COMPUTER SCIENCE

02- Overview, products & processes

Rick Adrion

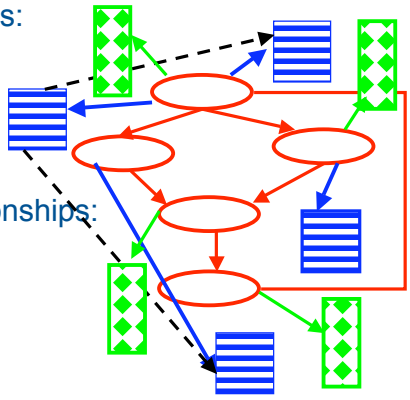
COMPUTER SCIENCE

The nature of software

- software is a complex, intricately interconnected data aggregate
- software development is the process of creating such a complex product, while continuously assuring that it remains consistent
- software engineering combines some of the approaches of classical engineering with some of the abstract approaches of mathematics

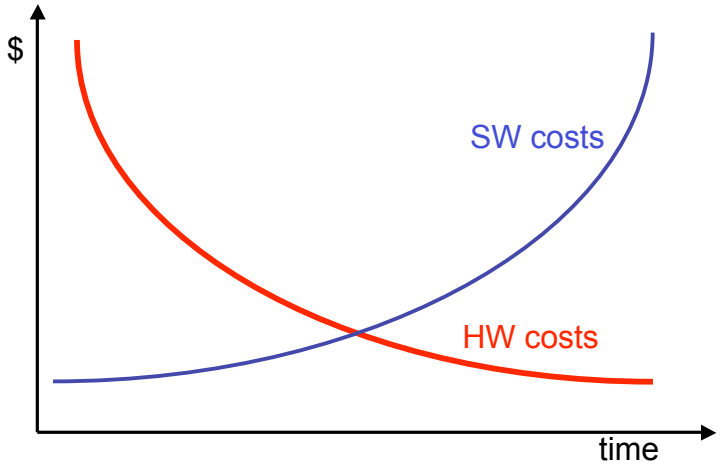
What is software?

- a software "product" is a complex web of intertwined software objects, connected by a multitude of diverse relations and constraints
- some types of objects:
  - source code
  - designs
  - test cases
  - documentation
- some types of relationships:
  - is invoked by
  - is derived from
  - is consistent with
  - is a version of



Hardware versus Software

- hardware costs are decreasing and software costs are increasing



Hardware versus Software

- once upon a time software was flexible and hardware was difficult to change.
- now software is brittle and expensive to change and maintain, while hardware has become much easier to design due to advances in CAD, ASICs, and FPGAs.
- even drivers -- software intended to be customized to hardware and be replaced -- are obstacles to success of new hardware. (See ATM vs. Gigabit Ethernet.)
- moreover, distribution of this inflexible media in binary has dramatically reduced opportunities for innovation in instruction sets and compilers.

Dave Patterson, University of California at Berkeley

Hardware versus Software

- Is hardware development done better than software development?
  - Yes, but...
    - software systems tend to be more complex
    - tend to do new applications in software and well-understood applications in hardware
    - despite the use of more rigorous and systematic processes, hardware systems fail too

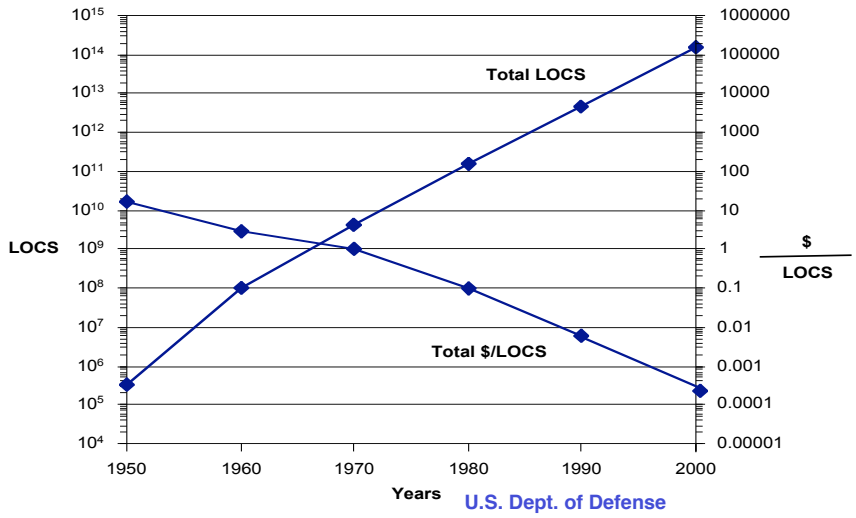
COMPUTER SCIENCE

What is novel about software?

