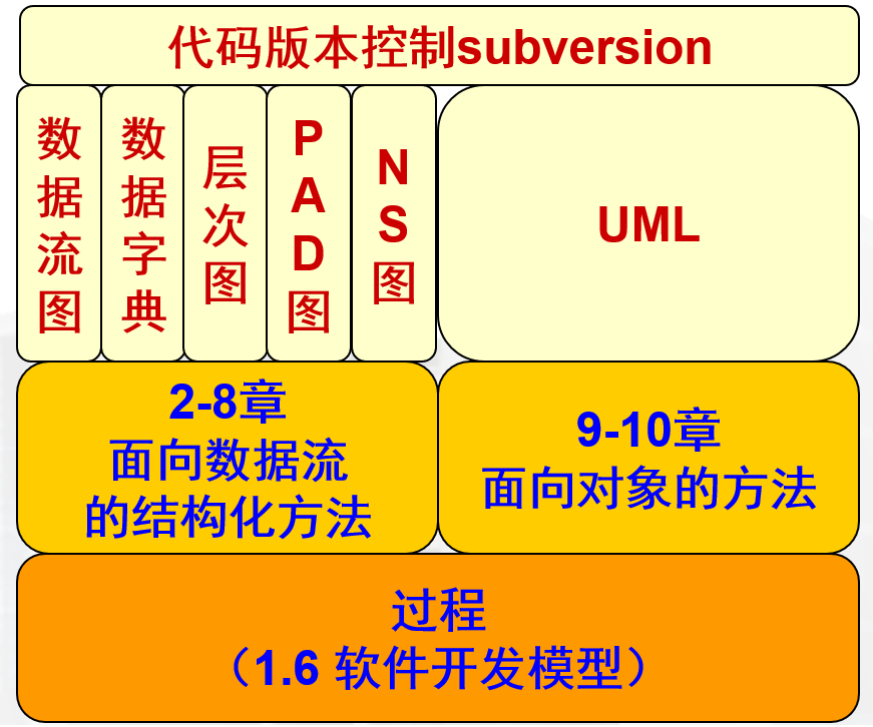
> *author*：王杰永
> *class:* 计算机科学与技术*19-3*

