ENES 489P Hands-On Systems Engineering Projects

Systems Engineering Drivers

Mark Austin

E-mail: austin@isr.umd.edu

Institute for Systems Research, University of Maryland, College Park

Topic 2: Systems Engineering Drivers

1. Systems Engineering Drivers: Technical Viewpoint
   - Information-Centric Systems,
   - Growing importance of Systems Integration,
   - Need for Error-Free Software,
   - Agility in System Development,
   - Formal Approaches to Trade Studies.
2. Systems Engineering Drivers: Signature Applications
   - Automobile Electronics,
   - Washington DC Metro System.
3. Systems Engineering Drivers: Management Viewpoint
   - User/customer involvement,
   - Clear statement of requirements.

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: 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 1: Information-Centric Systems

Exemplars of Early Work

- Great Pyramid of Giza, Egypt (20 year construction; finished 2556 BC).
- Construction of the Great Wall of China (220 BC).

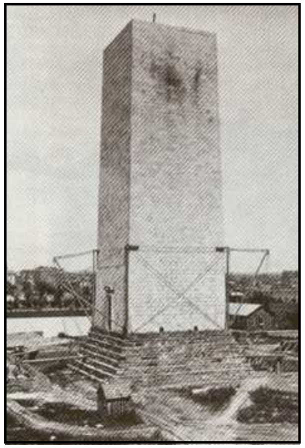
Challenge 1: Information-Centric Systems

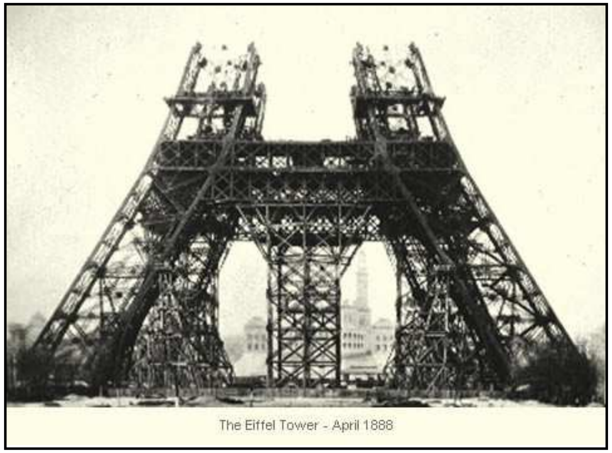
Industrial Revolution (1750 – 1850)

Year	Milestone
1708	Jethro Tull's mechanical seed sower $\longrightarrow$ large-scale planting/cultivation.
1765	Invention of the spinning jenny/wheel automates weaving of cloth.
1775	Watt's first efficient steam engine.
1801	Robert Trevithick demonstrates a steam locomotive.
1821	Faraday demonstrates electro-magnetic rotation $\longrightarrow$ electric motor.
1834	Charles Babbage analytic engine $\longrightarrow$ forerunner of the computer.
1854	Bessemer invents steel converter.
1863	Siemens-Martin open hearth process makes steel available in bulk.

Challenge 1: Information-Centric Systems

Advances in Construction (1750 – 1850)





- Left: Base of the Washington Monument; middle, base of the Eiffel Tower; right, Skyscraper construction.

Advances in Medicine (1750 – 1850)

- During 1730 - 1749. 74.5% of children born in London died before the age of five.
- By 1810 - 1829. 31.8% of children born in London died before the age of five.

