



NEW PLAN

软件高级架构师

· 一 段 新 征 程 ·





01

软件工程



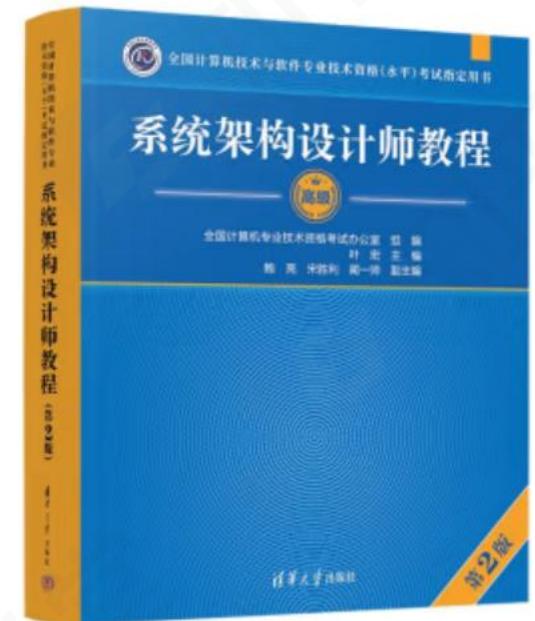
章节介绍



➤ 本章节在历年考试过程中的分值占比大概是13-15分，为架构设计师这门科目中2大重点之一，它不仅仅在选择题中考分值占比巨大，而且在案例分析和论文中都会出到题目，也是老师推荐同学们去看书的章节。对应书本是第五章，我们的课程和书本以及考试在这儿是有区别的，书本上少了软件维护，但是考试仍然会考软件维护，所以我们课程里面是有的，项目管理部分我们是单独拿出去作为一个单独的章节了

➤ 被考到知识点有：

- 软件工程基础：生命周期，能力成熟度模型，开发模型，开发方法，软件产品线，逆向工程
- 系统分析和需求工程：需求分类、需求获取、分析、定义、验证、管理。
- 系统设计：处理流程设计、系统设计、人机界面设计。
- 测试基础：测试原则、测试阶段、测试用例设计、调试、软件度量。
- 系统运行和维护：系统转换、系统维护、系统评价



➤ 在改版之后的考试中，分别考察的知识点为：

- 2023年11月：mccabe度量法、灰盒测试、喷泉模型、敏捷开发、需求分析工具(petri网)、PDCA、自动化测试适用类型、变更管理顺序、单元测试、web新型测试(A/B测试、链接测试)、W模型
- 2024年05月：敏捷开发、RUP4+1视图、需求分析、净室软件工程、静态测试、灰盒测试、系统测试依据、软件测试判断
- 2024年11月：RUP、白盒测试（2分）、系统维护类型、可复用资产、内聚分类、系统测试对应阶段、测试脚本、螺旋模型、数据流图

软件工程过程是指为获得软件产品包括以下**4**个方面活动:

- P(Plan): 软件规格说明。规定软件的功能及其运行时的限制。
- D(Do): 软件开发。开发出满足规格说明的软件。
- C(Check): 软件确认。确认开发的软件能够满足用户的需求。
- A(Action): 软件演进。软件在运行过程中不断改进以满足客户新的需求。

软件系统工具通常可以按软件过程活动将软件工具分为:

- 软件开发工具: 需求分析工具、设计工具、编码与排错工具、测试工具等。
- 软件维护工具: 版本控制工具、文档分析工具、开发信息库工具、逆向工程工具、再工程工具。
- 软件管理和软件支持工具: 项目管理工具、配置管理工具、软件评价工具、软件开发工具的评价和选择。

软件设计四个活动: **数据设计、架构(体系结构)设计、人机界面(接口)设计和过程(功能)设计**

软件开发生命周期:

- 软件定义时期：包括可行性研究和详细需求分析过程，任务是确定软件开发工程必须完成的总目标，具体可分成问题定义、可行性研究、需求分析等。
- 软件开发时期：就是软件的设计与实现，可分成概要设计、详细设计、编码、测试等。
- 软件运行和维护：就是把软件产品移交给用户使用。

软件系统的文档可以分为用户文档和系统文档两类:

- 用户文档主要描述系统功能和使用方法，并不关心这些功能是怎样实现的；
- 系统文档描述系统设计、实现和测试等各方面的内容。

软件工程生命周期：

- 可行性分析与项目开发计划：主要确定软件的开发目标及其可行性。
- 需求分析：确定软件系统的功能、性能、数据和界面等要求，从而确定系统的逻辑模型，该阶段产生的主要文档有软件需求说明书。
- 概要设计：概要设计就是设计软件的结构，明确软件由哪些模块组成，这些模块的层次结构、调用关系、功能是什么样的。设计该项目总体数据结构和数据库结构，即应用系统要存储什么数据，这些数据是什么样的结构，它们之间有什么关系。该阶段产生的主要文档有概要设计说明书。
- 详细设计：详细设计阶段的主要任务是把功能描述转变为精确的、结构化的过程描述。即该模块的控制结构是怎样的，先做什么，后做什么，有什么样的条件判定，有些什么重复处理等，并用相应的表示工具把这些控制结构表示出来。该阶段产生的主要文档有详细设计文档。
- 编码：编码阶段就是把每个模块的控制结构转换成计算机可接受的程序代码
- 测试：测试是保证软件质量，该阶段产生的主要文档有软件测试计划、测试用例和软件测试报告。
- 维护：软件维护是软件生存周期中时间最长的阶段。



能力等级	特点	关键过程区域
初始级	软件过程的特点是杂乱无章，有时甚至很混乱，几乎没有明确定义的步骤，项目的成功完全依赖个人的努力和英雄式核心人物的作用。	
可重复级	建立了基本的项目管理过程和实践来跟踪项目费用、进度和功能特性， 有必要的过程准则来重复以前在同类项目中的成功。	软件配置管理、软件质量保证、软件子合同管理、软件项目跟踪与监督、软件项目策划、软件需求管理
已定义级	管理和工程两方面的软件过程已经 文档化、标准化 ，并综合成整个软件开发组织的标准软件过程。所有项目都采用根据实际情况修改后得到的标准软件过程来发和维护软件。	同行评审、组间协调、软件产品工程、集成软件管理、培训大纲、组织过程定义、组织过程集点
已管理级	制定了软件过程和产品质量的详细度量标准 。对软件过程和产品质量有定量的理解和控制。	软件质量和定量过程管理
优化级	加强了定量分析，通过来自过程质量反馈和来自新观念、新技术的反馈使过程能 不断持续地改进 。	过程更改管理、技术改革管理和缺陷预防



能力等级	特点	关键过程区域
初始级	过程不可预测且缺乏控制	
已管理级	过程为项目服务	需求管理、项目计划、配置管理、项目监督与控制、供应商合同管理、度量和分析、过程和产品质量保证
已定义级	过程为组织服务	需求开发、技术解决方案、产品集成、验证、确认组织级过程焦点、组织级过程定义、组织级培训、集成项目管理、风险管理、集成化的团队、决策分析和解决方案、组织级集成环境
定量管理	过程已度量和控制	组织过程性能、定量项目管理
优化级	集中于过程改进和优化	组织级改革与实施、因果分析和解决方案

软件过程模型习惯上也称为软件开发模型，它是软件开发全部过程、活动和任务的结构框架。

- 瀑布模型
- V模型
- 增量模型
- 演化模型（原型模型、螺旋模型）
- 喷泉模型
- 基于构件的开发模型
- 形式化方法模型等
- 统一过程模型（RUP）
- 敏捷开发方法

瀑布模型：将软件生存周期中的各个活动规定为依线性顺序连接的若干阶段的模型，包括需求分析、设计、编码、测试、运行与维护。它规定了由前至后、相互衔接的固定次序，如同瀑布流水逐级下落

瀑布模型特点：

- 从上一项开发活动接受该项活动的工作对象作为输入。
- 以文档作为驱动、适合于软件需求很明确的软件项目的模型。
- 每个阶段都需要对该项活动的实施工作成果进行评审。若其工作成果得到确认，则继续进行下一项开发活动；否则返回前一项，甚至更前项的活动。
- 在瀑布模型中，需求或设计中的错误往往只有到了项目后期才能够被发现，对于项目风险的控制能力较弱，从而导致项目常常延期完成，开发费用超出预算。

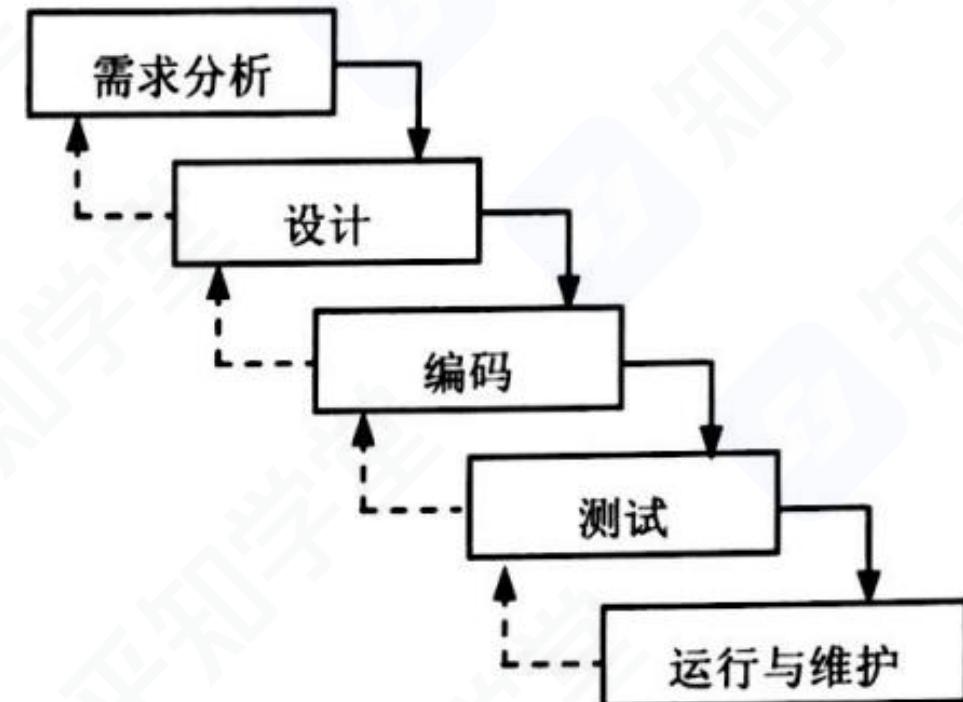


图 5-2 瀑布模型



V模型：V模型描述了质量保证活动和沟通、建模相关活动以及早期构建相关的活动之间的关系。V模型提供了一种将验证确认活动应用于早期软件工程工作中的方法。

V模型特点：

- 单元测试的主要目的是针对编码过程中可能存在的各种错误；
- 集成测试的主要目的是针对详细设计中可能存在的问题；
- 系统测试主要针对概要设计，检查系统作为一个整体是否有效地得到运行；
- 验收测试通常由业务专家或者用户进行，以确认产品能真正符合用户业务上的需要。
- V模型适用于需求明确和需求变更不频繁的情形。

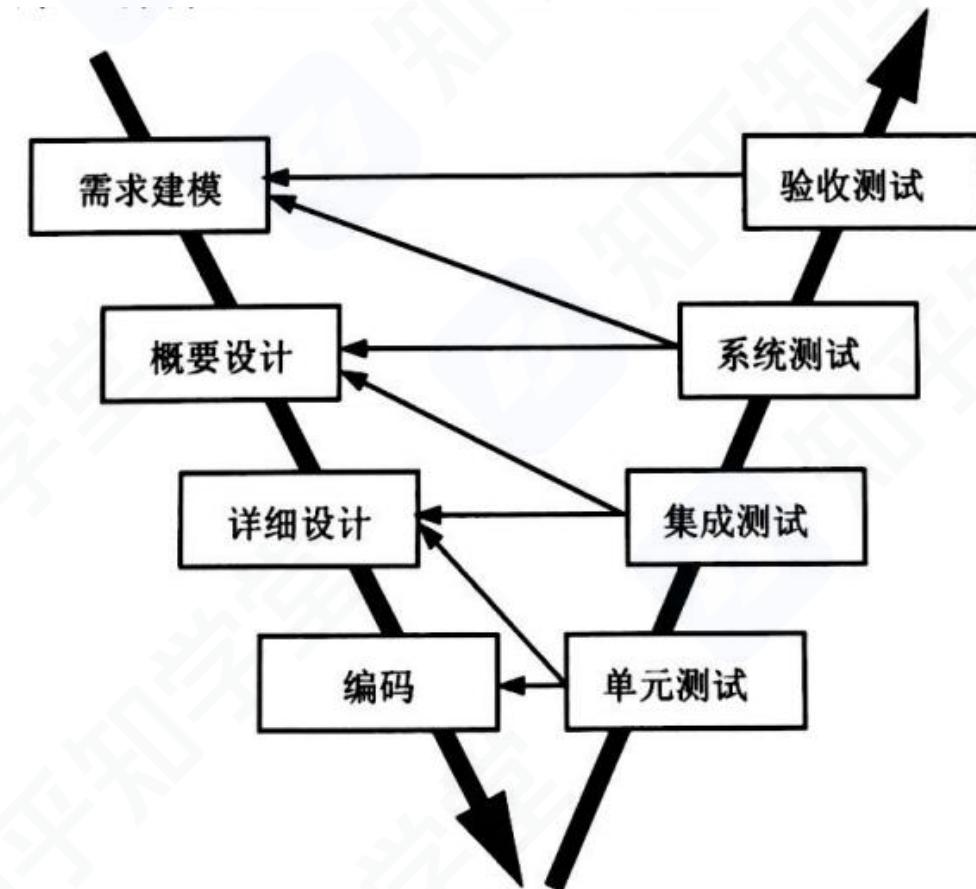
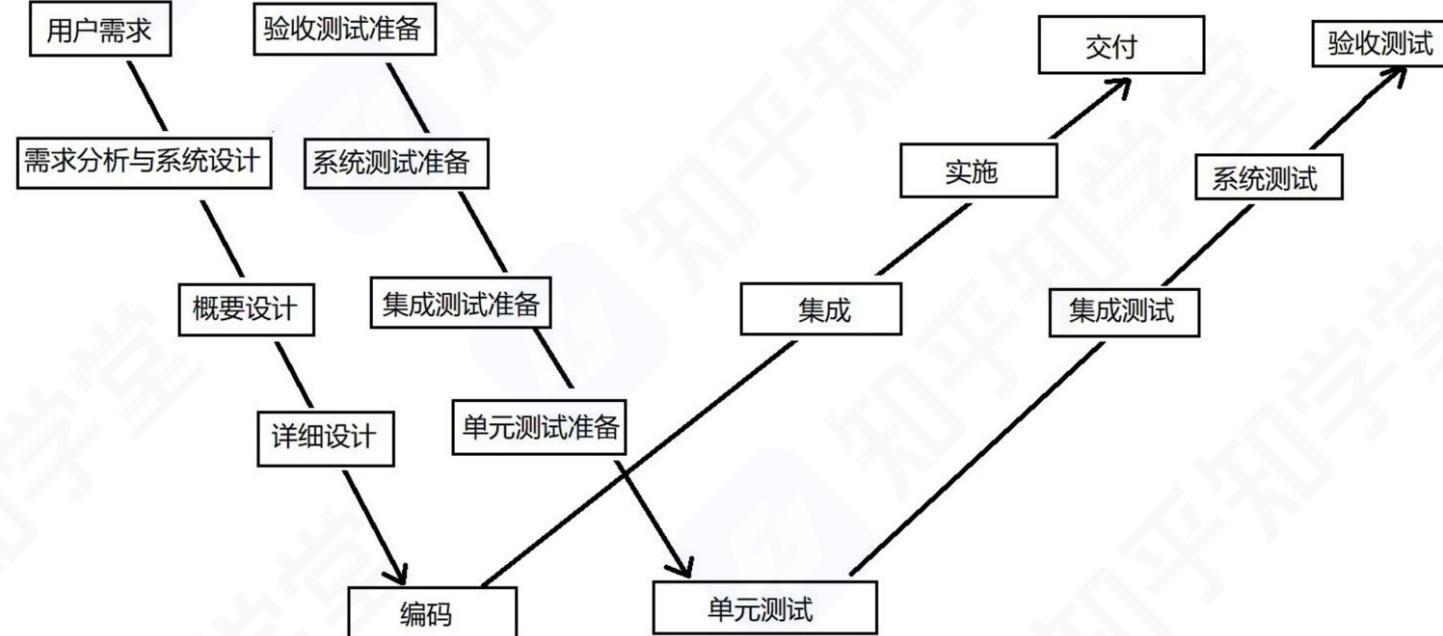


图 5-3 V 模型

W模型的定义：W模型是对V模型的一种改进。W模型中，软件开发和测试是紧密结合的，每个开发活动完成后就同步进行测试活动：需求分析完成后进行需求测试；设计完成后进行设计测试；编码完成后进行单元测试；集成完成后进行集成测试；系统构建完成后进行系统测试；完成交付准备工作之后进行验收测试。（W模型也叫双V模型）

W模型的缺点：W模型中开发活动都是串行的，开发和测试也是一种线性的关系，只有开发活动完成了才能进行测试活动。这种方式使得W模型无法适应敏捷、迭代开发，以及灵活的变更调整。



H模型的定义：H模型中的测试活动是一个独立的流程，只要满足了测试就绪条件，就可以开始测试活动。这种灵活的组织方式，使得H模型完全具备了前两个模型的优点：既可以与所有的开发活动紧密结合，又足够灵活满足敏捷和迭代的开发模型。

W模型的缺点：H模型的灵活也造就它难以驾驭的特点。如果管理者没有足够的经验就实施H模型，可能会事倍功半，测试活动的成本收益比会比较低。

总结：根据以上测试3种模型的特点，建议一般的软件开发过程采用W模型，实施敏捷和迭代开发的可以考虑采用H模型。





原型模型：

- 演化模型是迭代的过程模型，使得软件开发人员能够逐步开发出更完整的软件版本。典型的演化模型有原型模型和螺旋模型等。
- 原型模型开始于沟通，其目的是定义软件的总体目标，标识需求，然后快速制订原型开发的计划，并构建原型。交付给客户使用，并收集客户的反馈意见，这些反馈意见可在下一轮中对原型进行改进。

原型模型特点：

- 原型不必满足目标软件的所有约束，其目的是能快速、低成本地构建原型；
- 原型方法比较适合于用户需求不清、需求经常变化的情况。
- 构造方便、快速，造价低。对用户的需求是动态响应、逐步纳入的，不适合大型项目

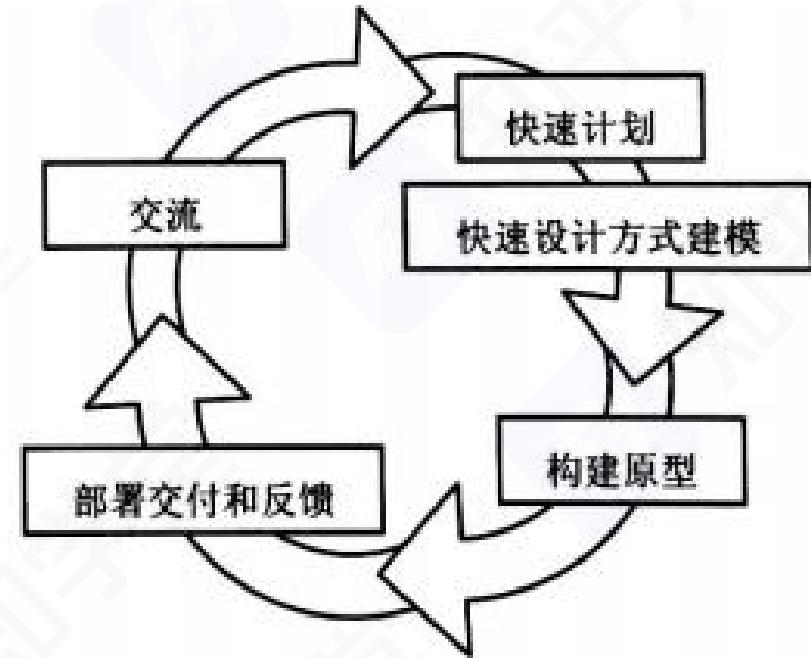


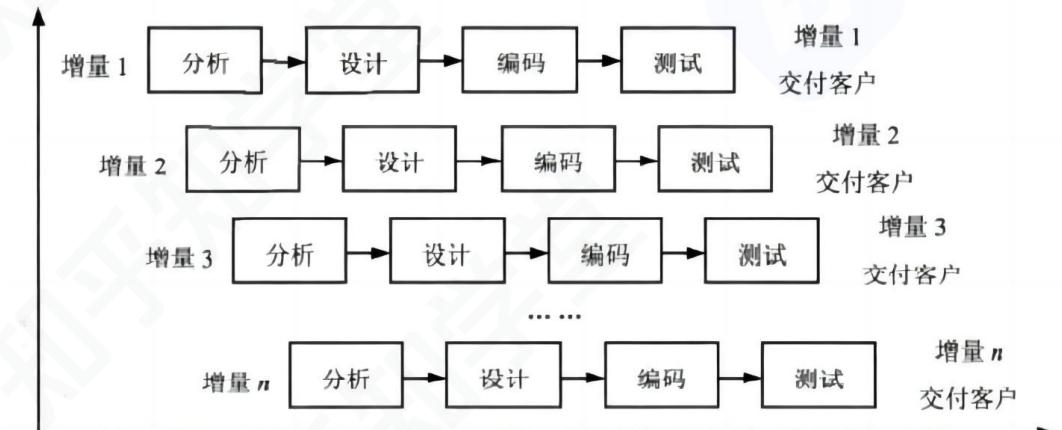
图 5-5 原型模型



增量模型：增量模型将需求分段为一系列增量产品，每一增量可以分别开发。该模型采用随着日程时间的进展而交错的线性序列，每一个线性序列产生软件的一个可发布的“增量”，第1个增量往往是核心的产品。客户对每个增量的使用和评估都作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。

增量模型特点：

- 由于并不是从系统整体角度规划各个模块，因此不利于模块划分，难点在于如何将客户需求划分为多个增量
- 与原型不同的是增量模型的每一次增量版本都可作为独立可操作的作品，而原型的构造一般是为了演示
- 第一个可交付版本所需要的成本和时间很少
- 开发由增量表示的小系统所承担的**风险不大**
- 由于很快发布了第一个版本，因此可以**减少用户需求的变更**





螺旋模型：

- 螺旋模型是一个演化软件过程模型，将原型实现的迭代特征与线性顺序(瀑布)模型中控制的和系统化的方面结合起来。在螺旋模型中，软件开发是一系列的增量发布。
- 开发过程具有周期性重复的螺旋线状。四个象限分别标志每个周期所划分的四阶段：**制订计划、风险分析、实施工程和客户评估**

螺旋模型特点：

- 融合了瀑布模型的线性渐进与原型模型的演化迭代
- 强调风险驱动，贯穿整个软件生存周期
- 需求管理贯穿始终，强调需求的动态变化
- 适用于高风险、不确定性大的复杂系统
- 需要丰富的风险管理经验
- 增加了客户参与度，通过定期评审确保客户满意
- 可能导致成本增加和时间延误，特别是迭代次数过多时

