# 重点

## 耦合类型有哪些？区别是什么？

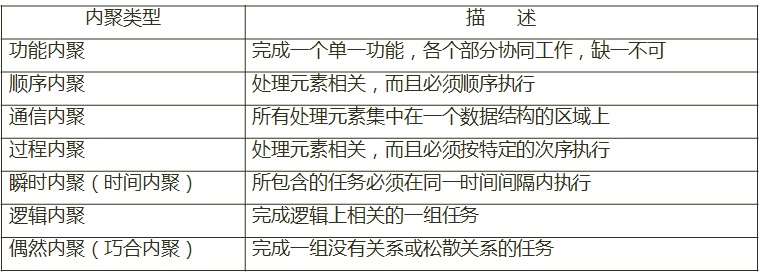
A．公共   
B.控制   
C.标记   
D.数据

公共耦合指通过一个公共数据环境相互作用的那些模块间的耦合。

耦合是指两个或两个以上的模块相互依赖于对方的一个量度。它可以分为非直接耦合、数据耦合、标记耦合、控制耦合、外部耦合、公共耦合及内容耦合等。  
    当一个模块直接修改或操作另一个模块的数据或者直接转入另一个模块时，就发生了内容耦合。

## 图　内聚类型有哪些？区别是？

如果一个模块的所有成分都操作同一数据集或生成同一数据集，则称为通信内聚。



功能内聚：完成一个单一功能，各个部分协同工作，缺一不可。

顺序内聚：处理元素相关，而且必须顺序执行。

通信内聚：所有处理元素集中在一个数据结构的区域上。

过程内聚：处理元素相关，而且必须按特定的次序执行。

瞬时内聚：所包含的任务必须在同一时间间隔内执行（如初始化模块）。

逻辑内聚：完成逻辑上相关的一组任务。

偶然内聚：完成一组没有关系或松散关系的任务。

## 系统的 维护类型有哪些？区别是？

1. 改正性  
   B.适应性  
   C.改善性  
   D.预防性

系统的可维护性的评价指标包括：可理解性、可测试性、可修改性。

可理解性：指别人能理解系统的结构、界面、功能和内部过程的难易程度。模块化、详细设计文档、结构化设计和良好的高级程序设计语言等，都有助于提高可理解性。

可测试性：诊断和测试的容易程度取决于易理解的程度。

可修改性：诊断和测试的容易程度与系统设计所制定的设计原则有直接关系。模块的耦合、内聚、作用范围与控制范围的关系等，都对可修改性有影响。

## ２基本的开发模型，各自擅长什么？

螺旋模型是一种演化软件开发过程模型，它兼顾了快速原型的迭代的特征以及瀑布模型的系统化与严格监控。螺旋模型最大的特点在于引入了其他模型不具备的风险分析，使软件在无法排除重大风险时有机会停止，以减小损失。同时，在每个迭代阶段构建原型是螺旋模型用以减小风险的途径。螺旋模型更适合大型的昂贵的系统级的软件应用。

瀑布模型严格遵循软件生命周期各阶段的固定顺序：计划、分析、设计、编程、测试和维护，上一阶段完成后才能进入到下一阶段。瀑布模型的优点是：可强迫开发人员采用规范的方法；严格规定了各阶段必须提交的文档；要求每个阶段结束后，都要进行严格的评审。但瀑布模型过于理想化，而且缺乏灵活性，无法在开发过程中逐渐明确用户难以确切表达或一时难以想到的需求。该模型比较适合于需求明确，对交付时间有严格要求的开发。  
原型模型基于这样一种客观事实：并非所有的需求在系统开发之前都能准确地说明和定义。因此，它不追求也不可能要求对需求的严格定义，而是采用了动态定义需求的方法。它适用于需求不明确的开发环境。  
螺旋模型综合了瀑布模型和演化模型的优点，还增加了风险分析。螺旋模型包含了四个方面的活动：制订计划、风险分析、实施工程、客户评估。采用螺旋模型时，软件开发沿着螺旋线自内向外旋转，每转一圈都要对风险进行识别和分析，并采取相应的对策。螺旋模型比较适合大规模的开发，它对风险控制有很高的要求。