Challenge 1: Information-Centric Systems

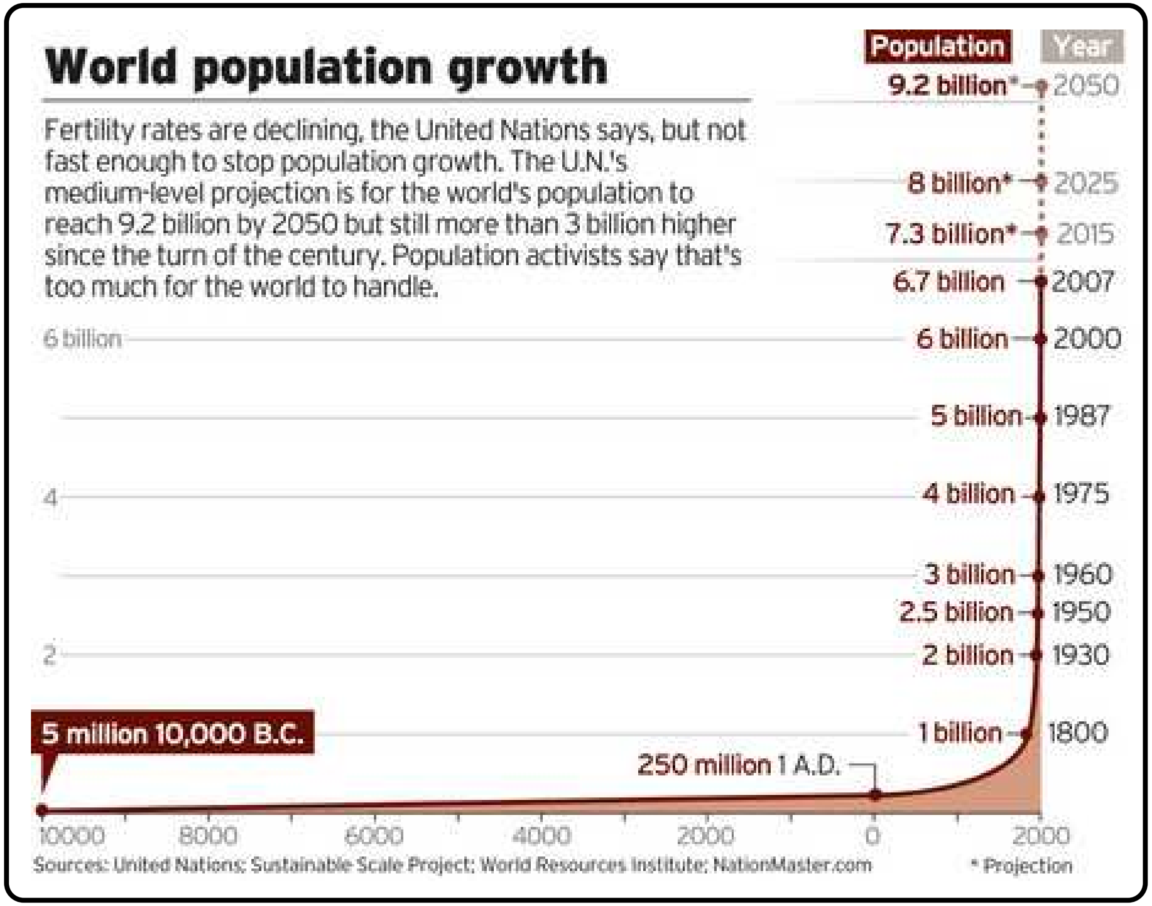
Early Skyscrapers

Skyscrapers (1890s) create habitable spaces in tall buildings for office workers.

Enablers	**Example: Empire State Building**
• New materials → design of tall structures having large open interior spaces. • Elevators (1857) → vertical transportation building occupants. • Mechanical systems → delivery of water, heating and cooling. • Collections of skyscrapers → high-density CBDs/commuter society.	

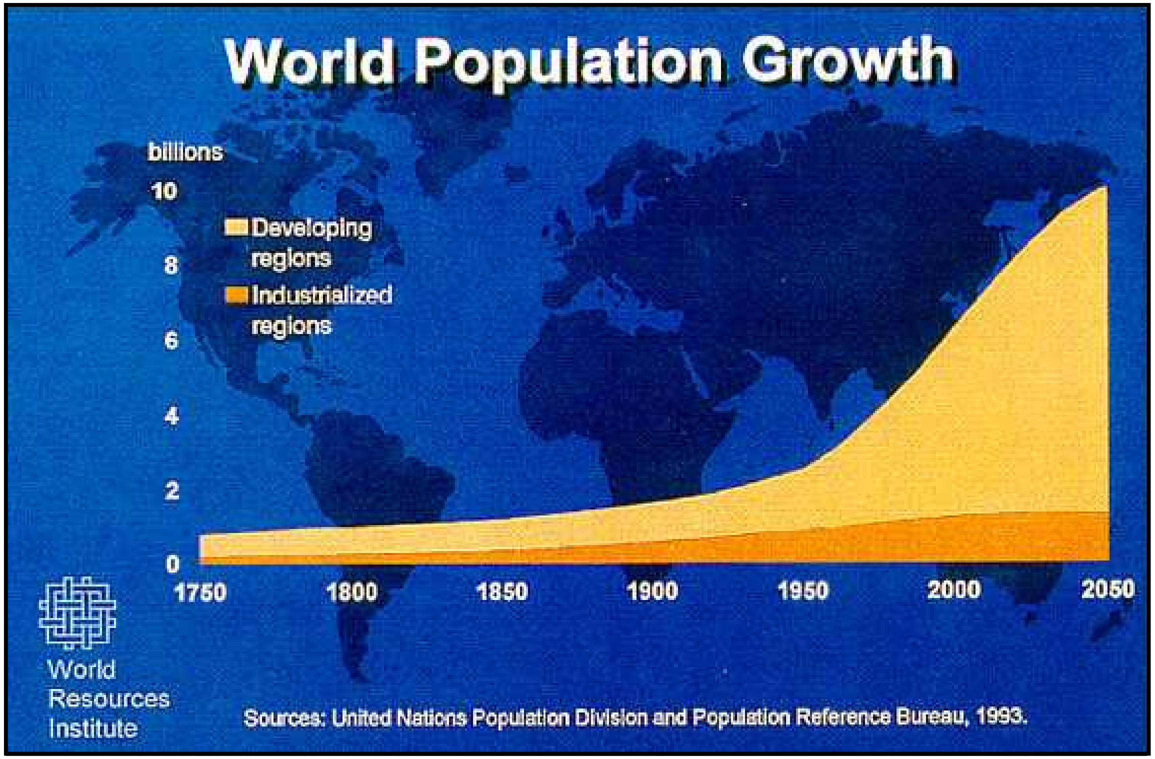
Challenge 1: Information-Centric Systems