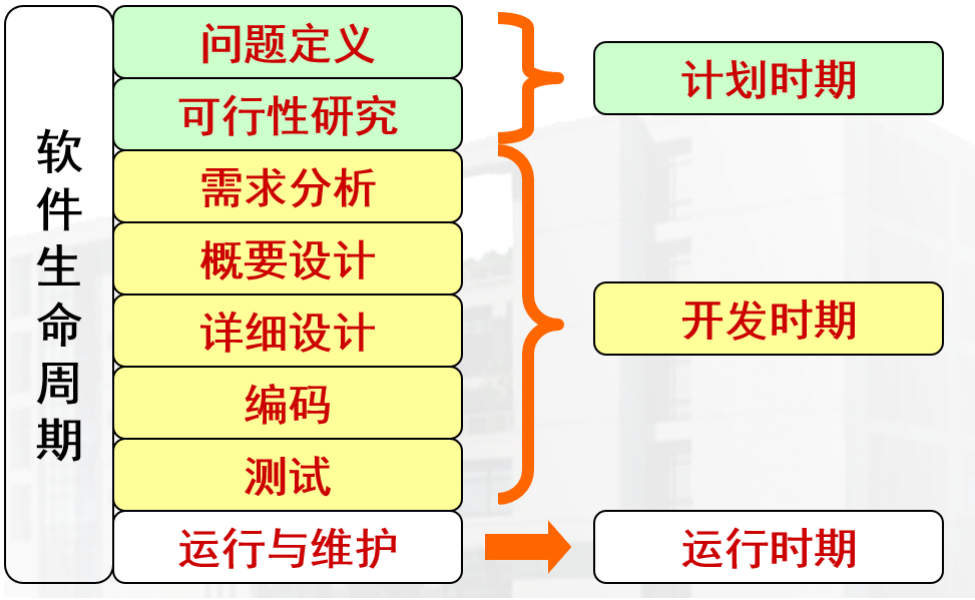
一、软件工程概述

软件工程是计算机软件开发的过程、方法和工具的学科。我们的软件工程课程主要讲了如下内容



1. 软件生命周期

软件生命周期是从软件目标的提出、定义、开发、维护，直到最终被丢弃的整个过程。



比较有趣的是，大部分人觉得软件中大部分错误是编码错误，但据统计，设计错误占软件错误的63%，编码仅占**37%**。

2. 软件开发模型

软件开发模型是对软件过程的建模，需要确定任务及执行顺序，保证质量和适应需求变化。

常见的软件开发模型：

- 瀑布模型
- 原型模型
- 增量模型
- 螺旋模型

二、软件开发的计划时期

通常，在计划时期，我们要完成问题定义以及可行性研究分析。

1. 问题定义

2. 可行性研究

在最短时间内，花费最小代价，确定定义的项目是不是可能实现和值得开发的。

- 经济可行性
- 技术可行性

- 运行可行性
- 法律可行性

三、需求分析

在需求分析阶段，我们要建立目标系统的逻辑模型，并形成《软件需求规格说明》。

> 目标系统的逻辑模型即反复与用户交流，调查“系统必须要做什么”，建立原型系统。
>
> 数据流图、数据字典、加工处理

1. 数据流图(Data Flow Diagram)

顶层DFD是系统的基本逻辑模型，包含一个加工处理和若干输入输出流

> 顶层*DFD*图中不应该出现外部存储

分层细化DFD的原则：

- 父子图平衡：将一个加工处理分解为一系列子加工时，分解前后的输入输出数据流必须相同。
- 区分全局文件与外部项

DFD图绘制规则：

- 顶层数据流图不要出现数据存储
- 数据存储之间、外部实体之间、数据存储与外部实体之间不能出现数据流
- 数据流是单向的，不能出现双向箭头。
- 任何加工都必须有输入和输出数据流

2. 数据字典

包括数据元素（数据项）、数据流、数据存储。

数据流

对DFD图中数据流的描述。

> 数据流名：
>
> 别名：
>
> 组成：
>
> 备注：

数据存储

对DFD图中的数据存储的描述

> 文件名：
>
> 组成：
>
> 组织：
>
> > 按某一字段，某种顺序排列
>
> 备注：

数据元素

数据元素是对数据流、数据存储中的字典条目（组成）的解释

> 数据项名：
>
> 别名：
>
> 取值：
>
> 备注：

3. 加工逻辑

加工逻辑用来描述DFD中每个加工能够“做什么”。

通常使用结构化语言、判定表、判定树、IPO图来描述加工逻辑。

4. 结构化分析方法(Structured Analysis)

在20世纪70年代中期，提出来了一种成为结构化设计(*structured design*)的软件设计技术。而在70年代后期，又有人提出了与SD配套的结构化分析(SA)技术，合称为结构化分析与设计方法。

SA的指导思想是自顶向下、逐步分解（对应于我们DFD图的绘制步骤），SA方法的步骤有：

- 分层细化DFD图
- 定义数据字典
- 定义加工逻辑

四、概要设计

在需求分析阶段得出了DFD图、数据字典之后，从需求分析阶段的结果除法，概要设计进行软件结构设计与数据设计，编写《概要设计说明书》。

- 软件结构设计：是概要设计的主要工作，分析系统由哪些模块组成，优化模块间的关系。
- 数据设计：将需求分析阶段创建的数据字典，转换为实现系统所需的数据结构（ER图等）。

1. 软件结构设计

通过软件结构设计，建立软件良好的模块结构，确定模块、模块间的关系

内聚与耦合

对于软件结构，有这么一句话：宁要塔型，不要饼型，提倡翁型

- 内聚：模块内部各成分之间的关联越高，即内聚越强

  > 功能内聚、顺序内聚、通信内聚、过程内聚、时间内聚、逻辑内聚、偶然内聚。

> 内聚程度依次降低

- 耦合：模块之间的联系越小，即耦合越松散

> 非直接耦合、数据耦合、特征耦合、控制耦合、外部耦合、公共耦合、内容耦合（最高的耦合）
>
> 耦合程度依次升高。

图形工具

层次图（H图）与结构图（SC图）

结构化设计方法（**SD**）

将系统的逻辑模型DFD图转换为软件结构图（H图、SC图）

2. 数据设计（PPT没有？？）

五、详细设计

在详细设计阶段，我们要确定每个模块的算法与数据结构，为每个模块设计一组测试用例，编写《详细设计说明书》

1. 详细设计工具

为了确定每个模块的算法，我们使用了描述算法的工具——程序流程图、N-S盒图以及PAD图。

2. 人机界面设计

人机界面应该具有一些特性：

- 可使用性
- 灵活性
- 可靠性

界面设计要遵守的一些原则：

- 用户界面适合于软件的功能
- 容易理解
- 风格一致
- 及时反馈信息
- 出错处理
- 适应各种用户
- 国际化
- 个性化
- 合理的布局
- 和谐的色彩

