COSC 340: Software Engineering

Introduction

Audris Mockus

(adapted from slides by Ravi Sethi, University of Arizona)

What is Software Engineering?

- Joint NATO workshop met in 1968 to discuss the *software crisis*
  - Individual approaches to program development do not scale up
  - The crisis led to a number of issues, including: projects running over-time / budget, low-quality software, code that was hard to maintain, etc.
- Quote from the report:
  - "The phrase 'software engineering' was deliberately chosen as being provocative, in implying the need for … the types of theoretical foundations and practical disciplines that are traditional in established branches of engineering."

What do software engineers do?

- Analyze users' needs and then design, test, and develop software to meet those needs
- Recommend software upgrades for customers' existing programs and systems
- Design each piece of an application or a system and plan how the pieces will work together
- Create a variety of models and diagrams (such as flowcharts) that instruct programmers how to write software code
- Ensure that a program continues to function normally through software maintenance and testing
- Document every aspect of an application or a system as a reference for future maintenance and upgrades
- Collaborate with other computer specialists to create optimum software

Growing need for software engineers

- Employment expected to grow 17% (2014 – 2024)
  - Much faster than the average for all occupations
- Median annual wage for software systems developers was $100,690 in 2015
  - Top 10% earned more than $153,710
  - Median for application developers was $98,260
- Qualities and Skills
  - Analytical, creativity, problem solving
  - Communication, customer-service, inter-personal
  - Big picture, attention to detail

Definitions of Software Engineering*

- “The establishment and use of sound engineering principles (methods) in order to obtain economically software that is reliable and works on real machines" [Bauer 1972]
- "Software engineering is that form of engineering that applies the principles of computer science and mathematics to achieving cost-effective solutions to software problems." [SEI 1990]
- "The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software" [IEEE 1990]

* IEEE and ACM Curriculum Guidelines [2004]

Definition of Software Engineering

- **Software Engineering** is
  - The art and science of
  - developing reliable software systems that
  - address customer needs,
  - subject to cost and schedule constraints

CUSTOMER-RELATED

TEAM-RELATED

Art & Science

Processes,
Principles, Practices

PRODUCT-RELATED

ORGANIZATION-RELATED

CUSTOMER-RELATED

Requirements
Stakeholder Needs
Usage Scenarios

TEAM-RELATED

Development
Activities
Skills, Culture

Art & Science
Processes,
Principles, Practices

Technology
Artifacts
Tools

PRODUCT-RELATED

Constraints
Cost, Schedule
Business, Legal, Regulatory

ORGANIZATION-RELATED

Software Engineering: Alternative Definition

- "Multi-person development of multi-version programs"

Multi Person Coordinate teams Design for modularity	**Multi Person** **Multi Version** X
Single Person	**Multi Version** Develop program families Evolve & maintain releases

Security Vulnerability Due to a Software Bug

“Software products are among the most complex of ... systems, and software by its very nature has intrinsic, essential properties (e.g., complexity, invisibility, and changeability) that are not easily addressed”

Apple SSL Security Vulnerability

- Why does it matter?
  - TLS and its predecessor SSL were designed to prevent eavesdropping and tampering of communications across a network
  - Examples: potentially affects secure browsing, credit card info, ...
  - Hundred of millions of iOS (iPhone, iPad) and OS X (Mac) devices
- Schedule of systems and security updates
  - iOS 6.x – 6.1.5 since 9/19/12 fixed in 6.1.6 on 2/21/14
  - OS X 10.9 – 10.9.1 10/22/13 10.9.2 on 2/25/14
- Acronyms
  - TLS: Transport Layer Security
  - SSL: Secure Sockets Layer

Apple SSL Security Vulnerability

- Extra goto in the code for the handshake algorithm
  - This sequence of conditionals was duplicated six times!
  - As it appears in the SSLVerifySignedServerKeyExchange() function

```
if ((err = ReadyHash(&SSLHashSHA1, &hashCtx)) != 0)
    goto fail;
if ((err = SSLHashSHA1.update(&hashCtx, &clientRandom)) != 0)
    goto fail;
if ((err = SSLHashSHA1.update(&hashCtx, &serverRandom)) != 0)
    goto fail;
if ((err = SSLHashSHA1.update(&hashCtx, &signedParams)) != 0)
    goto fail;
    goto fail;
if ((err = SSLHashSHA1.final(&hashCtx, &hashOut)) != 0)
    goto fail;
```

Ethics

“Because of their role in developing software systems, software engineers have significant opportunities to do good or cause harm.”

