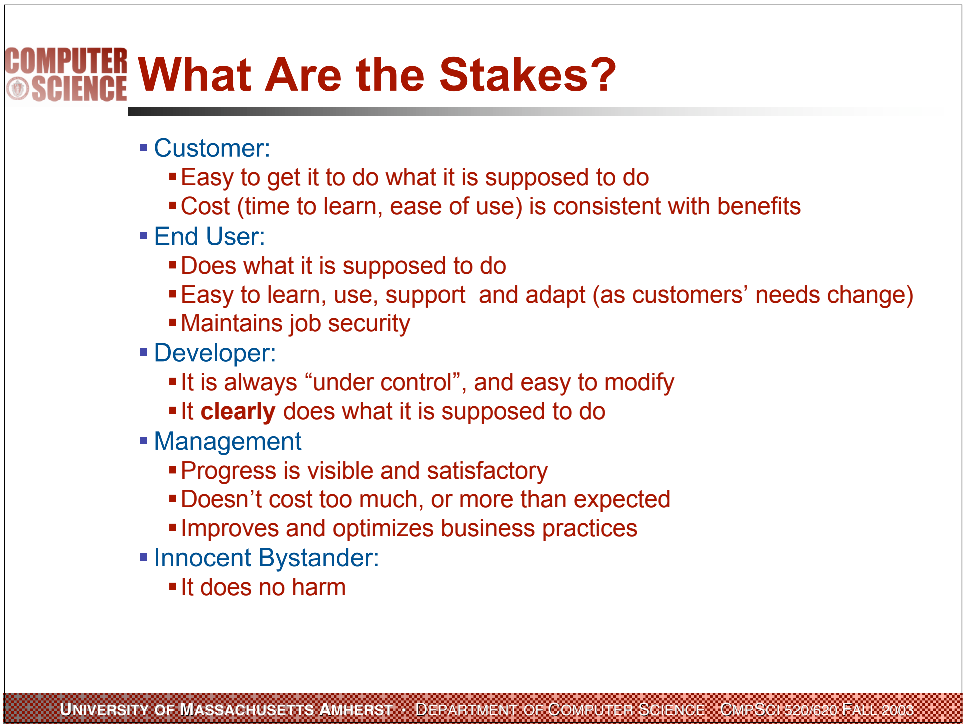
03 - Process

Rick Adrion

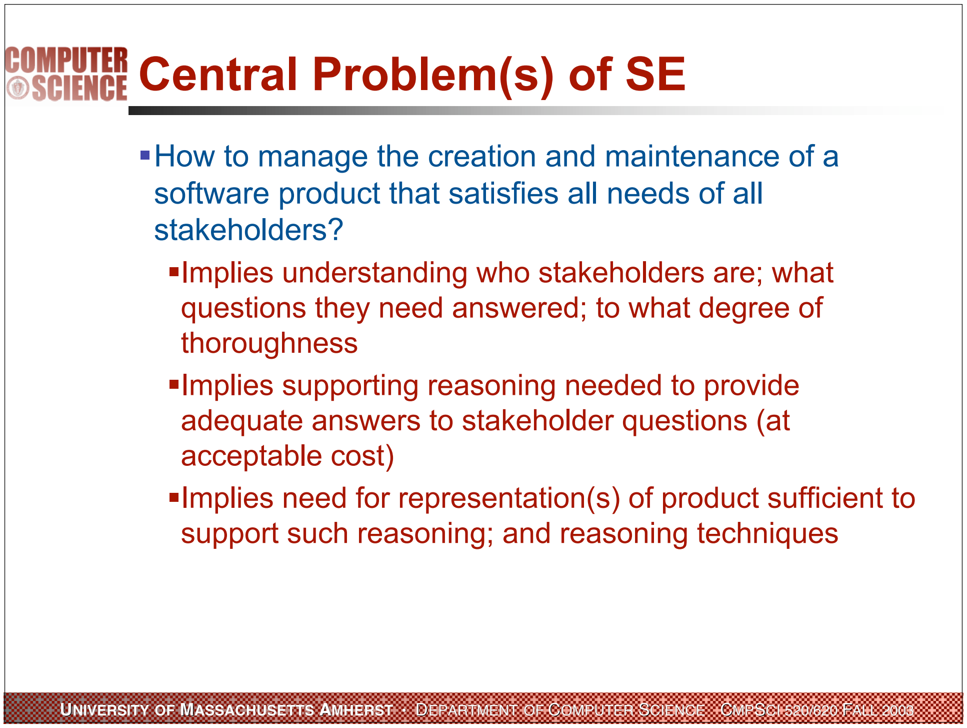
concept of stakeholders

- 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



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



Central Problem(s) of SE

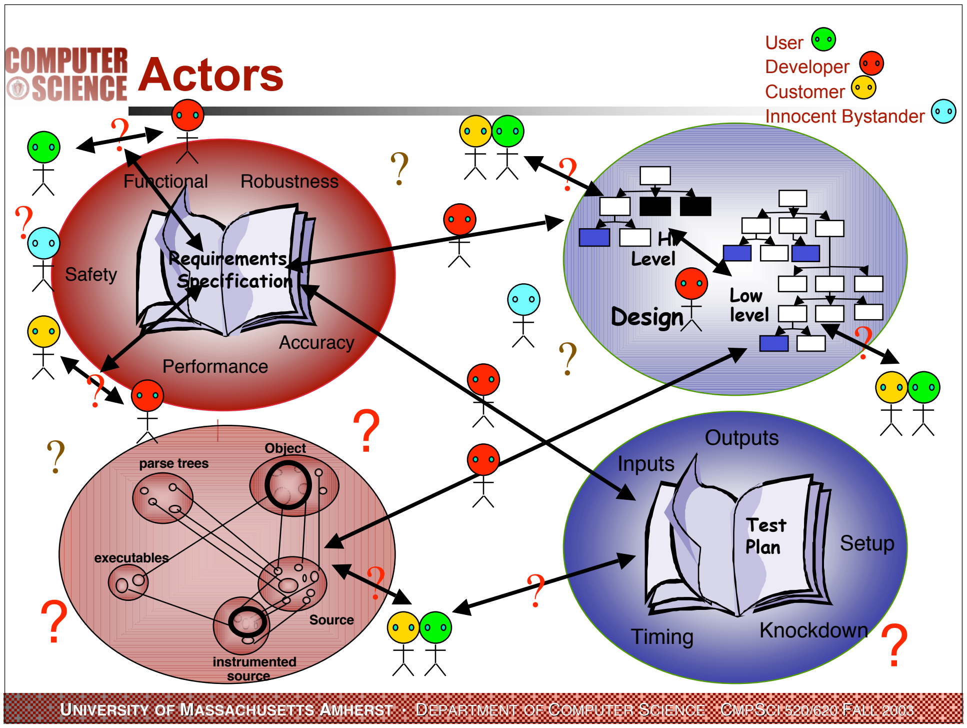
- How to manage the creation and maintenance of a software product that satisfies all needs of all stakeholders?
  - Implies understanding who stakeholders are; what questions they need answered; to what degree of thoroughness
  - Implies supporting reasoning needed to provide adequate answers to stakeholder questions (at acceptable cost)
  - Implies need for representation(s) of product sufficient to support such reasoning; and reasoning techniques

Interconnections Among Components

- Vehicles for answering stakeholder questions
  - Define what it means for product to be “correct”
- Ideally
  - Specified early in project
  - Created incrementally as product is built
  - Keep stakeholders satisfied continually
- Some examples
  - Test results must be consistent with expectations (ie. The manual is right)
  - Executables must correspond to source
  - Code must implement design
  - Lower levels of design must be elaborations of higher
  - Test data must really represent expected usage
  - Proofs of concepts really connect proofs to concepts

What do Stakeholders Want to Know?

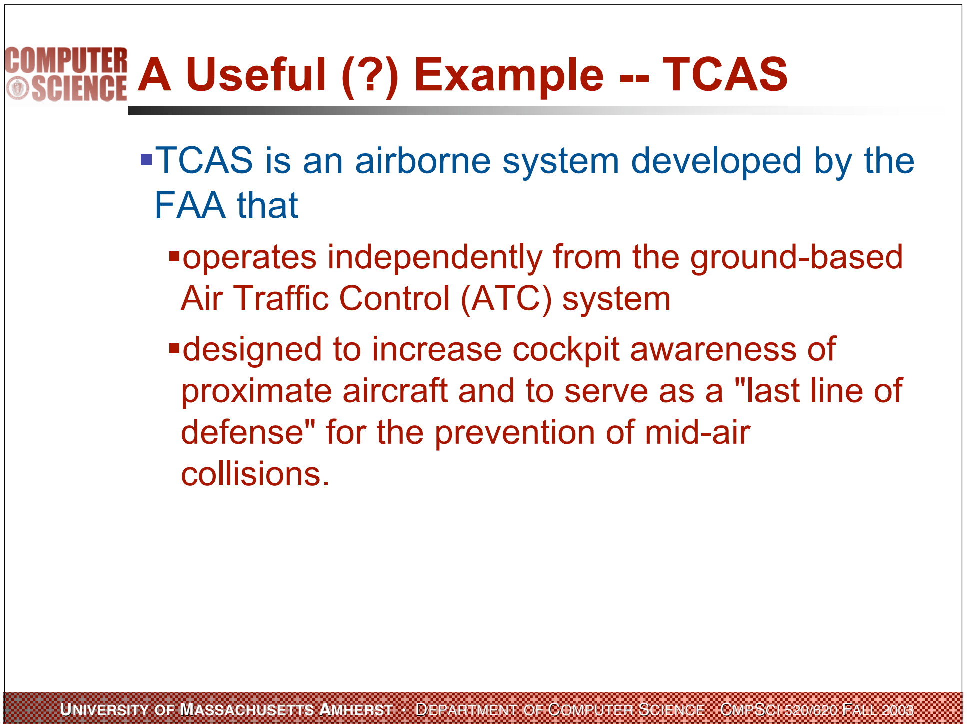
- Some examples of understandings needed:
  - What does the product do and how do we know
  - What is the product supposed to do
  - How does it work
  - What would happen if I did .....
  - Suppose we change ....
- There are infinitely many such questions
  - For each there are endless varieties of 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



Actors

User
Developer
Customer
Innocent Bystander

Functional
Robustness
Safety
Requirements Specification
Accuracy
Performance
Hi Level
Design
Low level
parse trees
Object
executables
Source
instrumented source
Outputs
Inputs
Test Plan
Setup
Timing
Knockdown
?



