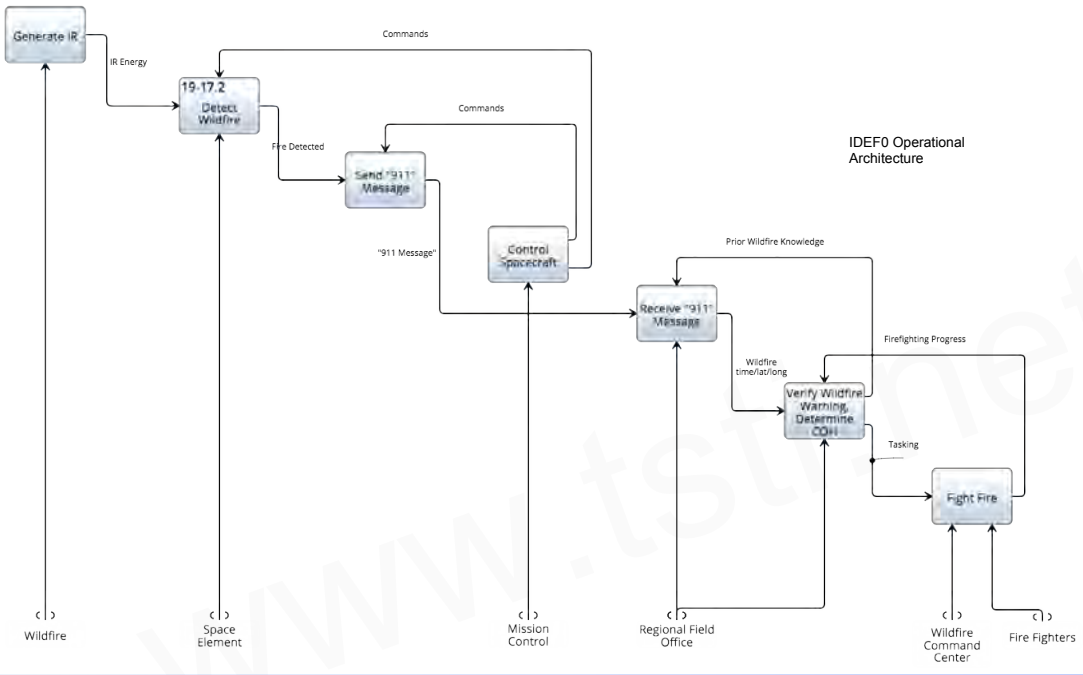
Copyright ©2020 Teaching Science and Technology Inc.

Dynamic Functional Partitioning: Operational Scenarios using IDEF0 Diagrams

- One of the first widely used modeling language was Integrated Computer-Aided Manufacturing (ICAM) Definition language
  - Began in the 1970s at the US Air Force Materials Laboratory, Wright-Patterson Air Force Base in Ohio and completed in the 1980s. Renamed Integration DEFinition (IDEF) in 1999.
  - As a modeling language, IDEF cover a wide range of applications for capturing systems engineering products.
  - It was developed under funding from U.S. Air Force and is still in use by US Department of Defense (DoD) agencies, and are in the public domain.



Basic IDEF0 Diagram



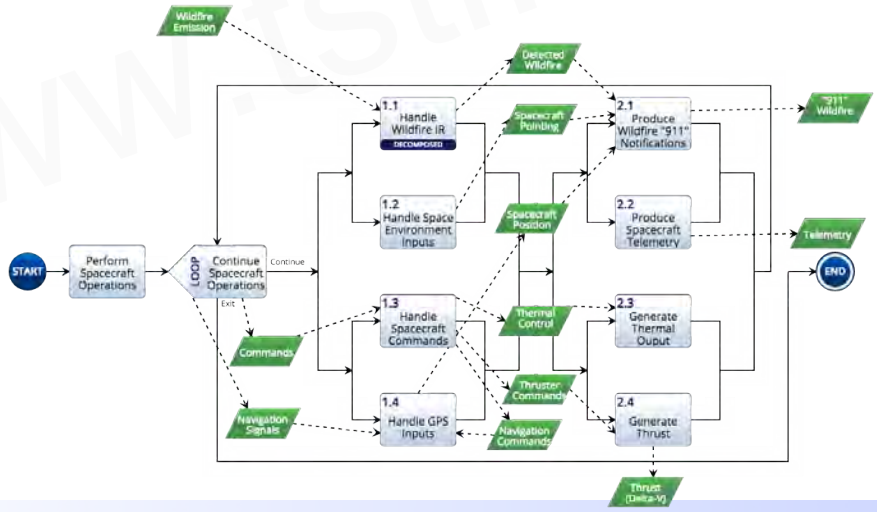
IDEF0 Operational Architecture

Copyright ©2020 Teaching Science and Technology Inc.

Copyright ©2020 Teaching Science and Technology Inc.

Dynamic Functional Partitioning: Integrated Functional Architecture

- From Multiple Scenarios to an Integrated Architecture of Behaviors and Functions—an Enhanced Functional Flow Block Diagram (EFFBD)
  - Here we show a complete functional architecture that includes the interactions from all the individual scenarios as well as the interactions between the scenarios.
    - The boxes are functions, the ovals are inputs, commands and outputs, and the circles and lines represent the logical execution flow of the functions.
    - The "LP" flow circle represents an execution loop and "LE" represents the loop exit (condition when the execution flow exits the loop). (IR is infrared; GPS is Global Positioning System.)



Revised
ASSE FIGURE 5-24

Copyright ©2020 Teaching Science and Technology Inc.

