

# SYS 581

# Introduction to Systems Engineering

*Instructor:*  
*Onur Asan*

*TA: Bijun Wang*



# Course Objectives

This course enhances the ability of design engineers, program and project managers, and others who work with complex systems to:

- Understand the context in which systems operate, the customers they serve and the stakeholder needs they must satisfy
- Work across the boundaries between traditional engineering disciplines to ensure that a system as a whole fulfills its purpose
- Appreciate the importance of interfaces and understand how to work across them effectively
- Collaborate with others of different disciplines and backgrounds as members of a diverse team

## Learning activities

- Engaged participation in class sessions for discussion, activities, and guest lectures;
- Self-directed required assignments and reading;
- Personal reflection/design project
- Group work for the project during the class
- Didactic lectures/Video sessions



# Philosophy

- Commitment to inclusion and accessibility, please discuss any special needs you may have with me.
- Collaboration.
- Experiential learning.
- Project versus additional concepts



# A little information about your instructor



# Background

- **PhD and MS: Industrial and Systems Engineering**
  - University of Wisconsin-Madison



## What I am doing now

- Professor at Medical College of Wisconsin and adjunct at University of Wisconsin-Milwaukee

Design and evaluation of clinical and consumer health technologies

Applying human factors tools into health care problems.



### Impact of clinical health information technologies

- Communication
- Human and system centered outcomes
- Error and safety

### Design of consumer health information technologies

- Designing for diverse users
- Designing for groups (families, patients in teams)

### Patient and provider safety

- Technology implementation
- Organizational effects of implementation

Technology mediated collaboration in health

usability, user centered

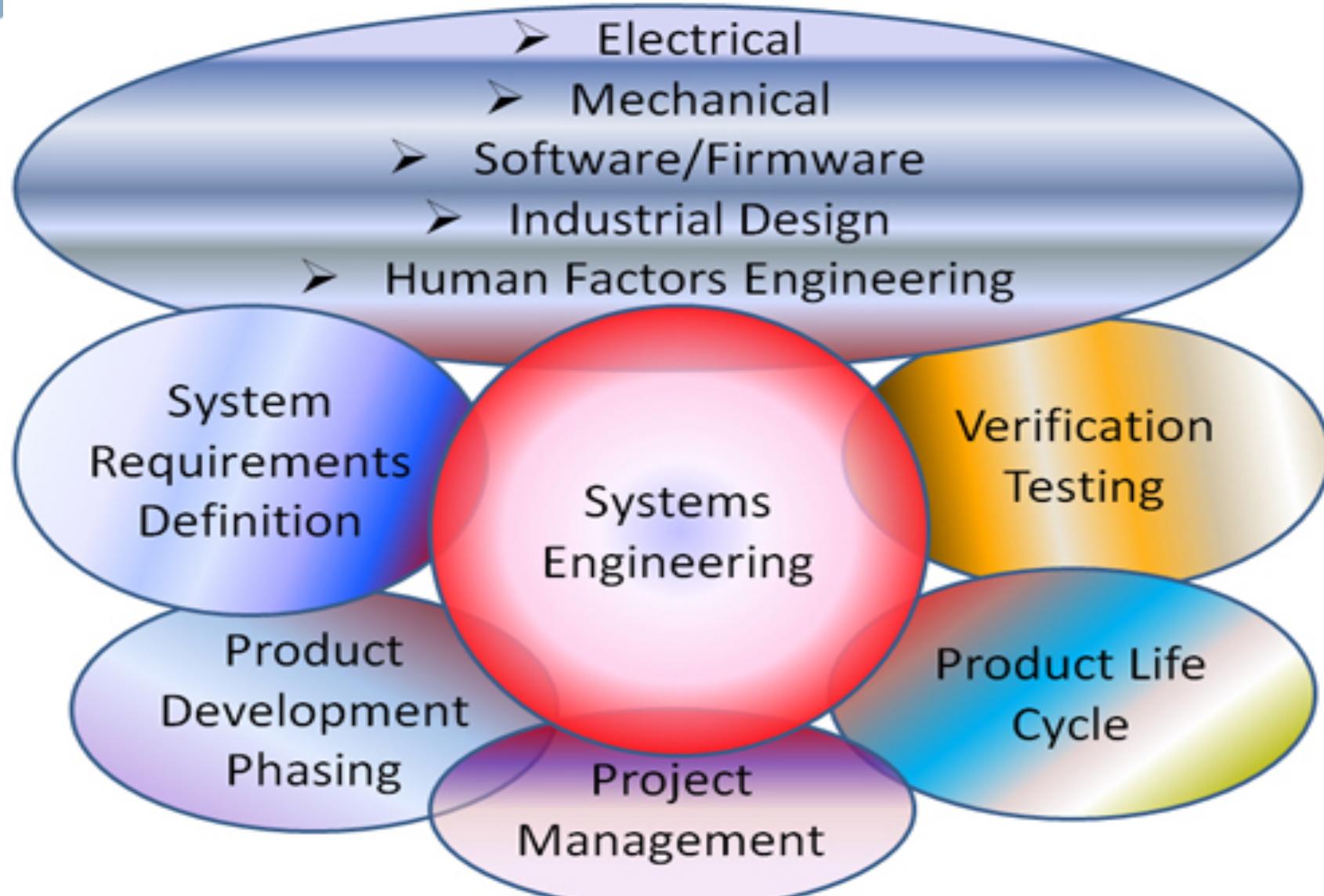


# Getting to know you!(Please show yourself)

- Name
- Any prior experience with Systems Engineering
- Name one thing you want to learn in this class
- Brief Response: Name one system
- 1 min

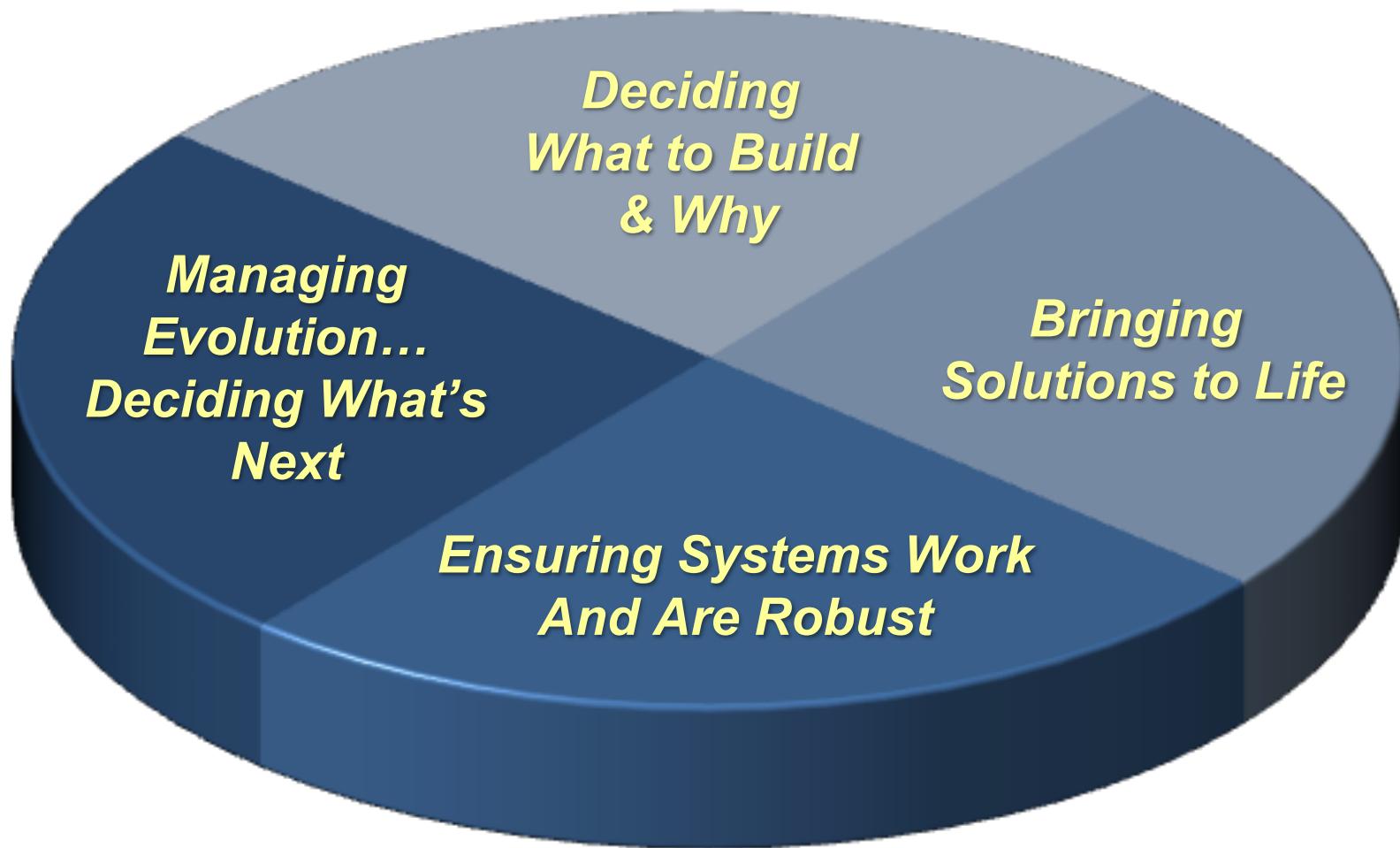


- Is there any System Engineer here in the class?



# SYS 581

## Introduction to Systems Engineering





# Course Design

*Deciding  
What to Build  
& Why*

Defining the  
Problem

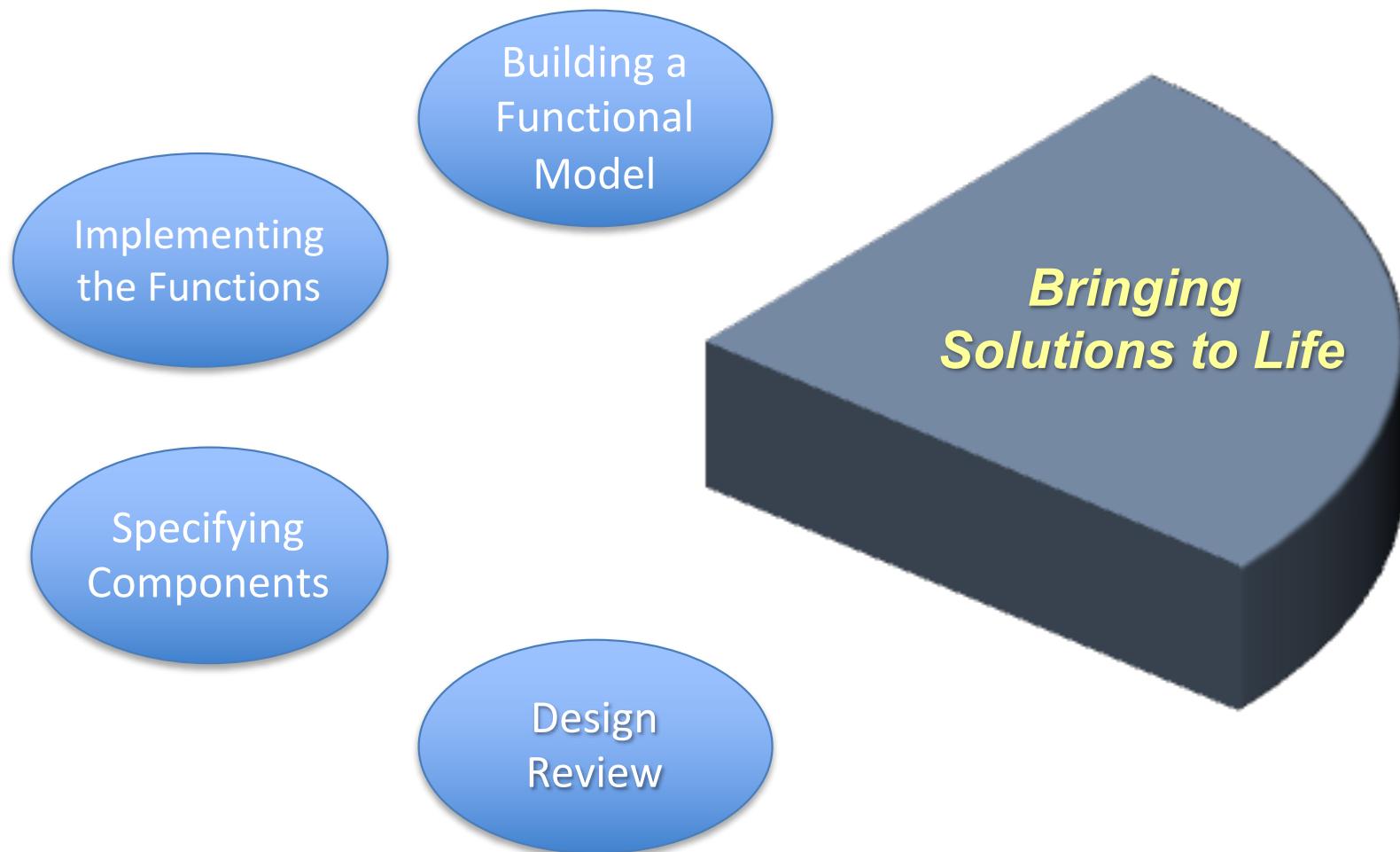
Concept  
Review

Developing  
a Solution

Formulating  
a Proposal

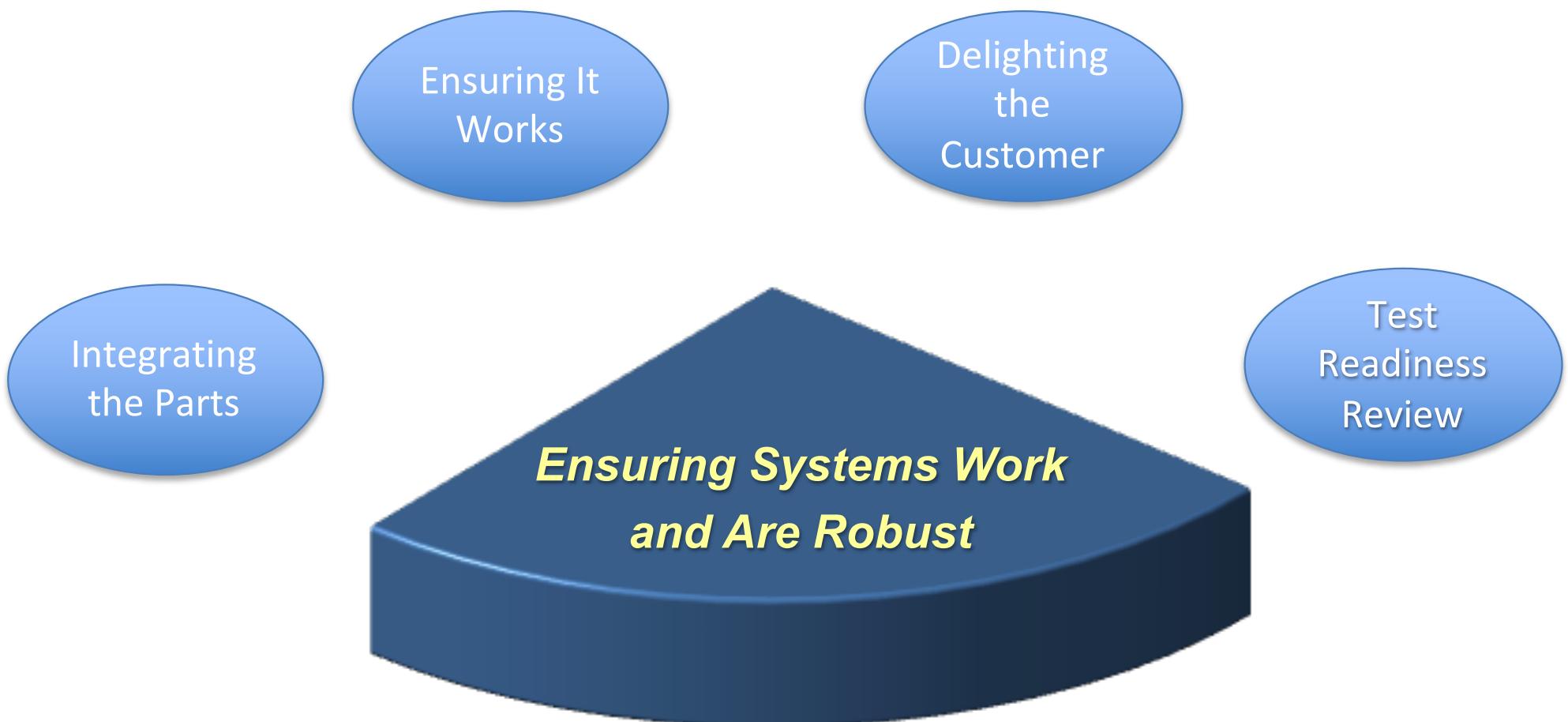


# Course Design





# Course Design





# Course Design



# Grading

- Homework (In class+ home assignments+discussion board) 20%
- Midterm Exam 30%
- Class Project related activities 50%
- Concept Review 10%
- Design Review 10%
- Test Readiness Review 10%
- Final Project Report 20%
- Team Evaluation forms

# Intended Learning Outcomes

**After successfully completing this course, students will be able to:**

- Define an operational need and distinguish between the need and its solution
- Elicit a complete set of stakeholder requirements and translate them into specifications for the system that will be designed to meet them
- Develop a system architecture and flow system requirements down to every component that must be designed or procured
- Develop integration and test plans to ensure that the system satisfies its requirements at every level of the architecture and meets all the needs of its stakeholders
- Understand emerging trends in system complexity and evolving systems engineering approaches for dealing with them.



# The History of Systems Engineering

Systems Engineering has been informally practiced since antiquity

- Great Wall of China, Egyptian Pyramids, Roman Aqueducts
- Mainly a “workforce” problem to build large infrastructures

The term “Systems Engineering” can be traced back to Bell Labs (1940s)

- [https://en.wikipedia.org/wiki/Bell\\_Labs](https://en.wikipedia.org/wiki/Bell_Labs)
- Beginning of new methods to better handle complexity

Formal Systems Engineering really started after WWII

- 1950's and 1960s: Cold War, Apollo Lunar Program, ICBMs etc...
- Complex Engineering Systems: Air Traffic Control, High Speed Rail, Nuclear, Healthcare



# How would you define Systems Engineering?

- Turn to your neighbor and discuss for about 8 minutes (Breakout room):
  - What is your definition of Systems Engineering?
  - Can you agree amongst yourselves?
  - What are the key elements of a definition?



# Some Definition of System Engineering?



"System engineering is a robust approach to the design, creation, and operation of systems. In simple terms, the approach consists of identification and quantification of system goals, creation of alternative system design concepts, performance of design trades, selection and implementation of the best design, verification that the design is properly built and integrated, and post-implementation assessment of how well the system meets (or met) the goals."— NASA Systems Engineering Handbook, 1995.



"An interdisciplinary approach and means to enable the realization of successful systems"— INCOSE handbook, 2004



# What is a System?



# Is This a System?





# Is This a System?





# Is This a System?





# So what is a System...

*How about:*

A set of elements...  
that work together...  
to achieve common purpose.

***“A set of interrelated components working together to achieve a common purpose.”***

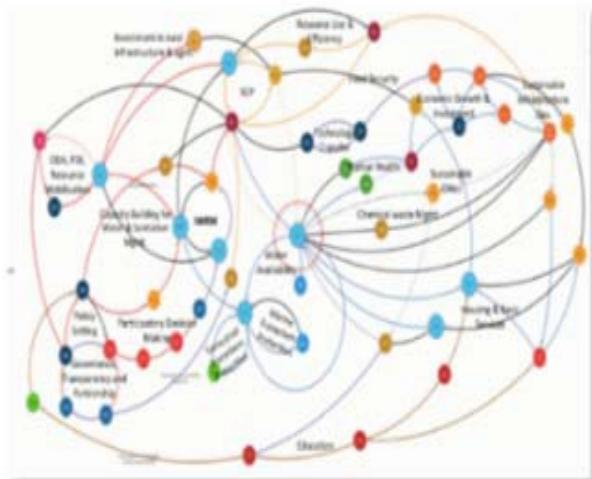
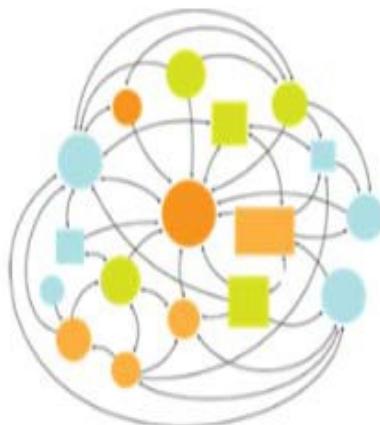




- Give me example of a “system”

# Principles of Systems Thinking

- Think outside before inside.
- Think what and why before how.
- Think relationships not just elements.
- Think long term not just initial capability.
- Think circles not lines.

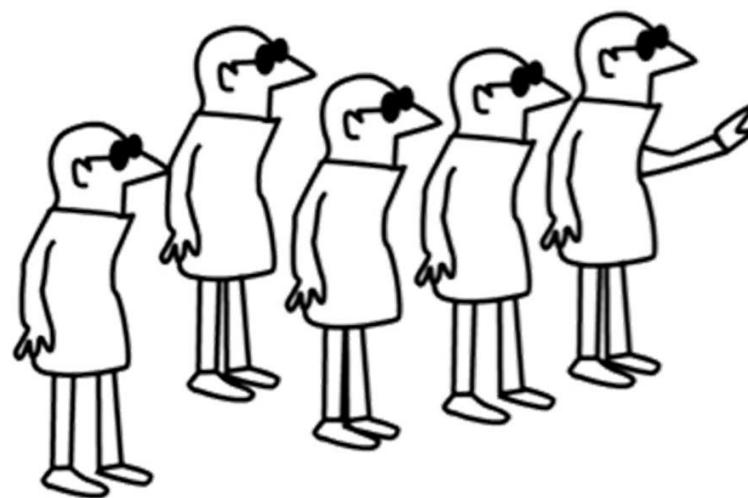


“ Any intelligent fool can make things bigger, more complex, and more violent. It takes a touch of genius — and a lot of courage to move in the opposite direction.

Ernst F. Schumacher, Development Economist & Statistician

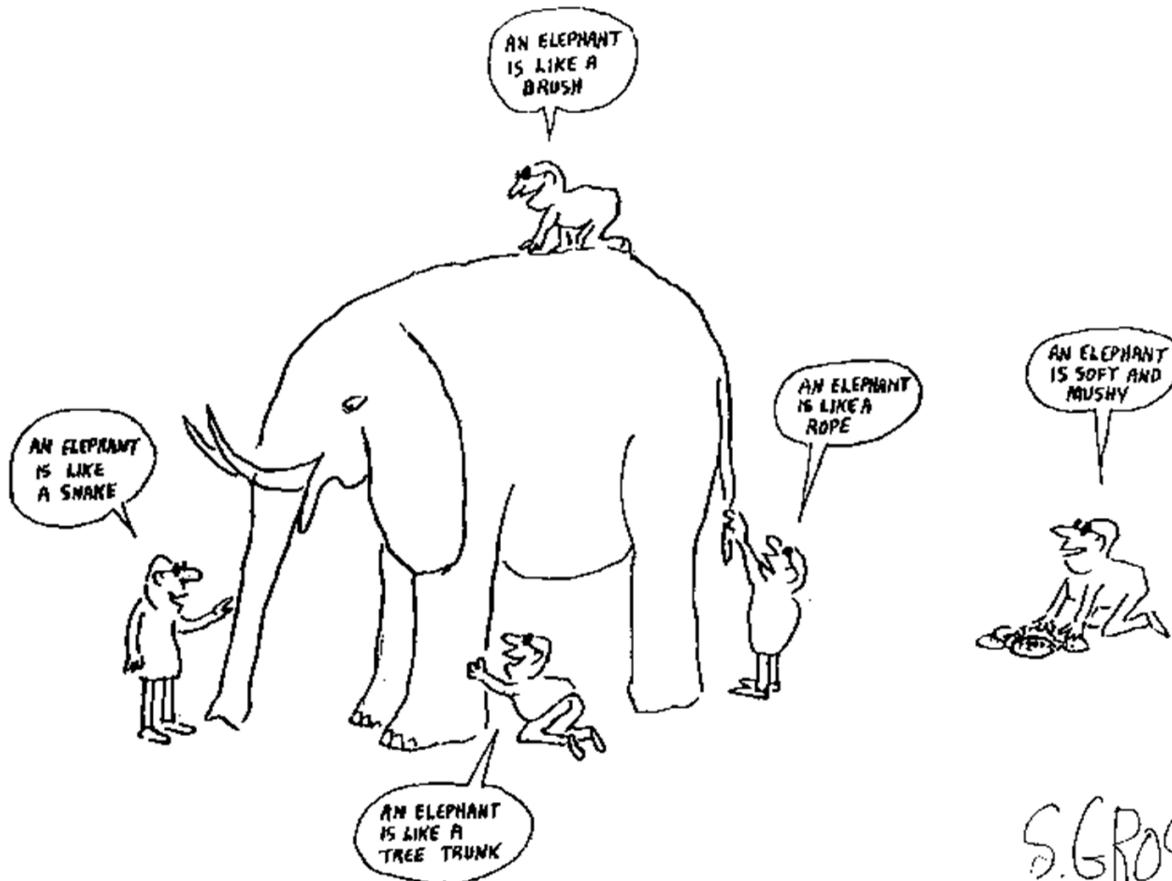


# What is a Systems Perspective?



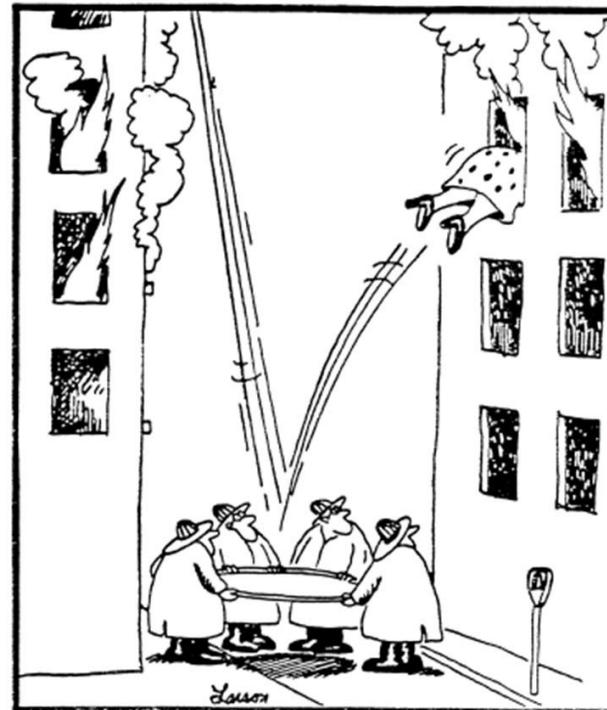


By fixating on the parts of a system,  
we miss understanding the  
whole





Lacking a Systems Perspective... the solution to one problem can easily lead to a new and bigger problem later on or somewhere else in the system



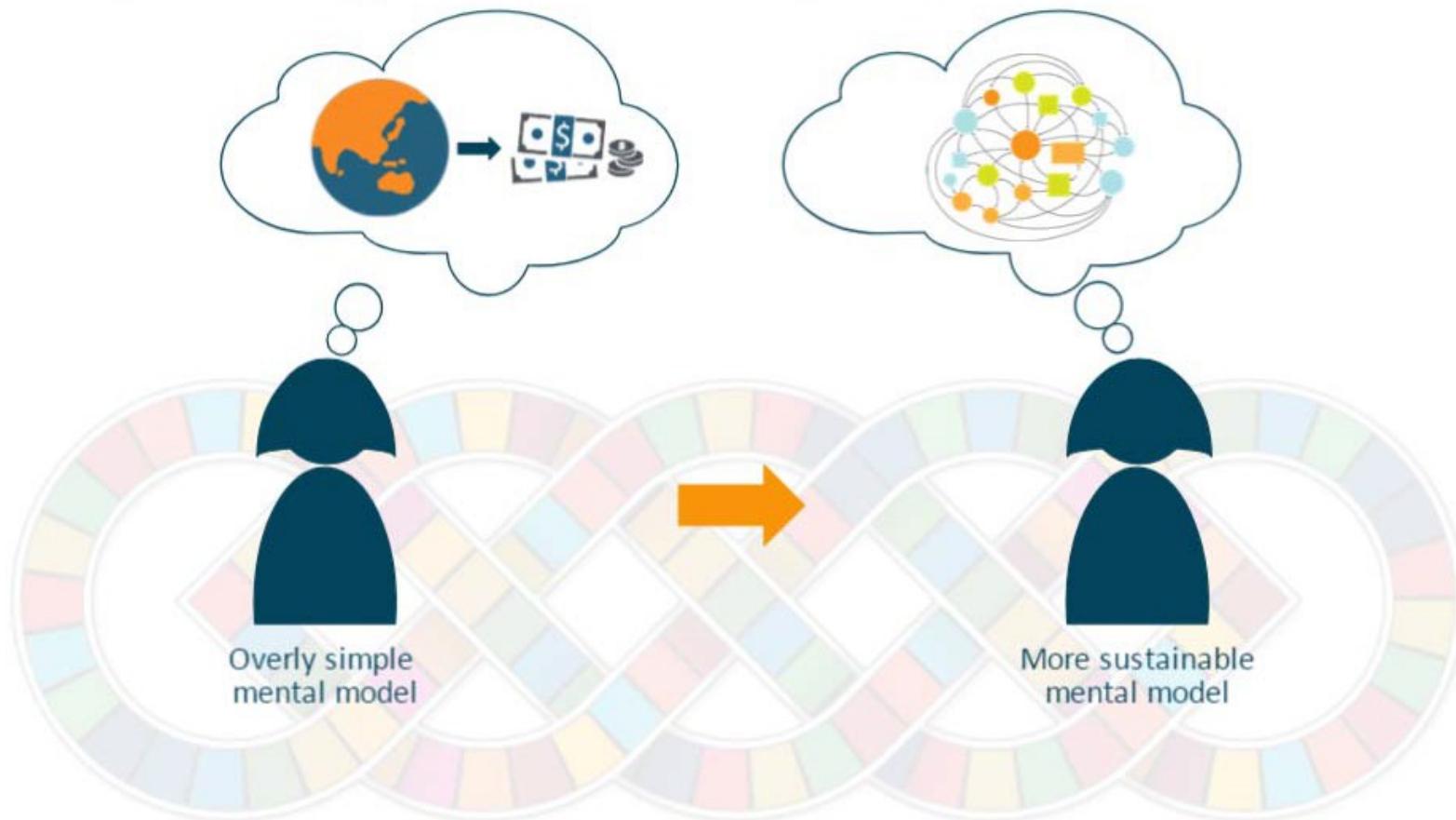
Artist:  
Gary Larson

**“unintended consequences”**



## Systems Thinking in a Nut-Shell

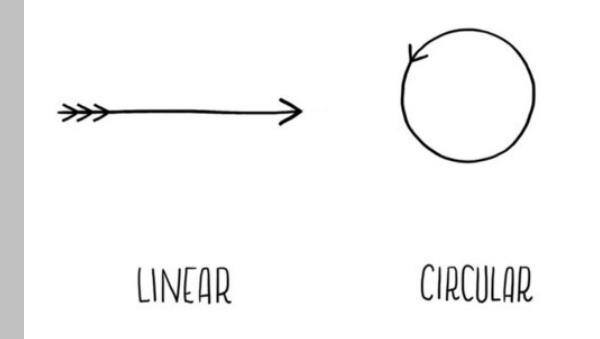
Systems thinking is the practice of examining, and improving, our mental models





# Systems Thinking

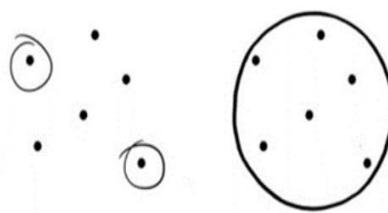
- System mindsets are needed for dealing with complex problem solving
  - 4 fundamental concepts:
    - INTERCONNECTEDNESS
    - SYNTHESIS
    - FEEDBACK LOOPS
    - CAUSALITY





## Analysis

*Is about dissection of complexity into manageable components. Analysis fits into the mechanical and reductionist worldview, where the world is broken down into parts*



ANALYSIS

SYNTHESIS

## Synthesis

**sees the interconnectedness**

### Synthesis

**Is about understanding the whole and the parts at the same time, along with the relationships and the connections that make up the dynamics of the whole.**

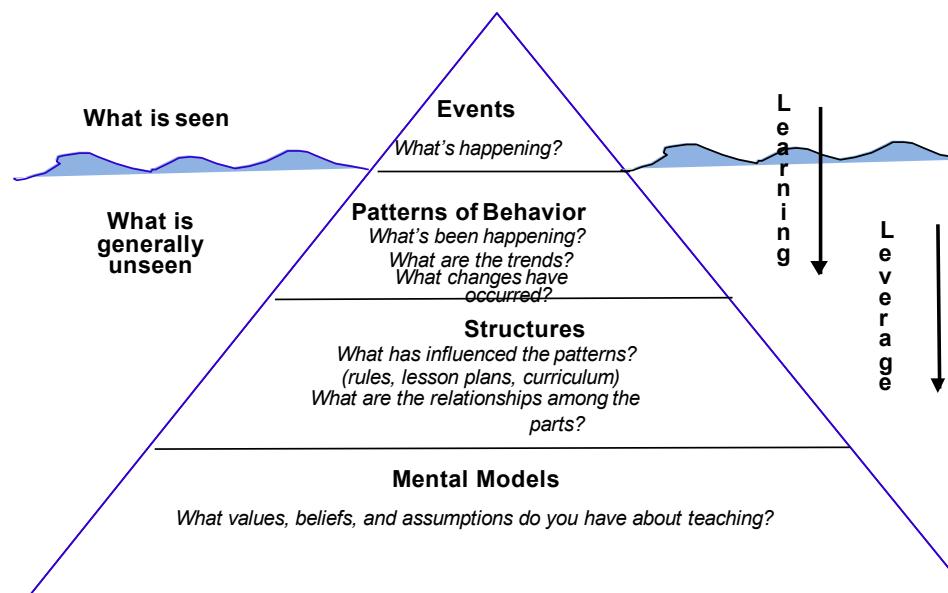


# What is Systems Thinking?

- Systems thinking is a vantage point from which you see a whole, a web of relationships, rather than focusing only on the detail of any particular piece. Events are seen in the larger context of a pattern that is unfolding over time. -  
*isee systems, inc.*
- Systems thinking is a perspective of seeing and understanding systems as wholes rather than as collections of parts. A whole is a web of interconnections that creates emerging patterns. – Peter Senge

# Systems Thinking helps us to . . .

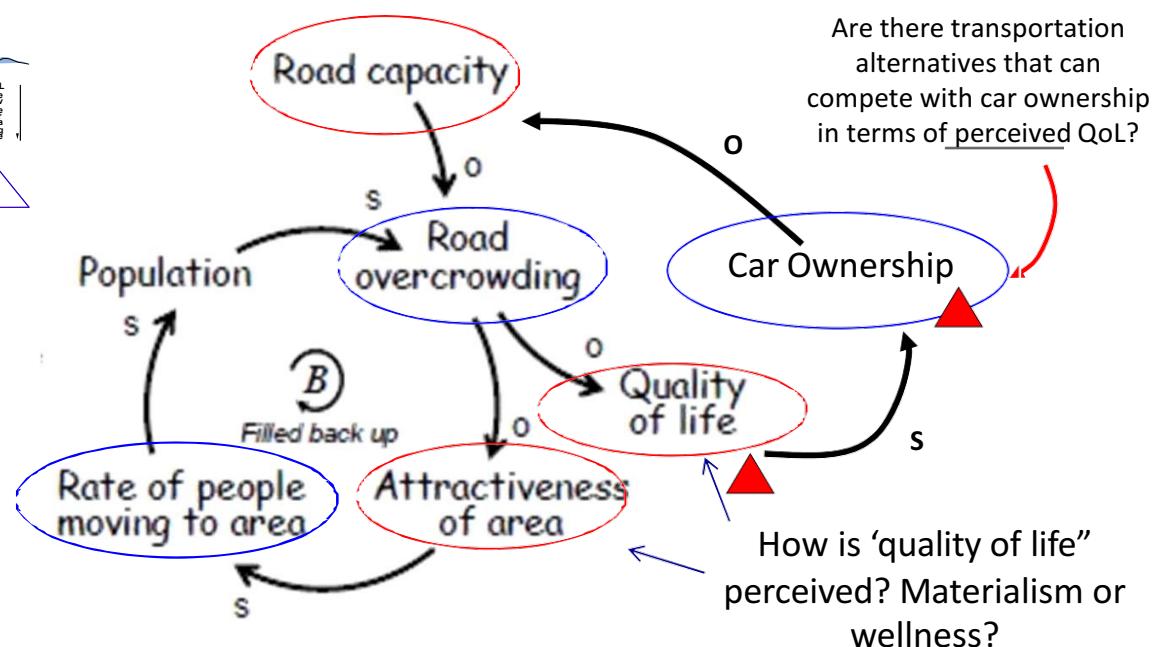
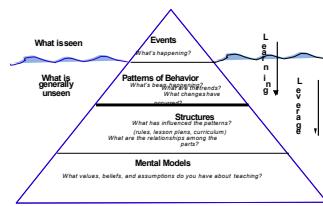
- . . . move the focus away from **events** and **patterns of behavior** (*which are symptoms of problems*) and toward **systemic structure** and the underlying **mental models**



**Source:** Senge, Peter, *The Fifth Discipline*, 1996.

# Systems Thinking helps us to . . .

- . . . To find the most important places for intervention to change the long-term behaviour of a system.



Source: Senge, Peter, The Fifth Discipline, 1996.

## **“Think outside before inside.”**

### **Analysis vs. Synthesis**

#### **Analysis**

1. Take what I want to understand apart
2. Identify the behaviors of the parts taken separately
3. Aggregate an understanding of the parts into an understanding of the whole

#### **Synthesis**

1. Identify the system in which what I want to understand is a part
2. Explain the behavior of the containing whole
3. Disaggregate the understanding of the whole to identify the role or function of the desired part

*“From a very early age, we are taught to break apart problems, to fragment the world.” ...Peter Senge, The Fifth Discipline*

**“Think what and why before how.”**

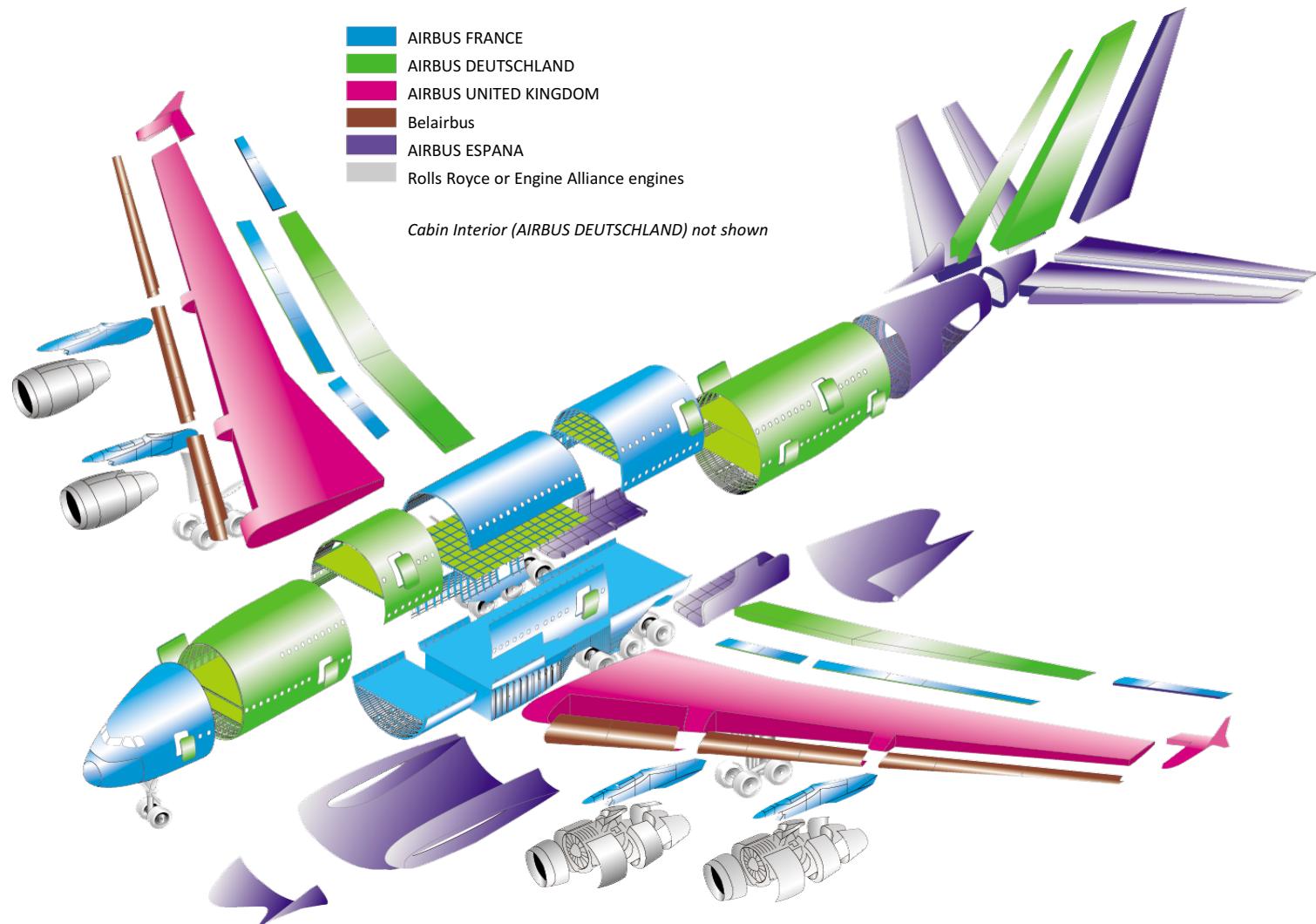
**What problem was this invention trying solve?**

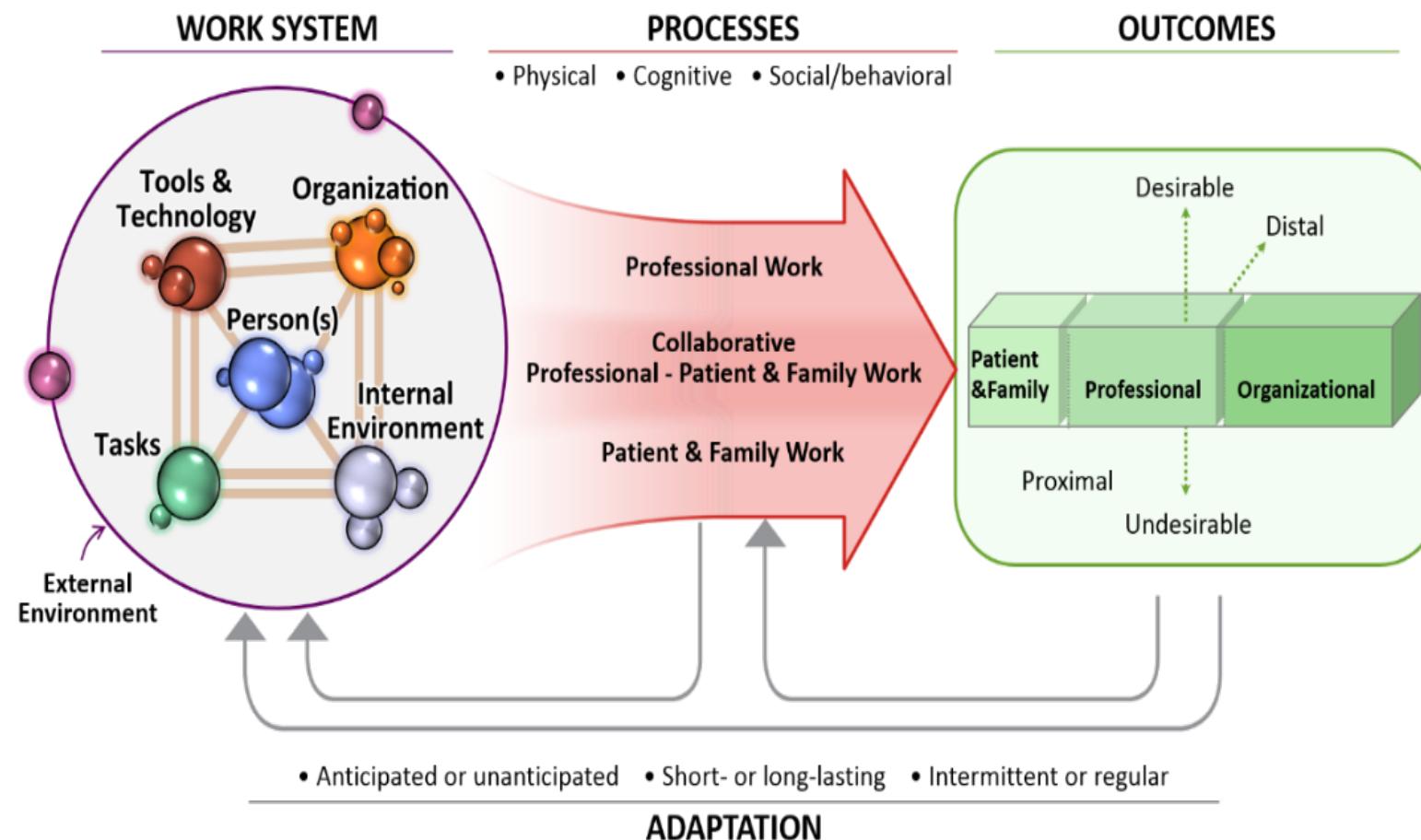
The Segway



# “Think relationships not just elements.”

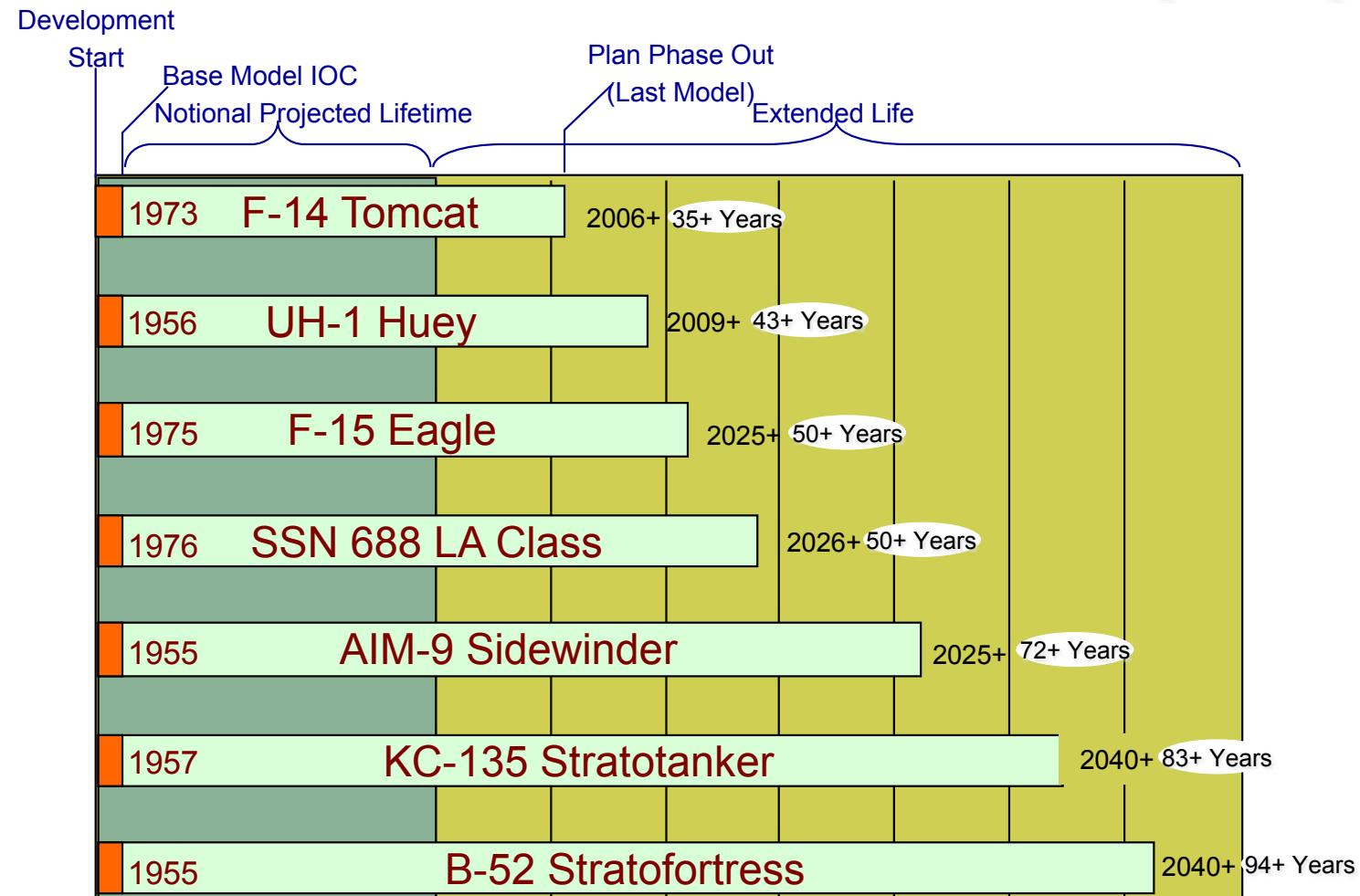
## Complex systems usually fail at interfaces.





# “Think long term not just initial capability.”

## The future often lasts a long, long time.



Similar Reality for Enterprise Level IT Systems

Many Applications Pre-Date the Internet and Client-Server Architectures

## **“Think circles not lines.”** Language shapes perceptions.

Western languages are biased toward linear cause and effect relationships.

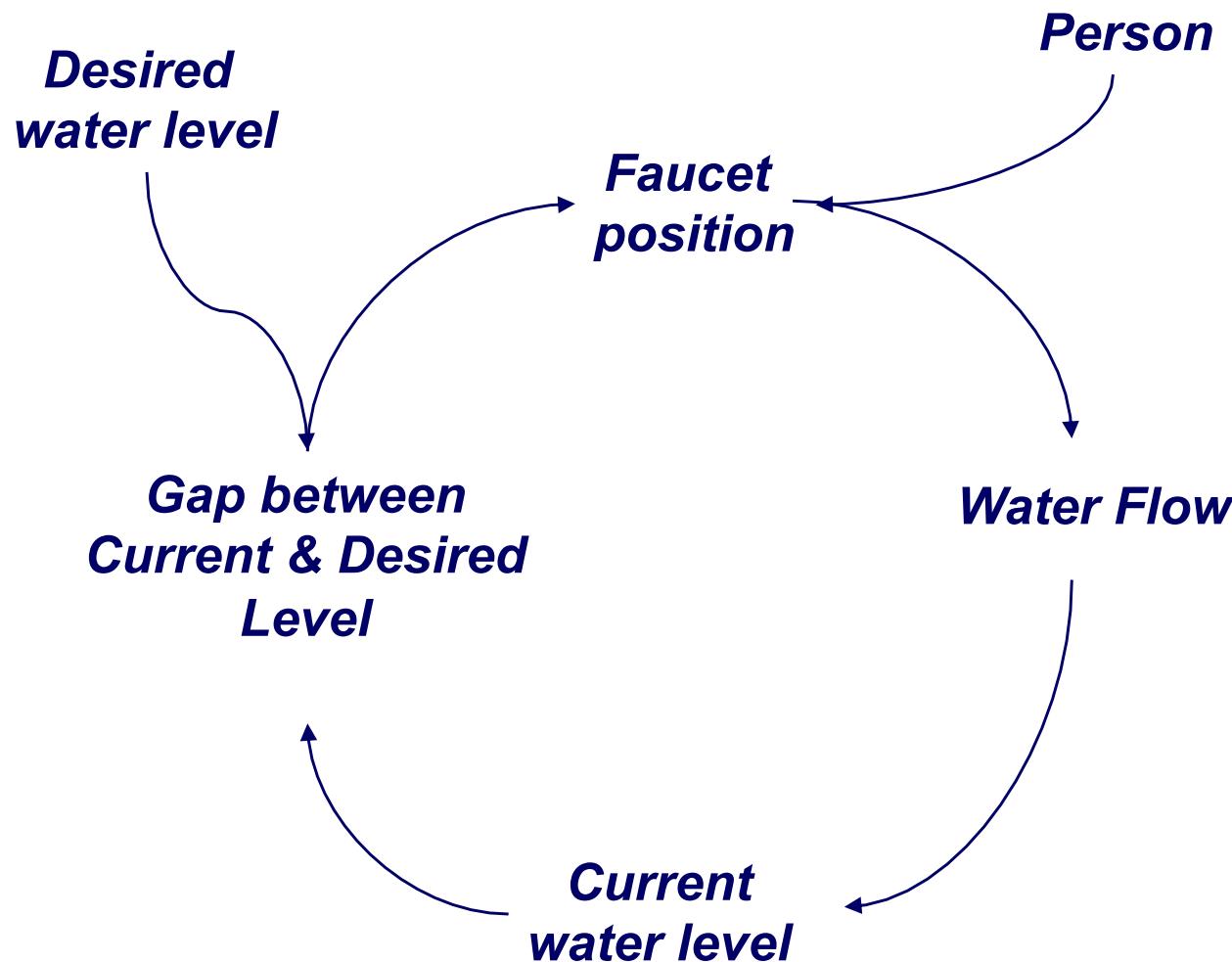
**Subject → Action → Object**

To see system-wide interrelationships, we need a language of interrelationships.

- A language made up of circles.



# Filling a glass of Water





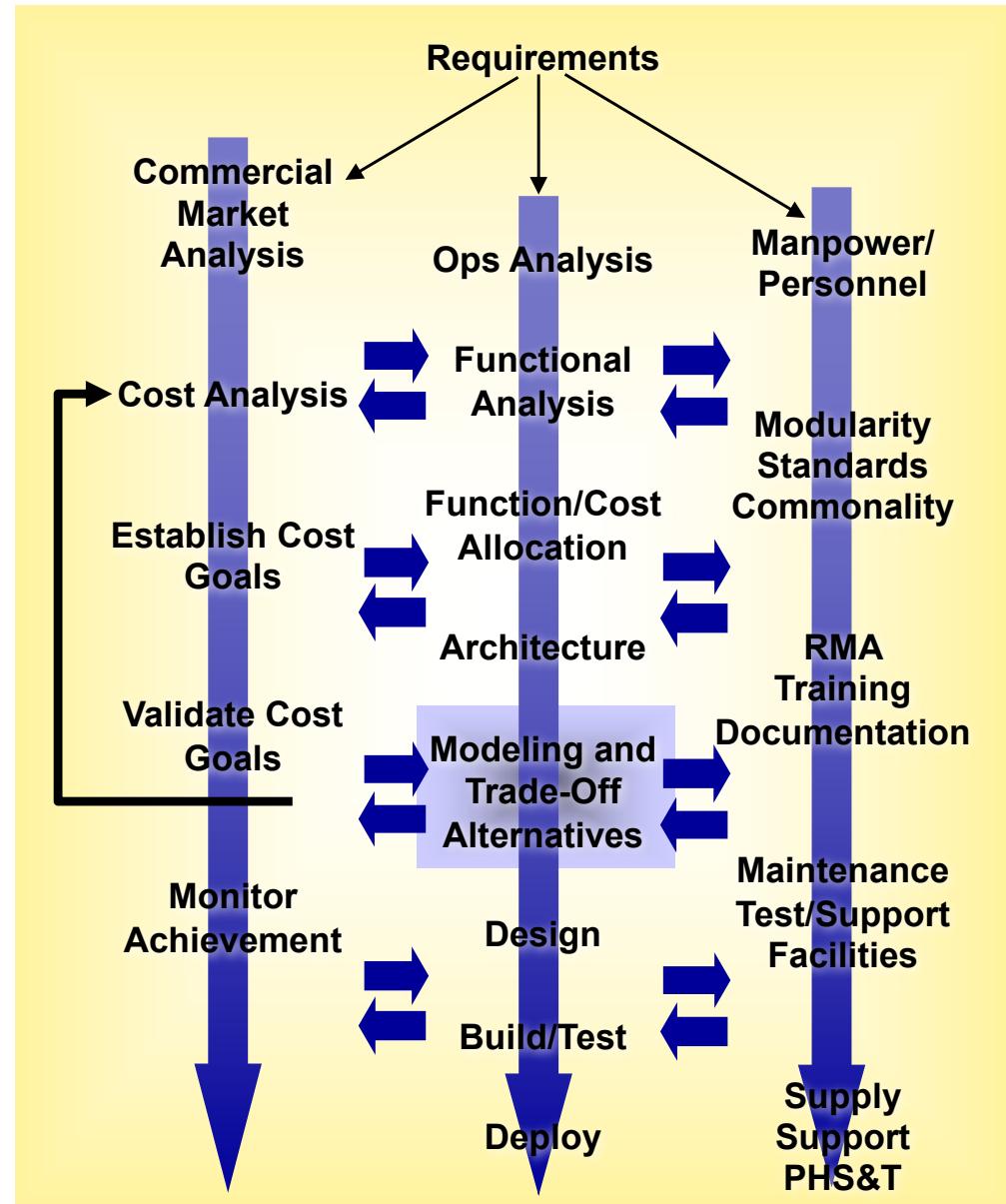
# Role of Systems Engineering

*The application of systems thinking to the design, development and sustainment of complex technical systems.*

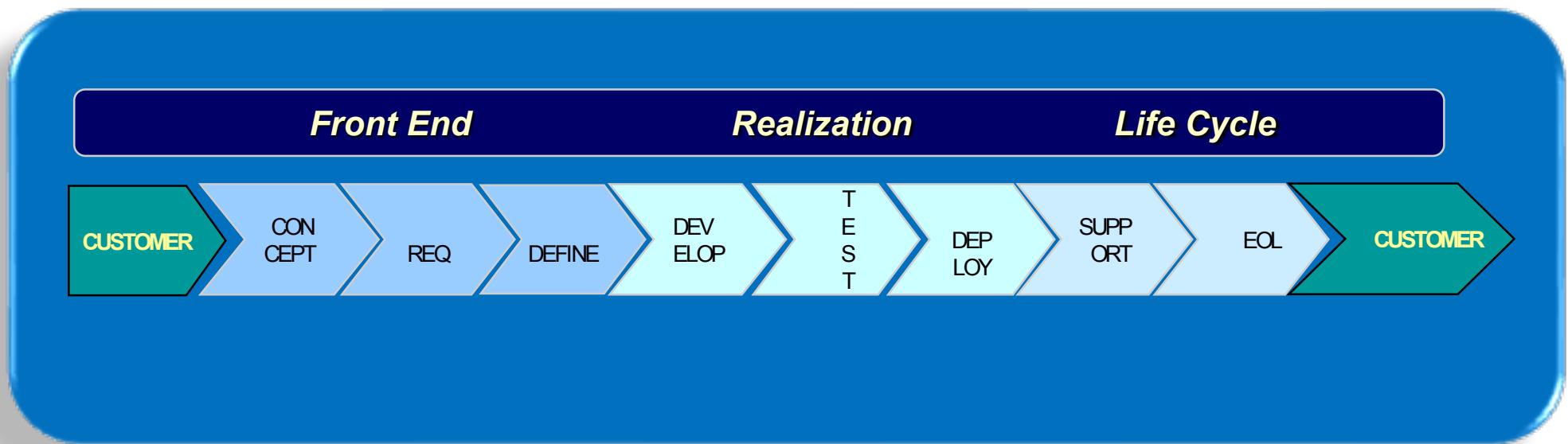
# Systems Engineering

A framework to

- Identify a **customer need**, an operational deficiency, or a market opportunity
- Translate the need into a **model** of a system that meets the need
- Focus on the **interfaces** not just the elements
- Address all **lifecycle** issues up-front during design for maximum leverage

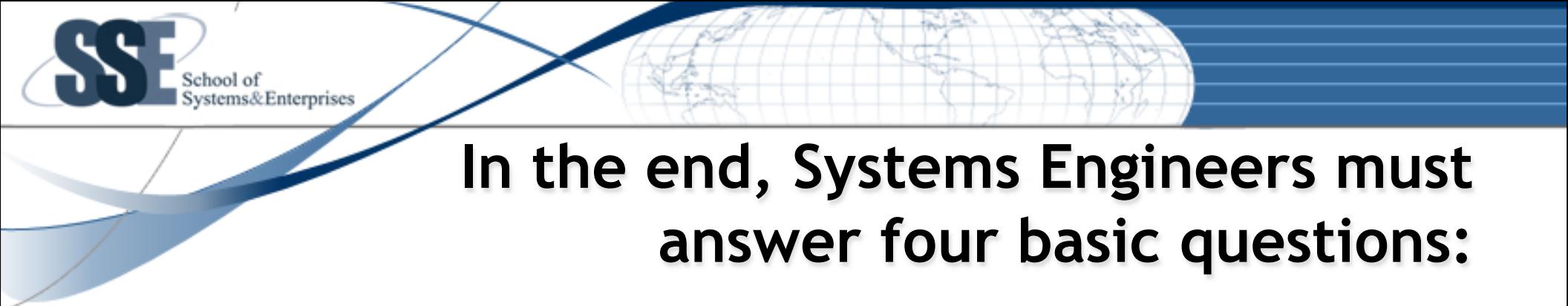


# Address all lifecycle issues up-front during design



*When should you  
Ensure that a  
System Works and is Robust?*

*How ?*



# In the end, Systems Engineers must answer four basic questions:

## 1. Will the system work?

- And how do you know?

## 2. Is it robust?

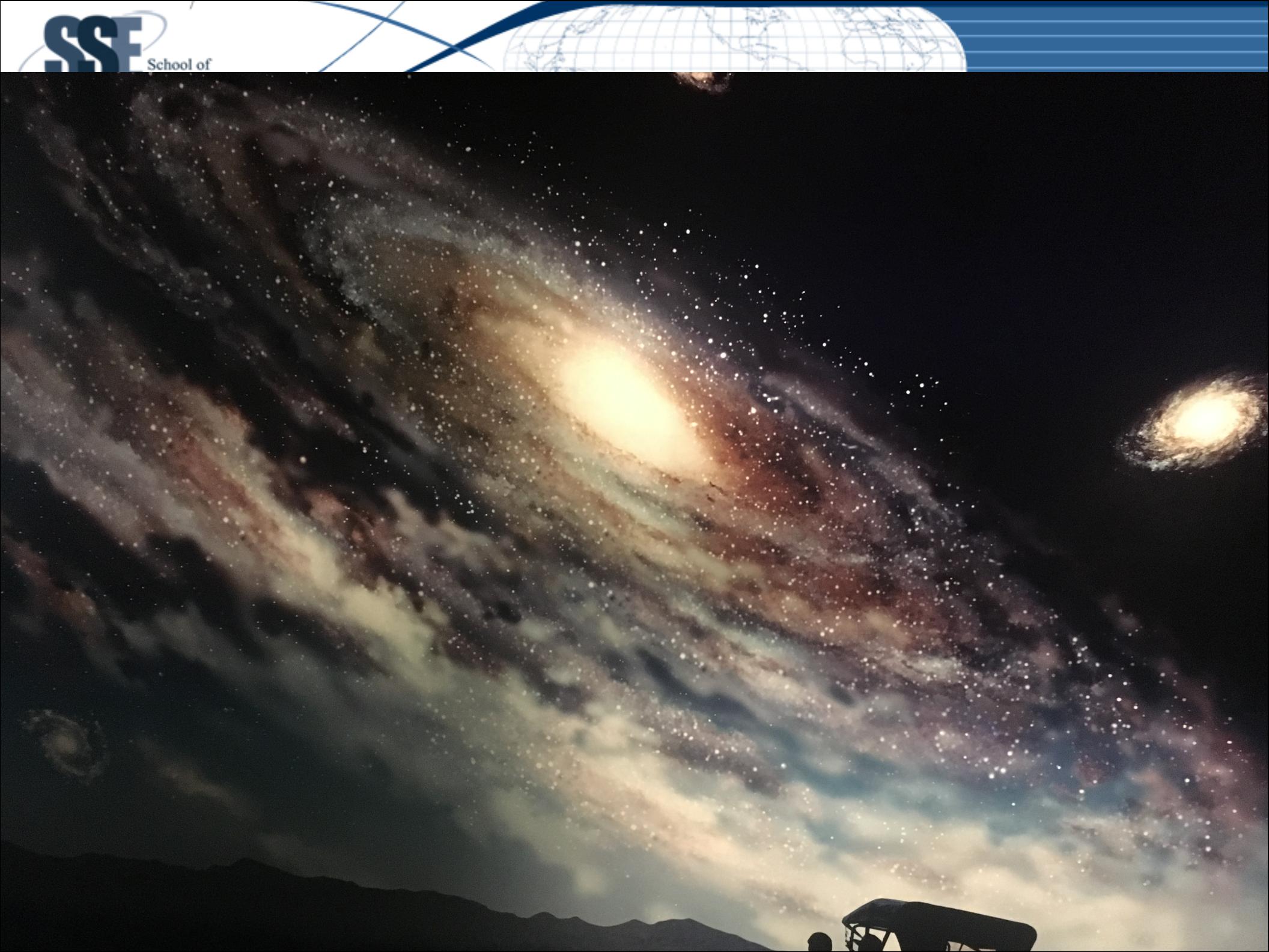
- What happens when the design conditions change?

## 3. Is it efficient?

- What is the relationship between means and ends?

## 4. What might be its unintended consequences?

*Dr. Michael Griffin  
former NASA Administrator*





# Will the system work?

Before



After

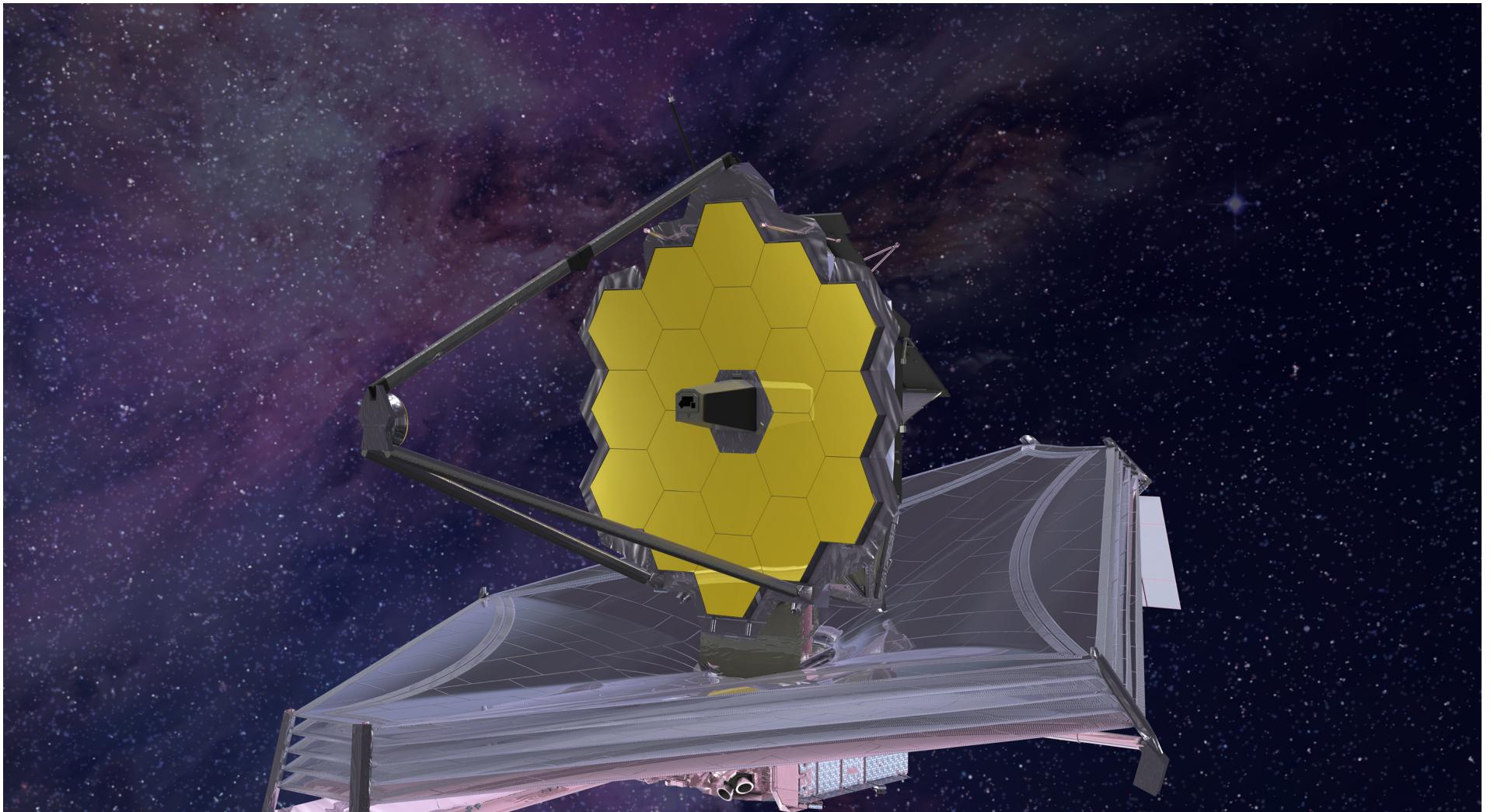


## Hubble Space Telescope (1.5 billion dollar)

- Spacing between lens and mirror off by 1.3 millimeters during grinding and polishing
- Caused an aberration of 0.001 arcseconds from design specification
- Error was 100,000 times the desired 1/50 of the wavelength of light
- Tests that could have detected the problem were not conducted because they were deemed too costly (~\$10M)
- Price tag for the Space Shuttle repair mission was \$674M!



John Webb







# Is the system robust?

## Robust

*- capable of performing  
without failure under a  
wide range of conditions*

*webster.com*



# Is the system efficient? How do means relate to ends?

## Denver International Airport

- Planned for totally automated baggage-handling system. Made this decision with only 2 years until airport opening
- Required 100 distributed PCs with new algorithms to optimize scheduling of carts
- Also needed uninterrupted power supplies to avoid system “crashes”
- **Result: Airport opening delayed 16 months, at a cost of \$500M. And the baggage system was abandoned**

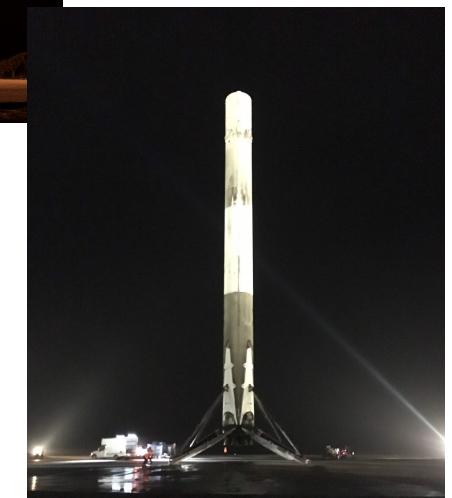
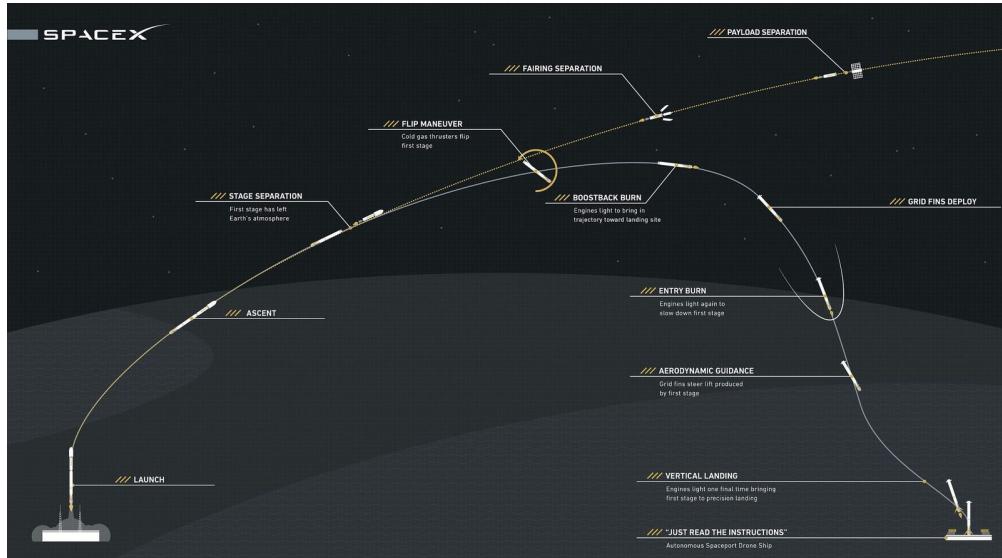




# What might be the unintended consequences?



# When complex systems work, it's not just luck!



*And it's not just that the design engineers did a better job this time!*



- Project Introduction
- Let's assign the groups and you will choose the subsystem to work on.