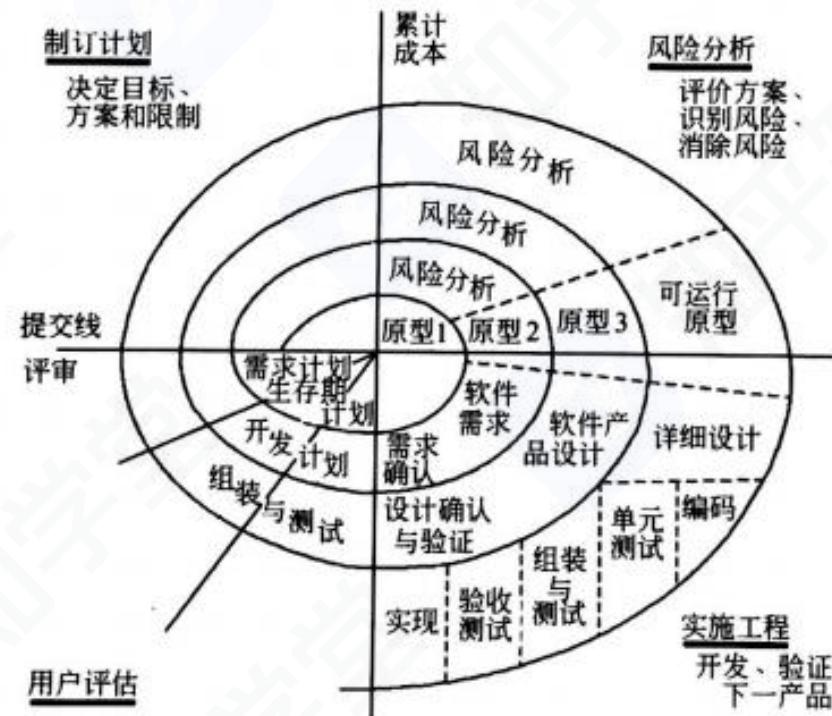


图 5-6 螺旋模型

RUP把软件开发生命周期划分为多个循环，每个循环生成产品的一个新的版本，每个循环依次由**4**个连续的阶段组成，每个阶段完成确定的任务。这**4**个阶段如下。

- 初始阶段：定义最终产品视图和业务模型，并确定系统范围。
- 细化阶段：设计及确定系统的体系结构，制订工作计划及资源要求。
- 构造阶段：构造产品并继续演进需求、体系结构、计划直至产品提交。
- 移交阶段：把产品提交给用户使用。

RUP的特点:

- 用例驱动：需求分析、设计、实现和测试等活动都是用例驱动的。
- 以体系结构为中心：包括系统的总体组织和全局控制、通信协议等。是一个多维的结构，会采用多个视图来描述。
- 迭代与增量。把整个项目开发分为多个迭代过程。在每次迭代中，只考虑系统的一部分需求，进行分析、设计、实现、测试和部署等过程；每次迭代是在已完成部分的基础上进行的，每次增加一些新的功能实现，以此进行下去，直至最后项目的完成。

~~典型的4+1视图模型中：~~

- ~~分析人员和测试人员关心的是系统的行为，会侧重于用例视图，~~
- ~~最终用户关心的是系统的功能，会侧重于逻辑视图，~~
- ~~程序员关心的是系统的配置、装配等问题，会侧重于实现视图，~~
- ~~系统集成人员关心的是系统的性能、可伸缩性、吞吐率等问题，会侧重于进程视图，~~
- ~~系统工程师关心的是系统的发布、安装、拓扑结构等问题，会侧重于部署视图。~~



敏捷开发方法:

- 敏捷开发的总体目标是通过“尽可能早地、持续地对有价值的软件的交付”使客户满意。通过在软件开发过程中加入灵活性，敏捷方法使用户能够在开发周期的后期增加或改变需求。
- 敏捷过程的典型方法有很多，实现了敏捷方法所宣称的理念就称之为敏捷宣言。
- 敏捷宣言：个体和交互胜过过程和工具、可以工作的软件胜过面面俱到的文档、客户合作胜过合同谈判、响应变化胜过遵循计划。

核心思想:

- 敏捷方法是适应型，而非可预测型。
- 敏捷方法是以人为本，而非以过程为本。
- 以原型开发思想为基础，采用增量式开发，发行版本小型化。

敏捷开发方法-极限编程 (XP)：XP是一种轻量级（敏捷）、高效、低风险、柔性、可预测的、科学的软件开发方式。它由**价值观、原则、实践和行为**4个部分组成，彼此相互依赖、关联，并通过行为贯穿于整个生存周期。



软件复用：是将已有软件的各种有关知识用于建立新的软件，以缩减软件开发和维护的花费。软件复用是提高软件生产力和质量的一种重要技术。早期的软件复用主要是代码级复用，被复用的知识专指程序，后来扩大到包括领域知识、开发经验、设计决定、体系结构、需求、设计、代码和文档等一切有关方面。

逆向工程：软件的逆向工程是分析程序，力图在比源代码更高抽象层次上建立程序的表示过程，逆向工程是设计的恢复过程。

逆向工程的四个级别：

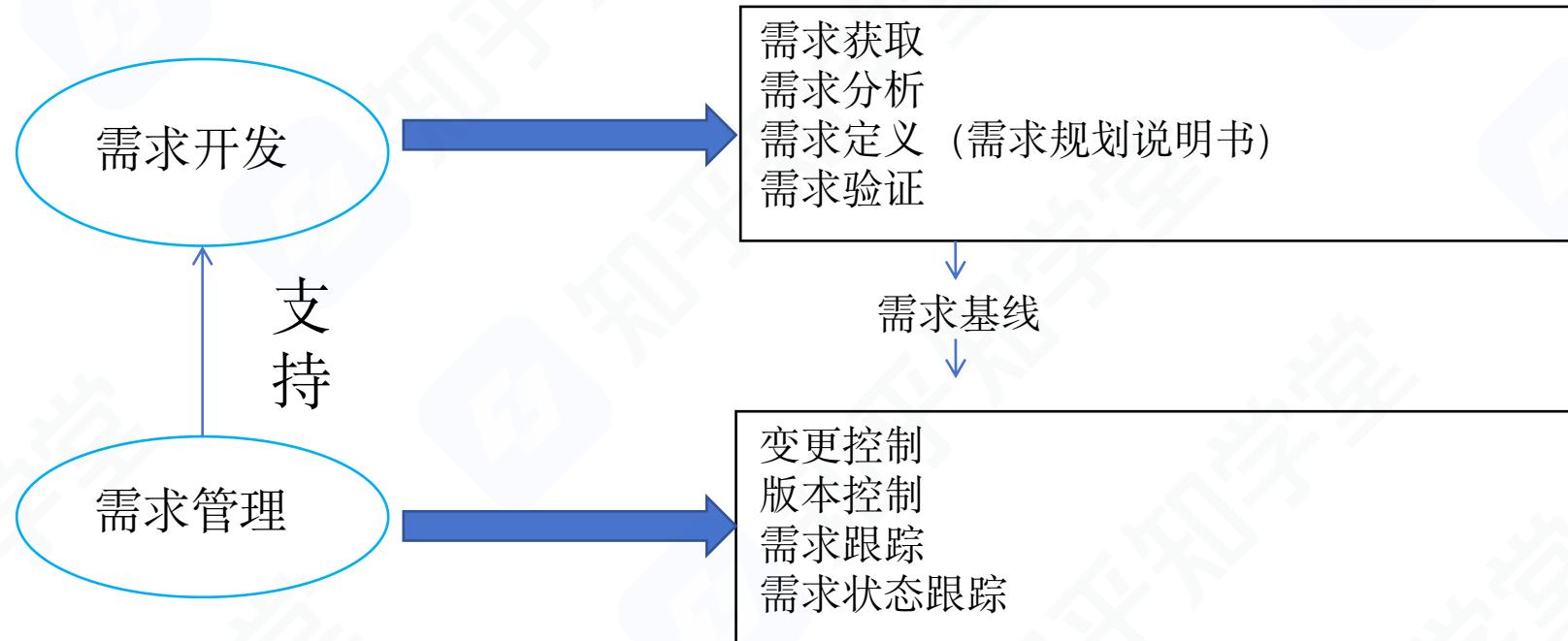
- 实现级：包括程序的抽象语法树、符号表、过程的设计表示。
- 结构级：包括反映程序分量之间相互依赖关系的信息，例如调用图、结构图、程序和数据结构。
- 功能级：包括反映程序段功能及程序段之间关系的信息，例如数据和控制流模型。
- 领域级：包括反映程序分量或程序诸实体与应用领域概念之间对应关系的信息，例如E-R模型。

其中，领域级抽象级别最高，完备性(完整)最低，实现级抽象级别最低，完备性最高。

软件需求:指用户对系统在功能、行为、性能、设计约束等方面的期望。是指用户解决问题或达到目标所需的条件或能力,是系统或系统部件要满足合同、标准、规范或其他正式规定文档所需具有的条件或能力,以及反映这些条件或能力的文档说明。

需求分类:

- 书本上的分类: 功能需求、性能需求、环境需求、界面需求、文档需求、数据需求、资源使用需求、安全保密要求、可靠性要求、成本消耗和开发进度需求、其他非功能性需求
- 常见的分类:
 - **业务需求**: 企业或客户对系统高层次的目标要求
 - **用户需求**: 描述的是用户的具体目标,或用户要求系统必须能完成的任务。即描述了用户能使用系统来做什么
 - **系统需求**: 从系统的角度来说明软件的需求,包括功能需求、非功能需求和设计约束等



常见的需求获取法包括：

- 用户访谈：1对1-3,找有代表性的用户进行访谈，对提问者的水平是有要求的。其形式包括结构化(有剧本)和非结构化(随意发挥)两种。
- 问卷调查：用户多，无法一一访谈，收集到的需求不够精准，比较杂乱，比较考验问卷编写者的水平
- 采样：从种群中系统地选出有代表性的样本集的过程，类似于数学中的数理统计。样本数量= $0.25 * (\text{可信度因子}/\text{错误率})^2$
- 情节串联板：一系列图片，通过这些图片来把需求给进行叙述出来，这样虽然生动，但是耗时
- 联合需求计划(JRP)：通过联合各个关键用户代表、系统分析师、开发团队代表一起，通过有组织的会议来讨论需求。
- 需求记录技术：任务卡片、场景说明、用户故事、Volere白卡

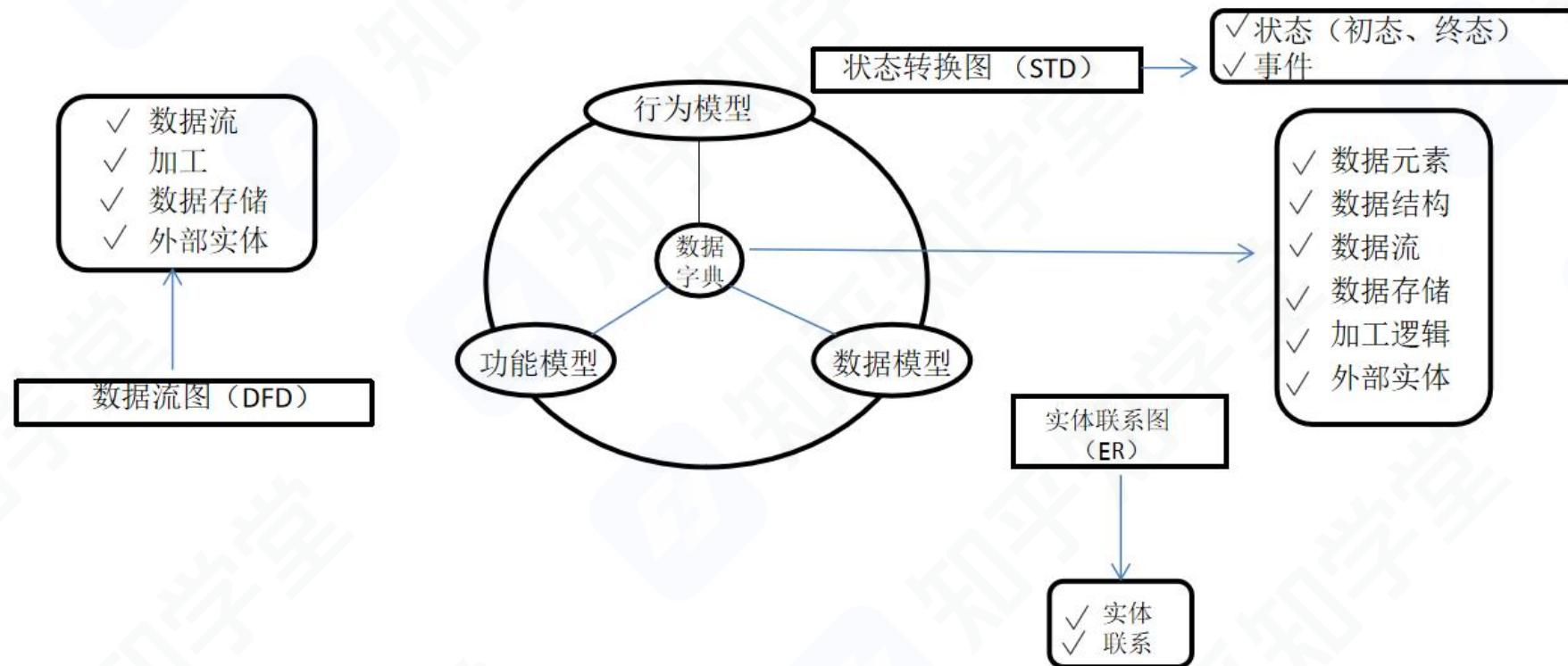
需求分析：一个好的需求应该具有无二义性、完整性、一致性、可测试性、确定性、可跟踪性、正确性、必要性等特性，因此，需要分析人员把杂乱无章的用户要求和期望转化为用户需求，这就是需求分析的工作。

常见的需求分析任务包括：

- 绘制系统上下文范围关系图（数据流图）
- 创建用户界面原型
- 分析需求的可行性
- 确定需求的优先级
- 为需求建立模型
- 创建数据字典
- 使用QFD(QFD:质量功能部署，把需求和QFD进行关联)

结构化特点：自顶向下，逐步分解，面向数据。

三大模型：功能模型(数据流图)、行为模型(状态转换图)、数据模型(ER图)以及数据字典。





数据流图DFD基本图形元素：外部实体、加工、数据存储、数据流。



(a) 外部实体 (External Agent)



或



(b) 加工 (Process)



或



(c) 数据存储 (Data Store)



(d) 数据流 (Data Flow)

图 6-8 DFD 的基本图形元素

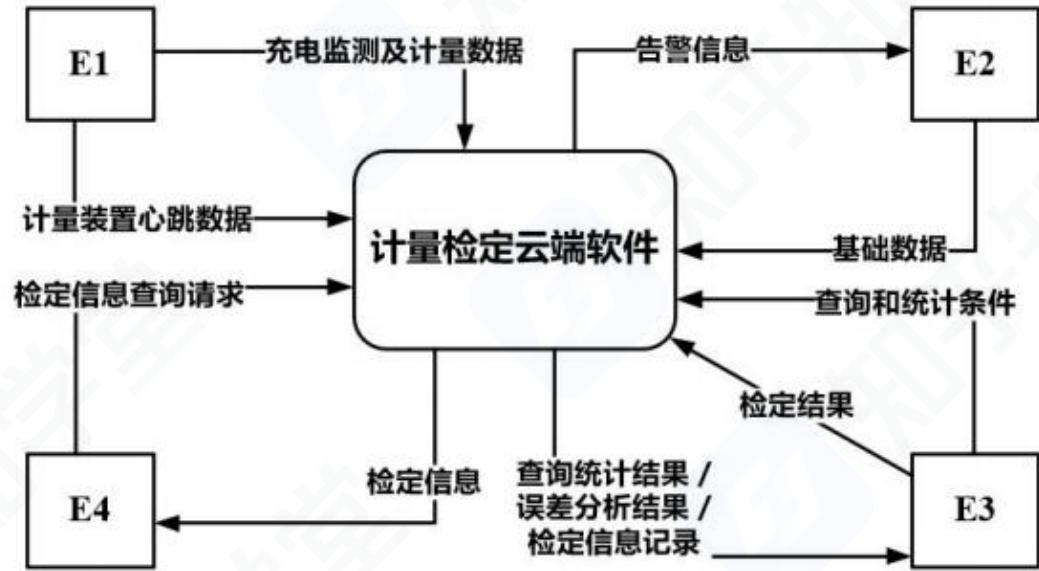


图 1-1 上下文数据流图

2022年下半年 软件设计师 下午试卷

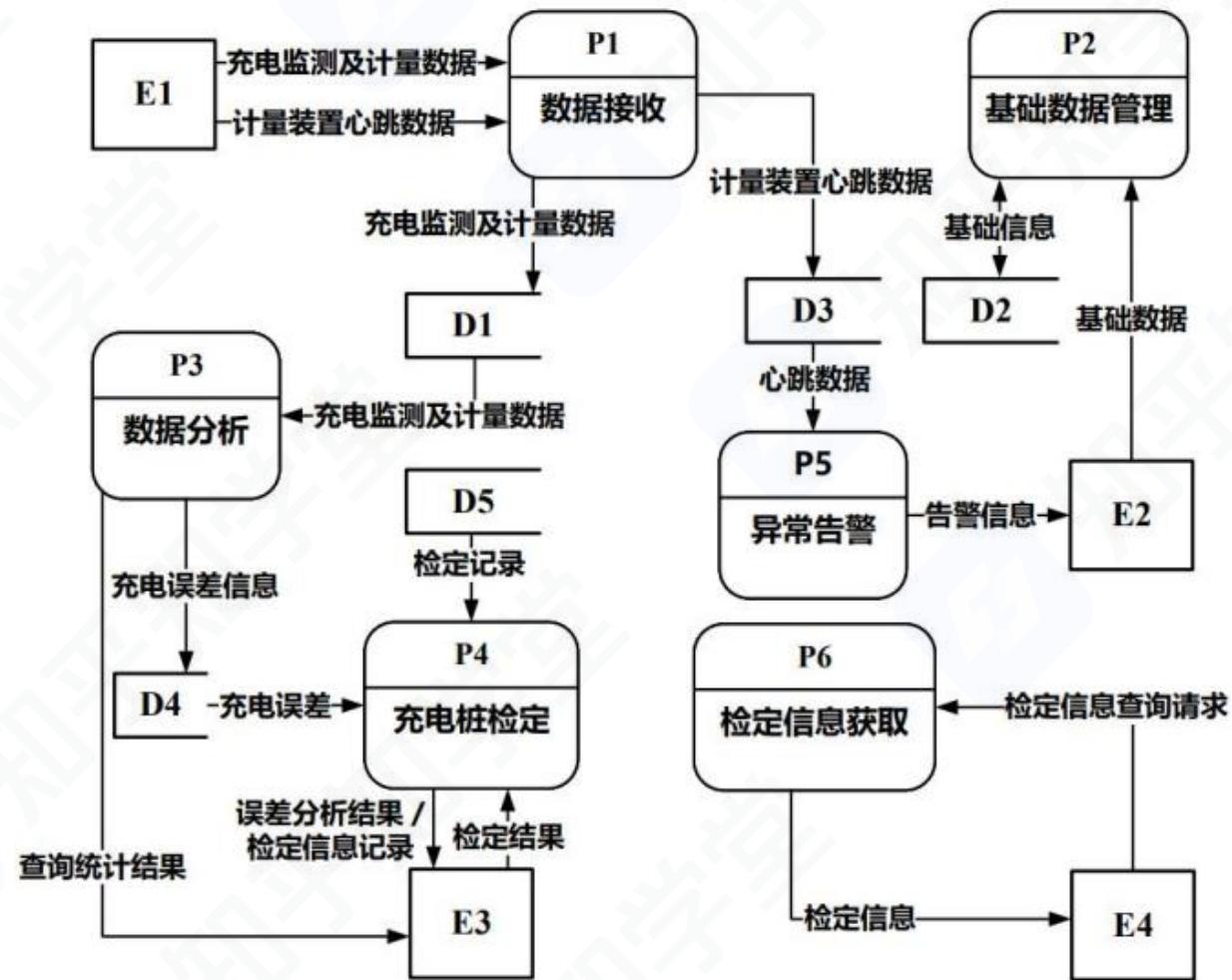


图 1-2 0 层数据流图

需求定义(软件需求规格说明书SRS):是需求开发活动的产物， 编制该文档的目的是使项目干系人与开发团队对系统的初始规定有一个共同的理解， 使之成为整个开发工作的基础。 SRS是软件开发过程中最重要的文档之一， 对于任何规模和性质的软件项目都不应该缺少。

需求定义方法

- 严格定义也称为预先定义 (结构化定义) ， 需求的严格定义建立在以下的基本假设之上：所有需求都能够被预先定义。开发人员与用户之间能够准确而清晰地交流。采用图形(或文字)可以充分体现最终系统，适合需求明确的情况。
- 原型方法，迭代的循环型开发方式，需要注意的问题：并非所有的需求都能在系统开发前被准确地说明。项目干系人之间通常都存在交流上的困难，原型提供了克服困难的一个手段。特点：需要实际的、可供用户参与的系统模型。有合适的系统开发环境。反复是完全需要和值得提倡的，需求一旦确定，就应遵从严格的方法。

需求验证：也称为需求确认，目的是与用户一起确认需求无误，对需求规格说明书**SRS**进行评审和测试，包括两个步骤：

- 需求评审：正式评审和非正式评审。
- 需求测试：设计概念测试用例，设计场景来测试需求，没有代码。

需求验证通过后，要请用户签字确认，作为验收标准之一，此时，这个需求规格说明书就是需求基线，不可以再随意更新，如果需要更改必须走需求变更流程。

定义需求基线：通过了评审的需求说明书就是需求基线，下次如果需要变更需求，就需要按照流程来一步步进行。需求的流程及状态如下图所示：



图 11-17 需求状态的变化

需求跟踪：也称之为双向跟踪。分为两种方式：

- 正向跟踪表示用户原始需求是否都实现了，正向跟踪一般是用来判断产品有没有少实现；
- 反向跟踪表示软件实现的是否都是用户要求的，不多不少，可以用原始需求和用例表格(需求跟踪矩阵)来表示，反向跟踪一般是用来判断产品有没有多实现；

使用方式：若原始需求和用例有对应，则在对应栏打对号，若某行都没有对号，表明原始需求未实现，正向跟踪发现问题；若某列都没有对号，表明有多余功能用例，软件实现了多余功能，反向跟踪发现问题。



用例 原始需求	UC-1	UC-2	UC-3	UC-n
FR-1					
FR-2					
.....					
FR-m					

()是关于需求管理正确的说法。

- A.为达到过程能力成熟度模型第二级，组织机构必须具有3个关键过程域
- B.需求的稳定性不属于需求属性
- C.需求变更的管理过程遵循变更分析和成本计算、问题分析和变更描述、变更实现的顺序
- D.变更控制委员会对项目中任何基线工作产品的变更都可以做出决定

