Week 8 - Design Concepts 设计概念

1. Design in Software Engineering Context 软件工程中的设计

1.1 Basic Concepts 基础概念

Design elements include the following 设计包含的要素：

- Principles 原则
- Concepts 概念
- Practices 实践

The goal of design is to produce a model or representation with the following characteristics
设计目标是产生具有以下特征的模型或表示(model/representation)：

- Firmness (no bugs) 稳固性(无 bug)
- Commodity (useful/valuable) 实用性(有用/有价值)
- Delight (pleasurable experience) 愉悦性(良好体验)

1.2 From Requirements to Design 从需求到设计

Software design is 软件设计是：

- The last software engineering action in modeling activity, preparing for construction (coding and testing)

  > 建模活动中的最后一个软件工程动作，为构建(编码和测试)做准备

- A process to transform user requirements into a form suitable for programming

  > 一个将用户需求转化为适合编程的形式的过程

Note: In practice, requirements and design are interrelated. Work during the design process often helps clarify and refine requirements.
注意：在实践中，需求和设计是相互关联的。特别是在设计过程中的工作往往能帮助明确和完善需求。

Design levels include 设计层次包括：

1. Architectural Design 架构设计：

   > - Defines relationships between major structural elements 定义主要结构元素之间的关系
   > - Determines architectural styles and patterns 确定架构风格和模式
   > - Defines implementation constraints 定义实现约束

2. Interface Design 接口设计：
   - Describes system communication methods 描述系统通信方式
   - Defines human-computer interaction 定义人机交互方式
3. Component-level Design 组件级设计：
   - Transforms architectural elements into program descriptions 将架构元素转化为程序描述
   - Focuses on implementation details 关注实现细节
4. Data/Class Design 数据/类设计：
   - Converts class models into concrete implementations 将类模型转换为具体实现
   - Designs necessary data structures 设计必要的数据结构

2. Quality Control in Design Process 设计过程的质量控制

2.1 Quality Control Goals 质量控制目标

Three main goals 三个主要目标：

1. Requirements Implementation 需求实现：
   - Implement all explicit requirements 实现所有显式需求
   - Meet implicit requirements 满足隐式需求
   - Satisfy stakeholder expectations 满足利益相关者期望
2. Readability and Understandability 可读性和可理解性：
   - Code generation guidelines 代码生成指南
   - Testing and maintenance reference 测试和维护参考
3. Completeness 完整性：
   - Provide complete software view 提供完整软件视图
   - Cover data, functional, and behavioral domains 覆盖数据、功能和行为领域

2.2 Quality Control Importance 质量控制重要性

Key values 关键价值：

- Defect Prevention: Early problem detection 缺陷预防：早期发现问题
- Cost Effectiveness: Lower cost of fixing issues during design phase 成本效益：设计阶段修复问题成本较低

- User Satisfaction: Better meeting user needs 用户满意度：更好满足用户需求
- Enhanced Reliability: Ensuring system stability 增强的可靠性：确保系统稳定运行
- Compliance and Standards Adherence: Meeting industry standards and regulatory requirements 合规性与标准遵守：符合行业标准和法规要求
- Documentation and Traceability: Facilitating maintenance and auditing 文档和可追溯性：便于维护和审计
- Overall System Quality Improvement 系统质量的整体提升

2.3 FURPS Quality Attributes FURPS 质量属性

Five core attributes 五个核心属性：

1. Functionality 功能性：
    - Evaluate feature set and capabilities 评估功能集和能力
    - Assess function generality 评估功能通用性
    - Check system security 检查系统安全性
2. Usability 可用性：
    - Human factors 人为因素
    - Overall aesthetics 整体美感
    - Consistency 一致性
    - Documentation 文档
3. Reliability 可靠性：
    - Failure frequency and severity 失败频率和严重程度
    - Output accuracy 输出准确性
    - Mean Time To Failure (MTTF) 平均故障时间
    - Recovery ability 恢复能力
    - Predictability 可预测性
4. Performance 性能：
    - Processing speed 处理速度
    - Response time 响应时间
    - Resource consumption 资源消耗

- Throughput 吞吐量
- Efficiency 效率

5. Supportability 可支持性：
   - Extensibility 可扩展性
   - Adaptability 适应性
   - Maintainability 可维护性
   - Testability 可测试性
   - Compatibility 兼容性
   - Configurability 可配置性
   - Ease of installation 安装便捷性
   - Problem location capability 问题定位能力