Trends in World Population Growth



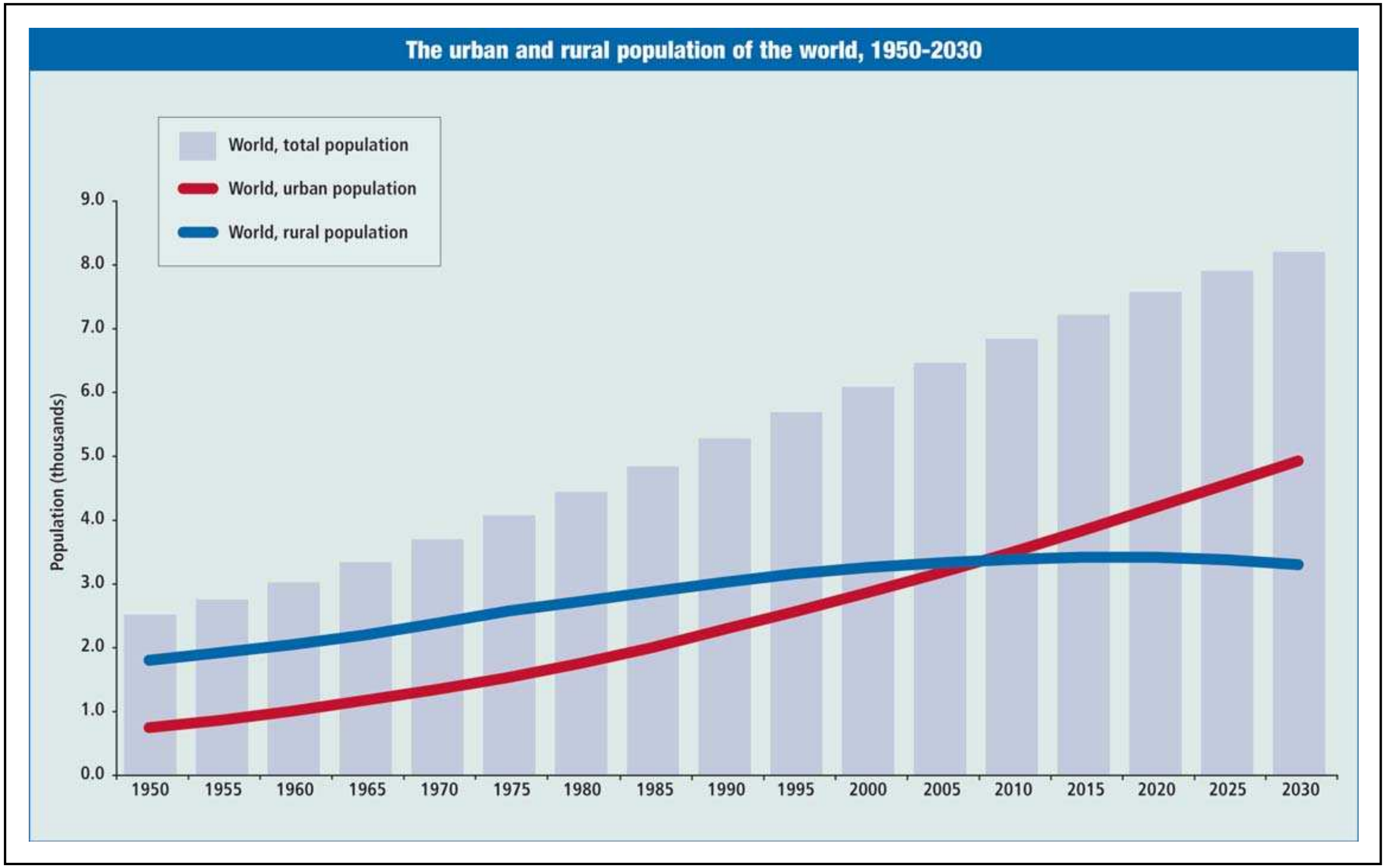
Challenge 1: Information-Centric Systems

Trends in World Population Growth



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

Rural to Urban Population Drift



Challenge 1: Information-Centric Systems

Urbanization in America

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

Cities are responsible for:

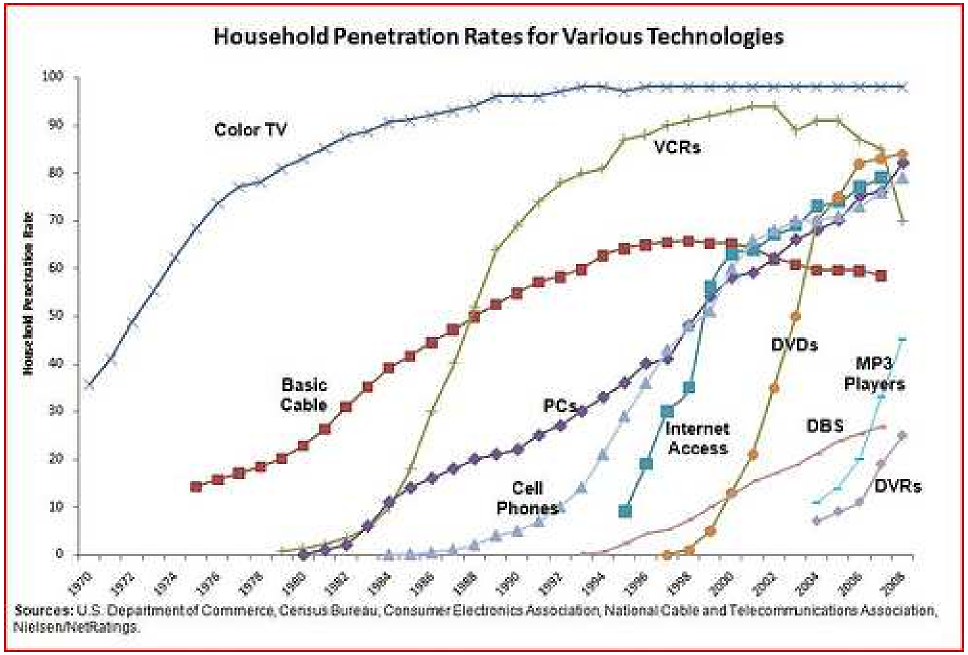
- Two thirds of the energy used,
- 60 percent of all water consumed, and
- 70 percent of all greenhouse gases produced worldwide.

