PURDUE UNIVERSITY®

CS 307: Software Engineering

Lecture 2: Software Engineering, Ethics, and Software Quality

Prof. Jeff Turkstra

Reminders

- Team assignments due Friday, January 13
  - Email jeff@purdue.edu
    - Even if it's "please assign me to a team"
- Consultation meetings

Today

- Software Engineering
- Ethics
- Stakeholders and Software Quality

Software Engineering

- IEEE: (1) the application of a systematic, disciplined, quantifiable approach to the development, operation, maintenance of software; that is, the application of engineering to software. (2) the study of approaches as in (1)

Solving Problems

- The goal of software engineering
- Within constraints
  - Cost
  - Time
  - Customer
  - Others
- And with rigor

Solutions

- Require effective communication
  - "I know you believe you understood what you think I said, but I am not sure you realize that what you heard is not what I meant…"
- Sometimes one can
  - Assemble from existing components
    - Not always build from scratch
    - Avoid NIH - "Not Invented Here"
  - Buy, not assemble or build

Systematic development and evolution

- An engineering process involves applying well-understood techniques in an organized and disciplined way
- Many practices have been formally standardized
- Most development is evolutionary

Why?

- Software engineering techniques are needed because systems have constraints
  - Time constraints
  - Knowledge/capacity constraints
  - Finite resources
  - etc
- Teamwork and coordination are required
  - Key challenge: dividing up the work and ensuring that all parts work together properly
- Necessity for "high quality"

Software *Engineering*

- The term Software Engineering was coined in 1968
  - People began to suggest that the principles of engineering should be applied to software development
- Engineering is a licensed profession
  - In order to protect the public
  - Engineers design artifacts following well-accepted practices which involve the application of science, mathematics, economics, …
  - Ethical practice is also a key tenet of the profession

In many countries, software engineering does not (yet) require an engineering exam or license

Differences with other Engineering Fields

- Foundations are primarily in computer science
  - As opposed to the natural sciences
- Focus on discrete rather than continuous mathematics
- Concentrates on abstract/logical entities instead of concrete/physical artifacts
- No “manufacturing” phase
- “Maintenance” primarily refers to continued development, or evolution
  - Not conventional wear and tear

Commonalities

- Engineers proceed by making a series of decisions, carefully evaluating options, and choosing an approach at each decision-point that is appropriate for the current task in the current context
  - Appropriateness can be judged by tradeoff analysis, which balances costs against benefits

- Engineers measure things, and when appropriate, work quantitatively; they calibrate and validate their measurements; and they use approximations based on experience and empirical data

- Engineers emphasize the use of a disciplined process when creating a design and can operate effectively as part of a team in doing so

- Engineers can have multiple roles
  - Research, development, design, production, testing, construction, operations, management, and others such as sales, consulting, and teaching

- Engineers use tools to apply processes systematically. Therefore, the choice and use of appropriate tools is key to engineering

- Engineers, via their professional societies, advance by the development and validation of principles, standards, and best practices

Ethics

- A theory or system of moral values
- The principles of conduct governing an individual or group

Engineering ethics

- Well-developed area of professional ethics
  - Like medical and legal ethics
- Engineers today are expected to both learn about and live up to ethical standards as a condition of their membership in the profession

Ethics in Software Engineering

- Software Engineering Code of Ethics and Professional Practice (Version 5.2) as recommended by the ACM/IEEE-CS Joint Task Force
- http://www.acm.org/about/se-code

Short version

"Software engineers shall commit themselves to making the analysis, specification, design, development, testing and maintenance of software a beneficial and respected profession. In accordance with their commitment to health, safety and welfare of the public, software engineers adhere to the following Eight Principles:"

Public

Software engineers shall act consistently with the public interest.

Client and Employer

Software engineers shall act in a manner that is in the best interests of their client and employer consistent with the public interest

Product

Software engineers shall ensure that their products and related modifications meet the highest professional standards possible.

Judgment

Software engineers shall maintain integrity and independence in their professional judgment.

Management

Software engineering managers and leaders shall subscribe to and promote an ethical approach to the management of software development and maintenance.

Profession

Software engineers shall advance the integrity and reputation of the profession consistent with the public interest.

Colleagues

Software engineers shall be fair and supportive of their colleagues.

Self

Software engineers shall participate in lifelong learning regarding the practice of their profession and shall promote an ethical approach to the practice of the profession.

