# 目录

[目录 1](#_Toc60941443)

[第一章 3](#_Toc60941444)

[1. 软件的定义 3](#_Toc60941445)

[2. 软件危机的概念以及提出该概念的时间 3](#_Toc60941446)

[3. 软件危机产生的原因以及解决措施 3](#_Toc60941447)

[4. 软件工程的概念 3](#_Toc60941448)

[5. 软件工程的4个发展阶段 3](#_Toc60941449)

[6. 软件工程的知识体系· 3](#_Toc60941450)

[7. 软件工程三要素 4](#_Toc60941451)

[8. 软件生命周期的概念以及分为几个阶段，并各阶段的任务 4](#_Toc60941452)

[9. 传统的软件生命周期模型及各自特点（瀑布、原型和螺旋） 4](#_Toc60941453)

[10. 面向对象的生命周期模型有哪些（RUP(初始、细化、构造、交互)，构件复用）及各自特点 5](#_Toc60941454)

[11. 敏捷软件开发过程模型的价值观 5](#_Toc60941455)

[第二章 5](#_Toc60941456)

[1. 可行性分析的概念 5](#_Toc60941457)

[2. 可行性分析的三个重要方面 5](#_Toc60941458)

[3. 成本效益分析 5](#_Toc60941459)

[4. 系统流程图的意义及其绘制 6](#_Toc60941460)

[5. 会撰写可行性分析报告 6](#_Toc60941461)

[第三章 6](#_Toc60941462)

[1. 需求分析的概念和任务 6](#_Toc60941463)

[2. 软件需求包含哪些类别 6](#_Toc60941464)

[3. 结构化分析的概念、指导思想和基本原则 7](#_Toc60941465)

[4. 结构化分析常用的工具有哪些？各有什么作用和特点 7](#_Toc60941466)

[第四章 9](#_Toc60941467)

[1. 软件设计的阶段 9](#_Toc60941468)

[2. 总体设计阶段的主要任务是什么 9](#_Toc60941469)

[3. 常见的体系结构风格 9](#_Toc60941470)

[4. 模块化、信息隐蔽、分解、耦合、内聚 9](#_Toc60941471)

[5. 数据流的分类 9](#_Toc60941472)

[6. 结构化设计建模工具 10](#_Toc60941473)

[7. 详细设计过程的任务是什么 11](#_Toc60941474)

[8. 数据设计（数据结构、文件、数据库设计） 12](#_Toc60941475)

[9. 详细设计三大类工具（图形、表格和语言）（程序流程图、PAD图、盒图、过程设计语言、判定表和判定树） 12](#_Toc60941476)

[10. 结构化软件设计说明书 13](#_Toc60941477)

[第五章 13](#_Toc60941478)

[1. 对象及其三要素 13](#_Toc60941479)

[2. 面向对象的定义 14](#_Toc60941480)

[3. 对象、类、实例、封装、继承、多态 14](#_Toc60941481)

[4. 对象模型的五个层次以及OOA的五项建模活动 14](#_Toc60941482)

[5. 使用UML建模进行软件分析与设计：用例图、类图、时序图、部署图、构件图 14](#_Toc60941483)

[6. 面向对象的设计准则 16](#_Toc60941484)

[7. 面向对象分析与设计说明书 16](#_Toc60941485)

[第六章 16](#_Toc60941486)

[1. 软件实现的主要任务 16](#_Toc60941487)

[2. 软件实现的过程 16](#_Toc60941488)

[3. 软件实现的输入和输出 16](#_Toc60941489)

[4. 面向对象实现的准则 17](#_Toc60941490)

[5. 软件版本管理 17](#_Toc60941491)

[6. 软件编程语言的发展与分类 17](#_Toc60941492)

[7. 软件实现阶段有哪些文档 18](#_Toc60941493)

[第七章 18](#_Toc60941494)

[1. 软件测试的定义 18](#_Toc60941495)

[2. 软件测试的最主要任务 18](#_Toc60941496)

[3. 软件测试的步骤有哪些？每个步骤的主要内容和测试防范什么？ 18](#_Toc60941497)

[4. 软件测试方法主要分哪几种？软件测试方法主要分为哪几类？每类方法又分别有哪些测试技术方法？ 18](#_Toc60941498)

[5. 黑盒测试的分类（等价类、边界值等） 18](#_Toc60941499)

[6. 等价类和边界值测试用例的设计 19](#_Toc60941500)

[7. 白盒测试的语句覆盖、分支覆盖、条件覆盖等 20](#_Toc60941501)

[8. 软件调试的任务是什么？有哪两部分组成 20](#_Toc60941502)

[9. 软件调试的方法 20](#_Toc60941503)

[10. 软件维护的概念和分类（完善、适应、纠错、预防） 21](#_Toc60941504)

[11. 软件测试分析文档 21](#_Toc60941505)

[第八章 21](#_Toc60941506)

[1. 项目管理的五个过程、十大知识领域，重要的几个领域的含义 21](#_Toc60941507)

[2. 范围管理、成本（可行性分析部分讲过的投资收益比）和进度管理的内容 22](#_Toc60941508)

[3. 进度估算的方法（定额估算法、经验公式法、CPM等） 22](#_Toc60941509)

[4. 进度管理的图形工具（甘特、网络、里程碑、资源等） 22](#_Toc60941510)

[5. 常用的网络图（单、双）、关键路径的分析、工期的估算 23](#_Toc60941511)

[6. 质量管理、风险管理的概念 24](#_Toc60941512)

[7. 质量特性 24](#_Toc60941513)

[8. 风险管理（识别、评估、预防、监控） 24](#_Toc60941514)

[9. 熟悉利用PROJECT进行任务分解、计算工期、分配资源、查看成本 24](#_Toc60941515)

[软件项目 24](#_Toc60941516)

# 第一章

1. 软件的定义

软件=程序+数据+文档+服务

数据是信息的表达方式和载体，是使程序正常进行信息处理的结构及表示。

文档是与程序开发、维护及使用有关的技术数据和图文资料。

服务主要指对各种软件用户的服务，包括提供软件产品使用说明书、推销服务及售后技术支持

1. 软件危机的概念以及提出该概念的时间

软件危机的概念：在计算机软件的研发、运行、维护和管理的过程中，出现一系列严重问题的现象

提出该概念的时间：20世纪60年代

1. 软件危机产生的原因以及解决措施

原因：

1. 软件开发的规模、复杂度和需求量都在不断增加。
2. 软件（系统）需求分析与设计不完善、有欠缺，致使软件开发、维护和管理或文档出现问题。
3. 没有按照工程化方式运作，开发过程无统一标准、准则和规范方法。
4. 研发人员之间或同用户相互交流沟通不够或文档资料不完备。
5. 软件测试调试不规范、不细致，提交的软件质量。
6. 在软件运行过程中，忽视了正常有效的维护和管理。

措施：

1. 技术方法。运行软件工程的技术、方法和标准进行规范。
2. 开发工具。选用先进高效的软件工具，采用切实可行的实施策略。
3. 组织管理。研发机构需要组织高效、管理制度和标准严格规范、职责明确、质量保证有力、团结互助、齐心协力，注重文档及服务。
4. 软件工程的概念

软件工程=工程原理 + 技术方法 + 管理技术

1. 软件工程的4个发展阶段

传统软件工程、对象工程、过程工程、构建工程

1. 软件工程的知识体系·



1. 软件工程三要素

软件工程方法、软件工具、软件工程过程

1. 软件生命周期的概念以及分为几个阶段，并各阶段的任务

软件生命周期是从开始研发软件到软件停止使用的整个过程。

七个阶段

1. 开发策划：完成问题定义、可行性研究、项目申报立项和制订开发计划与准备，明确“要解决的问题是什么”
2. 需求分析：需求分析和定义阶段任务不是具体地解决问题，而是确定软件必须具备的具体功能、性能等，即“必须做什么”及其他指标要求。
3. 总体设计：也称概要设计，主要设计软件系统的总体结构，结构的组成模块，模块的层次结构、调用关系及功能，并设计总体数据结构等。
4. 详细设计：对模块功能、性能、可靠性、接口、界面、网络和数据库等进行具体设计的技术描述，并转化为过程描述。
5. 编写程序：又称为编码实现，将模块的控制结构转换成程序代码。
6. 测试：为了保证软件系统要求和质量标准，在设计测试用例的基础上对软件进行检测、调试、修改和完善等。
7. 运行维护：对交付并投入使用的软件进行维护，并记录保存文档。
8. 传统的软件生命周期模型及各自特点（瀑布、原型和螺旋）