Method Benefits/Shortcomings

Method	Benefits of Method	Shortcomings of Method
Use operating modes	•Easy for operators and customers to understand •Addresses all major functions	•May hide opportunities to reuse functions across similar operating modes •Gets very complicated if operating modes interact a lot
Use inputs and outputs	•Addresses (conserves) all inputs and outputs (I/O) •Provides common functions for similar types of I/O •Good for functions that don't handle a lot of complicated processing or control	Doesn't cover complicated control and processing, so we may need to combine this method with another for complex functions
Use stimulus-response threads (use cases or operational scenarios)	•Matches the operational architecture and thus the users' point of view •Describes all uses of the system •Simple to understand if scenarios don't interact a lot	•May hide opportunities to reuse functions across scenarios •Can get very complicated if many scenarios or similar scenarios exist
Segment into fundamental and custodial functions	•Is logical •Used in many systems because custodial functions differ from those for normal operations	•Partitions only part of the system's functions •Must use with other methods for complete partitioning
Apply the Hatley-Pirbhai template	•A simple method that applies to many different types of functions •Effectively partitions at a high level	•Doesn't offer detailed understanding of processing or control functions •May need another method at a lower level to understand processing

ASSE Table 5-3

Copyright ©2020 Teaching Science and Technology Inc.

Copyright ©2020 Teaching Science and Technology Inc.

Benefits/Shortcomings (cont'd)

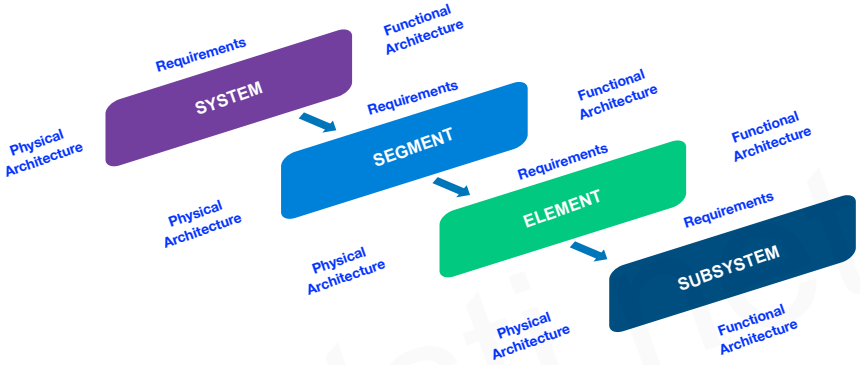
Method	Benefits of Method	Shortcomings of Method
Use the function states	•Very good to describe functions with multiple steps •Similar to a state machine that is familiar to many engineers •Easy for most people to comprehend •Can help architects discover the details of functions not completely known	•Architects must understand the start and end states up front •Gets complicated for long processing strings or complex functions
Use processing rates	•Is simple •Matches how functions will apply, because hardware or software separates them by processing rates	•Partitions only part of the system's functions •Must use with other methods for complete partitioning
Use organizational structure	•Matches how functions will apply, because they eventually allocate to components that different organizations will handle •Used in organizations whose structure is set or difficult to change	Because it forces functions to meet the organization's requirements, may inhibit creativity or trade-offs for a more efficient solution
requirements	Good when functional requirements are well-defined and complete	May develop a partial solution, because functional requirements don't encompass nonfunctional characteristics
Consider stakeholder priorities	•Enables effective support and maintenance of systems •Supports giving customers the "best bang for the buck" by creating an architecture that meets their priorities	•Partitions only part of the system's functions •Must use with other methods for complete partitioning
Match the physical architecture	•Matches how functions will apply •Normally used (for all or part) when we must incorporate a legacy system, commercial-off-the-shelf items, or purchased products	Because it forces one-to-one mapping of the functional and physical architectures, may inhibit creativity or trade-offs for a more efficient solution

ASSE Table 5-3

Copyright ©2020 Teaching Science and Technology Inc.

Design Solution—Resulting in the Physical Architecture

- ✦Develops alternative physical implementations addressing requirements and functional architecture including:
  - Product, process and people entities
  - Internal and external interfaces
  - Reiterate requirements and functional architecture if they are technically infeasible
  - Integrates with the program decision processes
- ✦Important aspects
  - Interoperability and modular open systems approach
  - Up and downward traceability of requirements
  - Product demonstrations according to applicable review/ phase exit criteria
- ✦Output: Physical architecture resulting from a trade off between alternatives
  - Generic Physical Architecture (often, the dominant design or reference architecture), and the Instantiated Physical Architecture
  - Framework for design definition documentation (specs and baselines)
  - Basis for the Work Breakdown Structure