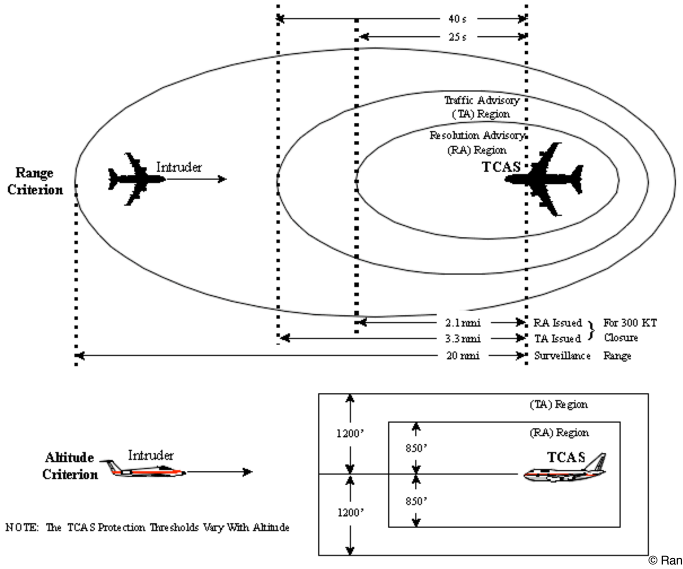
A Useful (?) Example -- TCAS

- TCAS is an airborne system developed by the FAA that
  - operates independently from the ground-based Air Traffic Control (ATC) system
  - designed to increase cockpit awareness of proximate aircraft and to serve as a "last line of defense" for the prevention of mid-air collisions.

Two levels of TCAS systems

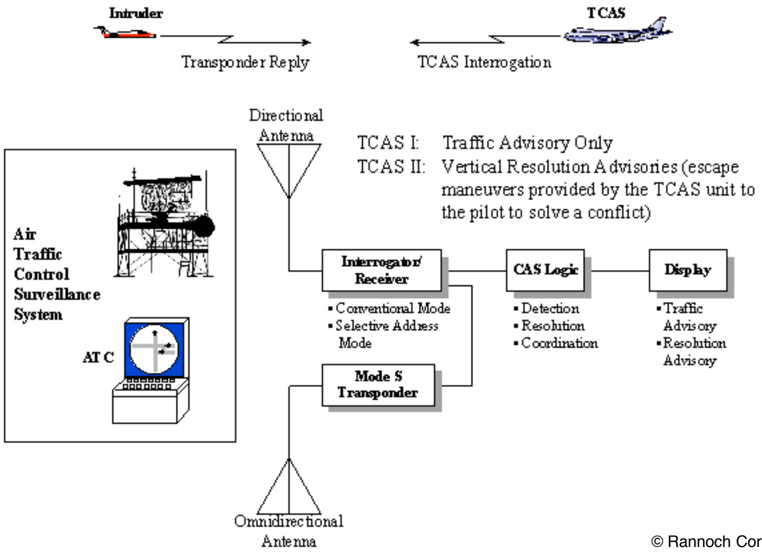
- TCAS I was developed to accommodate the general aviation (GA) community and the regional airlines.
  - issues ‘Traffic Advisories’ (TAs) to assist pilots in visual acquisition of intruder aircraft.
  - mandated on aircraft with 10 to 30 seats, although TCAS II may be installed instead.
- TCAS II is a more sophisticated system which provides the information of TCAS
  - analyzes the projected flight path of approaching aircraft and issues ‘Resolution Advisories’ (RAs) to the pilot to resolve potential mid-air collisions
  - required internationally in aircraft with more than 30 seats or weighing more than 15,000 kg.

TCAS overview


40 s
25 s
Traffic Advisory
(TA) Region
Resolution Advisory
(RA) Region
TCAS
Range
Criterion
Intruder
2.1 nmi
3.3 nmi
20 nmi
RA Issued
TA Issued
Surveillance
For 300 KT
Closure
Range
(TA) Region
(RA) Region
1200'
850'
850'
1200'
TCAS
Altitude
Criterion
Intruder
NOTE: The TCAS Protection Thresholds Vary With Altitude

© Rannoch Corporation 1998

TCAS Overview


Intruder
TCAS
Transponder Reply
TCAS Interrogation
Directional Antenna
Air Traffic Control Surveillance System
ATC
Interrogator/ Receiver
- Conventional Mode
- Selective Address Mode
CAS Logic
- Detection
- Resolution
- Coordination
Display
- Traffic Advisory
- Resolution Advisory
Mode S Transponder
Omnidirectional Antenna

TCAS I: Traffic Advisory Only
TCAS II: Vertical Resolution Advisories (escape maneuvers provided by the TCAS unit to the pilot to solve a conflict)

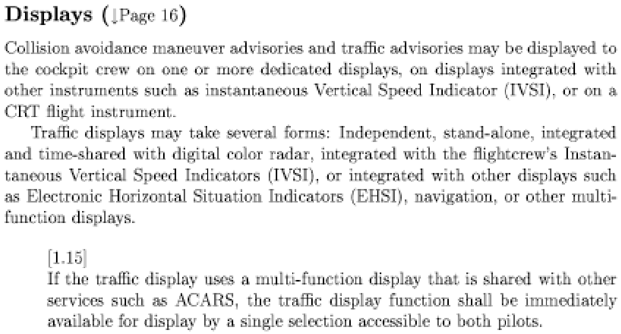
© Rannoch Corporation 1998

Stakeholders in TCAS

- the “innocent bystander” = passengers, people living near airports
- the customer & the end user = commercial and private pilots; air traffic controllers
- the end users’ management = airlines
- the developer = FAA, contractors (system, software), consultants
- the regulators/monitors = NSTB, FAA, Congress

Answers to Stakeholder Questions

- Problem of Providing Satisfactory answers
- Many important questions are vague
  - Is this software user-friendly? How to design appropriate pilot interfaces?



Displays (↓Page 16)

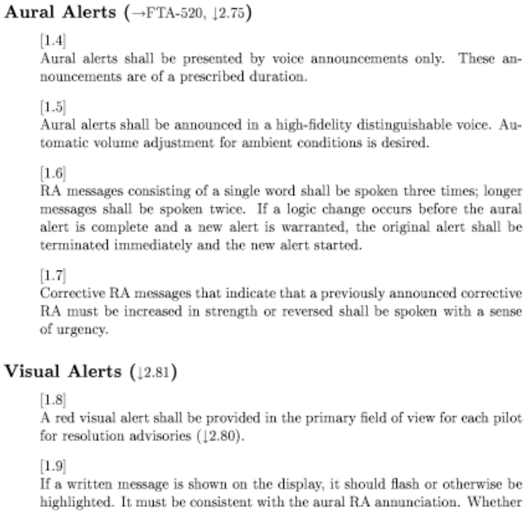
Collision avoidance maneuver advisories and traffic advisories may be displayed to the cockpit crew on one or more dedicated displays, on displays integrated with other instruments such as instantaneous Vertical Speed Indicator (IVSI), or on a CRT flight instrument.

Traffic displays may take several forms: Independent, stand-alone, integrated and time-shared with digital color radar, integrated with the flightcrew's Instantaneous Vertical Speed Indicators (IVSI), or integrated with other displays such as Electronic Horizontal Situation Indicators (EHSI), navigation, or other multi-function displays.

> [1.15]
> If the traffic display uses a multi-function display that is shared with other services such as ACARS, the traffic display function shall be immediately available for display by a single selection accessible to both pilots.

Answers to Stakeholder Questions

- Is this software user-friendly? How to design appropriate pilot interfaces?

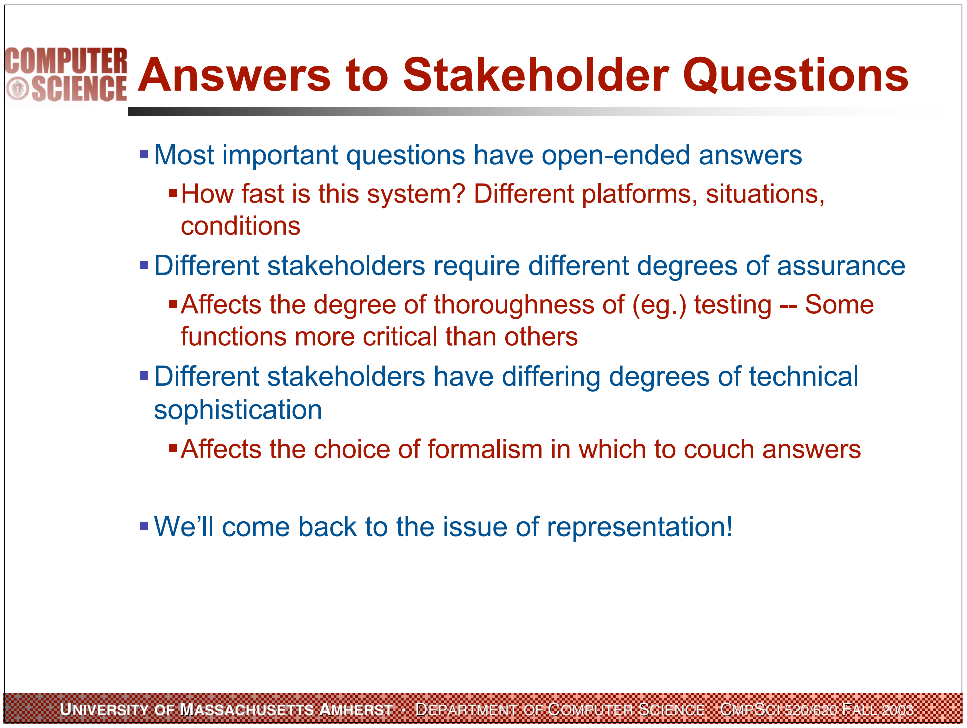


Aural Alerts (→FTA-520, ↓2.75)

> [1.4]
> Aural alerts shall be presented by voice announcements only. These announcements are of a prescribed duration.
>
> [1.5]
> Aural alerts shall be announced in a high-fidelity distinguishable voice. Automatic volume adjustment for ambient conditions is desired.
>
> [1.6]
> RA messages consisting of a single word shall be spoken three times; longer messages shall be spoken twice. If a logic change occurs before the aural alert is complete and a new alert is warranted, the original alert shall be terminated immediately and the new alert started.
>
> [1.7]
> Corrective RA messages that indicate that a previously announced corrective RA must be increased in strength or reversed shall be spoken with a sense of urgency.

