DA-IICT

IT 314: Software Engineering

Introduction to Software Engineering

Saurabh Tiwari

1

---

what's a software anyway?" ...

A software is a computer programs along with the associated documents and the configuration data that make these programs operate correctly.

A program is a set of instructions (written in form of human-readable code) that performs a specific task.

Software Engineering

- Software Engineering is am engineering discipline that's applied to the development of software in a systematic approach (called a software process)

- It is the application of theories, methods, and tool to design build a software that meets the specifications efficiently, cost-effectively, and ensuring quality.

- It's not only concerned with the technical process of building a software, it also includes activities to manage the project, develop tools, methods and theories that support the software production.

What is Software Engineering? Definition

- "Software Engineering is the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software."   *[IEEE'90]*

- "Software Engineering is intended to mean the best-practice processes used to create and/or maintain software, whether for groups or individuals."   *[SEYP]*

- "Software Engineering is an act of applying a collection of techniques, methodologies and tools that help with the production of a high quality software system, … with a given budget, before a given deadline, while change occurs."
  `   *[Textbook]*

Software Engineering vs. Computer Science
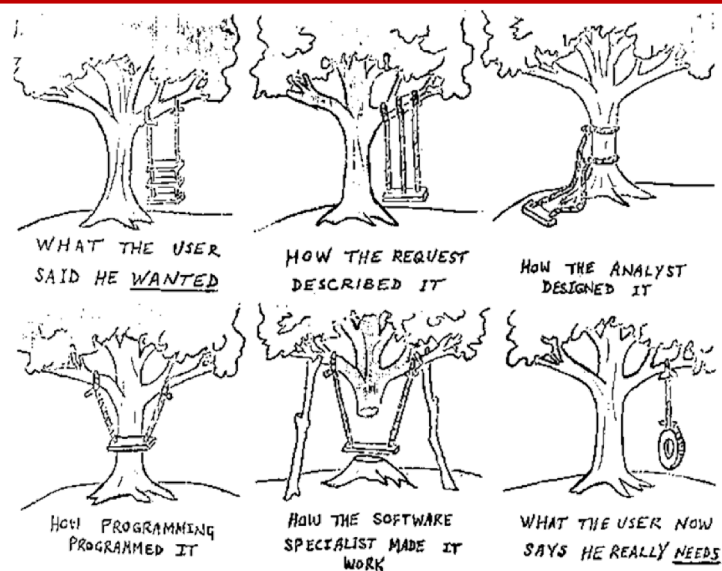
- Computer science is concerned with theory and fundamentals (algorithms, programming languages, theories of computing, AI and hardware design)

- Software engineering is concerned with the practicalities of developing and delivering useful software (activities).

Software Engineering vs. Computer Science
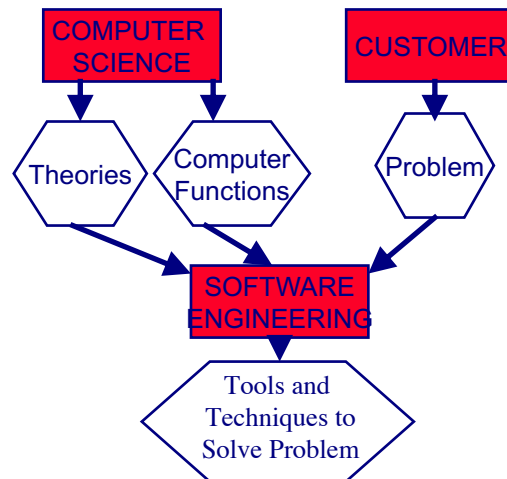
Communication is a critical element

Software Engineering vs. System Engineering

- System engineering is concerned with all aspects of computer-based systems development including hardware, software and process engineering.

- Software engineering is part of this process concerned with developing the software infrastructure, control, applications and databases in the system.

- System engineers are involved in system specification, architectural design, integration and deployment.
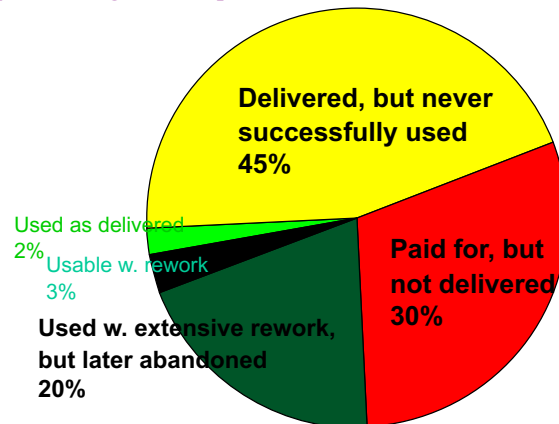
Problems Behind the "Software Crisis"

