

统一过程模型的图、撰写用例规约、UML 用例图、UML 活动图、UML 泳道图、UML 状态图 (P140)、UML 顺序图 (P141)、UML 类图、

## 第一章

1. IEEE 定义: 软件工程是(1)将系统化、规范化、可量化的方法应用于软件的开发、运行和维护，即将工程化方法应用于软件；(2) 在(1)中所述方法的研究。
2. 软件与硬件的区别: 本质逻辑与物理；软件是设计开发的，并非传统意义上生产制造的；软件不会磨损；大部分软件是按需定制的。
3. 遗留软件的特点: 生命周期长、业务关键性、质量差

## 第二章

1. 软件工程与软件过程的区别: 软件过程是工作产品构建时所执行的一系列活动、动作和任务的集合。软件过程定义了软件工程化中采用的方法，但软件工程还包括该过程中应用的技术—技术方法和自动化工具。

2. 软件工程的三个要素: 过程、方法和工具。

软件工程的目标（根基）: 质量关注点。

3. 软件工程的通用过程框架定义了 5 个框架活动和 8 个普适性活动:

五种框架活动: 沟通、策划、建模、构建、部署。

    8 个普适性活动: 项目跟踪控制、风险管理、质量保证、配置管理、技术评审、测量、可复用管理、工作产品的准备和生产。

4. 课本 21 页软件过程框架图

每个框架活动由一系列软件工程动作构成；每个软件工程动作由任务集合来定义，这个任务集合明确了工作任务、工作产品、质量保证点、项目里程碑。

    （任务集的例子 P22、P23）

5. 过程流（P22 图） 描述了在执行顺序和执行时间上，如何组织框架中的活动、动作和任务。主要类型有：线性过程流、迭代过程流、演化过程流、并行过程流。

6. 过程模式模板（非重点） P24

7. 过程评估与改进: 以改进为目标，评估力求理解软件的当前状态。

用于过程改进的 CMMI 标准评估方法提供了五部的过程评估模型：启动、诊断、建立、执行、学习。

用于组织内部过程改进的 CMM 评估

8. 瀑布模型（经典生命周期）: 特点—文档驱动

优点: 消除非结构化软件；降低软件的复杂度，促进软件开发工程化

缺点: 实际项目开发中很少遵守瀑布模型提出的顺序；客户难以清楚的描述真正的需求；客户要等到开发周期的晚期才能看到程序运行的测试版本；在线性过程的开始和结束，容易发生“阻塞状态”

适用于: 需求确定、采用线性方式完成的工作中。瀑布模型适合于项目开发而不是产品开发。信息管理系统一般适合于用瀑布模型。因为这类系统业务功能较为明确，架构比较单一，技术难点较少。图书馆系统、销售管理系统都是。

瀑布模型图和 V 模型图 (P27)

9. 增量模型融合了瀑布模型的线性方式和原型模型的迭代特征。 P29 图

当使用增量模型时，第一个增量往往是核心的产品，也就是说第一个增量实现了基本的需求，但很多补充的特征还没有发布。客户对每一个增量的使用和评估，都做为下一个增量发布的新特征和功能，这个过程在每一个增量发布后不断重复，直到产生了最终的完善产品。增量模型强调每一个增量均发布一个可操作

