02-Products/Processes/Stakeholders

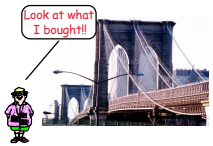
- Readings:
  - **Fundamentals of Software Engineering**, Ch. 2, 7
  - [ljO97] Leon J. Osterweil, "Software processes are software too, revisited: an invited talk on the most influential paper of ICSE 9," **Proceedings of the 19th International conference on Software Engineering** May 1997

Why engineer software?

- Scope and Impact on Society
  - New and Expanding Applications
  - Safety
  - Security and privacy
- Economics
  - A trillion dollar a year industry?
  - Slow to get technology to market
    - SRI->Xerox->Apple->Microsoft
    - Unix
  - Is the U.S. losing the innovation race?

Why engineer software?

- Quality
  - deliver prototypes & "leave it to the marketplace"
  - unsophisticated consumers
  - tolerate high failure rates
    - There are many recalls for automobiles; if something is not right, then it has to be fixed and usually for free.
    - A defect in software will be "fixed" in "the next" release (at some cost to upgrade) or as a part of "maintenance" (also at some cost)



- Other manifestations of poor quality:
  - optimizing software that is suboptimal
  - software that is too slow/costs time and money
  - software that is incomprehensible
  - software that is more trouble to use than it is worth

Barriers -- Industry's short term focus

- "bottom line orientation"
  - emphasis on development, time to market -- not life cycle
  - cost of capital, return on investment -- see "Made in America"
- startups cannot invest in R&D until product established in marketplace
  - without the R&D, takes too long for next or improved product
  - market niche strategy driven by investors
- software houses
  - intensely competitive
  - often don't use own technology
  - keep development cost down, fix later
- unsophisticated industries
  - lack of technical expertise
  - lack of administrative experience
- overselling the technology

Past Approaches

- Use more people
  - creates chaos, more problems
- Create "better" programming languages
  - bad programs can be written in any language
- Design before writing code
  - how do you get the design right?
- are you are designing the right program?
  - start by "baselining" requirements
  - but they inevitably change, what then?
- Test programs longer
  - find more bugs, but is that good?

More Approaches

- Train managers better
  - but how do you manage this kind of product?
- Software tools to help people write programs better
  - software tools are often bad software
  - people don't know how to use them
  - cost of capital issues
- Use superior software processes
  - how do you know when a process is “good”
- Train people better
  - what to teach them?

The silver lining? Struggling with hard technological problems can lead to good science

Our Strategy

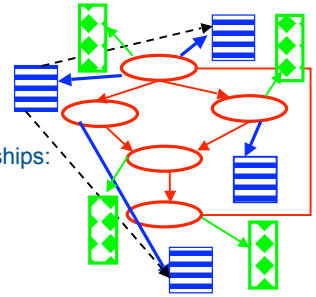
- Start with engineering problems
- Look for deeper scientific questions
  - e.g., What constitutes quality in software? What is an error?Can errors be proven to be absent? How can adding more people can make things come out worse?Is software based on any science?governed by any laws?
- Hypothesize conceptual foundations
- Base engineering solutions on conceptual foundations
- Use experience with engineering solutions to advance the science:
  - validate hypotheses
  - new (sub-) hypotheses
  - revise hypotheses
  - suggest new issues and questions

Products, Processes & Technologies

- Software Product
  - The complete set, or any of the individual items of the set, of computer programs, procedures, and associated documentation and data designated for delivery to a customer or end user. [IEEE-STD-610]
- Software Process
  - The process or set of processes used by an organization or project to plan, manage, execute, monitor, control and improve its software related activities. [ISO/IEC JTC1/SC7:15504-9]
- SoftwareTechnology
  - The theory and practice of various sciences (to include computer, cognitive, statistical sciences, and others) applied to software development, operation, understanding, and maintenance; more specifically, we view software technology as any concept, process, method, algorithm, or tool, whose primary purpose is the development, operation, and maintenance of software or software-intensive systems. [CMU/SEI-97-HB-00]

Software Products

- 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



Software Processes

- Activities related to the development, verification, evolution, evaluation, management, etc. of software products
- Examples of high level processes:
  - Develop requirements
  - Do Object Oriented Design
  - Formally verify
- Examples of low level processes
  - Archive result of running a test case
  - Compile a piece of code
- Tend to be concurrent (coordination of people, systems)
- Usually (regrettably) informal or undefined

