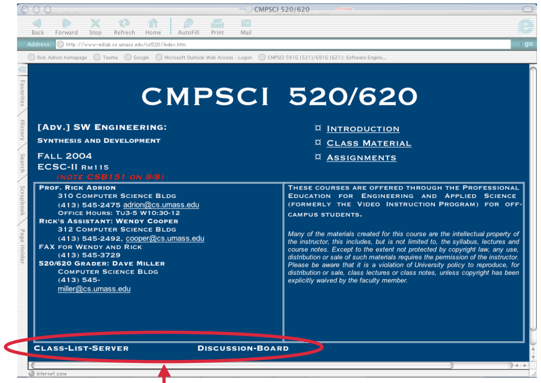
Welcome to 520/620

- Rick Adrion
  310 Computer Science Building
  (413) 545-2475
  adrion@cs.umass.edu
  Office Hours: Tuesdays 3-5 pm & Wednesdays 10:30 am - noon
- Wendy Cooper (on maternity leave until November)
  312 Computer Science Building
  (413) 545-2492
  cooper@cs.umass.edu
- TA: Dave Miller
  miller@cs.umass.edu

These courses are offered through the Professional Education for Engineering and Applied Science (formerly the Video Instruction Program) for off-campus students.

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



CMPSCI 520/620

[Adv.] SW Engineering:
Synthesis and Development
Fall 2004
ECSC-II Rm115
(note CSB151 on 9/8)

- ¤ Introduction
- ¤ Class Material
- ¤ Assignments

Prof. Rick Adrion
310 Computer Science Bldg
(413) 545-2475 adrion@cs.umass.edu
Office Hours: Tu3-5 W10:30-12
Rick's Assistant: Wendy Cooper
312 Computer Science Bldg
(413) 545-2492, cooper@cs.umass.edu
FAX for Wendy and Rick
(413) 545-3729
520/620 Grader: Dave Miller
Computer Science Bldg
(413) 545-
miller@cs.umass.edu

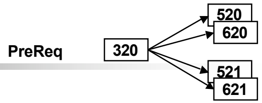
These courses are offered through the Professional Education for Engineering and Applied Science (formerly the Video Instruction Program) for off-campus students.

Many of the materials created for this course are the intellectual property of the instructor, this includes, but is not limited to, the syllabus, lectures and course notes. Except to the extent not protected by copyright law, any use, distribution or sale of such materials requires the permission of the instructor. Please be aware that it is a violation of University policy to reproduce, for distribution or sale, class lectures or class notes, unless copyright has been explicitly waived by the faculty member.

Class-List-Server Discussion-Board

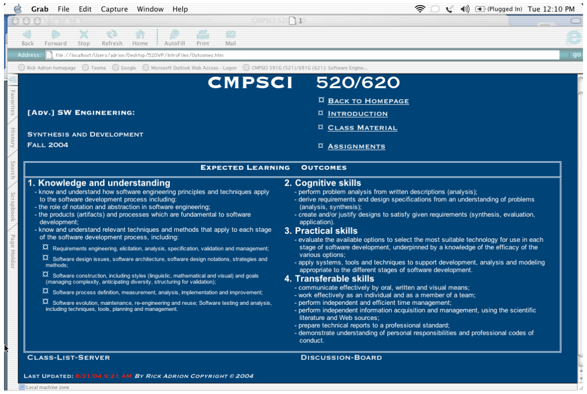
To be established

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


Grab
File
Edit
Capture
Window
Help
(Plugged In) Tue 12:10 PM
Back
Forward
Stop
Refresh
Home
AutoFill
Print
Mail
Address: file://localhost/Users/adrion/Desktop/520WP/IntroFiles/Outcomes.htm
go
Rick Adrion homepage
Teoma
Google
Microsoft Outlook Web Access - Logon
CMPSCI 591G (521)/691G (621): Software Engine...
Favorites
History
Search
Scrapbook
Page Holder
Local machine zone

CMPSCI 520/620

[Adv.] SW Engineering:

Synthesis and Development
Fall 2004

- ¤ Back to Homepage
- ¤ Introduction
- ¤ Class Material
- ¤ Assignments

Expected Learning Outcomes

1. 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.

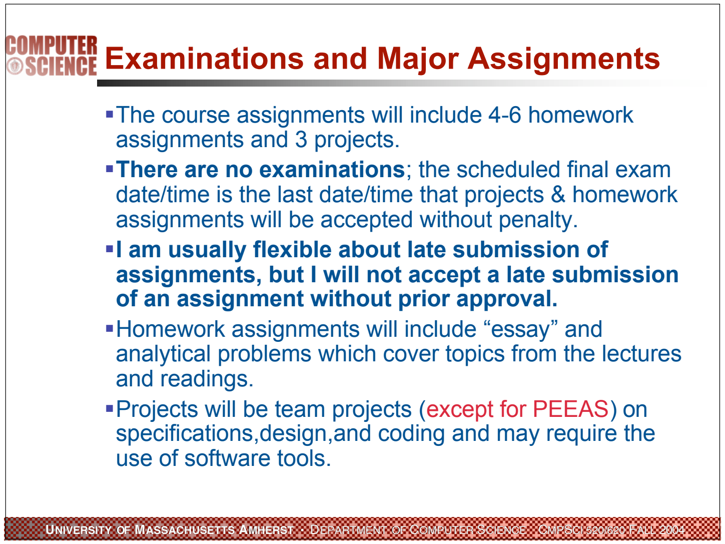
2. Cognitive skills
- perform problem analysis from written descriptions (analysis);
- derive requirements and design specifications from an understanding of problems (analysis, synthesis);
- create and/or justify designs to satisfy given requirements (synthesis, evaluation, application).

3. 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.

4. 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.

