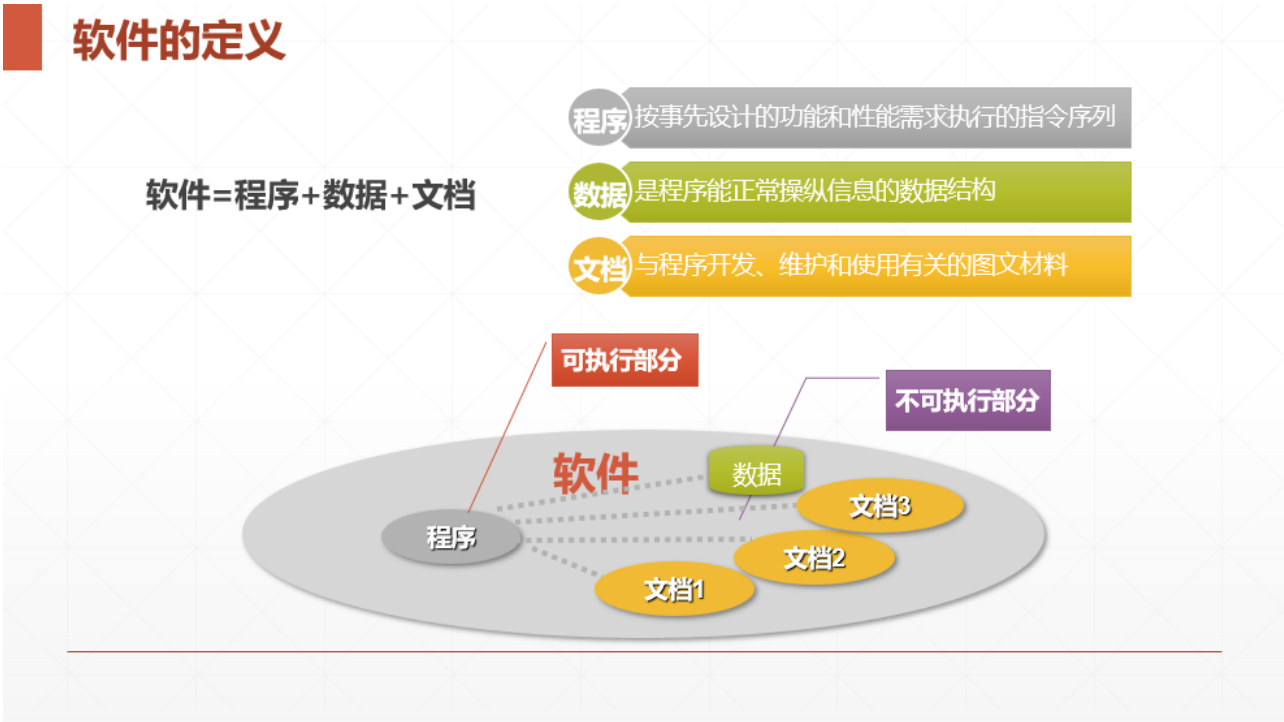
chapter 1

软件的概念和特点

概念：

软件=程序+数据+文档



特点：

软件是开发的或者是工程化的，而不是制造的

软件生产是简单拷贝，而不是重复开发

软件产品易于多次修改，且总是要求修改

软件开发的环境对产品影响较大

软件开发的时间和工作量难以估计

软件开发进度难以客观衡量

软件的测试非常困难

软件不会磨损和老化，但会退化

软件维护不是简单更换元器件，变更容易产生新的问题

软件的特征

软件危机的概念和产生的原因

定义：
在计算机软件的开发和维护过程中所遇到的一系列严重问题。 （效率和质量下降）

什么是软件危机

产生的原因：
客观：软件本身特点：逻辑部件、规模庞大
主观：不正确的开发方法：忽视需求分析、错误认为：软件开发=程序编写、轻视软件维护

产生软件危机的原因

客观：软件本身特点	主观：不正确的开发方法
• 逻辑部件 • 规模庞大	• 忽视需求分析 • 错误认为：软件开发=程序编写 • 轻视软件维护

软件工程的定义、三要素、应用软件工程的原因

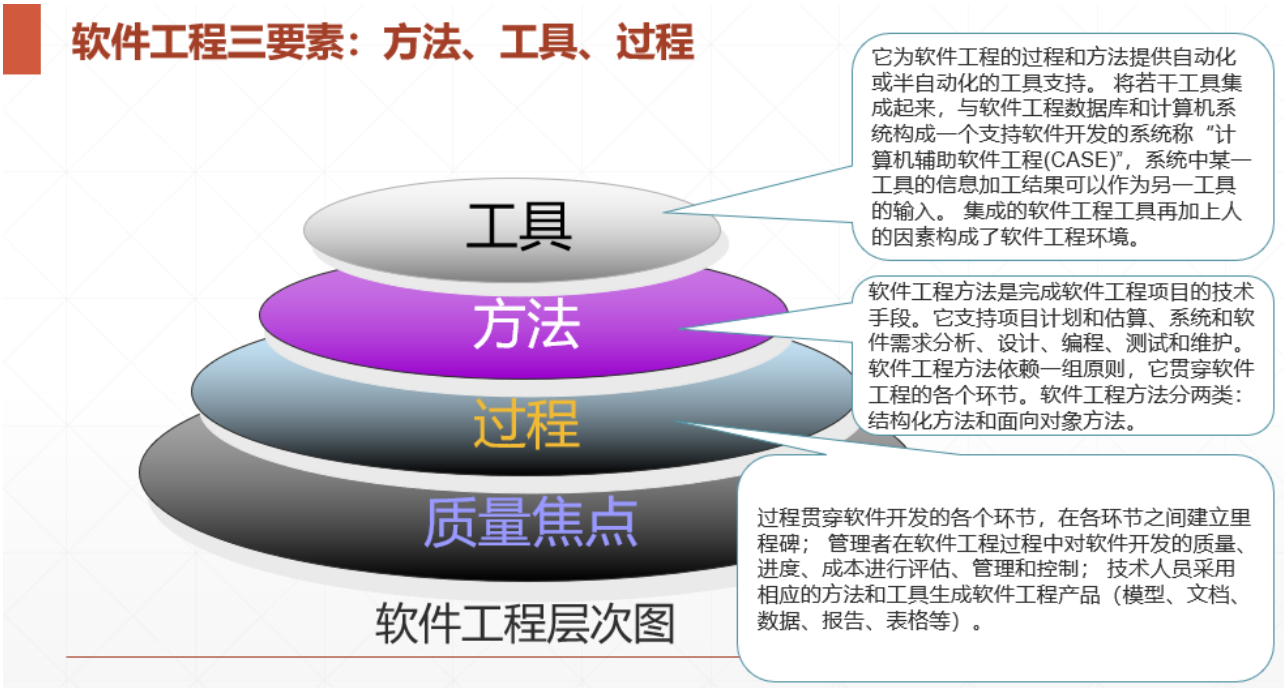
定义：

IEEE计算机协会将软件工程定义为：（1）应用系统化的、学科化的、定量的方法，来开发、运行和维护软件，即，将工程应用到软件。（2）对（1）中各种方法的研究。

软件工程的定义

定义	IEEE计算机协会将软件工程定义为：（1）应用系统化的、学科化的、定量的方法，来开发、运行和维护软件，即，将工程应用到软件。（2）对（1）中各种方法的研究。
目标	软件工程的目标是在给定的时间和预算内，按照用户的需求，开发易修改、高效、可靠、可维护、适应力强、可移动、可重用的软件。

软件工程三要素：方法、工具、过程



应用软件工程的原因：PPT上没有找到对应的内容，以下来自文心一言：

> 应用软件工程有很多重要的原因，其中一些包括：
>
> 提高软件质量：通过采用工程化的原则和方法，软件工程能够提高软件的质量和可靠性。这包括减少错误、提高一致性、提高软件的可维护性和可扩展性等。
> 降低开发风险：软件工程提供了一种系统化的方法来管理软件开发过程中的风险，例如需求不明确、技术难题和人员问题等。通过这种方式，软件工程能够降低开发过程中的风险，避免潜在的问题和灾难。
> 提高开发效率：软件工程通过使用经过时间考验的技术和方法，以及有效的项目管理，能够提高开发效率。这可以减少开发时间和资源浪费，提高开发效率，降低开发成本。
> 满足用户需求：软件工程通过用户需求分析和调查，确保开发的软件满足用户的需求和期望。这样可以为用户提供更好的产品和服务，提高用户满意度。
