Professional Practice (IT)

CSC110

Week # 01

Professional Practice

- **Credit Hours: 3(3,0)**
- **Objectives:**

A Computing graduate as professional has **some responsibilities** with respect to the society. This course develops student understanding about **historical, social, economic, ethical, and professional issues** related to the discipline of **Computing**. It identifies key sources for information and opinion about **professionalism and ethics**. Students analyze, evaluate, and assess ethical and professional computing case studies.

Professional Practice

Course Objectives:

It is expected that at the end of this course the student will be able to;

1. **Understand** about historical, social, economic, ethical, and professional issues related to the discipline of Computing
2. **Identify key sources** for information and opinion about professionalism and ethics.
3. Analyze, evaluate, and assess **ethical and professional computing** case studies.
4. **Use, misuses, and risks** of software, information security and privacy.
5. Business practices and the economics of software.
6. Intellectual property and software law (**cyber law**)
7. Software related **contracts**
8. Software house organization etc.

Professional Practice

Course Outcomes

At the end of the course, the student should be able to;

- Understand the importance of computing in industry and other professional domains.
- Importance of professional responsibilities and ethical issues.
- Professional practice and professionalism
- Importance of intellectual property and Cybercrimes.
- Understand how to write contracts and request for proposals.
- How to organize a team for successful deployment of software.

Professional Practice

Course Outline

Historical, social, and economic context of Computing (software engineering, Computer Science, Information Technology); Definitions of Computing (software engineering, Computer Science, Information Technology) subject areas and professional activities; professional societies; professional ethics; professional competency and life-long learning; uses, misuses, and risks of software; information security and privacy; business practices and the economics of software; intellectual property and software law (cyber law); social responsibilities, software related contracts, Software house organization.

Recommended Text Books:

- Professional Issues in Software Engineering, M.F. Bott et al.

Reference Books:

- Deborah G. Johnson, “Computer Ethics”, Pearson Education.

Need to remember:

- Minimum Class attendance 80%
- Evaluation Policy
  - Assignments 10
  - Quizzes / Class Participation 15
  - 1st Sessional (after 4 weeks) 10
  - 2nd Sessional (after 9 weeks) 15
  - Final examination (after 16 weeks) 50
  - **Total** **100**
- Un-marked Issues but Important:
  - Class discipline
  - Professionalism and Ethical Practice
  - Trustworthy and Self-confidence

Computing

In this section, we will cover the following topics;

Computing and its applications

Computing history

Computer software

Computer users

Five main sub-discipline of computing

Computing:

- **Computing** is any Problem-oriented activity requiring, benefiting from, or creating algorithmic processes - e.g. through computers.

- "In a general way, we can define computing to mean any goal-oriented activity requiring, benefiting from, or creating computers.

- Thus, computing includes
  - designing and building hardware and software
  - processing, structuring, and managing various kinds of information
  - doing scientific studies using computers
  - making computer systems behave intelligently
  - creating and using communications and
  - entertainment media
  - finding and gathering information relevant to any particular purpose, and so on.

- The list is virtually endless, and the possibilities are vast.“
- All facts are related to SE directly or indirectly.

History of computing:

- The history of computing is longer than the history of computing hardware and modern computing technology and includes the history of methods intended for pen and paper or for chalk and slate, with or without the aid of tables.

- Computing is intimately tied to the representation of numbers.

- The earliest known tool for use in computation was the abacus, and it was thought to have been invented 2400 BC. Its original style of usage was by lines drawn in sand with pebbles.

Computer:

- A computer is a machine that manipulates data according to a set of instructions called a computer program.

- The program has an executable form that the computer can use directly to execute the instructions.

- The same program in its **human-readable source code** form, enables a programmer to study and develop the algorithm.

Computer software:

- Computer software or just "**software**", is a collection of computer programs and related data that provides the instructions for telling a computer what to do and how to do it.

- software is a set of ***programs, procedures, algorithms*** and its ***documentation*** concerned with the operation of a data processing system.
- Direct mode (commands)
- Indirect mode(statements)

Application & System software:

- Application software, a computer software designed to help the user to perform specific tasks.
- Apps may be bundled with the computer and its system software, or may be published separately. Some users are satisfied with the bundled apps and need never install one.
- Application software applies the **power** of a particular computing platform or system software to a particular purpose.
- Platform dependability is also an issue.

- System software, is computer software designed to operate and control the computer hardware and to provide a platform for running application software.

Computer Network:

- A computer network, often simply referred to as a network, is a collection of hardware components and computers interconnected by communication channels that allow sharing of resources and information.

- Networks may be classified according to a wide variety of characteristics such as the medium used to transport the data, communications protocol used, scale, topology, and organizational scope.

- Computer networking is sometimes considered a sub-discipline of electrical engineering, telecommunications, computer science, information technology or computer engineering, since it relies upon the theoretical and practical application of these disciplines.

Computer User:

- A user is an agent, either a human agent (end-user) or software agent, who uses a computer or network service.
- Computer User can be a:
  - End User

    The term end-user refers to the ultimate operator of a piece of software, but it is also a concept in software engineering, referring to an idea of that group of end-users of computers.
  - Computer Programmer

    A programmer, computer programmer, or coder is a person who writes computer software. The term *computer programmer* can refer to a specialist in one area of computer programming or to a generalist who writes code for many kinds of software.

Why Computer Programming?

- Computer programming in general is the process of writing, testing, debugging, and maintaining the source code and documentation of computer programs.

- The purpose of programming is to invoke the desired behavior (customization) from the machine.

- The process of writing high quality source code requires knowledge of both the application's domain *and* the computer science domain. The highest-quality software is thus developed by a team of various domain experts, each person a specialist in some area of development.

Why Computer Programming?

- Steps of programming for SE
- Customer’s behaviour
- Maximum utilization of Machine
- Must know CS and application domains

Threat:

- *A programmer*
- *Technical programmer*
- *Open source programmer*
- *Professional*
- *Individual who can create "Killer applications".*

- *Programmer* may apply to a range of program quality, from hacker to open source contributor to professional.

- A single programmer could do most or all of the computer programming needed to generate the proof of concept to launch a new "killer" application
- (any computer program that is so necessary or desirable that it proves the core value of some larger technology).

Five sub-disciplines of the *computing* field:

- Computer Engineering
- Software Engineering
- Computer Science
- Information Systems
- Information Technology

Sub-disciplines of computing

Computer Engineering:

- Computer engineering is a discipline that integrates several fields of electrical engineering and computer science required to develop computer hardware and software.

- Computer engineers usually have training in electronic engineering (or electrical engineering), software design, and hardware-software integration instead of only software engineering or electronic engineering.

- Computer engineers are involved in many hardware and software aspects of computing, from the design of individual microprocessors, personal computers, and supercomputers, to circuit design.

- This field of engineering not only focuses on how computer systems themselves work, but also how they integrate into the larger picture.

Sub-disciplines of computing (see later)

Software Engineering (SE):

- Software engineering (SE) is the application of a systematic, disciplined, measurable approach to the design, development, operation, and maintenance of software, and the study of these approaches; that is, the application of engineering to software.

- The first reference to the term is the 1968 NATO Software Engineering Conference and was meant to provoke thought regarding the perceived "software crisis(difficulty of writing useful and efficient computer programs in the required time)" at the time.

