Systems Engineering*:*

Imperatives, Definitions, Technology & Talent

What is Output/Impact??

Clas A. Jacobson
Chief Scientist
United Technologies Systems & Controls Engineering (SCE)
United Technologies Corporation
Clas.Jacobson@utsce.utc.com
860.610.7652 860.830.4151 (mobile)

A Joint LCCC and ACCESS Workshop
May 4-6, 2015
Lund University, Lund, Sweden
State of the Art, Recent Advances and Future Directions in Model-Based Engineering and Model-Based Systems Engineering

THANKS

Alberto Sangiovanni Vincentelli, Alberto Ferrari, Richard Murray, Eelco Scholte, John Cassidy, Scott Kaslusky, Kevin Otto, Satish Narayanan, Karl Astrom, Manfred Morari, Scott Bortoff, Mark Myers, Greg Provan, Johan Akesson…and others…

KEY POINTS

Product development processes – how products are developed – are under pressure to deliver more with less. More functionality, shorter schedules, more software, more criticality – these are all drivers that push current approaches beyond what the processes and people can deliver. (Cost vs cost/benefit…)

Systems engineering is a science. Systems engineers are not (only) “experienced engineers” – there are methods & tools that can and should be applied in a discipline and taught – not just processes. A large amount of analysis.

Methods and tools define systems engineering (a) requirements analysis, (a) architecture analysis, (c) model based development and (d) design flows.

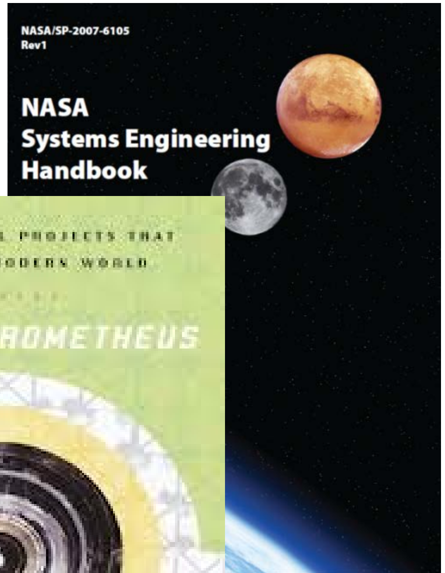
Implications: all about leadership, output & impact…

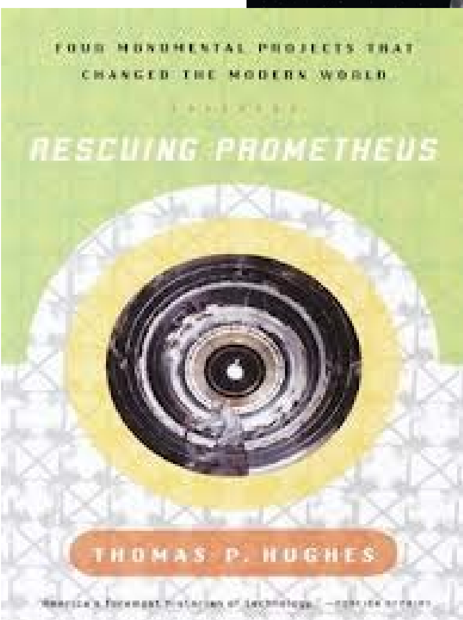
For industry – recognition and adoption of systems engineering is a competitive positioning – needs to be done correctly and efficiently…

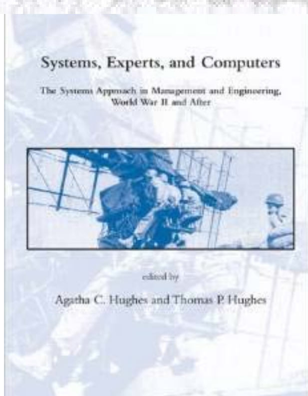
For academia – curricula in systems engineering do not exist and real experience in systems engineering largely lacking in academia. Customers and (national) needs are not being met.

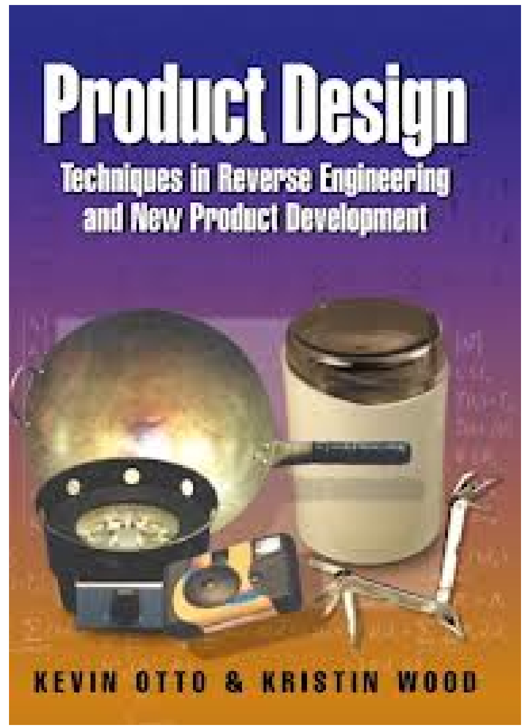
For research entities – funding programs need definition, scope and industrial partnering. NSF, DARPA, EU programs all need to be encouraged.

FAVORITE REFERENCES









INVITED PAPER

Quo Vadis, SLD? Reasoning About the Trends and Challenges of System Level Design

Recognizing common requirements for co-design of hardware and software in diverse systems may lead to productivity gains, lower costs and first-pass design success.

By Alberto Sangiovanni-Vincentelli, *Fellow IEEE*

ABSTRACT | System-level design (SLD) is considered by many as the next frontier in electronic design automation (EDA). SLD means many things to different people since there is no wide agreement on a definition of the term. Academia, designers, and EDA experts have taken different avenues to attack the problem, for the most part springing from the basis of traditional EDA and trying to raise the level of abstraction at which integrated circuit designs are captured, analyzed, and synthesized from. However, my opinion is that this is just the tip of the iceberg of a much bigger problem that is common to all *system* industry. In particular, I believe that notwithstanding the obvious differences in the vertical industrial segments (for example, consumer, automotive, computing, and communication), there is a common underlying basis that can be explored. This basis may yield a novel EDA industry and even a novel engineering field that could bring substantial productivity gains not only to the semiconductor industry but to all system industries including industrial and automotive, communication and computing, avionics and building automation, space and agriculture, and health and security, in short, a real technical renaissance.

In this paper, I present the challenges faced by industry in system level design. Then, I propose a design methodology, platform-based design (PBD), that has the potential of addressing these challenges in a unified way. Further, I place methodology and tools available today in the PBD framework and present a tool environment, Metropolis, that supports PBD and that can be used to integrate available tools and methods together with two examples of its application to separate industrial domains.

KEYWORDS | Embedded software; embedded systems; models and tools; platform-based design (PBD); system-level design (SLD); system-level design environments

I. INTRODUCTION

Electronic design automation (EDA) has played a pivotal role in the past 25 years in making it possible to develop a new generation of electronic systems and circuits. However, innovation in design tools has slowed down significantly as we approach a limit in the complexity of systems we can design today satisfying increasing constraints on time-to-market and correctness. The EDA community has not succeeded as of today in establishing a new layer of abstraction universally agreed upon that could provide productivity gains similar to the ones of the traditional design flow (Register Transfer Level (RTL) to GDSII) when it was first introduced. Nor has it been able to expand significantly into new adjacent markets to increase its total available market. Among the adjacencies of interest, I believe the *electronics system market* has great potential since system companies that are now facing significant

Manuscript received May 13, 2006; revised December 7, 2006. This work was supported in part by the Gigascale System Research Center, the Center for Hybrid and Embedded Software Systems (CHESS) at the University of California, Berkeley, which receives support from the National Science Foundation (NSF award CCR-0225610), the State of California Micro Program, and the following companies: Agilent, Bosch, DGIST, General Motors, Hewlett Packard, Infineon, Microsoft, National Instruments, and Toyota. This work was also supported by the European Project SPEEDS and the Networks of Excellence Artist2 and Hycon.
The author is with the Department of Electrical Engineering and Computer Science, University of California, Berkeley, CA 94720 USA (e-mail: alberto@eecs.berkeley.edu).
Digital Object Identifier: 10.1109/JPROC.2006.890107

0018-9219/$25.00 ©2007 IEEE

Vol. 95, No. 3, March 2007 | Proceedings of the IEEE 467

HISTORY: SHORT, BIASED…

WW II (“physics”) (Rhodes, Morse…)

Cold War (Hughes)

Space (NASA, Rechtin)

Automotive (ASV)

Cyber (NAE)

Crisis (OSD, cybersecurity...today)

AGENDA

Why?

What?

How?

Implications

AGENDA

Why?
- Systems, systems views
- Product development processes, risk/variability, problems
- Need for a change…

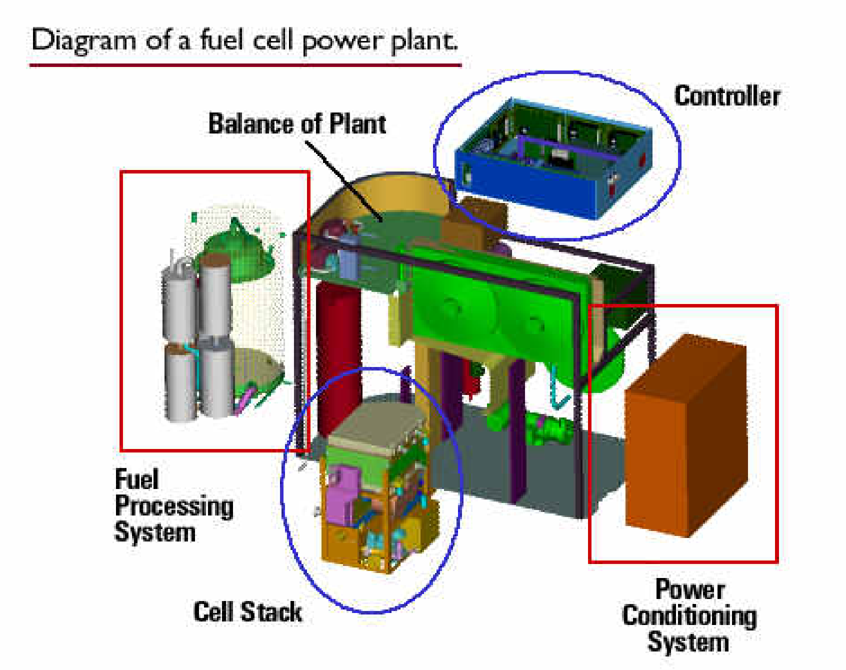
What?

How?

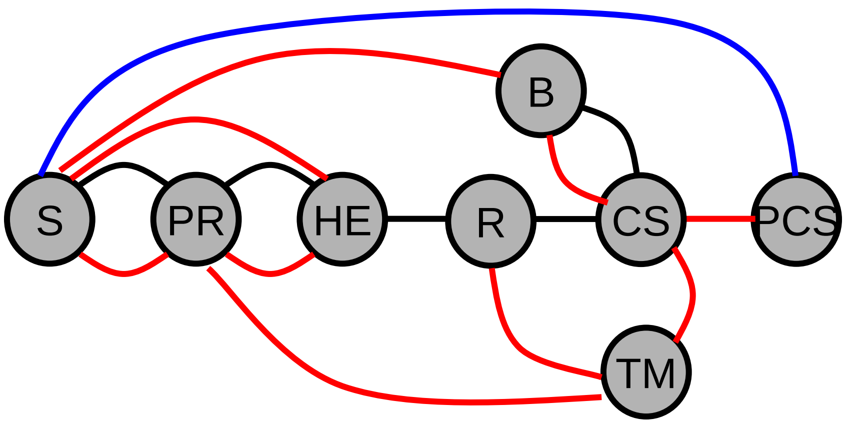
Implications

Fuel Cell Power Plant

Robust power production : Tightly Coupled Large Scale Dynamics



Graphical representation of Mass, Energy and Information transfer in a fuel cell power plant



- ***Spatially distributed***
- ***Interconnected system***
- ***Multi-scale dynamics and control***
    - Slow thermal dynamics and very fast reaction mechanisms leading to very stiff systems.
    - Very fast electrochemical reactions leading to algebraic constraints.

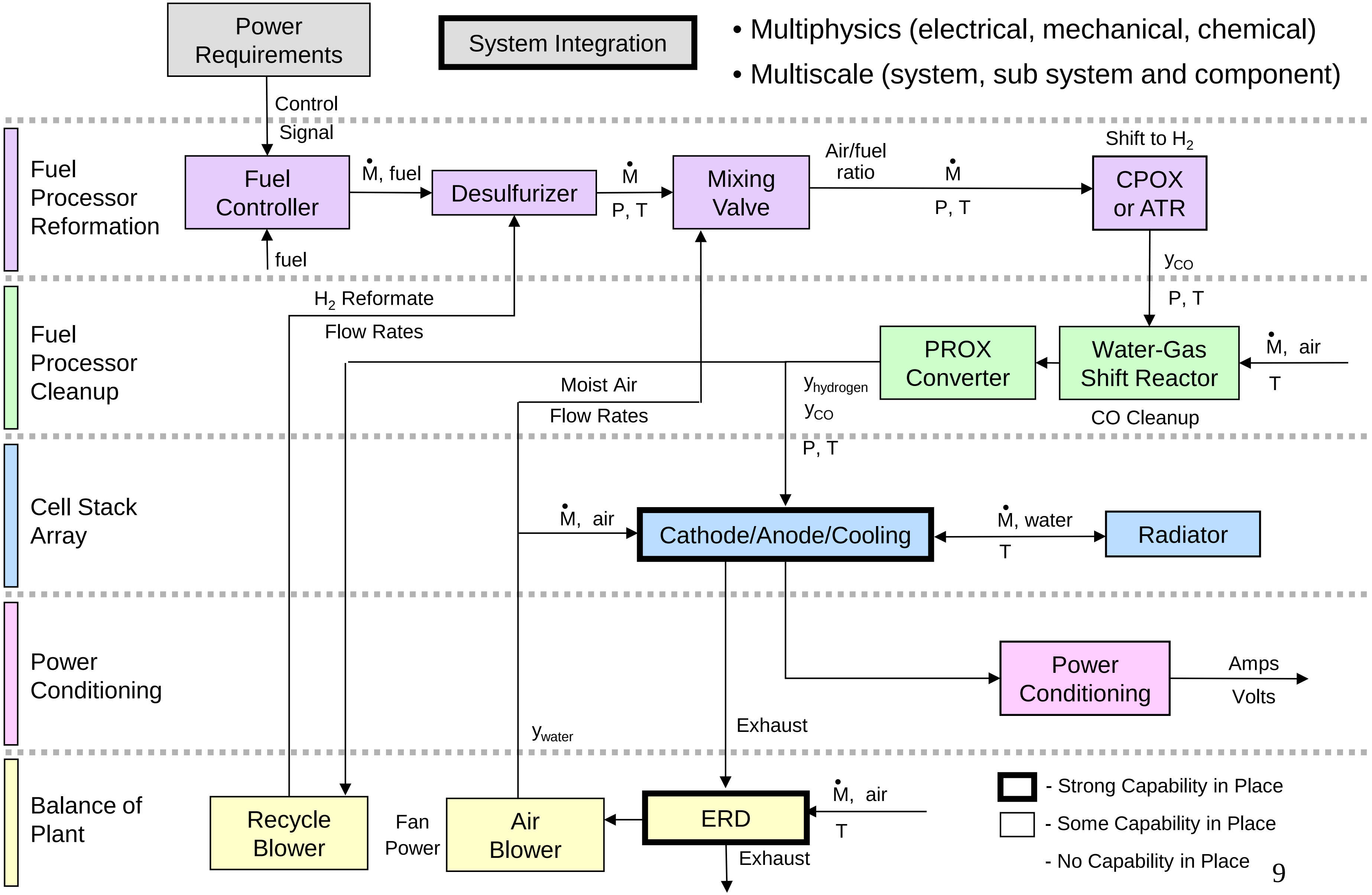
Fuel Cell Power Plant Model

- > 500 dynamic state variables
- Monolithic DAE system.
- Wide separation of time scales.
- Highly nonlinear.
- Large operating range
- Simulation FPS, CS alone takes min
- Subsystem simulations are robust.
- Full system takes ~30 min to hrs
- Full system simulations are not robust