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.**

Challenge 1: Information-Centric Systems

Accelerating pace of technology innovation



Observation: Humans perceive change as being a linear phenomena, but mathematics tells us that rates of change are constant and actual change is exponential ...

Challenge 1: 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 1: 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 1: Information-Centric Systems

Information-Age Systems

Developed under the premise that advances in

- Computing,
- Sensing, and
- Communications

technologies will allow for

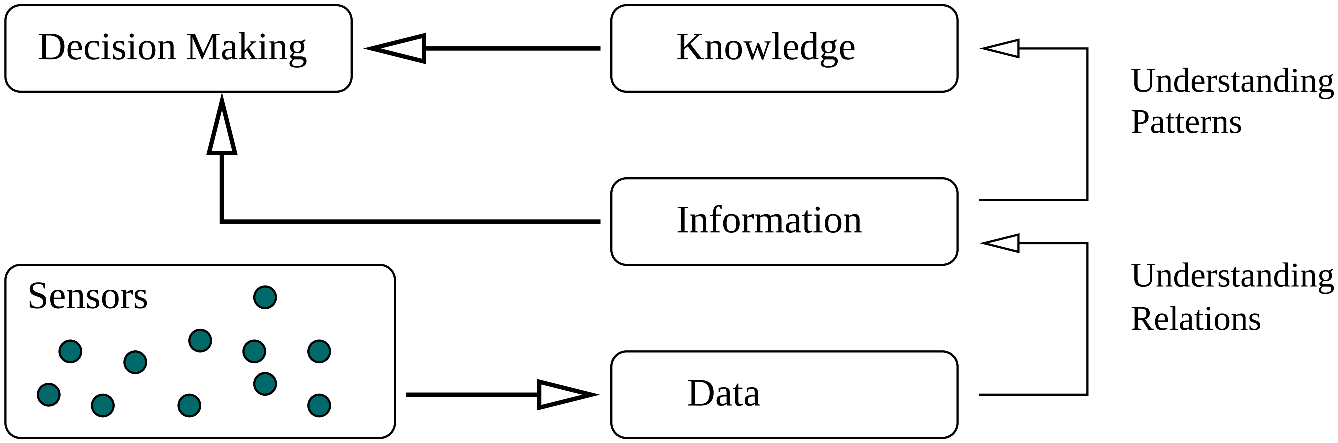
... new types of systems where human involvement is replaced by automation.

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 1: 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 1: Information-Centric Systems

Man and Machine

The traditional role of man and machine is facilitated by complementary strengths and weaknesses.

Man	Machine
• Good at formulating solutions to problems (algorithms). • Can work with incomplete data/information. • Creative. • Reasons logically, but very slow... • Performance is static.	• Electo-mechanical machine that can manipulate Os and 1s. • Very specific abilities. • Requires precise decriptions of problem solving procedures. • Dumb, but very fast. • Performance doubles every 18 months.

Challenge 1: Information-Centric Systems

Sensible Problem Solving Strategy

Let engineers and computers do what they are best at. This strategy:

1. Accelerates the solution procedure.
2. Enables the analysis of problems having size and complexity beyond manual examination.