# Class Schedule (1 of 2)

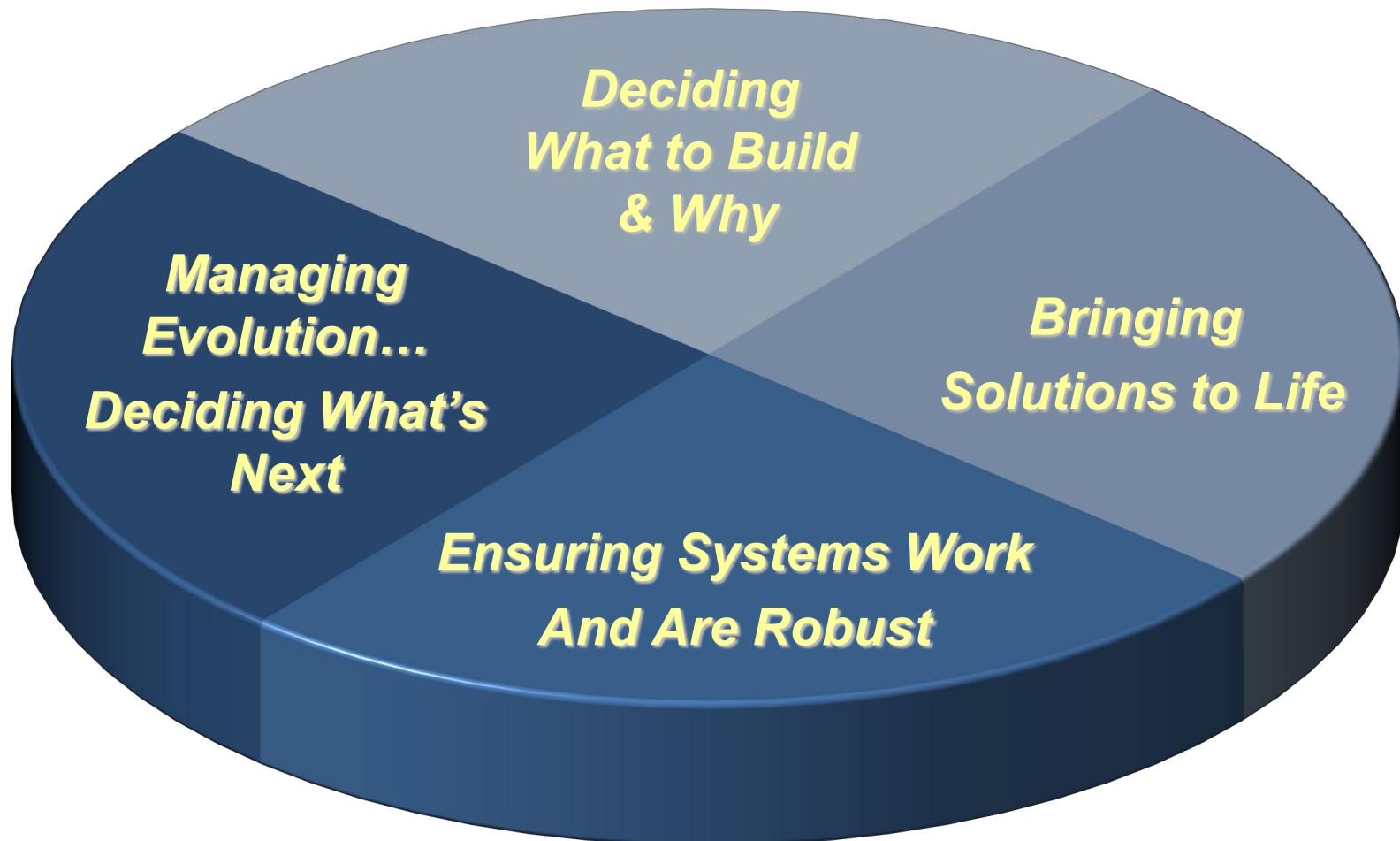
| Week | Topic   |
|------|---|
| 1    | <ul style="list-style-type: none"><li>• Thinking in Terms of Systems</li></ul> <p><i>Deciding What to Build and Why</i></p>                                     |
| 2    | <ul style="list-style-type: none"><li>• Defining the Problem</li></ul>  |
| 3    | <ul style="list-style-type: none"><li>• Developing a Solution</li></ul>   |
| 4    | <ul style="list-style-type: none"><li>• Formulating a Proposal</li></ul> <p><i>Bringing Solutions to Life</i></p>   |
| 5    | <ul style="list-style-type: none"><li>• Building a Functional Model<ul style="list-style-type: none"><li>• (Concept Review Storyboards due)</li></ul></li></ul> |
| 6    | <ul style="list-style-type: none"><li>• Concept Review</li></ul>  |
| 7    | <ul style="list-style-type: none"><li>• Implementing the Functions</li></ul>  |
| 8    | <ul style="list-style-type: none"><li>• Specifying Components</li></ul>   |
| 9    | <ul style="list-style-type: none"><li>• Design Review</li></ul>   |



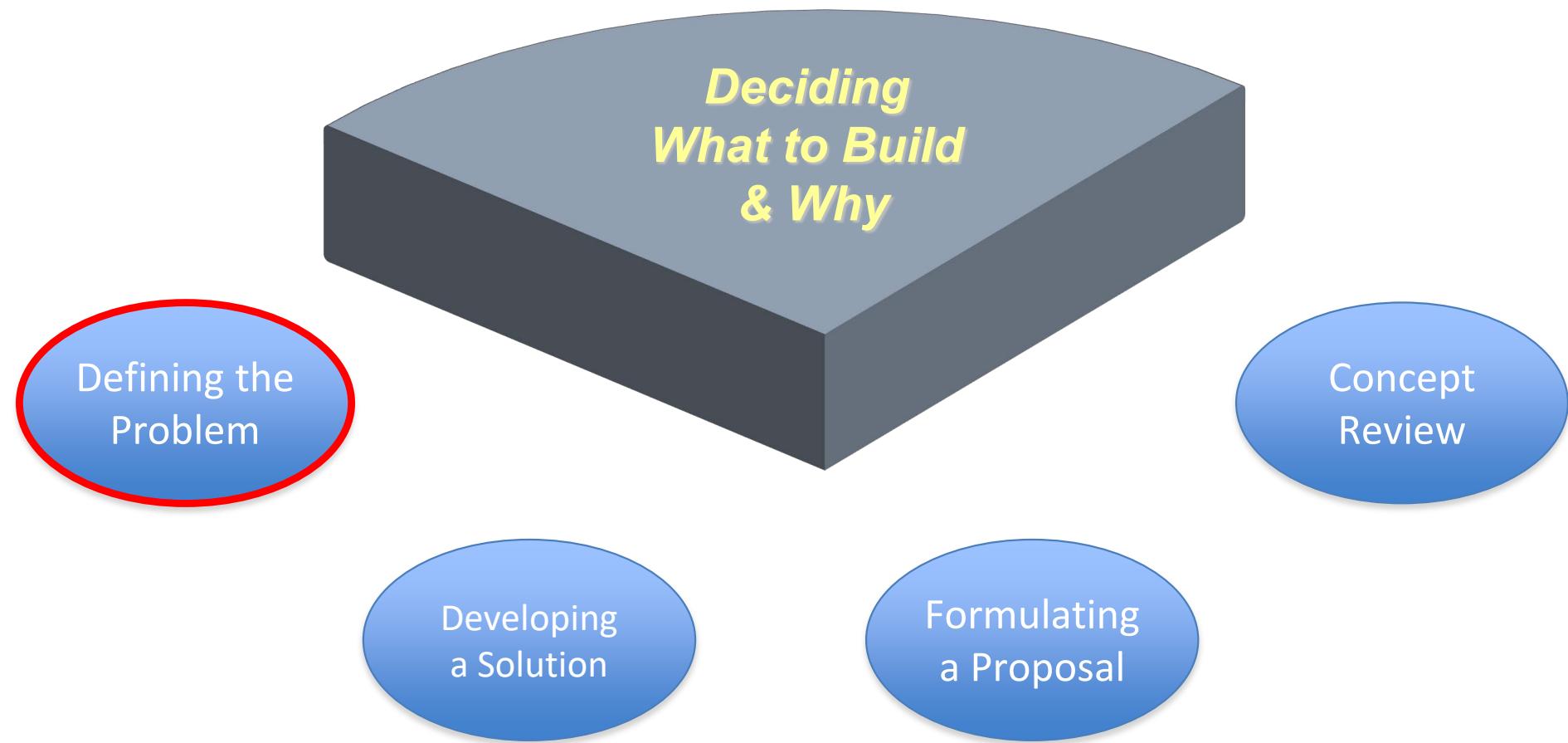
# Class Schedule (2 of 2)

| Week | Topic  |
|------|--|
|      | <i>Ensuring the System Works and Is Robust</i>   |
| 10   | <ul style="list-style-type: none"><li>• Integration and Test</li></ul>                           |
| 11   | <ul style="list-style-type: none"><li>• Ensuring Systems Work and Are Robust (M&amp;S)</li></ul> |
| 12   | <ul style="list-style-type: none"><li>• Designing for the Lifecycle</li></ul>                    |
| 13   | <ul style="list-style-type: none"><li>• Test Readiness Review</li></ul>                          |
|      | <i>Managing Evolution...Deciding What's Next</i>   |
| 14   | <ul style="list-style-type: none"><li>• Technology and Innovation</li></ul>                      |
| 15   | <ul style="list-style-type: none"><li>• Final Project Submission</li></ul>                       |

# SYS 581 Introduction to Systems Engineering



# SYS 581 Introduction to Systems Engineering



# Defining the Problem

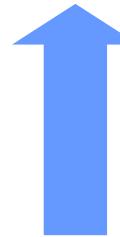
“The beginning is the most important part of the work.”

*...Plato, 400 B.C.*

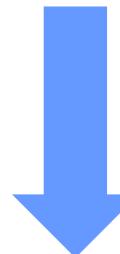
# Defining the Problem Key Questions:

1. What operational need or market opportunity is your system intended to address?
2. Who are the most important stakeholders and what are the key requirements of each?
3. What are the three to five most important features of your system that will distinguish it from those of your competitors?

**Why?**



**What?**



**How?**

# Needs and Opportunities

- What operational need or market opportunity is the system intended to address?
  - Who has the need?
  - What is it they wish to do - or do better?
  - How do they satisfy the need today, even if not fully or well?
  - Who would be willing to pay to have the need met? Why?

# What Problem Are We Trying to Solve?



How the customer explained it



How the Project Leader understood it



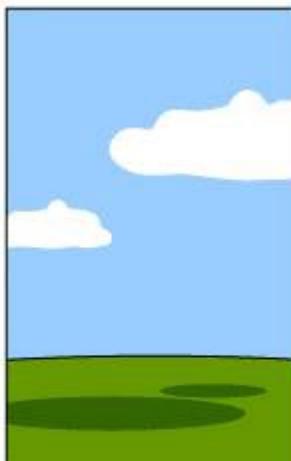
How the Analyst designed it



How the Programmer wrote it



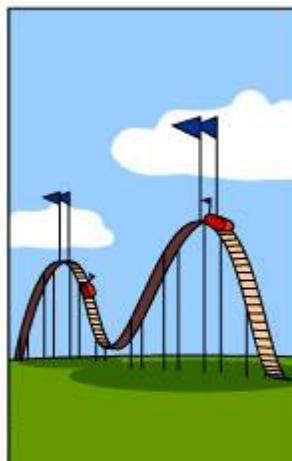
How the Business Consultant described it



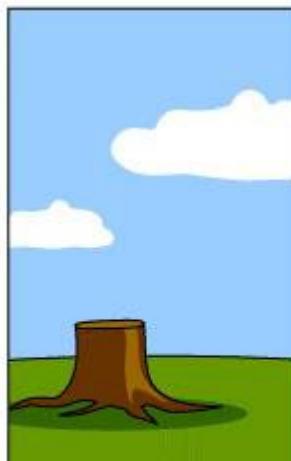
How the project was documented



What operations installed



How the customer was billed



How it was supported



What the customer really needed

# Be sure you are talking about a need, not a solution...

- ❖ A need exists in the domain of the customer, not the domain of the system
- ❖ A need exists independent of any solution to that need (or even the possibility of a solution)
- ❖ A need is expressed in the language of the customer (or user), not the language of the engineers



# Who is a Stakeholder?

- A group or an individual who is **affected by** or is in some way **accountable** for the outcome of an undertaking
  - Stakeholders can be classified as:
- **Customers** – An organization or individual that has **requested** a product and will **receive** the product to be delivered. Examples:
  - An end user of the product
  - The acquiring agent for the end user
  - The requestor of the work product from a technical effort
- **Other interested parties** who provide broad overarching **constraints** within which the customers' needs must be achieved, or who **have influence** on success of the system. Examples:
  - Those affected by the resulting product
  - Those affected by the manner in which the product is realized or used
  - Those who have a responsibility for providing life-cycle support services (e.g. design, manufacturing, operations, maintenance)



# Examples of Stakeholders

| Relative to Org | Stakeholder                           | Typical Expectations   |
|-----------------|---------------------------------------|--|
| External        | Customer                              | Expected level of product quality, delivered on time, affordable, life cycle support & services                          |
|                 | Subcontractors/vendors                | Well defined requirements  |
|                 | Local, State, National Public         | Products must not contaminate the environment  |
| Internal        | Org Management                        | Internal Commitments met (cost, schedule), good status provided, compliance with org policies, directives and procedures |
|                 | PM                                    | Expected technical work products delivered on time and can be used for decision making                                   |
|                 | Technical Team members                | clear tasks, job security, rewards, teamwork   |
|                 | Functional Organizations (e.g., test) | Test support products available, clear test requirements, recognition for project help                                   |

# If a stakeholder asks you “for” something, ask them “why.”

"People don't want to buy a quarter-inch drill. They want a quarter-inch hole!"

“The structure of a market, seen from the customers' point of view, is very simple: *They just need to get things done!*”

*Prof. Theodore Levitt*

Don't assume that the original needs statement is completely representative of the functional deficiency... or even true!  
(Rechtin, 1994)

# Stakeholders and their Requirements

- Who are the most important system stakeholders and what are the key requirements of each?
  - Who is the customer? If different from the customer, who is the user?
  - What other people or organizations will interact with the system once it is operational and in use? (Active stakeholders)
  - Are there other important stakeholders who, while they will not directly interact with the system, nevertheless have important requirements that the system must satisfy? (Passive Stakeholders)
  - What does each of the key stakeholders need or want the system to do? (System capabilities)
  - What other system attributes are important to one or more of the stakeholders? (System Characteristics)
  - How did you determine the requirements of each stakeholder? What evidence do you have that their requirements are valid and essential?

# Types of Stakeholders

- Active Stakeholders
  - Individuals, entities or other systems that will actively interact with the “system” once it is operational and in use
- Passive Stakeholders
  - Individuals, entities, other systems, standards, protocols, procedures, regulations that also impose requirements on the system
- Customer
  - The one who pays the bill!
  - May be either Active or Passive



# Examples of Stakeholders

| Relative to Org | Stakeholder                           | Typical Expectations   |
|-----------------|---------------------------------------|--|
| External        | Customer                              | Expected level of product quality, delivered on time, affordable, life cycle support & services                          |
|                 | Subcontractors/vendors                | Well defined requirements  |
|                 | Local, State, National Public         | Products must not contaminate the environment  |
| Internal        | Org Management                        | Internal Commitments met (cost, schedule), good status provided, compliance with org policies, directives and procedures |
|                 | PM                                    | Expected technical work products delivered on time and can be used for decision making                                   |
|                 | Technical Team members                | clear tasks, job security, rewards, teamwork   |
|                 | Functional Organizations (e.g., test) | Test support products available, clear test requirements, recognition for project help                                   |



# Stakeholder Brainstorming.



Toyota Hybrid



Iphone



Stevens Tower

# Stakeholder Requirements

- **Capabilities** - requirements that describe functions desired by the stakeholders
  - Examples?
- **Characteristics** - requirements that reflect other important system attributes
  - Examples?

# “Voice of the Customer”

Can take many approaches:

- Surveys or interviews
- Scenario development or user experience articulation
- “Camping out” or “fly on the wall” approaches
- Product value analysis
- Work with the lead or most innovative customers
- Input from the technologists, engineers, and architects, internal to the organization

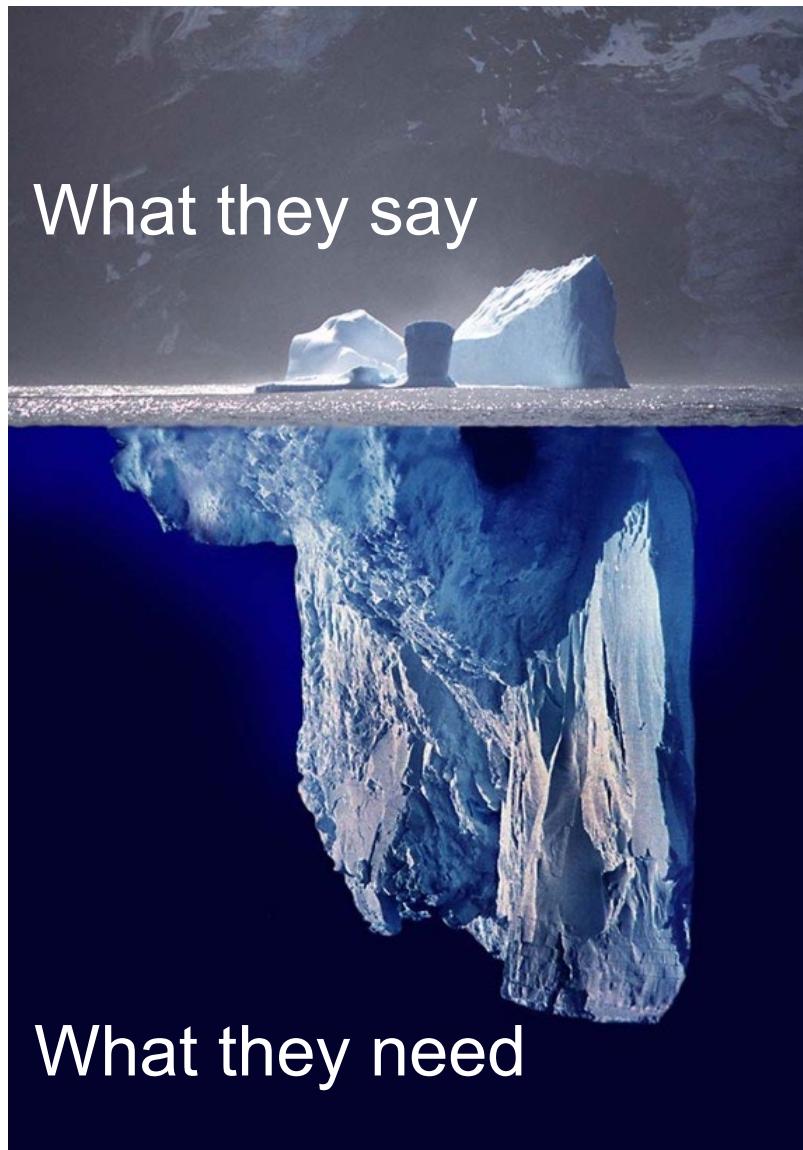
# Sometimes stakeholders speak in a requirements voice...

- Statements defining capabilities the stakeholder is missing today that can lead to innovative solutions
- Helps us understand the “why” of features/functions asked for by the stakeholders; this helps us deliver differentiated solutions
- Helps us understand the problem (stated or unstated) that the customer is experiencing today
- Example:
  - *“I need to have the same communications system features at home as I have at work or while traveling.”*

## ...and sometimes they “paint” verbal pictures or images.

- Verbal impressionistic characterizations of the stakeholder’s environment
  - A word picture - what life is like for them, and what motivates them
- Images are NOT:
  - Statements of the stakeholder’s need for the product
  - Statements of the stakeholder’s requirements of the product
- Fit the theme: “What pictures or scenes come to mind when you visualize?”
  - Scenes of the stakeholder’s current environment, not future projections
- Rich, vivid, and specific - NOT broad
- Examples:
  - ***“It’s a problem - if they are not dialing, I am losing money”***

# The Iceberg principle: “The most important requirements are beneath the surface!”



## Ask probing questions

- Why?; could you explain?; can you give me an example; what else...?



## Listen actively

- Paraphrase content
- Reflect emotion



# Ask probing questions

- The principal shortcoming of interviewers is a failure to probe
  - Customers talk in terms of features
    - “We need three additional telephone lines”
  - Interviewer needs to probe to find out why that feature is important
  - Determine what the customer need is
    - “I want to add staff”
- The interviewer’s favorite words and phrases should be:
  - Why?, Could you explain?, Could you share?, Describe; give me an example, What else.....?
- Go for the emotional response
  - What did you *hate* most about X? What was the “worst”, “best”, “most loved”, etc.
- Don’t settle or stop with vague responses
  - I.e, “easy”, “fast”, and “good” aren’t good enough

## *Listen actively*

- Paraphrase content: “*So what I think you’re saying is...*”
- Reflect emotion: “*It sounds like ... really makes you angry.*”

- Turn your neighbor and ask about their needs with probes. ( Roles: system engineer/ customer)
- You are designing a technology which will help them to maintain a healthy eating or living habit. Understand/identify that activity and related needs and list their exact needs

# Stakeholder Expectations Definition - Best Practice Process Flow Diagram

## Activities



### Input

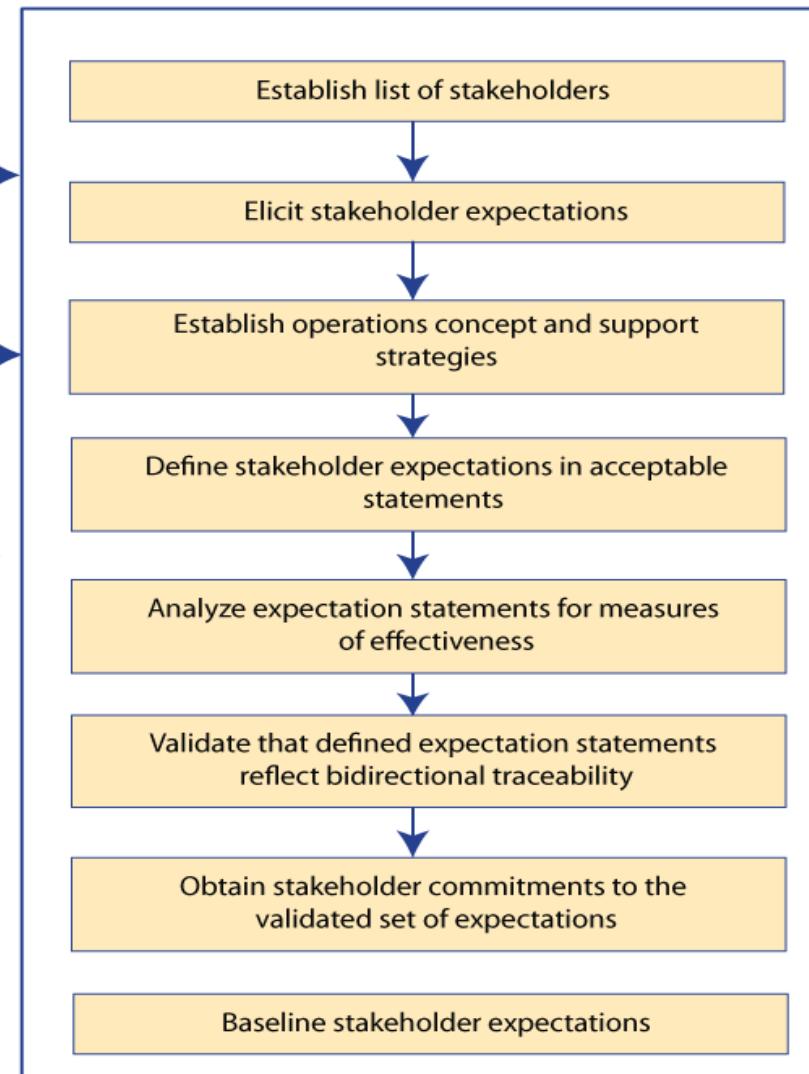
*From Project*

Initial customer expectations

Other stakeholder expectations

*From Design Solution Definition (recursive loop) and Requirements Management and Interface Management processes*

Customer flowdown requirements



### Output

*To Technical Requirements Definition and Requirements Management and Interface Management processes*

Validated stakeholder expectations

*To Technical Requirements Definition and Configuration Management processes*

Operations concept

Enabling product support strategies

*To Technical Requirements Definition and Technical Data Management processes*

Measures of effectiveness

# Requirements Definition

- Requirements describe the **necessary functions and features of the system** we are to conceive, design, implement and operate.
  - Performance
  - Schedule
  - Cost
  - Other Characteristics (e.g. lifecycle properties)
- Requirements are often organized hierarchically
- At a high level requirements focus on what should be achieved, not how to achieve it
- Requirements are specified at every level, from the overall system to each hardware and software component.
- **Critically important to establish properly ( for good design)**
- **Many of the cost overruns presented in Lecture 1 are caused by over-ambitious or missing requirements**

# Capturing Stakeholder Requirements

- Begin with the Stakeholder's Requirements Voice (from transcripts)
  - “We need to be able to use the new screens without a lot of new documentation or training; moving the buttons around and making it pretty doesn’t do it.”
- Support with Customer Images
  - “Lost opportunity: These people who are using the new screens are answering 60% of all calls and losing 40%”
- Generate Key Words - (based on stakeholder experience)
  - data screens, input, output, viewing, customer service, information access, documentation, training
- Write the Customer Requirement:
  - The screen is visually pleasing. (*poor*)
  - The screen looks similar to existing screens, with a minimum number of differences (*good*).

# Requirements Guidelines

- No abstract language
  - Poor: Screen is pleasing -- “pleasing” means different things to different people
  - Good: Screen has the same look as existing screens
- No statement of “how to” -- should be solution free
  - Poor: Data used by call center should be entered by skilled administrative assistants -- indicates “how” vs. “what” functionality is required
  - Good: Data can be input with a minimum number of errors.
- Requirement should be quantifiable
  - Poor: User can access information easily -- “easily” is not quantifiable
  - Good: User can access information in a minimum number of steps

# How Much Input is Enough?

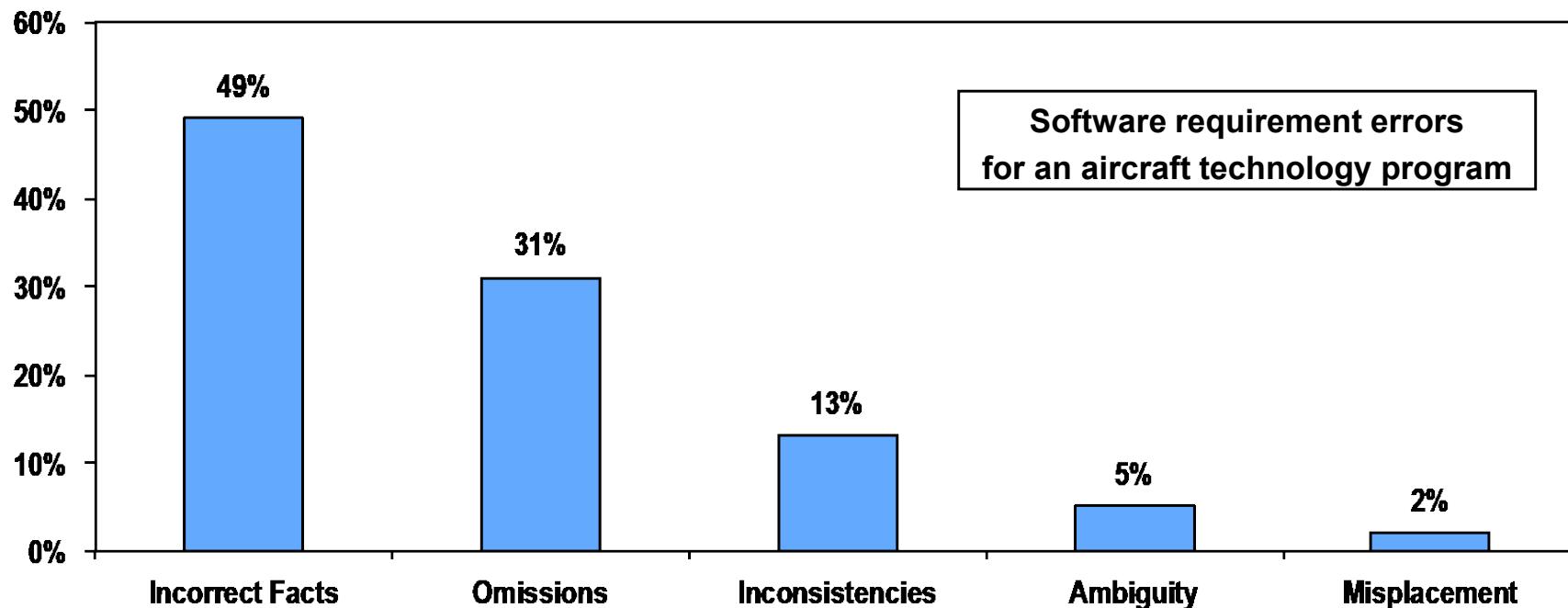
## Capturing the Voice of the Customer at Lexus



- Toyota launched its luxury automobile project in 1983
- Japanese designers lived in California while creating initial designs
  - Rented a home in upscale neighborhood
  - Rented and drove luxury cars
  - Lived the target buyer lifestyle
- After 5 months, the team returned to Japan with exterior and interior design proposals
- Lexus vehicles were an immediate success - well received by target buyers

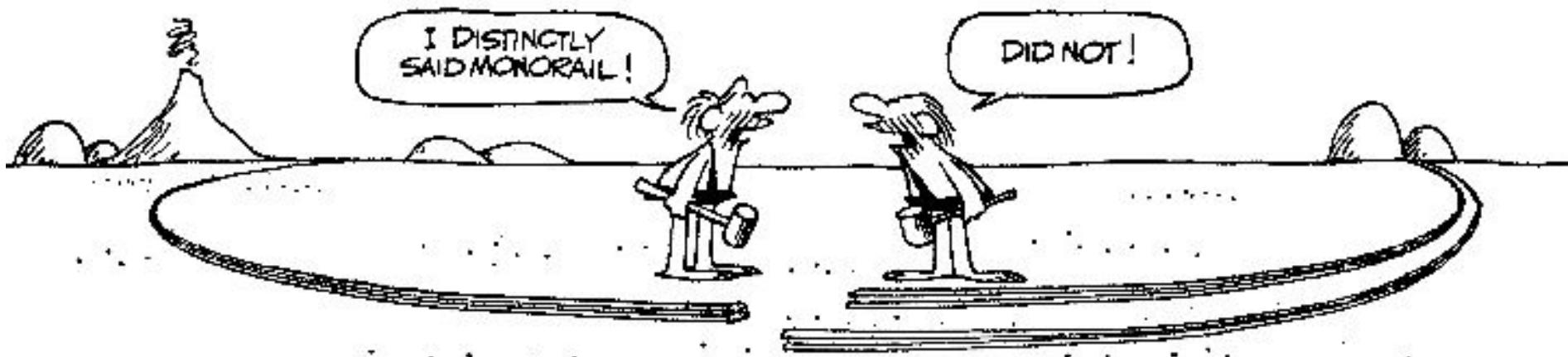
# Most requirements errors will not be discovered during design.

## Types of Requirement Errors



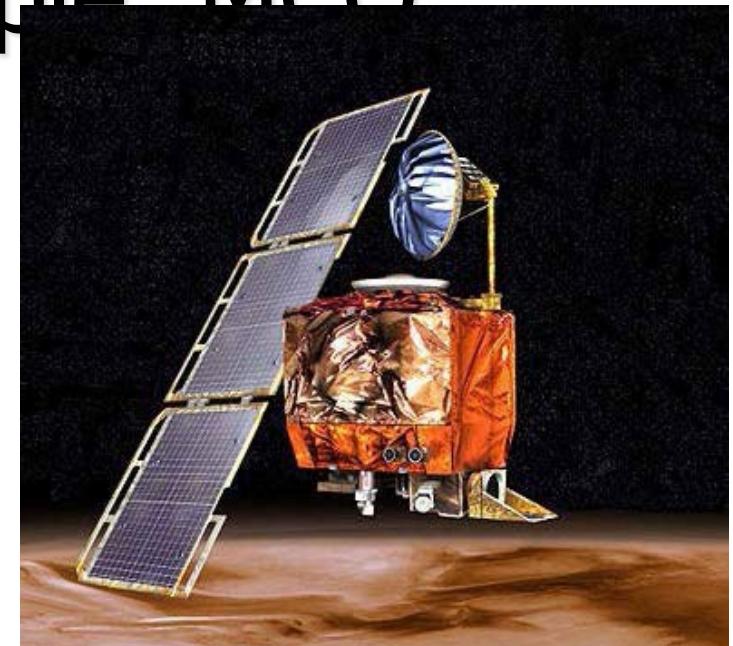
Ref: Hooks and Farry, Customer-Centered Products, Amacom, 2001, p 6

# Most requirements errors will not be discovered during design.



# Poor requirements example: MCO

- Mars Climate Orbiter (MCO) was launched by NASA on December 11, 1998
  - Intended to study Martian climate, weather and surface changes and act as communications relay back to Earth
  - **However, disintegrated during orbit insertion on Sept 23, 1999 → approach too close → requirements not followed**
- Units confusion problem: Ground Software produced output in non-SI units (lbf-sec) instead of SI units: Ns
  - Calculation of total momentum produced by engine burns needed by GNC
- Contract between NASA and Lockheed Martin did specify SI-units
  - This requirement was flowed down to the Software Interface Specification (SIS), but not verified later and not implemented in the AMD



This image is in the public domain.

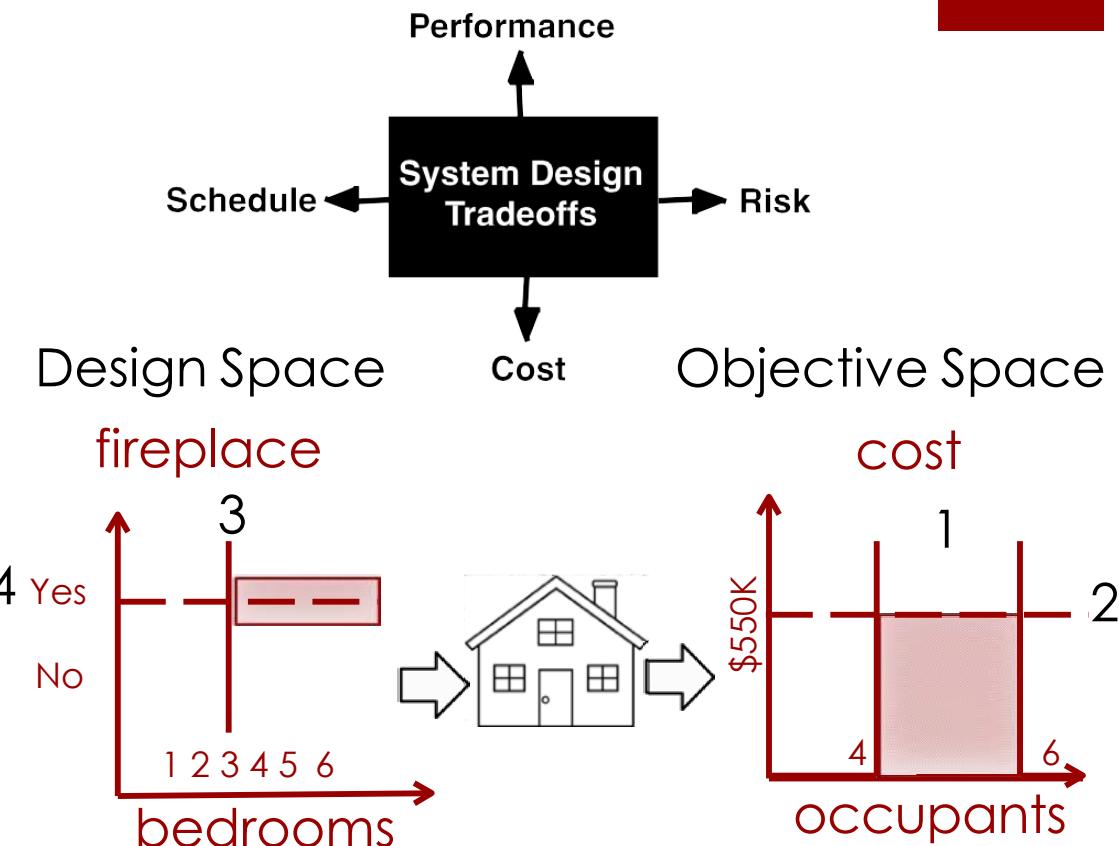
From the accident report:

*"Items that the mission assurance manager could have addressed for MCO included ensuring that the AMD file met the requirements of the SIS .."*

[ftp://ftp.hq.nasa.gov/pub/pao/reports/1999/MCO\\_report.pdf](ftp://ftp.hq.nasa.gov/pub/pao/reports/1999/MCO_report.pdf)

# Requirements set *constraints* and *goals* in the design and objective space

- When designing systems we always have tradeoffs between performance, cost, schedule and risk
- “**Shall**” ... Requirements help set constraints and define the boundaries of the design space and objective space
- “**Should**” ... requirements set goals once “shall” requirements are satisfied
  - Two main spaces:
  - Design Space – the things we decide as engineers
  - Objective Space – the things our systems/products achieve and what our customers care about



1. “The house shall sleep between 4 and 6 people”
2. “The total build cost should be less than \$550K”
3. “The house shall have at least 3 bedrooms”
4. “The house should have a fireplace”

# Requirements vs. Specifications

- Requirements specify what the product or system shall/should do:
  - **Functions** it shall perform
  - How well it should perform these
  - Degree of automation of the system (what operators must do)
  - Compatibility with other devices etc...
- Specifications describe how the system is built and works
  - The **Form** it is made of
  - Materials used in the system
  - Overall dimensions
  - Schematics, Blueprints etc...
  - User Interface

## “Description”

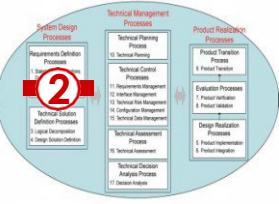
- Large enough to accommodate big dishes
- 1,200 Watts of power to reheat food quickly
- One touch settings for different food types (rice, pizza, frozen meals) ...
  - Etc...



Kenmore Elite  
Countertop 2.2 cu ft  
Microwave Oven

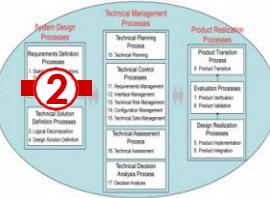
## “Specification”

- Stainless steel exterior
- Dimensions: 24" x 14" x 19"
  - Weight: 45.5 lbs
- General warranty: 1 year
- Power cord: included
- Etc ...



# Purpose of **Technical Requirements Definition**

- The Technical Requirements Definition Process
  - Is used to **transform** the baselined stakeholder **expectations** (input) into unique, quantitative, and measurable technical **requirements** (output)
    - Requirements
      - Come in many flavors
    - Should be expressed as well-written “**shall**” **statements** that can be used for defining a design solution



## Development (1/2)

- Establishes the **basis for agreement** between the stakeholders and the developers on what the product is to do
  - Reduces the development effort because **less rework** is required to address poorly written, missing, and misunderstood requirements.
  - Forces the relevant stakeholders to consider rigorously all of the requirements **before** design begins
  - Careful review can reveal omissions, misunderstandings, and inconsistencies **early** in the development cycle
- Provides a **basis for estimating costs and schedules**
  - The description of the product to be developed as given in the requirements is a **realistic basis** for estimating project costs and can be used to evaluate bids or price estimates

# Importance of Technical Requirements Development (2/2)



- Provides a baseline for verification
- Organizations can develop their validation and verification plans much more productively from a **good** requirements document.
- The requirements document provides a baseline against which **compliance** can be measured.
- The requirements are also used to provide the stakeholders with a **basis for acceptance** of the system.
- **Facilitates transfer of the product to new users.**
- **Serve as a basis for later enhancement or alteration of the finished product.**



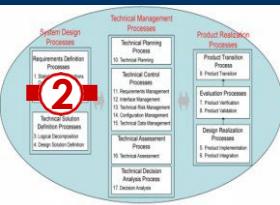
# Types of Requirements

- **Functional Requirements** define what functions need to be done to accomplish the mission objectives
  - Example: The Thrust Vector Controller (TVC) shall provide vehicle control about the pitch and yaw axes.
  - This statement describes a high level function that the TVC must perform.
    - Statement has form of Actor – Action Verb – object acted on
- **Performance Requirements** define how well the system needs to perform the functions
  - Example: The TVC shall gimbal the engine a maximum of 9 degrees, +/- 0.1 degree
- **Constraints** are requirements that cannot be traded off with respect to cost, schedule or performance
  - Example: The TVC shall weigh less than 120 lbs.



# Types of Requirements

- **Constraints** are requirements that cannot be traded off with respect to cost, schedule or performance
  - Example: The TVC shall weigh less than 120 lbs.
    - **Interface Requirements**
  - Example: The TVC shall interface with the J-2X per conditions specified in the CxP 72262 Ares I US J-2X Interface Control Document, Section 3.4.3.
    - **Environmental requirements**
  - Example: The TVC shall use the vibroacoustic and shock [loads] defined in CxP 72169, Ares 1 Systems Vibroacoustic and Shock Environments Data Book in all design, analysis and testing activities.
- **Other -ilities requirement types described in the SE Handbook include: human factors, reliability requirements, and safety requirements.**



# Attributes of Acceptable Requirements

- A complete sentence with a **single** “shall” per numbered statement
  - Characteristics for each Requirement Statement:
    - **Clear** and **consistent** – readily understandable
    - **Correct** – does not contain error of fact
  - **Feasible** – can be satisfied within natural physical laws, state of the art technologies, and other project constraints
    - **Flexibility** – Not stated as to how it is to be satisfied
    - **Without ambiguity** – only one interpretation makes sense
      - **Singular** – One actor-verb-object requirement
  - **Verify** – can be proved at the level of the architecture applicable
  - Characteristics for pairs and sets of Requirement Statements:
    - **Absence of redundancy** – each requirement specified only once
      - **Consistency** – terms used are consistent
    - **Completeness** – usable to form a set of “design-to” requirements
    - **Absence of conflicts** – not in conflict with other requirements or itself

# Requirements Writing “Workshop”

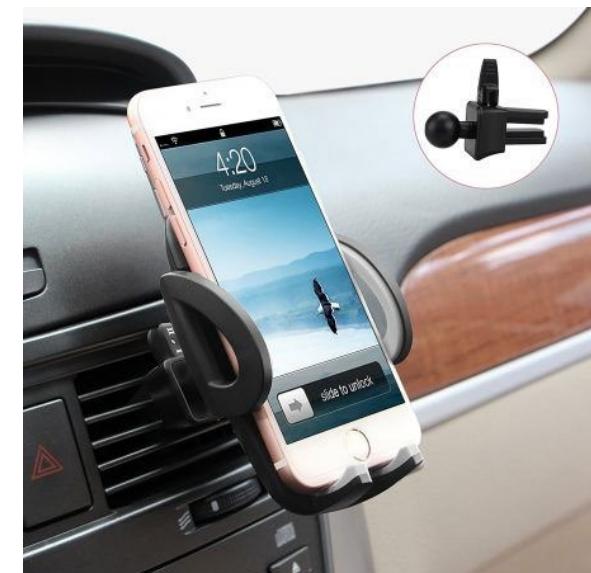
- Turn to your partner exercise (10 min)
- Together write a good requirement that was (possibly) used in the development of one of the following solutions
  - A – Mr. Sticky tape for trapping flies
  - B – New BMW i3 electric car
  - C – Phone holder



A



B



C

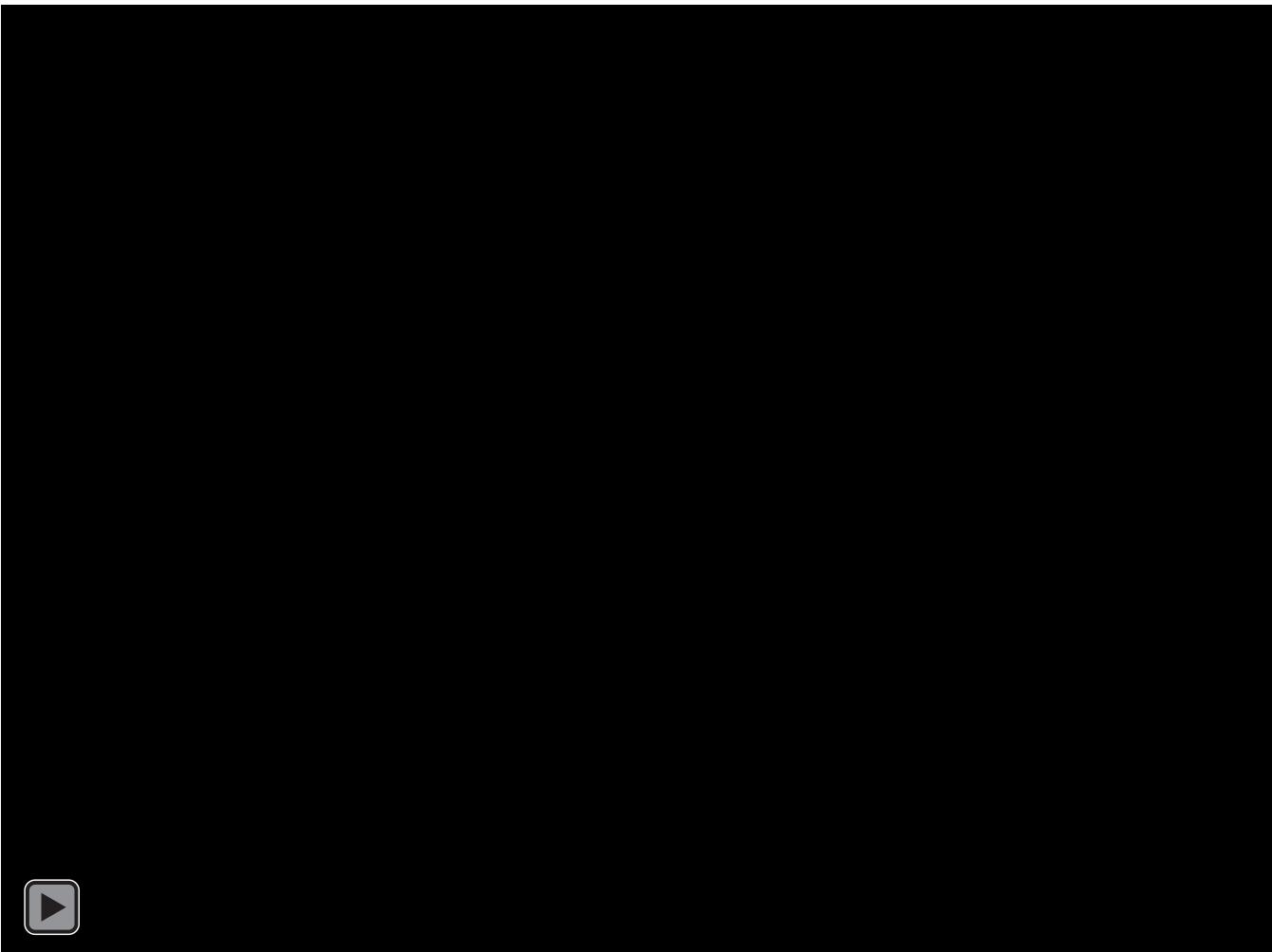
# Acceptance Criteria

- What are the three to five most critical criteria for judging that your system is a success?
  - Why are these the most important criteria?
  - Why are they sufficient for assuring system success?
  - How do they distinguish the system from other potential solutions that meet the need?

# Acceptance Criteria frame the Operational Concept

- **Vision**
  - The benefit to be enjoyed by the stakeholders once the system is implemented
- **Acceptance Criteria**
  - Three to five results that will define project success
  - Sometimes referred to as mission requirements, product pillars or conditions of satisfaction
  - Are not part of the trade space
  - Must be agreed to by all parties up front

# Example: Acceptance Criteria for the Apollo Program



*President John F. Kennedy,  
Address to Joint Session of Congress, May 25, 1961* <sup>46</sup>

# Acceptance Criteria for the Apollo Program

- “Landing a man on the moon...
- Returning him safely to the earth...
- Before this decade is out” (i.e. by 31 Dec 1969)

*President John F. Kennedy  
Address to Joint Session of Congress  
May 25, 1961*

# Reflections of a Project Manager

*“The Acceptance Criteria helped everyone (both stakeholders and the development team) focus on the critical and core capabilities of the system. It killed discussions about this whistle or that bell and focused us on the core functionality to be implemented.”*

# Defining Acceptance Criteria: The Key Questions

1. "What do we have to do to succeed?"
2. "Good! If that's all we did, would we be willing to declare success?"
3. Repeat 1&2 until the answer is yes.
4. If you have more than 5 criteria, ask "why" for each and begin again.

# Acceptance Criteria: Additional Examples

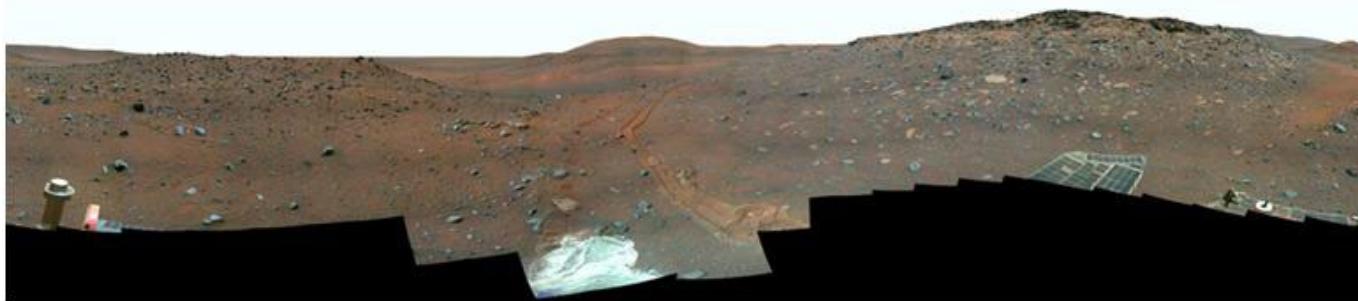
| SR-71<br>Blackbird   | Global Hawk<br>ATD  | FAA<br>NextGen  |
|--|---|---|
| <ul style="list-style-type: none"><li>★ 80,000 feet</li><li>★ Mach 3</li><li>★ Half the radar cross-section of the U-2</li></ul> | <ul style="list-style-type: none"><li>★ High altitude</li><li>★ Long range and endurance</li><li>★ Affordable</li></ul> | <ul style="list-style-type: none"><li>★ 3x the traffic</li><li>★ Safeguard the environment</li><li>★ Improve safety</li></ul> |

# Acceptance Criteria: Commercial Examples

| Handspring Visor   | Polycom SoundStation  | Iomega Zip Drive   |
|--|---|--|
| <ul style="list-style-type: none"> <li>◆ \$149 base price (\$249 for deluxe)</li> <li>◆ Use Palm OS and USB</li> <li>◆ Expansion slot for plug-and-play</li> <li>◆ Ready in twelve months</li> </ul> | <ul style="list-style-type: none"> <li>◆ Full-duplex, high-quality sound</li> <li>◆ Simple to use</li> <li>◆ Look "first-class," like it belongs in an executive conference room</li> </ul> | <ul style="list-style-type: none"> <li>◆ 100 Mb of data</li> <li>◆ "Fast enough"</li> <li>◆ Priced at \$200 to \$300</li> <li>◆ Simple to install</li> <li>◆ Ready in eleven months</li> </ul> |

# A More Recent Example: Mars Rovers (Spirit and Opportunity)

- The “Mission Success Statement” fit on a half sheet of paper, and carried very specific goals, such as the minimum number of 360-deg panorama images each rover was to collect on the surface of Mars (one) \*
- “The beauty of that was it lent a crystalline clarity to every single decision we ever had to make.” (*Steve Squyres, Principal Investigator of the MER mission, and Chairman, NASA Advisory Council*) \*



Panoramic photo mosaic taken by Spirit

\* Excerpts from Aviation Week & Space Technology, April 9, 2012

**By the way, Feb 18 2021**



# Acceptance Criteria and Agile

- In Agile, acceptance criteria refers to a set of predefined requirements that must be met in order to mark a user story complete.
- Acceptance criteria are also sometimes called the “definition of done” because they define the scope and requirements of user stories. They give developers the context needed to execute on a user story.

- **Acceptance Criteria Definition 1:** “Conditions that a software product must satisfy to be accepted by a user, customer or other stakeholder.” ([via Microsoft Press](#))
- **Acceptance Criteria Definition 2:** “Pre-established standards or requirements a product or project must meet.” ([via Google](#))

## A few traits of effective acceptance criteria



**Acceptance criteria should be testable.**



Since these requirements help formulate the definition of done for your engineers, they need to be easy to test. And the results of these tests must leave no room for interpretation. Tests should reveal straightforward yes/no or pass/fail results



**Criteria should be clear and concise.**



You aren't writing comprehensive documentation here. Keep your criteria as simple and straightforward as possible.

# A few traits of effective acceptance criteria

- **Everyone must understand your acceptance criteria.**
- Your criteria is useless if your developers can't understand it. If you're unsure about whether something is clear, take the time to ask and make adjustments until things are clear.
- **Acceptance criteria should provide user perspective.**
- Acceptance criteria is a means of looking at the problem at hand from a customer's standpoint. It should be written in the context of a real user's experience.

# Why Do You Need User Story Acceptance Criteria

1

Managing  
Expectations

2

Defining Scope  
and Reducing  
Ambiguity

3

Establishing  
Testing Criteria

4

Defending  
Against scope  
creep mid sprint

- **The Bottom Line:**

Acceptance criteria is an important component of every user story that an agile team works on.

It clearly defines the scope, desired outcomes of, and testing criteria for pieces of functionality that the delivery team is working on.

The process of creating and agreeing on acceptance criteria itself is also an invaluable communication opportunity between developers and product

- **HW 1: Due next week**

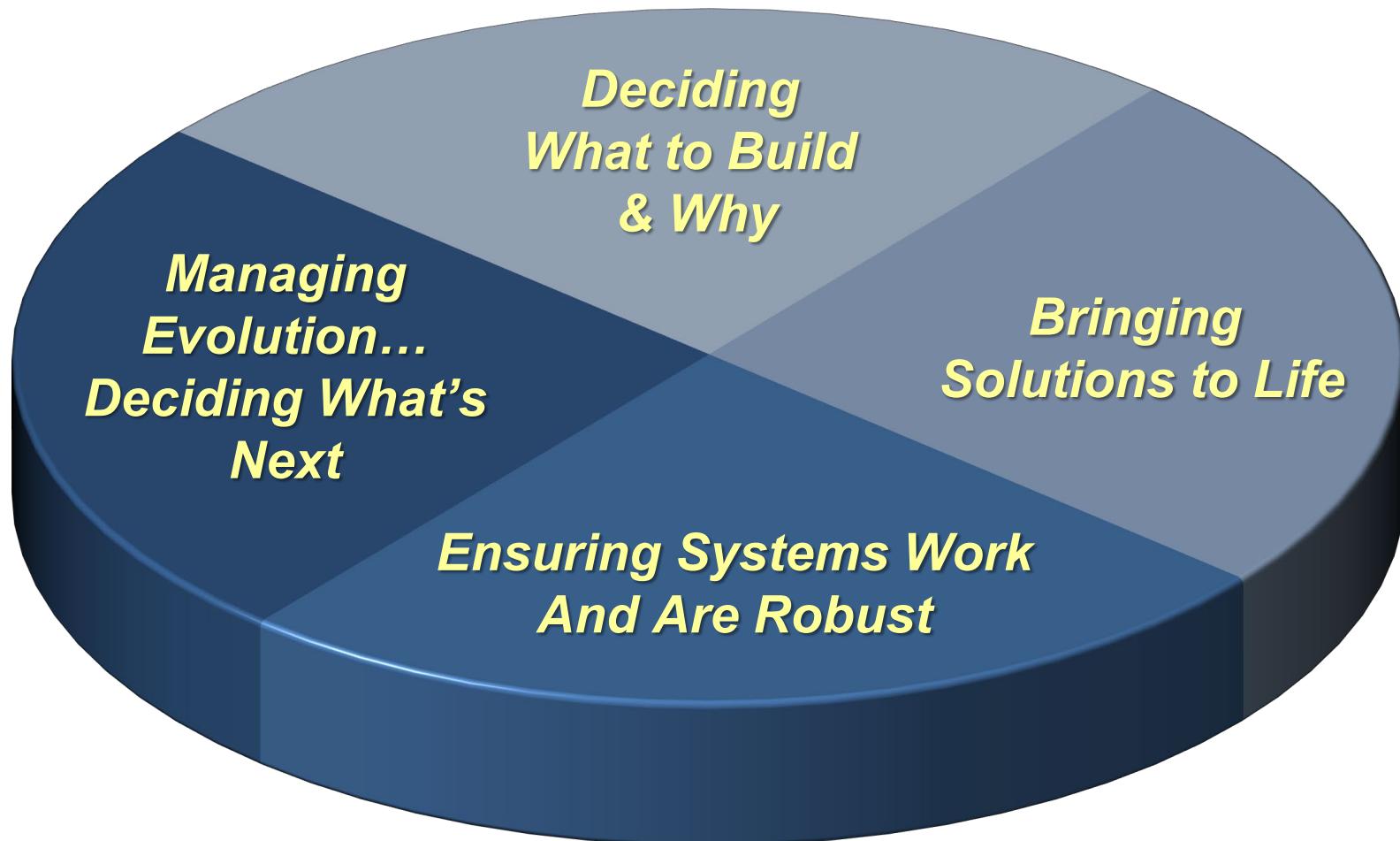
- 2 IT Project/System Failures since 2000 ( total min 2 max 3 pages)
- Due : February 6<sup>th</sup> at noon by Canvas

# Defining the Problem

## Key Questions:

1. What operational need or market opportunity is your system intended to address?
  - *Assessment Criteria: The need for the system is well understood, fully described in the language of the stakeholders and free of solutions.*
2. Who are the most important stakeholders and what are the key requirements of each?
  - *Assessment Criteria: The key stakeholders have been identified and their most important requirements defined, validated and clearly stated.*
3. What are the three to five most important features of your system that distinguish it from those of your competitors?
  - *Assessment Criteria: Features are specific, quantifiable (or readily observable), and important to the customer.*

# Course Design



# Class Schedule (1 of 2)

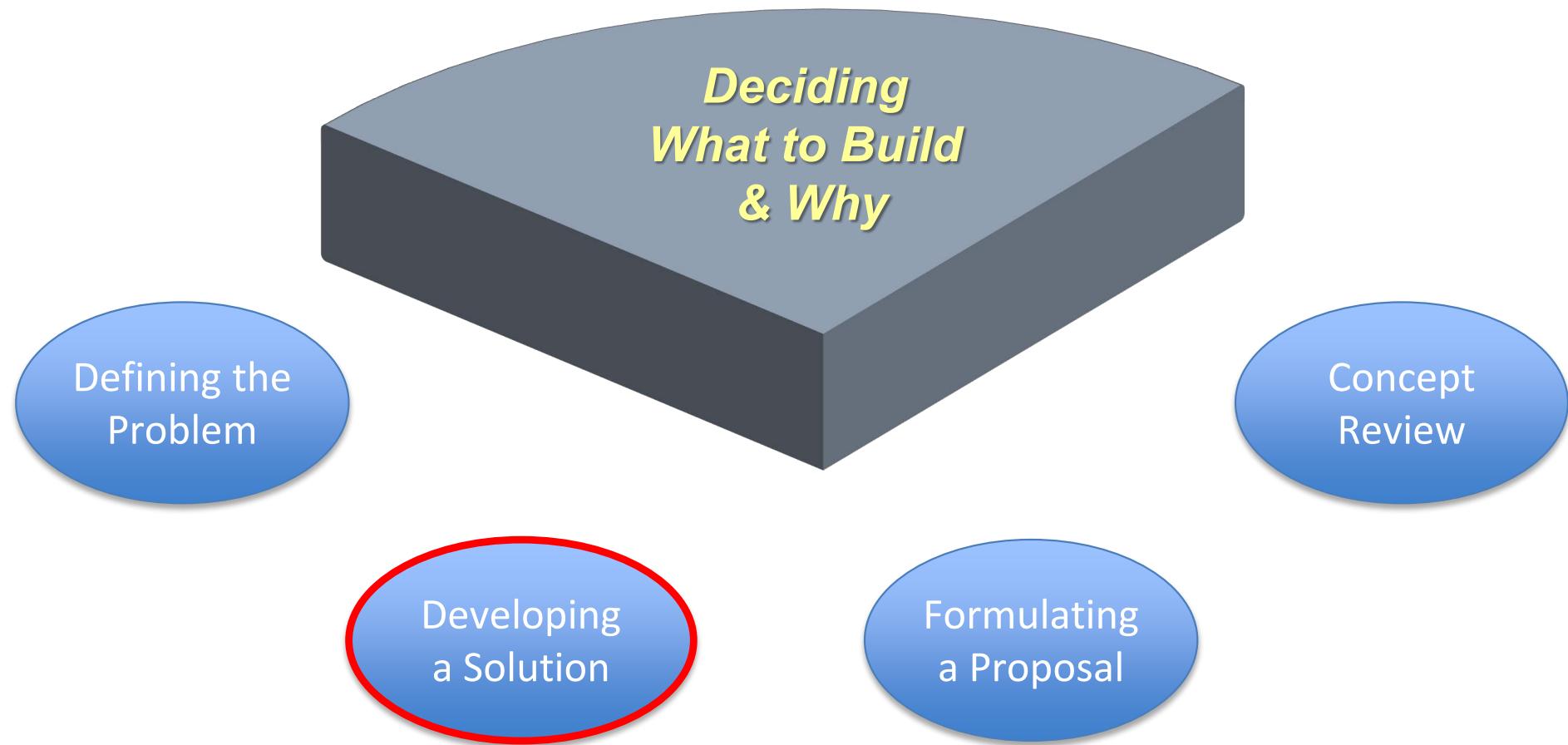
| Week | Topic  |
|------|--|
| 1    | <ul style="list-style-type: none"><li>• Thinking in Terms of Systems</li></ul> <p><i>Deciding What to Build and Why</i></p>                                |
| 2    | <ul style="list-style-type: none"><li>• Defining the Problem</li></ul>   |
| 3    | <ul style="list-style-type: none"><li>• Developing a Solution</li></ul>  |
| 4    | <ul style="list-style-type: none"><li>• Formulating a Proposal</li></ul> <p><i>Bringing Solutions to Life</i></p>  |
| 5    | <ul style="list-style-type: none"><li>• Building a Functional Model<ul style="list-style-type: none"><li>• (Concept Review slides due)</li></ul></li></ul> |
| 6    | <ul style="list-style-type: none"><li>• Concept Review</li></ul>   |
| 7    | <ul style="list-style-type: none"><li>• Implementing the Functions</li></ul>   |
| 8    | <ul style="list-style-type: none"><li>• Specifying Components</li></ul>  |
| 9    | <ul style="list-style-type: none"><li>• Design Review</li></ul>  |



# Class Schedule (2 of 2)

| Week | Topic  |
|------|--|
|      | <i>Ensuring the System Works and Is Robust</i>   |
| 10   | <ul style="list-style-type: none"><li>• Integration and Test</li></ul>                           |
| 11   | <ul style="list-style-type: none"><li>• Ensuring Systems Work and Are Robust (M&amp;S)</li></ul> |
| 12   | <ul style="list-style-type: none"><li>• Designing for the Lifecycle</li></ul>                    |
| 13   | <ul style="list-style-type: none"><li>• Test Readiness Review</li></ul>                          |
|      | <i>Managing Evolution...Deciding What's Next</i>   |
| 14   | <ul style="list-style-type: none"><li>• Technology and Innovation</li></ul>                      |
| 15   | <ul style="list-style-type: none"><li>• Final Project Submission</li></ul>                       |

# Course Design

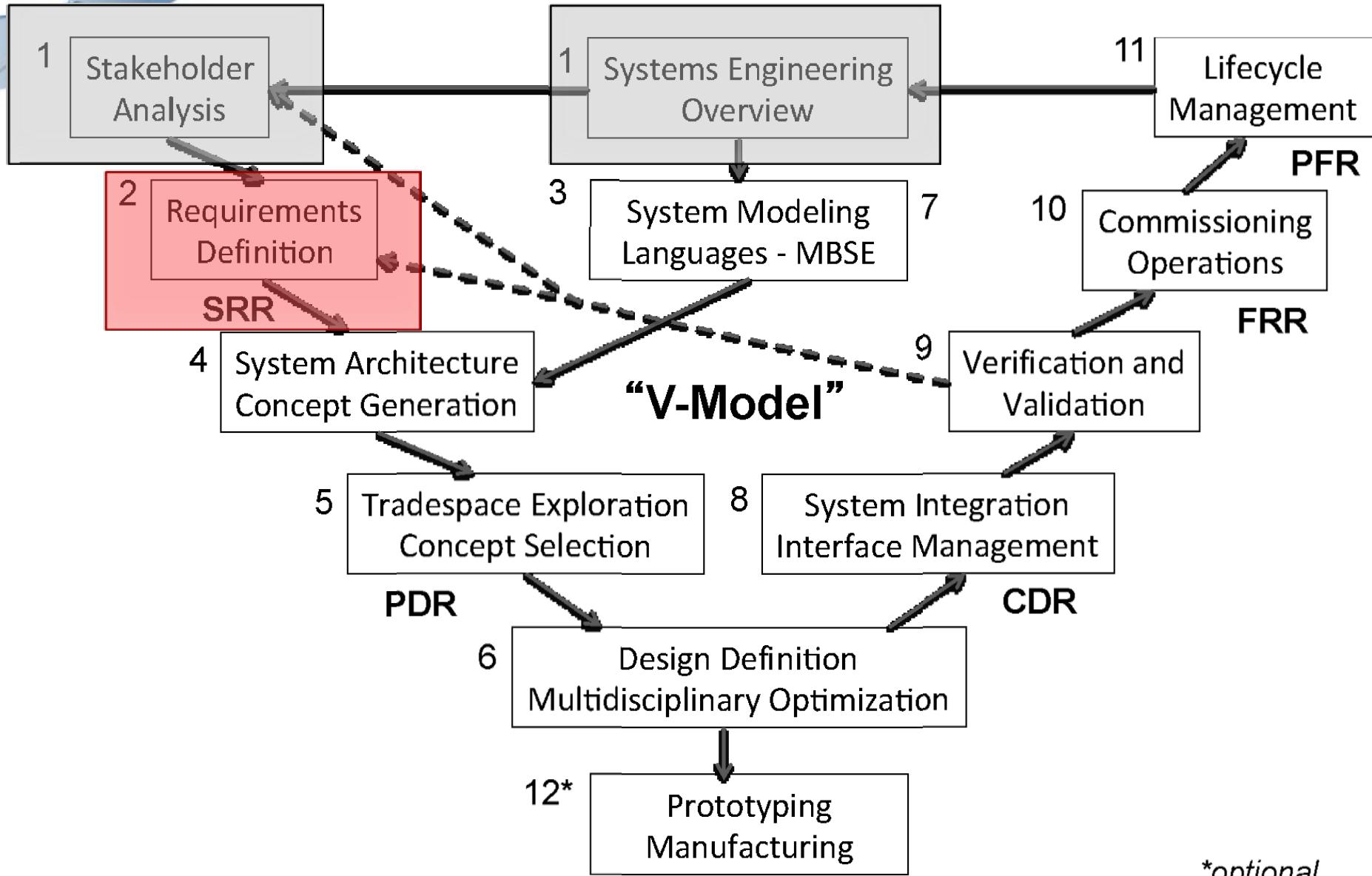




# IT Failure Discussion

# The “V-Model” of Systems Engineering

16.842/ENG-421 Fundamentals of Systems Engineering



\*optional



# Developing a Solution

# Developing a Solution

## *Key Questions:*

1. What is your proposed **system concept**?  
What alternative concepts did you consider and why did you select the one you proposed?
  
2. How will your proposed concept operate within the **larger context** to achieve its **intended purpose**?
  
3. What are the **key specifications** that will drive your system's design and development?

# Developing a Solution

## *Key Questions:*

- 1. What is your proposed **system concept**?  
What alternative concepts did you consider and why did you select the one you proposed?
- 2. How will your proposed concept operate within the **larger context** to achieve its **intended purpose**?
- 3. What are the **key specifications** that will drive your system's design and development?

# System Concepts

1. What is your **proposed system concept**?

What **alternative system concepts** did you consider and **why** did you select the one you proposed?

- **What criteria were used** to compare the alternatives?
- How well did **each concept satisfy** each of the **selection criteria**?
- On what **basis** was the preferred concept selected?

# Creating System Concepts - A Legend?

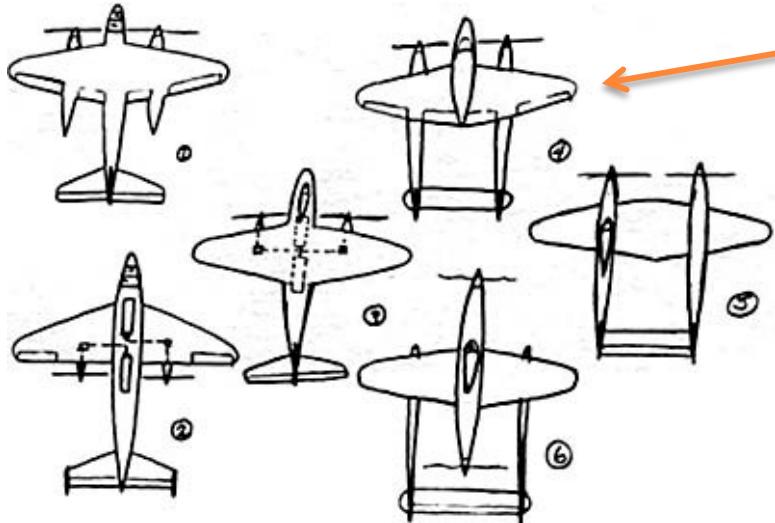
- Question on a physics exam at the University of Copenhagen:
  - "Describe how to use a barometer to determine the height of a skyscraper."
- Proposed Solution: "Lower it from the roof on a string and measure the length of the string."
- Physics-based solutions:
  - "Drop it from the roof and measure the elapsed time."
  - "Measure its shadow and use proportions."
  - "Turn it into a pendulum and measure the period."
  - "Measure its length and use it as a ruler."
  - "Measure the pressure differential." = *the “boring” solution*
- Better yet? - "Offer it to the super in exchange for the answer."