> 提高软件开发的可维护性：软件工程的采用可以使软件更易于维护和修改。这包括设计良好的软件架构、模块化设计、清晰的代码结构等。这样可以在不影响其他部分的情况下修改和扩展软件。
> 促进团队合作：软件工程强调团队合作和沟通，这可以促进团队成员之间的协作和配合。通过明确的角色和责任分配，以及有效的沟通机制，可以更好地协调团队成员的工作，确保项目的顺利完成。
> 总之，应用软件工程可以提高软件的质量、降低开发风险、提高开发效率、满足用户需求、提高软件开发的可维护性和促进团队合作。因此，它是软件开发过程中不可或缺的重要环节。

也就是：

> 提高软件的质量、降低开发风险、提高开发效率、满足用户需求、提高软件开发的可维护性、促进团队合作、消除软件危机

chapter 2

软件生命周期概念

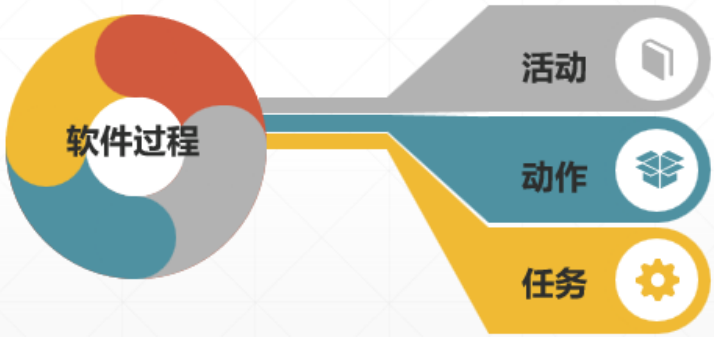
> 定义：
> 软件产品或软件系统从设计、投入使用到被淘汰的全过程

软件过程概念

> 定义：
> 软件过程是在工作产品构建过程中，所需完成的活动、动作和任务的集合

软件过程

软件过程是在工作产品构建过程中，所需完成的工作活动、动作和任务的集合。



活动主要实现宽泛的目标，与应用领域、项目大小、结果复杂性或者实施软件工程的重要程度没有直接关系。

动作包含了主要工作产品生产过程中的一系列任务。

任务关注小而明确的目标，能够产生实际产品。

常见的几种软件过程模型：瀑布、增量、原型、螺旋、敏捷等，比较各自优缺点

来自一位同学的博客：

软件过程模型	优点	缺点
瀑布模型	1.结构清晰，易于理解和管理 2. 适用于需求稳定的项目 3.易于控制和评估项目进展	1.需求变更困难 2.难以应对需求不明确或复杂的项目 3.高风险
增量模型	1.允许早期交付部分功能 2.更好地适应需求变更 3.缩短开发周期	1.集成和测试复杂性增加 2.需要明确定义增量之间的接口和集成策略 3.需要足够的技术能力和资源
原型模型	1.快速验证和演示概念 2.帮助理解需求和功能 3.反馈和修改成本低	1.原型与最终产品之间存在差异 2.可能忽视系统的内部结构和质量 3.需要客户积极参与
螺旋模型	1.强调风险管理 2.允许灵活性和迭代开发 3.适用于大型和复杂项目	1.技术能力和风险管理经验要求高 2. 项目管理和控制要求高 3.开发周期较长
敏捷模型	1.高度灵活 2.快速响应需求变更和客户反馈 3.强调团队合作和持续交付 4.鼓励客户参与,提高产品质量 5.通过迭代开发和持续集成提高产品质量。	1. 自组织和自我管理团队要求高 2. 需要客户积极参与和提供及时反馈 3. 需要良好的沟通和协作能力