2.4 Design Guidelines 设计指导原则

Eight technical criteria 八个技术标准：

1. Architecture Characteristics 架构特征：
   - Use recognizable architectural styles/patterns
     > 使用可识别的架构风格/模式
   - Components have good design characteristics
     > 组件具有良好设计特征
   - Support evolutionary implementation
     > 支持演进式实现
2. Modularity Requirements 模块化要求：
   - Logical partitioning
     > 逻辑分区
   - Reasonable subsystem division
     > 合理的子系统划分
3. Distinct Representations 区分表示：
   - Data 数据
   - Architecture 架构
   - Interfaces 接口

> - Components 组件

4. Data Structure 数据结构：

   > - Suitable for classes to be implemented
   >   > 适合待实现的类
   > - Based on recognizable data patterns
   >   > 基于可识别的数据模式

5. Component Independence 组件独立性：

   > - Exhibit independent functional characteristics
   >   > 展现独立功能特征

6. Interface Simplification 接口简化：

   > - Reduce complexity of connections between components
   >   > 减少组件间连接复杂度
   > - Simplify interaction with external environment
   >   > 简化与外部环境的交互

7. Method Repeatability 方法可重复性：

   > - Based on requirements analysis
   >   > 基于需求分析
   > - Use repeatable design methods
   >   > 使用可重复的设计方法

8. Clear Representation 表示清晰：

   > - Use effective notation
   >   > 使用有效表示法
   > - Clearly communicate design meaning
   >   > 清晰传达设计含义

3. Design Concepts 设计概念

3.1 Definition 定义

Design concepts are 设计概念是：

- Foundational ideas guiding system creation
  > 指导系统创建的基础思想
- Principles for organizing software systems
  > 组织软件系统的原则

- Guidelines affecting architecture and user experience

  > 影响架构和用户体验的指南

3.2 Core Concepts 核心概念

Main concepts include 主要包括：

- Abstraction 抽象
- Modularity 模块化
- Functional Independence 功能独立性
- Coupling 耦合
- Cohesion 内聚
- Object-Oriented Design 面向对象设计

3.3 Abstraction 抽象

Characteristics 特点：

- Highlight key features

  > 突出关键特征

- Hide unnecessary details

  > 隐藏非必要细节

- Layered description: from high-level abstraction to low-level implementation

  > 分层次描述：从高层抽象到低层实现

3.4 Modularity 模块化

Characteristics 特征：

- Cluster similar/related functions

  > 聚类相似/相关功能

- Establish boundaries

  > 设立边界

- Provide communication interfaces

  > 提供通信接口

Advantages 优势：

- Improve manufacturing efficiency

  > 提高制造效率

- Save time
  > 节省时间
- Support independent development
  > 支持独立开发

Trade-off considerations 权衡考虑：

- Increased module count leads to higher integration costs
  > 模块数量增加会导致集成成本上升
- Need to balance module size and integration costs
  > 需要在模块大小和集成成本之间找到平衡点
- Can analyze optimal module count through cost/effect curve
  > 可以通过成本/效果曲线来分析最佳模块数量

3.5 Functional Independence 功能独立性

Definition 定义：

- Degree of module independent operation
  > 模块独立运行的程度
- Characterized by low coupling and high cohesion
  > 以低耦合和高内聚为特征

3.5.1 Coupling 耦合

Definition 定义：

- Degree of interdependence between modules
  > 模块间相互依赖程度
- Goal is to achieve loose coupling
  > 目标是实现松散耦合
- Interface-based interaction
  > 基于接口的交互

Example code 示例代码：

```
// Tight coupling example 紧耦合示例
class Author {
    private String skypeID;
    // ... other code
}
```

```
class Editor {
    private Author author;
    public void contact() {
        // Direct access to private variable, high coupling
        // 直接访问私有变量，高度耦合
        String id = author.skypeID;
    }
}

// Improved loose coupling example 改进后的松耦合示例
class Author {
    private String skypeID;
    public String getSkypeID() {
        return skypeID;
    }
}

class Editor {
    private Author author;
    public void contact() {
        // Access through public method, reduced coupling
        // 通过公共方法访问，降低耦合
        String id = author.getSkypeID();
    }
}
```

3.5.2 Cohesion 内聚

Definition 定义：

- Measure of related responsibilities
  > 相关职责的度量
- Focus on single task/purpose
  > 关注单一任务/目的
- Elements working together
  > 元素协同工作

Types 类型：

1. Method Cohesion 方法内聚
   > - All statements in method serve a single purpose
   >   > 方法内的所有语句都应该服务于单一目的
2. Class Cohesion 类内聚

