软件工程题目

笔记本： 考研专业课
创建时间： 2018/11/2 19:57 **更新时间：** 2018/12/23 12:14
作者： 15801850335@163.com
URL： https://www.cnblogs.com/gaofei-1/p/6435442.html

名词解释：https://wenku.baidu.com/view/0a742f330b4c2e3f57276374.html
https://wenku.baidu.com/view/8b7c25145ef7ba0d4b733bee.html?rec_flag=default&sxts=1542699549021
全面内容：https://wenku.baidu.com/view/f019cbe06c175f0e7dd13780.html?rec_flag=default&sxts=1543826956924
题目：https://wenku.baidu.com/view/fdb5162f0066f5335a81216b.html
https://wenku.baidu.com/view/aee3ff2478563c1ec5da50e2524de518964bd3d7.html

16. 软件过程

软件过程是人们用于开发和维护软件及其相关过程的一系列活动，包括软件工程活动和软件管理活动。

17. 软件过程能力

软件过程能力是描述（开发组织或项目组）遵循其软件过程能够实现预期结果的程度，它既可对整个软件开发组织而言，也可对一个软件项目而言。

18. 软件过程性能

软件过程性能表示（开发组织或项目组）遵循其软件过程所得到的实际结果，软件过程性能描述的是已得到的实际结果，而软件过程能力则描述的是最可能的预期结果，它既可对整个软件开发组织而言，也可对一个特定项目而言。

19. 软件过程成熟度

软件过程成熟度是指一个特定软件过程被明确和有效地定义，管理测量和控制的程度。

20. 软件成熟度等级

软件成熟度等级是指软件开发组织在走向成熟的途中几个具有明确定义的表示软件过程能力成熟度的平台。

21. 关键过程域

每个软件能力成熟度等级包含若干个对该成熟度等级至关重要的过程域，它们的实施对达到该成熟度等级的目标起到保证作用，这些过程域就称为该成熟度等级的关键过程域。

22. 关键实践

关键实践是指对关键过程域的实践起关键作用的方针、规程、措施、活动以及相关基础设施的建立。

23. 软件能力成熟度模型

软件能力成熟度模型是指随着软件组织定义、实施、测量、控制和改进其软件过程，软件组织的能力也伴随着这些阶段逐步前进，完成对软件组织进化阶段的描述模型。

2010 软件工程真题

一. 名词解释：

1. 软件工程的用例驱动：用例获取系统的功能需求，它们“驱动”需求分析之后的所有阶段的开发。
2. ER 图：实体-关系图简称为 ER 图。通常，使用 ER 图来建立数据模型，用 ER 图描绘的数据模型也称为 ER 模型。
3. 模块化：将软件分成具有一定结构的模块的过程称为模块化。
4. α 测试和β 测试：

   α 测试是由一个用户在开发者的场所来进行的，软件在开发者对用户的“指导”下进行测试，开发者负责记录错误和使用中出现的问题，所以， α 测试是在一个受控的环境中进行的。

   β 测试是由软件的最终用户在一个或多个用户场所来进行的，不象α 测试，开发者通常不在测试现场，因此β 测试是软件在一个开发者不能控制的环境中的“活的”应用。

5. 请简述CMM的概念以及五级标准的含义。

答：CMM即能力成熟度模型是用于评价软件机构的软件过程能力成熟度的模型。是对于软件组织在定义、实施、度量、控制和改善其软件过程的实践中各个发展阶段的描述。

①初始级（无秩序的）

②可重复级（有一定的规范，可以复用一些成功的项目）

③已定义级（定性）

④已管理级（定量）

⑤优化级（新思想、新方法、通过反馈不断改进）

二. 简单题：

1. 敏捷软件开发宣言是轻量级开发方式的宣言，宣言强调它和重量级开发方式的不同。请简要叙述宣言的内容并概述其中一个敏捷过程模型。

   答：**宣言内容：**

   1) 个体和交互胜过过程和工具；
   2) 可工作软件胜过宽泛的文档；
   3) 客户合作胜过合同谈判；
   4) 响应变化胜过遵循计划。

   一个敏捷过程模型：

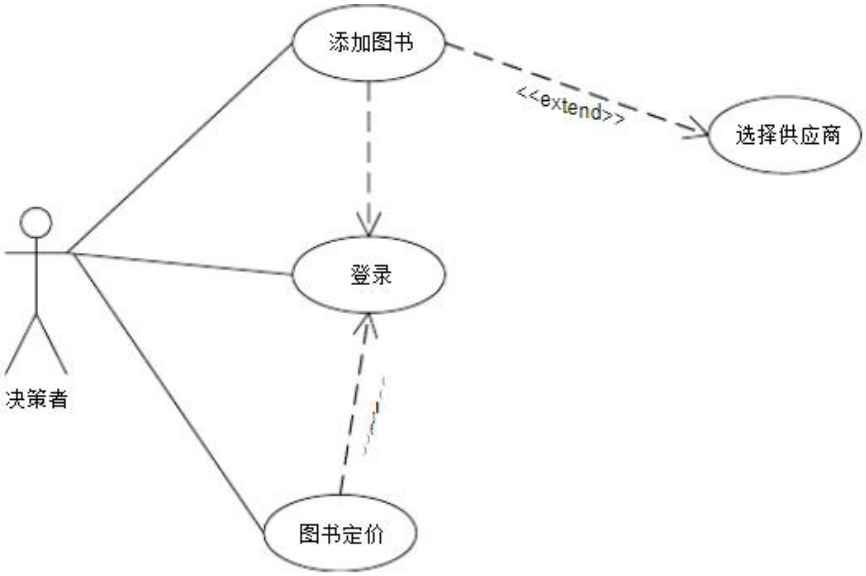
   极限编程（Extreme Programming，XP），生命周期：策划、设计、编码、测试。XP 原则：现场客户、简单设计、测试驱动、结对编程、代码全体拥有、持续集成、小型发布