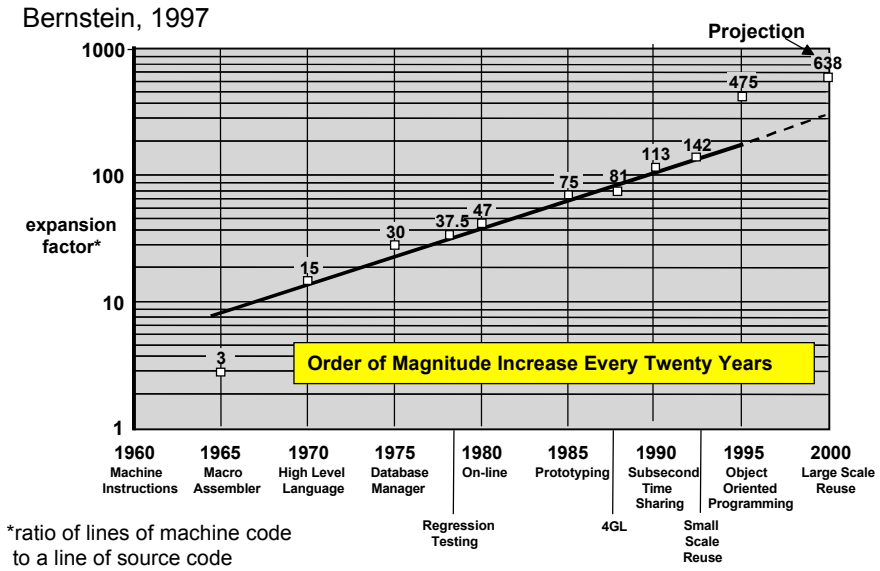
- product is unprecidentedly complex
- application horizons expand very fast--with human demands/imagination
- construction is human-intensive
- solutions require unusual rigor
- extremely malleable--can modify the product all too easily

COMPUTER SCIENCE

Lines of Code in Service



Trends in Software Expansion



Past Approaches

- Use more people
- Create "better" programming languages
- Design before writing code
- Test programs longer
- Train managers better
- Software tools to help people write programs better
- Use superior software processes
- Train people better

The silver lining?

- Struggling with hard technological problems can lead to good science
- Superficial Questions in Software
  - How can we prevent errors in software?
  - How can we deliver on time?
  - How can we keep our customers happy?
- Deeper Software Questions
  - What constitutes quality in software?
  - What is an error?Can errors be proven to be absent?
  - How can adding more people make things come out worse?
  - Is SW like anything else?
  - Is software based on any science?governed by any laws?

Our Strategy

- Start with engineering problems
  - look for deeper scientific questions
  - hypothesize conceptual foundations
  - find engineering solutions based on conceptual foundations
- Use experience with engineering solutions to advance the science:
  - validate hypotheses
  - new (sub-) hypotheses
  - revise hypotheses
  - suggest new issues and questions

Software Is not unique

- Studying such analogs can be useful:
  - Help us learn about computer software
  - Find points of similarity
  - Suggest successful approaches to be emulated
  - Avoid known mistakes

There are products that share some of its characteristics

Analogy 1: Custom Home Building

- Product Components
  - Customer needs
    - number, type and size of rooms, location, style,
    - what else?
  - Constraints
    - zoning, covenants, regulations
  - Preliminary design
    - architect sketches
    - plans from book
  - Detailed design
    - blueprints
  - Construction
    - Carpenters, plumbers, other skilled craftspeople
  - Maintenance & Evolution
    - instructions, manuals
    - additions/ remodeling as needed

Required Domain knowledge?