Visual Alerts (↓2.81)

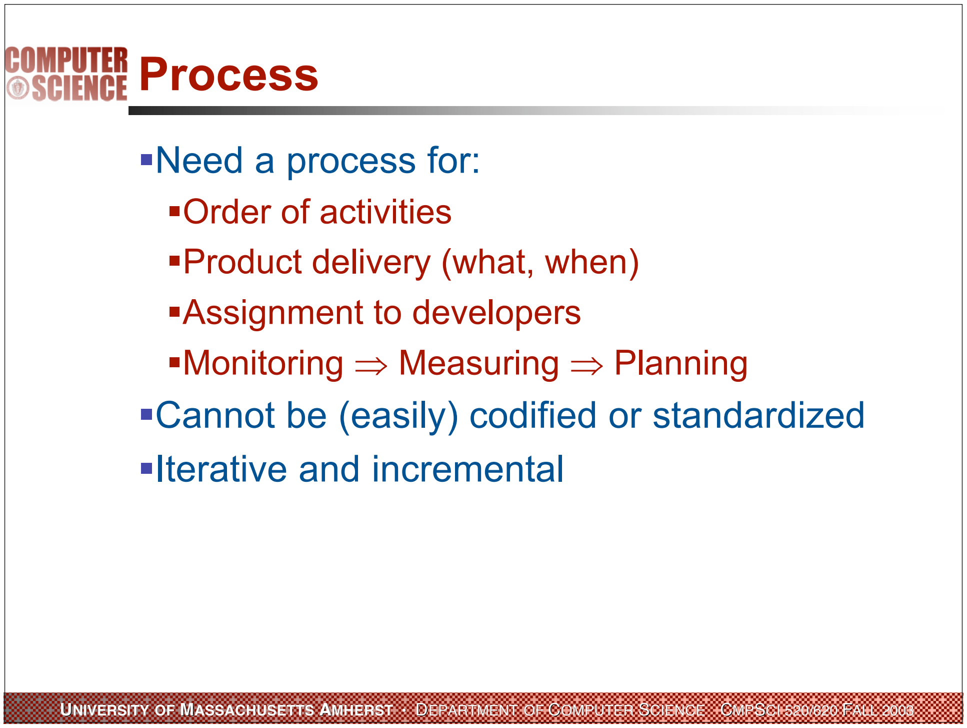
> [1.8]
> A red visual alert shall be provided in the primary field of view for each pilot for resolution advisories (↓2.80).
>
> [1.9]
> If a written message is shown on the display, it should flash or otherwise be highlighted. It must be consistent with the aural RA annunciation. Whether



Answers to Stakeholder Questions

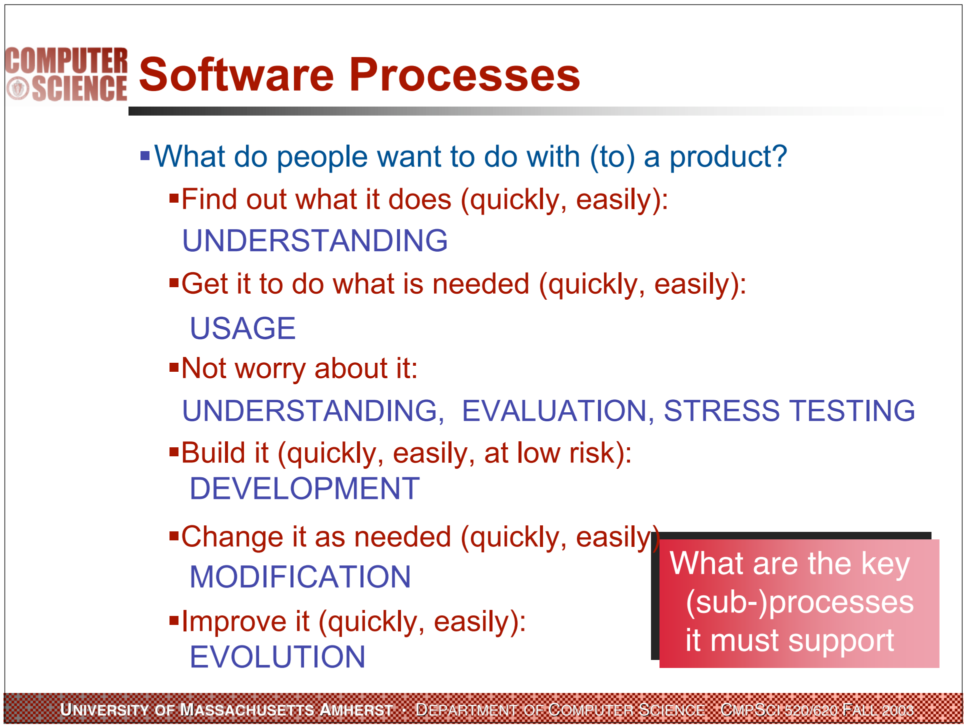
- Most important questions have open-ended answers
  - How fast is this system? Different platforms, situations, conditions
- Different stakeholders require different degrees of assurance
  - Affects the degree of thoroughness of (eg.) testing -- Some functions more critical than others
- Different stakeholders have differing degrees of technical sophistication
  - Affects the choice of formalism in which to couch answers

- We'll come back to the issue of representation!



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

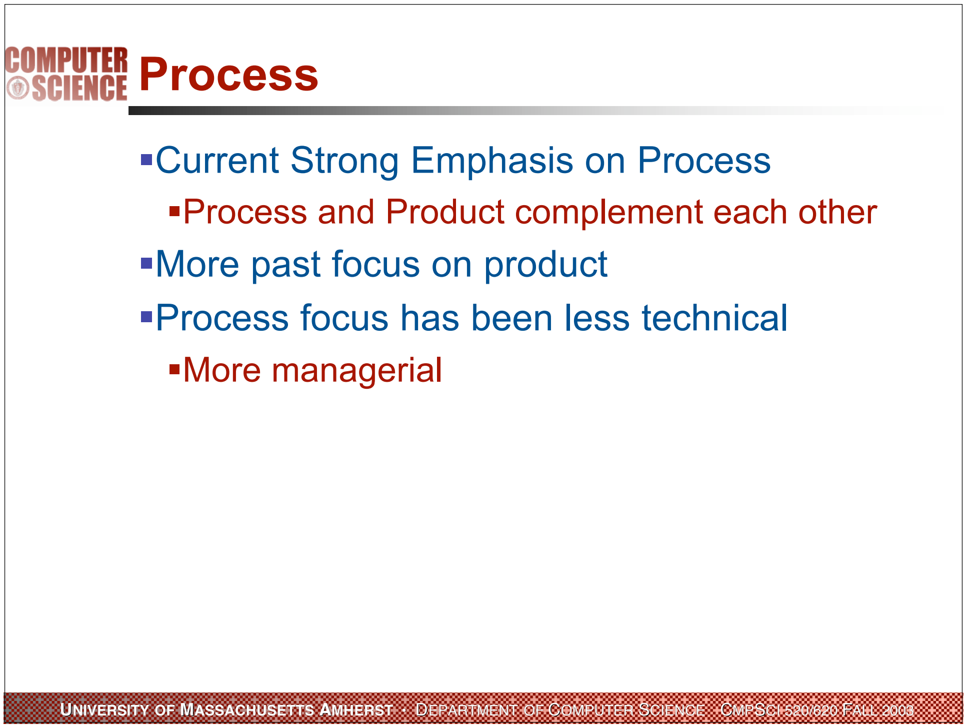


COMPUTER SCIENCE

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

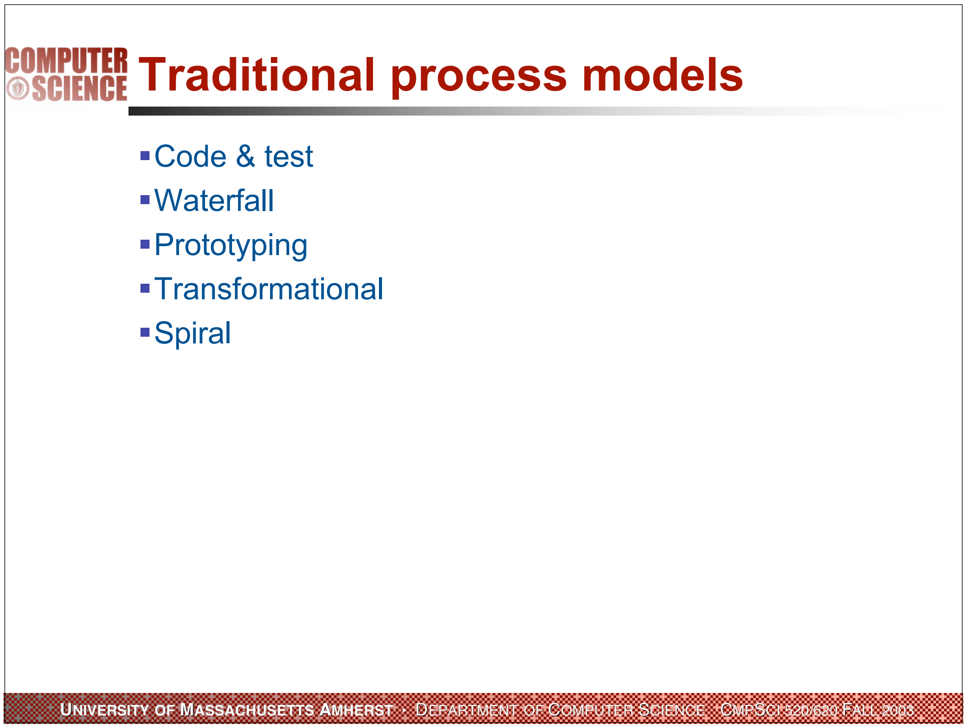
What are the key (sub-)processes it must support