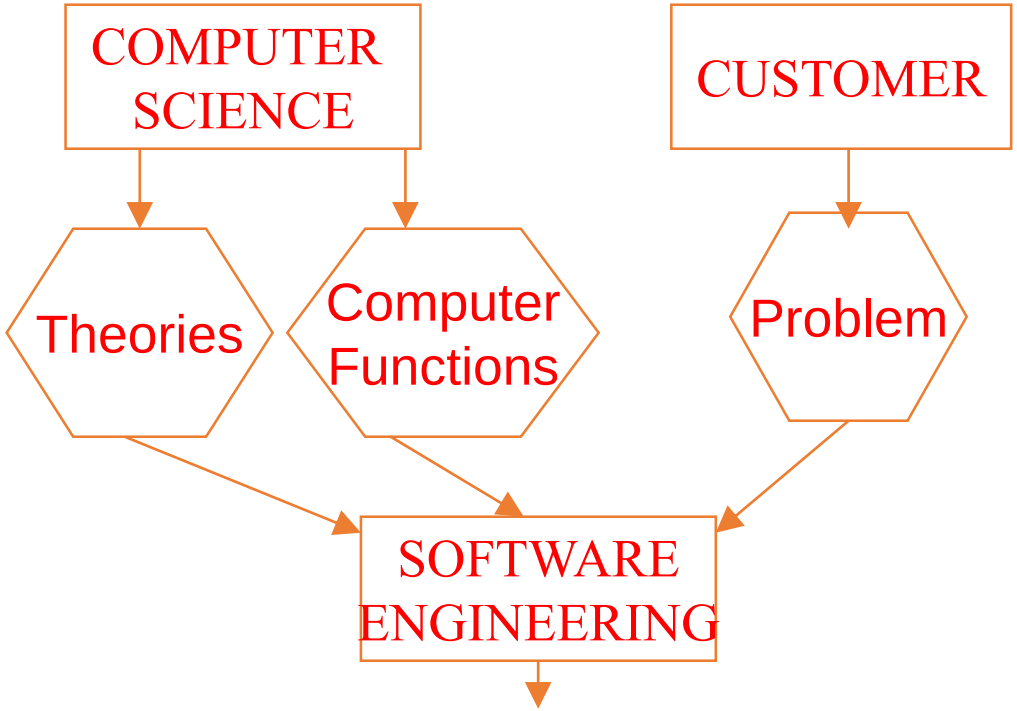
- The generally accepted concepts of Software Engineering as an engineering discipline have been specified in the Guide to the Software Engineering Body of Knowledge (SWEBOK) that is an internationally accepting standard.

Sub-disciplines of computing (see later)

Computer Science (CS):

- Computer science or computing science is the scientific and practical approach to computation and its applications.
- A computer scientist specializes in the theory of computation and the design of computational systems.
- Its subfields can be divided into practical techniques for its implementation and application in computer systems and purely theoretical areas.
- focus on the challenges in implementing computations such as programming language theory , computer programming and complex systems, human-computer interaction etc.

Software Engineering vs. Computer Science



We will discuss in more detail in next class

Sub-disciplines of computing

Information System (IS):

- "Information systems" is the study of complementary networks of hardware and software that people and organizations use to collect, filter, process, create, and distribute data.

- The study bridges business and computer science using the theoretical foundations of information and computation to study various business models and related algorithmic processes within a computer science discipline.

- **Computer Information System(s)** (CIS) is a field studying computers and algorithmic processes, including their principles, their software and hardware designs, their applications, and their impact on society while IS emphasizes functionality over design such as MIS, AIS, DSS etc.

Sub-disciplines of computing

Information Technology (IT):

- Information technology (IT) is the application of computers and telecommunications equipment to store, retrieve, transmit and manipulate data, often in the context of a business or other enterprise.

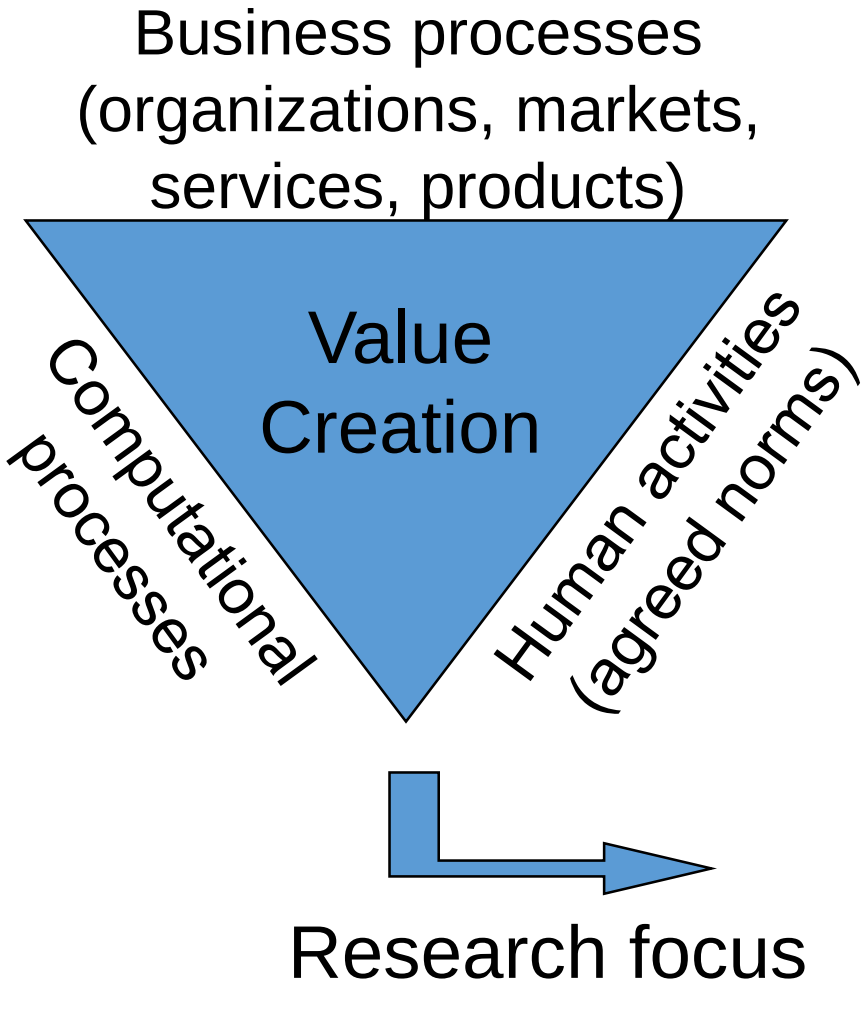
- The term is commonly used as a alternative word for computers and computer networks, but it also involves other information distribution technologies such as television and telephones.

- Several industries are associated with information technology, such as computer hardware, software, electronics, semiconductors, internet and telecom equipment, e-commerce and computer services.

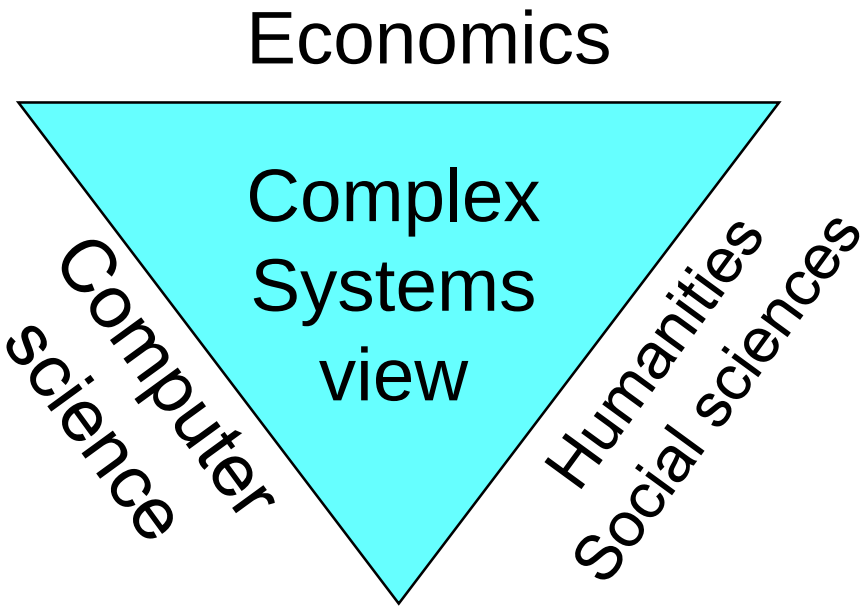
System Administrator:

- Systems administrator, is a person employed to maintain and operate a computer system and/or network.
- The duties of a system administrator are wide-ranging, and vary widely from one organization to another.
- System administrators are usually charged with installing, supporting and maintaining servers or other computer systems, and planning for and responding to service outages and other problems.
- Other duties may include scripting or light programming, project management for systems-related projects, supervising or training computer operators, and being the consultant for computer problems beyond the knowledge of technical support staff.