Problem: Create a product that provides shelter, sanitation, food preparation facilities, recreation

Solution Medium: Dwelling unit

Lots of interconnection and interaction among these components

Strengths of Analogy 1

- problem solving to meet real-world need
- single solution medium
- completed house is not whole solution, but only a component that
  - is constrained(zoning, etc.)
  - must "fit in" with utilities, neighbors, customer lifestyle
  - must evolve with resident (user) needs and changing environment

Weakness of Analogy 1

- Customer familiarity with
  - Solution medium
  - Descriptive terms
- Tangibility and Visibility of the
  - intermediate products
    - Changes are not always costly
  - final product
  - "well understood" application domain
- Too specific:
  - Custom: one need, one house; few stakeholders

Analogy 2: Legislation

- Product Components
  - "Customer" needs
    - Media, polls, reaction to political platforms, hearings, public opinion, reaction to legal decisions
    - what else?
  - Constraints
    - Constitution, Legislative Authority, previous legislation, court decisions
  - Preliminary design
    - Congressional staff drafts
    - Agency input
    - (sub) committee hearings
  - Detailed design
    - Congressional staff (re-) drafts
    - Legislative plan or blueprint
    - More Agency input
    - More (sub) committee hearings

Required Domain knowledge?

Problem: Create a product to meet needs of citizens or solve problems, such as Common defense, domestic tranquility, establish justice

Solution Medium: Congress, Executive Agencies, Petition,

Lots of interconnection and interaction among these components

Analogy 2: Legislation

- Product Components
  - Construction
    - Proposed Bill/Law
    - Amendments
    - Final Congressional & Executive approval
  - Maintenance & Evolution
    - Implementing bureaucracy
    - Amendments
    - Court decisions

Problem: Create a product to meet needs of citizens or solve problems, such as Common defense, domestic tranquility, establish justice

Solution Medium: Congress, Executive Agencies, Petition,

Lots of interconnection and interaction among these components

UNIVERSITY OF MASSACHUSETTS AMHERST • DEPARTMENT OF COMPUTER SCIENCE CMPSCI 520/620 FALL 2003

Strengths of Analogy 2

- Problem solving to meet real-world need
- Single solution medium
- Laws/agencies are not the solution, but only a component that
  - must "fit in" with existing laws & regulations; court decisions, agency implementation
  - must evolve with societal (user) needs and changing environment
  - errors must be corrected by amendment, substitution, promulgation, courts

Strengths of Analogy 2

- Inadequacy of notation, representation
  - Natural (although stylized) language
  - Interpretation varies: implementors (bureaucracies); courts
- Many stakeholders
  - effected citizens & (public, private) sectors; (federal, state & local) agencies & legislatures
- Complexity
  - Stakeholders unfamiliar with details
  - Side effects; unexpected outcomes
    - So called “wicked problem”
  - Seldom independent
  - Changes must be carefully planned
  - Assumed context for implementation

Weakness? of Analogy 2

- Look at two simple examples:
  - FY04 UMass budget
    - Governor
    - House Ways & Means
    - House 1 (amended)
    - Senate
    - Conference
    - Governor's vetos
    - Legislative overrides
  - Social Security "Windfall Elimination"
- What are other analogies?
  - Plays and Movies? Recipes? Driving instructions (eg. rallyes)

Required Domain knowledge?

Products?
Process?

What are? UMass budget/SS bill

- Customer needs
- Preliminary design
  - Domain knowledge
- Detailed design
- Construction
- Maintenance & Evolution

- And what is the process?

Key Features in Product Development

- Problem enunciation, understanding
  - What is the problem to be solved?
- Solution formulation
  - How might the problem be solved?
- Solution reduction to practice
  - How will the problem actually be solved?
- Solution implementation
  - The actual solution to the problem
- Evidence of consistency
- and **interconnections among all of these**