在结构化分析中，用数据流图描述()。当采用数据流图对一个图书馆管理系统进行分析时，()是一个外部实体。

- A.对象之间的关系，用于对数据建模
- B.在系统中如何被传送或变换，以及如何对数据流进行变换的功能或子功能，用于对功能建模
- C.对外部事件如何响应，如何动作，用于对行为建模
- D.流图中的各个组成部分

- A.读者
- B.图书
- C.借书证
- D.借阅



系统设计基本原理:

- 抽象：把现实中的业务抽象到信息系统中
- 模块化：可组合、分解和更换的单元
- 信息隐蔽：将每个程序的成分隐蔽或封装在一个单一的设计模块中
- 模块独立：每个模块完成一个相对独立的特定子功能，且与其他模块之间的联系简单

模块的设计要求独立性高，就必须**高内聚，低耦合**

- 内聚：是指一个模块内部功能之间的相关性
- 耦合：是指多个模块之间的联系

内聚程度从低到高如下表所示：

内聚分类	定义	记忆
偶然内聚	一个模块内的各处理元素之间没有任何联系	无直接关系
逻辑内聚	模块内执行若干个逻辑上相似的功能，通过参数确定该模块完成哪一个功能	逻辑相似、参数决定
时间内聚	把需要同时执行的动作组合在一起形成的模块。	同时执行
过程内聚	一个模块完成多个任务，这些任务必须按指定的过程程序执行	指定的过程顺序
通信内聚	模块内的所有处理元素都在同一个任务	相通数据结构、相通输入输出
顺序内聚	一个模块中各个处理元素都密切相关同一功能且必须顺序执行，前一个功能元素的输出就是下一个功能元素的输入	顺序执行、输入为输出
功能内聚	最强的内聚，模块内的所有单元共同作用完成一个功能，缺一不可	共同作用、缺一不可

耦合程度从低到高如下表所示：

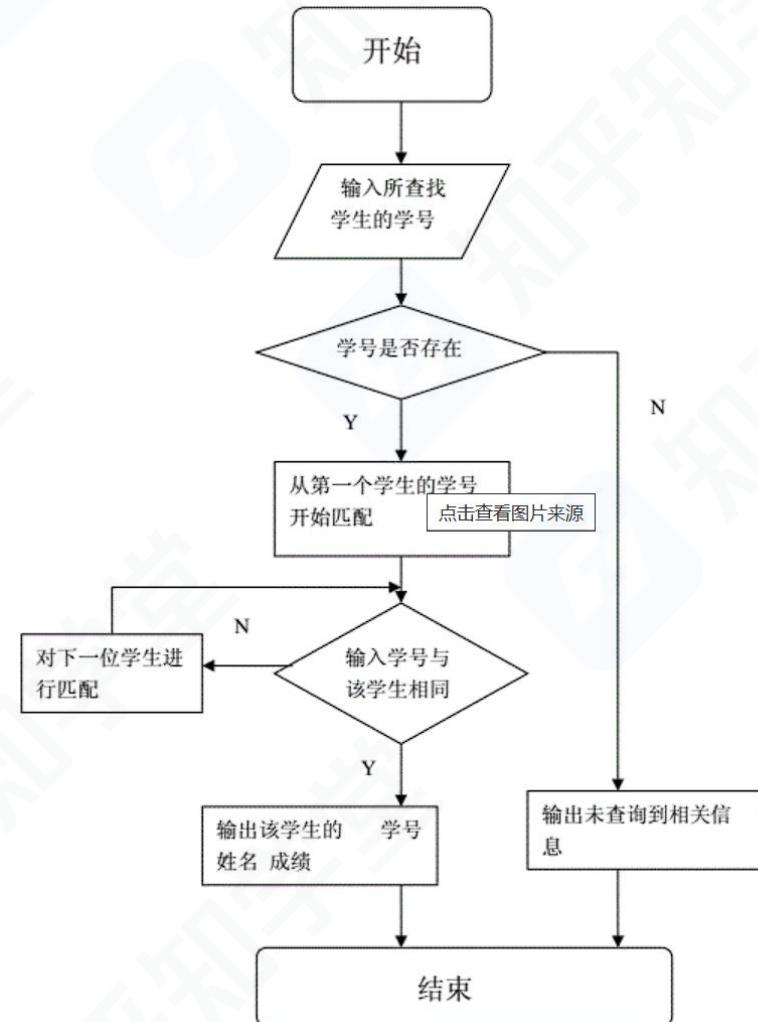
耦合分类	定义	记忆
无直接耦合	两个模块之间没有直接的关系，它们分别从属于不同模块的控制与调用，不传递任何信息。	无直接关系
数据耦合	两个模块之间有调用关系，传递的是简单的数据值，相当于高级语言中的值传递。	传递数据值调用
标记耦合	两个模块之间传递的是数据结构	传递数据结构
控制耦合	一个模块调用另一个模块时，传递的是控制变量	软件外部环境
外部耦合	模块间通过软件之外的环境联合（如I/O将模块耦合到特定的设备、格式、通信协议上）	软件外部环境
公共耦合	通过一个公共数据环境相互作用的那些模块间的耦合	外部公共数据
内容耦合	当一个模块直接使用另一个模块的内部数据。或通过非正常入口转入另一个模块内部时	模块内部关联

1. 系统总体结构设计：基本任务是采用某种设计方法，将一个复杂的系统按功能划分成模块、确定每个模块的功能、确定模块之间的调用关系、确定模块之间的接口，即模块之间传递的信息、评价模块结构的质量。
2. 数据结构及数据库设计
 - 数据结构的设计
 - 数据库的设计
3. 编写概要设计文档：主要有概要设计说明书、数据库设计说明书、用户手册以及修订测试计划。
4. 评审：对设计部分是否完整地实现了需求中规定的功能、性能等要求，设计方法的可行性，关键的处理及内外部接口定义的正确性、有效性、各部分之间的一致性等都一一进行评审。

1. 对每个模块进行详细的算法设计，用某种图形、表格和语言等工具将每个模块处理过程的详细算法描述出来。（流程图、IPO图、N-S图、PAD图）
2. 对模块内的数据结构进行设计。
3. 对数据库进行物理设计，即确定数据库的物理结构。
4. 其他设计。根据软件系统的类型，还可能要进行以下设计。
 - 代码设计
 - 输入输出格式设计
 - 用户界面设计
5. 编写详细设计说明书。
6. 评审。

程序流程图(Program Flow Diagram,PFD): 用一些图框表示各种操作，它独立于任何一种程序设计语言，比较直观、清晰，易于学习掌握。任何复杂的程序流程图都应该由顺序、选择和循环结构组合或嵌套而成。

IPO图：也是流程描述工具，用来描述构成软件系统的每个模块的输入、输出和数据加工。结构化设计中的数据流图就是一种IPO图。



N-S图(盒图): 比较容易表示嵌套和层次关系，并具有强烈的结构化特征。但是当问题很复杂时，N-S图可能很大，因此不适合于复杂程序的设计。

问题分析图(PAD): 是一种支持结构化程序设计的图形工具。PAD具有清晰的逻辑结构、标准化的图形等优点，更重要的是，它引导设计人员使用结构化程序设计方法，从而提高程序的质量。

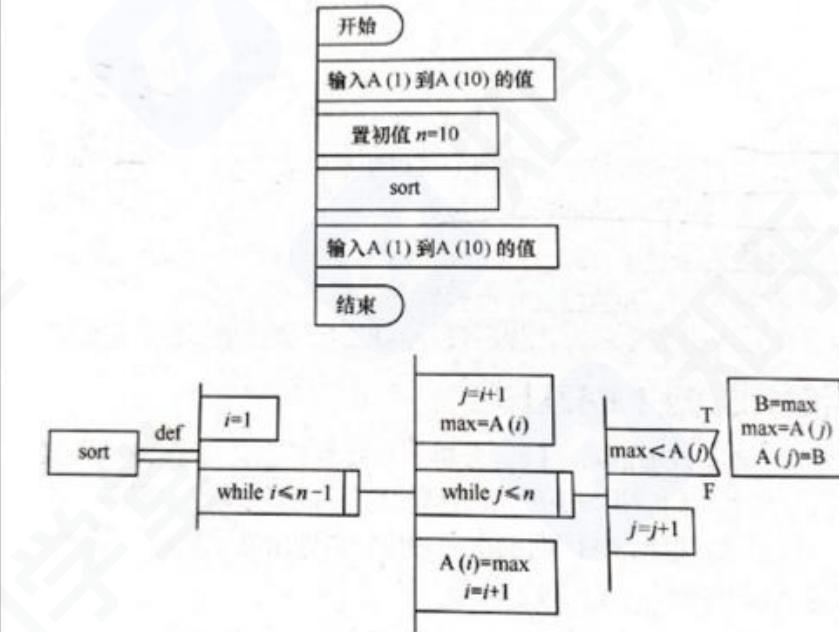
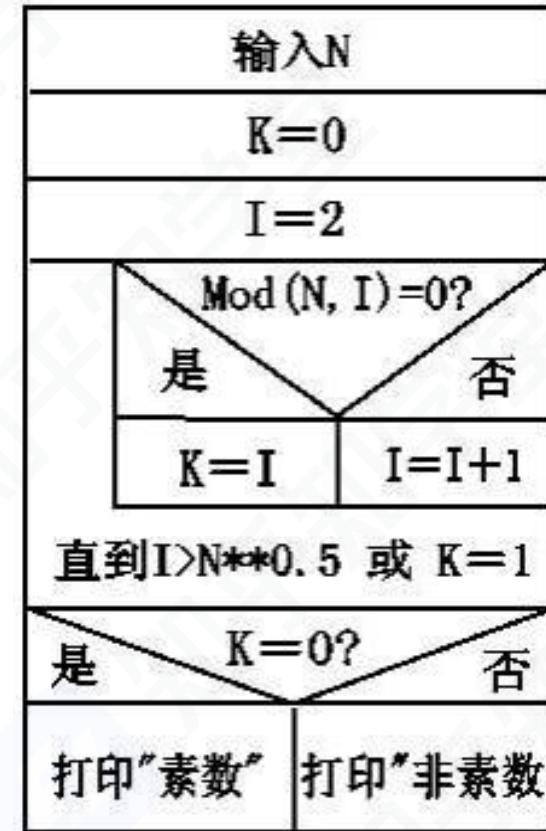


图 5.4 PAD 图示例

系统测试：为了发现错误而执行程序的过程，成功的测试是发现了至今尚未发现的错误的测试。

测试原则：

- 应尽早并不断的进行测试，比如V模型，从设计的时候就开始测试；
- 测试工作应该避免由原开发软件的人或小组承担；
- 在设计测试方案时，不仅要确定输入数据，而且要根据系统功能确定预期的输出结果；
- 既包含有效、合理的测试用例，也包含不合理、失效的用例；
- 检验程序是否做了该做的事，且是否做了不该做的事；
- 严格按照测试计划进行；
- 妥善保存测试计划和测试用例；
- 测试用例可以重复使用或追加测试。

软件测试方法可分为**静态测试**和**动态测试**。

静态测试: 指被测试程序不在机器上运行，而采用人工检测和计算机辅助静态分析的手段对程序进行检测，包括对文档的静态测试和对代码的静态测试。包括**桌前检查**、**代码审查**、**代码走查**的方式。使用这种方法能够有效地发现30%-70%的逻辑设计和编码错误。

- **桌前检查**: 在你自己编写代码后，你会进行桌前检查，这意味着你会仔细查看你的代码，以捕捉可能的错误、逻辑问题或风格不一致之类的问题。这是一种个人层面的检查，有助于在代码提交前修复问题。**代码审查**: 代码审查是团队中的成员一起对某个人编写的代码进行检查。通过代码审查，团队可以共同评估代码的质量，发现潜在的问题，并确保代码符合团队的标准和最佳实践。这有助于提高代码的稳定性和可维护性。
- **代码走查**: 代码走查是一种更广泛的审查实践，通常包括团队的开发人员、测试人员和其他相关人员。在代码走查过程中，团队会深入检查代码的各个方面，包括逻辑、性能、安全性等。目标是确保代码在整体上是健壮、高效且符合预期要求的。

动态测试：指在计算机上实际运行程序进行软件测试，一般采用白盒测试和黑盒测试方法(还有灰盒和自动化)。

- 黑盒测试：黑盒测试关注于测试软件的功能(功能性测试)，而不考虑内部实现细节。测试人员不需要知道代码的具体结构，而是根据软件的需求规格和功能来设计测试用例。这种方法类似于将测试人员置于一个“盒子”外，只观察软件的输入和输出，以确定是否按预期工作。
- 举例：假设你正在测试一个在线登录系统。对于黑盒测试，你会设计测试用例，包括输入不同的用户名和密码组合，然后观察系统的响应，验证是否成功登录、失败登录是否有适当的错误提示等，而不考虑系统内部的代码结构。
- 白盒测试：白盒测试关注于测试软件的内部逻辑和代码结构(结构性测试)，以确保代码按照预期方式执行。测试人员需要了解软件的代码，以设计测试用例，以覆盖不同的代码路径和分支情况，以及验证代码是否满足质量标准和最佳实践。
- 举例：假设你正在测试一个计算器应用。对于白盒测试，你会检查代码，确保加法、减法、乘法和除法等操作都正确实现。你可能会编写测试用例，测试各种输入情况，例如测试正数、负数、小数等，以确保代码在不同情况下都能正确执行。

1、单元测试:

- 也称为模块测试、算法测试。
- 测试的对象是可独立编译或汇编的程序模块、软件构件或OO软件中的类(统称为模块)。
- 测试依据是软件详细设计说明书。
- 侧重于测试模块中的内部逻辑和数据结构。
- 一般会采用白盒测试。

2、集成测试:

- 目的是检查模块之间，以及模块和已集成的软件之间的接口关系，并验证已集成的软件是否符合设计要求。
- 测试依据是软件概要设计文档。

3、系统测试：

- 测试对象是完整的、集成的计算机系统；
- 测试的目的是在真实系统工作环境下，验证完成的软件配置项能否和系统正确连接，并满足系统/子系统设计文档和软件开发合同规定的要求。
- 测试依据是用户需求或开发合同。
- 主要内容包括功能测试、健壮性测试、性能测试、用户界面测试、安全性测试、安装与反安装测试等，其中，最重要的工作是进行功能测试与性能测试。
 - 功能测试主要采用黑盒测试方法；
 - 性能测试主要指标有响应时间、吞吐量、并发用户数和资源利用率等。
- 系统测试通常由独立的测试团队执行，他们并不直接参与软件的开发过程。

4、确认测试：

- 主要用于验证软件的功能、性能和其他特性是否与用户需求一致。
- 测试依据是需求文档，确认测试是软件或产品开发的最后一个阶段，在系统测试完成后进行。
- 主要目标是确保软件或产品已经满足最终用户的期望和需求。
- 在确认测试中，最终用户（或其代表）将根据他们的实际使用情境，验证软件是否符合他们的业务流程和预期目标
根据用户的参与程度，通常包括以下类型：
 - 内部确认测试：主要由软件开发组织内部按照SRS进行测试。
 - Alpha测试：用户在开发环境下进行测试。
 - Beta测试：用户在实际使用环境下进行测试，通过改测试后，产品才能交付用户。
 - 验收测试：针对SRS，在交付前以用户为主进行的测试。其测试对象为完整的、集成的计算机系统。验收测试的目的是，在真实的用户工作环境下，检验软件系统是否满足开发技术合同或SRS。验收测试的结论是用户确定是否接收该软件的主要依据。除应满足一般测试的准入条件外，在进行验收测试之前，应确认被测软件系统已通过系统测试。



5、回归测试

- 测试目的是测试软件变更之后，变更部分的正确性和对变更需求的符合性，以及软件原有的、正确的功能、性能和其他规定的要求的不损害性(不能把其他好的模块改错)。

黑盒测试：将程序看做一个黑盒子，只知道输入输出，不知道内部代码，由此设计出测试用例，分为下面几类：

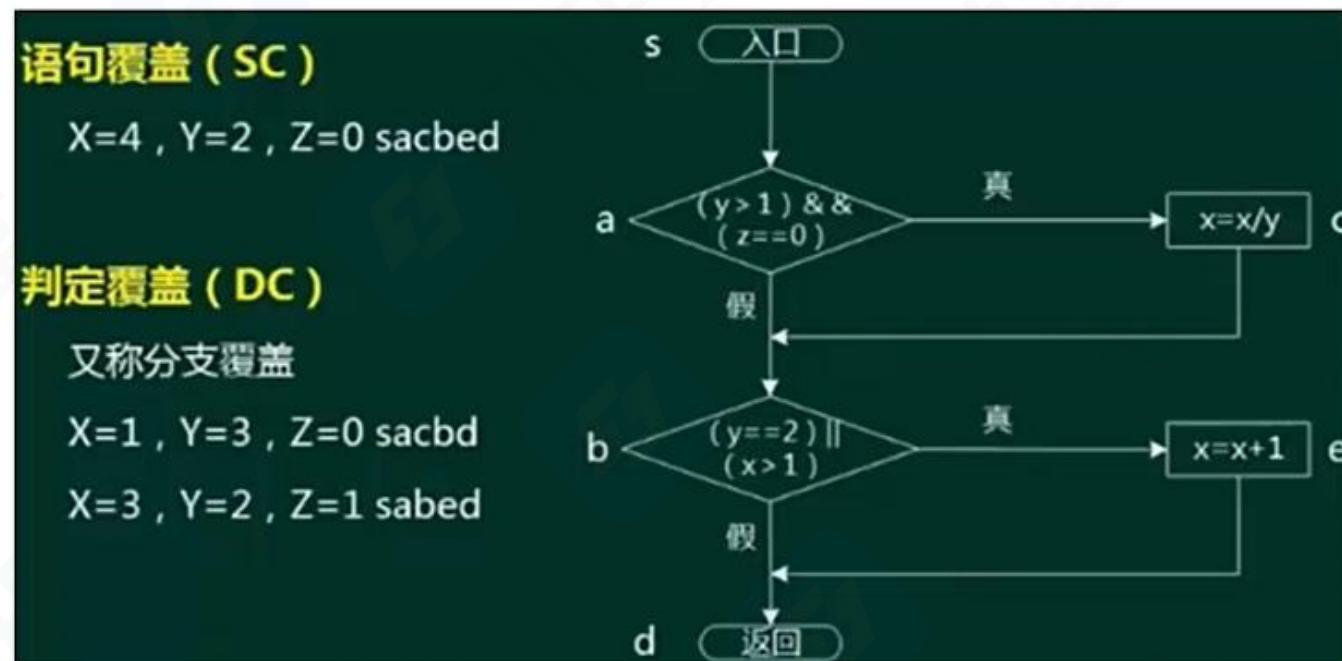
- 等价类划分：一种常用的黑盒测试技术，**用于减少测试用例的数量**，同时保证测试覆盖到尽可能多的情况。它通过将输入数据分为若干个等价类，认为每个等价类中的所有数据对系统的行为应该是等效的。这样，只需为每个等价类选择一个代表性的测试用例，而不是测试所有可能的输入。
- 边界值划分：**将每类的边界值作为测试用例**，边界值一般为范围的两端值以及在此范围之外的与此范围间隔最小的两个值，如年龄范围为0-150, 边界值为0,150,-1,151四个。
- 错误推测：没有固定的方法，**凭经验而言**，来推测有可能产生问题的地方，作为测试用例进行测试。
- 因果图：由一个**结果来反推原因的方法**，从自然语言描述的程序规格说明中找出因（输入条件）和果（输出或程序状态的改变），通过因果图转换为判定表。

2、某航空公司规定，乘客可以免费托运重量不超过 30 公斤的行李。当行李重量超过 30 公斤时，对头等舱的国内乘客超重部分每公斤收费 4 元，对其他舱的国内乘客超重部分每公斤收费 6 元，对外国乘客超重部分每公斤收费比国内乘客多一倍。根据描述绘出判定表。

决策规则号		1	2	3	4	5	6	7	8
条件	行李重量 $W \leq 30$	Y	Y	Y	Y	N	N	N	N
	国内乘客	Y	Y	N	N	Y	Y	N	N
应采取的行动	头等舱	Y	N	Y	N	Y	N	Y	N
	免费	X	X	X	X				
$(W - 30) \times 4$								X	
$(W - 30) \times 6$									X
$(W - 30) \times 8$									X
$(W - 30) \times 12$									X

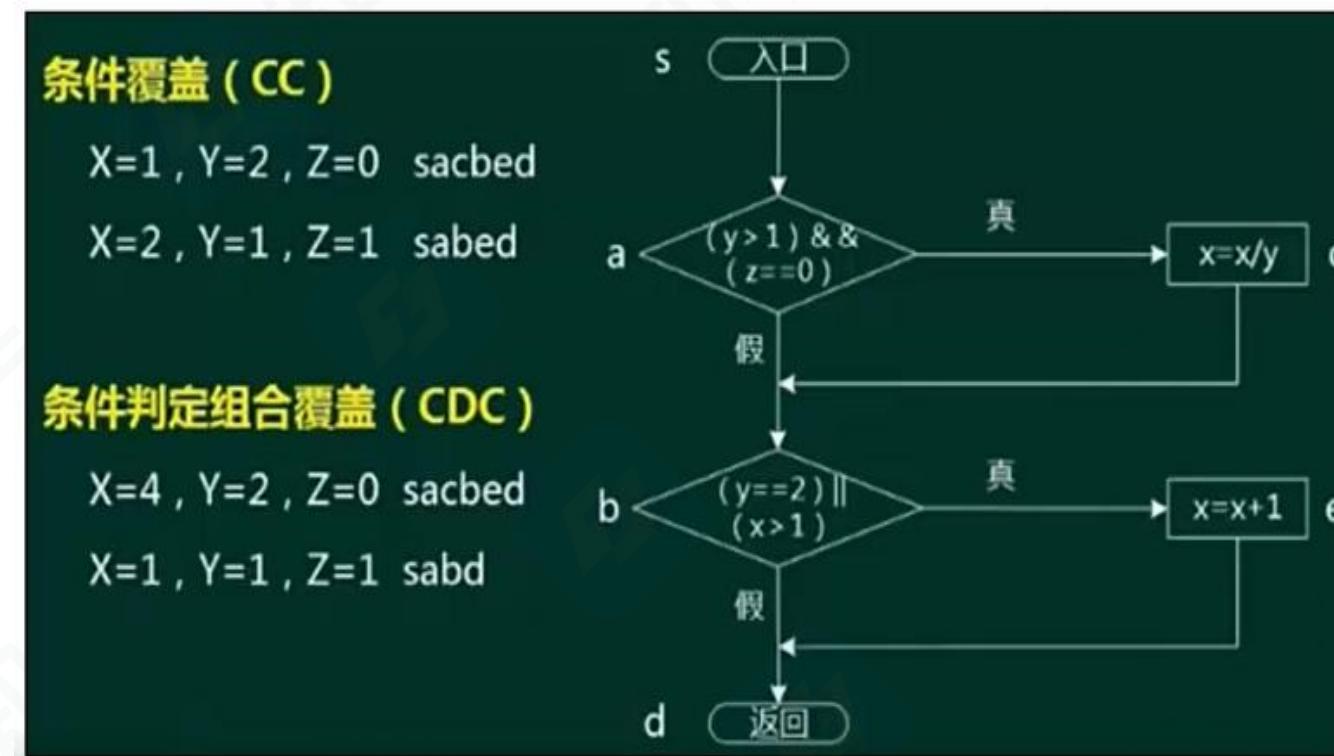
白盒测试：知道程序的代码逻辑，按照程序的代码语句，来设计覆盖代码分支的测试用例，覆盖级别从低至高分为下面几种：