的产品。

优点: (1) 人员分配灵活, 刚开始不用投入大量人力资源, 当核心产品很受欢迎时, 可增加人力实现下一个增量。

(2) 当配备的人员不能在设定的期限内完成产品时, 它提供了一种先推出核心产品的途径, 这样就可以先发布部分功能给客户, 对客户起到镇静剂的作用。

(3) 具有一定的市场。

(4) 可以规避技术风险。

缺点: 至始至终开发者和客户纠缠在一起, 直到完全版本出来。

增量模型主要适用于当需要快速推出可运行的版本, 而该版本不需要完整的功能时。

## 10. 原型开发模型 P29 图

缺点: (1) 没有考虑软件的整体质量和长期的可维护性。

(2) 大部分情况是不合适的操作算法被采用目的为了演示功能, 不合适的开发工具被采用仅仅为了它的方便, 还有不合适的操作系统被选择等等。

(3) 由于达不到质量要求产品可能被抛弃, 而采用新的模型重新设计。

优点: (1) 如果客户和开发者达成一致协议: 原型被建造仅为了定义需求, 之后就被抛弃或者部分抛弃, 那么这种模型很合适了。

(2) 迷惑客户抢占市场, 这是一个首选的模型。

客户和开发者必须在开始时达成一致: 原型被建造仅是为了定义需求, 之后就被抛弃了(或至少部分被抛弃), 实际的软件在充分考虑了质量和可维护性之后才被开发。

适用于: 需求不明确时。例如, 当客户有一个合理的需求, 但对细节没有任务线索时, 先开发一个原型。

## 11. 螺旋模型 P32 图

优点: 对于大型系统及软件的开发, 这种模型是一个很好的方法。开发者和客户能够较好地对待和理解每一个演化级别的风险。

缺点: (1) 需要相当的风险分析评估的专门技术, 且成功依赖于这种技术。

(2) 很明显一个大的没有被发现的风险问题, 将会导致问题的发生, 可能导致演化的方法失去控制。

(3) 对于要求固定预算的项目不适用。

特点: 风险驱动

## 12. 协同模型 (P33 图)

适合于不同的工程团队共同开发的系统工程项目。

## 13. 统一过程 (UP)

统一过程模型是一种“用例驱动、以体系结构(架构)为核心、迭代并且增量”的软件过程框架, 由 UML 方法和工具支持。

统一过程的五个阶段: 起始、细化、构建、转换、生产。(图 P38)

14. 软件过程的个人模型和团队模型, 都强调了成功软件过程的关键因素: 测量、策划和自我管理。

## 15. 知识结构

软件过程模型主要讲了惯用过程模型、专用过程模型、统一过程模型。

惯用过程模型包括: 瀑布模型、增量过程模型、演化过程模型(包括原型模型和螺旋模型)、协同模型。

其中, 演化过程模型具有迭代、递增特性, 其设计的目的是为了适应变更。

专用过程模型见 P42 小结第五段。

16. 软件生命周期: 软件计划与可行性研究、需求分析、软件设计、编码、软件测试、运行与维护。

### 第三章

1. 敏捷软件开发宣言:

个人和这些个人之间的交流胜过了开发过程和工具;

可运行的软件胜过了宽泛的文档;

客户合作胜过了合同谈判;

对变更的良好响应胜过了按部就班地遵循计划。

2. 敏捷方法最强制性的特点之一是它能够通过软件过程来降低有变更所引起的代价。

3. 普遍存在的变化是敏捷的基本动力。

4. 敏捷团队成员特点: 基本能力、共同目标、精诚合作、决策能力、模糊问题解决能力、相互信任和尊重、自我组织。

5. 敏捷理念的四个关键问题 P63 小结

6. 极限编程 (XP) 4 个框架活动: 策划、设计、编码和测试

策划: 用户故事: 一个用户通过系统完成它一个有价值的目标的事情。

设计原则: KIS 原则 (保持简洁原则)

重构是以不改变代码外部行为而改进其内部结构的方式来修改软件系统的过程。

编码:

结对编程: 两个人同时编写一段代码, 一般一个人负责实现, 一个人负责检查代码质量。

优点: 结对的两人完成其工作, 所开发代码和其他工作集成。

有些情况下, 这种集成工作由集成团队按日实施。

还有一些情况下, 结对者自己负责集成, 这种“连续集成”策略有助于避免兼容性和借口问题, 建立能及早发现错误的“冒烟测试”环境。

测试: XP 验收测试是由用户故事驱动的。

7. 其他敏捷过程模型: 自适应软件开发(ASD)、Scrum、动态系统开发方法(DSDM)、Crystal、特征驱动开发(FFD)、精益软件开发(LSD)、敏捷建模(AM)。

