ENSE 623: System Design Projects, Validation and Verification

Quick Review of ENSE 621 and ENSE 622

Mark Austin

E-mail: austin@isr.umd.edu

Institute for Systems Research, University of Maryland, College Park

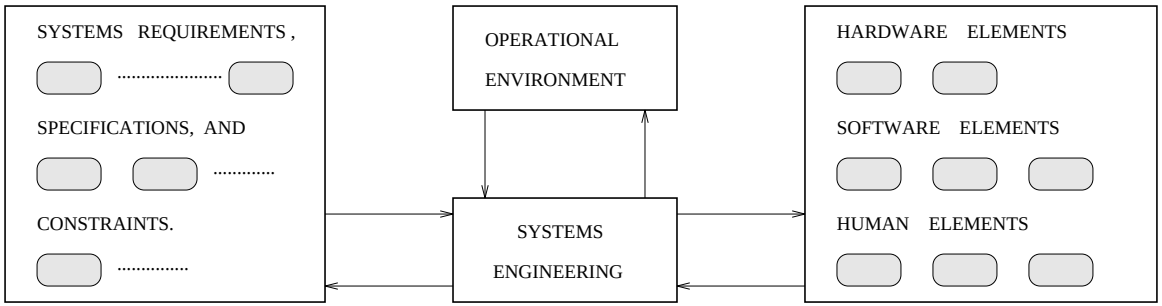
Quick Review of ENSE 621 and ENSE 622

Topics:

1. Our definition of Systems Engineering.
2. Systems Engineering in Mainstream US Industry.
3. Models of Systems Engineering Development (e.g., Waterfall, Spiral).
4. Economics of development.
5. Systems Engineering Drivers
6. Strategies for Systems Engineering Development
7. Key Steps in ENSE 621 and ENSE 622.

Our Definition of Systems Engineering

Systems engineering is a discipline that lies at the cross-roads of engineering and business concerns.



Specific goals are to provide:

1. A balanced and disciplined approach to the **total integration** of the system building blocks with the surrounding environment.
2. A methodology for systems development that focussed on **objectives**, **measurement**, and **accomplishment**.
3. A systematic means to acquire information, and sort out and identify areas for **trade-offs** in cost, performance, quality etc....

Practicing Systems Engineers

Typical concerns on the design side:

1. What is the required functionality?
2. How well should the system perform?
3. What about cost/econmics?
4. How will functionality/performance be verified and validated?

Typical concerns on the management side:

1. What processes need to be in place to manage the development?
2. What kind of support for requirements management will be needed?

Learning how to deal with these concerns in a systematic way is a challenging proposition driven, in part, by a constant desire to improve system performance and extend system functionality.

Understanding System Complexity

To understand a system, you really need to understand:

1. The ways in which it will be used,
2. The environment in which it will operate, and
3. The knowledge, technologies, and methods that go into making it.

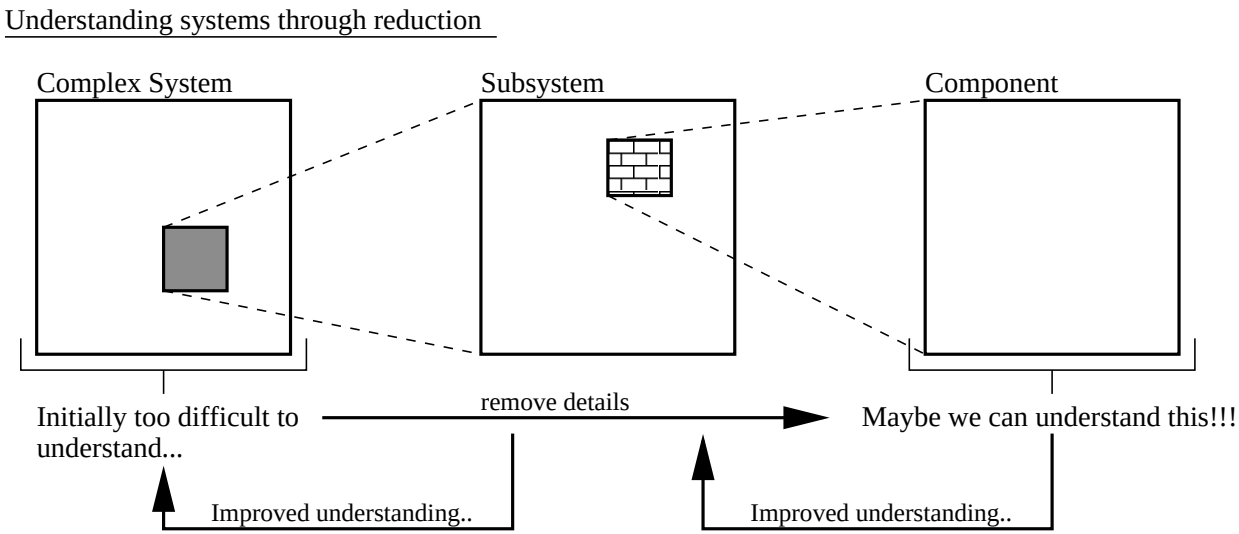
For a wide range of engineering applications this problem is quite tractable.

However as systems become more complex, we need to be strategic in the way we approach design, i.e., points to the importance of:

1. System Decomposition (to simplify design).
2. Abstractions (to simplify decision making in design).
3. Formal Analysis (our understanding of system behavior needs to be right).

Understanding System Complexity

Strategy: Put original problem aside and focus on understanding the collection of subsystems that make up the orginal system.

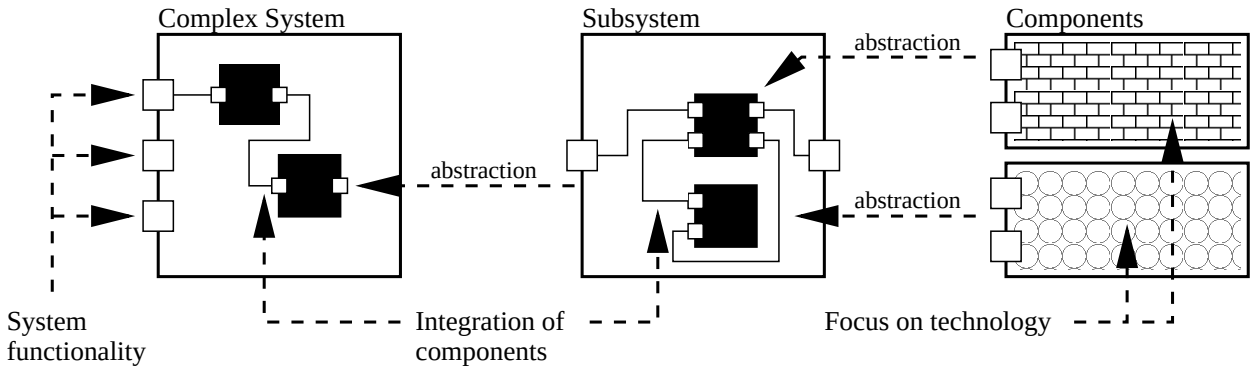


Common questions include:

1. What are the subsytems and how are they connected internally?
2. How does the system interact with the surrounding environment?

System Assembly via Integration of Abstractions

System assembly through integration of abstractions



Observations

Increasing importance of technology

Increasing range of functionality

Increasing opportunity for reuse of lower level entities

Engineering Concerns

Increasingly heterogeneous ... Increasingly homogeneous

Increasing use of abstraction

Increasing need for formal analysis

SE in Mainstream US Industry

Traditional engineering and systems engineering serve complimentary roles:

- **Traditional Engineering**.

  Focus on generation of knowledge needed to ceate new technologies and new things.

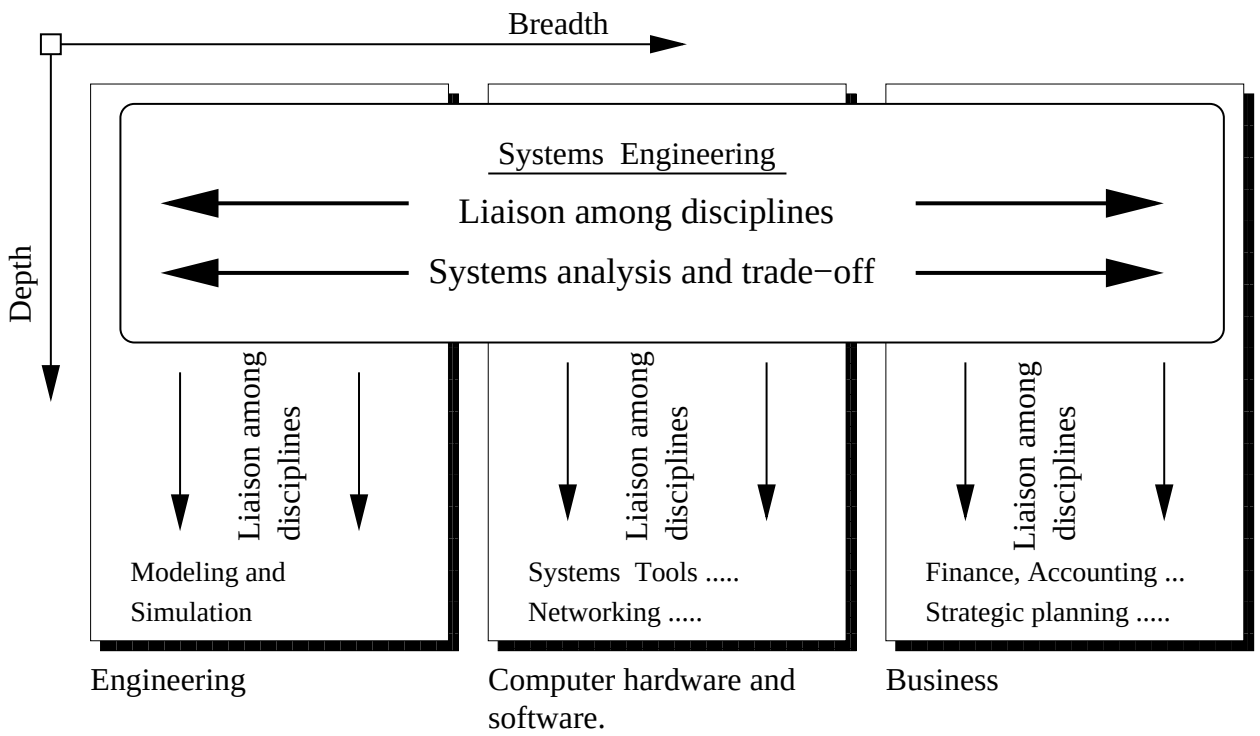
- **Systems Engineering**.

  Focus on understanding how existing technologies and things can be integrated together in new ways (...to create new kinds of systems).

So here's the bottom line:

... systems engineers need traditional engineers, and vice versa.

SE at the Organizational Level

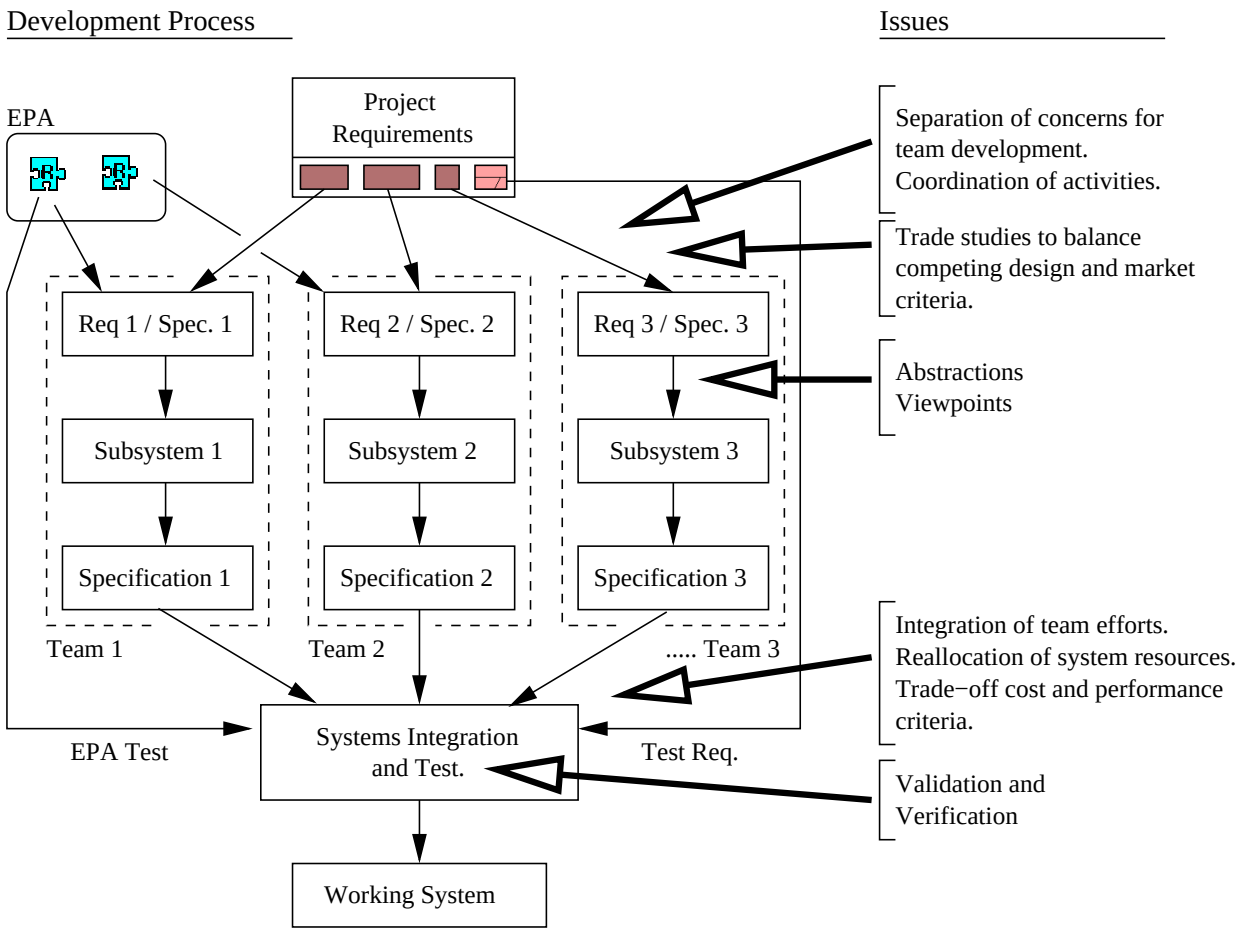


Focus on:

...liaison among disciplines, supported by formal methods for systems analysis and design.

SE at the Project Level

Systems are developed by teams of engineers – the team members must be able to understand one-another's work.

