System Synthesis

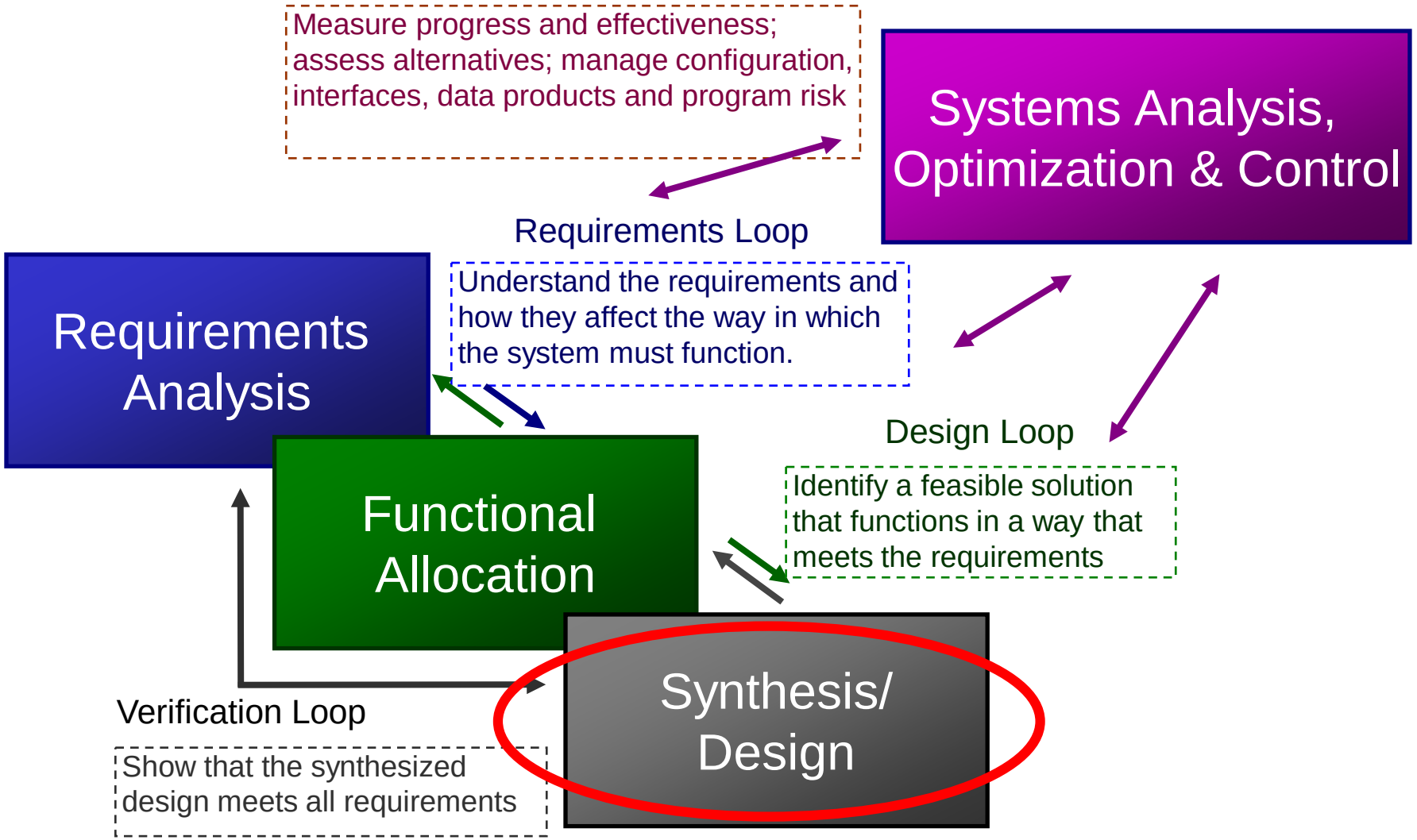
Module Purpose: System Synthesis

- Define *system synthesis* and describe it in context of the systems engineering process.
- Show how *system synthesis* is used to translate the functional architecture into an optimized physical architecture.
- Describe some heuristics and define some concepts useful for good *system design*, specifically:
  - *Modular designs* with low coupling, high cohesion and low connectivity.
  - *Robust designs* that meet requirements over a wide range of environmental or input parameters.