Typical products and processes

- Customer needs
- Constraints
- Required Domain Knowledge

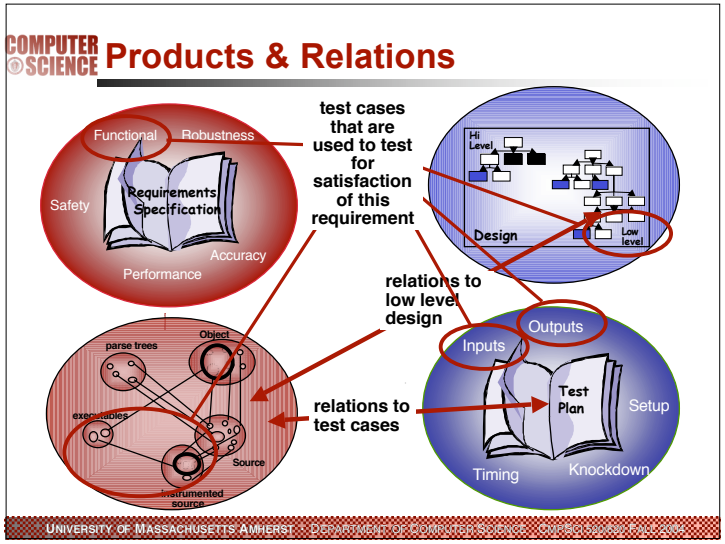
⇒ Requirements

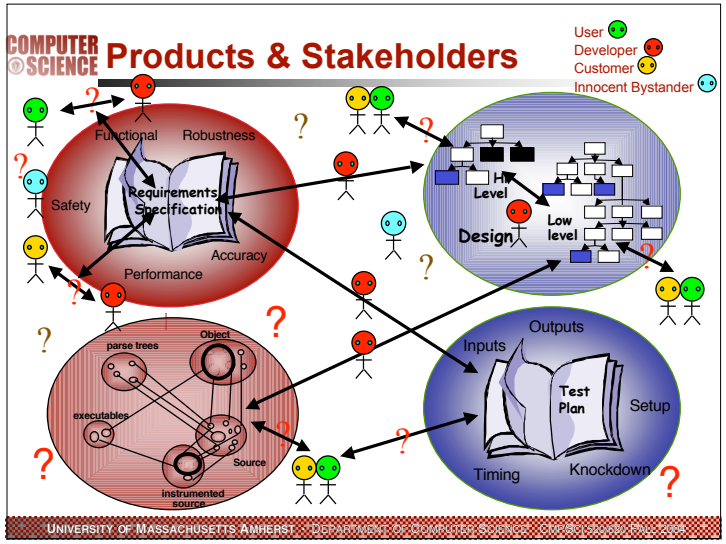
- Preliminary design
- Detailed design
- Construction
- Maintenance & Evolution

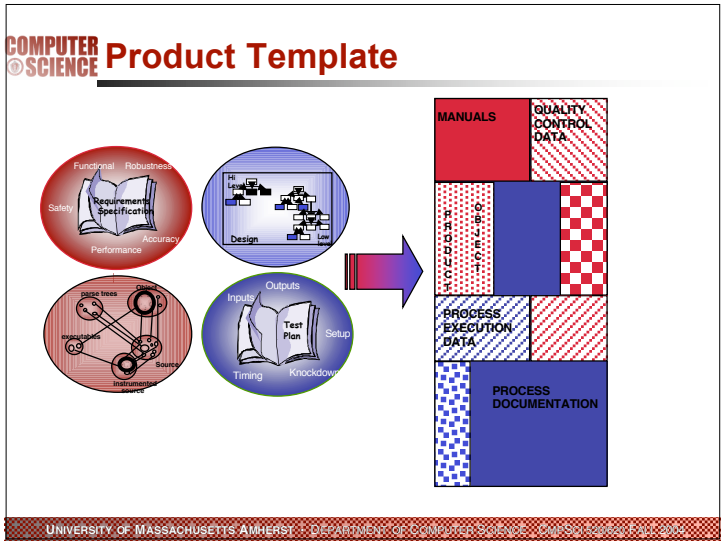
- Which are products and which are processes?