瀑布模型的特点：开发过程的顺序性；统筹兼顾不过早编程；严格要求保证质量。

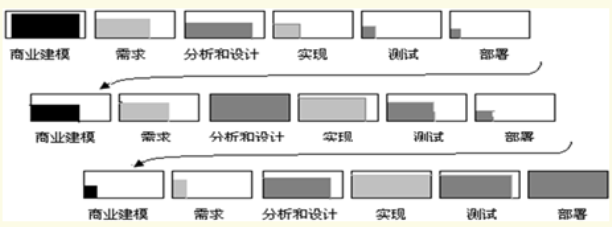
原型的特点：实际可行；具有最终系统的基本特征；构造方便、快速，造价低。

螺旋模型的特点：

1. 是一个演化软件过程模型，将原型实现的迭代特征与瀑布模型的线性顺序模型中控制的和系统化的方面结合起来；
2. 开发过程具有周期性重复的螺旋线状。强调了风险分析；
3. 采用循环的方式逐步深入；
4. 确定系列里程碑，确保客户满意
5. 面向对象的生命周期模型有哪些（RUP(初始、细化、构造、交互)，构件复用）及各自特点

统一开发过程RUP：用例驱动，以构架为中心，采用代和增量的开发策略

* 1. RUP的二维开发模型及其核心工作流。主要包括：商业建模、需求、分析与设计、实现、测试、核心支持工作流、部署、配置和变更管理、项目管理和环境。
  2. 开发中各阶段和里程碑，主要包括。初始阶段、细化阶段、构造阶段、交付阶段。
  3. RUP的迭代开发模式。如下图所示



1. 敏捷软件开发过程模型的价值观
2. 个体和交互胜过过程和工具
3. 可以工作的软件胜过面面俱到的文档
4. 客户合作胜过合同谈判
5. 响应变化胜过遵循计划

# 第二章

1. 可行性分析的概念

可行性分析也称可行性研究，是对拟研发软件项目(或称为申报的“拟研发立项问题”)分析论证可行性和必要性的过程。主要从技术、经济、社会等方面分析其可行性，并根据软件运行环境、软硬件及数据资源与处理要求、研发能力和效益等情况，确定立项开发的必要性，并在确定可行必要后提出初步方案，形成“可行性分析报告”，之后还需要进行立项并制定出研发计划，以便于进行有效研发。

1. 可行性分析的三个重要方面

技术可行性、经济可行性、社会可行性

1. 成本效益分析
2. 投入产出比。是指软件项目全部投资与产出增加值总和之比。R=K/IN(K为投资总额，IN为软件生存期内各年增加值的总和)
3. 货币的时间价值。由于利率的变化等因素，货币的时间价值能较准确地估算。假设年利率为i，若项目开发所需经费即投资为P元，则n年后可得资金数为F元：F=P\*(1+i)n；反之，若n年后可得效益为F元，则这些资金现在的价值为：P=F/(1+i)n
4. 系统流程图的意义及其绘制
5. 对于软件相关的具体主要物理系统的实际描述和表示
6. 全面了解系统业务处理过程和进一步分析系统结构的依据
7. 系统分析员、管理人员、业务操作人员相互交流确认的工具
8. 可直接在系统流程图上，拟出软件系统可实现处理的主要部分
9. 可利用系统流程图分析业务流程及其合理性。



1. 会撰写可行性分析报告

# 第三章

1. 需求分析的概念和任务

软件需求分析通常是软件需求分析师经过深入细致的调研和分析，准确理解用户需求和项目的功能、性能、可靠性等具体要求，将用户非形式化的需求表述转化为完整的具体需求定义，再将需求定义写成需求规划的过程。

任务：

1. 确定目标系统的具体要求
2. 建立目标系统的逻辑模型
3. 编写需求文档，验证确认需求
4. 软件需求包含哪些类别

软件需求分为三个层次：业务需求、用户需求、功能需求，此外还包括某些类别的非功能需求

1. 结构化分析的概念、指导思想和基本原则

结构化分析(Structured Analysis，SA)方法是面向数据流的需求分析方法，SA方法根据软件内部的数据传递、变换关系，自顶向下、逐层分解，绘出满足功能要求的模型。

结构化方法总的指导思想是“自顶向下、逐步求精”

其基本原则是抽象与分解。

1. 结构化分析常用的工具有哪些？各有什么作用和特点
2. 实体关系图(E-R图)是目前最常用的数据建模方法。主要用于在需求分析阶段清晰地表达目标系统中数据之间的联系及其组织方式，建立系统的实体数据模型(E-R模型)。实体模型由实体、联系和属性三个基本成分组成
3. 数据流图(DFD)是一种图形化的系统模型。运用图形方式描述系统内部的数据流程，表达系统的各处理环节之间的数据联系，是结构化系统分析方法的主要表达工具。DFD是结构化分析的最基本工具。
4. 数据词典(DD)是指存储数据源定义和属性(描述说明)文档，是数据描述的重要组成部分。数据字典有4类条目：数据流、数据项、文件及基本加工。在定义数据流或文件时，使用下表给出的符号。将这些条目按照一定的规则进行组织，构成数据字典。



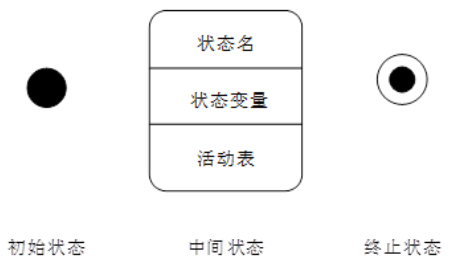
数据字典要求：完整性、一致性和可用性

1. 数据流条目。给出了DFD中数据流的定义，通常对数据流的简单描述为列出该数据流的各组成数据项。主要包括：数据流名称、别名及简述、数据流来源和去处、数据流组成、流通量。
2. 文件条目。给出某个文件的定义，文件的定义通常是列出文件记录的组成数据流，还可指出文件的组织方式。
3. 数据项条目。给出某个数据单项的定义，通常是该数据项的值类型、允许值等
4. 加工条目。是对DFD的补充，实际是“加工小说明”。由于“加工”是DFD的重要组成部分，一般应单独进行说明。因此，数据字典是对DFD中所包含各种元素定义(数据项、结构、特征、加工、流向等)的集合。对4类条目：描述数据流、数据项、文件及基本加工
5. 处理过程描述
6. 结构化语言是一种介于自然语言(英语或汉语)和形式语言之间的半形式化语言。专门用于描述一个功能单元逻辑要求。结构化语言有三种结构形式
7. 顺序结构。以一组祈使语句、选择语句、重复语句的顺序排列
8. 选择结构。常用IF-THEN( -ELASE)-ENDIF或 CASE-OF-ENDCASE等关键词构成的语句结构
9. 循环结构。常用 DO-WHILE-ENDDO或REPEAT-UNTIL等关键词的语句结构
10. 判定树(Decision Tree)也称为判断树或决策树，用判定树描述一个功能模块逻辑处理过程，其基本思路与结构化语言完全类似，是结构化语言的另一种更为直观方便的逻辑表现形式。