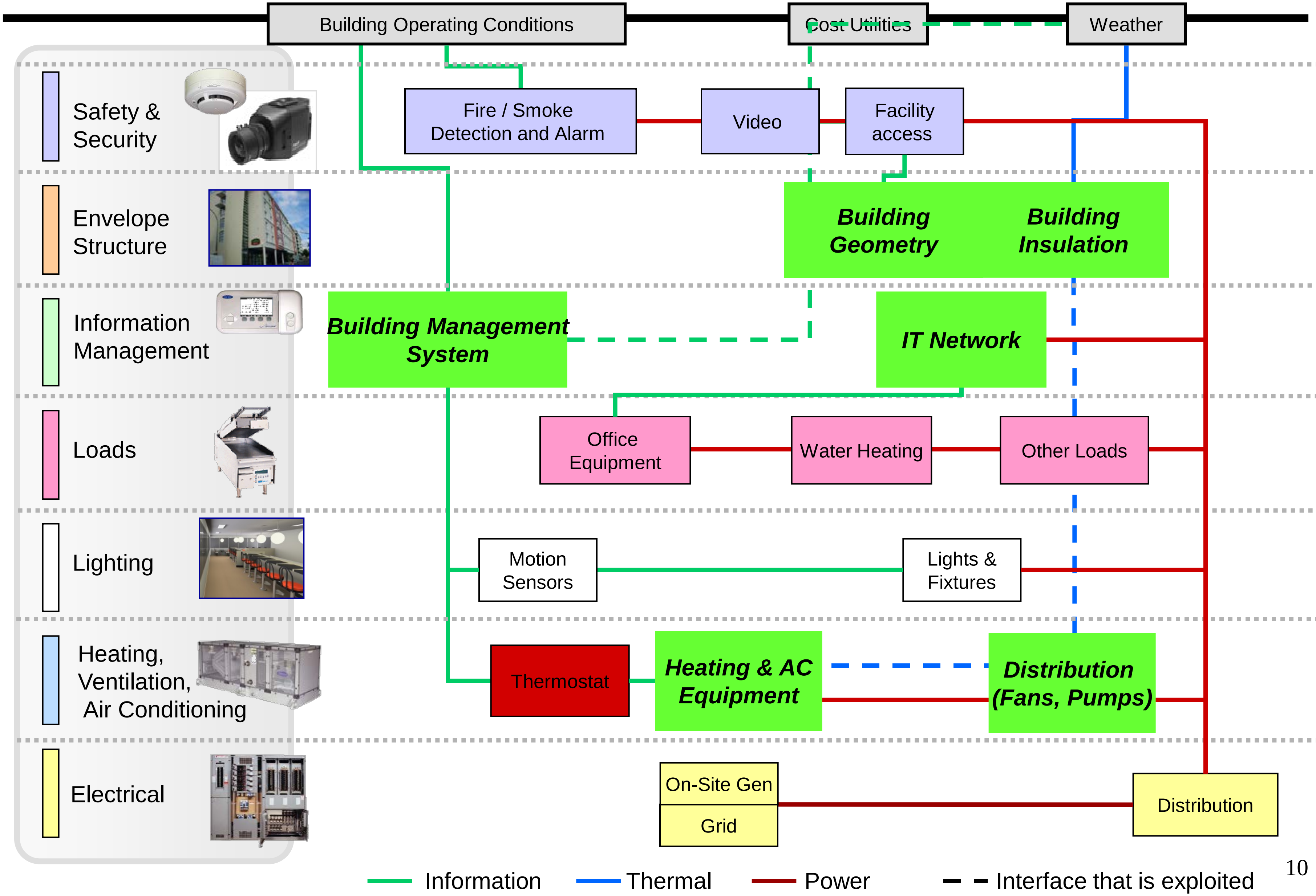
PEMFC Power Plant Dynamic Model (~2000)

Fuel cell functional decomposition

- Multidisciplinary scope (systems design, controls, component design)
- Multiphysics (electrical, mechanical, chemical)
- Multiscale (system, sub system and component)

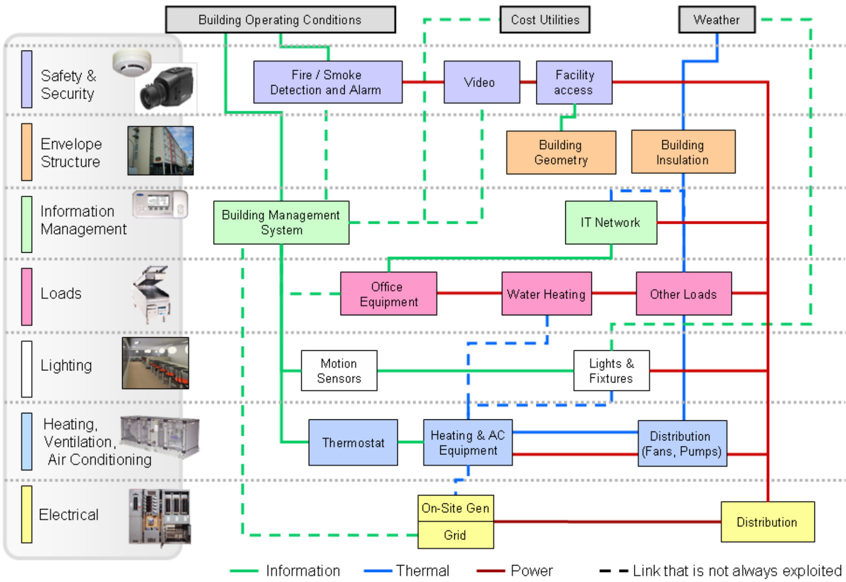


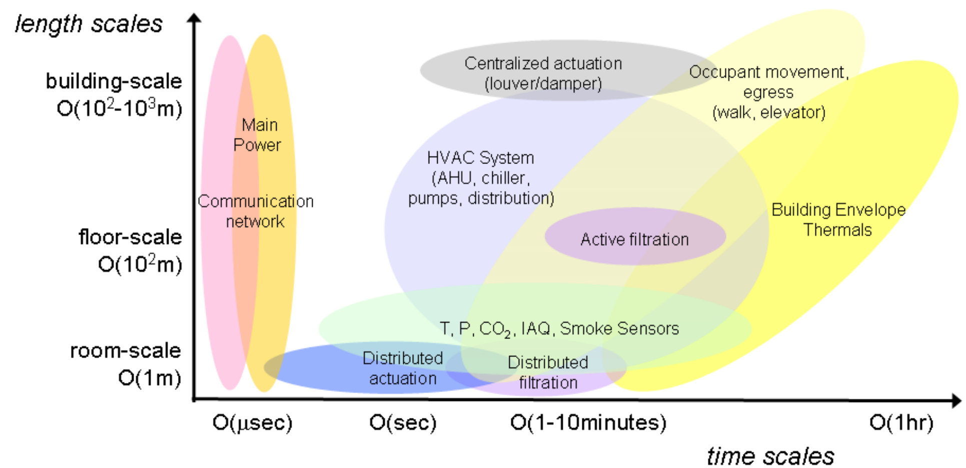
BUILDING SUBSYSTEM DECOMPOSITION



Building Systems Integration Challenge

Complex interconnections among building components*





- Components do not necessarily have mathematically similar structures and may involve different scales in time or space;
- The number of components may be large/enormous
- Components can be connected in a variety of ways, most often nonlinearly and/or via a network. Local and system wide phenomena may depend on each other in complicated ways
- Overall system behavior can be difficult to predict from the behavior of individual components. Overall system behavior may evolve along qualitatively different pathways that may display great sensitivity to small perturbations at any stage

* D.L. Brown, J. Bell, D. Estep, W. Gropp, B. Hendrickson, S. Keller-McNulty, D. Keyes, J. T. Oden and L. Petzold, Applied Mathematics at the U.S. Department of Energy: Past, Present and a View to the Future, DOE Report, LLNL-TR-401536, May 2008.

SYSTEMS

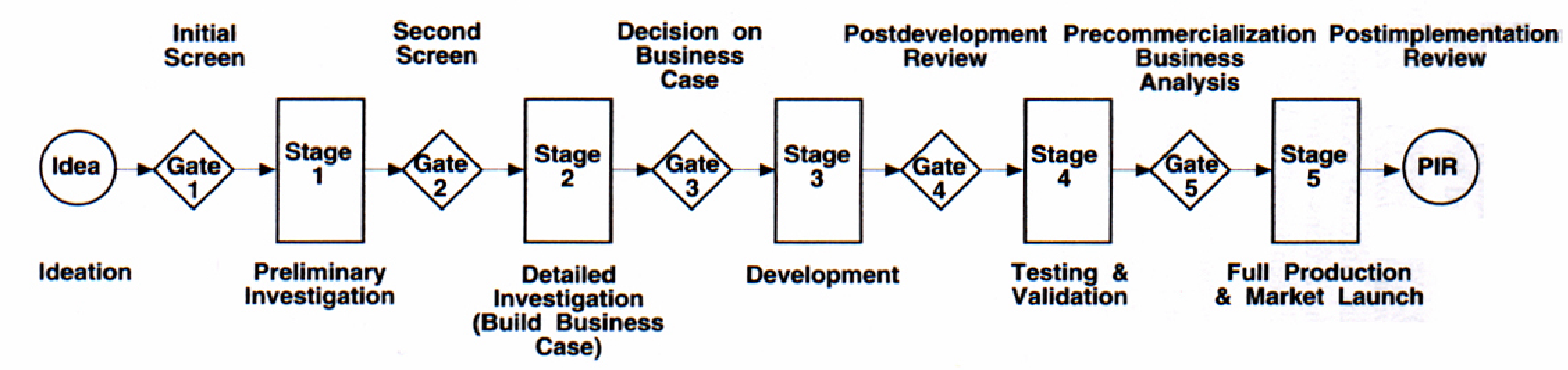
NASA Systems Engineering Handbook: "(1) The combination of elements that function together to produce the capability to meet a need. The elements include all hardware, software, equipment, facilities, personnel, processes, and procedures needed for this purpose. (2) The end product (which performs operational functions) and enabling products (which provide life-cycle support services to the operational end products) that make up a system."

INCOSE Systems Engineering Handbook: "homogeneous entity that exhibits predefined behavior in the real world and is composed of heterogeneous parts that do not individually exhibit that behavior and an integrated configuration of components and/or subsystems."

In the systems approach, concentration is on the analysis and design of the whole, as distinct from total focus on the components or the parts. The approach insists upon looking at a problem in its entirety, taking into account all the facets, all the intertwined parameters. It seeks to understand how they interact with one another and how they can be brought into proper relationship forthe optimum solution of the problem. The systems approach relates the technology to the need, the social to the technological aspects. It starts by asking exactly what the problem is and what criteria should dominate the solution and lead to evaluating of alternative avenues. As the end result, the approach looks for a detailed description of a specified combination of people and apparatus — with such concomitant assignment of function, designated use of matériel, and pattern of information flow that the whole system represents a compatible, optimum, interconnected ensemble yielding the operating performance desired. (Ramo)

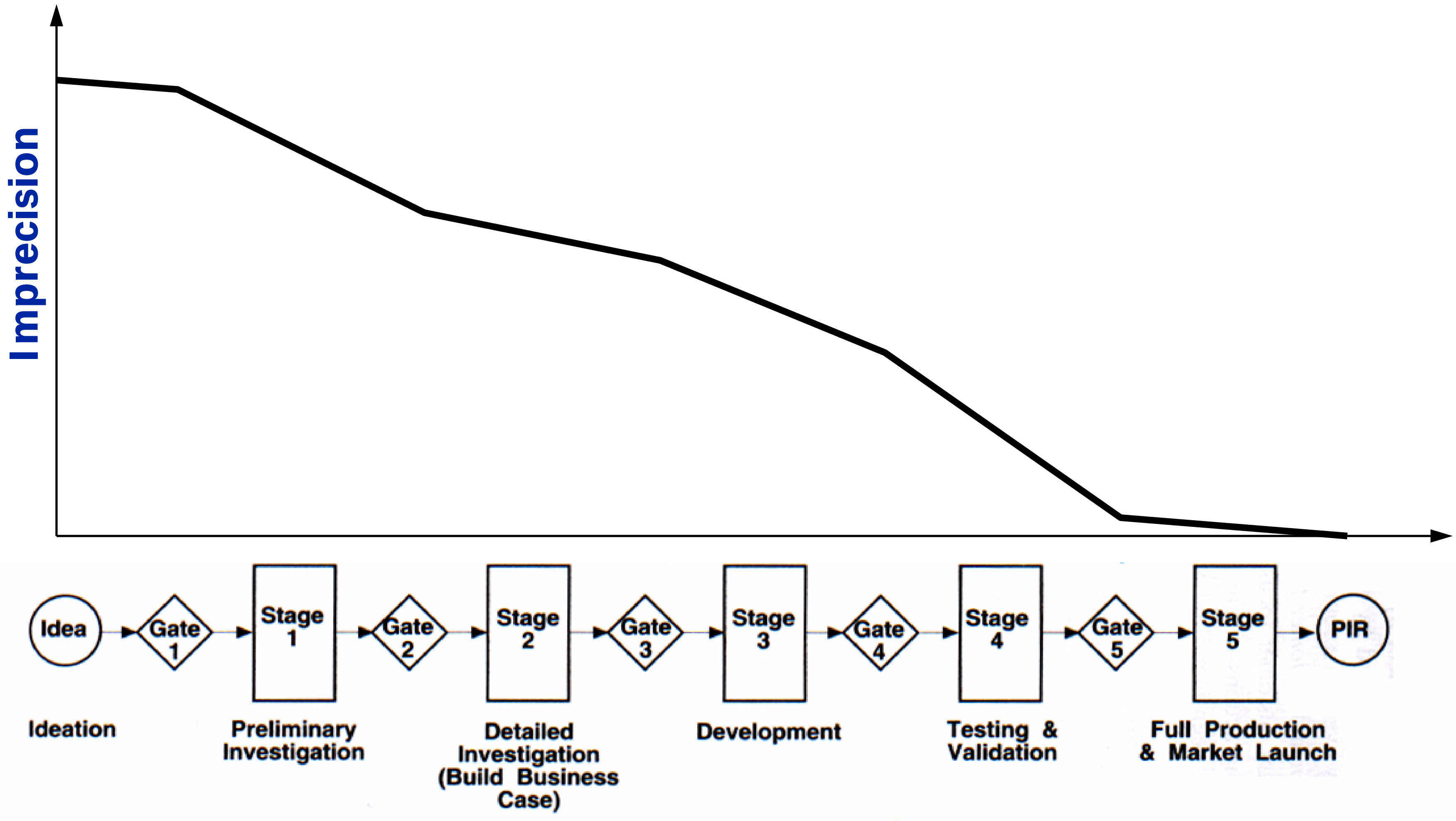
TYPICAL PHASE GATE DEVELOPMENT

Figure 5.4: A Generic Stage-Gate New Product Process (Cooper)



Use gates to bring everything to the same level of uncertainty. There you look for risks.