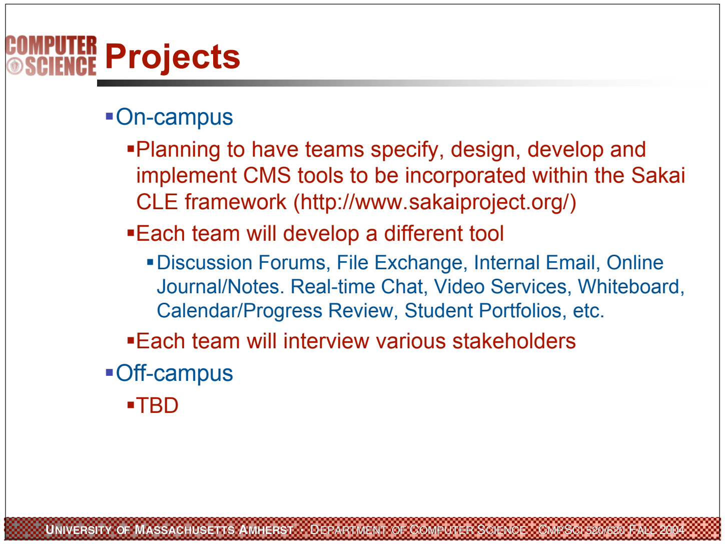
Class-List-Server Discussion-Board

Last Updated: 8/31/04 9:21 AM *By Rick Adrion Copyright © 2004*



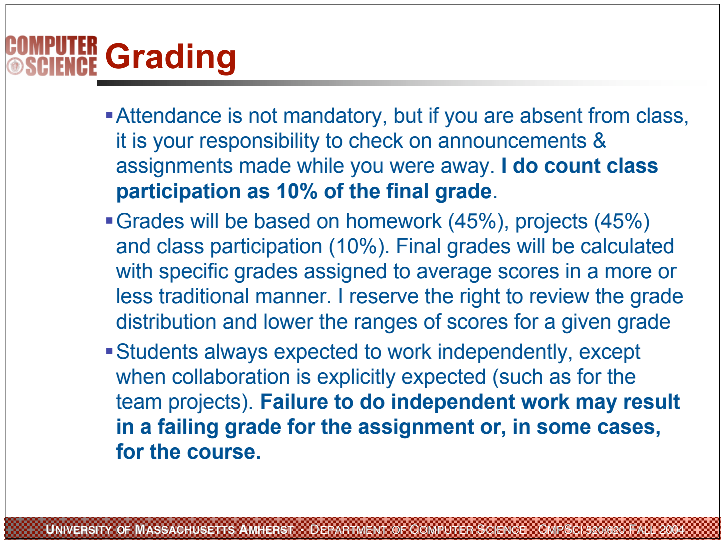
Examinations and Major Assignments

- 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 (except for PEEAS) on specifications,design,and coding and may require the use of software tools.



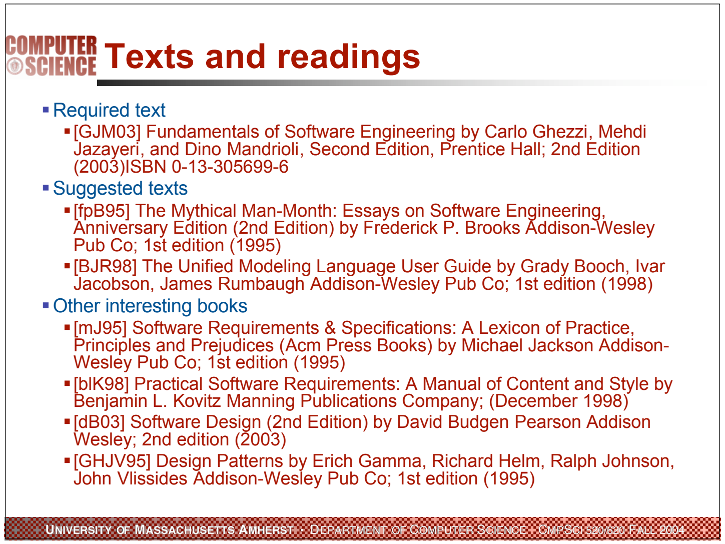
Projects

- On-campus
  - Planning to have teams specify, design, develop and implement CMS tools to be incorporated within the Sakai CLE framework (http://www.sakaiproject.org/)
  - Each team will develop a different tool
    - Discussion Forums, File Exchange, Internal Email, Online Journal/Notes. Real-time Chat, Video Services, Whiteboard, Calendar/Progress Review, Student Portfolios, etc.
  - Each team will interview various stakeholders
- Off-campus
  - TBD



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. I reserve the right to review the grade distribution and lower the ranges of scores for a given grade
- 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
  - [GJM03] Fundamentals of Software Engineering by Carlo Ghezzi, Mehdi Jazayeri, and Dino Mandrioli, Second Edition, Prentice Hall; 2nd Edition (2003)ISBN 0-13-305699-6
- Suggested texts
  - [fpB95] The Mythical Man-Month: Essays on Software Engineering, Anniversary Edition (2nd Edition) by Frederick P. Brooks Addison-Wesley Pub Co; 1st edition (1995)
  - [BJR98] The Unified Modeling Language User Guide by Grady Booch, Ivar Jacobson, James Rumbaugh Addison-Wesley Pub Co; 1st edition (1998)
- Other interesting books
  - [mJ95] Software Requirements & Specifications: A Lexicon of Practice, Principles and Prejudices (Acm Press Books) by Michael Jackson Addison-Wesley Pub Co; 1st edition (1995)
  - [blK98] Practical Software Requirements: A Manual of Content and Style by Benjamin L. Kovitz Manning Publications Company; (December 1998)
  - [dB03] Software Design (2nd Edition) by David Budgen Pearson Addison Wesley; 2nd edition (2003)
  - [GHJV95] Design Patterns by Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides Addison-Wesley Pub Co; 1st edition (1995)