瀑布模型具有可强迫开发人员采用规范的方法；严格规定了各阶段必须提交的文档；要求每个阶段结束后，都要进行严格的评审等优点。但瀑布模型过于理想化，而且缺乏灵活性，无法在开发过程中逐渐明确用户难以确切表达或一时难以想到的需求。  
原型模型一般是基于需求不容易明确这一事实：并非所有的需求在系统开发之前都能准确地说明和定义。因此，它不追求也不可能要求对需求的严格定义，而是采用了动态定义需求的方法。快速原型法适用于需求不够明确的项目。  
V模型一种典型的测试模型，该模型强调开发过程中测试贯穿始终。  
螺旋模型综合了瀑布模型和演化模型的优点，并在此基础上还增加了风险分析。采用螺旋模型时，软件开发沿着螺旋线自内向外旋转，每转一圈都要对风险进行识别和分析，并采取相应的对策。

## 极限编程

极限编程是一个轻量级的、灵巧的软件开发方法；同时它也是一个非常严谨和周密的方法。它的基础和价值观是交流、朴素、反馈和勇气；即，任何一个软件项目都可以从四个方面入手进行改善：加强交流；从简单做起；寻求反馈；勇于实事求是。XP是一种近螺旋式的开发方法，它将复杂的开发过程分解为一个个相对比较简单的小周期；通过积极的交流、反馈以及其它一系列的方法，开发人员和客户可以非常清楚开发进度、变化、待解决的问题和潜在的困难等，并根据实际情况及时地调整开发过程。XP就提倡结对编程（PairProgramming），而且代码所有权是归于整个开发队伍。其中的结对编程就是一种对代码的审查过程，XP主要解决代码质量低的问题，编码速度不能改变。

## 结构化设计，结构化开发方法

在结构化设计中，系统由多个逻辑上相对独立的模块组成，在模块划分时需要遵循如下原则：  
（1）模块的大小要适中。系统分解时需要考虑模块的规模，过大的模块可能导致系统分解不充分，其内部可能包括不同类型的功能，需要进一步划分，尽量使得各个模块的功能单一；过小的模块将导致系统的复杂度增加，模块之间的调用过于频繁，反而降低了模块的独立性。一般来说，一个模块的大小使其实现代码在1～2页纸之内，或者其实现代码行数在50～200行之间，这种规模的模块易于实现和维护。  
（2）模块的扇入和扇出要合理。一个模块的扇出是指该模块直接调用的下级模块的个数；扇出大表示模块的复杂度高，需要控制和协调过多的下级模块。扇出过大一般是因为缺乏中间层次，应该适当增加中间层次的控制模块；扇出太小时可以把下级模块进一步分解成若干个子功能模块，或者合并到它的上级模块中去。一个模块的扇入是指直接调用该模块的上级模块的个数；扇入大表示模块的复用程度高。设计良好的软件结构通常顶层扇出比较大，中间扇出较少，底层模块则有大扇入。一般来说，系统的平均扇入和扇出系数为3或4，不应该超过7，否则会增大出错的概率。  
（3）深度和宽度适当。深度表示软件结构中模块的层数，如果层数过多，则应考虑是否有些模块设计过于简单，看能否适当合并。宽度是软件结构中同一个层次上的模块总数的最大值，一般说来，宽度越大系统越复杂，对宽度影响最大的因素是模块的扇出。在系统设计时，需要权衡系统的深度和宽度，尽量降低系统的复杂性，减少实施过程的难度，提高开发和维护的效率。