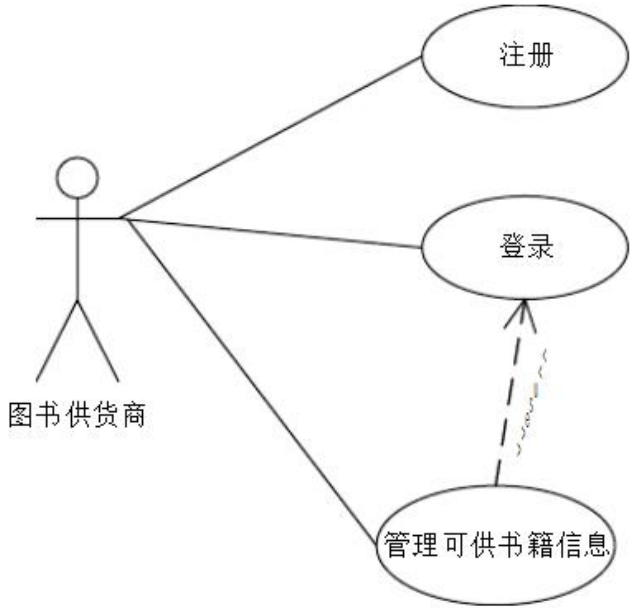
2. 判断题：

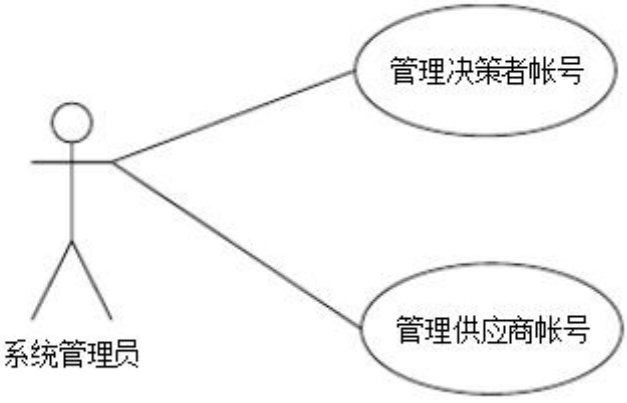
   1) 概要设计时应昼加强模块间的联系 （错）
   2) CMM 分 5 个阶段，从 1 级-5 级递增（对）
   3) 喷泉模型是典型的面向对象过程模型（对）
   4) 编码时应尽可能使用全局变量（错）
   5) 软件项目中，当进度出现问题的时候，应赶紧投入更多的人力，从而赶上计划。（错）

3. 根据网络书店采购，定价管理信息系统需求，区分用户、功能等，画 UML 图
“图书供货商”注册被审批后，可以登录系统发布其可提供的书籍信息，网络书店经营的“决策者”登录后可以添加待购书，其中，采购时可以选择以前系统中注册

过的“图书供货商”作为供货渠道，另外还可以为图书定价。“系统管理员”进行帐户管理。
具体来说：本系统提供给供应链上的“图书供货商”登录后输入可供图书商品信息。系统还提供网络书店经营“决策者”添加待购图书功能。其中可输入图书采购渠道，其采购渠道可以根据图书供应商，在注册后输入的供书信息进行选择，另外还可以进行图书价格确定等功能。“系统管理员”可以审批注册的“图书供应商”，日后也可以删除该供应商帐号，同时为“决策者”添加和删除相应帐户。
“决策者”无需注册，直接获得帐号和密码，可以在系统中修改密码。


添加图书
<<extend>>
选择供应商
登录
决策者
图书定价

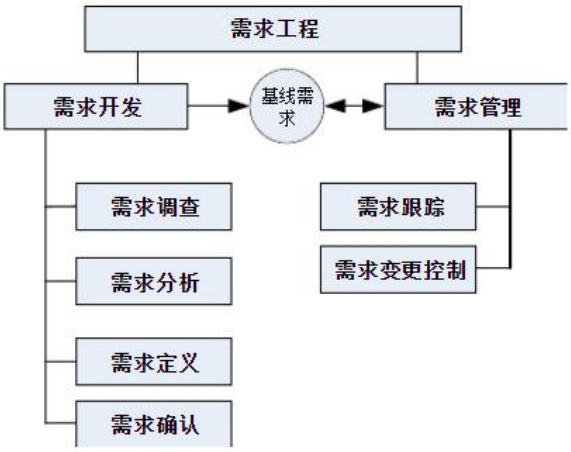

注册
登录
图书供货商
管理可供书籍信息


管理决策者帐号
管理供应商帐号
系统管理员

三． 论述题

对您参加与分析，设计或开发的一个软件项目作较为系统的简介，然后结合这个项目回答以下问题。

1）如何确定需求？确定需求的主要措施和经验。需求规格说明书包含哪些内容？



需求调查的目的是通过各种途径获取用户的需求信息（原始材料），产生《用户需求说明书》。

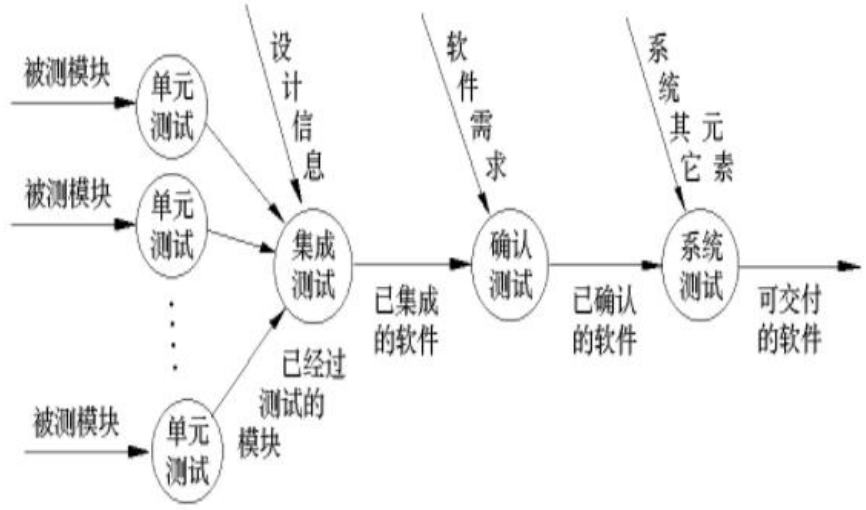
需求分析的目的是对各种需求信息进行分析，消除错误，刻画细节等。常见的需求分析方法有“问答分析法”和“建模分析法”两类。

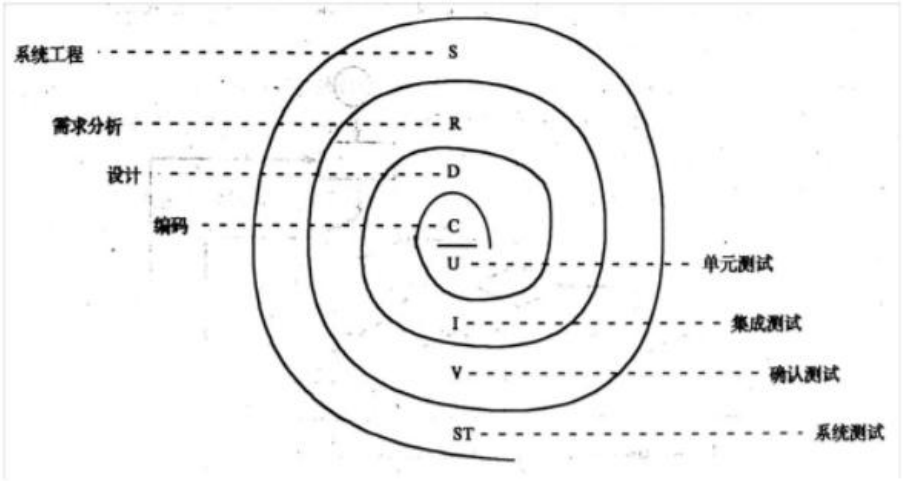
需求定义的目的是根据需求调查和需求分析的结果，进一步定义准确无误的产品需求，产生《产品需求规格说明书》。系统设计人员将依据《产品需求规格说明书》开展系统设计工作。

