软件工程导论
全书知识点梳理

第一章 概论

1.1 什么是软件工程？

软件工程是应用计算机科学、数学及管理科学等原理，开发软件的工程。软件工程借鉴传统工程的原则、方法，以提高质量、降低成本为目的。

1.2 软件工程生存周期分哪几个阶段？分别简述各个阶段的任务。

（1）计算机系统工程

计算机系统工程的任务是确定待开发软件的总体要求和范围，以及该软件与其他计算机系统元素之间的关系，进行成本估算，作出进度安排，并进行可行性分析。

（2）需求分析

需求分析主要解决待开发软件要“做什么”的问题，确定软件的功能、性能、数据、界面等要求，生成软件需求规约（也称软件需求规格说明）。

（3）设计

系统设计的任务是设计软件系统的体系结构，详细设计的任务是设计各个组成成分的实现细节，包括局部数据结构和算法。

（4）编码

编码阶段的任务是用某种程序设计语言，将设计的结果转换为可执行的程序代码。

（5）测试

测试阶段的任务是发现并纠正软件中的错误和缺陷。

（6）运行与维护

当发现了软件中潜在的错误或需要增加新的功能或使软件适应外界环境的变化等情况出现时，对软件进行修改。

1.3 简述 CMM 的 5 个等级。

1. 初始级
2. 可重复级
3. 已定义级
4. 已管理级
5. 优化级

1.4 简述各类软件过程模型的特点，如何选择合适的模型？（课件实例）

1. 瀑布模型：上一阶段的活动完成并经过评审才能开始下一阶段的活动，接受上一阶段活动的结果作为本阶段活动的输入，依据上一阶段活动的结果实施本阶段应完成的活动，对本阶段的活动进行评审。
2. 演化模型：从结构初始的原型出发，逐步将其演化成最终软件产品的过程。演化模型特别适用于对软件需求缺乏准确认识的情况。
3. 增量模型：将软件的开发过程分为若干个日程时间交错的线性序列，融合了瀑布模型的基本成分（重复地应用）和演化模型的迭代特征，特别适用于需求经常发生变化的软件开发。
4. 原型模型：开发人员和用户在“原型”上达成一致，缩短了开发周期，加快了工程进

度，降低成本。

5. 螺旋模型：将原型实现的迭代特征与瀑布模型中控制的和系统化的方面结合起来，不仅体现了这两种模型的优点，而且增加了风险分析。

6. 喷泉模型：各个阶段没有明显的界限，开发人员可以同步进行开发，可以提高软件项目开发效率，节省开发时间，适应于面向对象的软件开发过程。

7. 基于构件的开发模型：利用预先包装的构件来构造应用系统。

8. 形式化方法模型：易于发现需求的歧义性、不完整性和不一致性，易于对分析模型、设计模型和程序进行验证。

1.5 简述 CASE 工具和环境的重要性。

CASE 已被证明可以加快开发速度,提高应用软件生产率并保证应用软件的可靠品质。计算机专业人员利用计算机使他们的企业提高了效率,企业的各个部门通过使用计算机提高了生产率和效率,增强了企业的竞争力并使之带来了更多的利润。

习题

1、对于下列每一个过程模型,分别列举一个可以适用的具体软件项目,并说明在开发中如何应用该模型。

（1）某项目需要在一种新型机器上,为一种已知语言开发一个普通的编译器。

选用模型：瀑布模型

选用分析:由于该项目的语言是已知的,需求是明确的和稳定的,整个系统属于中小规模, 因此适合采用瀑布模型进行软件开发。

（2）某公司需要给火车站开发一个交互式火车车次查询系统,这是火车站首次使用该系统。

选用模型：快速原型模型

选用分析：本项目的主要问题在于用户需求方面，该系统与最终用户的交互是十分关键的，但是在项目的初期，因为是首次使用该系统，用户的需求基本上是不知道的，因此适合采用快速原型法来确定需求，在需求确定的基础上再开发最终的系统。

（3）某公司开发一个通用 CAD 软件产品，产品需求是逐步完善的，某些需求在一定范围内是明确的，某些需求需要进一步细化，但是迫于市场竞争的压力，产品需要尽快上市。

