Object-Oriented Software Engineering

Practical Software Development using UML and Java

Chapter 1:

Software and Software Engineering

Instructor : Humaira Afzal

Email : humairaafzal@cuilahore.edu.pk

Rules

Mobile phones – Silent or switch off

Arrive on time in class

If you do not understand a point, raise your hand and ask me to explain or contact during office hours

No disturbance!!!! No Misconduct!!!!

REMEMBER: Your first priority must be your studies

Out line

Software and software engineering

1.1 The nature of software

1.2 What is software engineering?

1.3 Software engineering as a branch of the engineering profession.

1.4 Stakeholders in Software Engineering

1.5 Software quality

1.6 Software engineering projects

1.7 Activities common to software projects

1.8 Difficulties and risks in Software Engineering as a whole

Objectives

You will learn about the following

- How does software differ from other products? What do we mean when we talk about high-quality software? What types of software are there and what are their main differences?
- How are software projects organized?
- How can we define software engineering?
- What activities occur in every software project?
- What should we keep in mind as we perform any software engineering activity?

1.1 The Nature of Software...

Software differs in important ways from the types of artifacts produced by other types of engineers.

Software is intangible

- Cannot feel the shape of a piece of software, and its design can be hard to visualize
- Hard to assess the quality and understand development effort

Software is easy to reproduce

- Cost is in its *development*

  —in other engineering products, manufacturing is the costly stage.

The Nature of Software ...

The industry is labor-intensive

- Hard to automate

Untrained people can hack something together

- Quality problems are hard to notice.

Software is easy to modify

- People make changes without fully understanding it
- As a side effect of their modifications, new bugs appear.

Software does not 'wear out'

- Software does not wear out with use like other engineering artifacts, but instead its design deteriorates as it is changed repeatedly.

The Nature of Software

Conclusions

- Much software has poor design and is getting worse
- Demand for software is high and rising
- We are in a perpetual ‘software crisis’
- We have to learn to ‘engineer’ software

Types of Software...

Custom

- For a specific customer
- Much custom software is developed in-house within the same organization that uses it; in other cases, the development is contracted out to consulting companies.
- Custom software is typically used by only a few people and its success depends on meeting their needs.
    - web sites, air-traffic control systems and software for managing the specialized finances of large organizations.

Types of Software...

Generic

- Sold on open market
- Requirements are determined largely by market research.
- Often called
  - COTS (Commercial Off The Shelf)
  - Shrink-wrapped software since it is commonly sold in packages wrapped in plastic.
    - word processors, spreadsheets, compilers, web browsers, operating systems, computer games and accounting packages for small businesses.

Types of Software...

Embedded

- Embedded software runs specific hardware devices which are typically sold on the open market. Such devices include washing machines, DVD players, microwave ovens and automobiles.

- Unlike generic software, users cannot usually replace embedded software or upgrade it without also replacing the hardware.

Types of Software

Real time software

- It has to react immediately (i.e. in real time) to stimuli from the environment (e.g. the pushing of buttons by the user, or a signal from a sensor).
    - Much design effort goes into ensuring that this responsiveness is always guaranteed.
    - Much real-time software is used to operate special-purpose hardware; in fact almost all embedded systems operate in real time.
    - Custom systems that run industrial plants and telephone networks are also real-time.
- Safety often a concern

Types of Software

Data processing software

- Data processing software is used to run businesses. It performs functions such as recording sales, managing accounts, printing bills etc.
- Design issues are how to organize the data and provide useful information to the users so they can perform their work effectively.
- Accuracy and security of data are key concerns

Some software has both aspects

For example, a telephone system has to manage phone calls in real time, but billing for those calls is a data processing activity.

Exercise

Classify the following software according to whether it is likely to be custom, generic or embedded (or some combination); and whether it is data processing or real-time.

(a) A system to control the reaction rate in a nuclear reactor.

(b) A program that runs inside badges worn by nuclear plant workers that monitors radiation exposure.

1.2 What is Software Engineering?...

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

  Software engineers have the responsibility to recognize situations when it would be most cost effective not to develop software at all, to develop simpler software or to purchase existing software.

- Software engineers must *communicate effectively* to identify and understand the problem

  Understand how people do their work, and to understand what impact any proposed software may have on its users' productivity.

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*
  - this is because software is normally continually changed over a period of years until it becomes obsolete.

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

  - Some software engineering techniques are aimed at increasing the quality of the design, whereas others are used to verify that sufficient quality is present before the software is released.

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

—Software engineers, like other engineers, therefore must ensure their systems can be produced within a limited budget and by a certain due date.