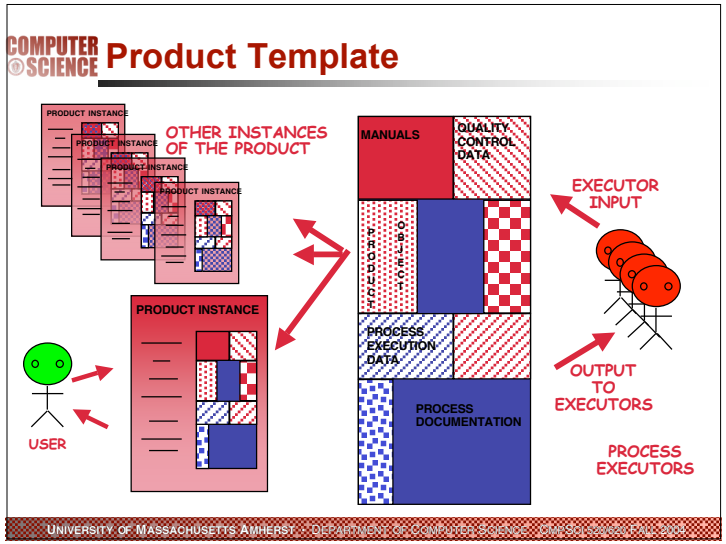
Stakes & Stakeholders

- Stakeholders may include: customer; "innocent bystander;" end user; end users' management; system administrator; developer; regulators/monitors; etc
- Stakes: ease of use; training; cost; cost/benefits; schedule; meets specs; optimizes business practices; does no harm; etc.
- A central concern of SE is to manage the creation and maintenance of a software product that satisfies all needs of all stakeholders
  - Implies understanding who stakeholders are; what questions they need answered; to what degree of thoroughness

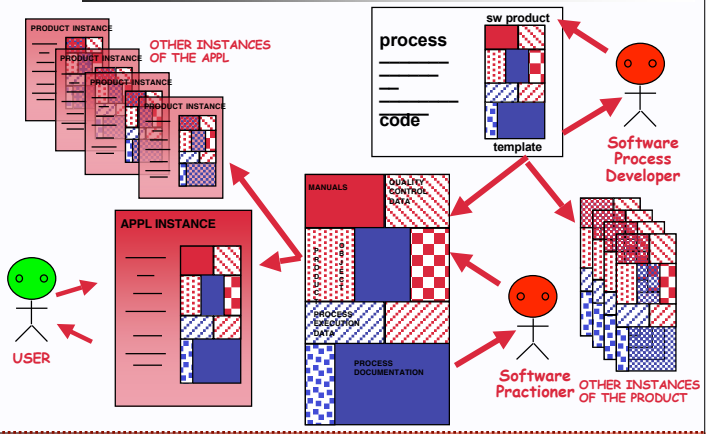








Software Process Instantiation



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

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: implementers (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:
  - FY05 State budget (funding, outside sections)
    - 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?

Example

University of Massachusetts For the operation of the University of Massachusetts; provided, that notwithstanding any general or special law to the contrary, the university may establish and organize auxiliary organizations, subject to policies, rules and regulations adopted by the board, to provide essential functions which are integral to the educational mission of the university; provided further, that notwithstanding any general or special law to the contrary, the university may enter into leases of real property without prior approval of the division of capital asset management and maintenance; provided further, …; provided further, that no funds appropriated in this item may be used for the issuance or renewal of student or employee identification cards which display a student or employee's social security number; provided further …; and provided further.. ..... $337,864,464

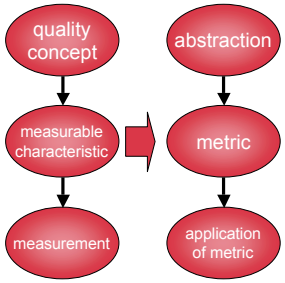
SECTION 242. Section 633 of chapter 26 of the Acts of 2003 is hereby amended by striking out the second and third paragraphs and inserting in place thereof the following paragraphs:-

Notwithstanding any general or special law to the contrary, the board of trustees for the university of Massachusetts system and the president of the university are hereby authorized and directed to establish a two year pilot program for out of state tuition retention at the flagship campus of the university at Amherst. The board shall promulgate… Notwithstanding any general or special law to the contrary, for employees of public higher education institutions who are paid from tuition retained pursuant to this section, fringe benefits shall be funded as if those employees' salaries were supported by state appropriations. …