选用模型：增量模型

选用分析：通用 CAD 软件产品具有一定的成熟度，某些需求和软件系统结构是可以确定的。但是实现该产品所有功能需要比较长的开发周期,为了尽快上市可以采用增量模型实行多版本的发布策略，既可以很快占领市场又可以为后继版本的需求定义奠定基础。

（4）某公司开发企业管理 ERP 系统，包括销售、库存、生产、财务、物流、人力资源等部分，在系统实施过程中不同的企业具有一定的需求差异。

选用模型：基于构件的开发模型

选用分析：企业 ERP 系统具有构件化的结构，在不同企业实施时应该尽量重用已有的组件。因此适合采用基于构件的开发模型开发该系统，在直接应用或修改使用的基础上，最终进行组件开发和系统集成。

（5）某公司开发一个汽车防抱死刹车控制系统。

选用模型：形式化方法模型

选用分析：由于该系统对安全性和可靠性要求极高，需要在系统运行之前进行相关性能的检查，性能检查由复杂的数学运算实现，因此适合采用形式化方法开发该系统。

第二章 系统工程

2.1 简述系统工程的任务

1. 识别用户的要求

   识别用户对基于计算机的系统的总体要求，标识系统的功能和性能范围，确定系统的功能、性能、约束和接口。

2. 系统建模和模拟

   一个基于计算机的系统通常可考虑建立以下模型：硬件系统模型、软件系统模型、人机接口模型、数据模型。

3. 成本估算及进度安排

   开发一个基于计算机的系统需要一定的资金投入和时间约束（交互日期），需进行成本估算，并作出进度安排。

4. 可行性分析

   主要从经济、技术、法律等方面分析所给出的解决方案是否可行。

5. 生成系统规格说明

   作为以后开发基于计算机的系统的依据。

2.2 简述可行性分析的任务

1. 经济可行性
   a) 成本
   b) 效益
   c) 货币的时间价值
   d) 投资的回收期
   e) 纯收入
2. 技术可行性
   a) 风险分析
   b) 资源分析
   c) 技术分析
3. 法律可行性
4. 方法的选择和折衷

第三章 需求工程

3.1 需求工程具体包括哪些步骤？每个步骤的具体任务是什么？

1. 需求获取：系统分析人员通过与用户的交流、对现有系统的观察及对任务进行分析。
2. 需求分析与协商：分析每个需求与其他需求的关系以检查需求的一致性、重叠和遗漏的情况，并根据用户的需求对需求进行排序。
3. 系统建模：通过合适的工具和符号系统地描述需求。
4. 需求规约：给出对目标软件的各种需求。
5. 需求验证：对功能的正确性、完整性和清晰性以及其他需求给予评价。
6. 需求管理：对需求工程所有相关活动的规约和控制。

3.2 列出在制定需求获取策略时的 3 种主要考虑因素。

1. 功能需求。考虑系统要做什么，在何时做，在何时及如何修改或升级。
2. 性能需求。考虑软件开发的技术性指标。
3. 用户或人对因素。考虑用户的类型。

3.3 什么是非功能性需求？举例说明。

非功能性需求是指软件产品为满足用户业务需求而必须具有且除功能需求以外的特性。软件产品的非功能性需求包括系统的性能、可靠性、可维护性、可扩充性和对技术和对业务的适应性等。

例如在银行管理系统中，由于银行数据量的庞大以及对银行账户的管理需求，用户对系统的性能、可靠性、可维护性要求很高。安全性是对银行用户个人信息保密的基本要求；在使用系统时，由于用户庞大，要求能快速安全的执行要求，这就对系统的性能有高需求；银行的用户的变动比较大，需要高要求的系统维护。

3.4 需求获取的方法和策略有哪一些？举例说明。

- 建立顺畅的通信途径
- 访谈与调查
- 亲身实践
- 会议
- 头脑风暴
- 概念建模
- 原型、仿真
- 自省
- 用户行为数据在线采集

第四章 设计工程

4.1 简述软件设计阶段的基本任务。

1. 数据/类设计：将分析类模型变换成类的实现和软件实现所需要的数据结构。
2. 体系结构设计：定义了软件的整体结构，由软件部件、外部可见的属性和他们之间的关系组成。
3. 接口设计：描述了软件内部、软件和协作系统之间以及软件同人之间的通信方式。
4. 部件级设计：将软件体系结构的结构性元素变换为对软件部件的过过程性描述。

