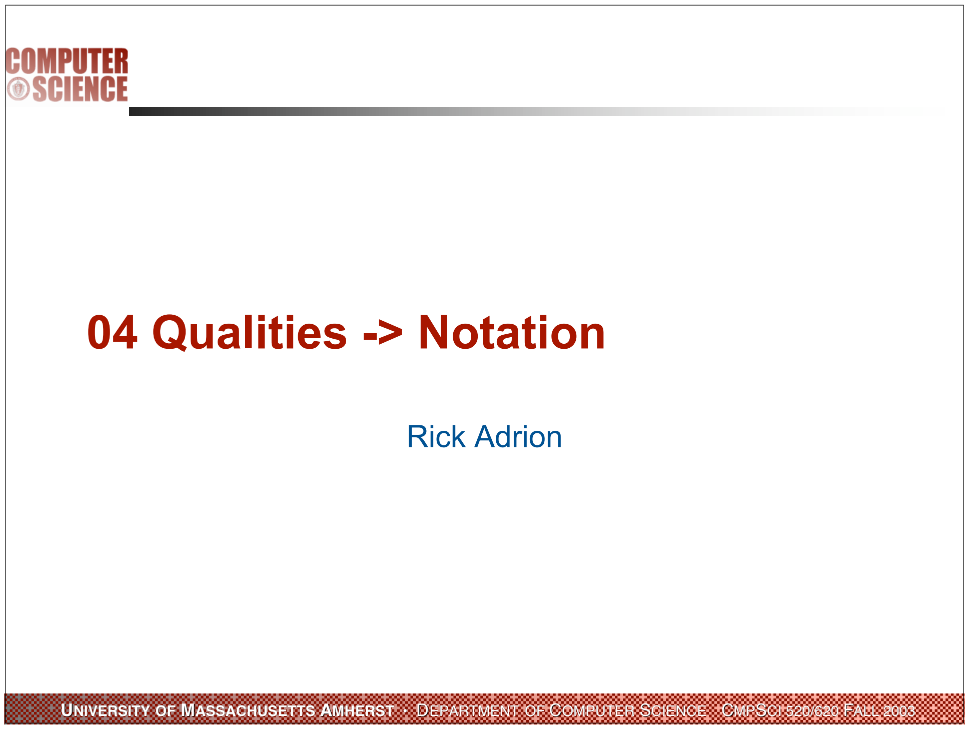


04 Qualities -> Notation

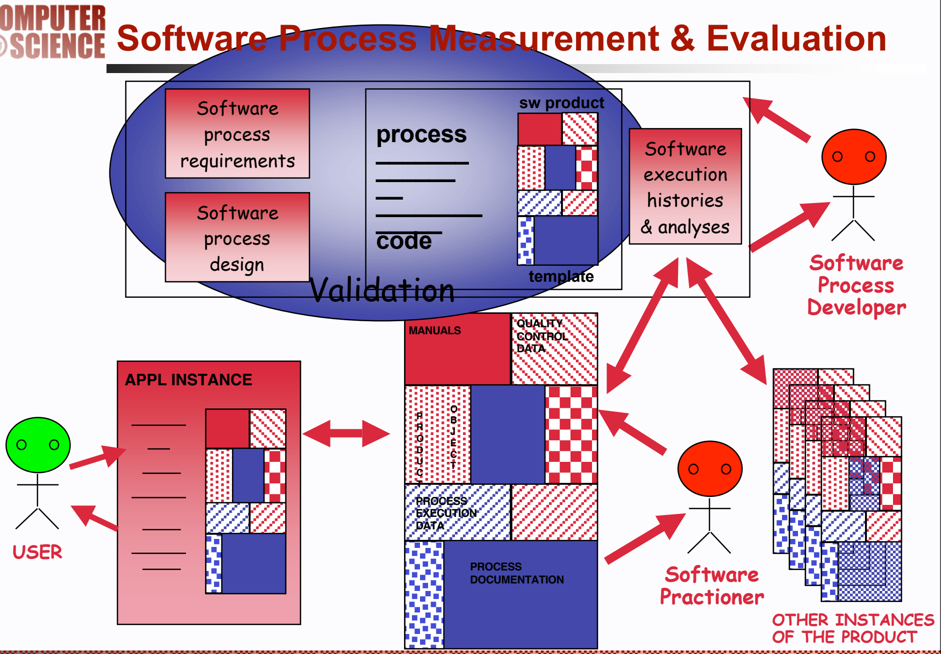
Rick Adrion

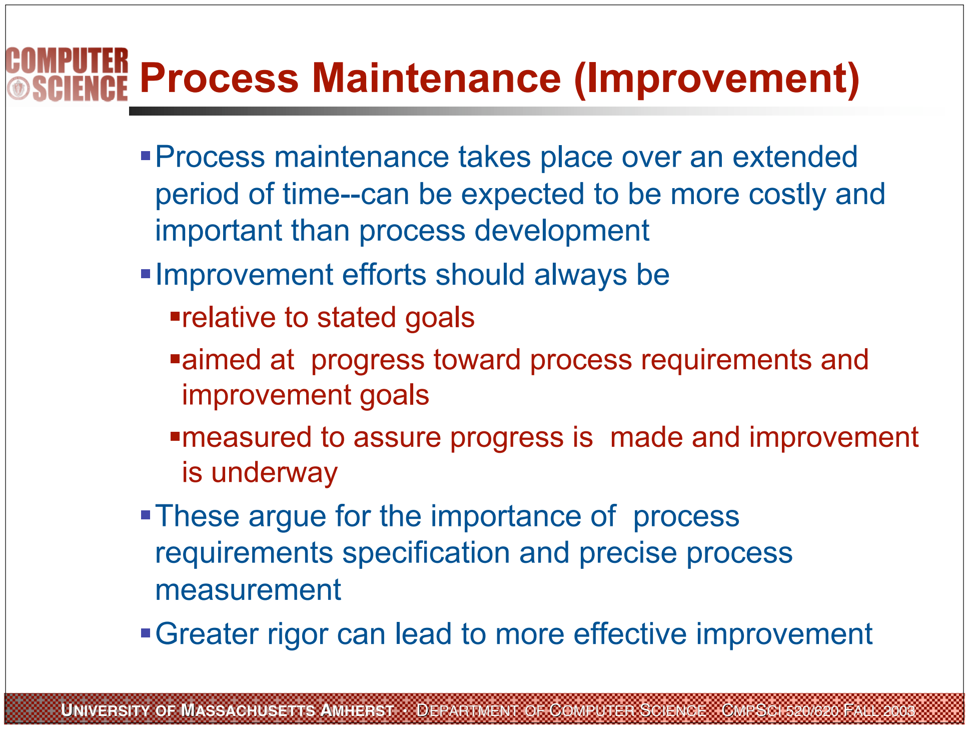
But first

A wrap-up
…. and a segue …

COMPUTER SCIENCE

Software Process Measurement & Evaluation

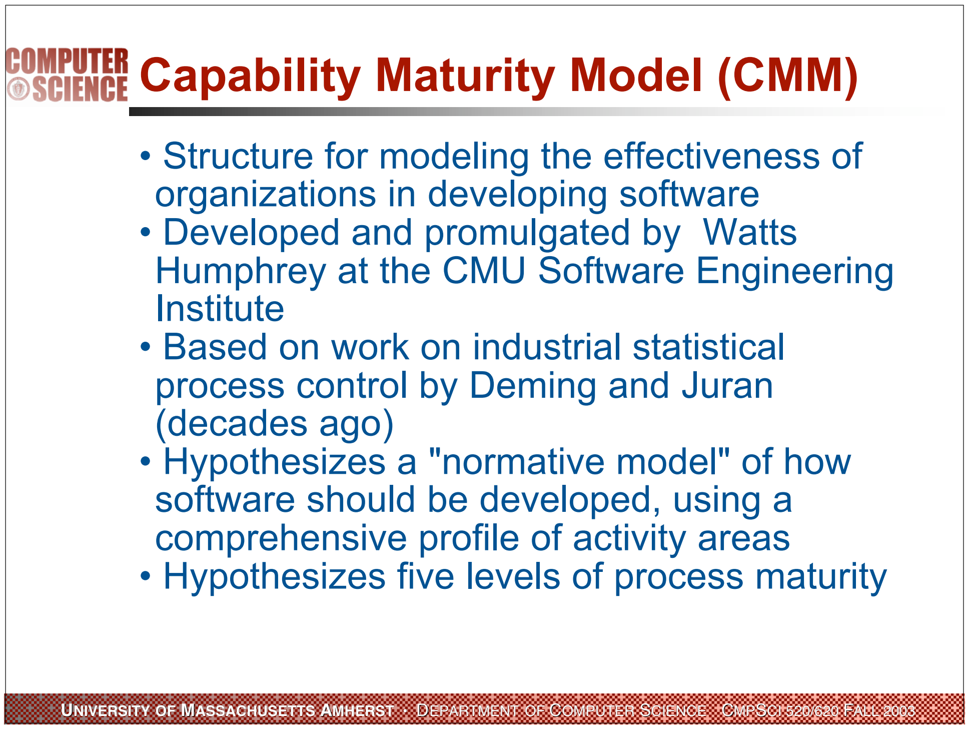




COMPUTER SCIENCE

Process Maintenance (Improvement)

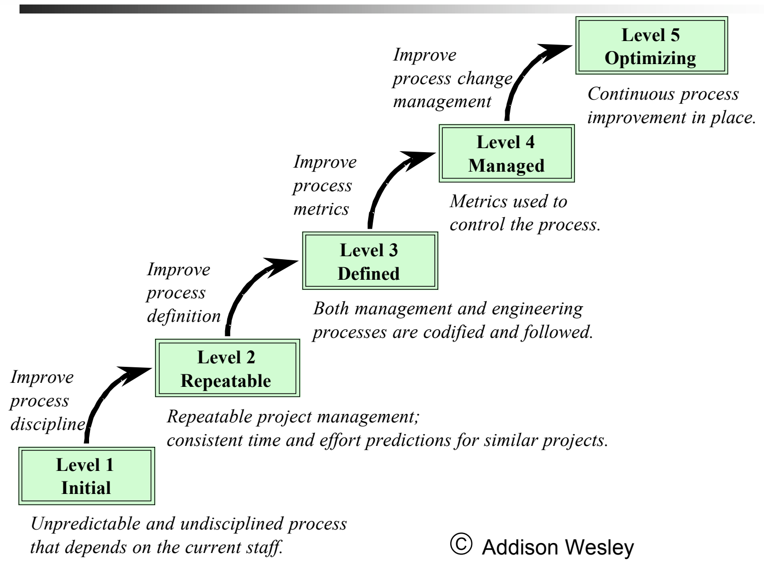
- Process maintenance takes place over an extended period of time--can be expected to be more costly and important than process development
- Improvement efforts should always be
  - relative to stated goals
  - aimed at progress toward process requirements and improvement goals
  - measured to assure 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



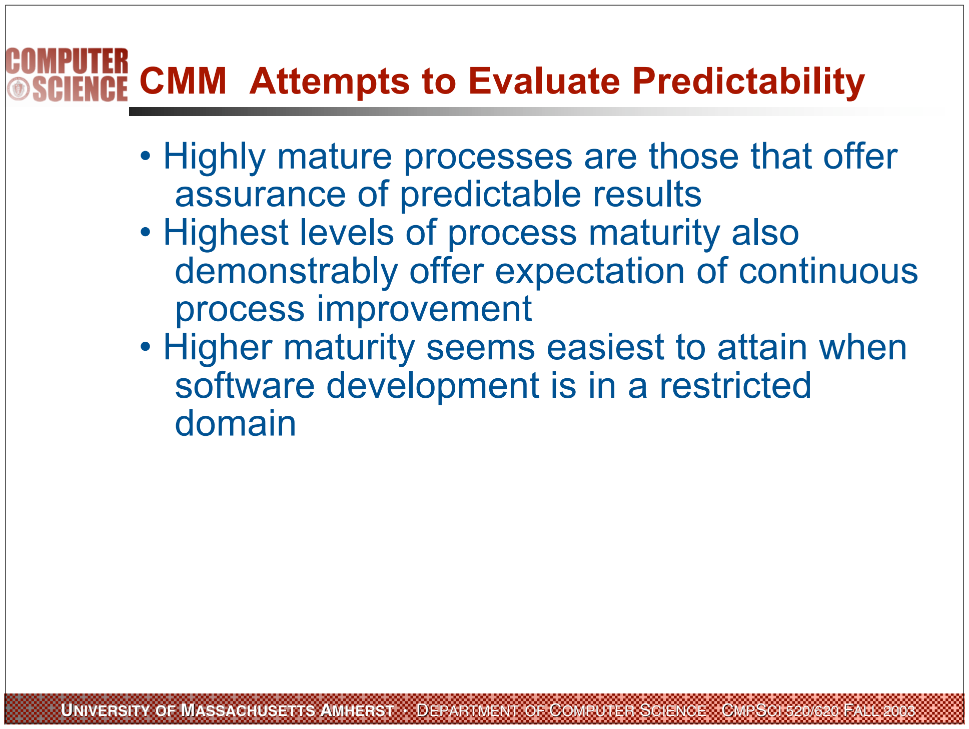
Capability Maturity Model (CMM)

- 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.