Texts and readings

- Other reading
  - [fbB87] Brooks, Jr., F.P., “No Silver Bullet; Essence and Accidents of Software Engineering,” IEEE Computer , April 1987, pp.10--19.
  - [dH92] David Harel, “Biting the Silver Bullet,” Computer, January 1992
  - [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
  - [ljO87] L. J. Osterweii. “Software Processes are Software Too,” Proceedings of the Ninth International Conference if Software Engineering , March 1987.
  - [rW98] Roel Wieringa, “A survey of structured and object-oriented software specification methods and techniques,” ACM Computing Surveys December 1998
  - [jmW90] Wing, J. M., “A Specifier's Introduction to Formal Methods,” IEEE Computer , September 1990, pp.8--24.
  - [dlP72] D.L. Parnas, “On the Criteria to be Used in Decomposing Systems,” CACM. Vol. 15. No. 5. 1972. pp. 1.053-1.058.
  - [LZ75] Liskov, B.H. and Zilles, S.N., “Specification Techniques for Data Abstractions,” IEEE Transactions on Software Engineering , March 1975, pp.7--19.
  - [carH69] Hoare, C.A.R., “An Axiomatic Basis for Computer Programming,” Communications of the ACM , October 1969, 6 pages.
  - [GHW85] Guttag, J.V., Horning, J.J. and Wing, J.M., “The Larch family of Specification Languages,” IEEE Software , September 1985, pp.24--36.
- Project Documents

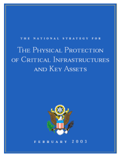
(Very) Tentative Calendar

Lec		5/620 Rdg	620 Rdg	Assgnmt Given	Date	Assgnmt Due
1	Overview	GJM03-Ch1, fpB95, fbB87, dH92			W September 8	
2	Products-Processes				M September 13	
3	Process				W September 15	
4	Quality-Notation				M September 20	
5	Notation- Graphs				W September 22	
6	Notation-Survey				M September 27	
7	Notation- State-Based				W September 29	
8	Notation- Formal				M October 4	
9	Requirements				W October 6	
	Columbus Day Holiday				*M October 11*	
10	Requirements				UMM October 13	
11	Requirements Specification				M October 18	
12	Requirements Analysis				W October 20	
13	Requirements Analysis				M October 25	
14	Software Architecture				W October 27	
15	Architecture, Patterns & Frameworks				M November 1	
16	Design				W November 3	
17	Design				M November 8	
18	Design				W November 10	
19	Analyzing Products				M November 15	
20	Analyzing Products				W November 17	
	“UMASS Thursday”				*M November 22*	
21	Analyzing Products				W November 24	
22	Representing, Managing & Analyzing Processes				M November 29	
23	Requirements Engineering				W December 1	
24	Reuse, Evolution & Maintenance				M December 6	
25	TBA				W December 8	
26	TBA				M December 13	

01-Introduction

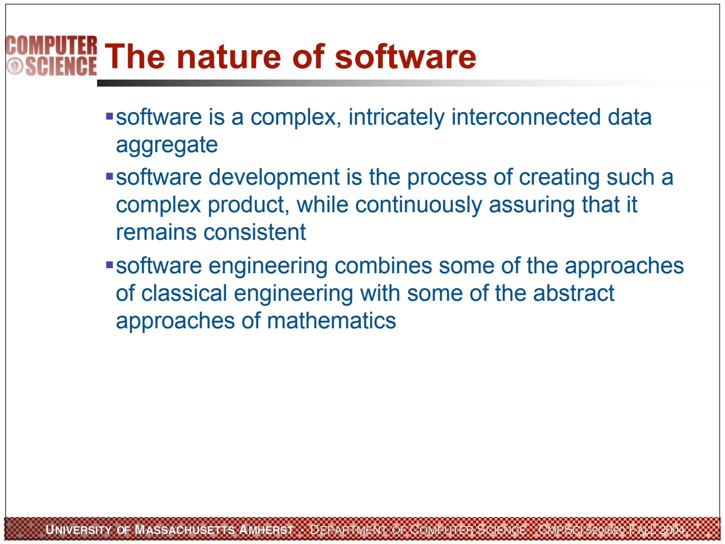
- Readings:
  - **Fundamentals of Software Engineering**, Ch. 1
  - **The Mythical Man-Month: Essays on Software Engineering, Anniversary Edition**
  - “No Silver Bullet; Essence and Accidents of Software Engineering,” **Computer**, April 1987
  - “Biting the Silver Bullet,” **Computer**, January 1992

Software as Key Infrastructure


THE NATIONAL STRATEGY FOR
THE PHYSICAL PROTECTION OF CRITICAL INFRASTRUCTURES AND KEY ASSETS
FEBRUARY 2003

- 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

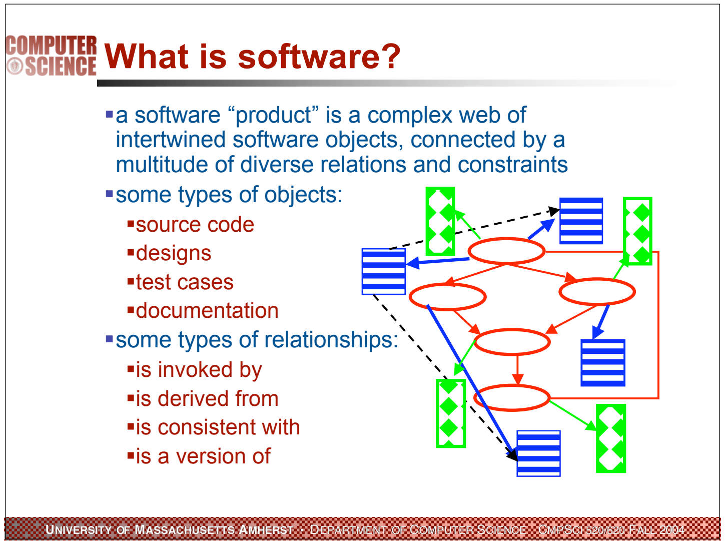
Software is the infrastructure in our infrastructure