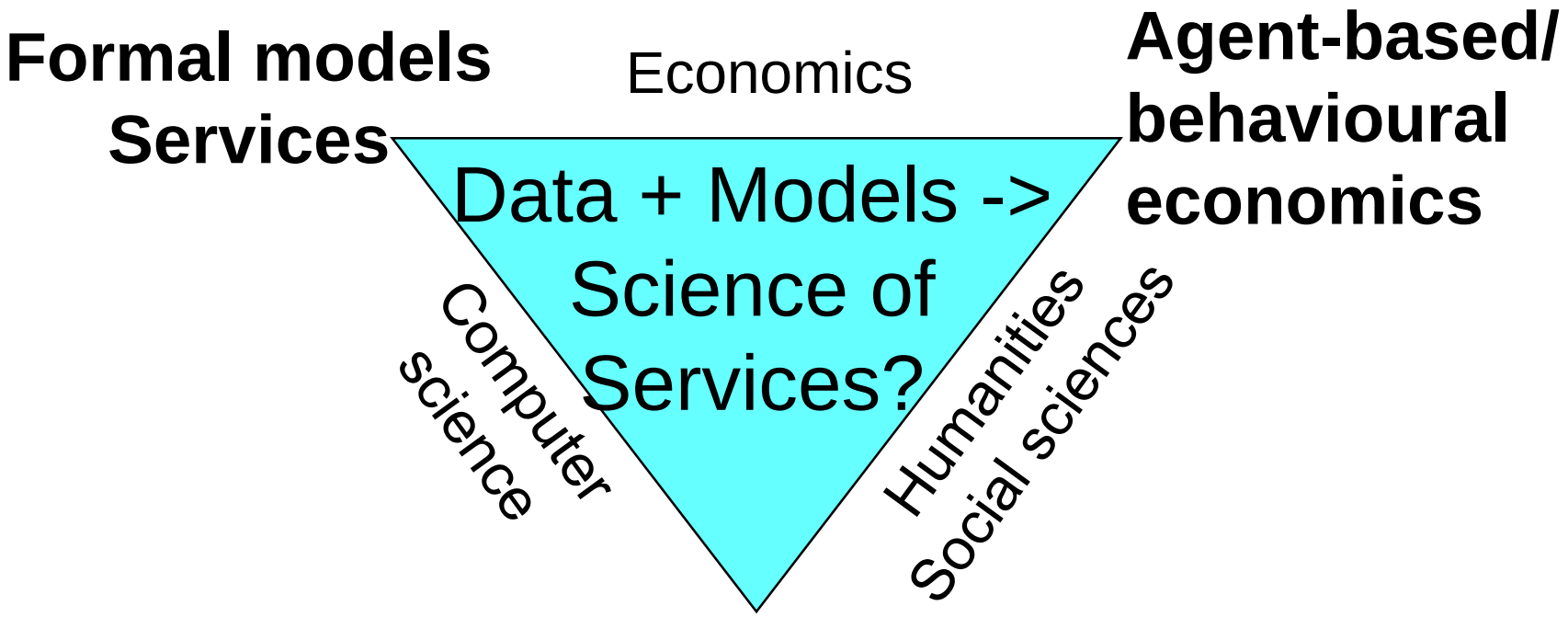
CS/IT reshapes business
Bridging human and computational processes



IT drives a novel
set of transformations:
What consequences
for society and business?



Towards "Service Science"



Services in the large:
How to manage millions of online services?

Professional Practice (IT)

CSC110

Week # 02/03

Science vs. Engineering

The difference between science and engineering:

- *Science* seeks to explain phenomena through **theory**, **hypothesis,** and **experiment**, in an effort to ascertain **natural laws**

  For example, chemistry investigates the structure of chemicals and their interactions

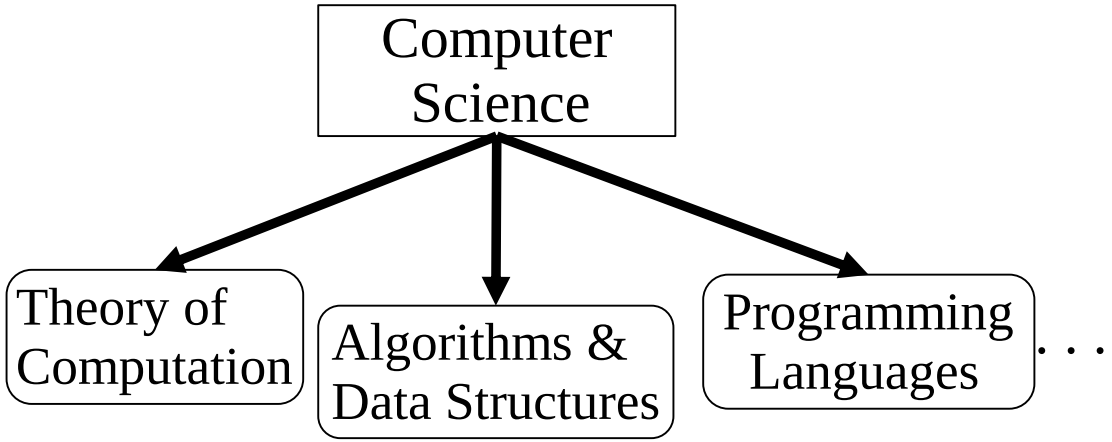
- *Engineering* seeks to apply natural laws to the solution of practical problems

  For example, chemical engineering might use the results of chemistry to come up with a better way of refining gasoline

Computer Science as a Science

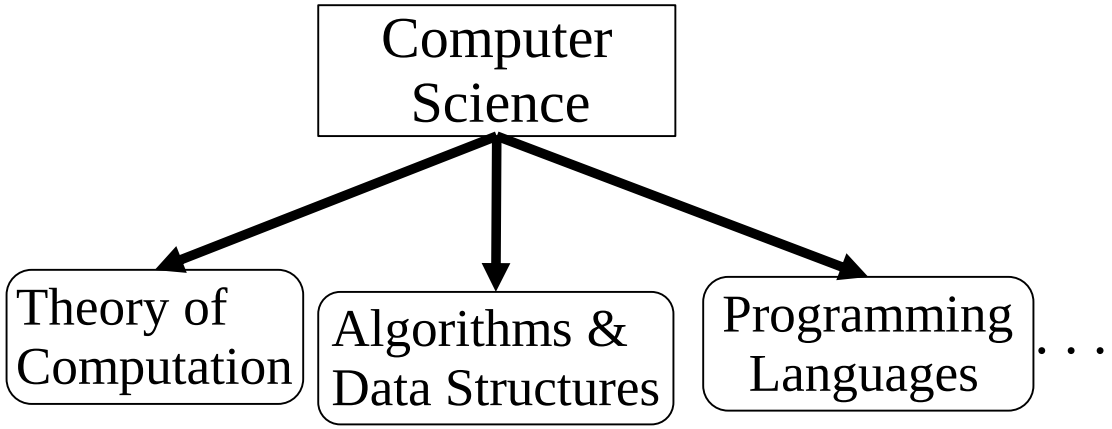
- **Theory:**
  - Computability, automata, discrete computational structures
  - Algorithm analysis
- **Hypothesis:**
  - That a certain algorithm will solve a problem
- **Experiment:**
  - Run a program implementing the algorithm

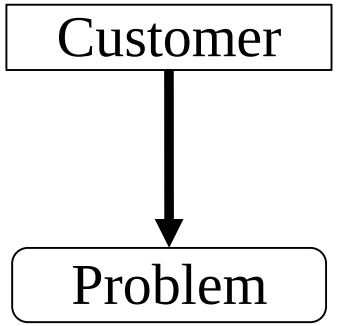
Computer Science as a Science (cont'd)