需求确认是指开发方和客户共同对需求文档进行评审，双方对需求达成共识后作出书面承诺，使需求文档具有商业合同效果。

需求说明书包括以下内容：项目背景、定义、参考资料、项目目标、假定和约束、功能需求、非功能需求等

2）项目进行过程中，需要做哪些测试？分别在哪些阶段进行？哪些工具及经验？





主要的测试工具：loadrunner、winrunner、QTP 等

3）如何进行软件配置管理？

a. 制订配置管理计划
b. 确定配置标识
c. 进行配置控制，实施变更管理
d. 配置审计
e. 记录并报告配置状态
f. 版本控制

1、需求工程：需求工程是软件工程的一个分支，它关注于软件系统所应予实现的现实世界目标、软件系统的功能和软件系统应当遵守的约束，同时它也关注以上因素和准确的软件行为规格说明之间的联系，关注以上因素与其随时间或跨产品族而演化之后的相关因素之间的联系。

2、需求：IEEE对需求的定义为：
①用户为了解决问题或达到某些目标所需要的条件或能力。
②系统或系统部件为了满足合同、标准、规范或其他正式文档所规定的要求而需要具备的条件或能力。
③对①或②中的一个条件或一种能力的一种文档化表述。

3、需求分析：需求分析是利用建模与分析技术对获取笔录的内容进行明确、整理、汇总，建立一个综合考虑问题域特性和需求的系统模型，然后根据系统模型将用户需求转化为系统需求的需求工程活动。

28、基线：基线是已经通过正式评审和批准的规格说明或产品，可以作为进一步开发的基础，并且只有通过正式的变更控制过程才能修改它。

29、需求基线：需求基线（Requirements Baseline）就是被明确和固定的需求集合，是项目团队需要在某一特定产品版本中实现的特征和需求集合。

30、需求跟踪：需求跟踪是一种有效的控制手段，能够在涉众的需求变化中协调系统的演化，保持各项开发工作对需求的一致性。需求跟踪是以软件需求规格说明文档为基线，在向前和向后两个方向上，描述需求以及跟踪需求变化的能力，分为前向跟踪（Pre－Traceabmty）和后向跟踪（Post－Traceability）两种。

16、简要说明结构化分析方法的局限性。
答：
结构化分析方法也有自身的局限性。
首先，虽然有了功能实体矩阵、实体生命历史和事件实体矩阵等分析技术，但是数据需求和处理需求的联接仍然不是一个容易的工作。
其次，结构化分析向结构化设计的过度(数据流图到结构图)中间有着难以处理的鸿沟。
再次，结构化分析过于重视对已有系统的建模，而这常常是难以实现的。

D、需求工程的定义：
参考答案：需求工程是指应用已证实有效的技术、方法进行需求分析，确定客户需求，帮助分析人员理解问题并定义目标系统的所有外部特征的一门学科。

基线：基线是指已经通过正式评审和批准的某规约或产品（一个版本），可作为进一步开发的基础，并只能通过正式的变更控制规程被改变。

多态性：多态性（Polymorphism）是指在父类中定义的属性或服务被子类继承后，可以具有不同的数据类型或表现出不同的行为。多态就是同一操作（方法）作用于不同的对象时，可以有不同的解释，产生不同的执行结果。

DFD图

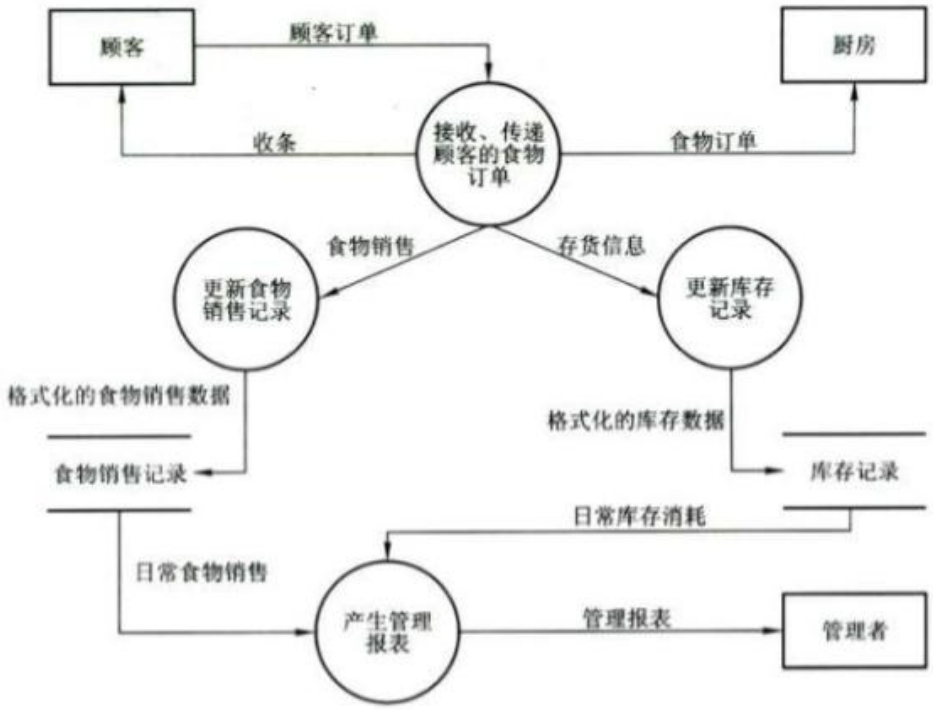


图 1用 DeMarco-Yourdon表示法表示的 DFD

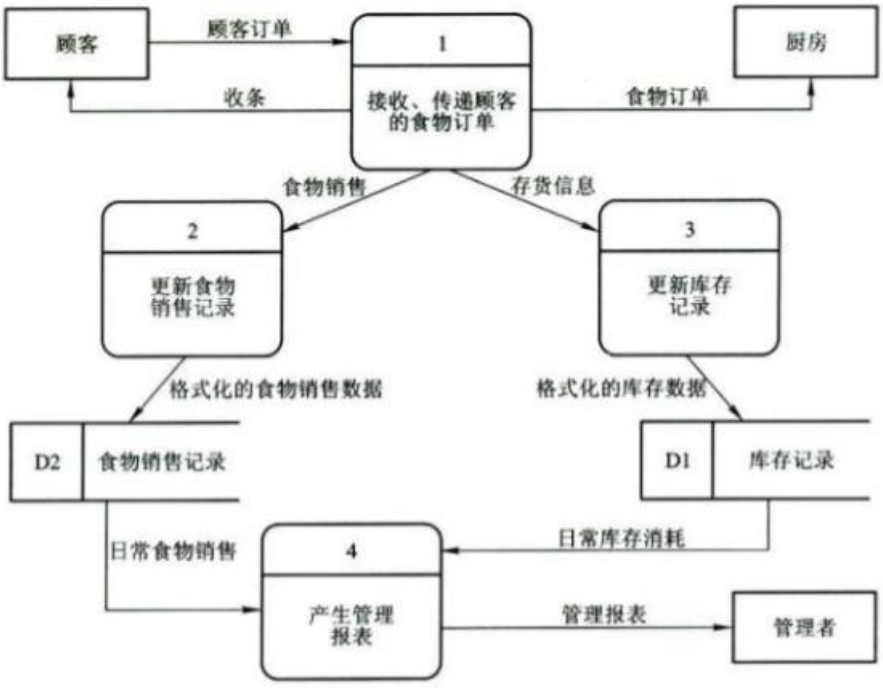


图 2用 Gane-Sarson表示法表示的 DFD

体系风格

编辑

对软件体系结构风格的研究和实践促进了对设计的复用，一些经过实践证实的解决方案也可以可靠地用于解决新的问题。体系结构风格的不变部分使不同的系统可以共享同一个实现代码。只要系统是使用常用的、规范的方法来组织，就可使别的设计者很容易地理解系统的体系结构。例如，如果某人把系统描述为"客户/服务器"模式，则不必给出设计细节，我们立刻就会明白系统是如何组织和工作的。