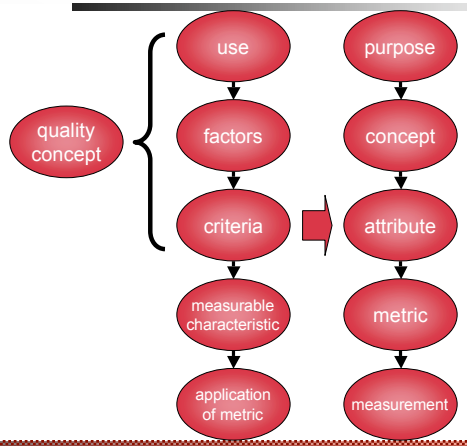
Another example

SECTION 1. WINDFALL ELIMINATION PROVISION RESTRICTED TO TOTAL MONTHLY AMOUNTS IN EXCESS OF $2,000. (a) IN GENERAL- Section 215(a)(7) of the Social Security Act (42 U.S.C. 415(a)(7)) is amended--
(1) in subparagraph (A), by inserting after `service'),' the following: `if the sum of the individual's primary insurance amount under paragraph (1) of this subsection and the portion of the monthly periodic payment which is attributable to noncovered service performed after 1956 (with such attribution being based on the proportionate number of years of such noncovered service) is greater than $2,000, then';
(2) in the second sentence of subparagraph (B)(i), by striking `(with such attribution being based on the proportionate number of years of such noncovered service)' and inserting `(as determined under subparagraph (A))';
(3) in the last sentence of subparagraph (B)(i), by striking `the larger of' and all that follows through `subsection (i))' and inserting the following: `the primary insurance amount determined under paragraph (1), reduced (before the application of subsection (i)) by the applicable percentage specified in clause (iii) of the excess of such amount over the larger of the two amounts computed under the preceding two sentences,'; and
(4) by adding at the end of subparagraph (B) the following new clause: `(iii) For purposes of clause (i), the applicable percentage in connection with any individual shall be the percentage specified in connection with such individual in the following table:

Quality



Quality - expanded



Need for metrics

- most "ilities" are qualitative, not quantitative
- how to determine
  - measurement ⇔ experimental
  - analysis ⇔ theoretical
  - simulation ⇔ compute/model

Software Qualities

- External product qualities
  - visible to the user of the system
  - reliability, robustness, performance (efficiency, usability, user-friendliness (human factors)), scalability
- Internal product qualities
  - affect the developers and maintainers
  - correctness (verifiability), maintainability (extensibility, repairability, reusability) portability (understandability, interoperability)

Software Qualities

- Process qualities
  - affect activities
  - productivity, timeliness, visibility

Process qualities ⇒ Internal qualities ⇒ External qualities

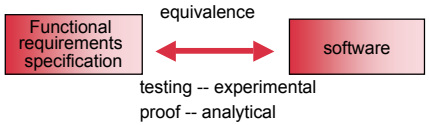
External Product Software Qualities

- reliability
  - "performs as expected" "depend on it"
  - too often release products with known "bugs"
  - SDI arguments
    - can't build correct system that complex -- Dave Parnas
    - doesn't have to be correct to be reliable -- Danny Cohen

most manufacturing ⇒ responsible for reliability
software industry ⇒ waive responsibility

External Product Software Qualities

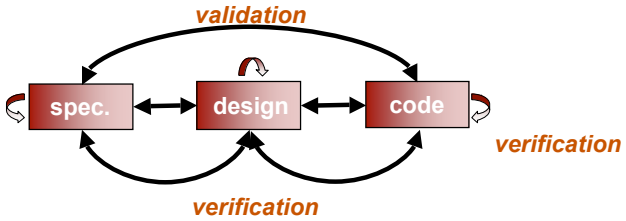
- robustness
  - performs "satisfactorily" even when environment & events unanticipated
- correctness



Reliability, Robustness & Correctness

- correct
  - does exactly what it is defined/specified to do
- reliable
  - does exactly what the user wants (expects?) it to do (under "normal" conditions)
- robust
  - does exactly what the user wants (expects?) it to do (under "abnormal" conditions)
- can apply to products and processes

Correctness as an Internal Product Software Quality



Verifiability

Verification	⇔ Correctness
Validation	⇔ Reliability (+ safety, security, etc.)
Certification	⇔ Legal & Contractual

More External Product SW Qualities

- Performance
  - efficient
    - produces results in an acceptable amount of time
  - usable -- end uses find it easy to use
    - easy to learn
    - easy to install
    - easy to operate
    - easy to advance

