软件工程重点总结 2018.01

1. 什么是软件工程：

软件工程是将系统化、规范化、可度量的方法应用于软件开发、运行和维护过程，即将工程化应用于软件中的方法的研究。从管理和技术两方面研究如何更好地开发和维护计算机软件的一门新兴学科。

2. 什么是软件危机：

*落后的软件生产方式无法满足迅速增长的计算机软件需求，从而导致软件开发与维护过程中出现一系列严重问题的现象。*如： 开发成本和进度的估计常常不准确； 用户对交付的软件系统不满意的现象经常发生；软件质量无保证、可靠性差； 软件可维护性非常低； 软件通常没有适当的文档资料； 软件成本在计算机系统总成本中所占比例逐年上升；

3. 什么是软件：

软件是计算机程序，规程以及运行计算机系统过程需要的相关文档和数据。在软件工程中，软件=程序+文档+数据。

4. 软件的生命周期（即开发所经历的阶段）：

软件生命周期是软件产品从考虑其概念开始到交付使用，直至最终退役为止的整个过程。分为如下三个阶段：

软件定义： 软件定义的任务是确定待开发的软件系统要做什么，又称为系统分析，可进一步划分为问题定义，可行性研究，需求分析三个阶段。

软件开发：软件开发的任务是具体设计和实现软件。通常由下面四个阶段组成：概要设计，详细设计，编码和单元测试、综合测试，前连个统称为系统设计，后两个统称为系统实现。

软件运行和维护：运行维护的任务是根据软件运行中的问题，对其进行修改，是系统能持久的满足用户需求。

5. 软件过程模型：

模型就是对实际事物、实际系统的一种抽象。所以软件过程模型是软件开发的全部过程、活动和任务的结构框架，直观的表达了软件开发的全过程，明确规定了要完成的主要活动、任务和开发策略。

作用：告诉人们应该去遵循一个什么样的过程和策略去开发软件系统。

关于以下原型的优缺点简单记住几条，考试扯扯就行了，主要理解各模型的过程。

1）瀑布模型： *这是个重点，老师画重点一直在说这个*

瀑布模型包含了各项软件工程活动，即制定开发计划、进行需求分析、软件设计、程序编码、测试及运行维护。*它规定了各项软件工程活动自上而下、相互衔接的固定次序*。瀑布模型是经典，是软件过程模型的里程碑。

特点：

上一项活动接受该项活动的工作对象作为输入；

利用这一输入实施该项应完成的内容；

产生本项活动的相关产出，作为输出传给下一项活动；

对本项活动执行情况进行平生。如果活动执行得到确认，则继续进行下一项活动；否则返回前一项，甚至对前一项的活动进行返工。

缺点：

在软件开发的初始阶段指明全部需求是非常困难的，有时甚至是不现实的。

需求一旦确定，用户和软件项目负责人需要等一段时间（经过设计、实现、测试、运行等阶段）才能得到一份软件的最初版本。

瀑布模型的软件活动是文档驱动的，当阶段之间规定过多的文档时，会大大增加软件开发的工作量，而且以文档的完成情况来评估项目完成进度时，往往会产生错误的结论。

2）原型模型： *这也是个重点，老师画重点一直在说这个*

原型表示软件的一个早期可运行的版本，反映最终系统的部分重要特征。*通俗的讲，原型模型就是先获取用户的基本需求，然后开发一个小型软件系统原型，然后根据用户的反馈，对原型进行不断的修改。*

作用：准确地确定用户的需求。

使用原型的策略：

废弃策略：先构造一个功能简单且性能要求不高的原型系统，根据用户的反馈，反复分析改进，形成好的设计思想，据此设计出完整、准确、一致、可靠的最终系统。原系统废弃不用。