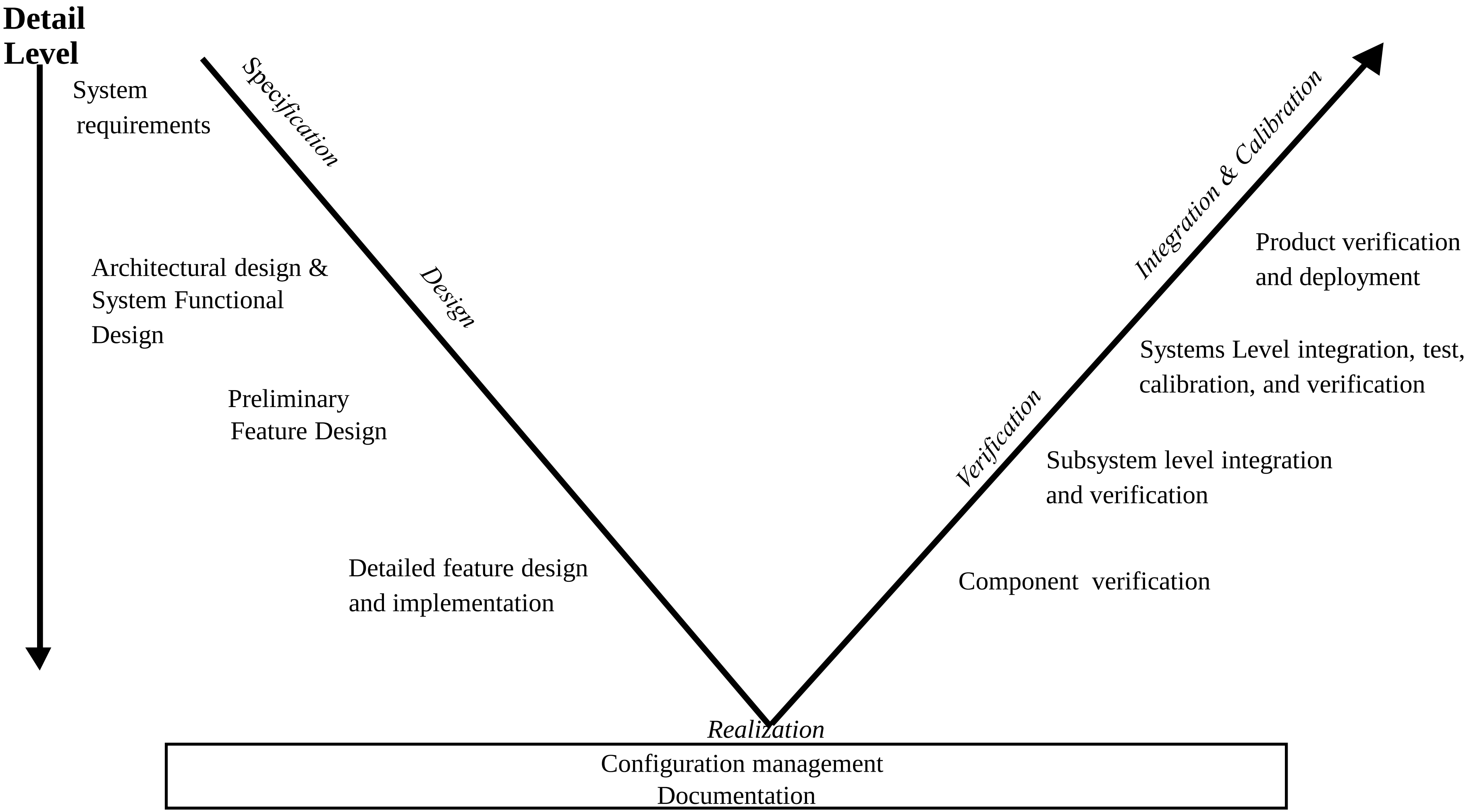
DESIGN SELECTION UNCERTAINTY



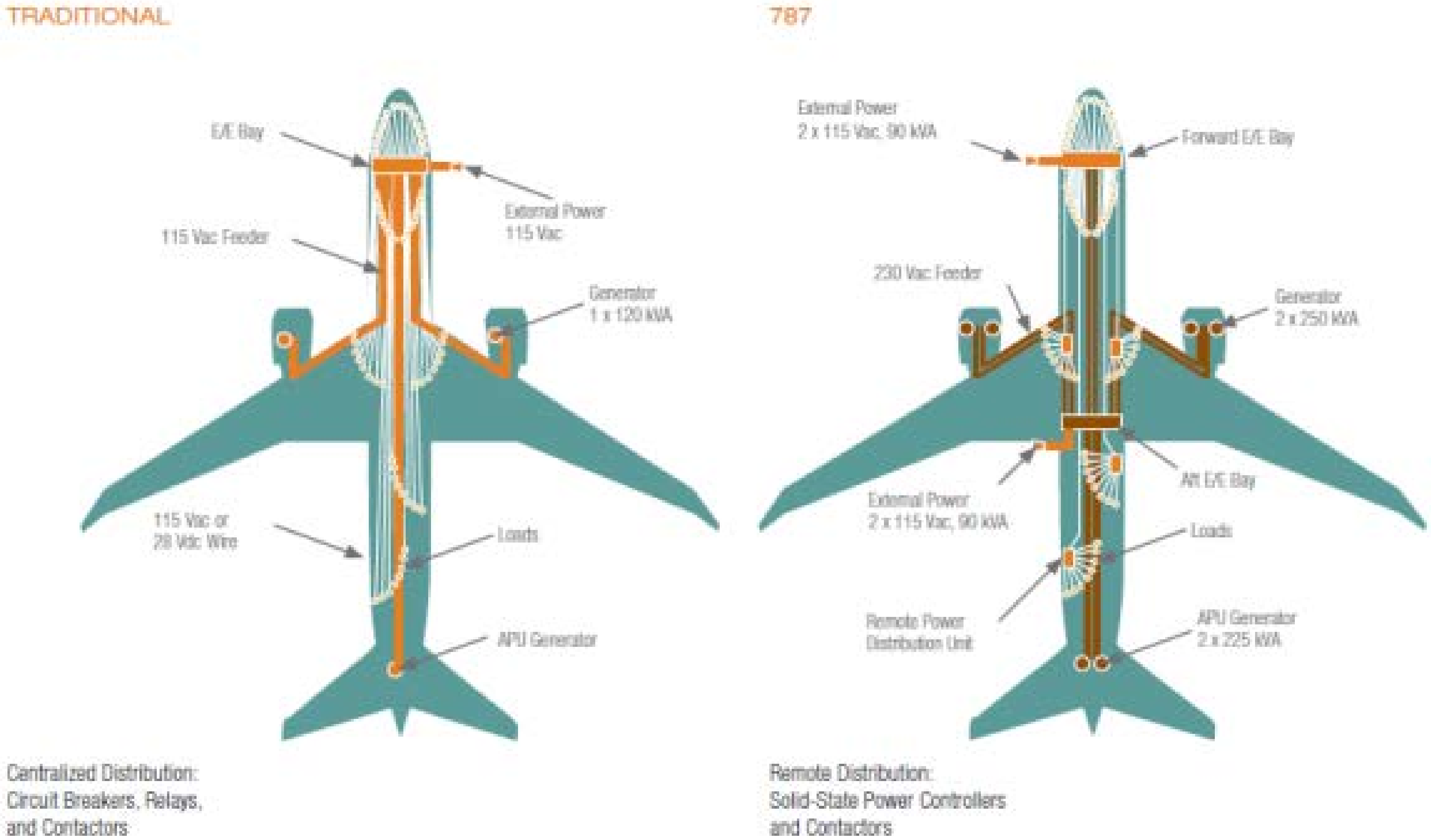
Systems Engineering ≡ Risk Management (Holding)

MODEL BASED SYSTEMS ENGINEERING

The “Design V” (NASA, MIL STD 499, ARP 4754a)



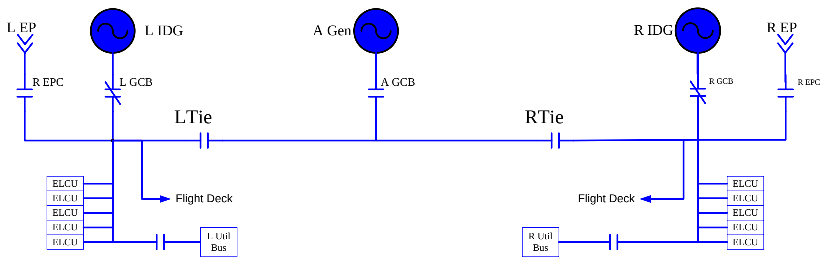
MORE ELECTRIC AIRCRAFT



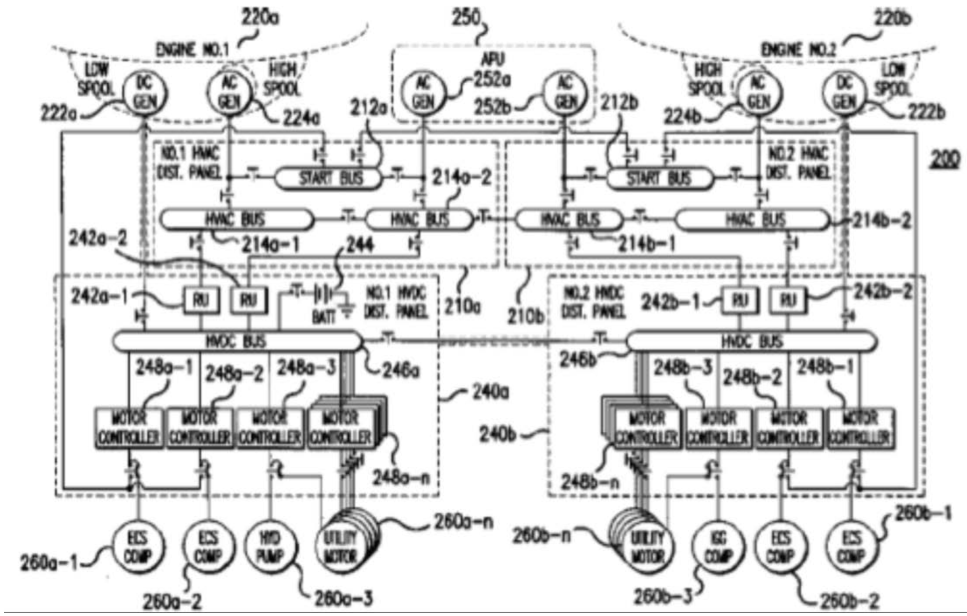
Source: *787 No-Bleed Systems:* Saving Fuel and Enhancing Operational Efficiencies by Mike Sinnett, Director, 787 Systems, Boeing, 2007