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.

Allocate System Performance and Resources to the Subsystems (step 2)

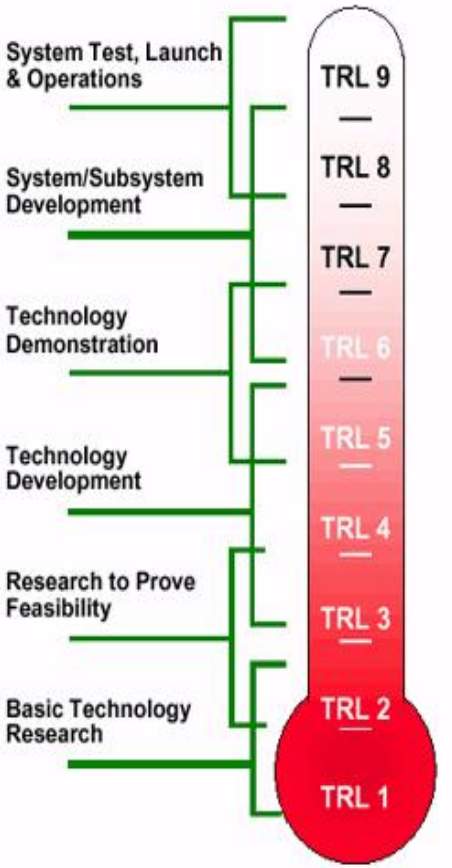
- The functional architecture still leaves resource and performance allocation to be determined.
- Performance and resource constraints are allocated (or derived) as subsystem alternatives are considered. e.g., system mass or power is allocated to the subsystems.
- Different solutions typically have different allocations.
- **First allocations will be sub-optimal, so start allocating knowing that you will re-allocate everything (probably many times).**
- The first time is usually no more than an ‘educated guess’, but is necessary to make progress.
- Make your assumptions explicit, document them and refine them as the design matures.

Define Subsystem Alternatives (step 3)

- For each functional requirement decide how it might be implemented.
- Start with an existing system/subsystem - at least for comparison - and an understanding of the current state-of-the-art (SOTA) and the merits of heritage solutions.
- Trade trees (discussed in trade study module) can be used to keep track of options for each physical implementation.
- Assess the performance, the limits of the approach, the robustness and risks of these options.
- Capture assumptions, interface implications, and the relative merits of alternatives. Why? So when things change you can go back and see the consequences of the new situation.