下面是Garlan和Shaw对通用体系结构风格的分类：

（1）数据流风格：批处理序列；管道/过滤器

（2）调用/返回风格：主程序/子程序；面向对象风格；层次结构

（3）独立构件风格：进程通讯；事件系统

（4）虚拟机风格：解释器；基于规则的系统

（5）仓库风格：数据库系统；超文本系统；黑板系统

收藏 | 1143

设计模式（设计模式概念）

编辑

设计模式（Design Pattern）是一套被反复使用、多数人知晓的、经过分类的、代码设计经验的总结。

使用设计模式的目的：为了代码可重用性、让代码更容易被他人理解、保证代码可靠性。 设计模式使代码编写真正工程化；设计模式是软件工程的基石脉络，如同大厦的结构一样。

设计模式百度百科链接：

https://baike.baidu.com/item/%E8%AE%BE%E8%AE%A1%E6%A8%A1%E5%BC%8F/1212549?fr=aladdin

总体来说设计模式分为三大类：

创建型模式，共五种：工厂方法模式、抽象工厂模式、单例模式、建造者模式、原型模式。

结构型模式，共七种：适配器模式、装饰器模式、代理模式、外观模式、桥接模式、组合模式、享元模式。

行为型模式，共十一种：策略模式、模板方法模式、观察者模式、迭代子模式、责任链模式、命令模式、备忘录模式、状态模式、访问者模式、中介者模式、解释器模式。

二、设计模式的六大原则

1、开闭原则（Open Close Principle）

开闭原则就是说对扩展开放，对修改关闭。在程序需要进行拓展的时候，不能去修改原有的代码，实现一个热插拔的效果。所以一句话概括就是：为了使程序的扩展性好，易于维护和升级。想要达到这样的效果，我们需要使用接口和抽象类，后面的具体设计中我们会提到这点。

2、里氏代换原则（Liskov Substitution Principle）

里氏代换原则(Liskov Substitution Principle LSP)面向对象设计的基本原则之一。 里氏代换原则中说，任何基类可以出现的地方，子类一定可以出现。 LSP是继承复用的基石，只有当衍生类可以替换掉基类，软件单位的功能不受到影响时，基类才能真正被复用，而衍生类也能够在基类的基础上增加新的行为。里氏代换原则是对"开-闭"原则的补充。实现"开-闭"原则的关键步骤就是抽象化。而基类与子类的继承关系就是抽象化的具体实现，所以里氏代换原则是对实现抽象化的具体步骤的规范。—— From Baidu 百科

3、依赖倒转原则（Dependence Inversion Principle）

这个是开闭原则的基础，具体内容：真对接口编程，依赖于抽象而不依赖于具体。

4、接口隔离原则（Interface Segregation Principle）

这个原则的意思是：使用多个隔离的接口，比使用单个接口要好。还是一个降低类之间的耦合度的意思，从这儿我们看出，其实设计模式就是一个软件的设计思想，从大型软件架构出发，为了升级和维护方便。所以上文中多次出现：降低依赖，降低耦合。

5、迪米特法则（最少知道原则）（Demeter Principle）

为什么叫最少知道原则，就是说：一个实体应当尽量少的与其他实体之间发生相互作用，使得系统功能模块相对独立。

6、合成复用原则（Composite Reuse Principle）

原则是尽量使用合成/聚合的方式，而不是使用继承。

1、单一职责原则

不要存在多于一个导致类变更的原因，也就是说每个类应该实现单一的职责，如若不然，就应该把类拆分。

2、里氏替换原则（Liskov Substitution Principle）

里氏代换原则(Liskov Substitution Principle LSP)面向对象设计的基本原则之一。 里氏代换原则中说，任何基类可以出现的地方，子类一定可以出现。 LSP是继承复用的基石，只有当衍生类可以替换掉基类，软件单位的功能不受到影响时，基类才能真正被复用，而衍生类也能够在基类的基础上增加新的行为。里氏代换原则是对“开-闭”原则的补充。实现“开-闭”原则的关键步骤就是抽象化。而基类与子类的继承关系就是抽象化的具体实现，所以里氏代换原则是对实现抽象化的具体步骤的规范。—— From Baidu 百科

历史替换原则中，子类对父类的方法尽量不要重写和重载。因为父类代表了定义好的结构，通过这个规范的接口与外界交互，子类不应该随便破坏它。

3、依赖倒转原则（Dependence Inversion Principle）

这个是开闭原则的基础，具体内容：面向接口编程，依赖于抽象而不依赖于具体。写代码时用到具体类时，不与具体类交互，而与具体类的上层接口交互。

4、接口隔离原则（Interface Segregation Principle）

这个原则的意思是：每个接口中不存在子类用不到却必须实现的方法，如果不然，就要将接口拆分。使用多个隔离的接口，比使用单个接口（多个接口方法集合到一个的接口）要好。

5、迪米特法则（最少知道原则）（Demeter Principle）

就是说：一个类对自己依赖的类知道的越少越好。也就是说无论被依赖的类多么复杂，都应该将逻辑封装在方法的内部，通过public方法提供给外部。这样当被依赖的类变化时，才能最小的影响该类。

最少知道原则的另一个表达方式是：只与直接的朋友通信。类之间只要有耦合关系，就叫朋友关系。耦合分为依赖、关联、聚合、组合等。我们称出现为成员变量、方法参数、方法返回值中的类为直接朋友。局部变量、临时变量则不是直接的朋友。我们要求陌生的类不要作为局部变量出现在类中。

6、合成复用原则（Composite Reuse Principle）

原则是尽量首先使用合成/聚合的方式，而不是使用继承。

依赖倒转原则

指在软件里面，把父类都替换成它的子类，程序的行为没有变化。简单的说，子类型能够替换掉它们的父类型。依赖性倒转其实可以说是面向对象设计的标志，用哪种语言编程并不是很重要。

接口隔离原则

定制服务的例子，每一个接口应该是一种角色，不多不少，不干不该干的事，该干的事都要干。

合成/聚合复用

合成/聚合复用原则（Composite/Aggregate Reuse Principle，CARP）经常又叫做合成复用原则。合成/聚合复用原则就是在一个新的对象里面使用一些已有的对象，使之成为新对象的一部分；新的对象通过向这些对象的委派达到复用已有功能的目的。它的设计原则是：要尽量使用合成/聚合，尽量不要使用继承。

依赖倒置原则 编辑

依赖倒置原则

A.高层次的模块不应该依赖于低层次的模块，他们都应该依赖于抽象。

B.抽象不应该依赖于具体实现，具体实现应该依赖于抽象。

概述 编辑

依赖倒置原则（Dependence Inversion Principle）是程序要依赖于抽象接口，不要依赖于具体实现。简单的说就是要求对抽象进行编程，不要对实现进行编程，这样就降低了客户与实现模块间的耦合。