追加策略：先构造一个功能简单且性能要求不高的原型系统，作为最终系统的核心，然后通过不断地进行扩充修改，逐步追加新要求，最后发展成为最终系统。

优点：

用户参与，尽早揭示软件中可能存在的风险及不确定因素，尤其是关于用户需求一致性方面的风险。

开发过程与用户培训过程同步,系统易维护，对用户更友好，产品柔性好。

缺点：

不经过系统分析并对系统进行整体划分，直接用原型模拟系统功能比较困难。

对于计算量大、逻辑性较强的程序模块，原型法很难真正构造出来供用户评价。

对于批处理系统，其大部分处理是内部进行的，应用原型法有一定困难。

项目文档容易被忽略，给后期原型的改进和维护造成困难。

对丢弃策略，很多工作被浪费掉，增加开发成本，降低开发效率。

以下模型划重点时一语带过，非重点。

3）螺旋模型

思想：使用原型及其他方法*来降低风险。* ***需求不明确****的大型软件系统的开发*

4）喷泉模型

比较灵活，具有迭代性和无间隙性。*可以随时回到上一个阶段。*

5）增量模型

可以让客户把需求逐步提出来。

6. **软件过程和面向对象，面向过程的区别：** *画重点的时候提出来的无厘头的问题*

   软件过程是指软件开发所经历的一系列阶段，而面向对象和面向过程只是这些阶段中某个阶段所采用的一些方法而已。（随便扯扯，有更好的回答欢迎反馈）

7. **可行性研究是干什么的？**

   可行性研究是研究这个软件项目是否值得去开发，需要从项目的技术可行性、经济可行性、操作可行性、社会可行性等方面进行研究和分析，并最终做出该项目是否具有可行性的结论。

8. **需求工程：**

   建立并使用完善的工程化方法 ，以较经济的手段获得准确表达用户需求的软件需求规格说明的一个学科。

9. 需求开发过程：

需求获取：通过访谈，问卷调查，专题讨论会等手段获取用户需求。

需求建模：用清晰、简明的方式将需求分析获得的信息记录下来，得到一个逻辑模型。

需求规格说明：用户和开发人员都充分了解用户需求后，将共同的理解以规范化的形式准确的表达出来，形成需求规格说明书。

需求评审：在将需求交付设计之前，对需求规格说明书进行彻底的检查和修改。

10. 需求管理过程：

软件开发过程中，可能会有需求的提出和更改，所以要进行需求管理。

需求变更控制：当出现需求变更时，就要对变更影响和成本进行分析，通过用户方和开发方组成的变更控制委员会来决策以规范和控制需求变更（肯定不能一方说改就改）

需求版本控制：它保证在需求文档中记录和反映所有的需求变化。

需求跟踪：维护需求的可跟踪性信息（哪个需求谁做了等等）。

需求状态跟踪：跟踪需求的状态（哪个需求进行到什么程度了，已完成还是已验证等等）

11. 模块化和模块独立性：

模块化是指解决一个复制问题时自顶向下逐层把软件划分成若干模块的过程，每个模块完成一个子功能。

模块独立性时指软件系统中每个模块只涉及软件要求的具体的子功能，而和软件系统中其他的模块的接口是简单的。

12. 面向对象的基本概念：

对象：将客观世界的实体抽象为问题空间中的对象。

类：把具有相同特征和行为的属性归在一起就形成了类。

消息：实现对象类之间的通信和任务传递。

封装：把对象属性和操作结合在一起，构成独立的单元，只能通过有限的接口于对象发生联系，不能直接存取对象的属性。

继承：子类可以自动拥有父类的全部属性和操作。

多态性：子类对象可以像父类对象一样使用，同样的消息可以发给父类也可以发给子类。

13. UML：

UML，统一建模语言，以面向对象图的方式来描述任何类型的系统模型。

14. 识别分析类：

识别实体类：实体类用于描述必须存储的信息，同时描述相关的行为。

识别边界类：边界类在系统与外界之间，为他们交换各种信息与事件。功能：输入、输出和过滤。

识别控制类：控制类与业务过程相关，它们控制整个业务的流程和执行次序。是一组操作的集合，用来协调各个边界类对象、实体类对象等。

15. 用户界面设计原则：

置用户于控制之下； 减轻用户的记忆负担； 保持界面一致；

16. 软件测试：

所有的软件问题都称为软件缺陷，不仅仅指 BUG。软件测试是为了发现软件缺陷，而无法证明软件的正确性。

静态黑盒测试：通过看产品说明书来查找缺陷。(好奇葩的方式)