ASD 生命周期: 思考、协作、学习。

ASD 学习方式: 客户反馈意见、技术评审、事后剖析。

Scrum(P57 看一下)

FFD 也看一下 (P59)

(其他的概述都在 P63 小结 4、5 段)

## 第五章 理解需求

1. 需求工程是指致力于不断理解需求的大量任务和技术。从软件工程的角度来看, 需求工程是一个软件工程动作, 开始于沟通活动并持续到建模活动。

2. 需求工程的 7 个活动: 起始、导出、精化、协商、规格说明、确认、需求管理。

3. 起始阶段 手段: 询问问题

导出需求遇到的问题: 范围问题、理解问题、易变问题。

精化是由一系列的用户场景建模和求精任务驱动的。这一阶段形成需求模型(分析模型), 以说明软件的功能、特征和信息。由用户场景提取分析类、定义类的属性、确定类的操作、明确类之间的关联和协作关系, 完成各种补充图。

**协商：**当各方用户或客户的需求产生冲突时，让他们对各自的需求进行排序，按照需求的优先级讨论冲突。

规格说明的形式多种多样：文档、模型、形式化的数学模型、场景、原型。

**确认：**确认需求的评审小组包括软件工程师、客户、用户和其他利益相关者。使用分析模型保证需求说明的一致性。

**需求管理：**跟踪表。

4. 启动需求工程的步骤：确认利益相关者、识别多重观点、协同合作、首次提问。

(1) 常见的利益相关者有：业务运行管理人员、产品管理人员、市场销售人员、内部和外部客户、最终用户、顾问、产品工程师、软件工程师、支持和维护工程师及其他人员。

(2) 协同合作中需求工程师的工作是标识公共区域和矛盾区域。

5. 导出需求的步骤：协作收集需求、质量功能部署（QFD）、用户场景、导出工作产品。

(1) 协同需求收集会议的基本原则：a. 软件工程师和客户共同举办和参与。b. 制定筹备与参与会议的规则。c. 拟定一个会议议程：既涵盖重点，又鼓励自由交流。d. 由一个主持人控制会议。e. 使用某种“定义机制”：工作表、活动挂图、不干胶贴纸，电子公告牌、聊天室、虚拟论坛。