Theory Hypothesis Experiment

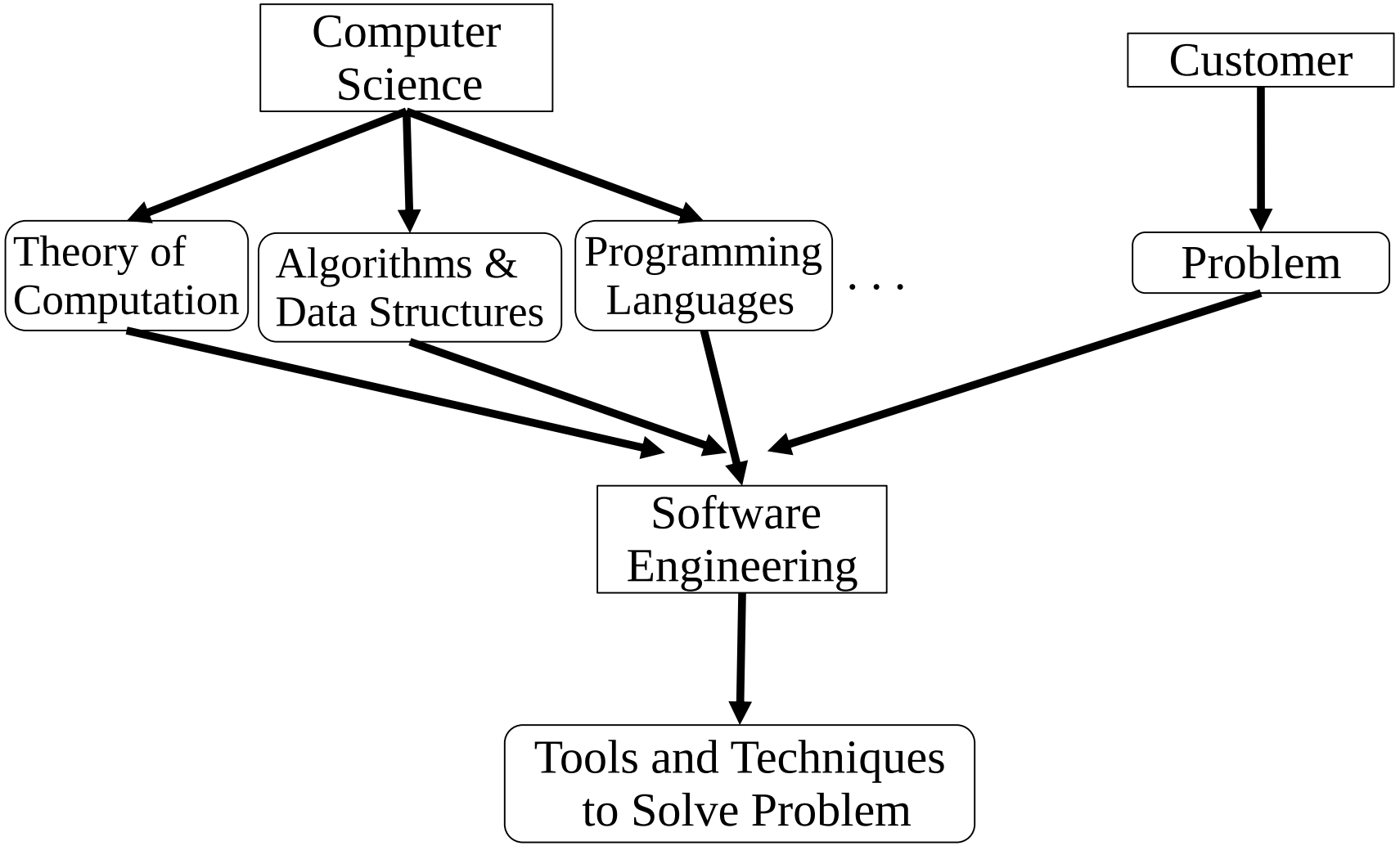
Adding a Customer





Requirements

The Difference Between Computer Science and Software Engineering



“Software Engineering” Definition

- Based on Webster definition of “engineering”

- The application of science and mathematics by which the properties of software are made useful to people

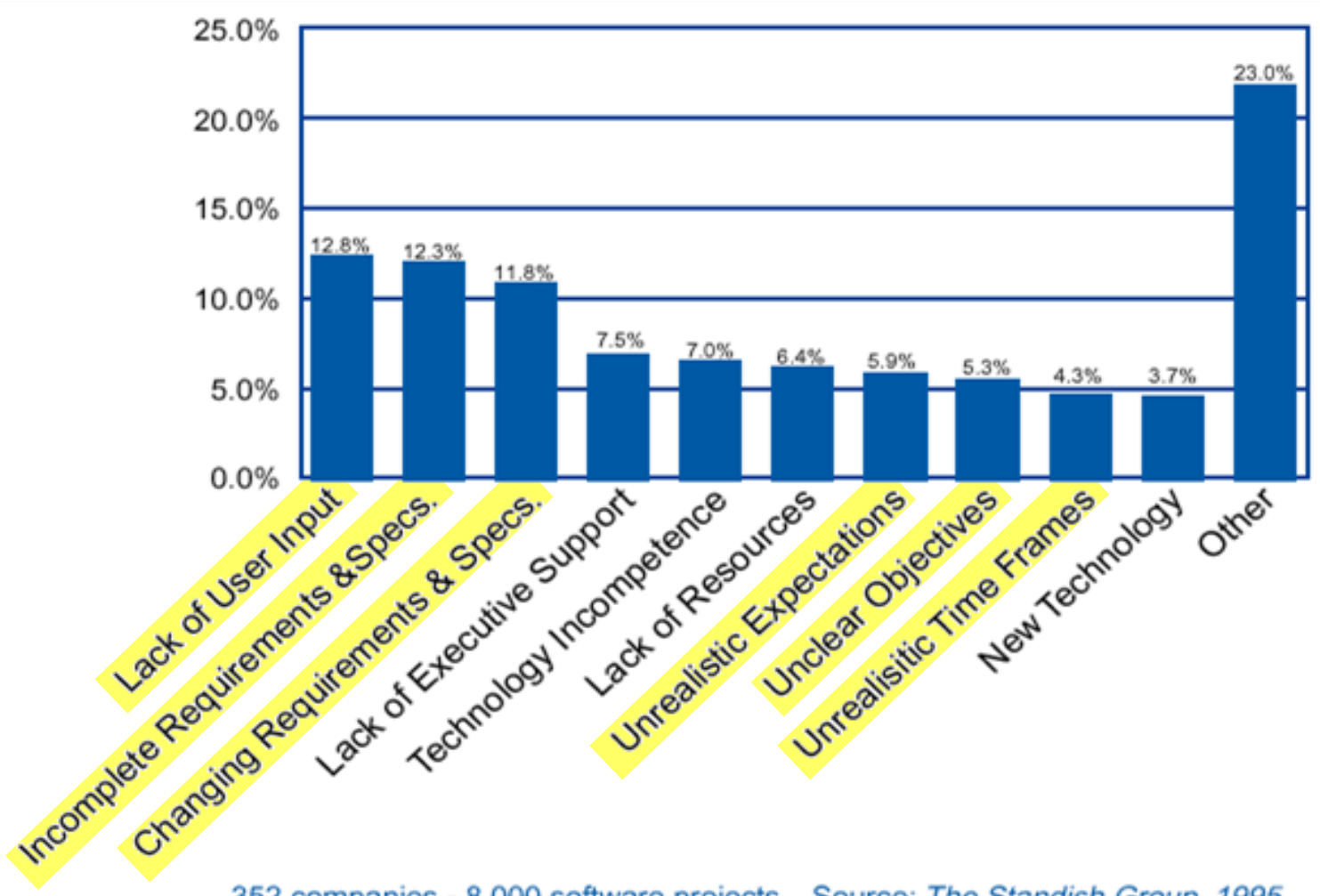
- Includes computer science and the sciences of making things useful to people
  - Behavioral sciences, economics, management sciences

History of the Role of Software

- In the 1950's software development was de-emphasized, because it only contributed to about 20% of overall system cost (SAGE)
- Programmers moved from machine language, to assembly language, to high-level language
- In 1968, a NATO report coined the term "software engineering"
- Hardware became faster and cheaper, outpacing the ability of software to keep up
- By the 1980's the software cost of a system had risen to 80%, and many experts pronounced the field "in crisis"