软件重构 编辑

软件重构是指在不改变软件的功能和外部可见性的情况下，为了改善软件的结构，提高软件的清晰性、可扩展性和可重用性而对其进行的改造。简而言之，重构就是改进已经写好的软件的设计。

一、软件过程

软件过程：是人们用于开发和维护软件及其相关过程的一系列活动，包括软件工程活动和软件管理活动。它定义了运用方法的顺序、应该交付的资料、为保证软件质量和协调变化所需要采取的管理措施，以及标志软件开发各个阶段任务完成的里程碑。软件过程是为了获得高质量软件所需要完成的一系列任务的框架，他规定了完成各项任务的工作步骤。

瀑布模型：适用于大型项目，功能、需求十分明确，无重大变化的软件系统的开发。
特点、优点：

1. 强迫开发人员采用规范的方法；
2. 严格地规定了每个阶段必须提交的文档；
3. 要求每个阶段交出的所有产品都必须经过质量小组的仔细验证。

缺点：

1. 在开发初期阶段，要求作出正确、全面、完整的需求分析，这是十分困难的；
2. 缺乏灵活性，难以适应需求不明确或需求经常变化的软件开发；
3. 开发早期存在的问题，往往要到交付使用时才发现，维护代价大。

演化模型：
在获取了一组基本需求后，通过快速分析，构造出该软件的一个初始版本，称为原型，然后根据用户在试用原型的过程中提出的反馈对原型进行改进，获得原型的新版本。演化模型采用迭代的思想，渐进的开发，逐步完整软件版本。

增量模型：
增量模型融合了瀑布模型的基本成分和演化模型的迭代特征。把软件的开发过程分成若干个日程时间交错的线性序列，每个线性序列产生一个可发布的“增量”，后一版本是对前一版本的修改和补充，重复增量发布过程，直至最终完善产品。

统一过程模型：
统一过程（RUP/UP，Rational Unified Process）是一种以用例驱动、以体系结构为核心、迭代并且增量的软件过程模型，由UML方法和工具支持，广泛应用于各类面向对象项目。
RUP 把整个软件开发生命周期分为多个循环，每个循环由四个阶段组成，每个阶段完成确定的任务，结束前有一个里程碑评估本阶段的工作。

. 6个核心工作流：业务建模、需求、分析设计、实现、测试、部署

. 3个核心支持工作流：配置与变更管理、项目管理和环境

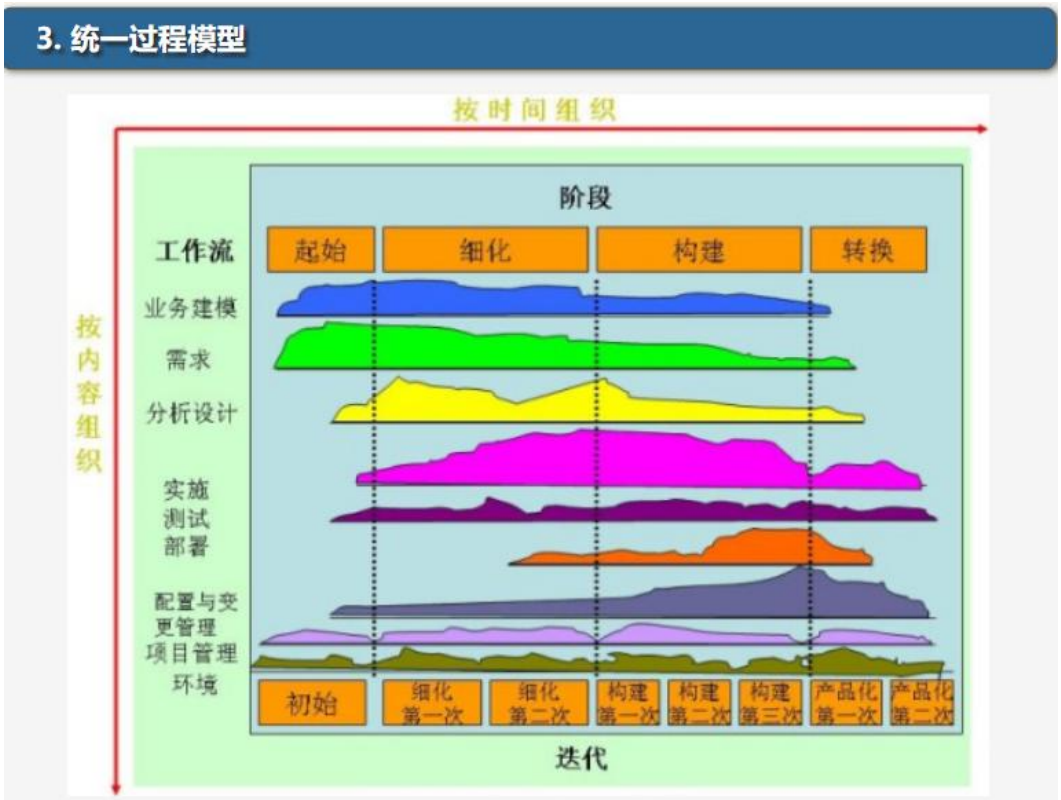


图 RUP 统一过程模型

RUP模型

UML过程的基本特征是：用例驱动，以架构为核心，迭代并且增量

用例驱动：

用例获取系统的功能需求，它们“驱动”需求分析之后的所有阶段的开发。在分析阶段，它们被用于获取所需的功能并经客户的确认。在设计和实现阶段，用例必须被实现，在测试阶段，用例用于验证系统。

以体系结构为中心：

首先定义一个基础的体系结构，然后将它原型化并加以评估，最后进行精化。体系结构给出系统的映象，它定义系统的不同部分、它们的关系和交互，它们的通信机制以及关于如何增加或修改体系结构中的成分的全部规则。体系结构涉及到功能性和非功能性两个方面，要定义一个易修改、易理解和允许复用的系统。

面向方面的软件开发

面向方面编程(Aspect Oriented Programming(AOP)），是目前软件开发中的一个热点，可以通过预编译方式和运行期动态代理实现在不修改源代码的情况下给程序动态统一添加功能的一种技术。利用AOP可以对业务逻辑的各个部分进行隔离，从而使得业务逻辑各部分之间的耦合度降低，提高程序的可重用性，同时提高了开发的效率。

螺旋模型：

螺旋模型是瀑布模型和演化模型的结合，并增加了风险分析。

喷泉模型：

支持面向对象开发的模型，以用户需求为动力，以对象为驱动的模型。软件开发过程自下向上周期的各阶段是相互重叠和多次反复的。体现迭代和无间隙的特征。

敏捷宣言：

1. “个体和交互”胜过“过程和工具”；
2. “可以使用的软件”胜过“宽泛的文档”；
3. “客户合作”胜过“客户谈判”；
4. “相应变化”胜过“遵循计划”。
5. 虽然右项也有价值，但我们认为左项价值更大。

敏捷开发特点：

根据维基百科上的定义：“（敏捷）更强调程序员团队与业务专家之间的紧密协作、面对面的沟通（认为比书面的文档更有效）、频繁交付新的软件版本、紧凑而自我组织型的团队、能够很好地适应需求变化的代码编写和团队组织方法，也更注重做为软件开发中人的作用。”

一个敏捷过程模型：

极限编程（Extreme Programming，XP），生命周期：策划、设计、编码、测试。XP 原则：现场客户、简单设计、测试驱动、结对编程、代码全体拥有、持续集成、小型发布