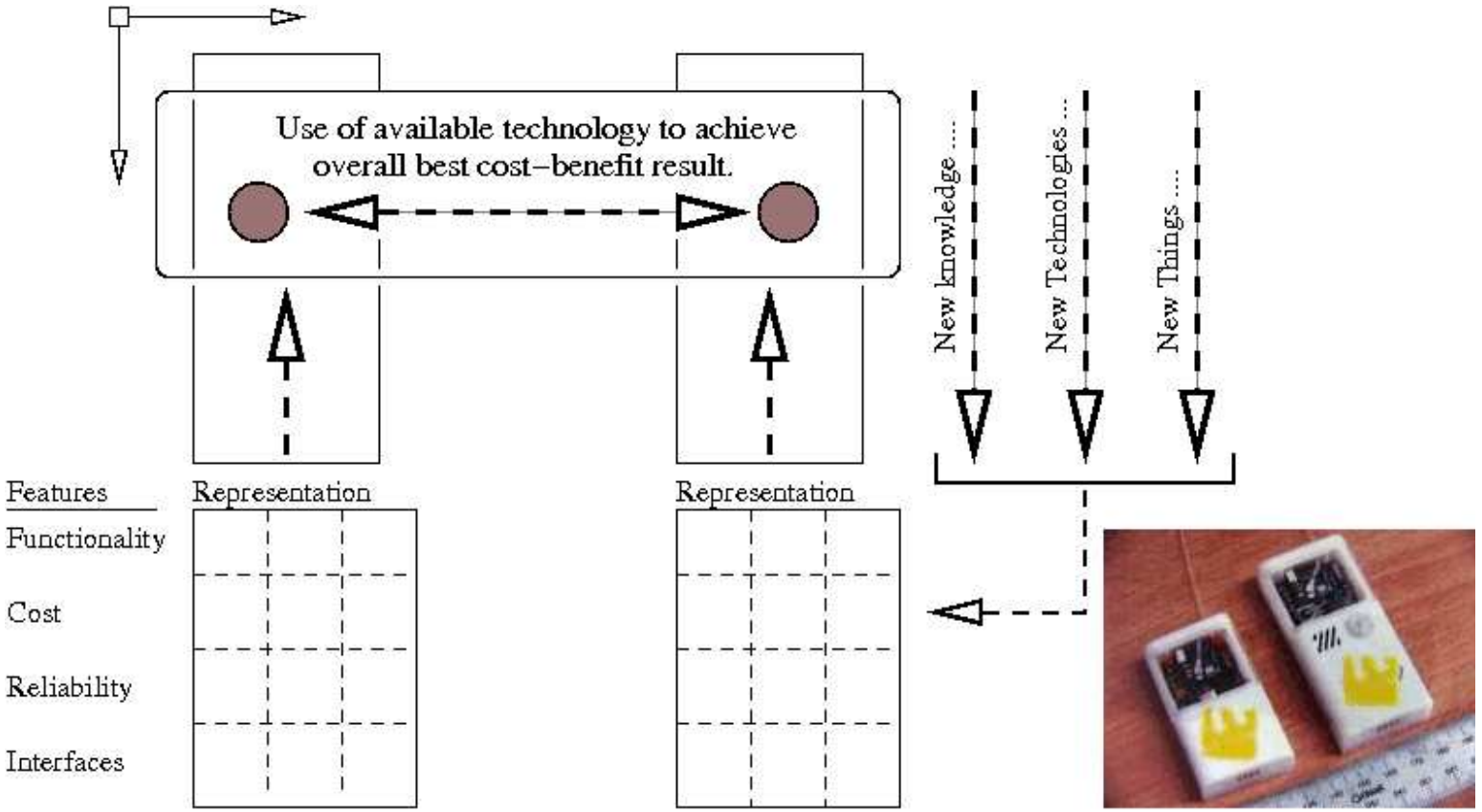


SE at the Project Level

Key concerns:

1. Partitioning of the design problem into several levels of abstraction and viewpoints suitable for concurrent development by design teams;
2. Synthesis of good design alternatives from modular components;
3. Integration of the design team efforts into a working system; and
4. Evaluation mechanisms that provide a designer with critical feedback on the feasibility of a system architecture, and make suggestions for design concept enhancement.
5. Formal methods for early validation/verification of systems.

SE at the Product Level



SE at the Product Level

Key concerns:

1. How to describe what a product does? Can this be done formally?
2. How to describe pre-conditions for using a product?
3. How to describe a products interfaces?
4. How to describe various representations (visual, mathematical).

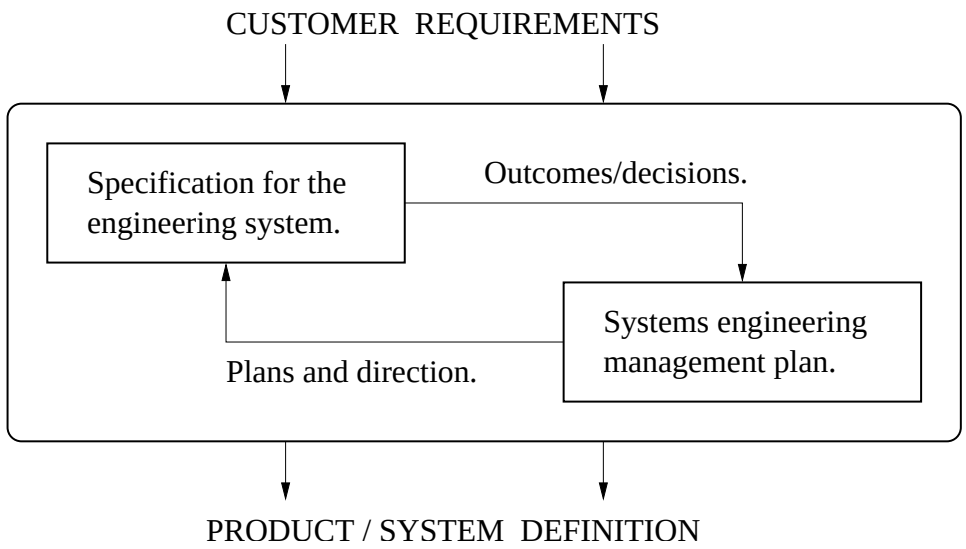
Systems Engineering Processes

Pre-defined plans of development ...

... provide the discipline to keep development activities predictable and on track.

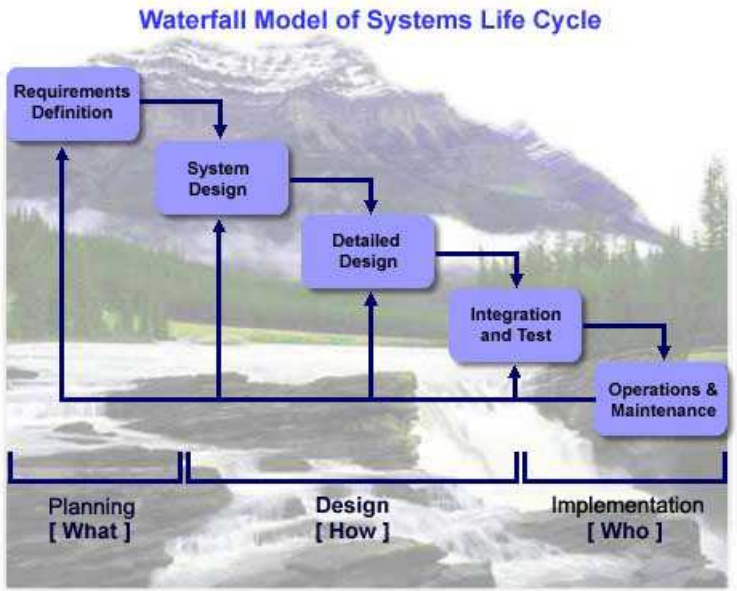
The project participants know what's expected and when.

Interaction of technical development and engineering management processes



During the past 3-4 decades this approach to system development has served many industry sectors (e.g., aerospace) well.

Waterfall Model of Development

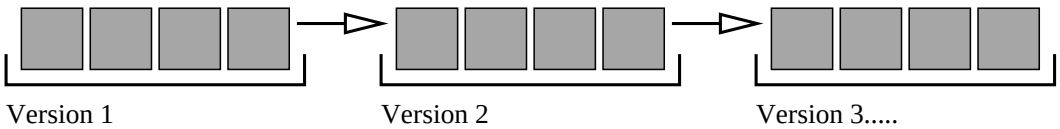


The waterfall model works well when:

... problem and solution method are well understood, requiring no large-loop corrections to development problems.

Iterations of Waterfall Development

Iterations of Waterfall Development.....

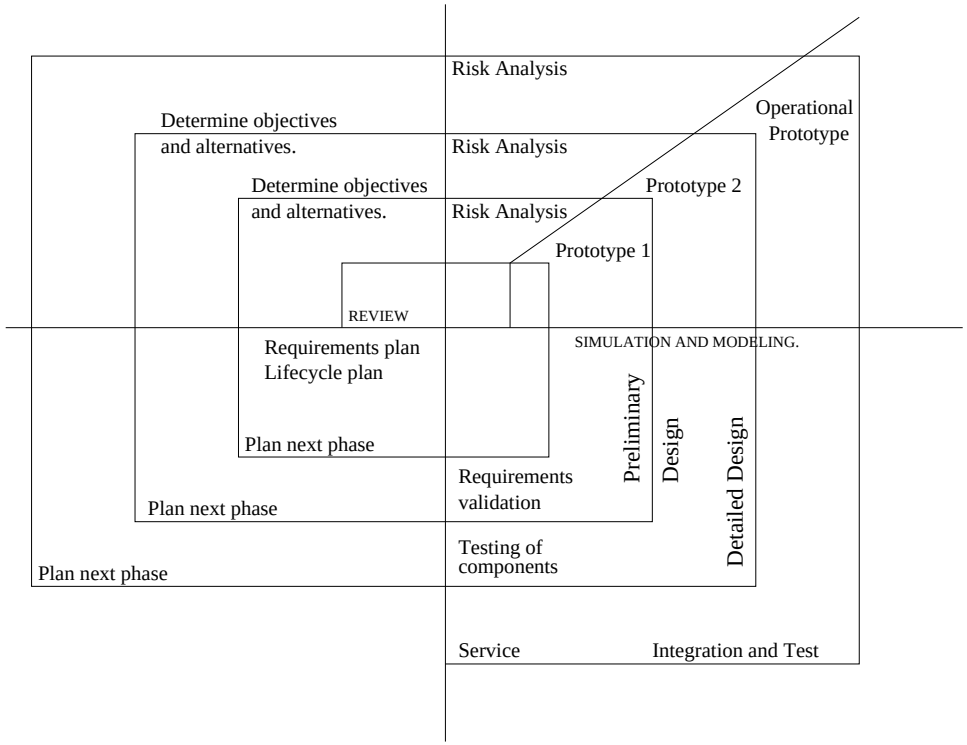


Limitations of Waterfall Model

- Changing requirements proved to be the biggest cause of cost overruns and schedule slips in the waterfall era.
- Users were found to be unable to define the requirements of a complex system without having had hands-on previous experience with the system – A Catch 22.

Spiral Model of Systems Development

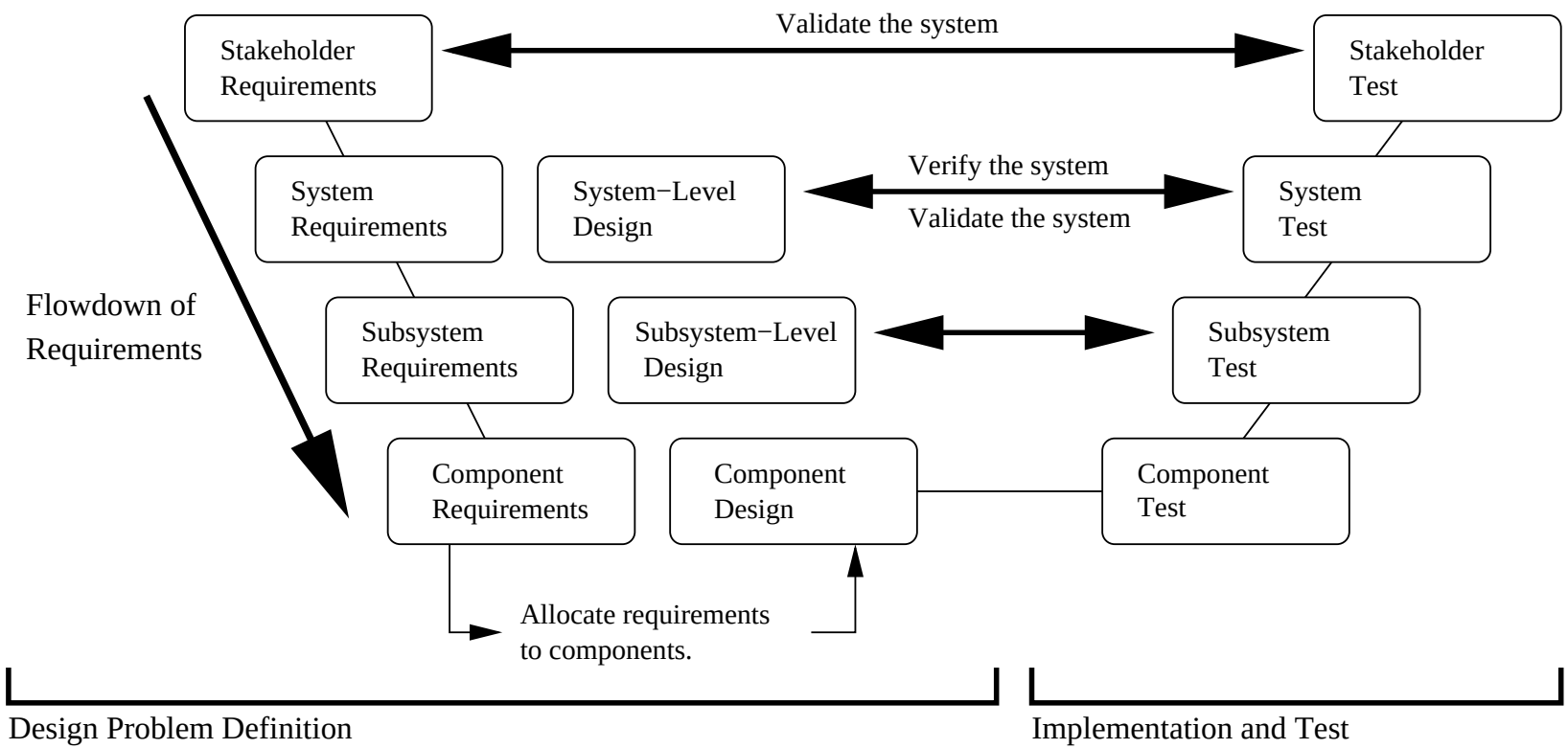
Spiral model corresponds to risk oriented iterative enhancement.



Categories of risk include: technical risk, schedule risk, cost risk, programmatic risk.

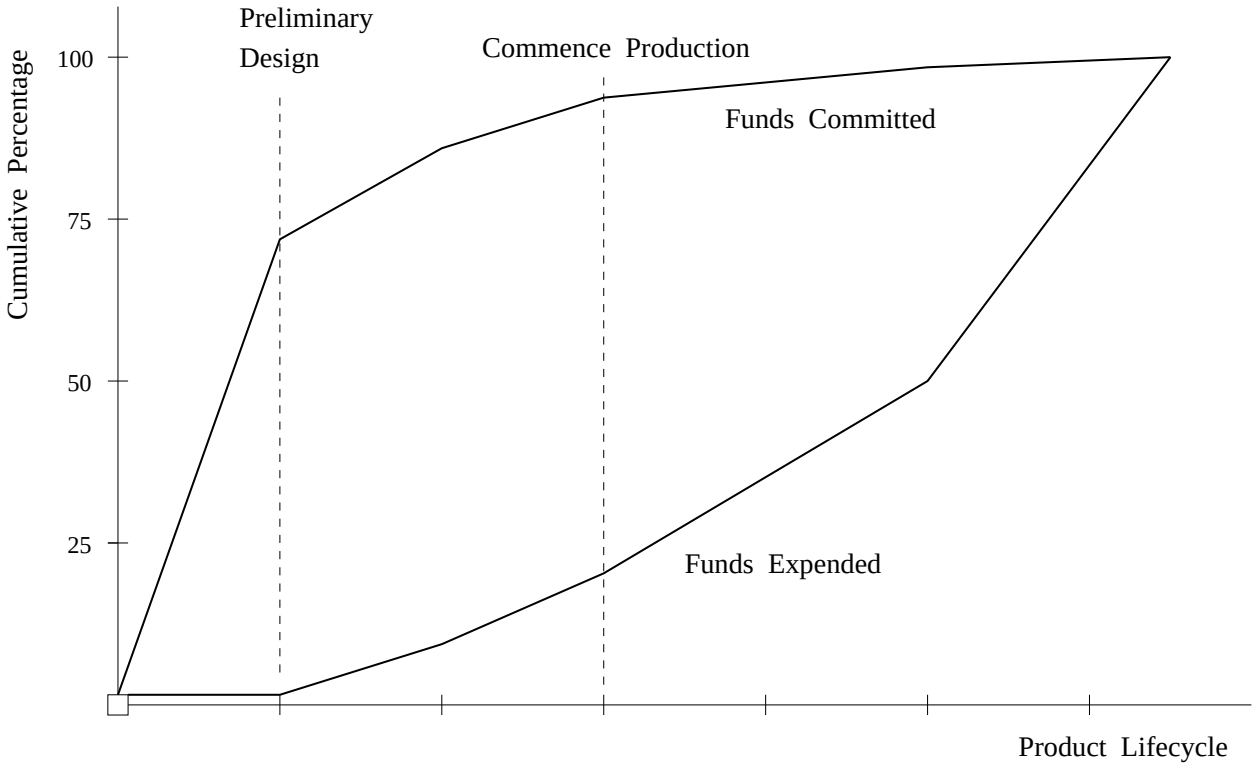
V-Model of Systems Development

Flowdown of requirements in the V-Model of system development.