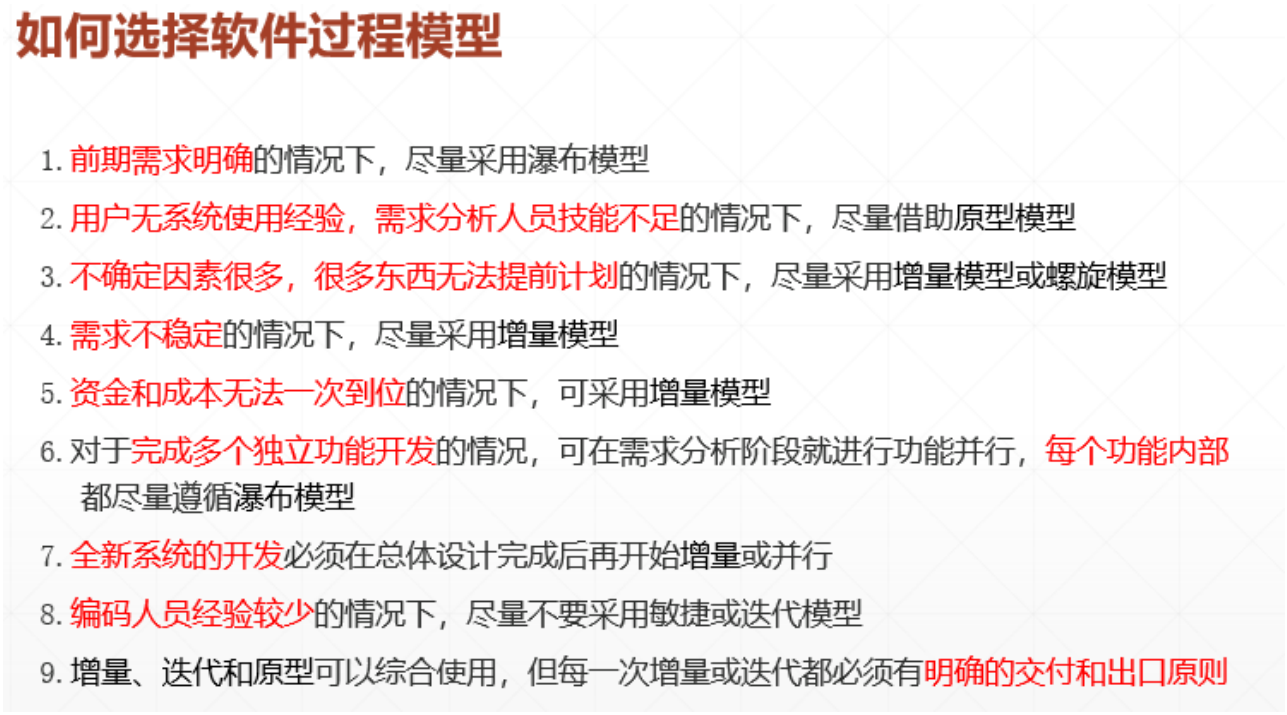
up根据ppt总结的：

软件过程模型	优点	缺点
瀑布模型	1.结构清晰，易于理解和管理2. 适用于需求稳定的项目3.易于控制和评估项目进展	
原型模型	减少需求不明确带来的风险	1、构造原型采用的技术和工具不一定主流 2、快速建立起来的系统加上连续的修改可能导致原型质量低下 3、设计者在质量和原型中进行折中 4、客户意识不到一些质量问题
增量模型	1、增量概念的引入，使得不需要提供完整的需求，只要有一个增量出现，开发就可以进行，软件能够更早投入市场。2、在项目初期，由于只开发部分系统，不需要投入太多的人力物力。3、整个开发过程中，产品逐步交付，软件开发能够较好地适应需求的变化，同时能够看到软件中间产品，提出改进意见，减少返工，降低开发风险。4、增量模型要求系统具有开放式体系结构，以便于增量的集成，同时也便于软件后期的维护。	1、开发者很难根据客户需求给出大小适合的增量。2、软件必须具备开放式体系结构，这在实践中往往是困难的3、如果不能进行良好的项目管理，采用增量模型的开发过程很容易退化成边做边改的方式，使软件过程控制失去整体性。
螺旋模型：	1、有助于目标软件的适应能力，支持用户需求的动态变化；2、原型可看作可执行的需求规格说明，易于为用户和开发人员共同理解，可作为继续开发的基础，为用户参与所有关键决策提供了方便；3、为项目管理人员及时调整管理决策提供了方便，进而降低开发风险。	1、如果每次迭代的效率不高，致使迭代次数过多，将会增加成本并推迟项目交付时间；2、使用该模型需要有相当丰富的风险评估经验和专门知识，要求开发队伍水平较高，否则会带来更大风险。
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	1、各个开发阶段是重叠的，在开发过程中需要大量的开发人员，因此不利于项目的管理 2、要求严格管理文档，使得审核的难度加大

软件过程模型	优点	缺点
基于构件的开发模型	1、软件复用思想2、降低开发成本和风险，加快开发进度，提高软件质量	1、模型复杂2、商业构件不能修改，会导致修改需求，进而导致系统不能完全符合客户需求3、无法完全控制所开发系统的演化4、项目划分的好坏直接影响项目结果的好坏
敏捷开发模型	对变化和不确定性有更快速更敏捷的反应，在快速的同时保持可持续的开发速度，能较好的地适应商业竞争环境下对项目提出的有限资源和有限开发时间的约束。	测试驱动开发可能会导致系统通过了测试但不是用户期望的，重构而不降低系统体系结构的质量是困难的。

瀑布模型的图附在下面：

过程模型的选择：（虽然大纲上没有，但感觉应该会考）



chapter 3

需求分析的概念

> 概念：
> 确定系统必须具有的功能和性能，系统要求的运行环境，并且预测系统发展的前景。
> 换句话说需求就是以一种清晰、简洁、一致且无二义性的方式，对一个待开发系统中各个有意义方面的陈述的一个集合

需求分析的过程：

> 过程：
> 需求确认与需求变更
>
> 其中需求确认包括：
> 需求获取、需求提炼、需求描述、需求验证

需求确认的步骤：

> 步骤：需求获取→需求提炼→需求描述→需求验证

需求分析三类建模：

> 功能模型、数据模型、行为模型。

面向对象的需求分析过程 中，三类模型各包含哪些内容？

	面向过程的需求分析	面向对象的需求分析
数据模型	实体-联系图（ERD） 数据字典（DD）	类图、类关系图
功能模型	数据流图（DFD）	用例图
行为模型	状态变迁图（STD）	活动图、时序图、状态图