- Increased size and complexity of systems

- Cost overruns

- Design bugs after implementation

- Maintenance ripple effect

- Requirements and design needed development tools, not just in the programming tools
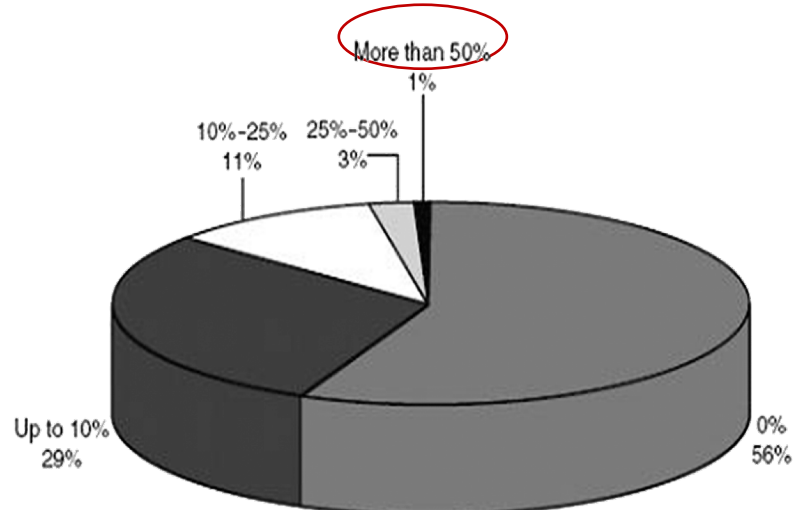
Why Software Engineering?

- Research from Standish Group Data on 9236 development projects completed in 2004.

  9 software projects totaling $96.7 million: Where The Money Went
  [Report to Congress, Comptroller General]

Delivered, but never successfully used 45%

Used as delivered 2%

Usable w. rework 3%

Used w. extensive rework, but later abandoned 20%

Paid for, but not delivered 30%

Abandoned or Cancelled Projects

More than 50%
1%

10%-25%
11%

25%-50%
3%

Up to 10%
29%

0%
56%

http://articles.directorym.net/An_Introduction_to_Catastrophe_Disentanglement_Lynchburg_VA-r923574-Lynchburg_VA.html

Weapons Against Software Crisis

- Improving software engineering methodologies

- High-level languages and tools that encourage and enforce these principles

Software Engineering Definition

The software crisis yielded yet another definition of software engineering:

•*Discipline whose aim is the production of fault-free software, delivered on time and within budget, that satisfies the client's needs.*

Costs of Software Engineering

- Software costs dominate computer systems costs.

  – Roughly 60% of costs are development costs, 40% are testing costs. For custom software, evolution costs often exceed development costs.

- Software maintenance costs are more than software development costs.

  – For systems with a long life the maintenance may be several times the development costs. And often even bad software has a long life.

What is Software Engineering?

- A problem solving activity

- A modeling activity

- A knowledge-acquisition activity

- A rationale-driven activity

Elaboration

SE: A Problem Solving Activity

- Analysis: Understand the nature of the problem and break the problem into pieces
- Synthesis: Solve them and put the solutions together to solve the original problem

- For problem solving we use
  - Techniques (methods):
    - Formal procedures for producing results using some well-defined notation
  - Methodologies:
    - Collection of techniques applied across software development and unified by a philosophical approach
  - Tools:
    - Instrument or automated systems to accomplish a technique

Difference from Other Engineering Disciplines

- The end product is abstract, not a concrete object like a bridge

- Costs are almost all human

- Easy to fix bugs, but hard to test and validate

- Software typically never wears out...but the hardware/OS platforms it runs on do

- Variations in application domain are open-ended, may require extensive new non-software, non-engineering knowledge for each project

- Changes occur in application and solution domains while the problem is being solved (Software Evolution)

Difference from Other Engineering Disciplines

- Software Engineering is a collection of techniques, methodologies and tools that help with the production of

  - a high quality software system

  - with a given budget

  - before a given deadline

- while change occurs.

Scientist vs Engineer

- **Computer Scientist**
  - ◆ **Proves theorems about algorithms, designs languages, defines knowledge representation schemes**
  - ◆ **Has infinite time…**
- Engineer
  - ◆ **Develops a solution for an application-specific problem for a client**
  - ◆ **Uses computers & languages, tools, techniques and methods**
- **Software Engineer**
  - ◆ **Works in multiple application domains**
  - ◆ **Has only 3 months...**
  - ◆ **…while changes occurs in requirements and available technology**

Software Engineering Concepts

An Example – TicketDistributor

TicketDistributor is a machine that issues tickets for trains. Travelers have the option of selecting a ticket for a single trip or for multiple trips, or selecting a time card for a day or a week. The machine computes the price of the requested ticket based on the area in which the trip will take place and whether the traveler is a child or an adult. The TicketDistributor must be able to handle several exceptions, such as traveler who do not complete the transaction, travelers who attempt to pay large bills, and resource outages such as running out of tickets, change, or power.