动态黑盒测试：不追究代码细节，通过使用软件来进行测试。

静态白盒测试：通过看说明，结构，代码来找缺陷。

动态白盒测试：利用查看代码功能和实现方式得到信息，来确定如何展开测试。

17. 软件测试策略：

单元测试：针对软件设计的最小单元程序模块进行测试的工作。

继承测试：按照一定的策略对单元测试的模块进行组装，并在组装过程中进行模块接口与系统功能测试；

确认测试：目的是验证软件的有效性。按照软件需求说明书对软件的功能和性能要求进行测试。

系统测试：为了测试软件安装到实际应用的系统中后，能否正常工作，以及对系统运行出现的各种情况的处理能力。

18. 软件维护：

软件维护是为了纠正错误或满足新的需求而修改软件的过程。大致分为：

纠错性维护； 适应性维护； 完善性维护； 预防性维护；

19. 逆向工程和再工程：

逆向工程：分析已有的程序，寻求比源代码更高级的抽象表示形式，比如通过反汇编、反编译和动态跟踪等方法，分析一个软件的实现过程，这种行为就是逆向工程。

再工程：是指在逆向工程所获得的信息的基础上修改重构已有的系统，产生一个新的版本。

20. 面向对象设计原则：
    a) 单一职责原则：一个对象只负责一件事。
    b) 开放-封闭原则：类可以扩展，但不可以修改。
    c) Liskov 替换原则：子类应该能替换父类，出现在父类能出现的任何地方。
    d) 接口隔离原则：其实就是接口的单一职责原则。
    e) 依赖倒置原则：高层模块和底层模块都应该依赖于抽象，抽象不依赖于细节实现，实现细节依赖于抽象
21. 设计模式：描述一个在我们周围不断重复发生的问题，以及该问题的解决方案。
22. 设计模式的作用和研究意义：
    a) 优化的设计经验
    b) 极高的复用性
    c) 丰富的表达能力
    d) 极低的耦合度
23. 经典设计模式：
    a) 抽象工厂模式
        i. 抽象工厂类
        ii. 具体工厂类
        iii. 抽象产品类
        iv. 具体产品类
    b) 适配器模式：将某个类的接口转换成客户希望的另一个类的接口，使接口不兼容的类能一起协作
    c) 策略模式：针对一组算法，将每一个算法封装到具有共同接口的独立的类中，使他们可以相互转换

以上内容为老师所画重点的全部内容，但是第十章未进行整理，老师提到的是 10.2、10.5 和 10.8 节，追求高分的同学请自行理解。

至于各种画图，这里不再总结，可以去找往年的假题，练习一下画图等等。

- 软件开发方法的三要素：**过程**、**方法**、和**工具**
- 两种主要的软件开发方法：**面向过程的方法**（又叫结构化方法）、**面向对象的方法**
- **可行性研究目的：用最小的代价在尽可能短的时间内研究并确定客户提出的问题是否有行得通的解决办法，系统分析师的工作！！**
- **软件过程模型：**对软件开发全部过程的抽象
- **软件开发模型：**软件过程模型
- **软件开发方法：**软件开发过程中所涉及的思想、方法、技术、和工具的集合（包含过程、方法、工具）
- **需求的概念：**
  - **定位：**可行性研究阶段后，系统设计之前（承上启下）
  - **任务：**最终形成一份经开发方和用户认可或达成共识的**软件需求规格说明书**
- **需求规格说明：描述了一个软件系统必须实现的功能和性能，以及影响该系统开发的约束。是软件系统的逻辑模型的重要组成部分！**
- public **+**，**protected #**，**private –**
- **设计模式的四个基本要素**
  - **模式名称**
  - **问题：描述了应该在何时使用模式。**
  - **解决方案：描述了设计的组成成分，它们之间的相互关系及各自的职责和协作方式**
  - **效果**：描述了模式应用的效果及使用模式应权衡的问题。
- **结构化分析的逻辑模型:**
  - **数据字典**：核心，包含了软件使用和产生所有**数据的描述**
  - **数据流图**：功能建模，描述系统的**输入数据流**如何经过一系列的**加工**变换逐步变换成系统的**输出数据流**
  - **实体—关系图**：数据建模，描述数据字典中**数据之间的关系**
  - **状态转换图**：行为建模，描述系统接收哪些**外部事件**，以及在外部事件的作用下的**状态迁移情况**
- **衡量模块独立性的两个准则**
  - **内聚**：指**模块内部各成分之间的联系**，也称块内联系或模块强度
  - **耦合**：指**一个模块与其它模块之间的联系**，也称为块间联系
- **模块的独立性高**
  - **块内联系强**
  - **块间联系弱**
- **结构化设计的基本概念**
  - **两种数据流类型：变换型和事务型**
  - **系统体系结构的描述工具：系统结构图**
  - **结构化设计过程**
-
  - **变换分析**
  - **事务分析**
- **软件测试是有风险的行为。**
- **并非所有软件缺陷都要修复。**
- **产品说明书经常变化。**
- **应该在测试之前就制定出测试计划。**