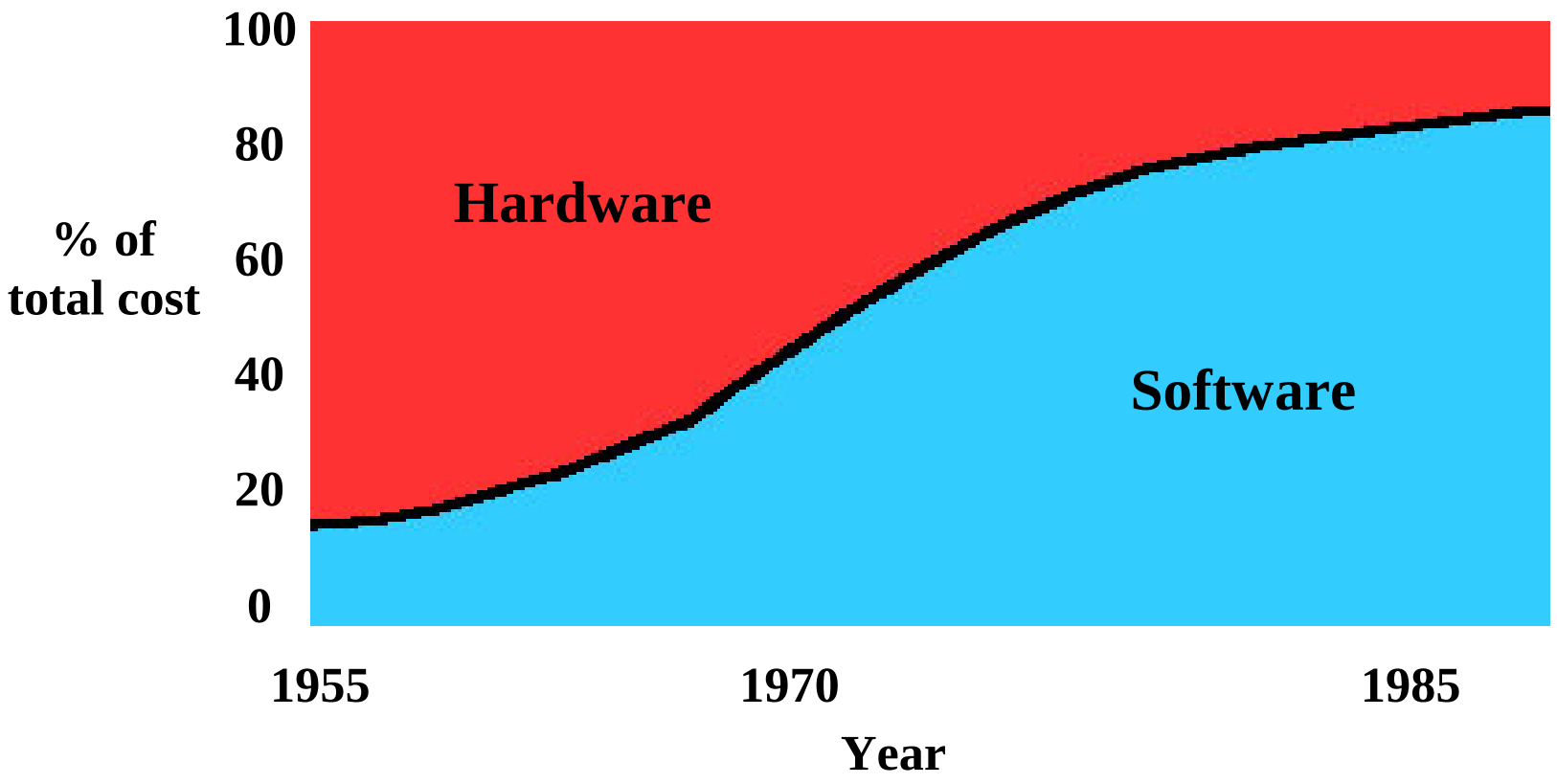
Why Software Projects Fail (software Crises)

- Average overrun: 89.9% on cost, 121% on schedule, with 61% of content



352 companies - 8,000 software projects. Source: *The Standish Group, 1995*

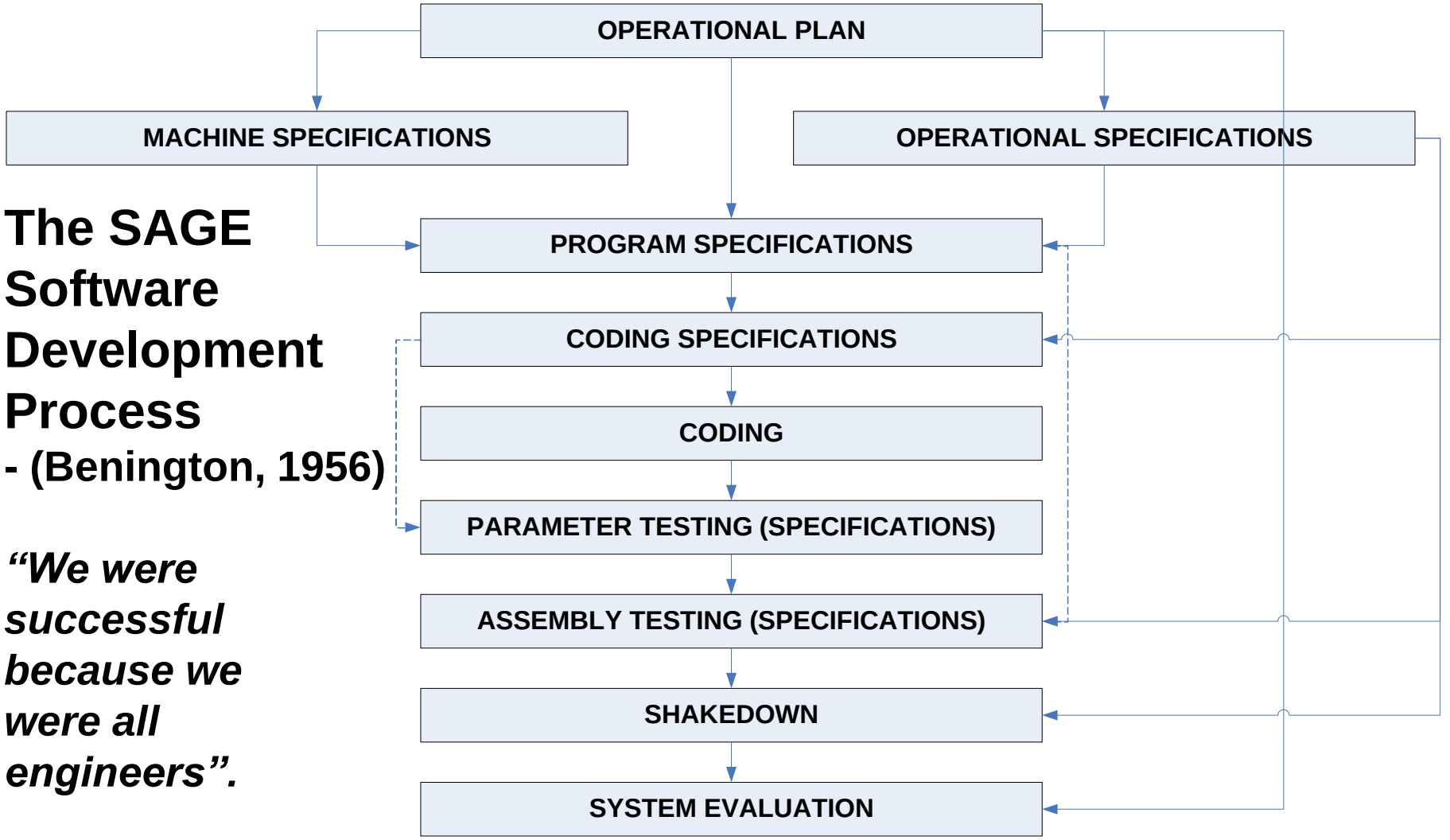
Large-Organization HW/SW Cost Trends (1973)



The SAGE Software Development Process

- (Benington, 1956)

"We were successful because we were all engineers".



Elements of the Continuing Software Crisis

- Software is **not delivered on time**
- Software is **over budget**
- Software is **too complex**. Complexity does not scale linearly: a 1000 line program is more than 10 times as complex as a 100 line program

Elements of the Software Crisis (cont'd)

- Software is **unmaintainable** due to:
  - poor design
  - poor documentation (most software can be understood only by its author, and then only within a few months of writing it)
- Software is **inefficient** (new versions of complex software require machine upgrades)
- Software is **unreliable** due to:
  - poor design
  - inadequate testing (market pressures, *beta* releases)
  - impossible testing

Key Factors Altering Software Engineering Practice Since 1970's

1. **Time-to-market criticality**: competition for market share
2. **Shifts in economics**: cheaper to add disk or memory than spend time optimizing code
3. **Desktop computing**: puts power in hands of users who demand more sophisticated tools
4. **GUIs**: make complex programs easier to use
5. Local and wide-area **networks**
6. **OOD** and **OOP**: enhance module reusability
7. Demise of the **waterfall** process model (see later)

The Goal of Software Engineering: Software Quality

Five perspectives on software quality:

1. **Transcendental view**: quality is recognizable but undefinable and unattainable, like a Nonphysical form
2. **Manufacturing view**: does product conform to a good process, and does it meet specs? (externally measured)
3. **User view**: product characteristics measured by fitness for purpose, e.g., correctness, reliability, maintainability (externally measured)
4. **Product view**: quality measured by *internal* characteristics (software "metrics") thought to be indicators of good *external* characteristics
5. **Value-based view**: quality depends upon the amount the customer is willing to pay (can manage conflicts among the other 4 views)

