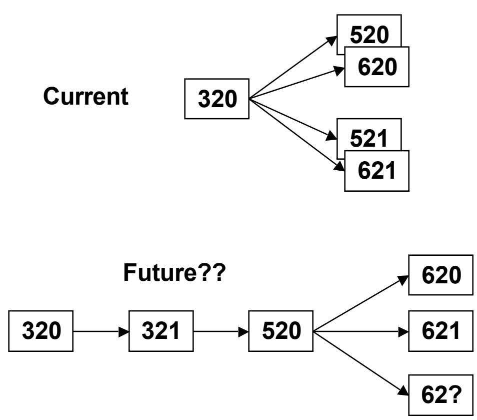
Welcome to 520/620

- Rick Adrion
  310 Computer Science Building
  (413) 545-2475
  adrion@cs.umass.edu
  Office Hours: Mondays 3-5pm & Tuesdays 9-11am
- Wendy Cooper
  312 Computer Science Building
  (413) 545-2492
  cooper@cs.umass.edu
- David (Ryan) Phillips
  drp@cs.umass.edu
- Other versions of these courses are offered through the Professional Education for Engineering and Applied Science (formerly the Video Instruction Program) for off-campus students. These 520X/620X courses are offered by Dean Leon Osterweil and will cover similar material, but with different lectures & assignments.

http://www-edlab.cs.umass.edu/cs520/

Background & prequisities



CMPSCI520/620

CMPSCI 520: Software Engineering: Synthesis and Development
Professor: Adrion
This course introduces students to the principal activities involved in developing high-quality software systems. The course stresses the use of defined, systematic processes in the creation of carefully defined and engineered software products. Among the topics covered are requirements analysis, software architecture, formal specification methods, process definition, software design methods, and test planning. Issues specific to the development of software by teams and groups will also be addressed. Students will be required to read selected papers from the literature and complete homework and projects. This course focuses on synthesis activities and complements CMPSCI 521, which focuses on software analysis techniques. Students are encouraged to take both courses but may take either course independently.
CMPSCI 620 : Advanced Software Engineering: Synthesis and Development
Professor: Adrion
As above. This course focuses on synthesis activities and complements CMPSCI 621, which focuses on software analysis techniques. Students are encouraged to take both courses but may take either course independently. Students who sign up for 620 will be expected to do more comprehensive projects, which emphasize critical and analytic thinking, than those who sign up for CMPSCI 520.

Learning Outcomes

- Knowledge and understanding
  - know and understand how software engineering principles and techniques apply to the software development process including:
    - the role of notation and abstraction in software engineering;
    - the products (artifacts) and processes which are fundamental to software development;
  - know and understand relevant techniques and methods that apply to each stage of the software development process, including:
    - Requirements engineering, elicitation, analysis, specification, validation and management;
    - Software design issues, software architecture, software design notations, strategies and methods;
    - Software construction, including styles (linguistic, mathematical and visual) and goals (managing complexity, anticipating diversity, structuring for validation);
    - Software process definition, measurement, analysis, implementation and improvement;
    - Software evolution, maintenance, re-engineering and reuse;
    - Software testing and analysis, including techniques, tools, planning and management.

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

Learning Outcomes (cont.)

- Cognitive (thinking) skills
  - perform problem analysis from written descriptions (analysis);
  - derive requirements and design specifications from an understanding of problems (analysis, synthesis);
  - create and/or justify designs to satisfy given requirements (synthesis, evaluation, application).
- Practical skills
  - evaluate the available options to select the most suitable technology for use in each stage of software development, underpinned by a knowledge of the efficacy of the various options;
  - apply systems, tools and techniques to support development, analysis and modeling appropriate to the different stages of software development.
- Transferable skills
  - communicate effectively by oral, written and visual means;
  - work effectively as an individual and as a member of a team;
  - perform independent and efficient time management;
  - perform independent information acquisition and management, using the scientific literature and Web sources;
  - prepare technical reports to a professional standard;
  - demonstrate understanding of personal responsibilities and professional codes of conduct

Examinations and Major Assignments

- The course assignments will include 4-6 homework assignments and 3 projects. **There are no examinations**; the scheduled final exam date/time is the last date/time that projects & homework assignments will be accepted without penalty. **I am usually flexible about late submission of assignments, but I will not accept a late submission of an assignment without prior approval.**
- Homework assignments will include “essay” and analytical problems which cover topics from the lectures and readings. Projects will be team projects on specifications and design, and may require the use of software tools.

