**简答题**

1. **软件工程的作用/必要性（Software Engineering）**

软件工程是建立和使用一套合理的工程原则，以便经济地获得可靠地、可以在实际机器上高效运行地软件；

IEEE给出的定义：（1）将系统化的、规范的、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。

软件过程构成了软件项目惯例控制的基础，建立了工作环境以便于应用技术方法、提交工作产品、建立里程碑、保证质量及正确管理变更；

软件工程方法为构件软件提供技术上的解决方法；

软件工程工具为过程和方法提供自动化或半自动化的支持；

1. **瀑布模型的优劣（The merits of the waterfall model）**

优点

①为项目提供了按阶段划分的检查点；

②当前一阶段完成后，只需要去关注后续阶段；

③可在迭代模型中应用瀑布模型；

④它提供了一个模板，这个模板使得分析、设计、编码、测试和支持的方法可以在该模板下有一个共同的指导；

缺点

①各个阶段的划分完全固定，阶段之间产生大量的文档，极大地增大了工作量。

②由于开发模型是线性的，用户只有等到整个过程的末期才能见到开发成果，从而增加了开发风险。

③通过过多的强制完成日期和里程碑来跟踪各个项目阶段；

④瀑布模型的突出缺点是不适应用户需求的变化；

1. **对象的概念、特征（The concept and characteristics of the object）**

**对象与传统数据对象的区别（The difference between object and traditional data）**

1）对象的定义：

定义1：对象是具有相同状态的一组操作集合。

定义2:由保存对象属性的数据和可以施加于这些数据的操作封装在一起构成的整体。对象是类的实例，对象给类以生命。

对象的特征：继承、封装、多态（面向对象）**不确定，保留**

对象与传统数据对象的区别：

对象是封装了数据结构及可以施加在这些数据结构上的操作的封装体，这个封装体有唯一标识它的名称，而且向外界提供一组服务（即共有的操作）。传统的数据是可见的，可被修改的，而实现对象操作的代码和数据是隐藏在对象内部的，在外面看不见的，更不能从外面去访问或修改这些数据或代码。对象将传统的数据进行封装处理。