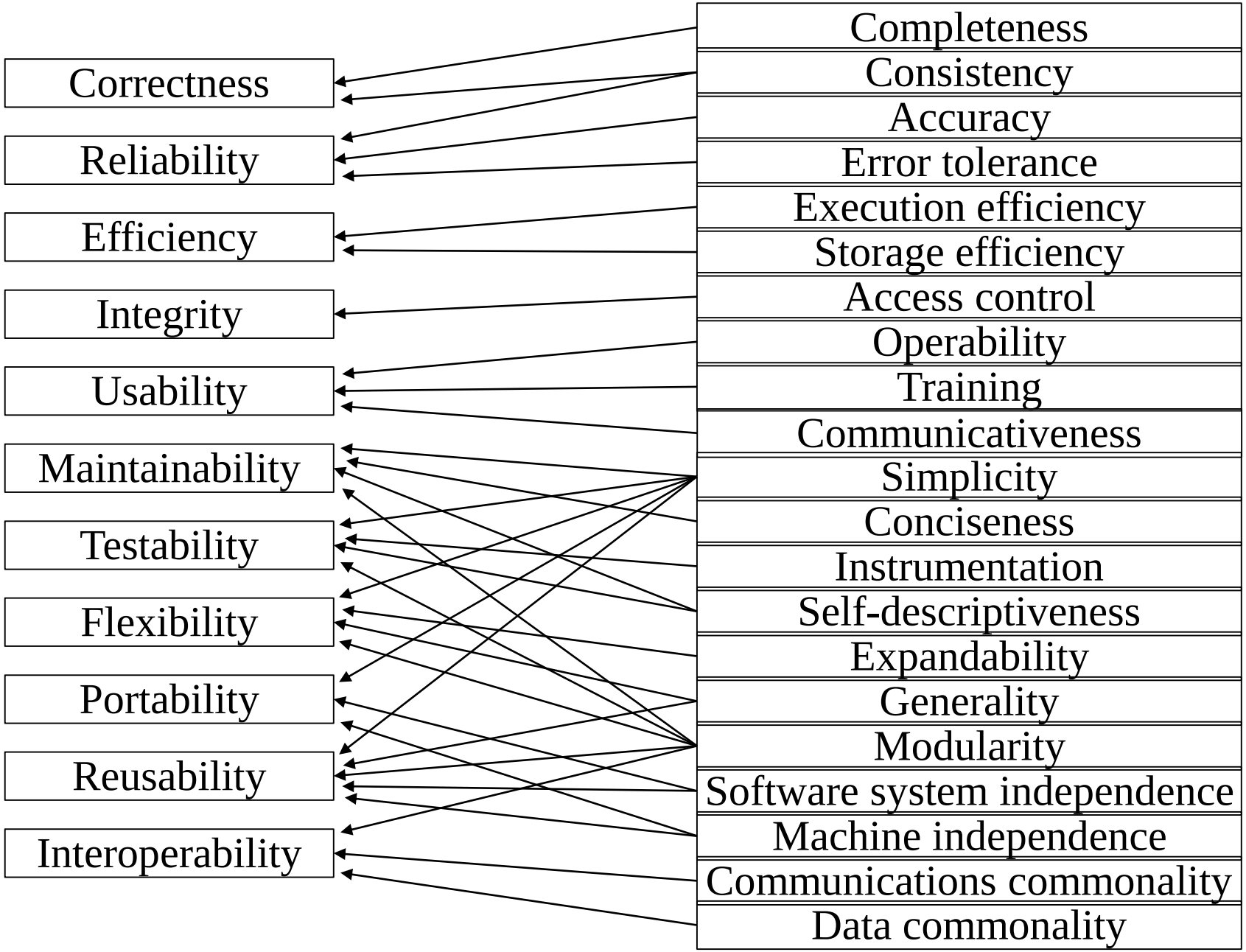
User Views

- Correctness
- Reliability
- Efficiency
- Integrity
- Usability
- Maintainability
- Testability
- Flexibility
- Portability
- Reusability
- Interoperability

Product Views

Completeness
Consistency
Accuracy
Error tolerance
Execution efficiency
Storage efficiency
Access control
Operability
Training
Communicativeness
Simplicity
Conciseness
Instrumentation
Self-descriptiveness
Expandability
Generality
Modularity
Software system independence
Machine independence
Communications commonality
Data commonality

Relating the User and Product Views



Elements of a Software Engineering Discipline

1. **Abstraction**: Identifying hierarchical *classes* of objects to reason about, ignoring detail
2. **Analysis and Design Methods and Notations**: For example, use of design patterns and UML
3. **User Interface Prototyping**: To help user and developer agree on requirements and software functions
4. **Software Architecture**: striving for independence of parts through modularity

Elements of a Software Engineering Discipline (cont'd)

5. **Software Process** (see later)
6. **Reuse**: Not just of software, but also sets of requirements, parts of designs, or groups of test scripts
7. **Measurement**: Trying to quantify project goals to evaluate progress (for example, number of bugs per 100 lines of code)
8. **Tools and Integrated Environments**: For example, CASE (computer-aided software engineering) tools