六、编码

程序设计语言的特性、程序的设计风格会深刻的影响软件的质量和可维护性。我们编码的目标是要产生正确可靠、简明清晰、具有较高效率的源程序。

七、测试

- 软件测试是为了证明程序有错，而不是证明程序无错误
- 一个好的测试用例是在于它能发现至今未发现的错误
- 一个成功的测试是发现了至今未发现的错误

1. 测试步骤

- 单元测试：集中对源代码实现的每一个程序模块进行测试
- 集成测试：把测试过的模块组装起来，主要对模块间接口开展测试
- 确认测试：检查已实现的软件是否满足了需求规格说明书中的各种需求
- 系统测试：把经过确认后的软件纳入实际运行环境中，与其他系统成分组合在一起测试。

2. 测试方法

> *Testing is the process of executing a program with the intent of finding errors.*

具体可以分为两大类测试方法

- 静态测试：主要进行代码复审，检查程序的静态结构，可以采用代码会审、走查、借助静态分析器进行自动化测试。
- 动态测试：在设定的测试用例上执行被测程序的过程。
    - 黑盒测试

- 白盒测试

白盒测试用例的设计

我们可以在程序流程图的基础上，使用逻辑覆盖法设计白盒测试用例。

对于逻辑覆盖法：

- 语句覆盖：将程序的每个语句至少执行一次
- 判定覆盖：每个判定的每个分支路径至少执行一次
- 条件覆盖：每个条件的真假两种情况至少执行一次

  > 满足条件覆盖不一定满足判定覆盖

- 判定/条件覆盖：每个条件的真假、每个判定的每个分支路径至少执行一次
- 条件组合覆盖：每个判定的所有条件的各种可能组合至少执行一次

我们也可以在程序同的基础上，使用路径测试法设计白盒测试用例。

对于路径测试法：

- 点覆盖：每个节点至少执行一次
- 边覆盖：每条边至少执行一次
- 路径覆盖：每条路径至少执行一次

黑盒测试用例设计

黑盒测试用例设计时，可以采取的方法有等价类划分、边界值分析、错误推测法。

3. 单元测试

4. 集成测试

5. 确认测试

6. 系统测试

> α测试：是用户在模拟操作环境下进行的测试，软件开发人员与*QA*人员也应参加
>
> β测试：是由软件的多个用户在实际操作环境下进行的测试。即是在开发者无法控制的环境下进行的软件现场应用。

7. 其他

除了上述测试外，测试还可以分为好多种类：

- 功能测试
- 可靠性测试
- 强度（压力）测试：高并发访问等
- 性能测试：通常与强度测试结合

  > 应该差不多吧

- 启动/停止测试
- 回复测试
- 配置测试
- 安全性测试
- 回归测试

8. 各种测试工具

> 体验了一下*NUnit*系列的*JUnit*与*CPPUnit*，感觉很棒

八、软件维护

九、面向对象软件工程

面向对象方法是以对象为核心的软件开发方法，按照人对客观世界认识的规律和解决问题的方法与过程开发软件，描述问题空间与实现在结构上一致。

UML是对象管理组织（OMG）采纳的基于面向对象技术的标准建模语言

> *Unified Modeling Language*——统一建模语言

> 面向对象 = 对象 + 类 + 继承 + 通信

1. UML模型基本组成

① 要素：Use Case、Actor、Class、Interface

② 关系：

- 关联关系：当一个类的对象作为另一个类的成员变量时，两个类之间有关联关系。

  > 聚合关系：整体与部分的关系，整体消失但对象不消失，部分的对象可以被多个整体共享
  >
  > > 球队与球员、电脑与*CPU*主存等
  >
  > 组合关系：整体与部分之间具有很强的所有关系和一致的生命周期，部分不能独立于整体而存在。
  >
  > > 活人与跳动的心脏