COMPUTER SCIENCE

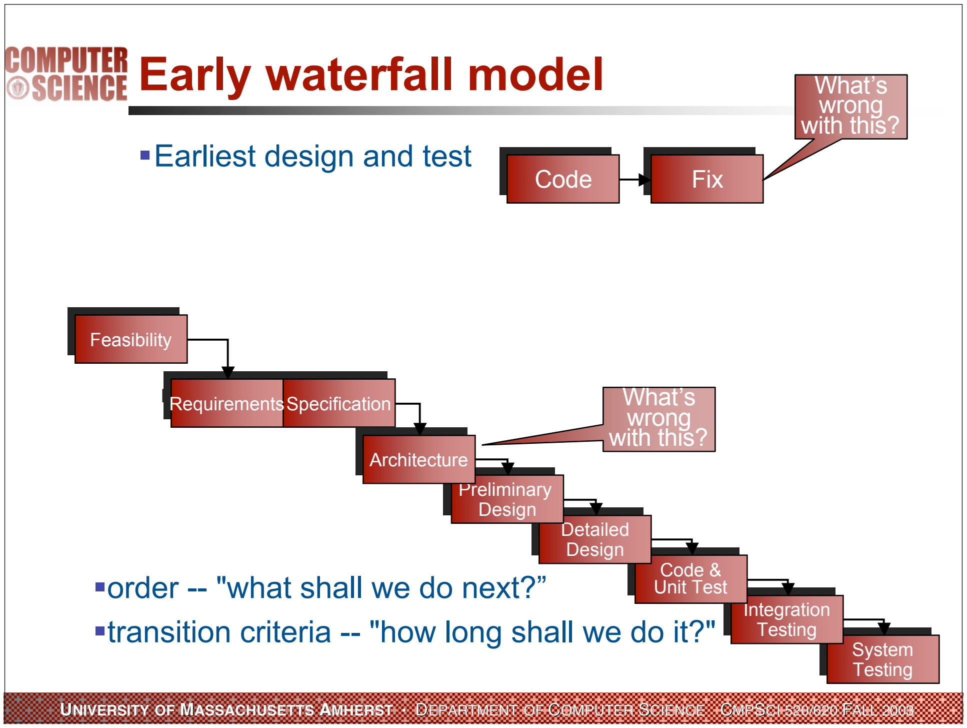
Process

- 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

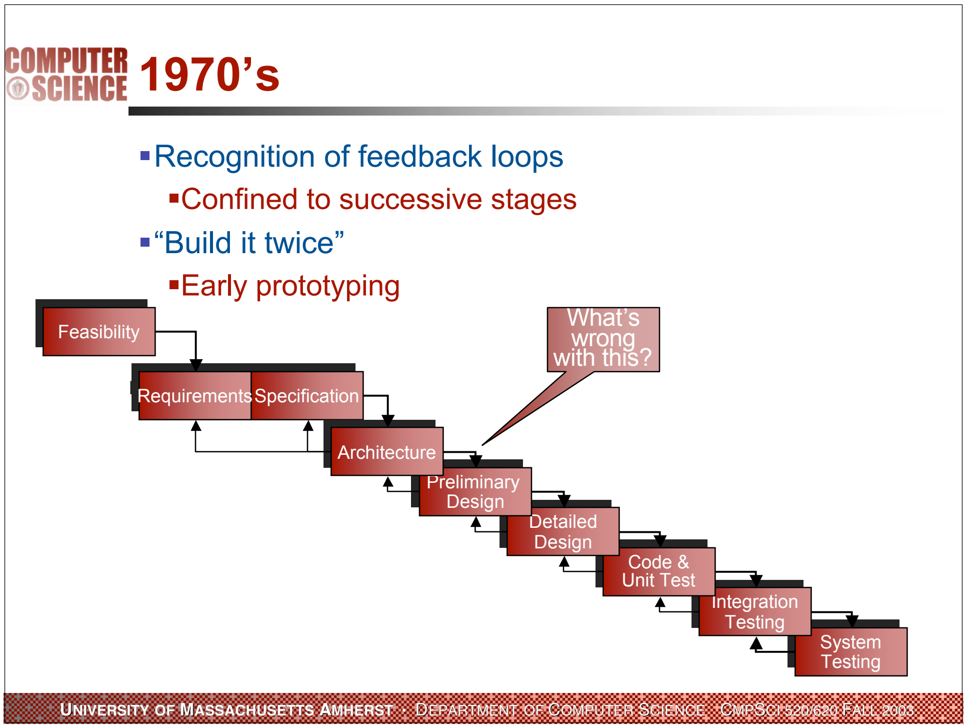


Early waterfall model

- Earliest design and test

Code
Fix
What's wrong with this?
Feasibility
Requirements
Specification
Architecture
Preliminary Design
Detailed Design
Code & Unit Test
Integration Testing
System Testing
What's wrong with this?

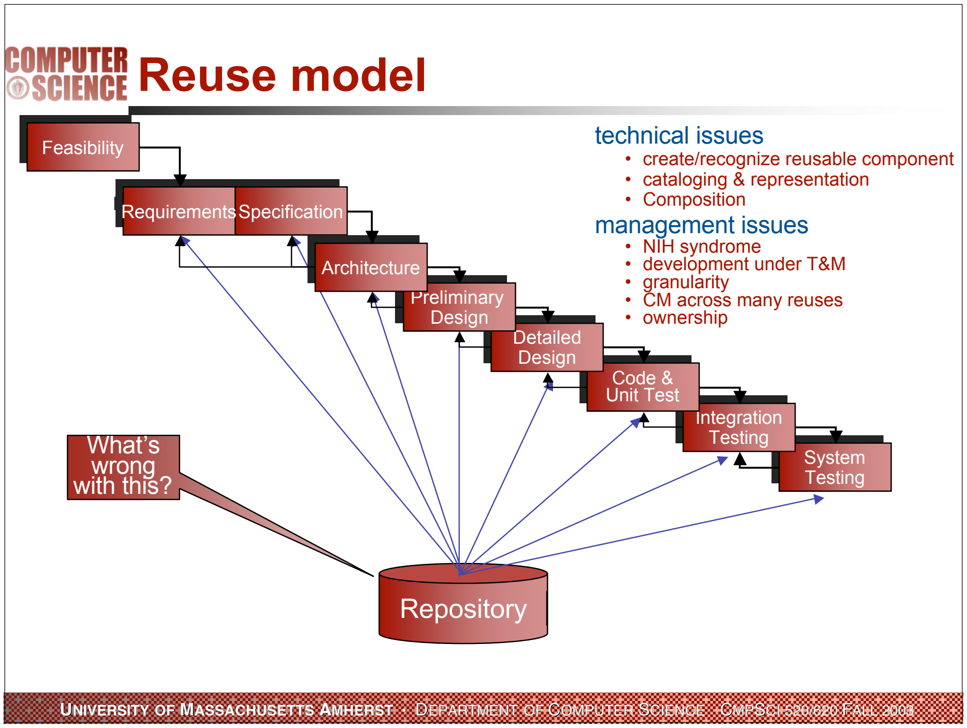
- order -- "what shall we do next?"
- transition criteria -- "how long shall we do it?"



1970's

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

Feasibility
Requirements
Specification
Architecture
Preliminary Design
Detailed Design
Code & Unit Test
Integration Testing
System Testing
What's wrong with this?

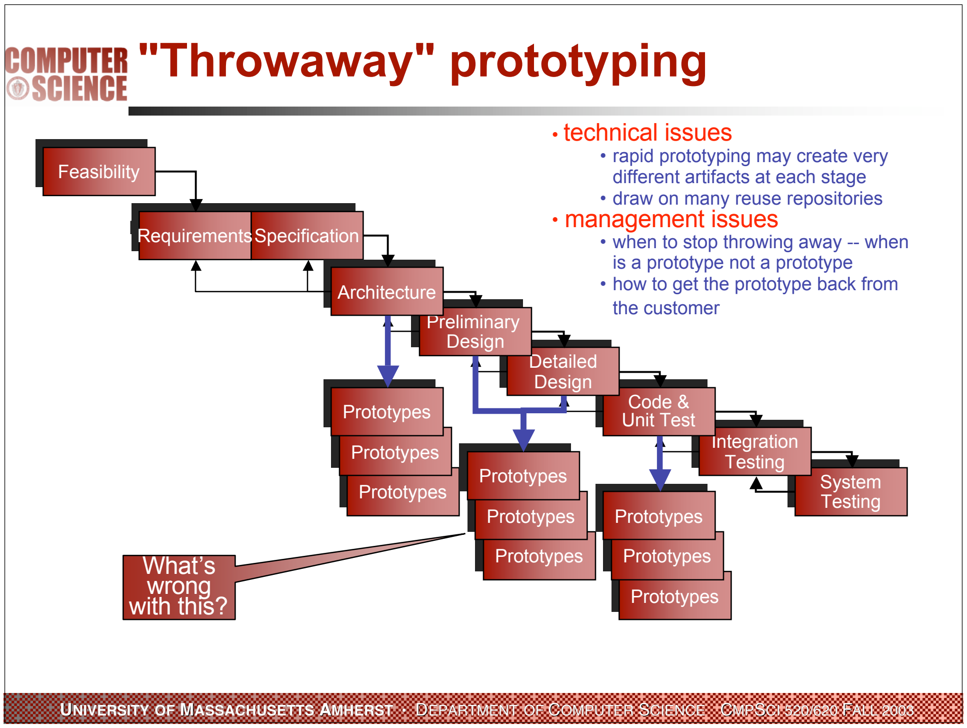


Reuse model

technical issues
- create/recognize reusable component
- cataloging & representation
- Composition

management issues
- NIH syndrome
- development under T&M
- granularity
- CM across many reuses
- ownership