UML图类型有哪些？ 掌握用例图和活动图作法。泳道划分活动图、分支及汇合、分叉及合并

九种常用的UML图总结_uml有哪些图-CSDN博客

文心一言：

> UML图是用于描述软件系统的各种图，它们帮助开发人员理解和管理复杂的系统。以下是UML图的一些主要类型：
>
> 类图（Class Diagram）：类图是描述系统中类的静态结构的图。它显示了类之间的关系，如关联、继承和聚合。
> 用例图（Use Case Diagram）：用例图用于描述系统的功能需求和参与者与系统的交互。
> 序列图（Sequence Diagram）：序列图显示了在系统中的对象如何交互以完成特定用例的过程。它强调时间顺序和对象间的消息传递。
> 协作图（Collaboration Diagram）：协作图与序列图相似，但更强调对象之间的关系而不是消息的顺序。
> 活动图（Activity Diagram）：活动图用于描述系统中的工作流程和逻辑流程。
> 状态图（State Diagram）：状态图用于描述对象的状态转换和可能的路径。
> 包图（Package Diagram）：包图用于描述系统的包和包之间的关系。
> 部署图（Deployment Diagram）：部署图用于描述系统在物理环境中的部署，包括硬件和软件组件的分布。
> 接口图（Interface Diagram）：接口图用于描述系统中类或组件的接口及其实现。
>
> 这些是UML图的一些常见类型，每种类型都有其特定的用途和表示方法。在实际的软件开发过程中，根据需要选择适当的UML图类型来描述和可视化系统。

用例图和活动图作法省略，大家可以去哔哩哔哩找教程看一下（

推荐一个up，感觉讲的很清楚：爱喝水的胖猫的个人空间-爱喝水的胖猫个人主页-哔哩哔哩视频 (bilibili.com)

chapter4待定。。。

系统设计分为概要设计和详细设计

> 软件设计的定义：在[IEEE610.12-90]中，软件设计定义为软件系统或组件的架构、构件、接口和其他特性的定义过程及该过程的结果。

与设计相关的**8**个概念：抽象、体系结构、设计模式、模块化、信息隐藏、功能独立、细化、重构。

抽象：忽略具体的信息将不同事物看成相同事物的过程
体系结构：软件的整体结构和这种结构为系统提供概念上完整性的方式
设计模式：在给定上下文环境中一类共同问题的共同解决方案
模块化：软件被划分为命名和功能相对独立的多个组件（通常称为模块），通过这些组件的集成来满足问题的需求
信息隐藏：模块应该具有彼此相互隐藏的特性 即：模块定义和设计时应当保证模块内的信息（过程和数据）不可以被不需要 这些信息的其他模块访问
功能独立：每个模块只负责需求中特定的子功能，并且从程序结构的其他部分看，该模块具有简单的接口

好处：
易于开发：功能被划分，接口被简化
易于维护（和测试）：次生影响有限，错误传递减少，模块重用

精化：逐步求精的过程
重构：不改变组件功能和行为条件下，简化组件设计（或代码）的一种重组技术

系统设计从体系结构、数据、接口和组件四方面进行设计。

面向过程和面向对象的系统设计，各自包含哪些设计内容？

GPT的答案

> 面向过程的系统设计：
>
> 1. 流程设计（***Procedural Design***）：
>   - 定义：强调解决问题的步骤和流程。
>   - 内容：算法、流程图、数据流图等。
> 2. 模块化设计（***Modular Design***）：
>   - 定义：将系统分解为独立的模块。
>   - 内容：定义模块的功能、输入和输出，模块之间的调用关系。
> 3. 顺序性设计（***Sequential Design***）：
>   - 定义：强调操作的顺序和执行流。
>   - 内容：确定操作的执行顺序，关注数据流程。
> 4. 层次化设计（***Hierarchical Design***）：
>   - 定义：按照层次结构组织系统组件。
>   - 内容：划分层次，定义层次间的接口。
> 5. 数据设计（***Data Design***）：
>   - 定义：定义数据结构和存储方式。

- 内容：数据表、记录结构、文件设计。

面向对象的系统设计：

1. 类设计（***Class Design***）：
    - 定义：基于对象的概念，设计类和类之间的关系。
    - 内容：类的属性、方法，继承关系，关联关系。
2. 对象设计（***Object Design***）：
    - 定义：确定系统中的对象及其行为。
    - 内容：对象的属性、方法，对象之间的交互。
3. 封装（***Encapsulation***）：
    - 定义：将数据和方法封装在对象中。
    - 内容：定义公共接口，隐藏实现细节。
4. 继承（***Inheritance***）：
    - 定义：实现代码的重用和扩展。
    - 内容：定义类之间的继承关系，子类继承父类的属性和方法。
5. 多态（***Polymorphism***）：
    - 定义：同一操作在不同对象上具有不同的行为。
    - 内容：方法重载、方法覆盖，接口实现。
6. 关联和聚合（***Association and Aggregation***）：
    - 定义：描述对象之间的关系。
    - 内容：定义对象之间的关联关系，包括一对一、一对多等。
7. 设计模式应用（***Design Patterns***）：
    - 定义：采用通用解决方案来解决特定问题。
    - 内容：应用常见设计模式，如工厂模式、观察者模式等。
8. 用例建模（***Use Case Modeling***）：
    - 定义：定义系统的功能需求。
    - 内容：识别用例，描述用例的行为。

总体而言，面向过程的系统设计更注重流程、模块和数据的组织，而面向对象的系统设计更注重对象、类、继承和多态等概念的应用。在实际应用中，很多系统设计可能会结合两者的特点，使用面向对象的设计思想来构建模块化、可维护且可扩展的系统。