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

Grading

- Attendance is not mandatory, but if you are absent from class, it is your responsibility to check on announcements & assignments made while you were away. **I do count class participation as 10% of the final grade**. Grades will be based on homework (45%), projects (45%) and class participation (10%). Final grades will be calculated with specific grades assigned to average scores in a more or less traditional manner (>93 = A; 88-92 = AB; 83-87 = B; 78-82 = BC; 73-77 =C; 68-72 = CD; 63-67 = D; <63 =F). I reserve the right to review the grade distribution and lower the ranges of scores for a given grade (e.g., perhaps 85-89 = AB; >90 = A).
- Students always expected to work independently, except when collaboration is explicitly expected (such as for the team projects). **Failure to do independent work may result in a failing grade for the assignment or, in some cases, for the course.**

Texts and readings

- Required text
  - *Requirements Analysis and System Design: Developing Information Systems with UML* by Leszek A. Maciaszek Pearson Addison Wesley; 1st edition (April 13, 2001)
- Suggested texts
  - *The Mythical Man-Month: Essays on Software Engineering, Anniversary Edition (2nd Edition)* by Frederick P. Brooks Addison-Wesley Pub Co; 1st edition (August 2, 1995)
  - *The Unified Modeling Language User Guide* by Grady Booch, Ivar Jacobson, James Rumbaugh Addison-Wesley Pub Co; 1st edition (September 30, 1998)
- Other interesting books
  - *Software Requirements & Specifications: A Lexicon of Practice, Principles and Prejudices* (ACM Press Books) by Michael Jackson Addison-Wesley Pub Co; 1st edition (August 23, 1995)
  - *Practical Software Requirements: A Manual of Content and Style* by Benjamin L. Kovitz Manning Publications Company; (December 1998)
  - *Software Design (2nd Edition)* by David Budgen Pearson Addison Wesley; 2nd edition (May 15, 2003)
  - *Design Patterns* by Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides Addison-Wesley Pub Co; 1st edition (January 15, 1995)
- Other reading
  - I will provide other reading, primarily papers from the literature.

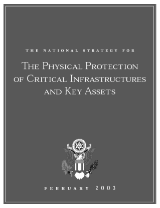
(Very) Tentative Calendar

Lec	Date	Due	Scheduled Lecture
1	9/3/03		Overview
2	9/8/03		Overview – Products & Processes
3	9/10/03		Historical Perspective
4	9/15/03		Rep.Products & Artifacts --Informal & Pictorial Approaches
5	9/17/03		Rep. Products & Artifacts --Dataflow & State Diagrams
6	9/22/03		Rep. Products & Artifacts -- State Machines & Petri Nets
7	9/24/03	HW #1	Rep. Products & Artifacts -- Mathematical & Logic Reps.
8	9/29/03		Introduction to UML
9	10/1/03		Introduction to UML
10	10/6/03		Requirements
11	10/8/03	HW #2	Requirements
	10/13/03		Columbus Day Holiday
12	10/15/03	Proj #1	Architecture
13	10/20/03		Design
14	10/22/03		Design
15	10/27/03	HW #3	Patterns & Frameworks
16	10/29/03		Development & Coding
17	11/3/03		Development & Coding
18	11/5/03		Analyzing Products
19	11/10/03	Proj. #2	Analyzing Products
20	11/12/03	HW #4	Representing Processes
21	11/17/03		Representing Processes
22	11/19/03		Managing Processes
23	11/24/03		Analyzing Processes
24	11/26/03	HW #5	Process Improvement
25	12/1/03		Version Control & Configuration Management
26	12/3/03		Reuse
27	12/8/03		Evolution & Maintenance
28	12/10/03	HW #6	Wrapup
		Proj #3	

01 - Overview

Rick Adrion

Software as Key Infrastructure



- In roads, bridges, schools, banks, hospitals......
- America's critical infrastructure
  - … provide the foundation for our national security, governance, economic vitality, and way of life
  - … their continued reliability, robustness, and resiliency create a sense of confidence and form an important part of our national identity and purpose
  - … frame our daily lives and enable us to enjoy one of the highest overall standards of living in the world.
- The facilities, systems, and functions that comprise our critical infrastructures
  - are highly sophisticated and complex
  - include human assets and physical and cyber systems that work together in processes that are highly interdependent