1. **实体类、边界类、控制类（Entity class boundary class control class）**

实体类：实体对象的抽象，通常来自域模型（现实世界）用来描述具体的实体，通常映射到数据库表格与文件中；

边界类：边界对象的抽象，通常是用来完成参与者（用户、外部系统）于系统之间交互的对象（例如：对话框、菜单、接口等）；

控制类：控制对象的抽象，主要用来体现应用程序的执行逻辑，将其抽象出来，可以使变化不影响用户界面和数据库中的表；

1. **高质量软件设计的特征（Characteristics of high quality software design）**

①设计应展示出这样一种结构：

1. 已经使用可识别的体系结构风格或模式创建；
2. 由展示出良好设计特征的构件构成；
3. 能够以演化的方式实现从而便于实现和测试；

②设计应该模块化：也就是说应将软件逻辑地划分为元素和子系统；

③设计应该包含数据、体系结构、接口和构件的清晰表示；

④设计应导出数据结构，这些数据结构适用于要实现的类，并从可识别的数据模式提取。

⑤设计应导出显示独立功能特征的构件

⑥设计应导出接口

⑦设计的导出应根据软件需求分析过程中获取的信息采可重复的方法进行；

⑧应使用能够有效传达其意义的表示法来表达设计；

1. **体系结构的概念及其重要性（The concept and importance of Architecture）**

概念：程序或计算系统的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相互关系。

作用

①对设计在满足既定需求方面的有效性进行分析；

②在设计变更相对容易的阶段，考虑体系结构可能的选择方案；

③降低与软件构造相关的风险；

1. **模块独立性划分标准（Standard for module independence）**

①内聚性：又称为块内联系，指模块的功能强度的度量，即一个模块内部各个元素彼此结合的紧密程度的度量，如果一个模块内各元素联系得越紧密则它的内聚性就越高；

②耦合性：也称块间联系，指软件系统结构中各模块间相互联系紧密程度的一种度量。模块之间联系越紧密其耦合性就越强，模块的独立性就越差。

耦合性和内聚性是模块独立性的两个定性标准，将软件系统划分模块使尽量做到高内聚和低耦合，提高模块的独立性。

1. **基于类的构建设计原则（Component based design principles）**

①开闭原则：模块应该对外延具有开放性，对修改具有封闭性；

②Liskov原则：子类可以替换它们的基类；

③依赖倒置原则：依赖于抽象而非具体实现

④接口分离原则：多个客户专用接口比一个通用接口要好

⑤发布复用等价性原则：复用的粒度就是发布的粒度；

⑥共同封装原则：一同变更的类应该合在一起；

⑦共同复用原则：不能一起复用的类不能被分到一组；

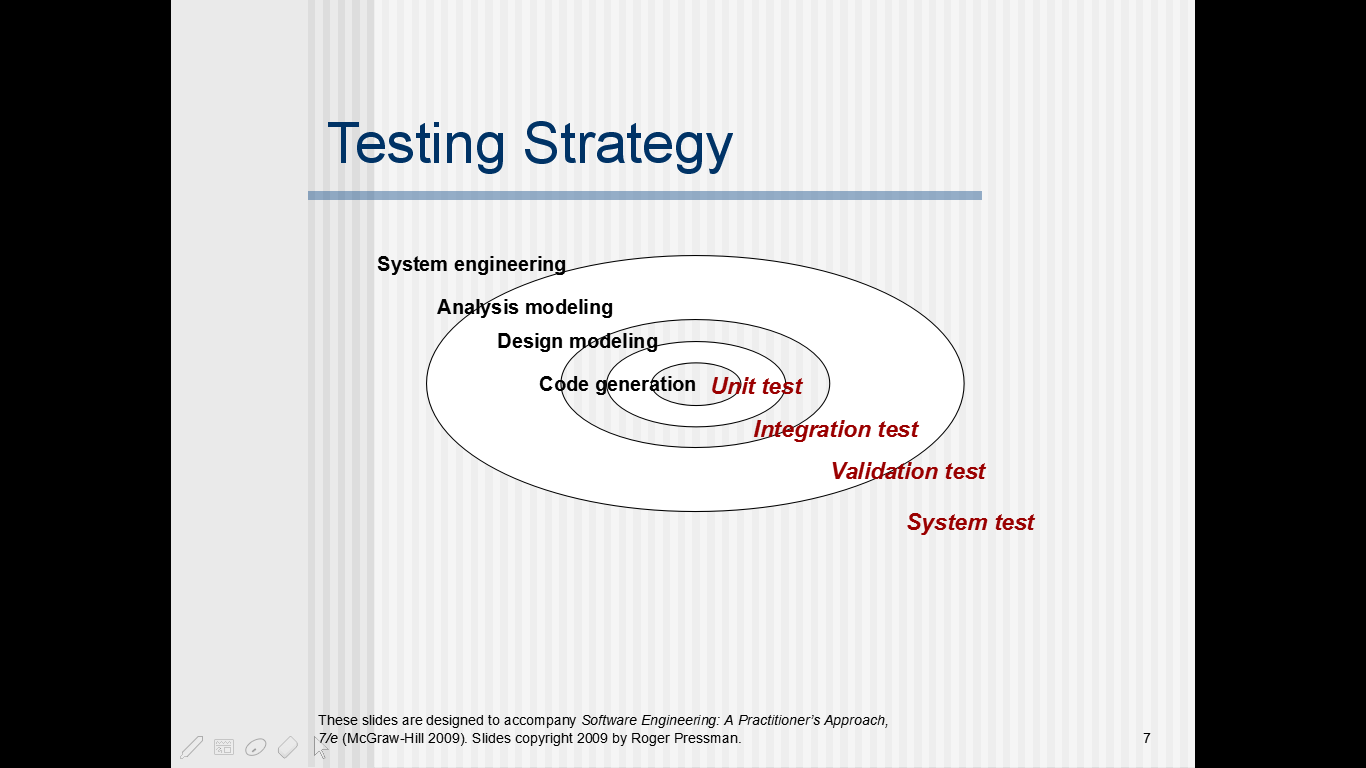
1. **界面设计黄金原则（Golden principles of interface design）**