- 语句覆盖SC：逻辑代码中的所有语句都要被执行一遍，覆盖层级最低，因为执行了所有的语句，不代表执行了所有的条件判断。
- 判定覆盖DC：逻辑代码中的所有判断语句的条件的真假分支都要覆盖一次。



白盒测试：

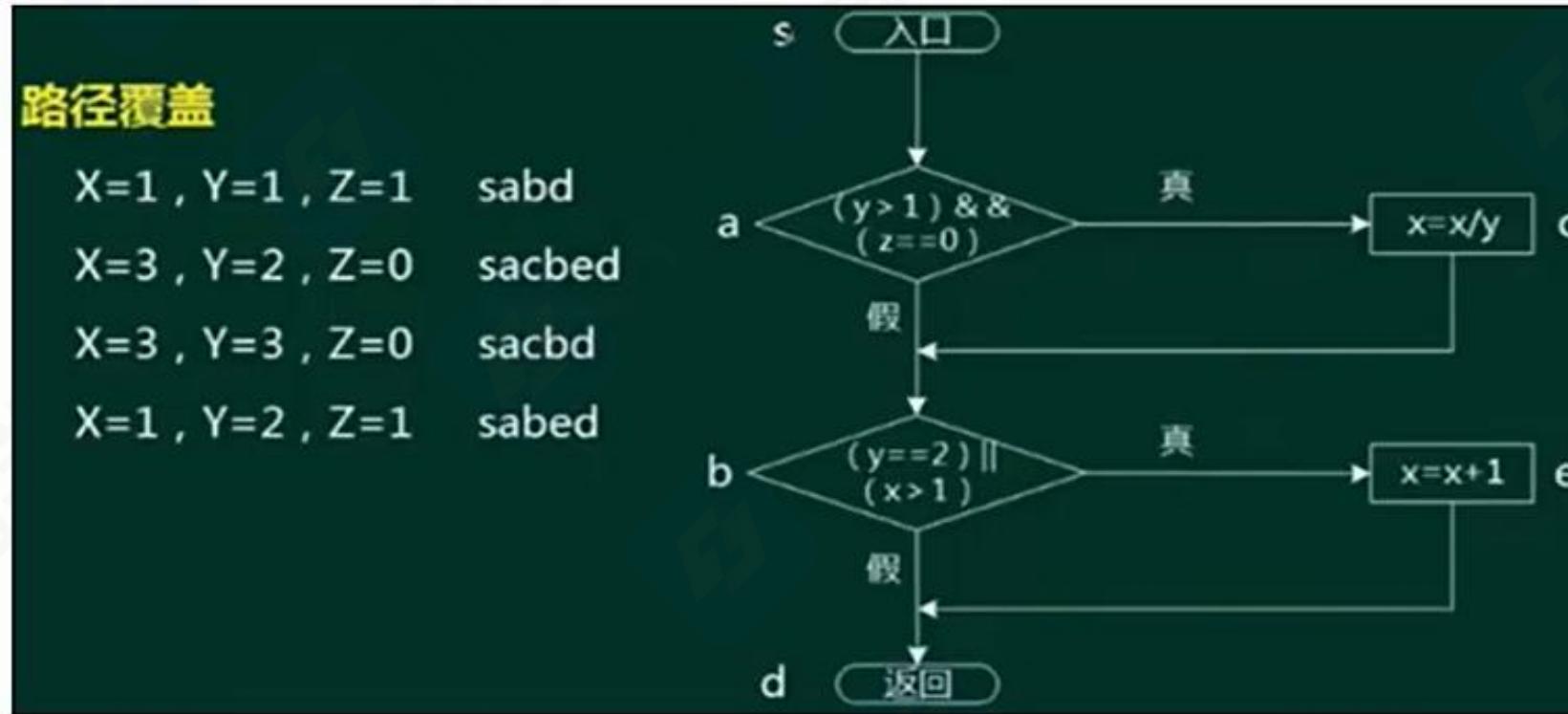
- 条件覆盖CC：针对每一个判断条件内(比如一个if)的每一个独立条件(if中的每个判定条件)都要执行一遍真和假。
- 条件判定组合覆盖CDC：同时满足判定覆盖和条件覆盖。





白盒测试：

- 路径覆盖：逻辑代码中的所有可行路径都覆盖了，覆盖层级最高。



白盒测试技术的各种覆盖方法中，(33) 具有最弱的错误发现能力。

- (33) A. 判定覆盖 B. 语句覆盖 C. 条件覆盖 D. 路径覆盖

针对月收入小于等于3500元免征个人所得税的需求，现分别输入3499,3500和3501进行测试，则采用的测试方法(16)。

- (16) A、判定覆盖 B、边界值分析 C、路径覆盖 D、因果图

系统转换是指新系统开发完毕，投入运行，取代现有系统的过程，需要考虑多方面的问题，以实现与老系统的交接，有以下三种转换计划：

- 直接转换：现有系统被新系统直接取代了，风险很大，适用于新系统不复杂，或者现有系统已经不能使用的情况。优点是节省成本，只适合小系统。
- 并行转换：新系统和老系统并行工作一段时间，新系统经过试运行后再取代，若新系统在试运行过程中有问题，也不影响现有系统的运行，风险极小，在试运行过程中还可以比较新老系统的性能，适用于大型系统。缺点是耗费人力和时间资源，难以控制两个系统间的数据转换。
- 分段转换：分期分批逐步转换，是直接和并行转换的集合，将大型系统分为多个子系统，依次试运行每个子系统，成熟一个子系统，就转换一个子系统。同样适用于大型项目，只是更耗时，而且现有系统和新系统间混合使用，需要协调好接口等问题。

数据转换与迁移：将数据从旧数据库迁移到新数据库中。有三种方法：系统切换前通过工具迁移、系统切换前采用手工录入、系统切换后通过新系统生成。

系统维护是整个系统开发过程中**耗时最长的**，系统的可维护性可以定义为维护人员理解、改正、改动和改进这个软件的难易程度，其评价指标如下：

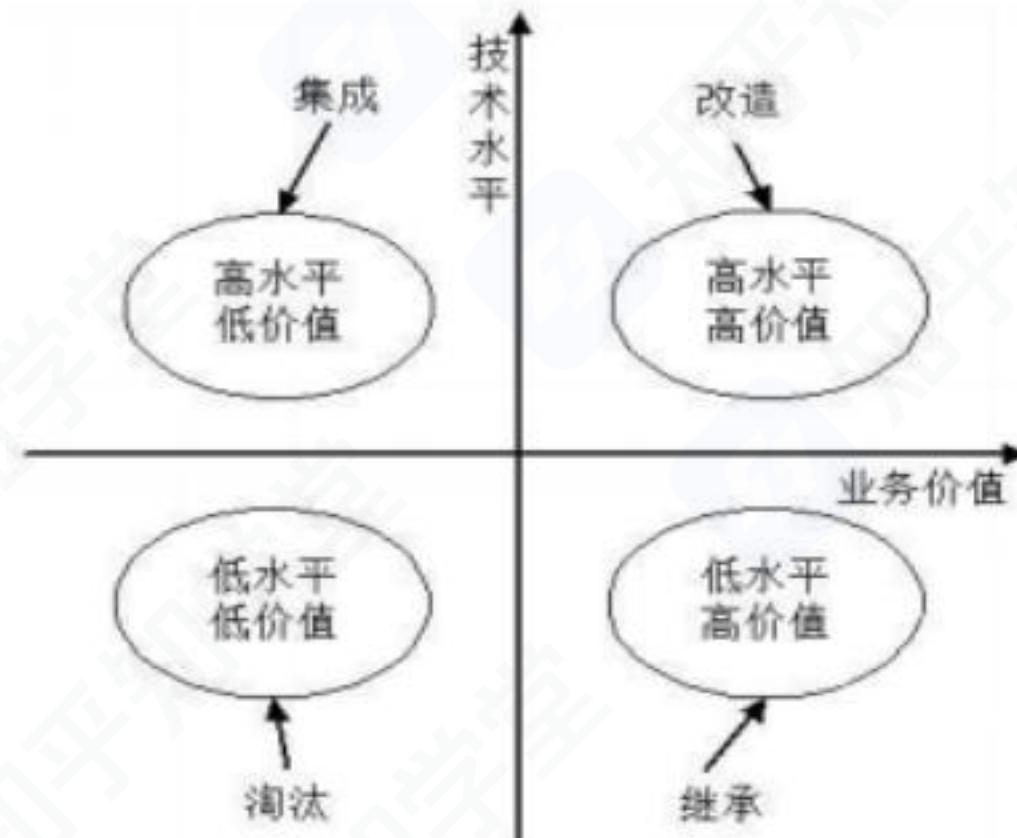
- 易分析性。软件产品诊断软件中的缺陷或失效原因或识别待修改部分的能力。
- 易改变性。软件产品使指定的修改可以被实现的能力，实现包括编码、设计和文档的更改。
- 稳定性。软件产品避免由于软件修改而造成意外结果的能力。
- 易测试性。软件产品使已修改软件能被确认的能力。

系统维护包括**硬件维护、软件维护和数据维护**，其中软件维护类型如下：

- **正确性维护**：发现了bug而进行的修改。
- **适应性维护**：由于外部环境发生了改变，被动进行的对软件的修改和升级。
- **完善性维护**：基于用户主动对软件提出更多的需求，修改软件，增加更多的功能，使其比之前的软件功能、性能更高，更加完善。
- **预防性维护**：对未来可能发生的问题进行预防性的修改。

遗留系统是指任何基本上不能进行修改和演化以满足新的变化了的业务需求的信息系统，它通常具有以下特点：

- 系统虽然完成企业中许多重要的业务管理工作，但仍然不能完全满足要求。一般实现业务处理电子化及部分企业管理功能，很少涉及经营决策。
- 系统在性能上已经落后，采用的技术已经过时。例如，多采用主机/终端形式或小型机系统，软件使用汇编语言或第三代程序设计语言的早期版本开发，使用文件系统而不是数据库。
- 通常是大型的软件系统，已经融入企业的业务运作和决策管理机制之中，维护工作十分困难。
- 没有使用现代信息系统建设方法进行管理和开发，现在基本上已经没有文档，很难理解。



净室软件工程(CSE): 是一种应用数学与统计学理论以经济的方式生产高质量软件的工程技术，力图通过严格的工程化的软件过程达到开发中的零缺陷或接近零缺陷，强调的是预防大于检查。净室方法不是先制作一个产品，再去消除缺陷，而是要求在规约和设计中消除错误，然后以“净”的方式制作，可以降低软件开发中的风险，以合理的成本开发出高质量的软件。

在净室软件工程背后的哲学是: 通过在第1次正确地书写代码增量，并在测试前验证它们的正确性，来避免对成本很高的错误消除过程的依赖。它的过程模型是在代码增量积聚到系统的过程的同时，进行代码增量的统计质量验证。它甚至提倡开发者不需要进行单元测试，而是进行正确性验证和统计质量控制。

净室软件工程(CSE)的理论基础主要是函数理论和抽样理论。

净室软件工程应用技术手段:

- 统计过程控制下的增量式开发。
- 基于函数的规范与设计。
- 正确性验证 (CSE的核心)
- 统计测试和软件认证。

净室软件工程在使用过程的一些缺点:

- CSE太理论化，需要更多的数学知识。其正确性验证的步骤比较困难且比较耗时。
- CSE开发小组不进行传统的模块测试，这是不现实的。
- CSE也会带有传统软件工程的一些弊端。

基于构件的软件工程(CBSE): 是一种基于分布对象技术、强调通过可复用构件设计与构造软件系统的软件复用途径。CBSE体现了“购买而不是重新构造”的哲学，将软件开发的重点从程序编写转移到了基于已有构件的组装。用于CBSE的构件应该具备以下特征。

- 可组装型：对于可组装的构件，所有外部交互必须通过公开定义的接口进行。同时它还必须对自身信息的外部访问。
- 可部署性：软件必须是自包含的，必须能作为一个独立实体在提供其构件模型实现的构件平台上运行。构件总是二进制形式，无须在部署前编译。
- 文档化：构件必须是完全文档化的，用户根据文档来判断构件是否满足需求。
- 独立性：构件应该是独立的，应该可以在无其他特殊构件的情况下进行组装和部署，如确实需要其他构件提供服务，则应显示声明。
- 标准化：构件标准化意味着在CBSE过程中使用的构件必须符合某种标准化的构件模型。构件模型定义了构件实现、文档化以及开发的标准，其包含的模型要素为：
 - 接口。构件通过构件接口来定义，构件模型规定应如何定义构件接口以及在接口定义中应该包含的要素，如操作名、参数以及异常等。
 - 使用信息。为使构件远程分布和访问，必须给构件一个特定的、全局唯一的名字或句柄。构件元数据是构件本身相关的数据，比如构件的接口和属性信息。
 - 部署。构件模型包括一个规格说明，指出应该如何打包构件使其部署成为一个独立的可执行实体。部署信息中包含有关包中内容的信息和它的二进制构成的信息。

构件模型提供了一组被构件使用的通用服务，这种服务包括以下两种。

- 平台服务，允许构件在分布式环境下通信和互操作。
- 支持服务，这是很多构件需要的共性服务。例如，构件都需要的身份认证服务、需要GPS服务之类。
- 中间件实现共性的构件服务，并提供这些服务的接口。

CBSE过程是支持基于构件组装的软件开发过程，过程中的**6个主要活动**：**系统需求概览、识别候选构件、根据发现的构件修改需求、体系结构设计、构件定制与适配、组装构件创建系统。**

CBSE过程与传统软件开发过程不同点：

- CBSE早期需要完整的需求(即需求明确)，以便尽可能多地识别出可复用的构件。
- 在过程早期阶段根据可利用的构件来细化和修改需求。如果可利用的构件不能满足用户需求，就应该考虑由复用构件支持的相关需求。
- 在系统体系统结构设计完成后，会有一个进一步的对构件搜索及设计精化的活动。可能需要为某些构件寻找备用构件，或者修改构件以适合功能和架构的要求。
- 开发就是将已经找到的构件集成在一起的组装过程。

构件组装是指构件相互直接集成或是用专门编写的“胶水代码”将它们整合在一起创造一个系统或另一个构件的过程。常见的组装构件有以下**3种**组装方式。

- **顺序组装**。通过按顺序调用已经存在的构件，可以用两个已经存在的构件来创造一个新的构件。如上一个构件输出作为下一个构件的输入。
- **层次组装**。这种情况发生在一个构件直接调用自另一个构件所提供的服务时。被调用的构件为调用的构件提供所需的服务。二者之间接口匹配兼容。
- **叠加组装**。这种情况发生在两个或两个以上构件放在一起创建一个新构件的时候。这个新构件合并了原构件的功能，从而对外提供了新的接口。外部应用可以通过新接口来调用原有构件的接口，而原有构件不互相依赖，也不互相调用。这种组装类型适合于构件是程序单元或者构件是服务的情况。

构件组装的三种不兼容问题(通过编写适配器解决):

- 参数不兼容。接口每一侧的操作有相同的名字，但参数类型或参数个数不相同。
- 操作不兼容。提供接口和请求接口的操作名不同。
- 操作不完备。一个构件的提供接口是另一个构件请求接口的一个子集，或者相反。



02

面向对象技术



- 本章节在历年考试过程中的分值占比大概是3-4分
- 本章节在新版教材里对应的是5.3.2以及2.6.2中的UML语言，与老版本的单独一章面向对象设计相比，减少了很多的内容，比如UML图，设计模式等，但是在改版之后的2次考试来看，仍然考到了老版本的知识
- 被考到知识点有：
 - 面向对象基本概念，面向对象分析
 - UML概述、关系、图
 - 设计模式
- 在改版之后的考试中，分别考察的知识点为：
 - 2023年11月：sysml需求图
 - 2024年05月：面向对象分析、UML构件图、设计模式
 - 2025年11月：UML图（2分）、设计模式（2分）

对象：基本的运行实体，为类的实例，封装了数据和行为的整体，如学生、汽车等真实存在的实体。对象具有清晰的边界、良好定义的行为和可扩展性。

消息：对象之间进行通信的一种方式称为消息。

类：是对象的抽象，定义了一组大体相似的对象结构，定义了数据和行为。

- 实体类：表示系统中的业务数据及其相关操作，通常对应现实世界中的实体对象，比如书本，人
- 边界类：作为系统与外部世界（如用户界面、外部系统）之间交互的接口，比如用户界面
- 控制类：负责实现系统的业务逻辑，处理数据流和控制应用程序的流程，比如用户认证



继承：父类和子类之间共享数据和方法的机制，是类与类之间的一种关系。

多态：通俗来说，就是多种形态，具体点就是去完成某个行为，当不同的对象去完成时会产生出不同的状态。在编程语言和类型论中，多态指为不同数据类型的实体提供统一的接口，多态由继承机制支持。包括以下四种多态：

- 参数多态：不同类型参数多种结构类型
- 包含多态：父子类型关系
- 过载多态：类似于重载，一个名字不同含义
- 强制多态：强制类型转换

覆盖（重写）：子类在原有父类接口的基础上，用适合于自己要求的实现去置换父类中的相应实现。即在子类中重定义一个与父类同名同参的方法。

重载：与覆盖要区分开，函数重载与子类父类无关，且函数是同名不同参数。

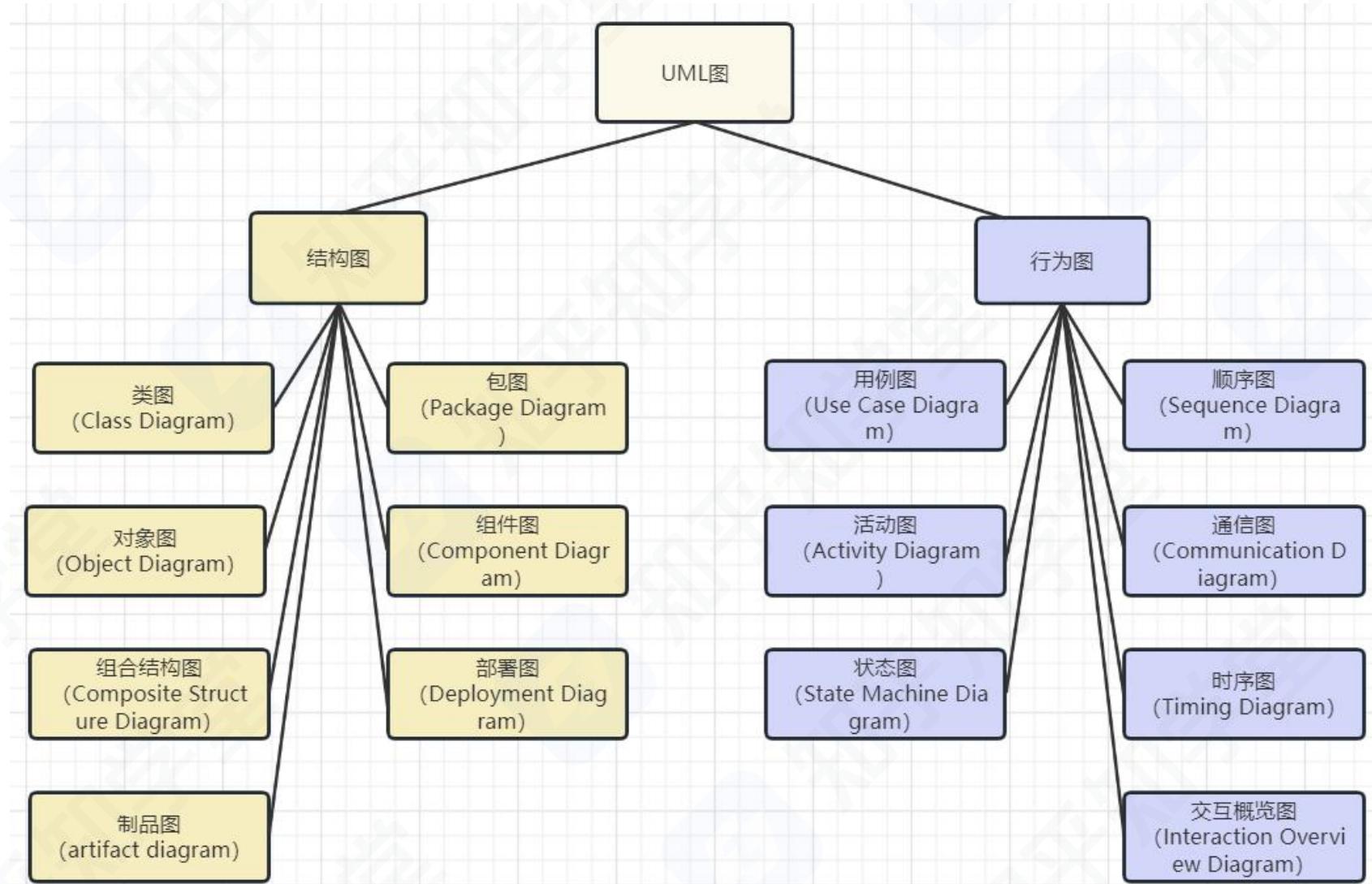
封装：一种**信息隐蔽**技术，其目的是使对象的使用者和生产者分离，也就是使其他开发人员无需了解所要使用的软件组件内部的工作机制，只需知道如何使用组件。

静态绑定是基于静态类型（类型在**编译时**就确定），在程序执行前方法已经被绑定；

动态绑定是基于动态类型（类型在**运行时**才能确定），运行时根据变量实际引用的对象类型决定调用哪个方法，动态绑定支持多态。



设计原则名称	定 义	使用频率
单一职责原则 (Single Responsibility Principle, SRP)	一个对象应该只包含单一的职责，并且该职责被完整地封装在一个类中	★★★★★☆
开闭原则 (Open-Closed Principle, OCP)	软件实体应当对扩展开放，对修改关闭	★★★★★
里氏代换原则 (Liskov Substitution Principle, LSP)	所有引用基类的地方必须能透明地使用其子类的对象	★★★★★
依赖倒转原则 (Dependence Inversion Principle, DIP)	高层模块不应该依赖低层模块，它们都应该依赖抽象。抽象不应该依赖于细节，细节应该依赖于抽象	★★★★★
接口隔离原则 (Interface Segregation Principle, ISP)	客户端不应该依赖那些它不需要的接口	★★☆☆☆
合成复用原则 (Composite Reuse Principle, CRP)	优先使用对象组合，而不是继承来达到复用的目的	★★★★★☆
迪米特法则 (Law of Demeter, LoD)	每一个软件单位对其他的单位都只有最少的知识，而且局限于那些与本单位密切相关的软件单位	★★★☆☆