chapter 5

质量保证**QA**：

定义：系统地监测和评估一个工程的各个方面，以最大限度地〃高正在由生产过程中 实现的质量的最低标准

软件质量保证**SQA**：

定义：遵照一定的软件生产标准、过程和步骤对软件质量进行评估的活动

软件质量的概念和关键点。

概念：明确表示是否符合功能和性能要求，明确地记载开发标准和所有专业开发软件的期望的隐性特点

关键点：

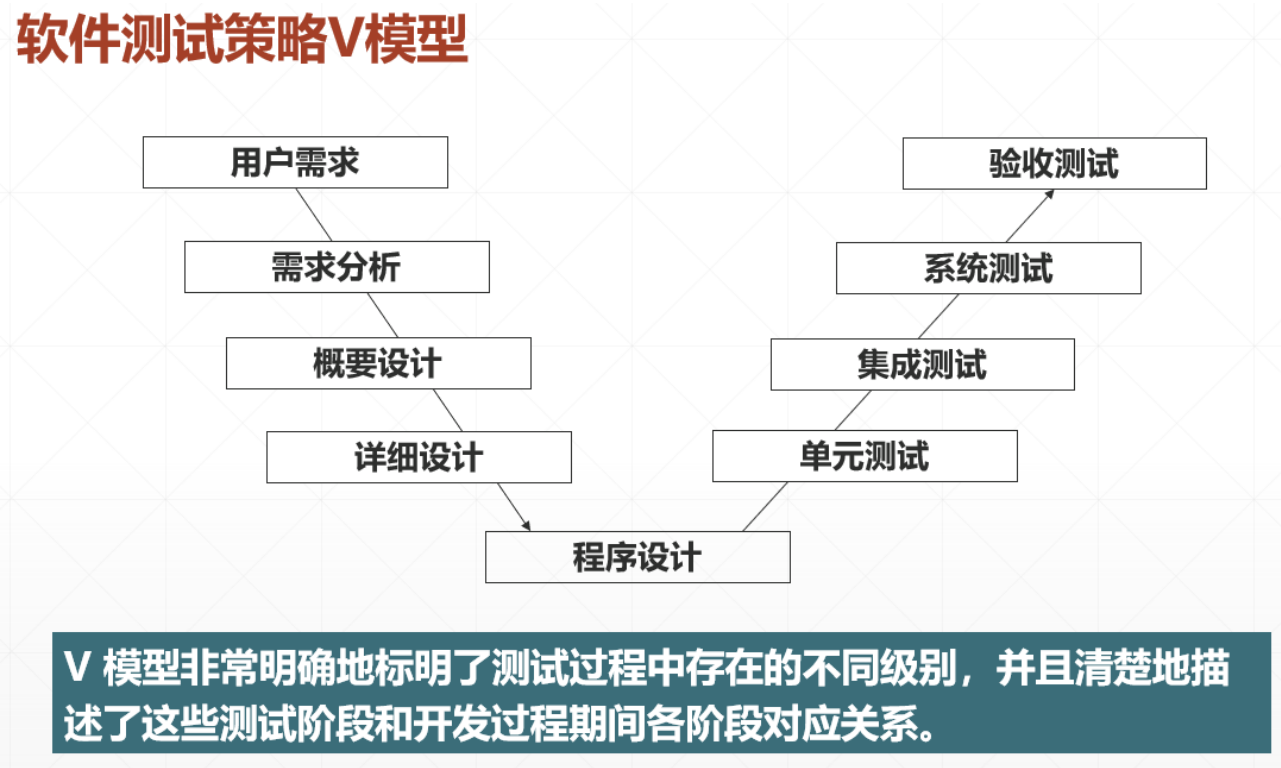
1、符合明确规定的功能和性能要求，即需求中的明确约定（前期约定）

2、符合明确的开发标准，包括行业、国家、国际标准等（明确标准）

3、符合所有软件开发专业的共性、隐性标准，如易用性、可维护性等（共性期望）

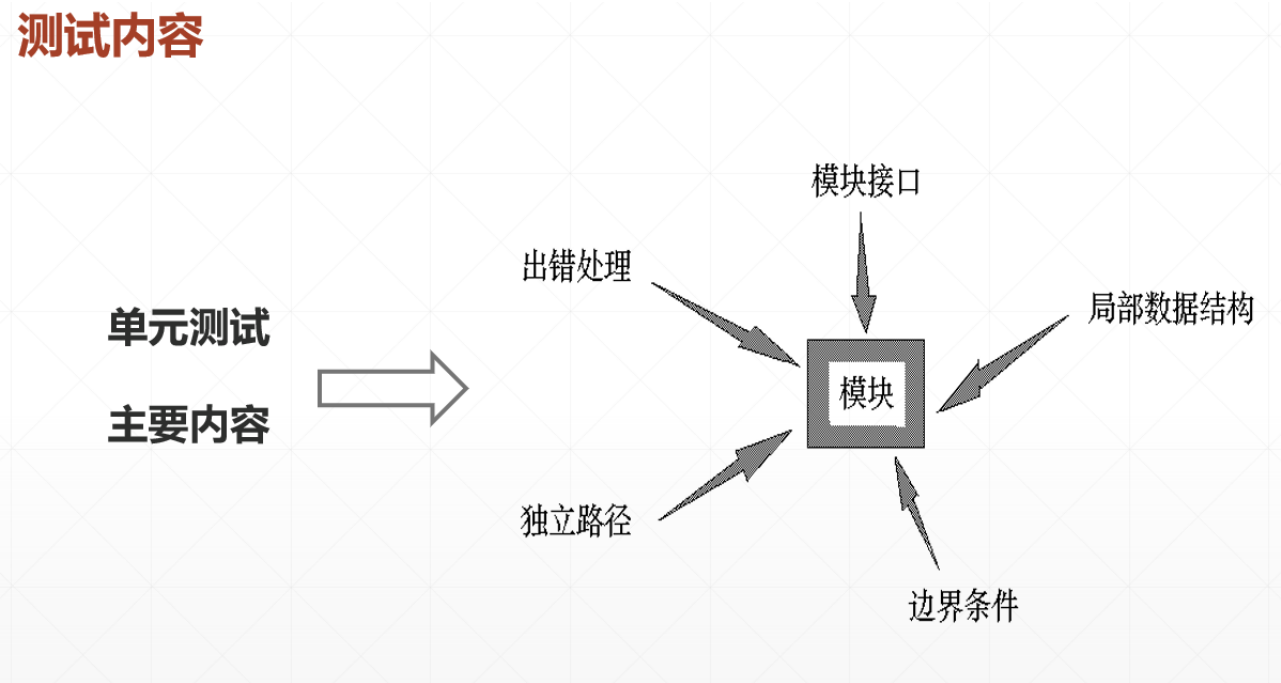
测试策略**V**模型概念，测试与开发的各阶段对应关系。

水平方向互相对应:



单元测试的内容、集成测试的分类、系统测试的分类、验收测试的分类。

单元测试内容：



集成测试的分类：

自顶向下的集成方法：该集成方式将模块按系统程序结构，沿控制层次自顶向下进行集成。
优点--自顶向下的集成方式在测试过程中较早地验证了主要的控制和判断点。选用按深度方向集成的方式，可以首先实现和验　　证一个完整的软件功能。
缺点--桩的开发量较大