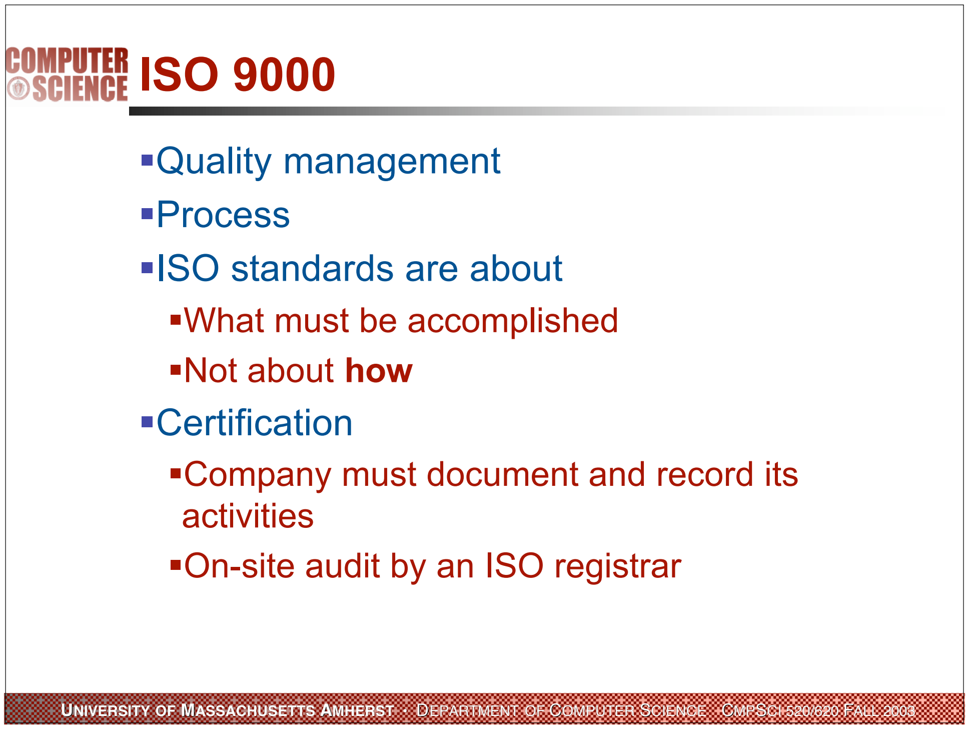
© Addison Wesley



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

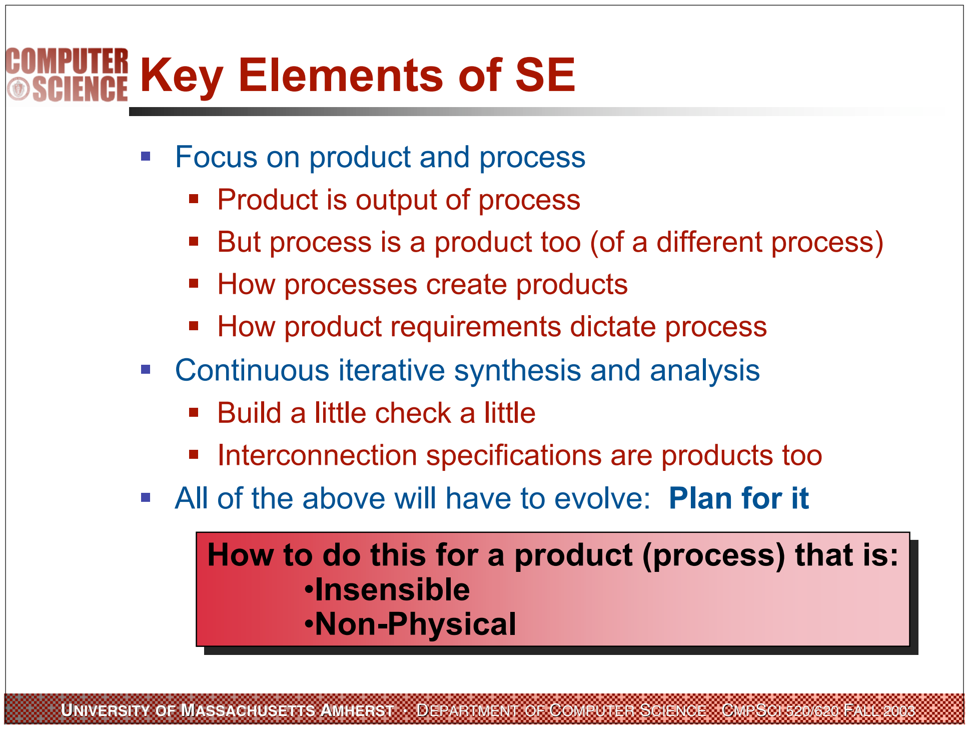
UNIVERSITY OF MASSACHUSETTS AMHERST • DEPARTMENT OF COMPUTER SCIENCE • CMPSCI 520/620 FALL 2003



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

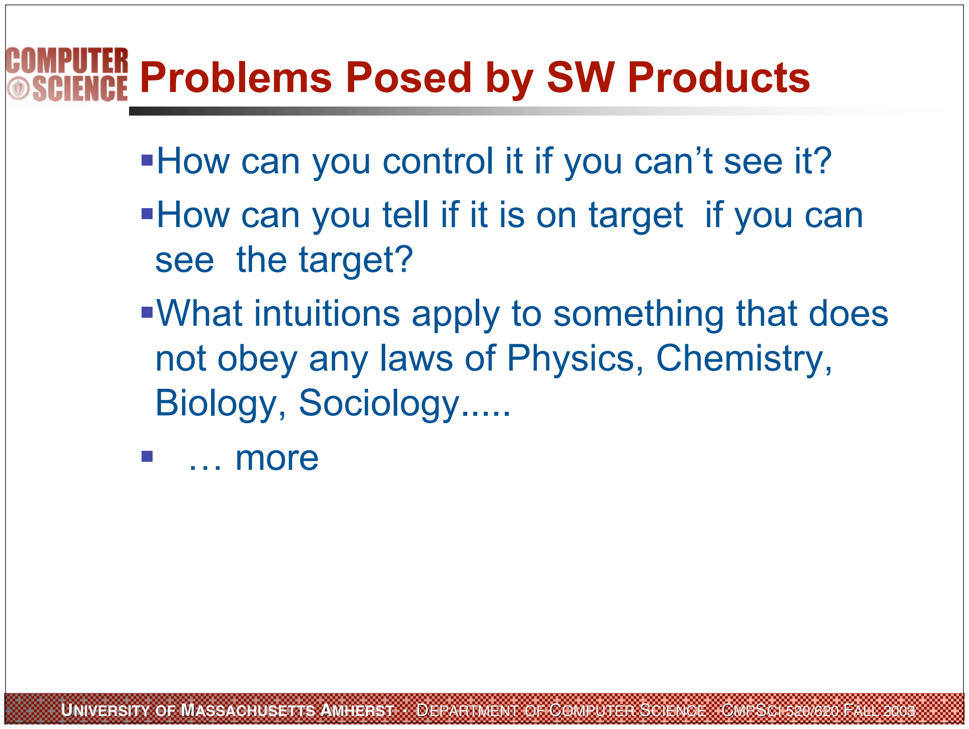
UNIVERSITY OF MASSACHUSETTS AMHERST • DEPARTMENT OF COMPUTER SCIENCE • CMPSCI 520/620 FALL 2003



Key Elements of SE

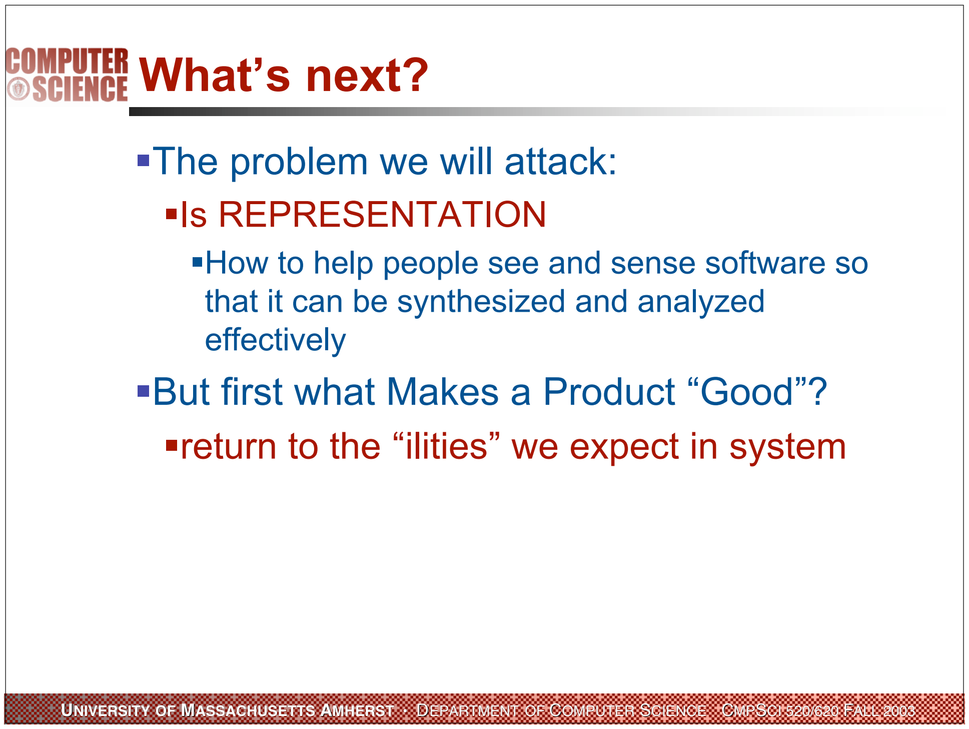
- Focus on product and process
  - Product is output of process
  - But process is a product too (of a different process)
  - How processes create products
  - How product requirements dictate process
- Continuous iterative synthesis and analysis
  - Build a little check a little
  - Interconnection 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

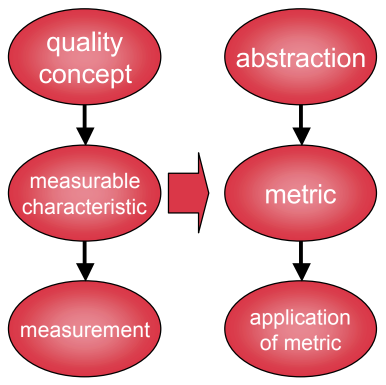
- How can you control it if you can’t see it?
- How can you tell if it is on target if you can see the target?
- What intuitions apply to something that does not obey any laws of Physics, Chemistry, Biology, Sociology.....
- … more

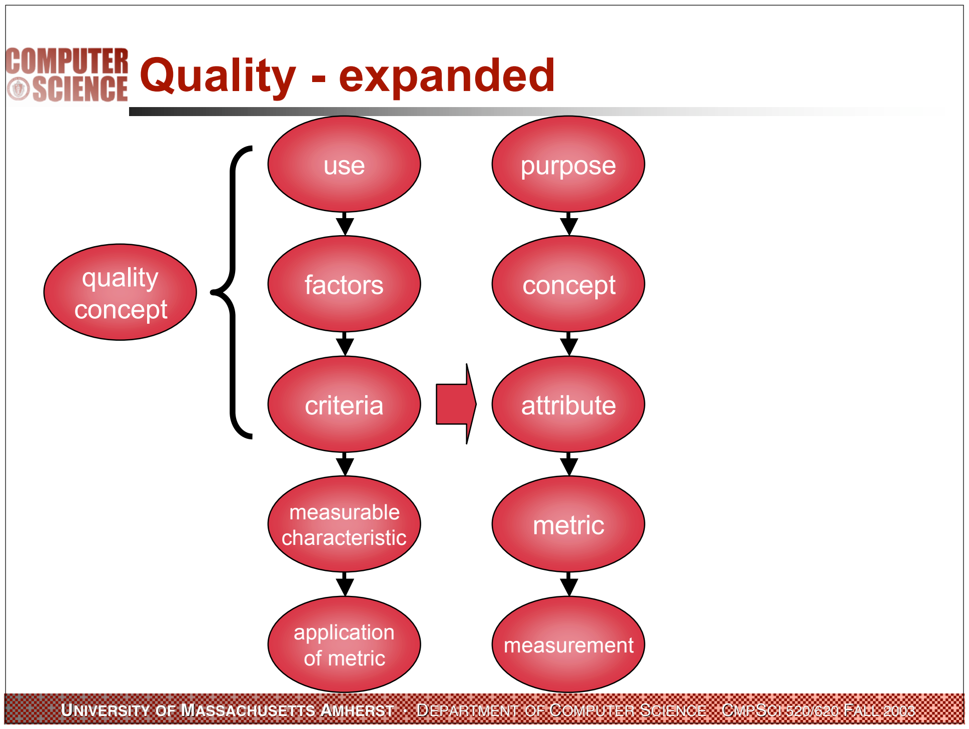


What's next?

