Object-Oriented Software Engineering

Practical Software Development using UML and Java

Chương 1:

Phần mềm và Công nghệ phần mềm

1.1 Công nghệ phần mềm là gì?

The process of solving customers' problems by the systematic development and evolution of large, high-quality software systems within cost, time and other constraints

Other definitions:

- 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).
- The Canadian Standards Association: The systematic activities involved in the design, implementation and testing of software to optimize its production and support.

What is Software Engineering?…

Solving customers' problems

- This is the *goal* of software engineering
- Sometimes the solution is to *buy, not build*
- Adding unnecessary features does not help solve the problem
- Software engineers must *communicate effectively* to identify and understand the problem

What is Software Engineering?…

Systematic development and evolution

- An engineering process involves applying *well understood techniques* in a organized and *disciplined* way
- Many well-accepted practices have been formally standardized
  - —e.g. by the IEEE or ISO
- Most development work is *evolution*

What is Software Engineering?…

Large, high quality software systems

- Software engineering techniques are needed because large systems *cannot be completely understood* by one person
- Teamwork and co-ordination are required
- Key challenge: Dividing up the work and ensuring that the parts of the system work properly together
- The end-product must be of sufficient quality

What is Software Engineering?

Cost, time and other constraints

- Finite resources
- The benefit must outweigh the cost
- Others are competing to do the job cheaper and faster
- Inaccurate estimates of cost and time have caused many project failures