## 测试，有几种类型，区别在于？注意什么，怎么计算测试用例，什么测试用例可以用

测试的目标是需求分析开始。从需求开始时就已经确定了产品的功能，在那些的用户场景、用户使用的功能。

选择测试

环路测试

软件测试的目的是为了发现尽可能多的缺陷。

### 测试策略

### A．自底向上  B.自顶向下  C.三明治  D.一次性  1、自顶向下集成

优点：较早地验证了主要控制和判断点；按深度优先可以首先实现和验证一个完整的软件功能；功能较早证实，带来信心；只需一个驱动，减少驱动器开发的费用；支持故障隔离。

缺点：柱的开发量大；底层验证被推迟；底层组件测试不充分。

适应于产品控制结构比较清晰和稳定；高层接口变化较小；底层接口未定义或经常可能被修改；产口控制组件具有较大的技术风险，需要尽早被验证；希望尽早能看到产品的系统功能行为。

### 2、自底向上集成

优点：对底层组件行为较早验证；工作最初可以并行集成，比自顶向下效率高；减少了桩的工作量；支持故障隔离。

缺点：驱动的开发工作量大；对高层的验证被推迟，设计上的错误不能被及时发现。

适应于底层接口比较稳定；高层接口变化比较频繁；底层组件较早被完成。

## 敏捷开发，有哪几种开发方法，各用于哪些方面

### 12个最佳实践分别是

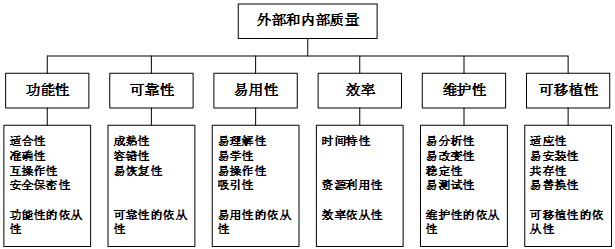
计划游戏，小型发布，隐喻，简单设计，测试先行，重构，结对编程，集体代码所有制，持续集成，每周工作40小时，现场客户及编码标准。其中系统的设计要能够尽可能早交付属于小型发布。  
小型发布可以使得集成更频繁，客户获得的中间结果越频繁，反馈也就越频繁，客户就能够实时地了解项目的进展情况，从而提出更多的意见，以便在下一次迭代中计划进去，以实现更高的客户满意度。

## 图　软件质量特性，有哪些，各自

可维护性

可移植性．．．

可移植性包含：适应性、易安装性、共存性和易替换性四个特性。



## McCabe度量法计算该程序的环路复杂性

怎么计算

## 结构化分析　数据流图

数据流图用来记录系统中的数据和数据在特定的过程中的流动，即数据如何被采集、处理、保存和使用的（围绕信息系统的功能）。外部实体指系统之外又与系统有联系的人或事物。它表达了该系统数据的外部来源和去处。

## 需求分析　软件开发过程

## 开发方法

1. 结构化   
   B.Jackson   
   C.原型化   
   D.面向对象

## **软件工程的基本概念。** 软件设计的任务是基于需求分析的结果建立各种设计模型，给出问题的解决方案。从工程管理的角度，可以将软件设计分为两个阶段：概要设计阶段和详细设计阶段。结构化设计方法中，概要设计阶段进行软件体系结构的设计、数据设计和接口设计；详细设计阶段进行数据结构和算法的设计。面对对象设计方法中，概要设计阶段进行体系结构设计、初步的类设计/数据设计、结构设计；详细设计阶段进行构件设计。 结构化设计和面向对象设计是两种不通过的设计方法，结构化设计根据系统的数据流图进行设计，模块体现为函数、过程及子程序；面向对象设计基于面向对象的基本概念进行，模块体现为类、对象和构件等。

## 敏捷过程的方法

水晶方法，Crystal ，是由 Alistair Cockburn 和 Jim Highsmith 建立的敏捷方法系列，其目的是发展一种提倡“机动性的”[1] 方法，包含具有共性的核心元素，每个都含有独特的角色、过程模式、工作产品和实践。Crystal 家族实际上是一组经过证明、对不同类型项目非常有效的敏捷过程，它的发明使得敏捷团队可以根据其项目和环境选择最合适的 Crystal 家族成员。