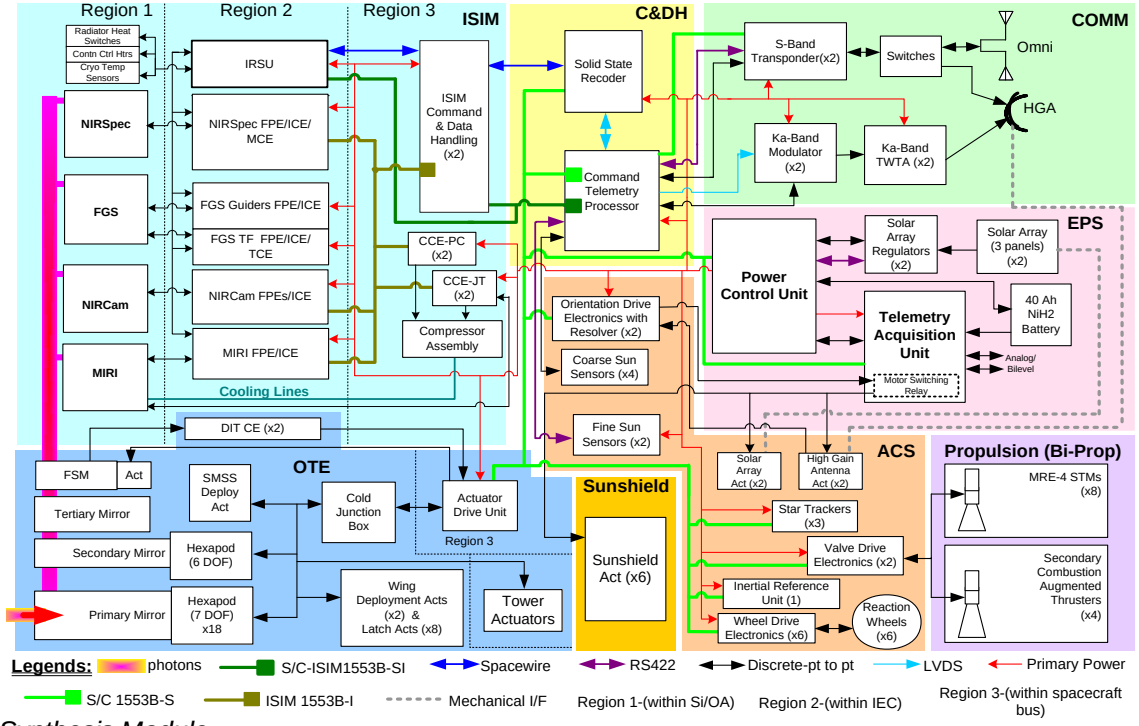
Assess Technology Alternatives and Their Maturity (step 4)

- Technology maturity is a measure of the use readiness of a considered technical approach for a **specified application and environment**.
- As such, technology maturity is also a measure of the risk associated with a particular approach **in the specified application and environment**.
- The Technology Readiness Level (TRL) is a commonly used measure of technical maturity (to be discussed in the technology module).



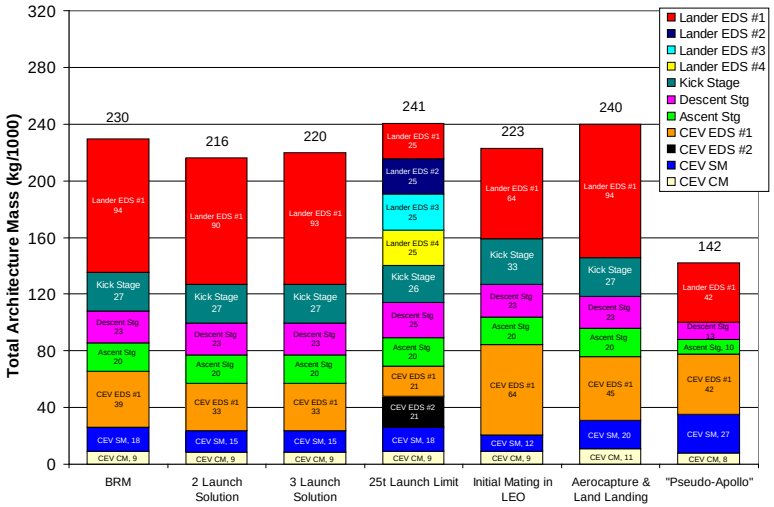
Define Physical Interfaces (step 5)

- A common and simple tool, the *N-squared diagram* helps define and record where interfaces exist (to be discussed in the interfaces module).
- Develop a *schematic block diagram* for the system and subsystems under development.
- *Schematic block diagrams* provide an easy reference for interfaces, redundancy and completeness checks.



Estimate Subsystem and System Performance of Each Alternative (step 6)

- Early performance estimates let you know if a given solution is viable. Compare with the past; extraordinary claims require extraordinary proof.
- Usually there are many options to consider, so many decisions have to be made. The first performance models are ‘quick and dirty’ (low fidelity), but with enough information to make some decisions and hence some system design progress.
- Because of the low fidelity of the performance estimates, record your assumptions, results, and the basis for any decisions. These estimates will be redone with new models, data or assumptions.