①用户操纵控制

②减少用户的记忆负担

③保持界面一致

1. **软件测试层次（Various stages of software testing）**

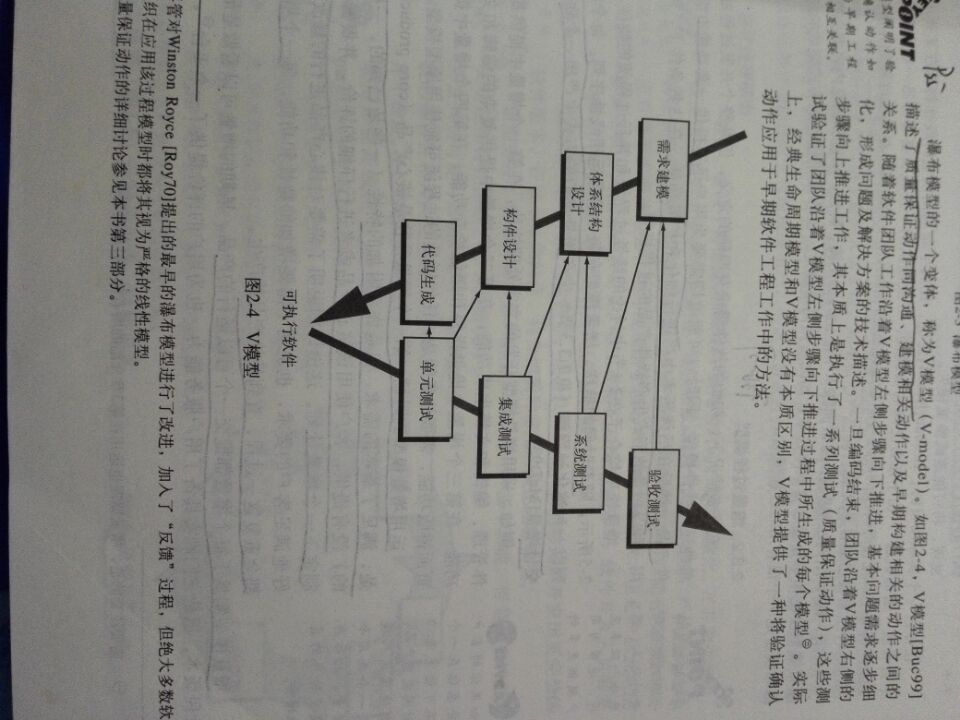


单元测试——详细设计/构建级设计

集成测试——体系结构设计

确认测试——需求

系统测试——系统工程



1. **黑盒、白盒测试（White box and black box testing method）**

黑盒测试

概念：也称行为测试，侧重于软件的功能需求，使软件工程师能设计出将测试程序所有功能需求的输入条件集；

适用场合：对程序内部结构不了解而只关心程序功能的场合

优点

①适用于各阶段测试；

②从产品功能角度测试；

③容易入手生成测试数据；

缺点

①某些代码得不到测试；

②如果规格说明有误，则无法发现；

③不易进行充分性测试

白盒测试

适用场合：更关注程序内部逻辑结构的测试

优点

①可构成测试数据使特定程序部分得到测试；

②有一定的充分性度量手段；

③可获得较多工具支持；

缺点

①不易生成测试数据；

②无法对未实现规格说明的部分进行测试；

③工作量大，通常用于单元测试，有应用局限

**选择填空**

1. **软件定义（the definition of software）**

软件是：

①指令的集合（计算机程序），通过执行这些指令可以满足预期的特征、功能和性能需求；

②数据结构。使得程序可以合理利用信息；

③软件描述信息（即文档），它以硬拷贝和虚拟形式存在，用来描述程序操作和使用；

1. **软件工程作用、目的**

目的：为高质量的软件开发提供一个科学的体系框架。

作用：将系统化的，规范化的，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；在以上所述方法中的研究。

1. **遗留系统（Legacy system）**

**遗留软件：**年代较久远，甚至过于久远了。具有生命周期长以及业务关键性的特点，同时存在质量差的特点。

**遗留系统发生演化（evolve）的原因**:

软件需要进行适应性调整，从而可以满足新的计算环境或者技术的需求；

软件必须升级以实现新的商业需求；

软件必须扩展使之具有与更多新的系统和数据库的互操作能力；

软件架构必须进行改建使之能适应多样化的网络环境。

1. **软件开发过程模型三要素**

过程、方法和工具

1. **五种框架活动（frame activities）**

沟通：包含了与客户（其他共同利益者）之间大量的交流和协作，还有需求获取和其他相关活动。

策划：为后续的软件工程工作制定计划。描述了需要执行的技术任务，可能的风险资金需求，工作产品，工作进度计划。

建模：需求分析:有助于开发者，客户更好的了解需求；设计：可以实现需求。

构建：偏码：手动，自动生成测试：为了发现偏码中的错误。

部署：软件交付到用户，用户对其进行评测并给出反馈意见。

1. **惯例模型种类（prescriptive process models）**

1）惯例模型种类：瀑布模型，增量过程模型，演化过程模型（原型开发和螺旋模型），协同模型，