极限编程（Extreme Programming，XP）是一门针对业务和软件开发的规则，它的作用在于将两者的力量集中在共同的、可以达到的目标上。它是以符合客户需要的软件为目标而产生的一种方法论，XP使开发者能够更有效的响应客户的需求变化，哪怕是在软件生命周期的后期。它强调，软件开发是人与人合作进行的过程，因此成功的软件开发过程应该充分利用人的优势，而弱化人的缺点，突出了人在软件开发过程中的作用。极端编程属于轻量级的方法，认为文档、架构不如直接编程来的直接。

敏捷开发是一种以人为核心、迭代、循序渐进的开发方法，常见的敏捷开发方法有极限编程法、水晶法、并列争球法和自适应软件开发方法。  
极限编程是一种轻量级的开发方法，它提出了四大价值观：沟通、简单、反馈、勇气。五大原则：快速反馈、简单性假设、逐步修改、提倡更改、优质工作。  
水晶法强调经常交付，认为每一种不同的项目都需要一套不同的策略、约定和方法论。  
并列争球法的核心是迭代、增量交付，按照30天进行迭代开发交付可实际运行的软件。  
自适应软件开发的核心是三个非线性的，重迭的开发阶段：猜测、合作、学习。

## 结构化开发方法

结构化开发方法是一种面向数据流的开发方法，其基本思想是软件功能的分解和抽象。结构化开发方法又称生命周期法，是迄今为止最传统、应用最广泛的一种信息系统开发方法。结构化开发方法采用系统工程的思想和工程化的方法，按用户至上的原则，结构化、模块化、自顶向下地对信息系统进行分析与设计。该方法严格按照信息系统开发的阶段性开展设计工作，每个阶段都产生一定的设计成果，通过评估后再进入下一阶段开发工作。因此，结构化开发方法具有以下优点：

• 开发工作的顺序性、阶段性适合初学者参与软件的开发

• 开发工作的阶段性评估可以减少开发工作重复性和提高开发的成功率

• 该方法有利于提高系统开发的正确性、可靠性和可维护性

• 具有完整的开发质量保证措施

结构化开发方法存在的不足主要是开发周期太长，个性化开发阶段的文档编写工作量过大或过于烦琐,无法发挥开发人员的个性化开发 能力。一般来说，结构化开发方法主要适用于组织规模较大、组织结构相对稳定的企业，这些大型企业往往业务处理过程规范、信息系统数据需求非常明确，在一定时期内需求变化不大。

### 数据流

面向数据流的设计是以需求分析阶段产生的数据流图为基础，按一定的步骤映射成软件结构，因此又称结构化设计。该方法由美国IBM公司L.Constantine和E.Yourdon等人于1974年提出，与结构化分析(SA)衔接，构成了完整的结构化分析与设计技术，是目前使用最广泛的软件设计方法之一。

各种软件系统，不论DFD如何庞大和复杂，一般可分为变换型和事务型，一个软件系统既可以只有一种数据流类型，也可以是两种数据流烈性。在结构化设计中，可以将数据流映射为软件系统的模块结构，不同类型的数据流有不同的映射方法。

## 增量模型

增量模型融合了瀑布模型的基本成分（重复应用）和原型实现的迭代特征，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”。当使用增量模型时，第1个增量往往是核心的产品，即第1个增量实现了基本的需求，但很多补充的特征还没有发布。客户对每一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。由于能够在较短的时间内向用户提交一些有用的工作产品，因此能够解决用户的一些急用功能。由于每次只提交用户部分功能，用户有较充分的时间学习和适应新的产品。

对系统的可维护性是一个极大的提高，因为整个系统是由一个个构件集成在一起的，当需求变更时只变更部分部件，而不必影响整个系统。

增量模型存在以下缺陷：

1） 由于各个构件是逐渐并入已有的软件体系结构中的，所以加入构件必须不破坏已构造好的系统部分，这需要软件具备开放式的体系结构。

2） 在开发过程中，需求的变化是不可避免的。增量模型的灵活性可以使其适应这种变化的能力大大优于瀑布模型和快速原型模型，但也很容易退化为边做边改模型，从而使软件过程的控制失去整体性。