- requires careful planning and sticking to the plan in a disciplined way.

1.3 Software Engineering and the Engineering Profession

The term Software Engineering was coined in 1968

- People began to realize 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 and economics.
- Ethical practice is also a key tenet of the profession

In many countries, much software engineering does not require an engineering license, but is still engineering

Software Engineering and the Engineering Profession

Ethics in Software Engineering:

Software engineers shall

- Act consistently with public interest
- Act in the best interests of their clients
- Develop and maintain with the highest standards possible
- Maintain integrity and independence
- Promote an ethical approach in management
- Advance the integrity and reputation of the profession
- Be fair and supportive to colleagues
- Participate in lifelong learning

1.4 Stakeholders in Software Engineering

Many people are involved in a software engineering project. Classify these stakeholders into four major categories

1. Users

- Those who use the software
- Users appreciate
    - software that is easy to learn and use, makes their life easier, helps them achieve more, or allows them to have fun.

2. Customers (also known as clients)

- Those who pay for the software
- They may or may not be users – the users may work for them.
- Customers appreciate software that helps their organization save or make money, typically by improving the productivity of the users and the organization as a whole.

1.4 Stakeholders in Software Engineering

3. Software developers

- These are the people who develop and maintain the software.
- Specialized roles in development team

4. Development Managers

- who run the organization that is developing the software.
- Their goal is to please the customer or sell the most software, while spending the least money.

All four roles can be fulfilled by the same person

Exercise:

How do you think each of the four types of stakeholders described above would react in the following situation?

You implement a system according to the precise specifications of a customer. However, when the software is put into use, the users find it does not solve their problem.

1.5 Software Quality...

The following are five of the most important attributes of software quality.

Usability

- Users can learn the system 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 re programming is not needed

Software Quality and the 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

Development manager:
sells more and
pleases customers
while costing less
to develop and maintain

Software Quality: Conflicts and Objectives

- Achieving high reliability - repeatedly checking for errors and adding redundant computations; achieving high efficiency, in contrast, may require removing such checks and redundancy.

- Improving usability may require adding extra code to provide feedback to the users, which might in turn reduce overall efficiency and maintainability.

Good engineering practice is setting objectives for quality when starting a project, and then designing the system to meet these objectives.

Exercise

E3: For each of the following systems, which attributes of quality do you think would be the most important and the least important?

(a) A web-based banking system, enabling the user to do all aspects of banking on-line.

(b) An air traffic control system.

1.6 Software Engineering Projects

Divide software projects into three major categories:

1) Evolutionary projects - Those that involve modifying an existing system.

2) Greenfield projects - Those that involve starting to develop a system from scratch.

3) Those that involve building most of a new system from existing components.

Software Engineering Projects

Evolutionary projects:

Evolutionary or maintenance projects can be of several different types:

- Corrective projects involve fixing defects.
- Adaptive projects involve changing the system in response to changes in the environment in which the software runs.
- Enhancement projects involve adding new features for the users.
- Re-engineering or perfective projects involve changing the system internally so that it is more maintainable, without making significant changes that the user will notice.

Software Engineering Projects

Greenfield projects:

- Projects to develop an entirely new software system from scratch are significantly less common than evolutionary projects.
- In a greenfield project you are not constrained by the design decisions and errors made by predecessors

Software Engineering Projects

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

- The third type of software project can be considered neither evolutionary nor new development.
- 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.

Example

For example, imagine an application framework for ticketing. Such a system would have basic capabilities for reserving and printing tickets for events or travel.

These functions would be well designed and tested by the original developers of the framework.

However, many details would need to be added to handle the particular needs of each new organization that adopts the framework. Selling tickets for a theater can be quite different from selling tickets for a sporting event, a tropical holiday package or even a cinema.

1.7 Activities Common to Software Projects...

Requirements and specification

—Domain analysis

- understanding the background needed so as to be able to understand the problem and make intelligent decisions.

—Defining the problem

- narrowing down the scope of the system by determining the precise problem that needs solving.

—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 requirements should be in hardware and what in software – necessary for embedded and real-time systems
  - —*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 behavioral modeling

Programming

Quality assurance

- Reviews and inspections
- Testing

Deployment

Managing the process

1.9 Difficulties and Risks in Software Engineering

- **Complexity and large numbers of details**
  a) it is easy to add new features
  b) software developers typically add features without fully understanding a system.
  c) The system may not have been originally designed to accommodate the features.
- **Uncertainty about technology**
- **Uncertainty about requirements**
- **Uncertainty about software engineering skills**
- **Constant change**
- **Deterioration of software design**
- **Political risks**