Software Development Steps

Requirements Analysis and Definition

System Design

Program Design

Program Implementation

Unit Testing

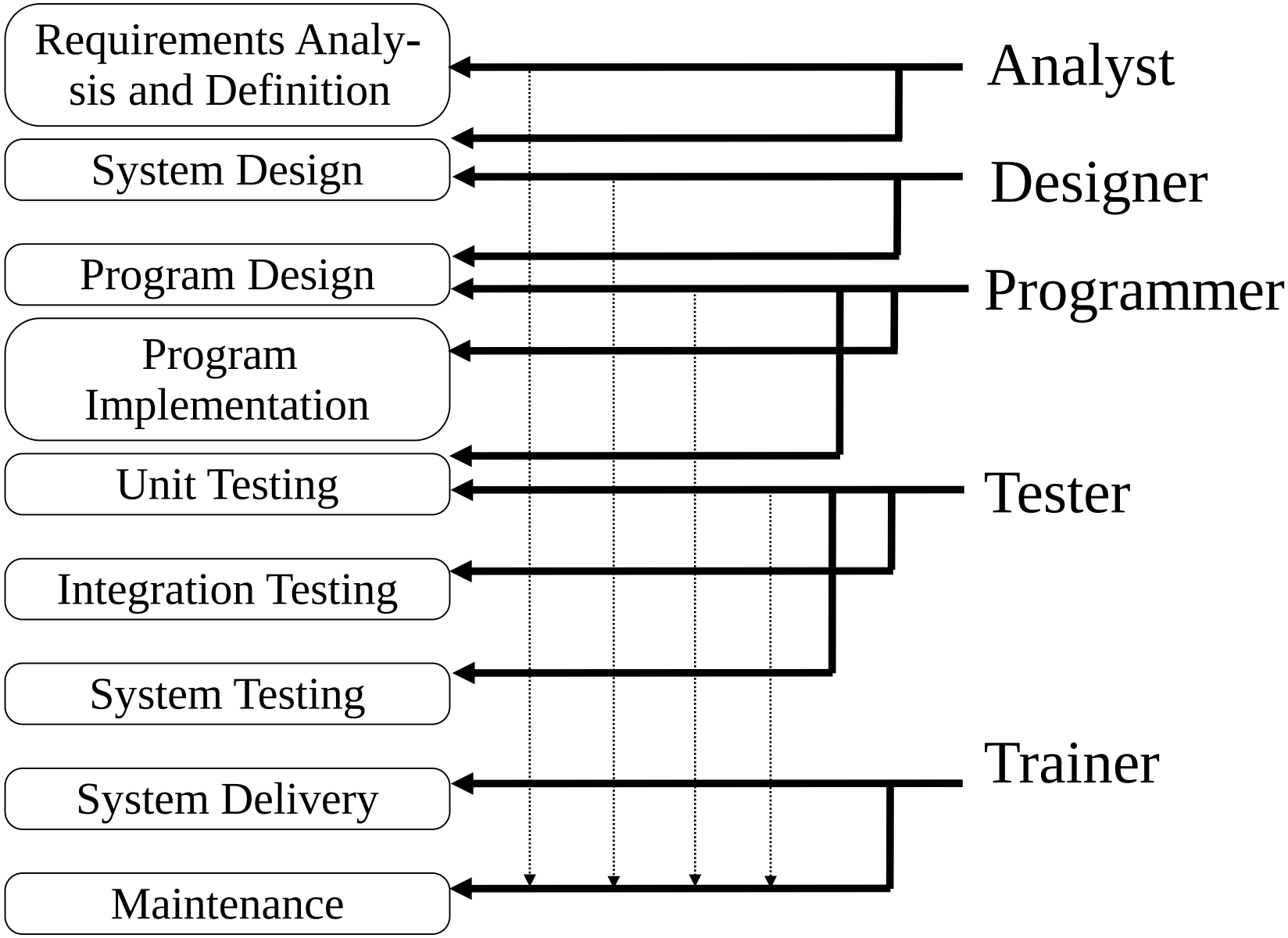
Integration Testing

System Testing

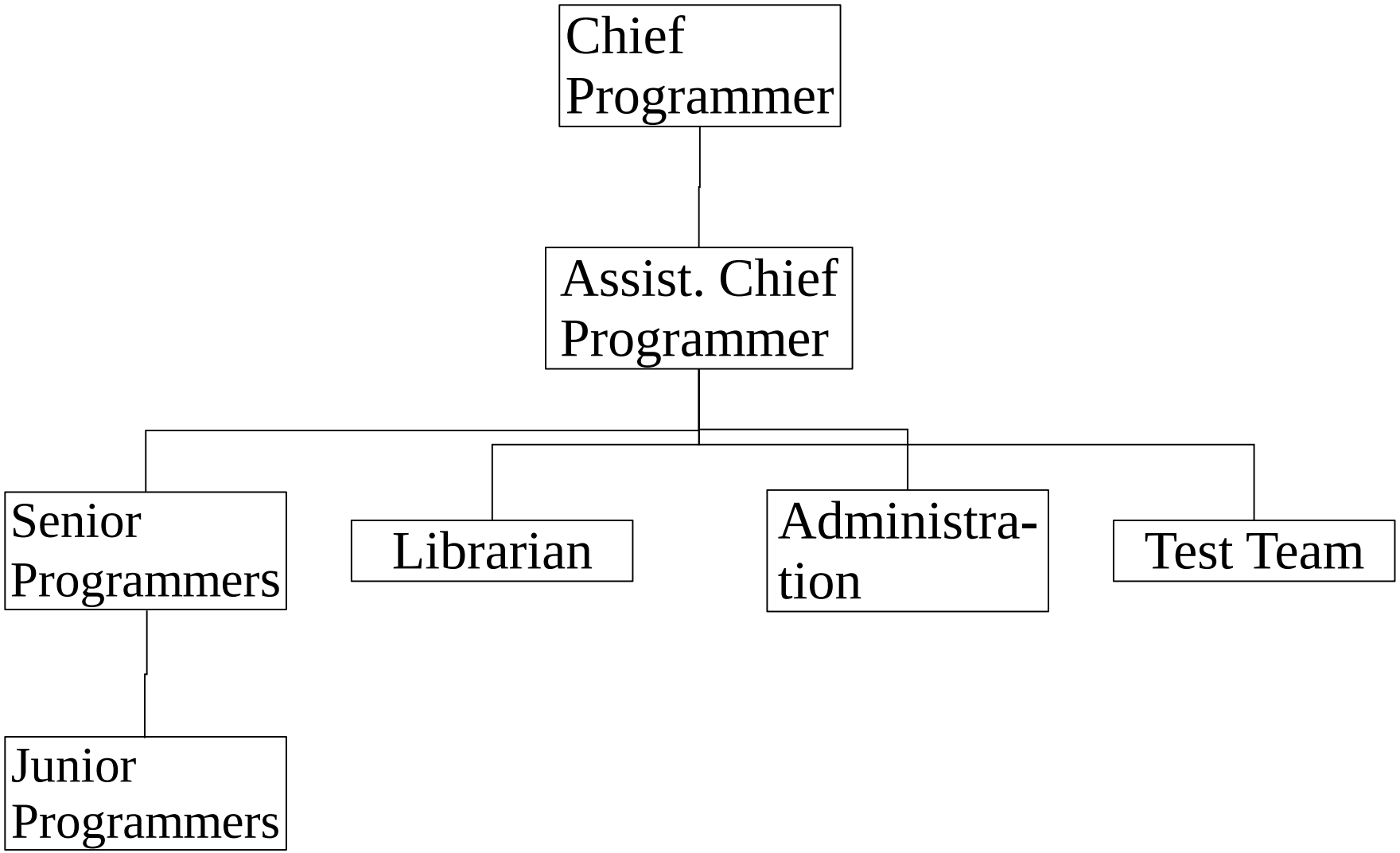
System Delivery

Maintenance

Software Developer Roles



Programming Team Organization



Other Software Personnel

- **Librarians: Prepare and store documents used during the life of the system:**
  - Requirements specifications
  - Design descriptions
  - Program documentation
  - Training manuals
  - Test data and schedules, etc.
- **Configuration Managers:**
  - Maintain cross references among requirements, design, implementation, and tests
  - Coordinate different versions of software across platforms, and from one release to another

The Future of Systems and Software

- **Eight surprise-free trends**
  1. Increasing integration of System Engineering and Software Engineering
  2. User/Value focus
  3. Software Criticality and Dependability
  4. Rapid, Accelerating Change
  5. Distribution, Mobility, Interoperability, Globalization
  6. Complex Systems of Systems
  7. COTS, Open Source, Reuse, Legacy Integration
  8. Computational Plenty

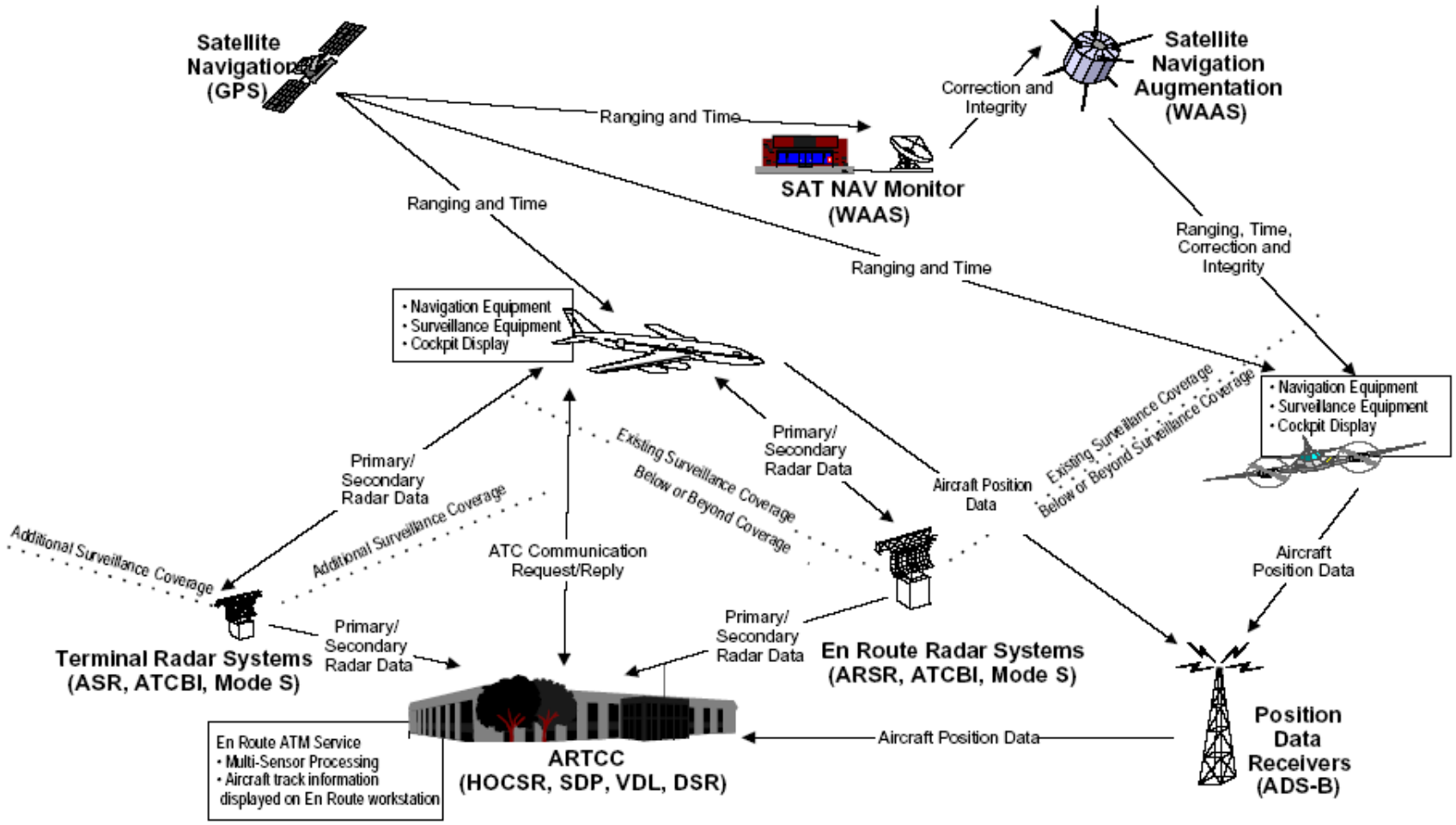
- **Two wild-card trends**
  9. Autonomy Software
  10. Combinations of Biology and Computing