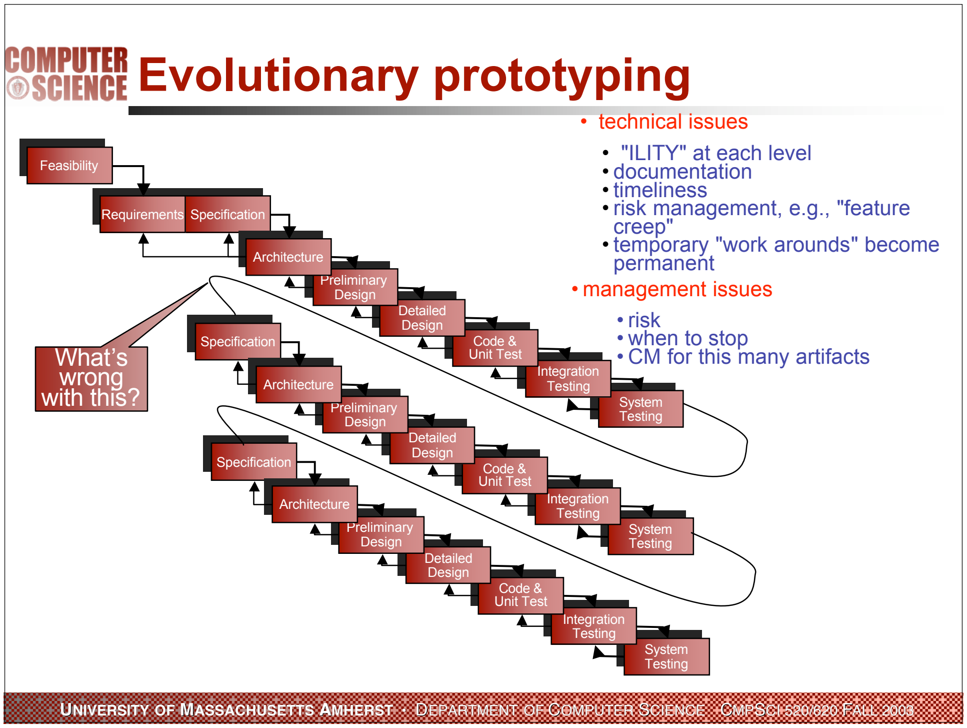
Feasibility
Requirements
Specification
Architecture
Preliminary Design
Detailed Design
Code & Unit Test
Integration Testing
System Testing
What's wrong with this?
Repository



"Throwaway" prototyping

- technical issues
  - rapid prototyping may create very different artifacts at each stage
  - draw on many reuse repositories
- management issues
  - when to stop throwing away -- when is a prototype not a prototype
  - how to get the prototype back from the customer

COMPUTER SCIENCE
Feasibility
Requirements
Specification
Architecture
Preliminary Design
Detailed Design
Code & Unit Test
Integration Testing
System Testing
Prototypes
Prototypes
Prototypes
Prototypes
Prototypes
Prototypes
Prototypes
Prototypes
Prototypes
What's wrong with this?
UNIVERSITY OF MASSACHUSETTS AMHERST · DEPARTMENT OF COMPUTER SCIENCE · CMPSCI 520/620 FALL 2003

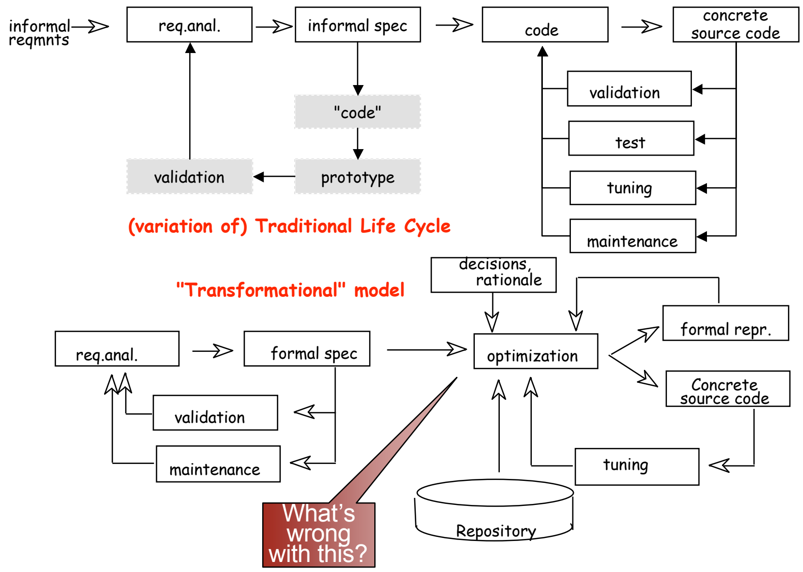


Evolutionary prototyping

- technical issues
  - "ILITY" at each level
  - documentation
  - timeliness
  - risk management, e.g., "feature creep"
  - temporary "work arounds" become permanent
- management issues
  - risk
  - when to stop
  - CM for this many artifacts

COMPUTER SCIENCE
Feasibility
Requirements
Specification
Architecture
Preliminary Design
Detailed Design
Code & Unit Test
Integration Testing
System Testing
Specification
Architecture
Preliminary Design
Detailed Design
Code & Unit Test
Integration Testing
System Testing
Specification
Architecture
Preliminary Design
Detailed Design
Code & Unit Test
Integration Testing
System Testing
What's wrong with this?
UNIVERSITY OF MASSACHUSETTS AMHERST · DEPARTMENT OF COMPUTER SCIENCE · CMPSCI 520/620 FALL 2003

Transformational Model


informal reqmnts
req.anal.
informal spec
"code"
prototype
validation
code
concrete source code
validation
test
tuning
maintenance

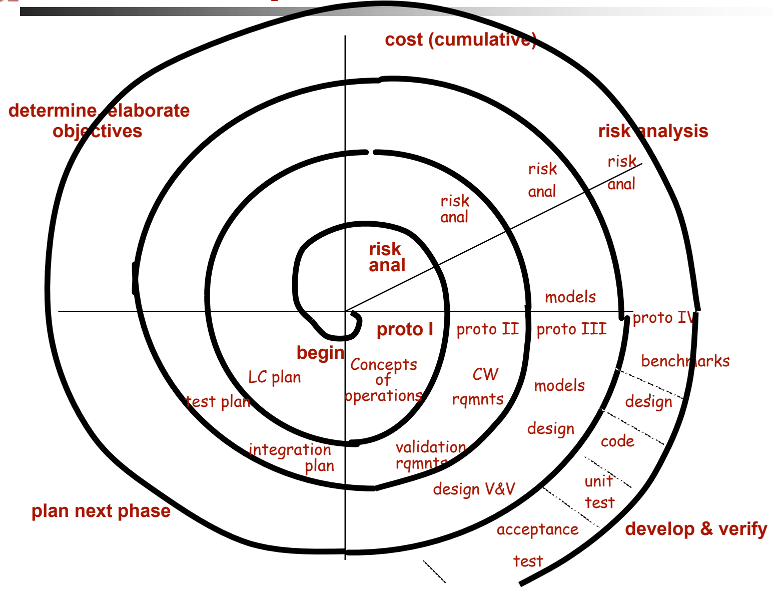
(variation of) Traditional Life Cycle

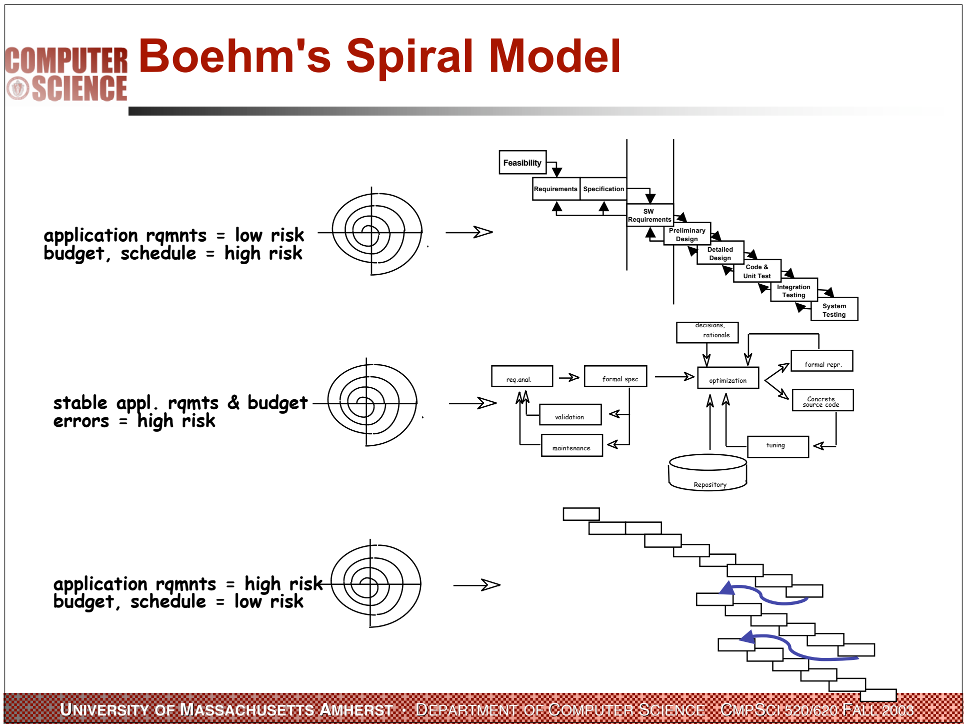
"Transformational" model

decisions, rationale
req.anal.
formal spec
optimization
formal repr.
Concrete source code
validation
maintenance
tuning
Repository

What's wrong with this?