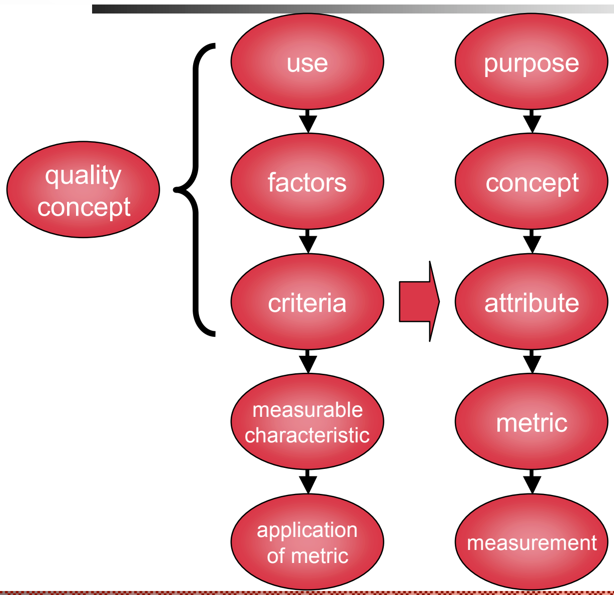
- The problem we will attack:
  - Is REPRESENTATION
    - How to help people see and sense software so that it can be synthesized and analyzed effectively
- But first what Makes a Product "Good"?
  - return to the "ilities" we expect in system

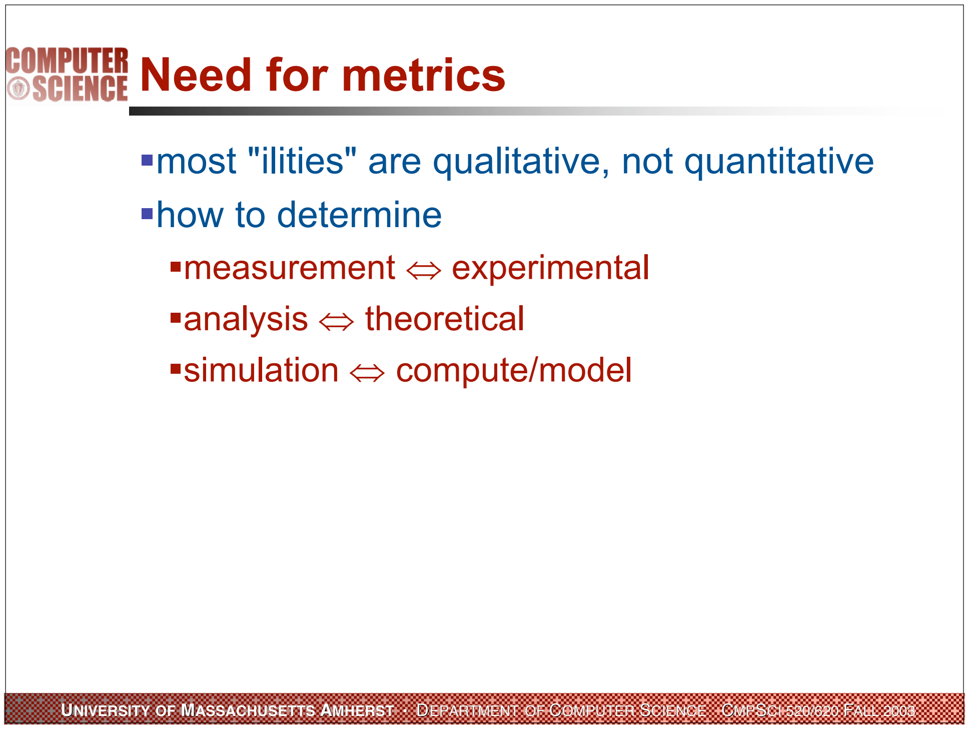
Quality





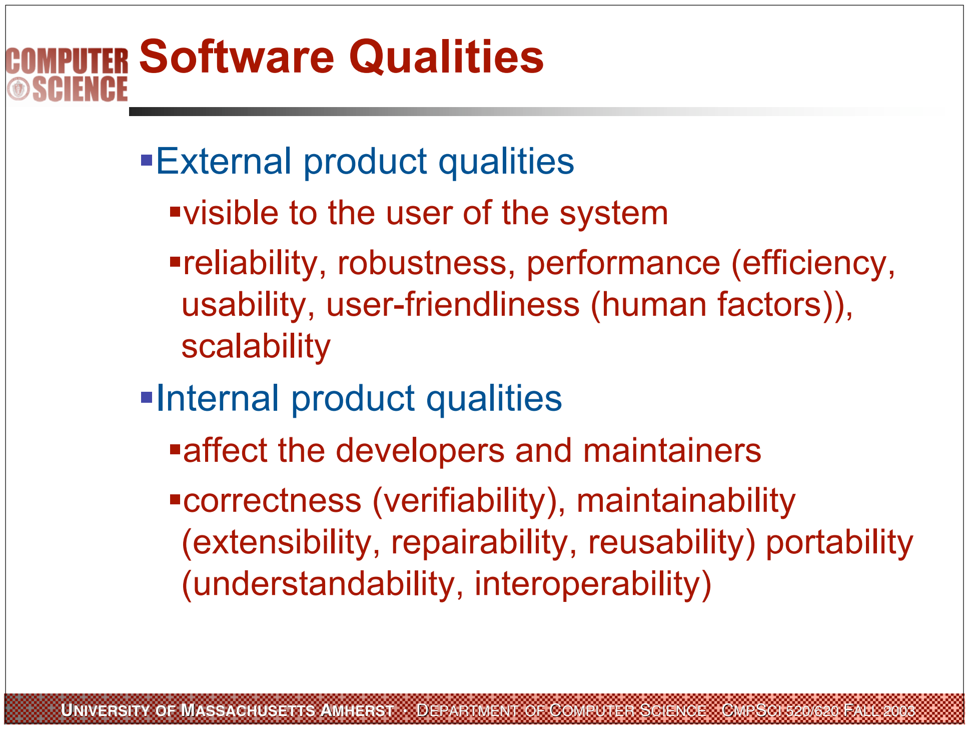
Quality - expanded





Need for metrics

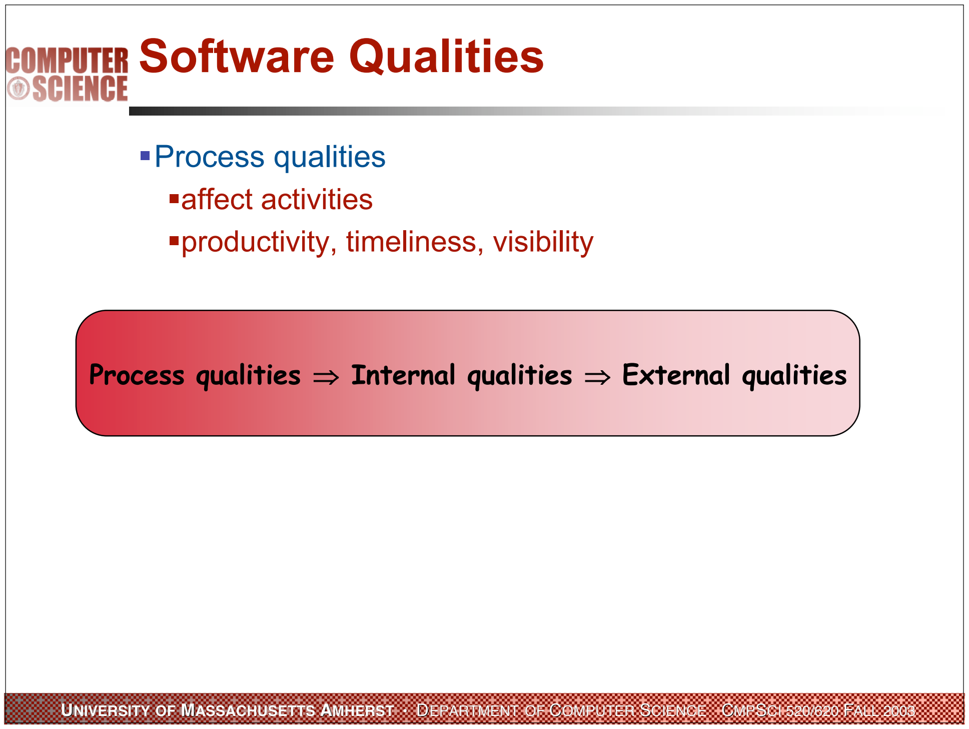
- most "ilities" are qualitative, not quantitative
- how to determine
  - measurement ⇔ experimental
  - analysis ⇔ theoretical
  - simulation ⇔ compute/model



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)

University of Massachusetts Amherst · Department of Computer Science · CMPSCI 520/620 Fall 2003

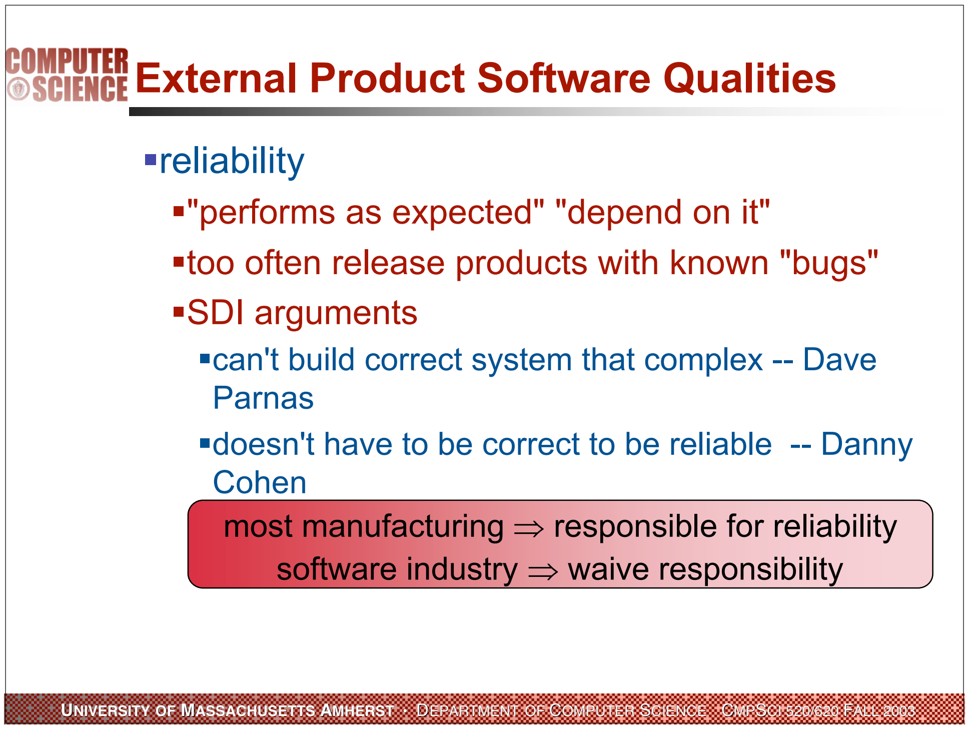


Software Qualities

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

Process qualities ⇒ Internal qualities ⇒ External qualities

University of Massachusetts Amherst · Department of Computer Science · CMPSCI 520/620 Fall 2003

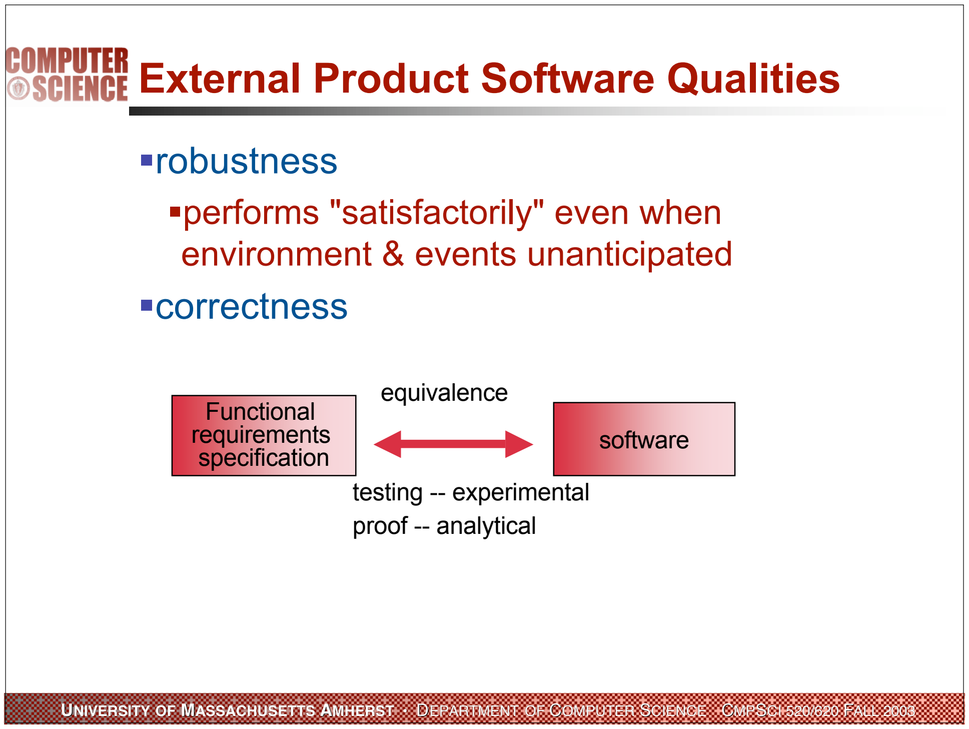


External Product Software Qualities

- 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

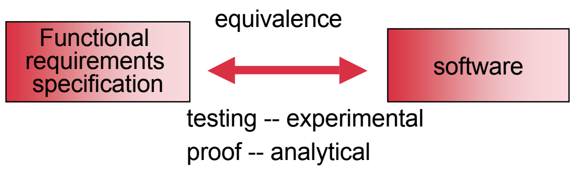
most manufacturing ⇒ responsible for reliability
software industry ⇒ waive responsibility