特点：描述一般组合条件较清晰。缺点：不易输入计算机。

1. 判定表也称为决策表，与结构化语言和判定树方法相比其优点是能够将所有的条件组合充分地表达出来。缺点是判定表的建立过程较为繁杂，且表达方式不如前两者简便。判定表由4个部分组成： 条件定义、条件取值及组合、结果(操作定义)、取值组合结果(操作)
2. 状态转换图：在对系统进行需求分析时，除建立系统的数据模型和功能模型外，有时还需要建立系统的行为模型。

状态：状态代表系统的一种行为模式。在状态转换图中，状态又可以分为：初始状态、终态、结束状态和中间状态。三类状态的表示符号如下图所示



初始状态：简称初态，系统启动时进入的状态。

最终状态：简称终态，系统运行结束时到达的状态。

状态迁移：一个状态向另一个状态转换。用箭头表示从种状态向另一种状态的迁移。

事件：是在某个特定时刻发生的事情，它是对引起系统做动作或(和)从一个状态转换到另一个状态的外界事件的抽象。

事件的语法格式为：事件说明[条件] / 动作表达式

1. 结构化分析建模
2. 功能建模（绘制数据流图，建造数据字典，数据流图的四种基本符号：数据源（终点）、数据加工、数据存储、数据流）
3. 数据建模（E-R）
4. 行为建模（状态图）
5. 数据字典

# 第四章

1. 软件设计的阶段

总体(概要)设计、详细设计

1. 总体设计阶段的主要任务是什么

软件总体设计的任务主要包括：

1. 软件的总体结构和模块外部设计。——软件结构图
2. 软件处理流程设计。——模块、子模块及数据调用关系
3. 确定软件的功能并分配(与程序结构的关系)。
4. 数据总体结构设计(逻辑-物理-数据结构与程序的关系)
5. 网络及接口概要设计。
6. 运行概要设计。
7. 出错处理概要设计(
8. 性能可靠性及安全保密概要设计
9. 维护概要设计
10. 常见的体系结构风格

管道/过滤器、层次系统、虚拟机

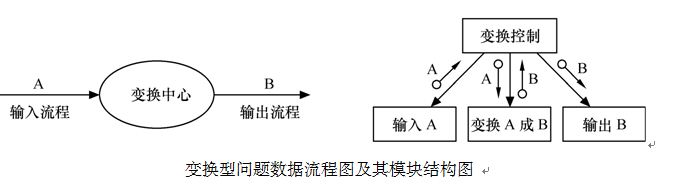
1. 模块化、信息隐蔽、分解、耦合、内聚
2. 软件工程模块化：模块是模块是指一个独立命名的，拥有明确定义的输入/输出和特性的程序实体。模块化(Modular)是将复杂软件划分为功能相对独立且易于处理的模块的过程，是概要设计主要的工作之一。
3. 信息隐蔽：指模块所包含的“过程及数据”信息，对于其他模块需要隐蔽。模块规定和设计应遵从：使包含在模块中的“过程或数据”信息，对于其它不需要这些信息的模块，不能访问或“不可见”。
4. 分解：子系统及模块划分（除了高内聚、低耦合）还应考虑：（1）模块大小适当。（2）模块的层次结构。
5. 内聚。标志一个模块内各个元素彼此结合的紧密程度，是信息隐蔽和局部化概念的自然扩展。设计时应该力求做到高内聚
6. 耦合。是对软件结构不同模块之间互连程度的度量，是影响软件复杂程度的一个重要因素。模块间的耦合程度将影响系统的可理解性、可测试性、可靠性和可维护性(力求做到低耦合)
7. 数据流的分类
8. 中心变换型(transform center)。如下图所示，该类图的特点是：DFD 图可以明显分为“输入—处理(变换)—输出”三部分，对这种类型的DFD 图的转换采用变换分析技术。



1. 事务处理型(transaction)。事务是指完成作业要求功能处理的最小单元。事务型数据流图如图4-9所示，输入流在经过某个“事务中心”时接收输入数据并分析确定其类型，然后根据所确定的类型为数据选择其中的一条加工路径。



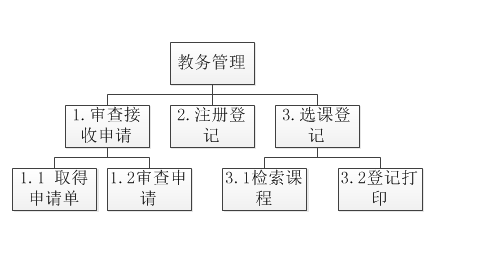
1. 结构化设计建模工具
2. 模块结构图（中心变换型）



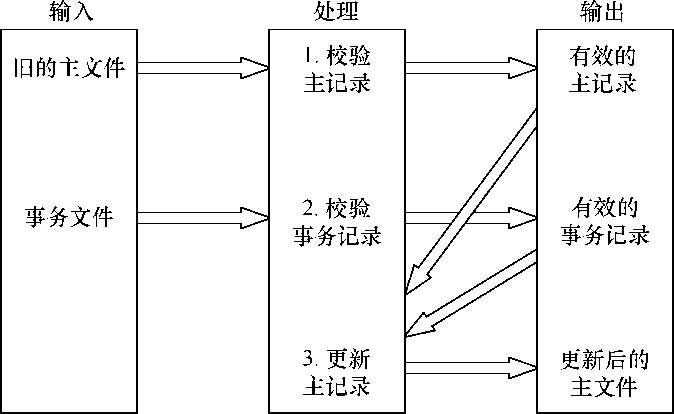
1. 事务型软件结构图



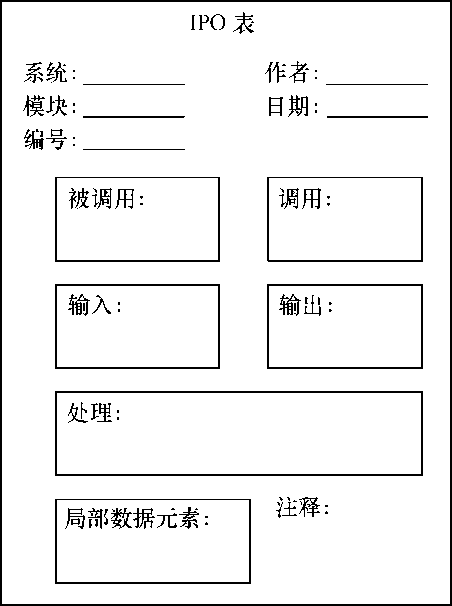
1. HIPO图
2. H图——层次图



1. IPO图——输入-输出-处理图



1. HIPO图——改进的IPO图（IPO表）的形式



1. 详细设计过程的任务是什么
2. 模块功能及(处理)算法设计。
3. 模块内的数据结构设计。
4. 模块接口设计。
5. 其他设计（BD、网络、代码、接口、界面等）。
6. 模块测试用例设计。
7. 编写详细设计文档。
8. 详细设计评审。
9. 数据设计（数据结构、文件、数据库设计）

数据库设计的任务及步骤为数据库应用程序（处理过程）和数据库（数据结构与数据表及数据）的需求分析、概念结构设计、逻辑结构设计、物理结构设计等阶段。

1. 需求分析。可以在软件需求分析阶段完成，也可以进一步深入企事业实际进行具体的调研分析，主要任务是调查、深入分析和确认业务用户数据需求与处理需求。
2. 概念结构设计。目的是获取数据库的概念数据模型，通常有两种方法；一是设计实体联系模型（E-R图），二是面向对象的方法，以类或对象形式表示数据及其之间联系。
3. 逻辑结构设计。主要任务是将概念模型转化为终端可以实现的传统数据模型，如转化为关系模型，需要根据所选数据库得到具体的关系数据模式，即二维表结构。
4. 物理结构设计。主要是根据数据模型及处理要求，选择存储结构和存取方法，以求获得最佳的存取效率。主要包括：数据库文件组织形式（顺序文件或随机文件）、索引文件组织结构、存储介质的分配、存取系统的选择等。
5. 详细设计三大类工具（图形、表格和语言）（程序流程图、PAD图、盒图、过程设计语言、判定表和判定树）

详细设计常用工具包括以下三种:图形、表格、语言.

