Introduction to Software Engineering

Software Products

Software **is some executable object like source code, object code, or a complete program.**

A *software product* is software plus all supporting items and services that together meet a user's needs.

Software Engineering

Software engineering **is the technical and managerial discipline concerned with the systematic invention, production, and maintenance of high quality software products delivered on time, at minimum cost.**

Projects

A *project* is a one-time effort to achieve a particular, current goal of an organization, subject to specific time or cost constraints.

Why Projects Fail

Projects usually fail because of management mistakes rather than technical mistakes—managerial issues are more important than technical issues in software engineering!

Management

Management **is the activity of bringing an organization's resources to bear to achieve its goals as efficiently as possible.**

Management Activities

- *Staffing*: acquiring, training and educating, and assigning human resources
- *Organizing*: structuring problems, resources, and work assignments
- *Controlling*: directing resources, tracking progress, adjusting plans and organizations
- *Leading*: setting goals and objectives, motivating, facilitating resolution of conflicts and problems, removing obstacles, and supporting people

Project Management

Project management **is the application of management principles to plan, staff, organize, control, and lead a project.**

Software Project Management

Software project management **is management of projects to create and maintain software systems.**

Software projects are difficult to manage because:

- it is hard to estimate time and costs accurately
- they are large and require many resources
- much coordination of sub-tasks is required
- it is hard to track progress

Software Project Failures

- Lack of project planning and control is a major cause of schedule slippage, cost overruns, poor quality, high maintenance costs, and outright project failures
- Early planning is difficult because so much is unknown—but the point of early planning is to discover project unknowns, because they are the source of problems
- Early planning is also essential as the basis for refined planning later on

Basic Definitions

A *milestone* is any significant event in a project.

A *deliverable* is any document, program, data, service, hardware, or object produced for an internal or external customer.

Software Cost Estimation

- Most planning efforts begin with an estimate of the cost, in terms of time, money, or effort, of a project
- Cost estimation is very difficult in software development because it depends on so many factors

Work Breakdown Structure

A *work breakdown structure (WBS)* is a tree whose root represents an entire project or product, and whose other nodes represent a hierarchical decomposition of the project or product.

Scheduling & Resource Assignment

Scheduling **is planning when a task will occur.**

Resource assignment **is planning which people and other assets to use for a task.**

Control, tracking, **or** ***monitoring*** **is comparing plans to reality and adjusting plans as needed.**

Scheduling & Resource Assignment Tools

Two graphical tools are widely used for these scheduling and resource assignment:

- Gantt Charts
- PERT/CPM Charts

Gantt and PERT charts are sometimes combined to incorporate the advantages of both.

Gantt Chart

A *Gantt chart* is a bar chart that plots tasks or resources against time.

PERT/CPM Chart

A *PERT/CPM chart* is a directed graph that shows tasks, their durations, and their precedence relationships, and allows estimation of project completion times.

Project Management Software

Improved Planning—all tasks and their relationships must be considered and documented in detail

Improved Cost & Time Estimation—the software computes them correctly and consistently

Improved Communication—detailed, up-to-date reports are easily generated

Improved Tracking—actual completion dates and resource costs are easy to add and incorporate into schedules and estimates

Overall Improvement—many inconsistencies are detected by the software

Software Requirements

A *software (product) requirement* is a feature, function, capability, or property that a software product must have.

Software Design

A *software design* is a specification of a software system that programmers can implement in code.

What vs How

The traditional way to distinguish requirements from design:

What a system is supposed to do is *requirements*

How a system is supposed to do it is *design*

Requirements vs Design

Determining requirements is really the first step of design.

Requirements **state client needs and solution constraints.**

Designs **state problem solutions.**

Software Requirements Specification

A *software requirements specification (SRS)* document is a statement of all the requirements for a software product.

Functional Requirements

Functional requirements **specify what the system should do, that is, the services the system should provide, and the way the system should interact with its users.**

Software Design Defined

A *software design* is a specification of a software system that programmers can implement in code.

Requirements vs. Design

Determining requirements is really the first step of design.

Requirements **state client needs and solution constraints.**

Designs **state problem solutions.**

Software Process

A *software process* is a collection of activities used to develop and maintain software products.

Testing

Testing **is the exposure of a system to trial input to see whether it produces correct output.**

Who Should Test Software?

The software writer should never test his or her own software because:

- **Testers who don’t believe they will find faults generally won’t find any faults**
- **Testers who have to fix any faults they find tend not to find very many**
- **Coders want code to be fault free, but effective testers must want to find faults**

Testing Phases

Unit Test—Test of code written by a single programmer

Integration Test—Test of several units combined to form a module, sub-system, or system

System or Alpha Test—Test of a finished program by system testers

Acceptance or Beta Test—Test of a finished program by end-users or their representatives

Unit Testing

A *unit* is a portion of a system implemented by a single programmer.

***Unit testing* is exercising a unit in isolation from the rest of the system.**

Integration Testing

Integration testing **is exercising several units combined to form a module, subsystem, or system.**

Integration Testing Methods

Top-down—Combine, test, and debug top-level routines that become the integration test harness for lower-level units

Bottom-Up—Combine and test low-level units into progressively larger modules and subsystems

Sandwich—Mainly top-down with bottom-up integration and testing applied to certain widely used components

Top-Down Integration & Testing Advantages & Disadvantages

+ Design errors may be found sooner
+ The top levels of the system are tested the most
+ Distributes integration and testing throughout the coding phase
+ Tends to make bug location easier
+ A working version of the system is available early
+ Minimizes test harness creation
– Maximizes stub creation; it may not even be possible to write good stubs in some cases
– May be impractical (too time-consuming) to use an entire large system as a test harness

Bottom-Up Integration & Testing Advantages and Disadvantages

+Works well early in the integration process: thorough testing is possible, and bugs are easy to find

–Breaks down when large subsystems are combined: thorough testing is difficult and bugs are hard to find

–Design errors tend to show up late

–Much test harness is needed at each step of integration

Sandwich Integration & Testing

- A practical compromise between top-down and bottom-up integration and testing strategies
- The method used most often in real projects

System or Alpha Test

A *system* or *alpha test* is an exercise of a completed system against its requirements, usually by a team of system testers, in the development environment.

Acceptance or Beta Test

An *acceptance* or *beta test* is an exercise of a completed system by a group of friendly end users, or by the customer, to determine whether the system is ready for deployment.

Test Cases

A *test case* is a unit of testing activity. Test cases have three parts:

- ***goal*—the aspect of the system being tested**
- ***input and system state*—data provided to the system under stated conditions**
- ***expected behavior*—the output or action the system should take according to its requirements**

Regression Testing

Regression testing **is running all test cases over again after any software change.**


“TERIMA KASIH”