CMM即能力成熟度模型：是用于评价软件机构的软件过程能力成熟度的模型。是对于软件组织在定义、实施、度量、控制和改善其软件过程的实践中各个发展阶段的描述。
①初始级（无秩序的，甚至是混乱的，几乎没有什么过程是经过妥善定义的）
②可重复级（建立了一定的过程规范，可以重复以前类似项目所取得的成功）
③已定义级（定性、用于管理和工程活动的软件过程已经文档化和标准化，并集成到整个组织的软件过程中。所有项目使用文档化的，组织批准的过程来开发和维护软件）
④已管理级（定量、收集软件过程和产品质量的详细度量值，能定量地理解和控制过程和产品）
⑤优化级（量化反馈和先进的新思想、新技术、促使过程不断改进）
CMMI，能力成熟度模型集成：
若干过程模型的综合和改进，是支持多个工程学科和领域的系统的、一致的过程改进框架，能适应现代工程的特点和需要，能提高过程的质量和工作效率。

二、软件需求

软件需求：

1. 用户解决问题或达到目标所需的条件或能力
2. 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
3. 反映（1）或（2）所描述的条件或能力的文档说明

需求工程的基本过程：

1. 需求获取：通过与用户的交流，对现有系统的观察及对任务进行分析，从而开发、捕获和修订用户的需求；

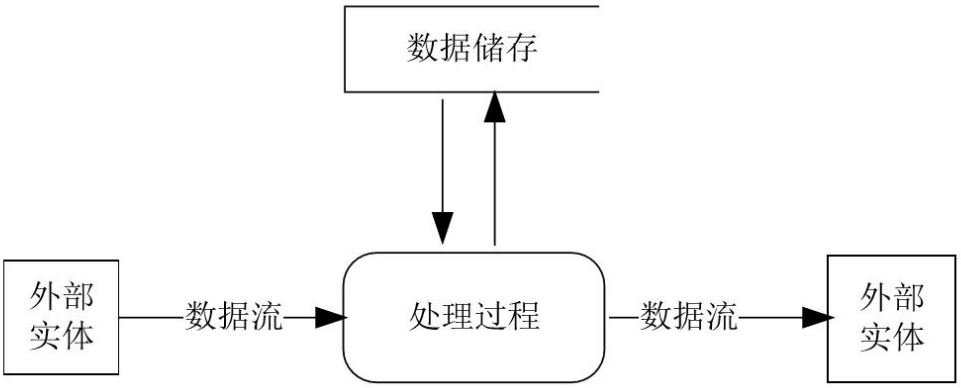
2. 需求建模：为最终用户所看到的系统建立一个概念模型，作为对需求的抽象描述，并尽可能多的捕获现实世界的语义；
3. 形成需求规格：生成需求模型构件的精确的形式化的描述，作为用户和开发者之间的一个协约；
4. 需求验证：以需求规格说明为输入，通过符号执行、模拟或快速原型等途径，分析需求规格的正确性和可行性，包含有效性检查，一致性检查，可行性检查和确认可验证性；
5. 需求管理：支持系统的需求演进，如需求变化和可跟踪性问题。

数据流图：

数据流图

■ 数据流图概念和作用

- 数据流图是过程建模的一种工具，用于分析、描述信息系统的数据转换和流动状况，显示系统内所有的基本成份及其相互联系的概况和细节。数据流图概括描述系统的内部逻辑，是理解表达用户需求、与用户沟通交流的工具，是新系统逻辑模型的最重要组成部分。



1．什么是用例图？用例图有什么作用？

定义：

由参与者（Actor）、用例（Use Case）以及它们之间的关系构成的用于描述系统功能的图成为用例图。（□ 2 分□ ）

作用：

用例图是从软件需求分析到最终实现的第一步，它显示了系统的用户和用户希望提供的功能，有利于用户和软件开发人员之间的沟通（1 分）。用例图可视化地表达了系统的需求，具有直观、规范等优点，克服了纯文字性说明的不足（□ 1 分□ ）。