*The student was Niels Bohr, the only Dane ever to win the Nobel Prize for Physics*

# How do you quickly develop a High-altitude interceptor for the Army?

Problem: Army needs

- High-altitude interceptor
- Max speed > 360 mph.
- Climb to 20,000 ft. in < six min.
- Must use Allison V-1710 engines with superchargers



*Kelly Johnson's drawings of concepts for this system*

*This concept became the famous Lockheed P-38 Lightning.  
Over 10,000 produced from 1939-1945*

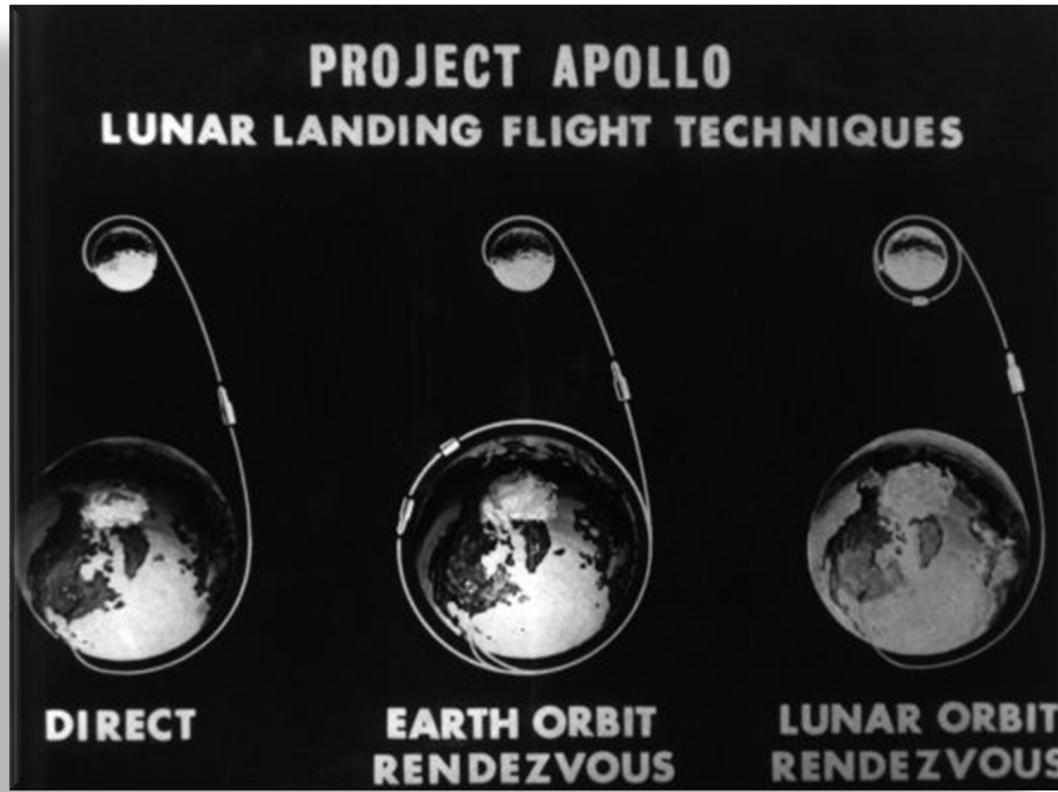


10803 AC

12

**Leader of famous “Skunk Works”**

# How do you get a Human to the Moon, and back?



*Originally rejected by NASA as too risky*

## Three Basic Concepts:

- Direct Ascent
- Earth Orbit Rendezvous (EOR)
- Lunar Orbit Rendezvous (LOR)



Dr. John C. Houbolt,  
NASA Rendezvous Committee

For the full story on the concepts generated for getting a man on the moon, see the excellent NASA article at:  
<http://www.nasa.gov/centers/langley/news/factsheets/Rendezvous.html>

# Generate Alternatives

After the problem has been framed, ask:  
**“How can we obtain the desired outcome?”**

- Challenge constraints – look at the problem from new angles
- Be creative, let the process diverge
- Gather information, if necessary
- Withhold judgment until the evaluation phase

# Issues in Concept Selection

- multiple criteria – how to deal with them ?
- what if there are ties between alternatives?
- group decision making versus individual decision making?
  - relates to stakeholder analysis
    - uncertainty
    - right criteria?
    - right valuation?
  - are the best alternatives represented?

# 'Simple' Methods of Concept Selection

- Pugh Matrix
  - Uses +, 0, - to score alternatives relative to a datum
    - Named after Stuart Pugh
  - The Pugh matrix helps determine which items or potential solutions are more important or 'better' than others. It is employed after capturing voice of the customer (VOC).

# Pugh Matrix Steps

## 1. Choose or develop the criteria for comparison.

- Based on a set of system requirements and goals.

## 2. Select the Alternatives to be compared.

- The alternatives are developed during concept generation.
- All concepts should be compared at the same level of abstraction and in similar language.

## 3. Generate Scores.

- Use a concept as **datum**, with all the other being compared to it
- Evaluate each alternative as being better (+), the same (S, o), or worse (-) relative to the datum.

## 4. Compute the total score

- Three scores will be generated, the number of (+), (-), (o)
- The overall total is the number of (+) minus the number of (-)
- The totals should not be treated as absolute in the decision making process but as guidance only.

## 5. Variations on scoring

- A number of variations on scoring Pugh's method exist.
- For example a seven level scale could be used for a finer scoring system where:
  - +3 meets criterion extremely better than datum

# Pugh Example (Simple)

## Evaluation Matrix

| Concept \ Criteria | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|--------------------|---|---|---|---|---|---|---|---|---|----|----|
| A                  | + | - | + | - | + | - | D | - | + | +  | +  |
| B                  | + | S | + | S | - | - |   | + | - | +  | -  |
| C                  | - | + | - | - | S | S | A | + | S | -  | -  |
| D                  | - | + | + | - | S | + |   | S | - | -  | S  |
| E                  | + | - | + | - | S | + | T | S | + | +  | +  |
| F                  | - | - | S | + | + | - |   | + | - | +  | S  |
| $\Sigma^+$         | 3 | 2 | 4 | 1 | 2 | 2 | U | 3 | 2 | 4  | 2  |
| $\Sigma^-$         | 3 | 3 | 1 | 4 | 1 | 3 |   | 1 | 3 | 2  | 2  |
| $\Sigma S$         | 0 | 1 | 1 | 1 | 3 | 1 | M |   | 1 | 0  | 2  |

# Pugh Matrix is useful for evaluating system concepts

|          | Conceptual System Design Alternatives |   |   |   |   |   |
|----------|---------------------------------------|---|---|---|---|---|
| Criteria | 1                                     | 2 | 3 | 4 | 5 | 6 |
| A        | +                                     | - | + | - | + | - |
| B        | +                                     | S | + | S | - | - |
| C        | -                                     | + | - | - | S | S |
| D        | -                                     | + | + | - | S | + |
| E        | +                                     | - | + | - | S | + |
| $\sum +$ | 3                                     | 2 | 4 | 0 | 1 | 2 |
| $\sum -$ | 2                                     | 2 | 1 | 4 | 1 | 2 |
| $\sum S$ | 0                                     | 1 | 0 | 1 | 3 | 1 |

Ref: Pugh, Stuart, Total Design: Integrated Methods for Successful Product Engineering, Addison-Wesley, 1991.

"+" to denote better than required performance

"-" to denote lower than required performance

"S" to denote performance on par with the requirement.

# Mars Colony Power Concept Alternatives

| Criteria:                       | Wind Power | Nuclear Power | Solar Power |
|---------------------------------|------------|---------------|-------------|
| Consistency of Power Generation | -          | +             | -           |
| Ease of Transport to Mars       | -          | -             | +           |
| Ease of Setup on Mars           | S          | -             | +           |
| Ease of Maintenance on Mars     | S          | -             | S           |
| Safety                          | +          | -             | +           |
| <b>Sum of +</b>                 | 1          | 1             | 3           |
| <b>Sum of S</b>                 | 2          | 0             | 1           |
| <b>Sum of -</b>                 | 2          | 4             | 1           |

# Partner Exercise (5 min)

- What do you see as the main advantages and potential disadvantages or pitfalls of the Pugh Matrix method?
  - Turn to your partner
  - Discuss for 5 minutes
  - Share

# What is Pugh Matrix for?

The Pugh matrix is for

- Structuring and representing an evaluation procedure
  - Serves as common visual
  - Provides a discipline
  - Helps break down self-sealing behavior
  - Encourages real teamwork
- Convergence
  - Eliminates weaker ideas
  - Retains a set of strong concepts
- Divergence
  - Helps to identify opportunities for combination

The Pugh matrix is NOT for

- Automatic decision making
  - “the scores or numbers ... are for guidance only and must not be summed algebraically.”
  - “it avoids the rigidity and false confidence of rating/weighting matrices”
- Completely controlling the process
  - “... stimulates creative unconstrained thinking due to its lack of rigorous structure”
- Trade studies
  - More on this today

Pugh, Stuart, 1991, *Total Design*, Addison-Wesley, New York.

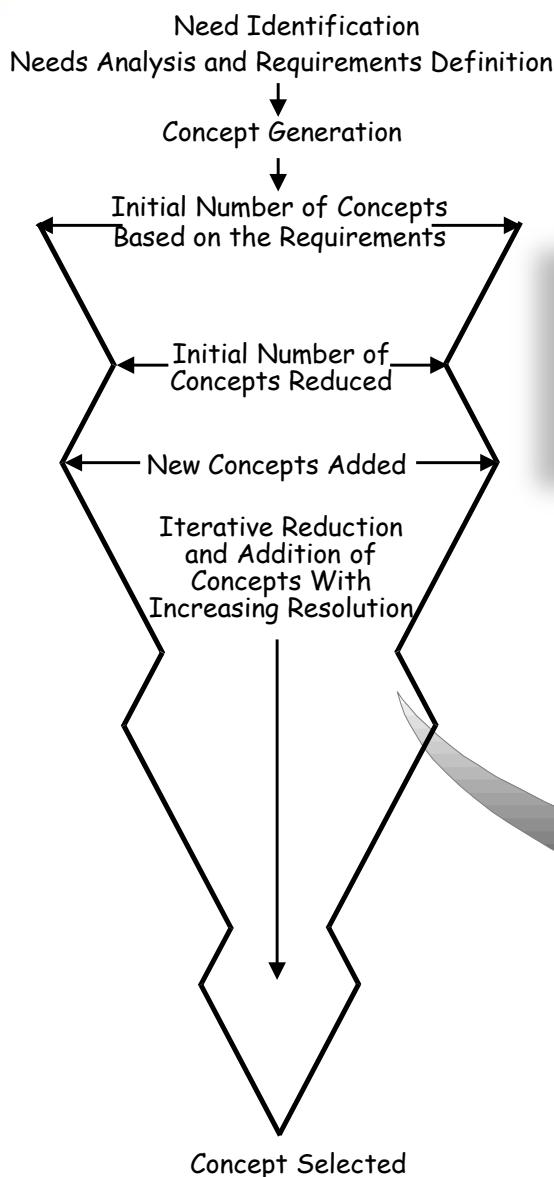
# Challenges

- “people who have a lot of experience ... exhibit an impatience ‘to get on with it’ and may consider that the procedure holds them back...”
- “strong willed individuals who have a lot of experience and whose initial concepts have not emerged in the final selection ... commence a defense based on emotion, experience, and bluster...”

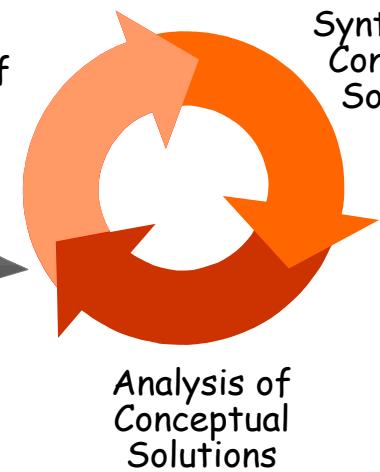
# Role of the Facilitator

- Controls the flow / pace of the session
- Records the results (creates the matrix)
- Maintains a tight discipline on the participants
  - Comparison to the datum concept
  - Preventing tangents
  - Encourages clarification of criteria
  - Encourages clarification of concepts
- Seeks opportunities for divergence (hybrids)

# System concept design is itself an iterative process.



*Great!*  
*How else could we do it?*



Ref: Pugh, Stuart, Total Design: Integrated Methods for Successful Product Engineering, Addison-Wesley, 1991.

# Evaluating and Selecting Concepts



***“There is no sense in being exact about something if you don’t even know what you are talking about.”***

***(John von Neumann, 1950)***

# System Concepts

1. What is your proposed **system concept**?

What **alternative concepts** did you consider and why was did you select the one you proposed?

- Assessment Criteria:

*Broad range of concepts defined and systematically analyzed against criteria linked to key stakeholder needs.*

# Developing a Solution

## *Key Questions:*

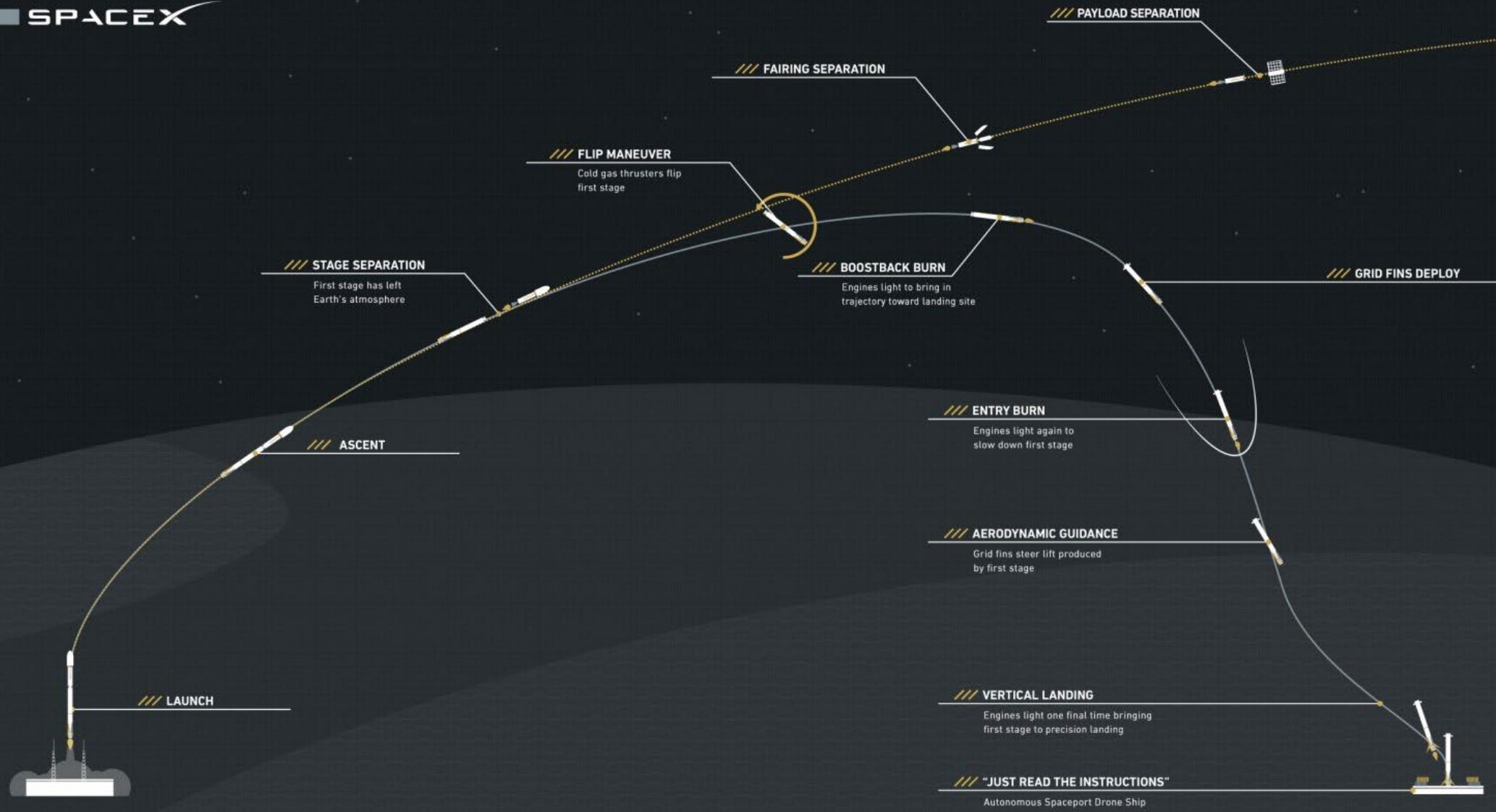
1. What is your proposed **system concept**?  
What alternative concepts did you consider and why did you select the one you proposed?
  
- 2. How will your proposed concept operate within the **larger context** to achieve its **intended purpose**?
  
3. What are the **key specifications** that will drive your system's design and development?

# The System and It's Context

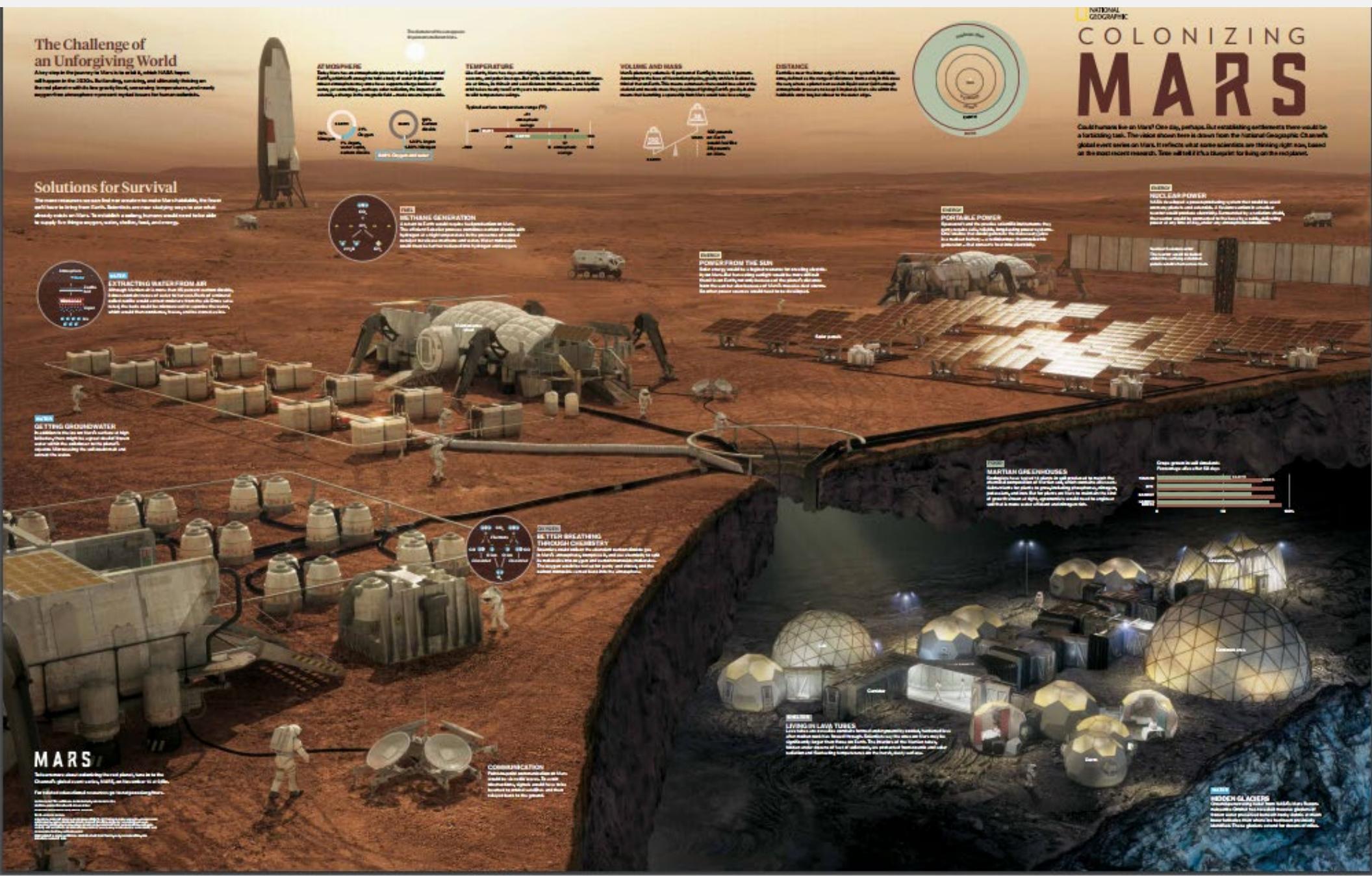
2. How will your **proposed concept operate within the larger context** to achieve its intended purpose?
  - What are the **key operational** and other scenarios in which the system will play an important role?
  - Which **users and external systems** will the system interact with?
  - What **sequences describe the interactions** between the system and its external systems for each of the key scenarios?

# A high level Operational Concept describes how the system will work within the larger context

SPACEX

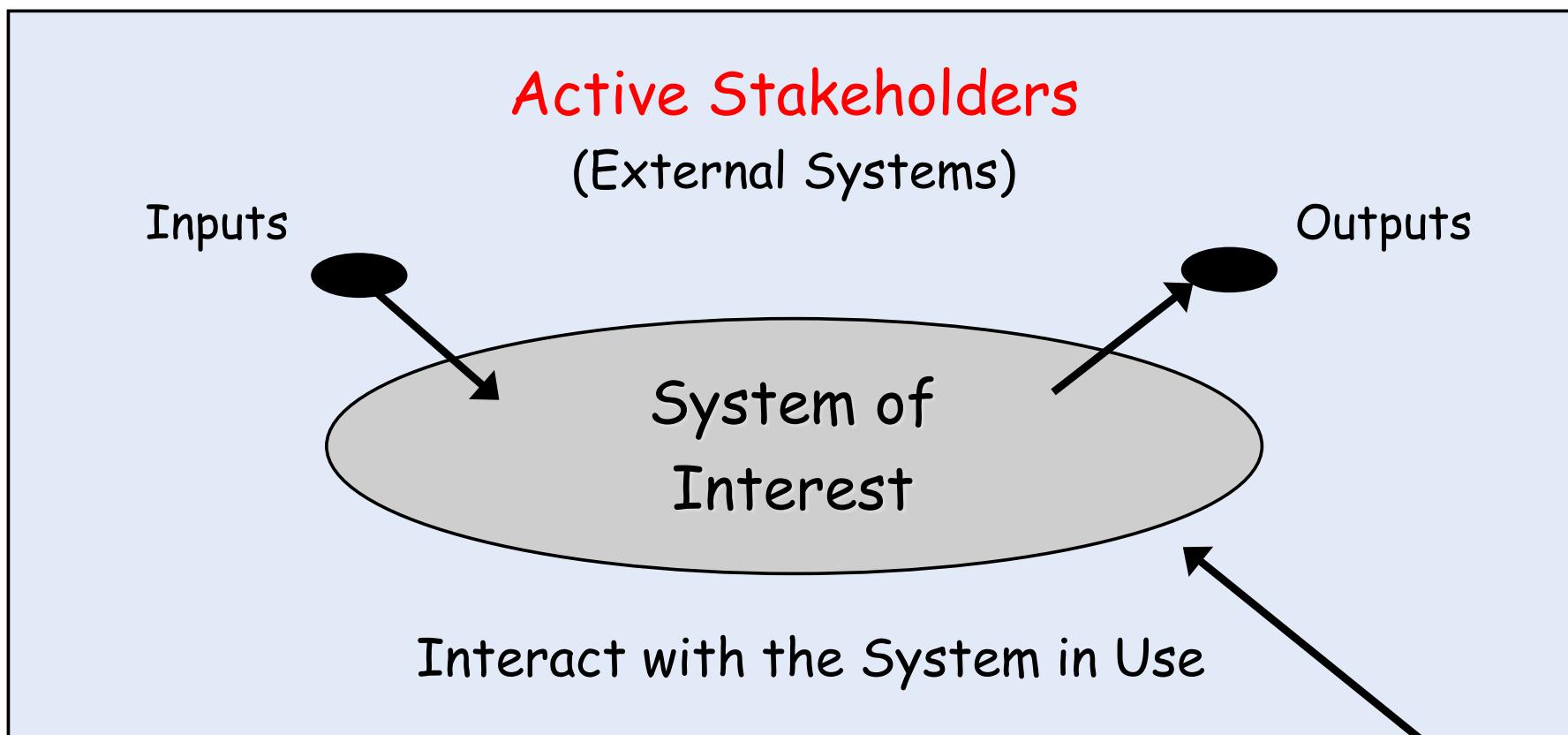


# A high level Operational Concept describes how the system will work within the larger context



A Context Diagram identifies the system boundary and active stakeholder interactions.

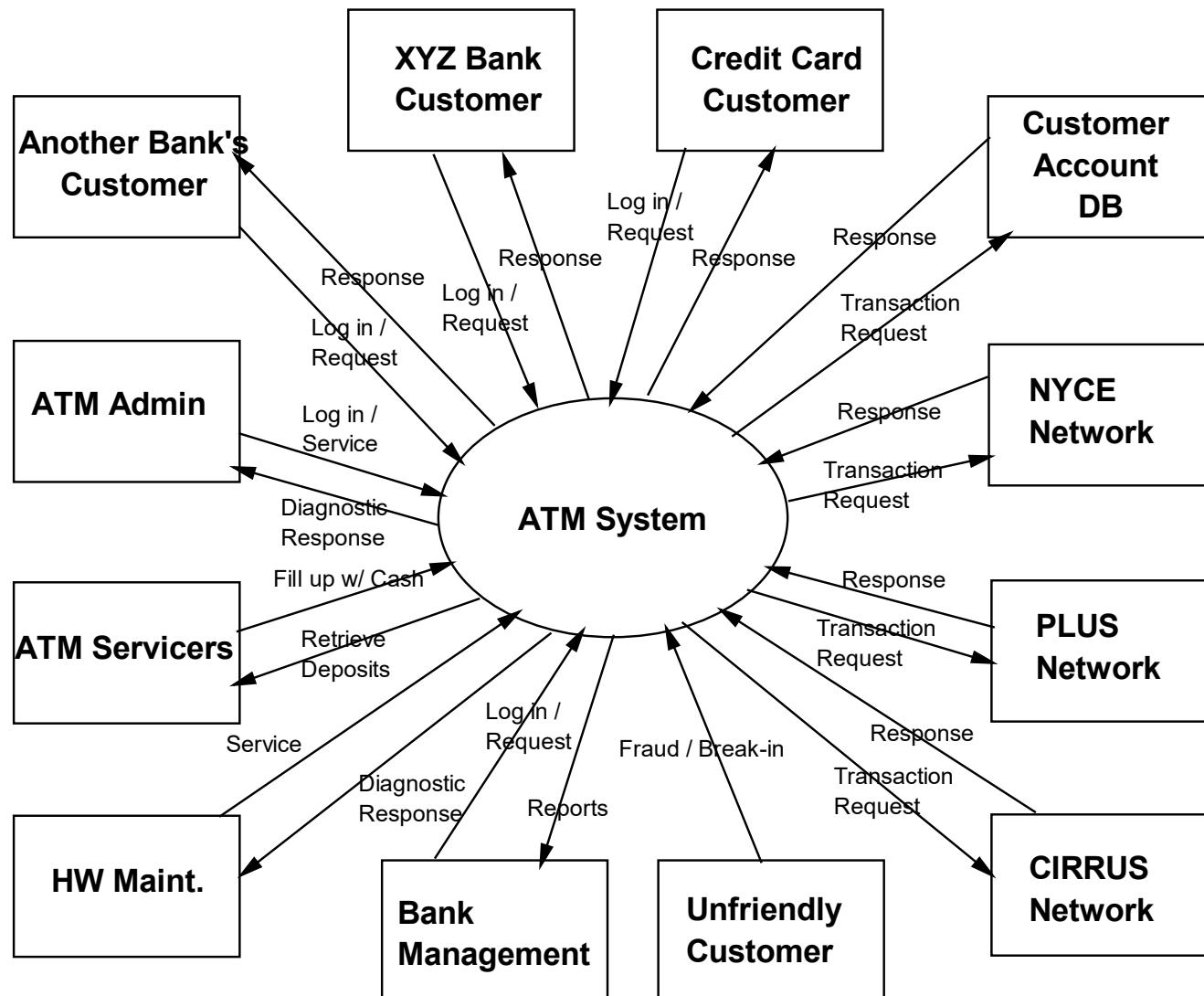
### Passive Stakeholders



Provide inputs to, but do not  
interact with, the System

Inputs

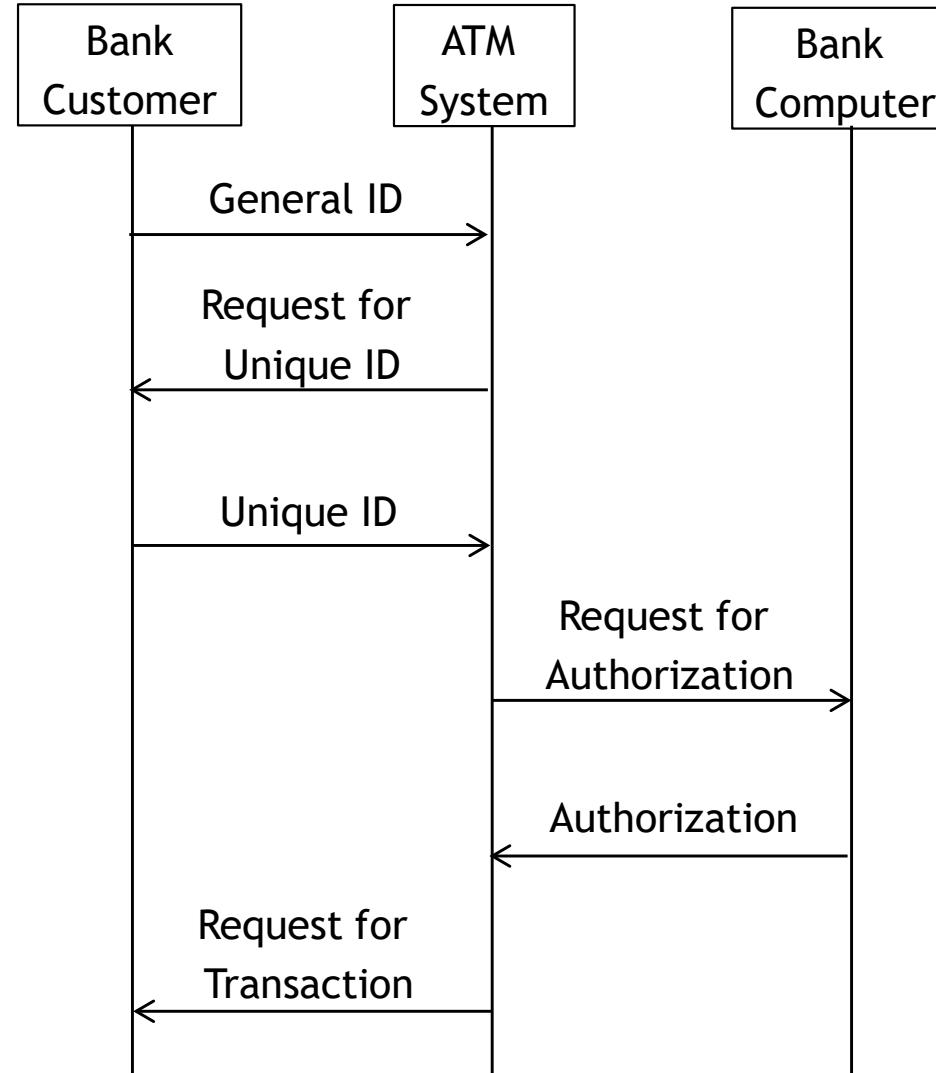
# Context Diagram for an *Automatic Teller Machine*



## Sequence Diagrams describe the interactions between the system and its external systems

- Represent the System of Interest as a single timeline
  - Provides a **black-box view** of the system
- Label the arrows with **nouns**, not verbs
  - Focuses on the inputs and outputs, not the act of exchanging them
- Restrict the diagram to first-order interactions between the system and its external systems
  - **Focuses on the boundary** between the system to be designed and its context

# Sequence Diagram for an ATM



# Sequence Diagram for Mars Transport System

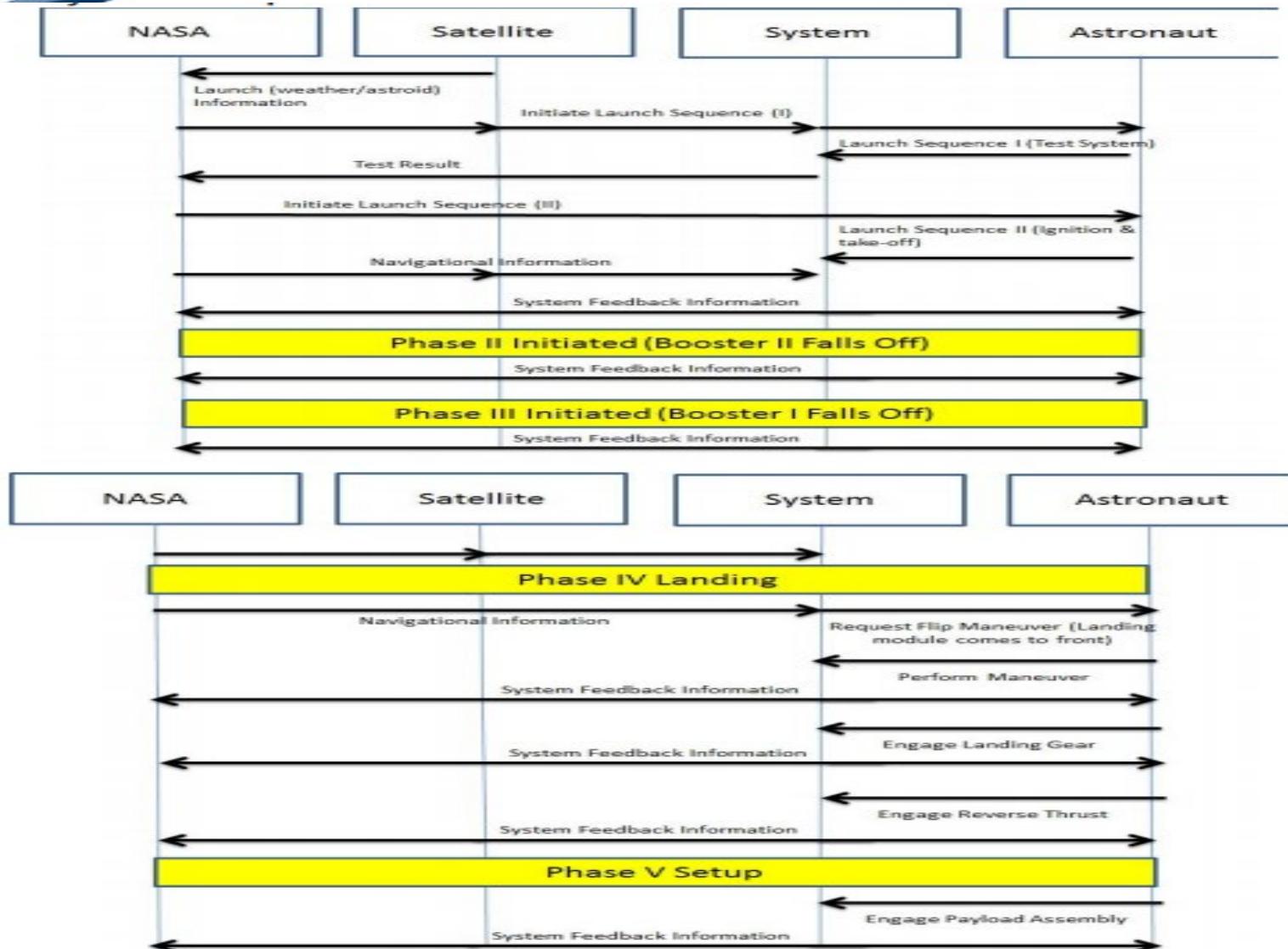


Figure 2: Sequence diagram

# The System and Its Context

2. How will your proposed concept operate within the **larger context** to achieve its **intended purpose**?

- Assessment Criteria:

- *The external systems with which the system will interact have been identified*
- *the system boundary has been clearly defined*
- *the interactions between the system and the external systems have been specified from a black box perspective*

# Developing a Solution

## *Key Questions:*

1. What is your proposed **system concept**?  
What alternative concepts did you consider and why did you select the one you proposed?
  
2. How will your proposed concept operate within the **larger context** to achieve its **intended purpose**?
  
3. What are the **key specifications** that will drive your system's design and development?  


# Translating Needs into Specifications

3. What are the **key specifications** that will drive the system's design and development?
  - To what **inputs** must the system respond and who or what will provide them?
  - What **outputs** must the system provide and to whom?
  - What **performance requirements** must the system's inputs and outputs satisfy?
  - What additional **quality attributes** must the system achieve?

# Input/Output Matrix

- **Intended inputs**
  - Resources necessary to produce or influence the production of desired outputs during each phase of the system life cycle
- **Unintended inputs**
  - Characteristics, often beyond our control, that arise from the environment in which the systems operates.
- **Desired outputs**
  - Required outputs over all system life cycle phases. Upon development and deployment, the system should demonstrate the realization of these outputs
- **Undesired outputs**
  - Undesired characteristics which, if anticipated in time, can be minimized

# Input/Output Matrix Example

|               | Inputs  |                                   | Outputs                               |  |
|---------------|---|-----------------------------------|---------------------------------------|--|
|               | Intended                                      | Unintended                        | Desired                               | Undesired                                    |
| Signal        | Pulse shape, data rate, signal to noise ratio | Electrical noise                  | Data rate, accuracy                   | Error rate, false alarm rate                 |
| Electrical    | Nominal voltage                               | Surge voltages and timing         | Voltage, current, frequency stability | Electromagnetic interference, electric shock |
| Mechanical    | Activation force                              | Shock and vibration               | Movement, resistance                  | Acoustic noise levels                        |
| Environmental | Normal temperature range                      | Temperature and humidity extremes | Particle density, air flow            | Heat, effluents                              |

# Input/Output Performance

- Quality
  - Accuracy, precision, error rate, security...
- Quantity
  - Size, throughput, number, intensity...
- Timing
  - Frequency, response time, availability...

# Being Clear About Requirements



*Translated into*

## Customer Needs

- Language of the Stakeholders
- What the customer wants
- May be *subjective*

## System Specifications

- Language of the Engineers
- What the engineers build
- Must be *objective*

# Quality Function Deployment (QFD)

|                   |  | Relevant Design Parameters |  |  |  |  |  |
|-------------------|--|----------------------------|--|--|--|--|--|
|                   |  |                            |  |  |  |  |  |
| Customer<br>Needs |  |                            |  |  |  |  |  |
|                   |  |                            |  |  |  |  |  |
|                   |  |                            |  |  |  |  |  |
|                   |  |                            |  |  |  |  |  |
|                   |  |                            |  |  |  |  |  |
|                   |  |                            |  |  |  |  |  |

**Relationships**  
*(Strong, Medium, Weak)*

# Quality Function Deployment

## Mobile Phone Example

| Customer<br>Needs        | Relevant Design Parameters |       |        |     |           |           |
|--------------------------|----------------------------|-------|--------|-----|-----------|-----------|
|                          | Length                     | Width | Weight | COG | Roughness | Curvature |
| “Feels Good in the Hand” | S                          | S     | S      | W   | S         | S         |
| “Phone operates all day” | W                          | W     | W      | M   |           |           |
| “Easy to hear”           | M                          | M     | W      | M   |           | W         |
|                          |                            |       |        |     |           |           |

# Rules for Writing Requirements

- Terminology
  - Use “**shall**” to indicate the limiting nature of a requirement
  - Statements of fact use “will”
  - Goals use “should”
  - This scheme should be agreed to by all key stakeholders at the outset!
- Grammatical Construction
  - A **subject** (The XYZ System...)
  - The word **“shall”**
  - A **relation statement** (e.g., less than or equal to)
  - The **minimum acceptable threshold with units**

# Rules for Writing Requirements

- **Use Proper Grammar**

- “The system shall stop the flow of liquid hydrogen in 0.5 seconds or less. The liquid stopping time is measured from the time the control signal for stopping is received until the flow through reaches zero.”

- **Avoid Compound Predicates**

- “The system shall fit ..., weigh ..., cost ...” (this causes traceability problems)

- **Avoid Negative Predicates**

- “The system shall not ...” (attempt to turn this into a positive statement of what the system shall do).

- **Avoid Ambiguous Terms**

- Verbs: “optimize,” “maximize,” and “minimize”
  - Adjectives: “adaptable,” “adequate,” “easy,” “flexible,” “rapid,” “robust,” “sufficient,” “supportable,” and “user-friendly”

# Characteristics of a “Sound” Requirement

| Attributes of individual requirements |  |
|---------------------------------------|--|
| Unambiguous                           | Every requirement should have only one interpretation  |
| Understandable                        | The interpretation of the requirement should be clear  |
| Correct                               | Consistent with what the system is required to do  |
| Concise                               | No unnecessary information is included in the requirement  |
| Traced                                | Each requirement is traced to some document or statement from the stakeholders - the rationale should be captured and verifiable |
| Traceable                             | Each derived requirement must be traceable to a “system” requirement via an explicit tracing scheme                              |
| Design independent                    | Requirements should not impose an implementation approach or solution or technology  |
| Verifiable                            | A finite, cost-effective approach has been defined to ensure that the requirement has been attained                              |

# Characteristics of a “Sound Set” of Requirements

| Attributes of a set of requirements |   |
|-------------------------------------|---|
| Unique                              | Requirements are not overlapping or redundant   |
| Complete                            | (a) Everything that the system is required to do throughout its life cycle is addressed; (b) responses to all possible (realizable) inputs throughout the system’s life cycle are defined; (c) the document is clear and self contained; (d) there are no “tbd” or “to be reviewed” statements. |
| Consistent                          | (a) Internal - no two subsets of requirements conflict; (b) external - no subset of the requirements conflict with external documents or stakeholders from whom these requirements have been derived.   |
| Comparable                          | The relative priorities of the requirements are included.   |
| Modifiable                          | Changes to the requirements can be made with relative ease, and with a method to ensure consistency   |
| Attainable                          | The solution space should not be a “null” set   |

## Voice of the Customer Potential Pitfalls

- Customers frequently express their needs in terms of "how" the need can be satisfied and not in terms of "what" the need is
  - It is important to ask "why" until they truly understand what their root need is
  - But, if the customer *really* wants the 'how', then capture that as the requirement

“MTBF = 10 Years/Failure”

# Translating Needs into Specifications

3. What are the **key specifications** that will drive the system's design and development?

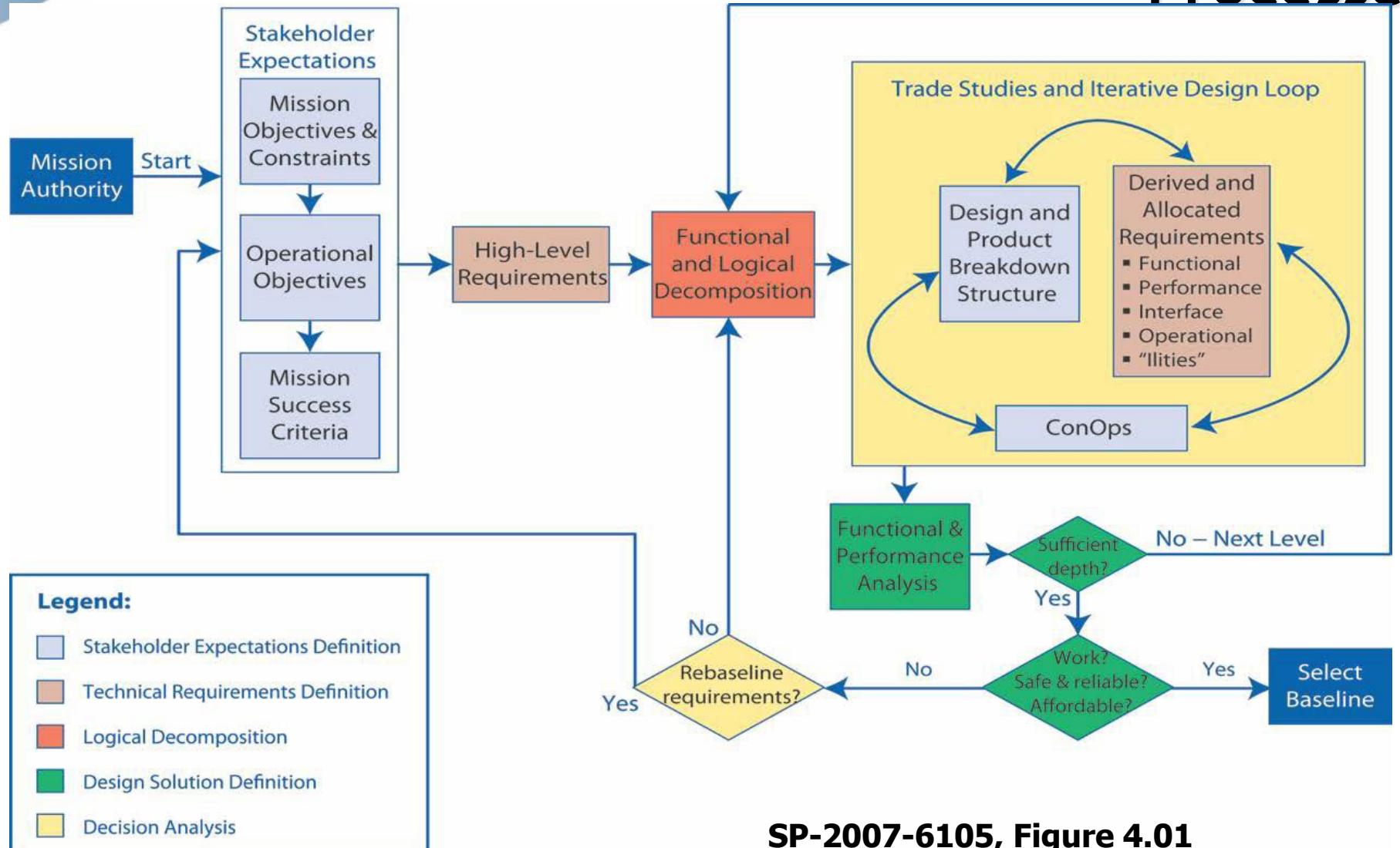
- Assessment Criteria:*

## *System requirements*

- a) have been derived from and are linked to the stakeholder requirements*
- b) describe what the system shall do, not how, and*
- c) are verifiable and properly written*



# Interrelationships Among the System Design Processes



SP-2007-6105, Figure 4.01

This graph is in the public domain.

# Creating an Operational Concept Key Questions:

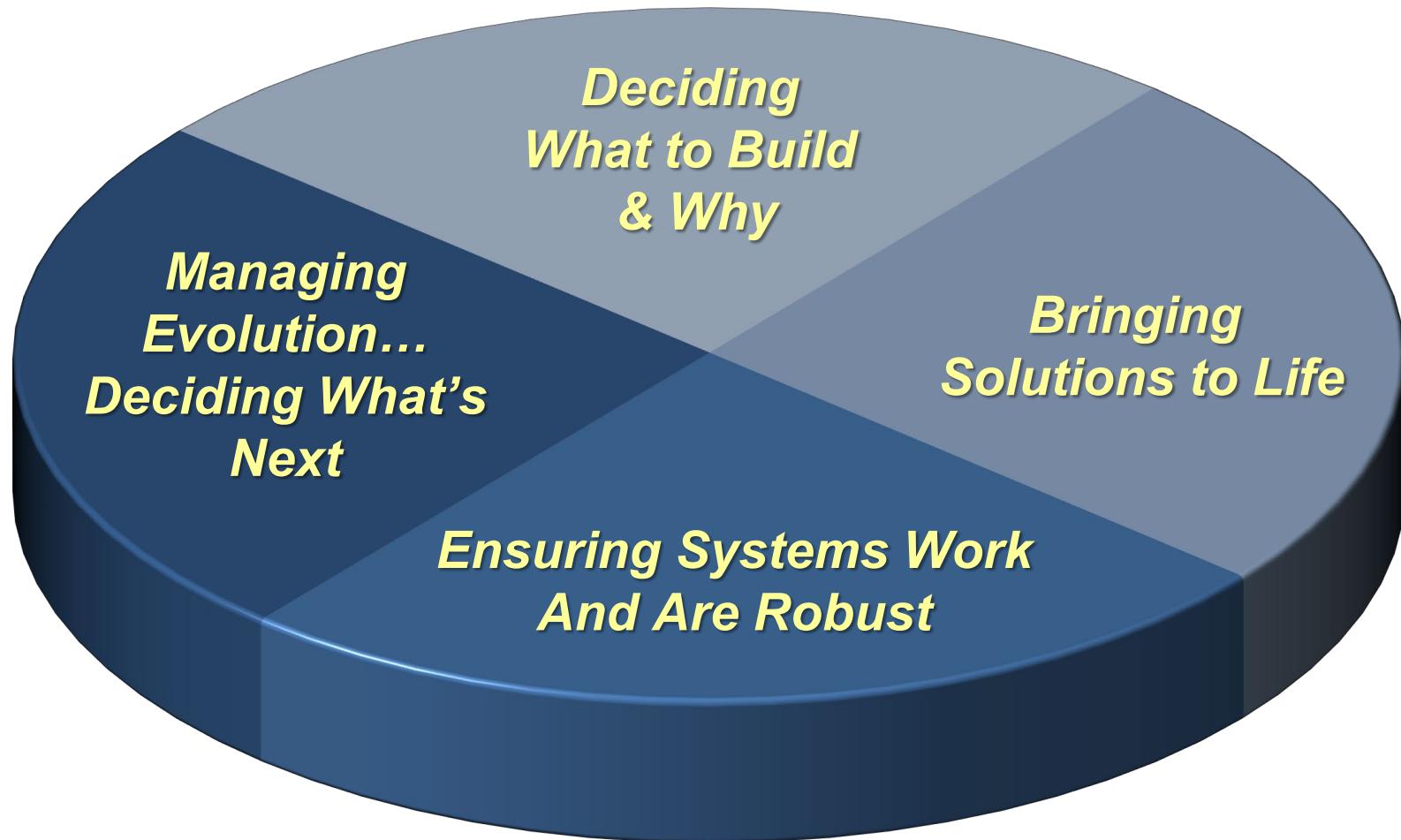
1. What is your proposed system concept? What alternative concepts did you consider and why did you choose the one you proposed?
  - *Assessment Criteria: Broad range of concepts defined and systematically analyzed against criteria linked to key stakeholder needs.*
2. How will your proposed concept operate within the larger context to achieve its intended purpose?
  - *Assessment Criteria: The external systems with which the system will interact have been identified, the system boundary has been clearly defined, and the interactions between the system and the external systems have been specified from a black box perspective.*
3. What are the key specifications that will drive the system's design and development?
  - *Assessment Criteria: System requirements a) have been derived from and are linked to the stakeholder requirements, b) describe what the system shall do but not how, and c) are verifiable and properly written.*

## In class Assignment

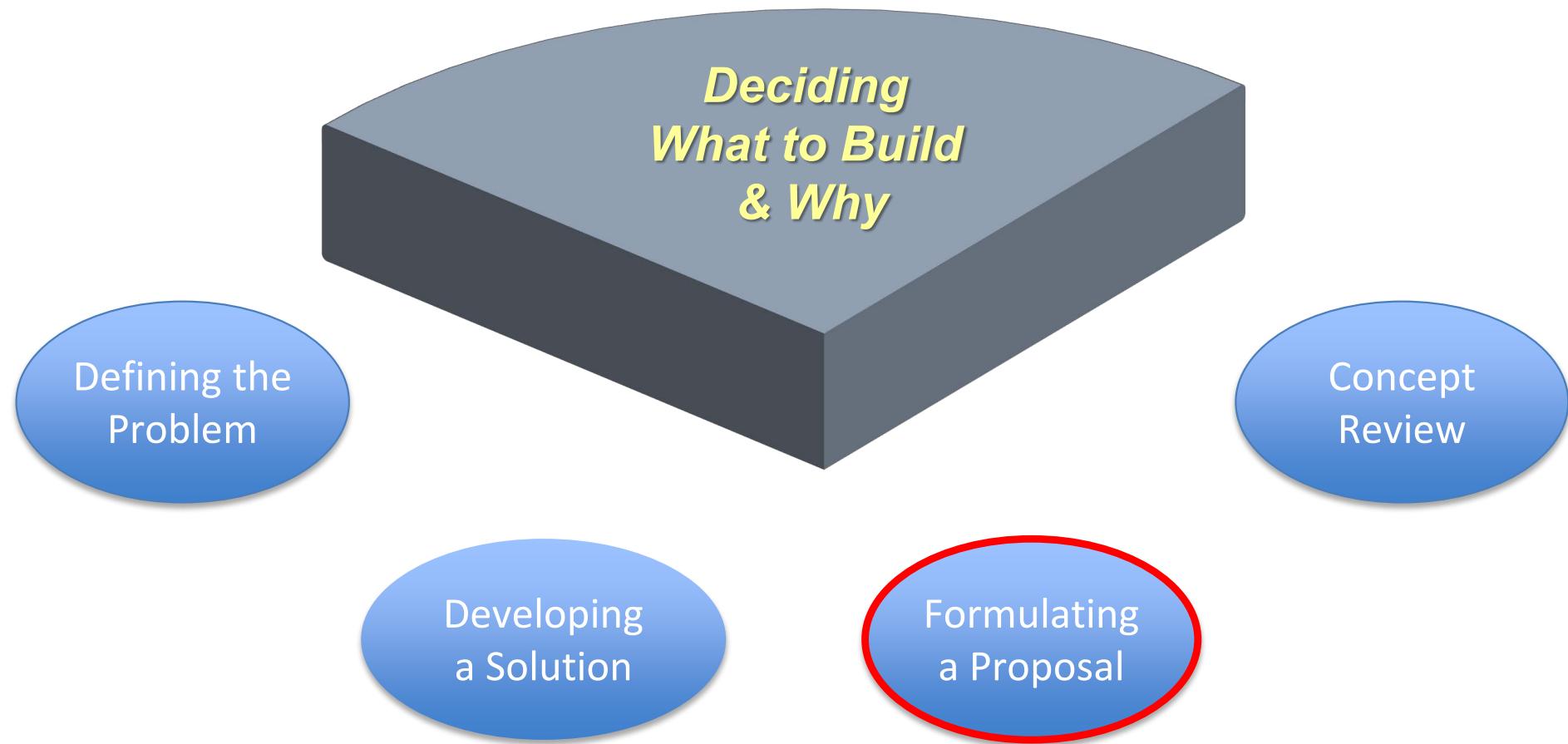
- Pugh Matrix Case Study (30 minutes)
- Submit to Canvas
- Concept Review in 2 weeks
- ( will upload a sample this week)

# Formulating a Proposal

# Course Design



# Course Design



- Astro Teller
- <https://www.youtube.com/watch?v=FYU-N2RWfso>
- Starship ( Artificial Gravity)
  - <https://www.youtube.com/watch?v=VyWE2bA4h6A>



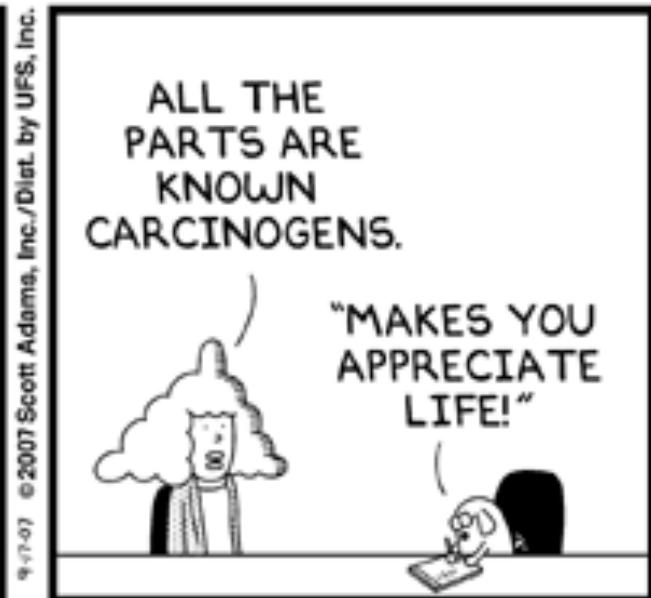
**What is a proposal?**

**Why would I need one?**

**Who would I present it to?**

# Why Proposal Writing is Different

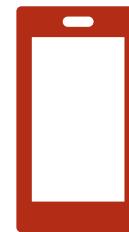
- Most engineers are trained to write technical documents
- Proposals are technical marketing documents



# Elements of a Good Proposal



Demonstrates your understanding of the customer's problem or the market opportunity



Describes your proposed solution



Identifies the unique benefits your solution will provide to the customer

# Tips for Proposal Writing

- Write/speak from the customer's or stakeholder's point of view
- Focus on what makes your proposal different
- Every assertion must be supported with evidence
- Every comparative must include what you are comparing to
- Superlatives must be avoided at all costs
- Tell a consistent story:

**Feature → Benefit → Proof**



***“I made this letter longer than usual because I lack the time to make it short.”***

*Blaise Pascal, 1657*

# Application to Proposal “Win Themes”

- A Win Theme is a high-level feature of your proposed solution that:
  - Provides extraordinary benefit to the customer and
  - Differentiates your solution from that of your competitors

- Win themes are the messengers that promote the value of your solution. Because win themes speak to the value of your solution, they turn compliant proposals into compelling ones. They make it easy for evaluators to express why they prefer your company and your solution. Win themes benefit the proposal team, too

# Win Themes

- Are few in number; 3-5 is reasonable
- Answer the question, “Why you?”
- Are the bullets you hope to see on your customer’s source selection rationale slide
- Should be woven throughout your entire proposal
- Must be supported with facts and data



Customer -focused messaging focused messaging



Limited,high-level ideas and concepts



Melding of the buyers priorities and the sellers  
differentiators/strengths



A compelling story

- Why do you think Win themes are important in writing proposals?
- Discussion

# Why are win themes important?

Help

Help organize your proposal strategy

Grab

Grab the evaluator's attention and differentiate your proposal

Demonstrate

Demonstrate your understanding and frame your solution for the buyer

Give

Give evaluators a reason to select you, even before they read the proposal

# Are These Good Win Themes?

- Our solution:
  - Satisfies all customer requirements
  - Will be developed by the best team
  - Is low cost



# Themes must support the strategy:

|   |   |
|---|---|
| <b>Win Strategy</b><br>(What and How sentences) | <ul style="list-style-type: none"><li>• Resilience: demonstrate compliance, quote service record</li><li>• Ghost competitor 1's weakness in space usage.<br/>Show that our solution needs 30% less space<br/>Stress extra cost and delivery risk with larger kit</li><li>• Experience: Show compliance, provide references show experience of team</li><li>• TCO: Neutralise Competitor 2 strength on price by stressing early benefits and savings over the full cycle</li><li>• Ghost Competitor 2's delivery performance by showing that our systems are shipping and in service now</li></ul> |
| <b>Win Themes</b><br>(Key messages)             | <p><b>Major:</b></p> <ul style="list-style-type: none"><li>• Compact cost effective solution</li><li>• Early delivery, early benefits</li><li>• Savings available this year</li></ul> <p><b>Minor:</b></p> <ul style="list-style-type: none"><li>• Proven resilient solution</li><li>• Experienced team</li></ul>   |

# STORYBOARDING

# What is a Storyboard

Storyboards have long been a part of our standard proposal best practices. We all know that the proposal giants include storyboarding as an integral part of the proposal development process, but where did this concept of storyboards originate?

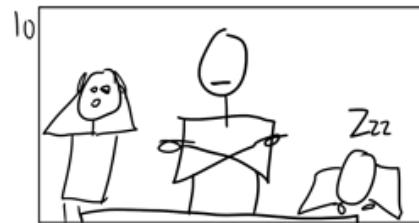


# Origin of the Storyboard

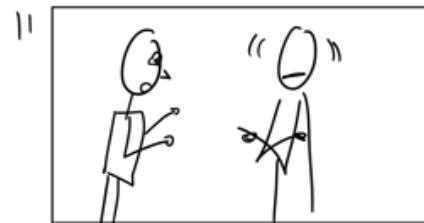
- The story board form widely known today was developed at the Walt Disney studio during the early 1930s. Diane Disney explained that the first complete storyboards were created for the 1933 Disney short *Three Little Pigs*.
- According to Christopher Finch in *The Art of Walt Disney*, Disney credited animator Webb Smith with creating the idea of drawing scenes on separate sheets of paper and pinning them up on a bulletin board to tell a story in sequence, thus creating the first storyboard.
- Storyboarding became popular in live-action film production during the early 1940s, and grew into a standard medium for pre-visualization of films.

# Storyboard Example

"CS2C: Fun with Storyboards" Expansion by Kenneth Chan. Insert frames 10-18 between frames 8 & 9.



Voiceover: The first stage is Denial. Student crosses arms stubbornly as classmate reacts in pain.



Classmate asks, "So, are you going to do it?" Vigorous head shaking. "NO WAY. My board's FINE."



Awkward pause as student gets over denial stage. Slowly accelerate zoom in on face in preparation for...



VO: The next stage is ANGER. "Why me?!" Stank face. Angry driving rock and roll music.



Daydream firing missiles and Care Bears Stare at evil instructor. Pew pew pew! Framed by blur filter.



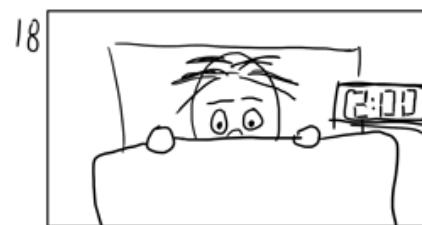
Cut back to student, huffing in exhaustion, but calming down. VO: "Theres gotta be a way out..."



The next stage is bargaining. Offering cookies and money to unimpressed instructor.



Depression sets in as student walks back to dark room in a slump. Crawls into bed and puts cover over head.



Overheat shot. Can't sleep. Slowly starts to uncover face. Eyes widen with inspiration and acceptance.

# Application of Storyboards to Proposals



Used to sketch out your proposal before committing the time and effort required to prepare a complete draft



Encourages graphic-centric rather than a text-centric approach



Allows the proposal team and the reviewers to see the “big picture” without getting lost in the details

# Why storyboarding is important?

- Storyboarding is essentially structured prewriting. The importance of prewriting has been studied and established for many fields, including business, which routinely uses such techniques as brainstorming, listing, and discussion to develop projects and departmental evaluations.
- Prewriting increases efficiency by helping writers understand the task, brainstorm, and then map and/or plan their writing before beginning a first draft.

# Storyboard

- ***Understand the task.*** One key benefit of storyboarding is that it helps authors understand the writing task. Before authors dive into writing their assigned section(s), they should review the limitations of their assigned task. They should make sure they understand the page limitations and the relevant RFP sections to address. During storyboarding, authors should also review and understand the proposal schedule and any major deadlines.

# Storyboard

- **Brainstorm.** Another key benefit of storyboarding is the opportunity to brainstorm. This brainstorming provides an opportunity for authors to analyze the customer, the competition, and the proposal strategy. When brainstorming, writers simply throw ideas out in whichever order and form they come out. The idea behind this is that once the writers get everything out their heads, they can more easily organize and structure those ideas.

# Storyboard

- **Analyze the customer.** One important outcome of storyboarding is gaining an understanding of the customer. Authors should work with the capture team to gain an understanding of who will likely read each section of the proposal. As a group, the proposal team should consider the following questions:
  - Who is the customer?
  - What is important to this customer?
  - What are the customer's major hot buttons?
  - What is their mission? What is the end-goal of the contract?
  - What are current issues they are facing?
  - What are some things they really like or appreciate with their current contractors?
  - What is their biggest fear with the upcoming contract?
  - What problems do they anticipate?
  - Is a technical person likely reading the section or someone else?

# Storyboard

- ***Consider the competition.*** Another important outcome of storyboarding is a greater understanding of the competition. Analyzing the major competitors will help the team identify areas to highlight team strengths that will “ghost” the weaknesses of the competition. This will also help the team to identify areas where the competition may have a leg up so that the team can proactively come up with ways to compensate. As part of the brainstorming sessions, the team should consider the following:
  - What are the likely strategies of the competitors?
  - What are their strengths and weaknesses?
  - How can we mitigate those strengths and highlight the weaknesses in our proposal?

# Storyboard

- ***Consider your position.*** Another key outcome of storyboarding is a stronger understanding of your team's position to win. This is especially important when consultant writers are involved. As part of the brainstorming sessions, the team should consider the following:
  - What are the team's strengths and weaknesses?
  - How can we mitigate those weaknesses and highlight the strengths in our proposal?
  - Are there any past performances or proof points to cite?

# Story Board

- ***Define the section strategy.*** Another key storyboarding outcome is defining the section strategy. Authors should consider the following questions:
  - What is your overall solution or strategy?
  - What are the major features of your solution that solve the customer's problems?
  - What are the benefits the customer will receive?
- ***Identify section themes.*** One of the most important outcomes of storyboarding are the section themes. Section themes should resonate with your overall win themes; however, section themes are specific to each section. Section themes should highlight a key feature of the section solution and the benefit that the customer receives from that feature.

## Theme Sentence

“To convince someone of something you must first state your case and then prove it. You cannot state your case and fail to prove it, nor can you prove it without first having stated it.” –

**Aristotle**

# Elements of a Proposal Storyboard

- |   |  |
|---|--|
| <ul style="list-style-type: none"><li>• Section Identifier (e.g. number, title)</li><li>• <i>Theme sentence (usually in boldface italics)</i></li><li>• Summary of Key Points (in bullet or outline form)</li></ul> | <ul style="list-style-type: none"><li>• Graphic (picture, graph, etc. that illustrates the theme)</li><li>• Figure Number</li><li>• Action Caption</li></ul> |
|---|--|

# Elements of a Proposal Storyboard

- |   |  |
|---|--|
| <ul style="list-style-type: none"><li>• Section Identifier (e.g. number, title)</li><li>• <i>Theme sentence (usually in boldface italics)</i></li><li>• Summary of Key Points (in bullet or outline form)</li></ul> | <ul style="list-style-type: none"><li>• Graphic (picture, graph, etc. that illustrates the theme)</li><li>• Figure Number</li><li>• Action Caption</li></ul> |
|---|--|

## Sections Theme statement

- State your case up front (themes are the first sentence in the section/subsection)
- Contain quantified benefits to the customer.
- Answer, “Why should the customer select your company?”
- Should be provided for every first-level section and second-level subsection
- Are typically formatted to stand out from the rest of the text

## Steps for Theme Statement

- Step 1: Determine what your government customer cares about.

Step 2: Determine the features of your approach that address your customer's concern.

Step 3: Determine the benefits that your customer receives from each identified feature.

Step 4: Quantify each benefit, if possible.

Step 5: Pick the feature/benefit combination(s) that will resonate most.

Step 6: Form your theme statement.

## Sample Theme Statements

*Customer Z receives immediate start-up, enhanced user productivity, and life-cycle savings of \$250,000 in software development costs with our COTS solution, which delivers all the required financial management functions listed in the RFP.*

*Our proprietary XYZ tool automates the ordering process, which reduces the risk of errors caused by manual entry, decreases associated labor costs, and results in fulfilment times that are up to four times faster.*

*Company Q delivers the lowest-risk transition to Customer M; as the incumbent contractor, we can transition the contract in weeks rather than months, without interruption to service, since we only require administrative adjustments.*

**Exercise (10 min): Write a Theme Sentence for a proposal section that demonstrates that you understand the customer strategy described on the following slide.**



# Our Strategy for the Journey to Mars

Living and working in space require accepting risk, and the journey is worth the risk. Crews must be protected from the unique hazardous environments of deep space and on the Martian surface.

Often, systems will have to operate autonomously or remain dormant for years in preparation for crew. Overcoming these challenges will be essential on the journey to Mars.

These technological and operational challenges fall into three categories:

**1. *Transportation:***

sending humans and cargo through space efficiently, safely, and reliably;

**2. *Working in space:***

enabling productive operations for crew and robotic systems;

**3. *Staying healthy:***

developing habitation systems that provide safe, healthy, and sustainable human exploration.

Bridging these three categories are the overarching logistical challenges facing crewed missions lasting up to 1,100 days and exploration campaigns that span decades.

# Once you have your theme sentences...

- Arrange them in a paragraph
- Adjust them until they tell your story
- This is the first draft of your proposal

# Elements of a Proposal Storyboard

- |   |  |
|---|--|
| <ul style="list-style-type: none"><li>• Section Identifier (e.g. number, title)</li><li>• <i>Theme sentence (usually in boldface italics)</i></li><li>• Summary of Key Points (in bullet or outline form)</li></ul> | <ul style="list-style-type: none"><li>• Graphic (picture, graph, etc. that illustrates the theme)</li><li>• Figure Number</li><li>• Action Caption</li></ul> |
|---|--|



**Figure 1: Grizzlies should be avoided. Grizzly bears are large, powerful, and fast animals that can cause serious injury. The best advice is to avoid coming in contact with bears like this one, whenever possible.**

## Action Caption Example

- Figure 1: Large, powerful and fast animals, grizzly bears can cause serious injury and should be avoided whenever possible.