Software is the infrastructure in our infrastructure

Key Infrastructures

- **Water** 1,800 federal reservoirs; 1,600 municipal wastewater facilities
- **Public Health** 5,800 registered hospitals
- **Emergency Services** 87,000 U.S. localities
- **Defense Industrial Base** 250,000 firms in 215 distinct industries
- **Telecommunications** 2 billion miles of cable
- **Energy**
  - **Electricity** 2,800 power plants
  - **Oil and Natural Gas** 300,000 producing sites
- **Banking and Finance** 26,600 FDIC insured institutions
- **Postal and Shipping** 137 million delivery sites

Key Infrastructures

- **Transportation**
  - **Aviation** 5,000 public airports
  - **Passenger Rail and Railroads** 120,000 miles of major railroads
  - **Highways, Trucking, and Busing** 590,000 highway bridges
  - **Pipelines** 2 million miles of pipelines
  - **Maritime** 300 inland/costal ports
  - **Mass Transit** 500 major urban public transit operators
- **Chemical Industry and Hazardous Materials**
  - 66,000 chemical plants
- **Key Assets**
  - **National Monuments and Icons** 5,800 historic buildings
  - **Nuclear Power Plants** 104 commercial nuclear power plants
  - **Dams** 80,000 dams
  - **Government Facilities** 3,000 government owned/operated facilities
  - **Commercial Assets** 460 skyscrapers

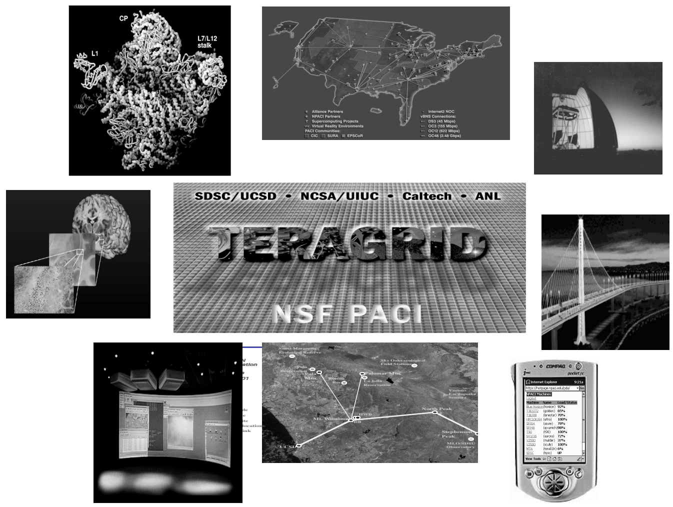
Software as Key Infrastructure for Science

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

What does the Future Look Like?

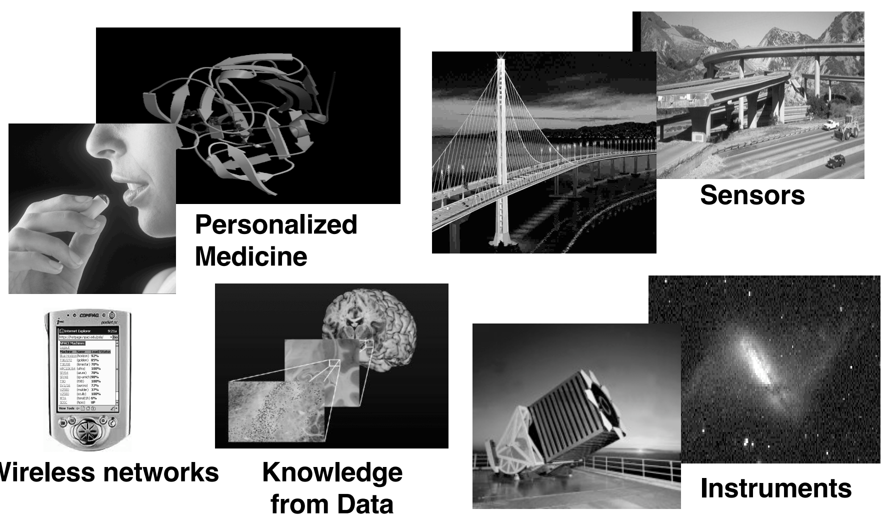
- Research
- Infrastructure
- People
- Data
- Software
- Hardware
- Instruments