1. 图形工具是将过程细节用图形进行表示， 如流程图、盒图、问题分析图(PAD)等。
2. 程序流程图：程序流程图也称为程序框图，是常用的一种算法表达工具。

优点：直观清晰，易学习掌握。

缺点：从发展趋势看，使用的人逐渐减少。不是求精的好工具，不可以表示数据结构，使程序员过早考虑程序控制细节，不利于理解大型程序。



1. 结构流程图：软件的结构流程图也称为盒图或N-S 图。表示软件的控制结构如图4-14所示。

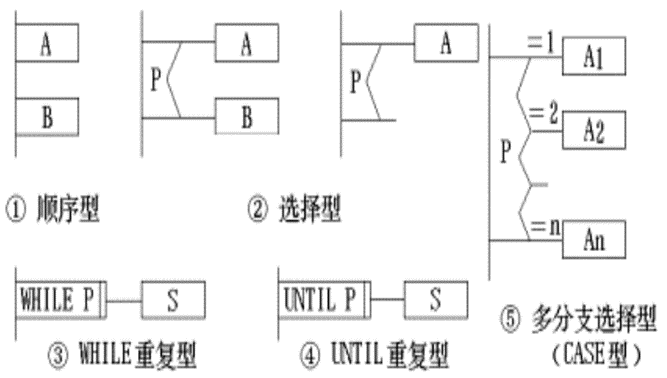
优点:图中不能随意转移控制（无流程线），程序结构更符合结构化程序设计的思想，有利于培养软件设计人员良好的设计风格。

缺点:程序嵌套层次较多时影响可读性而且不易修改。



1. 问题分析图(PAD)：问题分析图（PAD）采用了二维树形结构的图形符号表示程序的控制结构。

优点:程序结构层次清晰；支持自顶向下、逐步求精的设计过程；既可描述程序的逻辑结构，又能描述系统的数据结构；遵循机械的走树(Ttee Walk)规则，容易转换为高级语言源程序；易读、易懂、易记。



1. 表格工具。用于将处理过程细节用表格形式表示，如决策表（判定表）等。
2. 语言工具。主要用于将处理过程细节用语言形式表示，如结构化语言等。此外，还可以使用过程设计语言PDL。
3. 结构化软件设计说明书

# 第五章

1. 对象及其三要素

对象是描述客观事物的实体，是构成系统的基本单位。

对象具有三要素：对象标识、属性和服务。

1. 对象标识为对象的名字，用于唯一地识别系统内部对象，在定义或使用对象时指定。
2. 属性也称状态或数据， 用于描述对象的静态特征。在某些OOP（Object Oriented Programming面向对象编程）语言中，属性通常被称为成员变量或简称变量。
3. 服务也称操作、行为或方法， 用于描述对象的动态特征，在某些OOP语言中，通常被称为成员函数或简称函数。
4. 面向对象的定义

面向对象(Object Oriented，OO) 定义：面向对象= 对象 + 类 + 继承 + 消息通信，具有这4个概念的软件开发方法称为OOM。

面向对象方法（OOM）是面向对象技术和方法在软件工程中的全面运用，包括面向对象分析（OOA）、面向对象设计(OOD)、面向对象编程（OOP)、面向对象测试(OOT)和面向对象维护等方法

1. 对象、类、实例、封装、继承、多态
2. 封装（Encapsulation）有两层含义：一是对象是其全部属性和全部服务紧密结合而形成的一个不可分割的整体；二是对象如同一个密封的“黑盒子”，表示对象状态的数据和实现操作的代码都被封装在其中。封装是对象的一个重要特性。在面向对象的系统中，对象是一个封装数据属性和操作行为的实体。
3. 类（Class）也称对象类（Object Class）是对具有相同属性和服务的一组对象的抽象定义。类与对象是抽象描述与具体实例的关系，一个具体的对象被称为类的一个实例（Instance）。
4. 继承（Inheritance）是父类和子类之间共享数据结构和方法的一种机制，是以现存的定义的内容为基础，建立新定义内容的技术，是类之间的一种关系。继承有两种：单重继承、多重继承。

单重继承构成的类之间的层次关系为一树状，若将所有无子类的类，都看成还有一个公共子类， 多重继承构成的类之间的关系为一个网格， 而且继承关系可传递。

建立继承结构的优点有3个：一是易编程、易理解且代码短， 结构清晰；二是易修改，共同部分只在一处修改即可；三是易增加新类，只须描述不同部分。

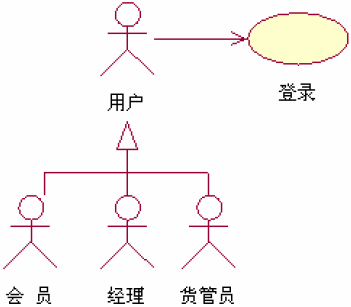
1. 多态性（Polymorphism）是指多种类型的对象在相同的操作或函数、过程中取得不同结果的特性。多态有多种不同形式，其中参数多态和包含多态统称为通用多态，过载多态和强制多态则统称为特定多态。

动态绑定(Dynamic-binding)是多态性的基石之一。将函数调用与目标代码块的连接延迟到运行时进行，只有发送消息时才与接收消息实例的一个操作绑定。

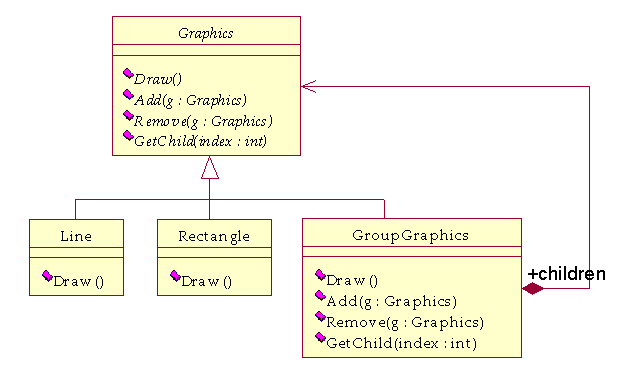
1. 对象模型的五个层次以及OOA的五项建模活动

对象模型常由五个层次组成：类与对象层、属性层、服务层、结构层和主题层，其层次对应着OOA过程中建立对象模型的五项主要活动：发现类和对象、定义属性、定义服务、划分结构、识别主题。

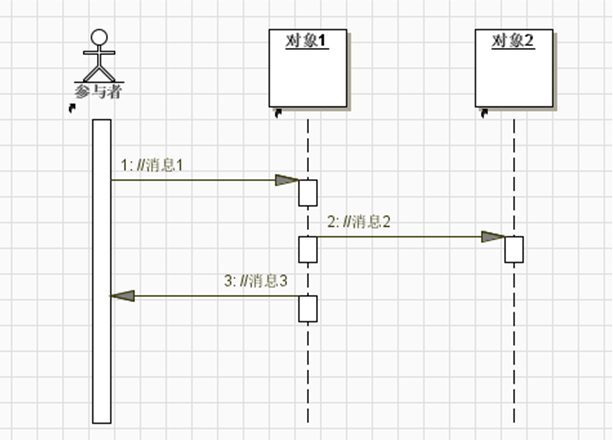
1. 使用UML建模进行软件分析与设计：用例图、类图、时序图、部署图、构件图
2. 用例图是从用户角度描述系统功能， 是用户所能观察到的系统功能的模型图，用例是系统中的一个功能单元