用例方法是完全从外部来定义系统的，它把需求和设计完全分离开来（1 分），使用户不用关心系统内部是如何完成各种功能的。

根据下面的叙述，绘制一幅关于顾客从自动售货机中购买物品的顺序图。

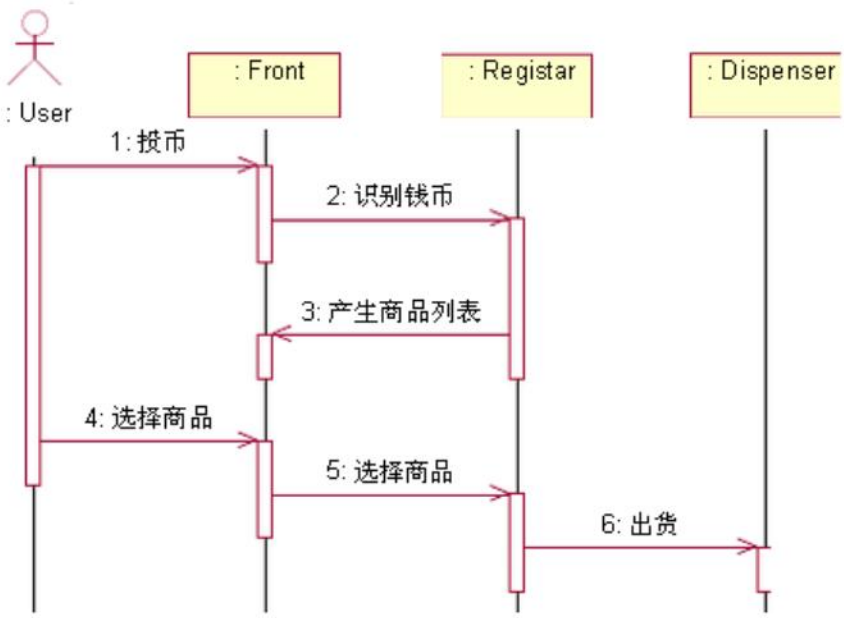
(1) 顾客（User）先向自动售货机的前端（Front）投币；

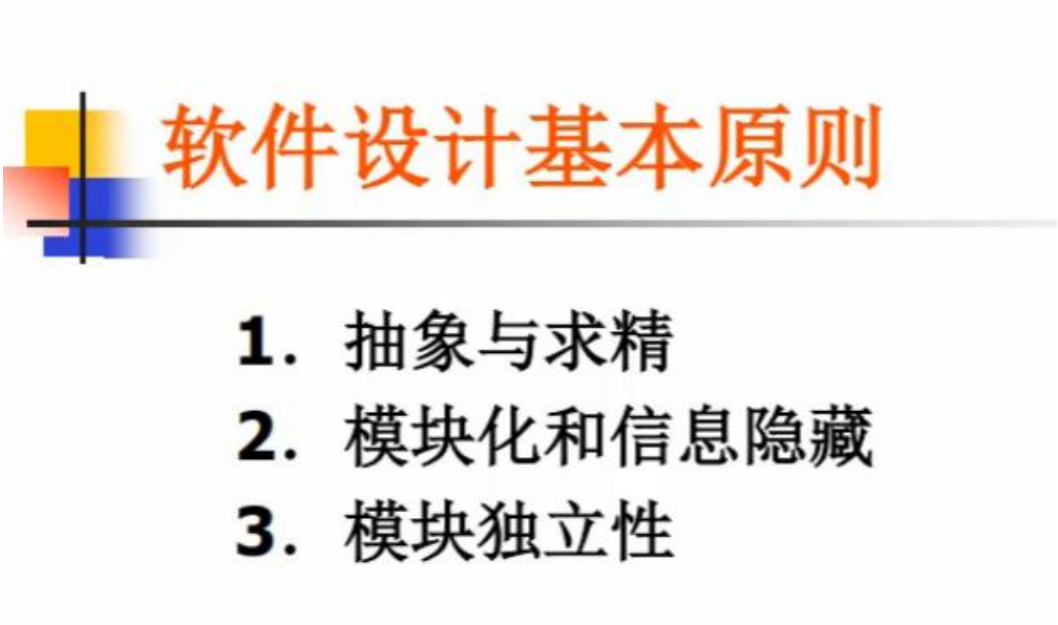
(2) 售货机的识别器（Register）识别钱币；

(3) 售货机前端（Front）根据 Register 的识别结果产生商品列表；

(4) 顾客选择商品；

(5) 识别器控制的出货器（Dispenser）将所选商品送至前端（Front）。





软件测试

测试用例（Test Case）是为某个特殊目标而编制的一组测试输入、执行条件以及预期结果，以便测试某个程序路径或核实是否满足某个特定需求。

中文名	测试用例	作　用	测试输入、执行条件以及预期结果
外文名	Test Case	类　型	测试程序

回归测试 编辑

本词条由"科普中国"科学百科词条编写与应用工作项目 审核 。

回归测试是指修改了旧代码后，重新进行测试以确认修改没有引入新的错误或导致其他代码产生错误。自动回归测试将大幅降低系统测试、维护升级等阶段的成本。

回归测试作为软件生命周期的一个组成部分，在整个软件测试过程中占有很大的工作量比重，软件开发的各个阶段都会进行多次回归测试。在渐进和快速迭代开发中，新版本的连续发布使回归测试进行的更加频繁，而在极端编程方法中，更是要求每天都进行若干次回归测试。因此，通过选择正确的回归测试策略来改进回归测试的效率和有效性是很有意义的。

软件测试覆盖率简介

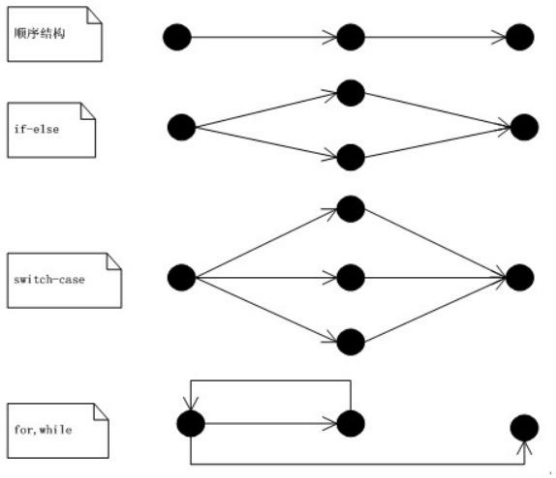
```
1、定义：覆盖率是用来度量测试完整性的一个手段，同时也是测试技术有效性的一个度量。
2、计算：覆盖率=（至少被执行一次的item数）/item的总数
3、特点
    1）通过覆盖率数据，可以检测我们的测试是否充分
    2）分析出测试的弱点在哪方面
    3）指导我们设计能够增加覆盖率的测试用例，有效提高测试质量，但是测试用例设计不能一味追求覆盖率，
```

圈复杂度 编辑

圈复杂度(Cyclomatic complexity)是一种代码复杂度的衡量标准，在1976年由Thomas J. McCabe, Sr. 提出。

在软件测试的概念里，圈复杂度用来衡量一个模块判定结构的复杂程度，数量上表现为线性无关的路径条数，即合理的预防错误所需测试的最少路径条数。圈复杂度大说明程序代码可能质量低且难于测试和维护，根据经验，程序的可能错误和高的圈复杂度有着很大关系。

圈复杂度主要与分支语句（if、else、，switch 等）的个数成正相关。可以在图1中看到常用到的几种语句的控制流图（表示程序执行流程的有向图）。当一段代码中含有较多的分支语句，其逻辑复杂程度就会增加。在计算圈复杂度时，可以通过程序控制流图方便的计算出来。通常使用的计算公式是V(G) = e – n + 2 , e 代表在控制流图中的边的数量（对应代码中顺序结构的部分），n 代表在控制流图中的节点数量，包括起点和终点（1、所有终点只计算一次，即便有多个return或者throw；2、节点对应代码中的分支语句）。



```java
public String case2(int index, String string) {
  String returnString = null;
  if (index < 0) {
    throw new IndexOutOfBoundsException("exception <0 ");
  }
  if (index == 1) {
    if (string.length() < 2) {
      return string;
    }
    returnString = "returnString1";
  } else if (index == 2) {
    if (string.length() < 5) {
      return string;
    }
    returnString = "returnString2";
  } else {
    throw new IndexOutOfBoundsException("exception >2 ");
  }
  return returnString;
}
```

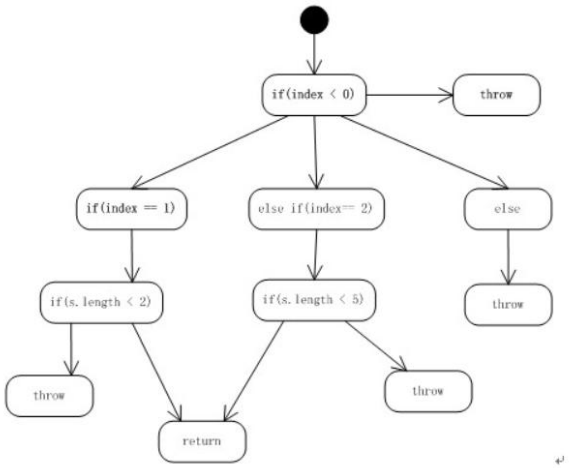


图3、case2的控制流图

根据公式 V(G) = e – n + 2 = 12 – 8 + 2 = 6 。case2的圈复杂段为6。说明一下为什么n = 8，虽然图上的真正节点有12个，但是其中有5个节点为throw、return，这样的节点为end节点，只能记做一个。

在开发中常用的检测圈复杂度的工具，PMD,checkstyle都可以检测到高复杂度的代码块。在代码的开发中，配合各种圈复杂度的检测插件，将高复杂度的代码进行适当的拆分、优化，可以大大提高代码整体的质量，减少潜在bug存在。