- Future infrastructure drives today's research agenda



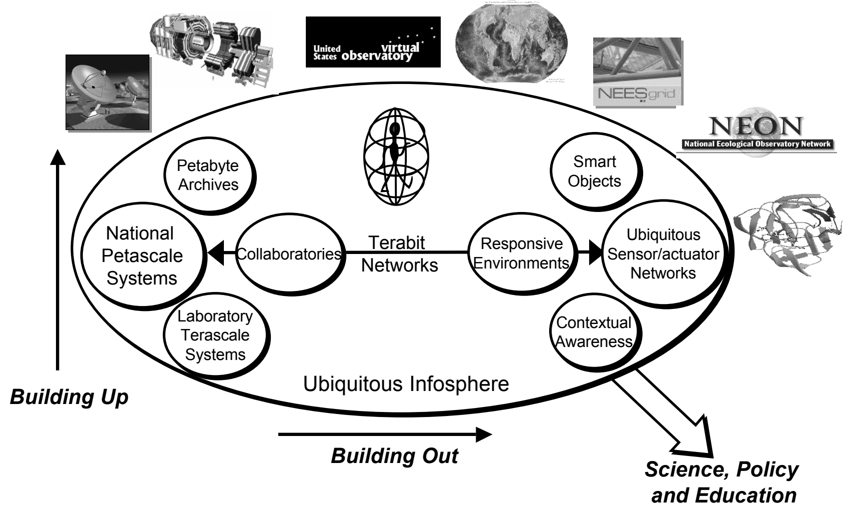
UNIVERSITY OF MASSACHUSETTS AMHERST • DEPARTMENT OF COMPUTER SCIENCE CMPSCI 520/620 FALL 2003

More Diversity, New Devices, New Applications



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

Futures: The Computing Continuum



The CRA Grand Challenges

- Create a Ubiquitous Safety.Net
  - A web of systems should be in place to mitigate the impact of and coordinate the response to natural and man-made disasters.
- Build a Team of Your Own
  - Humans will be able to pursue complex goals, aided by a team of robots and software agents to amplify physical capabilities and carry out specialized thought processes.
- Provide a Teacher for Every Learner
  - Students of all ages can receive virtual "one-on-one" instruction, tailored to their learning style, in an environment of unlimited digital resources.
- Build Systems You Can Count On
  - Reliability, security and availability should be second nature to the critical, complex information systems of tomorrow.
- Conquer System Complexity
  - We must overcome the complexity of large-scale information systems, which hinders their use and could prevent the preceding challenges from being realized.

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

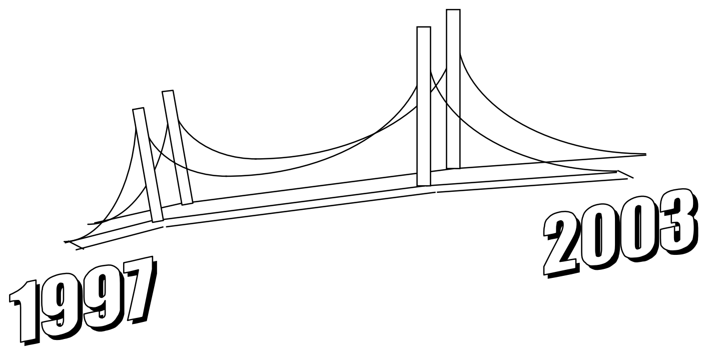
Software as Key Infrastructure

- And for most of the arts, humanities, and social sciences too:
  - large-scale longitudinal surveys
  - international and organizational databases, laboratories and collaboratories
  - methods for modeling behavior
  - techniques for incorporating spatial and biological information in models of human activity
  - historical databases
  - text analysis
  - statistics
  - visual representation
  - concordance
  - language study
  - textual and literary studies
  - access to worldwide collections (art, libraries, etc.)

Why Study Software Engineering?

- Practical engineering challenges
  - SW is
    - a critically important infrastructure component
    - a key enabler
      - militarily
      - economically
      - scientifically
      - culturally
  - BUT………..
    - expensive
    - usually of poor quality
  - It raises important scientific questions
    - foundation of the engineering solutions

SE: a bridge to the 21st century?



What is software engineering?

- is it engineering?
- is management?
- is it art?
- is it any kind of discipline?

What is software engineering?

- name coined at the NATO Science Committee Conference, October 1968
- engineering-- established, scientifically sound, practices that the typical practitioner follows

What is software engineering?