2）原型法的适用场合：原型模型比瀑布模型更符合人们认识事物的过程和规律，是一种较实用的开发框架。 它适合于那些不能预先确切定义需求的软件系统的开发，更适合于那些项目组成员（包括分析员、设计员、程序员和用户）不能很好交流或通信有困难的情况。

3）瀑布模型的适用场合：瀑布模型一般适用于功能、性能明确、完整、无重大变化的软件系统的开发。

1. **螺旋模型的特征（spiral model）**

螺旋模型结合了原型的迭代性质和瀑布模型的系统性和可控性的特点。它是一种风险驱动型的过程模型生成器，对于软件集中的系统，它可以指导多个利益相关者的协同工作。它有两个显著的特点：一是采用循环的方式逐步加深系统定义和实现的深度，同时降低风险；二是确定一系列里程，确保利益相关者都支持可行的和令人满意的系统解决方案。

1. **软件开发各阶段任务**

1）问题的定义及规划

此阶段是软件开发与需求放共同讨论，主要确定软件的开发目标及其可行性。

2）需求分析

在确定软件开发可行性的情况下，对软件需要实现的各个功能进行详细需求分析。需求分析阶段是一个很重要的阶段，这一阶段做的好，将为整个软件项目的开发打下良好的基础。“唯一不变的是变化本身”，同样软件需求也是在软件爱你开发过程中不断变化和深入的，因此，我们必须定制需求变更计划来应付这种变化，以保护整个项目的正常进行。

3）软件设计

此阶段中偶要根据需求分析的结果，对整个软件系统进行设计，如系统框架设计、数据库设计等。软件设计一般分为总体设计和详细设计。还的软件设计将为软件程序编写打下良好的基础。

4）程序编码

此阶段是将软件设计的结果转化为计算机可运行的程序代码。在程序编码中必定要制定统一、符合标准的编写规范。以保证程序的可读性、易维护性。提高程序的运行效率。

5）软件测试

在软件设计完成之后要进行严密的测试，一发现软件在整个软件设计过程中存在的问题并加以纠正。整个测试阶段分为单元测试、组装测试、系统测试三个阶段进行。测试方法主要有白盒测试和黑盒测试。

1. **建模（modeling）（分析模型包括哪些)（Analysis model）**

分析模型（需求模型）由各种元素组成：基于场景（用例）、基于数据（数据模型）、基于类、基于流和行为。

1. **对象的概念、特征（object）**

1）对象的定义：

定义1：对象是具有相同状态的一组操作集合。

定义2：对象是对问题域中某个东西的抽象，这种抽象反应了系统保存有关这个东西的信息或与它交互的能力。也就是说，对象是对属性值和操作的封装。

定义3：对象：：=〈ID，MS，DS，MI〉。其中，ID是对象的标识或名字，MS是对象中的操作。

2）对象的特征：

1. **结构化方法（Structured Method）、面向对象方法**

1）结构化方法（SD方法）是一种传统的软件开发方法，它是由结构化分析、结构化设计和结构化程序设计三部分有机组合而成的。它的基本思想：把一个复杂问题的求解过程分阶段进行，而且这种分解是自顶向下，逐层分解，使得每个阶段处理的问题都控制在人们容易理解和处理的范围内。

2）面向对象方法(Object-Oriented Method)是一种把面向对象的思想应用于软件开发过程中，指导开发活动的系统方法，简称OO (Object-Oriented)方法，是建立在“对象”概念基础上的方法学。