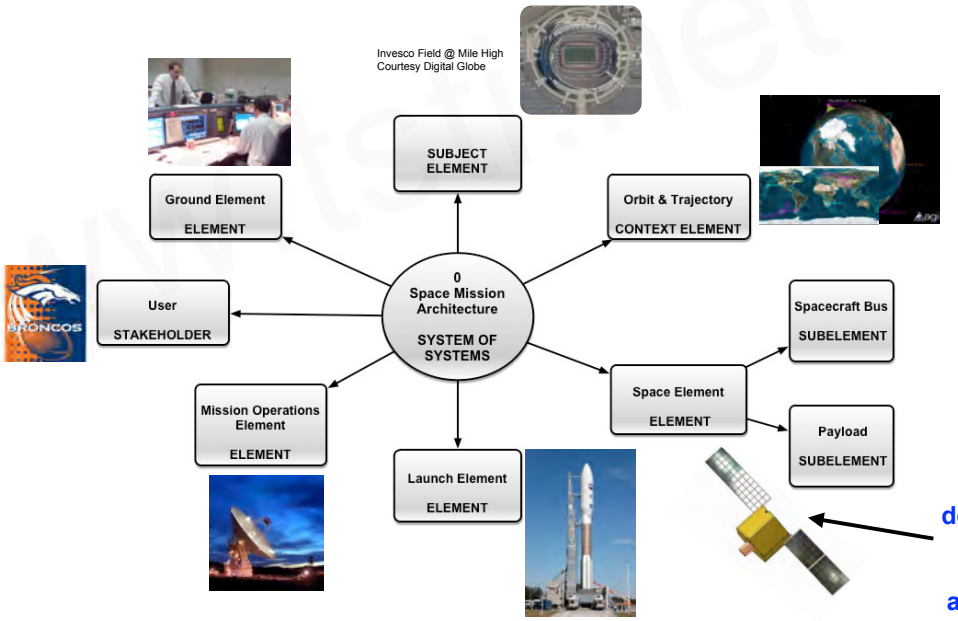


Copyright ©2020 Teaching Science and Technology Inc.

Copyright ©2020 Teaching Science and Technology Inc.

Physical Architecture—Mission-Level

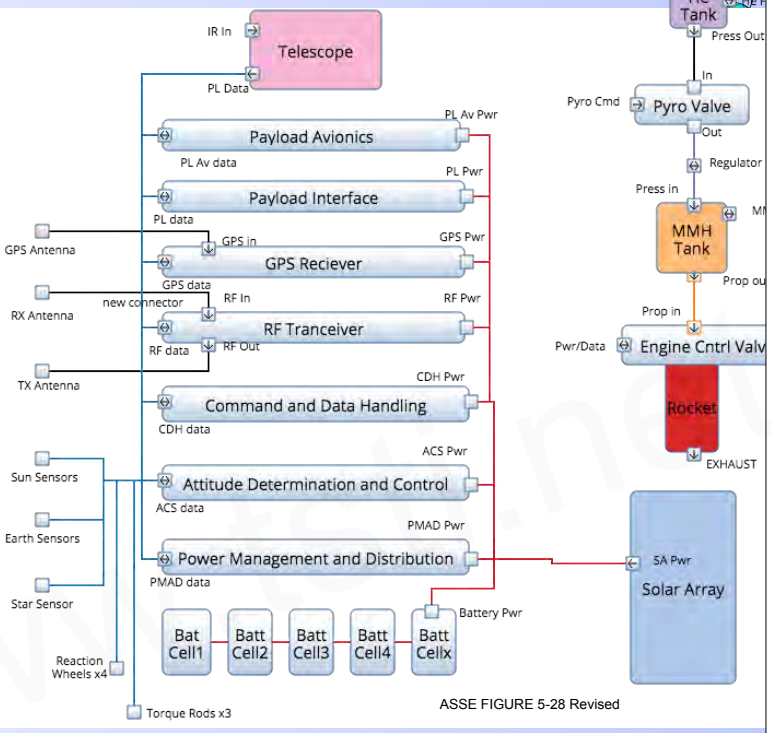
The Space Mission Physical Architecture reflects all of the trade-offs and decisions made during the design process at the Mission Systems Level



Copyright ©2020 Teaching Science and Technology Inc.

Develop a Generic System Physical Architecture

- At this point, we have a functional architecture that describes what the system of interest will do
  - We've also created a generic physical architecture that describes the types of components we think we need and the expected interfaces between them
  - To develop this architecture we need to know what physical options exits to perform specific functions. For this we're aided by a *Morphological Box*



ASSE FIGURE 5-28 Revised

Copyright ©2020 Teaching Science and Technology Inc.

Copyright ©2020 Teaching Science and Technology Inc.

Morphological Box

- One effective technique for allocating functions to components uses a *morphological box* to determine how functions or generic components may apply and instantiate the best physical components to match these applications.
  - Here the functionalities are the column headers, and the rows indicate possible physical solutions for each function.
  - We then select the best way to implement the function based on technical, cost, and scheduling requirements for the system of interest.

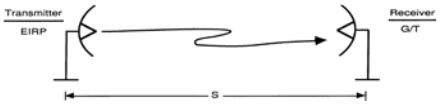
Communication Uplink	Communication Downlink	Attitude Sensors	Attitude Actuators	Power Generation	Power Storage	Navigation Method	Propulsion Technology
None (total autonomy)	UHF	Sun sensor	None (free tumble)	None (primary battery only)	Primary battery	Ground-based	Cold-gas
UHF	S-band	Star sensor	Passive spinning	Photovoltaic	NiCAD	Space-based GPS	Mono-propellant
S-band	Ka-band	Magnetometer	Dual spin	Fuel cell	Ni-H	Space-based optical	Bi-propellant
Ka-band	K-band	Accelerometer	Gravity gradient boom	RTG	NiM hydride		Solid
K-band	X-band	Gyro	Permanent magnet	Nuclear reactor	Li-Ion		Resistojet
X-band			Torque rods	Solar dynamic			Arcjet
			Reaction wheels				Ion
			Control moment gyros				Hall effect thruster
							Pulse plasma thruster
							Nuclear thermal

Adapted from ASSE TABLE 5-4

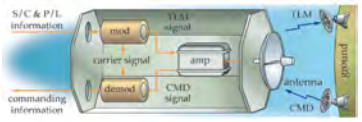
Copyright ©2020 Teaching Science and Technology Inc.