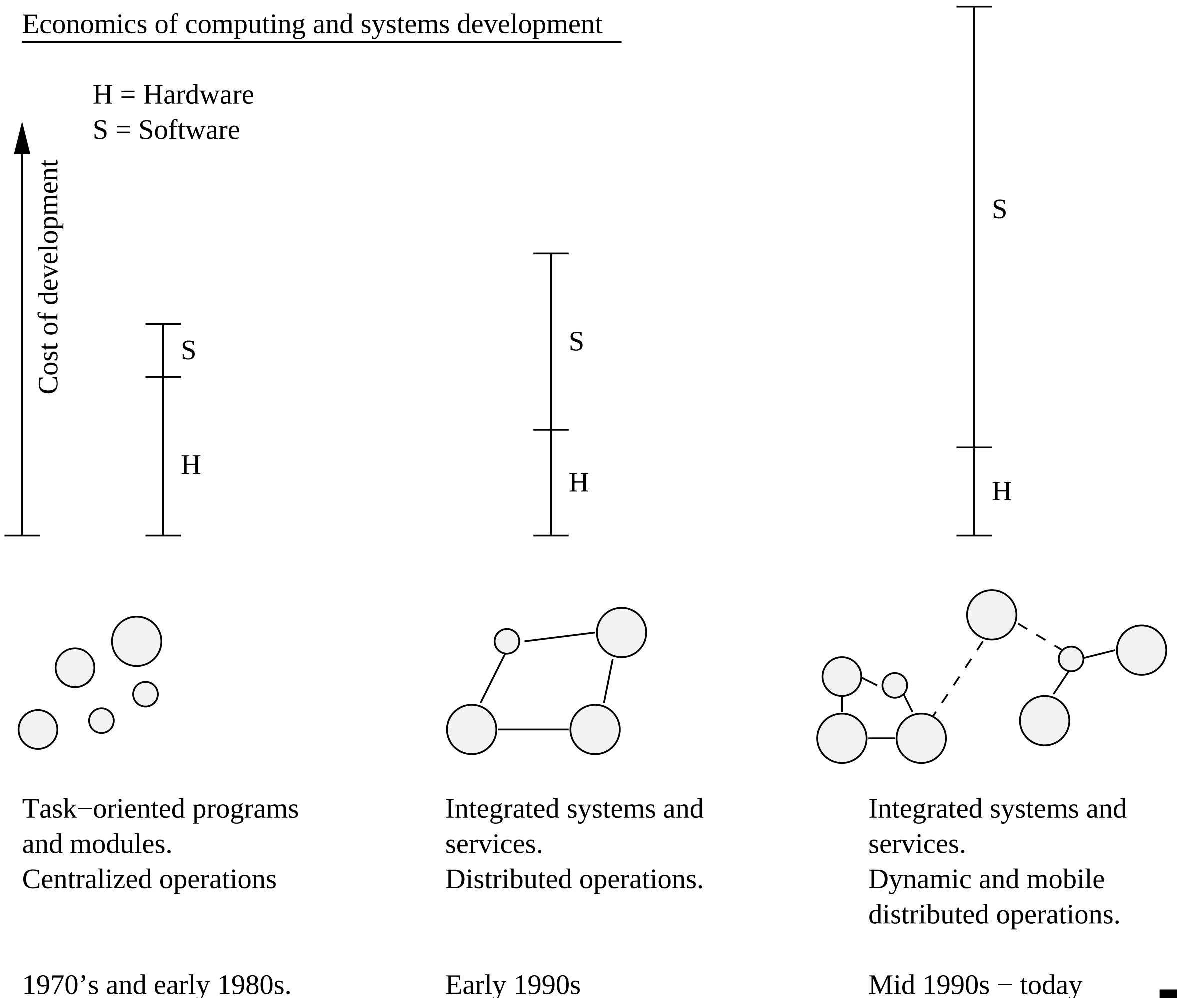
Getting things to work ...

> **... we need to describe to the computer solution procedures that are completely unambiguous.**

That is, we will need to look at data, organization and manipulation of data, and formal languages.

Challenge 1: Information-Centric Systems

Rapidly Expanding Expectations ...



Challenge 1: 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 1: 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 networking.	Embedded real-time control of physical systems.
Interaction	Touch, multi-touch, proximity.	....
Languages	XML, RDF, OWL.	New languages to support time-precise computations.

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

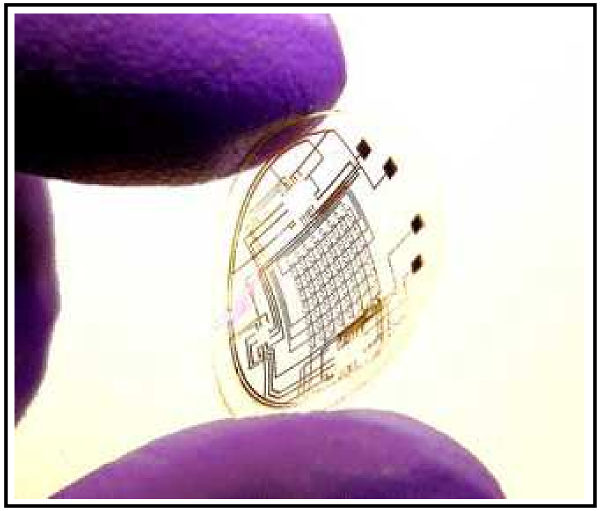
Challenge 1: Information-Centric Systems

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 1: Information-Centric Systems

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

Physical System Concerns

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**.
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 **reliability analysis**.
6. For design purposes, uncertainties in behavior are often handled through the use of safety factors.

Challenge 1: Information-Centric Systems

Software System Concerns

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. The main benefit of software is that ...

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

5. A small logical error can result in a system-wide failure.

Challenge 2: Systems Integration

Goals of Systems Integration

System integration involves ...

> **... joining existing disparate services or systems together into a single view or process for the user.**

Since many of the participating subsystems will have well-defined interfaces, integration involves joining the subsystems together by gluing their interfaces together.

Simple Idea

Improve system performance by promoting teamwork, i.e.,

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

So what's the catch?