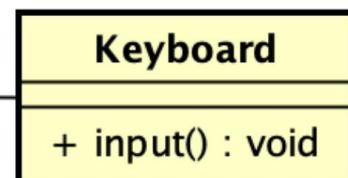
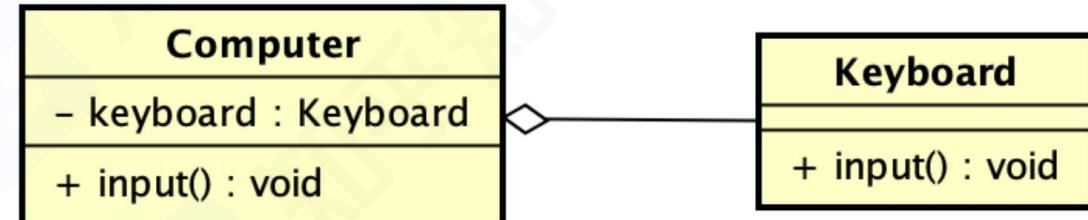
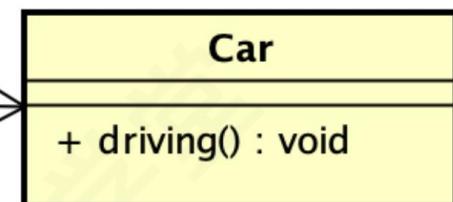
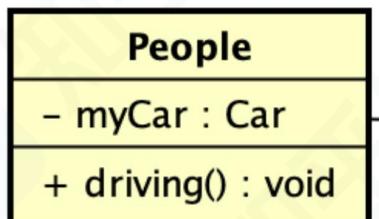
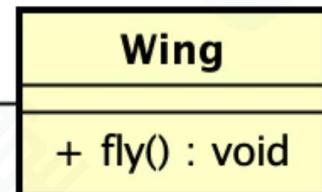
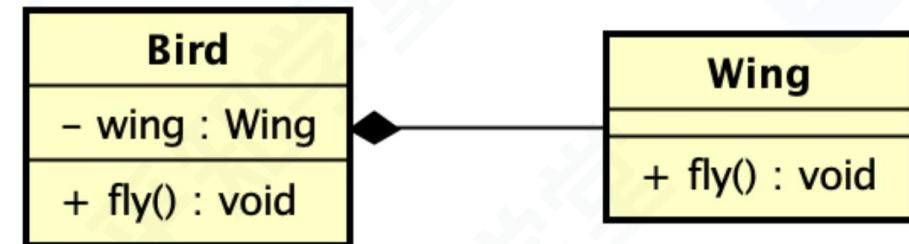
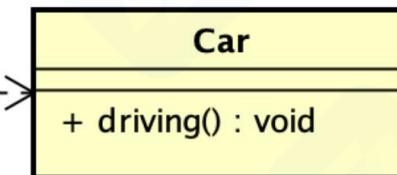
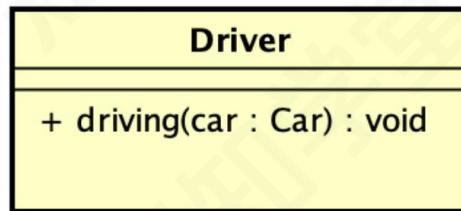




UML类型	目的	版本
类图	描述了系统中对象的类型以及它们之间存在的各种静态关系。	UML 1.x
组件图	描绘了系统中组件提供的、需要的接口、端口等，以及它们之间的关系。	UML 1.x, UML 2.0重新定义了
对象图	对象图是类图的一个实例，是系统在某个时间点的详细状态的快照。	UML 1.x
轮廓图	轮廓图提供了一种通用的扩展机制，用于为特定域和平台定制UML模型。	UML 2.0
组合结构图	描述了一个"组合结构"的内部结构，以及他们之间的关系。	UML 2.0
部署图	描述了系统内部的软件如何分布在不同的节点上。	UML 1.x
包图	描绘了系统在包层面上的结构设计。	UML 2.0
用例图	指由参与者、用例，边界以及它们之间的关系构成的用于描述系统功能的视图。	UML 1.x
活动图	描述了具体业务用例的实现流程。	UML 1.x
状态机图	描述了对象在它的整个生命周期里，响应不同事件时，执行相关事件的顺序。	UML 1.x
序列图	描述了在用例的特定场景中，对象如何与其他对象交互。	UML 1.x
时序图	时序图被用来显示随时间变化，一个或多个元素的值或状态的更改。	UML 2.0
交互概览图	交互概览图与活动图类似，但是它的节点是交互图。	UML 2.0
通讯图	描述了收发消息的对象的组织关系，强调对象之间的合作关系而不是时间顺序。	UML 1.x叫协作图，UML 2.0改名了



- 依赖：一个事物的语义依赖于另一个事物的语义的变化而变化，是一种使用关系，即一个类的实现需要另一个类的协助，普通箭头指向被使用者，比如Driver类指向Car类，代表Driver需要使用Car
- 关联：是一种拥有关系，它使得一个类知道另一个类的属性和方法。带普通箭头的实线，指向被拥有者。双向的关联可以有两个箭头，或者没有箭头。单向的关联有一个箭头。细分为组合和聚合
 - 聚合：是一种整体与部分的关系。且部分可以离开整体而单独存在。聚合关系是关联关系的一种，是强的关联关系；关联和聚合在语法上无法区分，必须考察具体的逻辑关系。
 - 组合：是一种整体与部分的关系。但部分不能离开整体而单独存在，组合关系是关联关系的一种，是比聚合关系还要强的关系



泛化：是一种继承关系，表示子类继承父类的所有特征和行为。

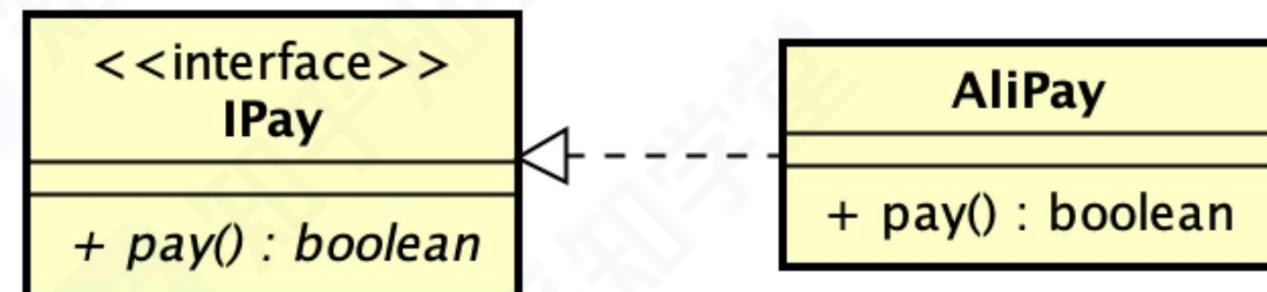
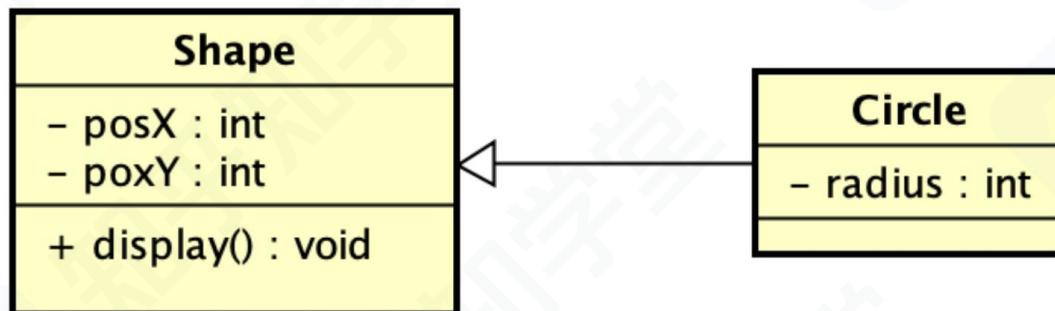
实现：是一种类与接口的关系，表示类是接口所有特征和行为的实现。



图 10-9 泛化



图 10-10 实现



类图：静态图，为系统的静态设计视图，展现一组对象、接口、协作和它们之间的关系。

多重度：指的是不同类之间的联系，类似于数据库设计的表与表的关系

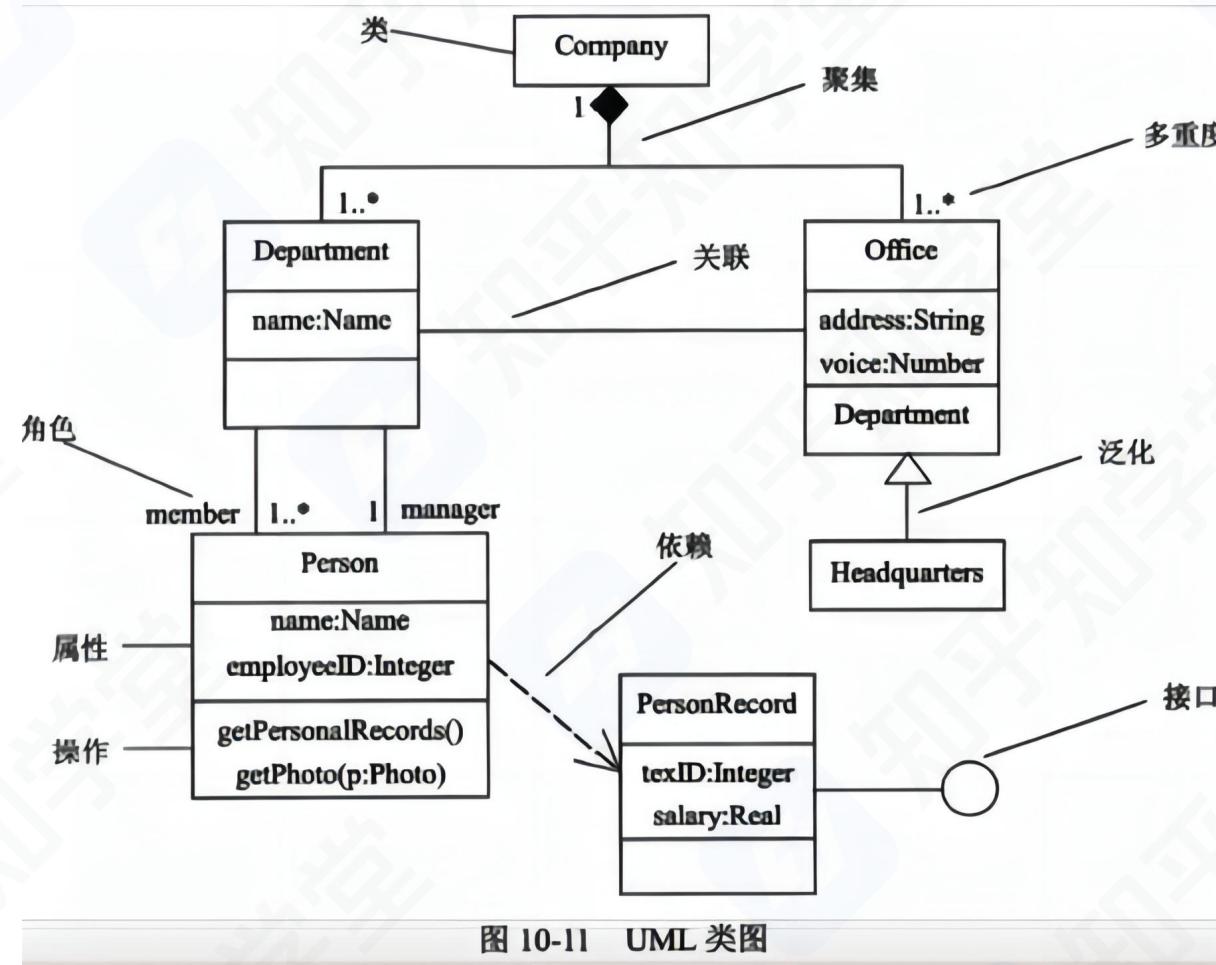
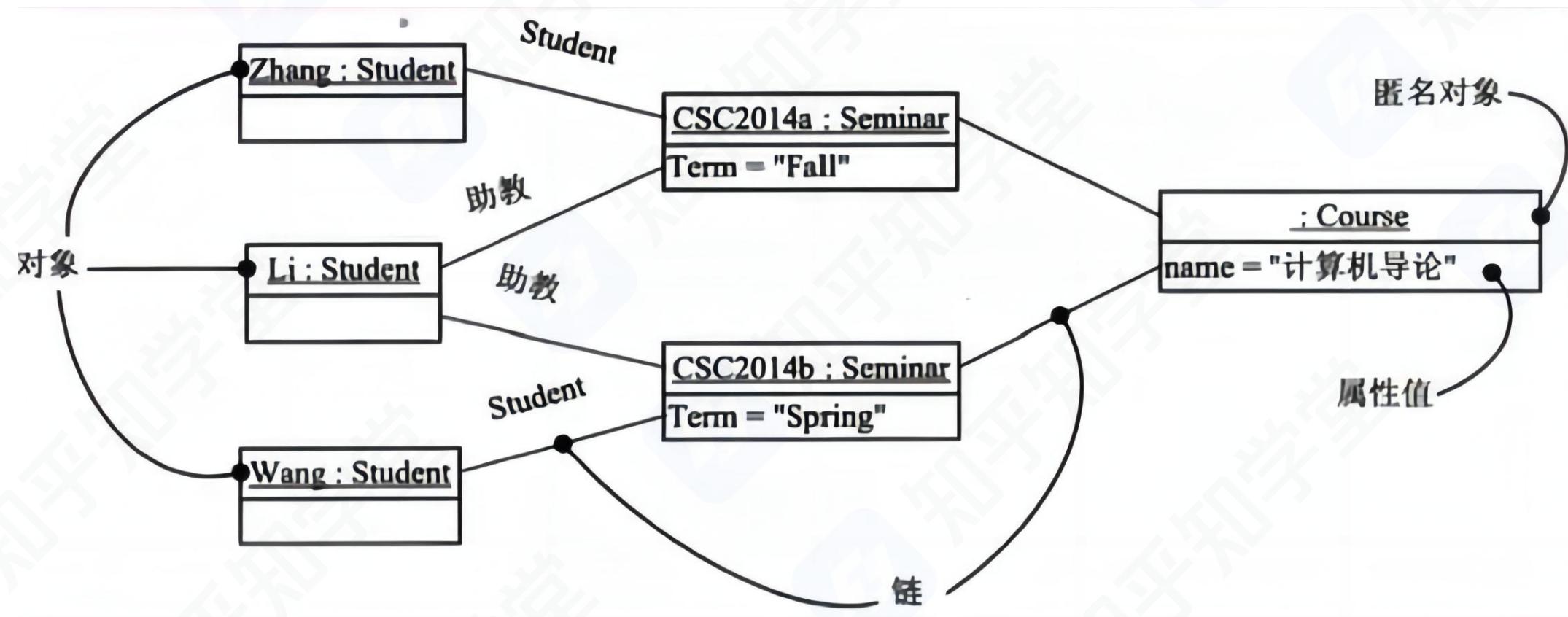


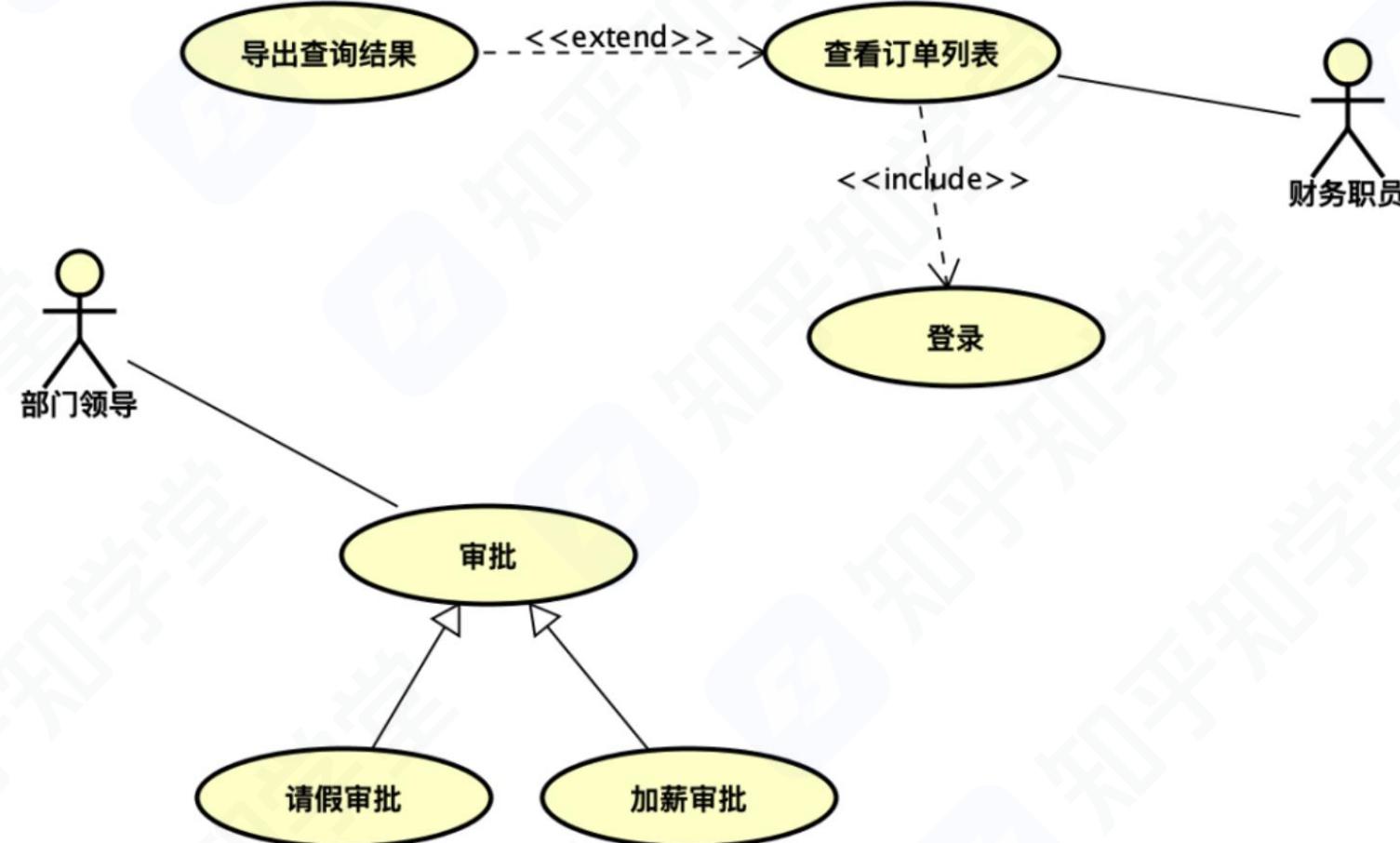
图 10-11 UML 类图

对象图：静态图，展现某一时刻一组对象及它们之间的关系，为类图的某一快照。在没有类图的前提下，对象图就是静态设计视图。



用例图：展现了一组用例、参与者以及它们之间的关系。

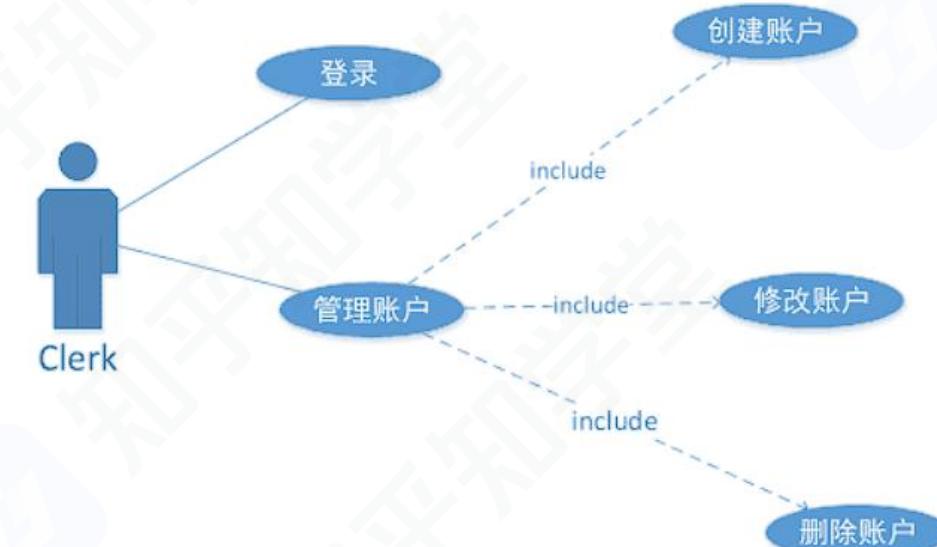
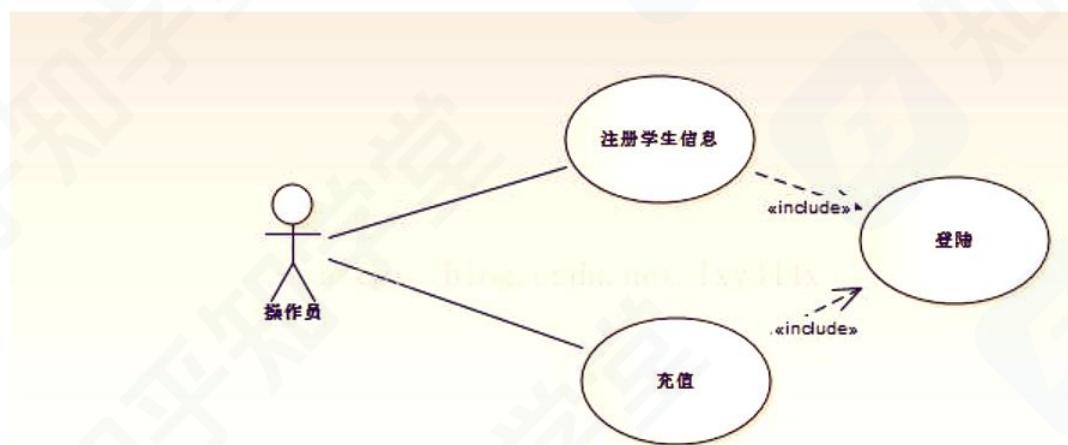
用例图中的参与者是人、硬件或其他系统可以扮演的角色；用例是参与者完成的一系列操作。用例之间的关系：包含 (include) 、扩展(extend)、泛化(generalize)。



(1) 当可以从两个或两个以上的用例中提取公共行为时，应该使用包含的关系来表示它们。其中这个提取出来的公用用例成为抽象用例，而把原始用例成为基本用例或基础用例。其中“<<include>>”是包含关系的构造型，箭头指向抽象用例。

例如，在机房收费系统中“注册学生信息”和“充值”两个用例都需要操作员或者管理员登陆，为此，可以定义一个抽象用例“用户登陆”。用例“注册学生信息”和“充值”与用例“用户登陆”之间的关系就是包含关系。