System Design Analysis and Trade Studies

- Allocating functions or generic components to the physical components creates fertile ground for trade studies, technical decision making, and iterative updates to the functions or generic components
- We apply
  - Laws of physics,
  - Engineering design principles,
  - Rules of thumb, and
  - A realistic appreciation for technology readiness
- to develop system performance models and simulations to determine how well a specific set of physical components will meet cost, schedule, performance requirements with acceptable risk
- Other types of analysis include:
  - Dynamic analysis—Assesses the system behavior's dynamic consistency and ability to execute
  - Timeline analysis—Establishes and analyzes integrated behavior timelines
  - Resource analysis—Monitors the amounts and dynamics of system resources, such as people, computer million instructions per second, memory, supplies, power, radar pulses, and number of interceptors
  - Flow analysis—Investigates how the system operation is affected based on the capacity of the link and the size of the items carried across this link

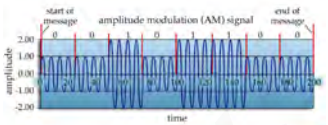


$$\frac{E_b}{N_o} = \frac{P_t L_t G_t L_s L_a G_r}{kRT_s} = \frac{[EIRP]L_s L_a}{kR}\left(\frac{G_r}{T_s}\right)$$



US Fig. 13-9

US Fig. 13-4



Design Input (units)	Analytical Result (units)	Reference
Frequency, f (GHz)	TX Output Power, P (dB)	10 Log P
Wavelength, λ (m)	TX Line Losses, L_t (dB)	Typical Value ~1.0
Modulation Technique	EIRP (dB)	$P_t+L_t+G_t$
Required Data Rate, R (bps)	Required Eb/No (dB) (1)	SMAD, Fig. 13-9
Required Bit Error Rate, BER	Space Losses, L_s (dB)	SMAD, Eq 13.22
TX Output Power, P (W)	Propagation & Polarization Losses, L_a (dB)	SMAD Fig. 13-10
TX Antenna Gain, G_t (dB)	RX Antenna Gain, G_r (dB) (at a given efficiency ~0.55 parabolic)	SMAD, Eq 13.7
Propagation Path (to a given elevation angle), S (km)	RX Antenna Beamwidth, ι_r (deg)	SMAD, Eq 13.19
RX Antenna Diameter, D_r (m)	RX Antenna Pointing Losses, L_p (dB)	SMAD, Eq 13.21
RX Antenna Pointing Error, ε_r (deg)	System Noise Temperature, T (K)	SMAD Table 13-10
	RX Eb/No (dB) (2)	SMAD, Eq 13.13
	Implementation Losses, L_i (dB) (3)	Typical Value ~2.0
	Margin (dB)	(2) - (1) + (3)

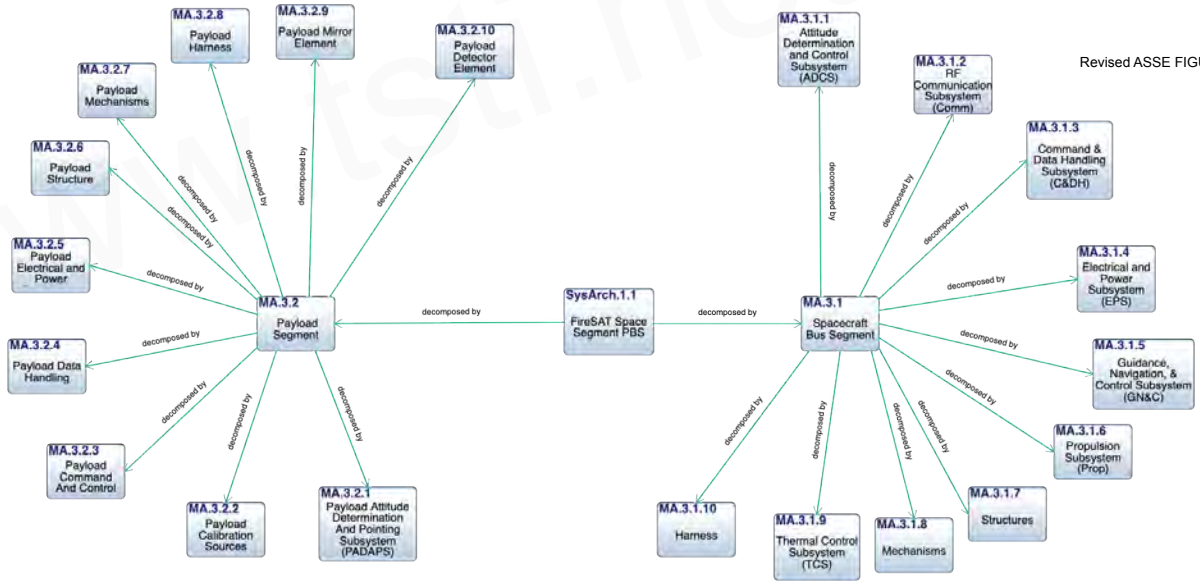
L-3 Comm Miniature S-Band T&C Transceiver MSX-765

Copyright ©2020 Teaching Science and Technology Inc.

Copyright ©2020 Teaching Science and Technology Inc.