Product Component Types

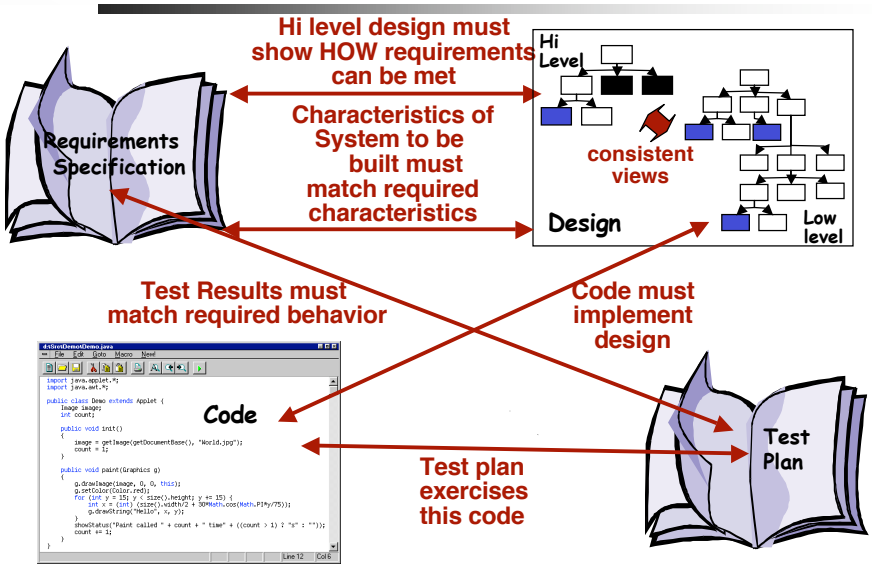
- Specification of customer/user needs/desires
  - Requirements
- Specification of potential solution or solution approach
  - Architecture
- Reduction of solution approach to practice
  - Design
- Solution
  - Implementation/Construction
- Evaluation of solution
  - Analysis/test results
- Changes/additions to solution
  - Maintenance & evolution

--and components interconnections with each other

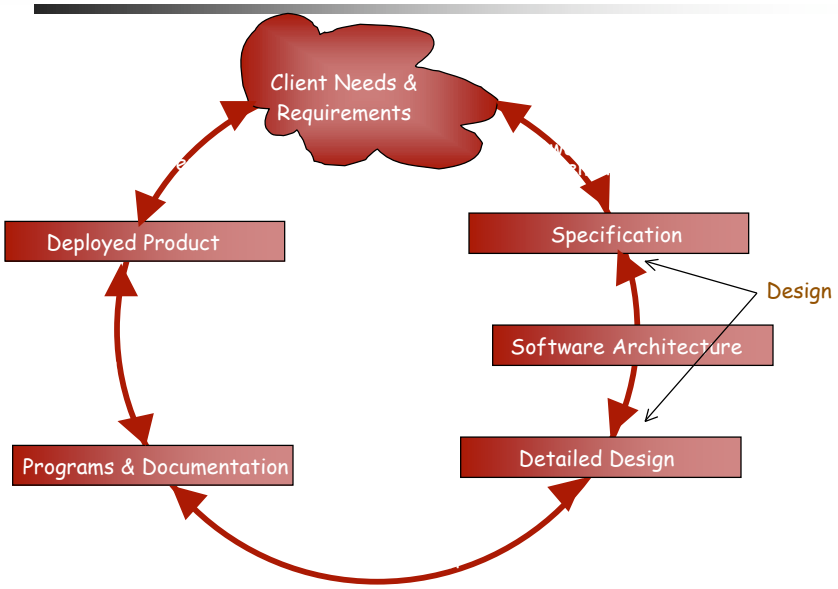
COMPUTER SCIENCE Applies Directly to Software Products

- Software -- ALL associated documents to assist with the development, operation, validation, and maintenance of programs/software systems
- Thus, a Software Product is a base of knowledge including:
  - problem specification (Requirements)
  - approach to its solution (Architecture)
  - solution approach & reduction to practice (Design)
  - solution itself (Code)
  - evidence that the solution "works"(Test and Analysis Results)
- And explicit interrelations specifying how all of the above most remain consistent