ARCHITECTURE & COMPLEXITY

767 Architecture



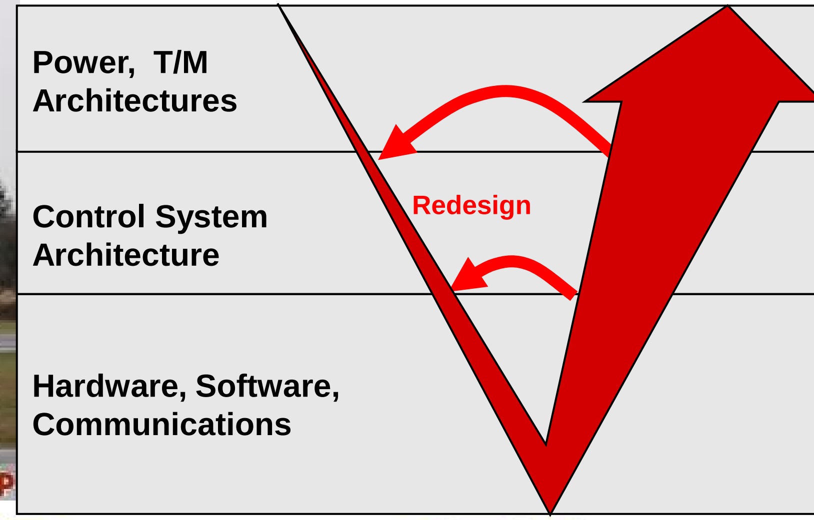
"More Electric" Architecture

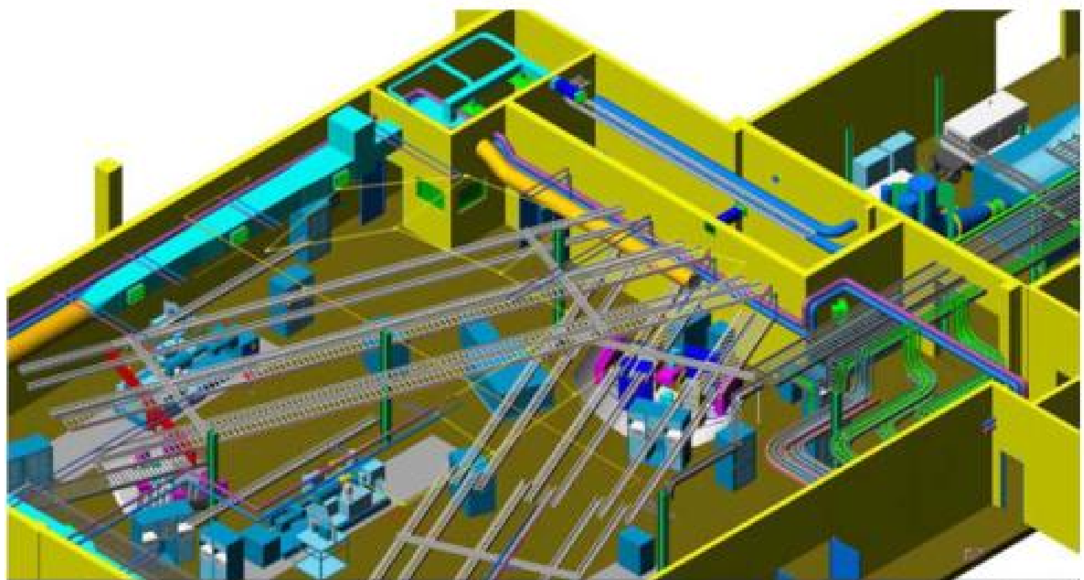


Growth in complexity driven by reliability

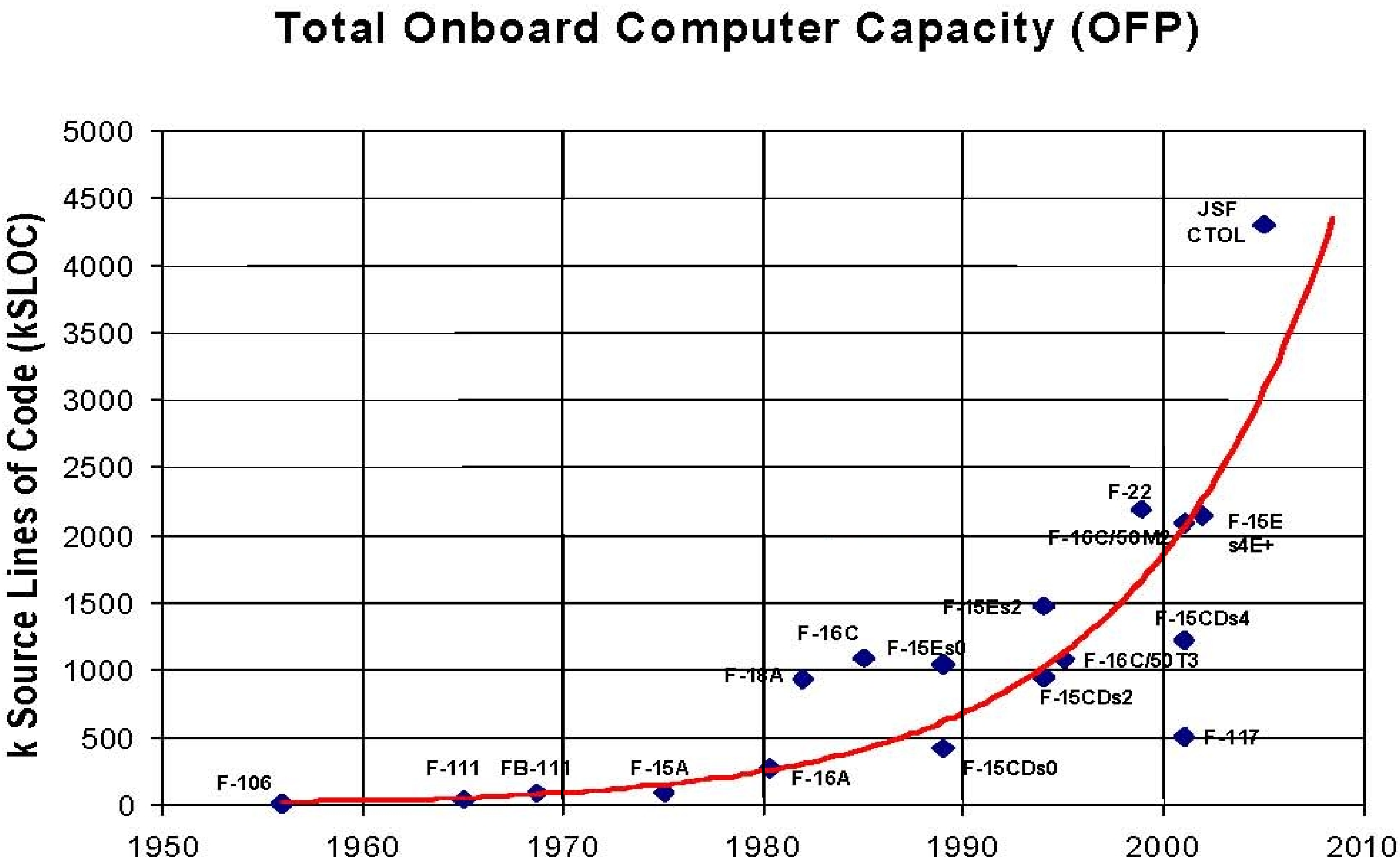
TESTING, DESIGN FLOW & REQUIREMENTS

Design Flow Burden on Integration & Testing





DoD Software is Growing in Size and Complexity



Source: "Avionics Acquisition, Production, and Sustainment: Lessons Learned -- The Hard Way", NDIA Systems Engineering Conference, Mr. D. Gary Van Oss, October 2002.

COPQ…

Assurance and Early and Continuous Validation

One of the great challenges for both defense and civilian systems is software quality assurance. Software assurance encompasses reliability, security, robustness, safety, and other quality-related attributes. **Diverse studies suggest that overall software assurance costs account for 30-50 percent of total project costs for most software projects.** Despite this cost, current approaches to software assurance, ***primarily testing and inspection***, are inadequate to provide the levels of assurance required for many categories of critical systems. As systems grow in size, complexity, interconnection, and use of third party components, these challenges will grow substantially. A further source of challenge is the dynamic nature of modern software architectures, including SOAs, architectures for autonomy and robotic systems, and other emerging architectural concepts.

Source?

DOD ISSUES IN INTEGRATED SYSTEMS

One area where the committee believes that new research would benefit DoD **is the management of engineering risk** in unprecedented large and ultra-scale systems. Such systems have engineering risks associated with early design commitments related to system functionality, non-functional attributes, and architecture. The research would focus on ways to mitigate these engineering risks at early stages of the process through new approaches to early validation, modeling, and architectural analysis.

D Software Research Needs and Priorities: A Letter Report
72.html

Preliminary Observations on DoD Software Research Needs and Priorities

A Letter Report

Committee on Advancing Software-Intensive Systems Producibility

Computer Science and Telecommunications Board
Division on Engineering and Physical Sciences

NATIONAL RESEARCH COUNCIL
OF THE NATIONAL ACADEMIES

THE NATIONAL ACADEMIES PRESS
Washington, D.C.
www.nap.edu

The second area where DoD has leading demand and could benefit from technological advancement is software quality assurance for defense systems. Software assurance encompasses reliability, security, robustness, safety, and other quality-related attributes. Defense systems often include commercial off-the-shelf components and may involve global development—global sourcing is a reality for major commercial software products and, additionally, for commercial custom software and service provisioning. **The needed research would focus on new ways for producers and consumers to create (and validate) a body of technical evidence to support specific claims in support of an overall judgment of fitness.**

The third area, which is just as important as the first two, is the reduction of requirements-related risk in unprecedented systems without too great a sacrifice in systems capability. The challenge in this area has two parts. **First, how can consequences of early commitments related to functional or nonfunctional requirements be understood** at the earliest possible time during development? And, second, **how can we make "requirements" more flexible over a greater portion of the system life cycle**? The committee believes that the most useful research for DoD would look at ways to achieve early validation—for example, through modeling, protoptying, and simulation—and also look at how iterative development cycles can be supported more effectively and, from the standpoint of risk in program management, more safely.

The Developer Approach: Standards

SAE Aerospace
An SAE International Group

AEROSPACE RECOMMENDED PRACTICE	SAE ARP4754	REV. A
	Issued 1996-11 Revised 2010-12 Superseding ARP4754	

(R) Guidelines for Development of Civil Aircraft and Systems

RATIONALE

This document provides updated and expanded guidelines for the processes used to develop civil aircraft and systems.

TABLE OF CONTENTS

SAE 4754, MIL STD 499, DO-178B

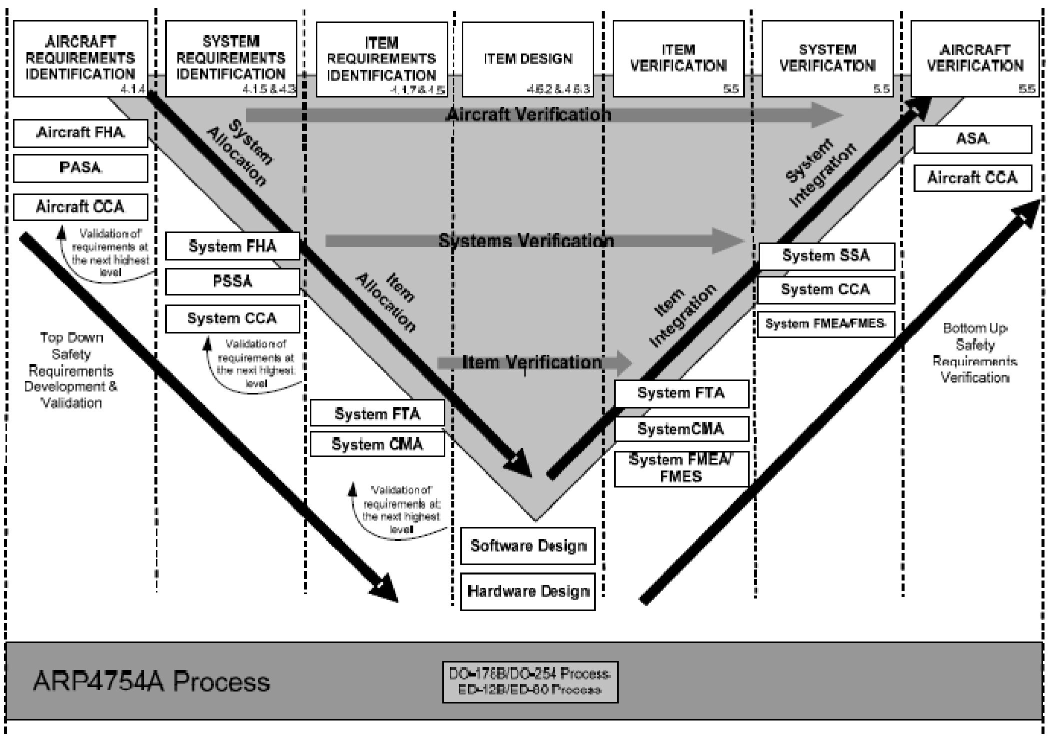


FIGURE 5 - INTERACTION BETWEEN SAFETY AND DEVELOPMENT PROCESSES

DRAFT

NOT MEASUREMENT SENSITIVE

MIL-STD-499C
Revised 24 March 2005
Superseding
MIL-STD-499B
6 May 1974

MILITARY STANDARD

SYSTEMS ENGINEERING

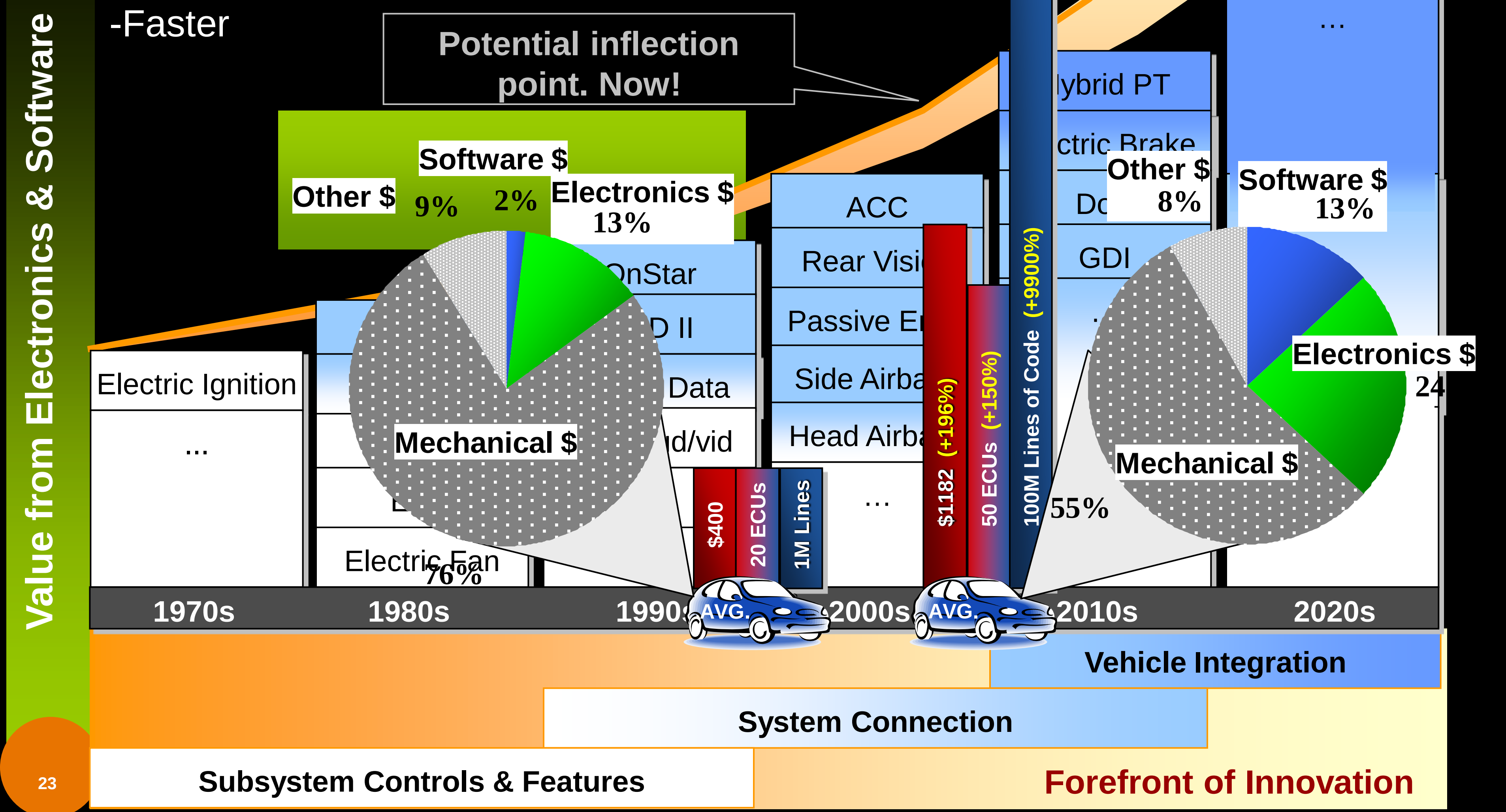
AMSC AREA MISC

DISTRIBUTION STATEMENT A. Approved for public release; distribution is unlimited.

