软件开发方法和过程

1. 软件开发的关键因素

- 压力与进步：探讨如何高效、低成本地开发优质软件产品。
- 软件开发的可行性变量：经济可行性取决于开发质量和生命周期与开发成本的匹配。

2. 软件产业的发展

- 提高整体的可行性：权衡生产成本、质量和生命周期。
- 软件开发方法的发展：不同新开发方法从不同角度实现同一目标。

3. 提高抽象层次

- 与软件技术发展密切相关的三个要素：计算机平台、人的思维模式和问题的基本特征。
- 提高解决问题的抽象层次：有效利用抽象手段解决软件开发问题。

4. 两个较为显著的进展

- Stephen J. Mellor指出：过去50多年里，软件开发取得了两个显著进展：开发出高抽象层次的程序设计语言和在更高抽象层次上实现软件复用。

5. 简单回顾软件开发的方法和过程

- 以机器为中心的计算：最早期程序员以0和1编写机器指令。
- 以应用为中心的计算：3GL的出现提高了生产力。
- 以企业为中心的计算：基于组件开发和分布式计算。

6. 中间件

- 提升抽象层次：中间件提供了位于操作系统层之上的计算抽象层。
- 中间件分化带来了异构性：不同技术标准的中间件产品并存，没有“最终赢家”。

7. OMG的标准

- OMG颁布的重要技术无关建模标准：UML、MOF、XMI、CWM。
- MDA框架规范：模型驱动软件开发的框架性标准。

8. MDA的主要思想

- 分离业务功能分析与设计和实现技术与平台的紧耦合关系。

9. MDA对抽象层次的划分

- Platform-Independent Model (PIM)：与实现技术和平台无关的模型。

面向对象的设计方法

1. 面向对象设计概述

- 软件设计的复杂性：软件可能是人类能制造的最复杂的实体，固有的复杂性导致开发困难、超支、延期等问题。
- 控制软件复杂性的手段：包括结构化设计方法、功能分解、抽象、模块化和信息隐藏。
- 结构化设计方法存在的问题：功能与数据分离、理解鸿沟、维护困难和限制软件可重用性。

2. 基于UML的面向对象设计

- UML的作用：运用面向对象概念构造系统设计模型，从分析模型构造设计模型，提供交流文档。
- 面向对象设计原则：包括类、对象、封装、消息、继承、多态性等。
- 4+1视图方法：逻辑视图、开发视图、处理视图和物理视图。

3. 面向对象设计阶段的建模

- 功能：用例图。
- 结构：类图、对象图、包图、复合结构图。
- 行为：状态机图、顺序图、活动图、通讯图。
- 实现：构件图、部署图。
- 时间：定时图。

4. 设计原则和过程

- 设计原则：模块化、层次分解、耦合、内聚、复用、体系结构和模式。
- 设计过程：确定分析到设计的映射、构造结构模型、行为模型和体系结构模型。

5. 设计类和类的精化设计

- 设计类：考虑实现因素，描述操作参数、属性和类型。
- 类的精化设计：精化类的属性和操作，明确定义操作参数和基本实现逻辑。

6. 类间关系和包与包图

- 类间关系：关联、聚集、泛化、依赖、实现和约束。
- 包与包图：任何大系统划分为小单元，包图描述包和包之间的静态关系。

7. 活动图和状态图设计

- 活动图：描述系统中各种活动的执行顺序，用于描述操作流程或用例处理流

UML图

2. UML图分类

- 结构图：包括类图、组件图、对象图、包图、复合结构图、部署图。
- 行为图：包括用例图、状态机图、活动图、序列图、通信图、交互概览图、时间图。
- 实现图：包括构件图和部署图。

3. 用例图（Use Case Diagram）

- 描述系统外部执行者与系统的用例之间的联系。
- 用例是系统提供的功能描述。
- 执行者是可能使用这些用例的人或外部系统。

4. 类图（Class Diagram）

- 描述系统中的类及其相互之间的关系。
- 反映系统中包含的各种对象的类型以及对象间的静态关系。

5. 对象图（Object Diagram）

- 类图的实例化，表示具体对象实例及其相互关系。

6. 包图（Package Diagram）

- 显示类的包以及这些包之间的依赖关系。

7. 复合结构图（Composite Structure Diagram）

- 描述类层次分解为内部结构。

8. 状态机图（State Machine Diagrams）

- 描述对象可能的状态以及事件发生时状态的转移。

9. 序列图（Sequence Diagram）

- 描述对象之间动态的交互关系，体现消息传递的时间顺序。

10. 活动图（Activity Diagram）

- 描述系统中各种活动的执行顺序，用于描述操作流程或交互流程。

11. 通信图（Communication Diagrams）

- 描述对象之间的消息收发关系，强调交

- Platform-Specific Model (PSM)：与具体实现技术和平台相关的应用模型。

10. 模型驱动工程（MDE）

- 以模型为首要软件制品：通过建模构造软件系统的业务模型，然后依靠模型转换驱动软件开发。

11. 软件开发的两个趋势

- 关注点从小规模编程向大规模编程转变。
- 语言和基础设施的不断演进。

12. 从语言的演化看开发方法

- 从汇编语言到高级语言：高级语言的演进速度和丰富程度极快。
- 语言设计上的两种基本思路：偏向于机器运行性能的考虑和偏向人的思维方式与习惯。

13. 面向对象的“窘境”

- 不同的解释和实现：深刻理解“对象模型”的本质是根本，也是难点。

14. 程序员的角色和软件本身形态的演化

- 程序员的角色在分化：从高智商的代名词、公司的“白领”慢慢变成“蓝领”，最后沦落为“码农”。
- 软件形态的演化：从传统服务器、桌面系统到分布式软件系统、基于中间件的软件系统、信息物理融合系统、基于云平台的软件系统。

程。

- 状态图：展示对象可能的状态以及事件发生时状态的转移情况。

8. 交互图和组件图

- 交互图：仅考虑顺序图，描述对象间的动态交互关系。
- 组件图：描述可重用的系统片段，具有良好定义接口的物理实现单元。

9. 部署图和数据库参与的类图

- 部署图：反映系统中软件和硬件的物理架构，表示系统运行时的处理节点和组件配置。
- 数据库参与的类图：分割数据库的行为，展示逻辑与物理之间的交互。

10. 总结

- 学习内容：面向对象设计和UML设计建模。
- 实践要求：会设计各种UML设计模型，包括类图、包图、顺序图、状态机图、活动图、组件图、配置图，并能使用基于UML的面向对象设计完成课程项目的设计。

互关系而非时间顺序。

12. 交互概览图（Interaction Overview Diagrams）

- 结合活动图和序列图的特点，强调控制流描述。

13. 构件图（Component Diagram）

- 描述软件构件及其依赖关系。

14. 部署图（Deployment Diagram）

- 描述系统中硬件和软件的物理配置情况和系统体系结构。

15. 时间图（Timing Diagram）

- 描述状态或值随时间的变化情况，以及带时间约束的交互行为。

16. UML支撑环境与扩展

- 模型驱动的软件开发环境。
- UML的扩展包括实时模型、可执行模型、企业计算等。

17. UML框架下的软件工程

- 讨论了UML在软件工程中的应用，包括统一软件过程和软件组件市场。