Economics of Development

Funding Commitments in Product Life-Cycle



Economics of Development

Cost of Correcting Design Errors

Project Phase	Bug Description	Relative Cost
Design	Design Team	1
Write and Test	Designer	10-20
Quality Assurance	QA Personnel	70-100
Shipment to Customer	Customer	Very-expensive

Systems Engineering Drivers

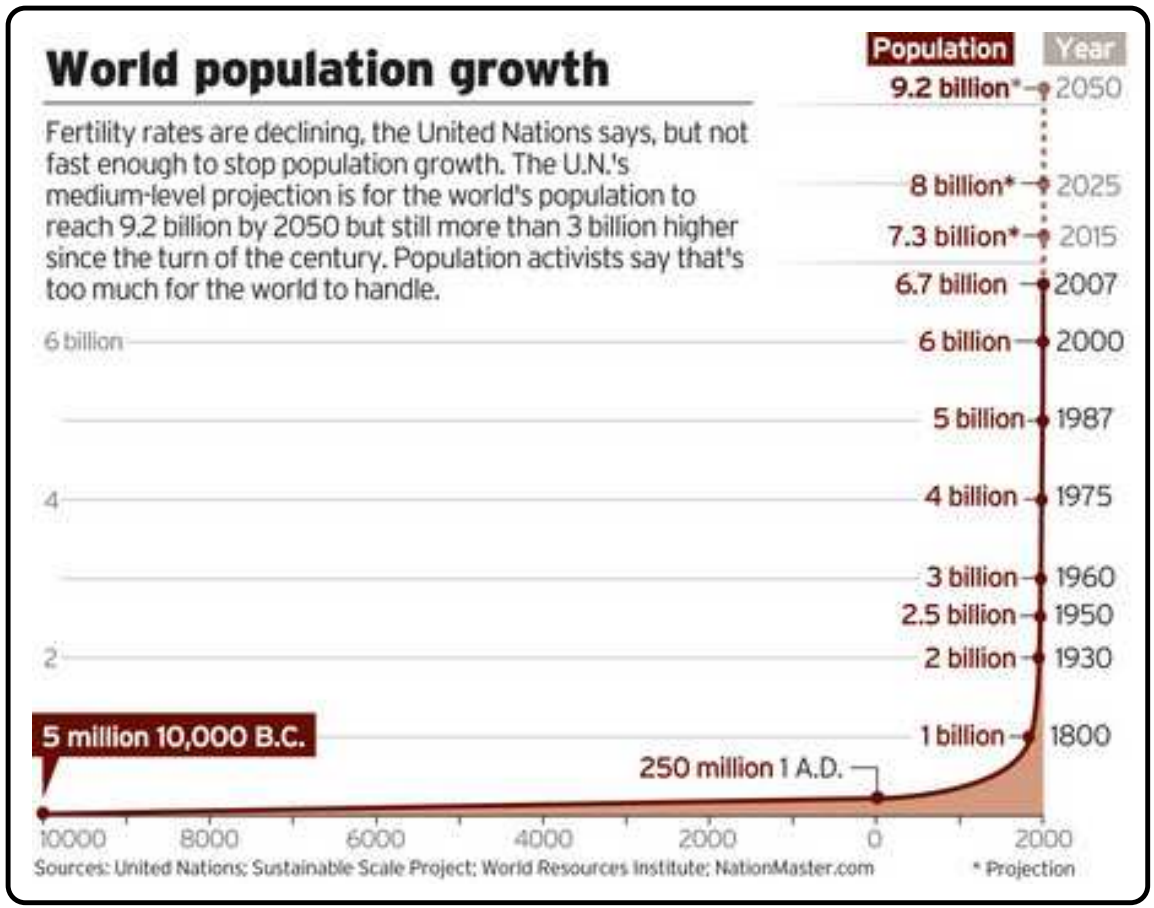
Systems Engineering Drivers

Several important developments that have rendered systems engineering methodologies, tools, and educational programs critical. They are:

1. Rapid changes in technology;
2. Fast time-to-market most critical;
3. Increasing higher performance requirements;
4. Increasing complexity of systems/products;
5. Increasing pressure to lower costs;
6. Increased presence of embedded information and automation systems that must work correctly; and
7. Failures due to lack of systems engineering.

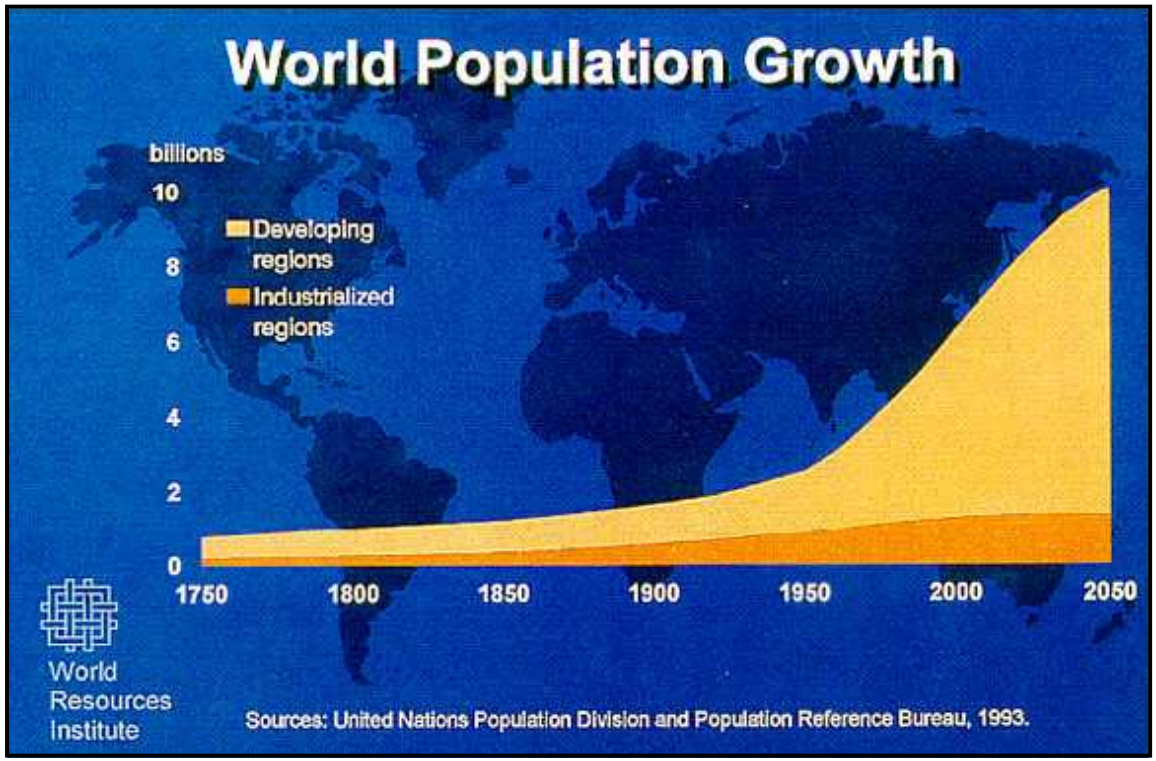
Challenge 1. Increasing Demand for Resources

Trends in World Population Growth



Challenge 1. Increasing Demand for Resources

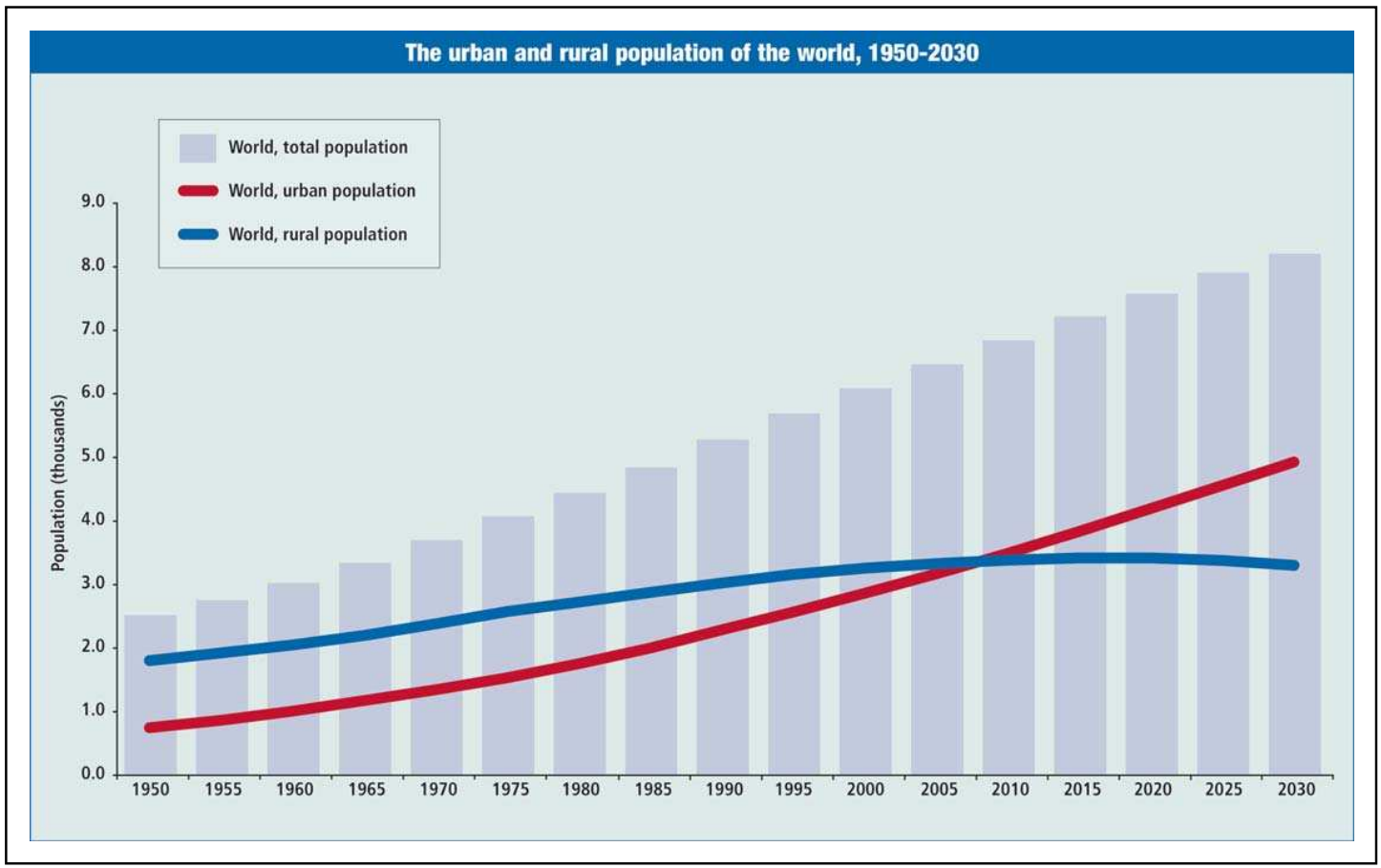
Trends in World Population Growth



Global population is growing along with growing affluence. This creates additional system demands. **Are these trends sustainable?**

Challenge 1. Increasing Demand for Resources

Rural to Urban Population Drift



Challenge 1. Increasing Demand for Resources

Urbanization in America

- In 2010, 82 percent of Americans lived in cities.
- By 2050 it will be 90 percent.

Cities are responsible for:

- Two thirds of the energy used,
- 60 percent of all water consumed, and
- 70 percent of all greenhouse gases produced worldwide.

Sustainable cities are looking at ways to ...

> **... improve their infrastructures to become more environmentally friendly, increase the quality of life for their residents, and cut costs at the same time.**

Source: SEIMENS, Sustainable Cities, USA.

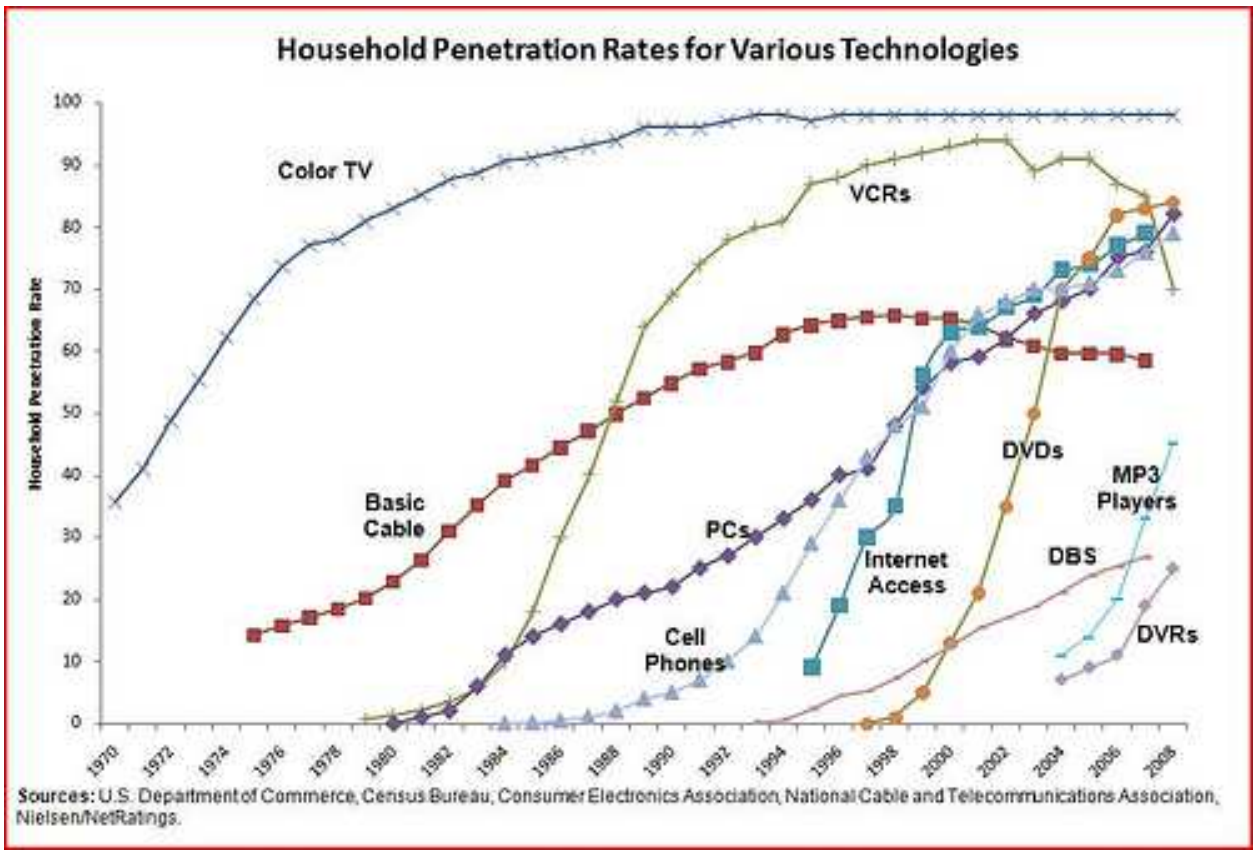
Challenge 2: Information-Centric Systems

Stages in a nation's economic evolution (Adapted from Tien, 2003).

