Lecture 20: What is Software Engineering?

Kenneth M. Anderson
Software Methods and Tools
CSCI 3308 - Fall Semester, 2002

Today's Lecture

- Discuss history of Software Engineering
- Discuss Several Definitions of Software Engineering
- Discuss qualities that software engineering strives to achieve in software

Historical Background: 30 years

- First Software Engineering Conference
  - NATO-sponsored conference in 1968
- "Software Crisis"
  - Systems were designed by identifying the hardware first
    - Software was allocated about 1-2% of the budget
  - However, software was causing all the problems (!) and thus needed more attention

Progression of SE

- An evolution of the programming activity
  - Early stages of computing
    - User/Developer were the same person
    - Problems were well-understood
      - First programs calculated metrics about artillery shells for the Navy!
  - High level languages began to appear in the 1950s
    - Along with the profession of "programmer"

SE Progression, continued

- 1960's
  - Large Software Systems for Commercial Ventures
    - Teams of Programmers
    - Separate end-users
    - Complex Problems
  - "Software Crisis" coined, as problems became apparent

The problem?

- Software is typically
  - late
  - over budget
  - faulty
  - costly to maintain
  - difficult to evolve
  - etc.

Consider the following:

- Loss of NASA's Mars Climate Observer
  - due to conversion error of English and Metric units!
  - even worse: problem was known but politics between JPL and Houston prevented fix from being deployed
- Leap-year bug
  - A supermarket was fined $1000 for having meat around 1 day too long on Feb. 29, 1988
- Denver International Airport
  - Luggage system: 16 months late, 3.2 billion dollars over budget!

SE Progression, continued

- 1968
  - Software Engineering formed
  - Many "solutions" put forward
    - New approaches to Project Management
    - New Team Organizations
    - Better Languages and Tools
    - Organizational Standards
- And here we are 35 years later! :-)

Multiple Definitions of SE

- There are many ways to define software engineering
  - We shall look at a few to try to gain a feel for an overall definition
  - These definitions come from textbooks, prominent software engineers, etc.

Software Engineering

- Software
  - Computer programs and their related artifacts
    - e.g. requirements documents, design documents, test cases, specifications, protocol documents, UI guidelines, usability tests, ...
- Engineering
  - The application of scientific principles in the context of practical constraints

What is Engineering?

- Engineering is
  - a sequence of well-defined, precisely-stated, sound steps, which follow a method or apply a technique based on some combination of
    - theoretical results derived from a formal model
    - empirical adjustments for unmodeled phenomenon
    - rules of thumb based on experience
- This definition is independent of purpose...
  - i.e. engineering can be applied to many disciplines

Software Engineering
(Daniel M. Berry)

- Software engineering is that form of engineering that applies:
  - a systematic, disciplined, quantifiable approach,
  - the principles of computer science, design, engineering, management, mathematics, psychology, sociology, and other disciplines,
- to creating, developing, operating, and maintaining cost-effective, reliably correct, high-quality solutions to software problems.

Software Qualities

- Correctness
- Reliability
- Robustness
- Performance
- User Friendliness
- Verifiability
- Maintainability
- Reusability
- Portability
- Understandability
- Interoperability
- Productivity
- Timeliness
- Visibility

Software Engineering Principles

- Rigor and Formality
- Separation of Concerns
- Modularity
- Abstraction
- Anticipation of Change
- Generality
- Incrementality

SE Research Topics (just a subset)

- Software Architecture
  - Design Patterns for Large Systems
- Web Services
  - Semantics of Component Frameworks
- Life Cycles
  - Understanding the pros/cons of XP
- Requirements Traceability
  - techniques for managing artifact relationships

SE “Hot Topics”

- Open Source and Agile Design Methods
- Refactoring and Design Patterns
  - especially “refactoring browsers/editors”
- Automated Testing and Test Driven Development
  - See for instance JUnit/HttpUnit
- Software Architecture
  - In particular “architecture patterns”