MIL-STD-499B 2.70

Electronics, Controls & Software
Shifting the Basis of Competition in Vehicles

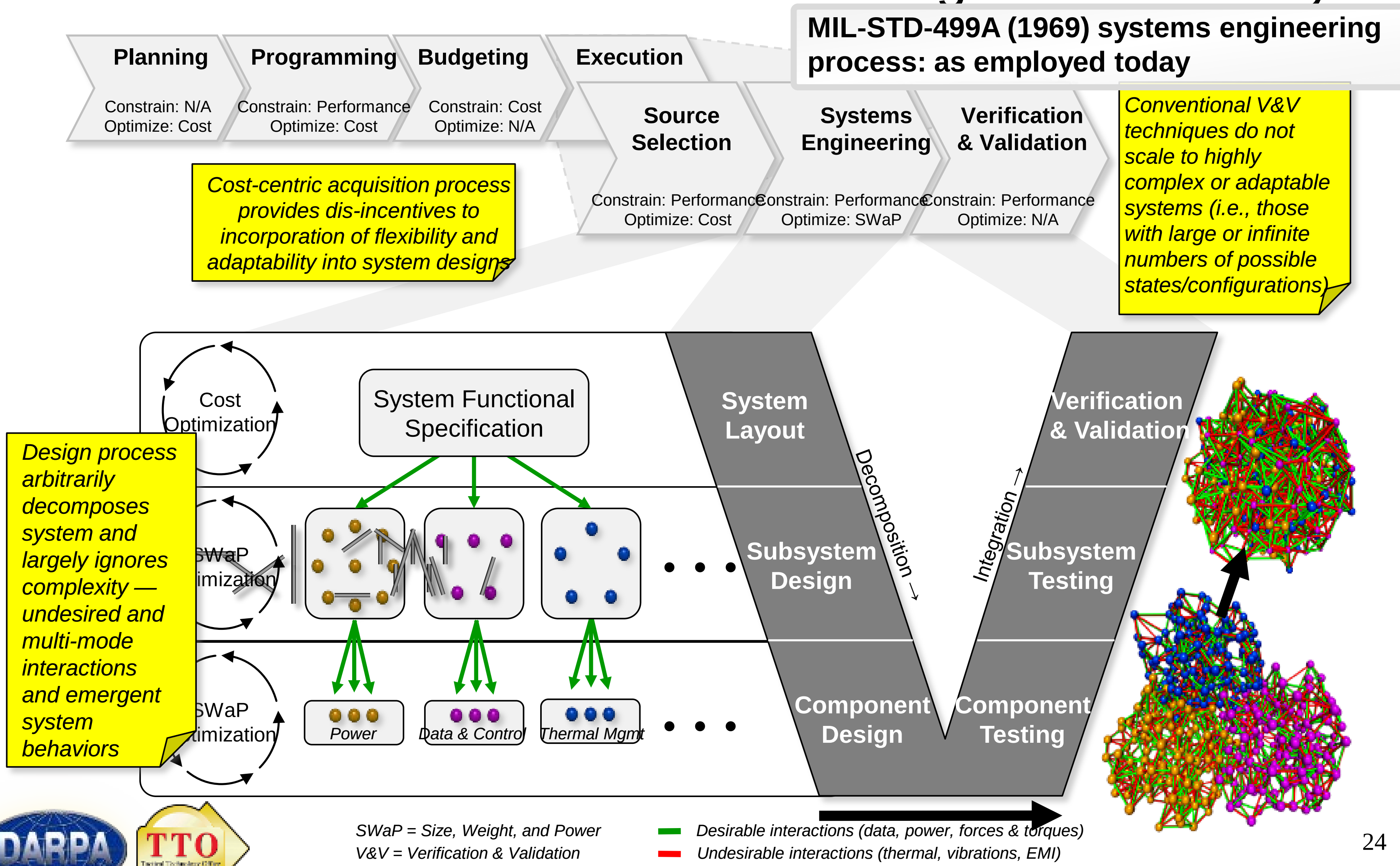
-More functions & features
-Less hardware
-Faster



Source: Matt Tsien, GM

STATUS QUO IN SYSTEM DESIGN (V MODEL)

There are several areas where change is necessary



UTC PRODUCTS

More integration…more software…
more complex operating modes

Agenda

Why?

What?

- Discipline…not just experience
- Definition – process + analysis

How?

Implications

SYSTEMS ENGINEERING

Is a discipline…

Not experience based (or not only)

Core skills

Arrange the design flow (stages…processes)

Produce the design artifacts (model based analyses carried out at each stage)

Project management and teaming and team selection (processes, standard, skills)

Scope

Systems engineering is a methodology for product system level design, optimization and verification that:

- Provides guarantees of performance and reliability against customer **requirements *(analysis)***
- Produces modular, extensible **architectures** for products ***(process + analysis)***
- Exploits **model-based analytical tools** and techniques ***(analysis…verification)***
- Coordinated execution of a prescriptive, repeatable and measurable **design process**

SYSTEMS ENGINEERING (UTC)

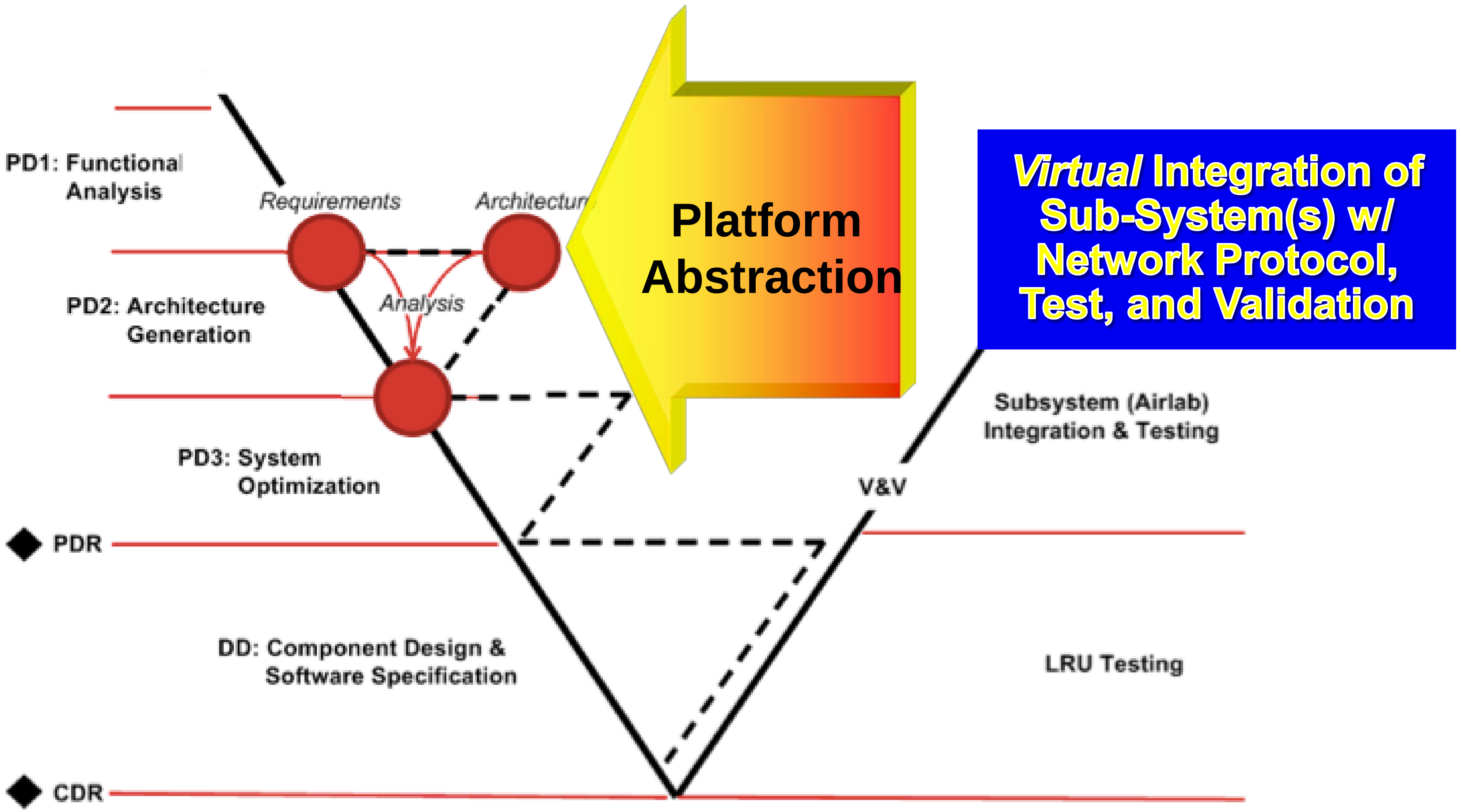
Systems engineering is the integrated product view and overall management of what will be delivered including components, communications and controls along with the coordinated product design including requirements elicitation and analysis, product development methodologies and allocation of requirements to subsystems and the validation, verification and certification.

Model based development is a core competence of methodologies and toolsets to accomplish the systems engineering task that involves translating requirements into product instantiation through a succession of combined behavioral, physics and computation/communication models which govern design decisions involving product architecture and quantify robustness and drive system testing and requirements verification. Model based development methodologies must be captured in engineering standard work to manage the work flow across the levels of abstraction of the design and models must form an integral part of the development process.

Controls is a key enabler in systems engineering that focuses on providing functionality that is often difficult to provide with a fixed design, moreover, controls can be used to reduce effects of uncertainty on product functionality. Control consists of the algorithmic connections between the physical components and the conversion of the performance requirements into product functionality.

PLATFORM-BASED DESIGN

Executable specs, early validation, virtual platforms



AGENDA

Why?

What?

How?

- Verification – rigorous requirements, formal methods
- Variability – robust design (uncertainty quantification)
- Architecture - identification (and evaluation) (models)
- Dynamics (not done here) (models)
- Optimization (not done here) (models)
- Contract based design (not done here) (models)

Implications

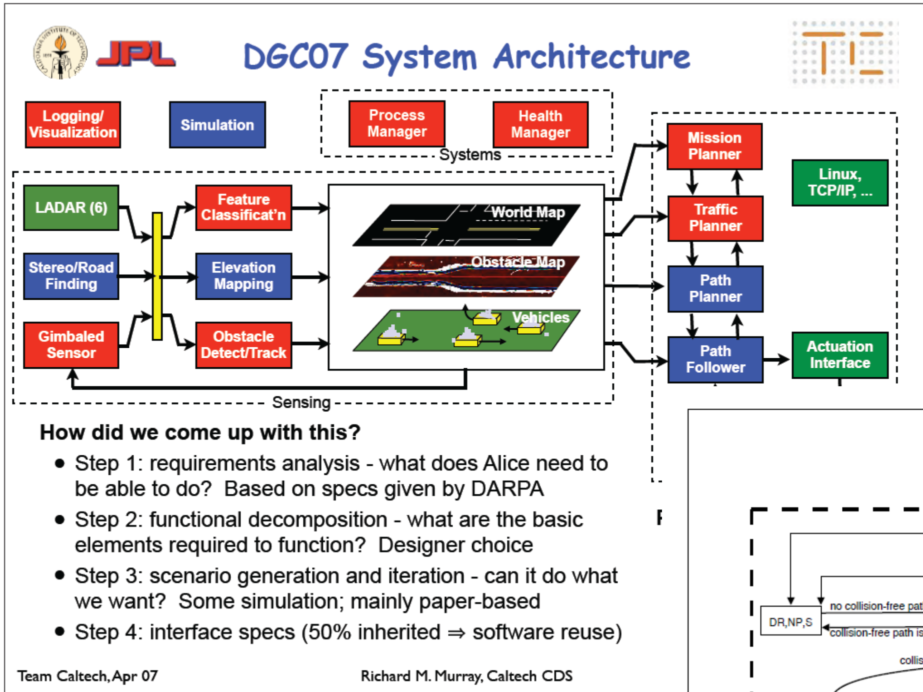
VERIFICATION…

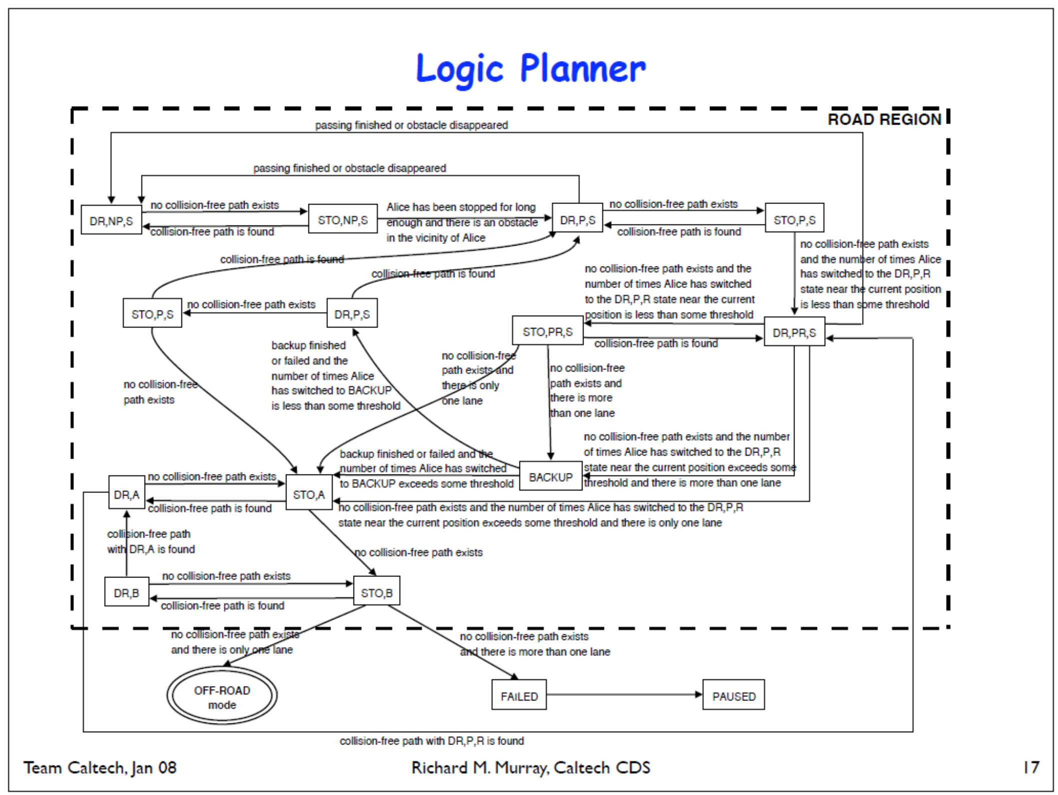
Alice Architecture

Richard M. Murray
Control and Dynamical Systems
California Institute of Technology

11 February 2009

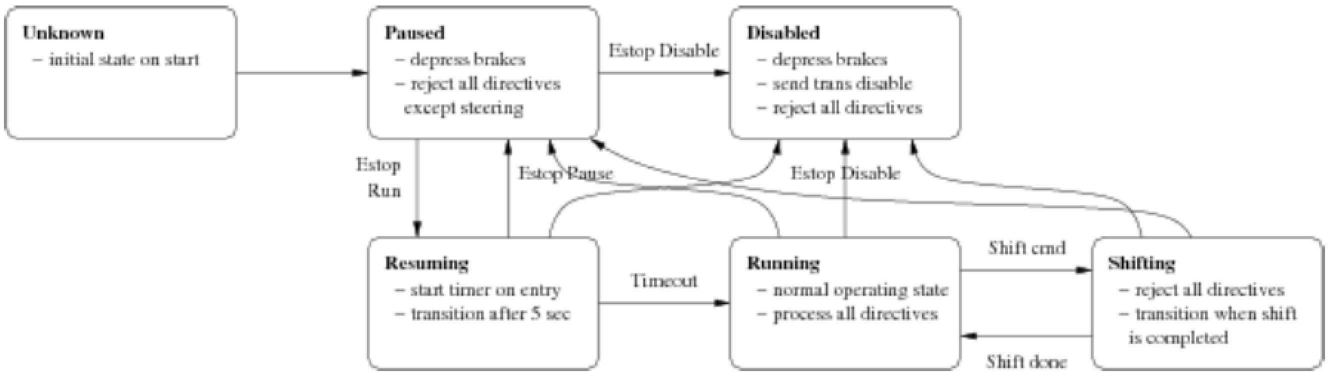
ARCHITECTURE…





METHODS & TOOLS…

GCDrive FSM Verification



Verification using temporal logic (Lamport's TLC, TLA+)

- Model follower, Actuation Interface, DARPA, accModule, transModule in TLC
- Shared variables: *state, estop, acc, acc_command, trans, trans_command*