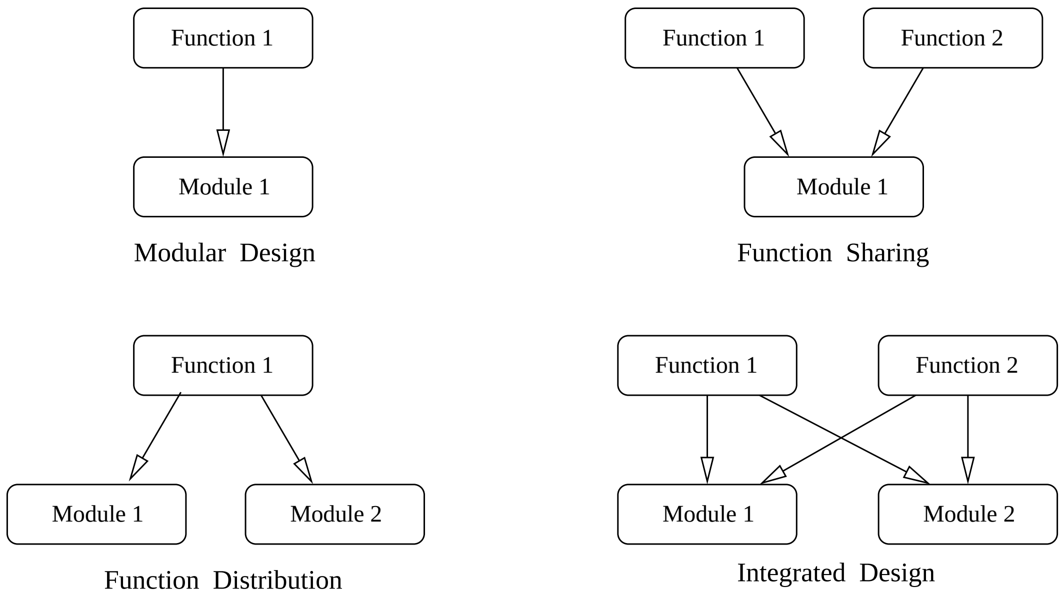
> **Integration requires concurrent consideration of each sub-systems functions and performance, together with models of connection and communication among sub-systems.**

Challenge 2: Systems Integration

Modular and Integrated Development of Systems

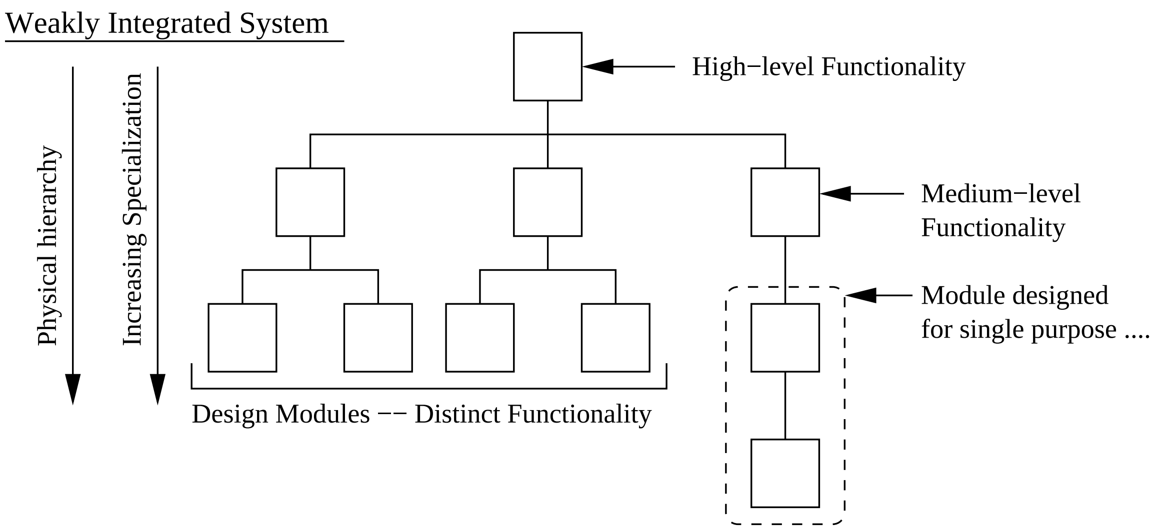
A modular architecture has well-defined, standardized, and decoupled interfaces which collectively allow for design changes to be made to one module, without generally requiring a change to other modules.