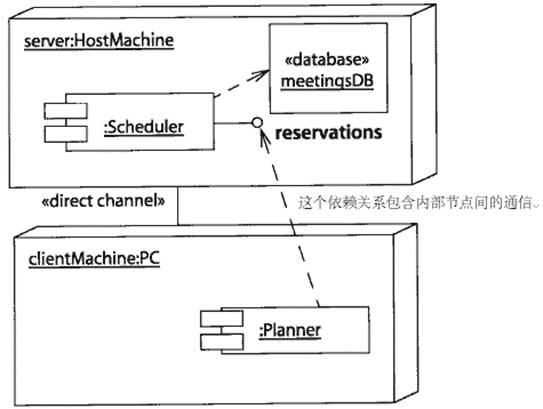
1. 类图描述系统中类的静态结构。不仅定义系统中的类，表示类之间的联系如关联、依赖、聚合等，也包括类的内部结构(类的属性和操作)



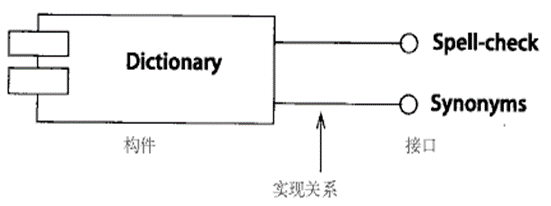
1. 顺序图显示对象之间的动态合作关系，它强调对象之间消息发送的顺序，同时显示对象之间的交互



1. 部署视图描述位于节点实例上的运行构件实例的安排。



1. 构件图为系统的构件建模型—构件即构造应用的软件单元—还包括各构件之间的依赖关系，以便通过这些依赖关系来估计对系统构件的修改给系统可能带来的影响



1. 面向对象的设计准则

OOD准则包括5个方面：

1. 抽象。强调实体本质内在属性，忽略无关紧要属性。
2. 信息隐蔽。
3. 高内聚。
4. 低耦合。
5. 可重用。
6. 面向对象分析与设计说明书

# 第六章

1. 软件实现的主要任务

软件实现主要任务:将软件详细设计的结果转换为目标软件。是对“详细设计”的工作进行具体实现，形成计算机/手机可运行程序。

1. 软件实现的过程

软件实现的过程包括编程和单元测试

1. 软件实现的输入和输出

软件实现的输入:软件详细设计说明书

软件实现的输出:源程序、目标程序和用户手册.

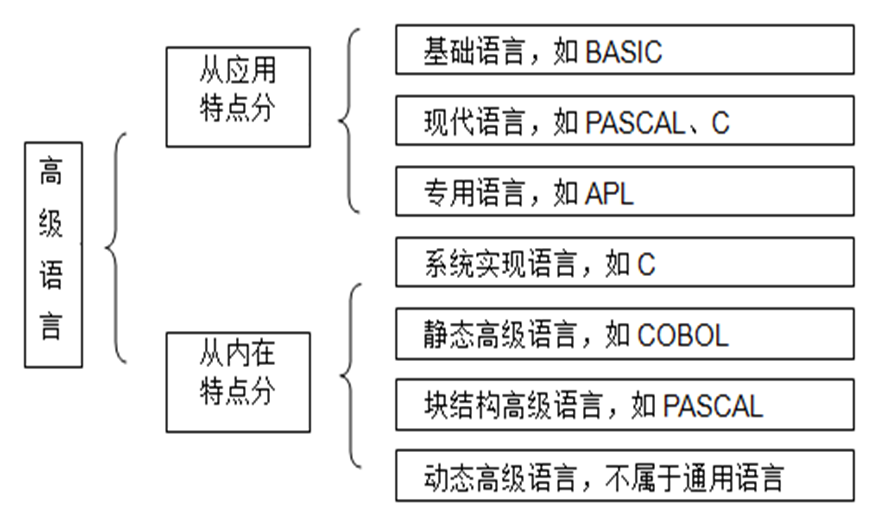
1. 面向对象实现的准则

实现的准则主要包括：

1. 高可重用性。
2. 高可扩充性。
3. 高可靠性及健壮性。
4. 软件版本管理

软件一般具有中间版本、α版本、β版本、发布版本和维护版本等，以不同标号标识。

1. 中间版本：软件未完成前且未进行正式测试的版本。一个软件可能会有多个中间版本，记录不同时间、不同人员开发的软件程序。
2. α版本：软件未完成但可以提交进行严格测试的软件版本。生存周期通常只有几天或几周。
3. β版本：能够稳定运行的软件，可交付给终端用户进行测试。是经过多个α版本的完整测试，确认没有问题后产生的，β版本提交给直接用户在实际工作中进行测试和检验，一般需要几个月的测试时间。
4. 发布版本：正式向社会发布或向用户提交使用的软件产品。
5. 维护版本：对发布的版本进行错误纠正，及功能或性能改进的版本。
6. 软件编程语言的发展与分类
7. 第一代语言：机器语言无需编译和转换就能被 CPU 直接使用。
8. 第二代语言：汇编语言。
9. 第三代语言：具有直接支持结构化构件，并具有很强的过程能力和数据结构能力特点的结构化与现代程序语言。分为三类：通用高级语言、面向对象的语言和专用语言。结构化程序语言Turbo C，面向对象的语言有C++、Java、Delphi等。
10. 第四代语言：超高级编程语言(4GL)， 不再涉及很多算法性细节.具有3个特征:强大数据管理能力，可对数据库有效存取、查询等操作;提供高效、非过程化命令，组成语言基本语句;可满足多功能一体化要求 。比如数据库查询语言、程序生成器等。
11. 分类



1. 软件实现阶段有哪些文档

软件实现文档包括：软件产品规格说明(SPS)、计算机编程手册(CPM)和软件用户手册(SUM)。

# 第七章

1. 软件测试的定义

软件测试（Software test）的定义是：在规定的条件下对软件进行检测性运行操作，以发现程序错误，衡量软件质量，并对其是否能满足设计要求进行评估的过程。

1. 软件测试的最主要任务

软件测试重点是测试软件的功能、性能和可靠性等是否符合用户需求指标，其中，“功能测试”是软件测试的最主要任务。

1. 软件测试的步骤有哪些？每个步骤的主要内容和测试防范什么？

通常具体的软件测试分为单元测试、集成测试、有效性（确认）测试和系统测试4个步骤，最后进行验收测试

1. 软件单元测试的主要内容包括：单元模块内和模块之间的功能测试、容错测试、边界测试、约束测试、界面测试、重要的执行路径测试，单元内的业务流程和数据流程等。
2. 软件集成测试的主要内容包括：系统集成后的功能测试、业务流程测试、界面测试、重要的执行路径测试、容错测试、边界测试、约束测试及接口测试等。
3. 有效性测试的主要内容包括：系统性的初始化测试、功能测试、用户需求确认、业务处理或数据处理测试、性能测试、安全性测试、安装性测试、恢复测试、压力测试等。
4. 系统测试是对整个程序系统及人工过程与环境的总测试，目标主要是发现并纠正软件开发过程中所产生的错误。
5. 软件测试方法主要分哪几种？软件测试方法主要分为哪几类？每类方法又分别有哪些测试技术方法？