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

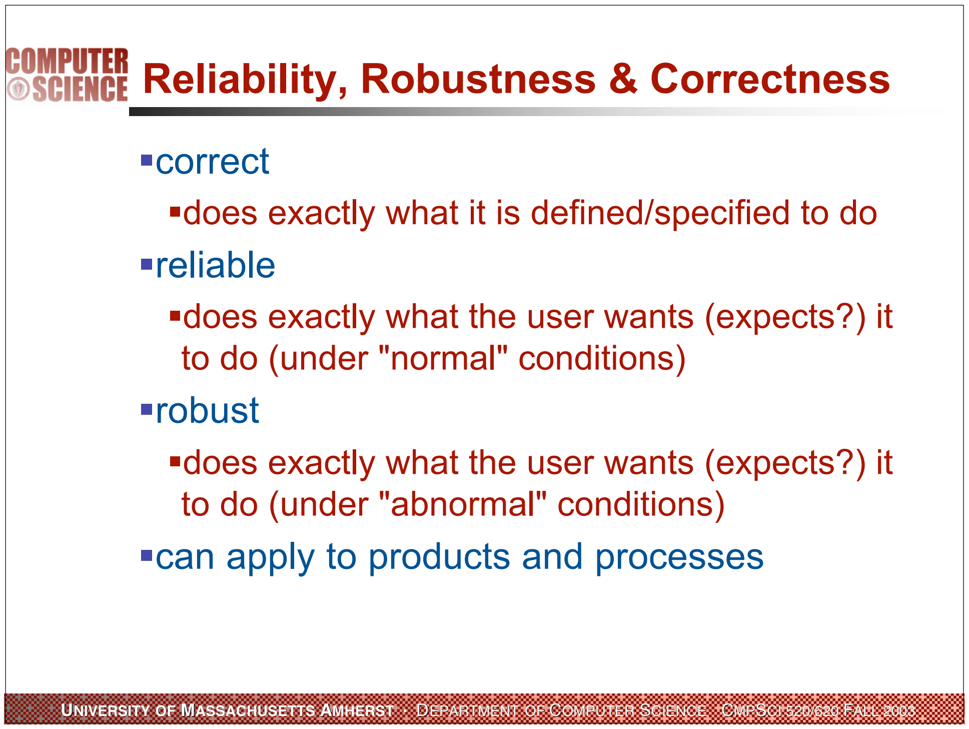


External Product Software Qualities

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

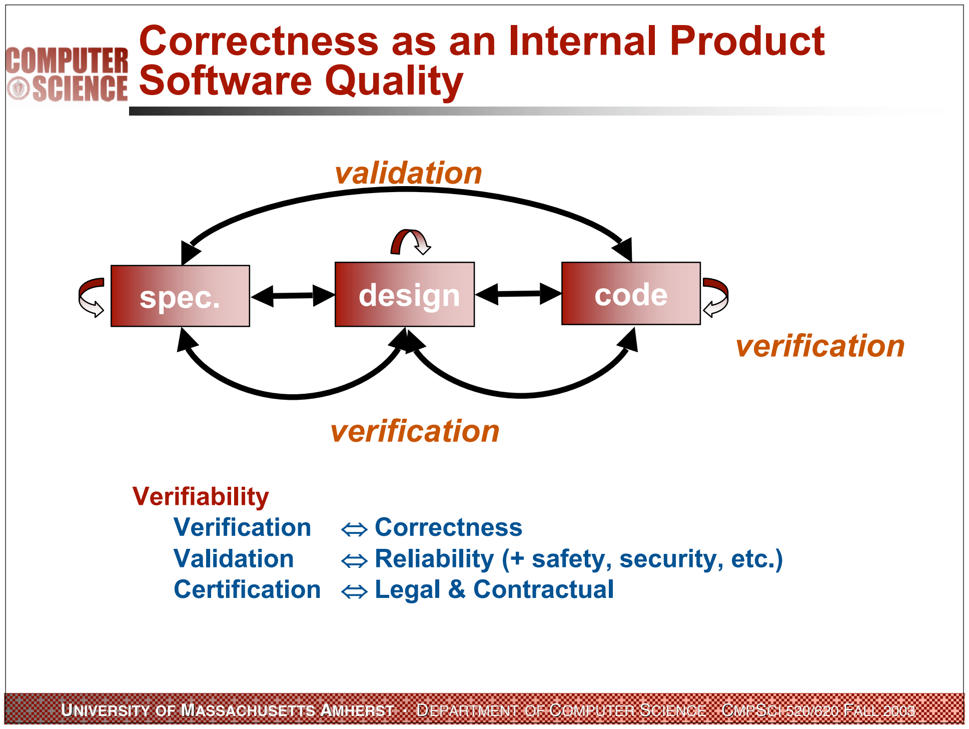


UNIVERSITY OF MASSACHUSETTS AMHERST · DEPARTMENT OF COMPUTER SCIENCE · CMPSCI 520/620 FALL 2003

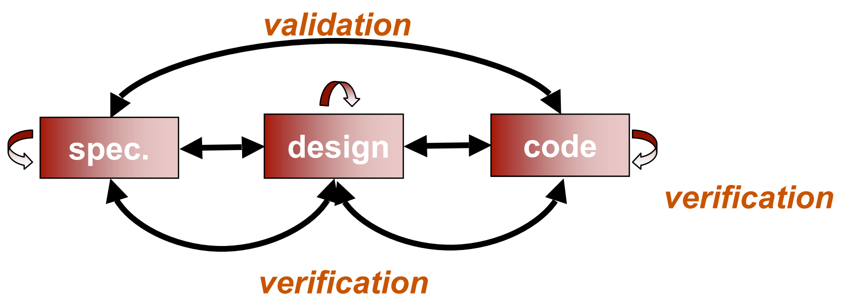


Reliability, Robustness & Correctness

- 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

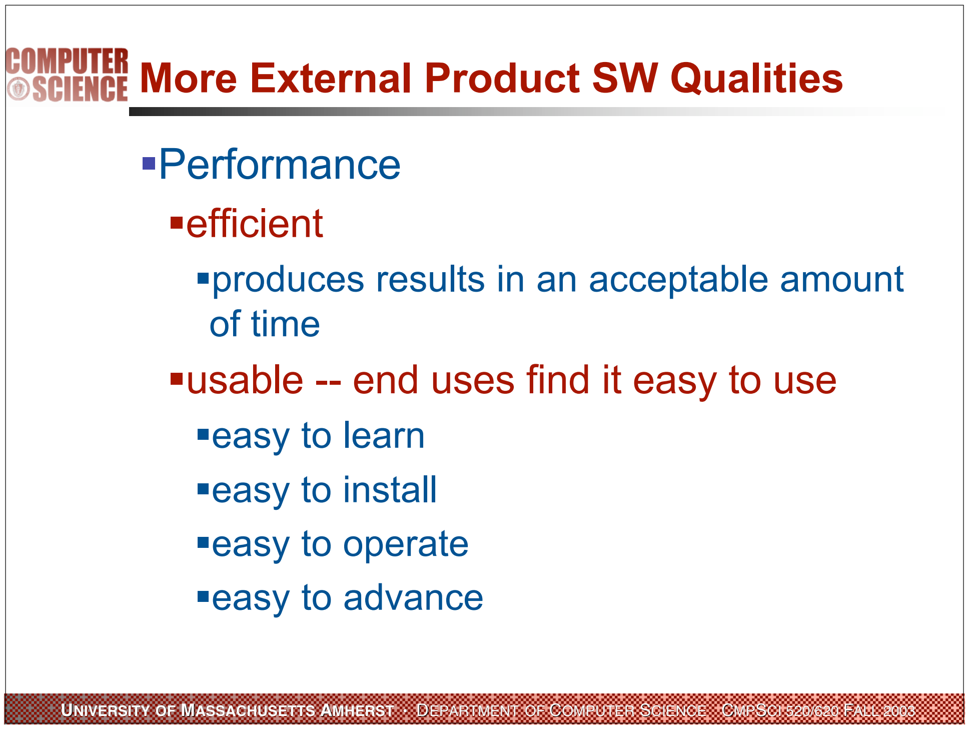


Correctness as an Internal Product Software Quality



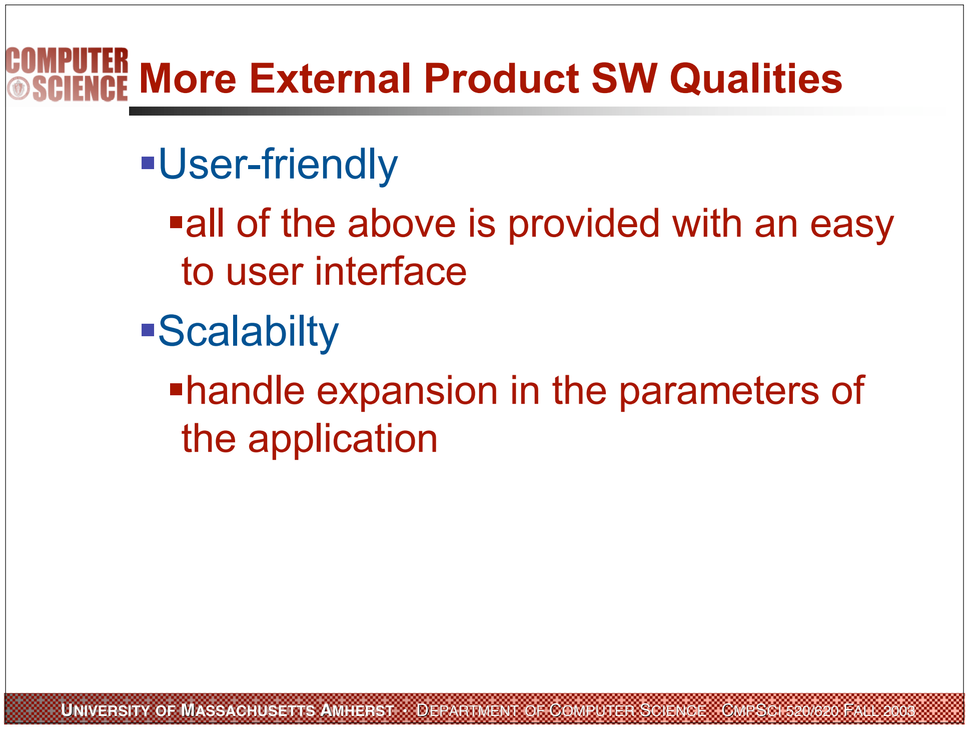
Verifiability

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



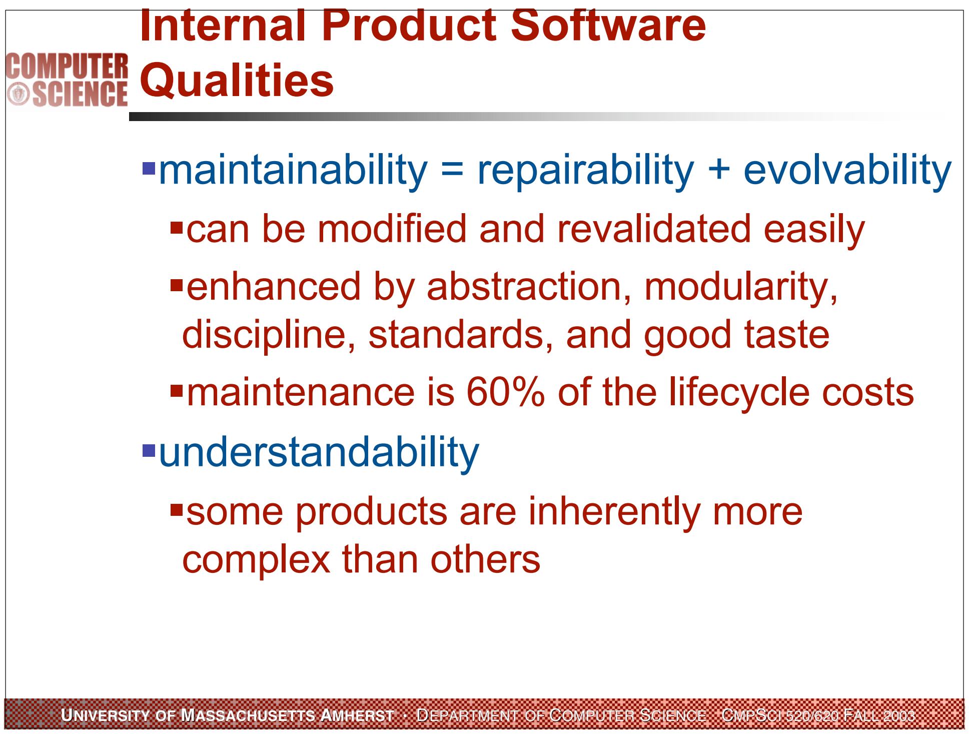
More External Product SW Qualities

- 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



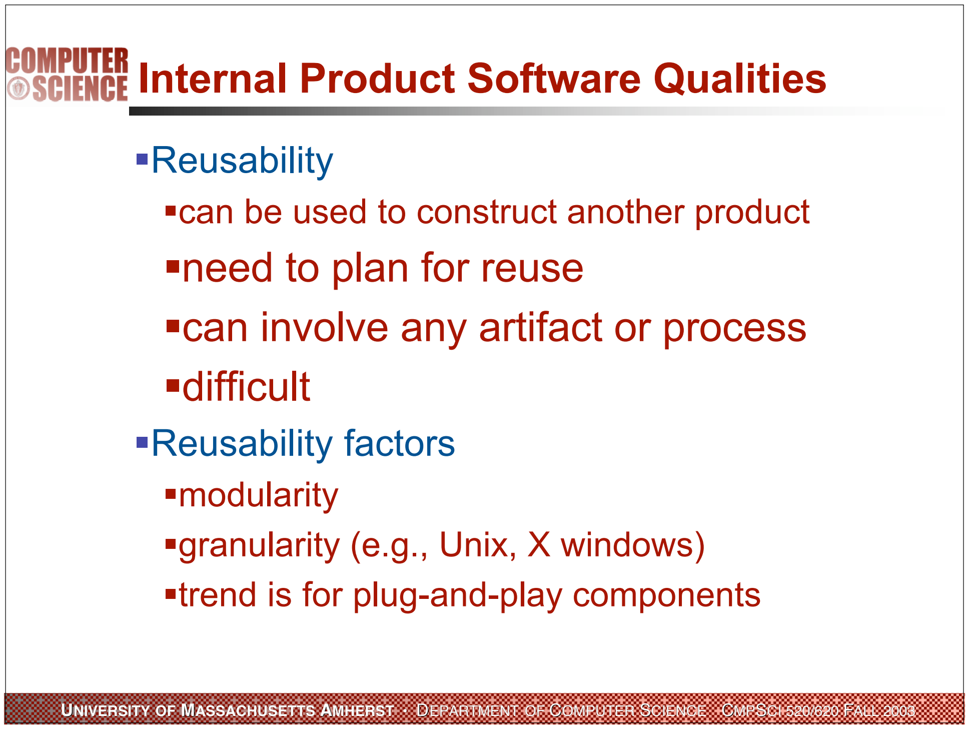
More External Product SW Qualities

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



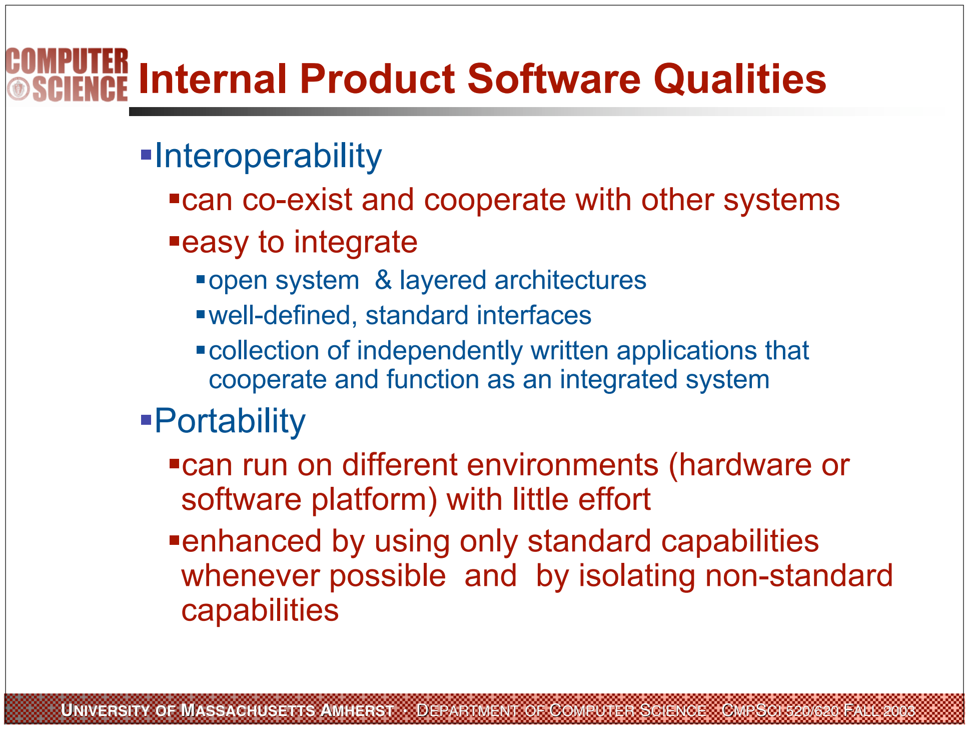