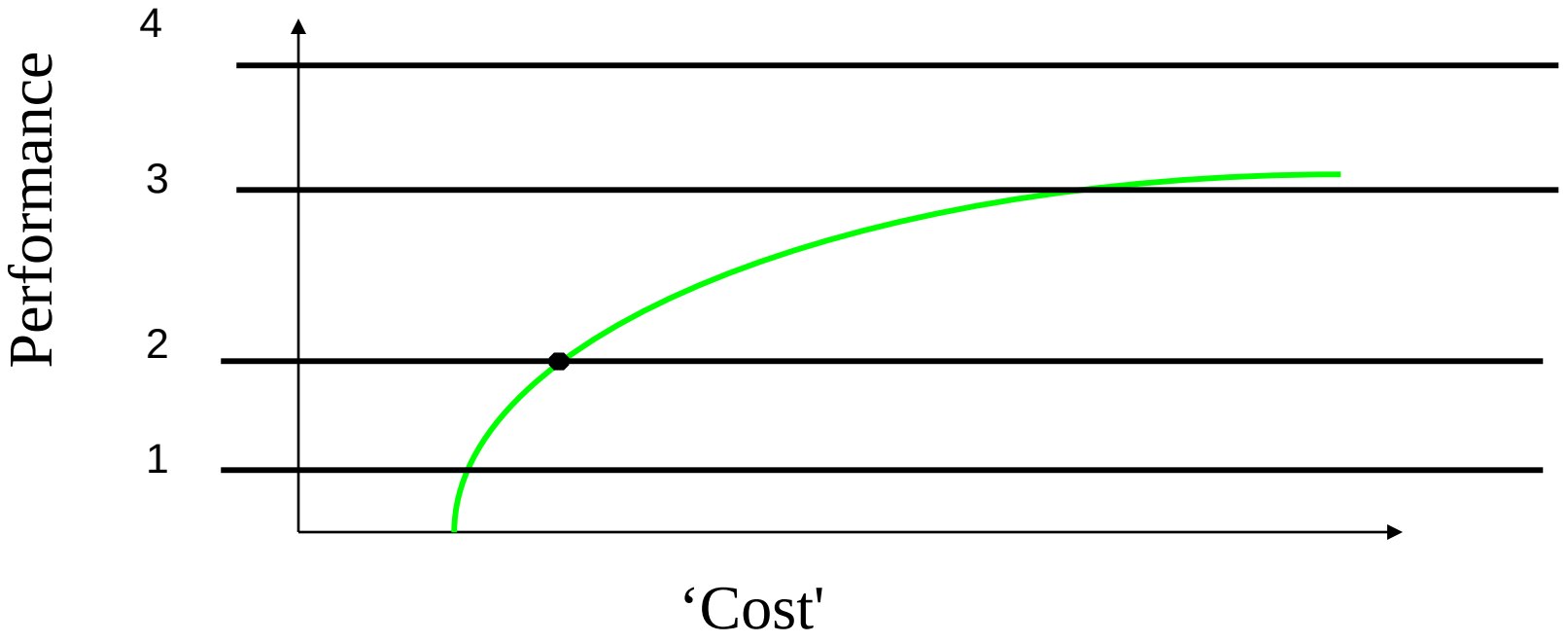


Use Performance-Resource Curves (step 7)

Introduction

Use performance-resource curves (utility curves) to identify break points.

A typical utility curve



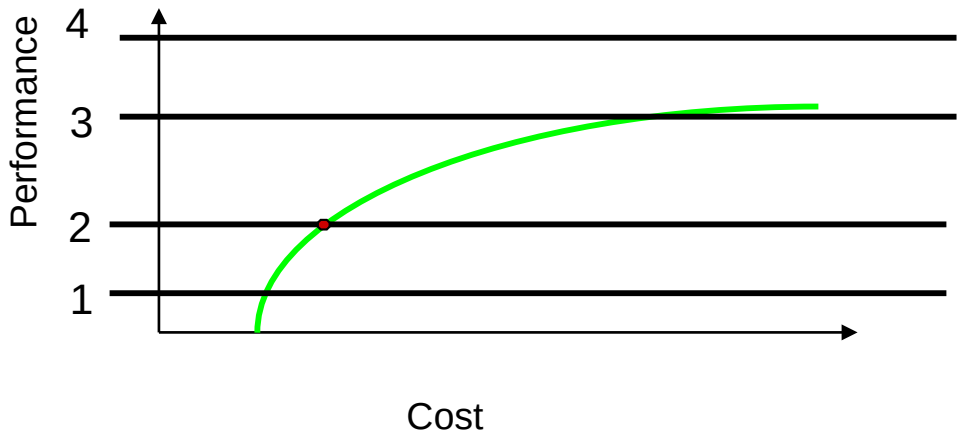
Use Performance-Resource Curves (step 7)