(2) 一个用例的功能太多时，可以使用包含关系建立若干个更小的用例。

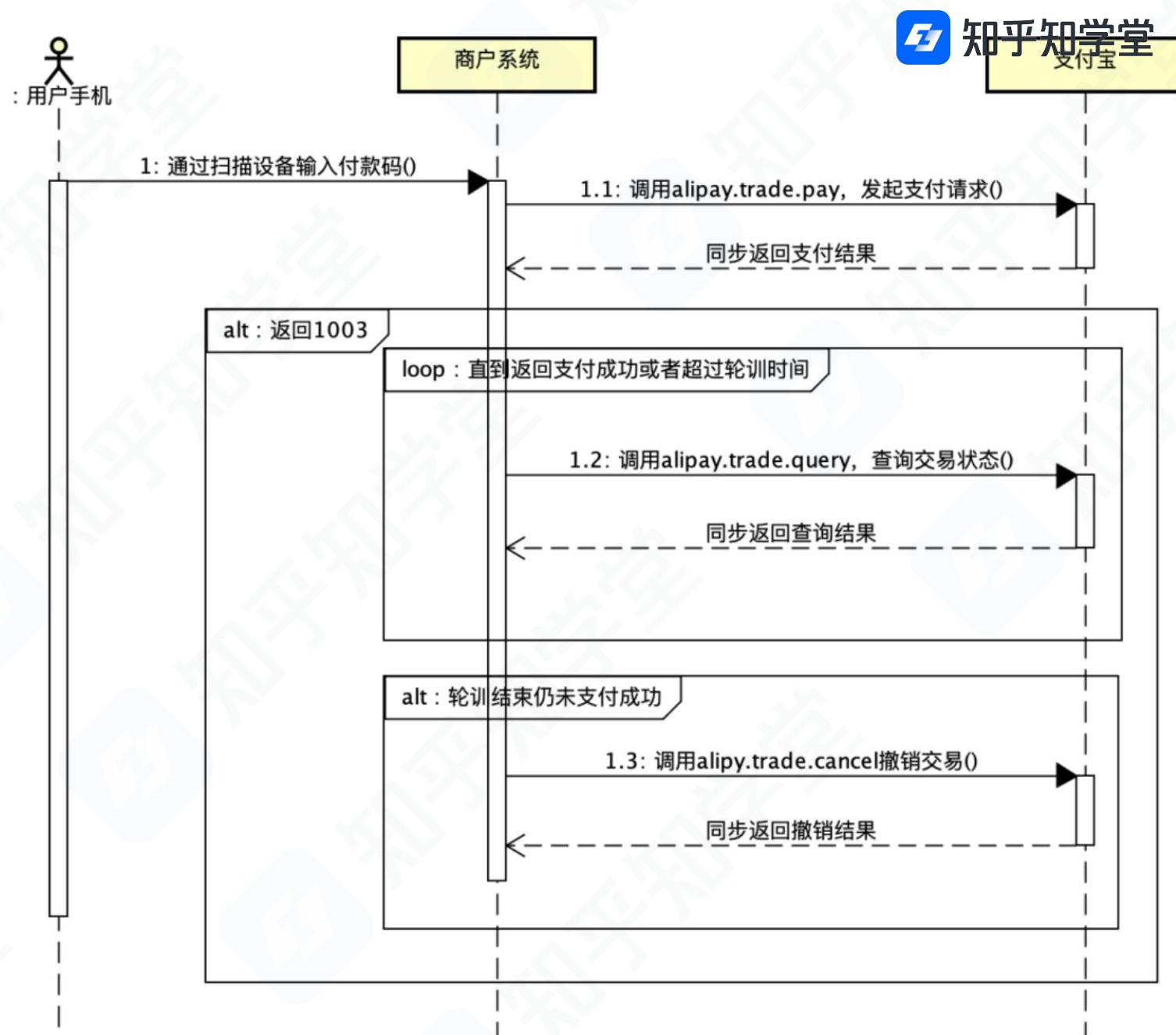


UML-序列图

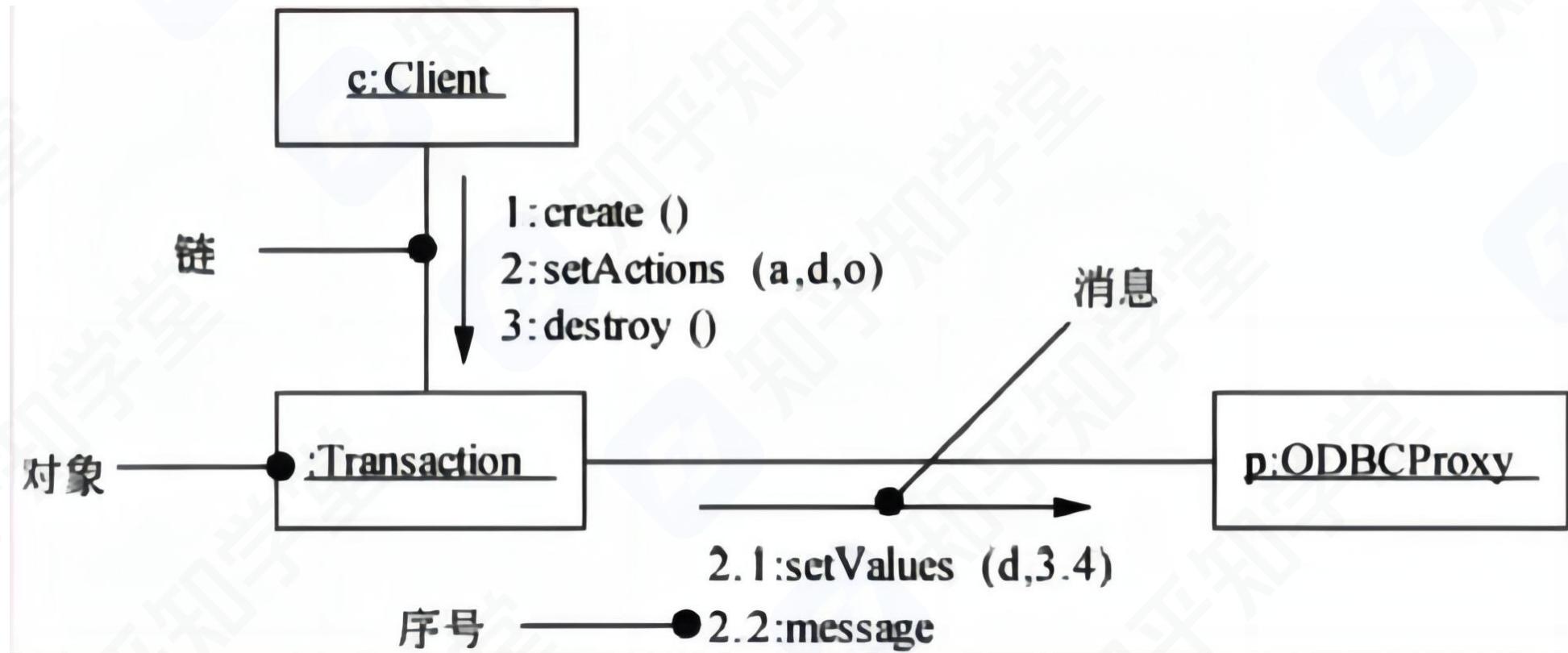
序列图：即顺序图，动态图，是场景的图形化表示，描述了以时间顺序组织的对象之间的交互活动。

- **同步消息**: 进行阻塞调用，调用者中止执行，等待控制权返回，需要等待返回消息，用实心三角箭头表示
- **异步消息**: 发出消息后继续执行，不引起调用者阻塞，也不等待返回消息，由空心箭头表示
- **返回消息**: 由从右到左的虚线箭头表示

注意：右图中展示的是支付宝条码支付场景的序列图。其中，loop是循环，alt是选择，序列图的其他关系这里就不介绍了。

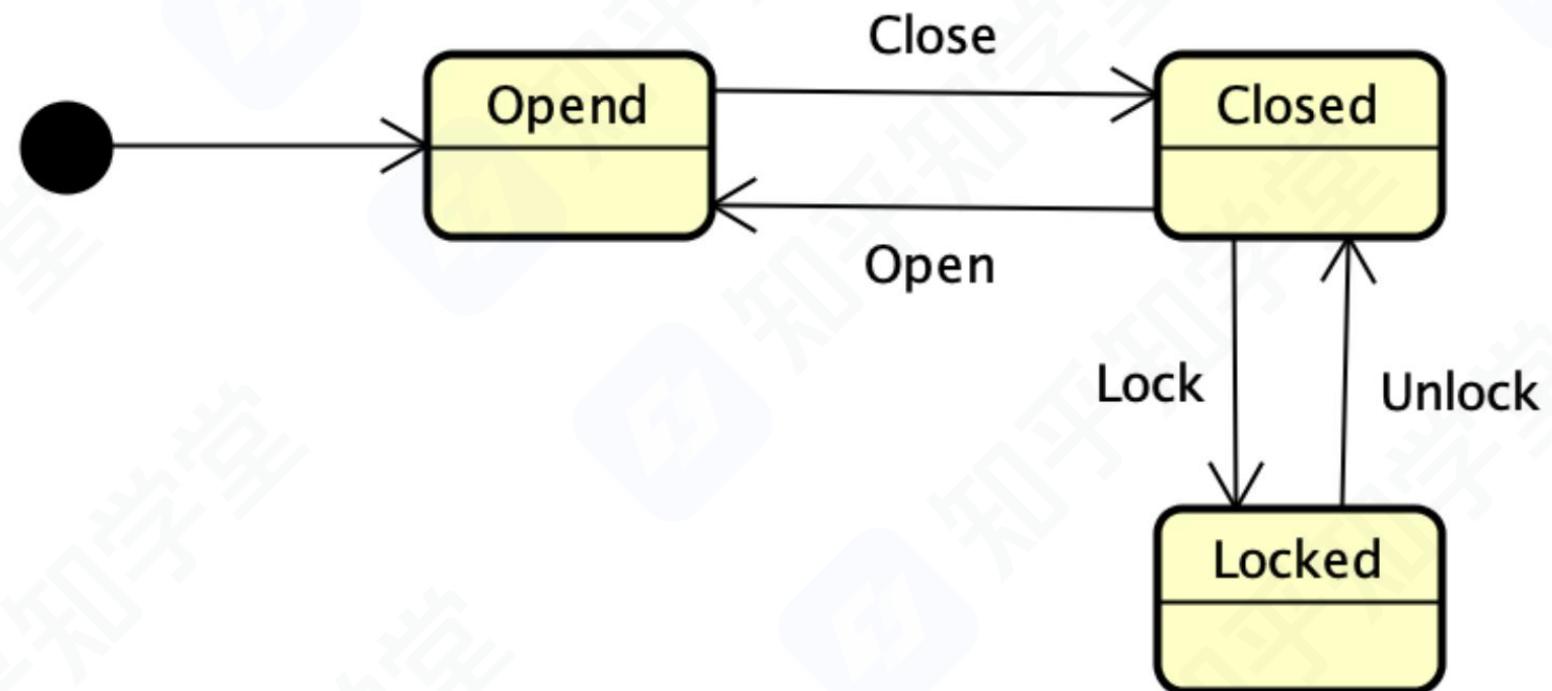


通信图：动态图，即协作图，是顺序图的另一种表示方法，也是由对象和消息组成的图，只不过不强调时间顺序，只强调事件之间的通信，而且也没有固定的画法规则，和顺序图统称为交互图



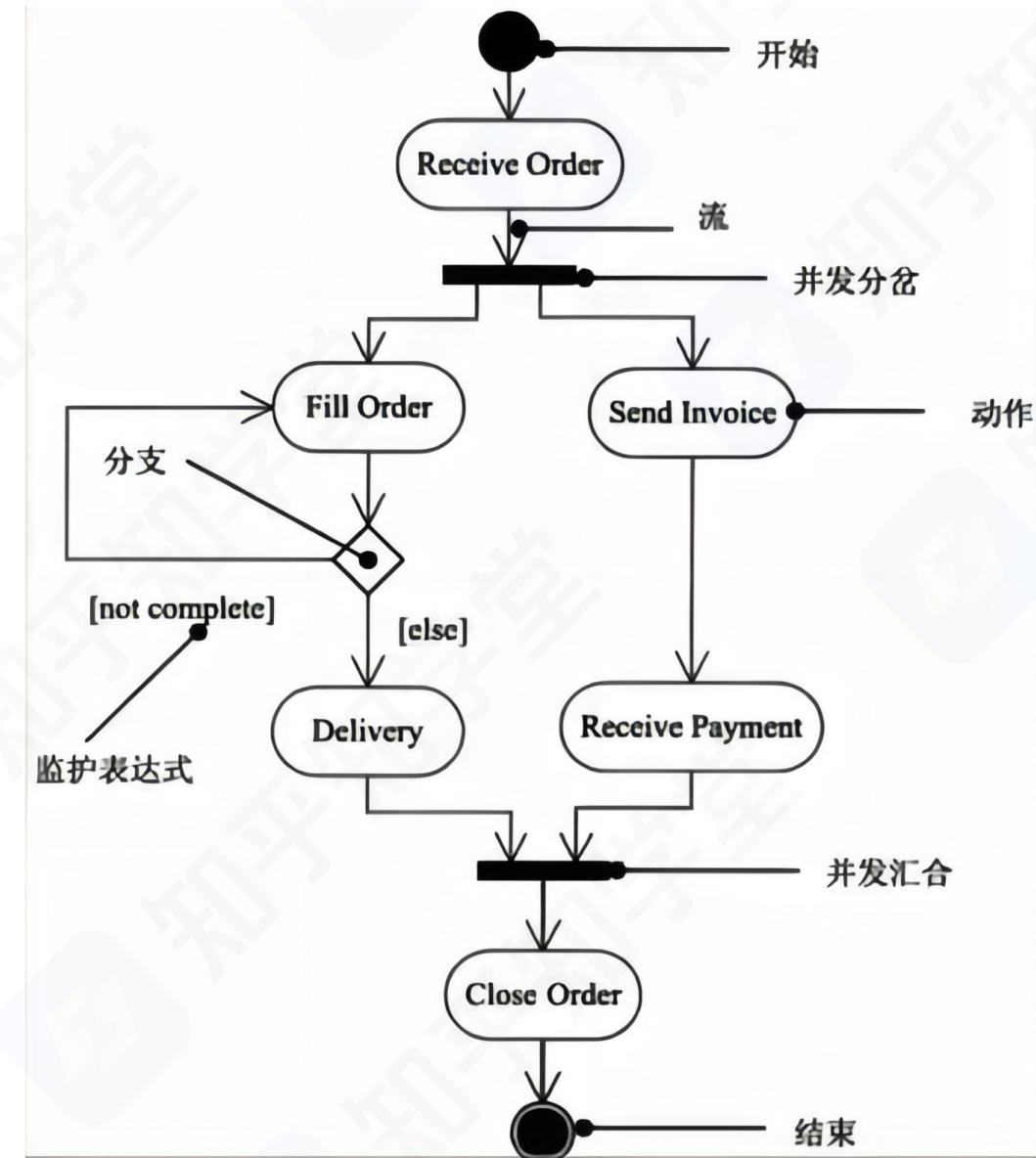
状态图：对一个单独对象的行为建模，指明对象在它的整个生命周期里，响应不同事件时，执行相关事件的顺序。

状态图中转换和状态是两个独立的概念，如下：图中方框代表状态，箭头上的代表触发事件，实心圆点为起点和终点。下图描述的就是门在生命周期里所经历的状态变化

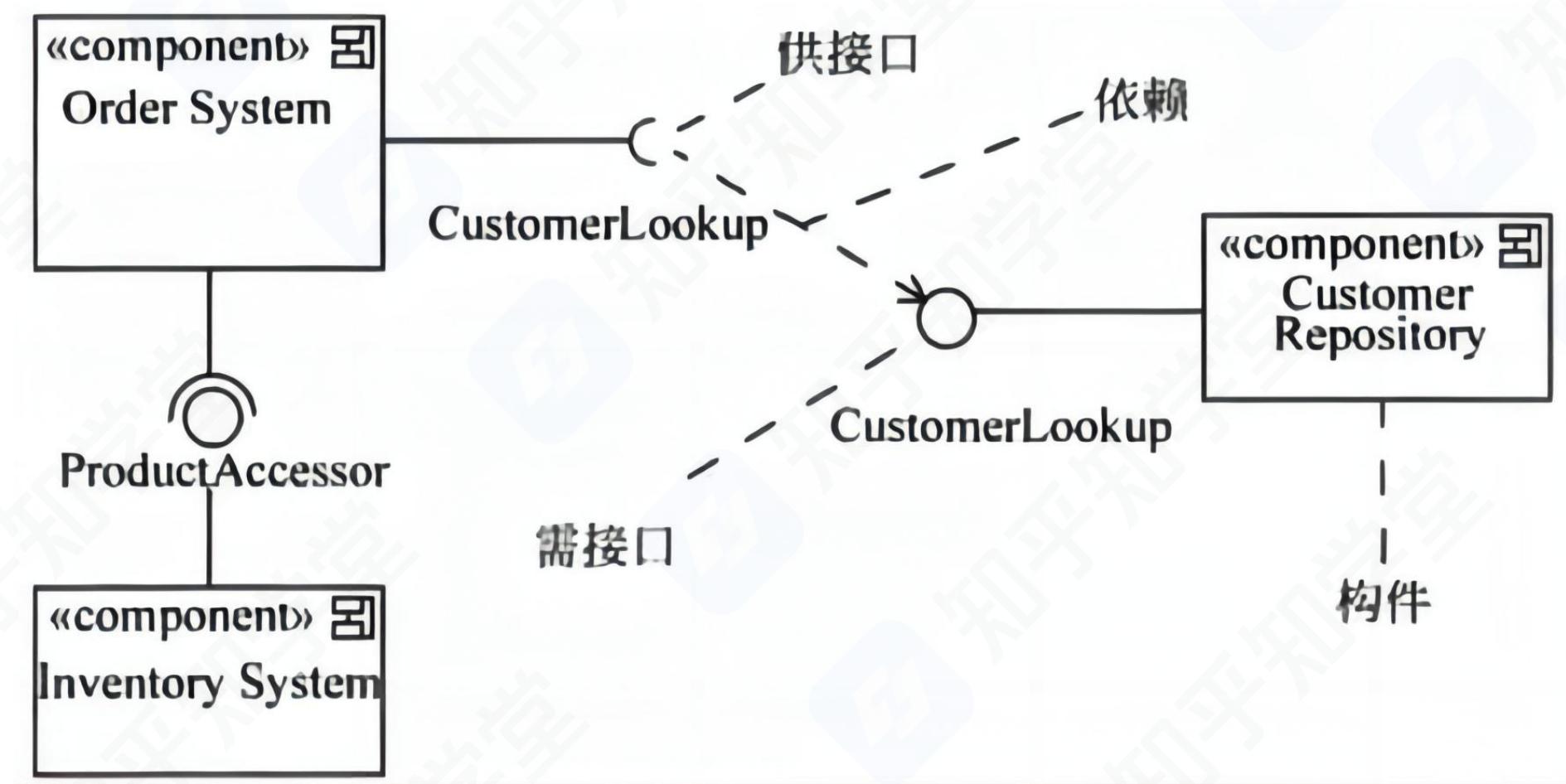


活动图：动态图，是一种特殊的状态图，展现了在系统内从一个活动到另一个活动的流程。

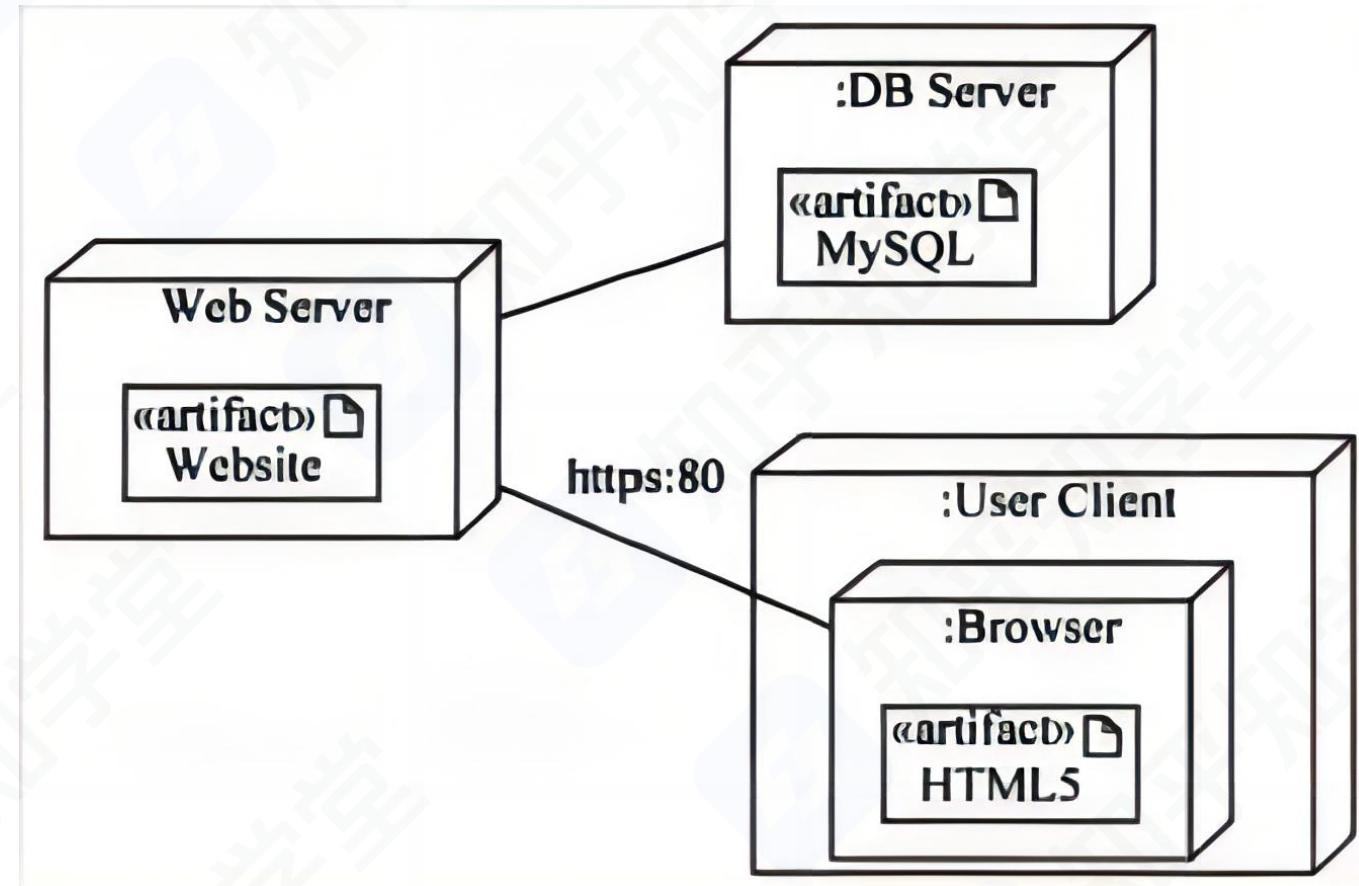
- 活动的分岔和汇合线是一条水平粗线。
- 每个分岔的分支数代表了可同时运行的线程数。
- 活动图中能够并行执行的是在一个分岔粗线下的分支上的活动。