# Concept Review Assignment

**Due on 2/22 (noon); to be presented on 2/22**

1. Describe your project proposal in a set of storyboards
  - One Executive Summary storyboard that summarizes your entire proposal ( 1-2 slides- big picture)
  - One additional storyboard for each of the six questions introduced in the lectures of the past three weeks ( 6 slides)
2. Prepare a 10- 12 minutes power point presentation that describes your proposal.
3. See the sample from last year

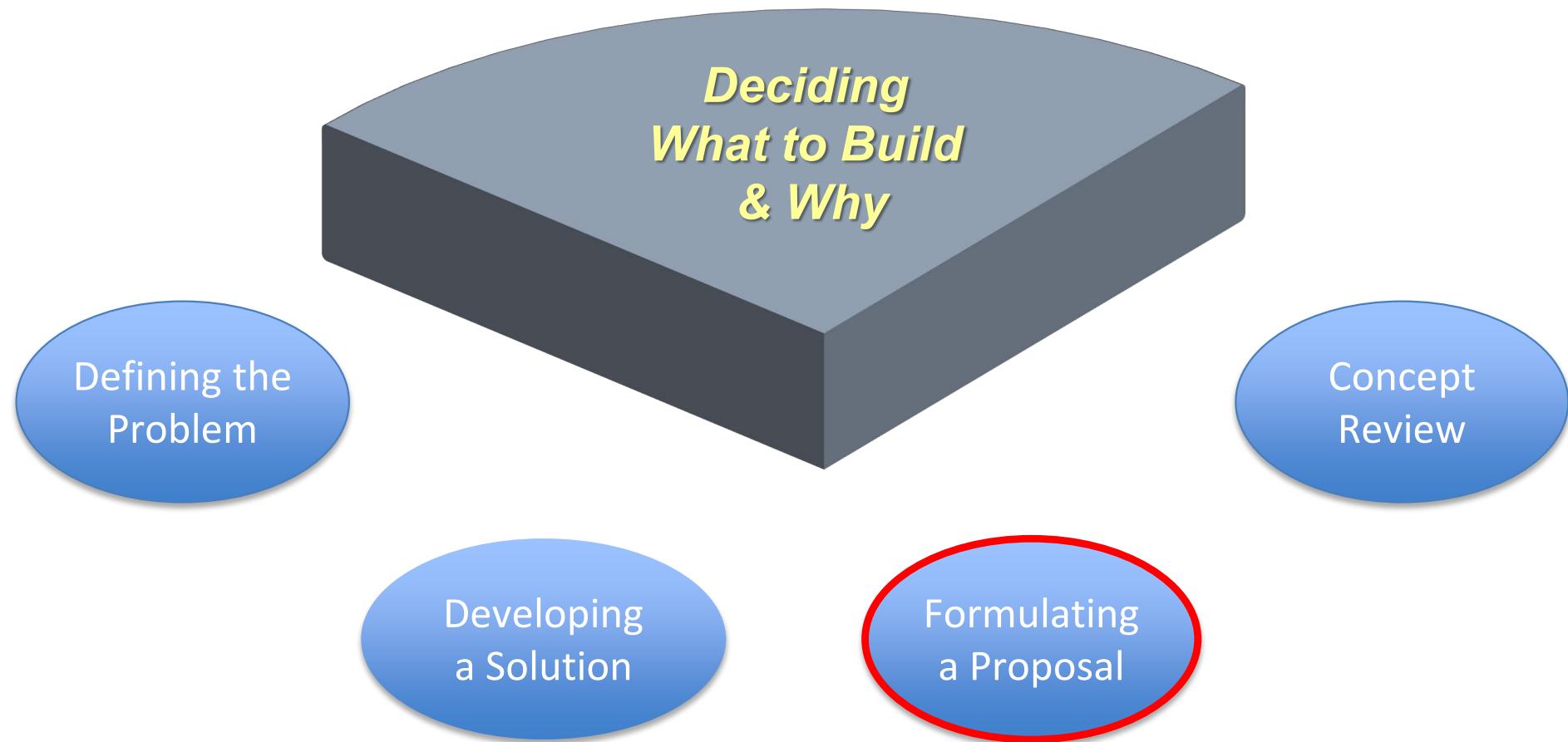
# Defining the Problem Key Questions:

1. What operational need or market opportunity is your system intended to address?
  - *Assessment Criteria: The need for the system is well understood, fully described in the language of the stakeholders and free of solutions.*
2. Who are the most important stakeholders and what are the key requirements of each?
  - *Assessment Criteria: The key stakeholders have been identified and their most important requirements defined, validated and clearly stated.*
3. What are the three to five most important features of your system that distinguish it from those of your competitors?
  - *Assessment Criteria: Features are specific, quantifiable (or readily observable), and important to the customer.*

# Creating an Operational Concept Key Questions:

4. What is your proposed system concept? What alternative concepts did you consider and why did you choose the one you proposed?
  - *Assessment Criteria: Broad range of concepts defined and systematically analyzed against criteria linked to key stakeholder needs.*
5. How will your proposed concept operate within the larger context to achieve its intended purpose?
  - *Assessment Criteria: The external systems with which the system will interact have been identified, the system boundary has been clearly defined, and the interactions between the system and the external systems have been specified from a black box perspective.*
6. What are the key specifications that will drive the system's design and development?
  - *Assessment Criteria: System requirements a) have been derived from and are linked to the stakeholder requirements, b) describe what the system shall do but not how, and c) are verifiable and properly written.*

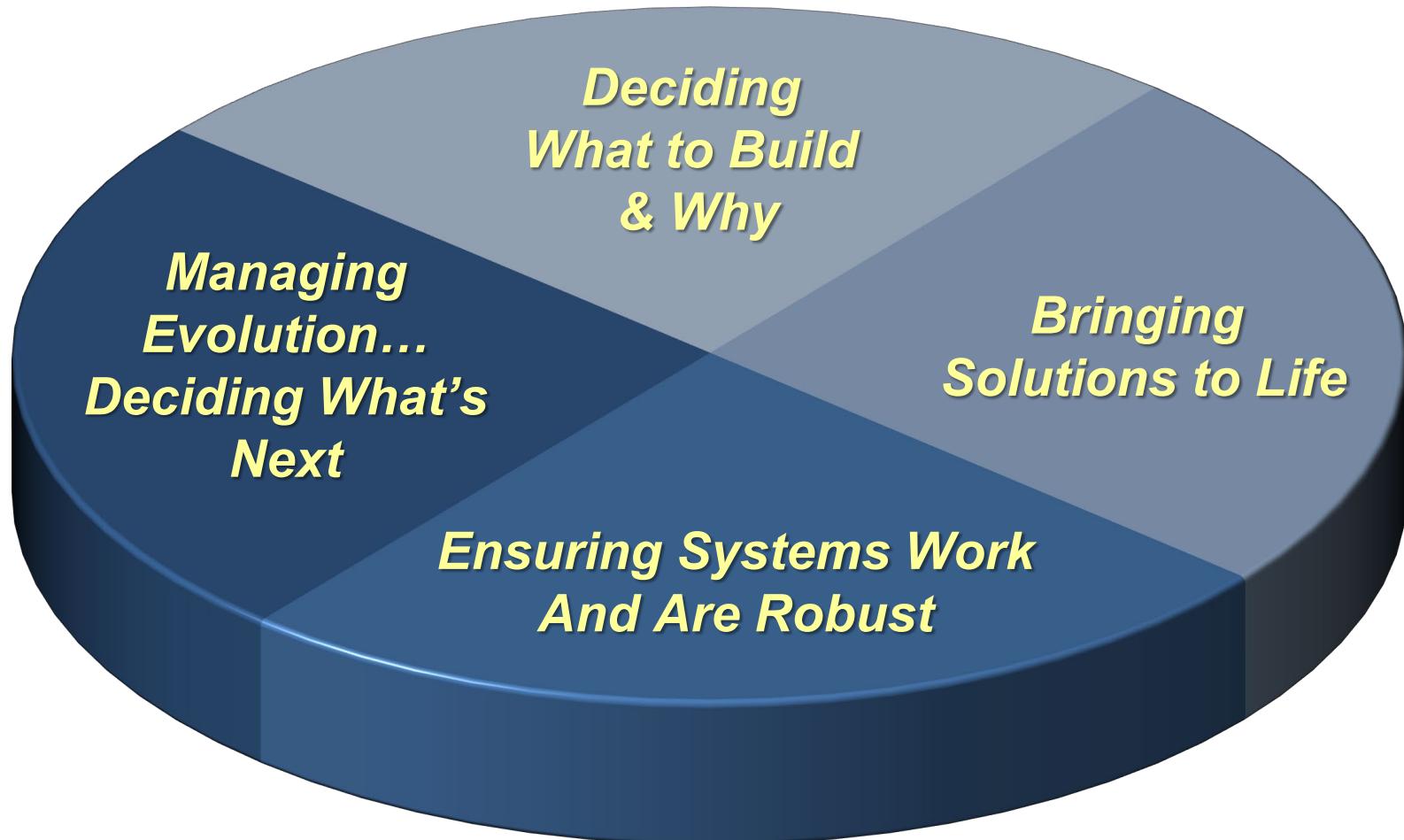
# Course Design



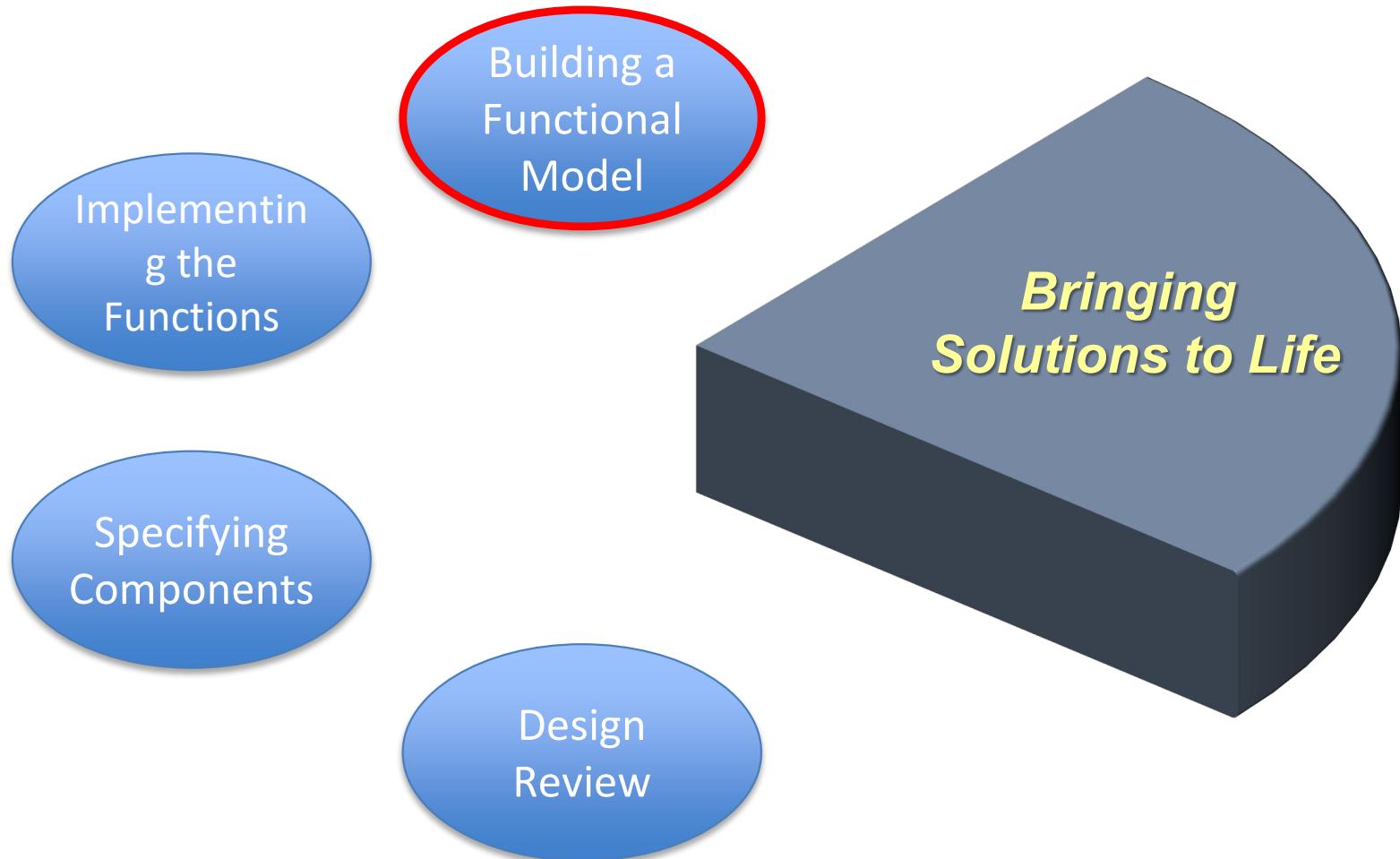
# Building a Functional Model

*“How will the system work?”*

# Course Design



# Course Design



# Deciding How the System Will Work

## Key Questions:

1. What top-level functions will your system have to perform in order to accomplish it's mission?
2. How else could you have partitioned the system functionality and why did you choose to partition it the way you did?
3. How will the top-level functions interact with each other and with their external environment?
4. How well will your functional architecture perform and what evidence can you provide to support that assessment?

# *Part 1: Introduction to System Architecture*

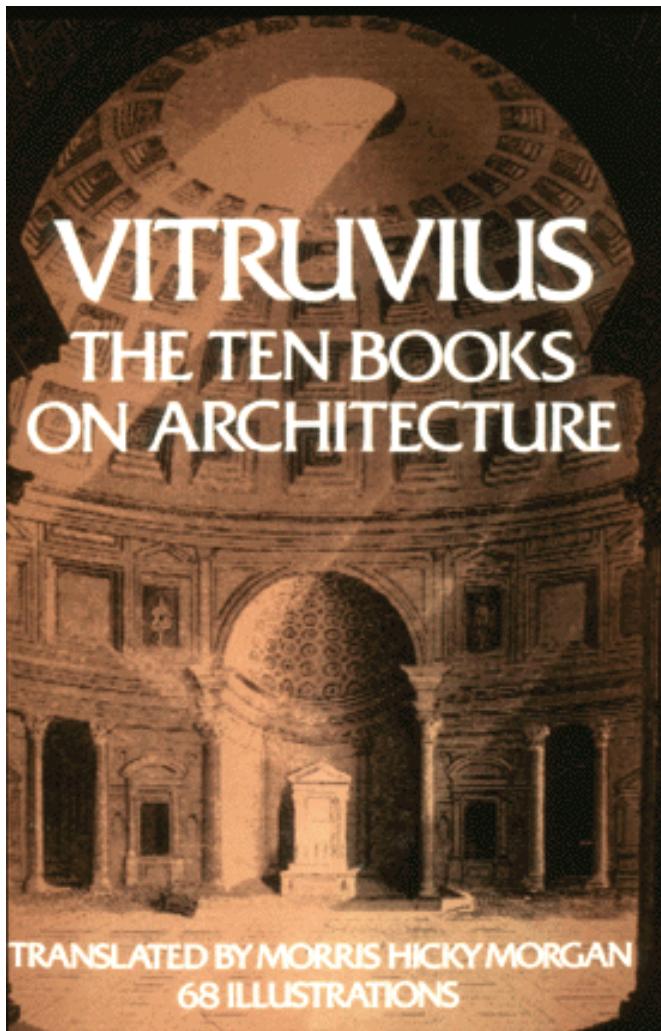
# Architecture as a Metaphor

- “...the systems engineer resembles an architect, who must generally have adequate substantive knowledge of building materials, construction methods, and so on, to ply his [or her] trade.
- “Like architecture, systems engineering is in some ways an art as well as a branch of engineering. Thus, aesthetic criteria are appropriate for it also. For example, such essentially aesthetic ideas as balance, proportion, proper relation of means to ends, and economy of means are all relevant in a systems-engineering discussion. Many of these ideas develop best through experience. They are among the reasons why an exact definition of systems engineering is so elusive.”

*Ref: Hendrik Bode, “The Systems Approach,” Report to the Committee on Science and Astronautics, U.S. House of Representatives by the National Research Council (U.S.), Panel on Applied Science and Technological Progress, 1967.*



# The Classical Architect



“The ideal architect should be a man [or woman] of letters, a skillful draftsman, a mathematician, a diligent student of philosophy, acquainted with music, not ignorant of medicine, learned in the responses of jurisconsults, and familiar with astronomy and astronomical calculations”

– *Vitruvius, 25 B.C.*

# Classical Virtues of Architecture



- ***Utilitas*** – usefulness; appropriate spatial accommodation
- ***Firmitas*** – structural soundness; robustness toward change
- ***Venustas*** – aesthetics and beauty; ability of a form to communicate its function and to fit into its environment

Ref: Vitruvius, De Architectura.

# What Is an Architect?

A creative and respected mediator between a “client” and a “builder”

- Envisions solutions to the client’s need that are feasible and that satisfy all the client’s requirements
- Communicates the vision to the client in a way that is compelling and leads to a decision to implement
- Communicates the vision to the builder in a way that results in its successful implementation
- Inspects and manages the implementation process to ensure that it is brought to a successful conclusion

# What Is a System Architect

A creative and respected mediator between a “client” and a “*design team*”

- Envisions solutions to the client’s need that are feasible and that satisfy all the client’s requirements
- Communicates the vision to the client in a way that is compelling and leads to a decision to implement
- Communicates the vision to the *design team* in a way that results in its successful implementation
- Inspects and manages the implementation process to ensure that it is brought to a successful conclusion

# The Role of a System Architect

## Challenges

- Determine user needs & how value is created
- Effectively manage complexity
- Accommodate change
- Work within organizational constraints
- Balance all of the above

## Required Skills

- Technical – To make it work
- Communication – To persuade others
- Political – To gain access to scarce resources necessary for the project's success

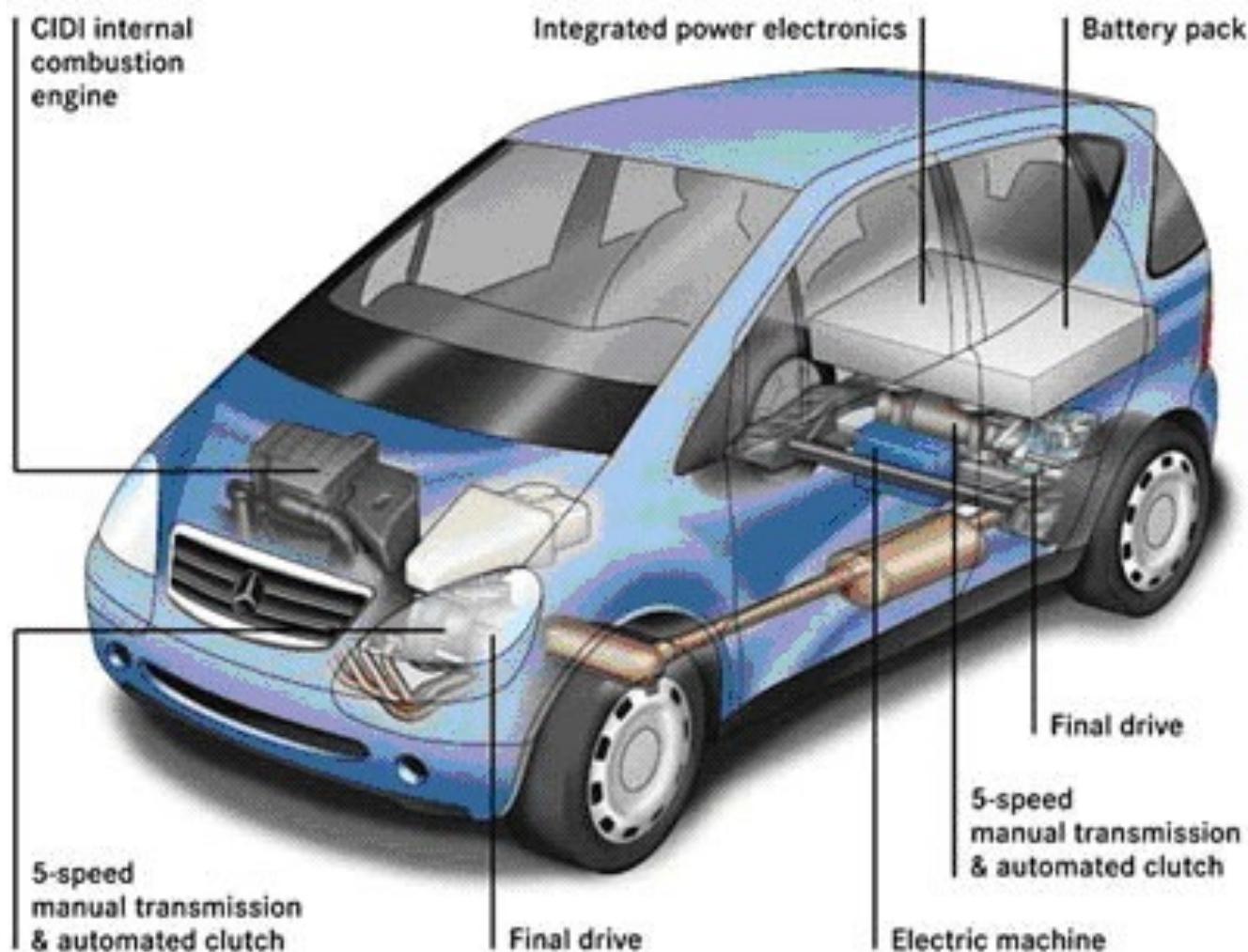
# Perspectives of a Classical Architect

- **Objective:** Achieve a good “fit” between the form to be designed and its context
  - Form – that over which we have control
  - Context – that which puts demands on the form
- **Challenge:** “We seek to produce harmony between a form we have not yet designed and a context we cannot fully describe.”
- **Process:**
  - Define the context as a set of requirements
  - Partition the requirements into clusters that are richly connected internally and as independent of each other as possible
  - Design a form that meets each cluster of requirements
  - Integrate the low level forms to produce the desired whole

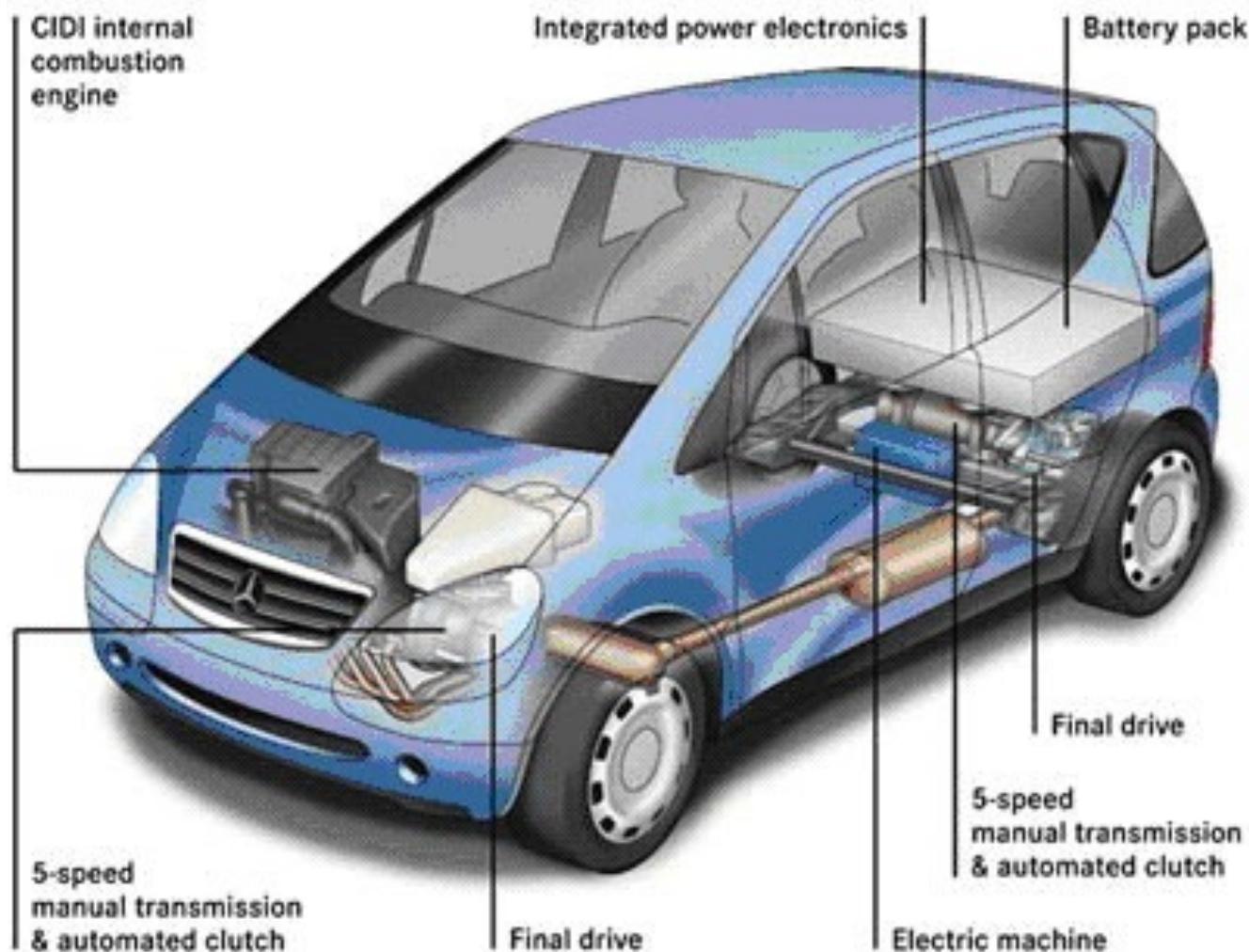
# System Architecture

- **System** – A set of components that work together to accomplish a common purpose
- **System Architecture** – The fundamental structure of a system: its elements, the roles they play, and how they are related to each other and to their environment

# What are the key components? How do they work together?



# How else could this be done? Why was it done this way?



# System Architecture and Design

- **System** - A set of components that work together to accomplish a common purpose
- **System Architecture** - The fundamental structure of a system: its elements, the roles they play, and how they are related to each other and to their environment
- **System Design** - an instantiation of the system architecture

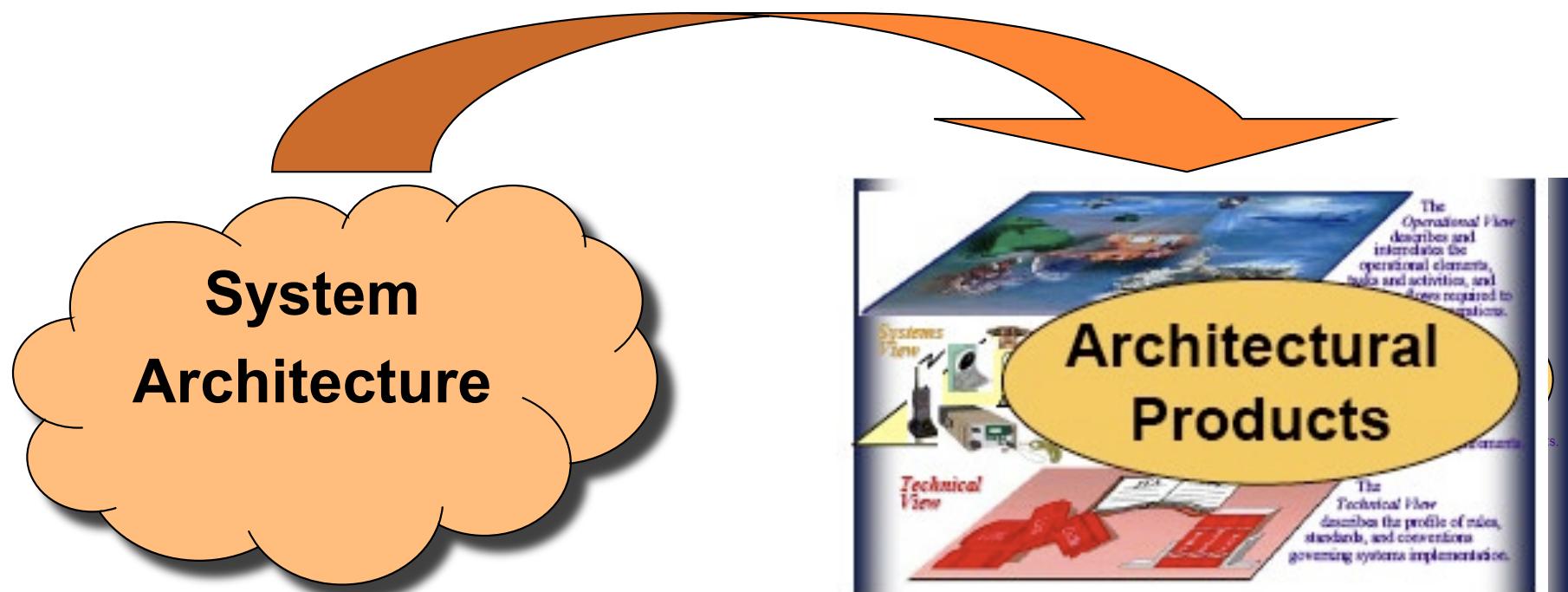
# All architecture is design; not all design is architecture.

|                 | Design  | Architecture  |
|-----------------|---|---|
| Scope           | For a specific implementation   | Can guide multiple designs and therefore implementations  |
| Goals           | Ensures the specific implementation meets specified requirements (functional, non-functional, and project)        | Ensures a number of implementations meet wider business targets and strategies                                  |
| Decision Points | Rationale for decisions is driven by implementation parameters (time to market, cost, features, resourcing, etc.) | Rationale for decisions is driven by business strategies (market responsiveness, competitive positioning, etc.) |
| Detail          | Defines all elements and interfaces to an implementation level of detail  | Defines some elements of the system as abstract types, and some interfaces at an implementation level of detail |

# System Architecture

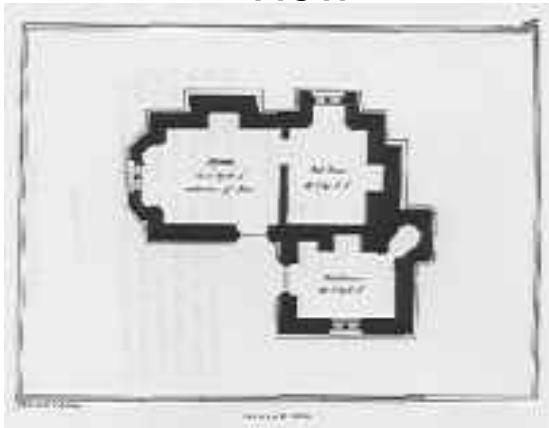
- **System** – A set of components that work together to accomplish a common purpose
- **System Architecture** – The fundamental organization of a system embodied in its components, their relationships to each other and to the environment, and the principles guiding its design and evolution. *Ref: ANSI/IEEE 1471-2000*
  - Also, the artifacts that describe this structure

# System Architecture - “The Map Is Not the Territory”



# Traditional Views of a Building Architecture

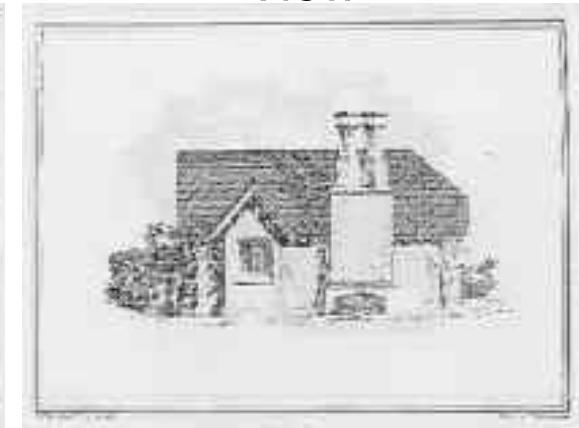
Plan  
View



Front Elevation  
View



Side Elevation  
View

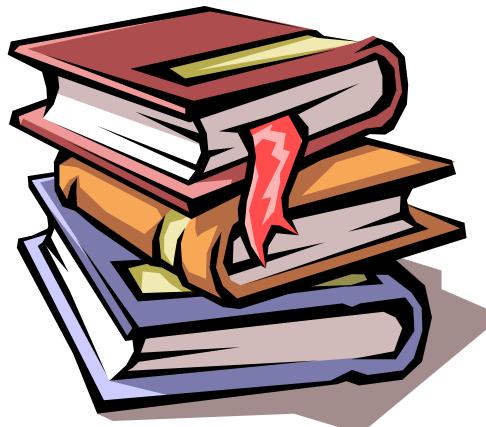


Today's architects don't just create documents, they build models.



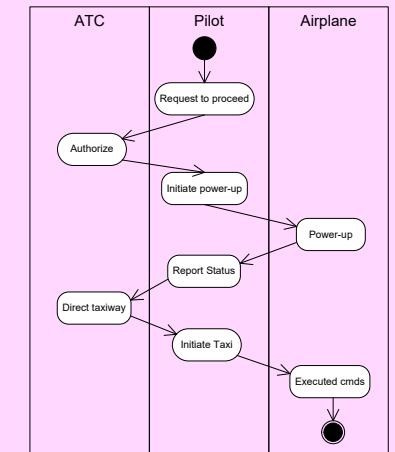
# SE Practices for Describing Systems

*Past*



- **Specifications**
- **Interface requirements**
- **System design**
- **Analysis & Trade-off**
- **Test plans**

*Present*



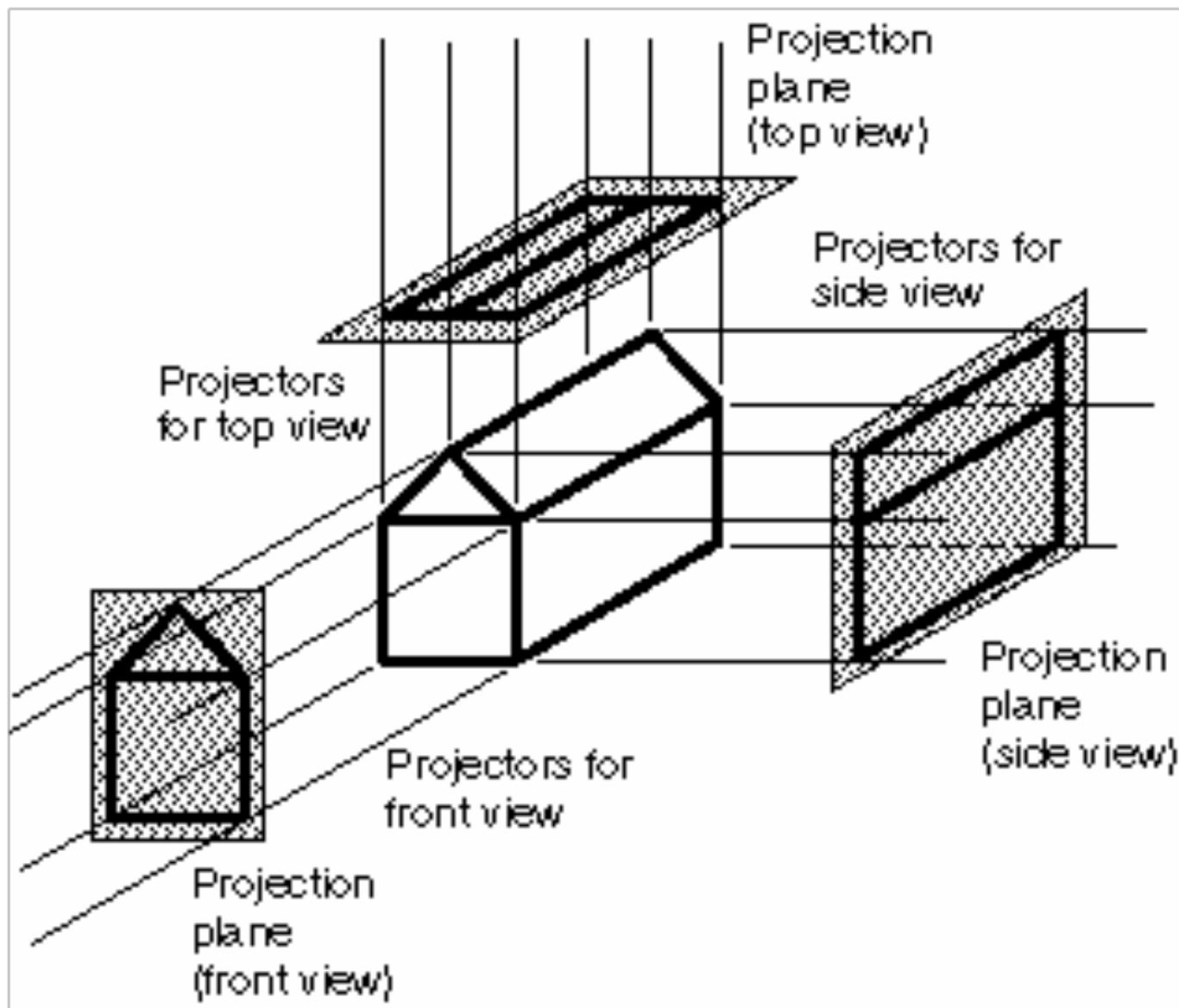
Moving from Document-Based to Model-Based Architecture

- Watch the video carefully, What do you think about this overall proposal? Pluses and Minuses
- (Canvas)
- <https://www.youtube.com/watch?v=XWJ-sE08ASg>

# Model-Based Architecture

- A system model:
  - Is the primary product of model-based systems engineering
  - Incorporates all the system requirements, functional elements, physical components and the relationships between them in a single repository
  - Requires some sort of tool, since there is no way to represent all this information at a single time
  - Is able to provide consistent “views” required by the system designers and a wide range of stakeholders, all derived from the same source

# Models and Views



# Traditional Views of System Architecture

- **Operational View**

- Defines the system from the users' (or operators') point of view – how the users will interact with the system and how the system will interact with external systems
- Shows the flow of data (inputs/outputs) between the system, its users and the external systems.

- **Functional View**

- Defines what the system must do – the capabilities or services it will provide and the tasks it will perform
- Shows the messages / data between functions

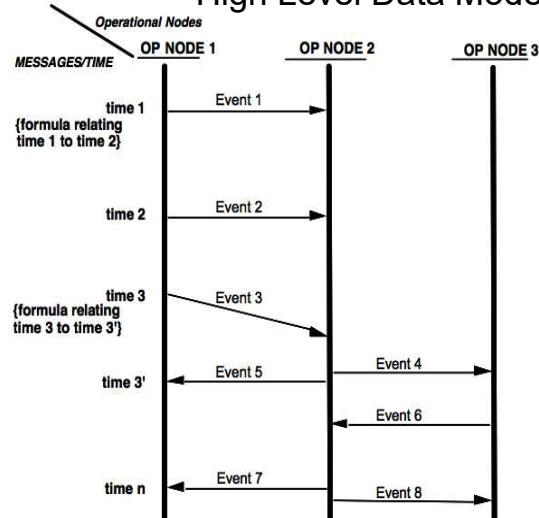
- **Physical View**

- Defines the partitioning of the system\* resources (hardware and software) needed to perform the functions. Also shows the interconnections between the resources - usually defined by the SE to the configuration item level – Computer Software (CSCI) or Hardware (HWCI)

# Traditional Views of System Architecture

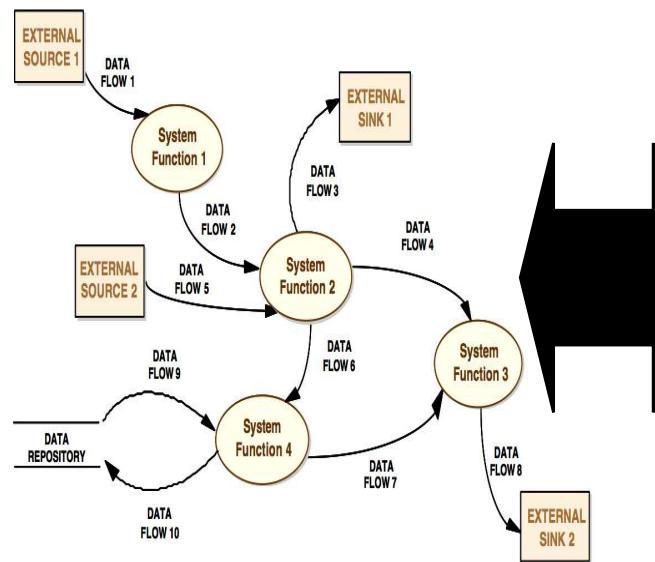
## Operational View

- 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 Case Scenarios
  - Sequence Diagrams
  - High Level Data Model



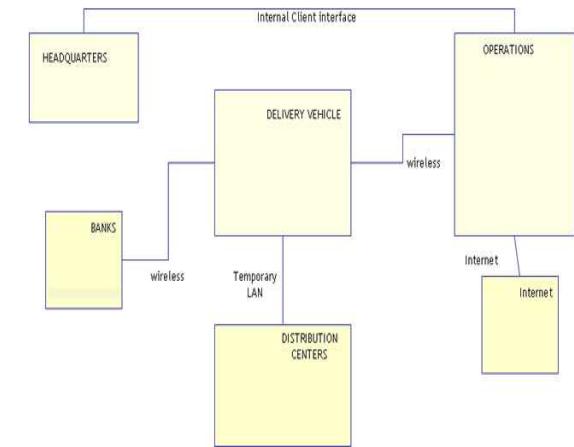
## Functional View

- Defines the capabilities, the services, or the functions provided by the system.
- Shows the messages / data between functions
- Usually described by:
  - IDEF0 Diagrams
  - Functional Flow Block Diagrams
  - N2 Diagrams



## Physical View

- Defines allocated resources (hardware and software)
- Shows the interconnections between the resources.
- Usually described by:
  - Physical Block Diagrams
  - Physical Interface Definitions





## *Part 2: The Functional View*

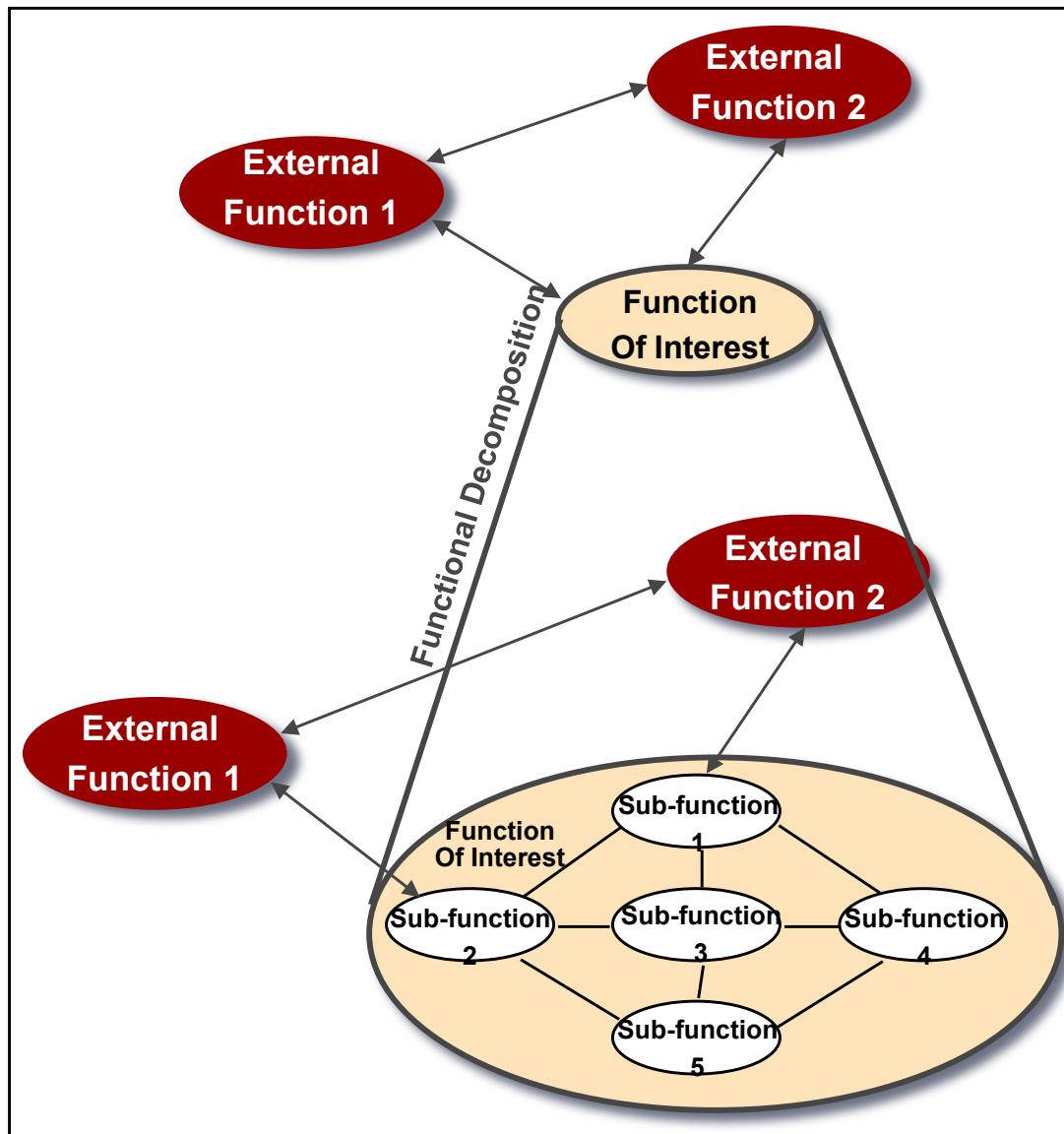
# Functional View of a System

- **Elements** – functions; the tasks the system performs
- **Interrelationships** between the elements – the data, information, material or energy exchanged between functions in order to accomplish the tasks

# Functions in Systems Engineering

- A function is a **process** that transforms inputs into outputs
- A function describes an **action** taken by the system or by one of its elements
- A function is represented by a **verb** or *verb-noun phrase*

# Functional Decomposition



# Decomposition vs. Composition

- Decompositon (top-down)
  - Partition system function a level at a time
  - Need sound definition of all inputs & outputs
- Composition (bottom-up)
  - Define many functionalities (bottom-level functions)
  - Synthesize functional hierarchy from many bottom-level functions
- Use both to develop the best solution

# Functional Building Blocks

- Signal functions - generate, receive, transmit, convert or distribute signals
- Information functions - analyze, interpret, query, store or convert information
- Material functions - support, enclose, store, reshape or transport matter
- Energy functions - generate, transform, or control: thrust, torque, power, heat or motion

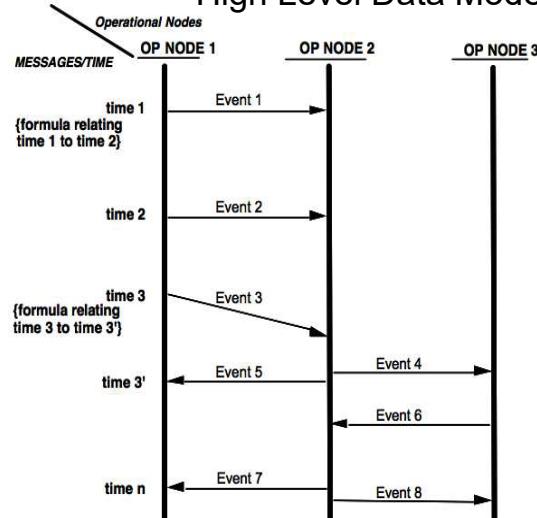
More examples?

Question - Are these elements hardware or software?

# Traditional Views of System Architecture

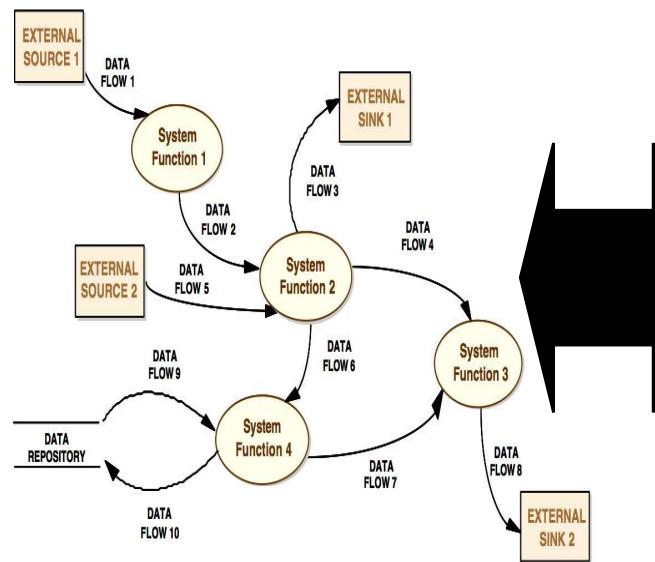
## Operational View

- 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 Case Scenarios
  - Sequence Diagrams
  - High Level Data Model



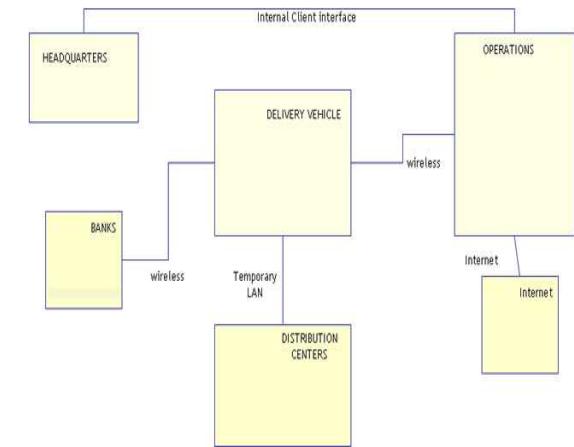
## Functional View

- Defines the capabilities, the services, or the functions provided by the system.
- Shows the messages / data between functions
- Usually described by:
  - IDEF0 Diagrams
  - Functional Flow Block Diagrams
  - N2 Diagrams



## Physical View

- Defines allocated resources (hardware and software)
- Shows the interconnections between the resources.
- Usually described by:
  - Physical Block Diagrams
  - Physical Interface Definitions

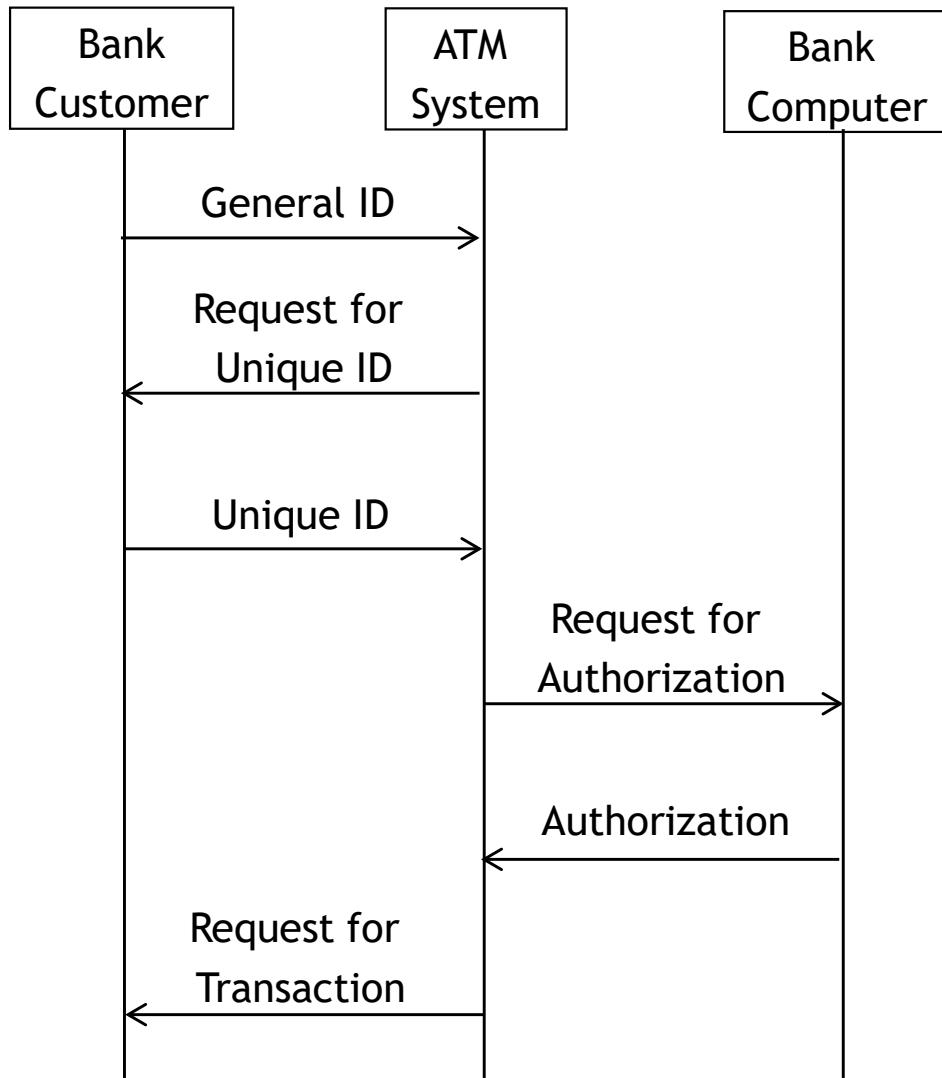




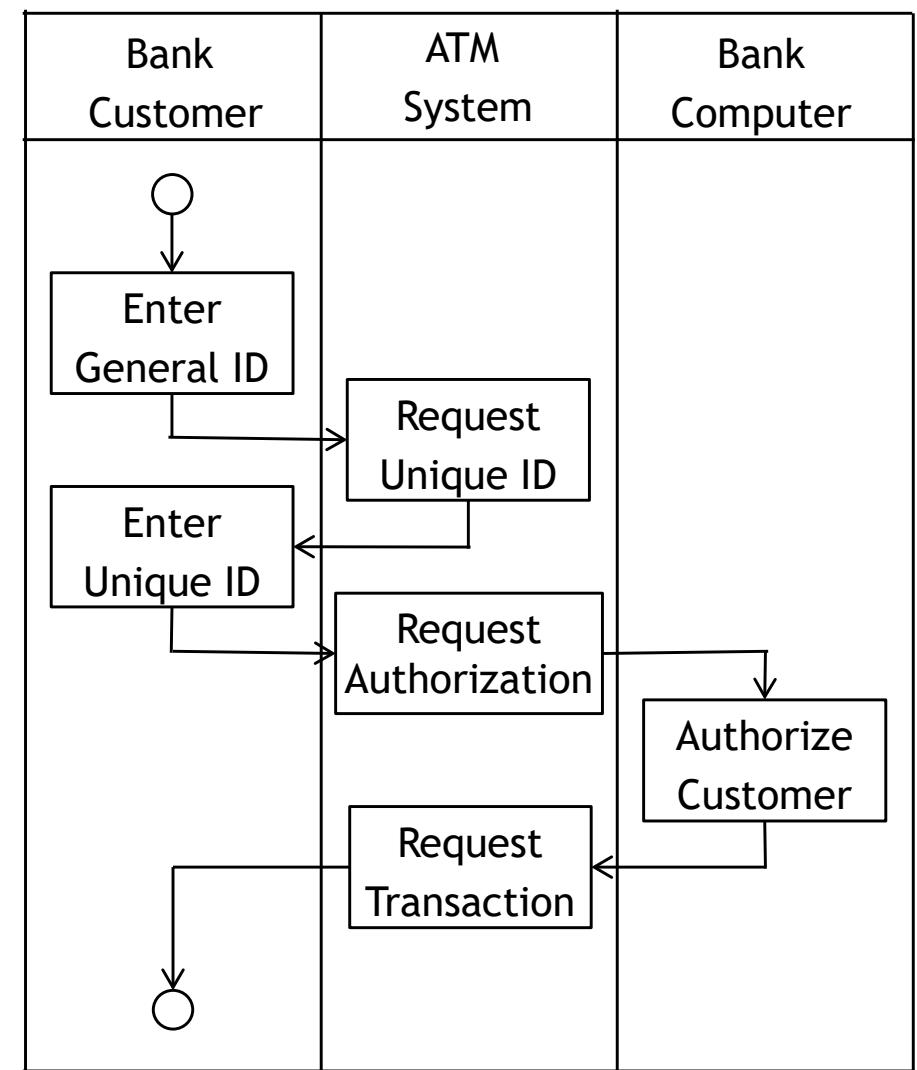
## *Part 3: ATM Example*

# ATM Example: Scenario Descriptions

**Sequence Diagram**

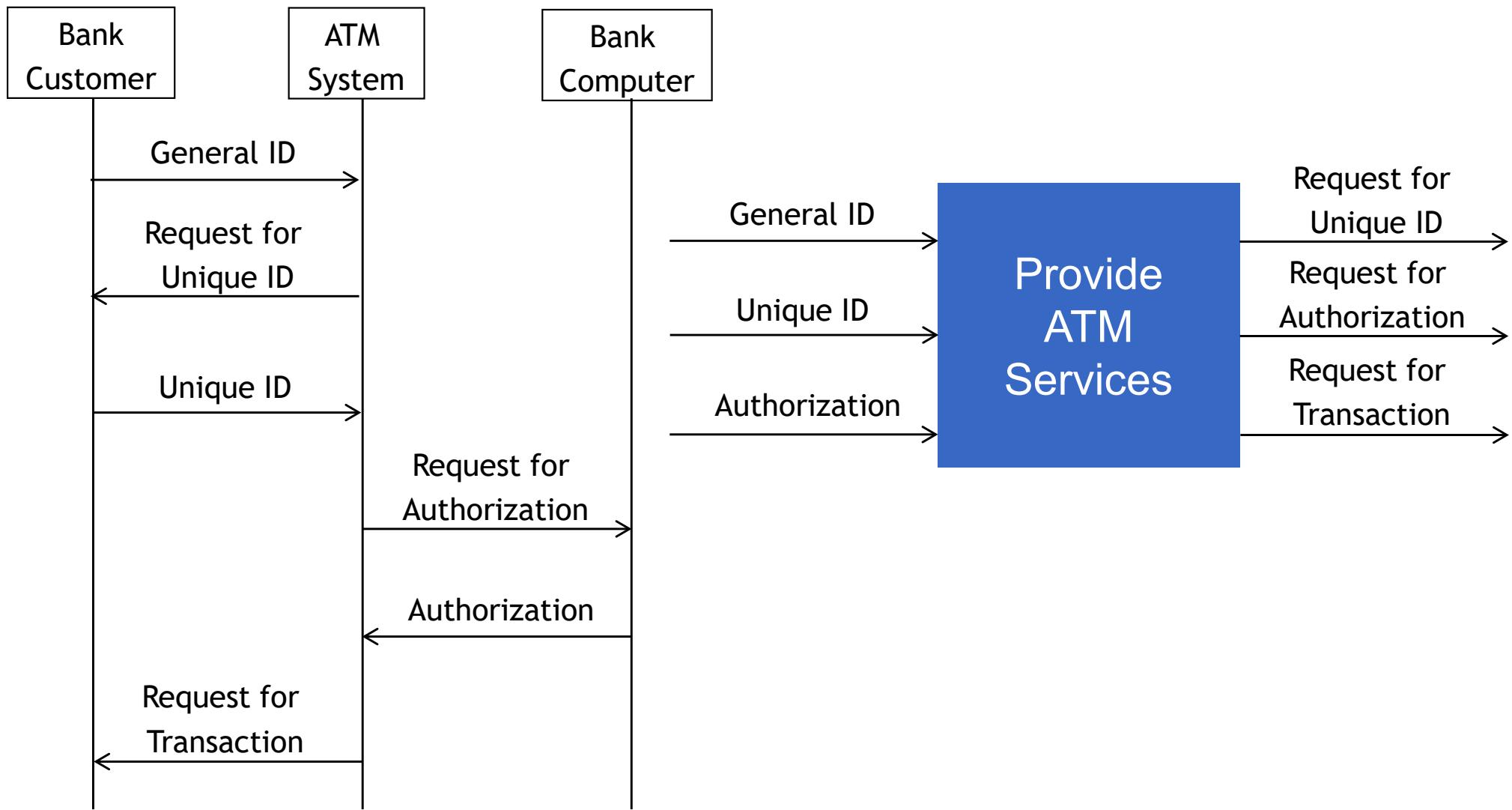


**Activity Diagram**



# ATM Example: Inputs and Outputs

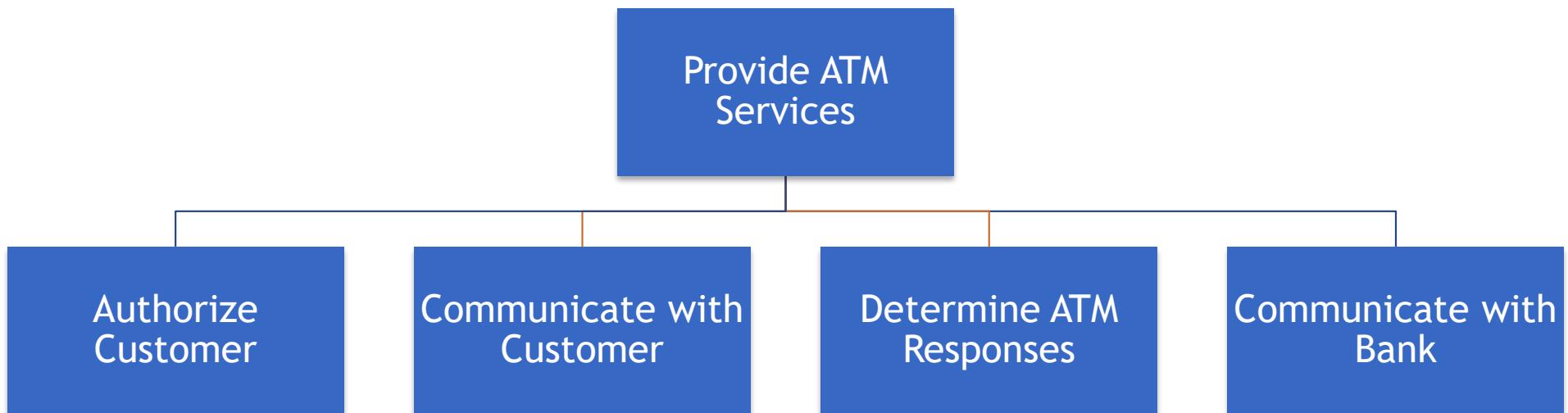
## Sequence Diagram



# ATM Example: First-Level Functional Decomposition

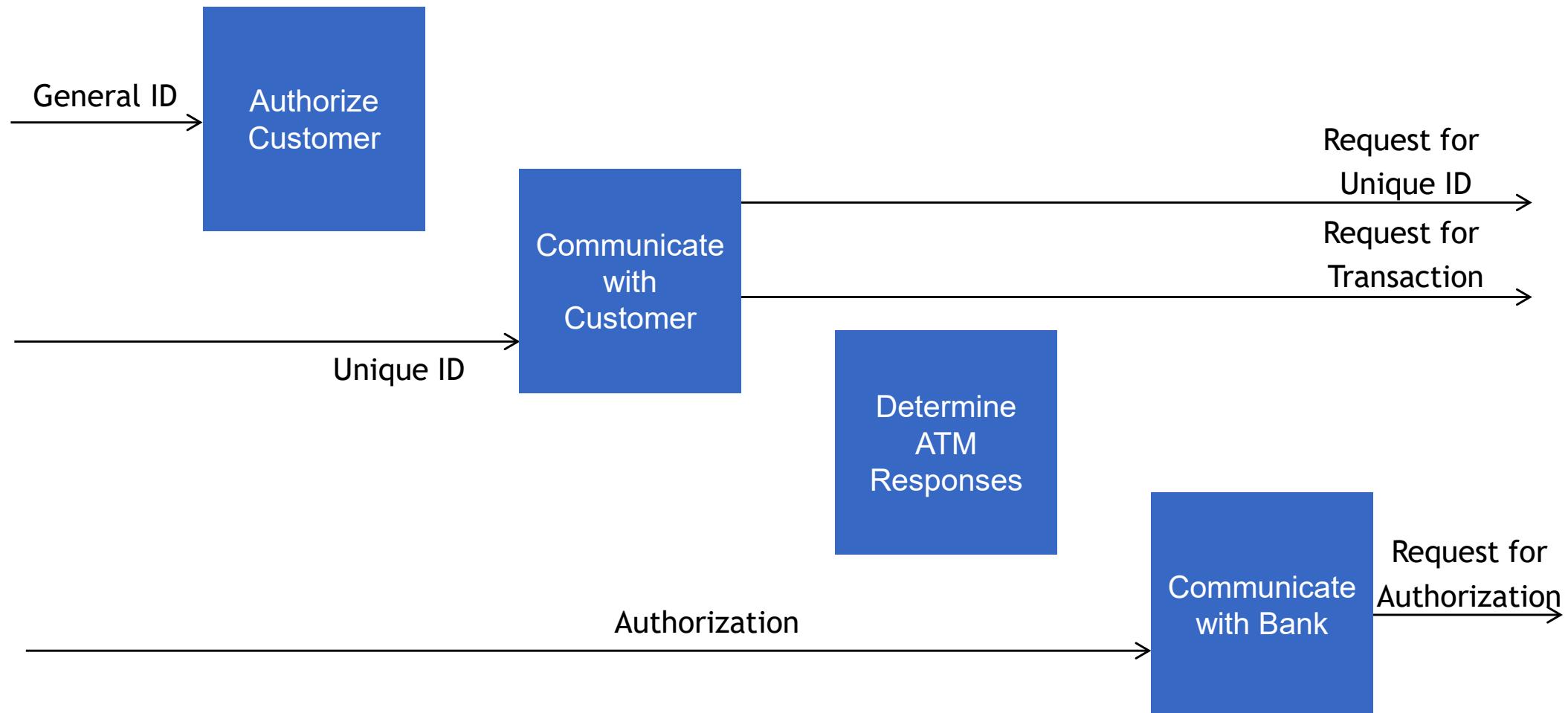
Provide ATM  
Services

# ATM Example: First-Level Functional Decomposition

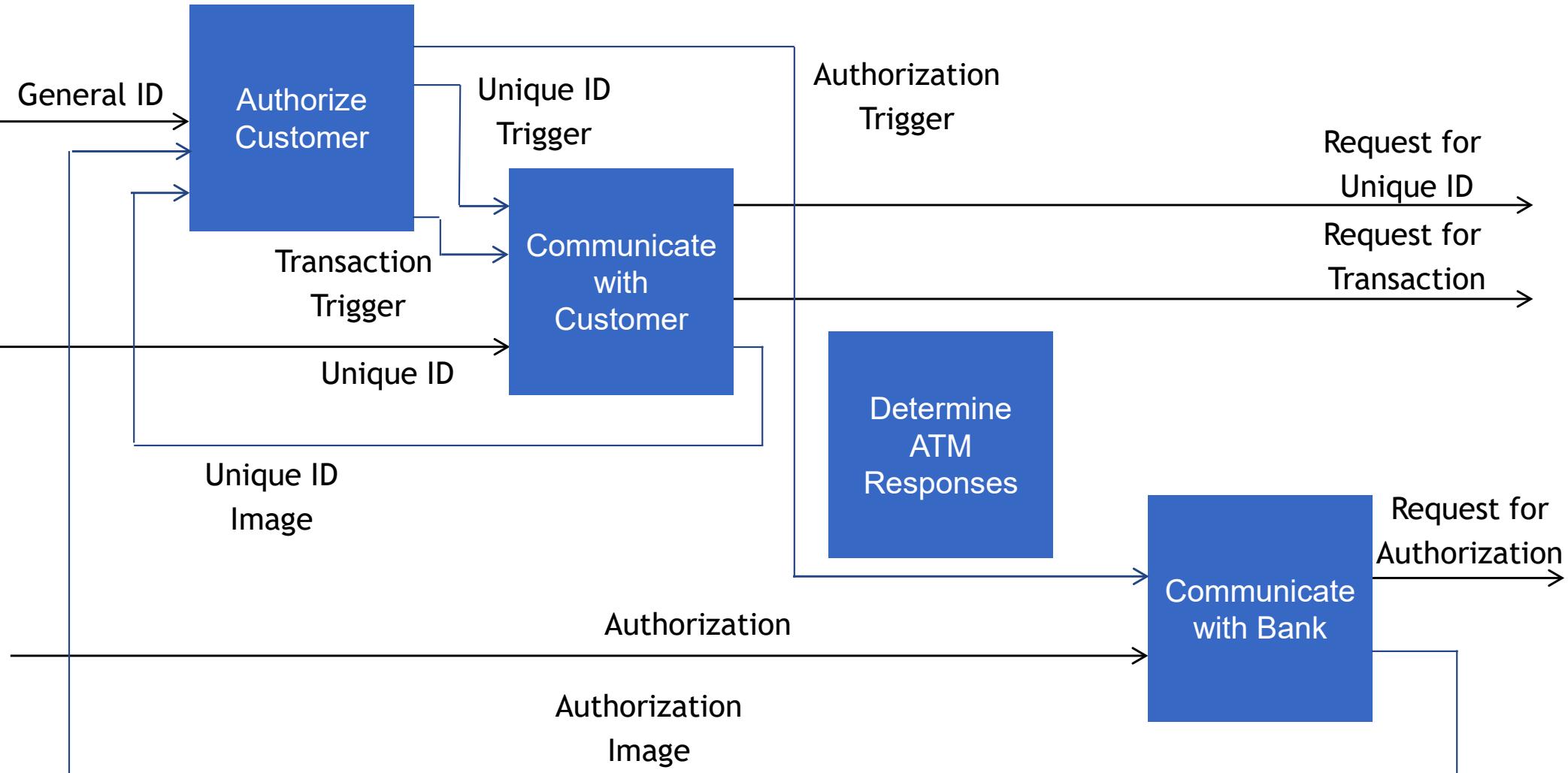




# ATM Example: Input and Output Assignment



# ATM Example: Sample Functional Flow



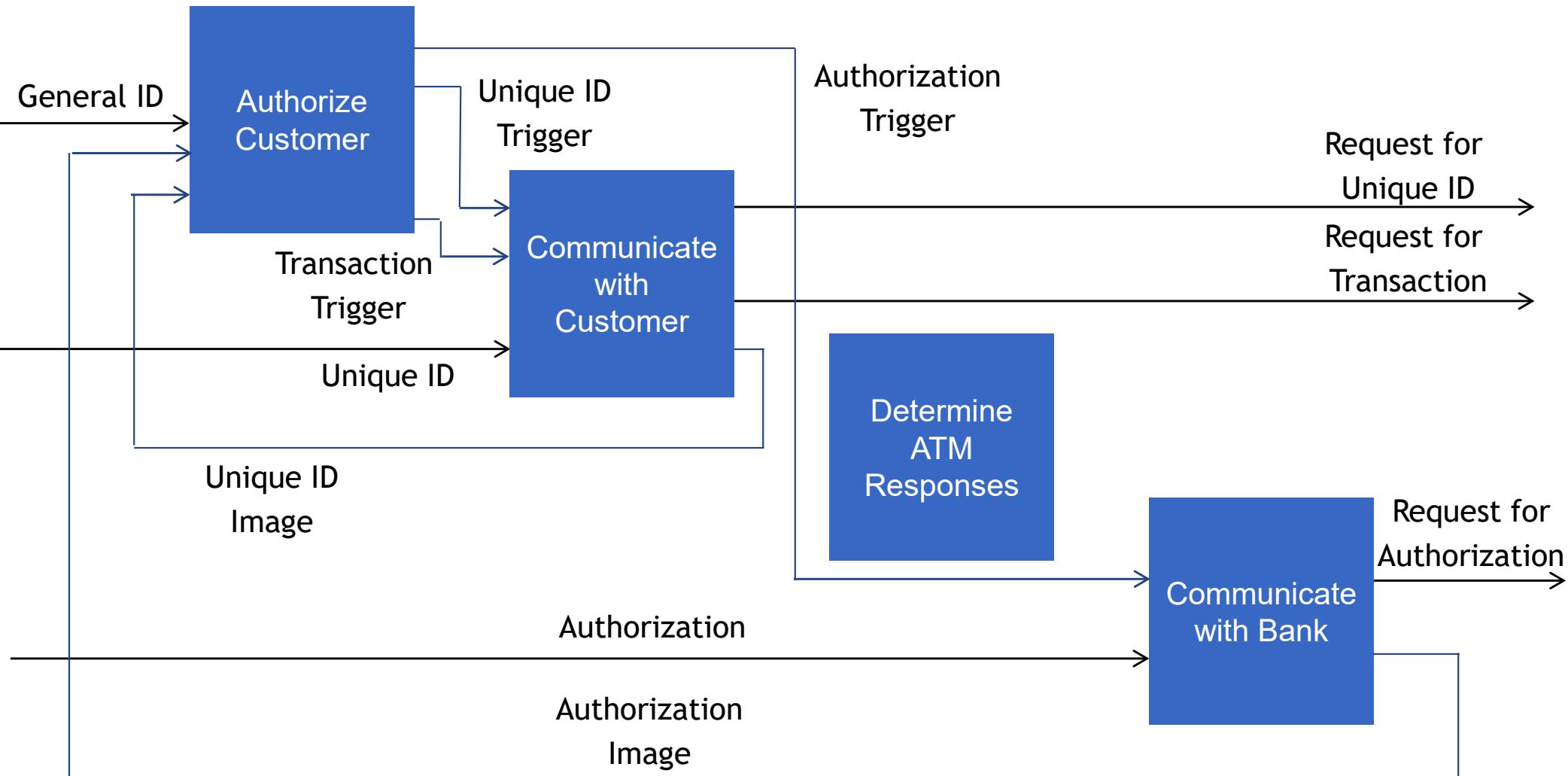
# ATM Example: N<sup>2</sup> CHART - EXTERNAL I/O

| GENERAL ID            | UNIQUE ID                       |                               | AUTH                     |                              |
|-----------------------|---------------------------------|-------------------------------|--------------------------|------------------------------|
| AUTHORIZE<br>CUSTOMER |                                 |                               |                          |                              |
|                       | COMMUNICATE<br>WITH<br>CUSTOMER |                               |                          | REQ FOR UID<br>REQ FOR TRANS |
|                       |                                 | DETERMINE<br>ATM<br>RESPONSES |                          |                              |
|                       |                                 |                               | COMMUNICATE<br>WITH BANK | REQ FOR AUTH                 |

# ATM Example: N<sup>2</sup> CHART - COMPLETE

| GENERAL ID         | UNIQUE ID                    |                         | AUTH                  |                              |
|--------------------|------------------------------|-------------------------|-----------------------|------------------------------|
| AUTHORIZE CUSTOMER | UID TRIGGER<br>TRANS TRIGGER |                         | AUTH TRIGGER          |                              |
| UID IMAGE          | COMMUNICATE WITH CUSTOMER    |                         |                       | REQ FOR UID<br>REQ FOR TRANS |
|                    |                              | DETERMINE ATM RESPONSES |                       |                              |
| AUTH IMAGE         |                              |                         | COMMUNICATE WITH BANK | REQ FOR AUTH                 |