Boehm's Spiral Model


cost (cumulative)
determine, elaborate objectives
risk analysis
risk anal
risk anal
risk anal
risk anal
proto I
proto II
proto III
proto IV
models
begin
Concepts of operations
CW rqmnts
models
benchmarks
LC plan
test plan
design
design
code
integration plan
validation rqmnts
design V&V
unit test
acceptance test
plan next phase
develop & verify



Boehm's Spiral Model

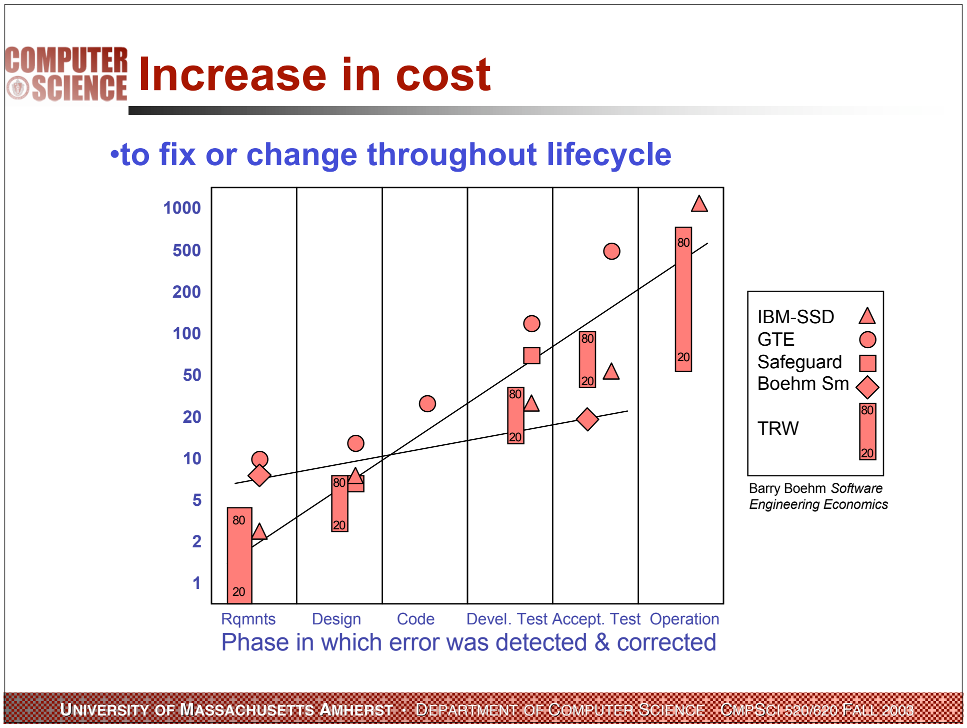
application rqmnts = low risk
budget, schedule = high risk

stable appl. rqmts & budget
errors = high risk

application rqmnts = high risk
budget, schedule = low risk

Feasibility
Requirements
Specification
SW Requirements
Preliminary Design
Detailed Design
Code & Unit Test
Integration Testing
System Testing
decisions, rationale
req.anal.
formal spec
optimization
formal repr.
Concrete source code
validation
maintenance
tuning
Repository

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

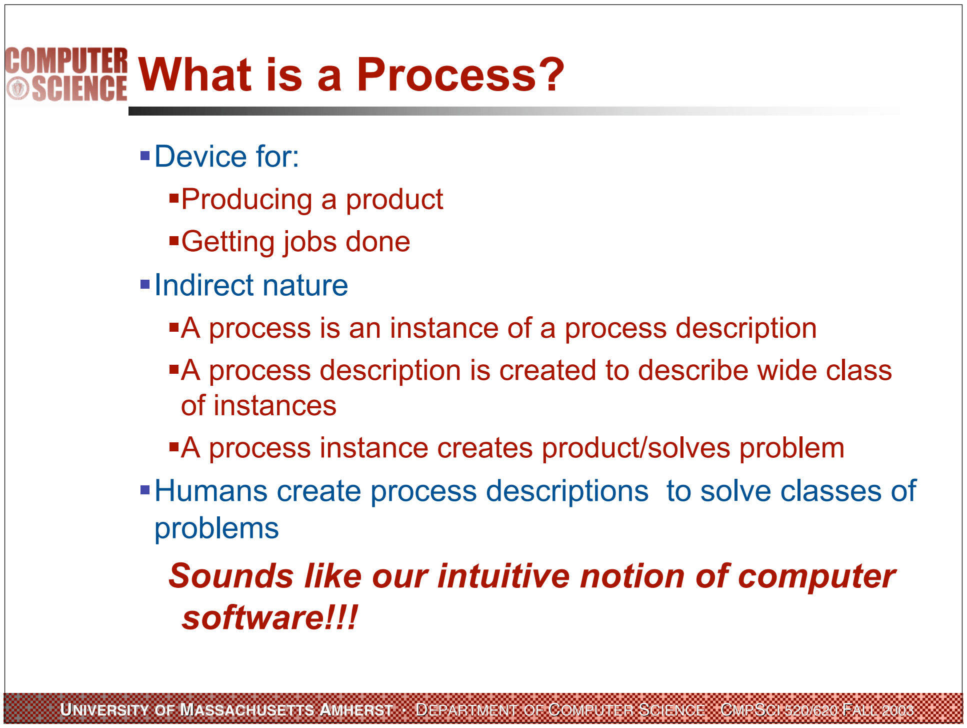


Increase in cost

- to fix or change throughout lifecycle

Barry Boehm *Software Engineering Economics*

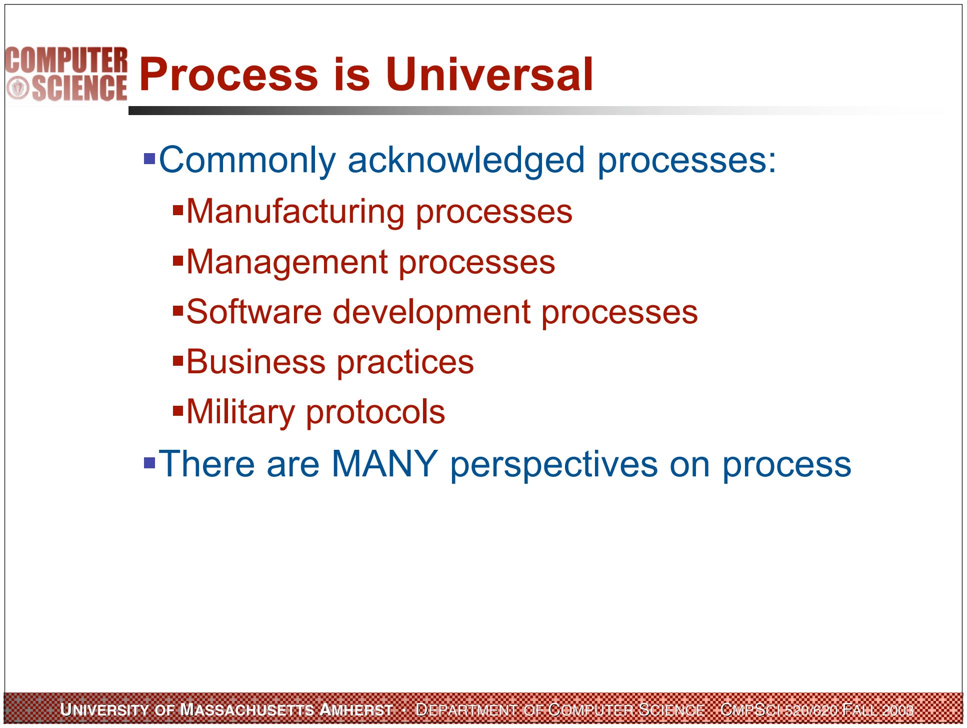
UNIVERSITY OF MASSACHUSETTS AMHERST • DEPARTMENT OF COMPUTER SCIENCE • CMPSCI 520/620 FALL 2003



What is a Process?

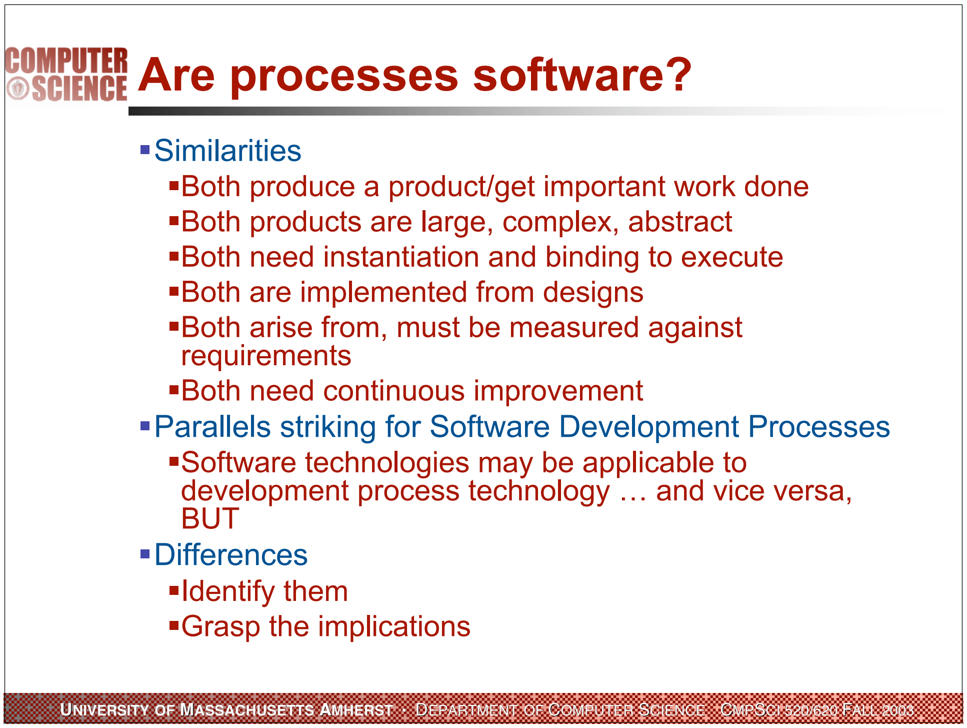
- Device for:
  - Producing a product
  - Getting jobs done
- Indirect nature
  - A process is an instance of a process description
  - A process description is created to describe wide class of instances
  - A process instance creates product/solves problem
- Humans create process descriptions to solve classes of problems

Sounds like our intuitive notion of computer software!!!