Four types of product architecture:



Challenge 2: Systems Integration

Nodal connectivity and functional influence in a weakly-integrated system

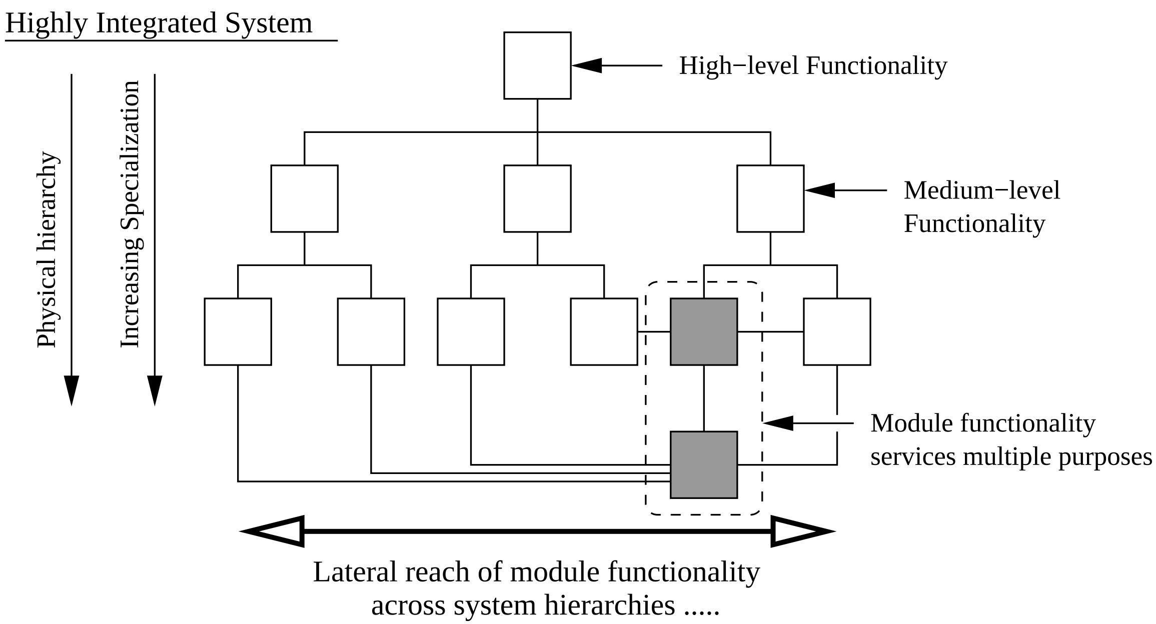


Key characteristics:

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:

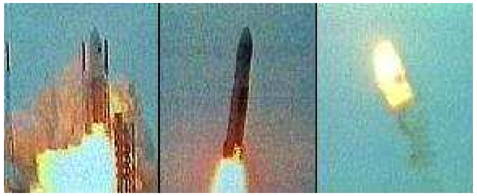
... software and communications technologies that must work correctly and with no errors.

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.

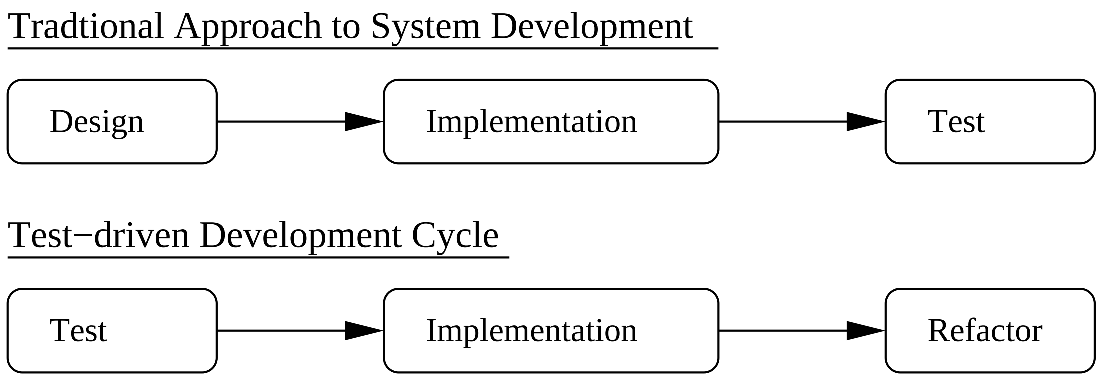
Still, it is well known that ...

> **... 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



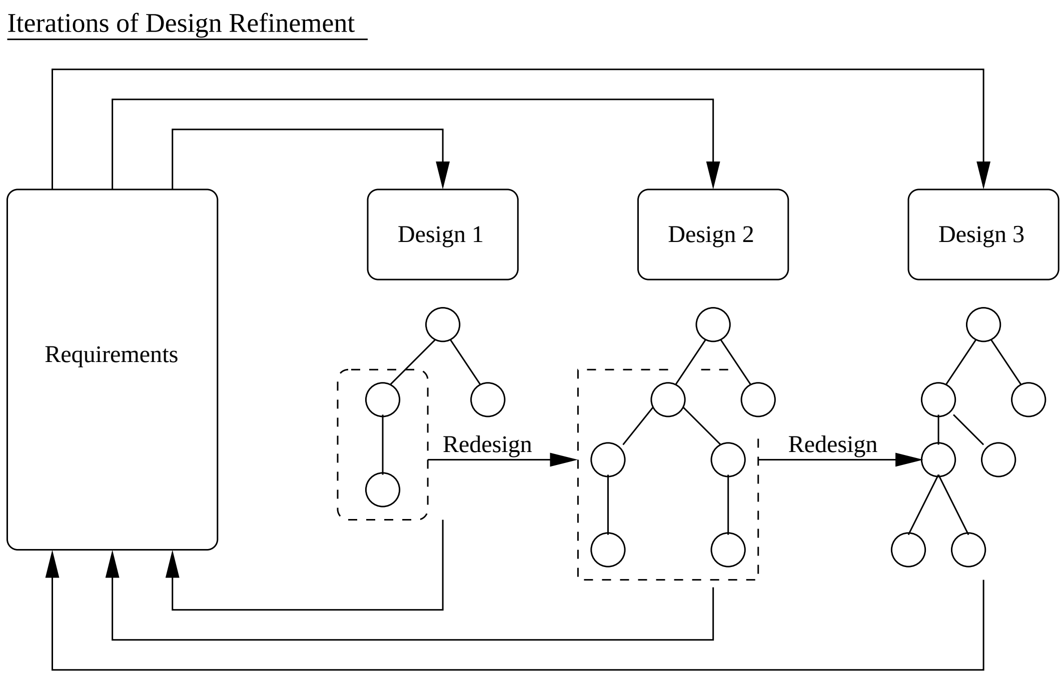
Workflows for test-driven development are based on a very simple tenet:

... 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.**

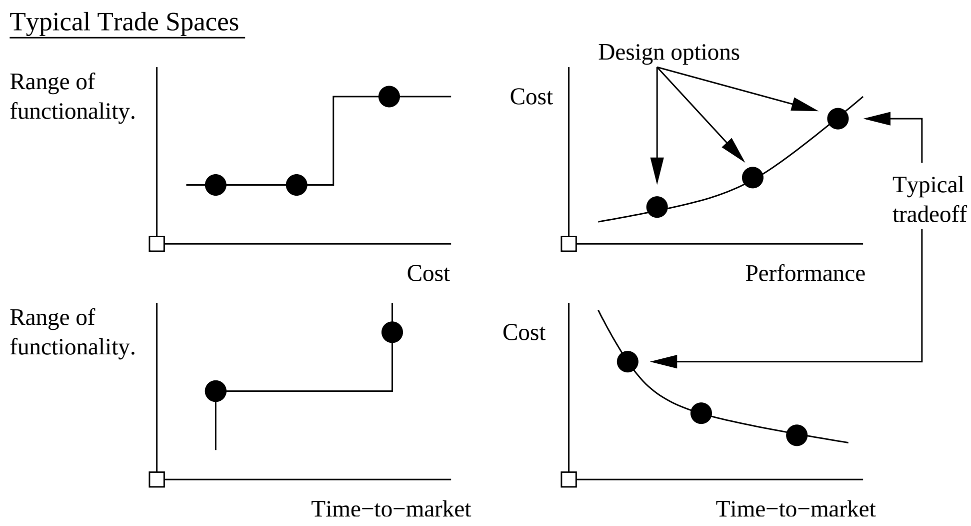
It also suggests that design developments should be persistent, meaning that step-by-step procedures for creating a design should be completely reversable.

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.**

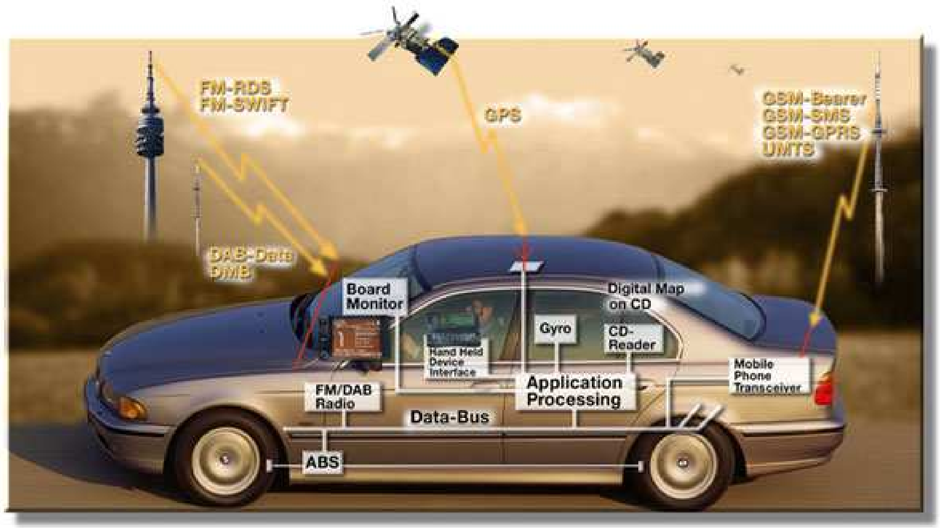
Challenge

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 Study 1: 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 1: 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 (30-70) 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 2: Washington DC Metro System

Washington D.C. Metro Train Crash (June 2009)

Case Study 2: Washington DC Metro System

Key points:

- Investigations invariably focus our attention on discrete aspects of machine or human error, whereas ...

  ... the real problem often lies in the relationship between humans and their automated systems.

- You really need to trace the cause of an accident back to the underlying fault.

- Safer automated systems leads to a paradox at the heart of all human-machine interactions:

  "...The better you make the automation, the more difficult it is to guard against these catastrophic failures in the future, because the automation becomes more and more powerful, and you rely on it more and more."

- In another incident the National Transportation Safety Board found that:

  ....the driver of the train had reported overshooting problems at earlier stops but was told not to interfere with the automated controls.

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).