The nature of software

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

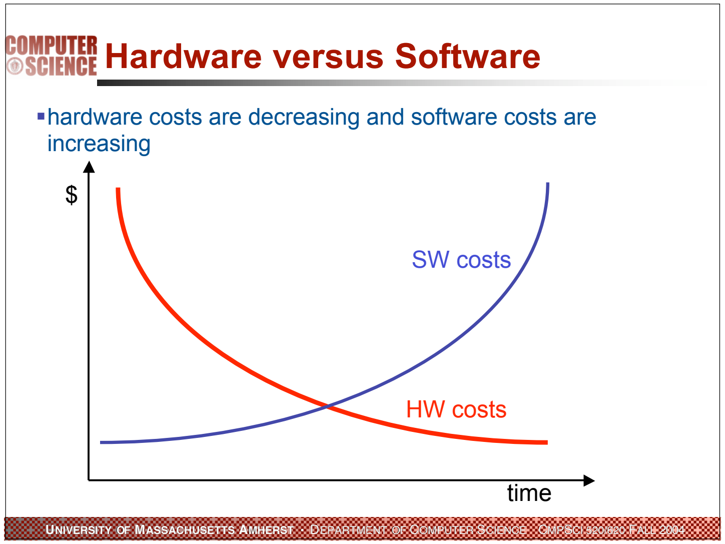
University of Massachusetts Amherst · Department of Computer Science · CMPSCI 520/620 Fall 2004



What is software?

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

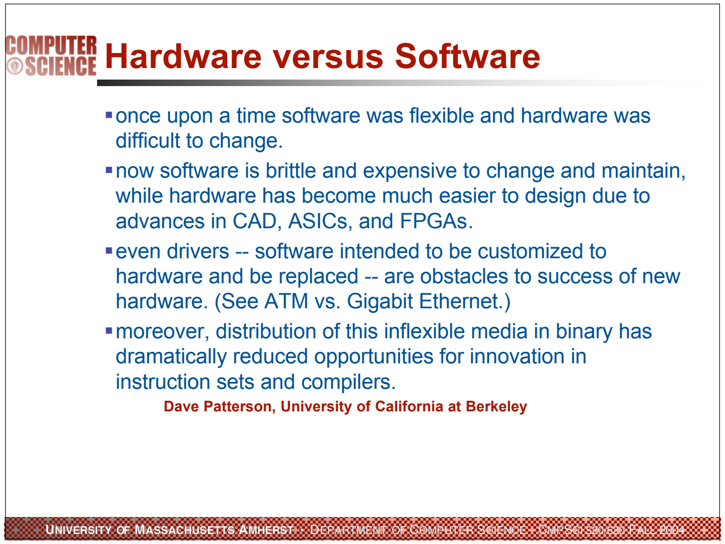
University of Massachusetts Amherst · Department of Computer Science · CMPSCI 520/620 Fall 2004



Hardware versus Software

- hardware costs are decreasing and software costs are increasing

University of Massachusetts Amherst · Department of Computer Science · CMPSCI 520/620 Fall 2004

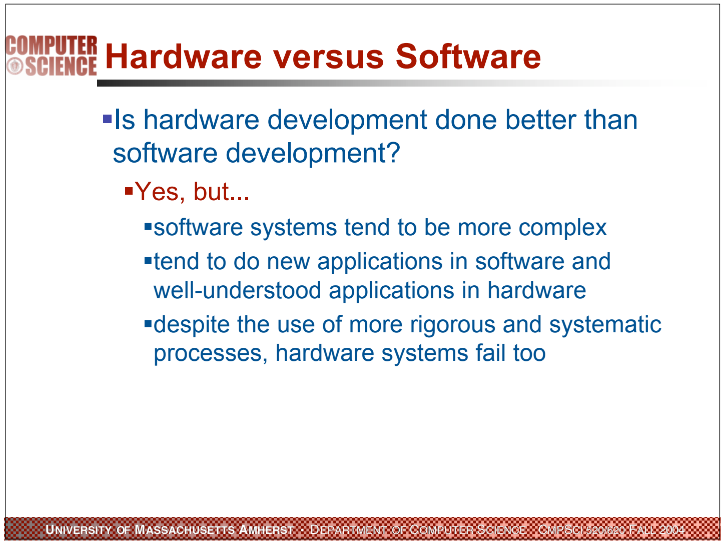


Hardware versus Software

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

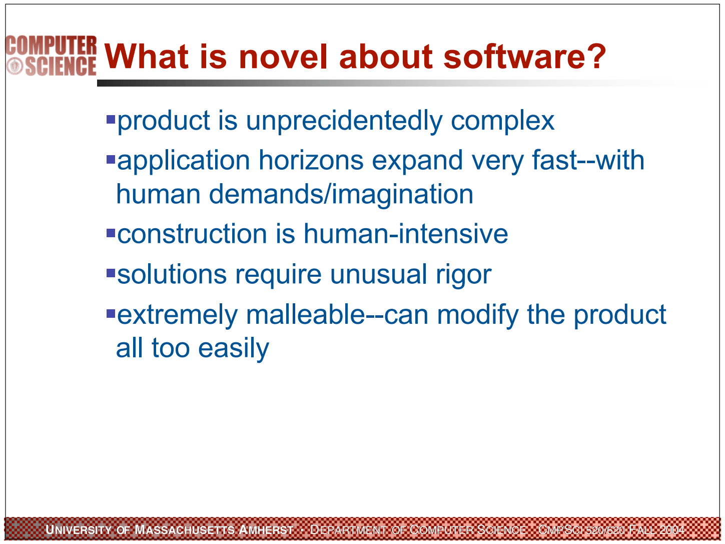
Dave Patterson, University of California at Berkeley