# How well does the architecture work? How do you know?



# Deciding How the System Will Work

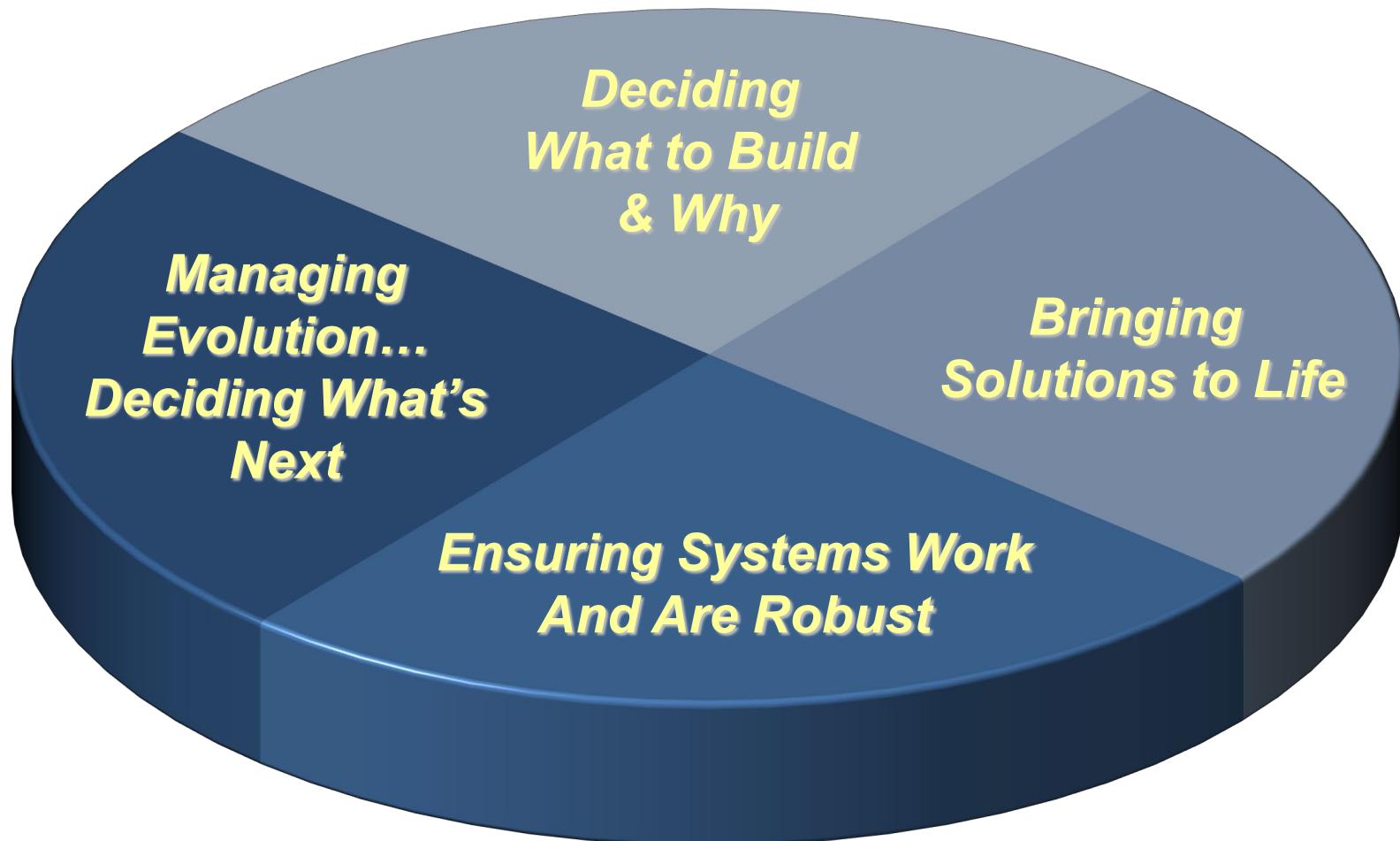
## Key Questions:

1. What top-level functions will your system have to perform in order to accomplish its mission?
  - *Assessment Criteria: Top-level functions are few in number (3-6) and fully represent the required system functionality.*
2. How else could you have partitioned the system functionality and why did you choose to partition it the way you did?
  - *Assessment Criteria: Alternative decomposition is plausible and the rationale for selecting the chosen partition is sound.*
3. How will the top-level functions interact with each other and with their external environment?
  - *Assessment Criteria: Key operational scenarios can be traced through the top-level functions.*
4. How well will your functional architecture perform and what evidence can you provide to support that assessment?
  - *Assessment Criteria: The functional architecture has been analyzed to verify that it achieves the key system requirements.*

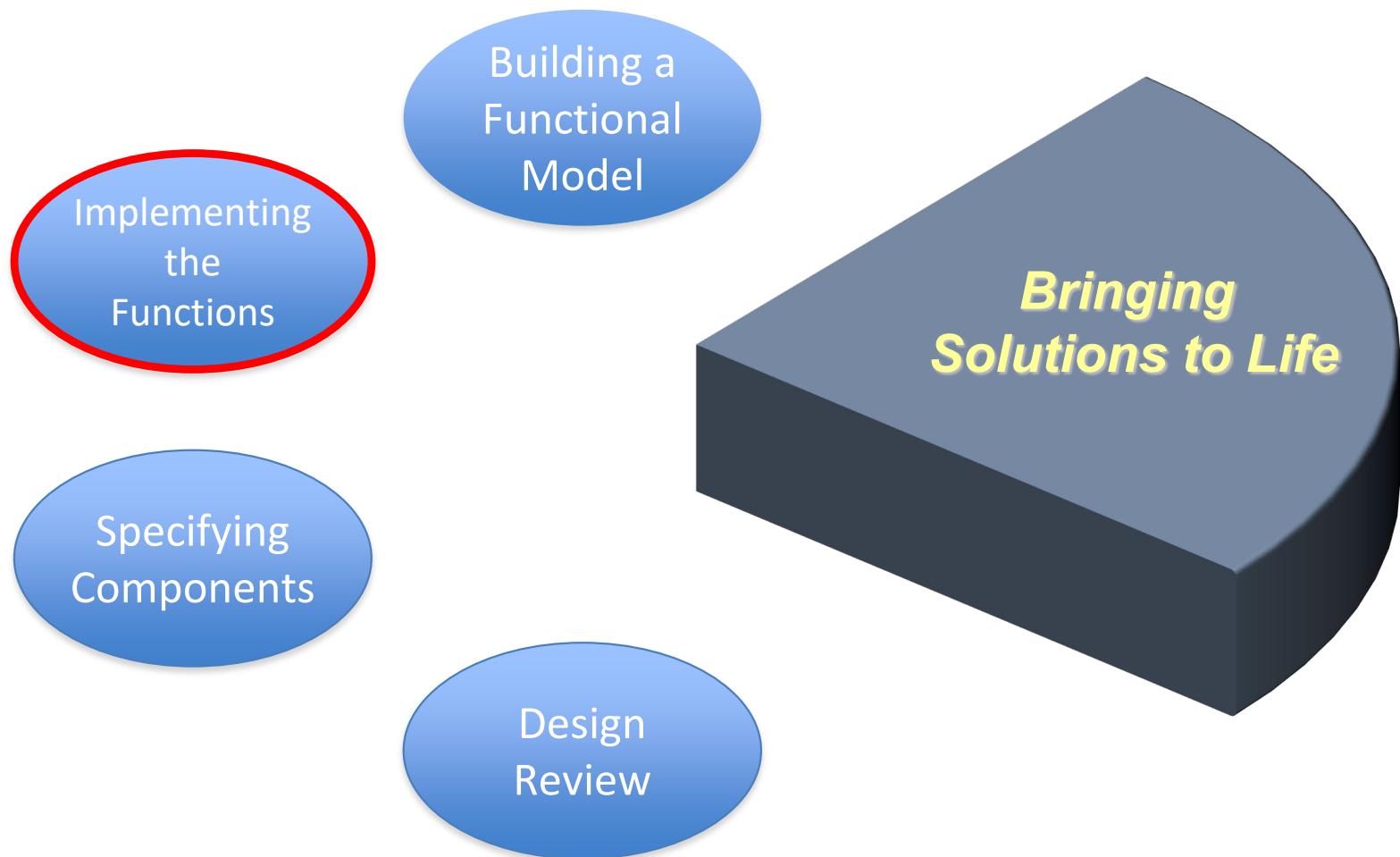
- Group Class work
- Submit to canvas
- 30 min
- Come up with 2-3 operational, and functional view of your system. And explain each with 2-3 sentences
- Feel free to use visuals



# Course Design



# Course Design



# Class Schedule (1 of 2)

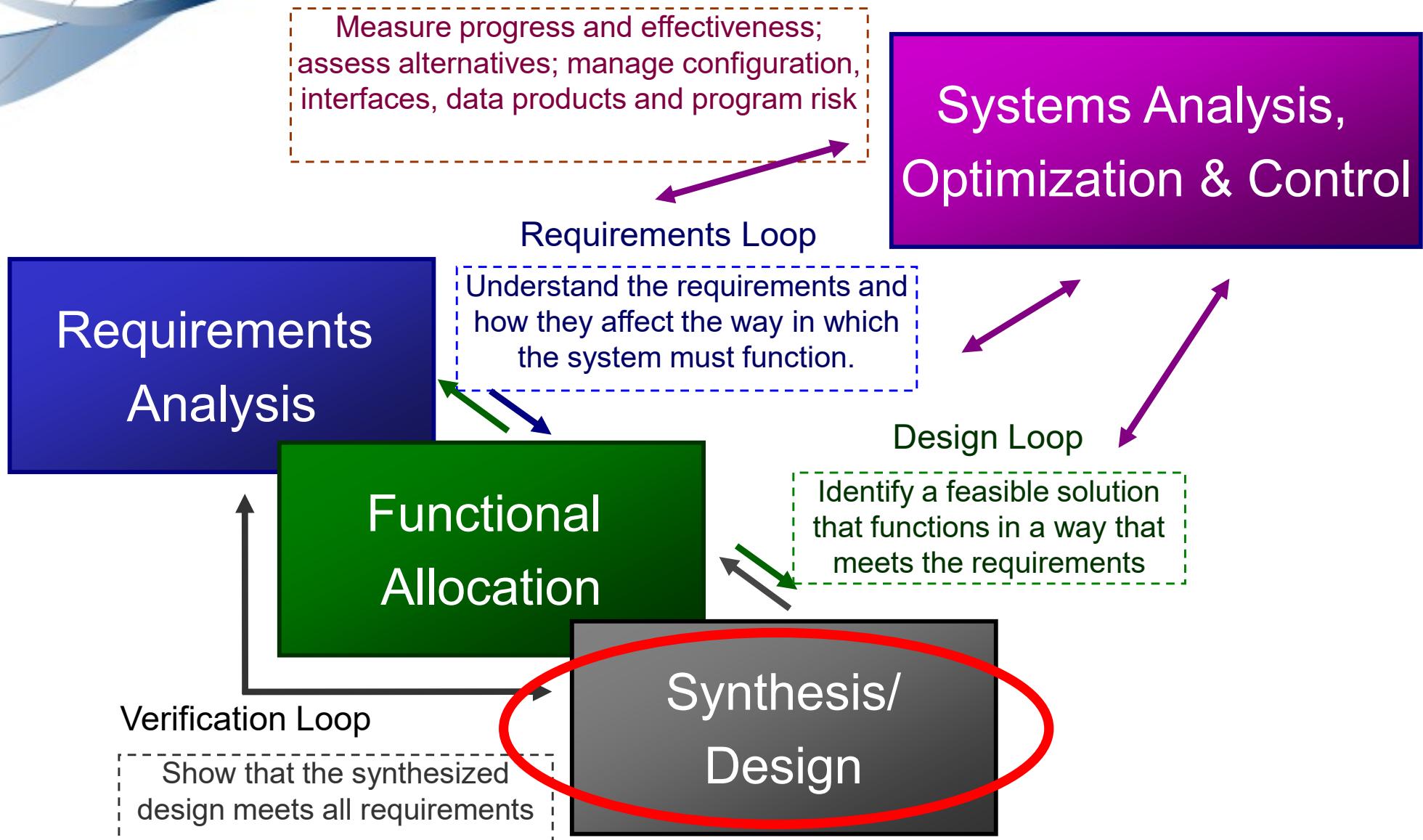
| Week | Topic  |
|------|--|
| 1    | <ul style="list-style-type: none"><li>• Thinking in Terms of Systems</li></ul> <p><b><i>Deciding What to Build and Why</i></b></p> |
| 2    | <ul style="list-style-type: none"><li>• Defining the Problem</li></ul>   |
| 3    | <ul style="list-style-type: none"><li>• Developing a Solution</li></ul>  |
| 4    | <ul style="list-style-type: none"><li>• Formulating a Proposal</li></ul>   |
| 5    | <ul style="list-style-type: none"><li>• Concept Review</li></ul> <p><b><i>Bringing Solutions to Life</i></b></p>                   |
| 6    | <ul style="list-style-type: none"><li>• Building a Functional Model</li></ul>  |
| 7    | <ul style="list-style-type: none"><li>• Implementing the Functions</li></ul>   |
| 8    | <ul style="list-style-type: none"><li>• Specifying Components</li></ul>  |
| 9    | <ul style="list-style-type: none"><li>• Design Review</li></ul>  |



# What is System Synthesis?

- *System synthesis*, also known as system design, translates the system functional architecture into a physical architecture. It creates a ‘how’ for every ‘what’ and ‘how well’.
- For each functional subsystem, alternative physical solutions are considered, trade studies are performed and a preferred solution picked.
- *System synthesis* is an iterative process - namely, as different physical architectures are considered functional or performance allocation may be changed to create a ‘balanced’ solution.
- A ‘balanced’ solution means that there is consideration of the overall system risk, cost, technical maturity and robustness for each combination of subsystems.
- The products of *system synthesis* include a physical architecture baseline (the ‘design-to’ baseline) and the subsystem trade study results.

# System Synthesis Within the Traditional Systems Engineering Process



# An Approach to System Synthesis - the

## steps

1. Begin with the functional architecture, its performance requirements and constraints (from functional analysis).
2. Allocate (or derive) subsystem performance and resource requirements.
3. Define physical subsystem alternatives.
4. Assess technology alternatives and their maturity.
5. Define physical interfaces.
6. Estimate subsystem and system performance of each combination of alternatives.
7. Use performance-resource curves (utility curves) to identify break points.
8. Determine the driving requirements and consider reallocation.
9. Select a preferred system design. i.e., the physical architecture with subsystem implementation plans and functional and performance allocations and system performance estimates.

# Implementing the Functions

*“How will the system be built?”*

# Implementing the Functions **Key Questions:**

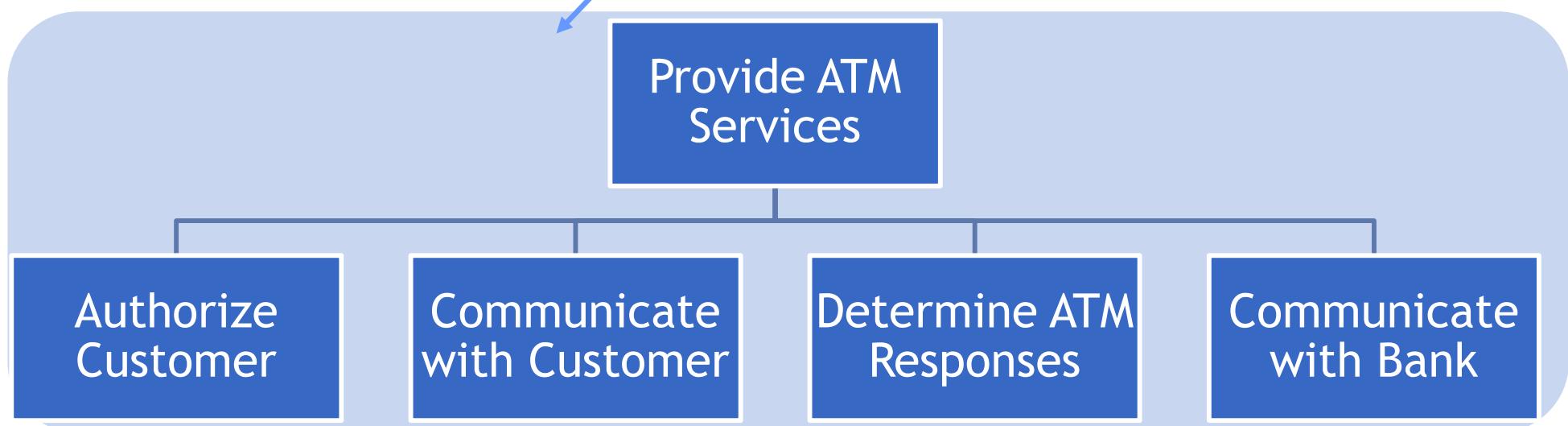
1. What are the major subsystems of your system and how do they map to the functions the system must perform?
  
2. What are the key interfaces between the subsystems and between them and the external systems with which the system interacts?

# System Architecture and Design

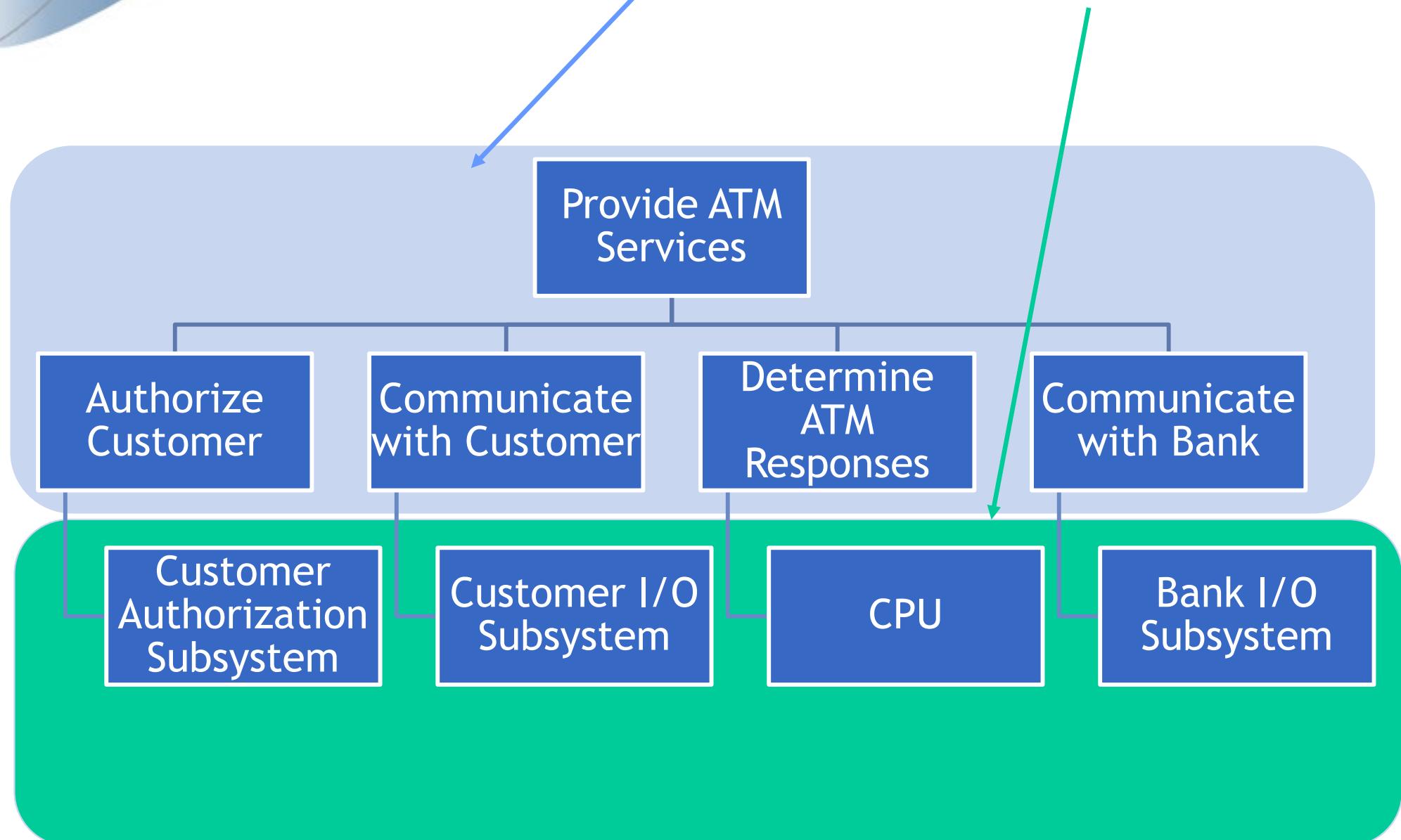
- **System** - A set of components that work together to accomplish a common purpose
- **System Architecture** - The fundamental structure of a system: its elements, the roles they play, and how they are related to each other and to their environment
- **System Design** - An instantiation of the system architecture

# *Allocating Functions to Subsystems*

# Allocating Functions to Subsystems



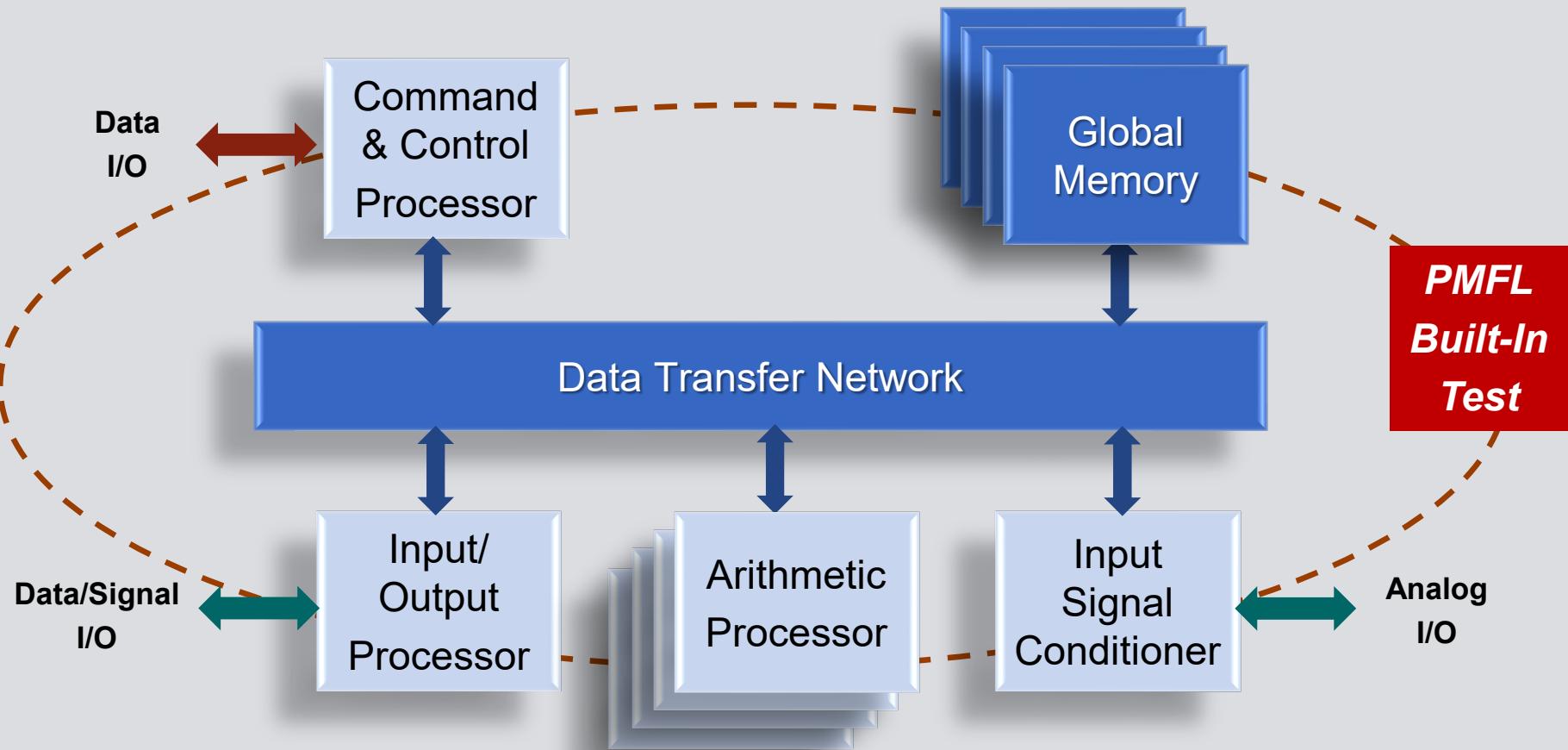
# Allocating Functions to Subsystems



# Allocating Functions to Subsystems

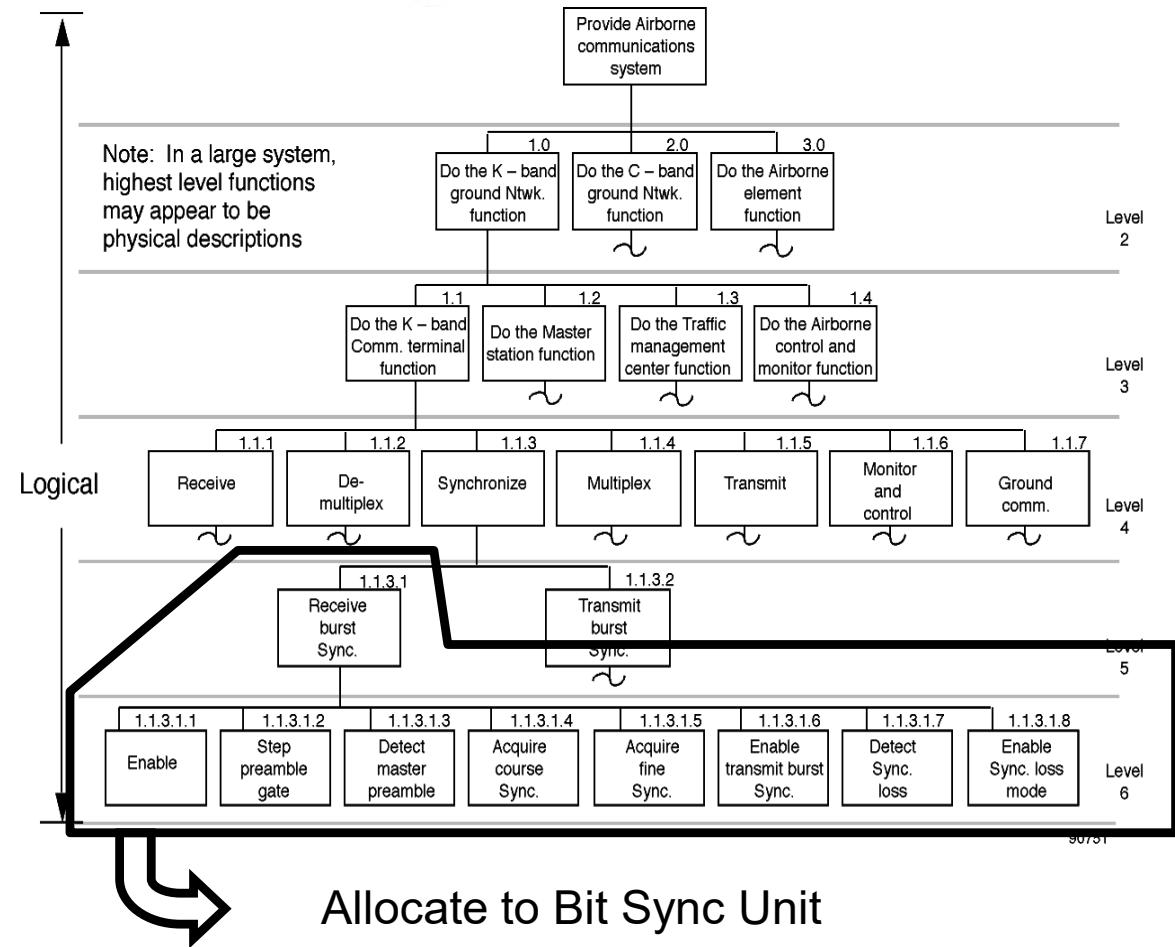
Subsystems may include:  
***Hardware, Software, Interfaces, Platforms, Standards***

## Enhanced Modular Signal Processor



# Functional Allocations to Lower Level Products

- Group functions that are strongly related to each other; separate elements that are unrelated.
- Choose subsystem allocations so that each can be independent of the others.
- Choose a configuration with minimal communications between subsystems.



“The most important aggregation and partitioning heuristics are to minimize external coupling and maximize internal cohesion.”

– Maier and Rechtin

# Modular Design for Spacecraft

- Typically spacecraft subsystems are modular by tradition, with subsystems filling familiar functions like propulsion, telecommunications, attitude determination and control.
- BUT modular designs for spacecraft do have difficult implementation decisions. For example, it is difficult to define the boundaries between:
  - Hardware and software functions
  - Data management (e.g., compression, stacking, error correction, etc) between instruments and the spacecraft
  - Data management between spacecraft and ground control



*The Cassini Spacecraft*

*Impact of*  
*Architectural Platforms*

## 1981 Dodge Aries



## Chrysler K-Car

1981–1995

Dodge Aries,  
Plymouth Reliant,  
Chrysler LeBaron,  
Dodge 400,  
Dodge Dart

~360,000 vehicles

## GM J-Car

Buick, Oldsmobile, and Cadillac  
1981–2005

~10,150,000 vehicles



1982 Cadillac Cimarron

# Platform-Based Development : Nokia Mobile Phones

## Lessons learned:

- Success *requires not only a platform, but a governance process for managing the platform*
- *There is a need for rules for maintaining separation between the platform and the variants*
- *The platform cannot be hijacked by the dominant customer*
- *It's difficult to assess the ROI of platform and how and when to invest in platforms versus products*
- *The platform must be reliable in its functionality and release schedule*

# What is the Platform?



iPhone 2G  
June 29, 2007



iPod Touch  
September 5, 2007

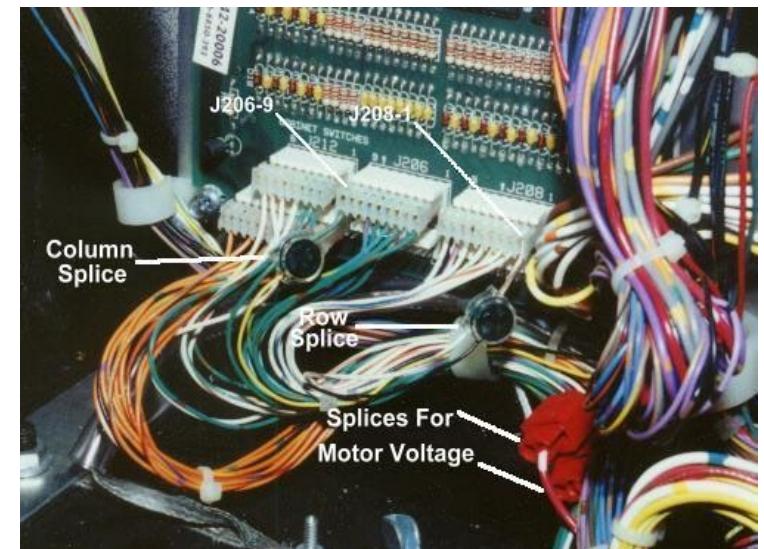


iPad  
March 12, 2010

# *Impact of Interfaces*

# Interfaces to other “things”

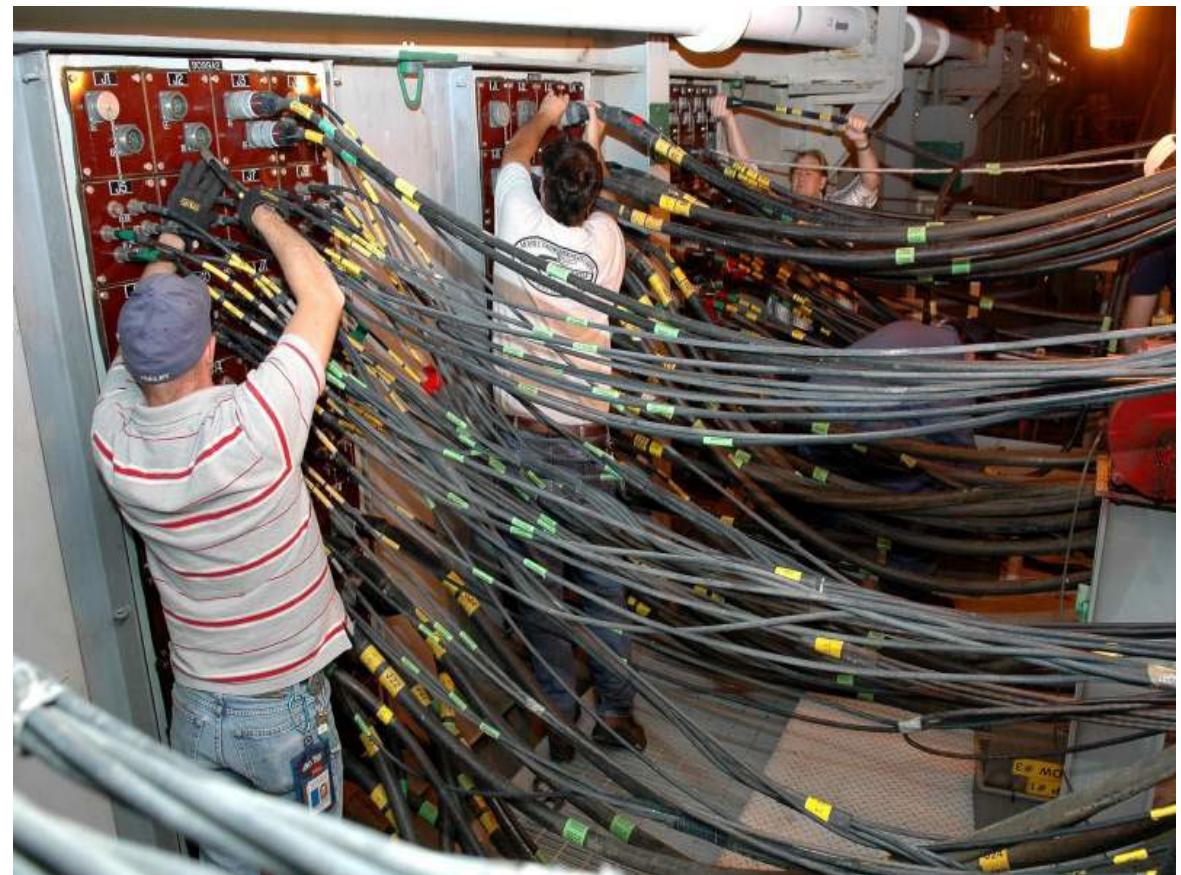
- A connection for ‘hooking together’ system components - external or internal.
- Have a logical and physical component responsible for carrying information or electromechanical energy from one component to another.
- Must identify interfaces, evaluate options, and allocate inputs/outputs to them.



- **Interfaces** are all about how systems and subsystems interact with each other and their operating environment.
- **Interface Purposes**
  1. Physically link or bind two or more system elements
  2. Adapt one or more incompatible system elements
  3. Buffer the effects of incompatible system elements
  4. Leverage human capabilities
  5. Restrain a system element and its usage

# Broad View of Interfacing

- Wide variety of ‘interfaces’ in all systems and products.
  - Software
  - Communication
  - Electrical
  - Mechanical
  - Optical
  - Acoustical
  - Chemical
  - Biological
  - Etc...



# Logical vs. Physical

- **Logical**

- Direct or indirect association between two entities
- Who communicates
- What scenarios
- When to communicate

- **Physical**

- How communication happens
- ‘Specialized’ interfaces

## Standardized vs Dedicated



- **Standardized** - standard, modular, interchangeable interfaces - complying with a ‘standard’ like RS-232, USB, Ethernet, etc.
- **Dedicated** - Unique, dedicated interfaces - often limiting compatibility with other systems

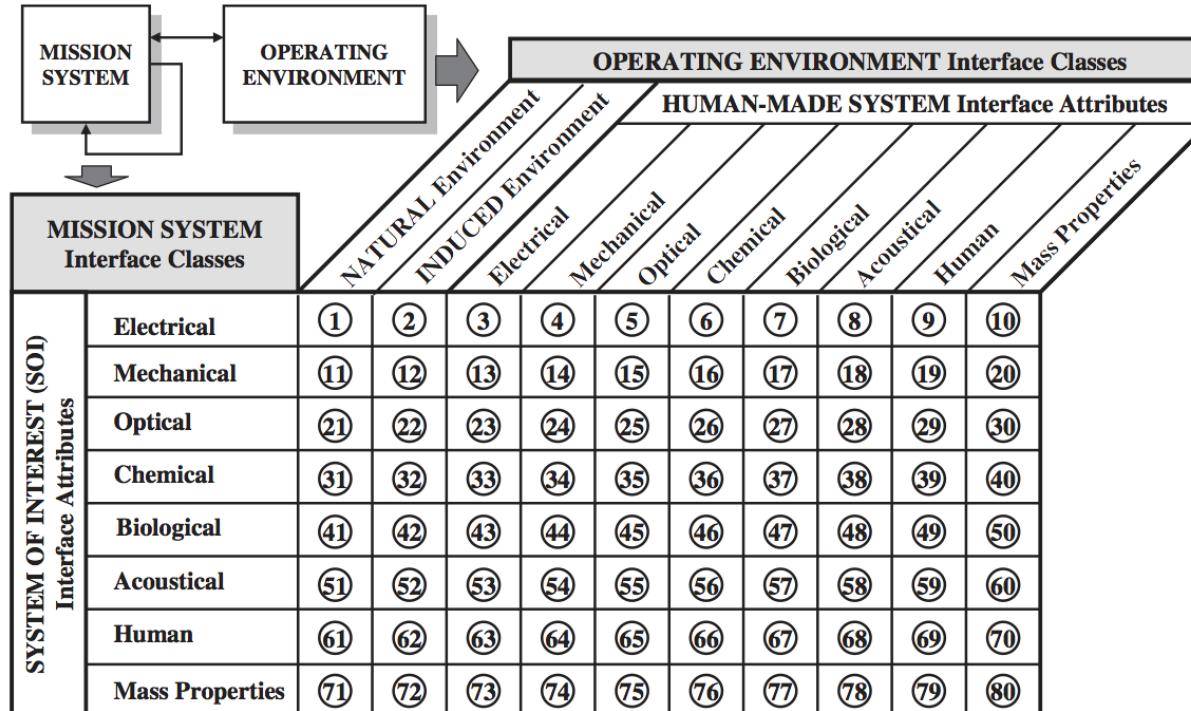
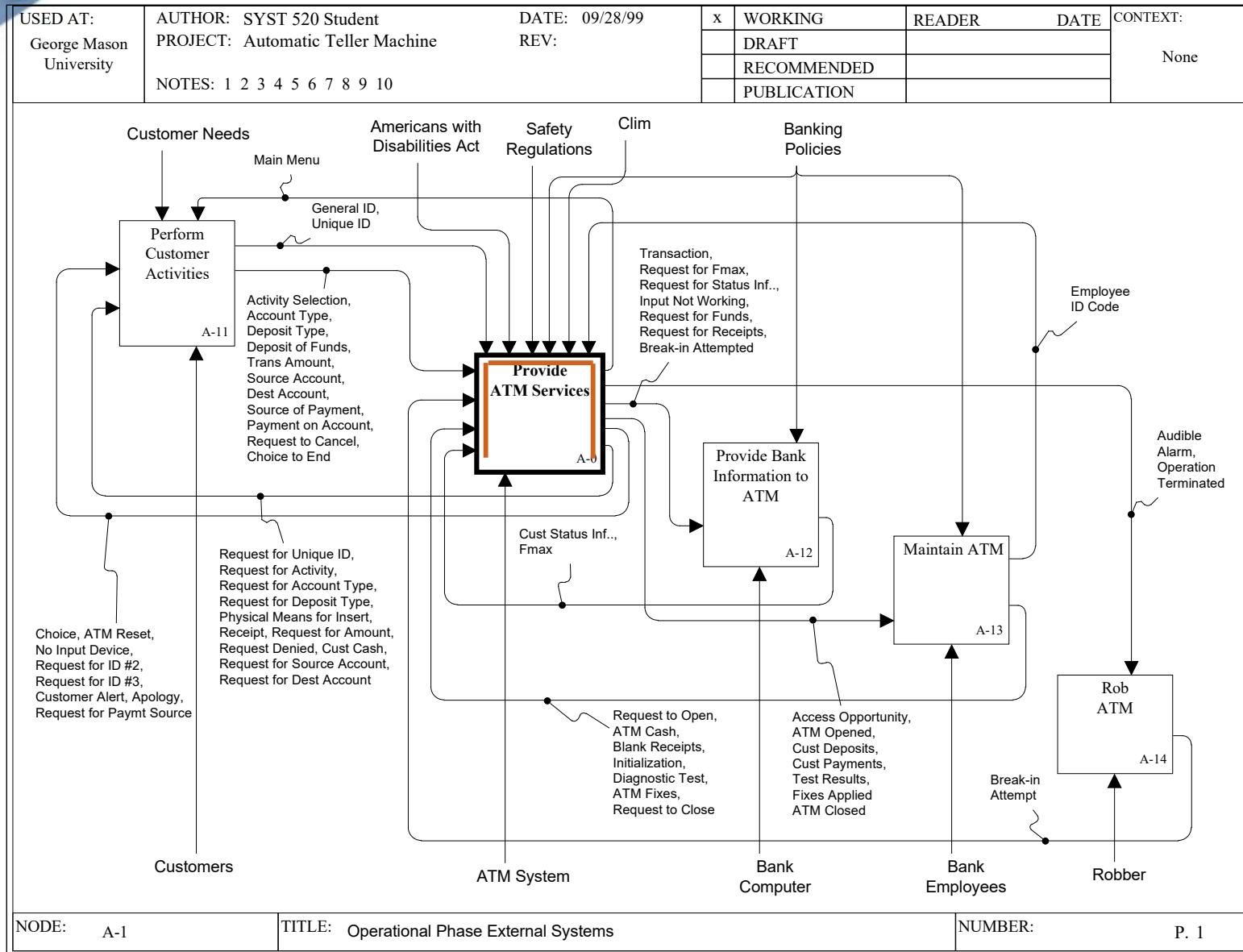


Figure 43.1 SYSTEM OF INTEREST (SOI) Interface Interactions Analysis Matrix

- Consider interactions between system and environment.
- Often focus on a few, not all interactions
- Environmental effects are natural, induced, and human-made.

# Where are the Interfaces?



At:  
**Inputs**  
**Controls**  
**Outputs**

# Interface Design Process

The same SE process !!

- Define Interface Requirements
- Identify the Items to Be Transported by the Interface
  - Define the Operational Concept
- Bound the Problem with an External Systems Diagram
  - Define the Objectives Hierarchy
  - Write the Requirements
- Select a High-Level Interface Architecture of Interface
  - Identify Several Candidate Architectures
  - Define Trial Interfaces for Each Candidate
  - Evaluate Alternatives against Requirements
  - Choose High-Level Interface Architecture
- Develop Functional Architecture of Interface
  - Specify Functional Decomposition
  - Add Inputs and Outputs
  - Add Fault Detection and Recovery Functions
- Develop Physical Architecture of Interface
- Identify Candidates based upon High-Level Architecture
  - Eliminate Infeasible Candidates
- Develop Operational Architecture of Interface
- Allocate Functions to Components of the Interface
- Analyze Behavior and Performance of Alternatives
  - Select Alternative
- Document Design and Obtain Approval
- Add Functions to Components Connected to Interface as Needed



# Interface Template

| Functional or Physical Element | Signals | Input or Output | Interface Behavior | Key Interface Attributes<br>(Wasson 43.1) | SOA | Physical Description | Applicable Standards |
|--------------------------------|---------|-----------------|--------------------|---|-----|----------------------|----------------------|
|                                |         |                 |                    |   |     |                      |                      |
|                                |         |                 |                    |   |     |                      |                      |
|                                |         |                 |                    |   |     |                      |                      |
|                                |         |                 |                    |   |     |                      |                      |
|                                |         |                 |                    |   |     |                      |                      |
|                                |         |                 |                    |   |     |                      |                      |
|                                |         |                 |                    |   |     |                      |                      |
|                                |         |                 |                    |   |     |                      |                      |
|                                |         |                 |                    |   |     |                      |                      |



# Interface Criticality

- Systems often fail at the Interfaces
- Systems Interfaces are uniquely determined by and owned by the Systems Engineers



# *Impact of Standards*

# Standards

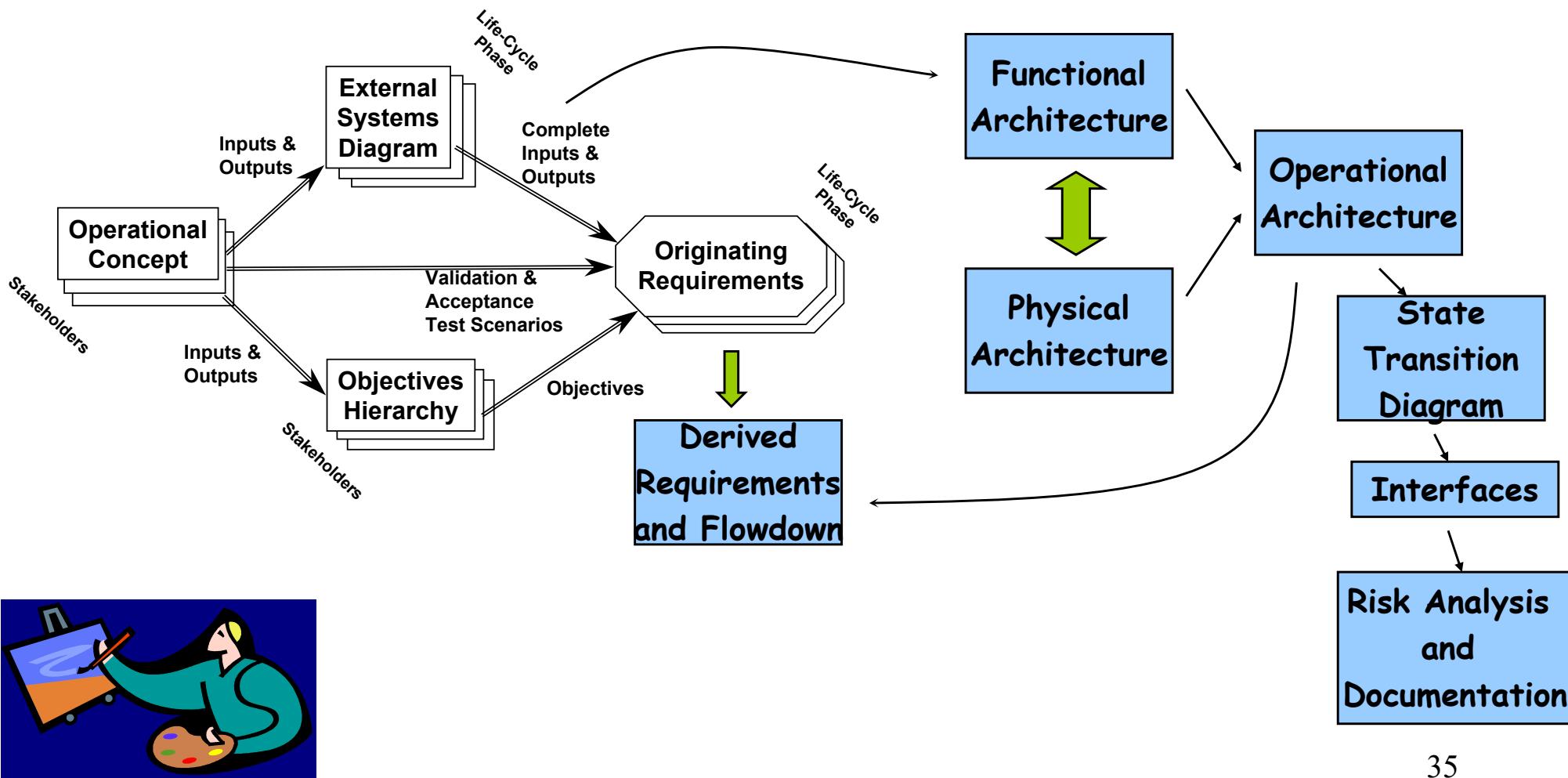
- **Formal standards** are negotiated and promulgated by accredited standards bodies, such as
  - International Organization for Standards (ISO)
  - International Telecommunications Union (ITU)
  - American National Standards Institute (ANSI)
- **De jure standards** are mandated by legal or other authorities
- **De facto standards** come into existence without any formal process via popular usage

## Benefits of Standards

- **Interchangeability** - ability to interchange components with different performance and cost characteristics
- **Interoperability** - the system can now operate with a wider variety of external systems, systems that have also adopted the same conventions
- **Portability** - systems can be moved and operate on other systems
- **Reduced cost and risk** for equivalent performance
- **Increased life cycle** is possible when long-lived standards are adopted

- Allocating Functions to Subsystems

# The Big Picture



# Operational Architecture

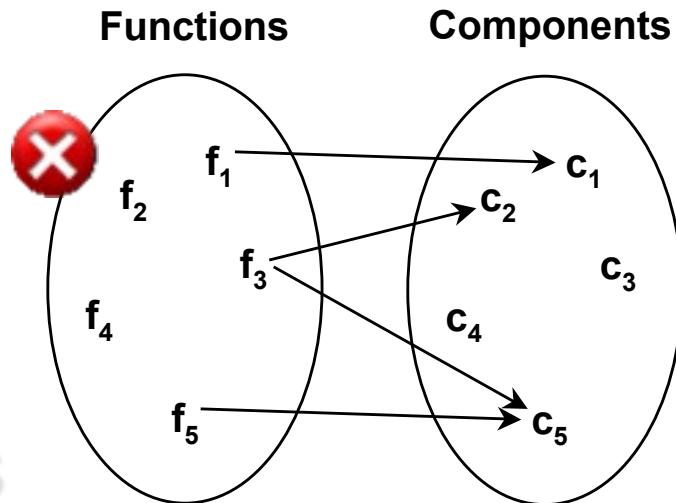
- Completes the functional to physical transition.
- Benefit of making major design decisions early : manageable blocks, chance of success, rapid development.
- Leave time for redesign, rework.

# Operational Architecture

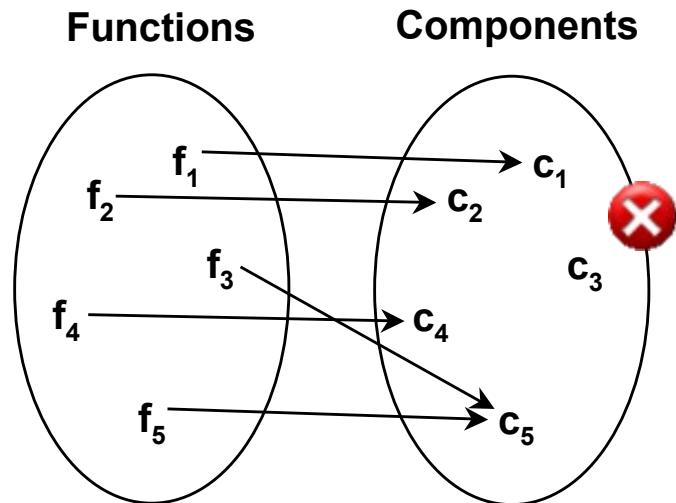
Provides a complete description of the system design including :

- Functional Architecture allocated to the Physical Architecture
- Derived I/O, Tech and Sys Wide, Trade Off, and Qualification requirements for each component
- Interface Architecture
- Complete Documentation

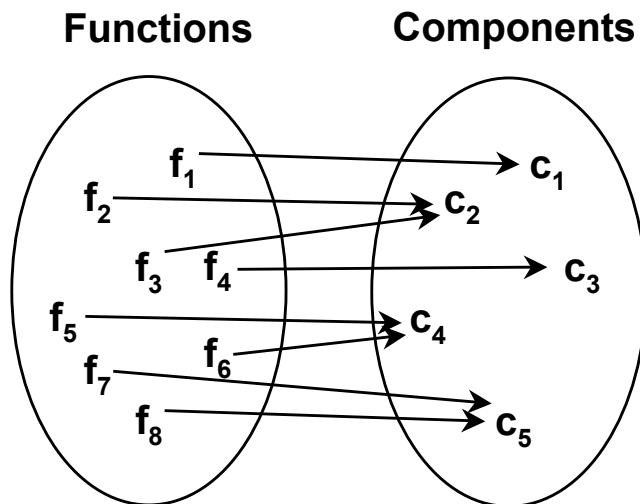
# Allocate Functions to Components



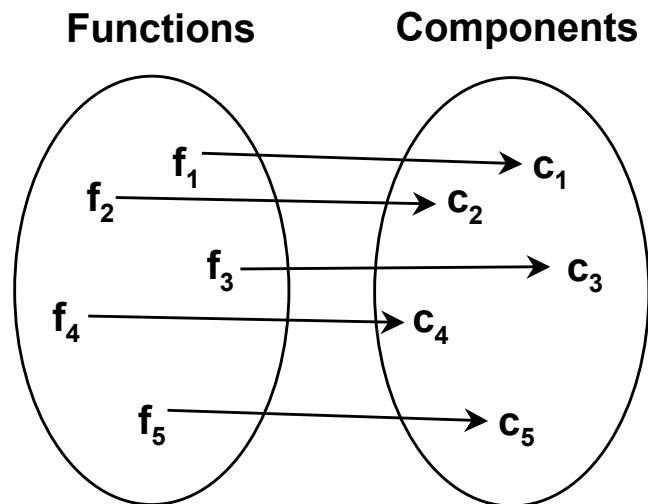
**Relation for the allocation of functions to components**



**Function for the allocation of functions to components**



**Onto, but not one-to-one function for the allocation of functions to components**



**One-to-one and onto function for the allocation of functions to components**

# Allocation Is Multi-objective Optimization Problem

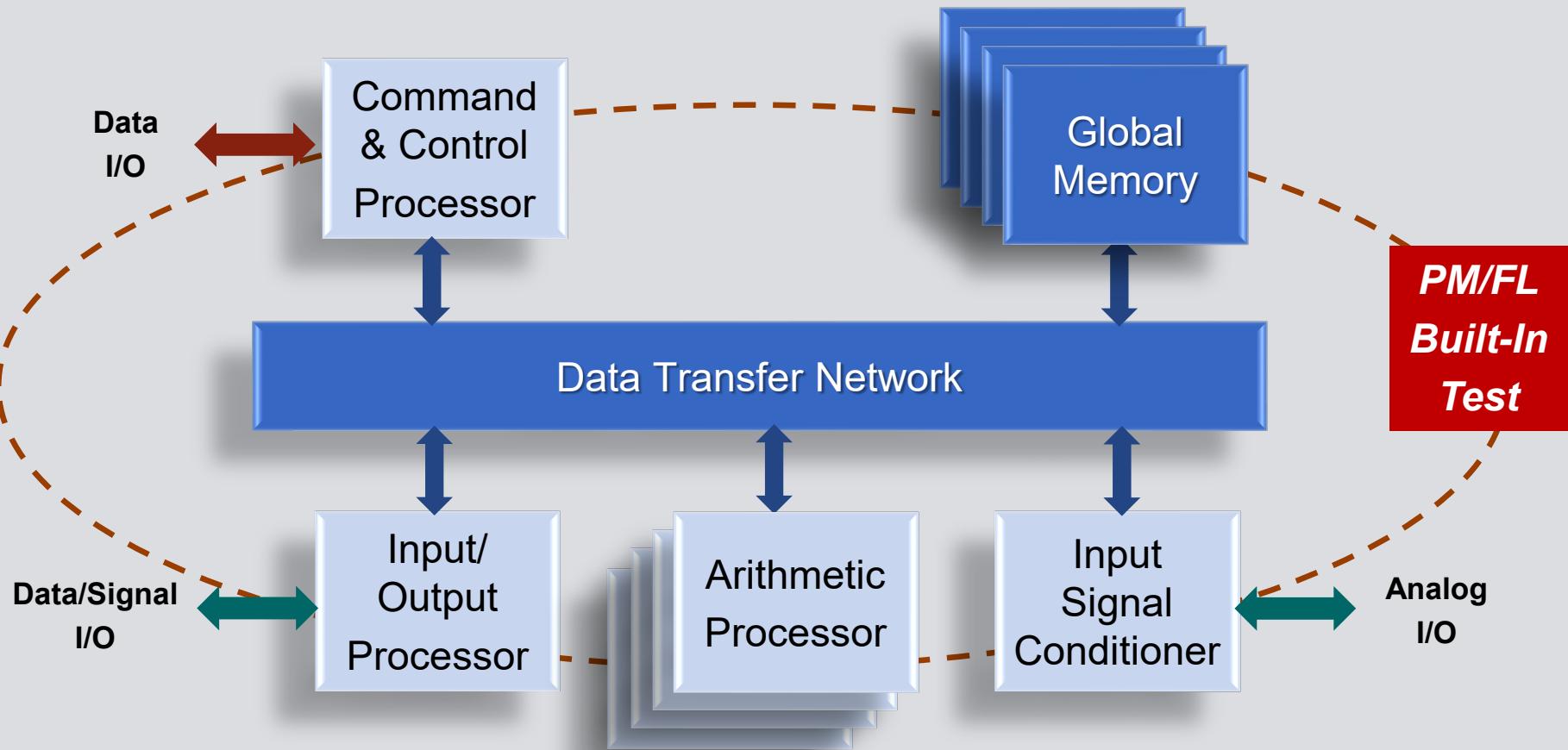
- Maximize the fundamental objectives
- Minimize the number and complexity of interfaces – *modularization*
- Maximize early critical testing opportunities - *risk minimization*
  - Equalizing risks
  - Localizing risks

# Allocating Functions to Subsystems

EXAMPLE

Subsystems may include:  
**Hardware, Software, Interfaces, Platforms, Standards**

## Enhanced Modular Signal Processor

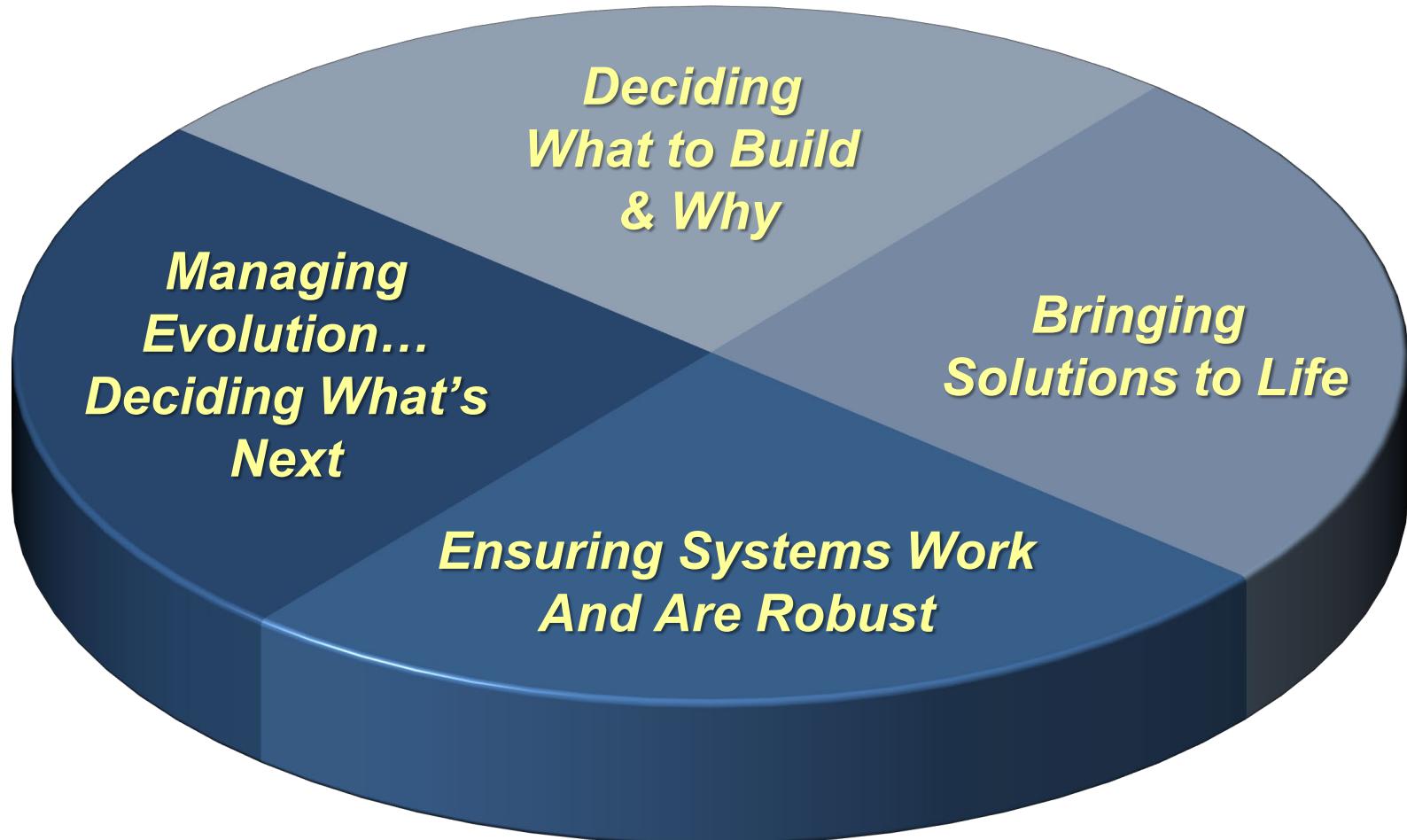


# Implementing the Functions Key Questions:

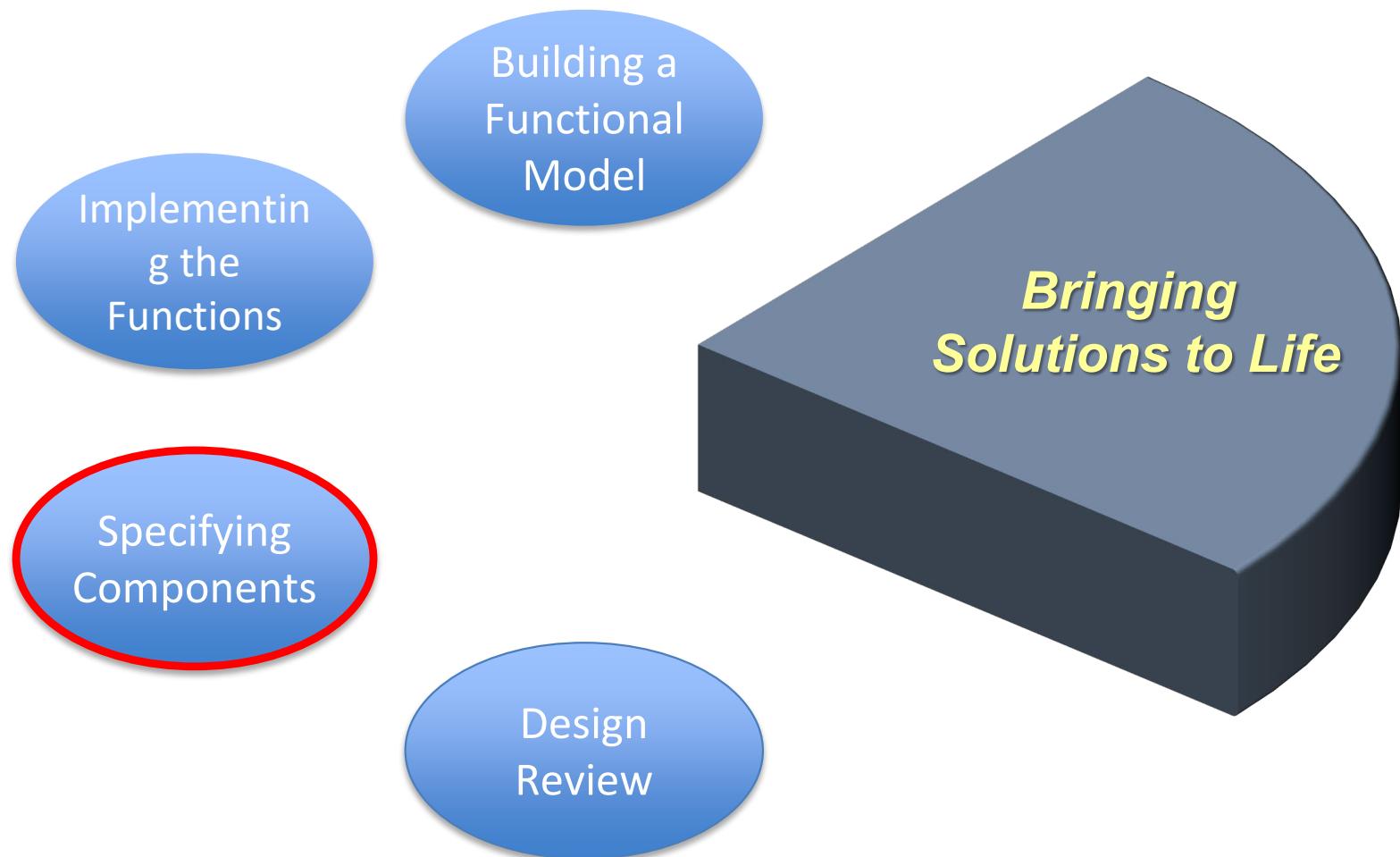
1. What are the major subsystems of your system and how do they map to the functions the system must perform?
  - *Assessment Criteria: Subsystems defined and mapped to the top-level functional architecture for the System Lifecycle.*
2. What are the key interfaces between the subsystems and between them and the external systems with which the system interacts?
  - *Assessment Criteria: External and internal interfaces defined and mapped to corresponding external and internal inputs and outputs.*

# Specifying Components

# Course Design



# Course Design



# Class Schedule (1 of 2)

| Week | Topic   |
|------|---|
| 1    | <ul style="list-style-type: none"><li>• Thinking in Terms of Systems</li></ul> <p><i>Deciding What to Build and Why</i></p>                                     |
| 2    | <ul style="list-style-type: none"><li>• Defining the Problem</li></ul>  |
| 3    | <ul style="list-style-type: none"><li>• Developing a Solution</li></ul>   |
| 4    | <ul style="list-style-type: none"><li>• Formulating a Proposal</li></ul> <p><i>Bringing Solutions to Life</i></p>   |
| 5    | <ul style="list-style-type: none"><li>• Building a Functional Model<ul style="list-style-type: none"><li>• (Concept Review Storyboards due)</li></ul></li></ul> |
| 6    | <ul style="list-style-type: none"><li>• Concept Review</li></ul>  |
| 7    | <ul style="list-style-type: none"><li>• Implementing the Functions</li></ul>  |
| 8    | <ul style="list-style-type: none"><li>• Specifying Components</li></ul>   |
| 9    | <ul style="list-style-type: none"><li>• Design Review</li></ul>   |



# Class Schedule (2 of 2)

| Week | Topic  |
|------|--|
|      | <i>Ensuring the System Works and Is Robust</i>   |
| 10   | <ul style="list-style-type: none"><li>• Integration and Test</li></ul>                           |
| 11   | <ul style="list-style-type: none"><li>• Ensuring Systems Work and Are Robust (M&amp;S)</li></ul> |
| 12   | <ul style="list-style-type: none"><li>• Designing for the Lifecycle</li></ul>                    |
| 13   | <ul style="list-style-type: none"><li>• Test Readiness Review</li></ul>                          |
|      | <i>Managing Evolution...Deciding What's Next</i>   |
| 14   | <ul style="list-style-type: none"><li>• Technology and Innovation</li></ul>                      |
| 15   | <ul style="list-style-type: none"><li>• Final Project Submission</li></ul>                       |

# Specifying Components

## Key Questions:

1. What are the key requirements for each of your major subsystems?
2. What tradeoffs were made in deriving your subsystem requirements?
3. What are the most important technical risks for your system and how will you mitigate each?

# Complex systems? How Complex Can it be?

15 min

- Think about an airplane United Flight 1230, which is planned to take off tomorrow at 9.30 pm.
- What are the factors ( think from system perspective) affecting on time performance of this system? In other words, list the factors which will make this flight to take off on time?

# *Part 1: Requirements Traceability and Flowdown*

# Requirements Flowdown

**Given:** A complete set of system requirements:

- Input/output requirements
- Functional requirements
- Performance requirements
- Lifecycle requirements
- Cost requirements
- Schedule requirements
- Test system requirements

**Develop:** A complete set of subsystem requirements for each subsystem (and ultimately, for each component)

- Input/output requirements
- Functional requirements
- Performance requirements
- Lifecycle requirements
- Cost requirements
- Schedule requirements
- Test system requirements

# Trace System-wide Requirements and Derive Subsystem-wide Requirements

- Trace system-wide/technology requirements to system
- For each system-wide/technology requirement, derive subsystem-wide requirements for each subsystem
- Trace each derived subsystem requirement to the appropriate subsystem



# Requirement Types

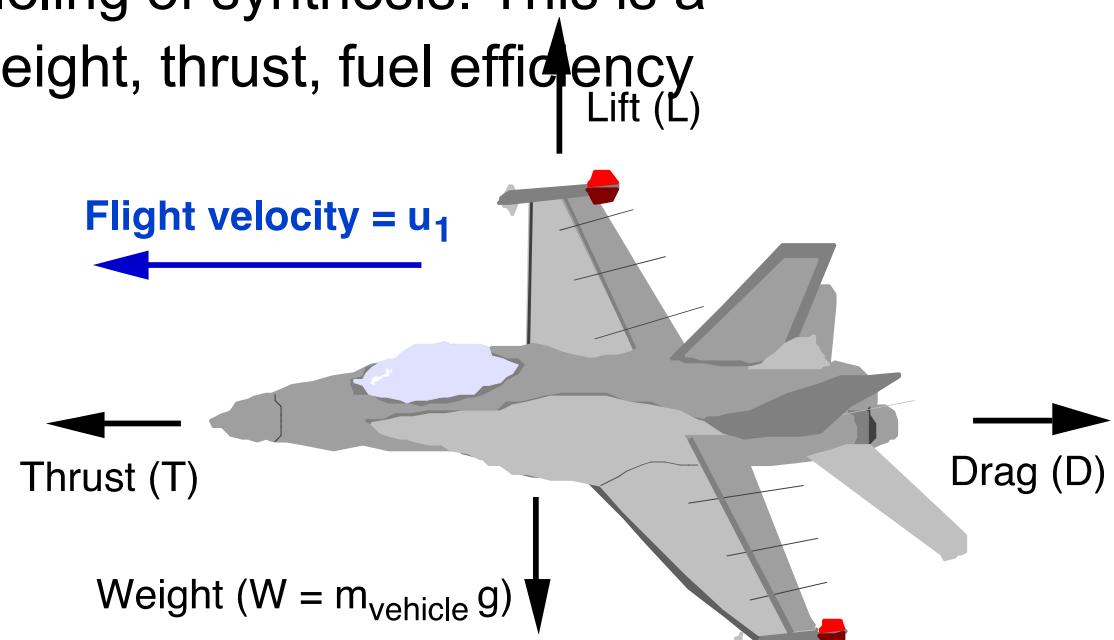
- **Input/output requirements** - what inputs must the system respond to and what outputs it must it produce in response
- **Functional requirements** - what the system must do; the tasks it must perform
- **Performance requirements** - what range of inputs must be accommodated and what quality attribute must the outputs possess (how well, how fast, how many, etc.)
- **Lifecycle requirements** - what additional attributes must the system demonstrate over its lifetime
- **Test system requirements** - what resources will we need in order to verify that the system meets its requirements

# Requirements Flowdown

- For each system requirement, derive subsystem requirements for each subsystem:
  - **Equivalence** is a simple flowdown technique that causes the subsystem requirement to be the same as the system requirement
  - **Apportionment** spreads a system-level requirement among the system's subsystems of the system, maintaining the same units, e.g., cost, reliability
  - **Synthesis** addresses those situations in which the system-level requirement is driven by complex contributions from several subsystems, requiring subsystem requirements flowed down from the system to be based upon some analytic model

# Synthesis Example: Breguet Range Equation

- A jetliner shall have an operational range of 6000 nautical miles.
- How would we flow this requirement down to the engine , airframe and fuel subsystem?
- Let's see the engineering modeling of synthesis. This is a conversation about lift, drag, weight, thrust, fuel efficiency and fuel capacity.
- It is complicated since aircraft has to carry its fuel which adds to its weight..