Result: FireSAT Physical Architecture

- This figure shows the partitioning of the spacecraft to lower-level physical components

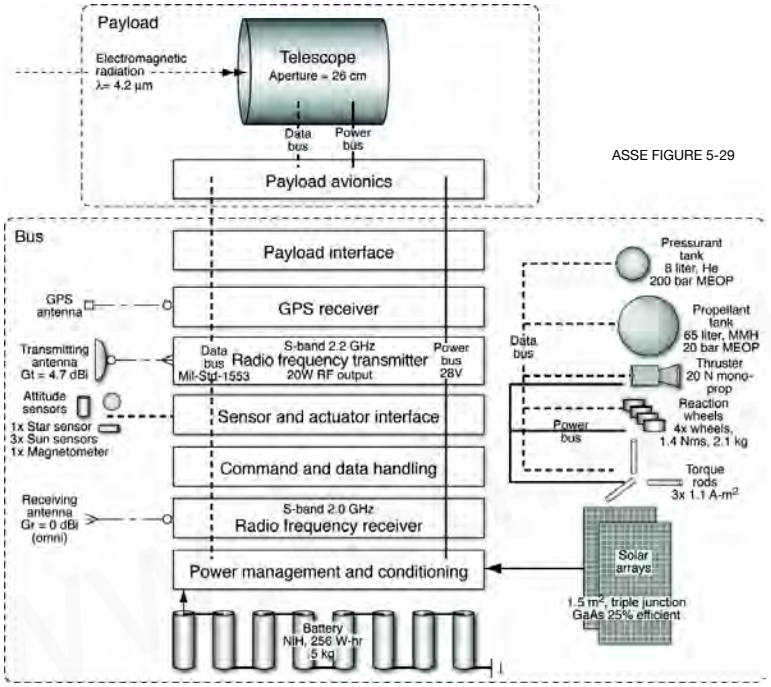


Revised ASSE FIGURE 5-33

Copyright ©2020 Teaching Science and Technology Inc.

Result: FireSAT Physical Architecture (cont'd)

- Instantiated architectures describe specific products or systems.
  - Normally they are developed from generic architectures and show the details of the physical components
- NOTE: it takes that's LOTS of analysis and numerous trade studies to get to this point!



ASSE FIGURE 5-29

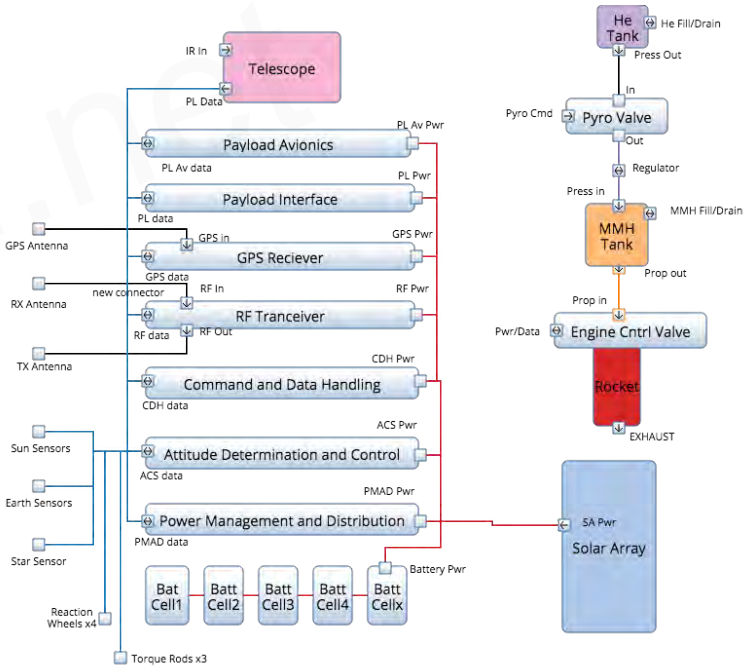
Copyright ©2020 Teaching Science and Technology Inc.

Copyright ©2020 Teaching Science and Technology Inc.

Result: FireSAT Physical Architecture (cont'd)

✦Instantiation/Specification Report

Name	Description
Attitude Determination and Control	
Battery	NiH 256 Whr, 5 kg
Command and Data Handling	
Engine Cntrl Valve	
GPS Reciever	
He Tank	8L He 200 bar MEOP
MMH Tank	65L 20 bar MEOP
Payload Avionics	
Payload Interface	
Power Management and Distribution	
Pyro Valve	
RF Tranceiver	TX: S-band 2.2 GHz, RX: S-band 2.0 GHz
Rocket	20N mono-prop thruster
Solar Array	1.5 m2 GaAs triple junction, 25% efficient
Telescope	4.2 micron, 26 cm aperture



Copyright ©2020 Teaching Science and Technology Inc.

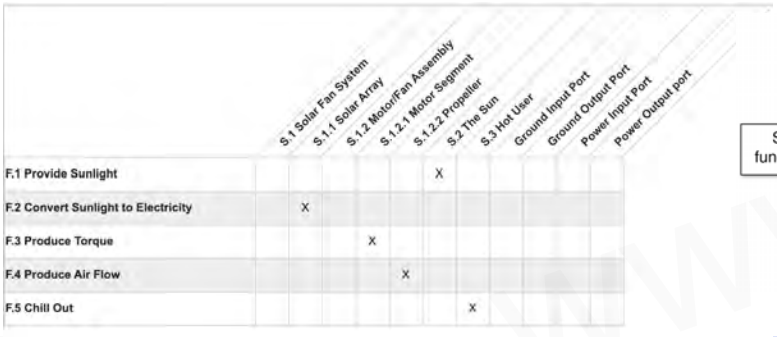
Allocating Functions to the Physical Components

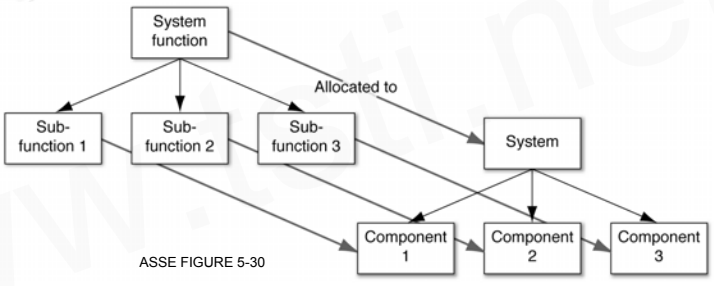
✦All functions must be allocated to some physical component, if we are to understand how each function gets done

- If a function is allocated to multiple physical components, we should decide whether to split the function into several sub-functions, so we can allocate each sub-function to a single component
- This step is important because the physical architecture drives requirements for the components and the interfaces between them.
- If we split a function across components, the requirements also split across their developers, so the interface between them may be nebulous
- An architecture is usually better when it's modular and the interfaces are well defined

✦Although this simplistic view implies each physical component should perform only one function, this is typically not the case in real life, nor should it be

- Architects must understand functions in the functional architecture and allocate them in the best way to the physical architecture.