(2) QFD 是一种将客户要求转化成软件技术的质量管理技术。它确认了三种需求：正常需求、期望需求、令人兴奋的需求。

(3) 导出工作产品的类别（P93）

6. 开发用例

见 P95 用例模板

7. 构建需求模型（需求工程 7 个活动中的精化阶段）

(1) 需求模型的元素

基于场景的元素：UML 活动图、用例

基于类的元素：UML 类图、协作图

行为元素：UML 状态图、顺序图

面向数据流的元素：数据流图、数据模型

(2) 分析模式的优点：第一，分析模式提高了抽象分析模型的开发速度；第二，分析模式有利于把分析模型转变到设计模型。

8. 协商需求

协商的目标是开发一个现实可行的项目计划。

“双赢”意味着什么？

(1) 找到了双方赢的条件。

(2) 合适的折衷。（在满足利益相关者要求的同时，也可以保证软件团队成员按照实际情况、在可实现的预算和期限内完成工作。）

(3) 后续开展软件活动的关键。

## 第六章 第七章 需求建模：场景、信息与类分析、流程、行为

1. 需求分析要明确“做什么”而不是“怎么做”。

需求分析的三个目标：1) 描述客户需要什么；2) 为软件设计奠定基础；

3) 定义在软件完成后可以被确认的一组需求。

2. 分析模型在系统描述和设计模型之间建立桥梁。（P107 图）

3. 分析的经验原则 6 条（P107） 域分析的 5 个输入和 4 个输出（P108 图）

## 4. 需求建模的方法

### (1) 结构化分析方法

核心：算法和数据结构

关键字：数据对象建模、操作数据对象的处理建模

### (2) 面向对象的分析

目的：定义域问题相关的所有类

UML 和统一过程主要是面向对象的分析方法。

(3) 结构化分析与面向对象分析的本质区别。

答：结构化分析的核心是“处理”，而面向对象分析的核心是“对象 / 类”。前者以“计算”为核心，而后者以“结构”为核心。

## 5. 基于场景建模

撰写用例规约、UML 用例图、UML 活动图、UML 泳道图

## 6. 数据建模概念

数据对象是由计算机软件处理的任意符合信息的表示。

数据属性命名某数据对象，描述其特征，并在某些情况下引用另一个对象。

关系指明数据对象相互“链接”的方式。

## 7. 基于类的建模

(1) 联系 P121 的具体例子) 如何确定某个潜在类是否应该真的成为一个分析类(即分析模型中类的特征): 保留信息、所需服务、多个属性、公共属性、公共操作、必要操作。

(2) 描述属性: 属性实在问题的环境下完整定义类的数据对象集合。

(3) 定义操作: 操作可粗略分为四种类型: 1) 以某种方式操作数据(如添加、删除、选择等); 2) 执行计算操作; 3) 请求某个对象的状态的操作; 4) 监视某个对象发生某个控制事件的操作。

(4) 类-职责-协作者建模 (CRC): 明确类的职责

类的分类: 实体类: 表示系统存储和管理的永久信息; 描述必须存贮的信息及其相关行为; 通常对应现实世界中的“事物”

边界类: 表示参与者与系统之间的交互

控制类: 表示系统在运行过程中的业务控制逻辑

职责: 给类分配职责的 5 个指导原则 (P125)

协作: 类有一种或两种方法实现其职责: 1) 类可以使用自身的操作控制各自的属性, 从而实现特定的职责; 2) 一个类可以和其他类合作。

类之间的三种关系: 1) is-part-of (是……的一部分) 关系; 2) has-knowledge-of(有……的知识) 关系; 3) depends-upon (依赖……) 的关系。

(5) 关联和依赖 (P129 图)

关联定义了类之间的联系, 多样性定义了一个类和另一个类之间的联系数量关系。

(6) 分析包

分析包用来集合一组相关的类。

## 8. 面向流程建模

(1) 创建数据流模型: 数据流图 (DFD) 在某一层的输入和输出必须和精化后该层的输入输出相同。

(2) 创建控制流模型

(3) 控制规格说明 (CSPEC) 使用两种不同的方式表现系统的行为, 1) 一个状

态图，是行为的序列说明。2) 程序激活表，即行为的组合说明，或者说是当有事件发生时，会引入流程模型的那个处理。

#### (4) 处理规格说明 (PSPEC)

PSPEC 和用例是同一事物吗？如果不是，请解释区别。

答：不是。处理规格说明用于描述出现在求精过程中最终层次的所有流程模型的处理，通常是在详细设计的时候用到，是系统某个功能的具体实现方法。而用例描述了一个用户如何使用系统的，并不涉及到系统的内部的行为，通常在需求分析阶段用到。

### 9. 生成行为模型

#### (1) 行为模型：表现系统能够动态行为

(2) 生成行为模型的步骤：1) 评估所有的用例，以保证完全理解系统内的交互顺序；2) 识别驱动交互顺序的事件，并理解这些事件如何与特定的对象相互关联；3) 为每个用例生成序列；4) 创建系统状态图；5) 评审行为模型以验证准确性和一致性。