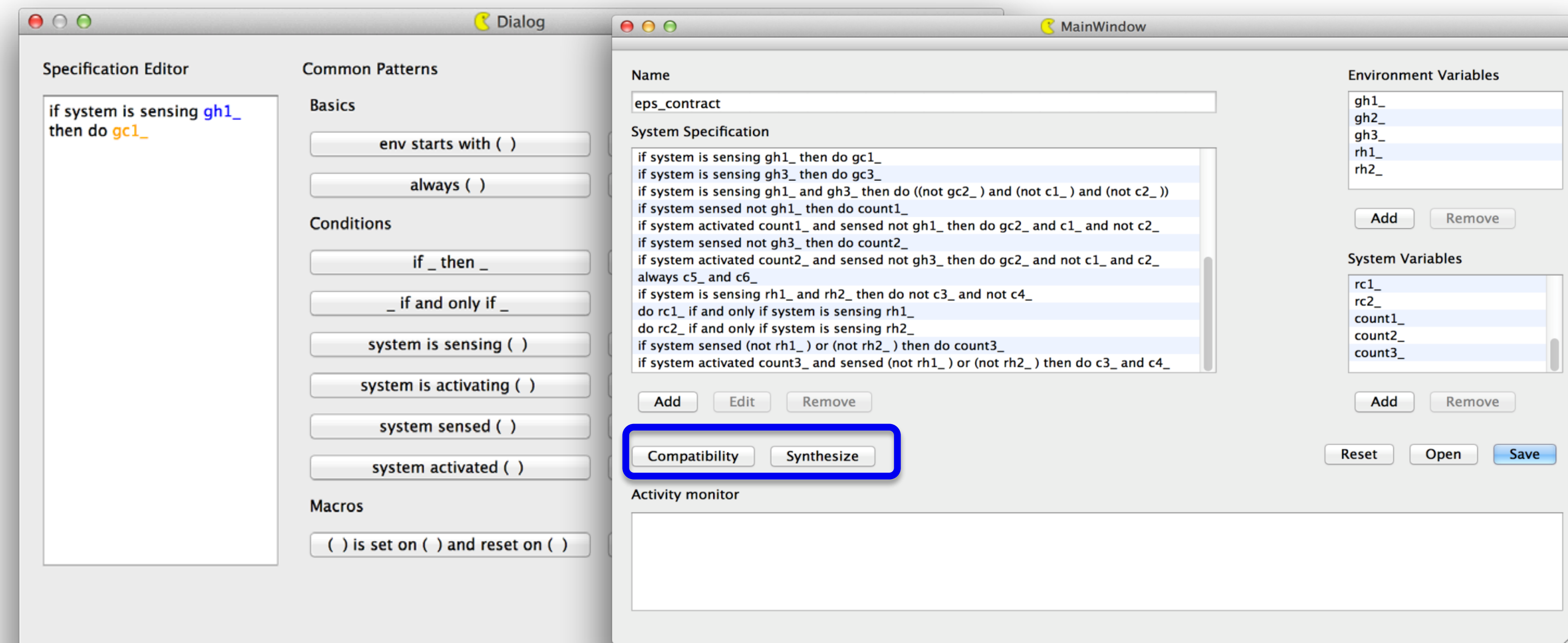
Verify the following properties

- $\Box((estop = \text{DISABLE}) \Rightarrow \Diamond\Box(state = \text{DISABLED} \wedge acc = -1))$
- $\Box((estop = \text{PAUSE}) \Rightarrow \Diamond(state = \text{PAUSED} \vee estop = \text{DISABLE}))$
- $\Box((estop = \text{RUN}) \Rightarrow \Diamond(state = \text{RUNNING} \vee state = \text{RESUMING}))$
- $\Box((state = \text{RESUMING}) \Rightarrow \Diamond(state = \text{RUNNING} \vee estop = \text{DISABLE} \vee estop = \text{PAUSE}))$
- $\Box((\text{state} \in \{\text{DISABLE, PAUSED, RESUMING, SHIFTING}\} \Rightarrow \text{acc} = -1)$

FROM STRUCTURED ENGLISH TO LOGIC

1. No AC bus shall be simultaneously powered by more than one AC source.
2. DC buses shall never unpowered.

Pattern-Based Contract Specification Language → **Temporal Logic** → **Compatibility? Consistency? Refinement?**



iCyPhy Source: P. Nuzzo, UCB

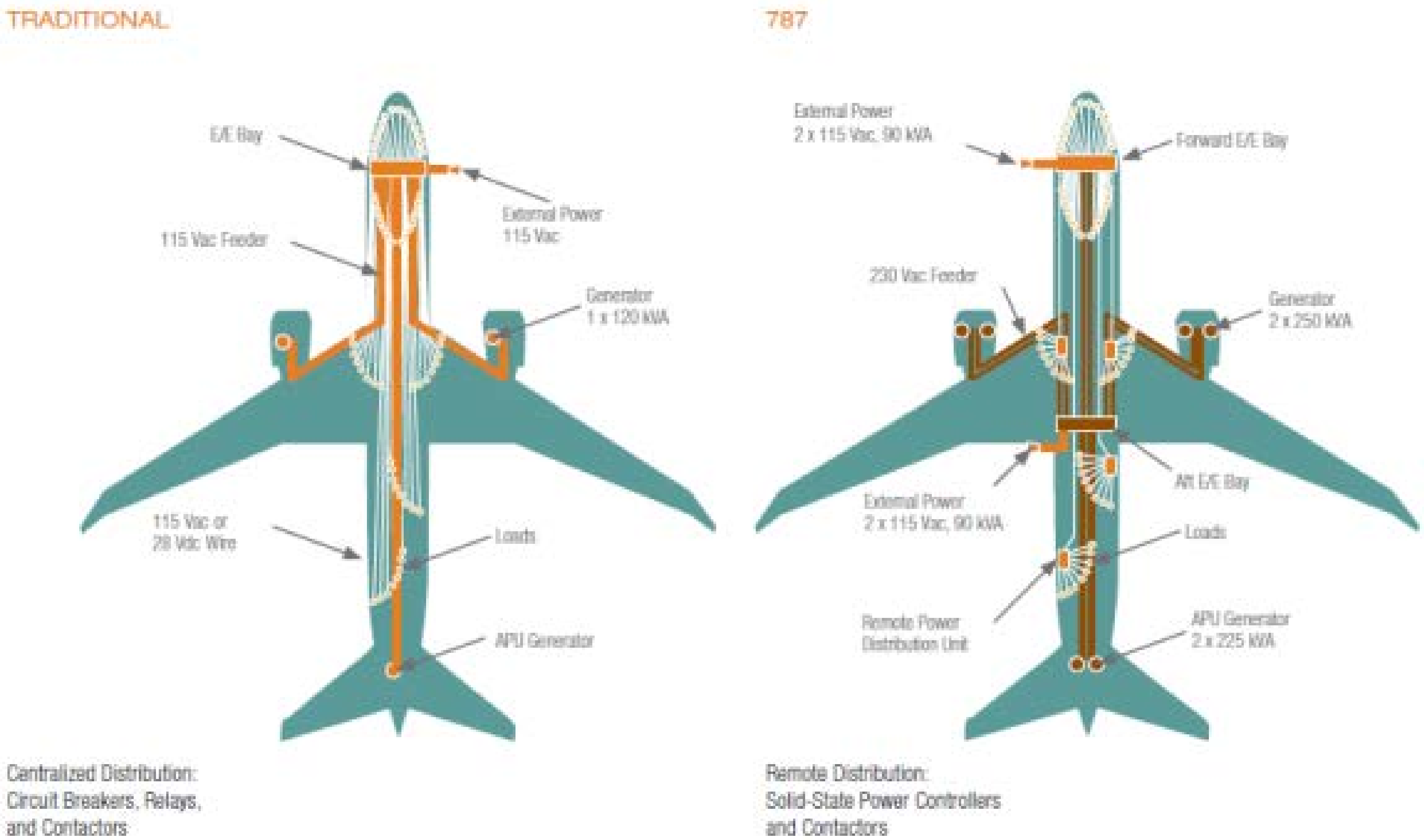
Analysis of Requirements: Overview

The Dilemma: complex systems leads to complex requirements
Complex behavior is difficult to capture in any natural language like English

Typical Requirement Flaws

- ***Ambiguity*** The natural language is not clear and it has to be "interpretation" is required
- ***Non-determinism*** The requirements allow to have choices at implementation level This does not mean that implementation must be non-deterministic.
- ***Inconsistency*** Some requirements are inconsistent to each other if they do not allow a solution that satisfies all of them.
- ***Vacuity*** A requirement is vacuous if by satisfying the other requirements it is implicitly satisfied.
- ***Realizability*** The requirement is not capable of being physically implemented
- ***Completeness*** All possible conditions would be covered.
- ***Extraneous*** The requirement does not belong to function being specified
- ***Negative*** Requirements makes verification difficult
- ***General*** Requirements makes verification difficult (always, under all conditions)

MORE ELECTRIC AIRCRAFT



Source: *787 No-Bleed Systems:* Saving Fuel and Enhancing Operational Efficiencies by Mike Sinnett, Director, 787 Systems, Boeing, 2007

SYSTEM SIZE AND INCREASED INTEGRATION

Increase reliance on electric power in aircraft raises complexity of system due to integration

Increased use of software and networks to provide system functionality

System Fault	Number of Configurations
No fault	1
Single contactor fault (Stuck Open)	~12
Single contactor fault (Stuck Open and Stuck Close)	~26
Single component fault (i.e. contactor, TRU, Bus, BPCU, GCU failure)	~40
Dual failure operation	~1,000

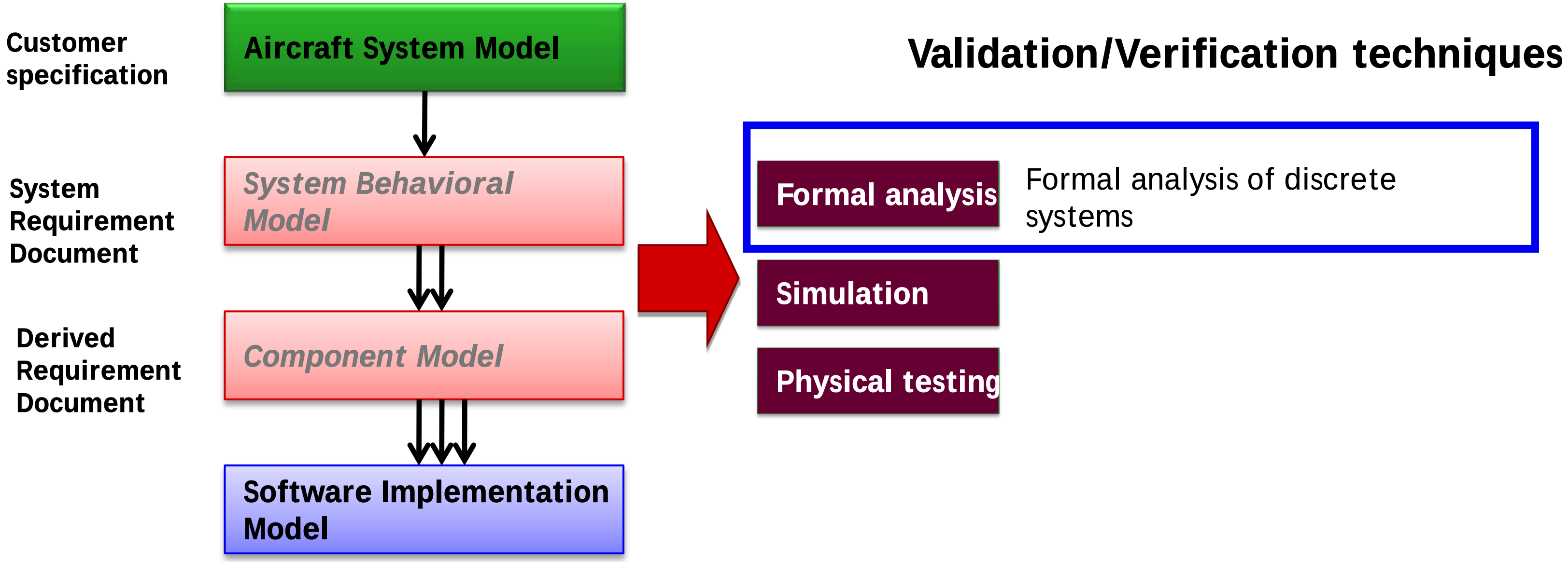
Typical conventional system (Single cruise mode system configuration)

MODEL BASED VERIFICATION TECHNIQUES

Need to manage complexity growth in cost/schedule effective manners

Develop models at the different abstraction layers to enable early and consistent guidance

Use analysis (formal analysis) to ***design and verify*** correct behavior at different layers

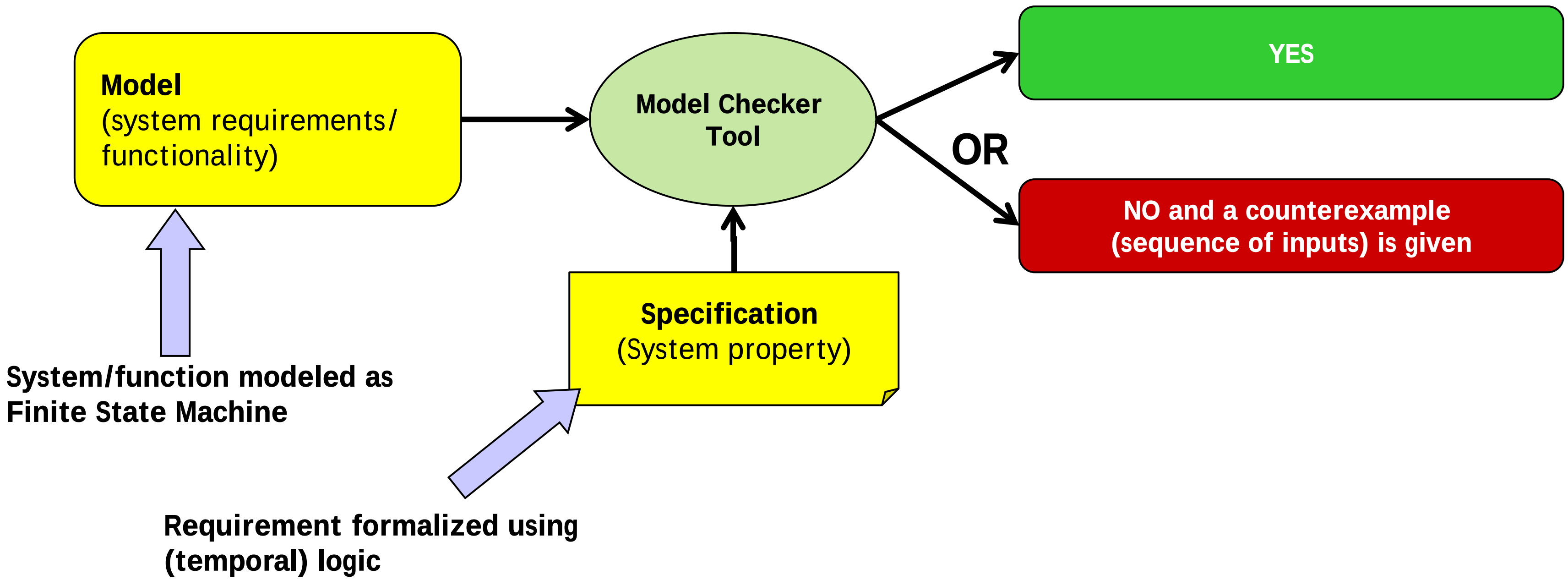


MODEL CHECKING

Use Formal Model of the controller/software and determine whether properties (i.e. requirements) are met for all possible input sequences