主要可分为黑盒测试和白盒测试两类，黑盒测试是功能测试，白盒测试是逻辑测试。

黑盒测试：等价分类法、边界值法、因果图法、错误判断法、决策表法

白盒测试：语句覆盖、判定覆盖、条件覆盖、判定/条件覆盖、条件组合覆盖、路径覆盖

1. 黑盒测试的分类（等价类、边界值等）
2. 等价分类法
3. 划分等价类。根据输入条件将输入数据划分为等价类。
4. 设计测试用例。
5. 边界值分析法

边界值分析法主要包括6个方面：

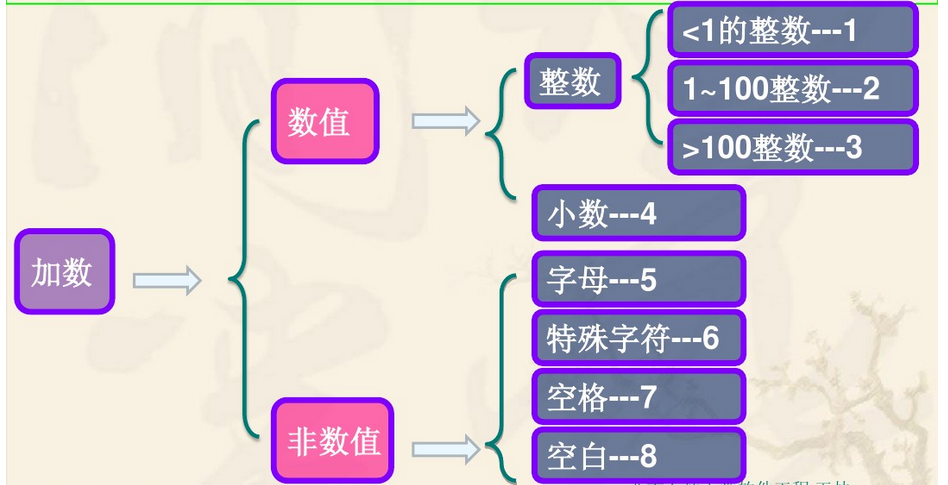
1. 测试数据。
2. 输出条件。
3. 输入边界数据。
4. 有序集测试用例。
5. 内部数据结构边界值。
6. 其他边界条件。
7. 等价类和边界值测试用例的设计