Explanation

'Cost' is not just money, but any resource (e.g., power efficiency, pointing jitter, or data downlink rate)

- ✓ *Performance requirement 1* - A little more resource and a lot more performance - consider adding performance to provide relief for other subsystems.
- ✓ *Performance requirement 2* - About maximum performance per cost point - a good place to be.
- ✓ *Performance requirement 4* - Impossible at any cost.
- ✓ *Performance requirement 3* - Possible, but costly and risky*.

* The last 10 percent of the performance sought generates one-third of the cost and two-thirds of the problems. Augustine's Law Number VII.



Augustine's Laws

Law Number V

One-tenth of the participants produce over one-third of the output. Increasing the number of participants merely reduces the average output.

Law Number X

Bulls do not win bullfights; people do. People do not win people fights; lawyers do.

Law Number XIII

There are many highly successful businesses in the United States. There are also many highly paid executives. The policy is not to intermingle the two.

Law Number XV

The last 10 percent of performance generates one-third of the cost and two-thirds of the problems.

Law Number XVI

In the year 2054, the entire defense budget will purchase just one aircraft. This aircraft will have to be shared by the Air Force and Navy 3-1/2 days each per week except for leap year, when it will be made available to the Marines for the extra day.

Augustine's Laws

Law Number XVII

Software is like entropy. It is difficult to grasp, weighs nothing, and obeys the Second Law of Thermodynamics; i.e., it always increases.

Law Number XXII

If stock market experts were so expert, they would be buying stock, not selling advice.

Law Number XXV

A revised schedule is to (an Aerospace) business what a new season is to an athlete or a new canvas to an artist.

Law Number XXVI

If a sufficient number of management layers are superimposed on each other, it can be assured that disaster is not left to chance.

Law Number XLVIII

The more time you spend talking about what you have been doing, the less time you have to spend doing what you have been talking about. Eventually, you spend more and more time talking about less and less until finally you spend all your time talking about nothing.

Determine the Driving Requirements (step 8)

- Determine the driving requirements (those that are the toughest to meet).
- **Make sure you understand the origins and motivations of the requirements and how they are distributed.**
- Distinguish between mission requirements and their derived or allocated subordinates.
  - *Mission requirements* are one-for-one tied to a system performance requirement or system constraint. Mission requirements cannot be varied for system optimization. An *example mission requirement*: **The Vehicle System shall have a mass of no more than 5,500 kg (restricted by transport capabilities).**
  - *Allocated or derived requirements* can be varied without *necessarily* changing the mission performance. Allocated or derived requirements can be reallocated for system optimization. An *example allocated requirement*: The communications subsystem shall have a mass of no more than 90 kg (this allocation is a fraction of the system mass requirement above).

Select A Preferred System Design (step 9)

- Use trade studies to compare alternatives and select a preferred system design (discussed in the trade studies module).
- Maintain a decision database because the priorities or constraints may change and the decisions revisited. Ask:
  - What decisions were made?
  - What was considered?
  - What criteria were used?
  - Formally review this database at the project milestone reviews.
- Use Measures of Effectiveness (MOEs) to compare alternatives. MOEs are metrics used to assess risk and performance, lifecycle implications, and customer priorities to support your decisions.
- Remember that system requirements define a successful solution. This way you will know when to ***stop trying*** to improve a design — **'better is the enemy of good enough'.**