- **穷举测试是不可能的。**
- 软件测试员必须测试程序的状态及其转换，既要进行**通过性测试**，又要进行**失效性测试**。
- **数据测试与状态测试**
- 系统测试包括**功能测试**、**压力测试**、**兼容性测试**、**安全测试**、安装测试以及恢复测试等。
- **软件测试过程:单元测试,集成测试,确认测试,系统测试**
- **测试面向对象程序的时候，除了继承传统的测试技术之外，还必须研究与面向对象程序特点相适应的新的测试技术。**
- **调试与测试是密不可分的两个活动。**
-

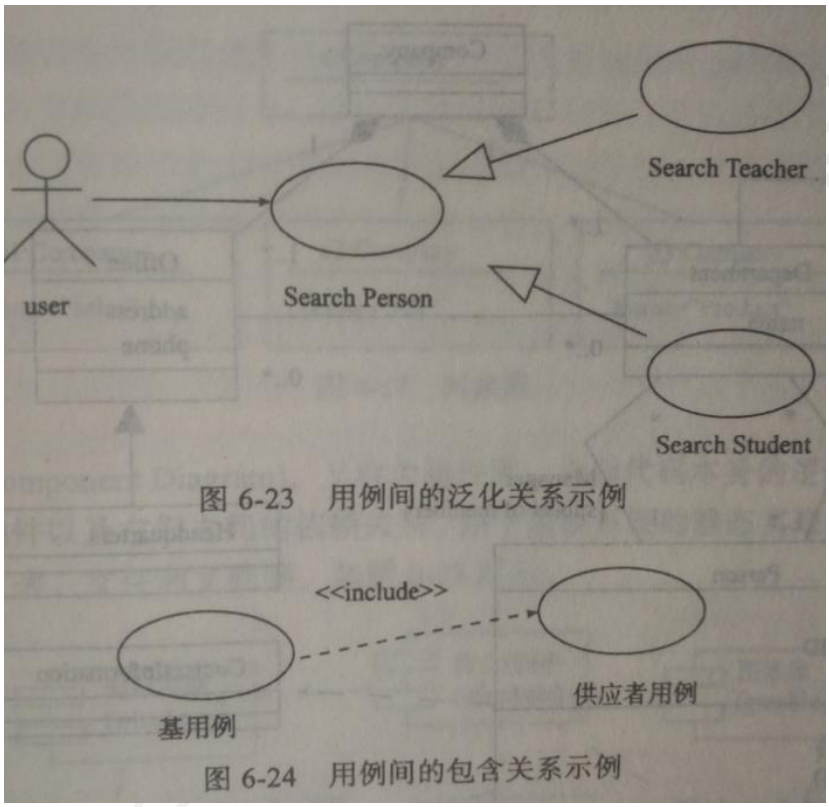


图 6-23　用例间的泛化关系示例

图 6-24　用例间的包含关系示例

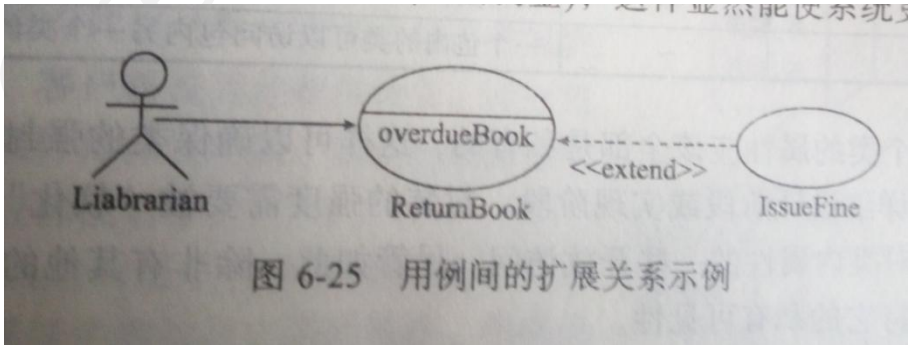


图 6-25　用例间的扩展关系示例

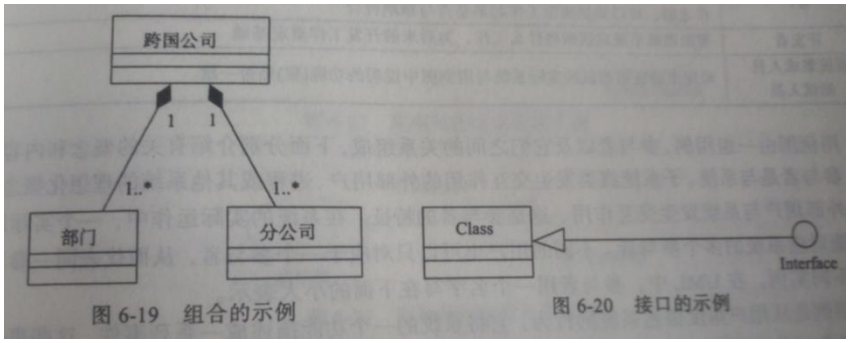


图 6-19 组合的示例　　图 6-20 接口的示例

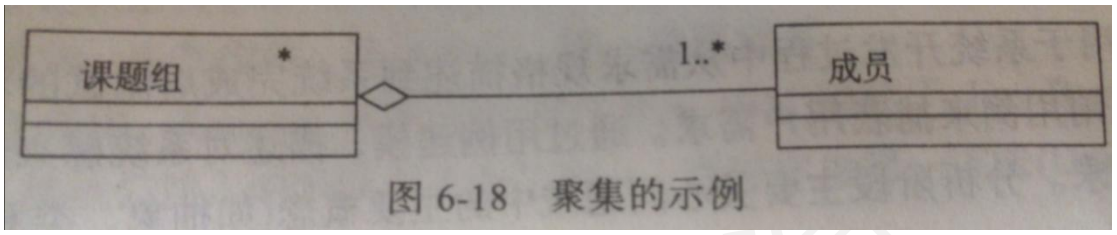
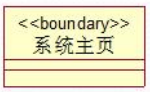