3）如果增量包之间存在相交的情况且未很好处理，则必须做全盘系统分析，这种模型将功能细化后分别开发的方法较适应于需求经常改变的软件开发过程。

## 统一过程UP

什么是 Rational 统一过程（ Rational Unified Process）?  
Rational Unified Process 是软件工程的过程。它提供了在开发组织中分派任务和责任的纪律化方法。它的目标是在可预见的日程和预算前提下，确保满足最终用户需求的高质量产品。  
统一过程模型是一种“用例驱动，以体系结构为核心，迭代及增量”的软件过程框架，由UML方法和工具支持。

RUP把一个项目分为四个不同的阶段：  
构思阶段 ：包括用户沟通和计划活动两个方面，强调定义和细化用例，并将其作为主要模型。  
细化阶段 ：包括用户沟通和建模活动，重点是创建分析和设计模型，强调类的定义和体系结构的表示。  
构建阶段 ：将设计转化为实现，并进行集成和测试。  
移交阶段 ：将产品发布给用户进行测试评价，并收集用户的意见，之后再次进行迭代修改产品使之完善。

## 软件文档的作用

在软件生产过程中，总是产生和使用大量的信息。软件文档在产品的开发过程中起着重要的作用。 提高软件开发过程的能见度。把开发过程中发生的事件以某种可阅读的形式记录在文档中。   
管理人员可把这些记载下来的材料作为检查软件开发进度和开发质量的依据，实现对软件开发的工程管理。 ? 提高开发效率。软件文档的编制，使得开发人员对各个阶段的工作都进行周密思考、全盘权衡、减少返工。并且可在开发早期发现错误和不一致性，便于及时加以纠正。 ? 作为开发人员在一定阶段的工作成果和结束标志。 ? 记录开发过程中有关信息，便于协调以后的软件开发、使用和维护。   
提供对软件的运行、维护和培训的有关信息，便于管理人员、开发人员、操作人员、用户之间的协作、交流和了解。使软件开发活动更科学、更有成效。   
便于潜在用户了解软件的功能、性能等各项指标，为他们选购符合自己需要的软件提供依据。   
从某种意义上来说，文档是软件开发规范的体现和指南。 ? 按规范要求生成一整套文档的过程，就是按照软件开发规范完成一个软件开发的过程。   
 所以，在使用工程化的原理和方法来指导软件的开发和维护时，应当充分注意软件文档的编制和管理。

## 几种维护，区别

在系统运行过程中，软件需要维护的原因是多样的，根据维护的原因不同，可以将软件维护分为以下四种：  
（1）改正性维护。为了识别和纠正软件错误、改正软件性能上的缺陷、排除实施中的误使用，应当进行的诊断和改正错误的过程就称为改正性维护。  
（2）适应性维护。在使用过程中，外部环境（新的硬、软件配置）、数据环境（数据库、数据格式、数据输入/输出方式、数据存储介质）可能发生变化。为使软件适应这种变化，而去修改软件的过程就称为适应性维护。  
（3）完善性维护。在软件的使用过程中，用户往往会对软件提出新的功能与性能要求。为了满足这些要求，需要修改或再开发软件，以扩充软件功能、增强软件性能、改进加工效率、提高软件的可维护性。这种情况下进行的维护活动称为完善性维护。  
（4）预防性维护。这是指预先提高软件的可维护性、可靠性等，为以后进一步改进软件打下良好基础。通常，预防性维护可定义为“把今天的方法学用于昨天的系统以满足明天的需要”。也就是说，采用先进的软件工程方法对需要维护的软件或软件中的某一部分（重新）进行设计、编码和测试。