University of Massachusetts Amherst · Department of Computer Science · CMPSCI 520/620 Fall 2004



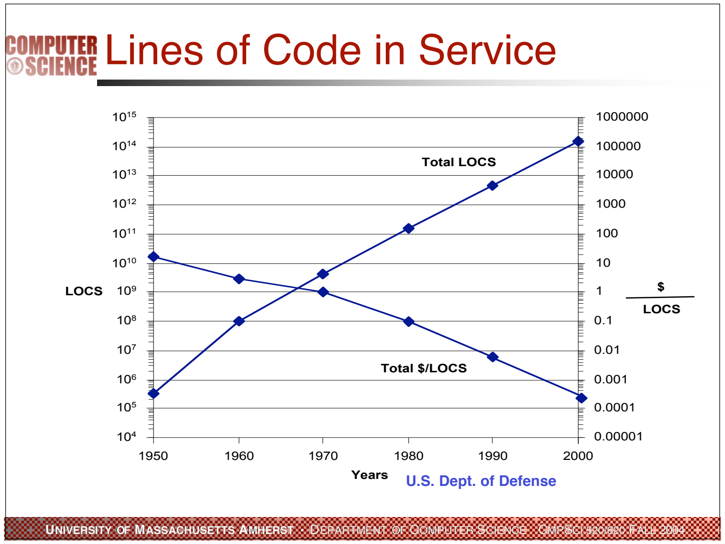
Hardware versus Software

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



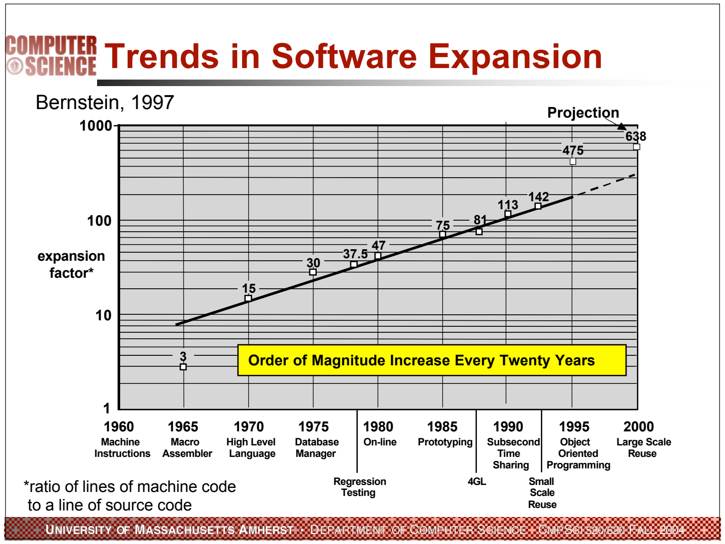
What is novel about software?

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



Lines of Code in Service

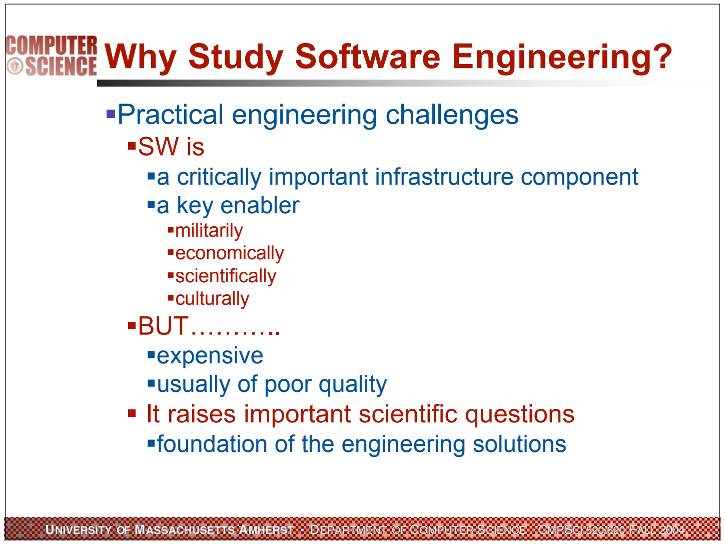
U.S. Dept. of Defense



Trends in Software Expansion

Bernstein, 1997

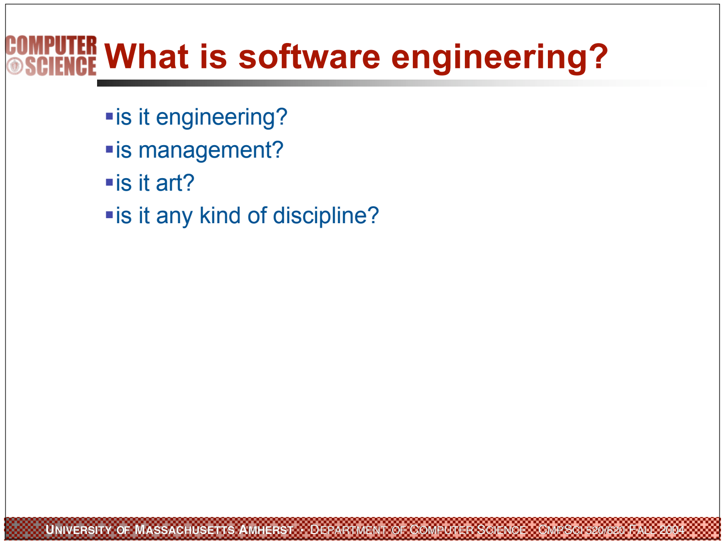
*ratio of lines of machine code to a line of source code