Characteristics	Stage 1 Mechanical Era	Stage 2 Electrical Era	Stage 3 Information Era
Economic Focus	Agriculture/Mining	Manufacturing	Services
Productivity Focus	Farming	Factory	Information
Underlying Technologies	Mechanical Tools	Electromechanical	Information
Product Lifecycle	Decades	Years	Months
Human Contribution	Muscle Power	Muscle/Brain Power	Brain Power
Living Standard	Subsistence	Quality of Goods	Quality of Life
Geographical Impact	Family/Locale	Regional/National	Global
Onset in the U.S.	Late 1700s.	Late 1800s.	Late 1900s.

Challenge 2: Information-Centric Systems

Accelerating pace of technology innovation



Challenge 2: Information-Centric Systems

We now have the ability to measure, sense, and see the exact condition of almost everything (IBM, 2009):

1. **More Instrumented.**

   By the end of 2010 there will be 1 billion trasistors per human and 30 billion RFID (radio frequency id) tags;

2. **More Interconnected.**

   Due to transformational advances in (wireless) communications technology, people, systems and objects can communicate and interact with each other in entirely new ways. Consider:

   We are heading toward one trillion connected objects (Internet of Things).

3. **More Intelligent.**

   More intelligent behavior means an ability to respond to changes quickly, accurately and securely, predicting and optimizing for future events.

Challenge 2: Information-Centric Systems

Need for Systems Thinking in Computer Science

Electronic components (hardware and software) are now ...

> **... hidden in a wide range of devices and that in 2008, the average person used 230 embedded chips every day.**

Examples:

- 80 chips in home appliances, 40 chips in at work, 70 chips in automobiles, and 40 chips in portable devices.

Embedded computer systems need to be designed with their own requirements:

- Reactivity: System response need to occur within a known bounded range and delay.
- Autonomy: Systems need to provide continuous service without human intervention.
- Dependability: Systems need to be resilient to attack and hardware/software failures.
- Scaleability: System performance needs to increase linearly with supplied resources.

Challenge 2: Information-Centric Systems

Industrial-Age Systems

Many present-day systems rely on human involvement as a means for sensing and controlling behavior, e.g.,

- Driving a car,
- Traffic controllers at an airport,
- Manual focus of a camera.

Key disadvantages:

- Humans are slow.
- Humans make mistakes.
- They also easily tire.

Challenge 2: Information-Centric Systems

Information-Age Systems

Developed under the premise that advances in

- Computing,
- Sensing, and
- Communications

technologies will allow for

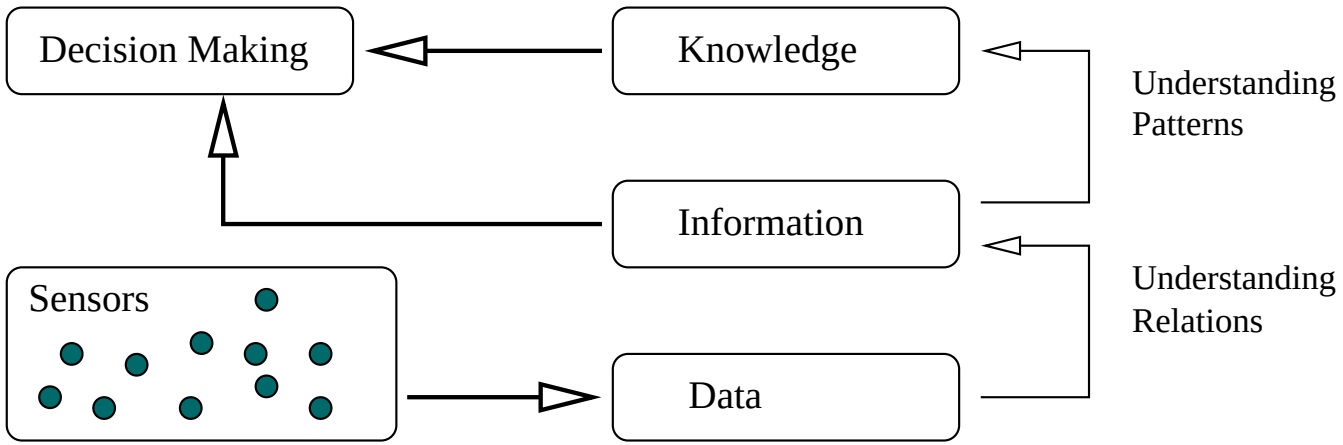
> **... new types of systems where human involvement is replaced (or partially replaced) by automation.**

and where critical constraint values in the design space are relaxed, e.g.,

- Autofocus camera,
- Electronic systems in automobiles and planes,
- Baggage handling systems at airports.

Challenge 2: Information-Centric Systems

Pathway from data to information and knowledge



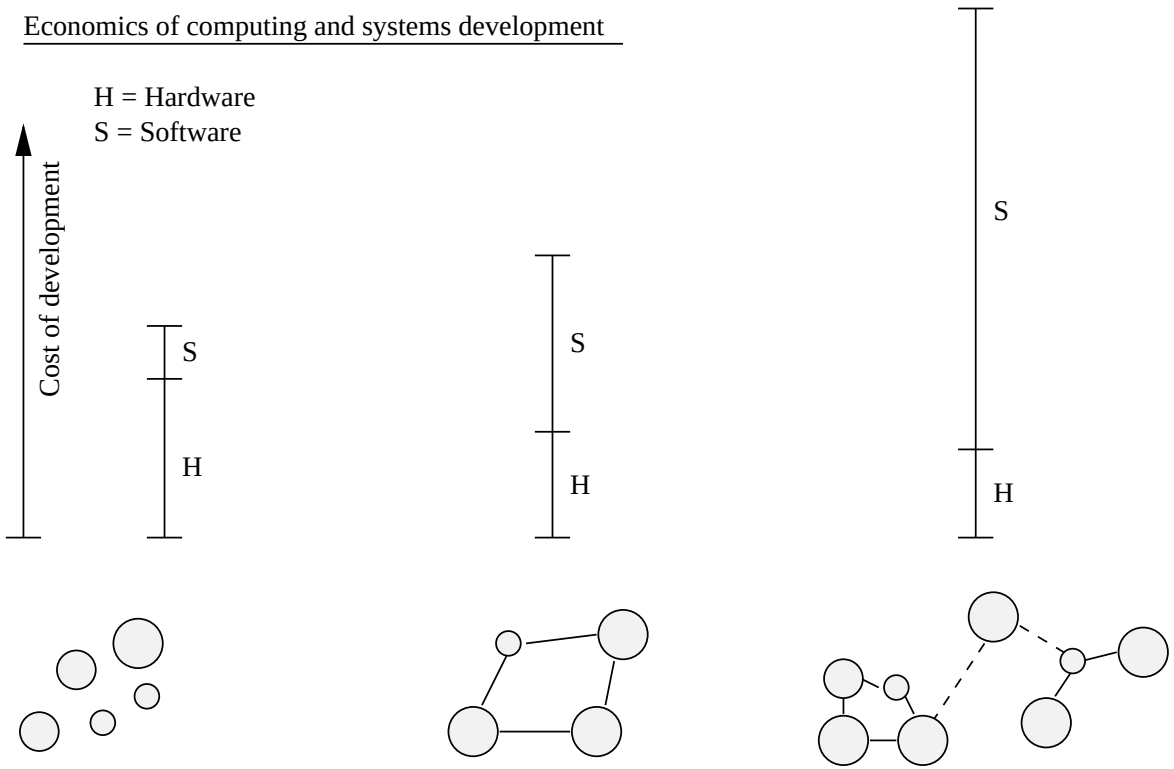
The generated information enables better (i.e., most timely, more accurate) decision making, which in turn, allows for extended functionality and improved performance.

Key Point

Algorithms for understanding relations and patterns will be implemented in software.

Challenge 2: Information-Centric Systems

Rapidly Expanding Expectations ...



Task−oriented programs and modules. Centralized operations

1970's and early 1980s.

Integrated systems and services. Distributed operations.

Early 1990s

Integrated systems and services. Dynamic and mobile distributed operations.

Mid 1990s − today

Challenge 2: Information-Centric Systems

History tells us that it takes about a decade for significant advances in computing capability to occur ...

Capability	1970s	1980s	1990s
Users	Specialists	Individuals	Groups of people
Usage	Numerical computations	Desktop computing	E-mail, web, file transfer.
Interaction	Type at keyboard	Graphical screen and mouse	audio/voice.
Languages	Fortran	C, C++, MATLAB	HTML, Java.

Table 1: Decade-long stages in the evolution of computing focus and capability.

In the 1990s, mainstream computing capability expanded to take advantage of networking.

Challenge 2: Information-Centric Systems

New Computing Infrastructure → New Languages

Capability	2000-present	2020-2030
Users	Groups of people, sensors and computers.	Integration of the cyber and physical worlds.
Usage	Mobile computing. Control of physical systems. Social net-working.	Embedded real-time control of physical systems.
Interaction	Touch, multi-touch, proximity.	....
Languages	XML, RDF, OWL, GoLang.	New languages to support time-precise computations.

Table 2: Decade-long stages in the evolution of computing focus and capability.

Challenge 2: Information-Centric Systems

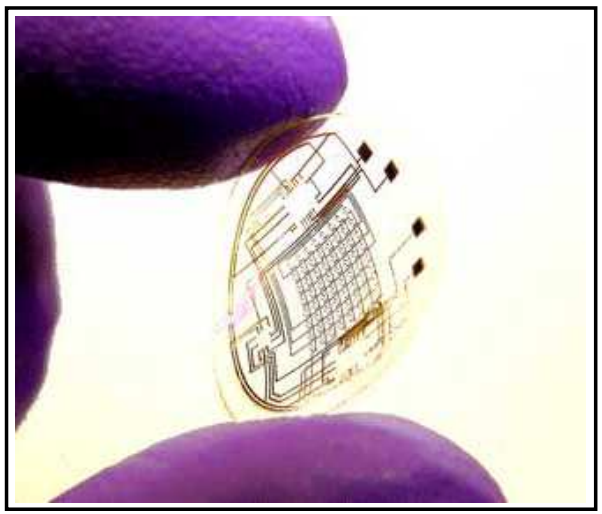
General Idea of CyberPhysical Systems

Embedded computers and networks will ...

... monitor and control the physical processes, usually with feedback loops where computation affects physical processes, and vice versa.

Two Examples

Programmable Contact Lens

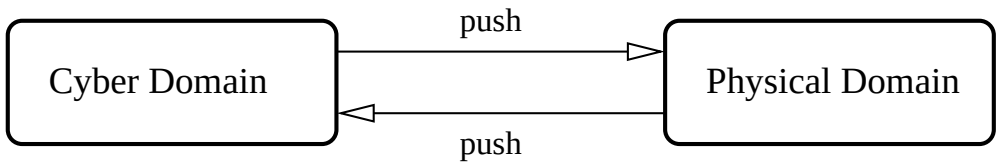


Programmable Windows

Challenge 2: Information-Centric Systems

Interplay of Physical and Cyber Systems

Cyber−Physical Systems Design −− A 10−20 year perspective.



C−P Requirements

Behavior must be robust to unexpected conditions.
C−P system must be adaptable to sub−system level failures.

C−P Structure

Cyber capability in every physical component.
Executable code
Networks of computation
Heterogeneous implementations

Spatial and network abstractions
−− physical spaces
−− physical and social networks.
−− networks of networks
Sensors and actuators.

C−P Behavior

Dominated by logic
Control, communications
Stringent requirements on timing
Needs to be fault tolerant

Physics from multiple domains.
Combined logic and differential equations.
Not entirely predictable.
Multiple spatial− and temporal− resolutions.

Challenge 2: Information-Centric Systems

Many modern engineering systems are a combination of physical and computational/software systems.

Physical Systems

1. Design success corresponds to notions of robustness and reliability.
2. Behavior is constrained by conservation laws (e.g., conservation of mass, conservation of momentum, conservation of energy, etc..).
3. Behavior often described by families of differential equations. (May involve multiple physics).
4. Behavior tends to be continuous – usually there will be warning of imminent failure.
5. Behavior may not be deterministic – this aspect of physical systems leads to the need for safety and reliability requirements.
6. For design purposes, uncertainties in behavior are often handled through the use of safety factors.

Challenge 2: Information-Centric Systems

Software Systems are Fragile

1. Design success corresponds to notions of correctness of functionality and timeliness of computation.
2. Computational systems are discrete and inherently logical. Notions of energy conservation ...etc... and differential equations do not apply.
3. Does not make sense to apply a safety factor. If a computational strategy is logically incorrect, then “saying it louder” will not fix anything.
4. A small logical error can result in a system-wide failure.

Software Systems are Flexible

The main benefit of software is that ...

... functionality can be programmed and then re-programmed at a later date.

Challenge 3: Importance of Systems Integration

Definition

System integration is the process of ...

... deliberate assembly of the parts of a system into a functioning whole.

The assembly process will include (this is not an exhaustive list):

- Physical assembly of the parts, providing consumables to make the parts work, connecting electronics to power sources, uploading and test of software.

Industrial-Age Approaches to System Integration

Since the 1970s established approaches to systems engineering have assumed that ...

... complications associated with a systems development can be kept in check through strategies of decomposition and separation of design concerns.

Challenge 3: Importance of Systems Integration

Amazing advances in network communications and software have allowed for new approaches to system development and expectations of system performance.

Process View of Systems Integration

Modern engineering systems are now ...

> **... designed by geographically distributed teams, and assembled from parts obtained from geographically distributed suppliers.**

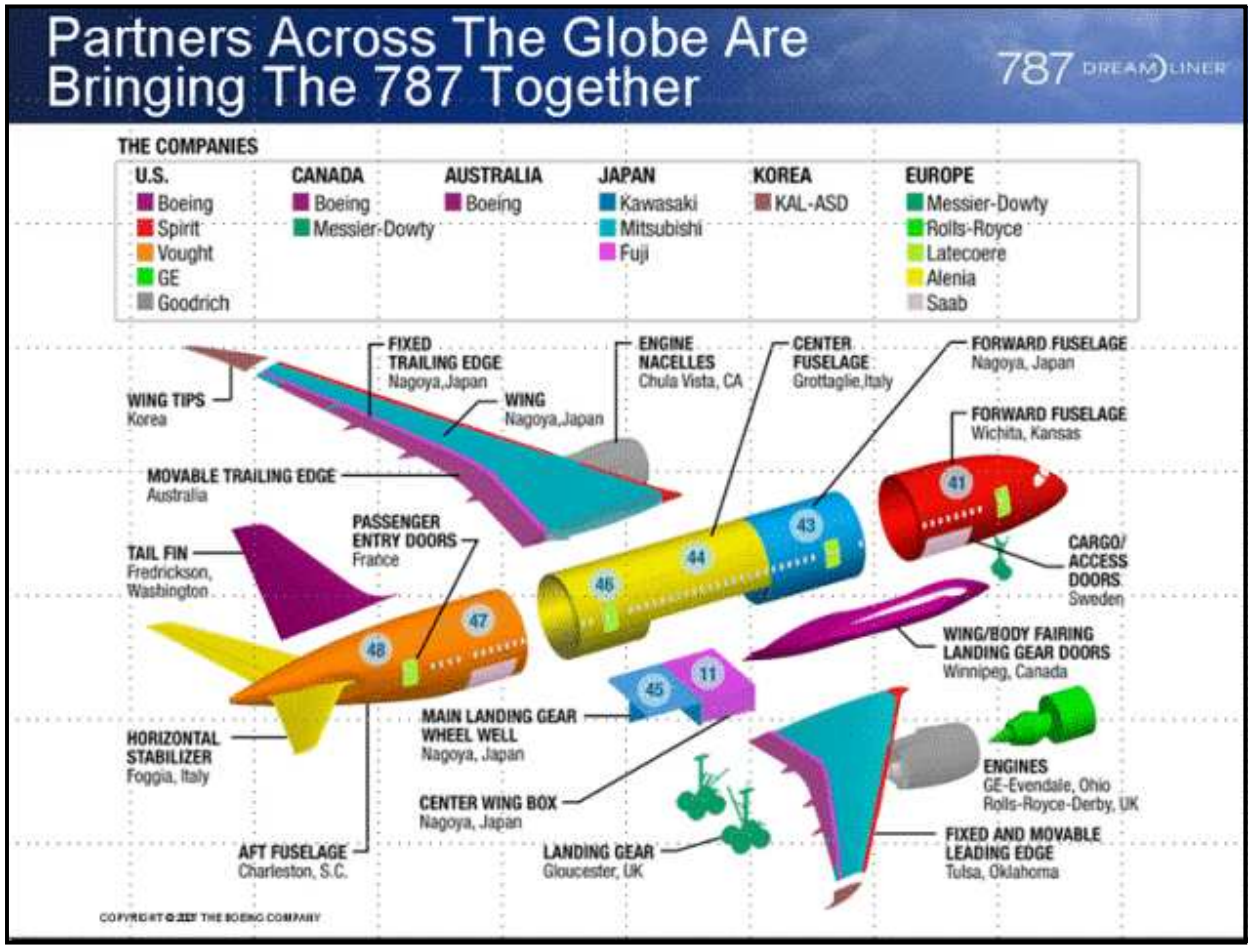
Product View of Systems Integration

These advances allow for ...