例:计算1~100（含1和100）的两个整数的和。

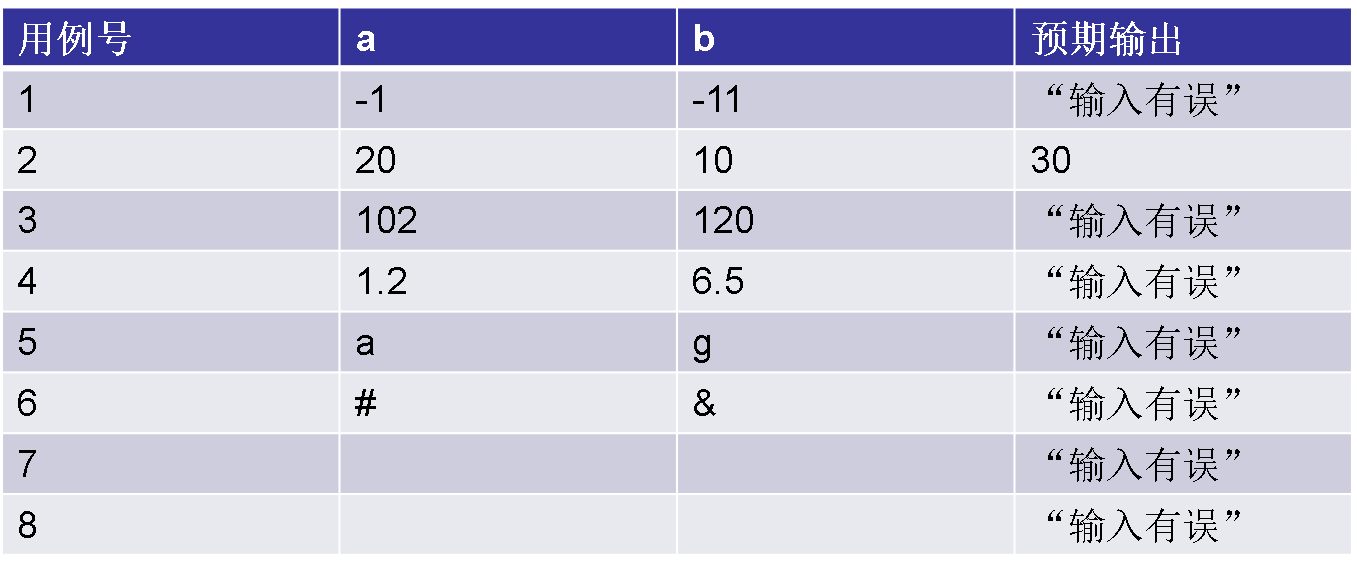
1. 利用等价类划分设计测试用例。



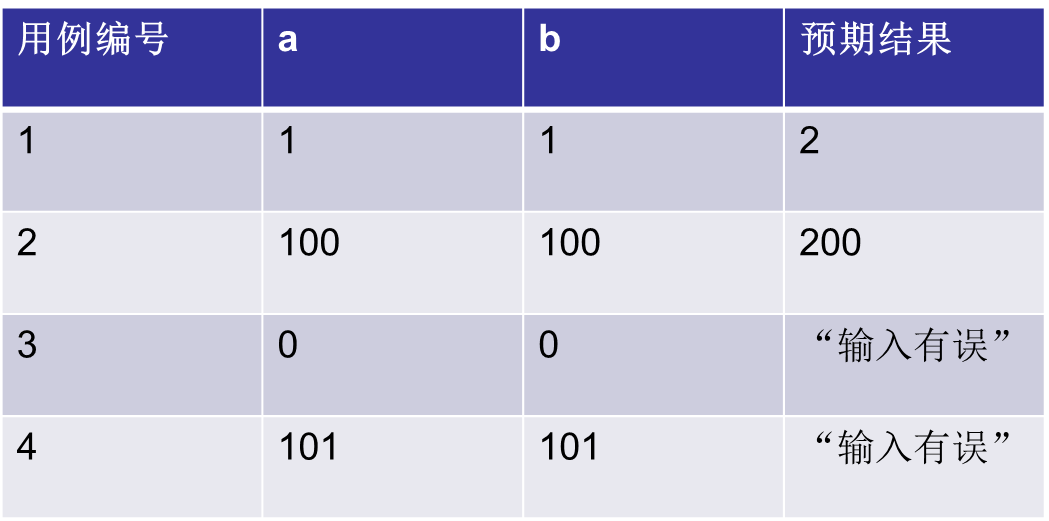
分析输入数据



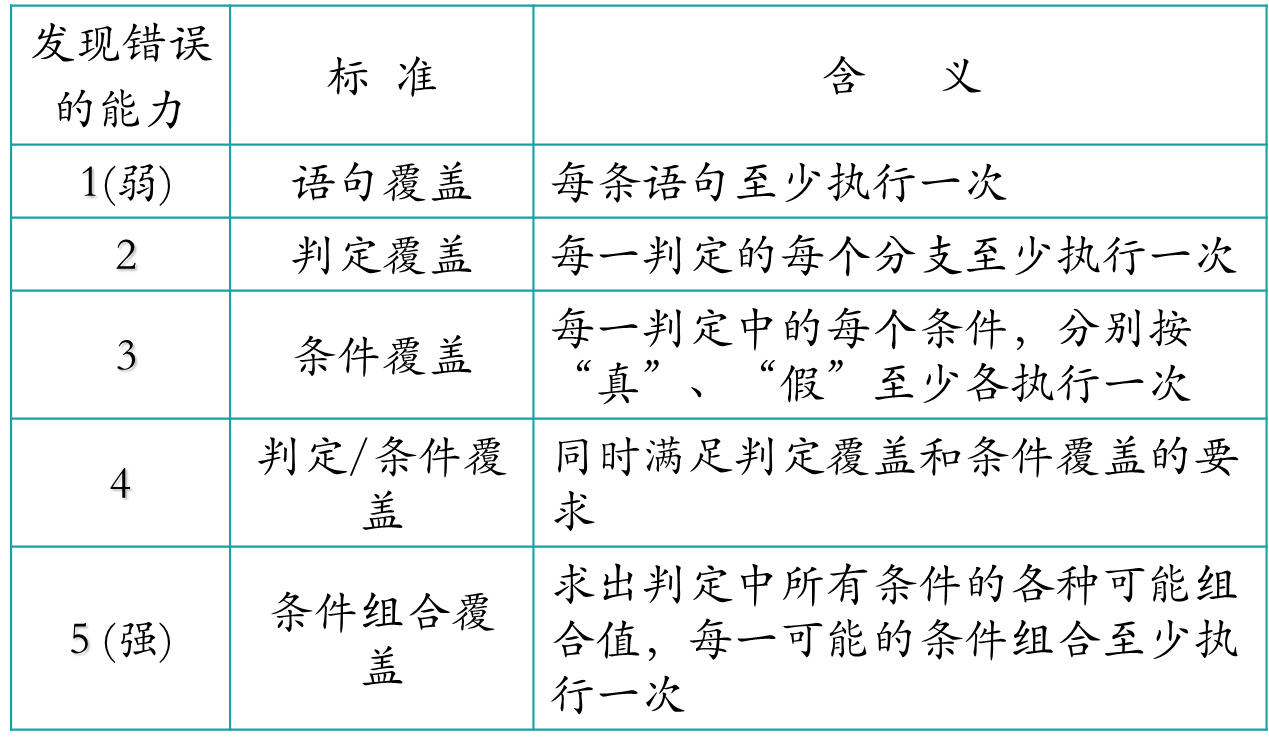
测试用例



1. 边界值测试用例实例



1. 白盒测试的语句覆盖、分支覆盖、条件覆盖等



1. 软件调试的任务是什么？有哪两部分组成

软件调试的任务：在软件测试的基础上进一步纠正和确认相关错误或问题。

1. 软件调试的方法

软件调试可采用4种方法：

1. 强行排错。主要技术和方法包括：
   1. 内存排错。
   2. 特定语句排错。
   3. 自动调试工具。
2. 回溯法排错
3. 归纳法排错
4. 演绎法排错。用演绎法排错4个步骤：
   1. 列举假设。
   2. 排除不正确假设。
   3. 进一步排查定位。
   4. 证明假设。
5. 软件维护的概念和分类（完善、适应、纠错、预防）

软件维护是指软件交付使用后，由于运行中存在的缺陷，或因业务需求及环境等变化，对软件进行微调的过程。目的是确保软件正常运行使用，提高用户满意度及服务信誉。

由上述原因产生的软件维护类型，主要包括4类：

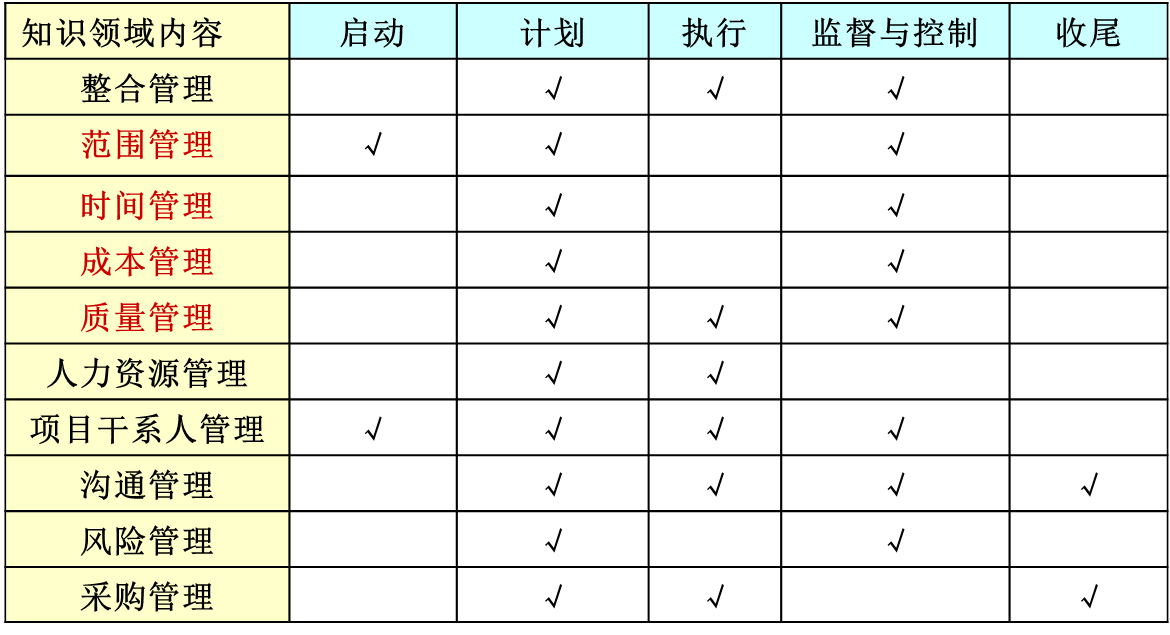
* 1. 完善性维护。
  2. 适应性维护。
  3. 纠错性维护。
  4. 预防性维护。

1. 软件测试分析文档

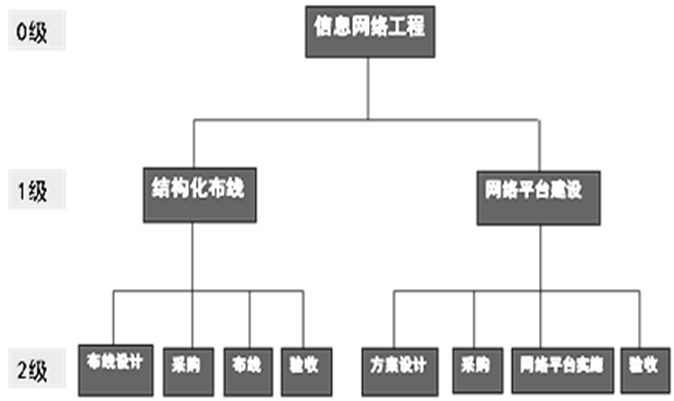
# 第八章

1. 项目管理的五个过程、十大知识领域，重要的几个领域的含义

项目管理过程:启动过程，计划过程， 执行过程，控制和收尾过程。



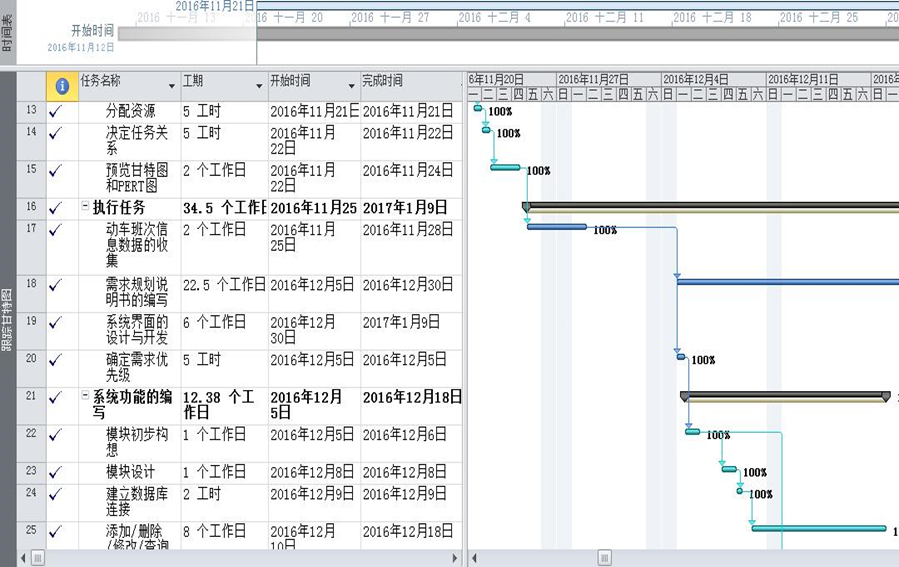
1. 项目范围管理：确保项目包括成功完成项目所需的全部工作，但有只包括成功完成项目所必须的工作过程。主要关心的是确定与控制哪些应该做哪些不应该做。
2. 项目时间管理：为确保项目按时完成所要求的各项过程，制定项目开展的时间计划。
3. 项目成本管理：保证项目在已经批准预算之内完成所必需的诸过程。主要关心项目活动所需资源的成本
4. 项目质量管理：保证项目能满足原先规定的各项要求所需要的过程。即“总体管理功能中决定质量方针、目标与责任的所有活动，并通过诸如质量规划、质量保证、质量控制、质量改进等手段在质量体系内机加一实施。”
5. 范围管理、成本（可行性分析部分讲过的投资收益比）和进度管理的内容
6. 进度管理：进度管理估算、项目进度安排
7. 范围管理： 软件项目有开发任务、管理和过程改进任务，每项任务应分解给相关的项目组，再由项目组分解落实到人.任务分解既有质和量要求，又有进度和费用方面的约束;既可按功能划分，也可按流程划分，分解方法和标准依据软件过程、项目特点、项目团队约束和项目经理经验而定。软件设计上的分层结构，通常成为任务分解结构WBS的依据.分解结果可用清单或/和图表形式表达。