白盒测试基本路径测试方法

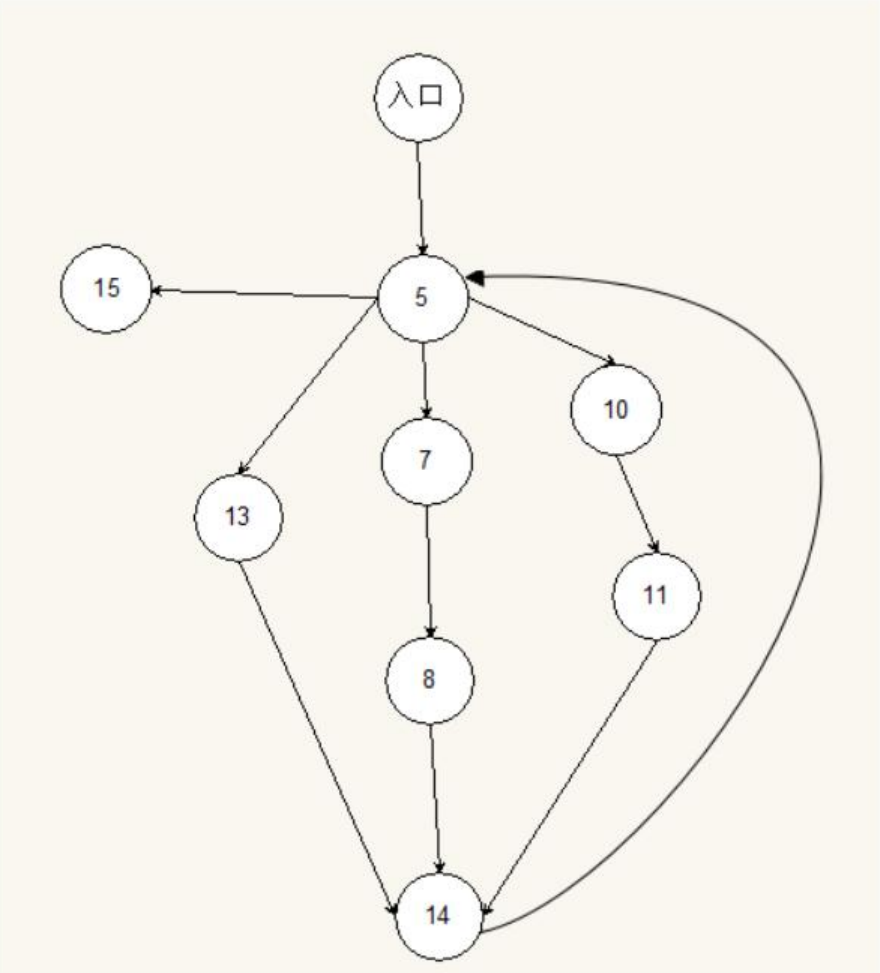
https://www.cnblogs.com/ITGirl00/p/3857446.html

白盒测试--基本路径测试法详细说明和举例

有一个函数如下：

```
1      public static  void SortNum(int numA, int numB)
2          {
3              int x = 0;
4              int y = 0;
5              while (numA--> 0)
6              {
7                  if (numB == 0)
8                      x = y +2;
9                  else
10                     if (numB == 1)
11                         x = y + 10;
12                     else
13                         x = y +20;
14             }
15         }
```

下面先画出程序控制流图



根据以上的控制流图，可以计算出以下路径：

路径1:5-7-8-14-5-15

路径2:5-10-11-14-5-15

路径3:5-13-14-5-15

路径4：5-15

导出测试用例：

下面使用**语句覆盖测试**：

	通过路径	输入数据	预期结果	测试结果
case 1	5-7-8-14-5-15	numA=2,numB=0	X=2	X=2
case 2	5-10-11-14-5-15	numA=2,numB=1	X=10	X=10
case 3	5-13-14-5-15	numA=2,numB=3	X=20	X=20
case 4	5-15	numA=0,numB=1	X=0	X=0

黑盒测试中等价类划分方法

https://www.cnblogs.com/tian-yong/p/5434058.html

Example1

> 某程序具有如下功能：输入3个正数a、b、c，分别作为三边的边长构成三角形，输出这3个数所构成的三角形类型。
> 用等价类划分方法为该程序进行测试用例设计。

划分等价类

分析思路： 步骤一、要求输入3个数，且3个数都为正数；参照规则5，划分为一个有效等价类和三个无效等价类。

- 有效等价类（1）：a>0; b>0; c>0;
- 无效等价类（2）：a<=0
- 无效等价类（3）：b<=0
- 无效等价类（4）：c<=0

步骤二、在有效等价类（1）的基础上，参照规则6，对该等价类进行细分；考察3个数能否构成三角形，参照规则5，划分为一个有效等价类和三个无效等价类。

- 有效等价类（5）：a>0; b>0; c>0; a+b>c; a+c>b; b+c>a
- 无效等价类（6）：a>0; b>0; c>0; a+b<=c
- 无效等价类（7）：a>0; b>0; c>0; b+c<=a
- 无效等价类（8）：a>0; b>0; c>0; a+c<=b

步骤三、在有效等价类（5）的基础上，参照规则6，对该等价类进行细分；考察3个数能否构成等边三角形，参照规则2，划分为一个有效等价类和一个无效等价类。

- 有效等价类（9）：a>0; b>0; c>0; a+b>c; a+c>b; b+c>a; a=b=c
- 无效等价类（10）：a>0; b>0; c>0; a+b>c; a+c>b; b+c>a; a!=b 或 b!=c 或 c!=a

步骤四、在无效等价类（10）的基础上，参照规则6，对该等价类进行细分；考察3个数能否构成等腰三角形，参照规则4，划分为三个有效等价类和一个无效等价类。

- 有效等价类（11）：a>0; b>0; c>0; a+b>c; a+c>b; b+c>a; a=b!=c
- 有效等价类（12）：a>0; b>0; c>0; a+b>c; a+c>b; b+c>a; b=c!=a
- 有效等价类（13）：a>0; b>0; c>0; a+b>c; a+c>b; b+c>a; c=a!=b
- 无效等价类（14）：a>0; b>0; c>0; a+b>c; a+c>b; b+c>a; a!=b; a!=c; b!=c

设计测试用例

序号	yyyynn	覆盖等价类	预期输出结果
–	–	覆盖有效等价类	–
1	199307	(1) (5)	日期格式有效
–	–	覆盖无效等价类	–
2	19June	(2)	日期格式无效
3	19Jun	(3)	日期格式无效
4	19June2	(4)	日期格式无效
5	198805	(6)	日期格式无效
6	205005	(7)	日期格式无效
7	198800	(8)	日期格式无效
8	199513	(9)	日期格式无效

测试和调试的区别

2015年09月07日 19:43:00 姚得芳 阅读数：893

测试的目的是找出存在的错误，而调试的目的是定位错误并修改程序以修改错误。

调试是测试之后的活动，测试和调试在目标，方法和思路上都有所不同

测试从一个已知的条件开始，使用预先定义的过程，有预知的结果；调试从一个未知的条件开始，结束的过程不可预计

测试过程可以实现设计，进度可实现确定；调试不能描述过程或者持续时间。

测试是在软件生命周期的全过程中，而调试只在软件生命周期的编码阶段中。

测试主要由测试人员进行，而编码主要由开发人员进行。

10. 软件测试的策略?

(1) (1) 在任何情况下都应使用边界值分析的方法。
(2) (2) 必要时用等价类划分法补充测试方案。
(3) (3) 必要时再用错误推测法补充测试方案。
(4) (4) 对照程序逻辑，检查已设计出的测试方案。
(5) (5) 根据对程序可靠性的要求采用不同的逻辑覆盖标准，再补充一些测试方案。