> **... development of systems where management of system functionality and improvements to performance are achieved through use of software.**

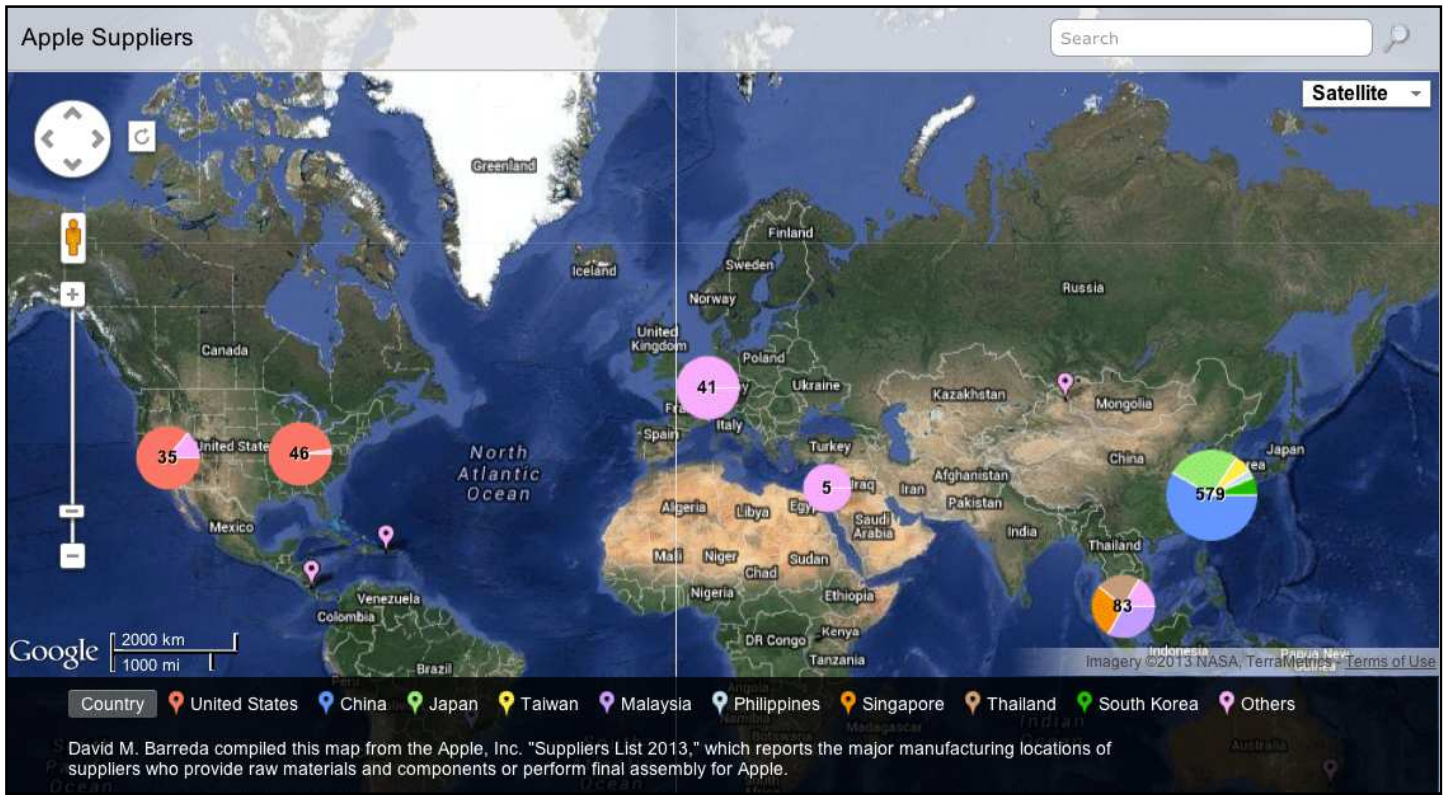
Challenge 3: Importance of Systems Integration

Suppliers for the Boeing 787 Aircraft



Challenge 3: Importance of Systems Integration

Google Maps display of suppliers for Apple products in 2013

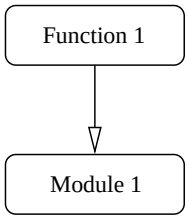


Challenge 3: Importance of Systems Integration

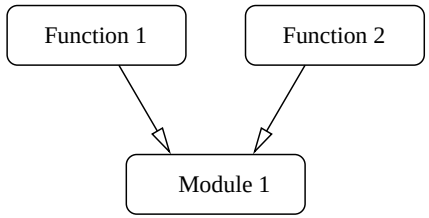
Modular Development of Systems

A modular architecture has well-defined, standardized, and decoupled interfaces which collectively allow for design changes to be made to one module, without generally requiring a change to other modules.

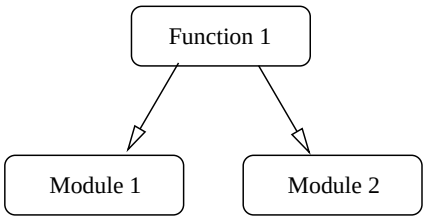
Four types of product architecture:



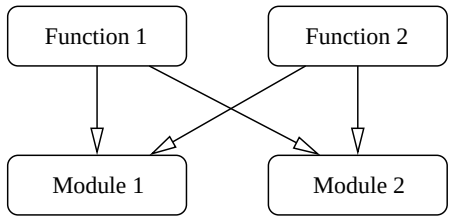
Modular Design



Function Sharing



Function Distribution



Integrated Design

Challenge 2: Systems Integration

Integration promotes teamwork.

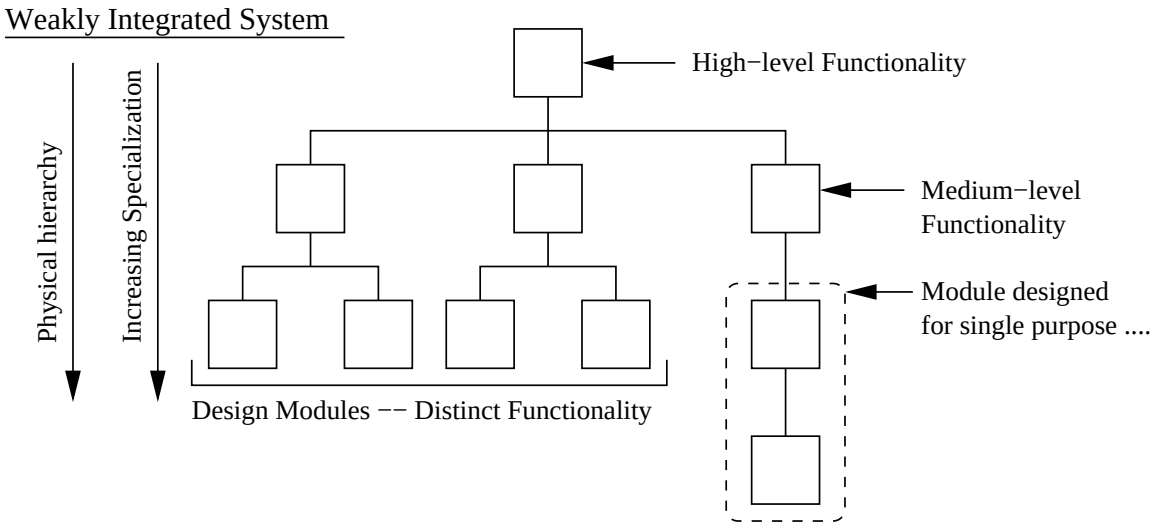
A system will function better when the sub-systems work together as a team rather than independently.

However, integration requires:

... concurrent consideration of each sub-systems functions and performance, together with models of connection and communication among sub-systems.

Challenge 2: Systems Integration

Nodal connectivity and functional influence in a weakly-integrated system

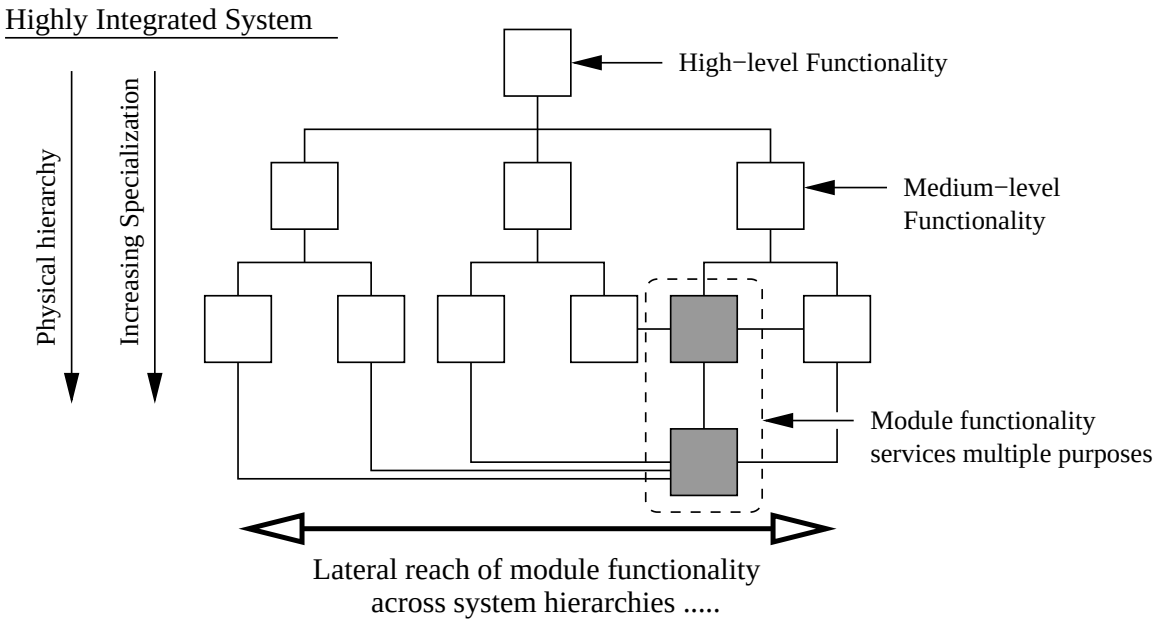


Key characteristics:

1. Collections of parts having interactions that are well understood.
2. Complexity is manifests itself through layers of progressively complicated detail, which tends to be discipline specific.

Challenge 2: Systems Integration

Nodal connectivity and functional influence in a highly-integrated system



Key characteristics:

1. Lateral influences dominate hierarchical relationships.
2. A change at almost any level may have system-wide consequences.
3. Impacts of decisions are less predictable and difficult to bound.

Challenge 3: Need for Error-Free Software

What computers and computer software bring to the table is an ability to design and efficiently implement systems that have

... wider ranges of functionality, better performance, and improved economics.

Complex engineering systems are becoming increasing reliant on:

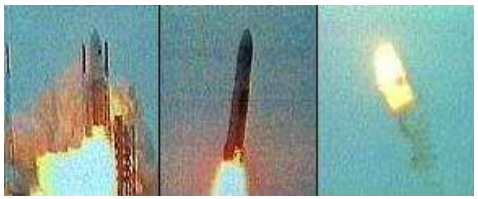
... software and communications technologies that must work correctly and with no errors.

Satisfying this criterion is complicated by the fact that...

... a small fault in the software implementation can trigger (or result in) system-level failures that are very costly and, sometimes, even catastrophic.

Challenge 3: Need for Error-Free Software

Case Study 1: Explosion of Ariane 5, 1996.



- The Ariane 5 rocket exploded on its maiden flight in June 1996 because the navigation package was inherited from the Ariane 4 without proper testing.
- Shortly after launch, an attempt to convert a 64-bit floating-point number into a 16-bit integer generated an overflow.
- The error was caught, but the code that caught it elected to shut down the subsystem. The rocket veered off course and exploded.

Challenge 3: Need for Error-Free Software

Case Study 2: Denver Airport Baggage Handling System.

- **1995.** The Denver airport baggage handling system was so complex (involving 26 miles of conveyors and 300 computers) that the development overrun prevented the airport from opening on time.

  Fixing the incredibly buggy system required an additional 50 percent of the original budget - nearly $200m.

- **2005.** Despite years of tweaking, it never ran reliably. Airport managers pull the plug, reverting to traditionally loaded baggage carts with human drivers (Jackson, Scientific American, June 2006).

Challenge 4: Agility in System Capability

Definition

For systems engineering purposes an agile system needs to ...

> **... respond quickly and effectively to rapid change, even in uncertain and unpredictable business environments.**

A slightly different defintion – an ideal agile system will ...

> **... proactively sense changes as opposed to simply being flexible in reaction to change.**

Implementation

Agility translates to implementations that strategically focus on:

- Measurement-directed sensing,
- Learning, and
- Taking appropriate actions.

Challenge 4: Agility in System Development

Systems Engineering with Pre-defined Plans of Development

Pre-defined plans of development (e.g., a Waterfall Model) ...

... provide the discipline to keep development activities predictable and on track.

The project participants know what's expected and when.

During the past 3-4 decades this approach to system development has served many industry sectors (e.g., aerospace) well.

Key Problem

As systems are required to adapt to change more quickly (i.e., with progressively shorter development times), ...

... pre-defined plans hinder progress through their lack of flexibility ...

and, as such, should be replaced by something better.

Challenge 4: Agility in System Development

Software Engineering Community

Agility in software engineering is facilitated by:

1. Freedom from the physical constraints normally associated with hardware,
2. Well developed technology for compiling high-level solutions procedures into executable code, and
3. Well developed technology for distributing software over networks and installing updates on target machines.

Together these three factors allow for environments where software can be programmed and then re-programmed and distributed as needed.