4.2 简述模块、模块化及模块化设计的概念。

模块是数据说明、可执行语句等程序对象的集合，是单独命名的，并且可以通过名字来访问的。模块化是指把软件按照规定原则，划分为一个个较小的，相互独立的但又相互关联的部件。模块化设计就是程序的编写不是开始就逐条录入计算机语句和指令，而是首先用主程序、子程序、子过程等框架把软件的主要结构和流程描述出来，并定义和调试好各个框架之间的输入、输出链接关系。

4.3 简述每种类型的模块耦合度和每种类型的模块内聚度。

耦合：

a) 内容耦合：当一个模块直接修改或操作另一个模块的数据,或者直接转入另一个模块时就发生了内容耦合。此时，被修改的模块完全依赖于修改它的模块。如果发生下列情形，两个模块之间就发生了内容耦合：
   i. 一个模块直接访问另一个模块的内部数据
   ii. 一个模块不通过正常入口转到另一模块内部
   iii. 两个模块有一部分程序代码重叠(只可能出现在汇编语言中)
   iv. 一个模块有多个入口。

b) 公共耦合：若一组模块都访问同一个公共数据环境，则它们之间的耦合就称为公共耦合。公共的数据环境可以是全局数据结构、共享的通信区、内存的公共覆盖区等。

c) 外部耦合：一组模块都访问同一全局简单变量而不是同一全局数据结构，而且不是通过参数表传递该全局变量的信息，则称之为外部耦合。

d) 控制耦合：如果一个模块通过传送开关、标志、名字等控制信息，明显地控制选择另一模块的功能，就是控制耦合。

e) 标记耦合：一组模块通过参数表传递记录信息，就是标记耦合。这个记录是某一数据结构的子结构，而不是简单变量。其实传递的是这个数据结构的地址。

f) 数据耦合：一个模块访问另一个模块时，彼此之间是通过简单数据参数(不是控制参数、公共数据结构或外部变量) 来交换输入、输出信息的。

g) 非直接耦合：两个模块之间没有直接关系，它们之间的联系完全是通过主模块的控制和调用来实现的

内聚：

1. 巧合内聚：讲几个模块中没有明确表现出独立功能的相同程序代码段独立出来建立的模块称巧合内聚模块。
2. 逻辑内聚：逻辑内聚是指完成一组逻辑相关任务的模块，调用该模块时，由传送给模块的控制性参数来确定该模块应执行哪一种功能。
3. 时间内聚：时间内聚是指一个模块中的所有任务必须在同一时间段内执行。
4. 过程内聚：过程内聚是指一个模块完成多个任务，这些任务必须指定的过程执行。
5. 通信内聚：通信内聚是指一个模块内所有处理元素都集中在某个数据结构的一块区域中。
6. 顺序内聚：顺序内聚是指一个模块完成多个功能，这些功能又必须顺序执行
7. 功能内聚：功能内聚是指一个模块中各个部分都是为完成一项具体功能而协同工作，紧密联系　不可分割。

4.4 描述信息隐蔽概念，并讨论信息隐蔽与模块独立两概念之间的关系。

1. 信息隐蔽指在设计和确定模块时，使得一个模块内包含信息(过程或数据)，对于不需要这些信息的其他模块来说,是不能访问的。在面向对象方法中，信息隐蔽是通过对象的封装性来实现的。
2. 信息隐蔽的概念与模块的独立性直接相关。

4.5 什么是模块的独立性？模块功能独立有何优点？如何度量独立性？

1、模块独立性：

A、模块独立性指每个模块只完成系统要求的独立的子功能,并且与其他模块的联系最少且接

口简单。

B、模块独立性是指模块内部各部分及模块间的关系的一种衡量标准，由内聚和耦合来度量。

2、优点：

A、具有独立的模块的软件比较容易开发出来。这是由于能够分割功能而且接口可以简化，当许多人分工合作开发同一个软件时，这个优点尤其重要。

B、独立的模块比较容易测试和维护。这是因为相对说来，修改设计和程序需要的工作量比较小，错误传播范围小，需要扩充功能时能够"插入"模块。总之，模块独立是优秀设计的关键，而设计又是决定软件质量的关键环节。