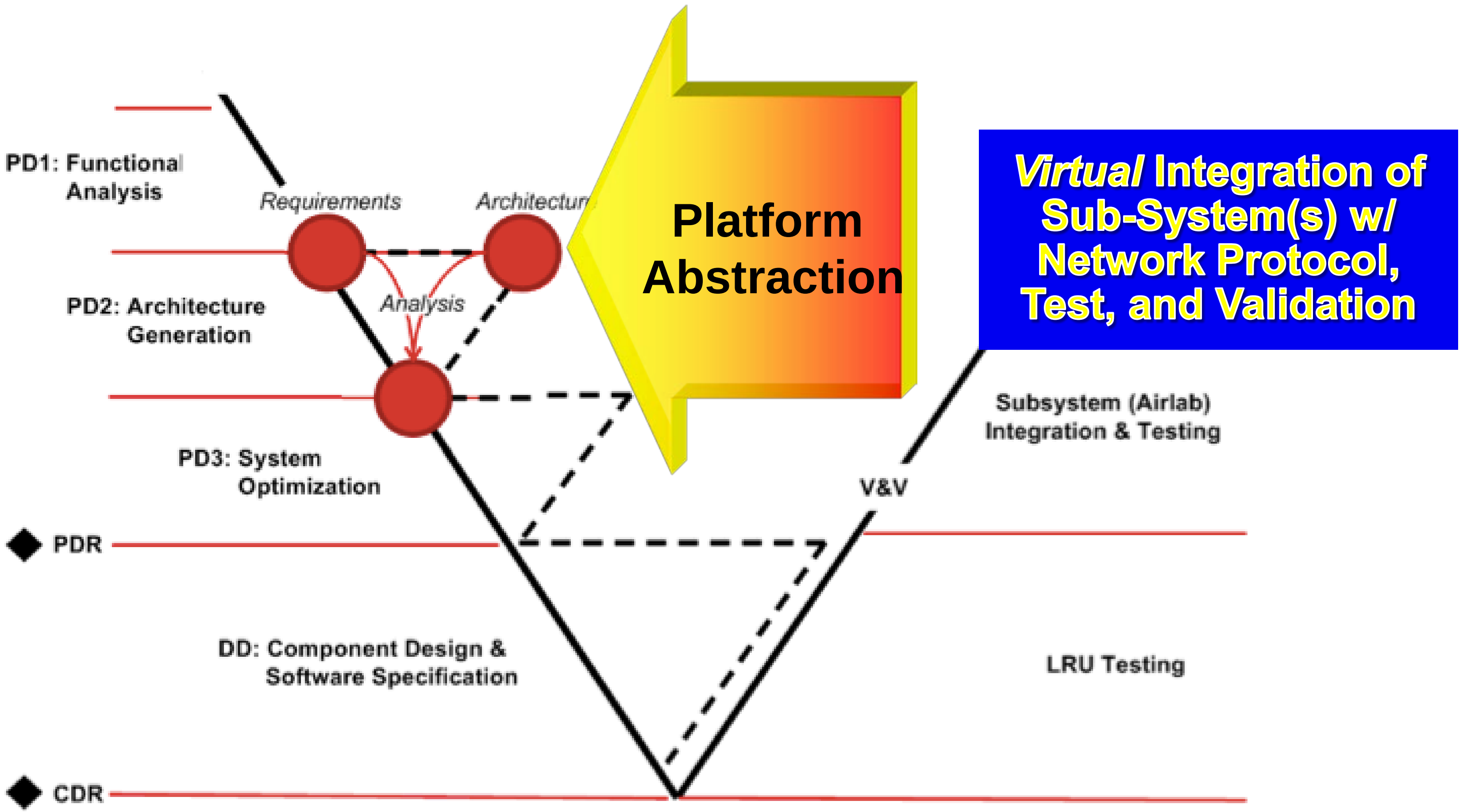
Looks at **all possible behaviors** of the system

Automated procedure if the system is **Finite State**



PLATFORM-BASED DESIGN

Executable specs, early validation, virtual platforms



AGENDA

Why?

What?

How?

- Verification – rigorous requirements, formal methods
- Variability – robust design (uncertainty quantification)
- Architecture - identification (and evaluation) (models)
- Dynamics (not done here) (models)
- Optimization (not done here) (models)
- Contract based design (not done here) (models)

Implications

FIXING DEFECTS…COST & PLACEMENT

Design Errors – What it Costs

The cost of fixing a single defect:

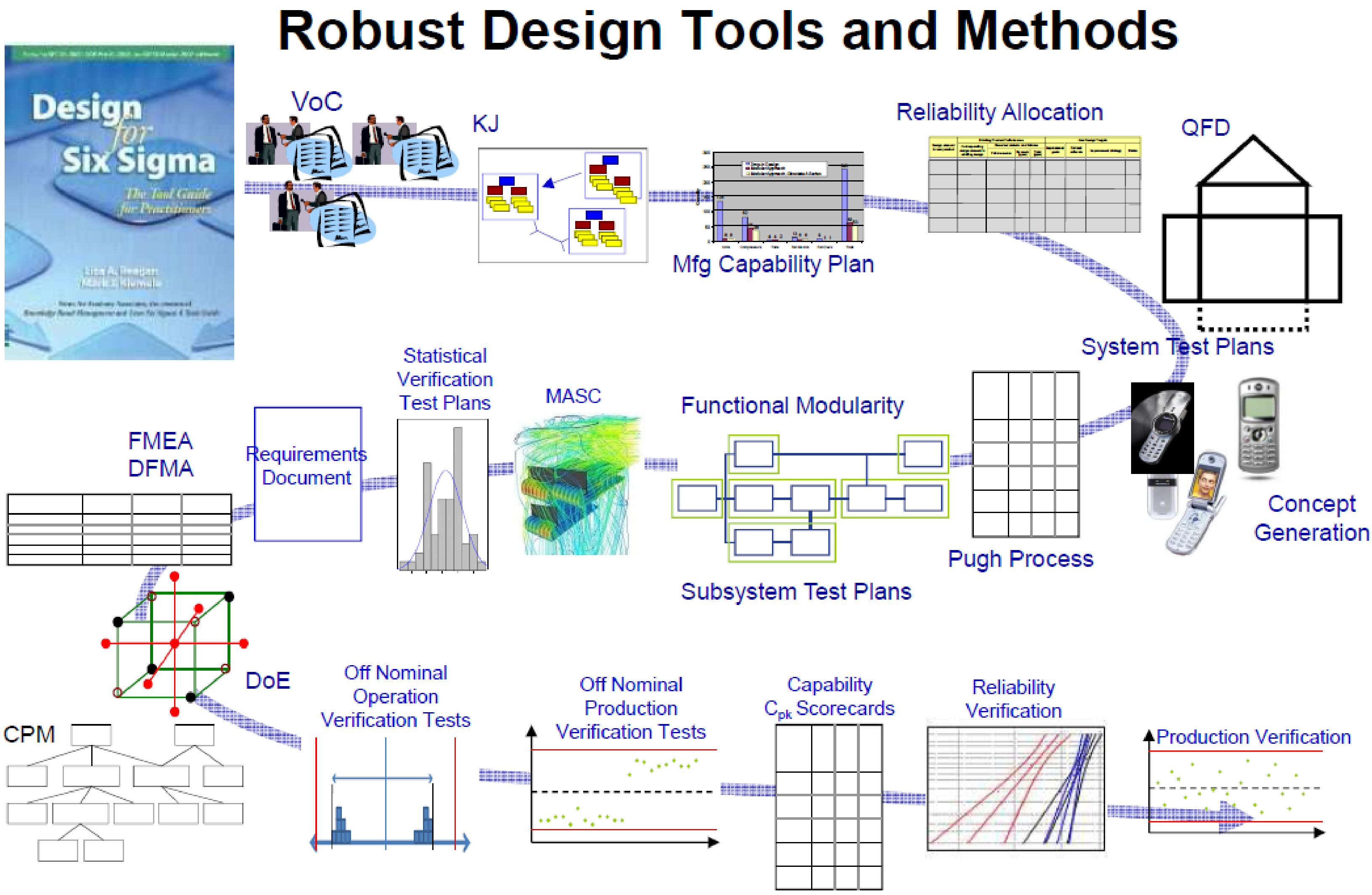
- $35 during the design phase
- $177 before procurement
- $368 before production
- $17,000 before shipment
- $690,000 on customer site

Mr. Hiroshi Hamada, President of Ricoh

Source: *European Community Quarterly Review*, Third Quarter 1996

RICOH
Image Communication

DESIGN FOR SIX SIGMA



BEST PRACTICE TO MITIGATE RISK

Design for Six Sigma: Methods & Tools

Customer voiced requirements

QFD conversion to measurable metrics

Concept Engineering

Target cascading

Potential (design) FMEA

Design of Experiments

Critical Parameter Management

VARIATION: LEADING INDICATOR OF YIELD

"Statistical Engineering"

Focus on variation in the context of the **Laws of Conservation of Energy & Mass**

σ Variation in **Energy** Flow, State & Transformations

These are ***leading indicators*** **of reliability & quality**

WHAT exactly do we measure?

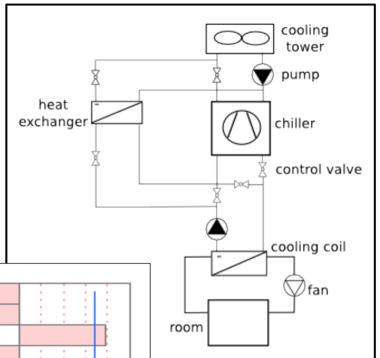
σ Variation in **Mass** Flow, State & Transformations

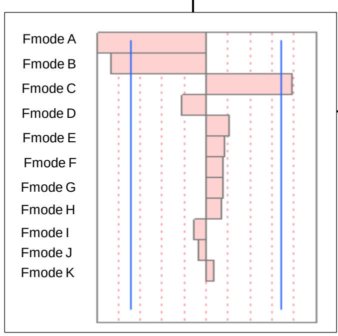
σ Variation in **Information** Flow, State & Transformations

CRITICAL PARAMETER MANAGEMENT

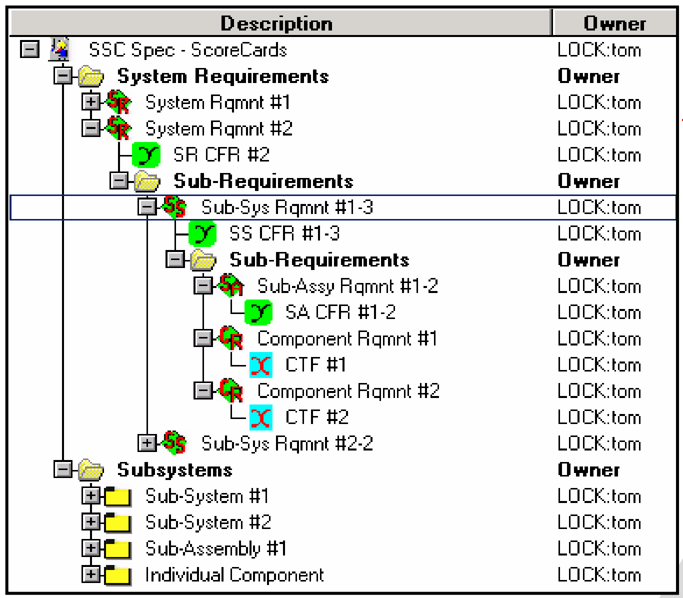
CPM is the analytic ability to compute performance variance statistics from the sensitivity impact of low level design variation all the way up to customer operation experience

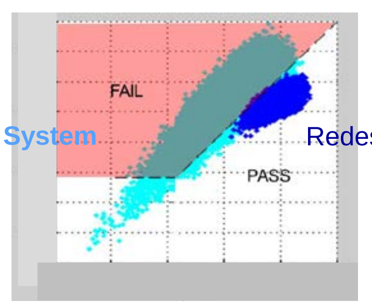
Model





Causes





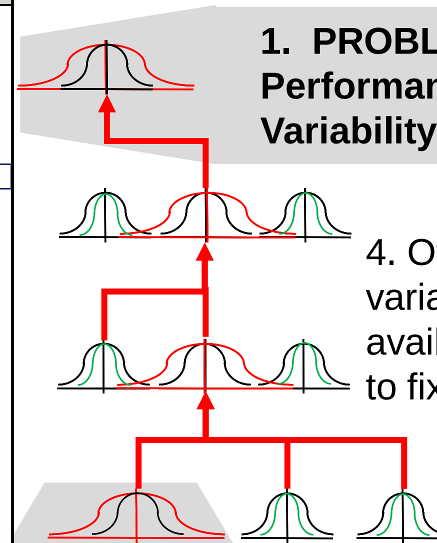
Initial System

Redesigned System

Yield Prediction

Rapid Mitigation

1. PROBLEM: Performance Variability Gap



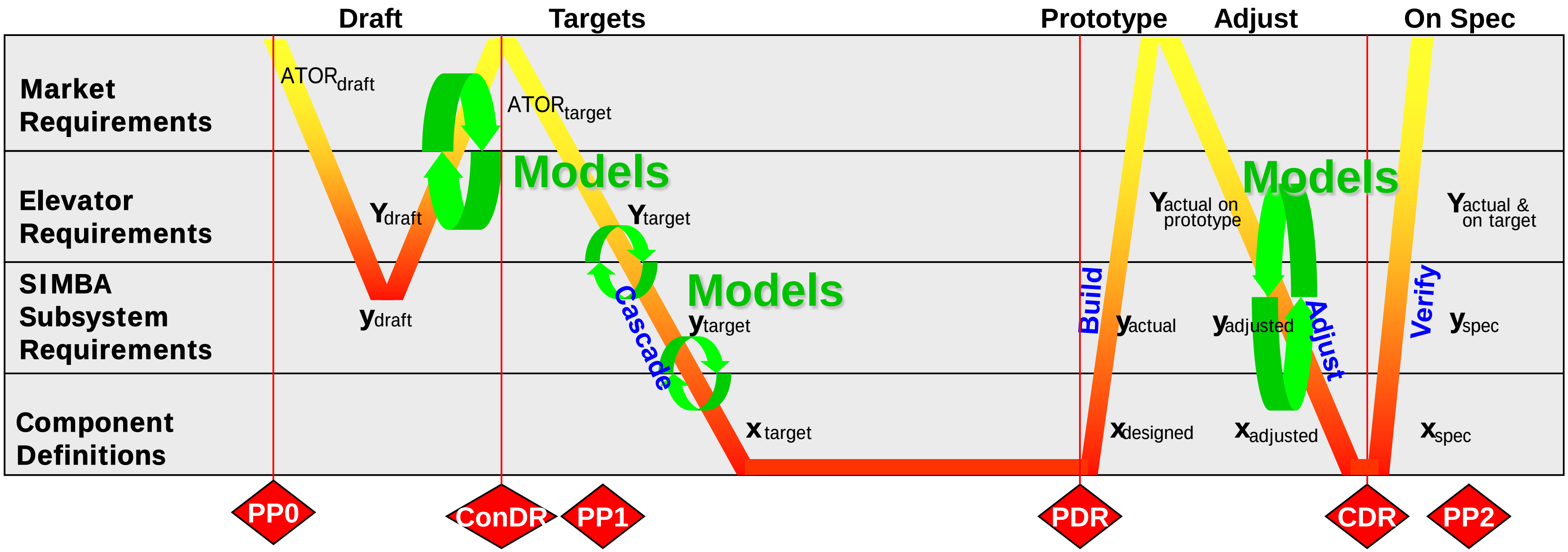
4. Other system variables available instead to fix problem

2. Root Cause of unexpected excess variation

3. Other component variables available instead to fix problem

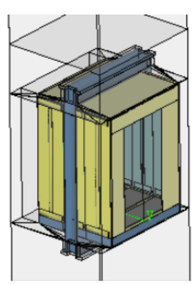
Model-Driven Product Development

MASC provides a basis for allocating system requirements to components.