Globalization: “The World is Flat”

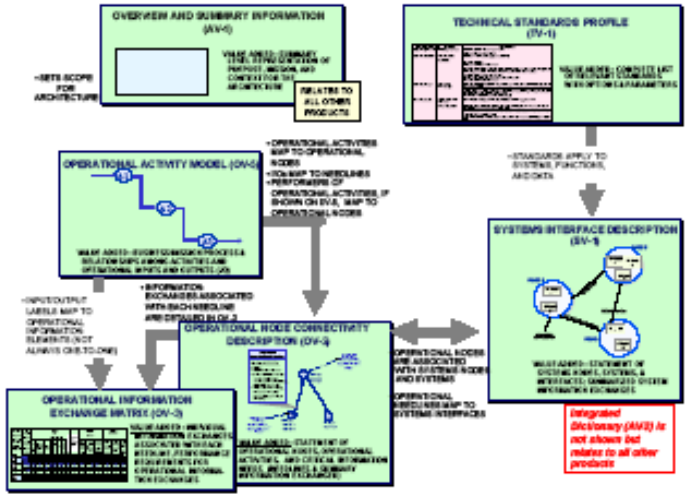
- Friedman, 2005

- **Information processing functions can be performed almost anywhere in the world**
  - **Low-cost global fiber-optic communications**
  - **Overnight global delivery services**
- **Significant advantages in outsourcing to low-cost suppliers**
  - **But significant risks also**
- **Competitive success involves pro-actively pursuing advantages**
  - **While keeping risks manageable**

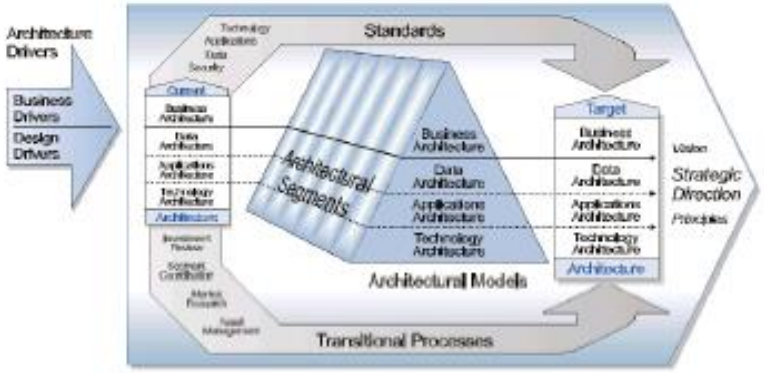
What does a SISOS look like?
- Network-Centric Air Traffic Control



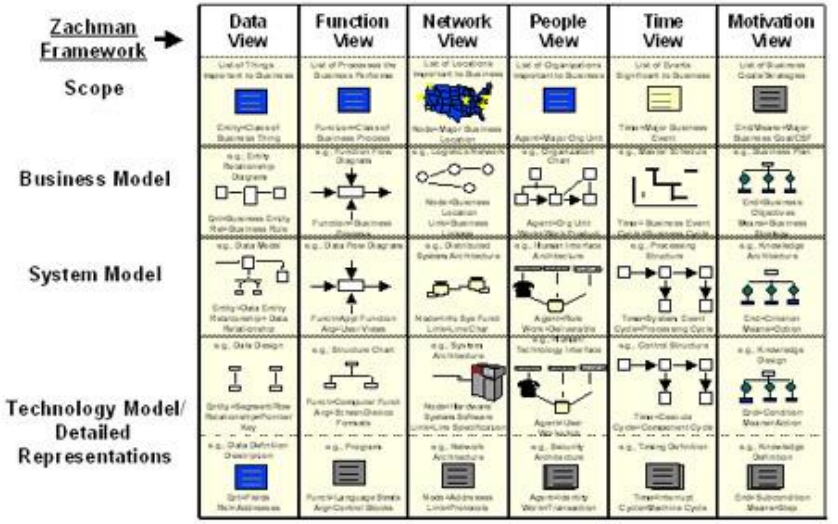
Integrated Enterprise Architectures



DOD Architectural Framework (DODAF)



Federal Enterprise Architectural Framework (FEAF)

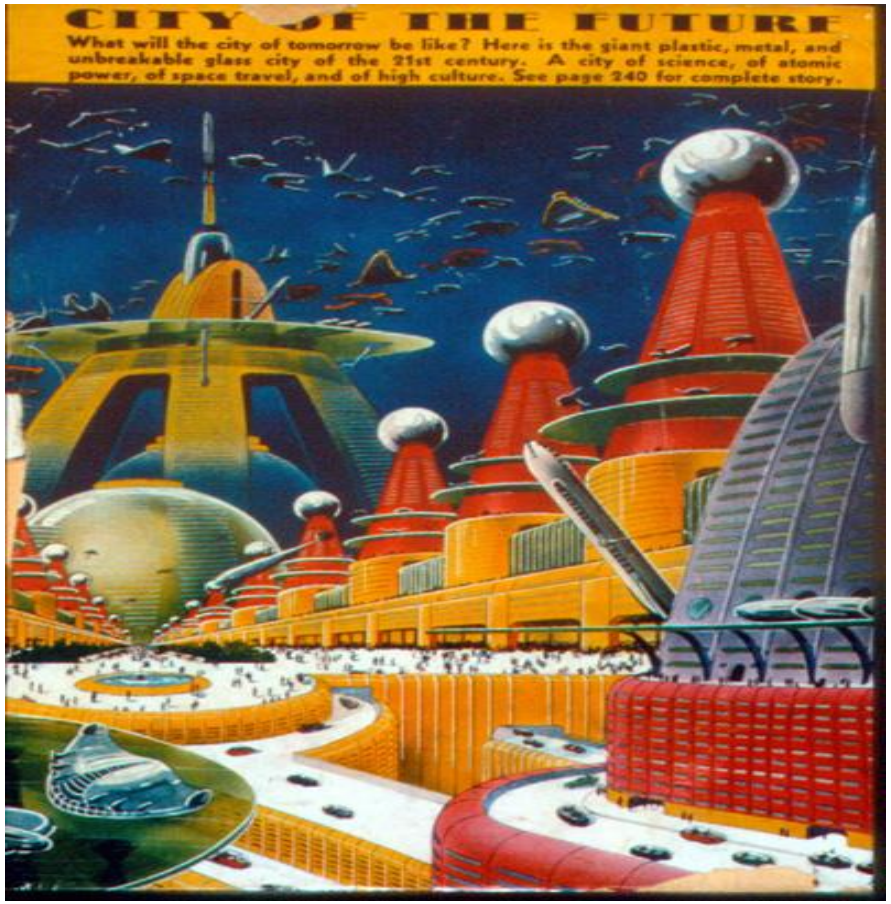


Zachman Framework

Persistence of Legacy Systems

- **Before establishing new-system increments**
  - **Determine how to undo legacy system**

1939's Science Fiction World of 2000



Actual World of 2000

Computational Plenty: Process Implications

- **New platforms:** smart dust, human prosthetics (physical, mental)
  - New applications: sensor networks, nanotechnology
- **Enable powerful self-monitoring software**
  - Assertion checking, trend analysis, intrusion detection, proof-carrying code, perpetual testing
- **Enable higher levels of abstraction**
  - Pattern programming, programming by example with dialogue
  - Simpler brute-force solutions: exhaustive case analysis
- **Enable more powerful software tools**
  - Based on domain, programming, management knowledge
  - Show-and-tell documentation
  - Game-oriented software engineering education

Wild Cards: Autonomy and Bio-Computing

- **Great potential for good**
  - Robot labor; human shortfall compensation
    - 5 Senses, healing, life span, self-actualization
  - Adaptive control of the environment
  - Redesigning the world for higher quality of life
    - Physically, biologically, informationally

- **Great potential for harm**
  - Loss of human primacy: computers propose, humans decide
  - Over-empowerment of humans
    - Accidents, terrorism, Enron California brownouts
  - New failure modes: adaptive control instability, self-modifying software, commonsense reasoning, bio-computer mismatches
  - V&V difficulties: cooperating autonomous agents, biocomputing

- **Forms and timing of new capabilities still unclear**