3、模块的独立程度可以由两个定性标准度量：内聚和耦合。

第五章 结构化分析与设计

5.1 简述数据流图的主要思想，概述使用数据流图进行需求分析的过程。

主要思想：数据流图描述输入数据流到输出数据流的变换（即加工），用于对系统的功能建模。

使用数据流图进行需求分析的过程：

1. 画出系统的输入和输出。
   A. 确定源和宿
   B. 确定加工
   C. 确定数据流
   D. 顶层图通常没有文件
2. 画出系统内部。
   A. 确定加工
   B. 确定数据流
   C. 确定文件
   D. 确定源和宿
3. 画出加工内部。
4. 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）。

5.2 分别采用数据流方法中的哪些技术来完成用户需求的精确化、一致化和完全化任务？

1. 父图和子图平衡。

父图和子图平衡是指任何一张 DFD 子图边界上的输入输出数据流必须与其父图中对应加工的输入输出数据流保持一致。

2. 数据守恒。

数据守恒包括以下两种情况：

（1） 一个加工所有输出数据流中的数据，必须能从该加工的输入数据流中直接获得，或者能通过该加工的处理而产生

（2） 加工未使用其输入数据流中的某些数据项。这表明这些未用到的数据项是多余的，可以从输入数据流中删去。（不一定是错误,但可能隐含潜在的错误）

3. 局部文件。

考虑分层数据流中一个文件应画在哪些 DFD 中，而不该画在哪些 DFD 中。

4. 一个加工的输出数据流不能与该加工的输入数据流同名。

同一个加工的输出数据流和输入数据流即使组成成份相同，仍应对它们取不同的名字，以表示它们是不同数据流，例如：例如，“报名单”和“合格报名单”。

允许一个加工有二个相同的数据流分别流向二个不同的加工。

5. 每个加工至少有一个输入数据流和一个输出数据流。
6. 在整套分层数据流中，每个文件应至少有一个加工读该文件，有另一个加工写该文件。
7. 分层数据流图中得每个数据流和文件都必须命名（除了流入或流出文件的数据流），并且与数据字典一致。
8. 分层 DFD 中的每个基本加工（即不再分解子图的加工）都应有一个加工规约。

5.3 在数据流图中，可否将两个加工用一个数据流相连？可否将两个源用一个数据流相连？为什么？

两个加工可以直接用数据流相连，两个源不能直接用数据流相连。因为数据流由一组固定成分的数据组成。在 DFD 中，数据流的流向可以有以下几种：从一个加工流向另一个加工，从加工流向文件（写文件），从文件流向加工（读文件），从源流向加工，从加工流向宿。

5.4 采用结构化分析方法绘制数据流图及数据字典（根据要求绘图并解决问题）。

5.5 描述基本加工的小说明（根据要求解决问题）。

5.6 结构图的基本成分包括什么？

- 模块
- 调用
- 数据

5.7 结构图的几个概念。

深度、宽度、扇入、扇出

第六章 面向数据结构的分析与设计

6.1 什么是面向数据结构的方法？

面向数据结构的方法是以数据结构为中心，从输入/输出的数据结构导出程序结构的一种软件需求分析与设计的方法。

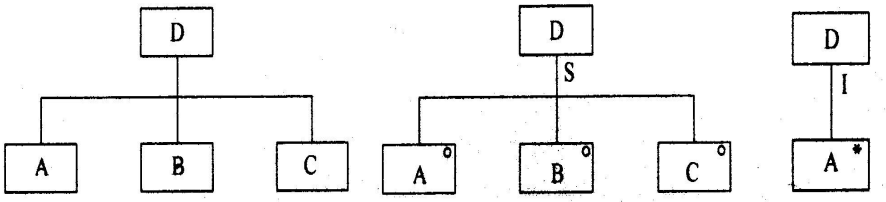
其特点如下：

1、以信息对象及其操作作为核心进行需求分析；

2、认为复合信息对象具有层次结构，并且可按顺序，选择，重复 3 种结构分解为成员对象信息；

3、提供由层次信息结构映射为程序结构的机制，从而为软件设计奠定良好的基础。

6.2 Jackson 图的三种结构。



三种元素类型：顺序元素、选择元素、重复元素。

第七章 面向对象方法基础

7.1 简述什么是 UML？

- UML 是一种 Language（语言）。
- UML 是一种 Modeling（建模）Language。
- UML 是 Unified（统一）Modeling Language。
- UML 是一种统一的、标准化的建模语言，UML 是一种应用面很广泛的建模语言。