## 测试 白盒测试 覆盖

语句覆盖（StatementCoverage） 又称行覆盖（LineCoverage），段覆盖（SegmentCoverage），基本块覆盖（BasicBlockCoverage），这是最常用也是最常见的一种覆盖方式，就是度量被测代码中每个可执行语句是否被执行到了，只管覆盖代码中的执行语句，却不考虑各种分支的组合等等。  
路径覆盖的含义是，选取足够多的测试数据，使程序的每条可能路径都至少执行一次（如果程序图中有环，则要求每个环至少经过一次）。

白盒测试常用的技术是逻辑覆盖，即考查用测试数据运行被测程序时对程序逻辑的覆盖程度。主要的覆盖标准有六种：语句覆盖、判定覆盖、条件覆盖、判定/条件覆盖、组合条件覆盖和路径覆盖。  
（1）语句覆盖  
语句覆盖是指选择足够多的测试用例，使得运行这些测试用例时，被测程序的每个语句至少执行一次。很显然，语句覆盖是一种很弱的覆盖标准。  
（2）判定覆盖  
判定覆盖又称分支覆盖，它的含义是，不仅每个语句至少执行一次，而且每个判定的每种可能的结果（分支）都至少执行一次。判定覆盖比语句覆盖强，但对程序逻辑的覆盖程度仍然不高。  
（3）条件覆盖  
条件覆盖的含义是，不仅每个语句至少执行一次，而且使判定表达式中的每个条件都取到各种可能的结果。条件覆盖不一定包含判定覆盖，判定覆盖也不一定包含条件覆盖。  
（4）判定/条件覆盖  
同时满足判定覆盖和条件覆盖的逻辑覆盖称为判定/条件覆盖。它的含义是，选取足够的测试用例，使得判定表达式中每个条件的所有可能结果至少出现一次，而且每个判定本身的所有可能结果也至少出现一次。  
（5）条件组合覆盖  
条件组合覆盖的含义是，选取足够的测试用例，使得每个判定表达式中条件结果的所有可能组合至少出现一次。显然，满足条件组合覆盖的测试用例，也一定满足判定/条件覆盖。因此，条件组合覆盖是上述五种覆盖标准中最强的一种。然而，条件组合覆盖还不能保证程序中所有可能的路径都至少经过一次。  
（6）路径覆盖  
路径覆盖的含义是，选取足够的测试用例，使得程序的每条可能执行到的路径都至少经过一次（如果程序中有环路，则要求每条环路径至少经过一次）。路径覆盖实际上考虑了程序中各种判定结果的所有可能组合，因此是一种较强的覆盖标准。

## 基本加工的说明有三种描述方式

对基本加工的说明有三种描述方式：结构化语言、判断表（决策表）、判断树（决策树）。基本加工逻辑描述的基本原则为：

1、对数据流图的每一个基本加工，必须有一个基本加工逻辑说明。

2、基本加工逻辑说明必须描述基本加工如何把输入数据流变换为输出数据流的加工规则。

3、加工逻辑说明必须描述实现加工的策略而不是实现加工的细节。

4、加工逻辑说明中包含的信息应是充足的，完备的，有用的，无冗余的。

## 划分模块

一个模块的作用范围（或称影响范围）指受该模块内一个判定影响的所有模块的集合。

一个模块的控制范围指模块本身以及其所有下属模块（直接或间接从属于它的模块）的集合。

一个模块的作用范围应在其控制范围之内，且判定所在的模块应在其影响的模块在层次上尽量靠近。如果再设计过程中，发现模块作用范围不在其控制范围之内，可以用“上移判点”或“下移受判断影响的模块，将它下移到判断所在模块的控制范围内”的方法加以改进。