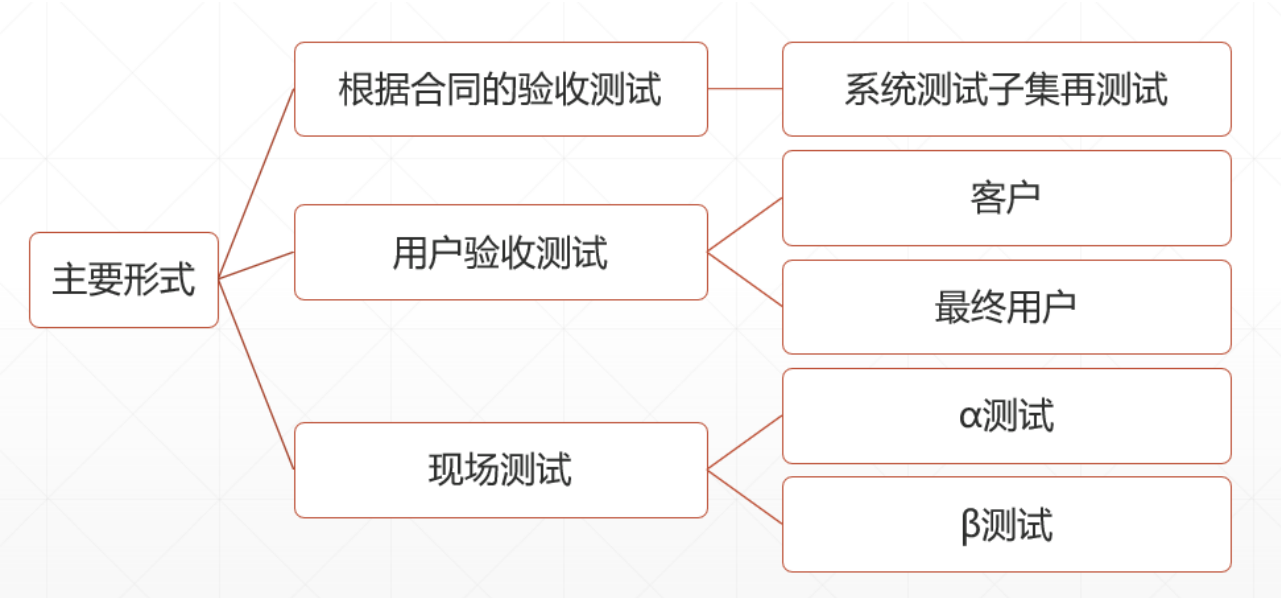
自底向上的集成方法：从软件结构最底层的模块开始，按照接口依赖关系逐层向上集成以进行测试。
优点--每个模块调用其他底层模块都已经测试，不需要桩模块
缺点--每个模块都必须编写驱动模块；缺陷的隔离和定位不如自顶向下。

SMOKE方法：将已经转换为代码的软件构件集成为构造（build）。一个构造包括所有的数据文件、库、可复用的模块以及实现一个或多个产品功能所需的工程化构件。

系统测试分类：

blog：系统测试分类和测试常用方法-CSDN博客

验收测试分类：



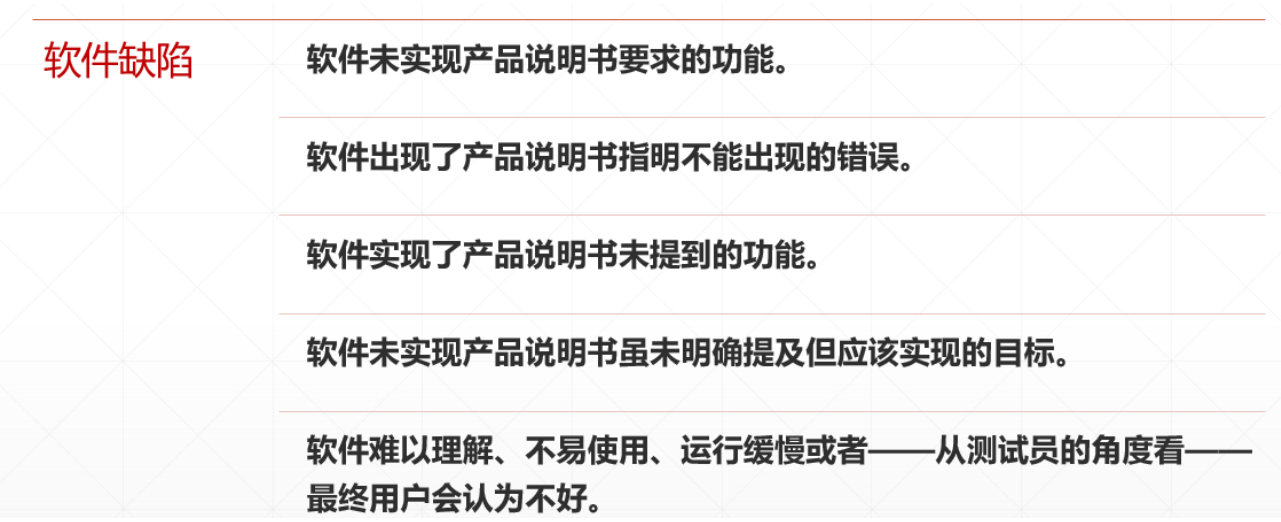
回归测试的概念

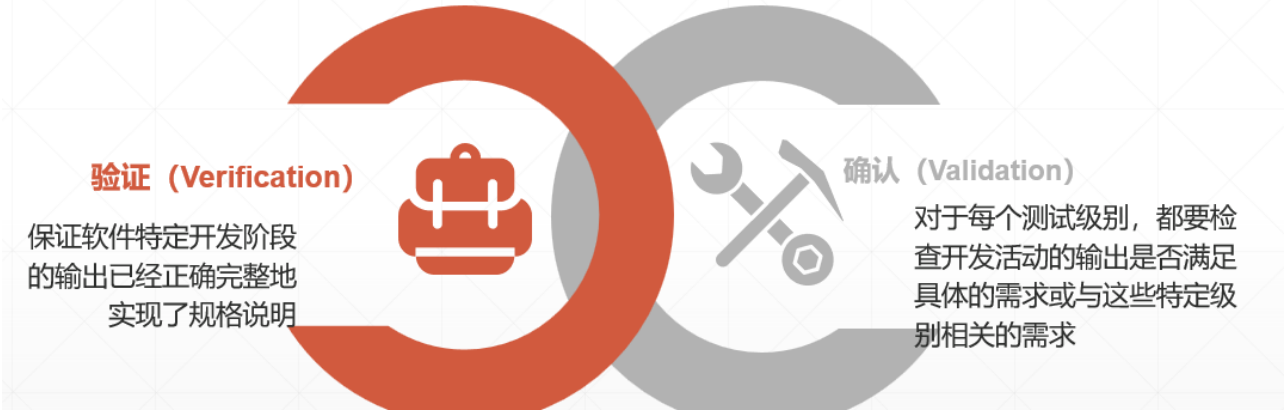
回归测试：指有选择地重新测试系统或其组件，以验证对软件的修改没有导致不希望出现的影响，以及系统或组件仍然符合其指定的需求。