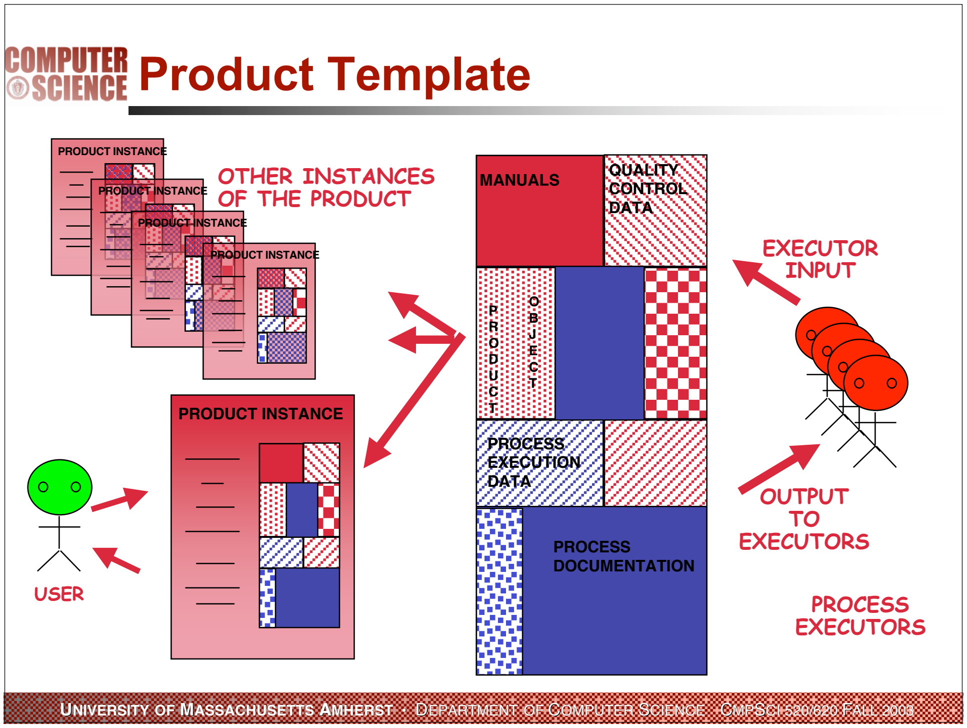
Process is Universal

- Commonly acknowledged processes:
  - Manufacturing processes
  - Management processes
  - Software development processes
  - Business practices
  - Military protocols
- There are MANY perspectives on process



Are processes software?

- Similarities
  - Both produce a product/get important work done
  - Both products are large, complex, abstract
  - Both need instantiation and binding to execute
  - Both are implemented from designs
  - Both arise from, must be measured against requirements
  - Both need continuous improvement
- Parallels striking for Software Development Processes
  - Software technologies may be applicable to development process technology … and vice versa, BUT
- Differences
  - Identify them
  - Grasp the implications



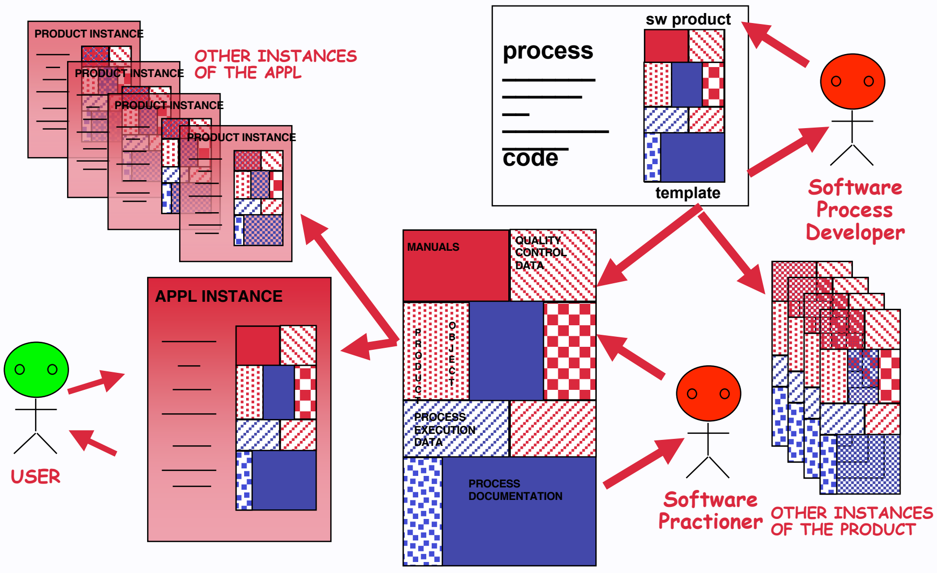
Product Template

PRODUCT INSTANCE
PRODUCT INSTANCE
PRODUCT INSTANCE
PRODUCT INSTANCE
OTHER INSTANCES OF THE PRODUCT
PRODUCT INSTANCE
USER
MANUALS
QUALITY CONTROL DATA
PRODUCT
OBJECT
PROCESS EXECUTION DATA
PROCESS DOCUMENTATION
EXECUTOR INPUT
OUTPUT TO EXECUTORS
PROCESS EXECUTORS

(Software) Processes Are Software Too

- Hypothesis:
  - Significant software product technology is directly applicable to developing software processes -- and executing them
- Process Programming
  - Application of classical software development technology to the development of software processes
- Software Process Programming
  - Using classical software development technology to develop the processes that are used to develop application software

Software Process Instantiation


PRODUCT INSTANCE
PRODUCT INSTANCE
PRODUCT INSTANCE
PRODUCT INSTANCE
OTHER INSTANCES OF THE APPL
process
code
sw product
template
Software Process Developer
APPL INSTANCE
USER
MANUALS
QUALITY CONTROL DATA
PRODUCT
OBJECT
PROCESS EXECUTION DATA
PROCESS DOCUMENTATION
Software Practioner
OTHER INSTANCES OF THE PRODUCT

Processes as Software

- Processes should be developed using **a (Process) development process**
  - Process Requirements
    - Key to designing suitable process
    - Basis for evaluation and improvement of process
  - Process Specification/Modeling/Design
    - Helps conceptualization, communication, visualization
    - Can be management aid
  - Process Code
    - Provides rigor and complete details
    - Basis for execution/tool support and integration
  - Process Measurements and Evaluation
    - Results of Static Analysis and Dynamic Measurement
    - Basis for Process Maintenance (**Process Improvement**)

Development Process Requirements

- Software process development must start with specification of software development process requirements
  - Provides basis for software process modeling and design
  - And the basis for effective process evaluation
- Different requirements should be expected to lead to different software development processes
- There is NO Single, "Ideal" Software Development Process
  - just as there is no single, fixed, ideal software product
  - in both cases, the software must be developed to satisfy previously specified requirements

Development Process Requirements

- Product specifications are process functional requirements
- Development schedules are process speed requirements
- There are also robustness, visibility, interface, etc. process requirements

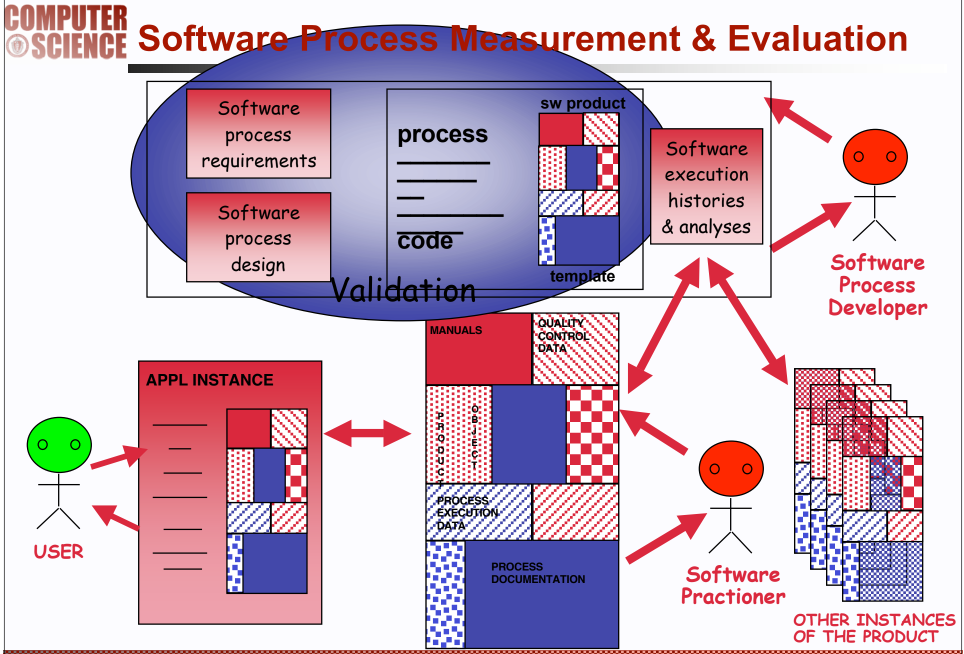
Other SW Process phases

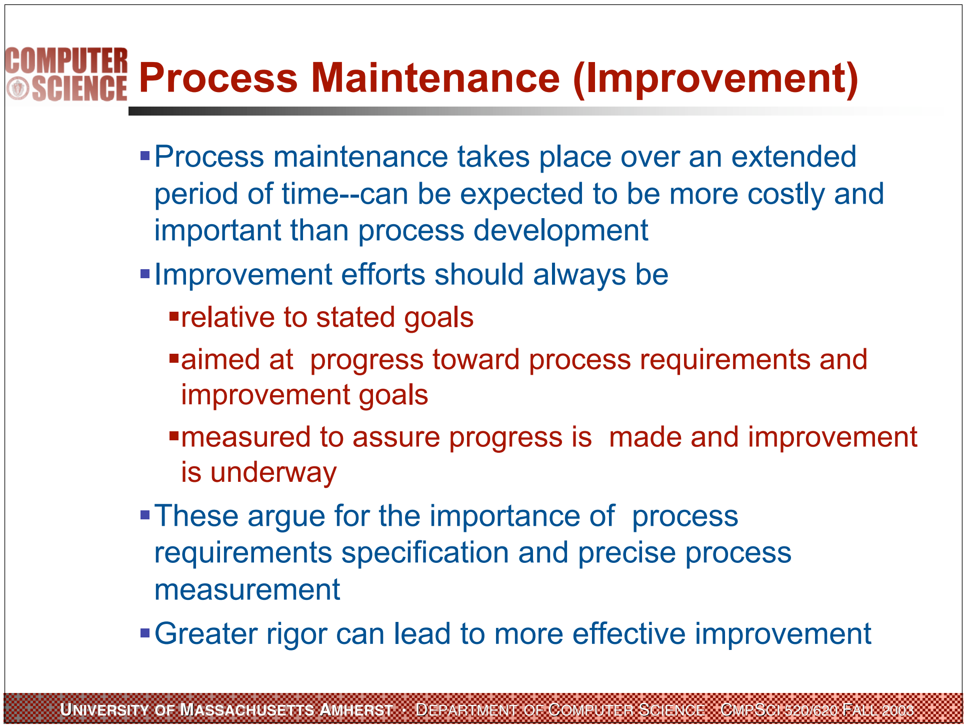
- Software Process Modeling/Design Goals
  - analogous to product design
  - basis for improved intuitive understanding, visualization, early analysis and validation of requirements, deeper understanding & conceptualizations of key software processes
  - helps demonstrate effective approach to satisfying product requirements
- Software Process Code
  - Uses languages with execution semantics
  - Provides details and elaborations upon process design; helps to grapple with details omitted from model/design
  - Vehicle for meshing process control with product data at arbitrarily low levels of detail
  - Provides superior visibility enabling better control
    - Basis for better predictability
    - Basis for process execution
  - Blueprint for tool integration