- 依赖关系：一个模型元素的变化会影响到另一个的模型元素
- 泛化关系：被称为继承关系，为通信而存在的关系
- 实现关系：类对接口的实现

③ 图：

- 静态图：类图、对象图、用例图
- 动态图：顺序图、状态图、协作图、活动图

2. 用例图

Use Case图用于描述拟建系统和外部环境的关系。

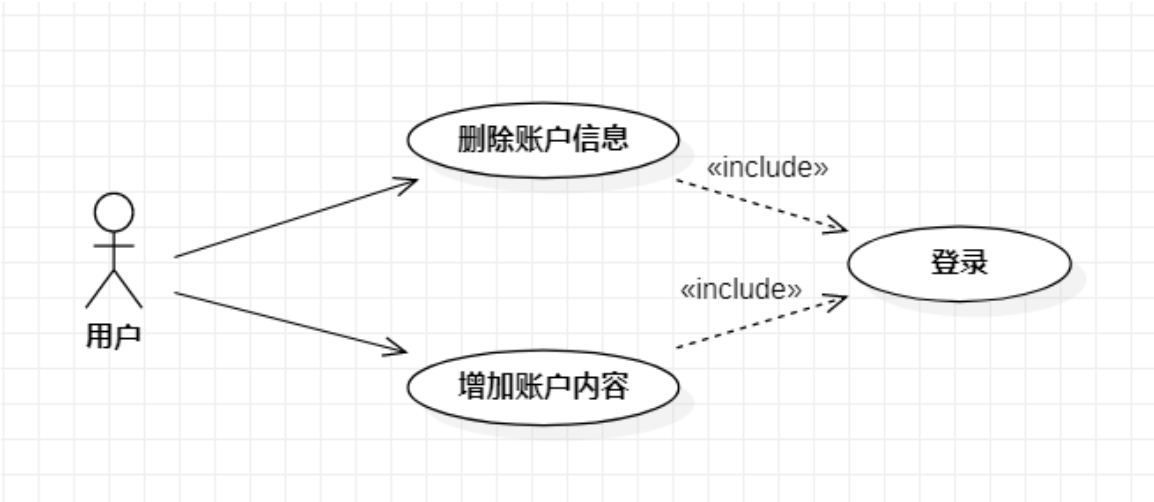
Actor

主导**Actor：** 初始化Use Case，主动要求得到结果，触发交互活动，至少有一个。

其他Actor：仅参与Use Case，在某个时刻与Use Case通信。

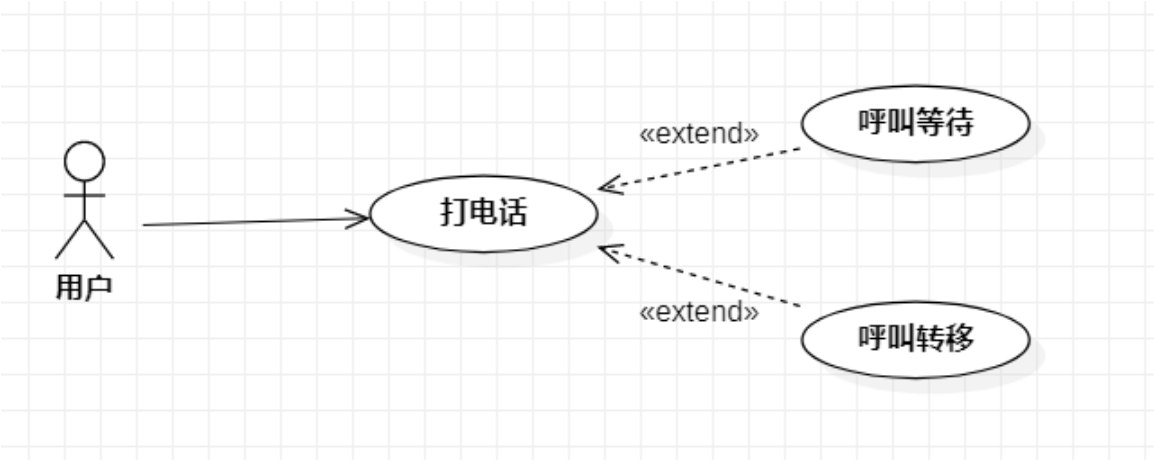
Use Case间的关系

- Include：包含关系



> 用户不论是对账户内容的增加还是删除，都需要先登录。表明删除*Use Case*与增加*Use Case*用例都*include*（包含）了登录*Use Case*

- Extend：用例的某一部分是可选的系统行为



> 用户打电话时，可以选择使用呼叫等待或是呼叫转移等增值服务，此时呼叫等待*Use Case*与呼叫转移*Use Case extend*了打电话*Use Case*的功能。

- Generalization

用例规约

> 用例图是骨架，用例规约是其内在的精髓

3. 类图

4. 序列图

- 对象
- 对象生存线
- 控制焦点
- 消息