# Synthesis Example: Breguet Range Equation

- Consider aircraft in **level flight** (Lift = weight) at **constant flight velocity**  $u_1$  (thrust = drag)

$$L = m_{\text{vehicle}} g,$$

$$D = \text{Thrust} = \frac{\eta_o \dot{m}_{\text{fuel}} Q_R}{u_1}$$

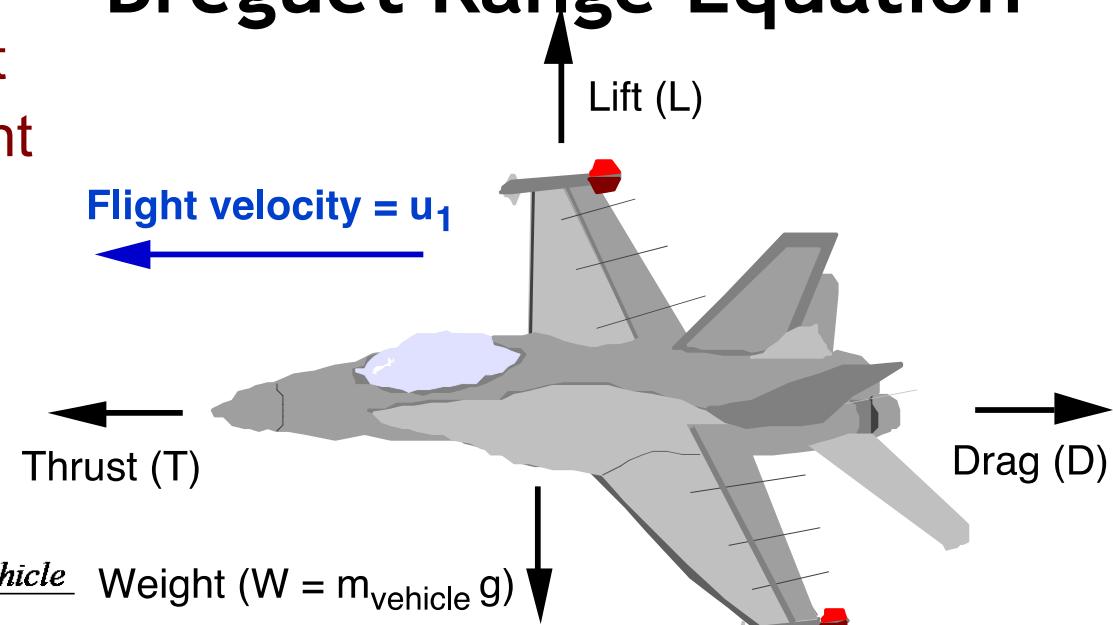
$$= \frac{\eta_o Q_R}{u_1} \frac{dm_{\text{fuel}}}{dt} = \frac{\eta_o Q_R}{u_1} \frac{-dm_{\text{vehicle}}}{dt}$$

- Combine expressions for lift & drag and integrate from time  $t = 0$  to  $t = R/u_1$  ( $R$  = range = distance traveled), i.e. time required to reach destination, to obtain **Breguet Range Equation**

$$\frac{D}{L} \frac{u_1 g}{\eta_o Q_R} dt = -\frac{dm_{\text{vehicle}}}{m_{\text{vehicle}}} \Rightarrow \int_0^{R/u_1} \frac{D}{L} \frac{u_1 g}{\eta_o Q_R} dt = - \int_{\text{initial}}^{\text{final}} \frac{dm_{\text{vehicle}}}{m_{\text{vehicle}}}$$

14       $\Rightarrow \frac{D}{L} \frac{u_1 g}{\eta_o Q_R} \frac{R}{u_1} = -\ln \frac{m_{\text{final}}}{m_{\text{initial}}} \Rightarrow$

$$R = \frac{L}{D} \frac{\eta_o Q_R}{g} \ln \frac{m_{\text{initial}}}{m_{\text{final}}}$$



Ref: USC Department of  
Mechanical Engineering

# Trace Qualification Requirements and Derive Subsystem Qualification Requirements

- Trace qualification requirements to system
- Derive qualification requirements for each subsystem
- Define at what point qualification will take place
- More on qualification later

# Technology and System-Wide Requirements and ‘Flowdown’

- We may have the following technology and system-wide requirements:
  - The system shall be blue,
  - The system shall weigh 100 Newtons,
  - The system shall achieve a top speed of 80 kph.
- How are these applied to subsystems ??

# Ensure All Requirements Can Be Verified

| Customer Requirements including Business Processes | System Requirements including Business Processes Changes | Final System Acceptance Criteria | FVT | CVT | TVT | ASCA | SIT | PST | UAT | Pre-Prod | Pilot |
|--|--|----------------------------------|-----|-----|-----|------|-----|-----|-----|----------|-------|
| C1   | SR 1.1   |                                  |     |     |     |      |     |     |     |          |       |
|  | BP 1.2   |                                  |     |     |     |      |     |     |     |          |       |
|  | SR 1.3   |                                  |     |     |     |      |     |     |     |          |       |
| C2   | SR 2.1   |                                  |     |     |     |      |     |     |     |          |       |
| C3   | SR 3.1   |                                  |     |     |     |      |     |     |     |          |       |
|  | SR 1.2   |                                  |     |     |     |      |     |     |     |          |       |
|  | BP 4.1   |                                  |     |     |     |      |     |     |     |          |       |
|  | SR 4.2   |                                  |     |     |     |      |     |     |     |          |       |
| C4   | SR 4.1   |                                  |     |     |     |      |     |     |     |          |       |
|  | BP 4.2   |                                  |     |     |     |      |     |     |     |          |       |

FVT - Functional Verification Test

CVT - Component Verification Test

TVT - Translation Verification Test

ASCA - Applications System Control & Auditability

SIT - System Integration Test

PST - Performance Stress Test

UAT - User Acceptance Test

Pre-Prod - Pre-Production Test

Legend: Enter one or more of the following in each Test Method cell  
(a blank cell indicates that the Test Method is not required for that requirement):

A = Analysis

D = Demonstration

I = Inspection

S/M = Simulation and Modeling

T = Test

# Ensure All Requirements Can Be Verified

| Customer Requirements including Business Processes       | System Requirements including Business Processes Changes | Final System Acceptance Criteria | FVT | CVT | TVT | ASCA | SIT | PST | UAT | Pre-Prod | Pilot |
|--|--|----------------------------------|-----|-----|-----|------|-----|-----|-----|----------|-------|
| C1   | SR 1.1   |                                  |     |     |     |      |     |     |     |          |       |
|  | BP 1.2   |                                  |     |     |     |      |     |     |     |          |       |
|  | SR 1.3   |                                  |     |     |     |      |     |     |     |          |       |
| System Requirements including Business Processes Changes | Component Requirements                                   | Acceptance Criteria              | FVT | CVT | TVT | ASCA | SIT | PST | UAT | Pre-Prod | Pilot |
| SR 1.1   | CO 1.1.1   |                                  |     |     |     |      |     |     |     |          |       |
|  | CO 1.1.2   |                                  |     |     |     |      |     |     |     |          |       |
|  | CO 1.1.3   |                                  |     |     |     |      |     |     |     |          |       |
| BP 1.2   | CO 2.1   |                                  |     |     |     |      |     |     |     |          |       |
| BP 4.Z   |  |                                  |     |     |     |      |     |     |     |          |       |

FVT - Functional Verification Test

CVT - Component Verification Test

TVT - Translation Verification Test

ASCA - Applications System Control & Auditability

SIT - System Integration Test

PST - Performance Stress Test

UAT - User Acceptance Test

Pre-Prod - Pre-Production Test

Legend: Enter one or more of the following in each Test Method cell  
(a blank cell indicates that the Test Method is not required for that requirement):

A = Analysis

D = Demonstration

I = Inspection

S/M = Simulation and Modeling

T = Test

# Verification Methods

| Method                      | Description  | Used During:   | Most Effective When:  |
|-----------------------------|--|--|---|
| Inspection (static test)    | Compare system attributes to requirements.   | During all segments of verification, validation, and acceptance testing for requirements that can be addressed by human examination.             | Success or failure can be judged by humans; examples include inspection of physical attributes, code walkthroughs, and evaluation of user's manuals.  |
| Analysis and simulation     | Use models that represent some aspect of the system. Examples of models might address system's environment, system process, system failures. | Used throughout qualification, but emphasis is early in verification and during acceptance.<br><br>Often used in conjunction with demonstration. | Physical elements are not yet available.<br>Expense prohibits instrumented test, and demonstration is not sufficient.<br><br>Issue involves all or most of the system's life span.<br><br>Issue cannot be tested (e.g., survive nuclear blast). |
| Instrumented test           | Use calibrated instruments to measure system's outputs.<br><br>Examples of calibrated instruments are oscilloscope, voltmeter, LAN analyzer. | Verification testing.  | Engineering test models through system elements are available.<br><br>Detailed information is required to understand and trace failures.<br><br>Life and reliability data is needed for analysis and simulation.                                |
| Demonstration or field test | Exercise system in front of unbiased reviewers in expected system environment.   | Primarily used for validation and acceptance testing.  | Complete instrumented test is too expensive.<br><br>High-level data/information is needed to corroborate results from analysis and simulation or instrumented test.   |

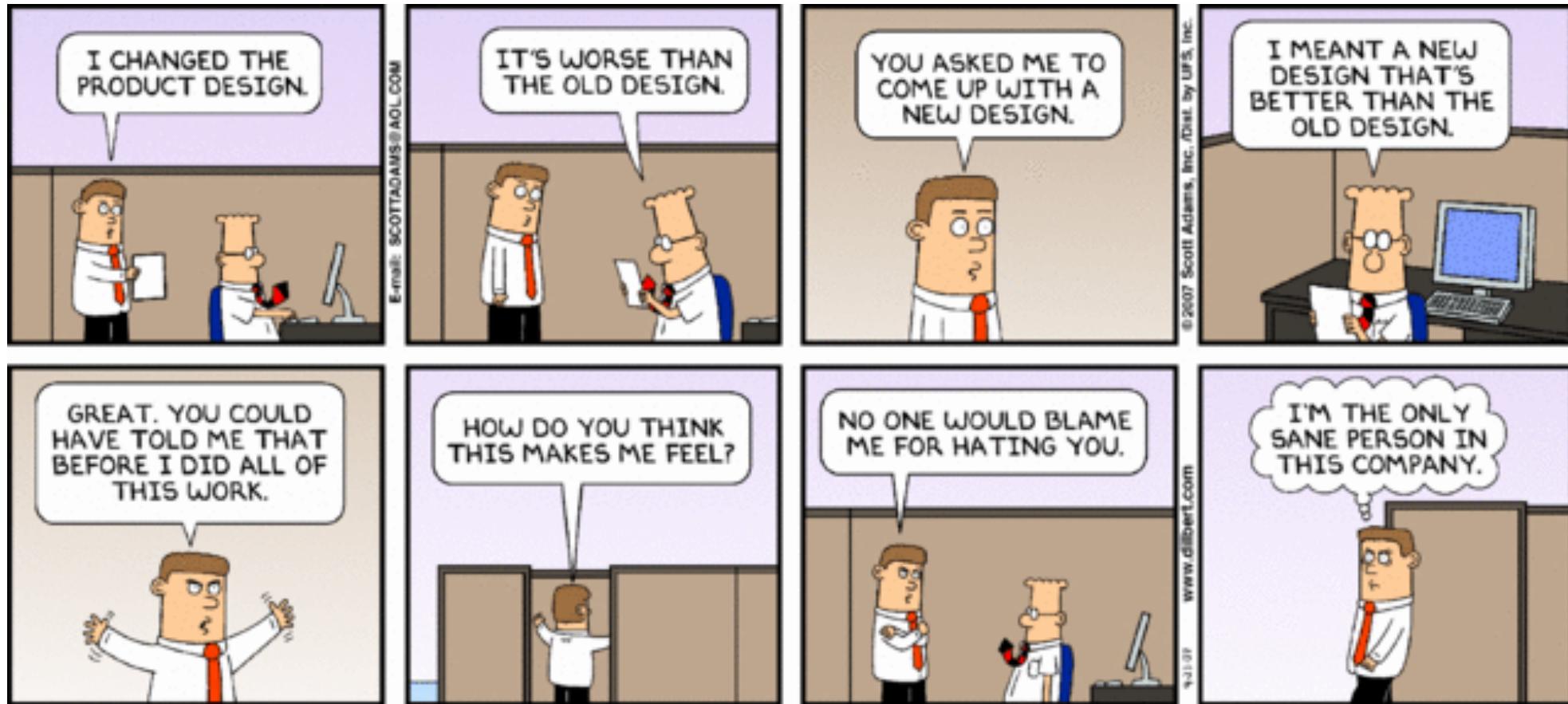


## *Part 2: Trade Studies*

# What is a Trade Study?

- A **trade study** (or trade-off study) is a formal tool that supports decision making.
- A **trade study** is an objective comparison with respect to performance, cost, schedule, risk, and all other reasonable criteria of all realistic alternative requirements; architectures; baselines; or design, verification, manufacturing, deployment, training, operations, support, or disposal approaches.
- A **trade study** documents the requirements, assumptions, criteria and priorities used for a decision. This is useful since new information frequently arises and decisions are re-evaluated.

# The Importance of Trade Studies



# Trade Studies Support Decision Making Throughout the Development Lifecycle

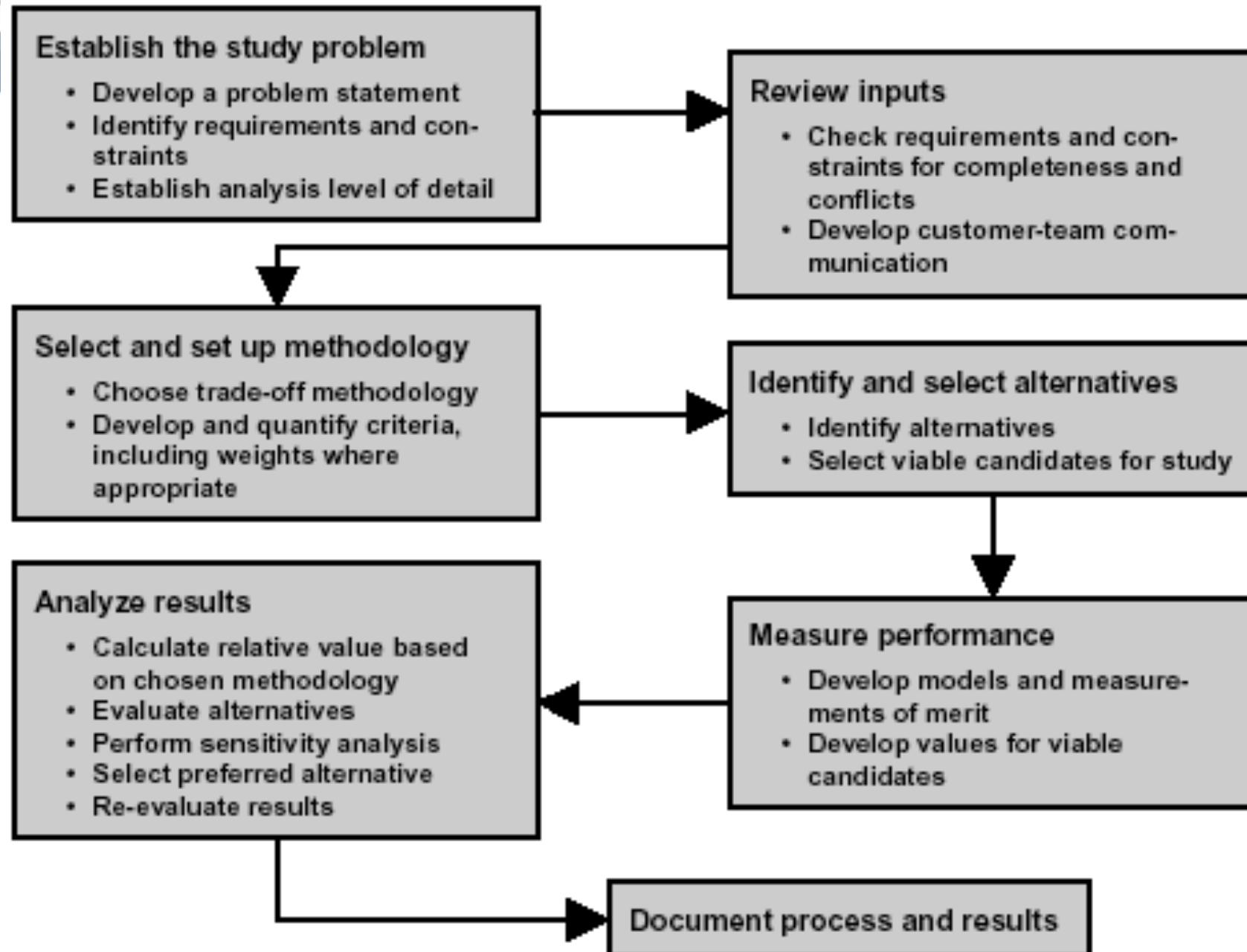
Trade studies support:

- Requirements development - e.g., to resolve conflicts; to resolve TBDs and TBRs
- Functional allocations - e.g., system architecture development
- System synthesis - e.g., assess the impact of alternative performance or resource allocations
- Investigate alternate technologies for risk or cost reduction
- Assess proposed design changes
- Make/buy decisions (i.e., build the part from a new design or buy from commercial, existing sources)

# Trade Study Process

- **Frame the Problem** - What decision do we need to make?
- **Generate Alternatives** - How many different ways can we produce the desired result?
- **Identify Decision Criteria** - What factors are important in comparing different options?
- **Assign Weights** - What is the relative importance of each of the criteria?
- **Evaluate the Alternatives** -
  - Determine how well each option performs against each of the criteria
  - Multiply the relative performance by the corresponding weight
  - Sum the results across the criteria
- **Select the Preferred Option** - How?

# The Trade Study Process (2/2)



# Frame the Problem

## Trade Studies made during the SE Process

- Resolve conflicting customer requirements.
- Develop requirements and acceptance criteria to define how the system will satisfy the customer requirements.
- Prioritize requirements and assess their cost and schedule implications with the customer
- Make architecture/requirements allocation decisions.
  - Build versus buy decisions.
  - Design decisions to the cost objectives.
  - Design decisions to meet the schedule objectives.
  - Decisions whether to perform functionality in software or hardware.
  - Decisions to reuse existing components or not.
  - Decisions whether to have a distributed solution or a centralized one.
  - Decisions on whether to use a network or point-to-point interfaces.
  - Decisions on whether to allocate a customer service to one configuration item or a different one.
- Decide whether to deliver all of the customer's requirements in one delivery or create multiple deliveries.
- Develop the maintenance concept for the system - levels of maintenance, where to store spare parts, what level of technical refresh program to develop.

# Evaluation Criteria – Measures (1/2)

- Trade studies depend upon having criteria for making decisions based on *measures of effectiveness* (voice of the customer) and *measures of performance* (voice of the engineer).
- **Measure of Effectiveness (MOE)** - A measure of how well mission objectives are achieved. MOEs are implementation independent - they assess ‘how well’ not ‘how’.
- Example measures of effectiveness include
  - Life cycle cost
  - Schedule, e.g., development time, mission duration
  - Technology readiness level (maturity of concept/hardware)
  - Crew capacity
  - Payload Mass

# Generate Alternatives

“Alternatives are the raw material of decision making”

-Smart Choices by Hammond, Keeney & Raiffa

After the problem has been framed, ask:

“How can we obtain the desired outcome?”

- Challenge constraints - look at the problem from new angles
- Be creative, let the process diverge
  - Gather information, if necessary
- Withhold judgment until the evaluation phase

How do we ensure that we are not just considering the same old alternatives while falling into a “status quo” trap?

# Generate Alternatives

|  |  |  |  |  |  |
|--|--|--|--|--|--|
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

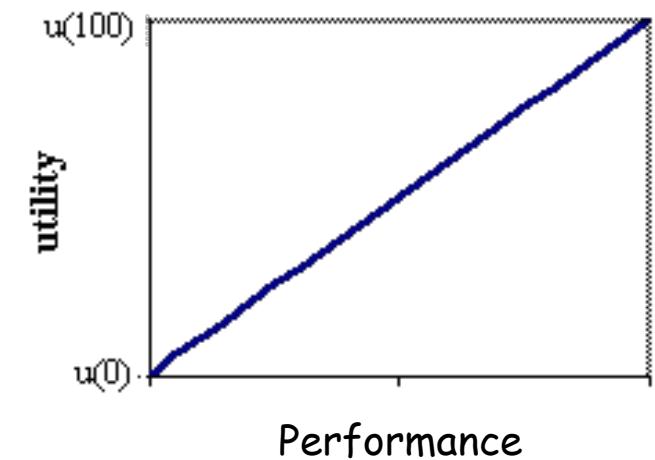
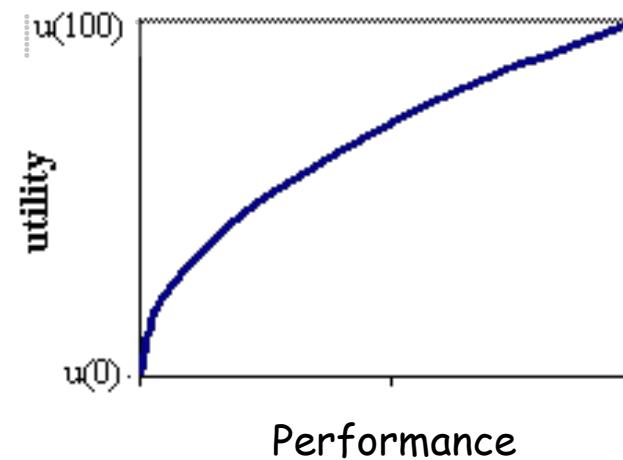
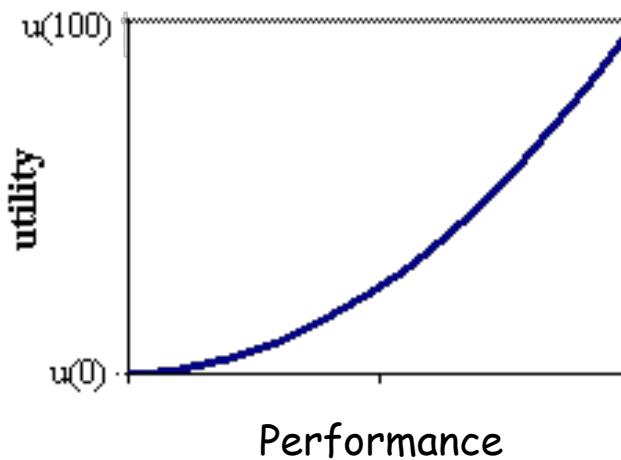
# Generate Alternatives

|  |  | Alternatives |   |   |   |
|--|--|--------------|---|---|---|
|  |  | A            | B | C | D |
|  |  |              |   |   |   |
|  |  |              |   |   |   |
|  |  |              |   |   |   |
|  |  |              |   |   |   |
|  |  |              |   |   |   |
|  |  |              |   |   |   |

# Identify Decision Criteria

|          |  | Alternatives |   |   |   |
|----------|--|--------------|---|---|---|
| Criteria |  | A            | B | C | D |
| 1        |  |              |   |   |   |
| 2        |  |              |   |   |   |
| 3        |  |              |   |   |   |
| 4        |  |              |   |   |   |
| 5        |  |              |   |   |   |
|          |  |              |   |   |   |

# Evaluate the Alternatives - The Utility Function



# Assign Weights

|          |         | Alternatives |   |   |   |
|----------|---------|--------------|---|---|---|
| Criteria | Weights | A            | B | C | D |
| 1        | 0.20    |              |   |   |   |
| 2        | 0.15    |              |   |   |   |
| 3        | 0.30    |              |   |   |   |
| 4        | 0.25    |              |   |   |   |
| 5        | 0.10    |              |   |   |   |
| Total    | 1.00    |              |   |   |   |

# Evaluate the Alternatives

|          |         | Alternatives |   |   |   |
|----------|---------|--------------|---|---|---|
| Criteria | Weights | A            | B | C | D |
| 1        | 0.20    |              |   |   |   |
| 2        | 0.15    |              |   |   |   |
| 3        | 0.30    |              |   |   |   |
| 4        | 0.25    |              |   |   |   |
| 5        | 0.10    |              |   |   |   |
| Total    | 1.00    |              |   |   |   |

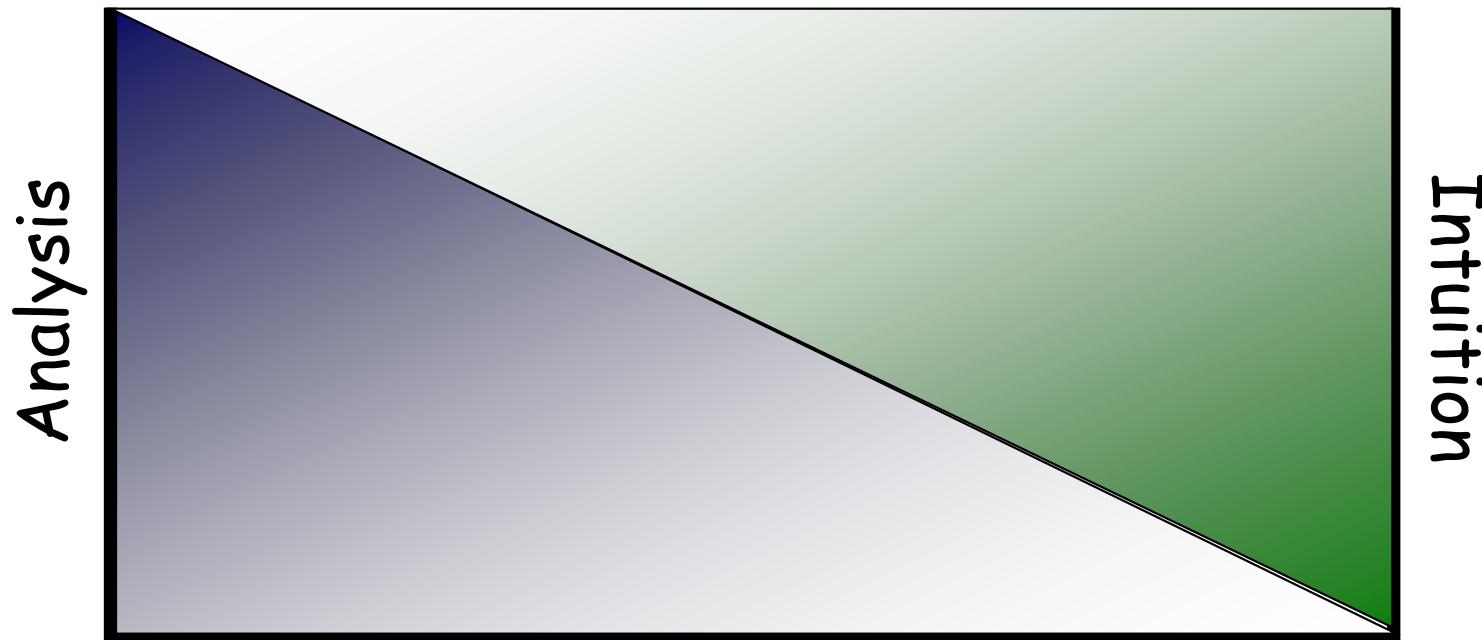
# Evaluate the Alternatives

|          |         | Alternatives |      |      |      |
|----------|---------|--------------|------|------|------|
| Criteria | Weights | A            | B    | C    | D    |
| 1        | 0.20    | 25           | 84   | 77   | 65   |
| 2        | 0.15    | 17           | 73   | 54   | 32   |
| 3        | 0.30    | 68           | 91   | 36   | 50   |
| 4        | 0.25    | 100          | 22   | 48   | 46   |
| 5        | 0.10    | 53           | 40   | 37   | 76   |
| Total    | 1.00    | 58.3         | 64.6 | 50.0 | 51.9 |

# Select the Preferred Option How?

|          |         | Alternatives |      |      |      |
|----------|---------|--------------|------|------|------|
| Criteria | Weights | A            | B    | C    | D    |
| 1        | 0.20    | 25           | 84   | 77   | 65   |
| 2        | 0.15    | 17           | 73   | 54   | 32   |
| 3        | 0.30    | 68           | 91   | 36   | 50   |
| 4        | 0.25    | 100          | 22   | 48   | 46   |
| 5        | 0.10    | 53           | 40   | 37   | 76   |
| Total    | 1.00    | 58.3         | 64.6 | 50.0 | 51.9 |

# Select the Preferred Option How?



*What role do analysis and intuition play in engineering decisions?*

# Trade Study Heuristics

1. Rules of Thumb:
  - ✓ Manage the number of options under consideration
  - ✓ Revisit the original problem statement
  - ✓ If a baseline solution is established, use it as a ‘yardstick’ to measure the alternatives.
2. Decisions are frequently made with imperfect information.
  1. Do not get stuck in ‘analysis paralysis’.
  2. Decide how deep the analysis must go. {Deep enough to make a decision with confidence, but no deeper.}
3. Does the decision feel right? If not, why?
4. Conduct further what-if scenarios by changing assumptions.
5. Reject alternatives that do not meet an essential requirement.
6. Ignore evaluation criteria that do not discriminate between alternatives.
7. Trades are usually subjective; numeric results usually give a false sense of accuracy.
8. If an apparent preferred option is not decisively superior, further analysis is warranted.

# Example Decision Matrix Trade Study

| Decision Matrix<br>Example for Battery |                      |        | ENTER SCORES                                  | Extend Old Battery Life | Buy New Batteries | Collect Experiment Data With Alternative Experiment | Cancelled Experiment |
|--|----------------------|--------|---|-------------------------|-------------------|---|----------------------|
| CRITERIA                               | Mandatory (Y=1/N=0)? | Weight | SCALE   |                         |                   |   |                      |
| Mission Success (Get Experiment Data)  | 1                    | 30     | 3 = Most Supportive<br>1 = Least Supportive   | 2                       | 3                 | 3   | 0                    |
| Cost per Option                        | 0                    | 10     | 3 = Least Expensive<br>1 = Most Expensive     | 1                       | 2                 | 3   | 1                    |
| Risk (Overall Option Risk)             | 0                    | 15     | 3 = Least Risk<br>1 = Most Risk               | 2                       | 1                 | 2   | 3                    |
| Schedule                               | 0                    | 10     | 3 = Shortest Schedule<br>1 = Longest Schedule | 3                       | 2                 | 1   | 3                    |
| Safety                                 | 1                    | 15     | 3 = Most Safe<br>1 = Least Safe               | 2                       | 1                 | 2   | 3                    |
| Uninterrupted Data Collection          | 0                    | 20     | 3 = Most Supportive<br>1 = Least Supportive   | 3                       | 1                 | 2   | 1                    |
| WEIGHTED TOTALS in %                   |                      | 100%   | 3   | 73%                     | 60%               | 77%   | 0%                   |

Preferred Solution

# ***Example Qualitative Decision Matrix For a Lunar Thermal Control Trade Study***

| <u>Characteristics</u>                                | <u>Single Phase Fluid</u> | <u>Two Phase Fluid</u> | <u>Heat Pipe</u> |
|---|---------------------------|------------------------|------------------|
| <u>Safety:</u> (3)<br>Operating Pressure              | Low                       | High                   | Low-Medium       |
| <u>Safety:</u><br>Toxicity                            | Fluid Dependent           | Fluid Dependent        | Fluid Dependent  |
| <u>Safety:</u><br>Flammability                        | Fluid Dependent           | Fluid Dependent        | Fluid Dependent  |
| <u>Reliability</u> (1)                                | High                      | Fair                   | Fair             |
| <u>Performance:</u> (2)<br>Pumping Cost               | Low                       | High                   | Fair             |
| <u>Complexity:</u> (4)<br>Controls                    | Simple                    | Nominal                | Complex          |
| <u>Complexity:</u> (5)<br>Manufacturing<br>Difficulty | Simple                    | Nominal                | Complex          |

*Key questions to ask:*

- Have the requirements and constraints truly been met?
- Is the tentative selection heavily dependent on a particular set of input values and assumptions, or does it hold up under a range of reasonable input values (i.e., is it 'robust')?
- Are there sufficient data to back up the tentative selection?
- Are the measurement methods sufficiently discriminating to be sure that the tentative selection is really better than the alternatives?
  - If close results, is further analysis warranted?
- Have the subjective aspects of the problem been fully addressed?
- Test the decision robustness.
  - Is the tentative selection very sensitive to an estimated performance or constraint? If so, explore the full reasonable range of each performance variable to understand the domain where your tentative selection is appropriate.



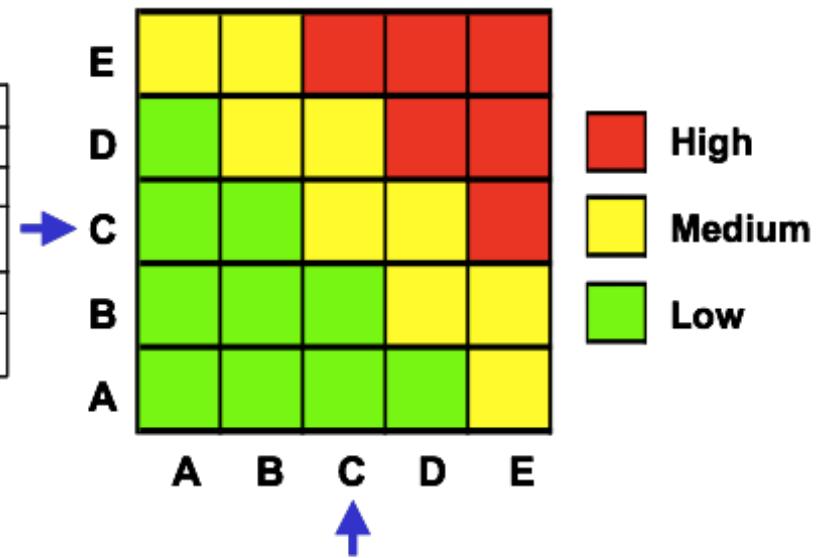
## *Part 3: Technical Risk*

# Technical Risk Management

- Technical risk management is an organized means of collecting and controlling project risks. This can involve qualitative and quantitative methods
- Risk
  - “...potential for a negative future reality that may or may not happen.” [DoD, 2001]
  - “...is characterized by the combination of the probability that a program or project will experience an undesired event... and the consequences, impact, or severity of the undesired event, were it to occur.” [NASA NPR 8000.4]
  - Is a vector consisting of a likelihood measure and a consequence measure

# Risk Management: The Classic “Stoplight” Matrix

| Probability of Occurrence   |       |
|---|-------|
| Design Difficulty   | Level |
| Requires breakthrough advance to meet requirements                    | E     |
| Substantial development is required                                   | D     |
| Moderate development is required to a new or existing item            | C     |
| Off-the-shelf item with minor modifications                           | B     |
| Qualified off-the-shelf item which meets all operational requirements | A     |



Consequence: Given the risk is realized, what is the magnitude of impact

| Cost          | Performance                             | Schedule   | Level |
|---------------|---|--|-------|
| ≥ 20%         | Unacceptable                            | Can't achieve key team or major program milestone        | E     |
| ≥ 15% - < 20% | Acceptable; no remaining margin         | Major slip in key milestone or critical path impacted    | D     |
| ≥ 10% - < 15% | Acceptable with medium margin reduction | Minor slip in key milestones, not able to meet need date | C     |
| ≥ 5% - < 10%  | Acceptable with small margin reduction  | Additional resources required, able to meet need date    | B     |
| ≥ 0% - < 5%   | Minimal or no impact                    | Minimal or no impact                                     | A     |



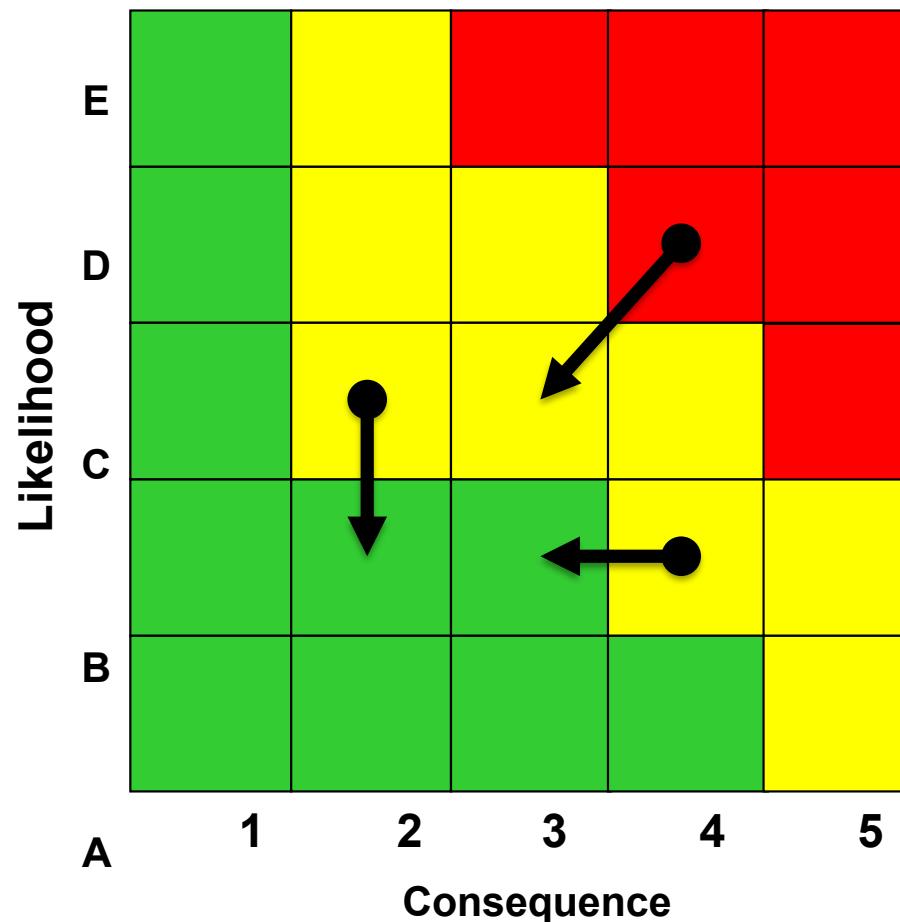
# There are many types of risk

| Risk type                     | Example  |
|-------------------------------|--|
| Technical or performance risk | Failure to meet a technical requirement or specification                   |
| Cost risk                     | Failure to stay within a cost cap  |
| Schedule risk                 | Failure to meet a critical milestone                                       |
| Programmatic risk             | Failure to secure long-term political support for a program or project     |
| Financial or market risk      | Failure of demand to materialize at a specified price                      |
| Liability risk                | Spacecraft de-orbits prematurely, causing damage over the debris footprint |
| Regulatory risk               | Failure to secure required approvals                                       |
| Operational risk              | Failure of the system during an operational mission                        |
| Safety risk                   | Event occurs resulting in serious injury or death                          |
| Supportability risk           | Failure to resupply required material to support operations                |

# Some sources of technical risk

- Technical complexity – many design constraints or many dependent development sequences must occur in the right sequence at the right time
- Organizational complexity – many independent organizations have to perform with limited coordination
- Inadequate margins and reserves (e.g., weight, power)
- Inadequate implementation plans
- Unrealistic schedules
- Total and year-by-year budgets mismatched with implementation risks
- Overly optimistic designs, pressured by mission expectations
- Limited engineering analysis and understanding due to limited models
- Limited understanding of operational environment

**Risk Mitigation involves taking action to reduce the likelihood, the consequence or both.**



# Technical Performance Measures (TPMs):

- Identify critical system parameters (*vital few*)
  - Functional - capabilities that must be provided
  - Non-functional - throughput, mass, reliability, and so on
- Establish interim milestones to assess progress
  - Validate design and key assumptions
  - Provide decision points for considering alternatives
- Are engineering metrics that can be measured throughout development
  - Traceable to top-level requirements
  - Controllable by individual designers
  - Directly feed risk management and mitigation
- Are **NOT** top level stakeholder requirements
  - Customers do not directly care about a TPM - they care about the key stakeholder requirement the TPM tracks

# TPM example - The Boeing 777

- Performance requirement
  - The 777 shall be able to fly non-stop between New York and Tokyo
  - The 777 shat have an operational range of xxx nautical miles
- Possible TPMs
  - Fuel consumption
  - Vehicle weight
  - Tank size
  - Drag
  - Power train friction
- Most important TPM = weight
  - Design weight reviewed monthly throughout the development program
  - “Weight-to-go” used to drive weight reduction initiatives for all components and subsystems
  - Every designer was able to understand his/her impact on aircraft weight



## Additional TPM Examples

- Processor resources utilized
- Processing time (for real time systems)
- Bandwidth required
- Power consumed
- Network resources utilized
- Others?

# Specifying Components Key Questions:

1. What are the key requirements for each of your major subsystems?
  - *Assessment Criteria: Verifiable subsystem requirements derived from key system requirements*
2. What tradeoffs were made in deriving your subsystem requirements?
  - *Assessment Criteria: Tradeoffs systematically analyzed against well-defined criteria and reasonable decisions*
3. What are the most important technical risks for your system and how will you mitigate each?
  - *Key technical risks identified, properly characterized as to likelihood and consequence, and suitable mitigation plans provided for each*

# Instructions for Design Review

- Prepare a 15 minute PowerPoint presentation that describes your system architecture and specifies the requirements for each of its major subsystems
- The presentation should not be in Q&A format but it should address each of the nine Key Questions in the three “Bringing Solutions to Life” lectures
- Post your presentation to the Design Review assignment in Canvas not later than 2:00 pm, Monday, April 4th

# Deciding How the System Will Work

## Key Questions:

1. What top-level functions will your system have to perform in order to accomplish its mission?
  - *Assessment Criteria: Top-level functions are few in number (3-6) and fully represent the required system functionality.*
2. How else could you have partitioned the system functionality and why did you choose to partition it the way you did?
  - *Assessment Criteria: Alternative decomposition is plausible and the rationale for selecting the chosen partition is sound.*
3. How will the top-level functions interact with each other and with their external environment?
  - *Assessment Criteria: Key operational scenarios can be traced through the top-level functions.*
4. How well will your functional architecture perform and what evidence can you provide to support that assessment?
  - *Assessment Criteria: The functional architecture has been analyzed to verify that it achieves the key system requirements.*

# Implementing the Functions Key Questions:

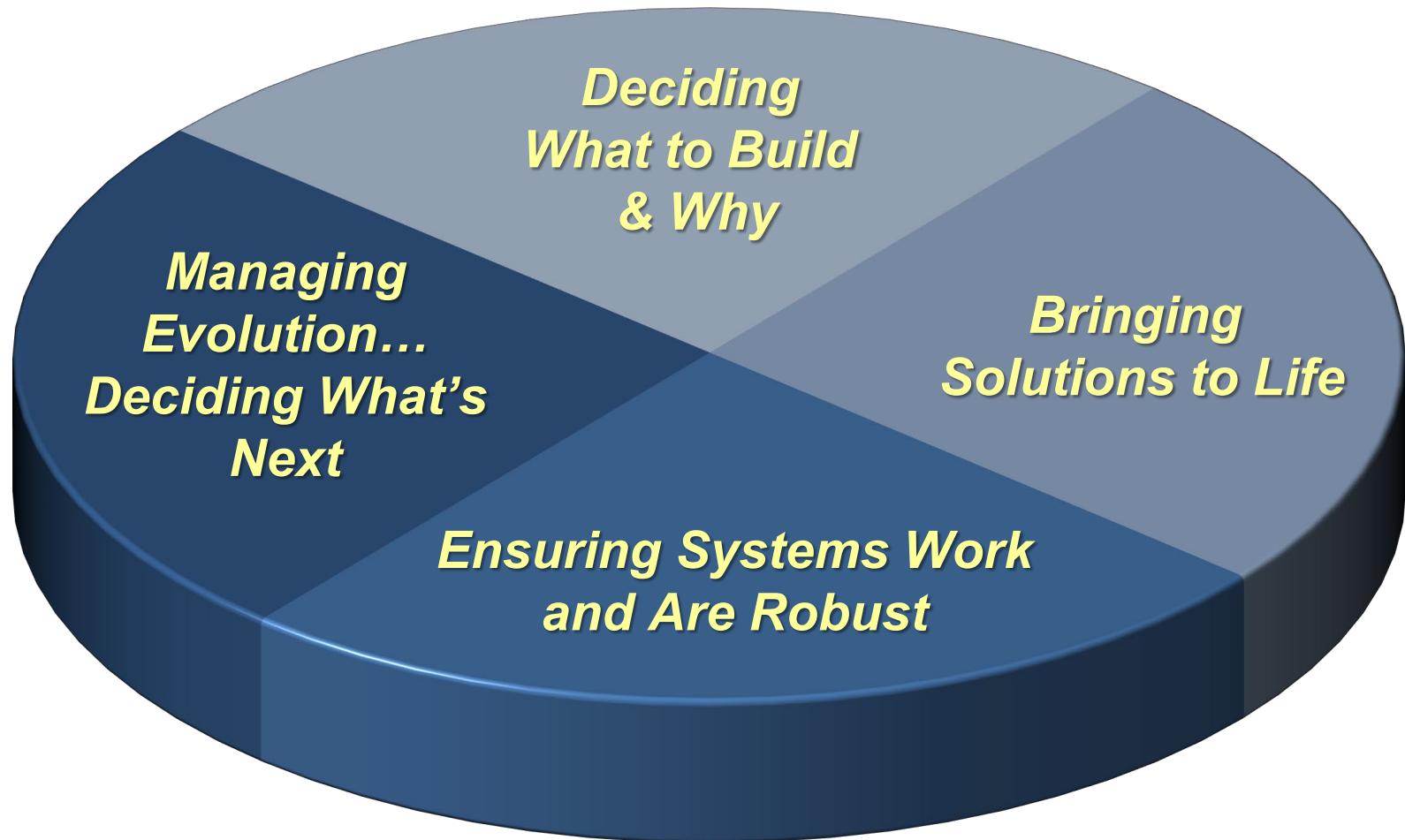
1. What are the major subsystems of your system and how do they map to the functions the system must perform?
  - *Assessment Criteria: Subsystems defined and mapped to the top-level functional architecture.*
2. What are the key interfaces between the subsystems and between them and the external systems with which the system interacts?
  - *Assessment Criteria: External and internal interfaces defined and mapped to corresponding external and internal inputs and outputs.*

# Specifying Components Key Questions:

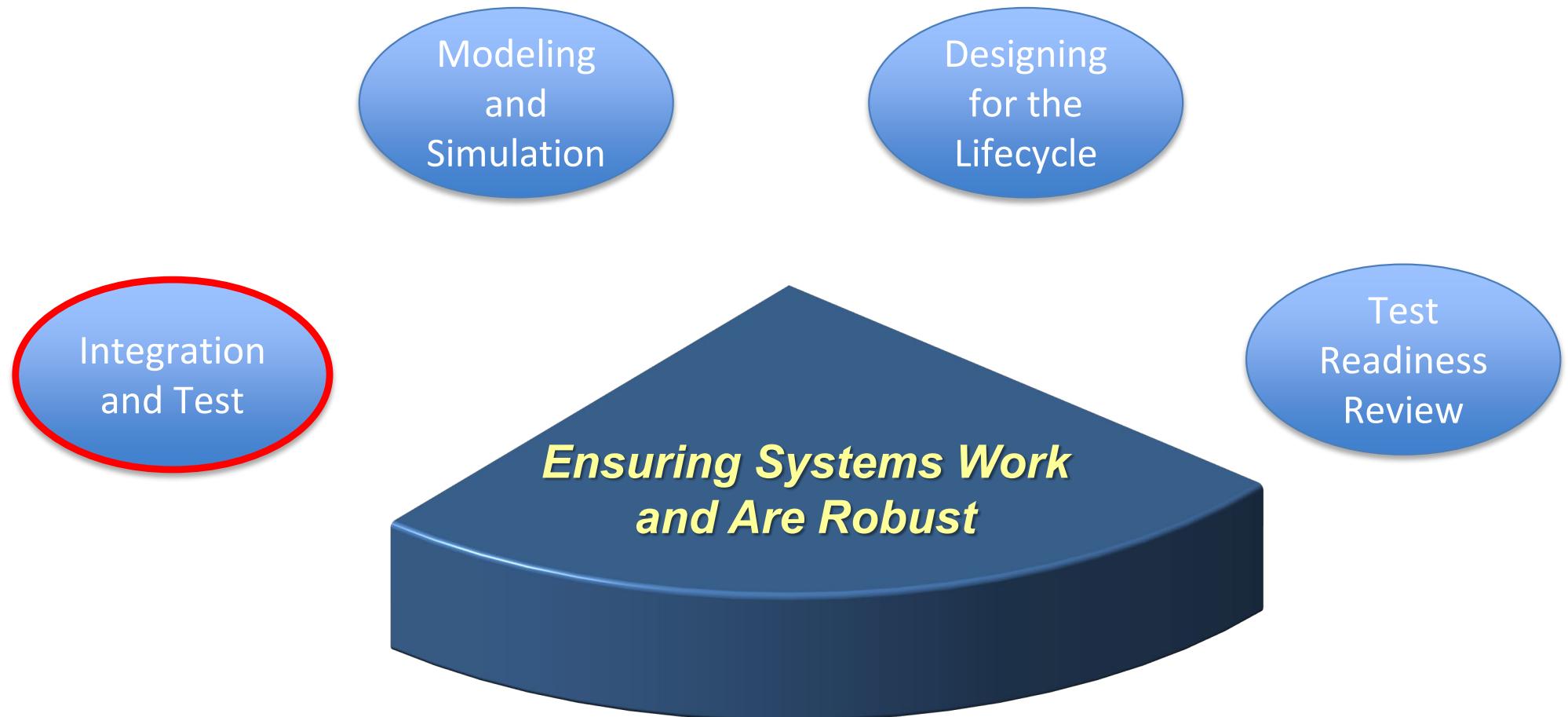
1. What are the key requirements for each of your major subsystems?
  - *Assessment Criteria: Verifiable subsystem requirements derived from key system requirements*
2. What tradeoffs were made in deriving your subsystem requirements?
  - *Assessment Criteria: Tradeoffs systematically analyzed against well-defined criteria and reasonable decisions*
3. What are the most important technical risks for your system and how will you mitigate each?
  - *Key technical risks identified, properly characterized as to likelihood and consequence, and suitable mitigation plans provided for each*

# Integration and Test

# Course Design



# Course Design



# Class Schedule (1 of 2)

| Week | Topic   |
|------|---|
| 1    | <ul style="list-style-type: none"><li>• Thinking in Terms of Systems</li></ul> <p><i>Deciding What to Build and Why</i></p>                                     |
| 2    | <ul style="list-style-type: none"><li>• Defining the Problem</li></ul>  |
| 3    | <ul style="list-style-type: none"><li>• Developing a Solution</li></ul>   |
| 4    | <ul style="list-style-type: none"><li>• Formulating a Proposal</li></ul> <p><i>Bringing Solutions to Life</i></p>   |
| 5    | <ul style="list-style-type: none"><li>• Building a Functional Model<ul style="list-style-type: none"><li>• (Concept Review Storyboards due)</li></ul></li></ul> |
| 6    | <ul style="list-style-type: none"><li>• Concept Review</li></ul>  |
| 7    | <ul style="list-style-type: none"><li>• Implementing the Functions</li></ul>  |
| 8    | <ul style="list-style-type: none"><li>• Specifying Components</li></ul>   |
| 9    | <ul style="list-style-type: none"><li>• Design Review</li></ul>   |

# Class Schedule (2 of 2)



| Week | Topic  |
|------|--|
|      | <i>Ensuring the System Works and Is Robust</i>   |
| 10   | <ul style="list-style-type: none"><li>• Integration and Test</li></ul>                           |
| 11   | <ul style="list-style-type: none"><li>• Ensuring Systems Work and Are Robust (M&amp;S)</li></ul> |
| 12   | <ul style="list-style-type: none"><li>• Designing for the Lifecycle</li></ul>                    |
| 13   | <ul style="list-style-type: none"><li>• Test Readiness Review</li></ul>                          |
|      | <i>Managing Evolution...Deciding What's Next</i>   |
| 14   | <ul style="list-style-type: none"><li>• Technology and Innovation</li></ul>                      |
| 15   | <ul style="list-style-type: none"><li>• Final Project Submission</li></ul>                       |

# Integration and Test Key Questions:

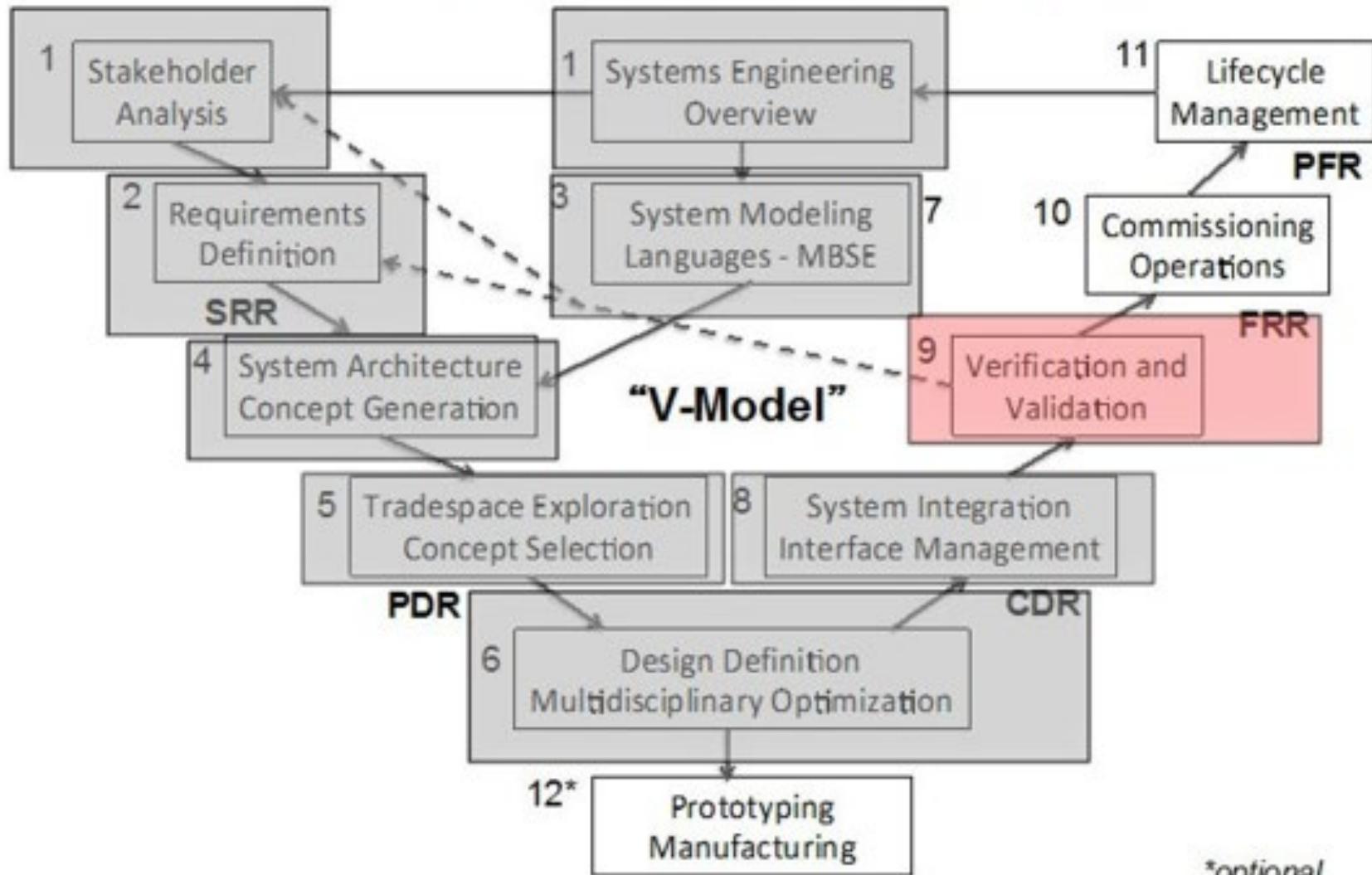
1. How will you integrate and test the components of your system? Why?
2. What resources and facilities will you need to verify that the system meets its specifications? How will you ensure they will be available when needed?
3. How will you prove to your customer that the system meets his/her needs in the operational environment?
4. How will you respond when problems arise in the operational environment?

# Integration, Verification and Validation



# The “V-Model” of Systems Engineering

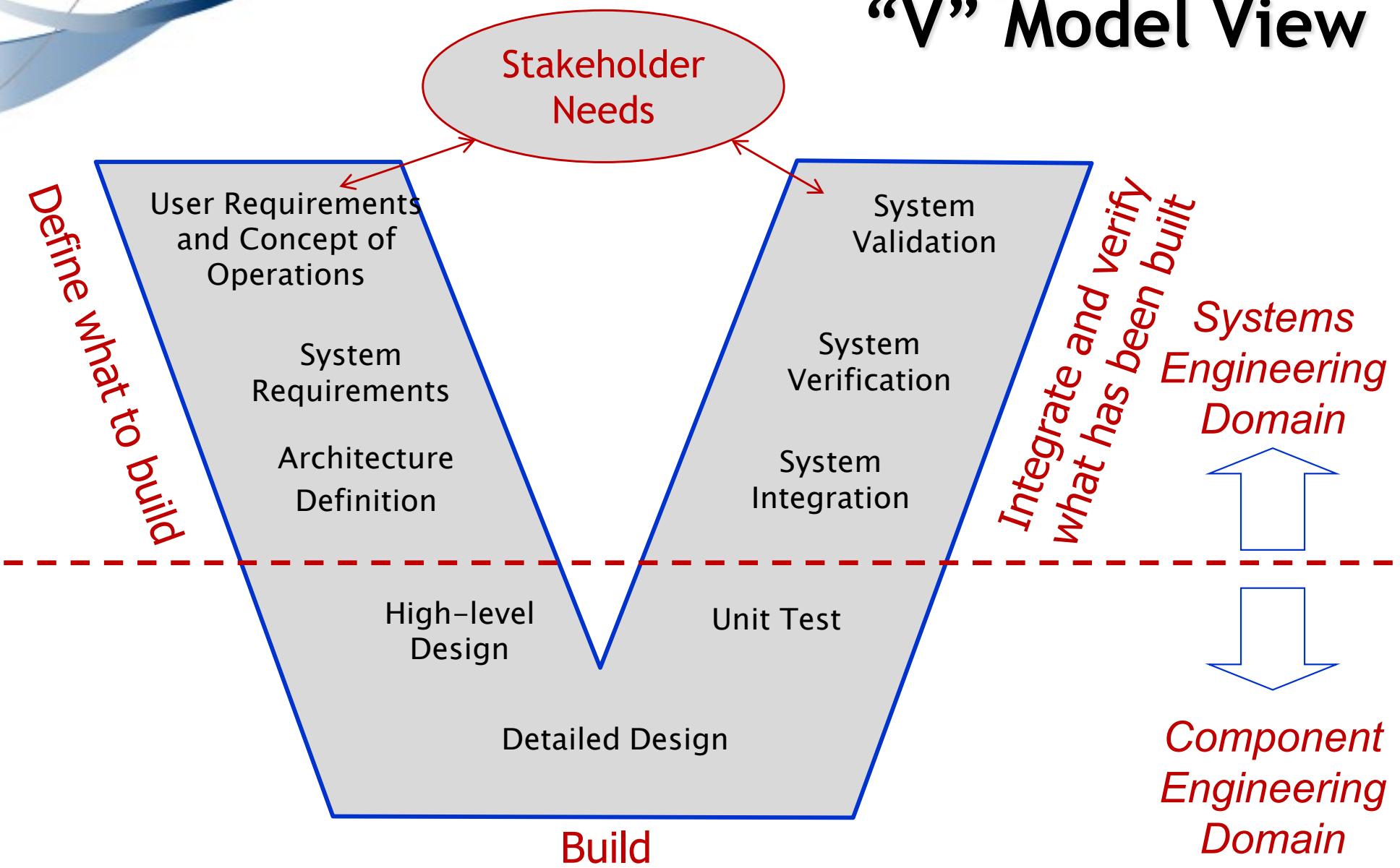
16.842/ENG-421 Fundamentals of Systems Engineering



\*optional

Numbers indicate the session # in this class

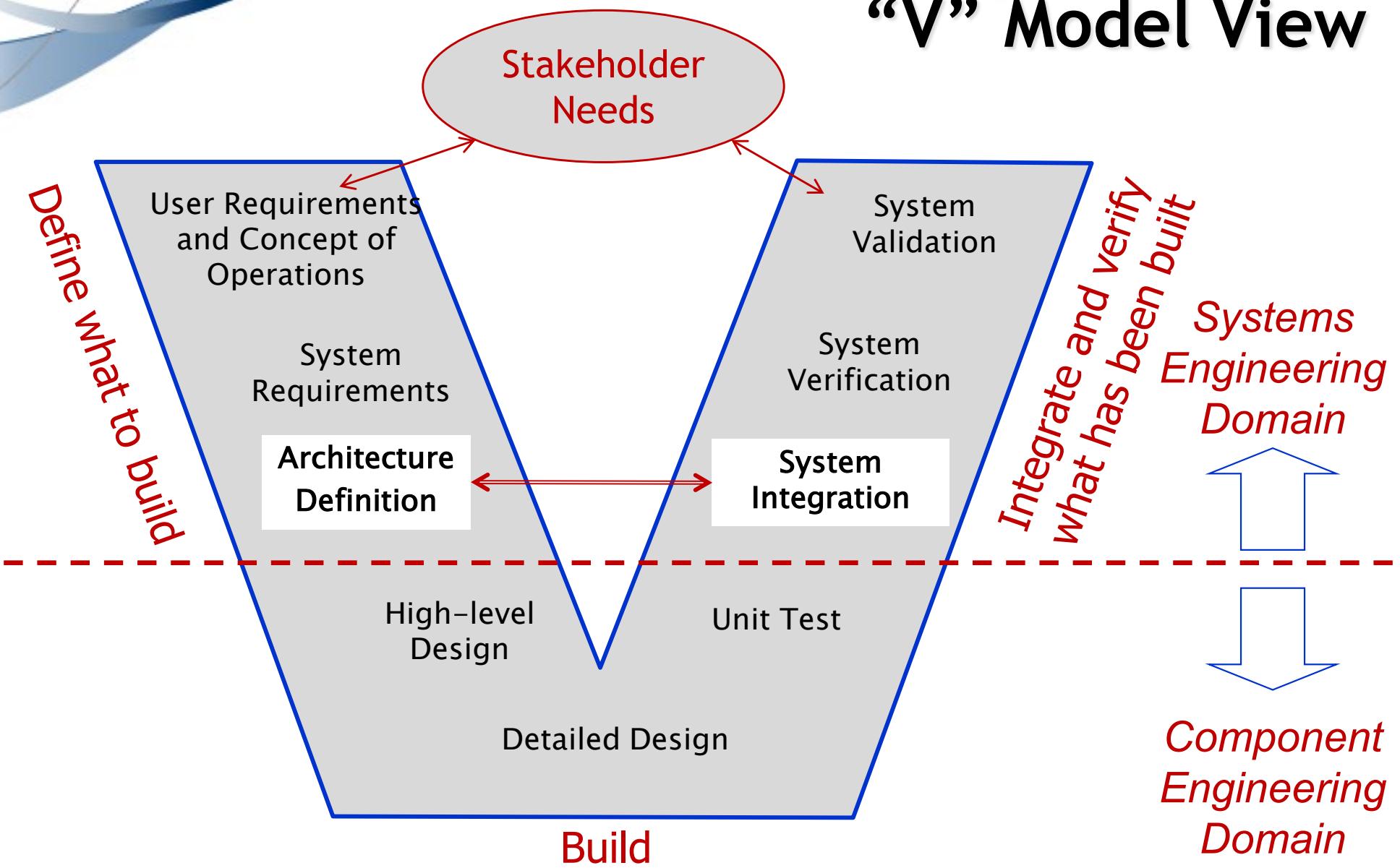
# “V” Model View



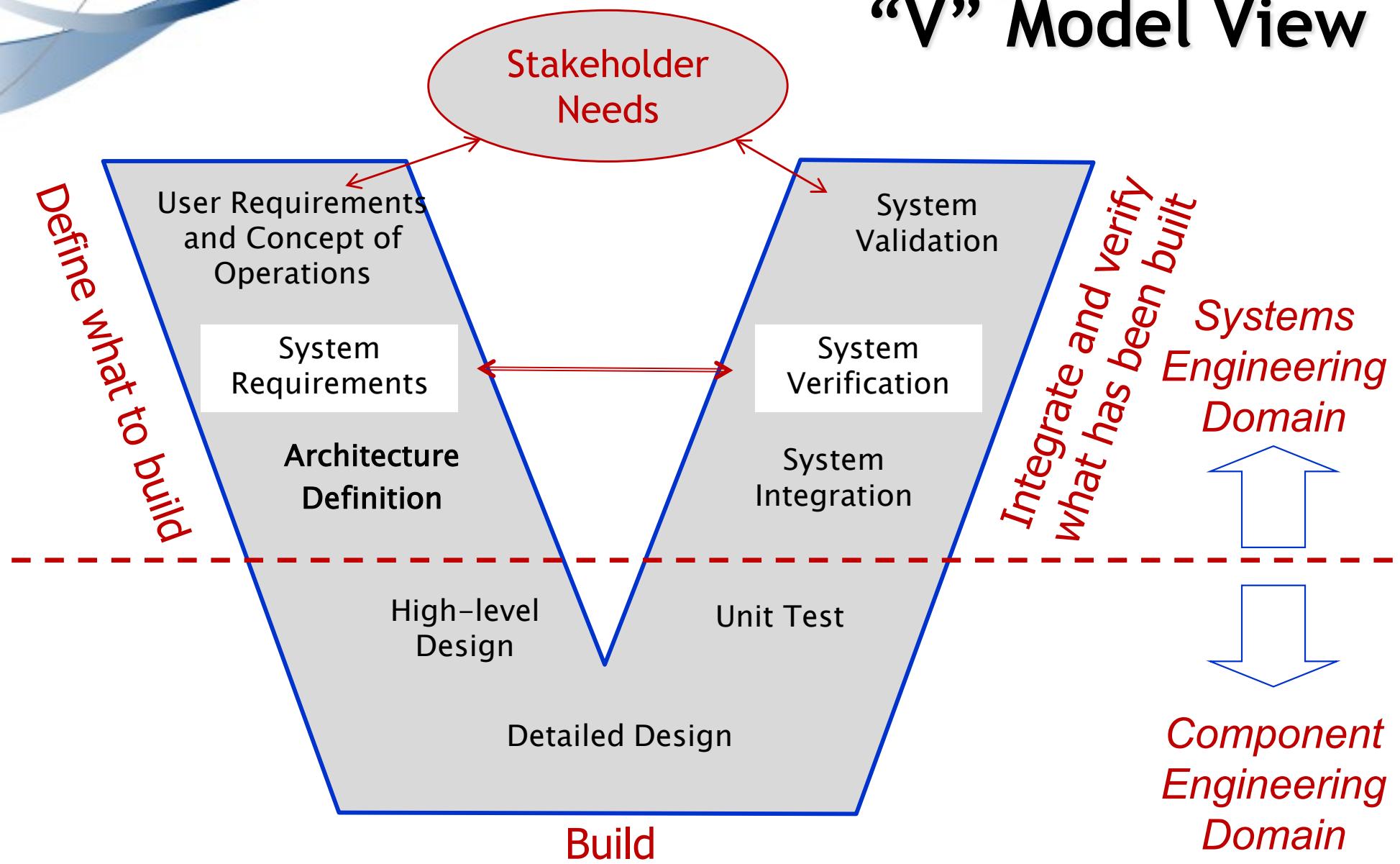
# System Integration

- That part of the system engineering process that unifies the product components and the process components into a coherent whole.
  - Ensures that the hardware, software, and human components of the system interact to achieve the system purpose and satisfy the customer's need
  - Closely related to architecture - it includes the synthesis of the components,, modules and subsystems identified through decomposition of the system during the architectural design

# “V” Model View



# “V” Model View

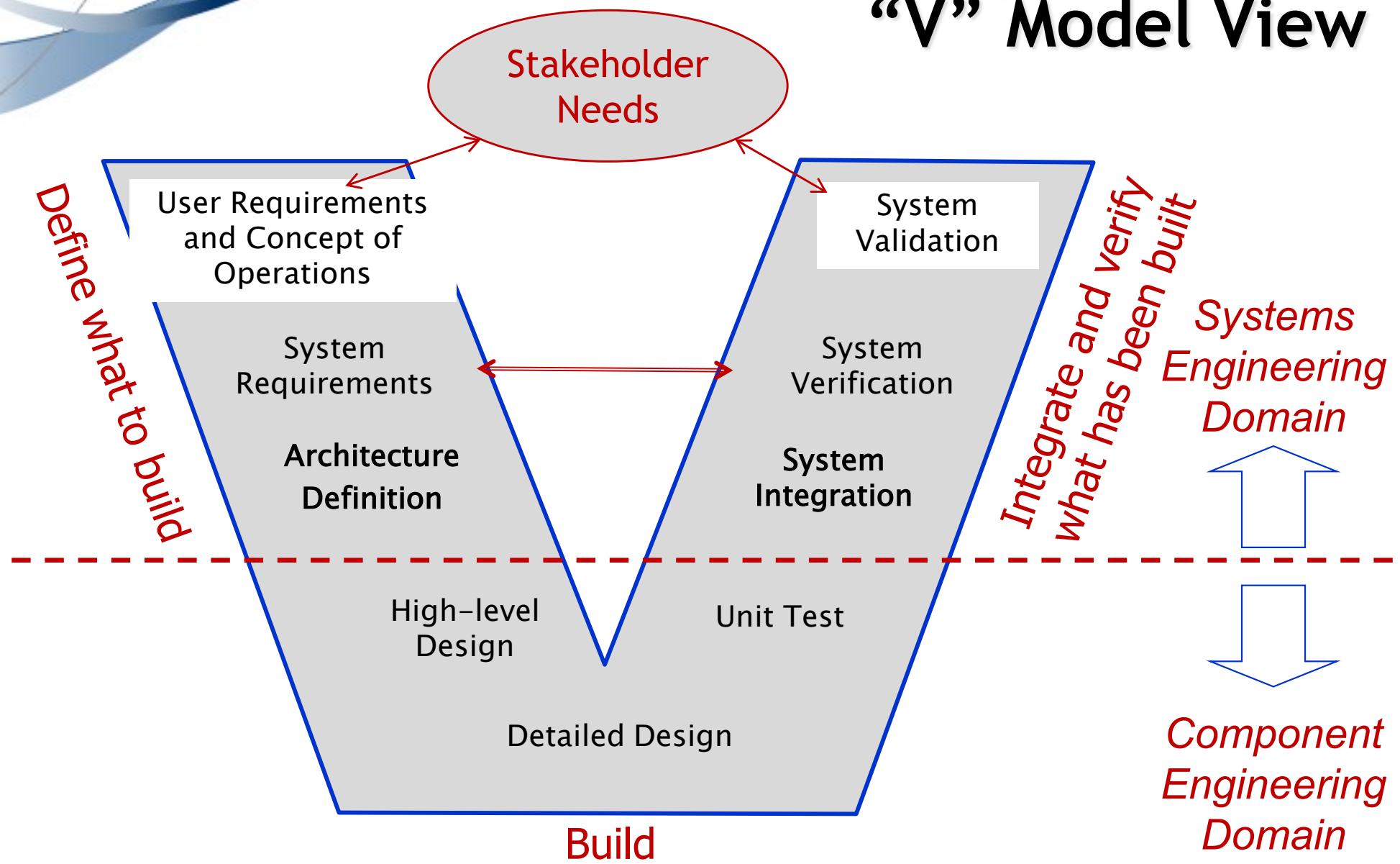


# System Verification

- Does the System meet the **System Requirements?**
  - Establishes the correspondence between a product or system and its specifications
  - Based on the Requirements Database and Traceability
  - Example: compare a module's software code against its technical design specifications document

***Did We Build the Product/System/Service Right?***

# “V” Model View



# System Validation

- Does the System satisfy the original **Stakeholder Needs?**
  - Ensure compliance against an original stakeholder need.
  - Example: compare the actual response of an online transaction system to what was originally expected, requested and approved.
  - Acceptance Tests are one component of System Validation
  - System Validation is based on Operational or Field Testing

**Did We Build the Right Product/System/Service?**

# Differences between V & V

## Differences Between Verification and Validation Testing

### Verification Testing

Verification testing relates back to the approved requirements set (such as an SRD) and can be performed at different stages in the product life cycle. Verification testing includes: (1) any testing used to assist in the development and maturation of products, product elements, or manufacturing or support processes; and/or (2) any engineering-type test used to verify the status of technical progress, verify that design risks are minimized, substantiate achievement of contract technical performance, and certify readiness for initial validation testing. Verification tests use instrumentation and measurements and are generally accomplished by engineers, technicians, or operator-maintainer test personnel in a controlled environment to facilitate failure analysis.

### Validation Testing

Validation relates back to the ConOps document. Validation testing is conducted under realistic conditions (or simulated conditions) on any end product to determine the effectiveness and suitability of the product for use in mission operations by typical users and to evaluate the results of such tests. Testing is the detailed quantifying method of both verification and validation. However, testing is required to validate final end products to be produced and deployed.

## Verification

- During development
- Check if requirements are met
- Typically in the laboratory
- Component/subsystem centric

*Was the end product realized?*

- ## Validation
- During or after integration
  - Typically in real or simulated mission environment
  - Check if stakeholder intent is met
  - Full-up system

*Was the right end product realized?*

# Concept Question

What is your name?