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

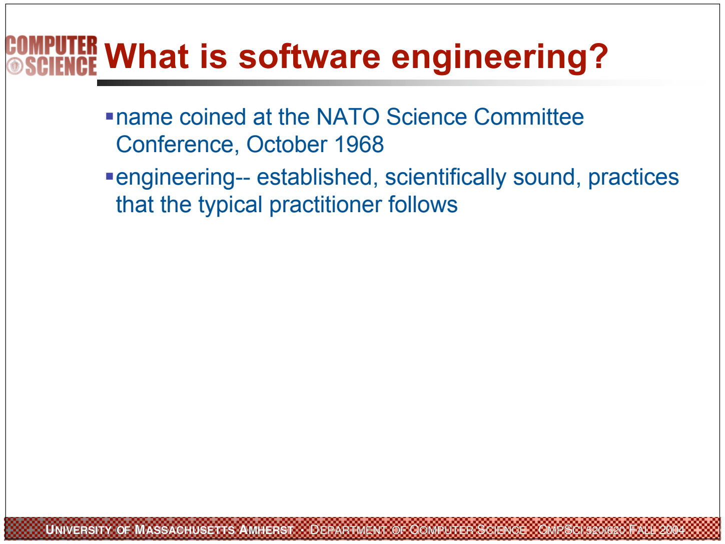
UNIVERSITY OF MASSACHUSETTS AMHERST • DEPARTMENT OF COMPUTER SCIENCE • CMPSCI 520/620 FALL 2004



What is software engineering?

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

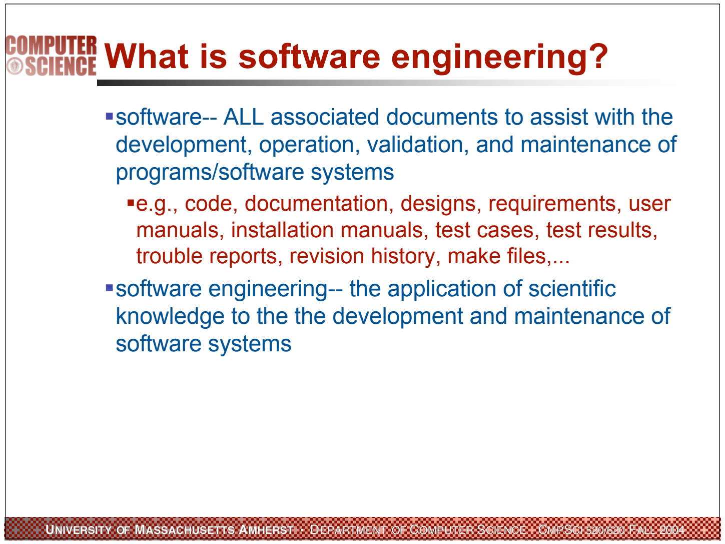
UNIVERSITY OF MASSACHUSETTS AMHERST • DEPARTMENT OF COMPUTER SCIENCE • CMPSCI 520/620 FALL 2004



What is software engineering?

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

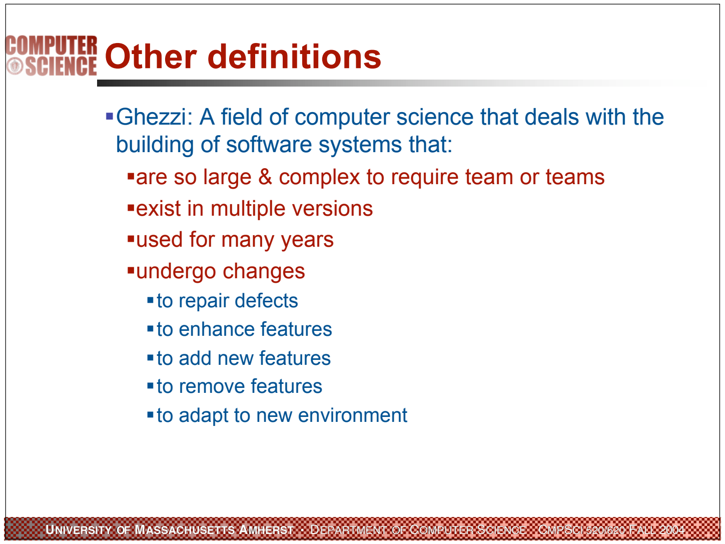
UNIVERSITY OF MASSACHUSETTS AMHERST • DEPARTMENT OF COMPUTER SCIENCE • CMPSCI 520/620 FALL 2004



What is software engineering?

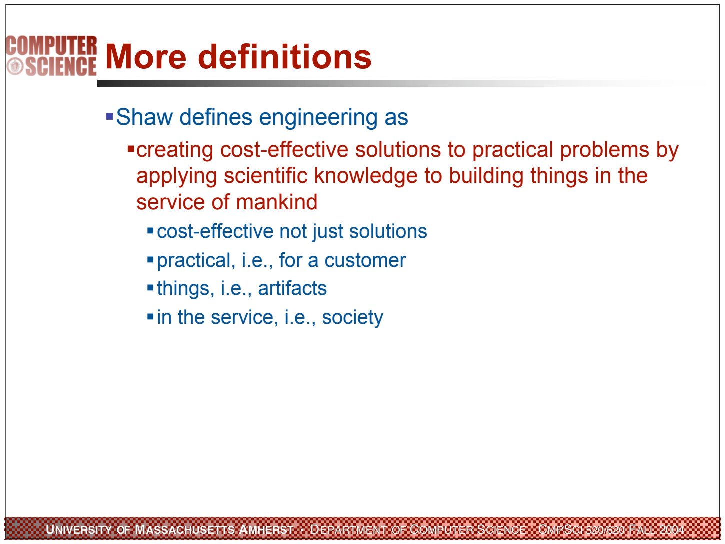
- software-- ALL associated documents to assist with the development, operation, validation, and maintenance of programs/software systems
  - e.g., code, documentation, designs, requirements, user manuals, installation manuals, test cases, test results, trouble reports, revision history, make files,...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

