Introduction to Software Engineering

Rajkumar Buyya
Grid Computing and Distributed Systems Lab
Dept. of Computer Science and Software Engineering
University of Melbourne, Australia
http://www.buyya.com

Software Engineering - Introduction

- Software Engineering is an *engineering discipline* which is concerned with *all aspects of software production* from the early stages of system requirements through to maintaining the system after is has gone into use.

Software Engineering/Computer Science

- *Computer Science* is concerned with the theories and methods which underlie computers and software systems.
- *Software Engineering* is concerned with the practical problem of producing software.

Notes

- *"Engineering discipline"* – Engineers make things work. They apply theories, methods and tools that are appropriate but use them selectively and always try to discover solutions to problems even when there are not applicable theories and methods to support them. Engineers have to work to organizational and financial constraints.
- "*All aspects of"* - Software engineering is not just concerned with the technical process of software development but also with activities such as software project management and the development of tools, methods and theories to support software production.

Software Crisis

- Have you ever received a bill for $0.00 ?
  - Did you respond by sending a cheque for $0
  - Have you any idea what this cheque did for our computer system ?
- Whether we dealing with billing software or word processing, software is being delivered:
  - Over Time
  - Over Budget
  - Low Quality
  - Full of bugs/residual faults
- Software engineering is an attempt to solve these problems.

Aspects of Software Engineering

- Historical Aspects
- Economical Aspects
- Maintenance Aspect
- Team Programming Aspects
- Design and Programming Aspects

Historical Aspects

- It is a fact that electric generators fail, but far less frequently than payroll products.
- It is true that bridges/cars/aero-planes sometimes collapse, but considerably less often than operating systems (e.g. MS Windows) do.
- In the belief that software could be engineered on the same footing as traditional engineering disciplines, a NATO study group coined the term "Software Engineering" in 1967.
- This was endorsed by the NATO Software Engineering Conference in 1968.

Scope of Software Engineering

- Why cannot other engineering techniques be used to build operating systems?
  - Attitude to collapse
  - Imperfect engineering
  - Complexity
  - Maintenance

Examples of Attitudes on Software

- People have the attitude that software collapse is not considered and unusual occurrence and therefore don't pay as much attention to design.
  - How many you rebooted your Microsoft Windows OS?
- Most of the time software engineers do not pay due attention to error scenarios, boundary conditions etc.
  - Divide by Zero conditions.
- Complexity of software is growing faster than the rate we can master it.
  - How many versions/bug fixes MS released within few years ?
- As a part of regular maintenance software engineers are expected to do major changes to software which is not the case in other fields of engineering.
  - Actually this has become survival strategy for companies like MS. Releasing new version every few months.

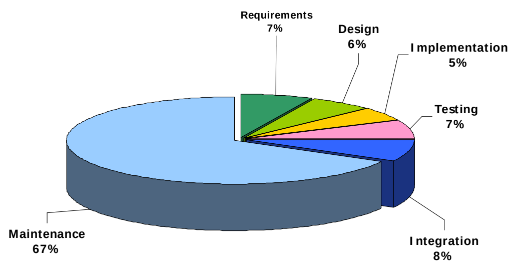
Economic Aspects

- Techniques should be economically viable
- A new coding method (CM_new) is 10% faster than the currently used method (CM_old). Should it be used?
  - Common sense answer
    - Of course
  - Software Engineering answer
    - What is the cost?

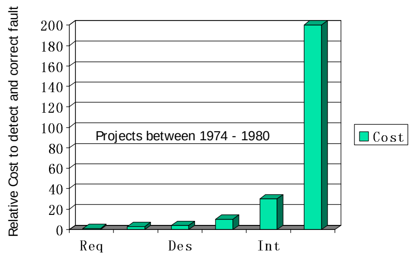
Maintenance Aspects

- Life Cycle – The series of steps software undergoes:
  - Requirements- Understand what the client wants
  - Specification – Understand what the product is supposed to do
  - Design – Identify the modules and the design
  - Implementation – Write code and unit test
  - Integration – Combine modules and test
  - Maintenance – Fixing problems and enhancements
  - Retirement – Product is no longer in use

Relative Cost/Effort



Cost to Detect and Fix Faults



13

Team Programming Aspect

- Large software products are developed by large software teams.
- Members have different responsibilities
  - e.g. – requirements, design, implementation, integration testing.
- Activities between teams have to be well organized for efficiency.
  - e.g. – meetings, interfaces

14

Software Design/Programming Aspects

Design Principles

15

Overview

- Introduce two commonly used design paradigms.
  - Structured Design paradigm
  - Object Oriented Design paradigm
- Understand general design principles.

16

Software Design - History

- Before 1975 most organizations did not use specific design techniques.
- 1975 – 1985 Structured Paradigm was introduced.
- Structured paradigm had certain short comings especially for large programs.
- Object Oriented paradigm was introduced and has become popular today.

17

Structured Paradigm

- Structured Designs are
  - Action (Function) Oriented

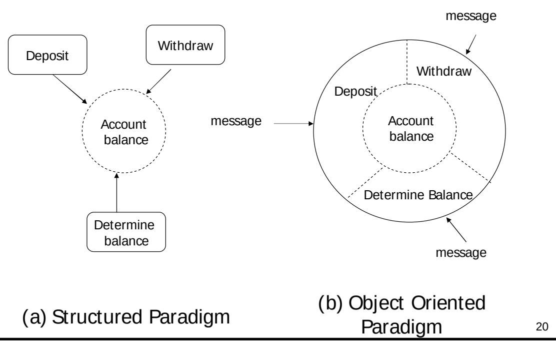
    OR
  - Data Oriented
  - But not both

18

Object Oriented Paradigm

- Both data and actions are of equal importance.
- Systems is a collection of interacting *Objects*.
- Object
  - Software component that incorporates *DATA* and the *ACTIONS* that are performed on the data.

Structured vs OO Example



Summary

- Software Engineering is an important discipline due to increased dependence of most of our modern life (e.g., banking, entertainment) on its products!
- By proper software engineering practices software can be built with:
  - Quality
  - Maintainability
  - On Time
  - On Budget

Reference

- Stephen Schach, *Classical and Object-Oriented Software Engineering with UML and Java,* Chapter 1, McGraw-Hill, New York, USA.
  - http://www.mhhe.com/engcs/compsci/schach5/samplech.mhtml
- Any other book on software engineering is also fine!