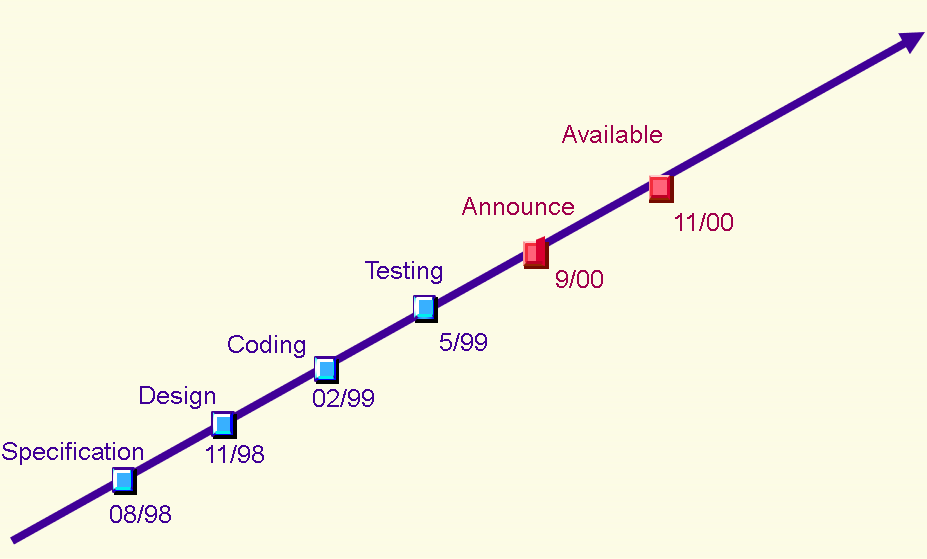
1. 进度估算的方法（定额估算法、经验公式法、CPM等）
2. 定额估算法。比较基本的估算项目历时的方法，公式为：T=Q/(R\*S)

[ T活动持续时间，可以用小时、日、周、月等表示。Q活动工作量，可以用人月、人天等单位表示。R人力或设备数量，可以用人或设备数表示。S开发效率，以单位时间完成的工作量表示。]

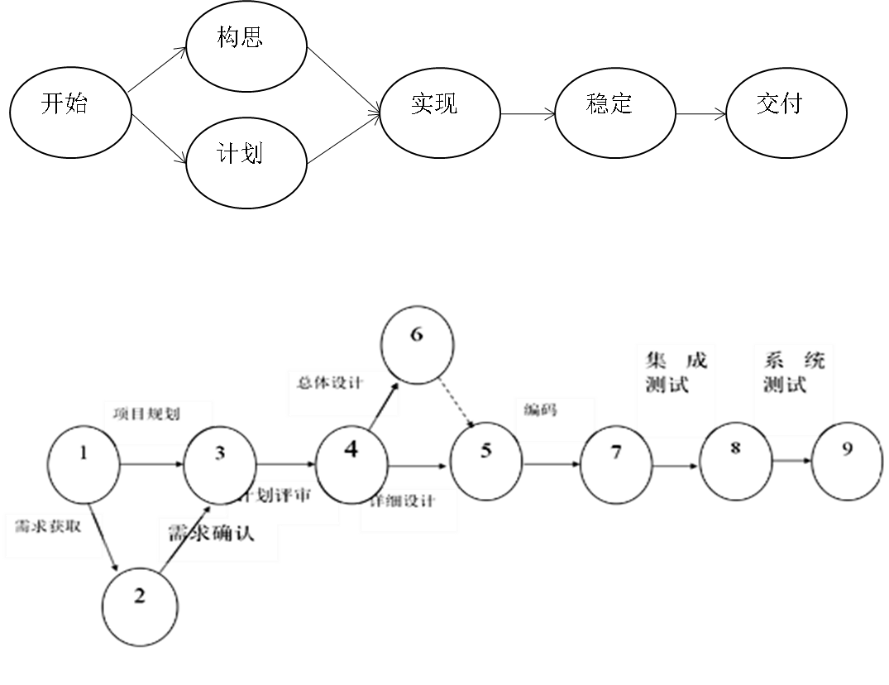
1. 经验公式法。根据大量项目数据统计而得出的模型，经验导出模型月进度为：D=a\*Eb 。其中，D表示月进度，E表示人月工作量，a为2~4的参数，b为1/3左右的参数，它们时依赖于项目自然属性的参数。经验导出模型有几种具体公式（如CcCoMo模型为D=2.5Eb，b为0.32~0.28）
2. PERT与CPM。关键路径法CPM是一种网络图方法。利用网路顺序图的逻辑关系和加权历时估算来计算项目历时，每个活动历时采用加权平均的算法：（O+4M+P）/6。其中，O是活动（项目）完成的最小估算值，或者说是最乐观值；P是活动（项目）完成的最大估算值，或者说是最悲观值；M是活动（项目）完成的最大可能估算值。
3. 进度管理的图形工具（甘特、网络、里程碑、资源等）
4. 甘特图（Gantt Chart）又称横道图，如图8-3所示，是活动进度与日历表的对照图。用水平线段表示活动的工作阶段，其中线段的长度表示完成任务所需时间，起点和终点分别表示任务的开始和结束时间。在甘特图中，任务的完成标准是交付相应文档和通过评审。甘特图表明项目的计划进度,并能动态反映当前开发进展状况,其缺点是不能表达出各任务之间复杂的逻辑关系。



1. 网络图是用网络分析的方法编制的进度计划图。主要可以描述每个活动及其之间的逻辑关系。计划评审技术 PERT 和关键路径法 CPM 都采用网络图表示项目的活动。
2. 里程碑



1. 常用的网络图（单、双）、关键路径的分析、工期的估算



关键路径是决定项目完成的最短时间；是时间浮动为0（Float=0）的路径；网络图中最长的路径；关键路径上的任何活动延迟，都会导致整个项目完成时间的延迟

1. 质量管理、风险管理的概念
2. 软件质量是指与软件产品满足各种需求（包括隐含需求）的能力相关特征的总和。质量管理主要内容包括三个过程：质量计划制定、质量保证和质量控制。
3. 软件项目的风险管理是指对软件项目可能出现的风险，进行识别、评估、预防、监控的过程。
4. 质量特性

质量特性包括功能性、可靠性、易使用性、高效性、可维护性、可移植性。

1. 风险管理（识别、评估、预防、监控）
2. 风险识别是寻找可能影响项目的风险和确认风险特性的过程。
3. 风险估算也称为风险预测，主要从风险发生的可能性（概率）和风险发生所产生的后果两个方面评估风险。
4. 风险计划是设计如何进行风险管理活动的过程，实际是一个风险管理方案（预案）。包括界定项目组织及成员风险管理的行动方案，选择合适的风险管理方法，确定风险判断的依据。降低风险的主要策略是回避风险、转移风险、损失控制和承担风险。
5. 风险分析的目的是建立应对处理及监控风险的策略。
6. 熟悉利用PROJECT进行任务分解、计算工期、分配资源、查看成本

# 软件项目