1.2 Stakeholders in Software Engineering

1. Users

- Those who use the software

2. Customers

- Those who pay for the software

3. Software developers

4. Development Managers

All four roles can be fulfilled by the same person

1.3 Software Quality...

Usability

- Users can learn it and fast and get their job done easily

Efficiency

- It doesn’t waste resources such as CPU time and memory

Reliability

- It does what it is required to do without failing

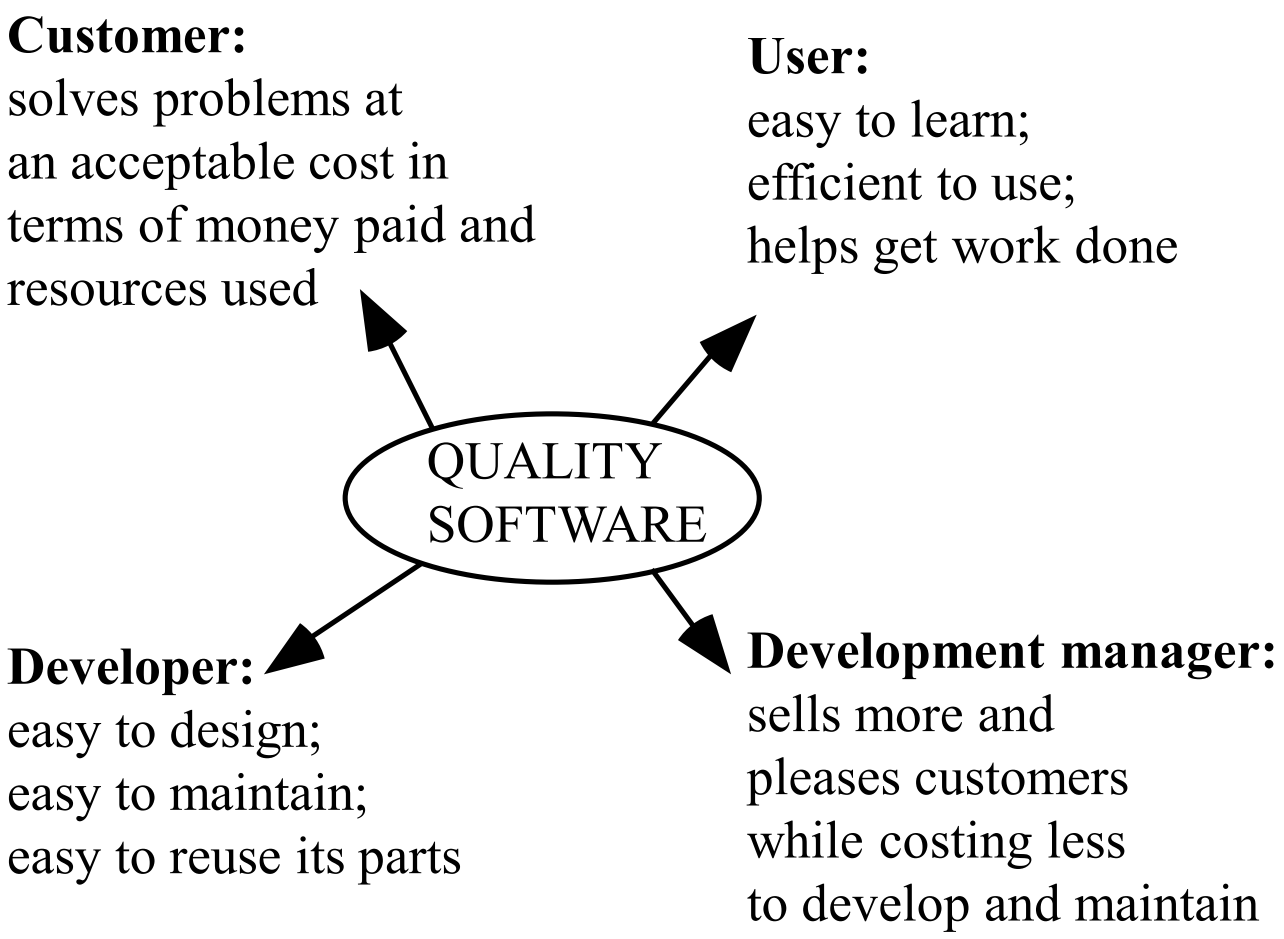
Maintainability

- It can be easily changed

Reusability

- Its parts can be used in other projects, so reprogramming is not needed

Software Quality and the Stakeholders



Software Quality: Conflicts and Objectives

The different qualities can conflict

- Increasing efficiency can reduce maintainability or reusability
- Increasing usability can reduce efficiency

Setting objectives for quality is a key engineering activity

- You then design to meet the objectives
- Avoids ‘over-engineering’ which wastes money

Optimizing is also sometimes necessary

- E.g. obtain the highest possible reliability using a fixed budget

Internal Quality Criteria

These:

- Characterize *aspects of the design* of the software
- Have an effect on the external quality attributes
- E.g.
  - —The amount of commenting of the code
  - —The complexity of the code

Short Term Vs. Long Term Quality

Short term:

- Does the software *meet the customer's immediate needs*?
- Is it sufficiently efficient for the volume of data we have *today*?

Long term:

- Maintainability
- Customer's future needs
- Scalability: Can the software handle larger volumes of data?

1.4 Software Engineering Projects

Most projects are *evolutionary* or *maintenance* projects, involving work on *legacy* systems

- Corrective projects: fixing defects
- Adaptive projects: changing the system in response to changes in
  - —Operating system
  - —Database
  - —Rules and regulations
- Enhancement projects: adding new features for users`
- Reengineering or perfective projects: changing the system internally so it is more maintainable

Software Engineering Projects

‘Green field’ projects

- New development
- The minority of projects

Software Engineering Projects

Projects that involve building on a *framework* or a set of existing components.

- A framework is an application that is missing some important details.
  - —E.g. Specific rules of this organization.
- Such projects:
  - —Involve plugging together *components* that are:
    - Already developed.
    - Provide significant functionality.
  - —Benefit from reusing reliable software.
  - —Provide much of the same freedom to innovate found in green field development.

1.5 Activities Common to Software Projects...

Requirements and specification

- Includes
  - —Domain analysis
  - —Defining the problem
  - —Requirements gathering
    - Obtaining input from as many sources as possible
  - —Requirements analysis
    - Organizing the information
  - —Requirements specification
    - Writing detailed instructions about how the software should behave

Activities Common to Software Projects...

Design

- Deciding how the requirements should be implemented, using the available technology
- Includes:
  - —*Systems engineering*: Deciding what should be in hardware and what in software
  - —*Software architecture*: Dividing the system into subsystems and deciding how the subsystems will interact
  - —*Detailed design* of the internals of a subsystem
  - —*User interface design*
  - —*Design of databases*

Activities Common to Software Projects

Modeling

- Creating representations of the domain or the software
  - —Use case modeling
  - —Structural modeling
  - —Dynamic and behavioural modeling

Programming

Quality assurance

- Reviews and inspections
- Testing

Deployment

Managing the process

1.6 Difficulties and Risks in Software Engineering

- **Complexity and large numbers of details**
- **Uncertainty about technology**
- **Uncertainty about requirements**
- **Uncertainty about software engineering skills**
- **Constant change**
- **Deterioration of software design**
- **Political risks**