1. **软件设计目的（Software Design）**

软件设计的目标是创作出坚固、适用和令人愉悦的模型或表示。

1. **软件设计模型包括哪些元素**

数据、体系结构、构件和接口

1. **主要体系结构有哪些种类（Architectural Design）**

典型的基本体系结构：功能结构、实现结构、并发结构、物理结构、开发结构。

1. **体系结构中目标系统所处环境的定义（同级系统、上级系统）**

与目标系统（为该系统所开发的体系结构设计）交互的系统可以表示为：

1. 上级系统：这些系统把目标系统作为某些高层处理方案的一部分。
2. 下级系统：这些系统把目标系统使用，并为完成目标系统的功能提供必要的数据和处理。
3. 同级系统：这些系统在对等的基础上相互作用（即信息或者由同级系统和目标系统产生，或者被目标系统和统计系统使用）。
4. 参与者：通过产生和消耗必要处理所需的信息，实现与目标系统交互的（人、设备）
5. **软件测试目的（证明系统缺陷）（Software Testing）**

软件测试的目的是发现软件设计和实现过程中的疏忽所造成的错误。

1. **软件测试有哪几个阶段及与软件开发阶段的对应关系（同简答10）**
2. **软件重用的概念（Software Reuse）**

软件重用是指在两次或多次不同的软件开发过程中重复使用相同或相似软件元素的过程。

1. **UML各种图的应用场合**
2. 类图。类图描述一组类、接口、协作和它们之间的关系。在OO系统建模中，最常见的图就是类图。类图给出了系统的静态设计视图，活动类的类图给出了系统的静态进程视图。   
   2)对象图。对象图描述一组对象及它们之间的关系。对象图描述了在类图中所建立的事物实例的静态快照。和类图一样，这些图给出系统的静态设计视图或静态进程视图，但它们是从真实实例或原型案例的角度建立的   
   3)构件图。构件图描述一个封装的类和它的接口、端口，以及由内嵌的构建和连接件构成的内部结构。构件图用于表示系统的静态设计实现视图。对于由小的部件构建大的系统来说，构件图是很重要的。构建图是类图的变体   
   4)组合结构图。组合结构图描述结构化类（如构件或类）的内部结构，包括结构化类与系统其余部分的交互点。组合结构图用于画出结构化类的内部内容   
   5)用例图。用例图描述一组用例、参与者及它们之间的关系。用例图给出系统的静态用例视图。这些图在对系统的行为进行组织和建模时是非常重要的。   
   6)顺序图。顺序图是一种交互图，交互图展现了一种交互，由一组对象或参与者以及它们之间可能发送的消息构成。交互图专注预提性的动态视图。顺序图是强调消息的时间次序的交互图。   
   7)通信图。通信图也是一种交互图，强调收发消息的对象或参与者的结构组织，该图反应了对象之间的消息交互，于顺序图想死，但与顺序图不同的是，协作图不但描述了对象之间的交互还描述了交互对象之间的链接关系。即通信图同时反映了系统的动态和静态特征。通信图成为协作图。   
   8)定时图。定时图也是一种交互图，强调消息跨越不同对象或参与者的实际时间，而不仅仅只是关心消息的相对顺序。   
   9)状态图。状态图描述一个状态机，由状态、转移、时间和活动组成。状态图给出了对象的动态视图。它对于接口、类或协作的行为建模尤为重要，而且它强调事件导致的对象行为，有助于对反应式系统建模。   
   10)活动图。活动图将进程或其他计算结构展示为计算内部一步步的控制流和数据流。活动图专注于系统的动态视图。它对系统的功能建模和业务流程建模特别重要，并强调对象间的控制流程。   
   11)部署图。部署图描述对运行时的处理节点及在其中生存构建的配置。部署图给出了架构的静态部署视图，通常一个节点包含一个或多个部署图。   
   12)制品图。制品图描述计算机中的一个系统的物理结构。制品包括文件、数据库和类似物理比特集合。制品图通常与部署图一起使用。制品也给出了他们实现的类和构建   
   13)包图。包图描述由模型本身分解而成的组织单元，以及他们之间的依赖关系。   
   14)交互概览图。交互概览图是活动图和顺序图的混合物。   
   其中结构性视图包括类图、对象图、包图、组合结构图、构件图、部署图和制品图，而行为性视图包括用例图、顺序图、通信图、定时图、状态图、活动图、交互概览图。其中顺序图、通信图、定时图和交互该蓝图统称为交互图。