Internal Product Software Qualities

- 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



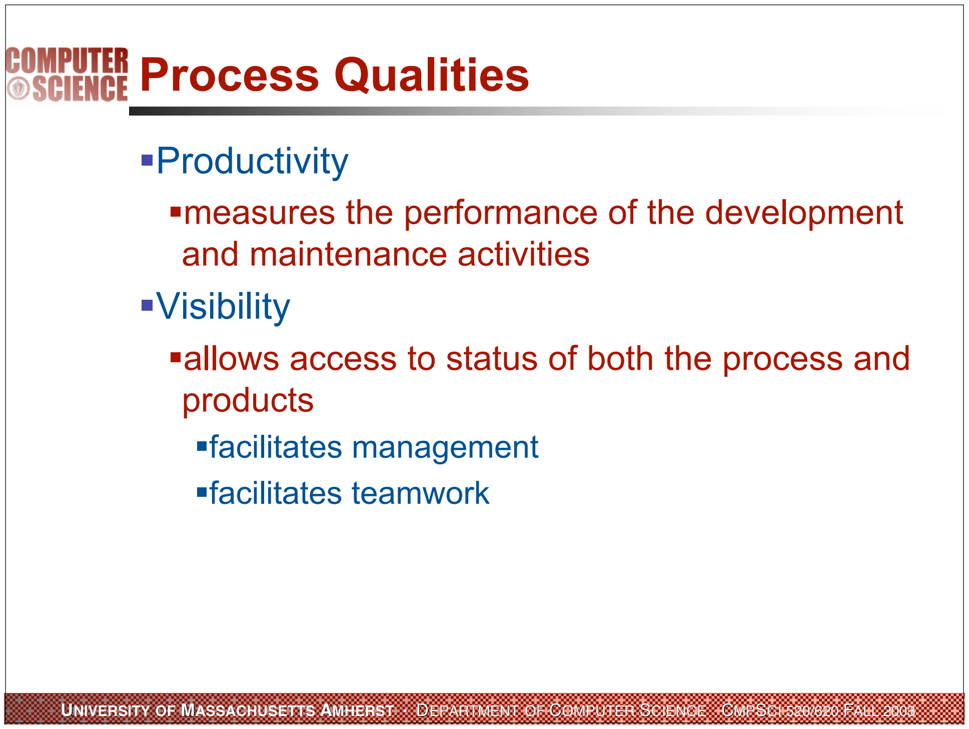
Internal Product Software Qualities

- 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



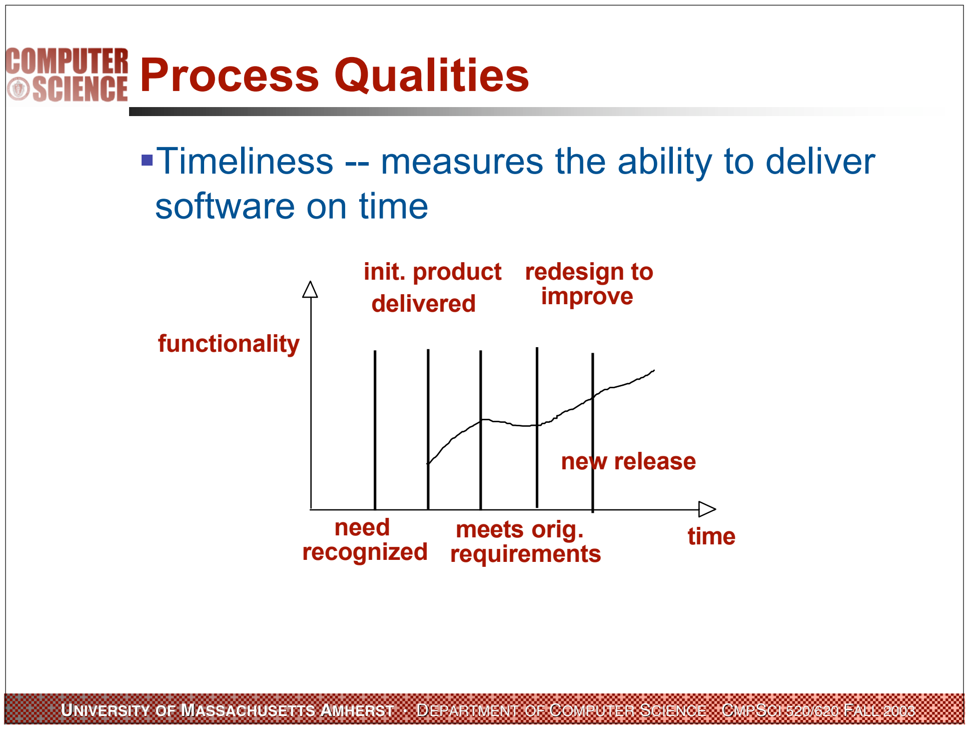
Internal Product Software Qualities

- 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



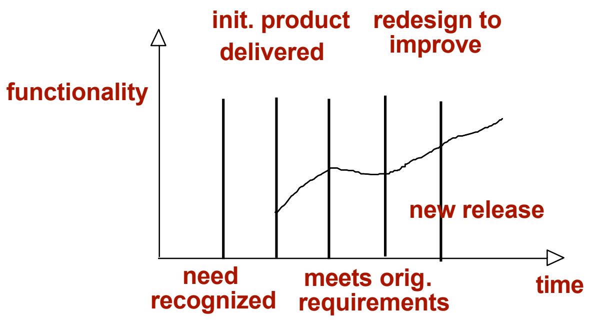
Process Qualities

- 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

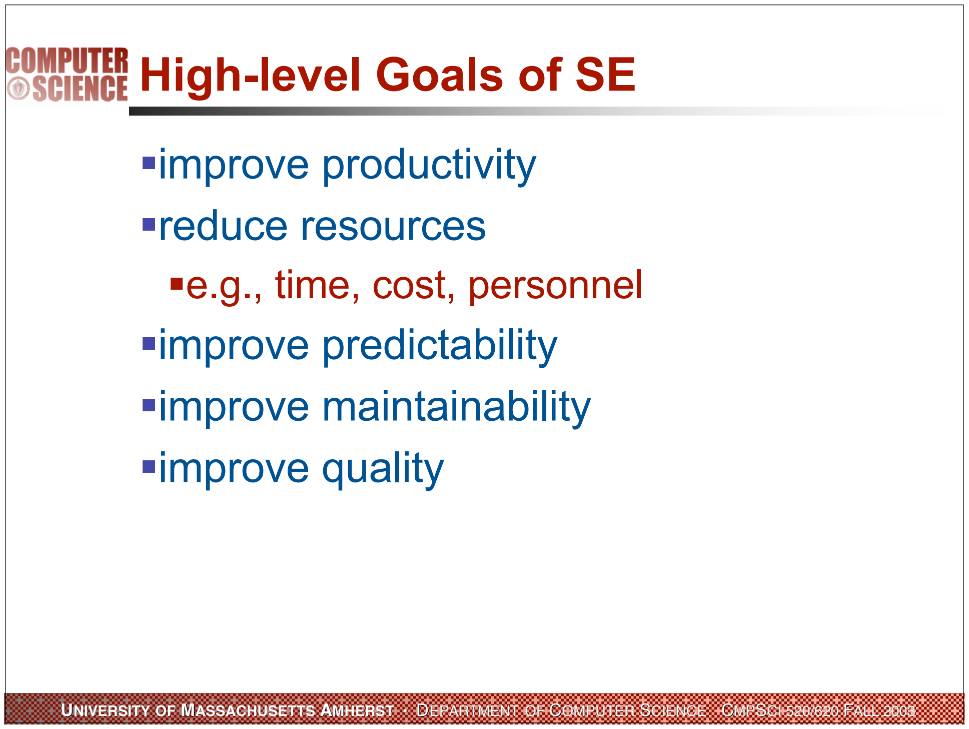


Process Qualities

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



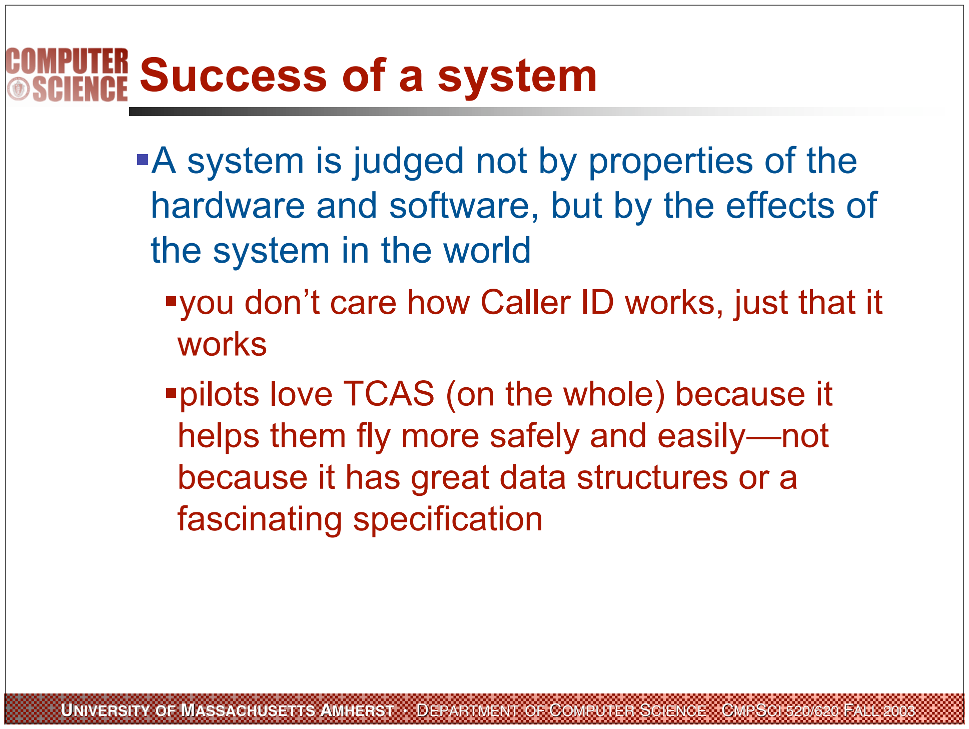
UNIVERSITY OF MASSACHUSETTS AMHERST · DEPARTMENT OF COMPUTER SCIENCE · CMPSCI 520/620 FALL 2003



High-level Goals of SE

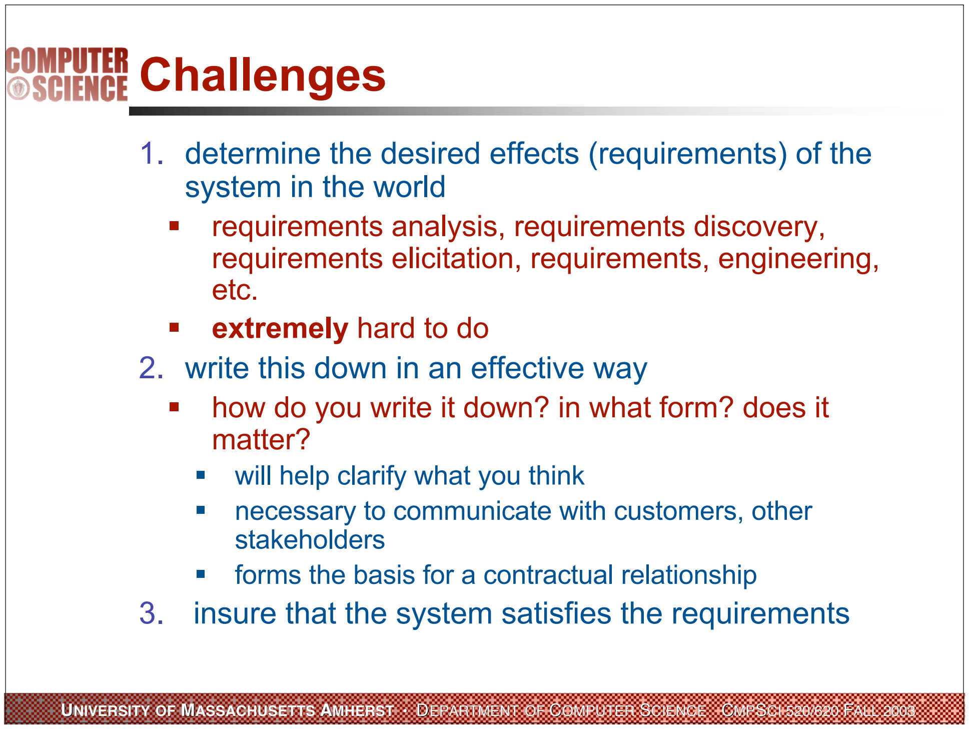
- improve productivity
- reduce resources
  - e.g., time, cost, personnel
- improve predictability
- improve maintainability
- improve quality