- software-- ALL associated documents to assist with the development, operation, validation, and maintenance of programs/software systems
  - e.g., code, documentation, designs, requirements, user manuals, installation manuals, test cases, test results, trouble reports, revision history, make files,...
- software engineering-- the application of scientific knowledge to the the development and maintenance of software systems

Other definitions

- Ghezzi: A field of computer science that deals with the building of software systems that:
  - are so large & complex to require team or teams
  - exist in multiple versions
  - used for many years
  - undergo changes
    - to repair defects
    - to enhance features
    - to add new features
    - to remove features
    - to adapt to new environment

More definitions

- Shaw defines engineering as
  - creating **cost-effective** solutions to **practical problems** by applying **scientific knowledge** to building **things** in the **service** of mankind
    - cost-effective not just solutions
    - practical, i.e., for a customer
    - things, i.e., artifacts
    - in the service, i.e., society

More definitions

- Parnas (original): multi-person construction of multi-version software
- Parnas (now): failed attempt by software researchers to stimulate the interest of engineers in software
  - Need computer scientists, but need software engineers who are educated differently, know how to apply "software science," know broader areas of knowledge, are skilled in the discipline of design and analysis and are licensed.

Why engineer software?

- **Scope and Impact on Society**
- Economics
- Quality

Scope and Impact on Society

- range of applications
  - simple ⇒ “super” systems
- current methods do not scale
- new and expanding applications
  - complex, embedded, real-time systems
  - computing/communications "convergence"
  - integrated & distributed information systems
  - “digital fabric”
- from Phillips:
  - TV can have 600Kbytes of code
  - VCR has 265Kbytes
  - Cell phone has 512Kbytes
  - Car radio has 64-256Kbytes

Scope and Impact on Society

- Safety
  - techniques are adapted from non-safety-critical
  - applications are adapted from non-safety-critical
- Security and privacy
  - tolerate breaches & failures
  - discourse, law way behind technology
- "O Ring effect"
  - technology removed for extended period of time
- "Firewall effect"
  - firewalls, barriers erected that limit the effectiveness & applicability, limit the evolution of technology

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

Scope and Impact on Society

- There have been many (in)famous software failures:
  - National long-distance telephone outage
  - London Ambulance Service disaster
  - European Space Agency booster destruction
- If you fly, your life depends on software
  - Airbus
  - AA flight in Cali, Columbia
- Your bank account depends on software
  - New York bank reconciliation failure
- Medical devices are controlled by software
  - Therac-20
- … and so on

Scope and Impact on Society

- Software Quality
  - 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

Why engineer software?

- Scope and Impact on Society
- **Economics**
- Quality

Economics

- A trillion dollar a year industry?
- A "make or break" key competency for
  - Many of the largest companies
  - Entire national economies
- Software can cost hundreds or thousands of dollars per line
  - Lifetime maintenance costs are higher still
- Software can represent major (the majority of) costs of
  - aircraft and spacecraft
  - communications systems
  - medical devices and care systems
  - a variety of surprising products
- Greatest costs may be due to software that is late or software that fails

Economics

- US losing the innovation race
  - Software is following hardware production "off shore" e.g., Singapore, Taiwan, India, Caribbean
  - Europe & Asia aim at market share applications
  - Japan concentrates on narrow application domains
    - SW factories
    - extensive reuse
    - high productivity/reliability
  - Significant installed base in US applications slows innovation

Economics

- One product industries
  - Lotus, Ingres
- One domain industries
  - Microsoft, Borland, TRW, Anderson, etc
- Over-commitment to proprietary systems in vertical markets
  - medical, finance, insurance
- Low investment in automation
- Slow to get technology to market
  - SRI->Xerox->Apple->Microsoft
  - Unix

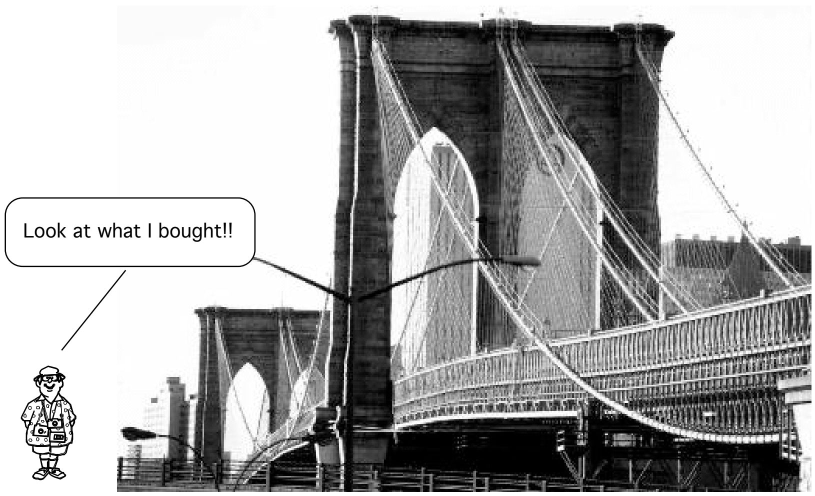
Why engineer software?