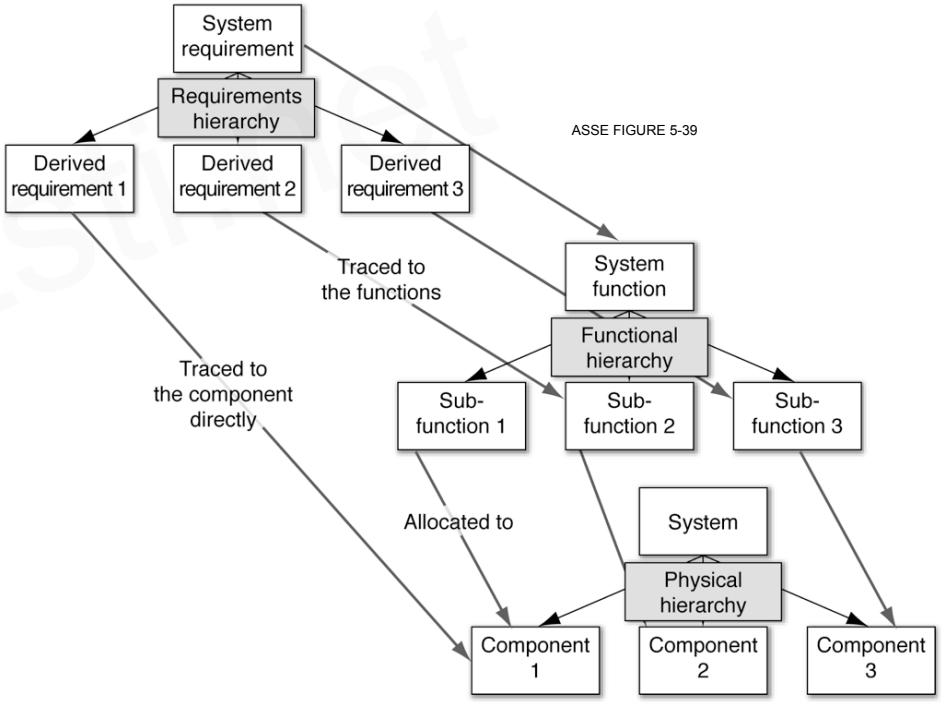
ASSE FIGURE 5-30

Copyright ©2020 Teaching Science and Technology Inc.

Copyright ©2020 Teaching Science and Technology Inc.

Allocating Requirements to Functions and Systems

- Defining Requirements for the Physical Components
  - We generally trace functional and input/output (I/O) requirements to functions.
  - We also trace nonfunctional requirements to functions or the physical components.
  - Either way, the requirements for a component include the ones traced to the component's function and the ones traced directly to the component.



Copyright ©2020 Teaching Science and Technology Inc.

Requirements Traceability

✦Tracing requirements gives us confidence that the solution meets the stakeholders' expectations.

- Traceability also begins the integration, verification, and validation planning process

✦Requirements management tools can help us trace requirements

- They may be as simple as general purpose software such as Microsoft Office or Apple iWork or dedicated tools such as DOORS™, Innoslate, Sparx EA or Cameo EA.
- We use them to trace requirements through the system lifecycle and to document that the customer accepts these requirements

✦Developing a Requirements and Verification Traceability Matrix

1. Translate stakeholder requirements into system requirements
2. Validate and trace system requirements back to the stakeholder requirements
3. Determine acceptance (or pass/fail) criteria that the project team will use as a system measurement during verification
4. Distribute and divide system requirements into component requirements down to the configuration item level while creating the functional and physical architectures
5. Verify and trace component requirements back to system–level requirements. During this time, determine acceptance criteria for the component specifications that the project team will use as a system measurement during verification

Copyright ©2020 Teaching Science and Technology Inc.

Copyright ©2020 Teaching Science and Technology Inc.

Assess the Architecture

- Architectural assessment metrics typically consider selected and specific aspects of the system solution, such as performance or flexibility and scalability or testability and maintenance, and so on.
- The metrics may come from the system or stakeholder requirements or from characteristics that all (or most) systems meet