**CHARTER 1**

应用领域——application domain

软件特点——characteristics of software

框架活动——framework activities

遗留软件——legacy software

实践——practice

原则——principles

软件工程——software engineering

软件神话——software myths

软件过程——software process

普适性活动——umbrella activities

**CHAPTER 2**

基于构件的开发——component-based development

并发模型——concurrent models

演化过程模型——evolutionary process models

形式化方法模型——formal methods models

通用过程模型——generic process models

增量过程模型——increment process models

个体软件过程——personal software models

惯用过程模型——prescriptive process models

过程模型——process patterns

任务集——task set

团队软件过程——team software process

统一过程——united process

**CHAPTER 3**

自适应软件开发——adaptive software development

敏捷过程——agile process

敏捷统一过程——agile unified process

敏捷性——agility

Crystal

DSDM——动态系统开发方法

Scrum

FDD——特征驱动开发

极限编程——extreme programming

工业极限编程——Industrial XP

精益软件开发——LSD（lean software development）

结对编程——pair programming

重构——refactoring

**CHARTER 5**

分析模型——analysis model

分析模式——analysis patterns

协作——collaboration

精化——elaboration

导出——elicitation

起始——inception

谈判——negotiation

质量功能部署——quality function development

需求工程——requirements engineering

需求管理——requirements management

规格说明——specification

利益相关者——stakeholder

用例——use cases

确认需求——validating requirements

**CHAPTER 6**

活动图——activity diagram

分析类——analysis classes

分析包——analysis packages

关联——associations

基于类建模——class-based modeling

CRC建模——class responsibility collaborator(CRC)

数据建模——data modeling

域分析——domain analysis

语法分析——grammatical parse

需求建模——requirement modeling

基于场景建模——scenario-based modeling

泳道图——swimlane

**CHAPTER 7**

分析模型——analysis patterns

行为模型——behavior model

配置模型——configuration model

内容模型——content model

控制流模型——control flow model

数据流模型——data flow model

功能模型——function model

交互模型——interaction model

导航建模——navigation modeling

处理规格说明——process specification

顺序图——sequence diagrams

**CHAPTER 8**

抽象——abstraction

体系结构——architecture

方面——aspects

内聚——cohesion

数据设计——data design

设计过程——design process

功能独立——functional independence

良好的设计——good design

信息隐蔽——information hiding

模块化——modularity

面向对象设计——object-oriented design

模式——patterns

质量属性——quality attributes

质量指导原则——quality guidelines

重构——refactoring

关注点分离——separation of concerns

逐步精化——stepwise refinement

**CHAPTER 9**

原型——archetype

体系结构描述——architectural description

体系结构——architecture

候选方案——alternatives

构件——component

复杂度——complexity

以数据流为中心——data centered

数据流——data flow

类型——genres

分层——layered

求精——refinement

模板——template

ATAM——体系结构权衡分析方法

分解——factoring

实例化——instantiation

映射——mapping

**CHAPTER 10**

内聚性——cohesion

构件——component

分类——classifying

适应——adaptation

组合——composition

合格性——qualification

传统的——traditional

基于构件的开发——component-based development

内容设计——content design

耦合——coupling

设计准则——design guidelines

领域工程——domain engineering

**CHAPTER 11**

可访问性——accessibility

命令标签——command labeling

设计评估——design evaluation

错误处理——error handling

黄金规则——golden rules

帮助设施——help facilities

界面——interface

一致性——consistent

记忆负担——memory load

响应时间——response time

任务分析——task analysis

任务细化——task elaboration

可用性——usability

用户分析——user analysis

**CHAPTER 17**

α测试——alpha test

β测试——beta test

类测试——class testing

配置评审——configuration review

调试——debugging

部署测试——deployment testing

独立测试组——independent test group

集成测试——integration testing

回归测试——regression testing

系统测试——system testing

单元测试——unit testing

确认测试——validation testing

V&V（验证与确认）——verification and validation

**CHAPTER 18**

基本路径测试——basis path testing

黑盒测试——black-box testing

边界值分析——boundary value analysis

控制结构测试——control structure testing

等价划分——equivalence partitioning

流图——flow graph

白盒测试——white-box testing

**CHAPTER 19**

类测试——class testing

簇测试——cluster testing

基于故障的测试——fault-based testing

多类测试——multiple class testing

划分测试——partition testing

随机测试——random testing