Seek 'Balanced' System Solutions (step 9)

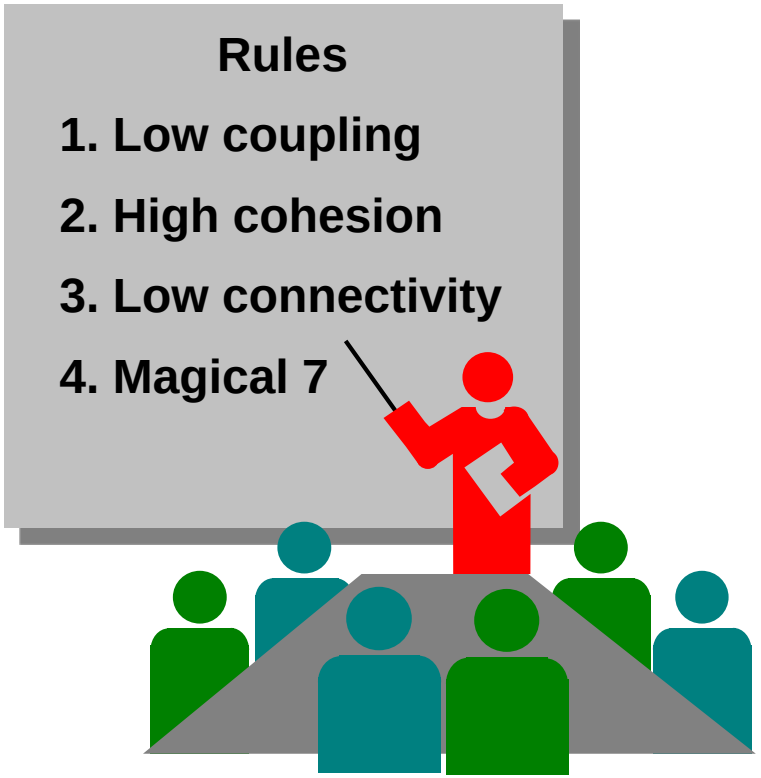
- **A 'balanced' system solution is one that has considered the system (i.e., total) risk, cost, performance, technical maturity and robustness of the set of subsystems. *No one subsystem dominates these system metrics.***

- One way to seek an optimal system design is to try to stress all subsystems equally. Each subsystem should be equally hard to develop, although they may be hard to develop for entirely different reasons. Try to share the development 'pain' among subsystems.

- To explore this notion, experiment with different resource or performance allocations. This is an area where team communications and acknowledging common goals are obviously necessary, since vague estimates of subsystem development difficulty will have to be shared and compared to yield new allocations.

Some Good System Synthesis Heuristics

- Develop and apply experience based development rules.
- Heuristics are codified and validated guidelines established by experience and past success.
- Examples:
  1. Choose subsystem boundaries so that each can be implemented independently.
  2. Group strongly related functions; separate those that are unrelated.
  3. Reduce the number and complexity of interfaces.
  4. Each level of a partitioned hierarchy should have 7±2 elements.



Inductive Methods

Some Definitions to Help Understand Modular Design (1/2)

- **Desirable attributes are modular subsystems or components which include low coupling, high cohesion and low connectivity.**

- **Coupling -** a measure of the relative dependence or information shared between subsystems.
  - Low coupling among subsystems results in a system that is less prone to 'ripple effects' when errors or changes occur within a subsystem.