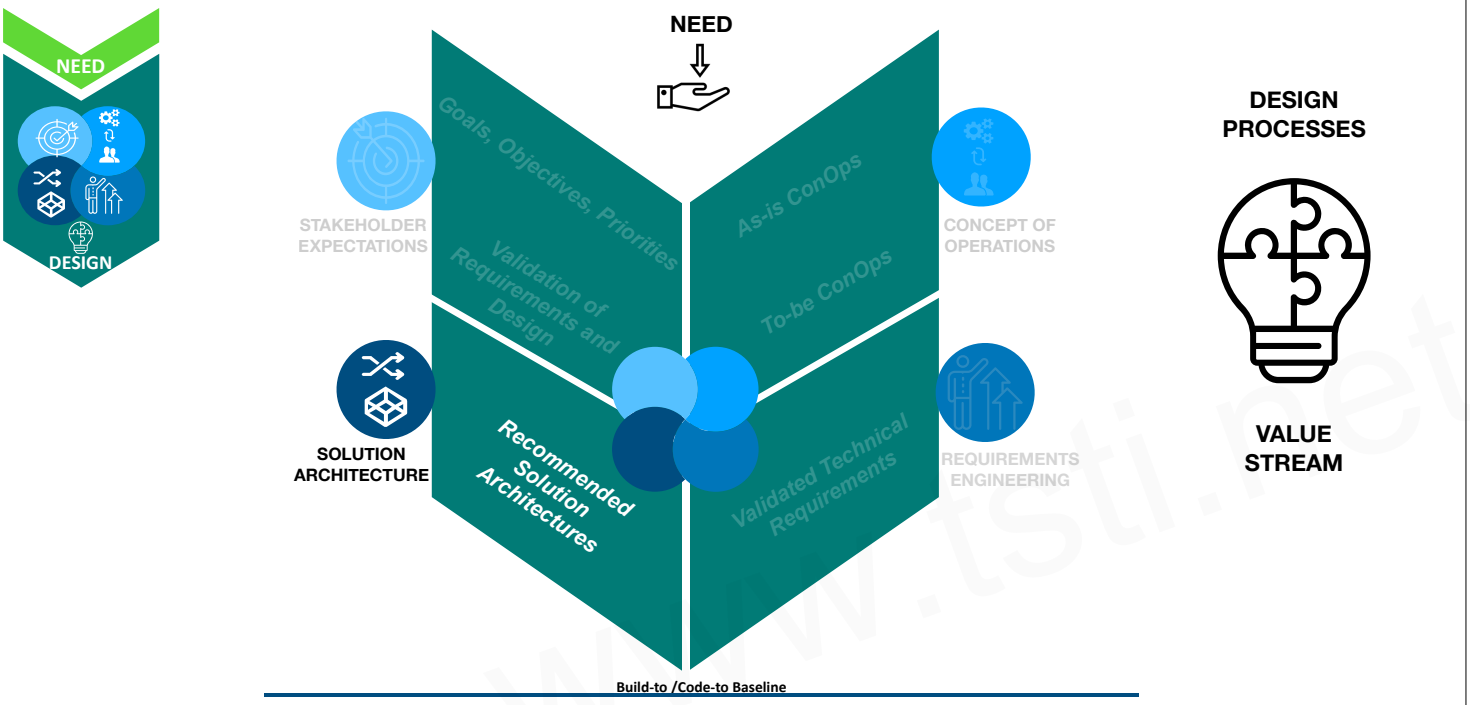
Commonality	Modularity	Standards Based	RMT

• Physical commonality (within the system)
  • Hardware commonality
  • Number of unique line replaceable units (LRUs)
  • Number of unique fasteners
  • Number of unique cables
  • Number of unique standards implemented
  • Software commonality
  • Number of unique software packages implemented
  • Number of languages
  • Number of compilers
  • Average number of software instantiations
  • Number of unique standards implemented
• Physical familiarity (from other systems)
  • % vendors known
  • % subcontractors known
  • % hardware technology known
  • % software technology known
• Operational commonality
  • % of operational functions automated
  • Number of unique skill codes required
  • Estimated operational training time—initial
  • Estimated operational training time—refresh from previous system
  • Estimated maintenance training time—initial
  • Estimated maintenance training time—refresh from previous system
	• Physical modularity
	  • Ease of system element upgrade
	  • Lines of modified code
	  • Number of labor hours for system rework
	  • Ease of operating system upgrade
	  • Lines of modified code
	  • Number of labor hours for system rework
	• Functional modularity
	  • Ease of adding new functionality
	  • Lines of modified code
	  • Number of labor hours for system rework
	  • Ease of upgrade existing functionality
	  • Lines of modified code
	  • Number of labor hours for system rework
	• Orthogonality
	  • Are functional requirements fragmented across multiple processing elements and interfaces?
	  • Are there throughput requirements across interfaces?
	  • Are common specifications identified?
	• Abstraction
	  • Does the system architecture provide options for information hiding?
	• Interfaces
	  • # of unique interfaces per system element
	  • # of different networking protocols
	  • Explicit versus implicit interfaces
	  • Does the architecture involve implicit interfaces?
	  • # of cables in the system
		• Open systems orientation
		  • Interface standards
		  • # of interface standards/# of interfaces
		  • Multiple vendors (more than 5) exist for products based on standards
		  • Multiple business domains apply/use standard (aerospace, medical, telecommunications)
		  • Standard maturity
		  • Hardware standards
		  • # of form factors/# of LRUs
		  • Multiple vendors (more than 5) exist for products based on standards
		  • Multiple business domains apply/Use standard (aerospace, medical, telecommunications)
		  • Standard maturity
		  • Software standards
		  • # of proprietary and unique operating systems
		  • # of non-standard databases
		  • # of proprietary middleware
		  • # of non-standard languages
		• Consistency orientation
		  • Common guidelines for implementing diagnostics and performance monitoring and fault localization
		  • Common guidelines for implementing OMI
			• Reliability
			  • Fault tolerance
			  • % of mission critical functions with single points of failure
			  • % of safety critical functions with single points of failure
			  • Critical points of delicateness (system loading)
			  • % processor loading
			  • % memory loading
			  • How critical is this?
			  • % network loading
			  • How critical is this?
			• Maintainability
			  • Expected mean time to repair
			  • Maximum fault group size
			  • Is system operational under maintenance?
			  • Accessibility
			  • Are there space restrictions?
			  • Are there special tool requirements?
			  • Are there special skills requirements?
			• Testability
			  • # of LRUs covered by BIT (BIT coverage)
			  • Reproducibility of errors
			  • Logging or recording capability
			  • Create system state at time of system failure?
			  • Online testing
			  • Is system operational during external testing?
			  • Ease of access to external test points?
			  • Automated input or stimulation insertion

ASSE FIGURE 5-37

Copyright ©2020 Teaching Science and Technology Inc.