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



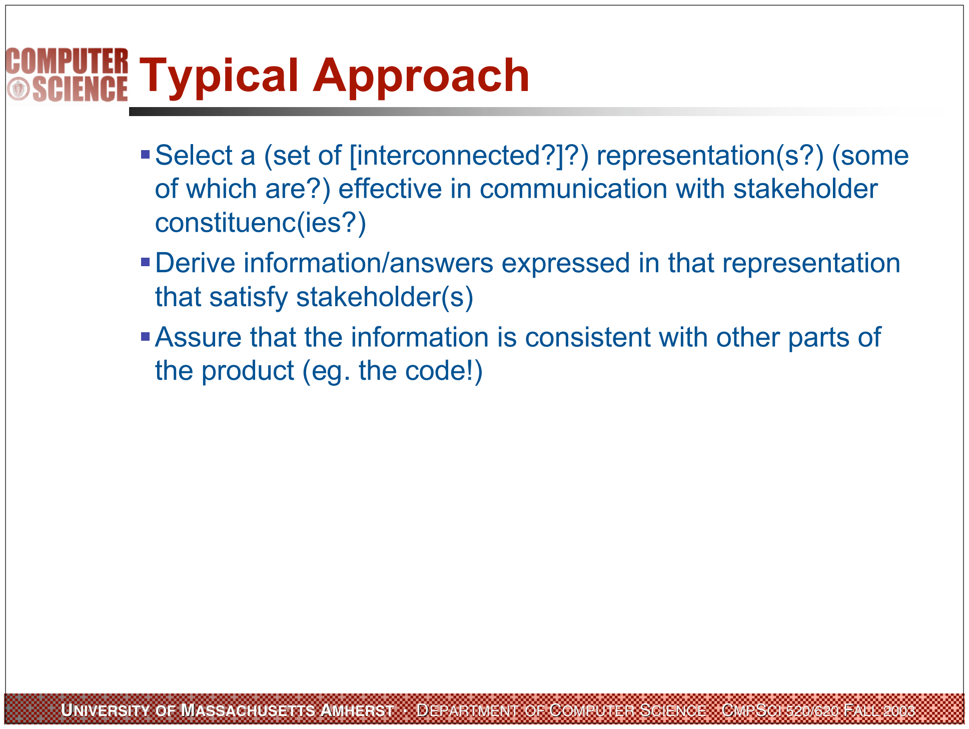
Success of a system

- 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



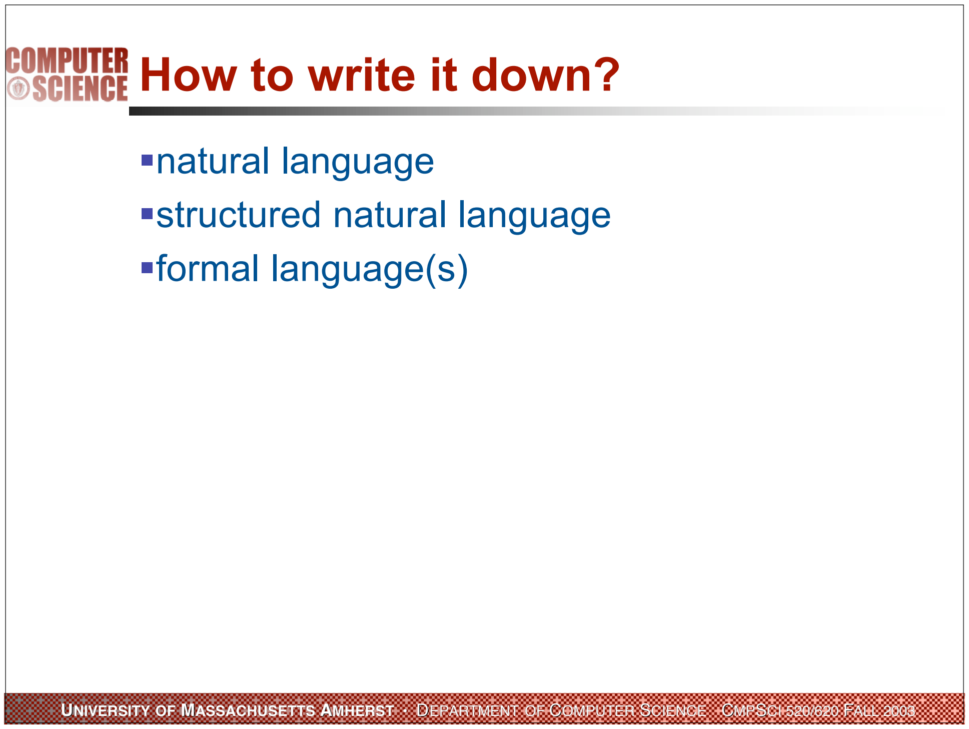
Challenges

1. determine the desired effects (requirements) of the system in the world
   - requirements analysis, requirements discovery, requirements elicitation, requirements, engineering, etc.
   - **extremely** hard to do
2. write this down in an effective way
   - how do you write it down? in what form? does it matter?
     - will help clarify what you think
     - necessary to communicate with customers, other stakeholders
     - forms the basis for a contractual relationship
3. insure that the system satisfies the requirements



Typical Approach

- Select a (set of [interconnected?]?) representation(s?) (some of which are?) effective in communication with stakeholder constituenc(ies?)
- Derive information/answers expressed in that representation that satisfy stakeholder(s)
- Assure that the information is consistent with other parts of the product (eg. the code!)



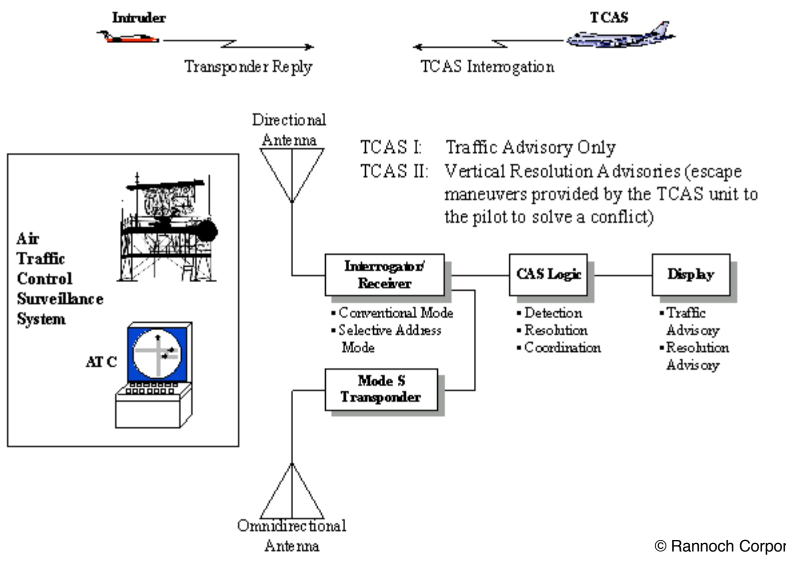
How to write it down?

- natural language
- structured natural language
- formal language(s)

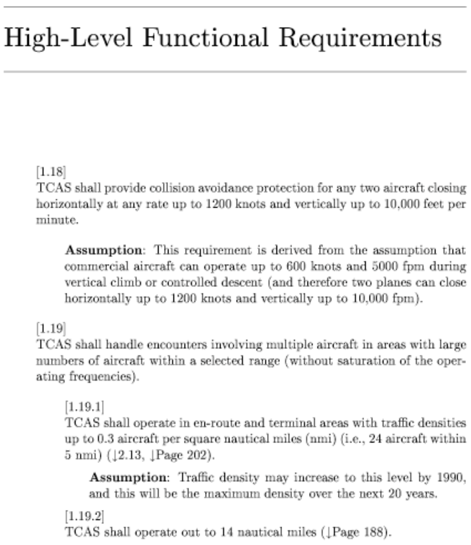
TCAS

- FAA needed proof that all collisions will be detected
- Statement of proof needed
- Other artifacts: body of proof, code, code structure representations, etc. Must be derived and shown to be consistent with each other

TCAS Overview



Levison “Intentional” Spec



High-Level Functional Requirements

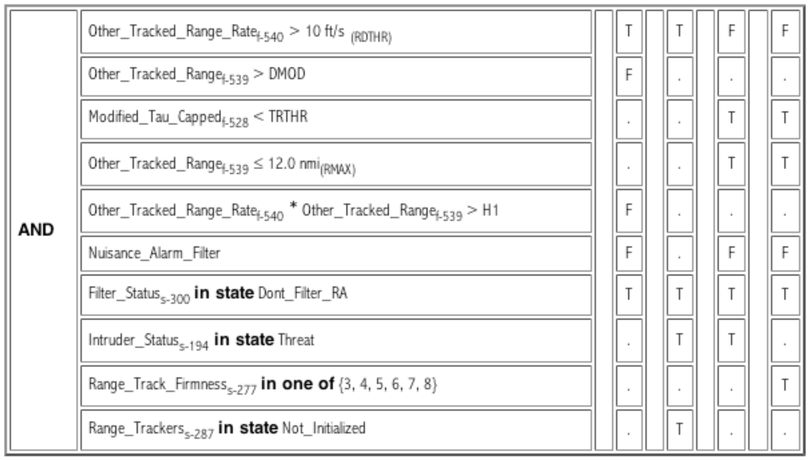
[1.18]
TCAS shall provide collision avoidance protection for any two aircraft closing horizontally at any rate up to 1200 knots and vertically up to 10,000 feet per minute.

> **Assumption**: This requirement is derived from the assumption that commercial aircraft can operate up to 600 knots and 5000 fpm during vertical climb or controlled descent (and therefore two planes can close horizontally up to 1200 knots and vertically up to 10,000 fpm).

[1.19]
TCAS shall handle encounters involving multiple aircraft in areas with large numbers of aircraft within a selected range (without saturation of the operating frequencies).

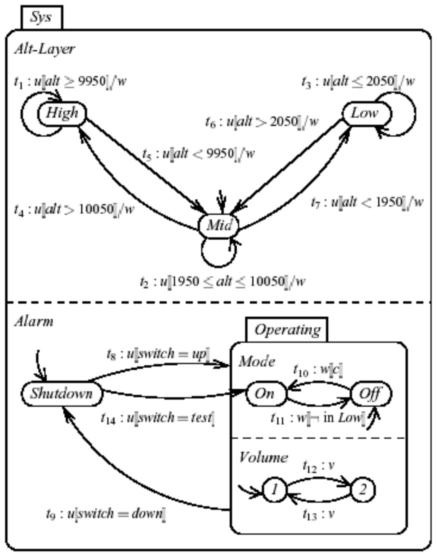
> [1.19.1]
> TCAS shall operate in en-route and terminal areas with traffic densities up to 0.3 aircraft per square nautical miles (nmi) (i.e., 24 aircraft within 5 nmi) (↓2.13, ↓Page 202).
>
> > **Assumption**: Traffic density may increase to this level by 1990, and this will be the maximum density over the next 20 years.
>
> [1.19.2]
> TCAS shall operate out to 14 nautical miles (↓Page 188).