COMPUTER SCIENCE Products



Process



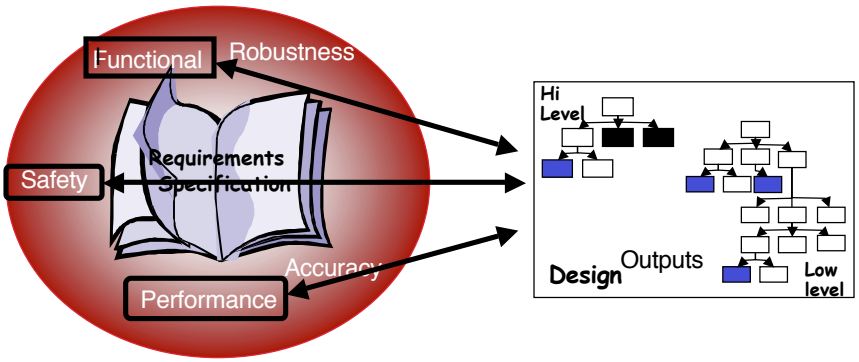
Requirements Analysis

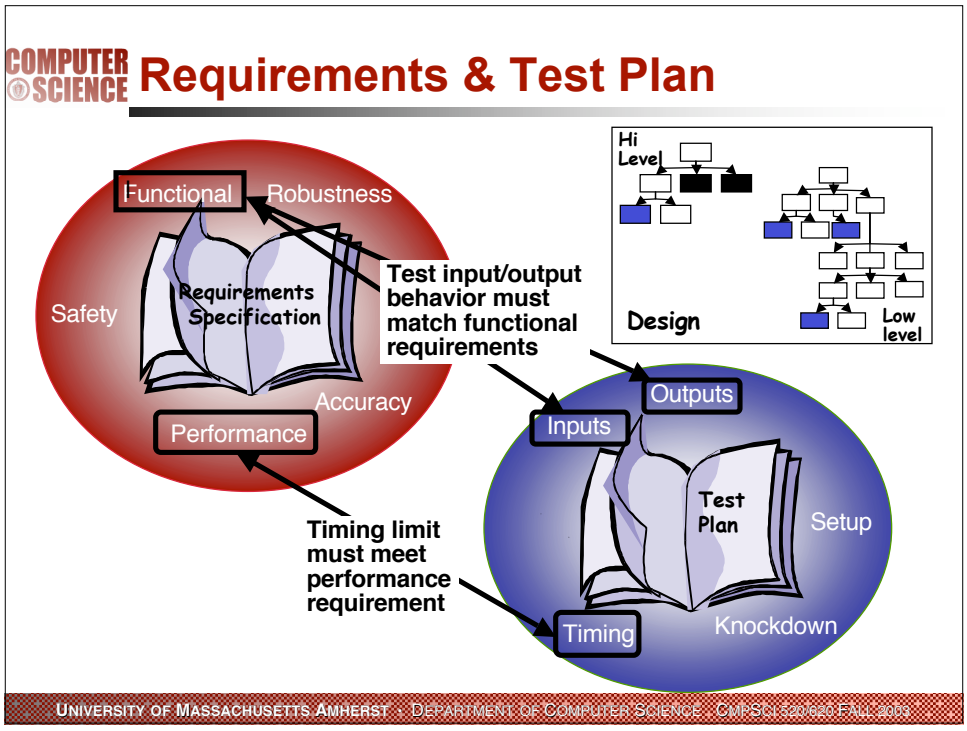
- Requirement
  - Condition or capability needed by a user to solve a problem or achieve an objective
  - Condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification or other formally imposed document
  - Documented representation of a condition or capability
- Functional requirements - functions that the system or a system component must perform
- Non-functional requirements - constraints, e.g., performance, UI, reliability, safety, security, portability, standards, economical & political aspects