- Scope and Impact on Society
- Economics
- **Quality**

Quality

- deliver prototypes
- "leave it to the marketplace"
- unsophisticated consumers
  - don't know what they want
  - tolerate high failure rates
    - "the computer is down"
  - misled by "coverups" in banking, finance, communications
  - lack of understanding of risks
    - Year 2000 (Y2K)

Unsophisticated consumers



Quality

- Analogy
  - 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)

Barriers to engineering software

- **industry's short term focus**
- shortage of skilled personnel
- inadequate investment in R&D
- coupling -- tend to "toss over the fence"
- lack of infrastructure
- standards

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

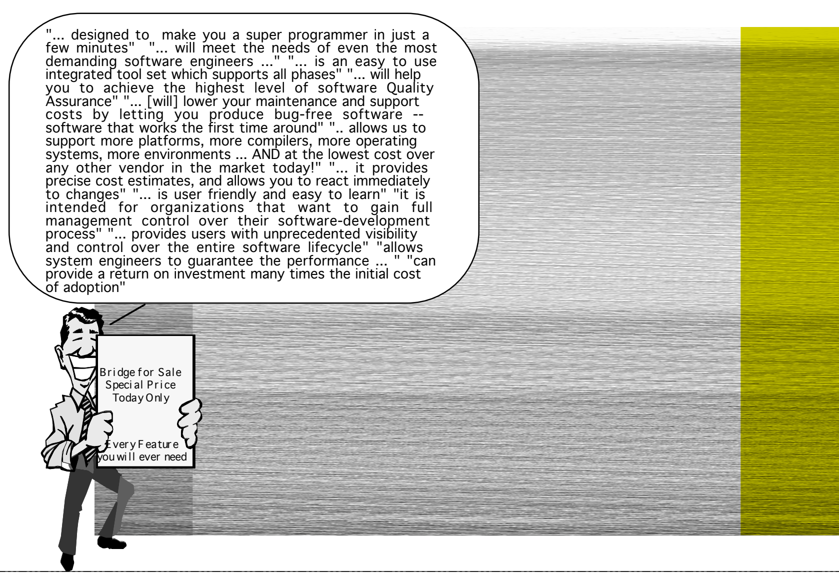
UNIVERSITY OF MASSACHUSETTS AMHERST • DEPARTMENT OF COMPUTER SCIENCE CMPSCI 520/620 FALL 2003

Industry’s short term focus

- 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

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

Overselling the technology



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

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
  - How to make accurate calendars?
    - Laws of planetary motion around the sun
  - How to cure colds and cancer?
    - Microbiology and molecular biology
  - How to keep bridges from falling down?
    - Statics, materials science
  - How to capture and store electricity
    - Electromagnetism

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?
- What do people do when they create software?
- How can adding more people can make things come out worse?
- What is software anyway?
  - Is it 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
- 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

What is the Nature of Software?

- It is an insensible, nonphysical
- It is a component of a larger system that "fits" with hardware, people, mechanical devices
- It transforms data using computers
- It changes constantly both proactively and reactively
- It has a complex structure
- It is usually very large, expensive, and lengthy to build

What's Different about Software?

- Software:
  - is extremely malleable – we can modify the product all too easily
  - Its construction is human-intensive, there are no real costs of materials
  - is intangible: no laws of physics are applicable
  - is not detectable by any of the five human senses
  - Its application horizons expand too fast with human demands/imagination
  - is unprecedentedly complex (a machine with millions of moving parts)
  - requires solutions of unusual rigor
  - has a discontinuous operational nature

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?
  - Preliminary design
    - architect sketches
    - plans from book
  - Detailed design
    - blueprints
  - Construction
    - Carpenters, plumbers, other skilled craftspeople
  - Maintenance & Evolution
    - instructions, manuals
    - additions/ remodelling as needed

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
  - 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
- 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?
  - 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

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: implementors (bureaucracies); courts
- Many stakeholders
  - effected citizens & (public, private) sectors; (federal, state & local) agencies & legislatures
- Complexity
  - Stakeholders unfamiliar with details
  - Side effects; unexpected outcomes
  - Seldom independent
  - Changes must be carefully planned

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)

Products?

Process?