Software Engineering Concepts

ROLES	RESPONSIBILTIES	EXAMPLE
Client	Requirements, Scope, Constraints like budget, schedule	Train Company
User	Domain Knowledge, use of the system	Traveler
Project Manager	Managing the project, estimation, planning, scheduling and tracking, hiring people, reporting	Alice (Boss)
Human Factor Specialist	Ensure Usability	Zoe (HCI Specialist)
Developer	Various roles in software development	John (Analyst), Marc (Programmer), Zoe (Tester)
Technical Writer	Documentation delivered to the client that include SRS, User manual	John

Example of roles in software development for TicketDistributor project

Software Engineering Concepts

WORK PRODUCT	TYPE	DESCRIPTION
Specification (SRS)	Deliverable	Contractual document between client and development organization
Operation Manual	Deliverable	Installation and configuration instructions
Status Report	Internal Work Product	Reports about tasks that are completed or in progress for the people in the hierarchy
Test Report	Internal Work Product	Test plans, test procedure, test cases and results
Release Report	Deliverable	Delivered functionality, resolved and unresolved issues

Example of work products in software development for TicketDistributor project

Software Development - Engineering Activities

A phased development (Process-oriented)

Requirements Specification

- Functional and Non-Functional Req. (Testable)
- Additional Constraints
- Interfaces to other systems including UI

Analysis Modeling

- Use cases
- Interaction Models
- Object Model (Problem Domain)
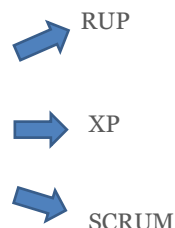
Design

- System Design
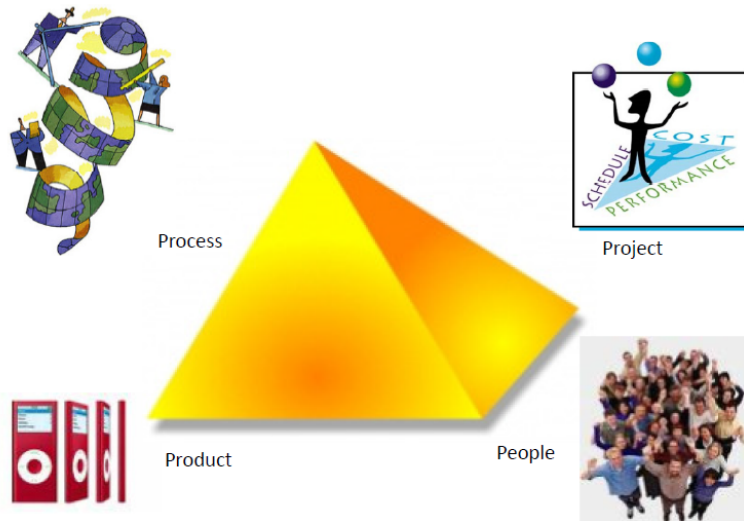- Object Design

Construction (Coding)

Testing

- At Different levels – Unit, Integration, System, User
- Regression Testing
- Specialized Testing like performance, security, load, Stress testing

Operation and Maintenance

- Deployment
- Change Management

RUP

XP

SCRUM

Software Engineering Dimensions

S/W Characteristics (Boehm's)

1. Finding and fixing a software problem after delivery of the product is 100 times more expensive than defect removal during requirements and early design phases.

2. Nominal software development schedules can be compressed up to 25% (by adding people, money, etc.) but no more.

3. Maintenance costs twice what development costs.

4. Development and maintenance costs are primarily a function of size, e.g. the number of source lines of code, in the product.

5. Variations in humans account for the greatest variations in productivity.

S/W Characteristics (Boehm's Top 10)

6. The ratio of software to hardware costs went from 15:85 in 1955 to 85:15 in 1985 and continues to grow in favor of software as the dominant cost.

7. Only about 15% of the development effort is in coding.

8. Walk-throughs catch 60% of the errors.

What are the attributes of good software?

- The software should deliver the required functionality and performance to the user and should be maintainable, dependable and acceptable.
- Maintainability
  - Software must evolve to meet changing needs;
- Dependability
  - Software must be trustworthy;
- Efficiency
  - Software should not make wasteful use of system resources;
- Acceptability
  - Software must accepted by the users for which it was designed. This means it must be understandable, usable and compatible with other systems.

What are the attributes of good software?

- From the Users Perspective
  - Correctness
  - Reliability
  - Efficiency
  - Maintainability
  - Usability
  - Robustness
- From the Developers Perspective
  - Consistency
  - Understandability
  - Testability
  - Compactness
  - Compatibility
  - Integrity

Questions??

Next Lectures…
SDLC, Software process, Software process models, CMM
Other Processes