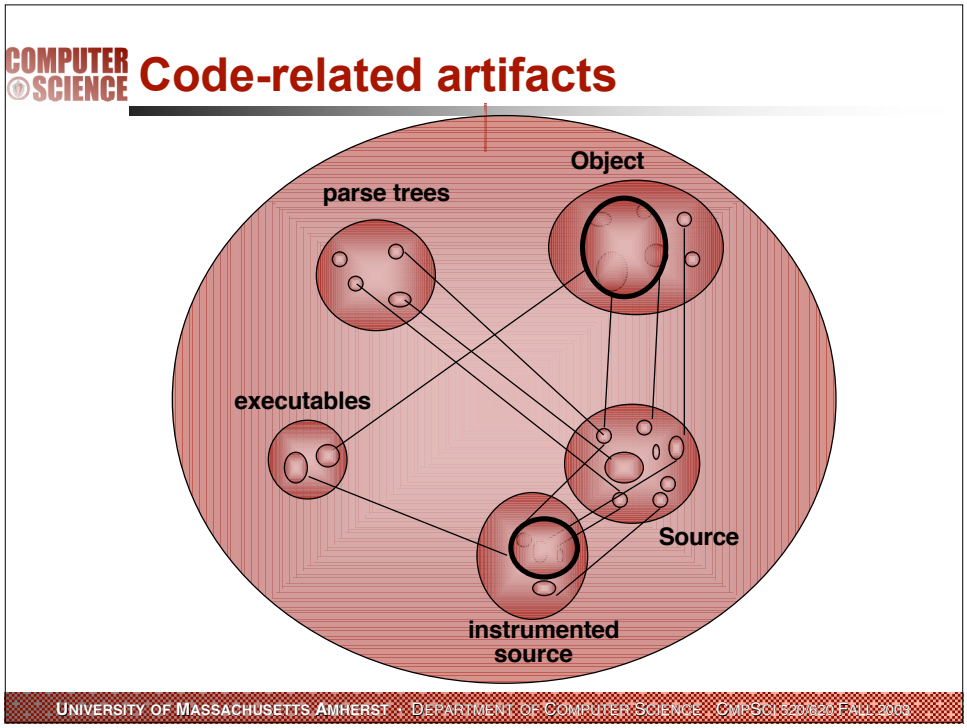
Interconnections

- Must assure that the components have the right relations to each other
  - Code implements design
  - Test execution results are consistent with expectations
  - Test data really represents expected usage
  - Proofs of concepts really connect proofs to concepts
- Desired/required interconnections specified early in project
  - Define what it means for product to be "correct"
- Interconnections validated as product is built
  - Testing/analysis/verification/validation
- Stakeholders view evidence of validation of interconnections

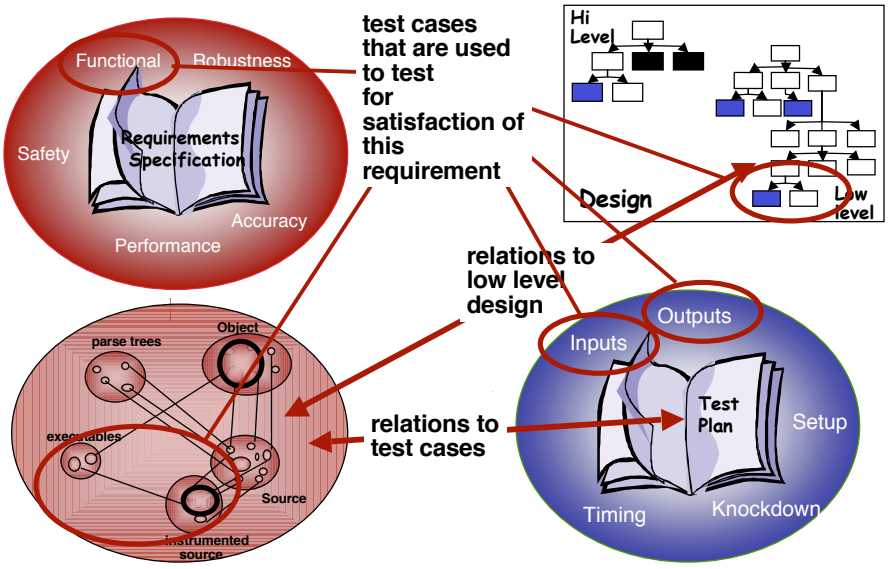
Requirements & Design







Interconnections



What Makes a Product "Good"?

- It meets the needs and expectations of all of its stakeholders
- Come back to the "ilities" we expect in systems

concept of stakeholders

- Stakeholders may include:
  - the person responsible for interacting with (or who is impacted by) the software—the customer, the "innocent bystander"
  - the person responsible for deploying the software—the end user
  - the person responsible for purchasing the software—the end users' management
  - the person responsible for managing the data repositories used by the system—the system administrator
  - the person responsible for modifying the runtime functions of the system—the developer
  - the person responsible for assuring appropriate use of the software—the regulators/monitors
  - the person responsible for approving new requirements for the system, etc

concept of stakeholders

- Stakeholders in legislative process:
  - the customer, the "innocent bystander" = citizens, clients, general public … others?
  - the end user = agency bureaucrat
  - the end users' management = agency heads
  - the system administrator = agency legal, IG; public attys, etc.
  - the developer = legislature (staff, representatives)
  - the regulators/monitors = courts; regulatory agencies, IGs, GAO, etc.

concept of stakeholders

- Stakeholders in SIS:
  - the customer, the "innocent bystander" = students, faculty, academic staff, …. others?
  - the end user = administrative staff, OIT
  - the end users' management = VPA&F, CIO
  - the system administrator = OIT
  - the developer = Peoplesoft, consultants, OIT, E*MPAC
  - the regulators/monitors = Registrar, Bursar, others?

What Are the Stakes?