   > - All members serve a clear concept
   >   > 类的所有成员应该服务于一个清晰的概念

3. Module Cohesion 模块内聚

   > - Components within module are closely related
   >   > 模块内的组件应该紧密相关

4. Component Cohesion 组件内聚

   > - Components should encapsulate related functionality
   >   > 组件应该封装相关的功能

3.6 Object-Oriented Design 面向对象设计

Definition 定义：

- System as collection of interacting objects
  > 系统作为交互对象的集合

- Objects encapsulate data and behavior
  > 对象封装数据和行为

- Promotes modularity and reusability
  > 促进模块化和可重用性

4. Design Model Elements 设计模型元素

4.1 Overview 概述

Design model is 设计模型是：

- Detailed framework
  > 详细框架

- Development process guide
  > 开发过程指南

Contains elements 包含元素：

1. Data Design Elements
   > 数据设计元素

2. Architectural Design Elements
   > 架构设计元素

3. Interface Design Elements
   > 接口设计元素

4. Component-Level Design Elements
   > 组件级设计元素

5. Deployment-Level Design Elements
   > 部署级设计元素

4.2 Data Design Elements 数据设计元素

Hierarchy example 层次示例:

1. Business Level 业务层次:
   > - Define high-level goals
   >   > 定义高层目标
   > - Identify required data
   >   > 识别所需数据
   > - Example: increase sales, optimize inventory, enhance satisfaction
   >   > 例如：增加销售、优化库存、提升满意度
   > - Determine required data types: product, customer, order, inventory
   >   > 确定所需数据类型：产品、客户、订单、库存

2. Application Level 应用层次:
   > - Design core functionality data usage
   >   > 设计核心功能数据使用
   > - Example: product catalog, order processing
   >   > 示例：产品目录、订单处理
   > - Implement specific functions: product catalog, order processing, inventory management, recommendation engine
   >   > 实现具体功能：产品目录、订单处理、库存管理、推荐引擎

3. Program Component Level 程序组件层次:
   > - Implement data structures
   >   > 实现数据结构
   > - Example: Product, Customer classes
   >   > 示例：Product、Customer类
   > - Specific implementation: Product, Customer, Order, Inventory classes and their data structures
   >   > 具体实现：Product、Customer、Order、Inventory类及其数据结构

4.3 Architectural Design Elements 架构设计元素

Key aspects 关键方面：

- Define overall software layout
  > 定义软件整体布局
- Determine component relationships and communication methods
  > 确定组件关系和通信方式
- Based on application domain information
  > 基于应用领域信息
- Consider requirements model elements
  > 考虑需求模型元素
- Utilize architectural styles and patterns
  > 利用架构风格和模式

4.4 Interface Design Elements 接口设计元素

Three key aspects 三个关键方面：

1. User Interface (UI) 用户界面：
   > - Human-computer interaction design
   >   > 人机交互设计
   > - User experience considerations
   >   > 用户体验考虑
2. External Interfaces 外部接口：
   > - Other systems
   >   > 其他系统
   > - Devices
   >   > 设备
   > - Networks
   >   > 网络
3. Internal Interfaces 内部接口：
   > - Between design components
   >   > 设计组件之间

   - Component communication methods
     > 组件通信方式

4. 图片示例：

4.5 Component-Level Design Elements 组件级设计元素

Key characteristics 关键特征：

- Detailed description of each software component's internal structure
  > 详细描述每个软件组件的内部结构
- Use UML diagrams for representation
  > 使用UML图表示
- Multiple levels of abstraction possible
  > 可以有多个抽象层次
- Include detailed program flow
  > 包含详细的程序流程

4.6 Deployment-Level Design Elements 部署级设计元素

Deployment configuration description 部署配置说明：

1. Web Server Web服务器：
   - Process static content
     > 处理静态内容
   - SSL termination
     > SSL终止
   - Reverse proxy
     > 反向代理
2. Application Server 应用服务器：
   - Run business logic
     > 运行业务逻辑
   - Handle API requests
     > 处理API请求
   - Horizontal scaling
     > 水平扩展
3. Database Server 数据库服务器：

   - Data persistence
     > 数据持久化
   - Data backup
     > 数据备份
   - Transaction management
     > 事务管理

4. Load Balancer 负载均衡器：
   - Traffic distribution
     > 流量分发
   - Health checking
     > 健康检查
   - Session persistence
     > 会话保持