Software Engineering Overview

Yuchuan Deng

目录

Software

0.1 What is Software

- Instructions (computer programs)
- Data structures
- Documentation (descriptive information)

It is important to note that:

1. Programs are executable sequences of instructions designed to complete specific functions and performance.
2. Data structures allow programs to process information correctly.
3. Documentation includes graphical and textual information needed to develop, use, and maintain the programs.

0.2 Software Crisis

0.2.1 What is Software Crisis

A software crisis refers to a series of severe problems encountered during the development and maintenance of computer software.

0.2.2 Software Crisis Involves Two Major Issues

1. How to develop software to meet the growing demand.
2. How to maintain the ever-increasing number of existing software.

0.2.3 The Software Crisis Manifests in Several Typical Ways

1. Estimates of software development costs and schedules are often inaccurate.
2. Users are frequently dissatisfied with ”completed” software systems.
3. The quality of software products is often unreliable.
4. Software is frequently unmaintainable.
5. Software typically lacks adequate documentation.
6. The proportion of software costs in the total cost of computer systems increases year by year.
7. The rate of improvement in software development productivity lags far behind the rapid proliferation and deepening of computer applications.

0.3 Software Characteristics (The Differences Between Software and Hardware)

- Software is developed, not manufactured.
- Software does not wear out, but it can deteriorate.
- Most software is custom-built, not assembled out of components.

0.4 Software Engineering

In summary, software engineering is an engineering discipline that guides the development and maintenance of computer software. It employs engineering concepts, principles, techniques, and methods to develop and maintain software, combining time-proven management techniques with the best available technical methods to economically develop high-quality software and maintain it effectively.

0.5 Seven Basic Principles of Software Engineering

1. Manage with a phased life cycle plan.
2. Conduct phase reviews consistently.
3. Implement strict product control.
4. Use modern programming techniques.
5. Ensure results are clearly reviewable.
6. Keep the development team small and skilled.
7. Recognize the necessity of continuously improving software engineering practices.

Software Process

The Essence Practice

- Understand the problem (communication and analysis)
- Plan a solution (modeling and software design)
- Carry out the plan (code generation)
- Examine the result for accuracy

Generic View

- What is software process? Collection of activities, actions & tasks
- Software Engineering Layers: Tools, Methods, Process, focus on quality
- Genetic Software Engineering Framework Activities: Communication, Planning, Modeling, Construction, Deployment

- Different process models for SE proposed, all process models define a set of framework activities, a collection of tasks that conducted to accomplish each activity, work products produced as a consequence of the tasks, and a set of umbrella activities that span the entire process
- Process Assessment and Improvement
- Process Flow (linear process flow, iterative process flow, evolutionary process flow, parallel process flow)
- Umbrella Activities

Process Models

- Waterfall Model
- Incremental Model
- Prototyping Model
- Spiral Model
- Unified Process Model

Evolutionary Models

Agile

- What is agile? Rapid, incremental
  - Rapid deliver products
  - Effective response to change
  - Customers join the team, Effective communication among all stakeholders
  - Project plan must be flexible
  - Organizing a team so that it is in control of the work performed, communicate facilely
- The Manifesto for Agile
  - Individuals and interactions over processes and tools
  - Working software over comprehensive documentation
  - Customer collaboration over contract negotiation
  - Responding to change over following a plan
- Extreme Programming (XP): planning, design, coding, test
  - Key points of XP
    - User stories
    - KIS
    - Pair-programming

    - Refactoring
    - Continuous integration
    - Incremental delivery

- Scrum (Requirements, Analysis, Design, Evolution, Delivery)
- Kanban
- DevOps (Development & Operations)

Requirement

The Definition of Requirements Engineering

- RE is the tasks and techniques that lead to understanding of requirements
- RE builds a bridge to design and construction
- Inception, Elicitation, Elaboration, Negotiation, Specification, Validation, Requirements Management
- Elicitation Work Products

Quality Function Deployment (QFD)

- Normal / Expected / Exciting

ERD

- Example (选课：老师-学生-课程；借书：图书、借阅人、管理员)

Use-case

- A use case describes the actions system takes to deliver something of value to the actor
- Scenario—often called use cases, provide a description of how the system will be used
- Example: 编写用例图、编写用例场景

Objectives of Requirements Model

- To describe what the customer requires
- To establish a basis for the creation of a software design
- To define a set of requirements that can be validated once the software is built

Rules of Thumb (需求分析建模原则)

Requirements Modeling

- Scenario-based Modeling
- CRC

What is UML

- Diagrams of UML for analysis modeling
  - Use-case diagram
  - Activity diagram
  - Class diagram
  - State diagram
  - Deployment diagram
  - Sequence diagram
  - Swimlane diagram
  - Collaboration diagram

Class-Based Modeling

- Identifying Analysis Classes
  - "Grammatical parse"
  - Selecting Classes-Criteria
- Class-responsibility-collaborator (CRC) Models
  - Responsibilities are the attributes and operations encapsulated by the class
  - Collaborations are those classes that are required to provide a class with…
- Collaborations

Design

Design Conceptions

- Information Hiding
- Separation of Concerns
- Modularity
- Refactoring
- Cohesion
- Coupling

Object-oriented Design Conceptions

- Class, attribute, operation
- Polymorphism
- Encapsulation
- Inheritance

Quality Attributes - FURPS

- Functionality
- Usability
- Reliability
- Performance
- Supportability

Four Design Models

- Data Design, Architectural Design, Interface Design, Component-level Design, Deployment level Design

Architectural Design

- Software Architecture: the structure of the system, which comprise···
- Why is Architecture important?
- Architectural styles
  - Data-centered architectures
  - Data flow architectures
  - Call and return architectures
  - Object-oriented architectures
  - Layered architectures

Component-level Design

- Component definition
- What is Component? O-O View; Conventional view
- The definition of Decision Table

Component-level Design (class-based)

- OCP (The Open-Closed Principle)
- LSP (The Liskov Substitution Principle)
- DIP (Dependency Inversion Principle)
- ISP (The Interface Segregation Principle)

Interface Design

- Golden Rules
  - Place the user in control
  - Reduce the user’ s memory load
  - Make the interface consistent
- User Interface Design Models
  - User model—a profile of all end users of the system
  - Design model
  - Mental model
  - Implementation model

Test

Test

- Testing is the process of exercising a program with the specific intent of finding errors prior to delivery to the end user.

Test Strategies

- Unit Test
- Integration Test
  - Top-down Integration
  - Bottom-up Integration
  - Sandwich Testing
- Validation Test
  - Configuration Review
  - Acceptance Testing
- System Test

  - Recovery Testing, Security Testing, Stress Testing, Performance Testing, Deployment Testing, Configuration Testing

Test Tactics

- White-box Test
  - Sometimes called glass-box testing, is a test-case design philosophy that uses the control structure described as part of component-level design to derive test cases
- Black-box Test
  - Black-box testing, also called behavioral testing, focus on the functional requirements of the software

Other

- Regression Testing
- Smoke Testing
- Debug, Debugging, Techniques

OO Test

- OO Unit Testing
- OO Integration Testing

基本路径测试