- Customer:
  - Easy to get it to do what it is supposed to do
  - Cost (time to learn, ease of use) is consistent with benefits
- End User:
  - Does what it is supposed to do
  - Easy to learn, use, support and adapt (as customers' needs change)
  - Maintains job security
- Developer:
  - It is always "under control", and easy to modify
  - It **clearly** does what it is supposed to do
- Management
  - Progress is visible and satisfactory
  - Doesn't cost too much, or more than expected
  - Improves and optimizes business practices
- Innocent Bystander:
  - It does no harm

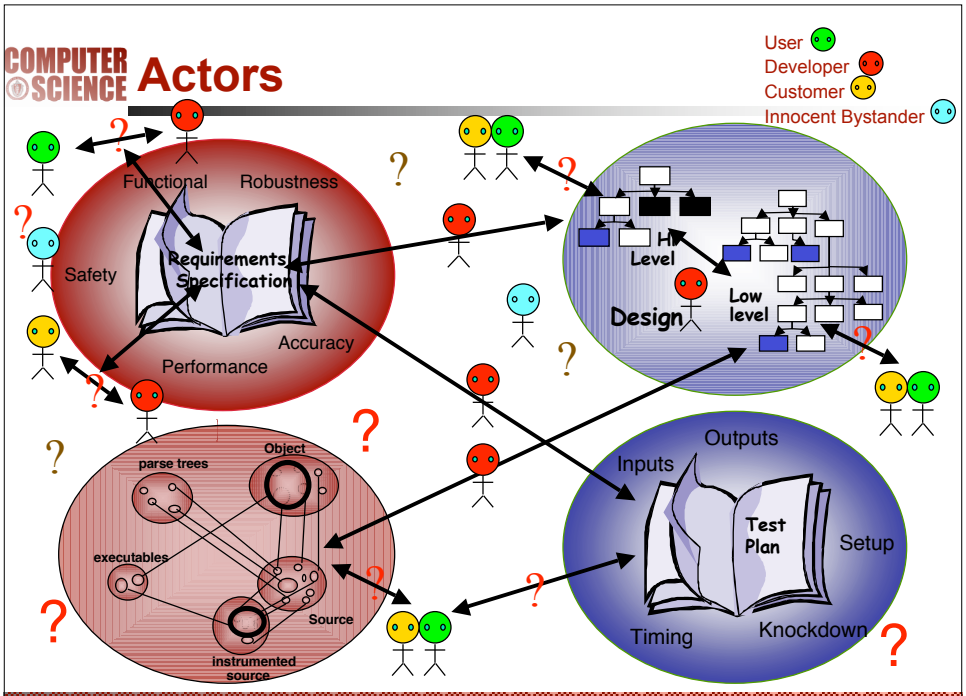
COMPUTER SCIENCE

Needs & objectives

- Meet business objectives
  - Current system
  - Future needs
  - System support
- Understand Context
  - People involved in implementation
  - People affected
- Keep projects under control
  - Being sure of the problem being solved
  - Estimating project costs
- Understand what quality means for a project
- Make needed repairs and changes easily

COMPUTER SCIENCE

Actors



What do Stakeholders Want to Know?

- Some examples of understandings needed:
  - What does the product do and how do we know
  - What is the product supposed to do
  - How does it work
  - What would happen if I did .....
  - Suppose we change ....
- There are infinitely many such questions
  - For each there are endless varieties of answers
  - The answers themselves form key parts of the product
- Superior products are tightly interconnected bundles of:
  - Component artifacts, used as the basis for:
    - These questions
    - Their answers
    - Solid basis for believing the answers

Process

- Need a process for:
  - Order of activities
  - Product delivery (what, when)
  - Assignment to developers
  - Monitoring ⇒ Measuring ⇒ Planning
- Cannot be (easily) codified or standardized
- Iterative and incremental

Software Processes

- What do people want to do with (to) a product?
  - Find out what it does (quickly, easily):
    UNDERSTANDING
  - Get it to do what is needed (quickly, easily):
    USAGE
  - Not worry about it:
    UNDERSTANDING, EVALUATION, STRESS TESTING
  - Build it (quickly, easily, at low risk):
    DEVELOPMENT
  - Change it as needed (quickly, easily)
    MODIFICATION
  - Improve it (quickly, easily):
    EVOLUTION

What are the key (sub-)processes it must support

Process

- Current Strong Emphasis on Process
  - Process and Product complement each other
- More past focus on product
- Process focus has been less technical
  - More managerial