UNIVERSITY OF MASSACHUSETTS AMHERST • DEPARTMENT OF COMPUTER SCIENCE • CMPSCI 520/620 FALL 2004



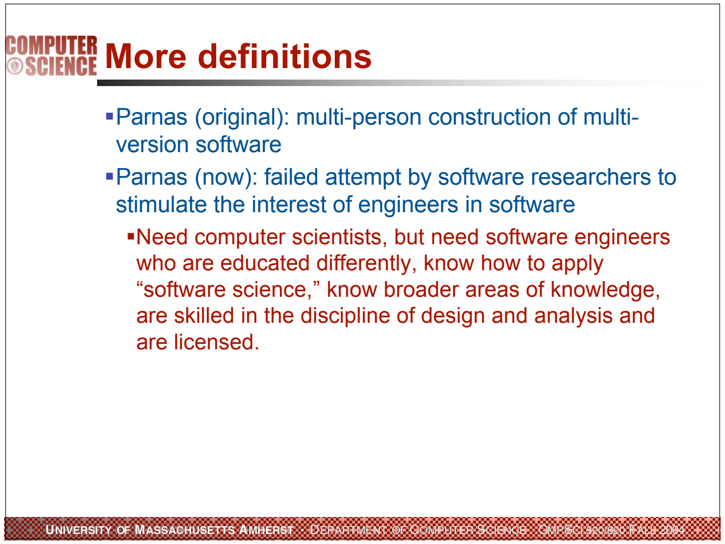
Other definitions

- 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



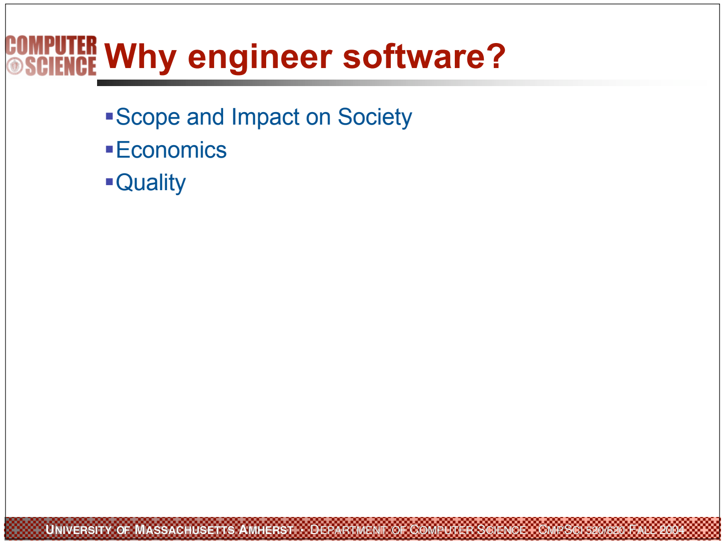
More definitions

- 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



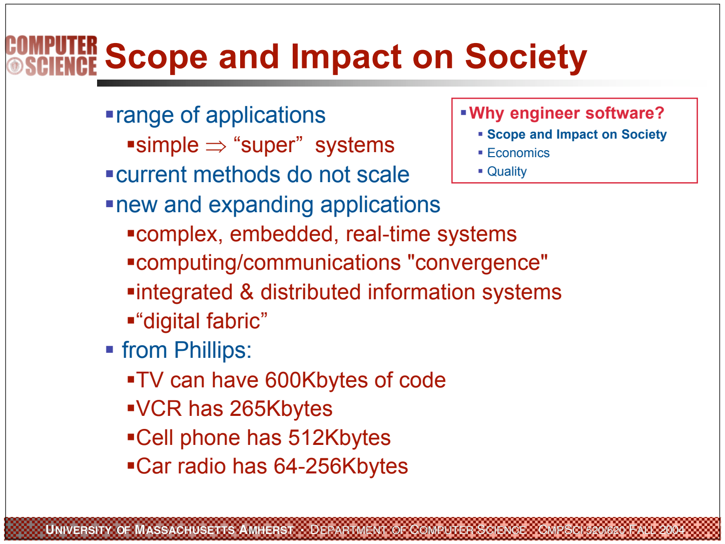
More definitions

- 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

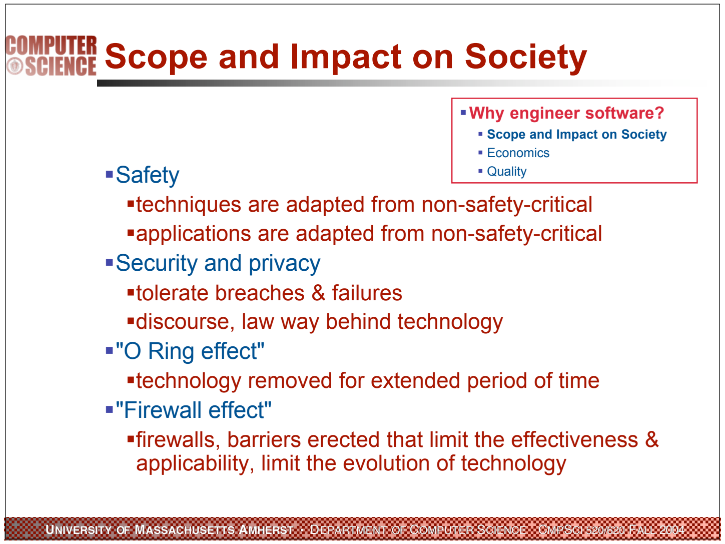


Scope and Impact on Society

- 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

- Why engineer software?
  - Scope and Impact on Society
  - Economics
  - Quality