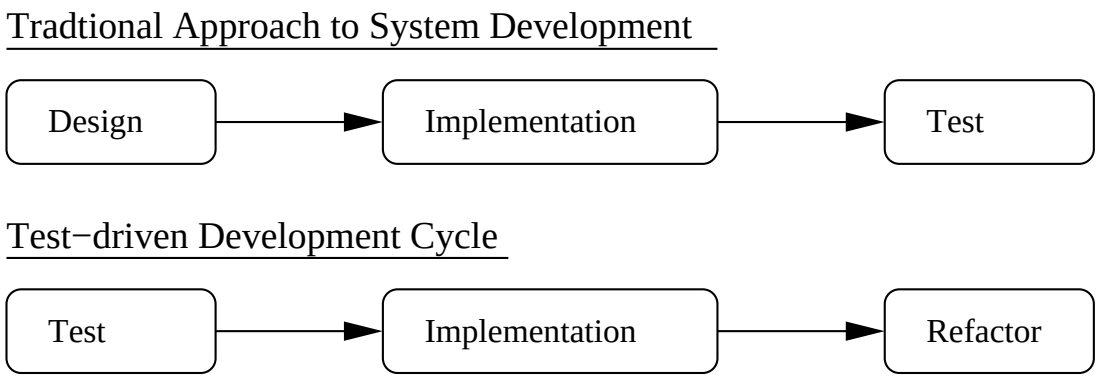
Still, it is well known that ...

> **... unless support for change (and extension) is explicity built into the system, then the system will probably not adapt as needed.**

Challenge 4: Agility in System Development

Test-Driven Software Development

Comparison of traditional and test-driven development cycles



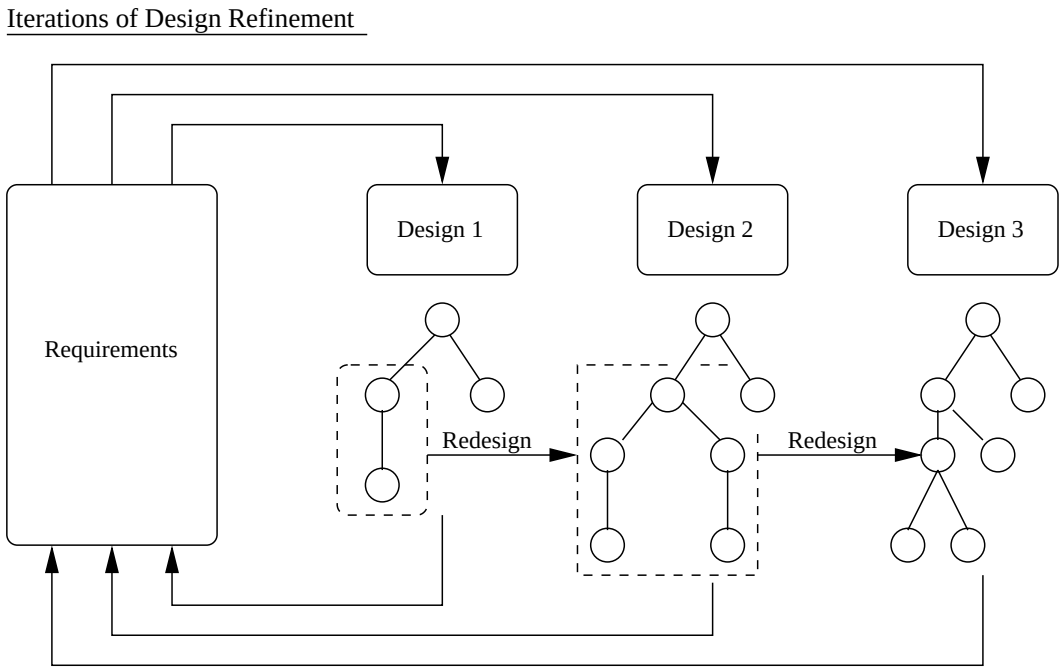
Workflows for test-driven development are based on a very simple tenet:

... you only ever write code to fix failing tests.

Challenge 4: Agility in System Development

Agility in Systems Engineering

Incremental refinement of a design over several iterations of development.



Requirements change for a variety of reasons: economics and environment.

Designs also change to fix mistakes, incorporate new technologies, and to account for changing capability.

Challenge 4: Agility in System Development

Agility in Systems Engineering

Unlike the software world,

> **... the systems engineering world needs to deal with stringent physical constraints, plus software, plus mixtures of hardware of software that could interchangable.**

This forces a focus on

> **... modular approaches to system implementation and the design of system interfaces as a first class entity.**

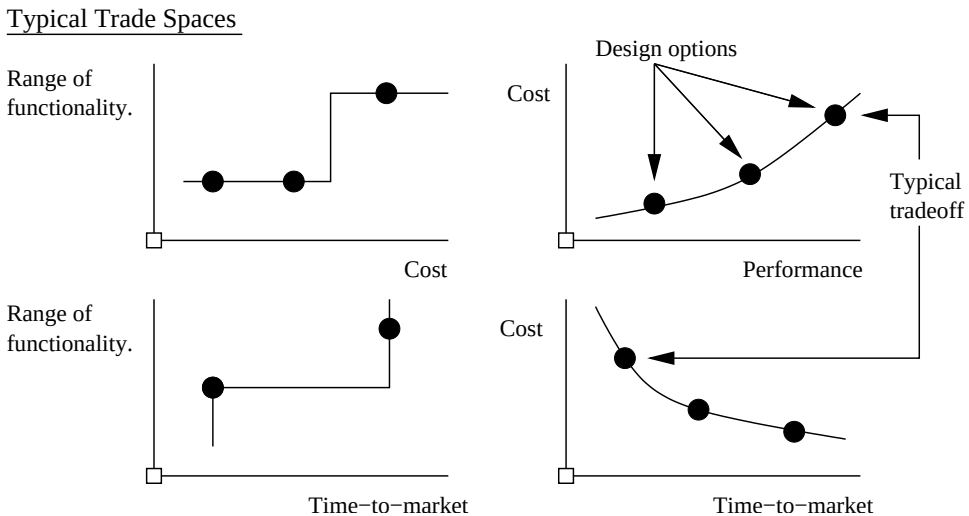
It also suggests that design developments should be persistent, meaning that step-by-step procedures for creating a design should be completely reversable.

Designers should be given the tools to recover from mistakes and/or quickly revise a design to meet a new set of requirements.

Challenge 5: Formal Support for Trade Studies

The purpose of a trade study is to ...

... examine the relative value and sensitivity of attributes associated with the design's measure of effectiveness.



This information is then used to guide decision making relating to the selection and treatment of design alternatives.

Challenge 5: Formal Support for Trade Studies

For the development of systems that are new and innovative, and/or extensible and/or highly adaptive,

> **... systems engineers may have neither the experience nor insight needed to satisfy the design constraints and balance the design objectives.**

Potential complications include:

> **... a lack of clarity on which parts of a design are best suited to participate in trade off studies.**

Challenge

Systems engineers need:

1. Better ways of identifying the trade spaces that are relevant to a new design situation, and
2. Formal approaches to trade-off analysis for systems that are either extensible and/or highly adaptive.

Case Studies

Case Study 1: Complexity in Aerospace Systems

During the past three decades aerospace systems have seen

... increased use of electrical systems to achieve functionality.

Example. F-16 and F-35 Military Jets

Fourth generation F-16 (production began 1974). Fifth generation F-35 (production began in 2006).

F−16

F−35

Case Study 1: Complexity in Aerospace Systems

Summary: F-16 System:

- 15 subsystems,
- O(10^3) interfaces,
- Less than 40% of the functions managed by software.

Summary: F-35 System:

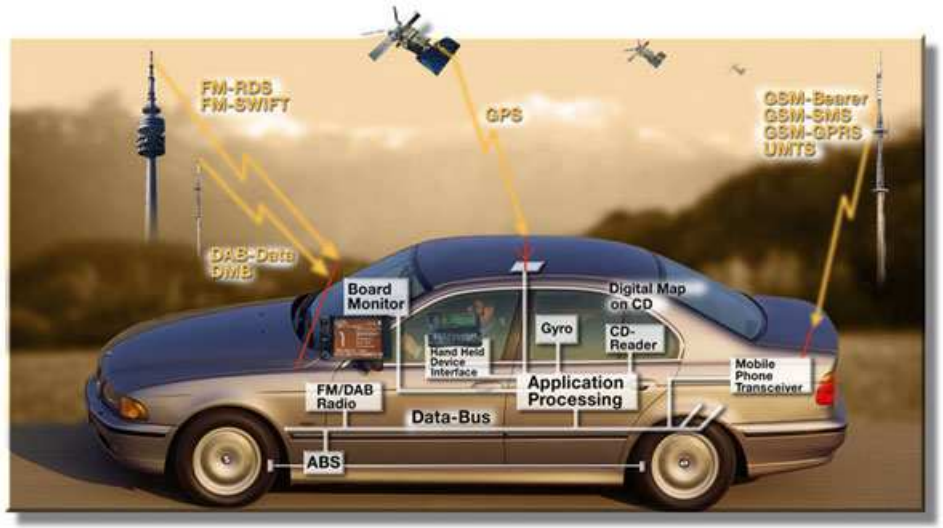
The F-35 offers 3-8 times the operational capability of the F-16 and F-18.

This innovation has come at the cost of increased technical complexity.

- 130 subsystems,
- O(10^5) interfaces;
- 90% of its functions are managed by software.

Case Study 2: Automobile Electronics

Electronics and Communications in a Modern Car.



In a modern automobile, the electronics and communication systems now account for 30% of the overall cost (W. Reitzle, BMW, 2000).

Source: A.S. Sangiovanni-Vincentelli, EE 249, UC Berkeley, Fall 2002.

Case Study 2: Automobile Electronics

Key points:

- The electronic systems in modern cars and trucks are ...

  ... packed with up to 100 million lines of computer code.

  You can think of a modern automobile as a network of (30) computers on wheels.

- The software in each unit is also made to work with other units. So,

  ... when a driver pushes a button on a key fob to unlock the doors, a module in the trunk might rouse separate computers to unlock all four doors.

- Throttle-by-wire technology (electronic throttle control) replaces cables and/or mechanical connections.

  Among other things, throttle by wire makes it easier for carmakers to add advanced cruise and traction control features.

- Electronic systems are engineered to protect against the kind of false signals or electronic interference that could cause sudden acceleration.

Case Study 3. America's Infrastructure Crisis

The Problem

In America, ...

... civil infrastructure is not considered to be a national priority.

A few key statistics:

- From 1950-1970, the US devoted 3% of its gross domestic product (GDP) to infrastructure spending.
- Since 1980, spending on infrastructure has been cut to 2% of GDP.
- China spends 5% of GDP on infrastructure.
- India spends 9% of GDP on infrastructure.

Case Study 3. America's Infrastructure Crisis

Key Problems

Two key problems:

- Much of America's infrastructure was built post World War II – it's now 50-60 years old, and being attacked by decay and neglect.
- The US Population is still growing! This puts additional demands on infrastructure.

Criticism

Quote from W.P. Henry, former president of ASCE:

Our infrastructure is in crisis mode ...

> **... how many more people must die needlessly because we do not take proper care of our infrastructure?**

Case Study 3. America's Infrastructure Crisis

Poster Child: Collapse of the Minneapolis Bridge over Interstate 35W.

The 40-year old steel deck truss crossing had been considered ...

> **... structurally deficient since 1990, but engineers with the Minnesota Department of Transportation had not believed the bridge to be in danger of imminent collapse.**

Thirteen commuters were killed and more than 100 were injured on August 1, 2007.

Case Study 3. America's Infrastructure Crisis

The Infrastructure Crisis extends beyond Bridges

Key quotes from ASCE's Infrastructure Crisis Report (Reid, 2008):

- Without "significant infrastructure investment" **aviation delays** are expected to cost this US economy $170 billion between 2000 and 2012.
- Improving the physical condition and service of the nation's **mass transit** systems will require between $30 billion and $45 billion a years, approximately 130 to 240 percent more than the total investment for 2004.
- More than 3,200 **dams** are currently classisfied as "unsafe" – meaning that their deficiencies leave them more susceptible to failure – a figure that has increase 80 percent since 1998.
- It took Congress eight years to pass the **water resources** bill, and then it was vetoed by President Bush!

Systems Management Challenges

Most important factors contributing to project failure.

Factor	Contribution
Incomplete requirements (*)	13.1%
Lack of User Involvement(*)	12.4%
Lack of resources	10.6%
Unrealistic expectations(*)	9.9%
Lack of executive support	9.3%
Changing requirements and specifications(*)	8.7%
Lack of planning	8.1%

Source: Surveys conducted by Standish Group (1995 and 1996).

Systems Management Challenges

Most important factors contributing to project success.

Factor	Contribution
User involvement(*)	15.9%
Management support	13.9%
Clear statement of requirements(*)	13.0%
Proper planning	9.6%
Realistic expectations(*)	8.2%
Smaller milestones	7.7%
Competent staff	7.2%
Ownership(*)	5.3%

Source: Surveys conducted by Standish Group (1995 and 1996).

Model-Based Systems Engineering

Model-Based Systems Engineering

Goals

Model-based systems engineering (MBSE) development is an approach to systems-level development in which

> **... the focus and primary artifacts of development are models (as opposed to documents)**.

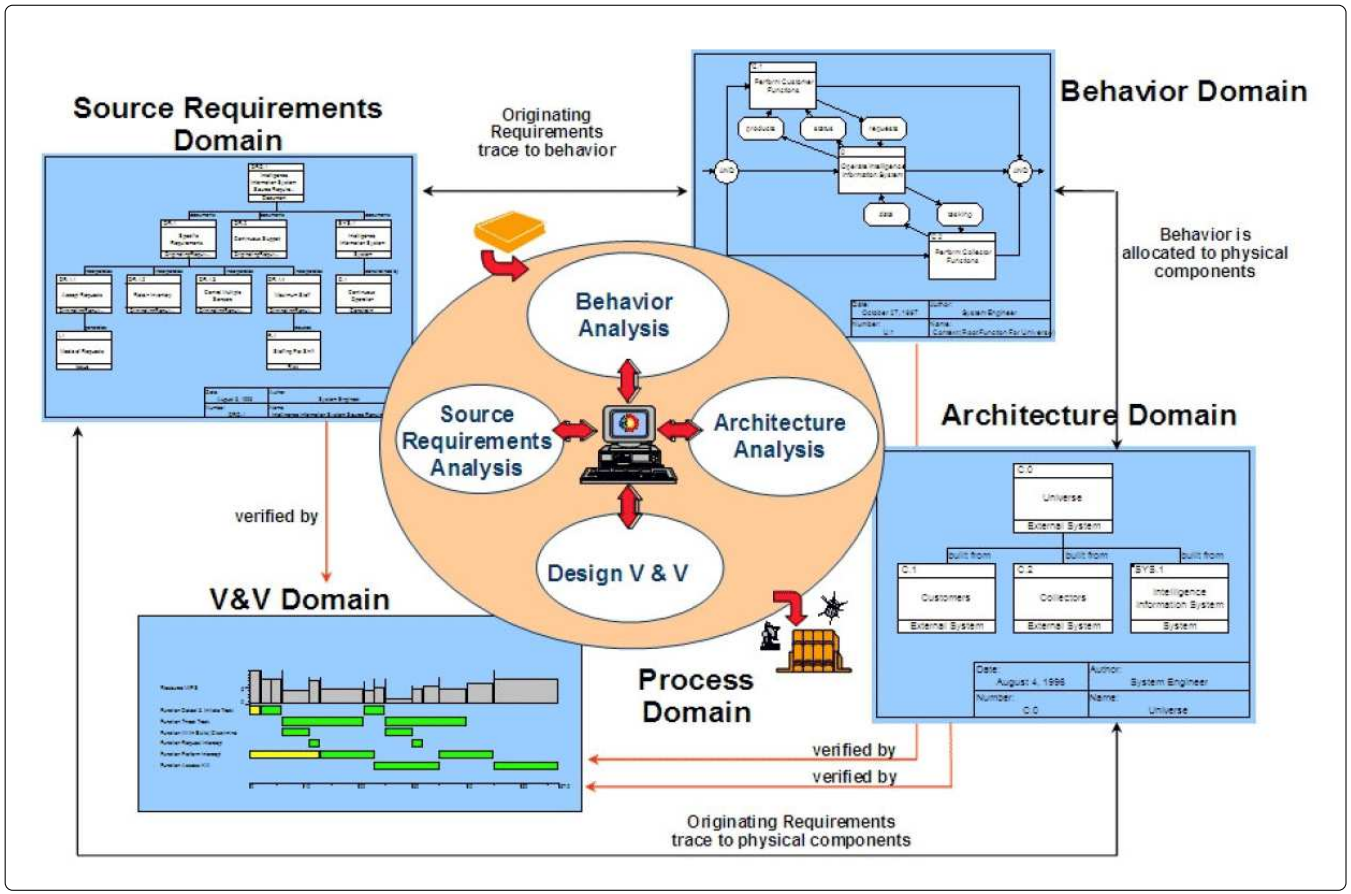
Approach and Benefits

MBSE procedures provide a formal basis for:

- Closing the gap between **what is needed** and **how the system will work**
- Assisting in the management of complex systems.
- Early and formal approaches to system validation and verification.