How would you classify the following activities? \*

|  | Verification          | Validation            | No sure               |
|--|-----------------------|-----------------------|-----------------------|
| Testing handling of a new car in snow conditions in Alaska | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |
| Frontal crash test in the lab                              | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |
| Testing of a new toy in a Kindergarten                     | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |
| Vehicle emissions testing on a dynamo                      | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |
| Satellite vibration testing on a shake table               | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |
| Field testing of Google glasses with 1,500 pilot users     | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |



# Integration

# Perspectives of a Classical Architect (from Week 6)

- **Objective:** Achieve a good “fit” between the form to be designed and its context
  - Form – that over which we have control
  - Context – that which puts demands on the form
- **Challenge:** “We seek to produce harmony between a form we have not yet designed and a context we cannot fully describe.”
- **Process:**
  - Define the context as a set of requirements
  - Partition the requirements into clusters that are richly connected internally and as independent of each other as possible
  - Design a form that meets each cluster of requirements
  - Integrate the low level forms to produce the desired whole

# Perspectives of a Classical Architect (from Week 5)

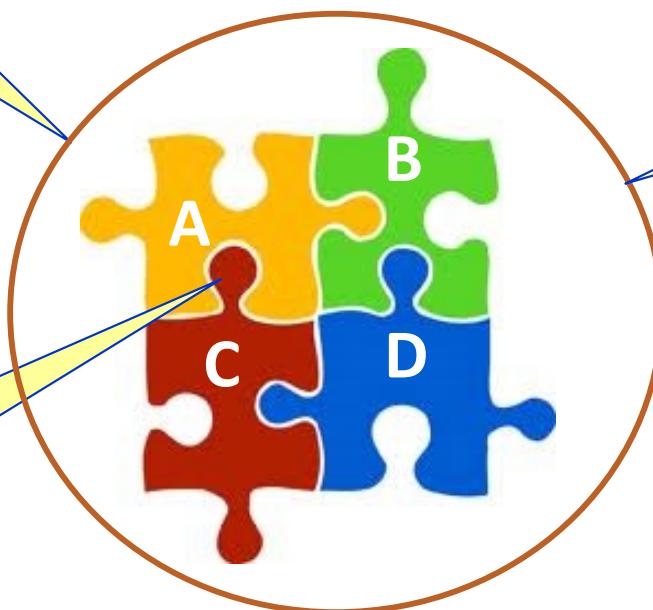
- **Objective:** Achieve a good “fit” between the form to be designed and its context
  - Form – that over which we have control
  - Context – that which puts demands on the form
- **Challenge:** “We seek to produce harmony between a form we have not yet designed and a context we cannot fully describe.”
- **Process:**
  - Define the context as a set of requirements
  - Partition the requirements into clusters that are richly connected internally and as independent of each other as possible
  - Design a form that meets each cluster of requirements
  - **Integrate the low level forms to produce the desired whole**

# Successful Integration is all about...

**Interface**  
Management  
 $A \leftrightarrow C$

**Risk** Mitigation  
Likelihood &  
Consequence

**Requirements**  
Flowdown  
 $A+B+C+D < S$

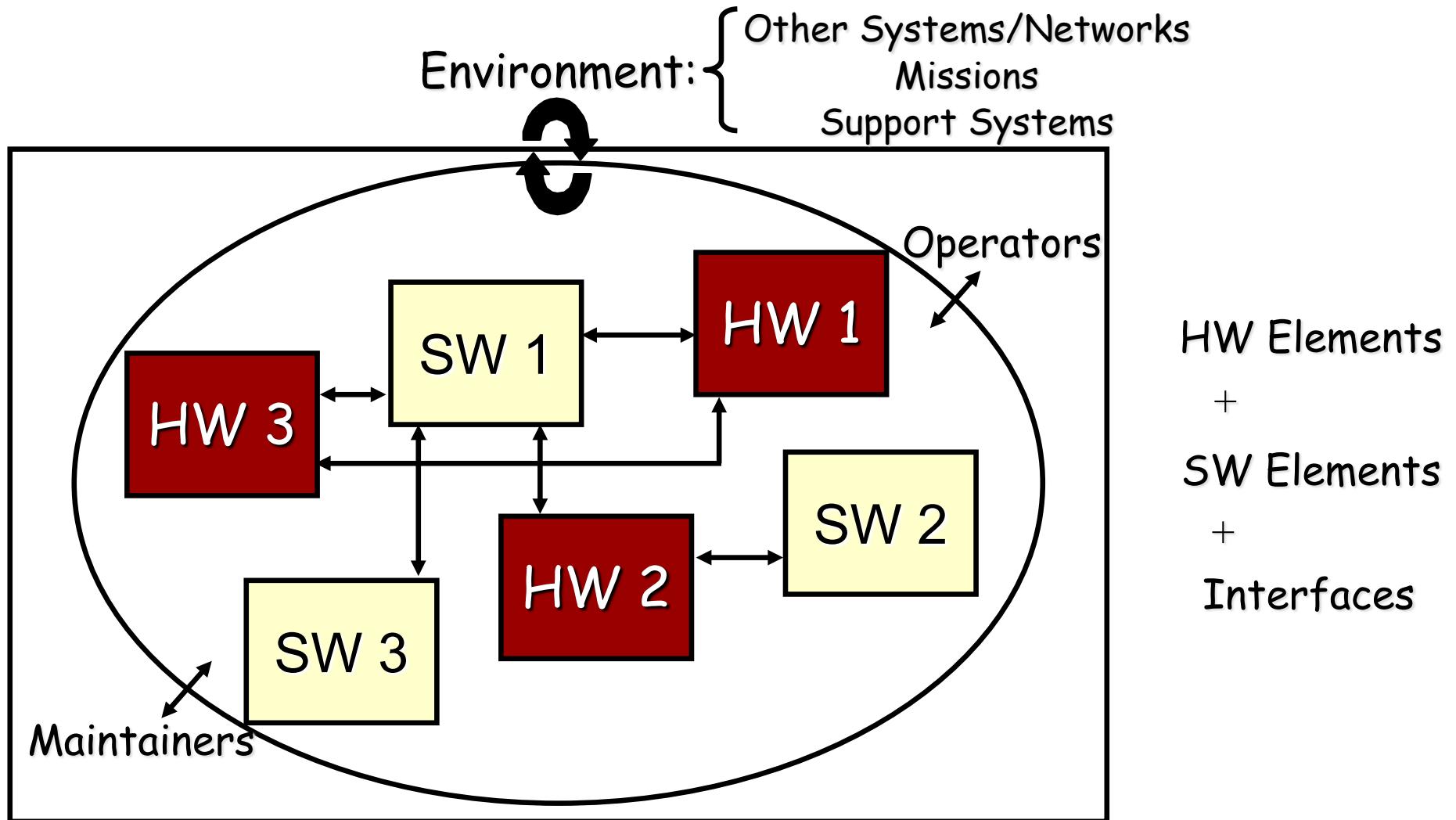




# Interface Management

# Not Just Components but Interfaces

(from Week 7)

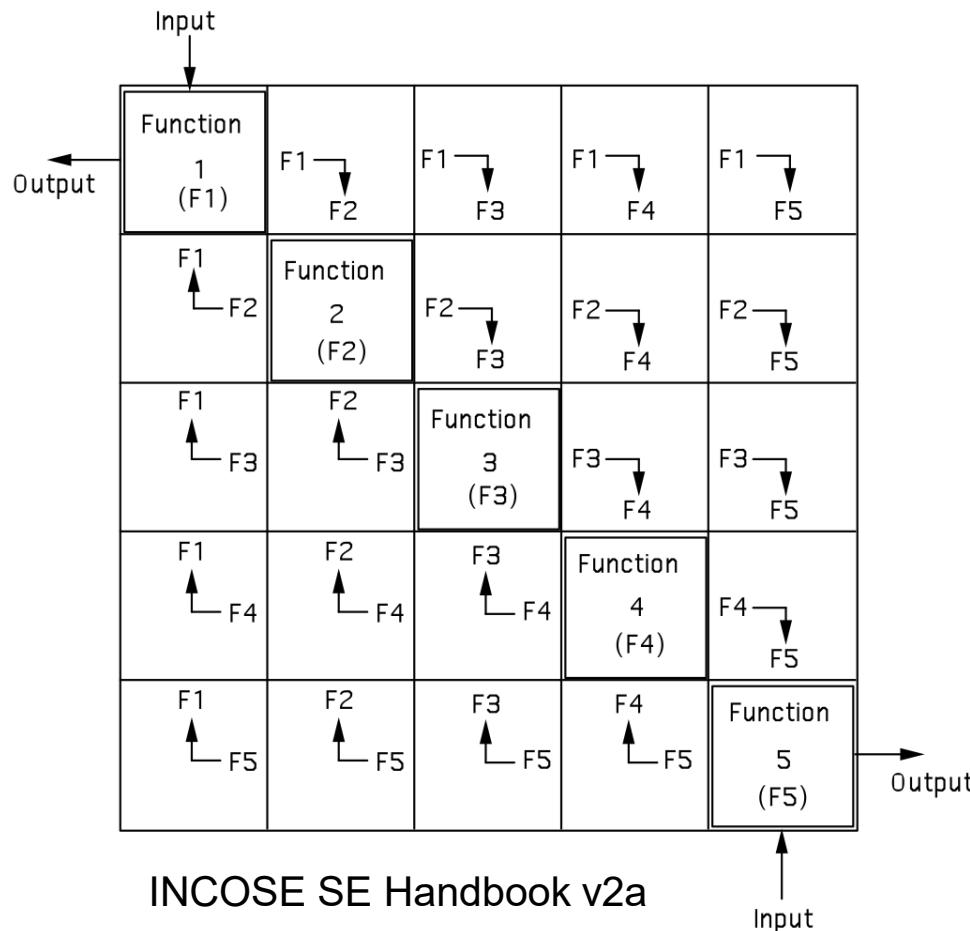


# Interface Criticality

(from Week 7)

- Systems usually fail at the interfaces
- Interfaces are uniquely determined by and owned by the systems engineers
- Three types of interfaces:
  - **Connectors** - facilitate the transmission or exchange of electricity, fluid, force, material, information, etc.
  - **Isolators** - inhibit such interactions
  - **Converters** - alter the form of the entity being transmitted or exchanged

# Functional N<sup>2</sup> Diagram)



- Functions are placed on the diagonal and are described as verb noun (or verb object) pairs.
- Functional interfaces (and flows) are in clockwise direction, e.g., F1 → F2 indicates a flow from F1 to F2 above the diagonal or F2 → F1 below the diagonal.
- Flows may be primary or feedback.
- Functional interfaces are items, not functions

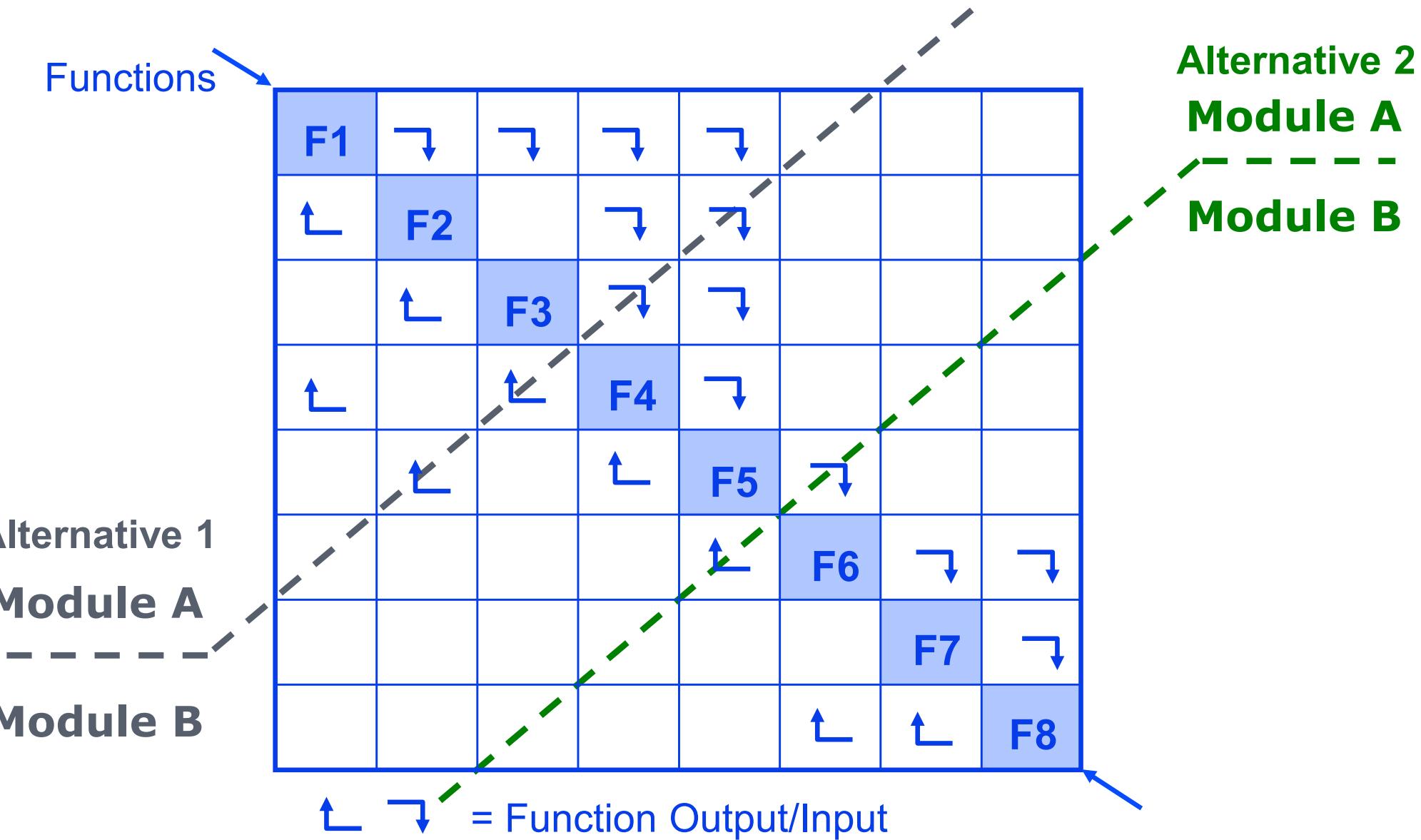
The N2 chart is used in functional analysis to identify interactions between functions.

It can also be used to indicate functional complexity. Each box on the diagonal is a task such as "move control surface," "measure position," or "compare position to desired."

The arrows indicate an output of one task, such as position data, being passed to another task as its input.

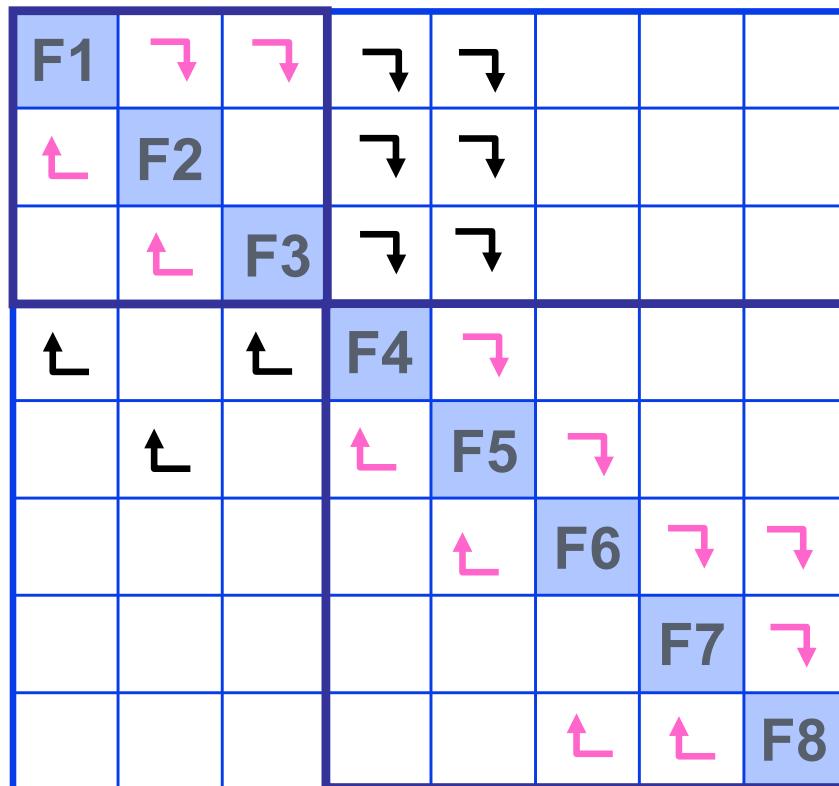
This basic definition of selected functions and their interaction is  
the basis of the “functional architecture.”

# Functional Allocation and Partitioning

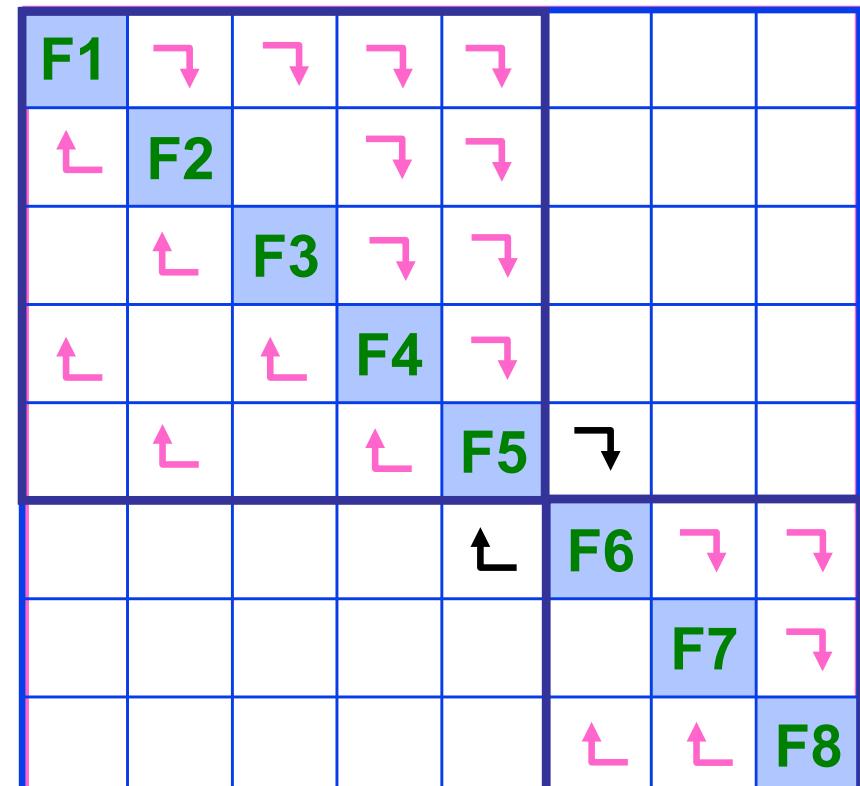


# Interface Impacts of Functional Partitioning

Alternative 1



Alternative 2



↑↓ = Function Output/Input

↑↓ = Intercomponent Output/Input

# Count the Number of Interfaces

<https://www.youtube.com/watch?v=qKpxd8hzOcQ>



# Requirements Flowdown

# Being Clear About Requirements

(from Week 3)



## Customer Needs

- Language of the Stakeholders
- What the customer wants
- May be subjective

## System Specifications

- Language of the Engineers
- What the engineers build
- Must be objective

# Requirements Flowdown (from Week 8)

- For each system requirement, derive requirements for each subsystem, and ultimately for each component:
  - **Equivalence** is a simple flowdown technique that causes the subsystem requirement to be the same as the system requirement
  - **Apportionment** spreads a system-level requirement among the system's subsystems of the system, maintaining the same units, e.g., cost, reliability
  - **Synthesis** addresses those situations in which the system-level requirement is driven by complex contributions from several subsystems, requiring subsystem requirements flowed down from the system to be based upon some analytic model

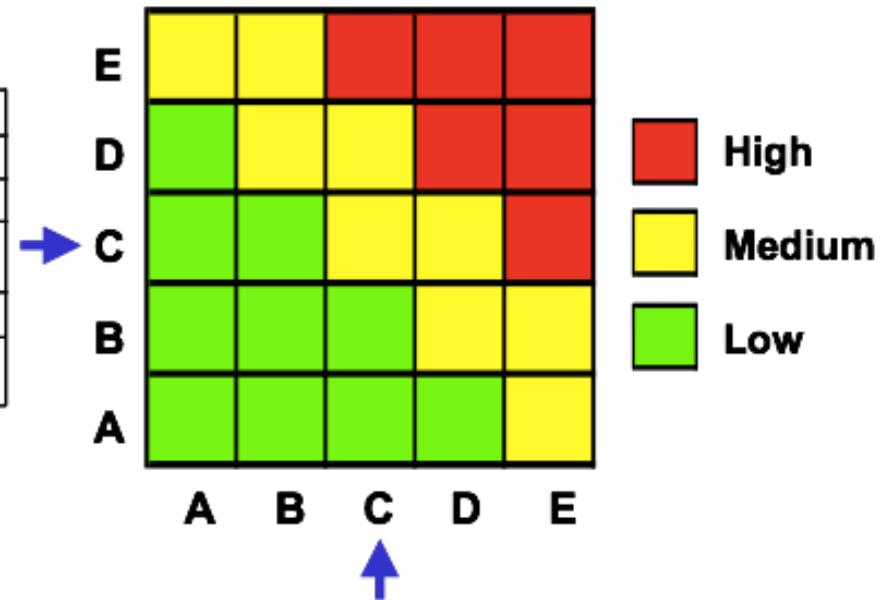
A 80 years old person needs to get all required food /grocery items without going to the store. He has disabilities, he cannot walk and he does not have a good memory , so he cannot track of what is left or needed in the refrigerator..he needs a system which will make him happy and have access food items in his refrigerator without really needing to go to grocery or call anybody.

# Risk Mitigation

# Technical Risk Assessment

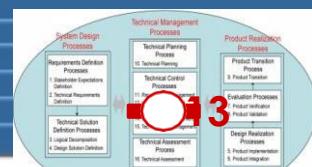
*(from Week 8)*

| Probability of Occurrence   |       |
|---|-------|
| Design Difficulty   | Level |
| Requires breakthrough advance to meet requirements                    | E     |
| Substantial development is required                                   | D     |
| Moderate development is required to a new or existing item            | C     |
| Off-the-shelf item with minor modifications                           | B     |
| Qualified off-the-shelf item which meets all operational requirements | A     |



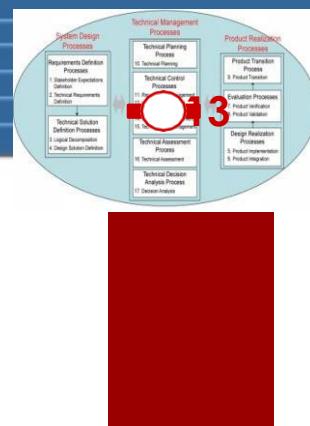
Consequence: Given the risk is realized, what is the magnitude of impact

| Cost          | Performance                             | Schedule   | Level |
|---------------|---|--|-------|
| ≥ 20%         | Unacceptable                            | Can't achieve key team or major program milestone        | E     |
| ≥ 15% - < 20% | Acceptable; no remaining margin         | Major slip in key milestone or critical path impacted    | D     |
| ≥ 10% - < 15% | Acceptable with medium margin reduction | Minor slip in key milestones, not able to meet need date | C     |
| ≥ 5% - < 10%  | Acceptable with small margin reduction  | Additional resources required, able to meet need date    | B     |
| ≥ 0% - < 5%   | Minimal or no impact                    | Minimal or no impact                                     | A     |



# Importance of Technical Risk Management

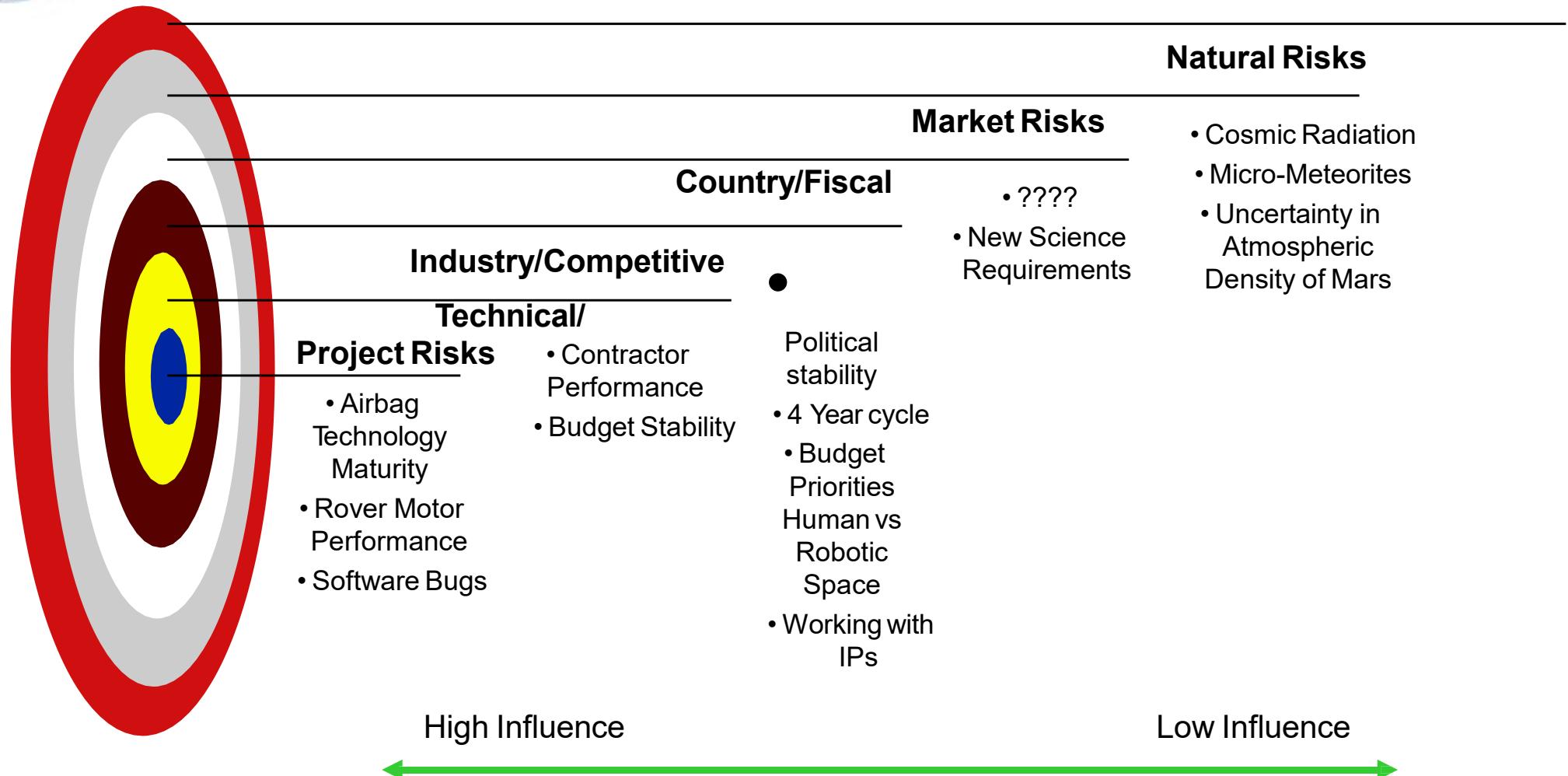
- Risk is defined as the combination of:
- The **probability** that a program or project will experience an undesired event and
- The **consequences**, impact, or severity of the undesired event, were it to occur
- The undesired event might come from **technical** or **programmatic** sources (e.g. a cost overrun, schedule slippage, safety mishap, health problem, malicious activities, environmental impact, or failure to achieve a needed scientific or technological objective or success criteria)
- Technical Risk Management is an organized, systematic risk-informed **decision-making** discipline that **proactively** identifies, analyzes, plans, tracks, controls, communicates, documents, and manages risk to increase the likelihood of achieving project goals



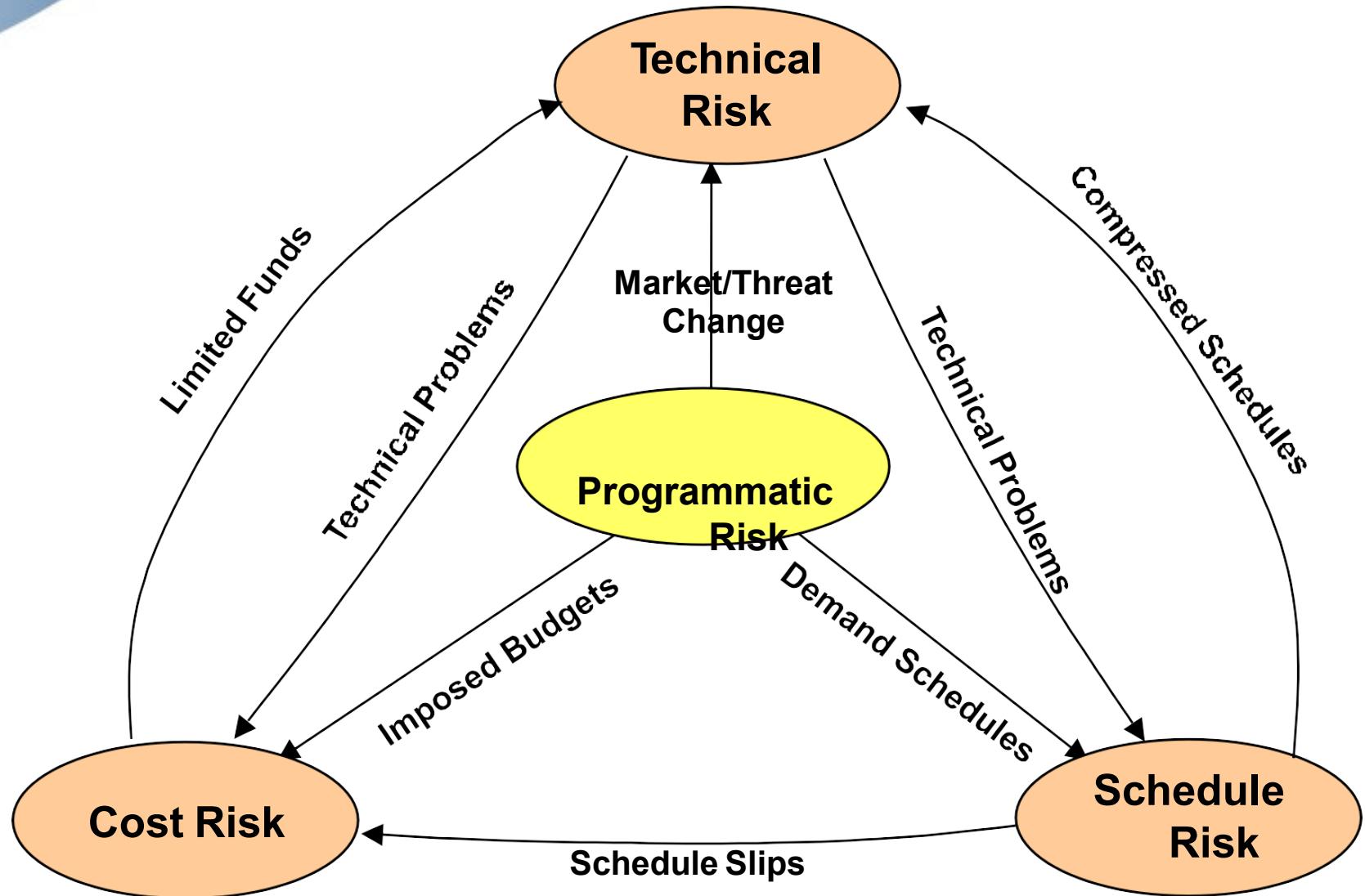
# What is Risk?

- Risk is a measure of **future uncertainties** in achieving program technical performance goals within defined cost and schedule constraints
- Risks can be associated with **all** aspects of a technical effort, e.g., threat, technology maturity, supplier capability, design maturation, performance against plan, etc., as these aspects relate within the systems structure and with interfacing products.
  - Risks have three components:
    1. Future **root cause**
    2. Probability or **likelihood** of that future root cause occurring
    3. **Consequences** (or effect) of that future occurrence

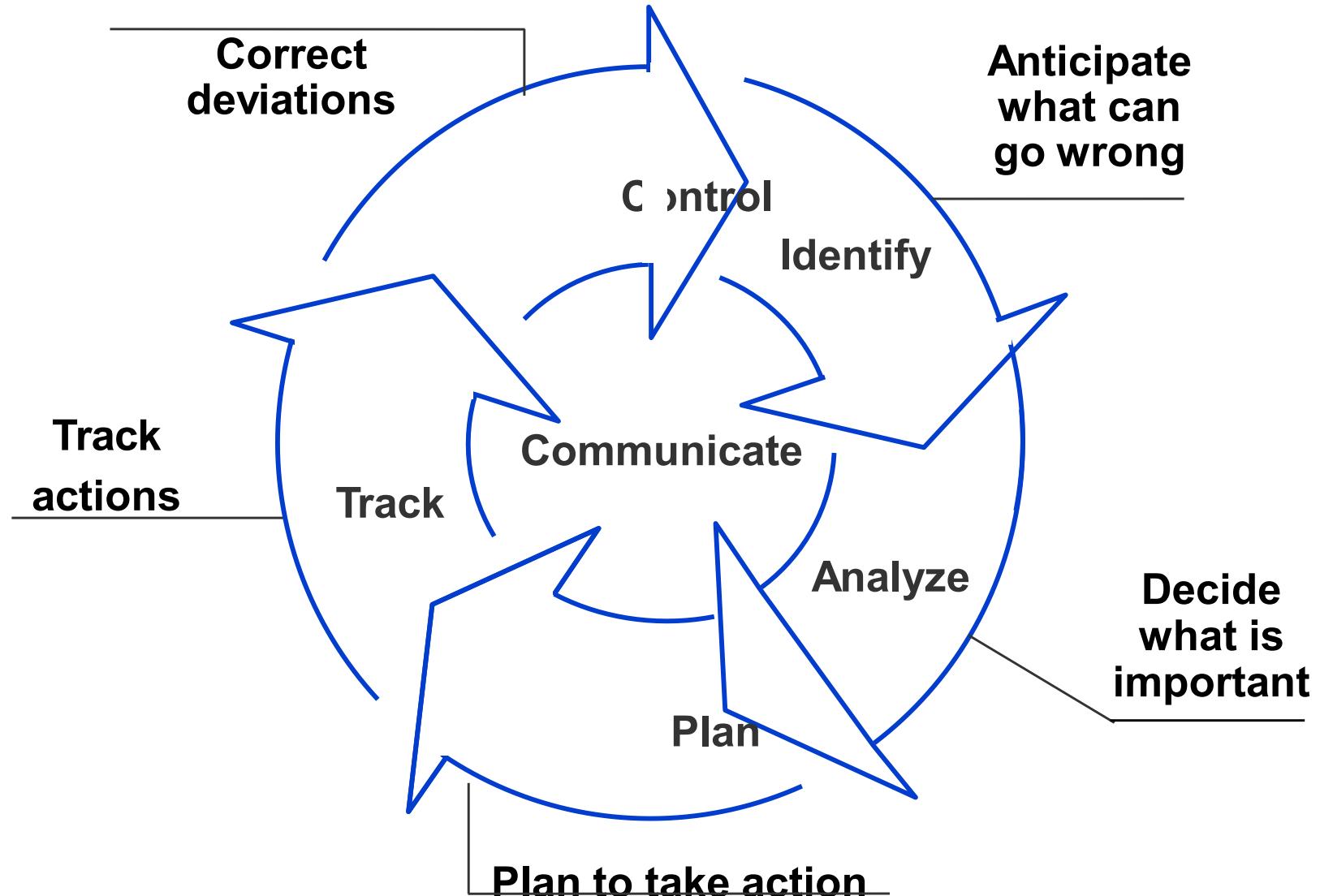
# Layers of Risk Model (e.g. for Mars Missions)



# Risk Categories – “Iron” Triangle



# A Risk Management Framework





# Verification Testing

*Did we build the system right?*



## Types of Testing

There are many different types of testing that can be used in verification of an end product. These examples are provided for consideration:

- Aerodynamic
- Burn-in
- Drop
- Environmental
- High-/Low-Voltage Limits
- Leak Rates
- Nominal
- Parametric
- Pressure Limits
- Security Checks
- Thermal Limits
- Acceptance
- Characterization
- Electromagnetic Compatibility
- G-loading
- Human Factors Engineering/  
Human-in-the-Loop Testing
- Lifetime/Cycling
- Off-Nominal
- Performance
- Qualification Flow
- System
- Thermal Vacuum
- Acoustic
- Component
- Electromagnetic Interference
- Go or No-Go
- Integration
- Manufacturing/Random Defects
- Operational
- Pressure Cycling
- Structural Functional
- Thermal Cycling
- Vibration

This image is in the public domain.

Source: NASA SE Handbook, Section 5.3 Product Verification



## Turn-to-your-partner Exercise

- **What kind of testing have you been involved in in the past? What was the purpose? What where the challenges? What went well? What were the results?**
  - Discuss for 10 min.
  - Share.

# Testing Strategies

1. Testing by Decree: “The programmer says it works fine.”
2. Testing by Rumor: “I heard there are no problems.”
3. Testing by E.S.P.: “I just feel that it has no bugs.”
4. Testing by Desperation: “It has to work!”
5. Testing by Analogy: “The other program had no problem, so this one should be fine too.”
6. Testing by Logic: “I see no reason why it shouldn’t work.”
7. Testing by Wishful Thinking: “I’m sure it will be fine.”
8. Testing by Prayer: “God, I hope there will be no fatal errors.”
9. Testing by Technical Support: “Ship it. We’ll fix it with TOOLS.”
10. Testing by Experience: “It worked one time.”
11. Testing by Assurance: The developer only made a small change, so he/she is sure that it will still work.”

# Integration and Verification Strategies

- Bottom-Up
- Top-Down
- Hybrid
- Big Bang
- Cross-path Component
- External

# Bottom-Up Integration

- Integration begins with the elementary pieces of the system (units, configurable items, ...)
- After each such piece is tested, components comprising multiple units are integrated and eventually tested
- This process continues until the entire system is assembled and tested
- This is the traditional systems engineering integration approach

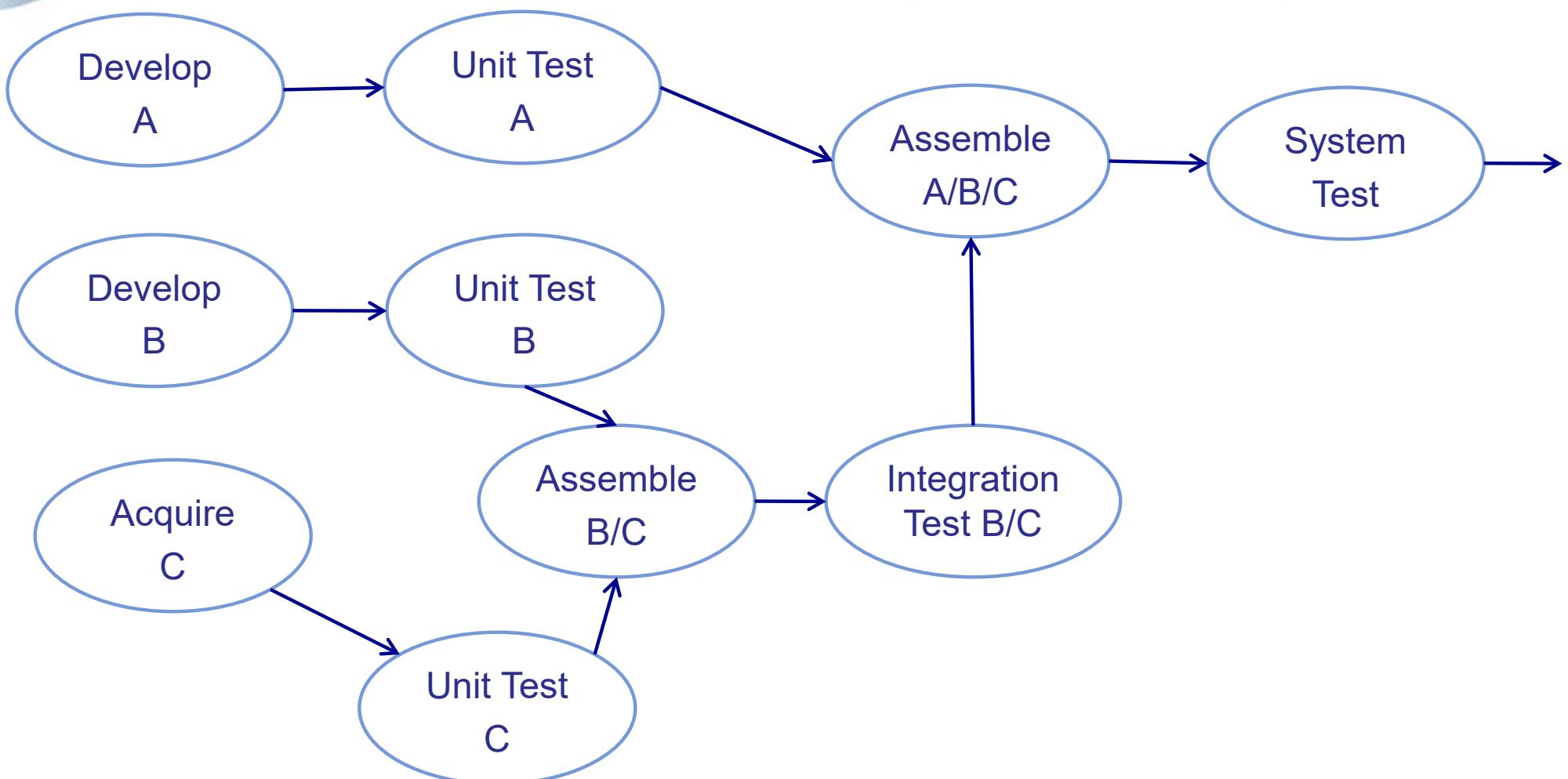
## Advantages

- It is easier to detect flaws in the tiniest pieces of the system
- Test conditions are easier to create
- Observation of the test results is easier

## Disadvantages

- “Scaffold” systems must be produced to support pieces as they are integrated
- System’s control structure cannot be tested until the end
- Major errors in the system design are typically not caught until the end
- System does not exist until the last integration test is completed

# Sample Bottom-Up Integration Sequence



# Top-Down Integration

- Integration begins with a major or top-level module
- All modules called from the top-level module are simulated by “stubs” (shell or model replica) or models
- Once the top-level module is qualified, actual modules replace the stubs and models until the entire system has been qualified.
- This is most useful for systems using large amounts of COTS components

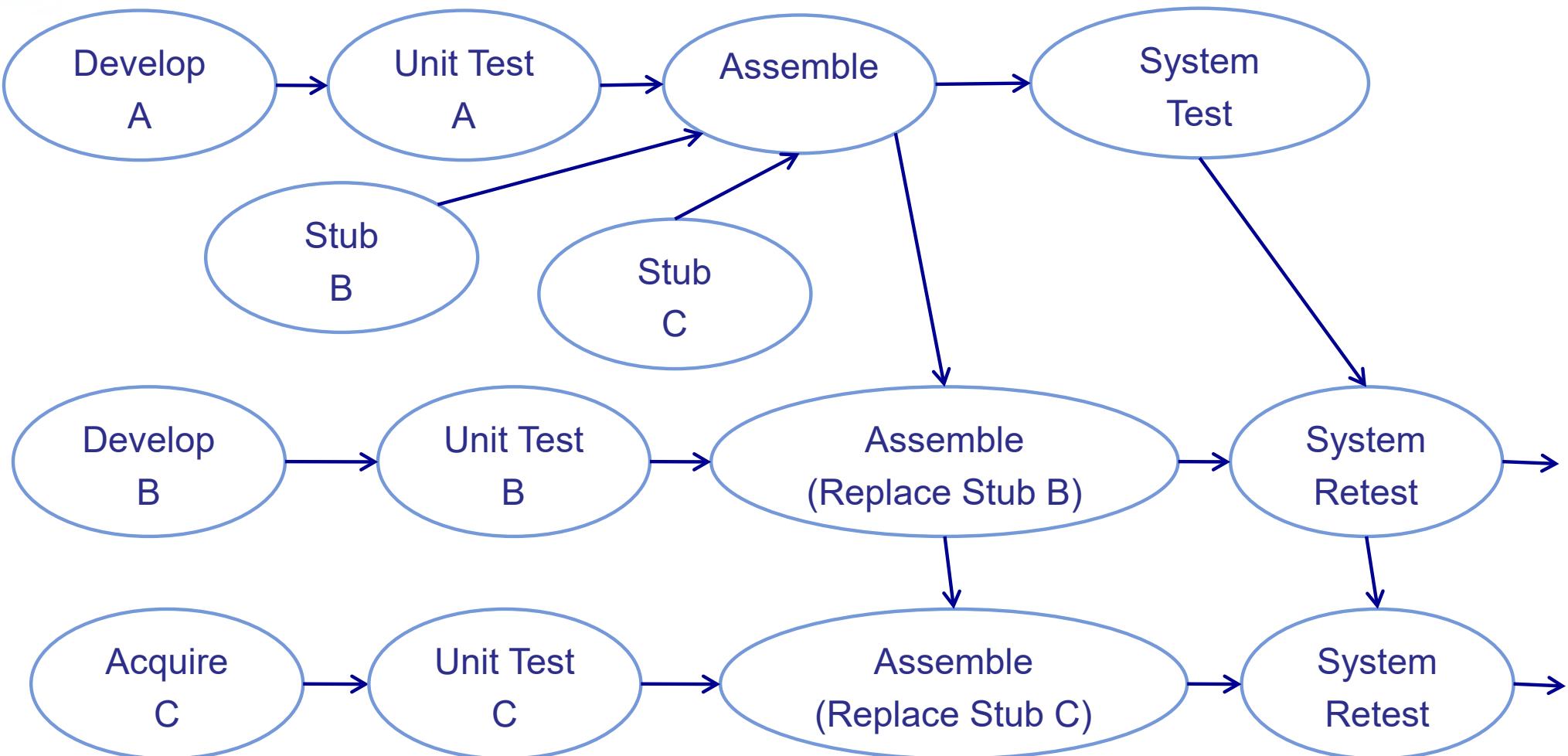
## Advantages

- Early demonstration of the system is allowed
- Representation of the test cases is easier
- More productive if major flaws occur toward the top of system

## Disadvantages

- Stubs and models have to be developed
- Representation of test cases in the stubs may be difficult
- Observation of test output may be artificial and difficult

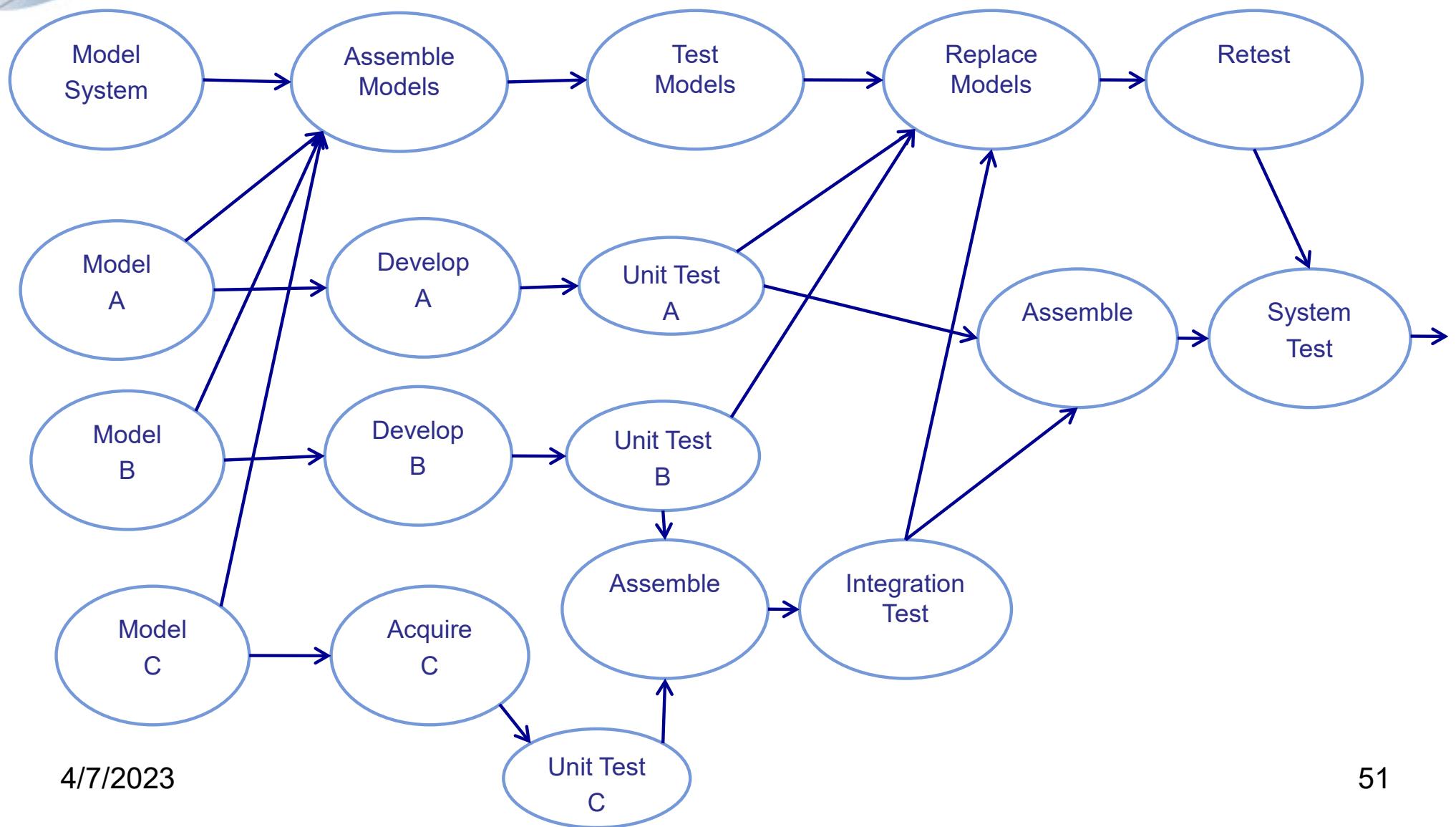
# Sample Top-Down Integration Sequence



# Hybrid Integration

- Mixture of bottom-up and top-down integration. The main focus is on bottom-up integration, with the objective to minimize the disadvantages by using techniques such as:
  - Model-based approaches
  - Early system integration by paying explicit attention to integration of system qualities

# Sample Hybrid Integration Sequence



# Big Bang Integration

- Untested CIs are assembled and the combination is tested
- This is a commonly used and maligned approach

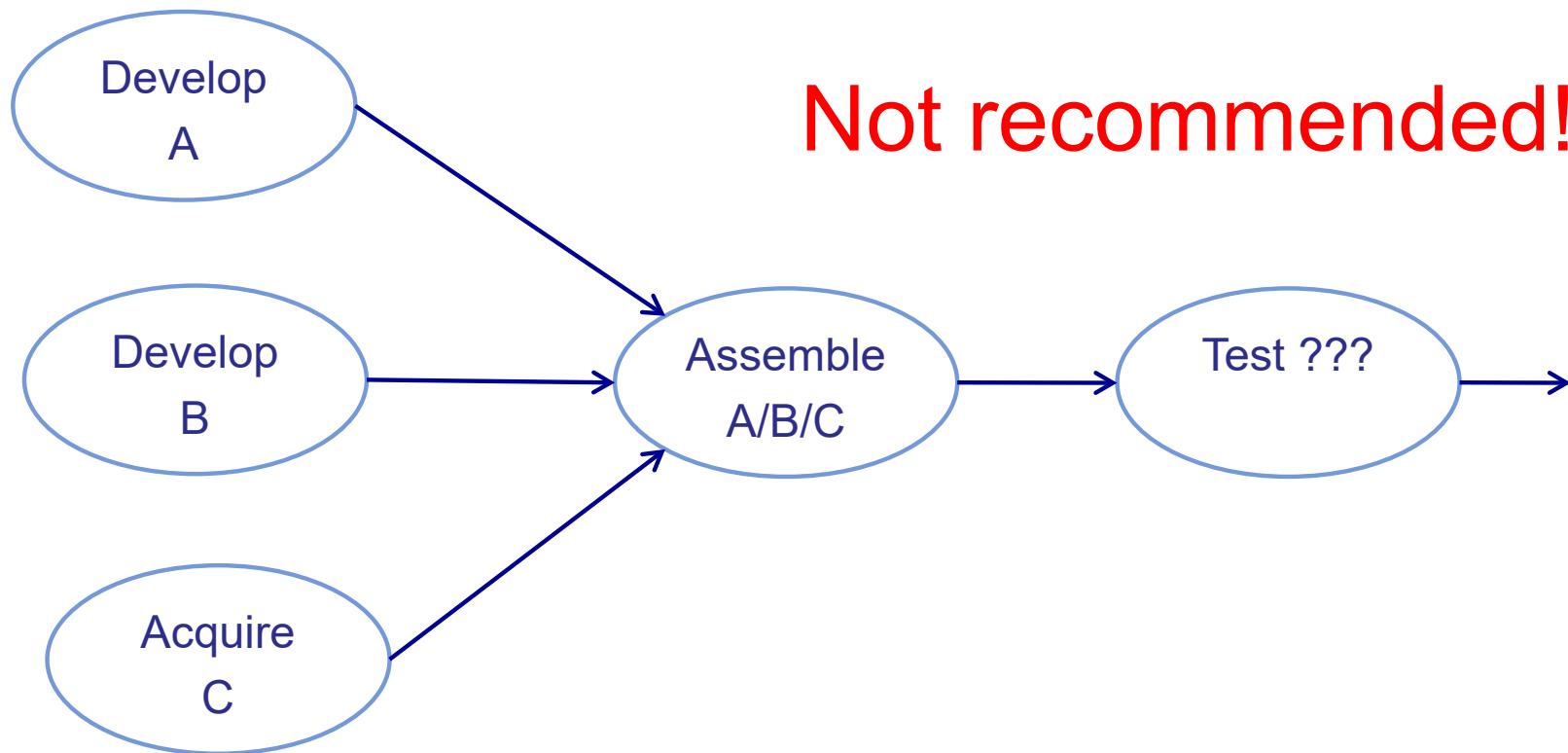
## Advantages

- Immediate feedback on the status of system elements is provided
- Little or no pretest planning is required
- Little or no training is required

## Disadvantages

- Source of errors is difficult to trace
- Many errors are never detected
- Late detection of design errors leading to complex and time consuming repair actions

# Sample Big Bang Integration Sequence

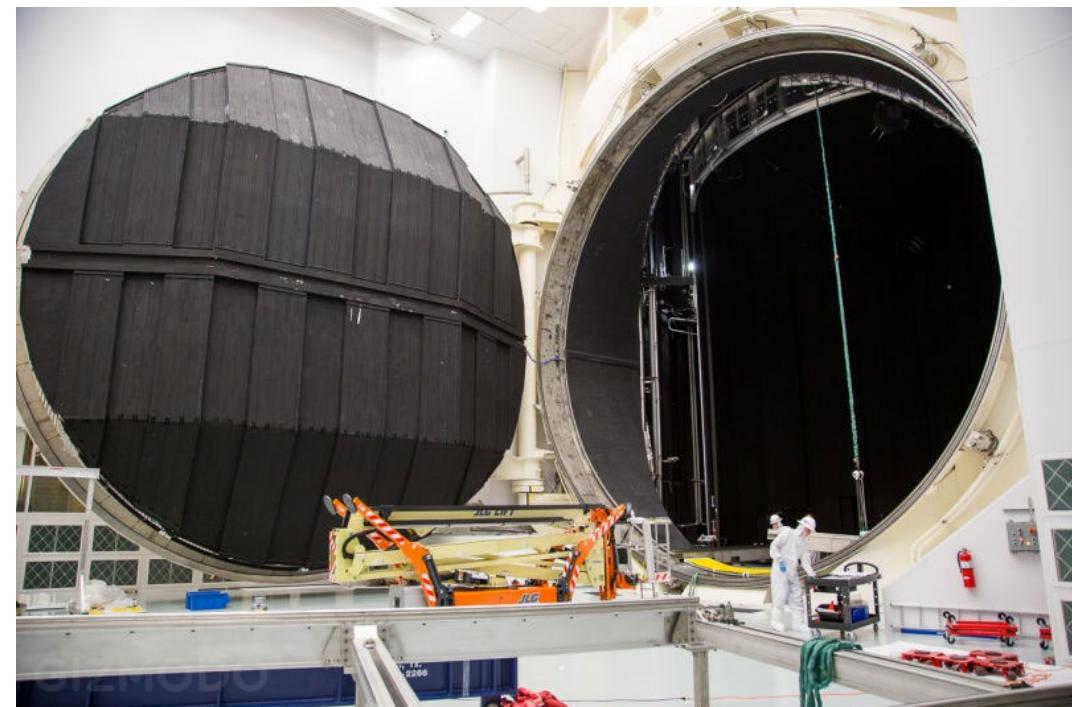
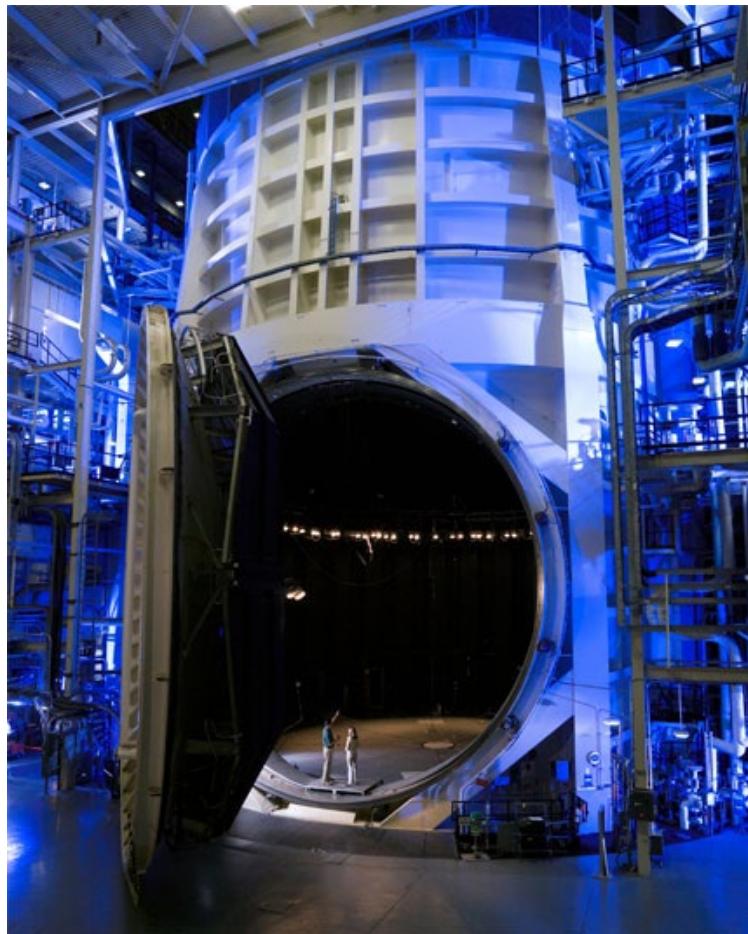


# Designing and Building the Verification System

- ▶ The verification system is ***another system*** that must be built while the system-under-test is being built
- ▶ Verification requirements must be defined:
  - Inspection - example: the drawing or specification against which the system shall be inspected
  - Analysis - example: the characteristics of the models to be used in the analysis
  - Demonstration - example: the stimulus for the demonstration and the response expected
  - Test - example: the HW and SW required to generate inputs and measure outputs
- ▶ Pass/fail criteria must be identified for all verification events



# Johnson Space Center Chamber A





**Johnson Space Center's Chamber A** is a 16.8 m (55 ft) diameter x 27.4 m (90 ft) high, thermal-vacuum test facility and is famous for testing the Apollo spacecraft, with and without the mission crew. Its usable test volume and high-fidelity space simulation capabilities are adaptable for thermal-vacuum testing of a wide variety of test articles, including entire space vehicles.

Door: 40' diameter; 40 tons; two person operation



# **Dynamic Test Stand**

## **Marshall Space**

## **Flight Center**

# Designing and Building the Verification System

[https://www.youtube.com/watch?v=\\_jfXX7qppbc](https://www.youtube.com/watch?v=_jfXX7qppbc)

<https://www.youtube.com/watch?v=73604Hz4Nk>

4

<https://www.youtube.com/watch?v=lgspliTfWIk>

# System Integration & Test

Integrate the system components

Verify that the system meets its specifications at every level of integration

Validate that the system satisfies the stakeholder needs

***What else is there to say?***



# Verification and Validation: A Fatal Flaw?

- **Verify** - to prove the truth of; confirm
- **Validate** - substantiate; confirm

**What's wrong with these definitions?**

- **Confirm** - establish the truth or correctness of something previously believed to be the true

**Question: What percentage of development occurs after testing has begun?**

# Example: Interface Definition

## Ariane 5 Launch System



- Ariane 5 is a launch system manufactured for the European Space Agency
- It succeeded Ariane 4, which had 113 successful launches between 15 June 1988 and 15 February 2003
- Eventually captured 50% of the market for commercial satellite launches
- It took 10 years and \$7 billion to produce Ariane 5
- The first Ariane 5 test flight took place on June 4, 1996 and had a price tag of \$500M.



# Flight 501: Failure of the Ariane 5

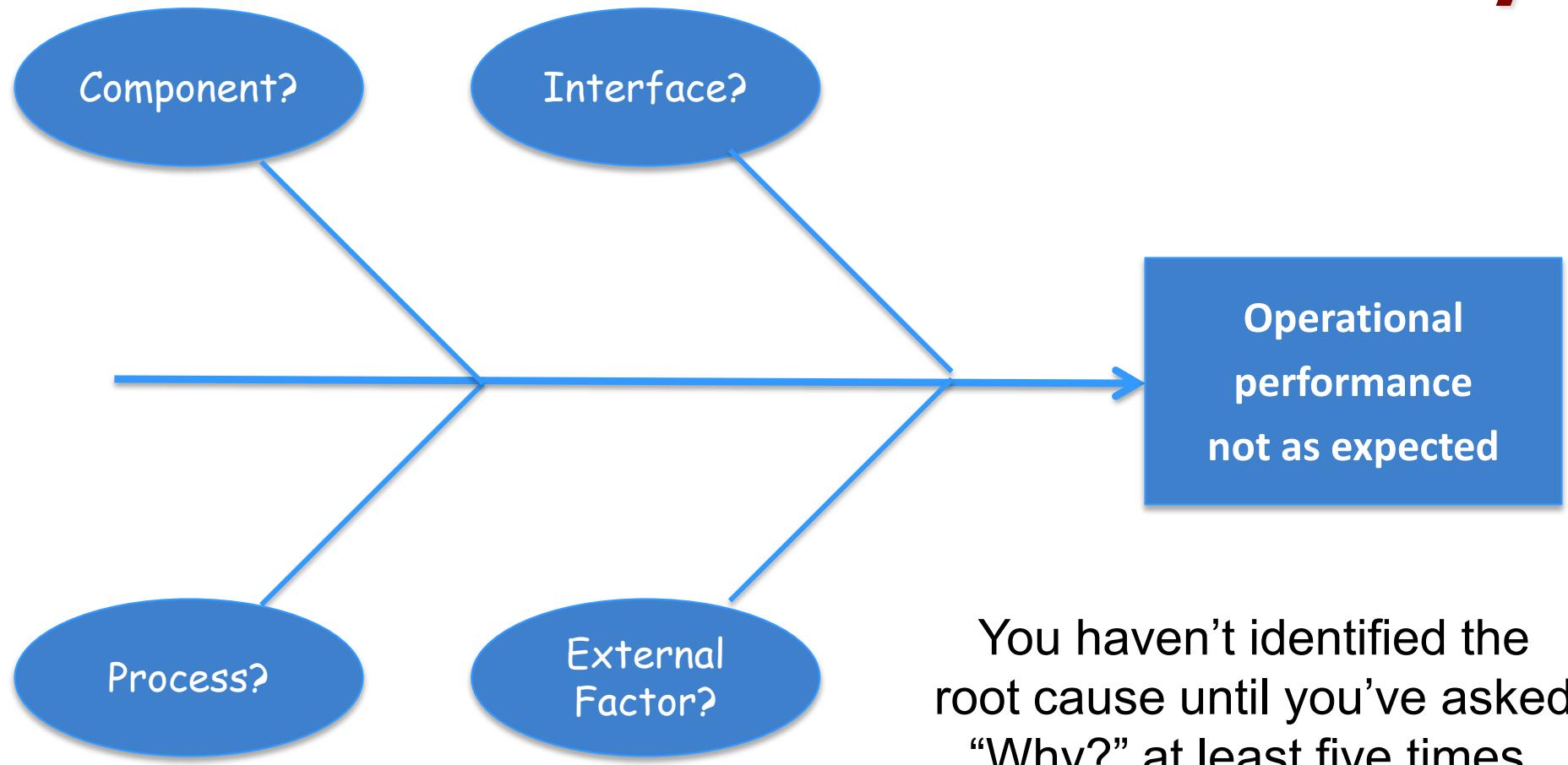


- On 4 June 1996, the maiden flight of the Ariane 5 ended in failure
- Everything was nominal until approximately 37 sec. after launch, when the vehicle suddenly veered off its flight path, broke up and exploded.
- Preliminary investigation of the flight data showed:
  - failure of the back-up Inertial Reference System, followed immediately by failure of the active Inertial Reference System
  - swiveling of the nozzles of the two solid boosters into the extreme position and, slightly later, of the Vulcain engine, causing the launcher to veer abruptly
  - self-destruction of the launcher triggered (correctly) by rupture of the links between the solid boosters and the core stage.

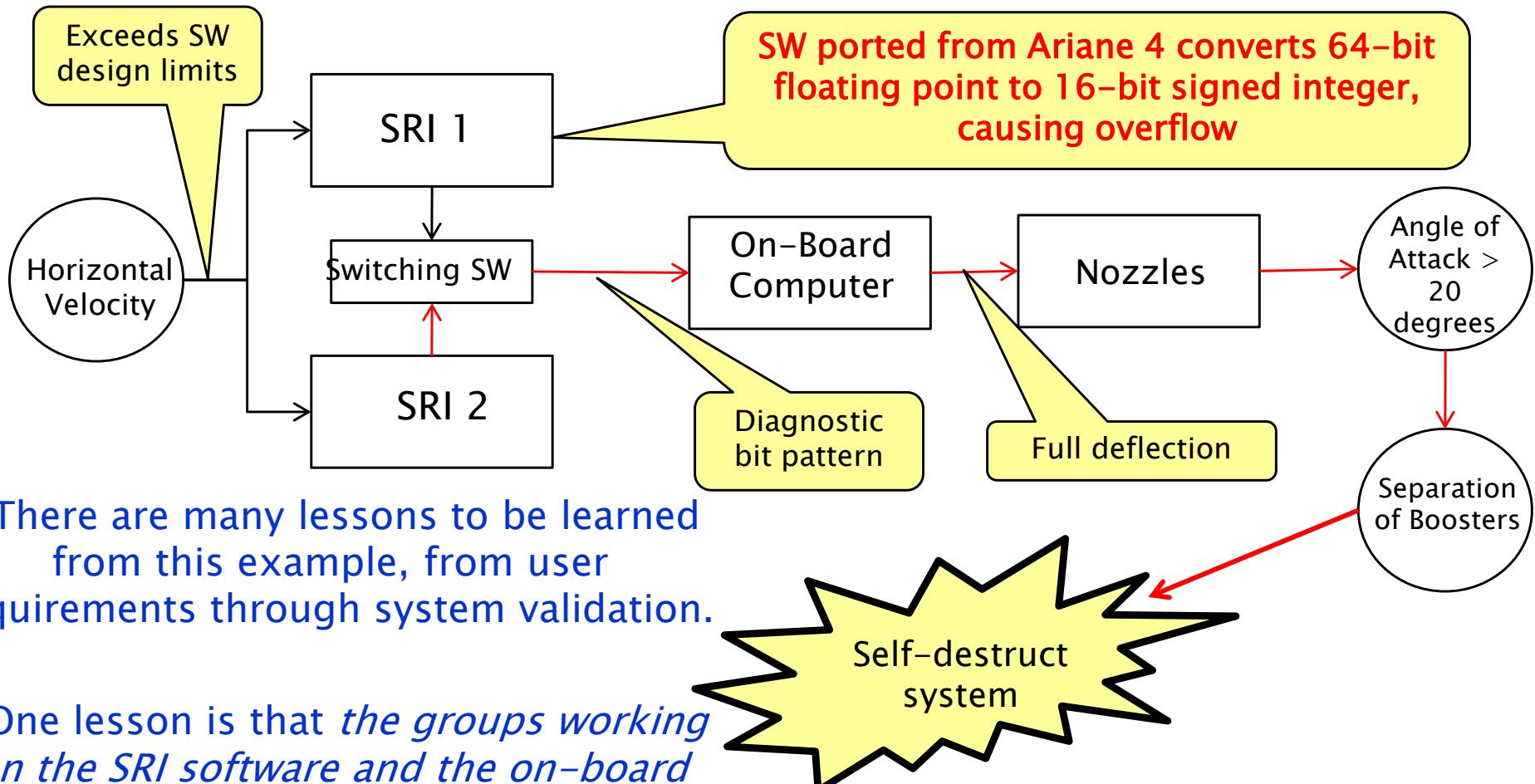


- [https://www.youtube.com/watch?v=PK\\_yguLapgA](https://www.youtube.com/watch?v=PK_yguLapgA)

# If it doesn't perform as expected... **Why?**



# Ariane 5 Failure Sequence



- There are many lessons to be learned from this example, from user requirements through system validation.