图 6-18 聚集的示例

三种分析类的图形表示

边界类：

系统主页



实体类：

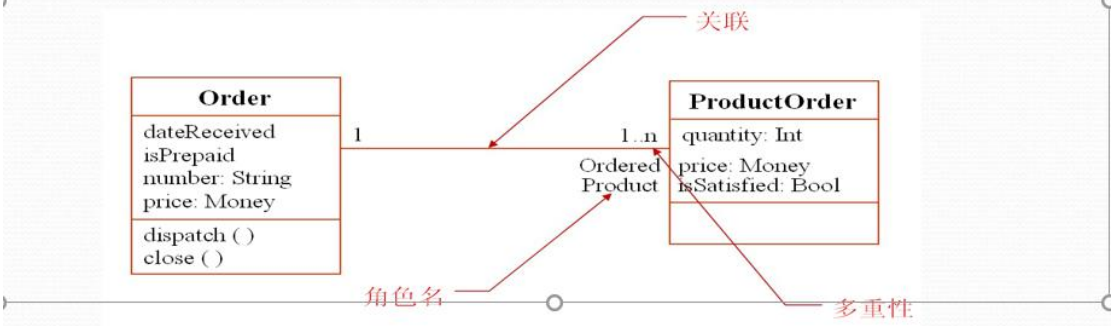
账户

控制类：

弹出

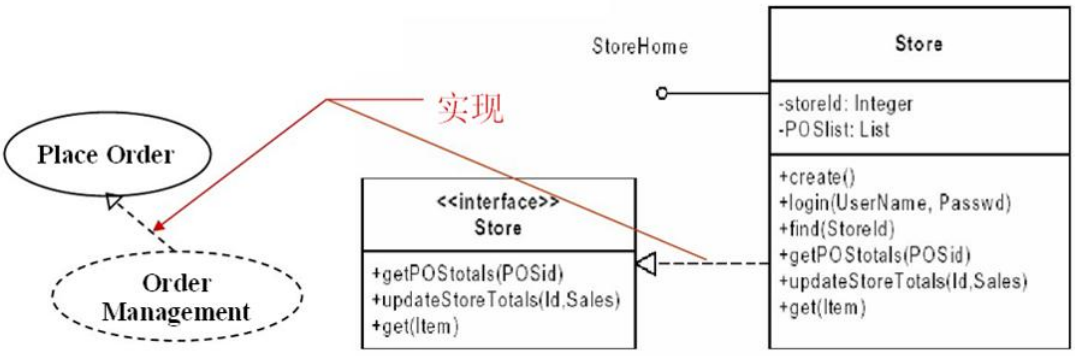
UML关系-1：关联（Association）

- ✓关联是一种结构关系，它描述了一组对象之间的静态连接
- ✓关联两端的类可以某种角色参与关联
  - 角色是关联中靠近它的一端的类对另一端的类呈现的职责
  - 如果关联上没有标出角色名，则隐含地用类的名称作为角色名



UML关系-4：实现（Realization）

- ✓实现是类之间的语义关系，其中的一个类指定了由另一个类保证执行的契约。
- ✓两种情况
  - 接口与实现它们的类或构件之间
  - 用例及其协作之间



数据流(data flow)：由一组固定成分的数据组成，代表数据的流动方向

加工(process)：描述了输入数据流到输出数据流的变换，即将输入数据流加工成输出数据流

文件(file)：使用文件、数据库等保存某些数据结果供以后使用

源或宿(source or sink)：软件系统输入数据的来源和输出数据的去向

小结

- 结构化分析是结构化方法在需求分析阶段的活动
- 结构化方法是面向数据流（或者过程）的传统方法，它以数据流为中心构建系统的逻辑模型
- 结构化分析的主要思想：抽象与分解
- 结构化分析的逻辑模型：数据字典、数据流图、实体-关系图、状态转换图
- 数据流图（DFD）描述输入数据流到输出数据流的变换(即加工)过程，用于对系统的功能建模。
- 数据流图由括源和宿、数据流、加工、文件组成
- 建立数据流图的步骤：第一步画系统的输入和输出，第二步画系统内部，第三步画加工内部，第四步重复第3步，直至每个尚未分解的加工都足够简单(即不必再分解)

小结(续)

- 分层数据流图的审查：一致性（父图子图的平衡、数据平衡、局部文件等），完整性
- 数据字典由不同的数据条目组成，而数据条目是对数据流图中的不同组成元素（源和宿、数据流、加工、文件）的说明
- 基本加工的说明描述的是一个基本加工应该做什么，描述方法有结构化语言、判定树、判定表
- 判定表由条件桩、条件条目、动作桩、动作条目组成

内聚

1.偶然性内聚
2.逻辑性内聚
3.时间性内聚
4.过程性内聚
5.通信性内聚
6.顺序性内聚
7.功能性内聚