模块的作用范围是指受该模块内部一个判定影响的所以模块集合，只有某一模块内含有依赖于该判定的操作，那么该模块就在该判定的作用范围内。  
模块的控制范围包括该模块本身以及该模块的所有下属模块。控制范围完全取决于系统的结构，与模板本身的功能并没有多大关系。  
在系统设计中，对于模块的作用范围和控制范围有两条规则：  
（1）对于任何一个判断，其作用范围应该是这个判断所在模块的控制范围的一个子集。换言之，所有受判断影响的模块应该从属于做出判断的那个模块。  
（2）受模块M判定影响的模块，最好局限于模块M本身或其直接下属模块。  
因此，一个模块的作用范围应该在其控制范围之内。

## SEI能力成熟度模型（SEICMM）把软件开发企业分为5个成熟度级别

CMM的分级结构及其过程描述：

（1）初始级：软件过程的特点是无秩序或说无定规的，有时甚至是混乱的。软件过程定义几乎处于无章法、无步骤可循的状态，软件产品所取得的成功往往依赖于极个别人的努力和机遇。

（2）可重复级：已建立了基本的项目管理过程，可用于对成本、进度和功能特性进行跟踪。对类似的应用项目，有章可循并能重复以往所取得的成功。

（3）已定义级：用于管理的和工程的软件过程均已文档化、标准化，并形成了整个软件组织的标准软件过程。全部项目均采用与实际情况相吻合的、适当修改后的标准软件过程来进行操作。

（4）已管理级：软件过程和产品质量有详细的度量标准。软件过程和产品质量得到了定量的认识和控制。

（5）优化级：通过对来自过程、新概念和新技术等方面的各种有用信息的定量分析，能够不断地、持续地对促进过程进行改进。

除第一级外，每一级都设定了一组目标，如果达到了这组目标，则表明达到了这个成熟级别，自然可以向上一更为成熟的高一级别迈进。CMM体系不主张跨级别的进化，因为从第二级开始，每一个低级别的实现均是更高级别实现的基础。

## 单元测试

在单元测试开始时，应该对通过所有被测模块的数据流进行测试。

①输入参数是否与形式参数是否一致。

②调用模块的实参与被调用模块形参是否一致。

③全局变量在各模块中的定义和用法是否一致

④外部输入、输出等等。

## 数据流图 加工 设计原则 典型例题 概念 试题76(2013年上半年试题29)

## 数据流图的设计原则： ①  数据守恒原则 ②  守恒加工原则 ③  对于每个加工，必须既有输入数据流，又有输出数据流 ④  外部实体与外部实体之间不存在数据流 ⑤  外部实体与外部存储之间不存在数据流 ⑥  数据存储与数据存储之间不存在数据流 ⑦  父图与子图的平衡原则 ⑧  数据流与加工有关，且必须经过加工

对于数据流图，必须要保持父图与子图平衡，且每个加工必须要有输入数据流和输出数据流，而每条数据流都至少有一端是加工，而且还要能保持数据守恒（即相关数据流经过加工后得到相应的输出数据流）。

## **代码行数度量法以程序的总代码行数作为程序复杂性的度量值。** **软件设计质量评审**

1. 评价软件的规格说明是否合乎用户的要求，即[总体设计](http://baike.baidu.com/view/228947.htm" \t "http://www.educity.cn/jiaocheng/_blank)思想和设计方针是否正确，需求规格说明是否得到了用户或单位上级机关的批准；需求规格说明与软件的[概要设计](http://baike.baidu.com/view/551728.htm" \t "http://www.educity.cn/jiaocheng/_blank)规格说明是否一致等。  
   (2)评审可靠性，即是否能避免输入异常(错误或超载等)、硬件失效及[软件失效](http://baike.baidu.com/view/3740235.htm" \t "http://www.educity.cn/jiaocheng/_blank)所产生的失效，一旦发生应能及时采取代替手段或恢复手段。  
   (3)评审保密措施实现情况，即是否提供对使用系统资格、对特定数据的使用资格及特殊功能的使用资格进行检查，在查出有违反使用资格情况后，能否向系统管理人员报告有关信息，是否提供对系统内重要数据加密的功能。  
   (4)评审操作特性实施情况，即操作命令和操作信息的恰当性，输入数据与输入[控制语句](http://baike.baidu.com/view/1359886.htm" \t "http://www.educity.cn/jiaocheng/_blank)的恰当性，输出数据的恰当性，应答时间的恰当性等。  
   (5)评审性能实现情况。  
   (6)评审软件是否具有可修改性、可扩充性、可互换性和可移植性。  
   (7)评审软件是否具有[可测试性](http://baike.baidu.com/view/2194552.htm" \t "http://www.educity.cn/jiaocheng/_blank)。  
   (8)评审软件是否具有复用性。