- One lesson is that *the groups working on the SRI software and the on-board computer software did not have a common view of the interface.*

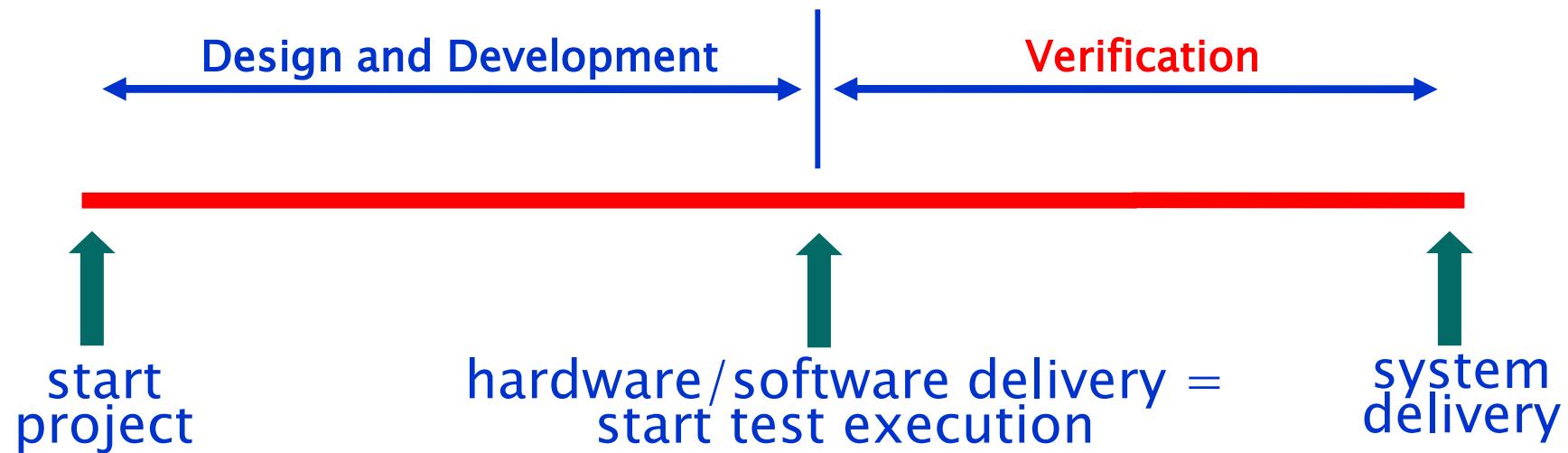
# Thought Experiment: Verification Accountability

- Suppose you are responsible for the verification of a newly developed system
  - You are accountable for system quality
  - You are given a date on which you will receive the integrated system and a date on which the system is to be deployed
- You will be promoted if you succeed, fired if you fail

***What keeps you up at night?***

# You are responsible for verification

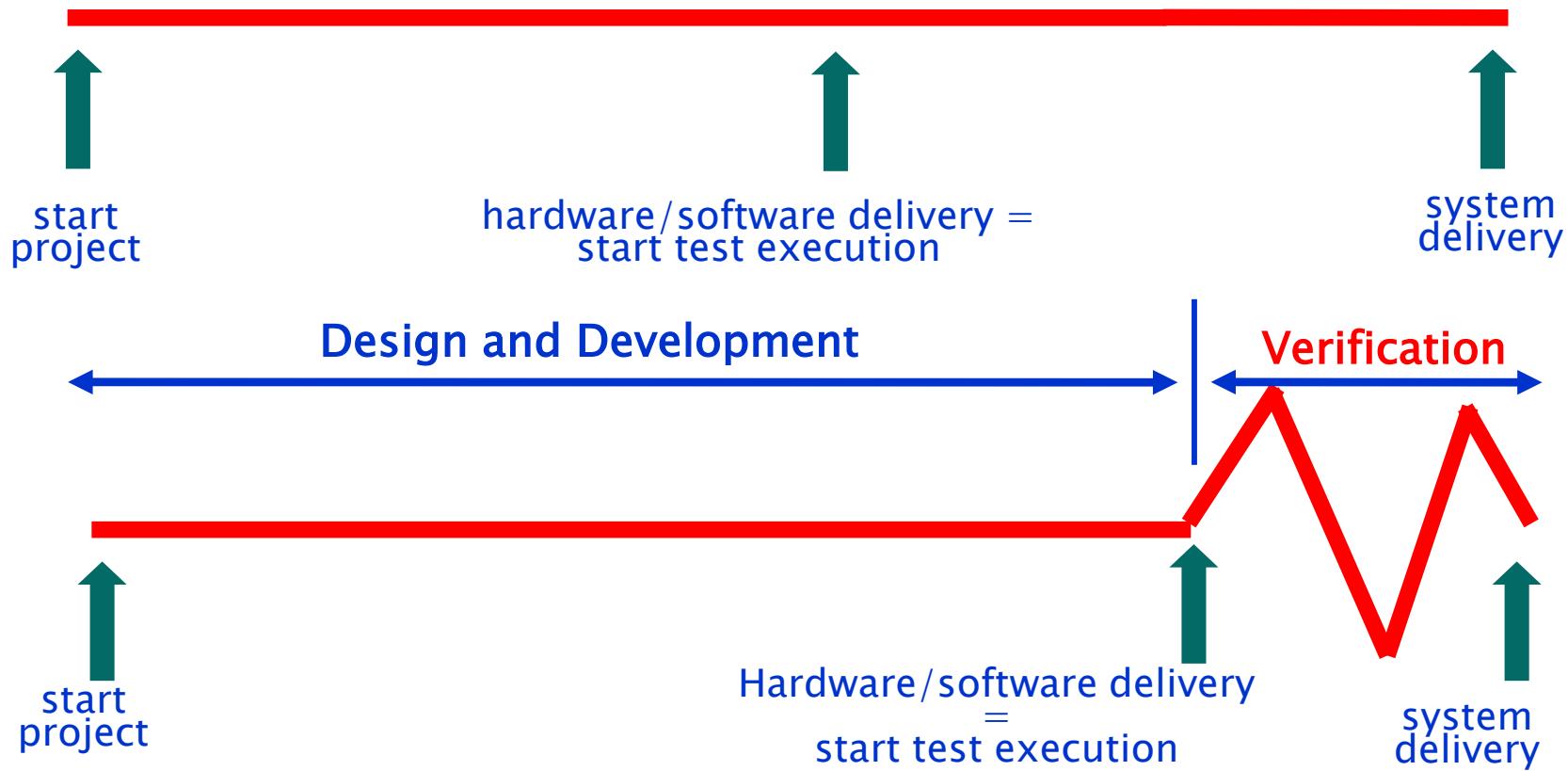
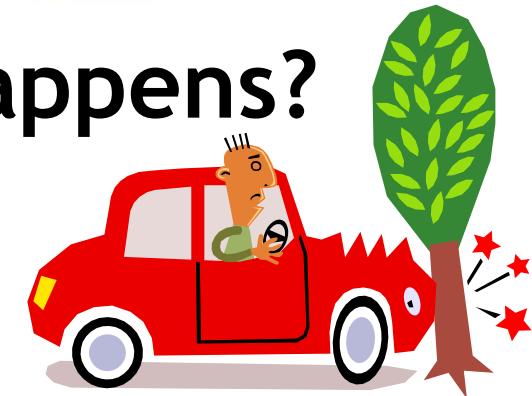
System Schedule:



# Do you think this ever happens?

Slipping test-start dates...

...fixed system delivery deadlines





# What's the Big Deal?



# Do you think this ever happens? *The F-35 Lightning II*



“More than 150 operational F-35s have been delivered to date, and each will need extensive modification to the Block 3F standard once development concludes.” ...*February 1, 2016*

# What keeps you up at night?

The system delivery to verification is delayed

The deployment date is firm

What do you do?

# Some Testing Principles

- Testing starts during the requirements phase
- The developer should generally not be the tester
  - Best practice is Independent Verification & Validation (IV&V)
  - But the developer should understand the tests!
- A test case specifies the test inputs and the expected outputs
- Test cases should also cover invalid and unexpected inputs
- Test cases should test that the item under test does what it should do and that it does not do what it should not do
- A test is successful when it detects an error
- No risk, no test



# System Validation

*Did we build the right system?*

# Verification vs. Validation

## Verification

- Controlled Environment
- Demonstrate that requirements are met
- Individual Tests
- May include some external interfaces
- Comprehensive
- Conducted by engineers
- Prove that implementation is correct

The system is built right!

## Validation

- Operational Environment
- Demonstrate that stakeholder needs are met
- Operational Scenarios
- Must includes all external interfaces
- Focuses on major features
- Conducted with stakeholders
- Determine that Customer is satisfied

The right system is built!

# Validation Challenges

Operational conditions are difficult to replicate:

- External system interfaces
- Operational processes and expertise
- Traffic levels and stress conditions
- Performance under all conditions

Exhaustive testing is never possible:

- Too many input combinations
- Too many input sequences
- Too many invalid and unexpected inputs

Real-world environment!

# Design and build the validation system

- ▶ The validation system is **yet another system** that must be built while the system-under-test is being built
- ▶ And the verification system must be preserved and maintained in order to diagnose problems

# System Validation Experience

- Despite all efforts to:
  - capture Stakeholder Needs,
  - develop System Requirements,
  - monitor and manage System Design,
  - test throughout Development and Integration,
  - conduct comprehensive, effective System Verification,
  - and execute Continuous Early Validation...
- The probability of identifying new issues with the system during final validation, deployment and operation is:

**Roughly 100%**

**Until you can reliably replicate a fault:**

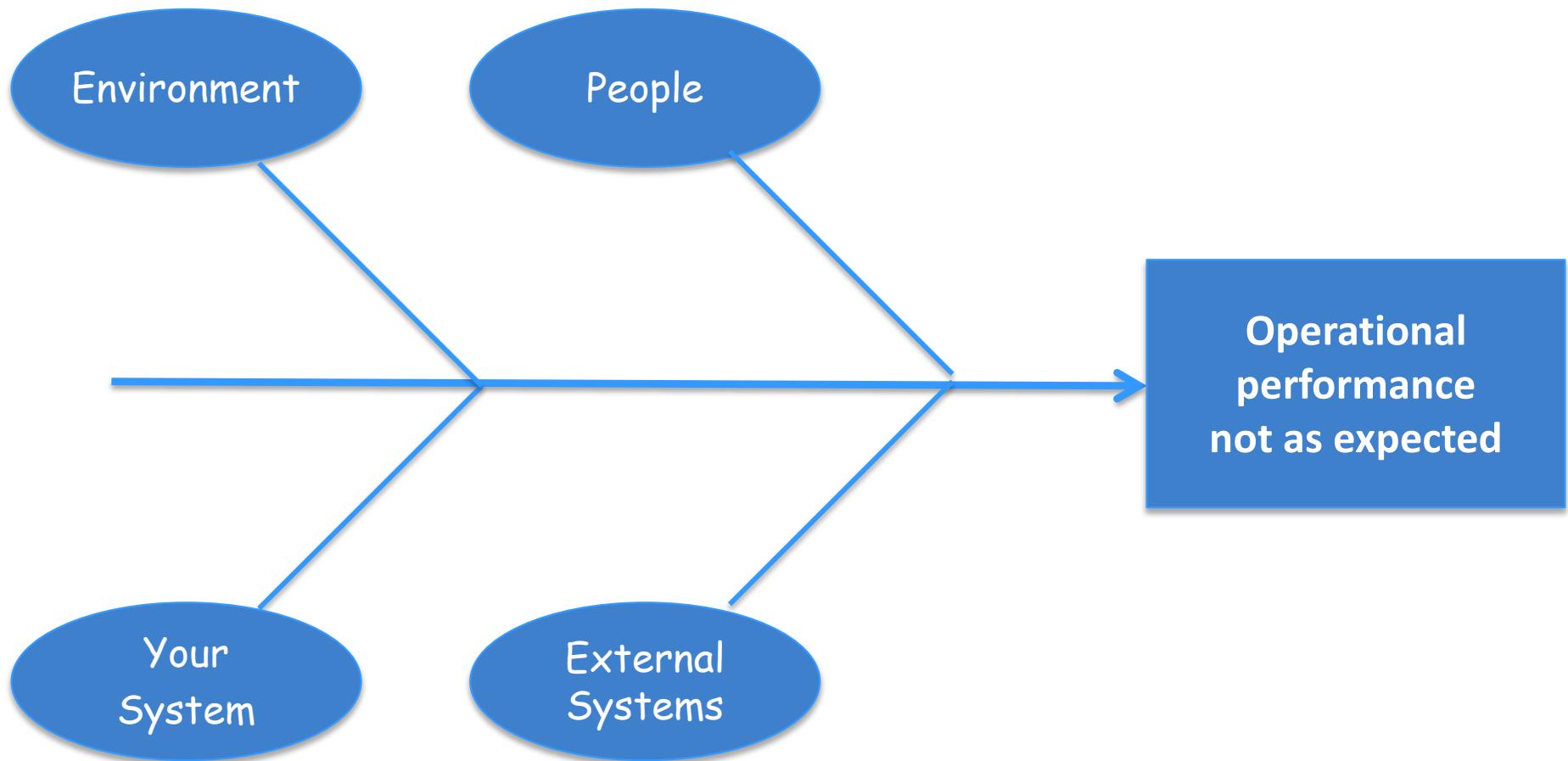
You have no knowledge of its cause...

You have no reasonable estimate of when the  
fault will be resolved...

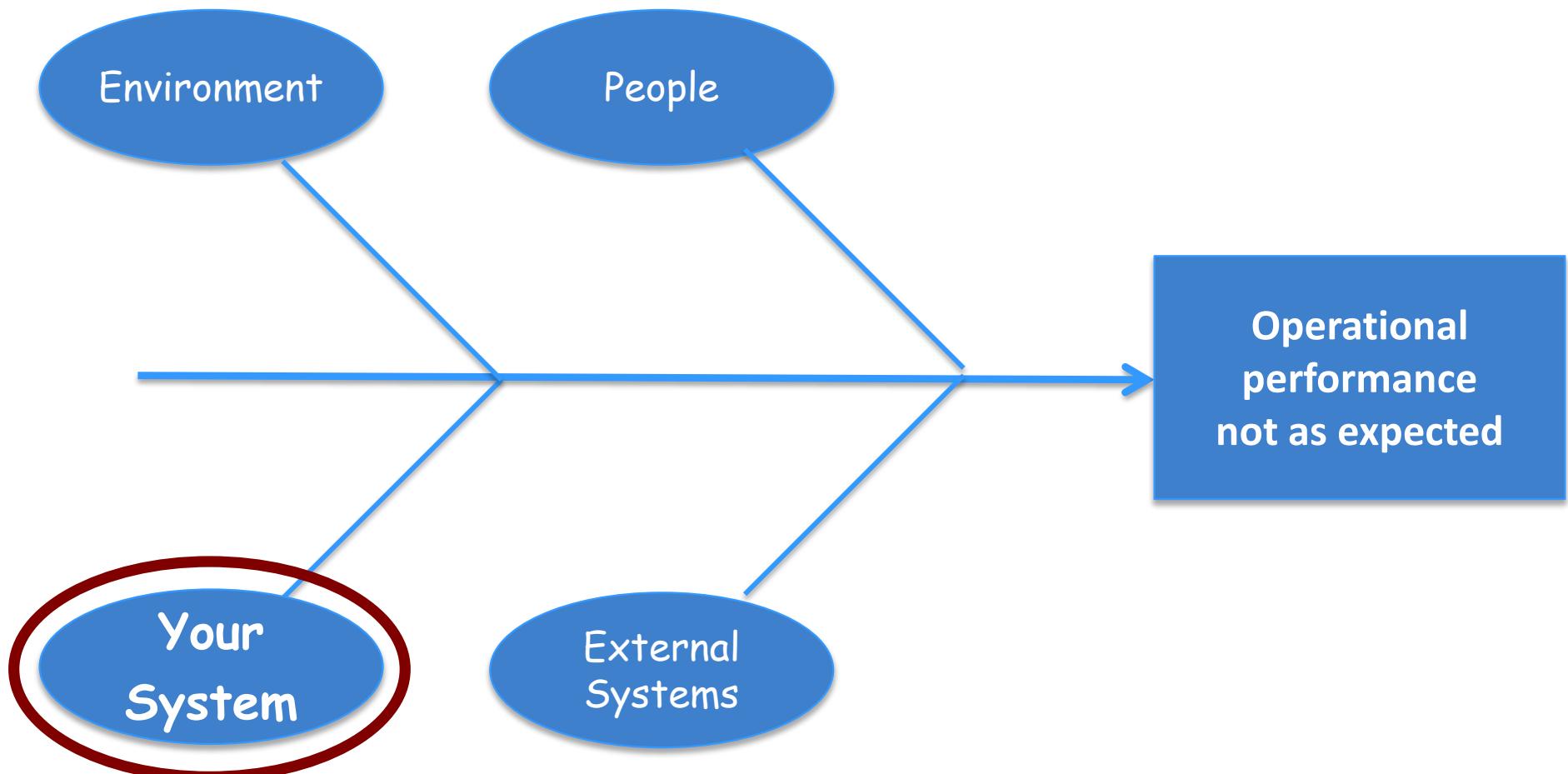
**You have nothing useful to say to your  
customer!**



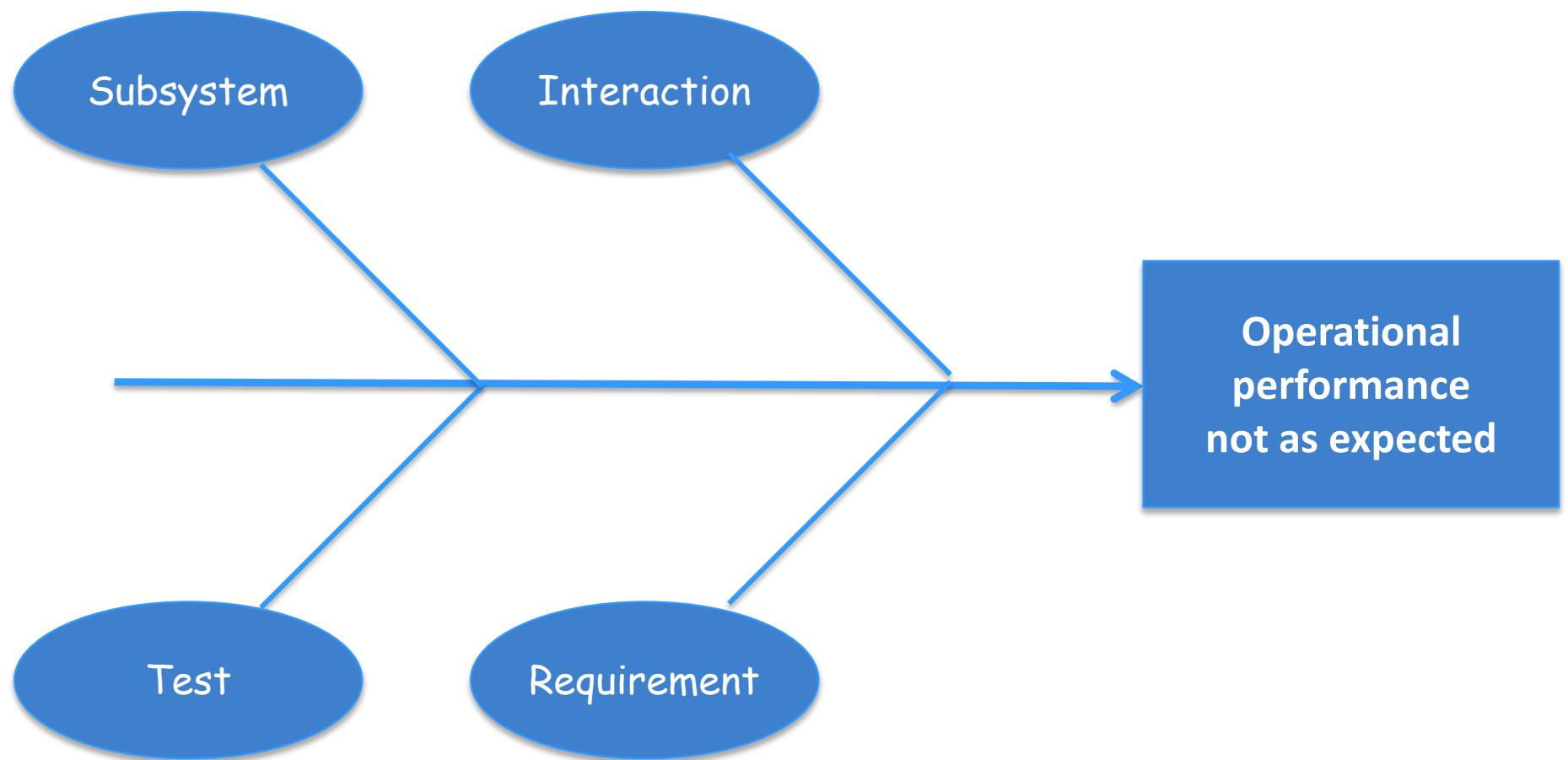
So it doesn't perform as expected...  
*Now what?*



# So it doesn't perform as expected... *Now what?*



# Once you're convinced it's your system, can you replicate the results in the lab?



# Boeing 787 Dreamliner Batteries



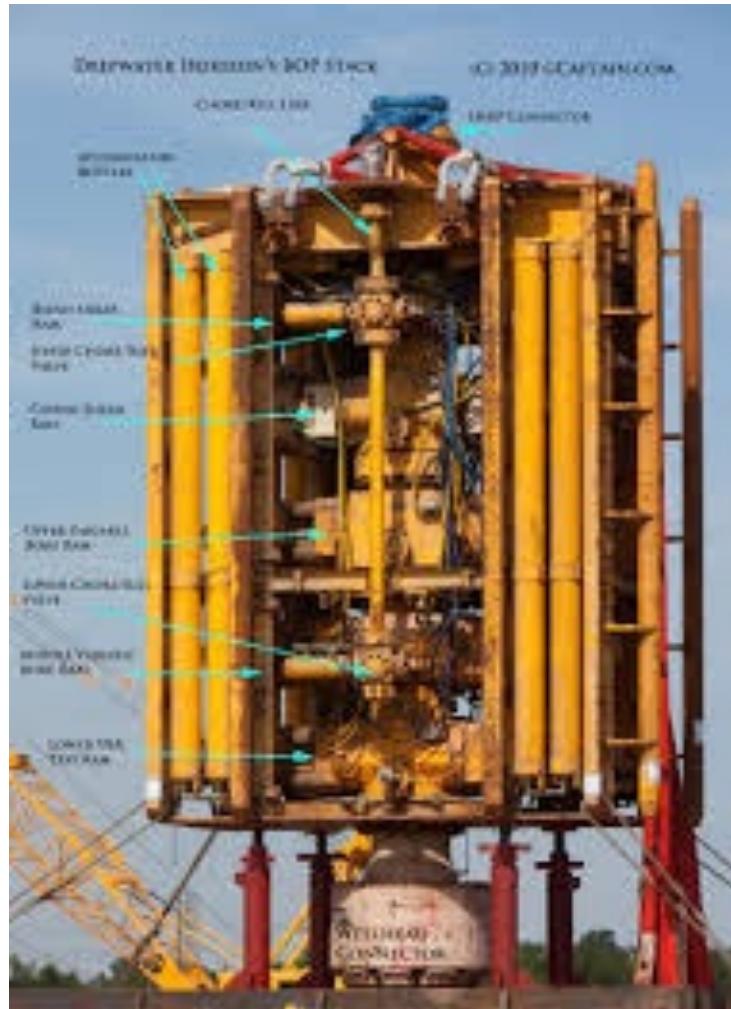
- GS Yuasa did not test the battery under the most severe conditions possible in service.
- The test battery was different than the final battery design certified for installation on the airplane.
- Boeing's safety assessment for the 787 battery did not provide the engineering rationale and justifications for a key assumption or consider the consequences if the assumption were incorrect.

*Ref: NTSB Incident Report NTSB/AIR-14/01 PB2014-108867*

# Hubble Space Telescope



- Two “null-corrector” tests were used to measure the shape of the mirror.
  - Two indicated there was a problem—but the company incorrectly believed the third was right.
- Two other tests could have been conducted but were not:
  - The primary mirrors could have been swapped and tests rerun.
  - An end-to-end test could have been conducted on the assembled system.
- The eventual repairs cost over \$100 million; total cost for the mission was \$674 million.



# Deepwater Horizon

According to a study reported in the New York Times:

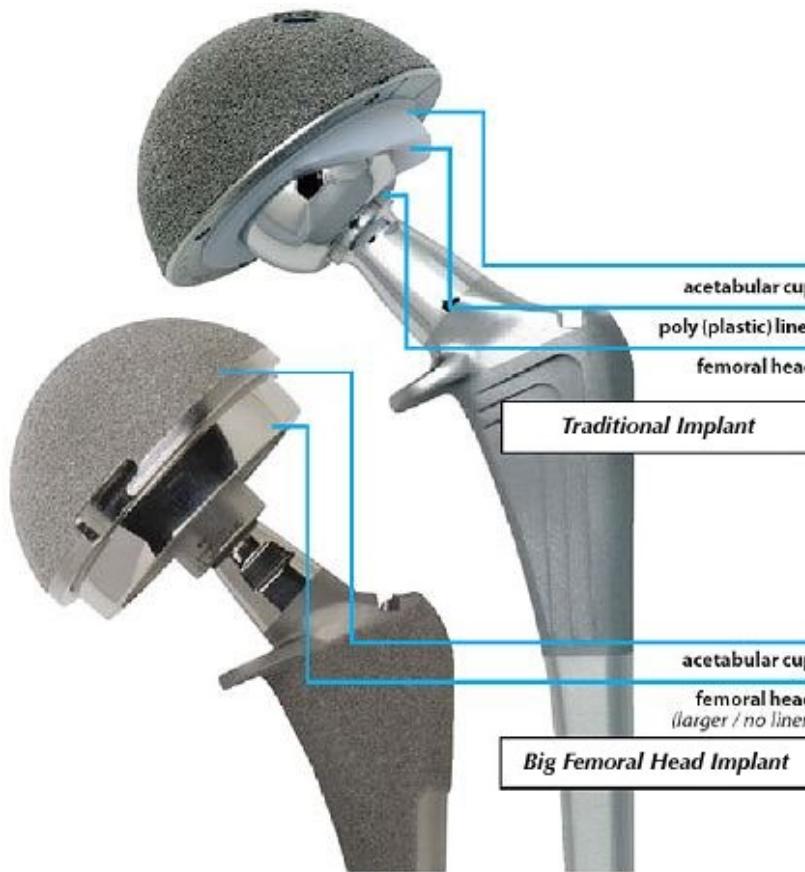
- There were only *62 failures out of nearly 90,000 tests conducted over several years.*
  - But the study raised questions about the effectiveness of these tests:
    - *“It is not possible to completely simulate the actual conditions of deep water wells.”*
  - Another study described a mentality of, “I don’t want to find problems; I want to do the minimum necessary to obtain a good test.”

# Toyota “Sudden Acceleration”



- The National Highway Traffic Safety Administration (NHTSA) enlisted NASA engineers who evaluated the electronic circuitry and analyzed more than 280,000 lines of code.
- The NASA engineers found no electronic flaws capable of producing dangerous high-speed unintended acceleration.
- The NHTSA is [still] considering taking several new actions as the result of the findings.

# All-metal Artificial Hips



- The F.D.A. does not require most all-metal hips to undergo clinical trials before sale.
- Instead, they are tested in labs on machines that simulate millions of steps to study the forces exerted by years of motion.
- These tests did not point to problems, but the simulations were based on idealized conditions, not on what would happen in the real world of doctors and patients.
- Small variations in how they are implanted can generate debris.

# A Word About Acceptance Testing

- **Validation**

- Operational Environment
- Demonstrate that stakeholder needs are met
- Operational Scenarios
- Must includes all external interfaces
- Focuses on major features
- Conducted with stakeholders
- Determine that customer is satisfied

## Acceptance Testing

- Acceptance tests are a subset of validation tests
- Involve specific tests required by the customer
- We perform acceptance tests in order to get paid
- We perform verification so we don't have to go back after the system is delivered

The right system is built!

The customer agrees!

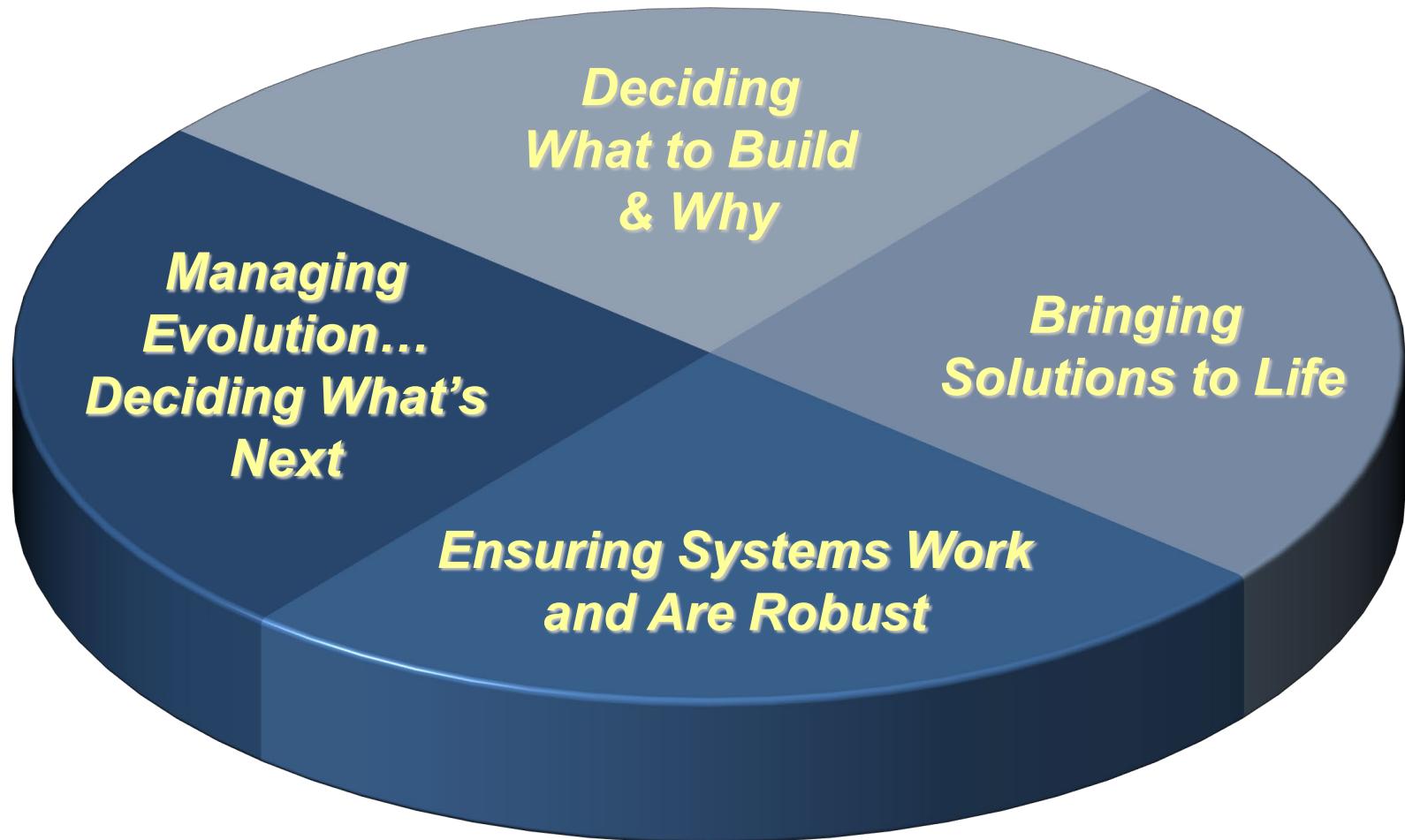
# Integration and Test Key Questions:

1. How will you integrate and test the components of your system?  
Why?
  - *Assessment Criteria: Plan in place to systematically test all key interfaces and risk items against corresponding specifications*
2. What resources and facilities will you need to verify that the system meets its specifications?
  - *Assessment Criteria: Test systems fully designed and plans in place to ensure they are available when needed*

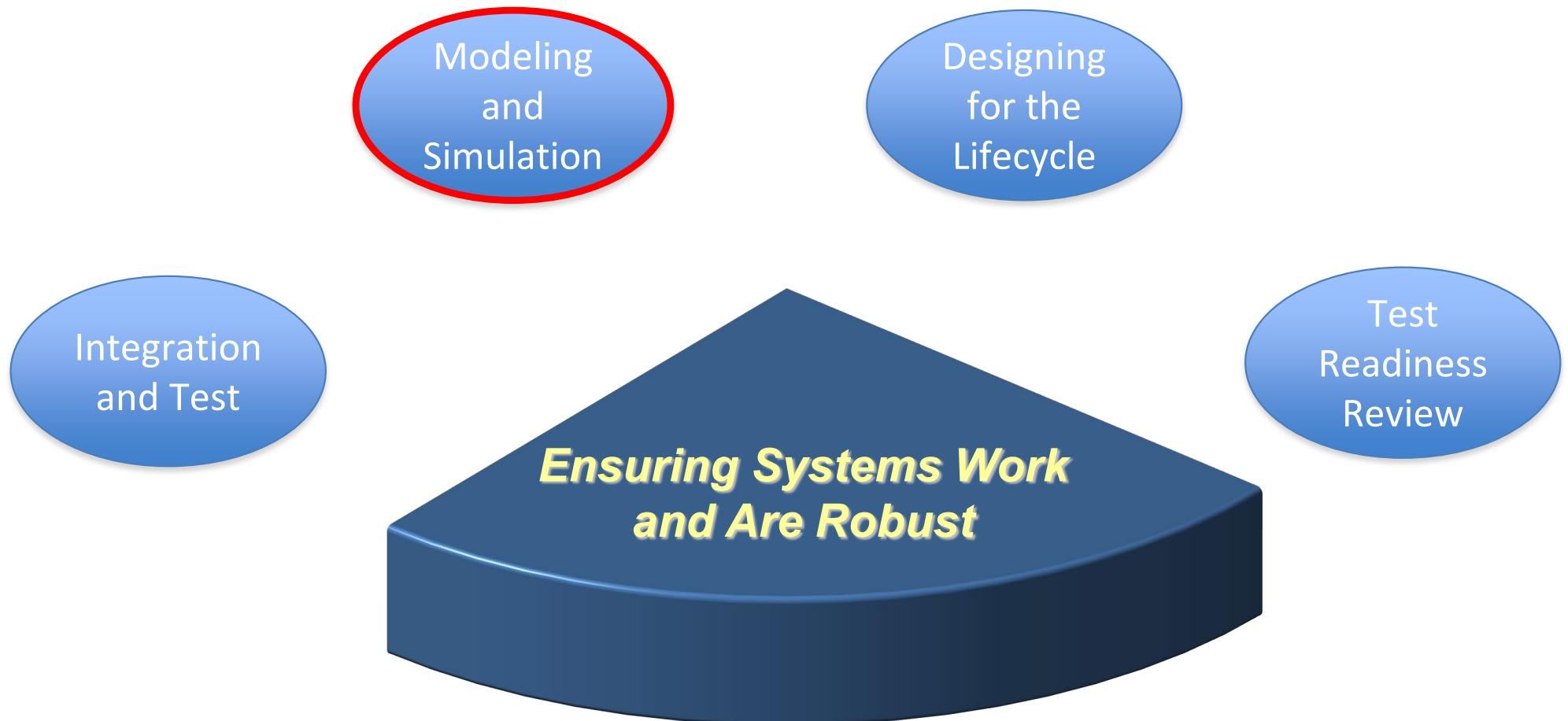
# Integration and Test Key Questions:

3. How will you prove to your customer that the system meets his/her needs in the operational environment?
  - *Assessment Criteria: Plans in place to demonstrate that the system meets the original customer needs and to gain final customer acceptance*
4. How will you respond when problems arise?
  - *Assessment Criteria: Plans and resources in place to diagnose, fix and test field-found faults and to certify that corrective actions have been successful*

# Course Design



# Course Design



# Modeling and Simulation Key Questions:

1. *How could you utilize Modeling and Simulation to Ensure that your System Works and is Robust?*  
*Why?*  
*When?*
  
2. *How could you generate the Simulation inputs?*
  
3. *How could you be sure that you can be confident in the Simulation results?*

**SSE**

School of  
Systems&Enterprises



# Experiments

- An experiment is the process of extracting information from a system by exercising its inputs.
- To perform an experiment on a system it must be both controllable (for inputs) and observable (for outputs).
- -Inaccessible and uncontrollable inputs are sometimes called disturbance inputs.
- -Unobservable outputs are sometimes called internal states.
- Practical problems associated with experiments.
  - -The experiment might be too expensive
  - -The experiment might be too dangerous (chemical systems)
  - -The system needed for the experiment might not yet exist

# Model

- A model of a system is anything an "experiment" can be applied to in order to answer questions about that system.
- This implies that a model can be used to answer questions about a system without doing experiments on the real system.
- A model is always related to the system it models and the experiments it can be subject to.
- A model of a system might be valid for one experiment on the model and invalid for another.
- The term model validation, always refers to an experiment or a class of experiments to be performed.

# Model

- A representation of an object, a system, or an idea in some form other than that of the entity itself. (Shannon)

## Type of Models

- Different kinds of models depending on how the model is represented
- -Mental model—a statement like "a person is reliable" helps us answer questions about that person's behavior in various situations.
- -Verbal model—this kind of model is expressed in words. For example, the sentence "More accidents will occur if the speed limit is increased" is an example of a verbal model. Expert systems is a technology for formalizing verbal models. Many models in sociology and psychology are verbal models.
- -Physical model—this is a physical object that mimics some properties of a real system, to help us answer questions about that system. For example, prototype buildings, airplanes, etc.-
- Mathematical model—a description of a system where the relationships between variables of the system are expressed.

# Simulation.

- A simulation is an experiment performed on a model
- Examples:-
- A simulation of an industrial process such as steel or pulp manufacturing, to learn about the behavior under different operating conditions in order to improve the process.
- -A simulation of vehicle behavior, e.g., of a car or an airplane, for the purpose of providing realistic operator training.
- -A simulation of a simplified model of a packet-switched computer network, to learn about its behavior under different loads in order to improve performance. It is important to realize that the experiment description and model description parts of a simulation are conceptually separate entities?
- The value of the simulation results is dependent on how well the model represents the system.

# Simulation

- Experiments are too expensive, too dangerous, or the system to be investigated does not yet exist.
- The time scale of the dynamics of the system is not compatible with that of the experiment.
- For example, it takes millions of years to observe small changes in the development of the universe.
- Variables may be inaccessible in real system. In simulation all variables can be controlled.
- Easy manipulation of models.
- Suppression of disturbances and suppression of second-order effects. This can allow us to isolate particular effects and thereby gain a better understanding of those effects.

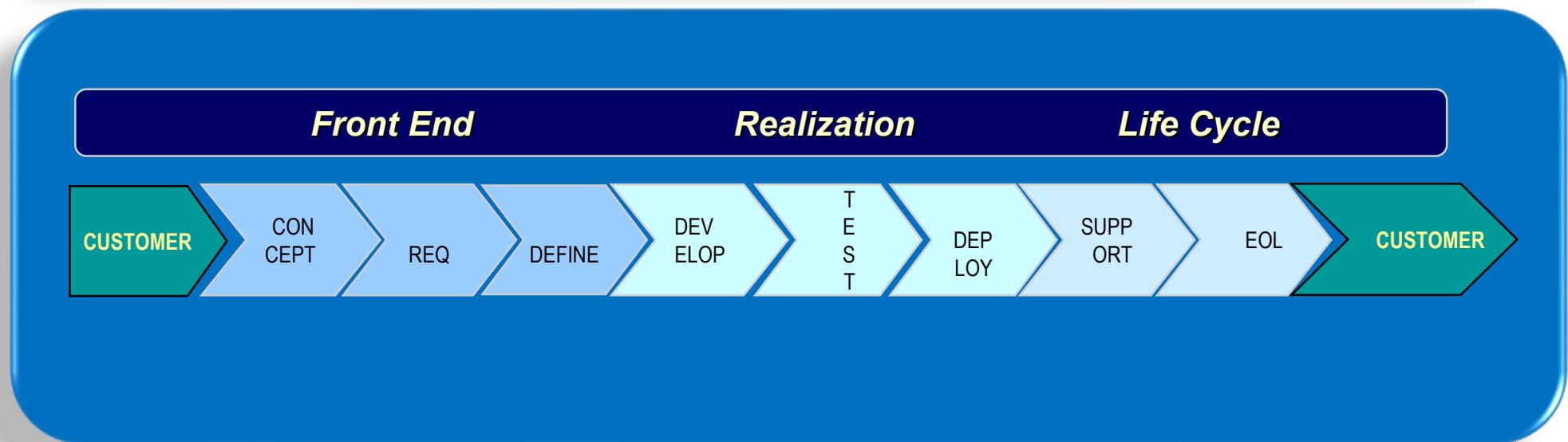


- Simulation Experience??

# Model and Simulation in Testing

- <https://www.youtube.com/watch?v=ulsn-EGs1p4&feature=youtu.be>
- <https://www.youtube.com/watch?v=B9eYCG0lr5Y&feature=youtu.be>

# *When should you Ensure that a System Works and is Robust?*



## *How ?*

# *Ensuring Systems Works and are Robust*

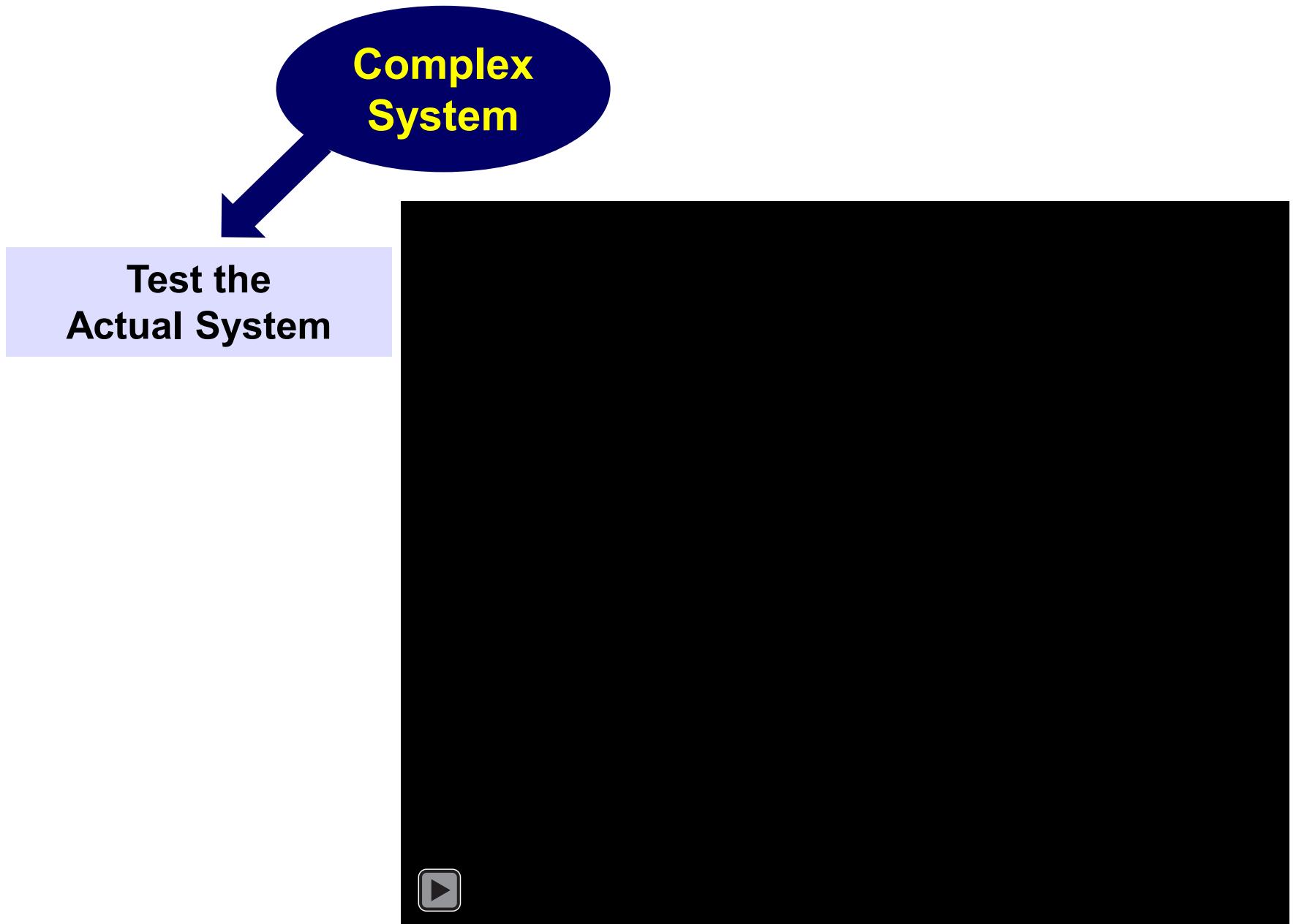
Complex  
System



**Test the  
Actual System**



# *Ensuring Systems Works and are Robust*

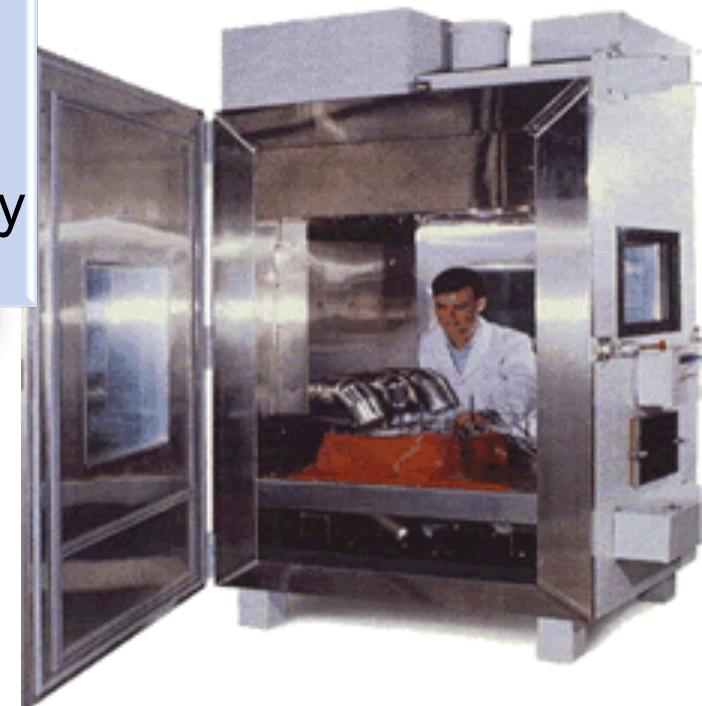


# *Ensuring Systems Works and are Robust*



## Stress Testing

Goal: Improve System Reliability



## Reliability Qualification Testing

Goal: System Reliability Measurement



# *Ensuring Systems Works and are Robust*

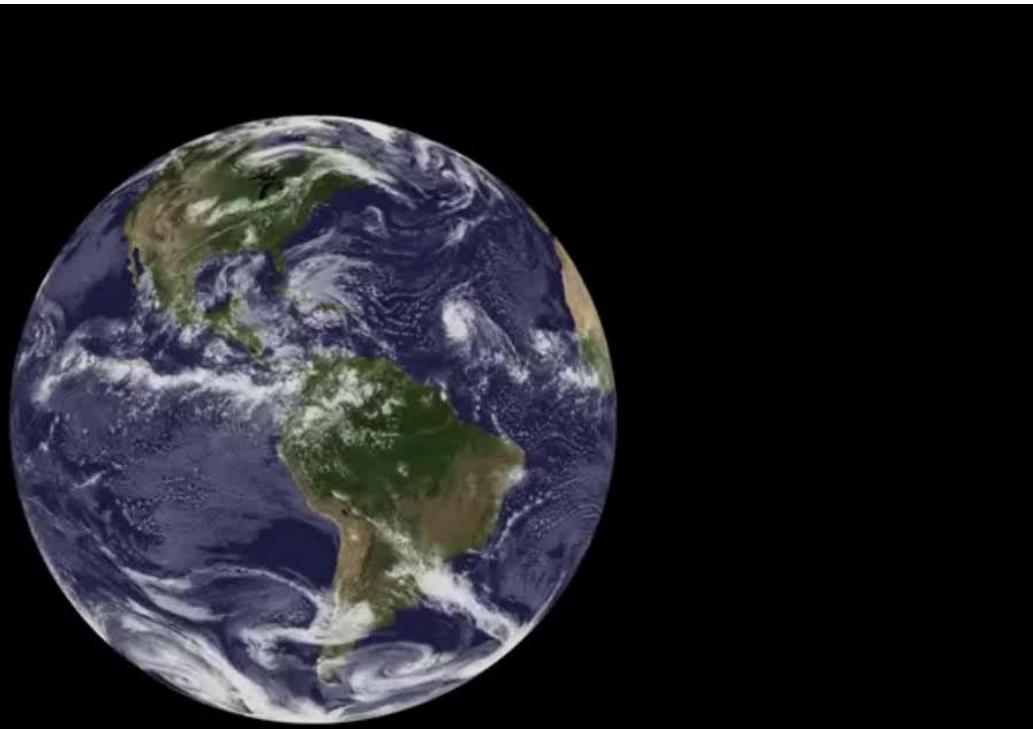
Complex System

Test a Model of  
the System

Physical  
Model



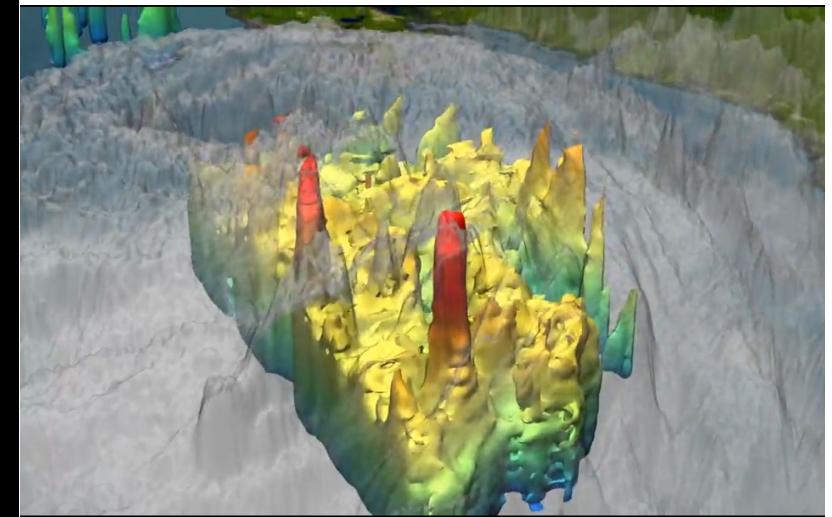
# *Ensuring Systems Works and are Robust*



**Complex  
System**

**Test a Model of  
the System**

**Mathematical/  
Computational  
Model**



# **SYSTEM INTEGRATION, VERIFICATION AND VALIDATION THROUGH MODELING AND SIMULATION**

## **System Design**

- Conceptual M&S
- Continuous M&S
- Stochastic M&S
- System Dynamics M&S
- Monte Carlo M&S
- Agent-Based M&S

### **Mechanical Design**

- Finite Element Methods FEM
- Vibration Analysis
- Stress/Strain Analysis
- Failure Analysis

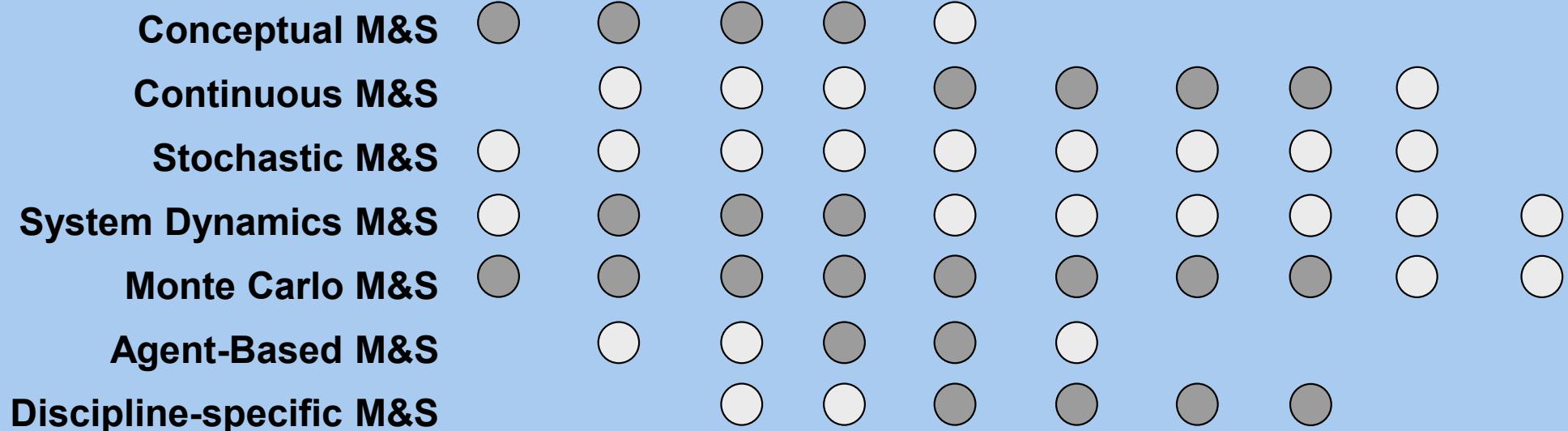
### **Electrical Design**

- SPICE
- Circuit Synthesis
- Functional verification
  - VHDL
- Test Vector Generation

### **Software Design**

- Unified Modeling Language UML
- Extensible Markup language XML
- Cost Estimation Model COCOMO

# System Modeling & Simulation



**Mathematical/  
Computational  
Model**

- = **Strong reliance on M&S**
- = **Medium reliance on M&S**
- = **Low reliance on M&S**

# **Complex Systems Design M&S**

## **Trends**

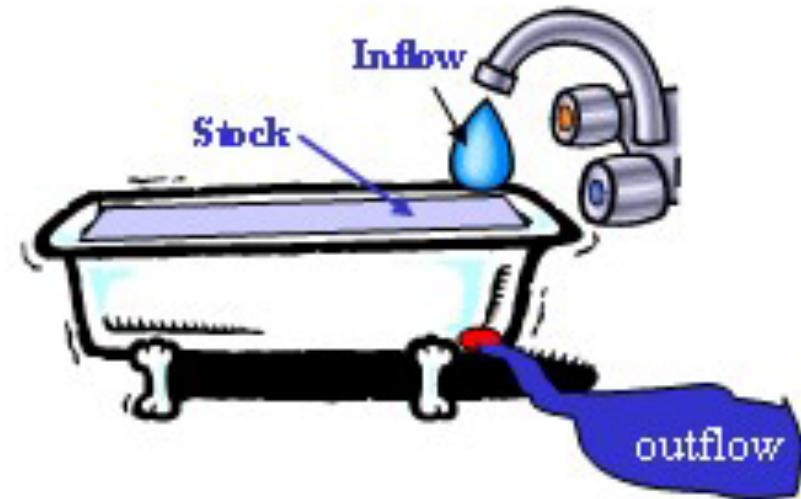
- System Dynamics M&S
- Agent-Based M&S
- MMOGs

# **Complex Systems Design M&S**

## **Trends**

- 
- **System Dynamics M&S**
  - Agent-Based M&S
  - MMOGs

# System Dynamics M&S

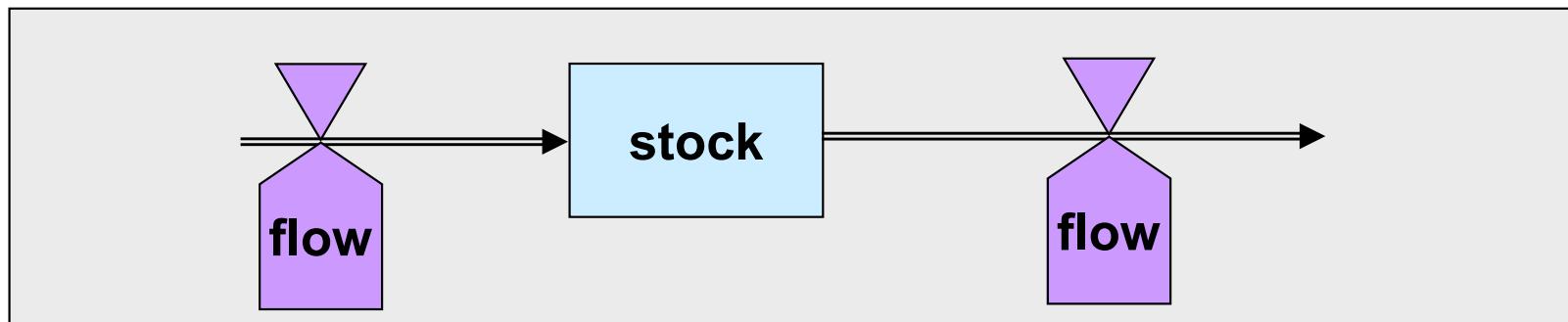


- Modeling the dynamic behavior of complex systems over time
  - Decomposing the system into simpler interconnected components
- Each component's state is defined by a set of differential equations that underlie the dynamics of the component
- Models the aggregated cause-effect relationship between variables

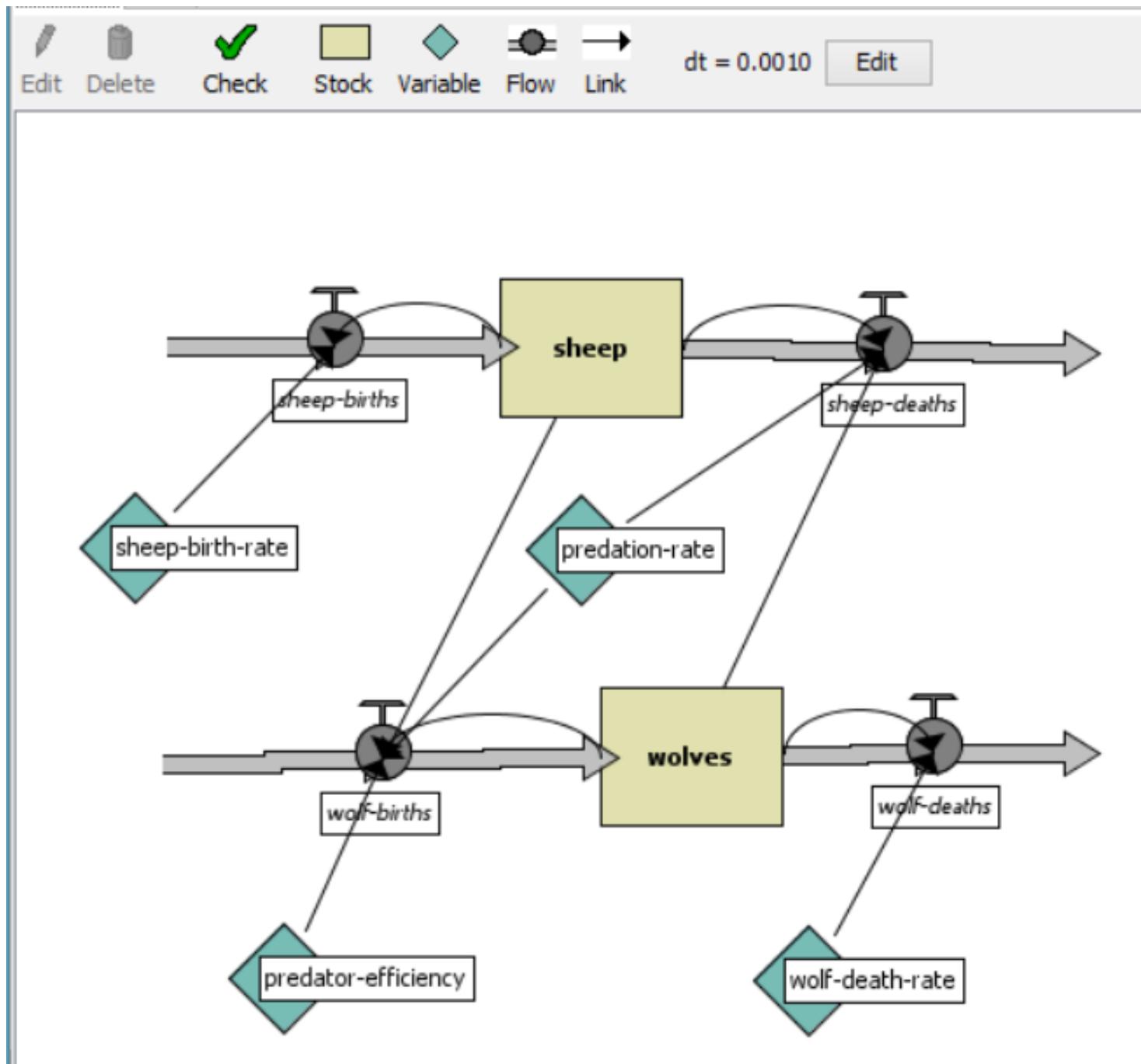
# Developing a System Dynamics Model

## *Stock and Flow Model*

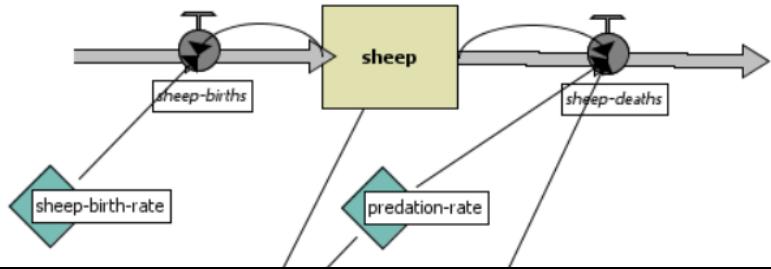
- **Stock (*level*)**
  - Something that is accumulated over time by inflows and/or depleted by outflows
    - stocks can only be changed via a *flow*
  - Stocks are shown graphically as a box
- **Flow (*rate*)**
  - Something that modifies a stock over time
    - inflows (adding to the stock)
    - outflows (subtracting from the stock)
  - Flows are shown graphically as metered (e.g., valve) interconnects



# System Dynamics Model



# System Dynamics Model



# Complex Systems Design M&S Trends

- System Dynamics M&S
- Agent-Based M&S
- MMOGs



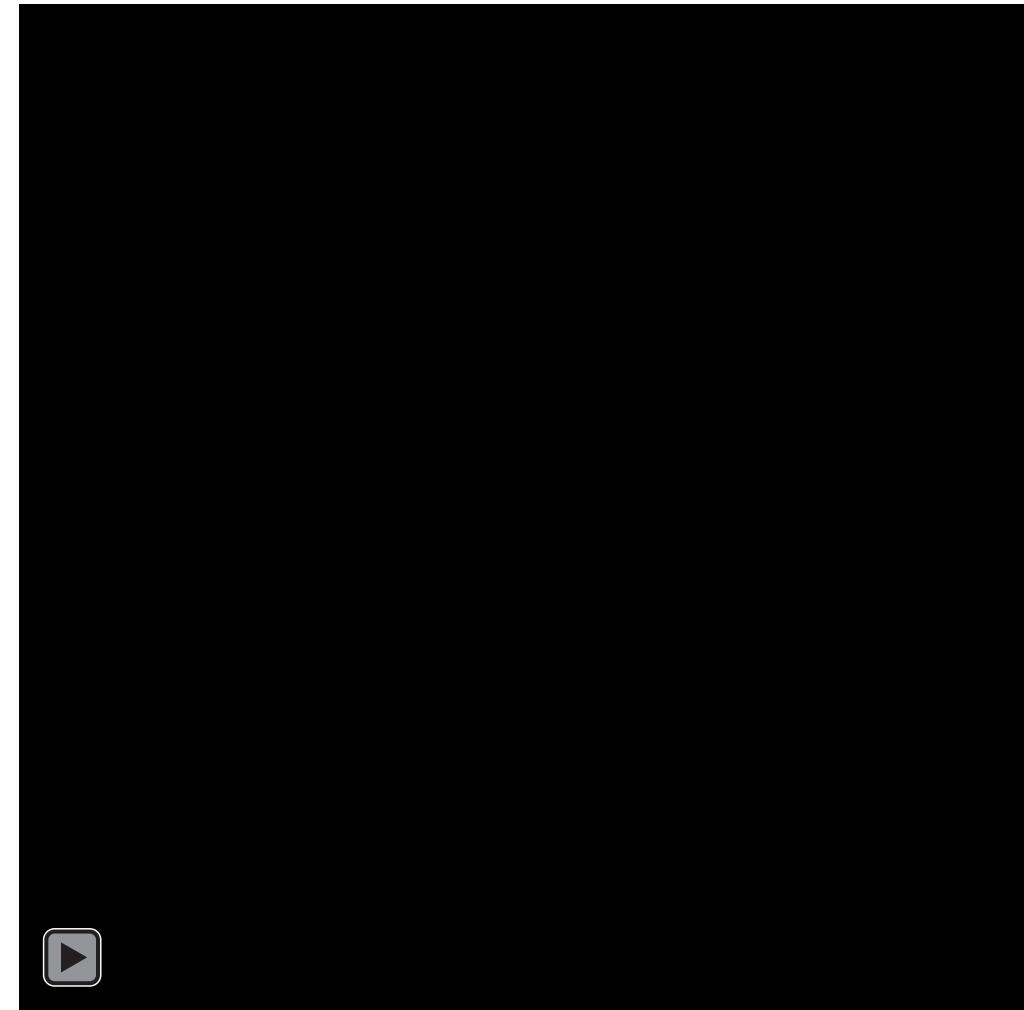
# *Agent-Based M&S*

➤ **Computational modeling approach used to study complex systems**

- Systems that are composed of multiple interacting entities and
- Exhibit emergent behavior

➤ **A bottom-up modeling approach**

- Agents
- Environment



# Agent-Based Models

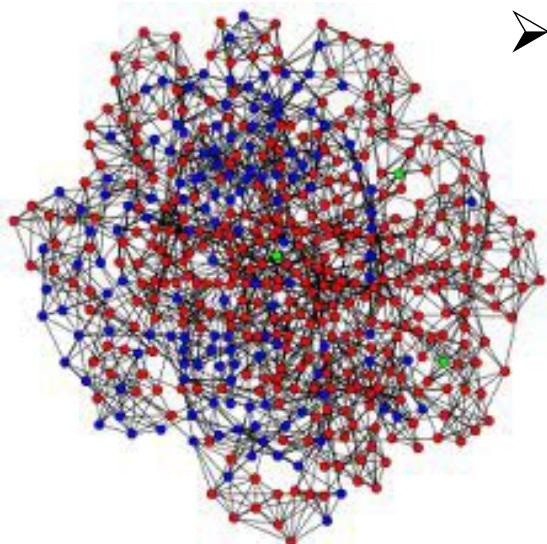
**“the modeling of (complex systems) is seen to be precisely the type of modeling challenge that agent-based modeling is designed to address”.**

*National Research Council  
2008*



## **Agents**

- Individual Entities
- Interact with other Agents in an Environment
  - Transmit information
  - Receive information
  - ‘Learn’
- Modeled to react ‘realistically’



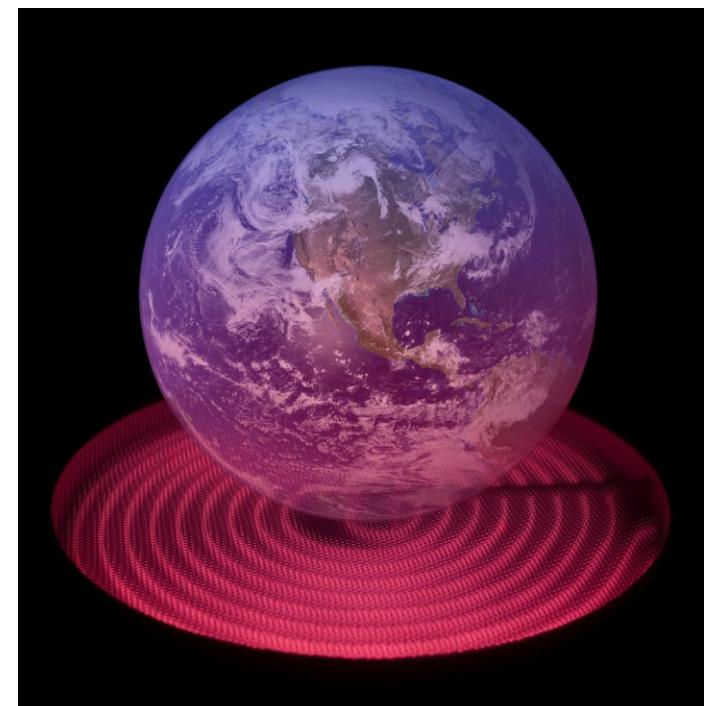
- Features
  - Autonomy
  - Social Ability
  - Reactivity
  - Pursuit of individual goals
  - Perception
  - Memory
  - Policy (rules)

## *Environment*

➤ Virtual world in which the agents operate

- Often represented as a graphical geometric space ('*specially explicit*')
  - e.g., 10 x 10 matrix
  - e.g., Floor plan
- Sometimes represent non-geometric space
  - e.g., Knowledge space models

➤ Modeled to represent 'reality'

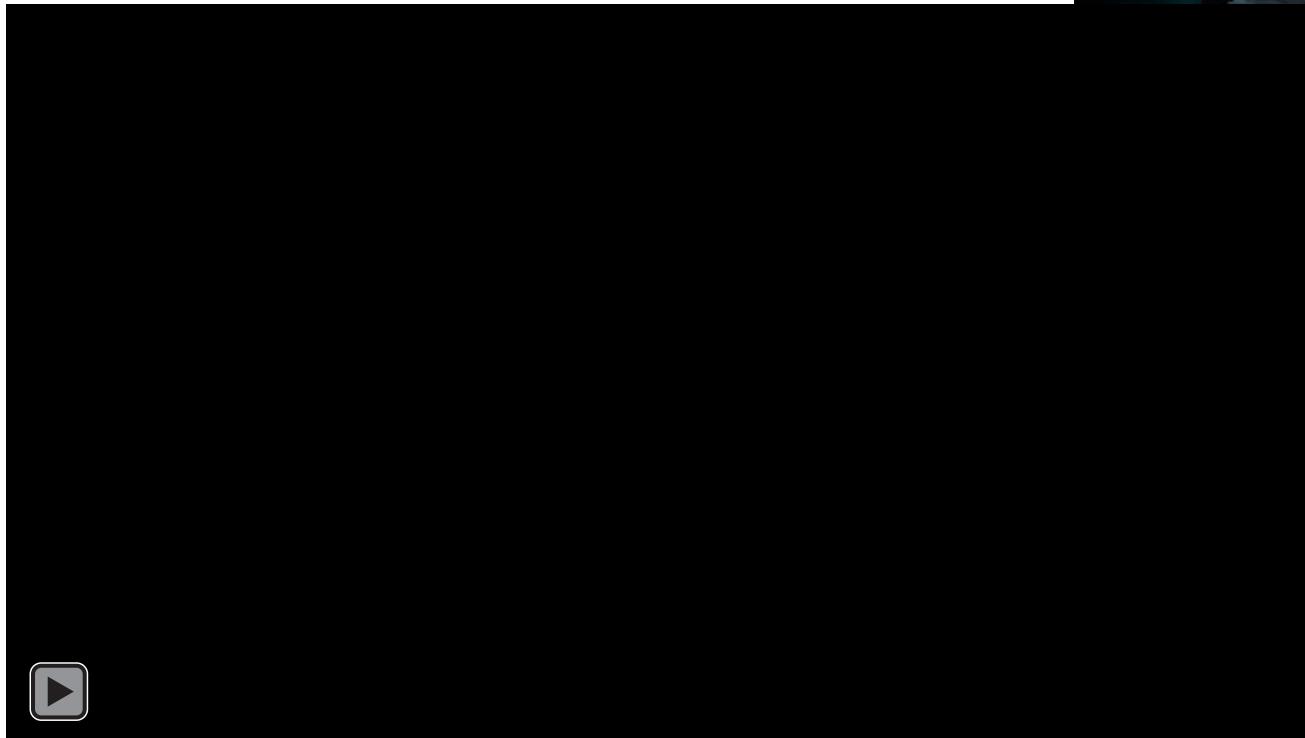


# Complex Systems Design M&S Trends

- System Dynamics M&S
- Agent-Based M&S
- **MMOGs**



# Massively Multiplayer Online Games (MMOGs)



- MMOGs are a recent addition to the modeling and simulation (M&S) tool suite
- MMOGs are simultaneously tools that allow hundreds or thousands players to interact with behavioral models, frameworks for building such models, and laboratories in which these models can be tested

# Design this System *Ensuring that it Works and is Robust*



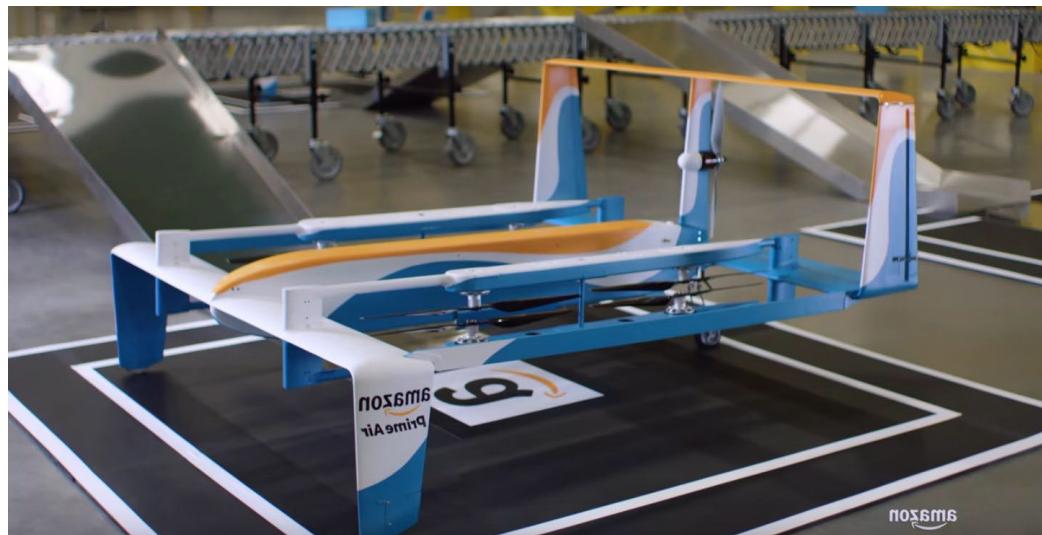
## U.S. Infectious Disease Containment and Control System





## Design this System *Ensuring that it Works and is Robust*

- A delivery System utilizing unmanned aerial vehicles (UAVs) to transport packages, food and other goods
  - Orders placed on-line
  - 30-minute delivery to residence
  - 10-mile delivery radius
  - Packages up to five pounds
  - 400 feet ceiling
  - Up to 300 UAVs per 10 miles<sup>2</sup>
  - 30 minute battery life
  - Autonomous “Sense and Avoid”



# Modeling and Simulation

## Key Questions:

1. *How could you utilize Modeling and Simulation to Ensure that your System Works and is Robust? Why? When?*
  - *Assessment Criteria: M&S plan is rational and well integrated into the overall system V&V plan.*  
*The objectives/goals of M&S are clearly defined.*
2. *How could you generate the Simulation inputs?*
  - *Assessment Criteria: Simulation inputs and experiments are clearly defined and correlated with the M&S objectives.*
3. *How could you be sure that you can be confident in the Simulation results?*
  - *Assessment Criteria: Simulation results are clearly correlated with the M&S objectives.*

# Homework/Participation

- Midterm Exam Next week