TCAS Truth Tables for Threat_Range_Test

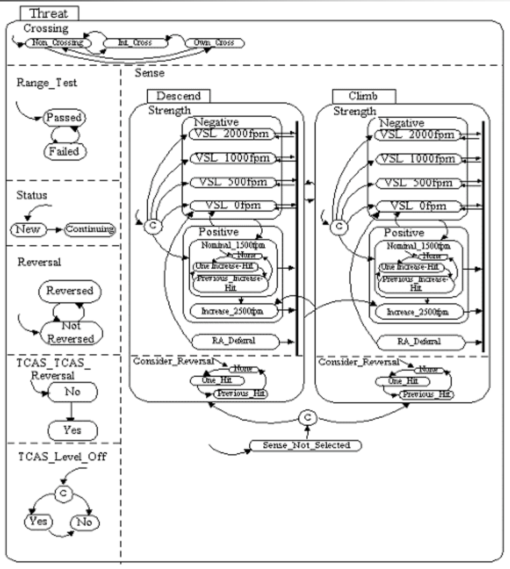


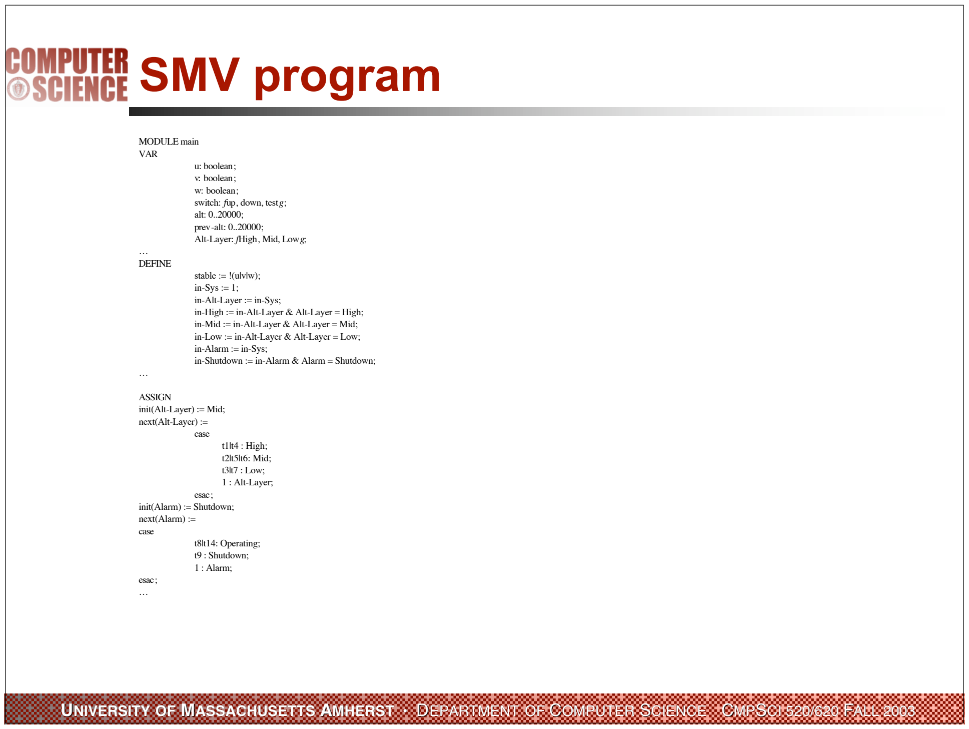
AND	Other_Tracked_Range_Rate$_{f\text{-}540}$ > 10 ft/s $_{(RDTHR)}$	T	T	F	F
	Other_Tracked_Range$_{f\text{-}539}$ > DMOD	F	.	.	.
	Modified_Tau_Capped$_{f\text{-}528}$ < TRTHR	.	.	T	T
	Other_Tracked_Range$_{f\text{-}539}$ ≤ 12.0 nmi$_{(RMAX)}$	.	.	T	T
	Other_Tracked_Range_Rate$_{f\text{-}540}$ * Other_Tracked_Range$_{f\text{-}539}$ > H1	F	.	.	.
	Nuisance_Alarm_Filter	F	.	F	F
	Filter_Status$_{s\text{-}300}$ **in state** Dont_Filter_RA	T	T	T	T
	Intruder_Status$_{s\text{-}194}$ **in state** Threat	.	T	T	.
	Range_Track_Firmness$_{s\text{-}277}$ **in one of** {3, 4, 5, 6, 7, 8}	.	.	.	T
	Range_Trackers$_{s\text{-}287}$ **in state** Not_Initialized	.	T	.	.

RMSL



TCAS Statechart

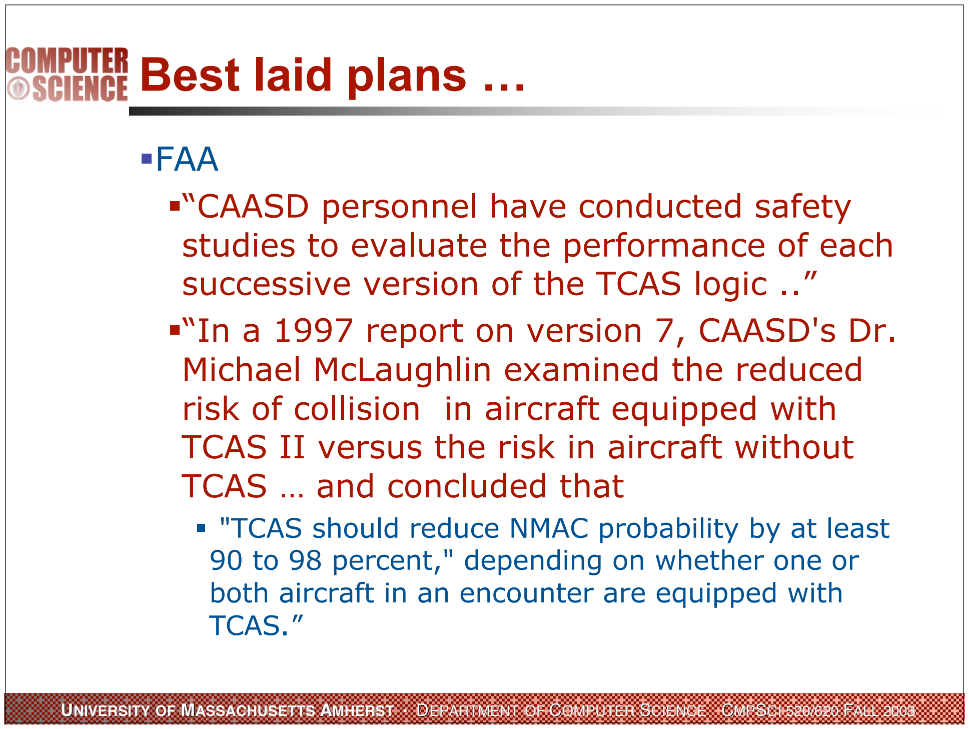




SMV program

```
MODULE main
VAR
        u: boolean;
        v: boolean;
        w: boolean;
        switch: fup, down, testg;
        alt: 0..20000;
        prev-alt: 0..20000;
        Alt-Layer: fHigh, Mid, Lowg;
…
DEFINE
        stable := !(u|v|w);
        in-Sys := 1;
        in-Alt-Layer := in-Sys;
        in-High := in-Alt-Layer & Alt-Layer = High;
        in-Mid := in-Alt-Layer & Alt-Layer = Mid;
        in-Low := in-Alt-Layer & Alt-Layer = Low;
        in-Alarm := in-Sys;
        in-Shutdown := in-Alarm & Alarm = Shutdown;
…

ASSIGN
init(Alt-Layer) := Mid;
next(Alt-Layer) :=
        case
            t1|t4 : High;
            t2|t5|t6: Mid;
            t3|t7 : Low;
            1 : Alt-Layer;
        esac;
init(Alarm) := Shutdown;
next(Alarm) :=
case
        t8|t14: Operating;
        t9 : Shutdown;
        1 : Alarm;
esac;
…
```



Best laid plans …

- FAA
  - "CAASD personnel have conducted safety studies to evaluate the performance of each successive version of the TCAS logic .."
  - "In a 1997 report on version 7, CAASD's Dr. Michael McLaughlin examined the reduced risk of collision in aircraft equipped with TCAS II versus the risk in aircraft without TCAS … and concluded that
    - "TCAS should reduce NMAC probability by at least 90 to 98 percent," depending on whether one or both aircraft in an encounter are equipped with TCAS."

… go oft astray

- The investigation into the chain of events behind mid-air collision over southern Germany has increasingly focused on the Swiss air traffic control agency Skyguide. Intially Skyguide blamed the Russian crew of one of the two aircraft for ignoring warnings to dive. But since then new important information has come to light: The pilot of the Russian Tu-154 was given conflicting instructions by air traffic control and his onboard computer The Russian pilot was given only 44 seconds warning A warning system at the control centre was switched off for maintenance Only one controller was on duty at the time The centre's radar system does not meet EU standards … BBC

How to write it down?

- natural language
- structured natural language
- pictorial notation
  - Box-and-Arrow Charts
  - Graphs
    - Flowgraphs
    - Parse Trees
    - Call graphs
    - Dataflow graphs
  - Charts, Diagrams
- data models
- formal language(s)
  - state-oriented
  - function-oriented
  - object-oriented

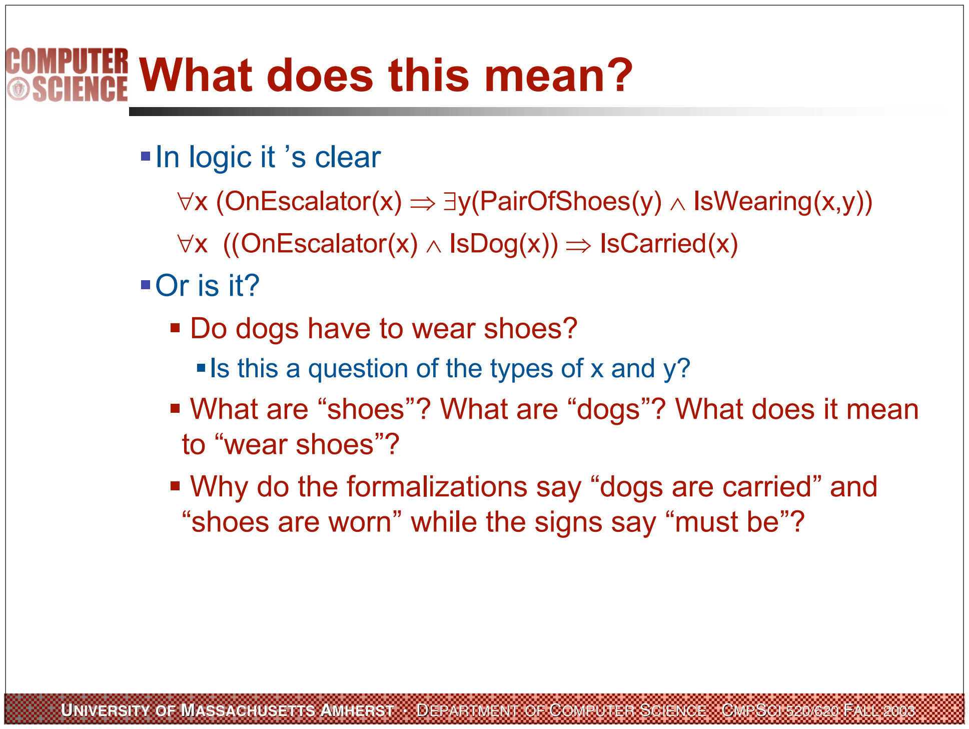
Which of these are best adapted to providing which types of answers to which types of stakeholders?

Natural Language

- Write in "plain English"
  - All stakeholders understand natural language (?)
  - Possible to augment with defined terms
  - Use of punctuation for clarification
  - Text/word processing systems help automate/maintain/alter
- Examples of Natural Language artifacts:
  - User manuals
  - Requirements specifications
  - Test Plans
  - Development status reports

Natural language

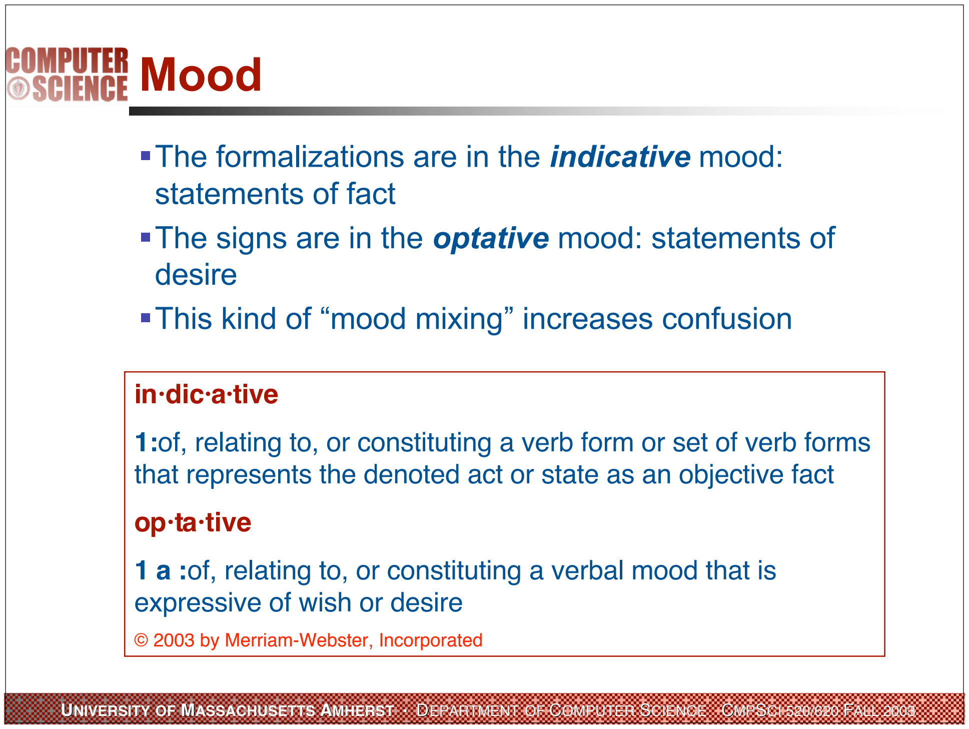
- Inherently ambiguous and also complex
- From one of Michael Jackson's books:
  - In an airport at the foot of an escalator are two signs
    - "Shoes must be worn."
    - "Dogs must be carried."



What does this mean?

- In logic it 's clear
  - $\forall x\ (OnEscalator(x) \Rightarrow \exists y(PairOfShoes(y) \land IsWearing(x,y))$
  - $\forall x\ ((OnEscalator(x) \land IsDog(x)) \Rightarrow IsCarried(x)$
- Or is it?
  - Do dogs have to wear shoes?
    - Is this a question of the types of x and y?
  - What are "shoes"? What are "dogs"? What does it mean to "wear shoes"?
  - Why do the formalizations say "dogs are carried" and "shoes are worn" while the signs say "must be"?