## 软件维护

①改正性维护改正性维护是指改正在系统开发阶段已发生而系统测试阶段尚未发现的错误。  
②适应性维护适应性维护是指使用软件适应信息技术变化和管理需求变化而进行的修改。  
③完善性维护完善性维护是为扩充功能和改善性能而进行的修改，主要是指对已有的软件系统增加一些在系统分析和设计阶段中没有规定的功能与性能特征。  
④预防性维护预防性维护为了改进应用软件的可靠性和可维护性，为了适应未来的软硬件环境的变化，应主动增加预防性的新的功能，以使应用系统适应各类变化而不被淘汰。

## 可维护性质量特性

可维护性质量特性是指与软件维护的难易程度相关的一组软件属性，它包含了易分析性、稳定性、易测试性和易改变性4个子特性。其中：  
易分析性是描述诊断缺陷或失效原因、判定待修改程度的难易程度的特性。  
稳定性是描述修改造成难以预料的后果的风险程度，风险程度越低，稳定性越好。  
易测试性是描述测试已修改软件的难易程度的特性。  
易改变性是描述修改、排错或适应环境变化的难易程度。

## **软件开发 各个阶段 概要设计 详细设计**

需求分析阶段的任务主要是要解决系统做什么的问题，即弄清楚问题的要求，包括需要输入什么数据，要得到什么结果，最后应输出什么。  
概要设计的主要任务是把需求分析得到的结果转换为软件结构和数据结构，即将一个复杂系统按功能进行模块划分、建立模块的层次结构及调用关系、确定模块间的接口及人机界面、确定数据的结构特性、以及数据库的设计等。  
详细设计是在概要设计的基础上更细致的设计，它包括具体的业务对象设计、功能逻辑设计、界面设计等工作。详细设计是系统实现的依据，需要更多地考虑设计细节。  
编码即编写程序代码，具体实现系统。

## 软件复杂性

软件复杂性主要表现在程序的复杂性。程序的复杂性主要指模块内程序的复杂性。它直接关联到软件开发费用的多少、开发周期长短和软件内部潜伏错误的多少。同时它也是软件可理解性的另一种度量。  
软件复杂性度量的参数很多，主要有：  
    (1) 规模，即总共的指令数，或源程序行数。  
    (2) 难度，通常由程序中出现的操作数的数目所决定的量来表示。  
    (3) 结构，通常用于程序结构有关的度量来表示。  
    (4) 智能度，即算法的难易程度。

## 软件测试

   软件测试的目的就是在软件投入生产性运行之前，尽可能多地发现软件产品（主要是指程序）中的错误和缺陷。软件测试是软件质量保证的主要手段之一。  
    一个好的测试用例是极有可能发现至今为止尚未发现的错误的测试用例。一次成功的测试是发现了至今为止尚未发现的错误的测试。一个高效的测试是指用少量的测试用例，发现被测软件尽可能多的错误。软件测试不能说明软件中不存在错误。

## 黑盒测试

黑盒：顾名思义　应该不能知道内部情况

黑盒测试又称功能测试。它把软件看做一个不透明的黑盒子，完全不考虑（或不了解）软件的内部结构和处理算法，它只检查软件功能是否能按照软件需求说明书的要求正常使用，软件是否能适当地接收输入数据并产生正确的输出信息，软件运行过程中能否保持外部信息（例如文件和数据库）的完整性等。  
常用的黑盒测试技术包括等价类划分、边界值分析、错误推测和因果图等。