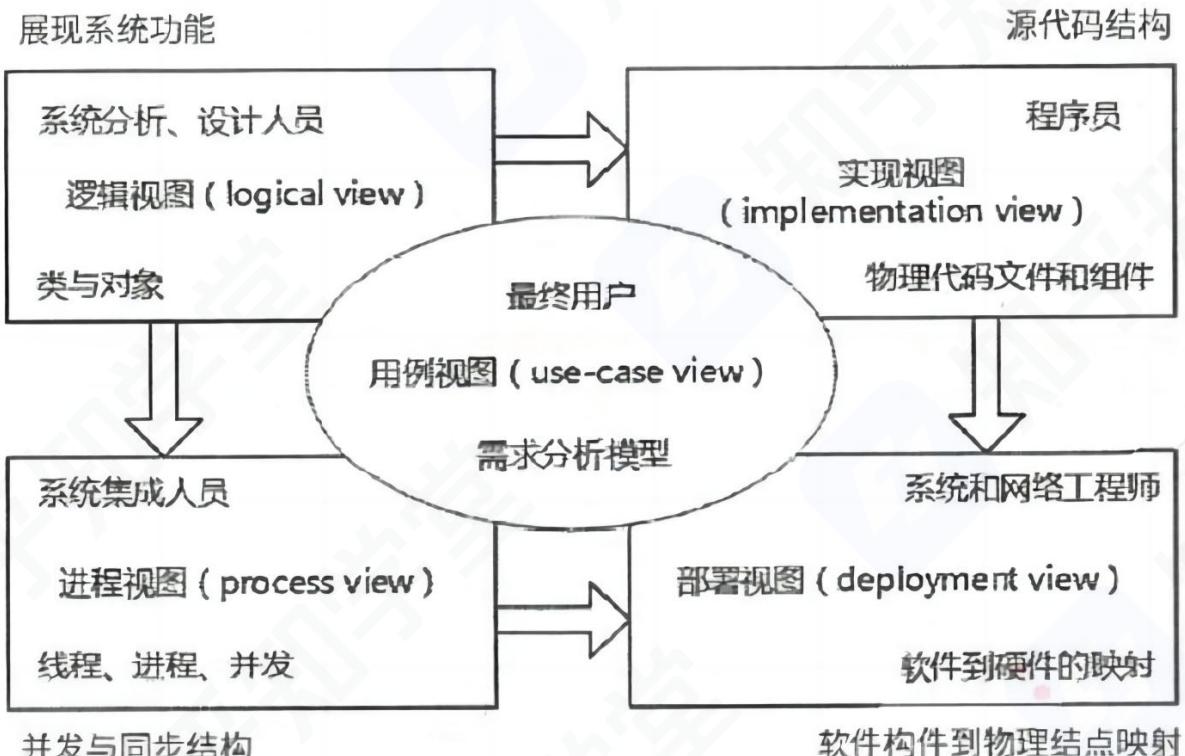


构件图(组件图): 静态图, 为系统静态实现视图, 展现了一组构件之间的组织和依赖。通常包括构件、接口以及各种关系



部署图：静态图，为系统静态部署视图，**部署图描述的事物理模块的节点分布**。它与构件图相关，通常一个结点包含一个或多个构件。其依赖关系类似于包依赖，因此部署组件之间的依赖是单向的类似于包含关系。





UML 4+1视图 (糅合后期的架构4+1视图)

- (1) 逻辑视图。逻辑视图也称为**设计视图**，它表示了设计模型中在架构方面具有重要意义的部分，即**类、子系统、包和用例实现的子集**。
- (2) 进程视图。进程视图是**可执行线程和进程作为活动类的建模**，它是**逻辑视图的一次执行实例**，描述了**并发与同步结构**。
- (3) 实现视图。实现视图对组成基于系统的**物理代码的文件和构件进行建模**。也叫**开发视图**
- (4) 部署视图。部署视图把**构件部署到一组物理节点上**，表示软件到硬件的映射和分布结构。也叫**物理视图**
- (5) 用例视图。用例视图是**最基本的需求分析模型**。也叫**统一的场景**

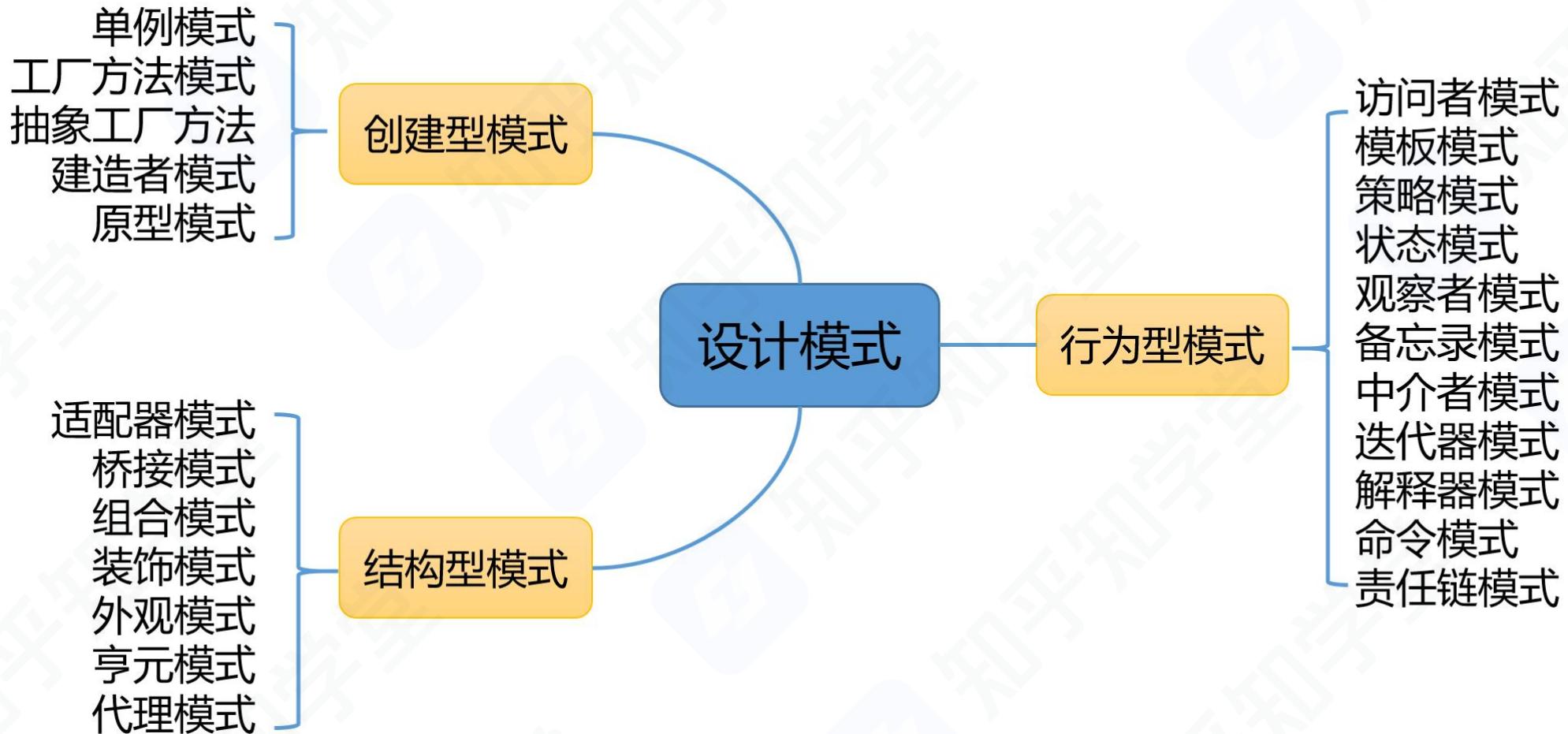
描述、SA模型的设计与分析方法，以及对SA设计经验的总结与复用等。有关SA模型描述的研究分为3个层次。

(1) SA的基本概念，即SA模型由哪些元素组成，这些组成元素之间按照何种原则组织。传统的设计概念只包括构件(软件系统中相对独立的有机组成部分，最初称为模块)以及一些基本的模块互联机制。随着研究的深入，构件间的互联机制逐渐独立出来，成为与构件同等级别的实体，称为连接子。现阶段的SA描述方法是构件和连接子的建模。近年来，也有学者认为应当把Aspect等引入SA模型。

(2) 体系结构描述语言(Architecture Description Language, ADL)，支持构件、连接子及其配置的描述语言就是如今所说的体系结构描述语言。ADL对连接子的重视成为区分ADL和其他建模语言的重要特征之一。典型的ADL包括UniCon、Rapide、Darwin、Wright、C2 SADL、Acme、xADL、XYZ/ADL和ABC/ADL等。

(3) SA模型的多视图表示，从不同的视角描述特定系统的体系结构，从而得到多个视图，并将这些视图组织起来以描述整体的SA模型。多视图作为一种描述SA的重要途径，也是近年来SA研究领域的重要方向之一。系统的每一个不同侧面的视图反映了一组系统相关人员所关注的系统的某一特定方面，多视图体现了关注点分离的思想。

把体系结构描述语言和多视图结合起来描述系统的体系结构，使系统更易于理解，方便系统相关人员之间进行交流，并且有利于系统的一致性检测以及系统质量属性的评估。学术界已经提出若干多视图的方案，典型的包括4+1模型(逻辑视图、进程视图、开发视图、物理视图，加上统一的场景)、Hofmesiter的4视图模型(概念视图、模块视图、执行视图和代码视图)、CMU-SEI的Views and Beyond模型(模块视图、构件和连接子视图、分配视图)等。此外，工业界也提出了若干多视图描述SA模型的标准，如IEEE标准1471-2000(软件密集型系统体系结构描述推荐实践)、开放分布式处理参考模型(RM-ODP)、统一建模语言(UML)以及IBM公司推出的Zachman框架等。需要说明的是，现阶段的ADL大多没有显式地支持多视图，并且上述多视图并不一定只是描述设计阶段的模型。



速记口诀：单抽元件厂

创建型设计模式	定义	记忆关键字
Abstract Factory 抽象工厂模式	提供一个接口，可以创建一系列相关或相互依赖的对象，而无需指定它们具体的类	抽象接口
Builder 构建器模式	将一个复杂类的表示与其构造相分离，使得相同的构建过程能够得出不同的表示	类和构造分离
Factory Method 工厂方法模式	定义一个创建对象的接口，但由子类决定需要实例化哪一个类。使得子类实例化过程推迟	子类决定实例化
Prototype 原型模式	用原型实例指定创建对象的类型，并且通过拷贝这个原型来创建新的对象	原型实例，拷贝
Singleton 单例模式	保证一个类只有一个实例，并提供一个访问它的全局访问点	唯一实例

速记口诀：外侨组员带配饰

结构型设计模式		定义	记忆关键字
Adapter 适配器模式		将一个类的接口转换成用户希望得到的另一种接口。它使原本不相容的接口得以协同工作	转换，兼容接口
Bridge 桥接模式		将类的抽象部分和它的实现部分分离开来，使它们可以独立的变化	抽象和实现分离
Composite 组合模式		将对象组合成树型结构以表示“整体-部分”的层次结构，使得用户对单个对象和组合对象的使用具有一致性	整体-部分，树形结构
Decorator 装饰模式		动态的给一个对象添加一些额外的职责。它提供了用子类扩展功能的一个灵活的替代，比派生一个子类更加灵活	附加职责
Facade 外观模式		定义一个高层接口，为子系统中的一组接口提供一个一致的外观，从而简化了该子系统的使用	对外统一接口
Flyweight 享元模式		提供支持大量细粒度对象共享的有效方法	细粒度，共享
Proxy 代理模式		为其他对象提供一种代理以控制这个对象的访问	代理控制

速记口诀：观摩（模）对（迭）策，责令解放（访），戒（介）忘台（态）

行为型设计模式	定义	记忆关键字
Chain of Responsibility 职责链模式	通过给多个对象处理请求的机会，减少请求的发送者与接收者之间的耦合。将接收对象链接起来，在链中传递请求，直到有一个对象处理这个请求	传递请求、职责链接
Command 命令模式	将一个请求封装为一个对象，从而可用不同的请求对客户进行参数化，将请求排队或记录请求日志，支持可撤销的操作	日志记录、可撤销
Interpreter 解释器模式	给定一种语言，定义它的文法表示，并定义一个解释器，该解释器用来根据文法表示来解释语言中的句子	解释器，虚拟机
Iterator 迭代器模式	提供一种方法来顺序访问一个聚合对象中的各个元素而不需要暴露该对象的内部表示	顺序访问，不暴露内部
Mediator 中介者模式	用一个中介对象来封装一系列的对象交互。它使各对象不需要显式地相互调用，从而达到低耦合，还可以独立的改变对象间的交互	不直接引用

速记口诀：观摩（模）对（迭）策，责令解放（访），戒（介）忘台（态）

行为型设计模式	定义	记忆关键字
Memento 备忘录模式	在不破坏封装性的前提下，捕获一个对象的内部状态，并在该对象之外保存这个状态，从而可以在以后将该对象恢复到原先保存的状态	保存，恢复
Observer 观察者模式	定义对象间的一种一对多的依赖关系，当一个对象的状态发生改变时，所有依赖于它的对象都得到通知并自动更新	通知、自动更新
State 状态模式	允许一个对象在其内部状态改变时改变它的行为	状态变成类
Strategy 策略模式	定义一系列算法，把它们一个个封装起来，并且使它们之间可互相替换，从而让算法可以独立于使用它的用户而变化	算法替换
Template Method 模板方法模式	定义一个操作中的算法骨架，而将一些步骤延迟到子类中，使得子类可以不改变一个算法的结构即可重新定义算法的某些特定步骤	子类按照父类的模板去实现
Visitor 访问者模式	表示一个作用于某对象结构中的各元素的操作，使得在不改变各元素的类的前提下定义作用于这些元素的新操作。	数据和操作分离



03

项目管理



- 本章节在历年考试过程中的分值占比大概是1-2分
- 本章节在改版之后已经做了大量的删减，近两年的考试就只是针对配置项、进度管理和关键路径进行了考察，我们只需要针对这几个点进行熟悉和掌握即可
- 被考到知识点有：
 - 软件配置项
 - 进度管理
 - 关键路径
- 在改版之后的考试中，分别考察的知识点为：
 - 2023年11月：软件配置项、进度管理
 - 2024年05月：关键路径
 - 2025年11月：进度管理

软件项目进度管理的目的：确保软件项目在规定的时间内按期完成。

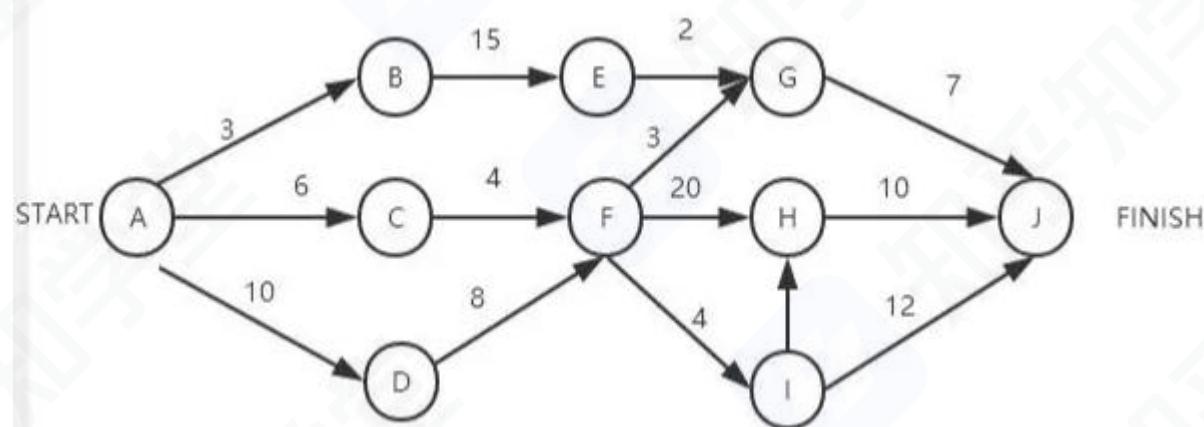
项目管理基本原则：划分、相互依赖、时间分配、工作量确认、确定责任、明确输出结果、确定里程碑

项目管理进度安排：为监控软件项目的进度计划和工作的实际进展情况，表示各项任务之间进度的相互依赖关系，需要采用图示的方法。

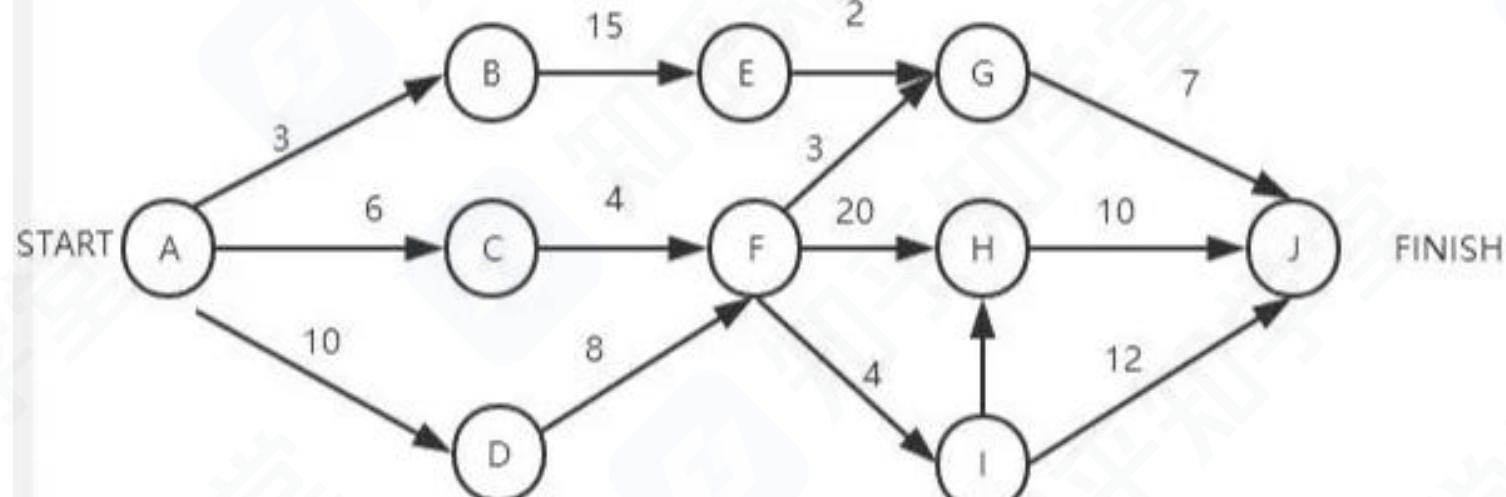
- Gantt图：又称为横道图，横轴表示时间，纵轴表示活动，以时间顺序表示活动，能反应活动间的并行关系，但无法反应活动之间的依赖关系，因此也难以清晰的确定关键任务和关键路径。
- Pert图：类似于前趋图，是有向图，反应活动之间的依赖关系，有向边上标注活动运行的时间，但无法反应活动之间的并行关系。



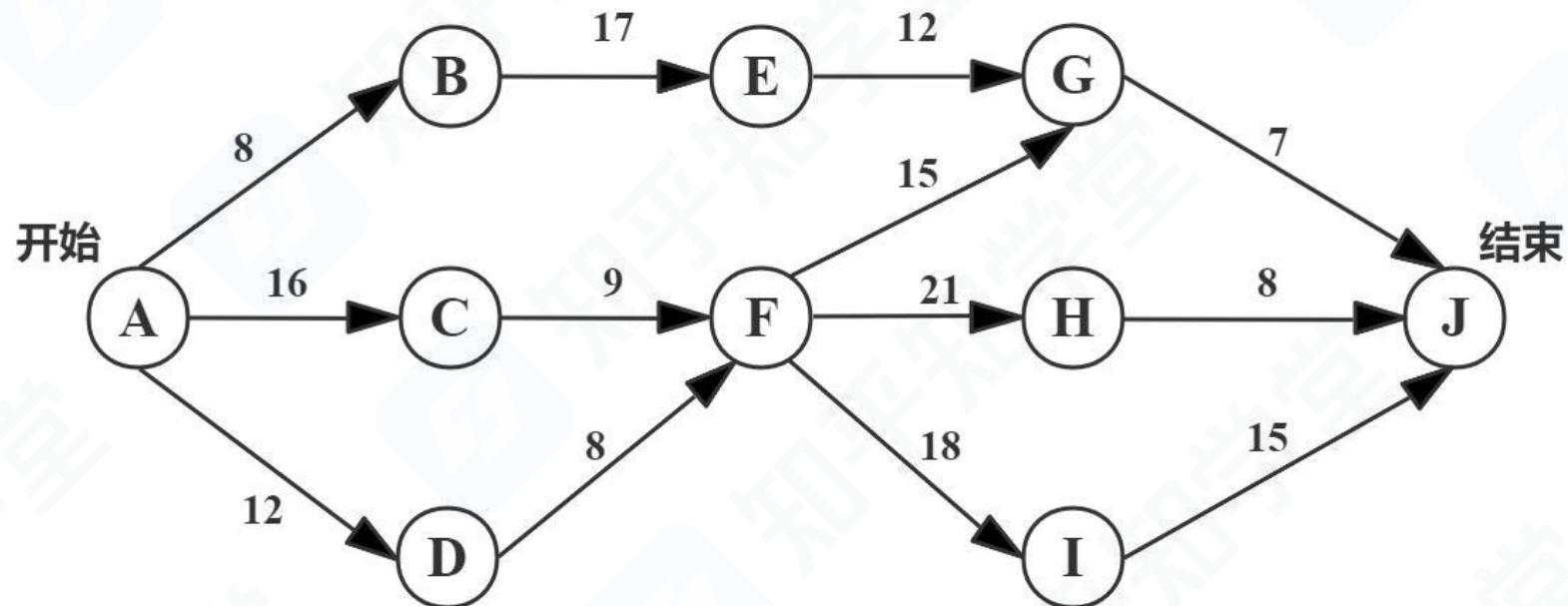
图 5-12 Gantt 图实例



所谓关键路径就是指项目完成必须要执行的环节和步骤，我们可以通过PERT图来求出项目要完成必须要经过哪些环节以及项目的总工期，注意，虽然关键路径是项目的最短工期，但却是从开始到结束时间最长的路径



例：下图是一个软件项目的活动图，其中顶点表示项目里程碑，连接顶点的边表示包含的活动，边上的数字表示完成该活动所需要的天数。则关键路径长度为（17）。若在实际项目进展中，在其他活动都能正常进行的前提下，活动（18）一旦延期就会影响项目的进度。



- (17) A. 34

- B. 47

- C. 54

- D. 58

- (18) A. A→B

- B. C→F

- C. D→F

- D. F→H

1. 某项目包含四道工序P、Q、R、S，依赖关系如下：P完成后，Q和R可同时开始，R完成后，S才能开始，各工序正常工期及直接费用：

- P: 3天 (2万元)
- Q: 5天 (4万元)
- R: 4天 (3万元)
- S: 2天 (1万元)