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



Mood

- The formalizations are in the ***indicative*** mood: statements of fact
- The signs are in the ***optative*** mood: statements of desire
- This kind of "mood mixing" increases confusion

in·dic·a·tive

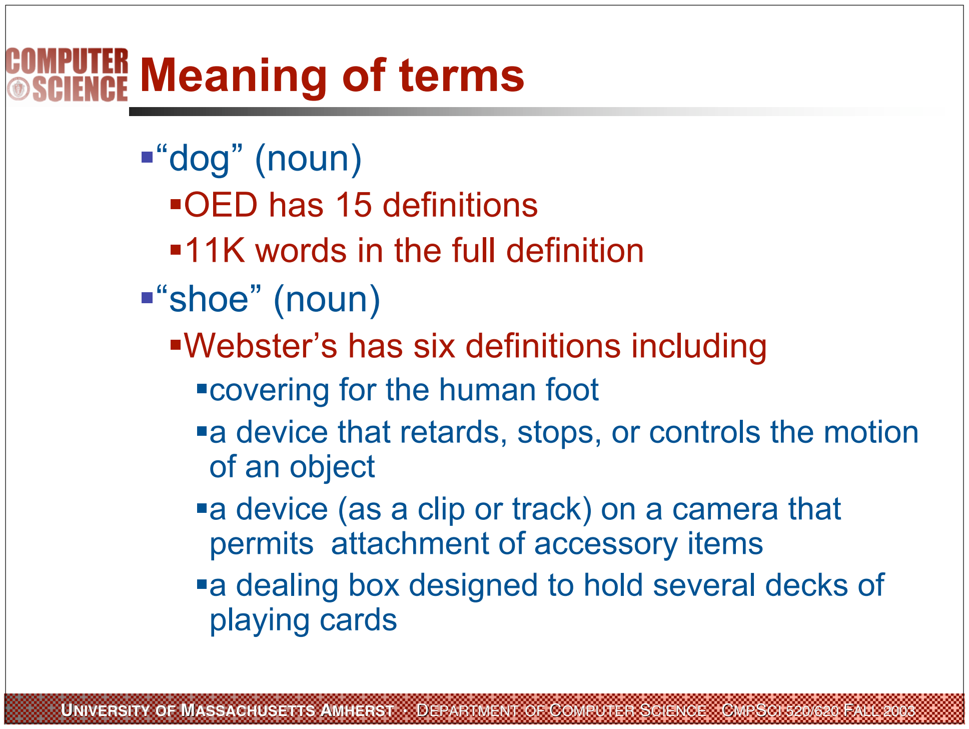
1:of, relating to, or constituting a verb form or set of verb forms that represents the denoted act or state as an objective fact

op·ta·tive

1 a :of, relating to, or constituting a verbal mood that is expressive of wish or desire

© 2003 by Merriam-Webster, Incorporated

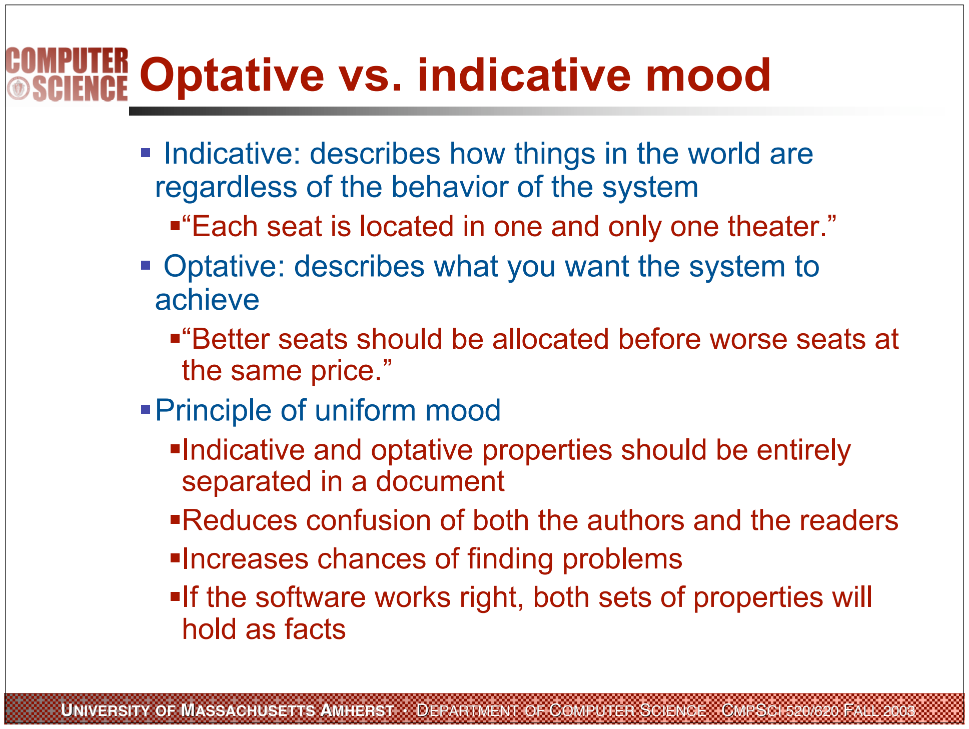
UNIVERSITY OF MASSACHUSETTS AMHERST • DEPARTMENT OF COMPUTER SCIENCE • CMPSCI 520/620 FALL 2003



Meaning of terms

- "dog" (noun)
  - OED has 15 definitions
  - 11K words in the full definition
- "shoe" (noun)
  - Webster's has six definitions including
    - covering for the human foot
    - a device that retards, stops, or controls the motion of an object
    - a device (as a clip or track) on a camera that permits attachment of accessory items
    - a dealing box designed to hold several decks of playing cards

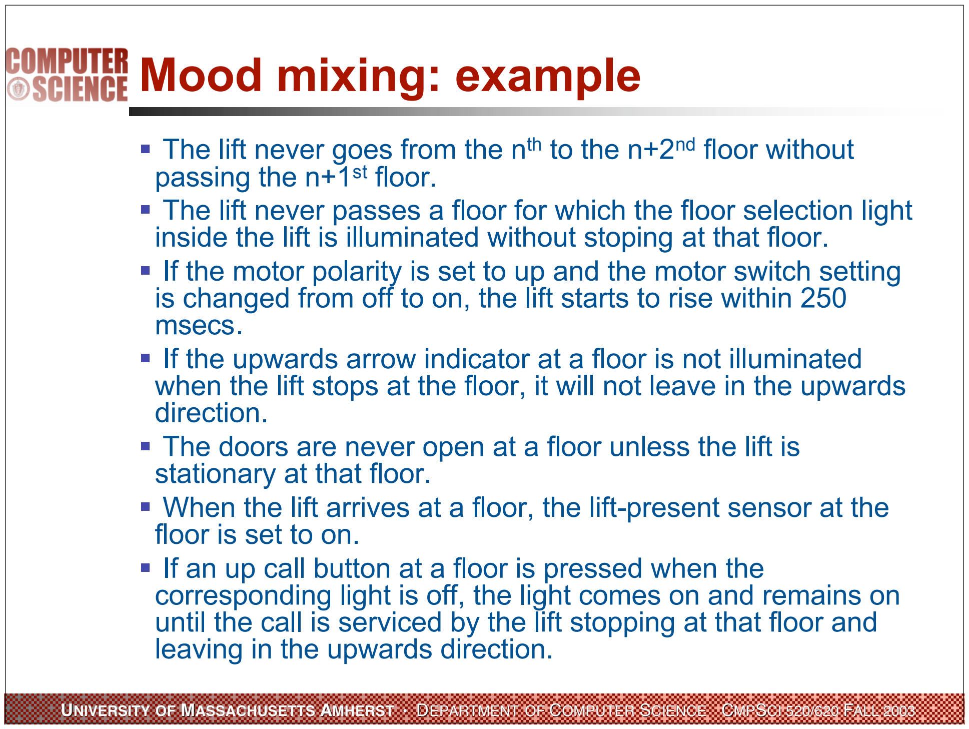
University of Massachusetts Amherst · Department of Computer Science · CMPSCI 520/620 Fall 2003



Optative vs. indicative mood

- Indicative: describes how things in the world are regardless of the behavior of the system
  - "Each seat is located in one and only one theater."
- Optative: describes what you want the system to achieve
  - "Better seats should be allocated before worse seats at the same price."
- Principle of uniform mood
  - Indicative and optative properties should be entirely separated in a document
  - Reduces confusion of both the authors and the readers
  - Increases chances of finding problems
  - If the software works right, both sets of properties will hold as facts

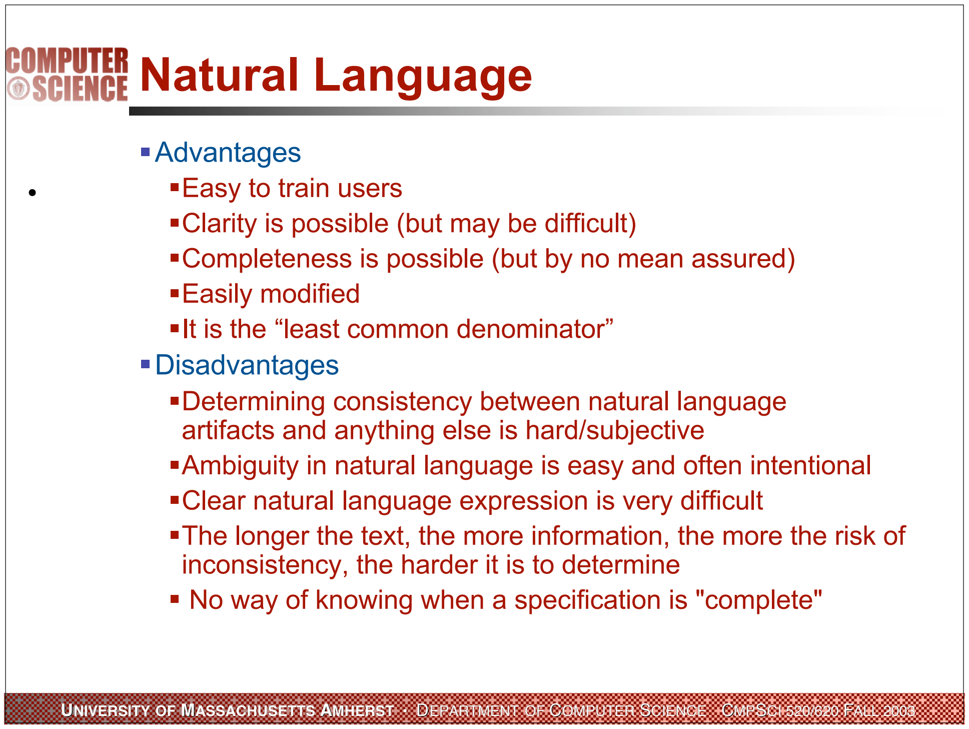
University of Massachusetts Amherst · Department of Computer Science · CMPSCI 520/620 Fall 2003



Mood mixing: example

- The lift never goes from the nth to the n+2nd floor without passing the n+1st floor.
- The lift never passes a floor for which the floor selection light inside the lift is illuminated without stoping at that floor.
- If the motor polarity is set to up and the motor switch setting is changed from off to on, the lift starts to rise within 250 msecs.
- If the upwards arrow indicator at a floor is not illuminated when the lift stops at the floor, it will not leave in the upwards direction.
- The doors are never open at a floor unless the lift is stationary at that floor.
- When the lift arrives at a floor, the lift-present sensor at the floor is set to on.
- If an up call button at a floor is pressed when the corresponding light is off, the light comes on and remains on until the call is serviced by the lift stopping at that floor and leaving in the upwards direction.

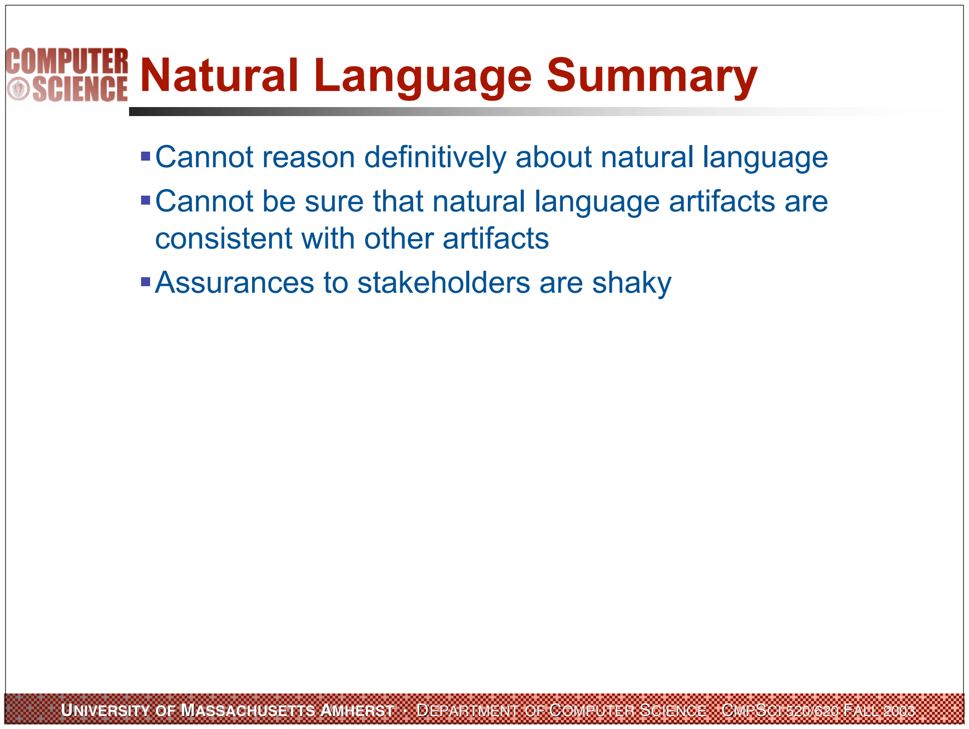
University of Massachusetts Amherst · Department of Computer Science · CMPSCI 520/620 Fall 2003



Natural Language

- Advantages
  - Easy to train users
  - Clarity is possible (but may be difficult)
  - Completeness is possible (but by no mean assured)
  - Easily modified
  - It is the “least common denominator”
- Disadvantages
  - Determining consistency between natural language artifacts and anything else is hard/subjective
  - Ambiguity in natural language is easy and often intentional
  - Clear natural language expression is very difficult
  - The longer the text, the more information, the more the risk of inconsistency, the harder it is to determine
  - No way of knowing when a specification is "complete"

University of Massachusetts Amherst · Department of Computer Science · CMPSCI 520/620 Fall 2003



Natural Language Summary

- Cannot reason definitively about natural language
- Cannot be sure that natural language artifacts are consistent with other artifacts
- Assurances to stakeholders are shaky

University of Massachusetts Amherst · Department of Computer Science · CMPSci 520/620 Fall 2003