Other SW Process phases

- Software Process Measurement and Evaluation
- An analogy to application software measurement and evaluation seems very useful here
  - Dynamic monitoring of process execution is analogous to interactive debugging of application software
  - Need to support analysis of processes too

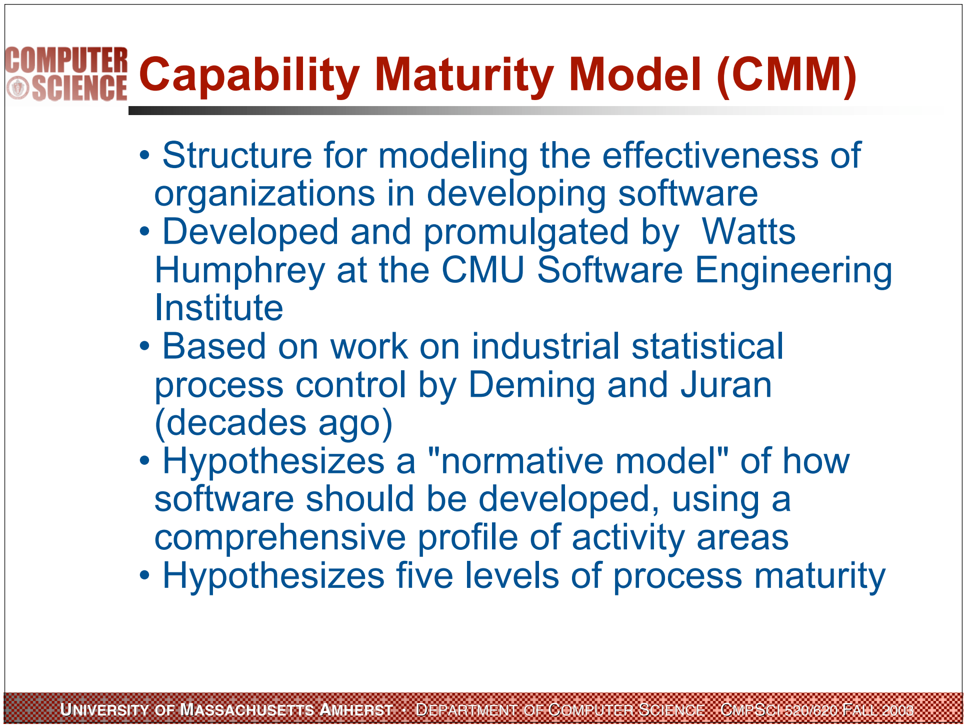
Software Process Measurement & Evaluation


Software process requirements
Software process design
process
code
sw product
template
Validation
Software execution histories & analyses
Software Process Developer
MANUALS
QUALITY CONTROL DATA
PRODUCT
OBJECT
PROCESS EXECUTION DATA
PROCESS DOCUMENTATION
APPL INSTANCE
USER
Software Practioner
OTHER INSTANCES OF THE PRODUCT



Process Maintenance (Improvement)

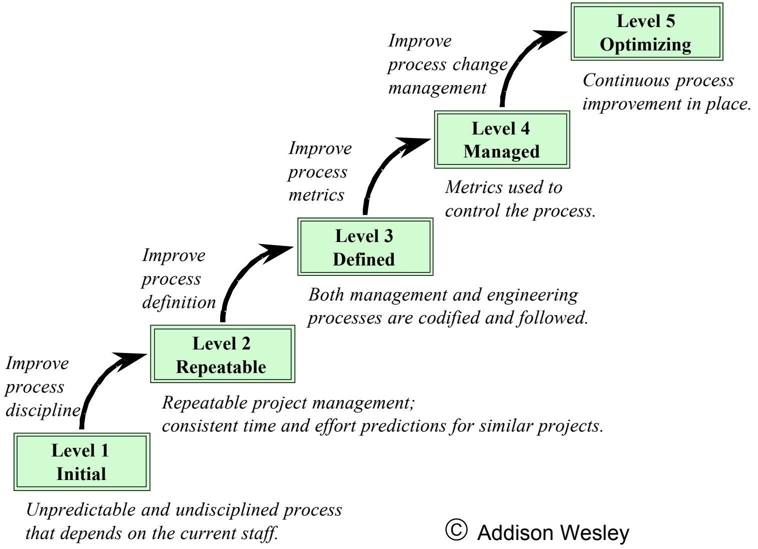
- Process maintenance takes place over an extended period of time--can be expected to be more costly and important than process development
- Improvement efforts should always be
  - relative to stated goals
  - aimed at progress toward process requirements and improvement goals
  - measured to assure progress is made and improvement is underway
- These argue for the importance of process requirements specification and precise process measurement
- Greater rigor can lead to more effective improvement



Capability Maturity Model (CMM)

- Structure for modeling the effectiveness of organizations in developing software
- Developed and promulgated by Watts Humphrey at the CMU Software Engineering Institute
- Based on work on industrial statistical process control by Deming and Juran (decades ago)
- Hypothesizes a "normative model" of how software should be developed, using a comprehensive profile of activity areas
- Hypothesizes five levels of process maturity

CMM



Level 5
Optimizing

Continuous process improvement in place.

Improve process change management

Level 4
Managed

Metrics used to control the process.

Improve process metrics

Level 3
Defined

Both management and engineering processes are codified and followed.

Improve process definition

Level 2
Repeatable

Repeatable project management; consistent time and effort predictions for similar projects.

Improve process discipline

Level 1
Initial

Unpredictable and undisciplined process that depends on the current staff.

© Addison Wesley

CMM Attempts to Evaluate Predictability

- Highly mature processes are those that offer assurance of predictable results
- Highest levels of process maturity also demonstrably offer expectation of continuous process improvement
- Higher maturity seems easiest to attain when software development is in a restricted domain

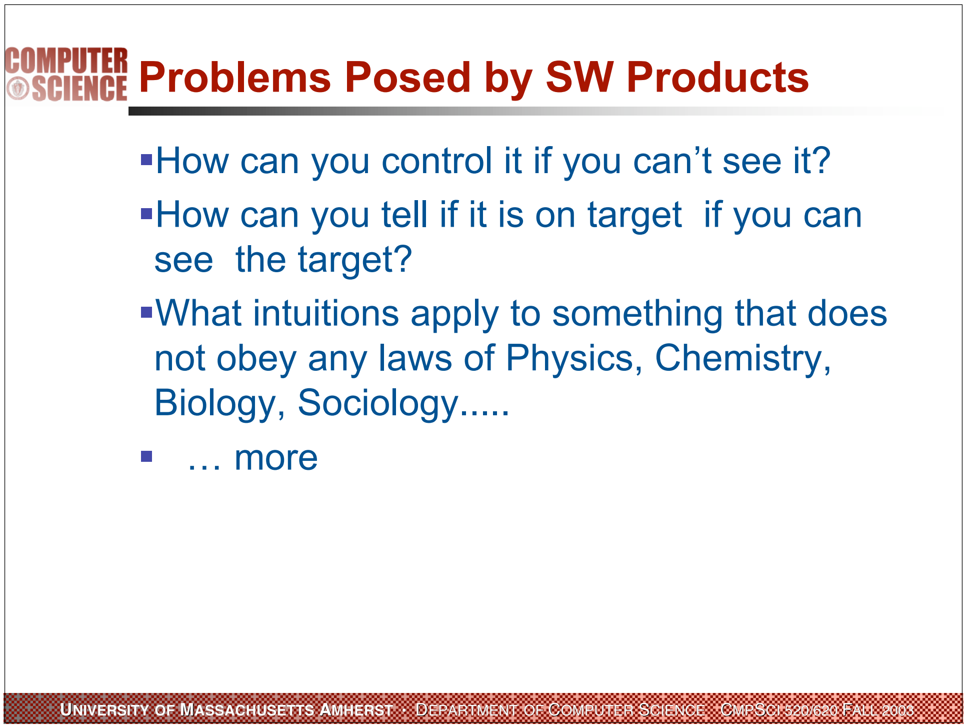
ISO 9000

- Quality management
- Process
- ISO standards are about
  - What must be accomplished
  - Not about **how**
- Certification
  - Company must document and record its activities
  - On-site audit by an ISO registrar

Key Elements of SE

- Focus on product and process
  - Product is output of process
  - But process is a product too (of a different process)
  - How processes create products
  - How product requirements dictate process
- Continuous iterative synthesis and analysis
  - Build a little check a little
  - Interconnection specifications are products too
- All of the above will have to evolve: Plan for it

How to do this for a product (process) that is:
•Insensible
•Non-Physical



Problems Posed by SW Products

- How can you control it if you can’t see it?
- How can you tell if it is on target if you can see the target?
- What intuitions apply to something that does not obey any laws of Physics, Chemistry, Biology, Sociology.....
- … more