University of Massachusetts Amherst · Department of Computer Science · CMPSCI 520/620 Fall 2004

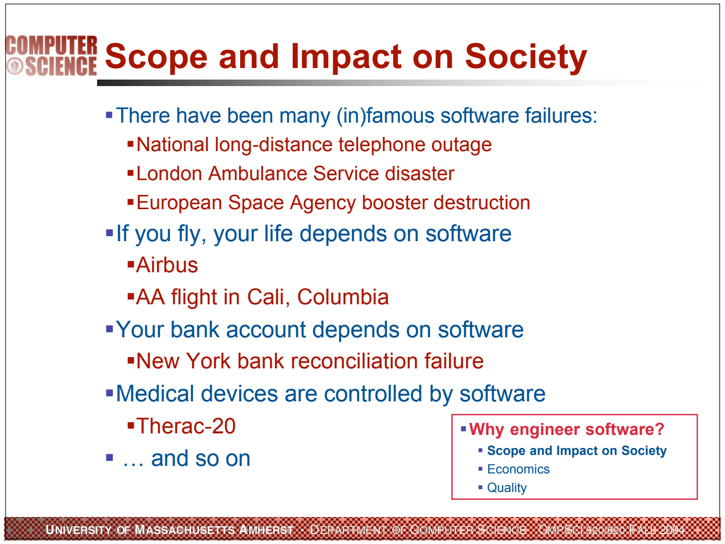


Scope and Impact on Society

- Why engineer software?
  - Scope and Impact on Society
  - Economics
  - Quality

- 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 2004

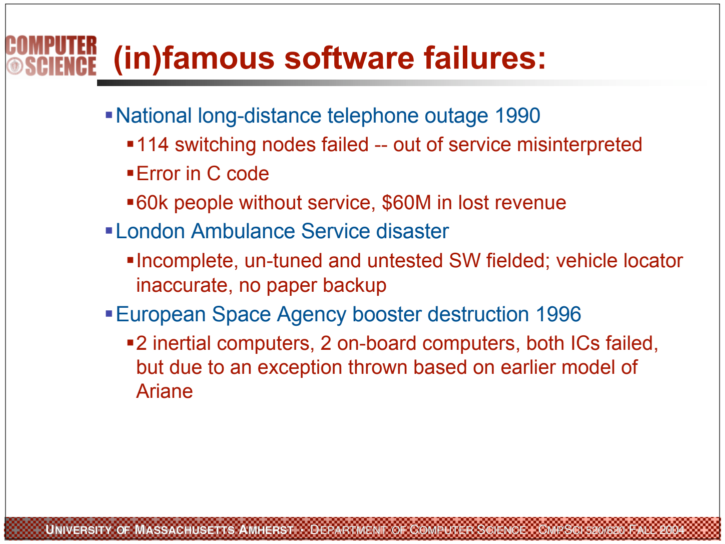


Scope and Impact on Society

- 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

- Why engineer software?
  - Scope and Impact on Society
  - Economics
  - Quality

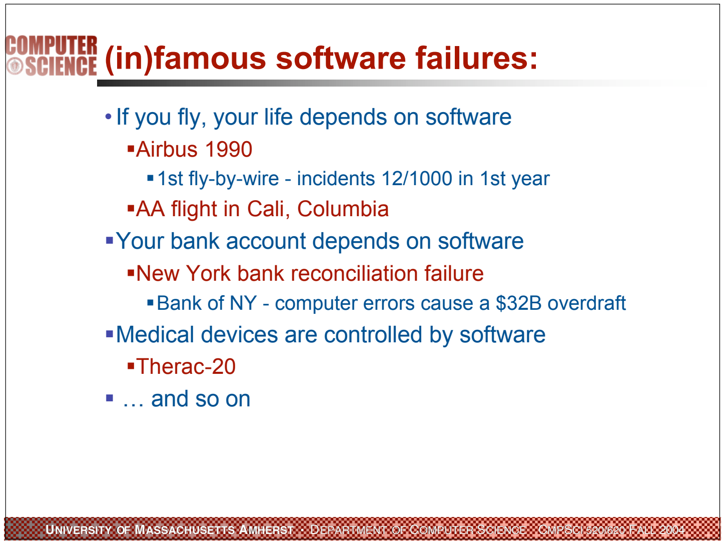
University of Massachusetts Amherst · Department of Computer Science · CMPSCI 520/620 Fall 2004



(in)famous software failures:

- National long-distance telephone outage 1990
  - 114 switching nodes failed -- out of service misinterpreted
  - Error in C code
  - 60k people without service, $60M in lost revenue
- London Ambulance Service disaster
  - Incomplete, un-tuned and untested SW fielded; vehicle locator inaccurate, no paper backup
- European Space Agency booster destruction 1996
  - 2 inertial computers, 2 on-board computers, both ICs failed, but due to an exception thrown based on earlier model of Ariane

University of Massachusetts Amherst · Department of Computer Science · CMPSCI 520/620 Fall 2004



COMPUTER SCIENCE

(in)famous software failures:

- If you fly, your life depends on software
  - Airbus 1990
    - 1st fly-by-wire - incidents 12/1000 in 1st year
  - AA flight in Cali, Columbia
- Your bank account depends on software
  - New York bank reconciliation failure
    - Bank of NY - computer errors cause a $32B overdraft
- Medical devices are controlled by software
  - Therac-20
- … and so on

UNIVERSITY OF MASSACHUSETTS AMHERST · DEPARTMENT OF COMPUTER SCIENCE CMPSCI 520/620 FALL 2004