We were Here



Copyright ©2020 Teaching Science and Technology Inc.

Copyright ©2020 Teaching Science and Technology Inc.

Solution Architecture—Objectives and Agenda

✦Learning Objectives

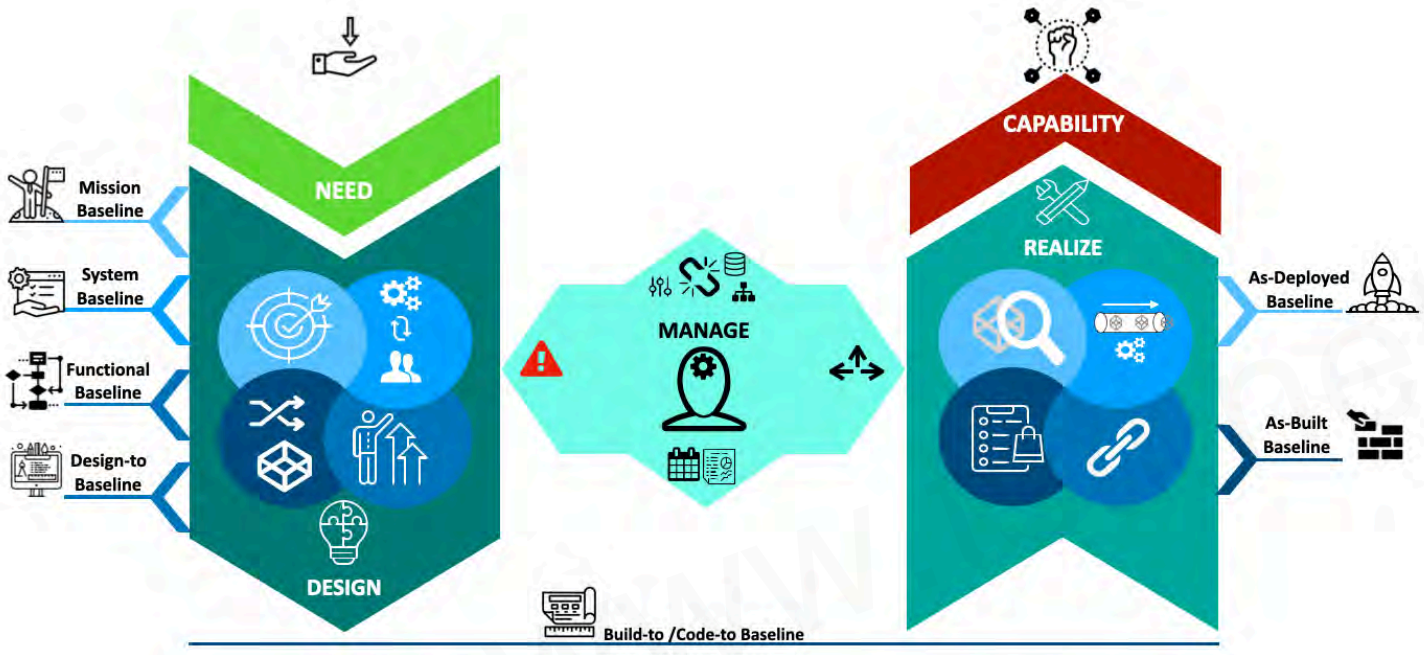
- At the end of this section you should be able to…
  - Describe the elements of functional and physical architectures and how they relate to each other (e.g. functions *performed by* physical)
  - Explain various approaches to functional architecture development
  - Apply systems engineering tools and techniques to capture and communicate functional and physical architectures

✦Agenda

- Creating Solution Architectures
- Defining Functional Partitions
  - Static Approaches
  - Dynamic Approaches
- Design Solution—Physical Architecture
- Allocation
  - Functional to Physical Allocation
  - Requirements Allocation
- Assessing Architectures

Copyright ©2020 Teaching Science and Technology Inc.

Systems Engineering—Review



Copyright ©2020 Teaching Science and Technology Inc.

Copyright ©2020 Teaching Science and Technology Inc.

Course Objectives and Agenda

✦Learning Objectives

- At the end of this course you should be able to…
  - Define key systems engineering terms
  - Explain fundamental systems engineering principles
  - Apply systems engineering tools and techniques to solve specific design, manage and realization challenges
  - *Develop relevant systems engineering artifacts for a given scenario that captures and communicates design, systems management and system realization decisions*

✦Agenda

- **Introduction to Systems Engineering**
- **DESIGN**
  - Bounding Scope
  - Engineering Requirements
  - **Developing the Solution Architecture**
- MANAGE
  - Managing Interfaces, Configuration and Data
  - Making Decisions
  - Managing Risk
  - Technical Planning
- REALIZE
  - Implementing and Integrating
  - Verification and Validation
  - Transition and DevOps
- Capstone Exercise
- FireSAT Case Study (Reference)

Copyright ©2020 Teaching Science and Technology Inc.

Thank You for Your Time and Attention!

Questions?

TSTI

Learn more at: www.tsti.net

Copyright ©2020 Teaching Science and Technology Inc.

Copyright ©2020 Teaching Science and Technology Inc.

www.tsti.net

Copyright ©2020 Teaching Science and Technology Inc.

www.tsti.net

Copyright ©2020 Teaching Science and Technology Inc.

Copyright ©2020 Teaching Science and Technology Inc.