测试技术常见术语的概念：软件缺陷、验证和确认、测试与质量保证、质量与可靠性、调试与测试、测试用例

主要测试方法：

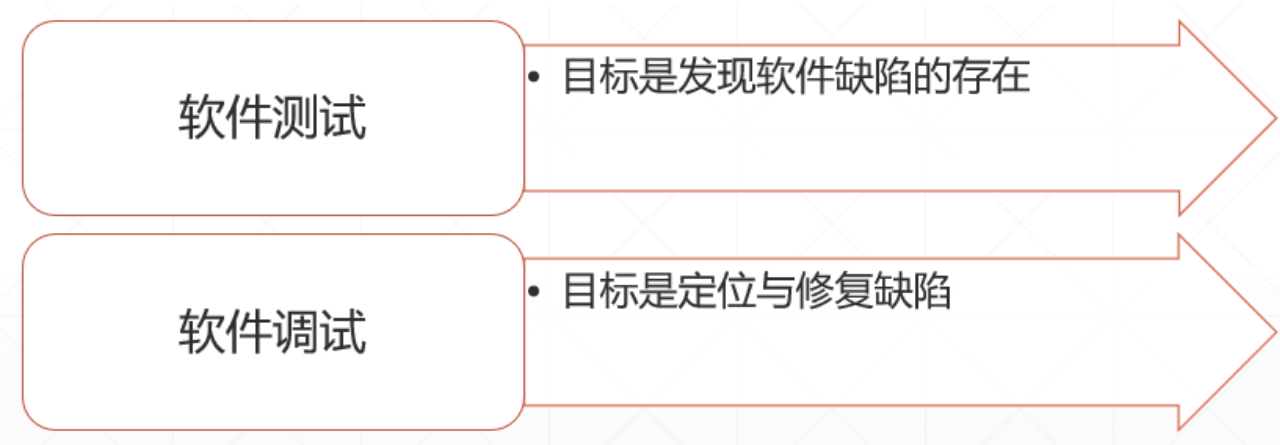
根据PPT总结：




验证（Verification）
保证软件特定开发阶段的输出已经正确完整地实现了规格说明
确认（Validation）
对于每个测试级别，都要检查开发活动的输出是否满足具体的需求或与这些特定级别相关的需求


软件测试
• 找出软件缺陷，并确保修复
软件质量保证
• 创建、执行改进过程并防止缺陷的标准和方法


质量与可靠性
可靠性（reliability）
功能性（functionality）
可维护性（maintainability）
可移植性（portability）
可用性（usability）
效率（efficiency）
01
02
03
04
05
06



测试用例（test case）：是测试输入、执行条件、以及预期结果的集合，是为特定的目的开发的，例如执行特定的程序路径或验证与指定的需求相符合。

输入条件	执行条件	预期结果
用户名 =yiyh，密码为空	输入用户名称，按 “登陆” 按钮。	显示警告信息 “请输入用户名和密码！”

输入条件	预期结果	实际结果
典型值		

主题。设计者、类型、测试名称、状态、描述、优先级、comment、步骤名、步骤描述、预期结果、评审人、评审备注、评审时间等……

来自文心一言：

以下是这些概念的解释：

软件缺陷：软件缺陷也被称为Bug，是计算机软件或程序中存在的某种破坏正常运行能力的问题、错误，或者隐藏的功能缺陷。缺陷的存在会导致软件产品在某种程度上不能满足用户的需要。

验证和确认：验证是通过提供客观证据对规定要求已得到满足的认定。确认是通过提供客观证据对特定的预期用途或应用要求已得到满足的认定。简单说，“验证”就是检查最终产品是否达到顾客使用要求，“确认”则是要证明所提供的（或将要提供的）产品适合其预计的用途。

测试与质量保证：测试通常也称为质量控制，是确保某种产品或服务质量的预防措施或方法。质量保证是过程中质量的过程或用于生产优质产品的方法。测试涉及产品在非常受控和各种条件下的实际操作或使用，结果被写入、编目、存储和解释，以评估产品是否符合公司和市场的规定标准。

质量与可靠性：质量主要指产品或过程满足需求的能力，而可靠性则是产品在长时间使用或多次使用中的表现。在某些情况下，质量和可靠性是密不可分的。如果一个产品在长时间使用中一直保持良好的性能，那么它的质量和可靠性都很好。
调试与测试：调试是在测试之后进行的活动，目的是修复在测试阶段发现的问题。测试的目的是发现错误，而调试的目的是修复错误。

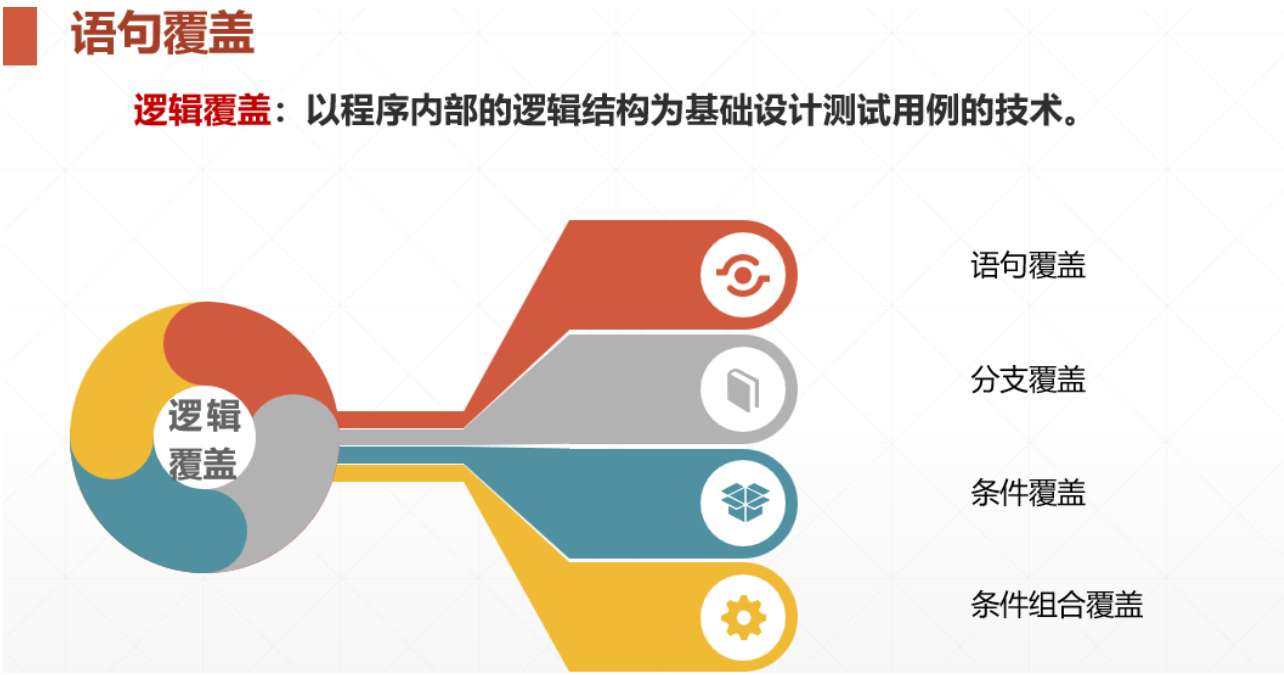
测试用例：测试用例是指对一项特定的软件产品进行测试任务的描述，体现测试方案、方法、技术和策略。其内容包括测试目标、测试环境、输入数据、测试步骤、预期结果等，用于评价一个程序和系统的特性或能力，并确定是否达到期望的结果。
这些概念都与软件开发和维护的过程紧密相关，每个阶段都对软件的最终质量和性能起着关键作用。如需更详细的解释，可以咨询计算机软件专业人士。