More External Product SW Qualities

- User-friendly
  - all of the above is provided with an easy to user interface
- Scalabilty
  - handle expansion in the parameters of the application

Internal Product Software Qualities

- maintainability = repairability + evolvability
  - can be modified and revalidated easily
  - enhanced by abstraction, modularity, discipline, standards, and good taste
  - maintenance is 60% of the lifecycle costs
- understandability
  - some products are inherently more complex than others

Internal Product Software Qualities

- Reusability
  - can be used to construct another product
  - need to plan for reuse
  - can involve any artifact or process
  - difficult
- Reusability factors
  - modularity
  - granularity (e.g., Unix, X windows)
  - trend is for plug-and-play components

Internal Product Software Qualities

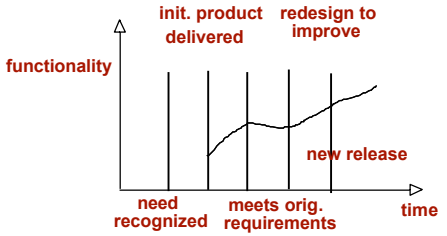
- Interoperability
  - can co-exist and cooperate with other systems
  - easy to integrate
    - open system & layered architectures
    - well-defined, standard interfaces
    - collection of independently written applications that cooperate and function as an integrated system
- Portability
  - can run on different environments (hardware or software platform) with little effort
  - enhanced by using only standard capabilities whenever possible and by isolating non-standard capabilities

Process Qualities

- Productivity
  - measures the performance of the development and maintenance activities
- Visibility
  - allows access to status of both the process and products
    - facilitates management
    - facilitates teamwork

Process Qualities

- Timeliness -- measures the ability to deliver software on time



High-level Goals of SE

- improve productivity
- reduce resources
  - e.g., time, cost, personnel
- improve predictability
- improve maintainability
- improve quality

Success of a system

- A system is judged not by properties of the hardware and software, but by the effects of the system in the world
  - you don't care how Caller ID works, just that it works
  - pilots love TCAS (on the whole) because it helps them fly more safely and easily—not because it has great data structures or a fascinating specification

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

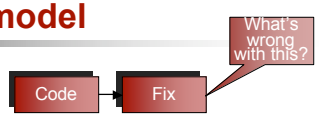
- Current Strong Emphasis on Process
  - Process and Product complement each other
- More past focus on product
- Process focus has been less technical
  - More managerial

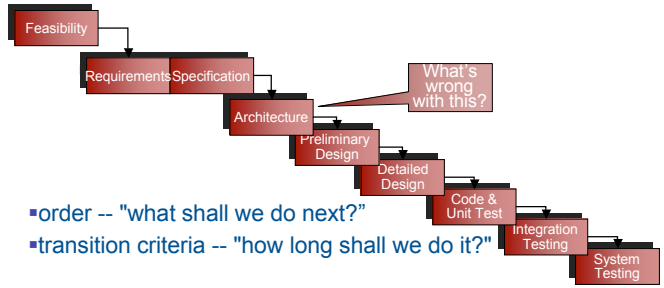
Traditional process models

- Code & test
- Waterfall
- Prototyping
- Transformational
- Spiral

Early waterfall model

- Earliest design and test

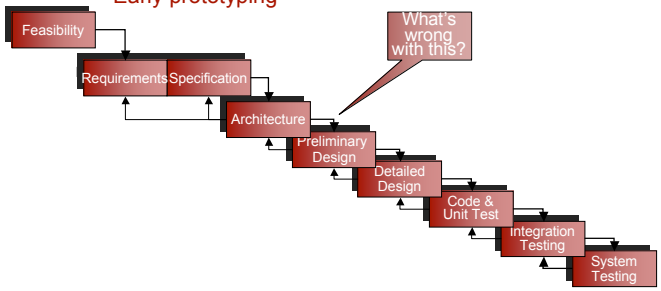




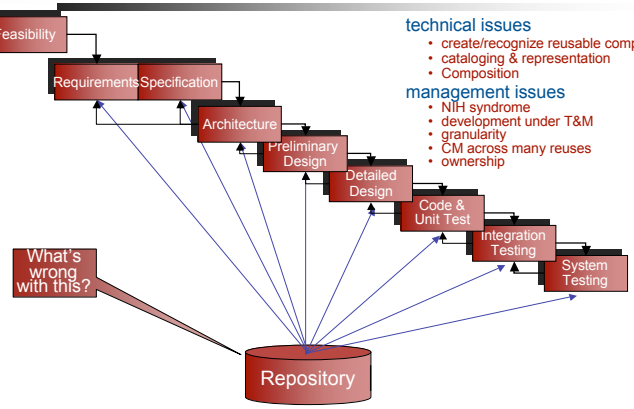
- order -- "what shall we do next?"
- transition criteria -- "how long shall we do it?"