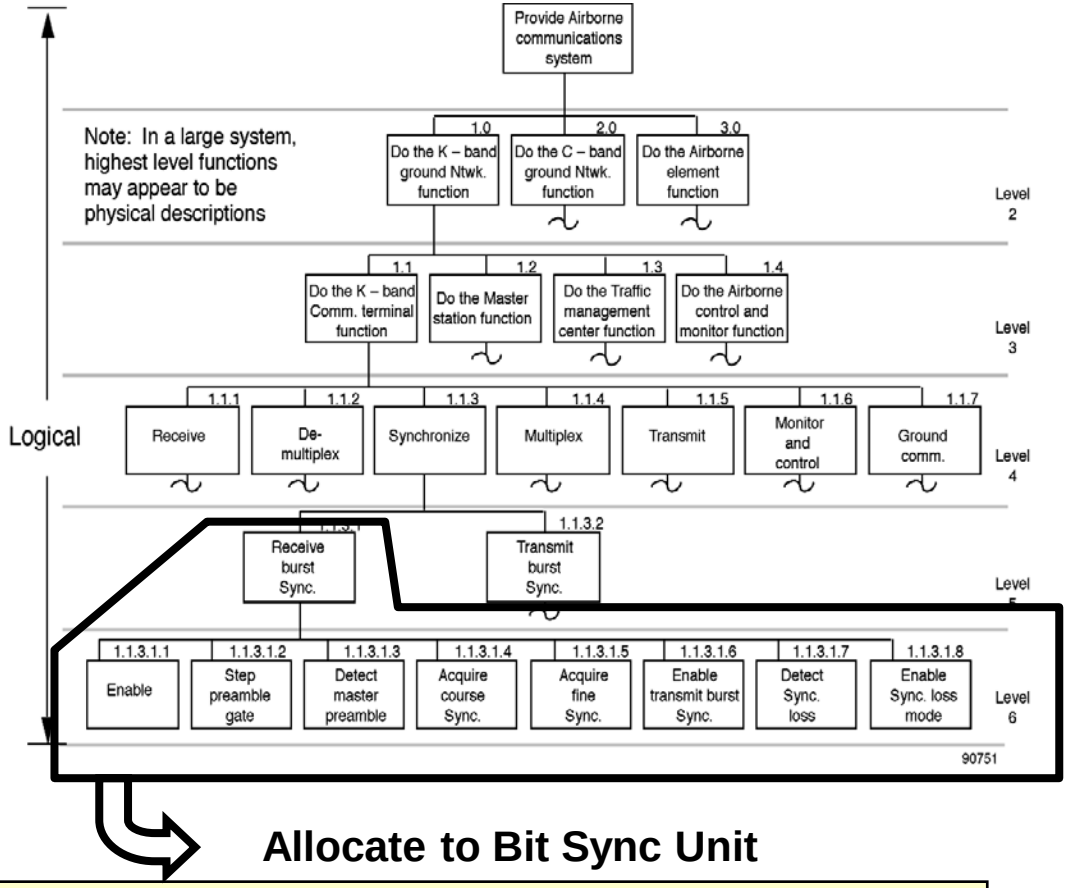
- **Cohesion -** a measure of the similarity of tasks performed within a subsystem.
  - High cohesion allows for use of identical or similar parts, or for use of a single component to perform multiple functions.

Some Definitions to Help Understand Modular Design (2/2)

- **Connectivity** refers to the relationship of internal elements within one module to internal elements within another module. High connectivity is undesirable in that it creates complex interfaces that may impede design, development, and testing.
- In short, develop subsystems with single-minded function and an aversion to frequent interaction with other subsystems. Keep interfaces simple and similar functions together.

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 the others.**
- **Choose a configuration with minimal communications between subsystems.**