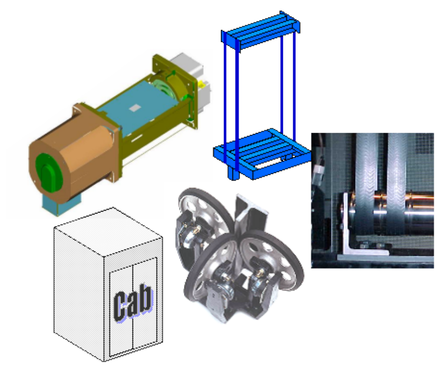


1 months   1-2 months   1-3 months

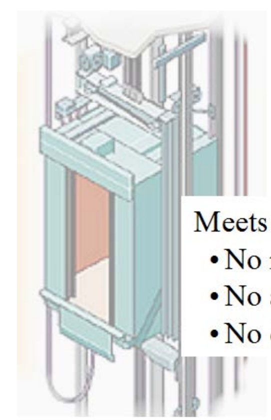
Marketing & Engineering
(by MASC & Customer Req.)



Model System
- predict feasible performance
- flow down requirements
- create component (SIMBA) level requirements



Subsystems developed with appropriate system level noise requirements



Meets Requirements
- No major redesigns
- No added cost
- No delays

AGENDA

Why?

What?

How?

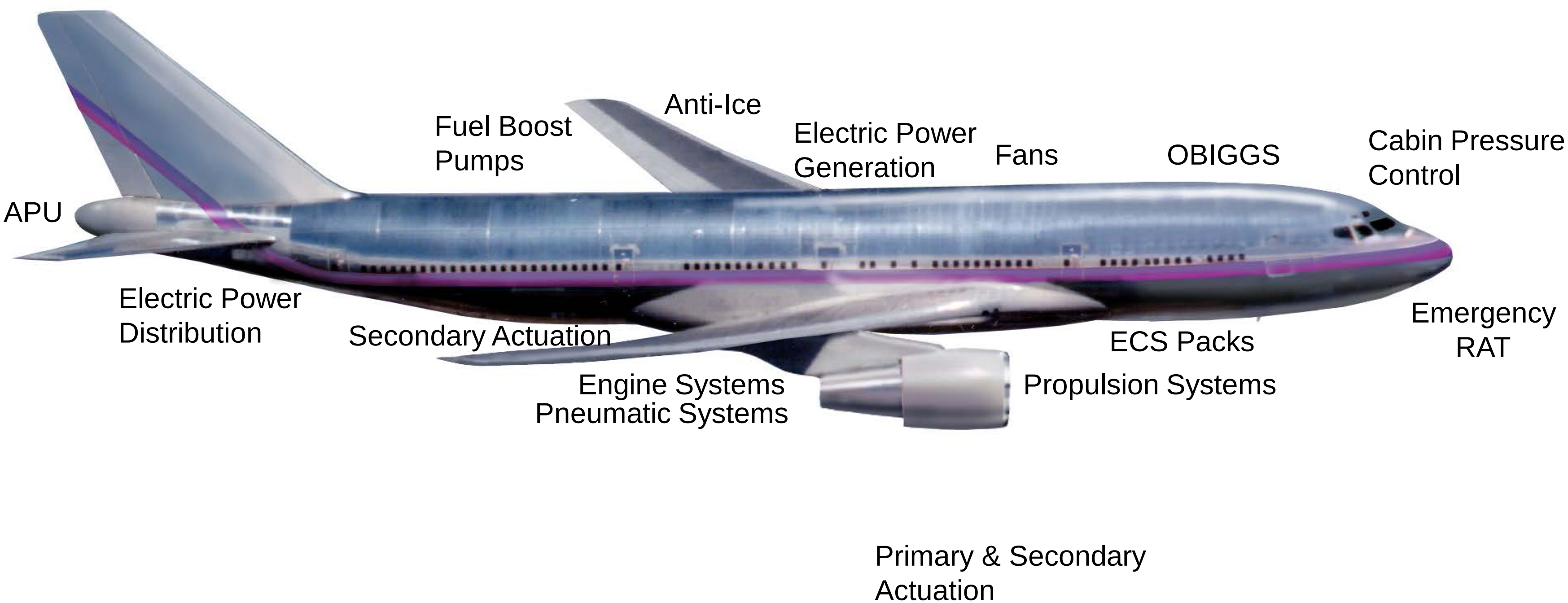
- Verification – rigorous requirements, formal methods
- Variability – robust design (uncertainty quantification)
- Architecture - identification (and evaluation) (models)
- Dynamics (not done here) (models)
- Optimization (not done here) (models)
- Contract based design (not done here) (models)

Implications

ITAPS: Integrated Total Aircraft Power Systems

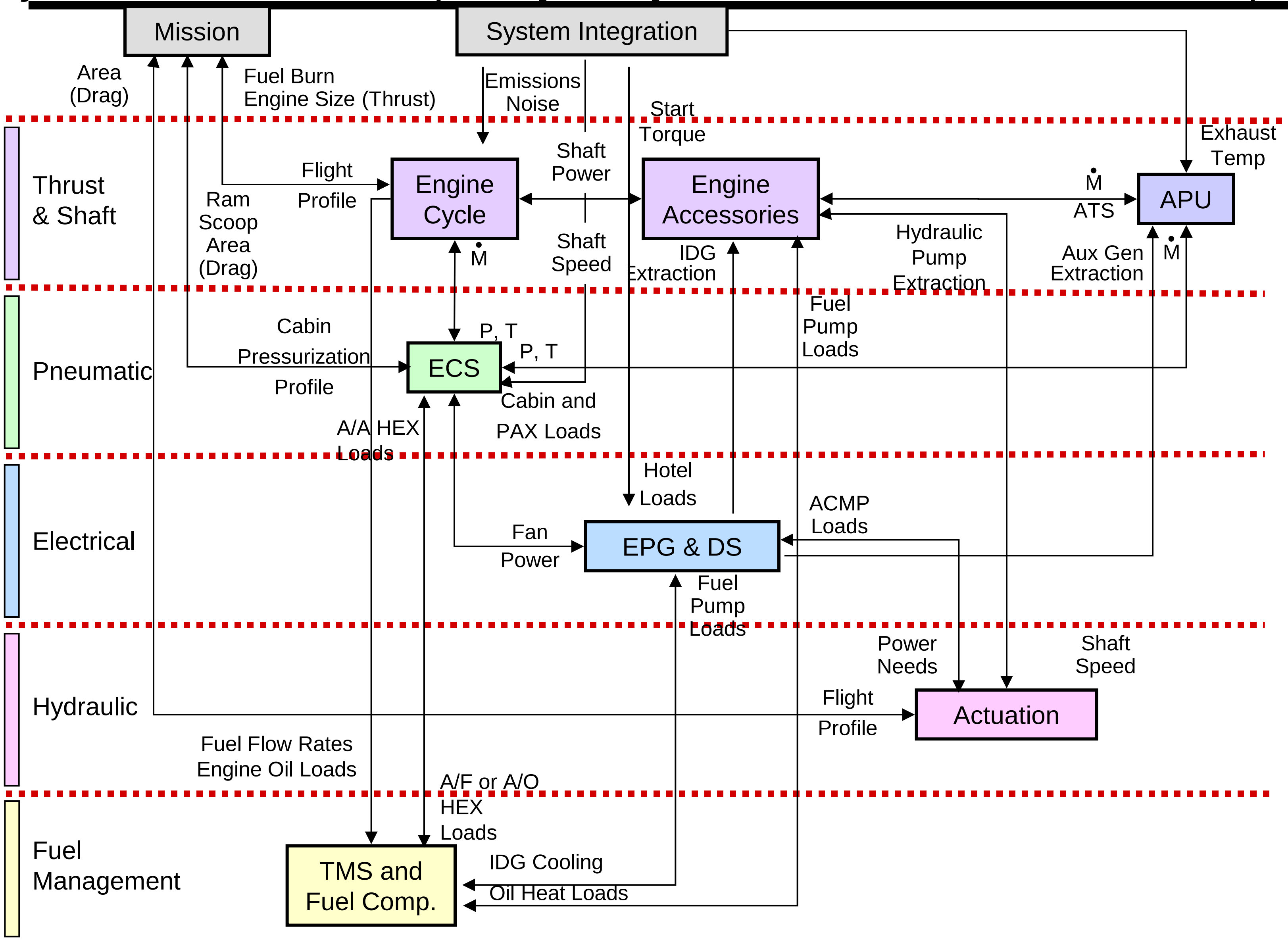
UTC is uniquely positioned to assist airframers in developing system solutions

- Complete Power, Fuel and Thermal Management Product Portfolio
- ITAPS has assembled this capability into a functional entity
  - Concepting methodology for generating ***integrated*** power system architectures
  - Integrated power system design tools
  - Process for accelerating technology development of enablers identified in studies
  - **ITAPS welcomes airframer participation, shares responsibilities**
  - **ITAPS is willing to partner to extend system scope beyond product line**



Concept Evaluation Phase

System-level multi-disciplinary analysis establishes vehicle-level impact



IMPLICATIONS

Industry

Product & product development – (potentially huge) impact to industry practice; *barriers are skills, scalability of methods, full embrace of computational methods & tools - numerics) and overall cost*

Policy

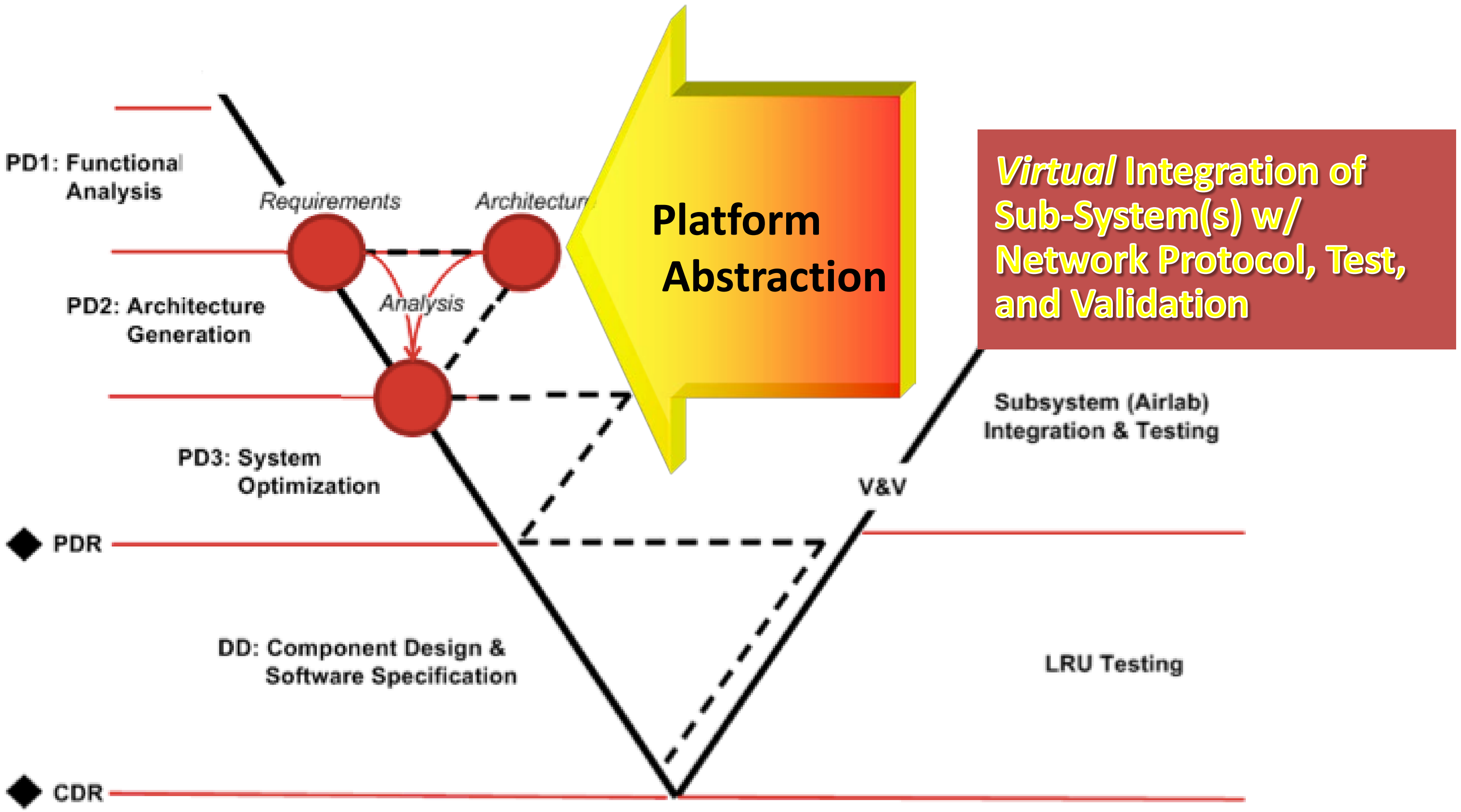
Research – opportunities and need (compelling – DARPA, NSF, EU)…*barrier is the need to effectively encourage and promote – missing National impact*

Academia

University curricula – fundamental changes needed, *barrier is faculty background & skills and siloing in departments; mathematics & methodology (& tools)*

Desired Design Approach: Platform-Based Design

Executable specs, early validation, virtual platforms



iCyPhy – UC Berkeley, Caltech, IBM, UTC

Structure of the Research Program

Task 0
Requirement formalization and Contract-Platform-Based Design Methodology

Task 1.1
Co-Simulation foundation and architecture

Task 1.2
Co-Simulation platform implementation

Task 2
MoCs for SysML

Task 3
Formal Control Synthesis enhancements

Task 4
Design Space Exploration and Performance Analysis

Task 5.1
Design Driver Modeling

Task 5.2
Design Driver Modeling

Task 5.3
Design Driver Modeling

KEY POINTS

Product development processes – how products are developed – are under pressure to deliver more with less. More functionality, shorter schedules, more software, more criticality – these are all drivers that push current approaches beyond what the processes and people can deliver. (Cost vs cost/benefit)

Systems engineering is a science. Systems engineers are not (only) “experienced engineers” – there are methods & tools that can and should be applied in a discipline.

Methods and tools define systems engineering (a) requirements analysis, (a) architecture analysis, (c) model based development and (d) design flows. A large amount of analysis.

Implications: all about leadership, output & impact…

For industry – recognition and adoption of systems engineering is a competitive positioning – needs to be done correctly and efficiently…

For academia – curricula in systems engineering do not exist and real experience in systems engineering largely lacking in academia. Customers and (national) needs are not being met.

For research entities – funding programs need definition, scope and industrial partnering. NSF, DARPA, EU programs all need to be encouraged.