Source: ACM Software Engineering Code of Ethics, https://www.acm.org/about/se-code

6 known accidents 1985-1987

– Series of “fixes”

- E. Texas Cancer Center, March 21, 1986

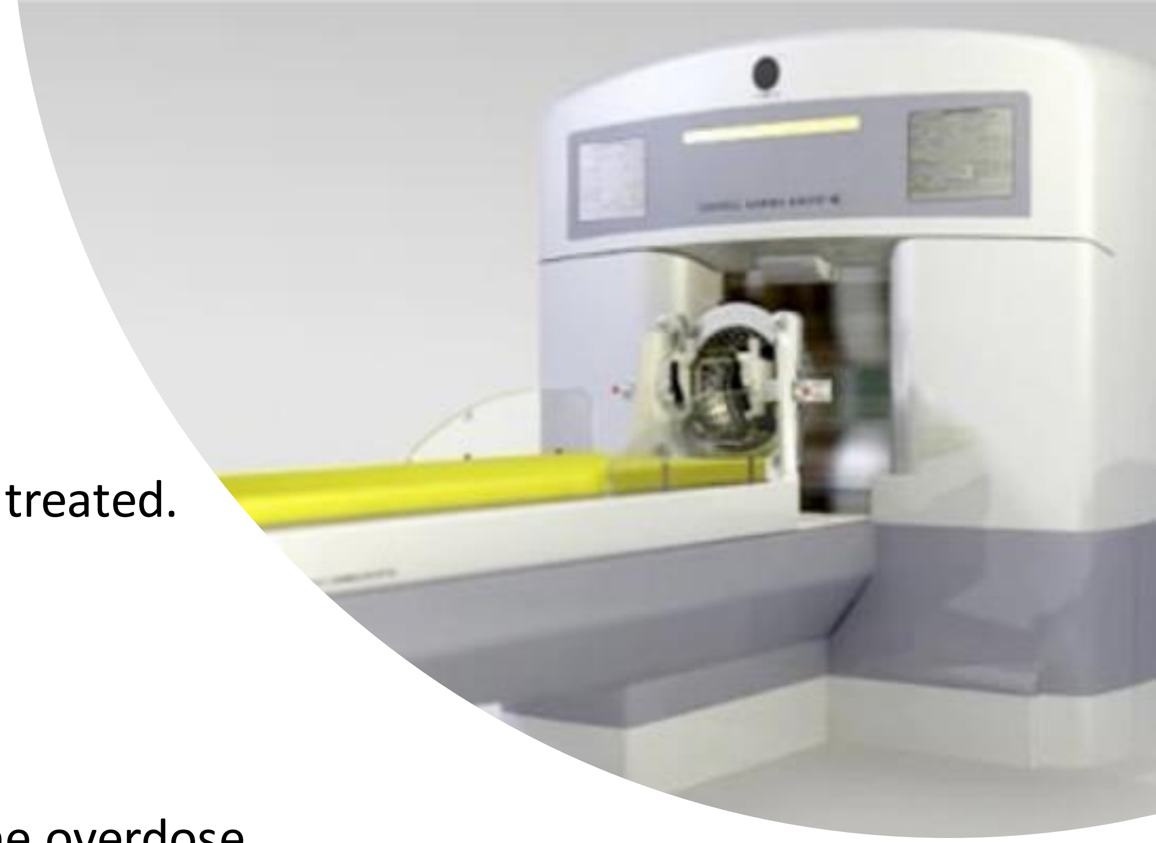
– Therac-25 had been in use for 2 years; over 500 patients treated.

– Male patient, in for his ninth treatment

– Planned dose: 180 rads

– Possible dose: 16,500-25,000 rads in less than 1 sec.

– 5 months later, the patient died from complications of the overdose



Therac-25: Fatal Radiation Overdose

From: Leveson and Turner [1993], Leveson [1995]

Lessons

- Confusing Reliability with Safety
  - Had worked tens of thousands of times – “highly reliable”, but not safe
- Lack of Defensive Design
  - No self checks or other error detection and error handling
- Failure to Eliminate Root Causes
  - A fix after each accident, rather than a thorough investigation
- Complacency
  - Often, it takes an accident to get attention, get funding, get action
- Unrealistic Risk Assessments
  - The first hazard analyses initially ignored software, then assumed that all software errors are equally likely

From: Leveson [1995]

Lessons

- Inadequate Software Engineering Practices
  - “Specifications and documentation should not be an afterthought”
  - Establish rigorous software quality assurance practices and standards
  - Keep designs simple; avoid dangerous coding practices
  - Design audit trails and error detection into the system from the start
  - Conduct extensive tests at the module and software level
  - System tests are not enough
  - Perform regression tests on all software changes
  - Carefully design user interfaces, error messages, and documentation
- Software Reuse [of “proven” subsystems?]
  - Safety is a property of the system, not of the software itself

From: Leveson [1995]

Ethics

- Contribute to society and human well-being
- Avoid harm to others
- Be honest and trustworthy
- Be fair and take action not to discriminate
- Honor property rights including copyrights and patent
- Give proper credit for intellectual property
- Respect the privacy of others
- Honor confidentiality
- Source: ACM Software Engineering Code of Ethics, https://www.acm.org/about/se-code