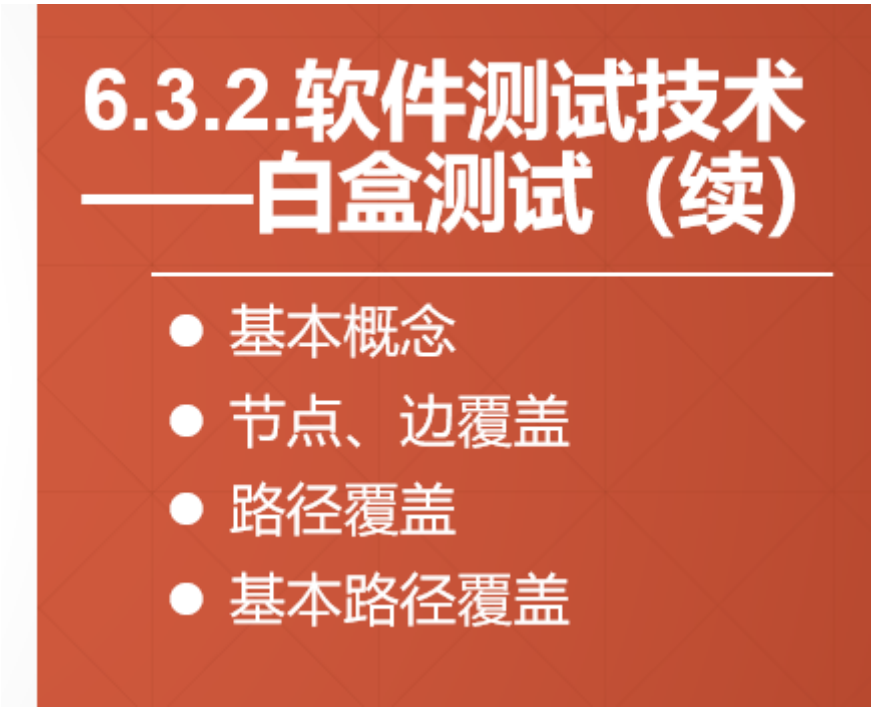
来自同学的blog:

1. **软件缺陷（Software Defect）：**
    1. **概念：** 软件中的错误或问题，可能导致程序在特定条件下产生不正确的结果。
    2. **说明：** 缺陷是指程序代码或设计中的错误，通常需要修复以提高软件的质量。
2. **验证与确认（Verification and Validation）：**
    1. **概念：** 验证是确认系统是否符合规格说明，而确认是确认系统是否满足最终用户的需求。
    2. **说明：** 验证主要关注构建过程，而确认关注系统交付给最终用户后的验证。
3. **测试与质量保证（Testing and Quality Assurance）：**
    1. **概念：** 测试是通过执行系统来发现软件缺陷的过程，而质量保证是一系列过程和活动，旨在确保软件满足预期的质量标准。
    2. **说明：** 测试是质量保证的一个组成部分，而质量保证包括整个软件开发周期中的各种活动，以确保最终交付的产品质量。
4. **质量与可靠性（Quality and Reliability）：**
    1. **概念：** 质量是软件满足规格和用户需求的程度，而可靠性是系统在一段时间内能够正常工作的概率。
    2. **说明：** 质量关注软件的特征，而可靠性关注软件的稳定性和可靠程度。
5. **调试与测试（Debugging and Testing）：**
    1. **概念：** 调试是识别和修复已知或潜在的缺陷的过程，而测试是通过执行软件以发现新的缺陷。
    2. **说明：** 调试通常在编码阶段进行，而测试可以在不同阶段执行，包括单元测试、集成测试和系统测试。
6. **测试用例（Test Case）：**
    1. **概念：** 一组输入、执行条件和预期结果，用于验证程序在特定条件下的正确性。
    2. **说明：** 测试用例是测试的基本单位，通过执行测试用例可以评估软件在不同情况下的行为。

白盒测试、黑盒测试、静态分析各有哪些方法？

白盒测试：





黑盒测试：

黑盒测试（GPT的答案）：

等价类划分（Equivalence Partitioning）：
方法： 将输入划分为不同的等价类，从每个等价类中选择测试用例。
目的： 通过测试一组代表性的输入，验证系统对每个等价类的处理能力。

边界值分析（Boundary Value Analysis）：
方法： 测试输入的边界值，包括最小值、最大值和中间值。
目的： 验证系统在输入边界处的行为，通常这些边界值是导致错误的关键点。

错误推测（Error Guessing）：
方法： 基于经验和直觉推测可能存在的错误，并设计相应的测试用例。
目的： 利用测试人员的经验来发现潜在的问题，尤其是在规格说明不完整的情况下。

状态转换测试（State Transition Testing）：
方法： 针对系统中的状态进行测试，关注状态之间的转换。
目的： 验证系统在不同状态下的行为，尤其是状态切换时的正确性

静态分析：

掌握等价类划分测试方法。（有效等价类和无效等价类划分、对应测试用例的设 计）

哔哩哔哩叭，概括不了一点儿。。。

chapter6

项目管理四要素：人员、产品、项目、过程（概念）

软件度量有哪些方法 **???**

我也不是很确定。。。

这是PPT上找到的：

> 面向规模度量：
> 面向功能点度量：
> 面向对象度量：
> 面向用例度量：

网上blog找到的资料：

> 生产率估计（基于规模（KLOC）、基于功能点（FP））、工作量度量（算法成本模型、COCOMO模型）。