Allocate to Bit Sync Unit

"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

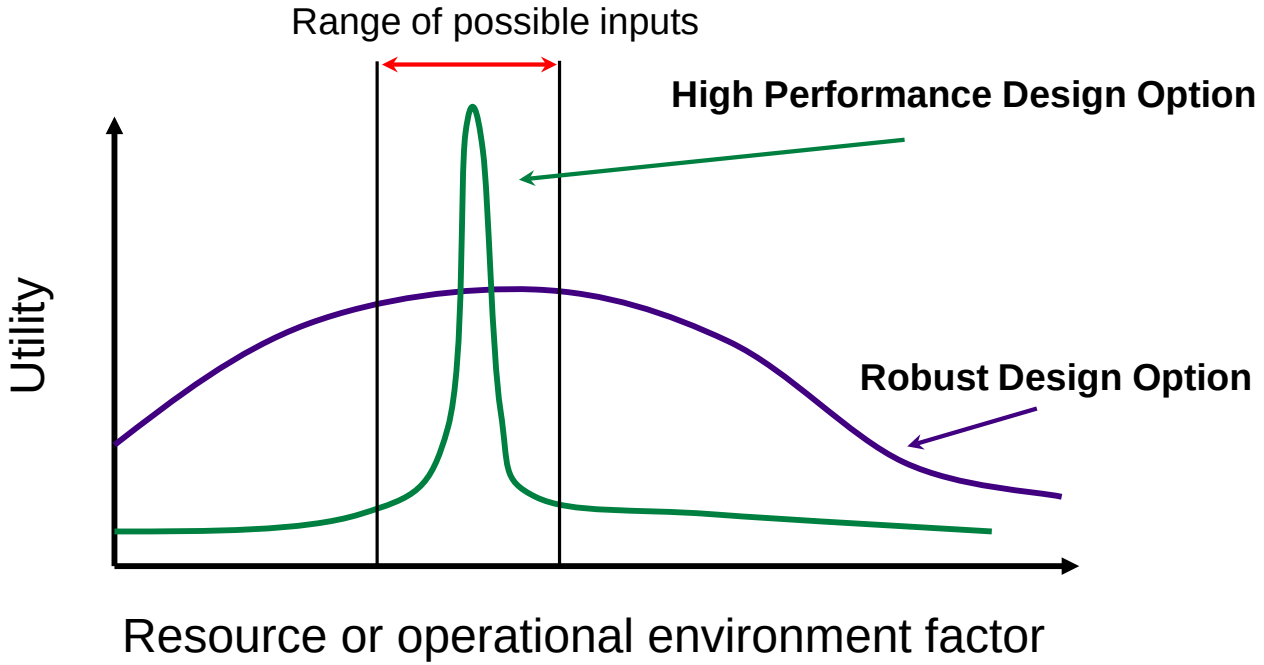
The Cassini Spacecraft

Robust Design

- **A *robust design* reduces the effects of uncertainty or variability without reducing the uncertainty or variability itself.**
- Subsystems that can perform over a wide range of environments or operating parameters are called ‘robust’.
- Sensitivity analysis determines the performance of a system or subsystem with variable input.
- For example, consider the output voltage from a power supply with a nominal output voltage of 18v DC with a nominal input of 110v AC when the input voltage is varied from 120v to 90v AC. For a robust design the output voltage may vary from 17.95v to 18.05v for a non-robust design the voltage may vary from 14.7v to 19.6v.

Robust Design

- **Robustness** is a measure of the ability of a system to absorb changes in requirements, constraints or failures while reducing the impacts on the performance, functionality, or composition of the mission or system. Two different design options are shown - one with high performance, one with robust performance.



For the range of possible inputs shown, the robust design is better than the high performance design.

Module Summary: System Synthesis (1/2)

- System synthesis produces a physical architecture from the functional architecture.
- System synthesis is an iterative process in that many subsystem solutions are considered, and the subsequent system merits are assessed before a baseline is established.
- *To assess the merits of candidate subsystems, consider their performance, risks, heritage, technical maturity, and limits of their performance.*
- One way to seek optimal system design is to choose performance and resource allocations to equally stress all subsystems. Share the pain among subsystems.
- **System requirements define a successful solution. Know when to stop trying to improve a design - better is the enemy of good enough.**

Module Summary: System Synthesis (2/2)

- Maintain a decision database and formally review it at each project milestone review.
- Use performance-resource (utility) curves to identify break points to help develop balanced solutions.
- Determine the ‘driving’ requirements.
  - Understand which requirements are the toughest to meet and consider reallocation to manage them.
- Subsystems with high cohesion, low complexity and low connectivity are good. In other words, keep interfaces simple and similar functions together.
- In system design decisions, consider robustness — a measure of a system’s sensitivity to environmental or resource variation.

True genius resides in the capacity for evaluation of uncertain, hazardous, and conflicting information.
~ Winston Churchill ~