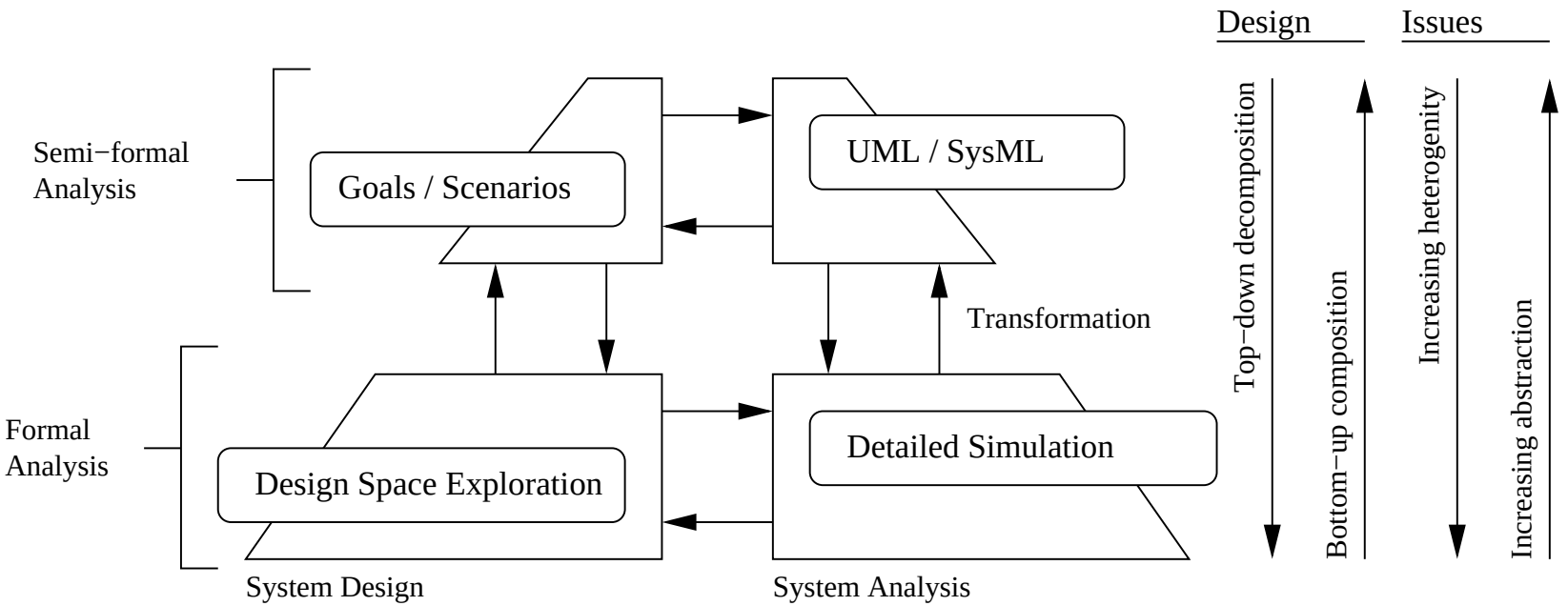
Model-Based Systems Engineering

Model-based systems engineering process at Vitech



Model-Based Systems Engineering

Multi-Level Approach Model-based Systems Engineering



Model-Based Systems Engineering

Orchestration of Good Design Solutions

1. **Semi-Formal Models**

   To allow for the efficient representation of ideas (e.g., goals and scenarios), representations for preliminary/tentative design need to be based on semi-formal models (e.g, UML and SysML).

2. **Formal Models**

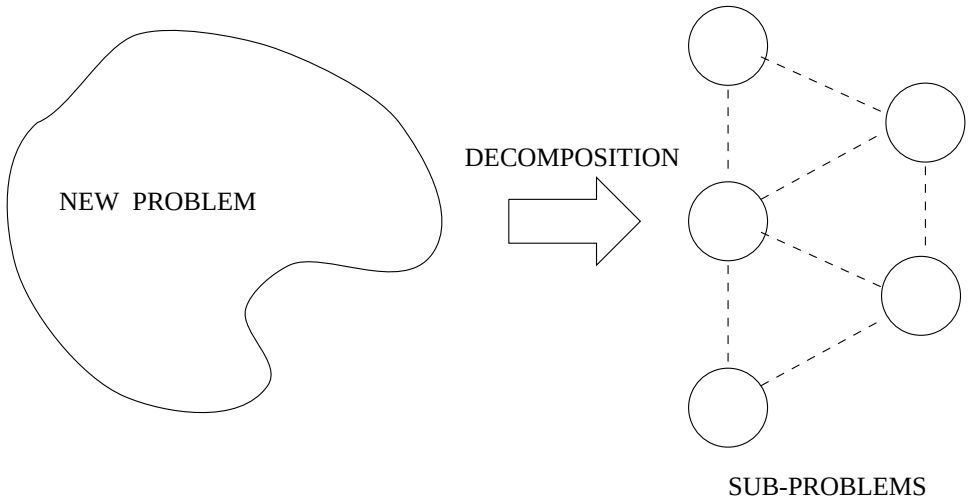
   To help prevent serious flaws in detailed design and operation, design representations and validation/verification procedures need to be based on formal languages having precise semantics.

3. **Abstraction**

   Abstraction mechanisms eliminate details that are of no importance when evaluating system functionality, system performance, and/or checking that a design satisfies a particular property.

Orchestration of Good Design Solutions

4. Decomposition

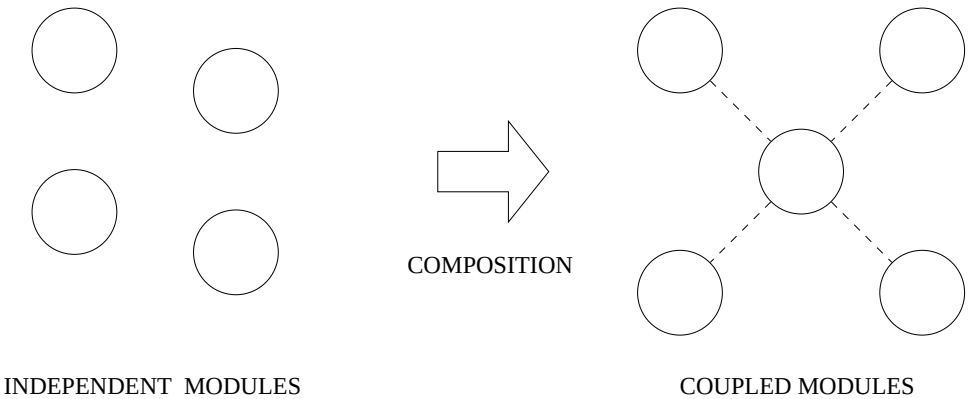


Model-Based Systems Engineering

Orchestration of Good Design Solutions

5. Composition

Composition is the process of systematically assembling a system from subsystems and components.



We seek, in particular, methods that allow for ...

... the systematic assembly of behavior models for complex systems from behavior models for simpler systems and components.

Established Strategies of Development

Simplify Design through Separation of Concerns

Complex systems are often characterized by ...

> **... many components, intertwined network structures, concurrent behaviors, and complicated communications and interactions among subsystems and components.**

To facilitate understanding of these design issues/concerns, we aim to

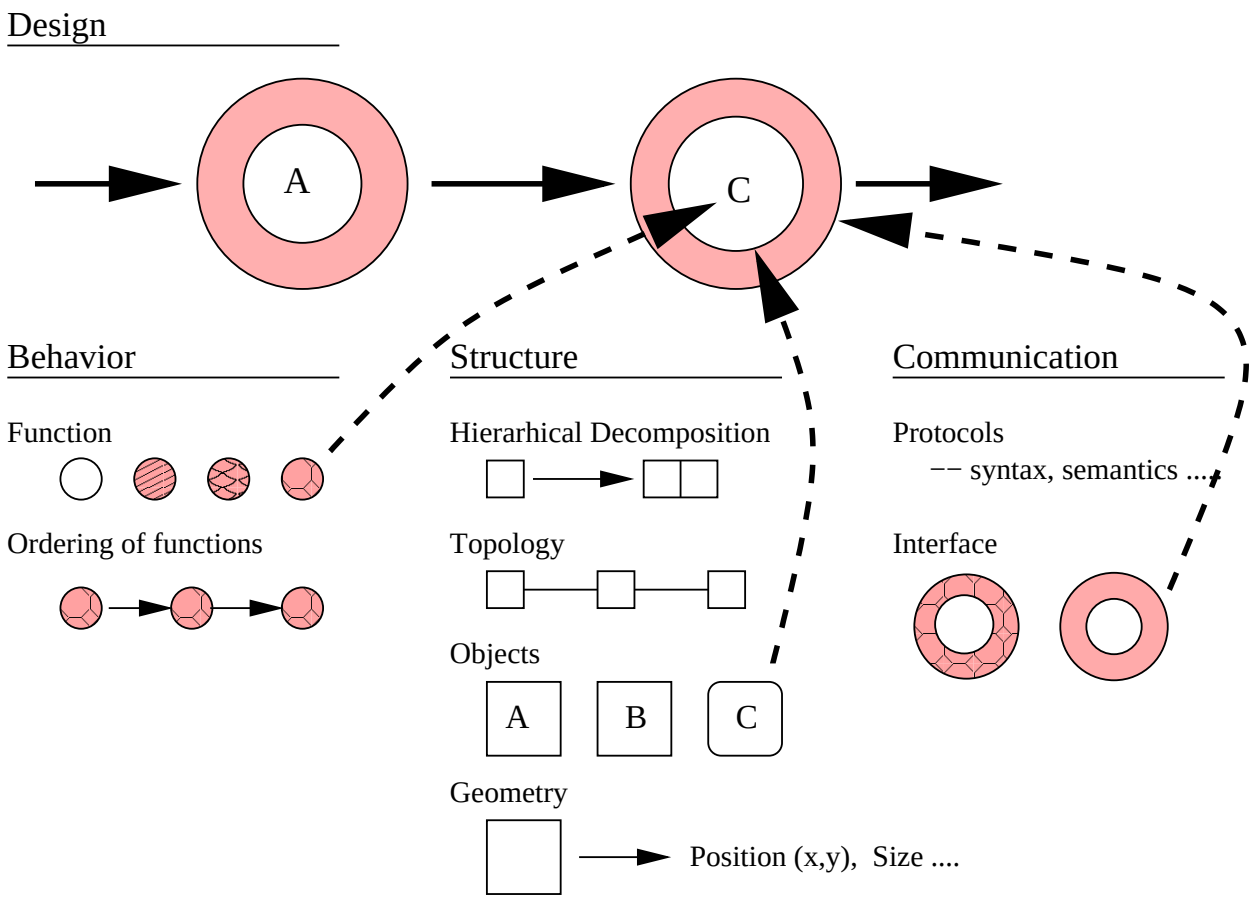
> **... pull a design apart and examine it from perspectives (or ”facets” or viewpoints) that are almost orthogonal, thereby factoring out so-called cross-cutting concerns.**

Achieving (almost) orthogonality of concerns is important because ...

> **... it means we can explore options in one viewpoint (or dimension of design) without affecting other concerns.**

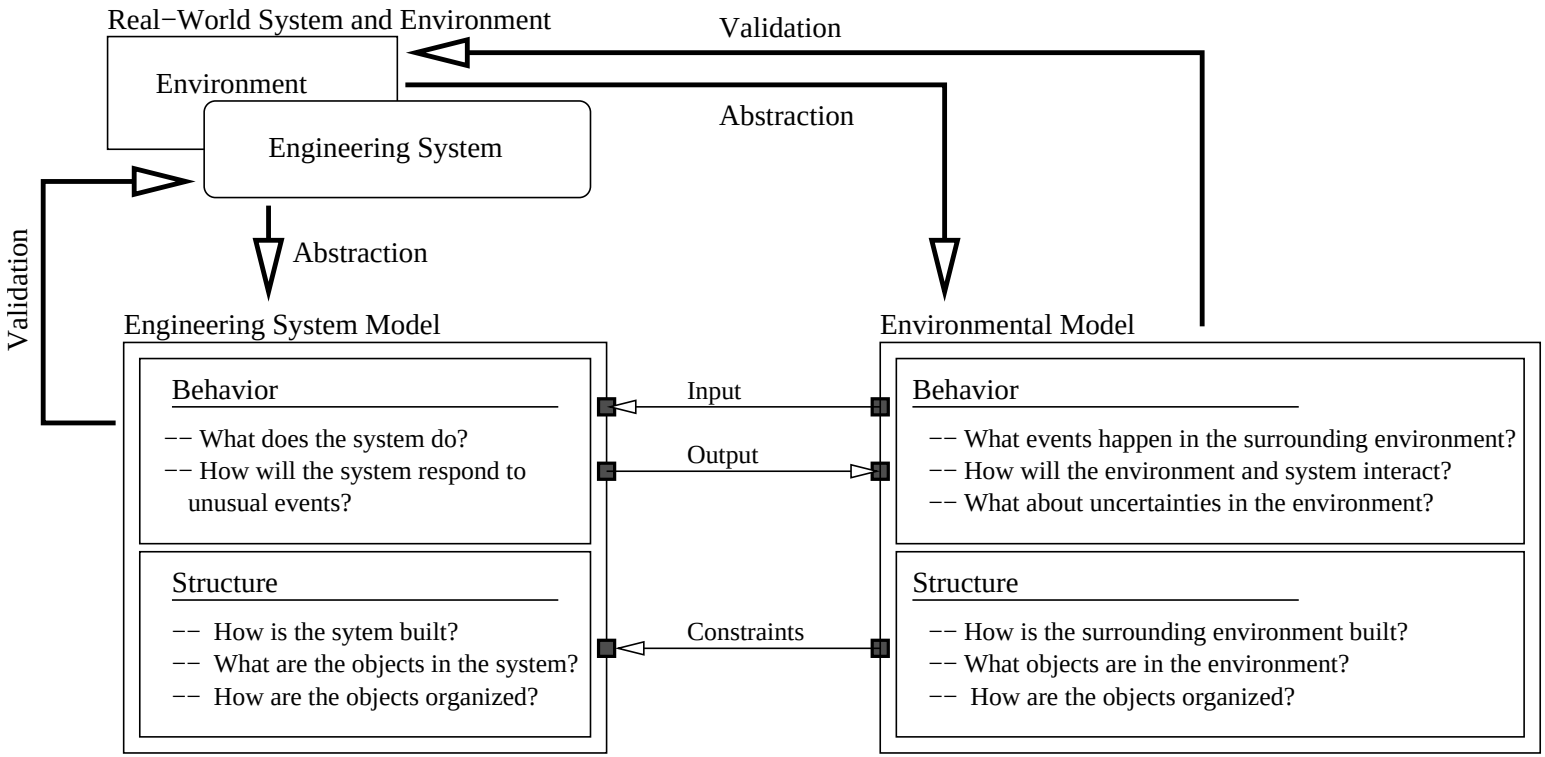
Established Strategies of Development

Example 1. Separation of concerns (e.g., structure, behavior, communication) in simple network.