Responsibility

- Software engineers have significant opportunities to do good or cause harm
  - Also to enable or influence others to do good or cause harm
- Software engineers must commit themselves to making software engineering a beneficial and respected profession

National Society of Professional Engineers

"to hold paramount the safety, health, and welfare of the public."

Stakeholders

- Users
  - Those who use the software
- Customers (Project Owners)
  - Those who pay for the software
- Software Developers (Software Engineers)
  - Those who create and maintain the software
- Development Managers (Project Coordinators)
  - Those who supervise the software development process

Quality

- Conformance to requirements
  - Crosby
- Fitness for use
  - Japan
- I know it when I see it
  - Guaspari
- Value to someone
  - Weinberg

Japan

- Business is done by reputation, not always by contract
- Something goes wrong after release?
  - Lose key currency of business: credibility and reputation

Software quality attributes

- For product operation
- For product revision
- For product transition

Operation attributes

- Correctness
- Efficiency
- Integrity
- Reliability
- Usability

Correctness

The degree to which a program satisfies the user's requirements.

- Does it do what the user wanted it to do?
- If not, why not?
- Does it do more than what was desired?
  - Who is paying for the extra features?

Efficiency

The amount of computing resources, time and space, required to perform a user-defined function or task

- Does it run as well as it can, subject to agreed upon resource constraints, on their computer system?
- If not, why not?

Integrity

How well is the software and data protected from security breeches and installation errors?

- Is access controlled?
- Can users tell if the system has been properly configured?
- Can they tell when it has not been configured properly?

Reliability

To what degree can the system be expected to perform its intended function without failure in the user's environment?

- What is the Mean Time Between Failures?
- What is the Mean Time to repair at user's site?

Usability

How much effort do the users have to spend learning to use the system efficiently?

- Can you use it?
- Can I use it?
- Can anyone use it?
- Would anyone want to learn how to use it?
- Is it worth the effort?

Revision attributes

- Flexibility
- Maintainability
- Testability

Flexibility

How much effort will be required to enhance the system?

- Can it be changed?
- Who can change it?

Maintainability

- How much effort will be required to locate and repair defects in the system?
  - Can you find the defects?
  - Can you repair them?
  - What will it cost to fix them?
  - Who is going to pay for the repairs?

Testability

How much effort will be required to test the structure and functionality of the system?

- Can you test the system?

Transition attributes

- Interoperability
- Portability
- Reusability

Interoperability

How much effort will be required to link this program or system to another?

- Can you interface it to another system?
- Can the user interface it to other programs?

Portability

How much effort will it take to transfer a program or system from one machine to another?

- Will it run on all your user's machines?
- Who will pay for the porting of the code?

Reusability

To what extent can the design, or system modules be used in other applications. How much effort will it take to reuse them?

- Can you find and extract the parts?

Internal quality

- Characterizes aspects of the design of the software
- Impacts external quality attributes
- For example
  - Amount of commenting of the code
  - Complexity of the code
  - Use of well-understood software patterns

Quality is everyone's business

- The customer and end user expect to receive and pay for a quality system.
- The developers have a right to expect to be able to build a quality system.
- Maintainers have a right to expect a quality system. One which can be adapted to the changing needs of the customer and end users.

Quality is free

- “What costs money are the unquality things – all the actions that involve not doing jobs right the first time.”
- “An erroneous assumption is that quality is an intangible and therefore not measurable. In fact, quality is precisely measurable by the oldest and most respected of measurements: cold hard cash”

Is quality free?

"Well, maybe. However, it takes a lot of time and effort to achieve." Crosby

Quality and Stakeholders

Customer:
solves problems at
an acceptable cost in
terms of money paid and
resources used

User:
easy to learn;
efficient to use;
helps get work done

Developer:
easy to design;
easy to maintain;
easy to reuse its parts

sells more and
pleases customers
while costing less

Tension

- Software qualities can conflict
  - Increasing efficiency may impact interoperability
    - Or portability, or maintainability, or …
  - Increasing usability may impact efficiency
  - Increasing integrity may impact efficiency
  - etc

Objectives

- Setting objectives for quality is a key engineering activity
  - You then design to meet the objectives
- Optimizing is always necessary
  - E.g., obtain the highest possible reliability using a fixed budget

Questions?