7.2 UML 有哪些视图？有哪些图？（红笔为已学）

UML2.0 包含的视图和图：

- 静态视图（类图）
- 设计视图（构件图）
- 用况视图（用况图）
- 状态机视图（状态机图）
- 活动视图（活动图）
- 交互视图（顺序图）
- 部署视图（部署图）
- 模型管理视图（包图，包图是类图的一种变种）
- 剖面

7.3 UML 包括哪四种事物？

（1）结构事物：类、接口、用例、主动类、构件和结点等。
（2）动作事物：状态等。
（3）组织事物：包。
（4）注释事物：给建模者提供信息，提供关于任意信息的文本说明，但没有语义作用。

第八章 面向对象建模

8.1 用况图。

什么是用况图，用况图的组成元素（用况、执行者、用况之间的关系），根据要求绘制用况图。

8.2 类图。

什么是类图，类图的组成元素（类、类的属性和操作、类之间的关系、重数），根据要求绘制类图。

8.3 状态图。

什么是状态图，状态图的组成元素（状态、状态之间的迁移、分支、状态内的迁移、复合状态），根据要求绘制状态图。

8.4 活动图。

什么是活动图，活动图的组成元素（活动、泳道、分支、分岔和汇合、对象流），根据要求绘制活动图。

8.5 顺序图。

什么是顺序图，顺序图的组成元素（对象和生命线、激活、消息、组合片段），根据要求绘制顺序图。

第九章 基于构件的软件开发

8.1 什么是构件？

根据 pressman 书中的定义：构件是某系统中有价值的、几乎独立的并可替换的一个部分，它在良好定义的体系结构语境内满足某种清晰的功能。
根据 brown 的定义：构件是一个独立发布的功能部分，可以通过其接口访问它的服务。
根据《计算机科学技术百科全书（第二版）》中的定义：软件构件是软件系统中具有相对独立功能，可以明确标识，接口由规约指定，与语境有明显依赖关系，可独立部署，且多由第三方提供的可组装软件实体。软件构件须承载有用的功能，并遵循某种构件模型。可复用构件是指具有可复用价值的构件。

在基于构件的软件开发中经常会使用到的商用成品构件，是指由第三方开发的满足一定构件标准并且可组装的软件构件。

8.2 构件的可变性分析和可变性机制。

为了满足不同的复用需求，需要在构件复用时可能发生变化的一个或多个位置上标识变化点(variation point)，同时为变化点附加一个或多个变体(variant)。

8.3 简述基于构件的软件开发过程。（结合图 9.1）

基于构件的软件开发过程：
领域工程的步骤：
1 领域分析
2 建立领域特定的基准体系结构模型
3 标识候选构件
4 泛化和可变性分析
5 构件重构

6　构件的测试
7　构件的包装
8　构件入库

应用系统工程的步骤：

1　建立应用系统的体系结构模型；
2　寻找候选构件；
3　评价和选择合适的构件；
4　构件的修改和特化；
5　开发未被复用的不分；
6　构件的组装；
7　集成测试；
8　评价被复用的构件，并推荐可能的新构件。

第十章 敏捷软件开发

10.1 什么是敏捷开发方法？

敏捷开发是一种基于更紧密的团队协作、能够有效应对快速变化需求、快速交付高质量软件的迭代和增量的新型软件开发方法。

10.2 有哪些有代表性的敏捷开发方法？

以 Scrum 为基础的方法论（包括 Scrum、Scrum/XP 混合等）体系仍然居于主流地位，使用率最高。其他还有：看板方法、精益创业、极限编程等。