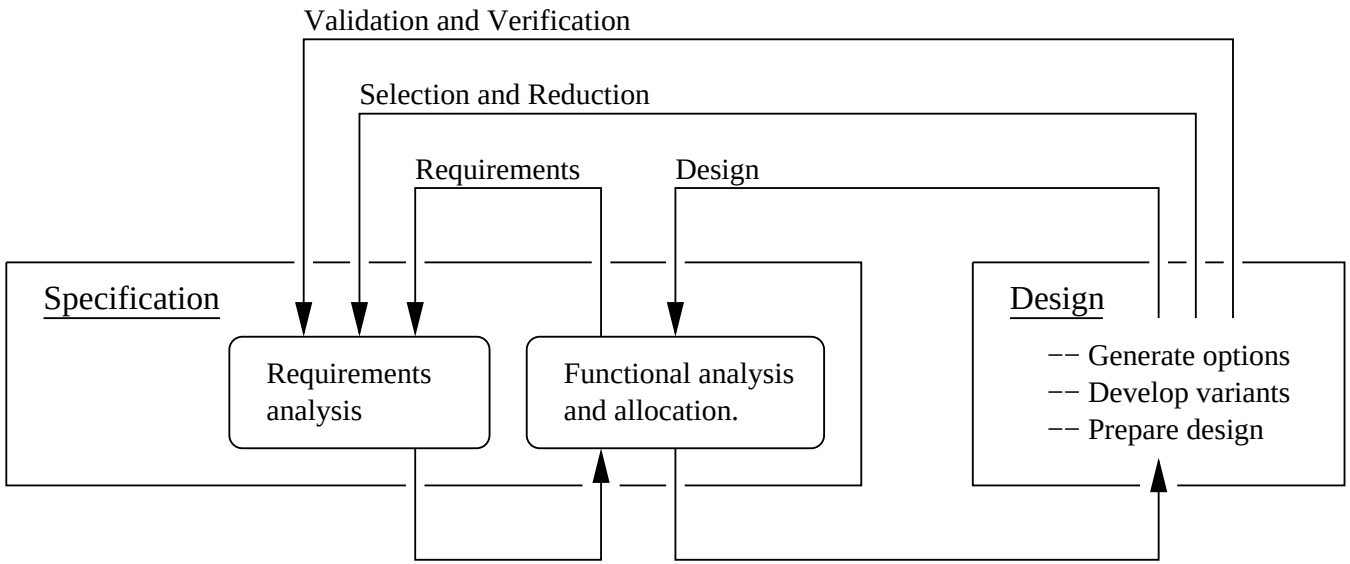
Established Strategies of Development

Example 2. Synthesis of models for engineering system and surrounding environment.



Established Strategies of Development

Example 3. Separation of SE activities/products – requirements, design and validation results.



Established Strategies of Development

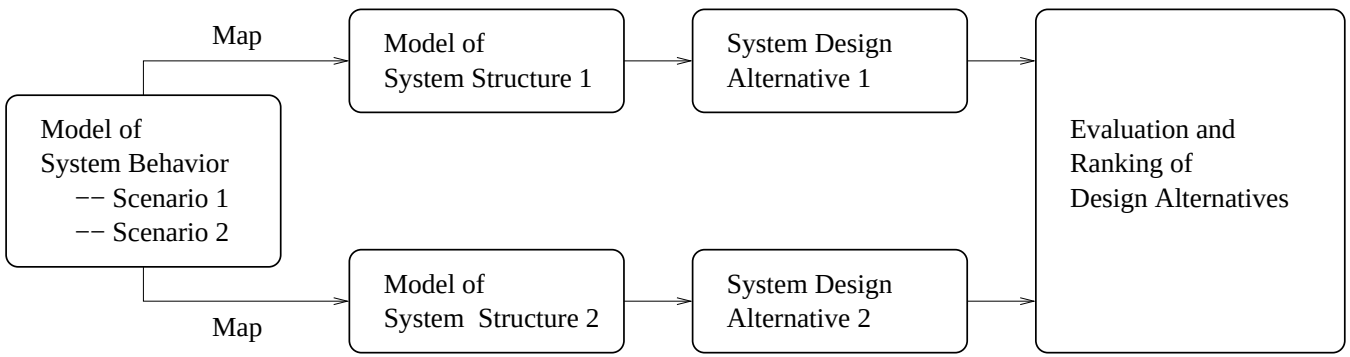
Function before Physical

We promote the description of systems in two orthogonal ways:

- The function that the systems is intended to provide,
- Candidate architectures for realizing the functionality.

Function-Architecture Co-Design

Map models of system behavior onto system structure alternatives.

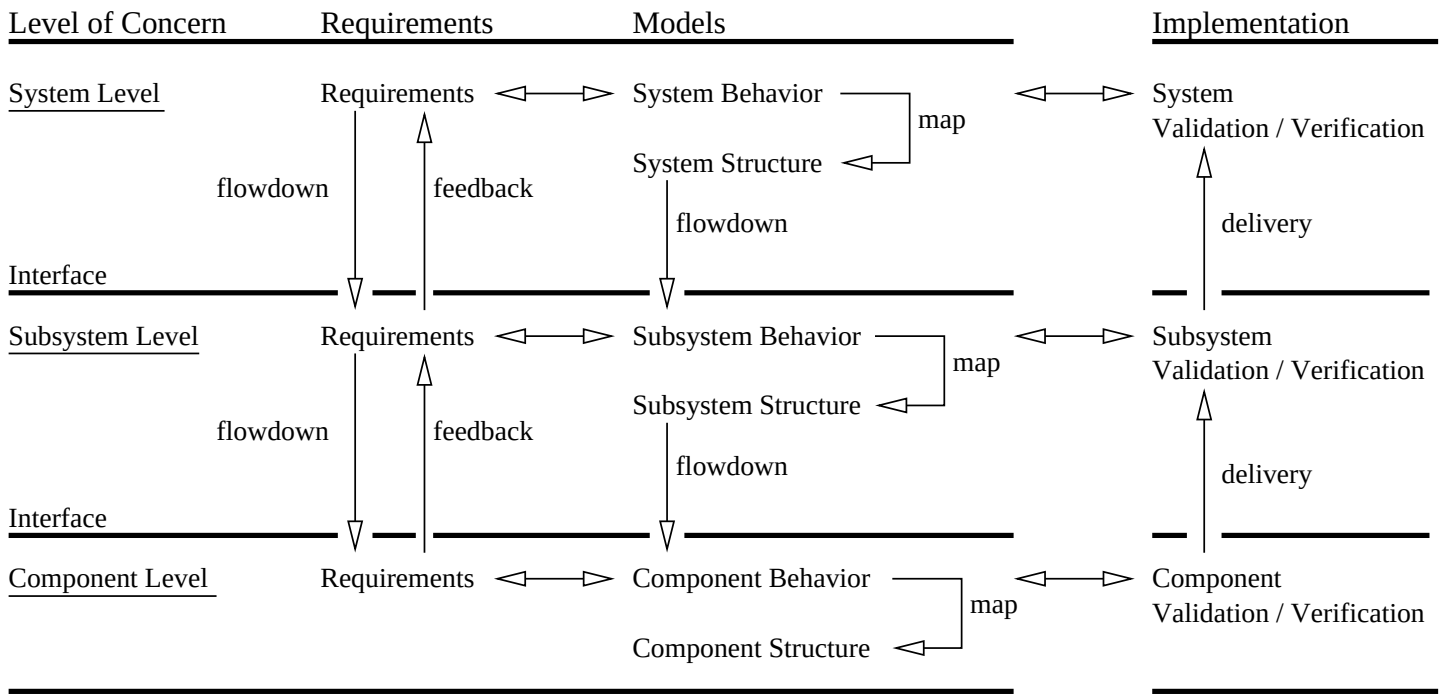


Identify measures of effectiveness. Then evaluate and rank design alternatives.

Established Strategies of Development

Layered Approach to Development

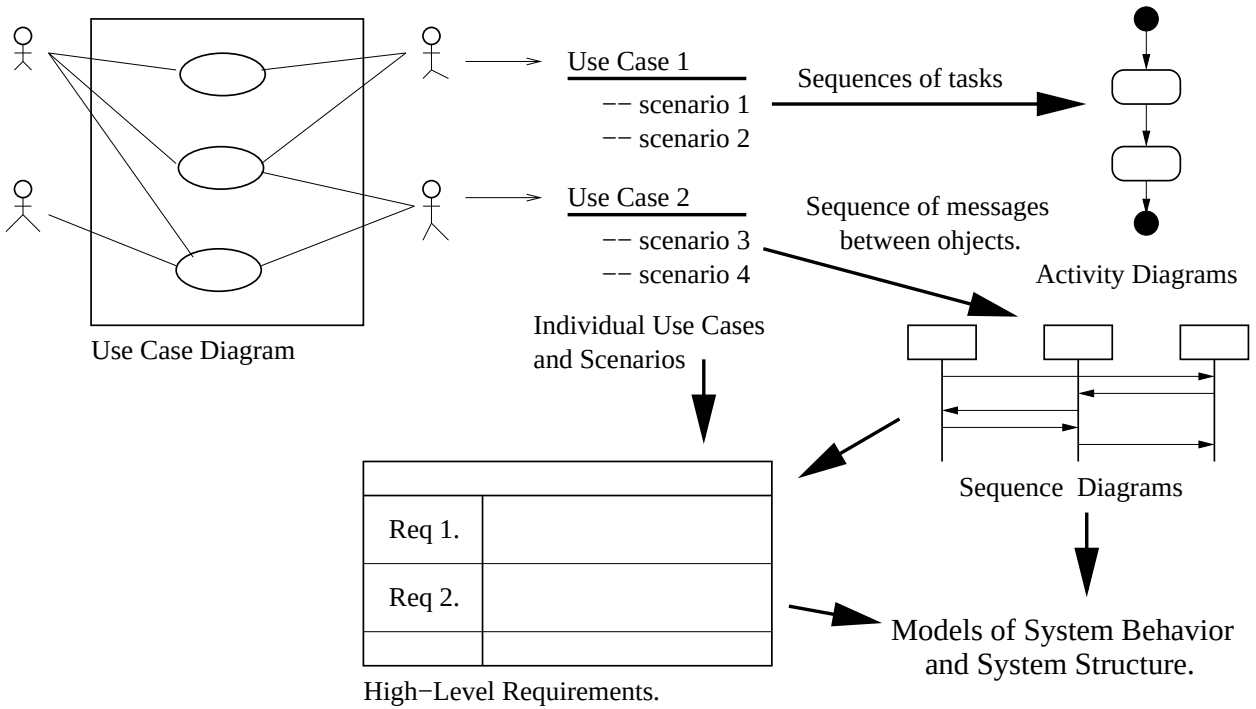
The tenet of “breadth before depth” leads to a layered approach to development.



Looking Back at ENSE 621/ENSE 622

Problem Definition. Development of an Operations Concept.

Pathway from goals and scenarios to simplified models of behavior and requirements.



Looking Back at ENSE 621/ENSE 622

Key Points:

- The functional description dictates what the system must do.

  Here, we employ a combination of use cases (and use case diagrams), textual scenarios, and activity and sequence diagrams to elicit and represent the required system functionality.

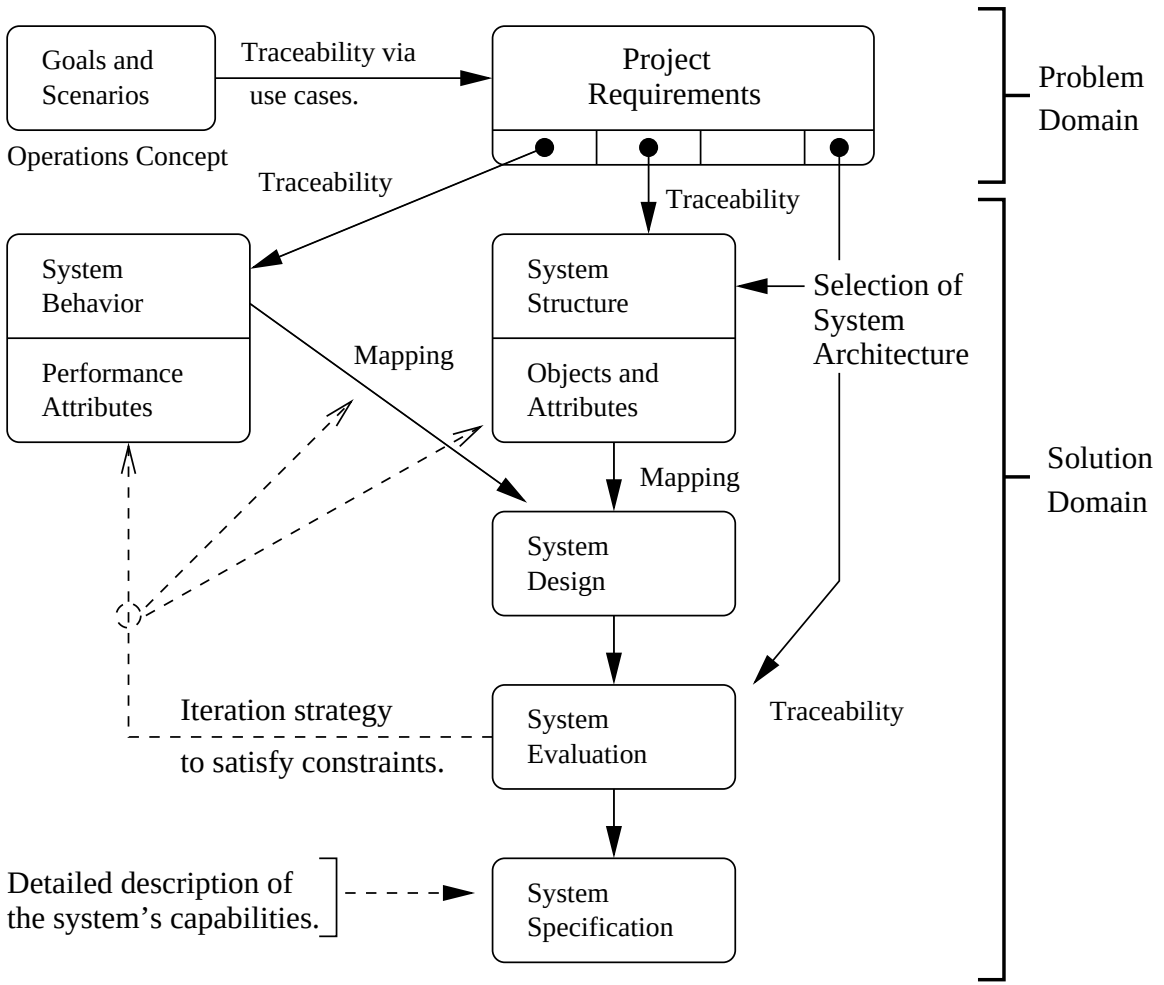
- A complete system description will also include statements on minimum levels of acceptable performance and maximum cost.

  Since a system does not actually exist at this point, these aspects of the problem description will be written as design requirements/constraints.

- Further design requirements/constraints will be obtained from the structure and communication of objects in the models for system functionality (e.g., required system interfaces).

Looking Back at ENSE 621/ENSE 622

Problem Solution. Pathway from Requirements to Models of System Behavior/Structure and System Design



Looking Back at ENSE 621/ENSE 622

Key Points:

- Requirements are organized according to the role they will play in the system-level design.
- Models of behavior specify **what** the system will actually do.
- Models of structure specify **how** the system will accomplish its purpose.
- The nature of each object/subsystem will be captured by its attributes. Attributes includes:
  - The attributes of the physical structure of the design,
  - The attributes of the environmental elements that will interact the the system.
  - Attributes of the system inputs and system outputs
- We create the system-level design by mapping fragments of system functionality/behavior onto specific subsystems/objects in the system structure.