(3) 状态图可以表现系统执行其功能时每个类对象的状态。UML 状态图为每个类呈现了主动状态和导致这些主动状态变化的事件。

顺序图可以表现系统执行其功能时从外部观察到的系统状态。顺序图表明事件如何依法从一个对象到另一个对象的转移。

如何从状态图区分顺序图？它们有何相似之处？

答：状态图描述一个对象状态的变迁，而顺序图描述几个对象之间交互的顺序。对象状态的变迁，通常是由事件激发的，这个事件和顺序图当中的消息有关。可以由多个对象的状态图，组合成多个对象交互组成的序列图。

## 第九章 体系结构设计

1. 软件体系结构是指系统的一个或者多个结构，它包括软件构件、构件的外部可见属性以及它们之间的关系。

### 2. 软件体系结构为什么重要（3个原因）

(1) 软件体系结构的表示有助于对计算机系统开发感兴趣的各方开展交流。

(2) 体系结构突出了早期的设计决策，这些决策对于随后的软件工程工作有深远的影响，同时对系统作为一个可运行的实体的最后成功有重要作用。

(3) 体系结构“构建了一个相对小的、易于理解的模型，该模型描述了系统如何构成以及其构件如何一起工作”。

3. 软件体系结构风格定义：描述特定领域中软件系统家族的组织方式的惯用模式，反映了领域中众多系统所共有的结构和语义特性，并指导如何将各个模块和子系统有效地组织成一个完整的系统。

4. 体系结构风格的简单分类：以数据为中心的体系结构、数据流体系结构、调用和返回体系结构（主程序/子程序体系结构、远程调用体系结构）、面向对象体系结构、层次体系结构。

具体描述见 P178

5. 体系统结构权衡分析方法 (ATAM) 六个步骤 P185

## 第十章 构件级设计

1. 构建级设计的目的是把设计模型转化为运行软件。

2. 构件的定义：系统中模块化的、可部署的和可替换的部件，该部件封装了实现并暴露一组接口。

3. 关于什么是构件的三种观点：面向对象的观点、传统观点、过程相关观点  
以面向对象的观点来看，构建是协作类的集合。P197 图
3. 基于类的构件的基本设计原则：开闭原则、Liskov 替换原则、依赖倒置原则、接口分离原则、发布复用等价性原则、共同封装原则、共同复用原则。  
开闭原则 OCP：构件应该对外延具有开放性，对修改具有封闭性。P201 图  
Liskov 替换原则 LSP：子类可以替换它们的基类。LSP 原则要求源自于基类的任何子类必须遵守基类与使用该基类的构件之间的隐含约定。  
依赖倒置原则 DIP：上层模块不应该依赖于下层模块，二者都应该依赖于抽象；抽象不应该依赖于细节，细节应该依赖于抽象。  
接口分离原则 ISP：多个用户专用接口比一个通用接口好。  
发布复用等价性原则 REP：复用的粒度就是发布的粒度。  
共同封装原则 CCP：一同变更的类应该封装在一起。  
共同复用原则 CRP：不能一起复用的类不能分到一组。
4. 基于类的构件级设计的指导方针：构件、接口、依赖与继承（具体描述见 P203）
5. 基于类的构件级设计的内聚性：在面向对象系统进行构件级设计时，内聚性是指构件或者类只封装那些相互关联密切，以及与构件或类自身有密切关系的属性或者操作。  
一般来说，内聚性越高，构件的实现、测试和维护就越容易。  
内聚性的分类：功能内聚、分层内聚、通信内聚。  
功能内聚：通过操作来体现。当一个模块只完成某一组特定操作并返回结果时，就称此模块是功能内聚的。  
分层内聚：由包、构件和类来体现。高层能够访问低层的服务，而低层不能访问高层的服务。  
通信内聚：访问相同数据的所有操作被定义在一个类中。
6. 基于类的构件级设计的耦合性：耦合是类之间彼此联系程度的一种定性度量。  
在构件级设计中，尽可能保持低耦合是一个重要目标。  
耦合分类：内容耦合、共用耦合、控制耦合、标记耦合、数据耦合、例程调用耦合、类型使用耦合、包含或导入耦合、外部耦合。  
具体描述见 P205