项目每日间接费用1.2万元。若正常推进无赶工，关键路径和总成本分别是多少？

- A. 关键路径 P→Q，总成本 18.6万元
- B. 关键路径 P→R→S，总成本 20.8万元
- C. 关键路径 P→R，总成本 19.2万元
- D. 关键路径 P→Q→S，总成本 21.4万元

1. 某项目包含四道工序P、Q、R、S，依赖关系如下：P完成后，Q和R可同时开始，R完成后，S才能开始，各工序正常工期及直接费用：

- P: 3天 (2万元)
- Q: 5天 (4万元)
- R: 4天 (3万元)
- S: 2天 (1万元)

项目每日间接费用1.2万元。若正常推进无赶工，关键路径和总成本分别是多少？

- A. 关键路径 P→Q，总成本 18.6万元 B. 关键路径 P→R→S，总成本 20.8万元
C. 关键路径 P→R，总成本 19.2万元 D. 关键路径 P→Q→S，总成本 21.4万元

2. 在题目1中，若通过赶工压缩工期（参数如下），求最短工期及对应总成本：

- P最多缩短1天，每天额外成本0.5万
 - Q最多缩短2天，每天额外成本0.8万
 - S最多缩短1天，每天额外成本0.3万
- A. 最短工期6天，总成本21.3万元 B. 最短工期7天，总成本20.8万元
C. 最短工期8天，总成本19.9万元 D. 最短工期7天，总成本22.1万元

软件配置管理（**Software Configuration Management**, 简称**SCM**）是软件项目管理的一个关键方面，旨在有效地管理和控制软件开发过程中的各种元素和变更。它涉及到跟踪、控制和管理软件系统的不同版本、构件、文档和配置项，以确保软件开发过程的有序性、可追踪性和可控性。

关键概念和任务：

- 配置项（**Configuration Items, CIs**）：配置项是软件项目中的各种组成部分，包括源代码、文档、库、二进制文件、脚本等。每个配置项都可以被版本化、控制和管理。
- 版本控制：**SCM**包括版本控制，它允许跟踪和管理配置项的不同版本。每个版本都有一个唯一标识符，以便在需要时回溯到特定版本。
- 变更管理：变更管理是**SCM**的一部分，它涉及处理和记录对配置项的变更请求、评审、批准和实施。这确保了变更有计划、受控制和可追踪的。
- 配置控制：配置控制是确保配置项在整个软件开发周期中保持一致性和稳定性的过程。它包括定义、文档化和执行配置管理策略。
- 构建和发布管理：**SCM**也涵盖了构建和发布管理，确保软件的不同版本可以按需构建和发布，以满足用户需求。

风险管理就是要对项目风险进行认真的分析和科学的管理，这样，是能够避开不利条件、少受损失、取得预期的结果并实现项目目标的，能够争取避免风险的发生或尽量减小风险发生后的影响。但是，完全避开或消除风险，或者只享受权益而不承担风险是不可能的。

风险管理计划编制：如何安排与实施项目的风险管理，制定下列各步的计划

- **风险识别：**识别出项目中已知和可预测的风险，确定风险的来源、产生的条件、描述风险的特征以及哪些项目可以产生风险，形成一个风险列表。
- **风险定性分析：**对已经识别的风险进行排序，确定风险可能性与影响、确定风险优先级、确定风险类型。
- **风险定量分析：**进一步了解风险发生的可能性具体由多大，后果具体由多严重。这一步并不是必须的
- **风险应对计划编制：**对每一个识别出来的风险来分别制定应对措施，这些措施组成的文档称为风险应对计划。包括消极风险和积极风险。
- **风险监控：**监控风险计划的执行，检测残余风险，识别新的风险，保证风险计划的执行，并评价这些计划对减少风险的有效性。



在信息系统项目中，从宏观上来看，风险可以分为项目风险、技术风险和商业风险。

项目风险是指潜在的预算、进度、个人(包括人员和组织)、资源、用户和需求方面的问题，以及它们对项目的影响。项目复杂性、规模和结构的不确定性也构成项目的(估算)风险因素。项目风险威胁到项目计划，一旦项目风险成为现实，可能会拖延项目进度，增加项目的成本。

技术风险是指潜在的设计、实现、接口、测试和维护方面的问题。此外，规格说明的多义性、技术上的不确定性、技术陈旧、最新技术(不成熟)也是风险因素。技术风险威胁到待开发系统的质量和预定的交付时间。如果技术风险成为现实，开发工作可能会变得很困难或根本不可能。

商业风险威胁到待开发系统的生存能力，主要有以下5种不同的商业风险：

- (1)市场风险：开发的系统虽然很优秀但不是市场真正所想要的。
- (2)策略风险：开发的系统不再符合企业的信息系统战略。
- (3)销售风险：开发了销售部门不清楚如何推销的系统。
- (4)管理风险：由于重点转移或人员变动而失去上级管理部门的支持。
- (5)预算风险：开发过程没有得到预算或人员的保证。



04

信息安全



- 本章节在历年考试过程中的分值占比大概是**2-4分**，对应新版本教材中的第四章，另外在下篇八大架构里面也额外增加了一个安全架构的章节，相对之前新增了不少内容，超纲率中等，会考到很多书本上没有涉及到的安全知识，比如**24年5月**就有一半的分值超纲了，**24年11月**除了有超纲的题，还考到了一道特别偏的题目：安全审计的四要素，同学们应对本章节需要心态放好，在了解课本上知识的同时，尽量拓展网安的其他知识
- 被考到知识点有：
 - 信息属性：保密性、可用性、完整性、不可抵赖性等
 - 安全需求：物理安全、网络安全、系统安全、应用安全
 - 网络安全技术：防火墙、入侵检测、入侵防御、网络攻击和威胁、计算机病毒和木马
 - 安全协议：**SSL**（在互联网上建立安全连接的协议）、**SET**（网上交易设计的协议）、**Kerberos**（计算机网络认证协议）、**PGP**（数据加密、数字签名）
- 在改版之后的考试中，分别考察的知识点为：
 - **2023年11月**：**SSL特性**、**RSA算法**(非对称加密算法)
 - **2024年05月**：机密性、不可否认性、基于任务的访问控制模型、安全等级、灾难恢复级别
 - **2024年11月**：安全审计四要素、信息属性（保密性和完整性）、数据安全治理



信息安全包括5个基本要素：**机密性、完整性、可用性、可控性与可审查性**。

机密性：机密性指的是信息只能被授权的人员或实体访问，**防止未经授权的人获取敏感信息**。

- 假设你要通过电子邮件发送包含敏感个人信息的文件给你的朋友。为确保机密性，你可以使用端到端加密的方式发送邮件，这样只有你和你的朋友能够解密和查看邮件的内容，其他人无法获取其中的信息。

完整性：确保信息在传输或存储过程中**不被篡改或损坏**，保持其准确性和完整性。

- 假设你是一家互联网银行的客户，你要发送一笔转账请求给银行。为确保请求的完整性，银行使用数字签名来对你的请求进行签名。当银行收到请求后，会验证数字签名的有效性，以确保请求没有被篡改，然后继续处理转账。

可用性：确保信息和资源在**需要时可供访问和使用**，避免因服务中断或不可用性导致的损失。

- 假设你是一家在线零售商，你的网站每天都有大量访问量。为确保网站的可用性，你可以设置冗余服务器，当一个服务器发生故障时，备用服务器可以接管服务，保持网站在线，避免服务中断。

可控性：是指对系统或信息进行管理和控制，**确保只有授权的人员可以进行合法的操作**。

- 假设你是一家大型公司的系统管理员，你要设置员工访问公司内部数据库的权限。根据员工的角色和职责，分配不同级别的访问权限，只有被授权的员工才能访问敏感数据。

可审查性：指对系统和操作进行**监控和审查的能力**，记录和跟踪系统中发生的事件，以便追查和解决安全问题。



信息安全的范围包括：设备安全、数据安全、内容安全和行为安全。

- 信息安全：是信息系统安全的首要问题，是信息系统安全的物质基础，主要涉及到保护计算机系统、网络设备和其他硬件设备免受恶意攻击和未经授权的访问，设备安全的主要目标是确保这些设备的操作和功能不会被破坏、篡改或滥用。它包括3个方面：设备的稳定性、可靠性、可用性。
- 数据安全：即采取措施确保数据免受未授权的泄露、篡改和损坏，包括3个方面：数据的秘密性、完整性、可用性。
- 内容安全：是信息安全在政治、法律、道德层次上的要求，包括3个方面：信息内容政治上健康、符合国家法律法规、
- 行为安全：关注人的行为和习惯，以预防社会工程学攻击和其他利用人为因素的安全威胁。行为安全措施可能包括培训员工识别垃圾邮件、不受信任的链接和附件，以及避免与陌生人共享敏感信息。行为安全的特性包括：行为的秘密性、完整性、可控性。

信息的存储安全包括：信息使用的安全、系统安全监控、计算机病毒防治、数据的加密和防止非法的攻击等。

网络安全包括：

- 网络安全漏洞：物理安全性、软件安全漏洞、不兼容使用安全漏洞、选择合适的安全哲理。
- 网络安全威胁：非授权的访问、信息泄露或丢失、破坏数据完整性、拒绝服务攻击、利用网络传播病毒。
- 安全措施的目标：访问控制、认证、完整性、审计、保密。

信息安全系统的组成框架包含三个体系：技术体系、组织结构体系和管理体系

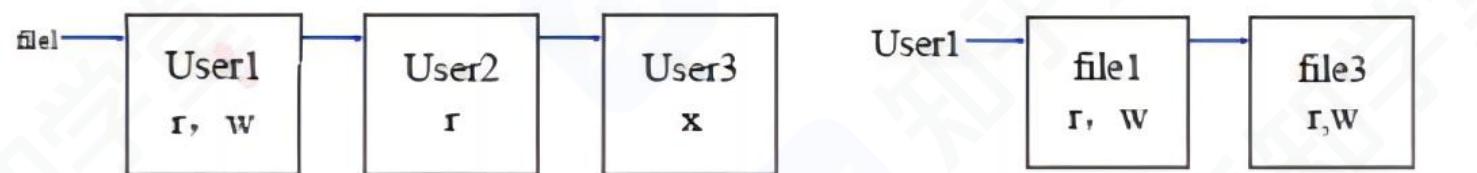
访问控制：是指主体依据某些控制策略或权限对客体本身或是其资源进行的不同授权访问，包括认证、控制策略实现和审计3方面的内容。访问控制的3要素：主体、客体和控制策略。

访问控制的实现技术

- **访问控制矩阵 (ACM)**。是通过矩阵形式表示访问控制规则和授权用户权限的方法。主体作为行，客体作为列。

	file1	file2	file3
User1	rw		rw
User2	r	rwx	x
User3	x	r	

- **访问控制表 (ACL)**。目前最流行、使用最多的访问控制实现技术。每个客体有一个访问控制表，是系统中每一个有权访问这个客体的主体的信息。这种实现技术实际上是按列保存访问矩阵。
- **能力表**。对应于访问控制表，这种实现技术实际上是按行保存访问矩阵。每个主体有一个能力表，是该主体对系统中每一个客体的访问权限信息。使用能力表实现的访问控制系统可以很方便地查询某一个主体的所有访问权限。



- **授权关系表**。每一行（或者说元组）就是访问矩阵中的一个非空元素，是某一个主体对应于某一个客体的访问权限信息。如果授权关系表按主体排序，查询时就可以得到能力表的效率；如果按客体排序查询时就可以得到访问控制表的效率。



为对抗攻击者的攻击，密钥生成需要考虑3个方面的因素：增大密钥空间、选择强钥（复杂的）、密钥的随机性（使用随机数）。

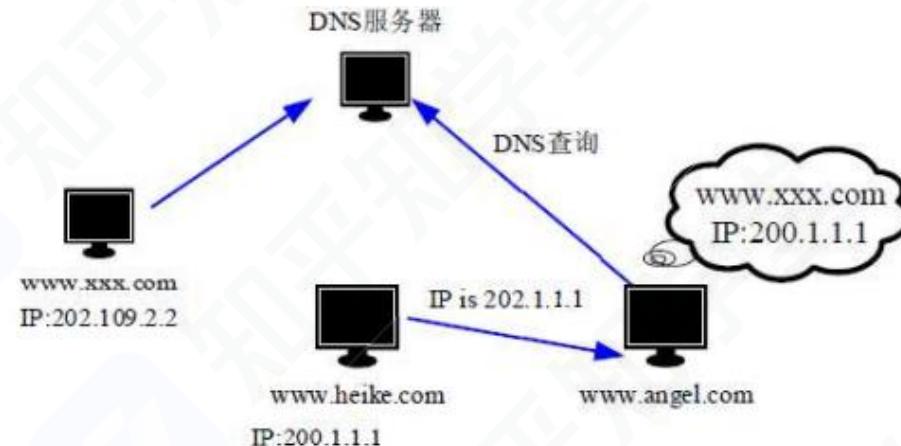
外部用户针对网络连接发动拒绝服务攻击主要有以下几种模式：消耗资源、破坏或更改配置信息、物理破坏或改变网络部件、利用服务程序中的处理错误使服务失效。

分布式拒绝服务DDoS攻击：是传统DoS攻击的发展，攻击者首先侵入并控制一些计算机，然后控制这些计算机同时向一个特定的目标发起拒绝服务攻击。克服了传统DOS受网络资源的限制和隐蔽性两大缺点。

拒绝服务攻击的防御方式：

- **加强对数据包的特征识别**，攻击者发送的数据包中是有一些特征字符串。通过搜寻这些特征字符串，就可以确定攻击服务器和攻击者的位置。
- **设置防火墙监视本地主机端口的使用情况**。如果发现端口处于监听状态，则系统很可能受到攻击。
- **对通信数据量进行统计**也可获得有关攻击系统的位置和数量信息。在攻击时，攻击数据的来源地址会发出超出正常极限的数据量。
- 尽可能的修正已经发现的问题和系统漏洞。

DNS欺骗：首先是冒充域名服务器，然后把查询的IP地址设为攻击者的IP地址，这样的话，用户上网就只能看到攻击者的主页，而不是用户想要取得的网站的主页了，这就是DNS欺骗的基本原理。也即改掉了域名和IP地址的对应关系。黑客是通过冒充DNS服务器回复查询IP的，如下图所示：



DNS欺骗的检测：

- **被动监听检测**：通过旁路监听的方式，捕获所有DNS请求和应答数据包，并为其建立一个请求应答映射表。如果在一定的时间间隔内，一个请求对应两个或两个以上结果不同的应答包，则怀疑受到了DNS欺骗攻击。
- **虚假报文探测**：采用主动发送探测包的手段来检测网络内是否存在DNS欺骗攻击者。如果向一个非DNS服务器发送请求包，正常来说不会收到任何应答，如果收到了应答包，则说明受到了攻击。
- **交叉检查查询**：在客户端收到DNS应答包之后，向DNS服务器反向查询应答包中返回的IP地址所对应的DNS名字，如果二者一致说明没有受到攻击，否则说明被欺骗。



端口扫描：端口扫描是指通过发送数据包到目标计算机或网络设备的各个端口，以检测哪些端口是开放的，并了解这些端口上运行的服务及其版本信息。其基本原理是：当一台计算机想要与另一台计算机上的特定服务进行通信时，它会向目标主机的相应端口发送数据包。如果目标端口是开放的，它会回应一个确认消息；如果端口是关闭的，通常不会有响应；而如果端口被防火墙过滤，可能会收到一个错误消息。端口扫描工具就是基于这一原理，通过大量发送数据包，并分析返回的结果，来判断端口的状态。

常见扫描方式：全连接扫描、半连接扫描（SYN扫描）和隐秘扫描（第三方肉鸡扫描）。

常见扫描工具：

- **Nmap**: 是一个开源的安全和端口扫描器，也是一个网络探索工具。它能够发现网络上的服务和主机，从而创建网络映射。该软件提供了多种功能，可帮助探测计算机网络、主机发现以及检测操作系统，还提供高级漏洞检测。
- **Nessus**: 世界上最著名的漏洞扫描程序之一，由Tenable Network Security设计。它可以检测未打补丁的服务和错误配置、弱密码（默认和常用）以及各种系统漏洞。
- **Metasploit**: 面向渗透测试的工具，网络安全专家可针对远程目标开发和执行漏洞利用代码。它本身是开源的，但专业版Metasploit Pro需要付费购买。



防范措施

- 配置防火墙规则：通过配置防火墙规则，可以阻止未授权的端口扫描和数据包进入网络。
- 定期更新软件和操作系统：及时安装最新的安全补丁和更新，以修复已知的安全漏洞。
- 关闭不必要的端口和服务：仅对外开放必要的端口和服务，减少潜在的攻击面。
- 使用强密码和身份验证：为重要的服务和账户设置强密码，并启用身份验证机制，以增加攻击者的难度。
- 定期监控和审计：持续监控网络流量和日志数据，及时发现并响应异常的网络活动和潜在的攻击迹象。

漏洞扫描：指对重要计算机信息系统进行检查，发现其中可能被黑客利用的漏洞。包括基于网络的漏洞扫描（通过网络远程扫描主机）、基于主机的漏洞扫描（在目标系统安装了代理扫描）。

GB17859—999标准规定了计算机系统安全保护能力的五个等级：

- **第一级用户自主保护级**: 实现基本的用户隔离和自主访问控制。比如：个人电脑操作系统,通过账号密码隔离不同用户,用户只能访问自己的文件和进程。
- **第二级系统审计保护级**: 在第一级基础上实现更细粒度的访问控制和审计。比如：公司内部办公系统,有账号密码登录,对安全相关的登录、文件访问进行日志记录,对不同部门的数据访问设置权限控制。
- **第三级安全标记保护级**: 在第二级基础上实现安全标记和强制访问控制。比如：政府内部文件系统,对文件设置密级标记,用户的访问权限按照政策与文件标记相匹配判定,进行强制访问控制。
- **第四级结构化保护级**: 在第三级基础上实现形式化安全策略和对所有资源的访问控制。比如：军方信息系统,有多级安全策略模型,对所有用户身份、存储数据、网络资源等实施严格的强制访问控制。
- **第五级访问验证保护级**: 实现抗篡改的访问监控器,对单个用户的所有访问进行仲裁。比如：核设施监控系统,通过抗篡改的访问控制器验证每个用户每次访问的权限,拒绝非授权访问。

安全风险管理：在风险评估实施前，应该考虑：

- 确定风险评估的范围。
- 确定风险评估的目标。
- 建立适当的组织结构。
- 建立系统性的风险评估方法。
- 获得最高管理者对风险评估策划的批准。

风险评估的基本要素为脆弱性、资产、威胁、风险和安全措施，与这些要素相关的属性分别为业务战略、资产价值、安全需求、安全事件和残余风险，这些也是风险评估要素的一部分。

风险计算模型包含信息资产、弱点/脆弱性、威胁等关键要素。每个要素有各自的属性，信息资产的属性是资产价值，弱点的属性是弱点被威胁利用后对资产带来的影响的严重程度，威胁的属性是威胁发生的可能性。

风险计算的过程如下：

- 对信息资产进行识别，并对资产赋值。
- 对威胁进行分析，并对威胁发生的可能性赋值。
- 识别信息资产的脆弱性，并对弱点的严重程度赋值。
- 根据威胁和脆弱性计算安全事件发生的可能性。
- 结合信息资产的重要性和发生安全事件的可能性，计算信息资产的风险值。



真题



知乎知学堂

在进行软件系统安全性分析时，（）保证信息不泄露给未授权的用户、实体或过程；完整性保证信息的完整和准确，防止信息被非法修改；（）保证对信息的传播及内容具有控制的能力，防止为非法者所用。

- A.完整性 B.不可否认性 C.可控性 D.保密性

- A.完整性 B.安全审计 C.加密性 D.可控性

完整的信息安全系统至少包含三类措施，即技术方面的安全措施、管理方面的安全措施和相应的（）。其中，信息安全的技术措施主要有：信息加密、数字签名、身份鉴别、访问控制、网络控制技术、反病毒技术、（）。

- A.用户需求 B.政策法律 C.市场需求 D.领域需求

- A.数据备份和数据测试 B.数据迁移和数据备份 C.数据备份和灾难恢复 D.数据迁移和数据测试

39-41.信息系统面临多种类型的网络安全威胁。其中，信息泄露是指信息被泄露或透露给某个非授权的实体；(39)是指数据被非授权地进行增删、修改或破坏而受到损失；(40)是指对信息或其他资源的合法访问被无条件地阻止；(41)是指通过对系统进行长期监听，利用统计分析方法对诸如通信频度、通信的信息流向、通信总量的变化等参数进行研究，从而发现有价值的信息和规律。

- | | | | |
|---------|------------|--------|---------|
| A.非法使用 | B.破坏信息的完整性 | C.授权侵犯 | D.计算机病毒 |
| A.拒绝服务 | B.陷阱门 | C.旁路控制 | D.业务欺骗 |
| A.特洛伊木马 | B.业务欺骗 | C.物理侵入 | D.业务流分析 |

67-68、某WEB网站向CA申请了数字证书。用户登录过程中可通过验证(67)确认该数字证书的有效性，以 (68)。

- | | | | |
|--------------|-------------|--------|---------|
| A.CA的签名 | B.网站的签名 | C.会话密钥 | D.DES密码 |
| A.向网站确认自己的身份 | B.获取访问网站的权限 | | |
| C.和网站进行双向认证 | D.验证该网站的真伪 | | |

2024年05月第50-51题：()保证信息在传输和存储过程中不被未授权的访问和泄露，()确保通信双方不能否认他们发送或接收的信息。

- | | | | |
|-------|-------|-------|--------|
| A.保密性 | B.完整性 | C.可用性 | D.抗否认性 |
| A.保密性 | B.完整性 | C.可用性 | D.抗否认性 |

2024年05月第64题：基于任务的访问控制(TBAC)模型由()组成

- A.主体、客体、属性、角色、策略
- B.主体、客体、角色、权限、访问策略
- C.业务流程、授权主体、受托客体、策略集
- D.工作流、授权结构体、受托人集、许可集

2024年05月第65题：()是一种网络安全标准，用于数据的传输安全。

- A.IPSec
- B.RSA
- C.TCSEC
- D.SHA-1



真题



2024年05月第66题：GB17859—999标准规定了计算机系统安全保护能力的五个等级，其中安全等级最高的是()

- A.访问验证保护级
- B.系统设计保护级
- C.安全标记保护级
- D.结构化保护级

2024年05月第67题：以下灾难中心建设中，灾难恢复级别最高的是 ()

- A.实时数据传输及完整设备支持
- B.数据0丢失和远程集群支持
- C.电子传输及完整设备支持
- D.实时数据传输及远程集群支持

2023年11月第10题：SSL协议不能实现以下()安全

- A.保密性
- B.完整性
- C.可用性
- D.不可抵赖性

2023年11月第26题：以下 ()属于非对称加密算法

- A.DES
- B.RSA
- C.IDEA
- D.AES



THE END

功不唐捐，玉汝于成！

• 开启新征程 •