1970's

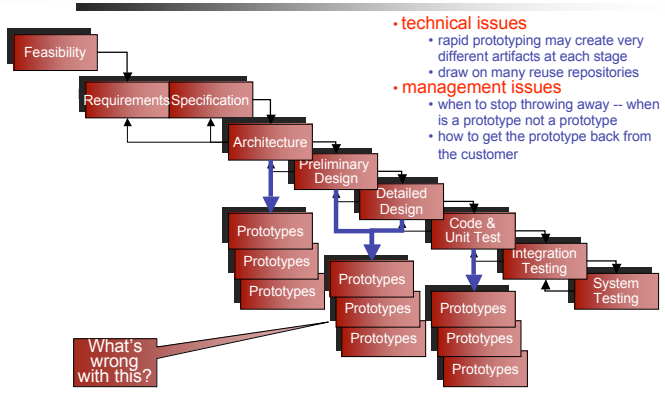
- Recognition of feedback loops
  - Confined to successive stages
- "Build it twice"
  - Early prototyping



Reuse model

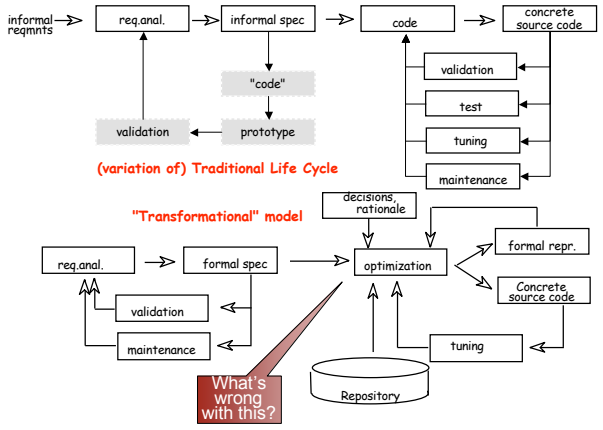


"Throwaway" prototyping

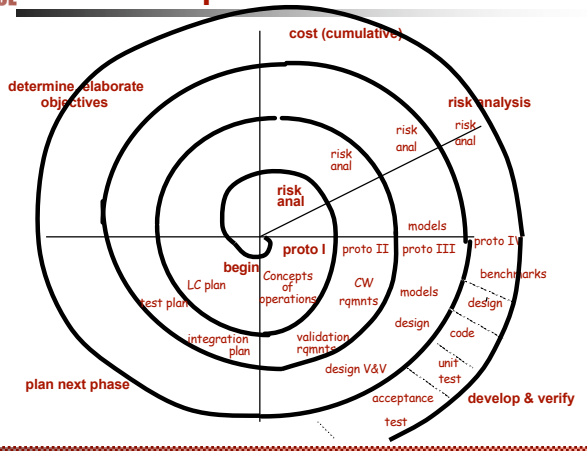


Evolutionary prototyping

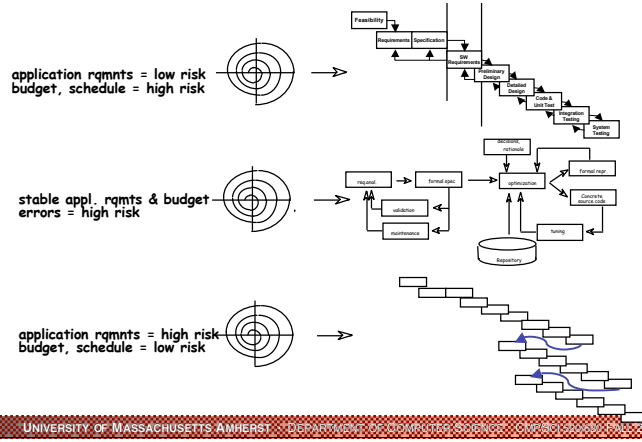
Transformational Model



Boehm's Spiral Model



Boehm's Spiral Model



Increase in cost

•to fix or change throughout lifecycle