10 系统开发和运行知识

软件生存周期

软件生存周期包括可行性分析与项目开发计划、需求分析、概要设计、详细设计、编码和单元测试、综合测试及维护阶段。

1、可行性分析与项目开发计划

主要任务是确定软件的开发目标及可行性。该阶段应该给出问题定义、可行性分析和项目开发计划。

2、需求分析

需求分析阶段的任务不是具体地解决问题，而是准确地确定软件系统必须做什么，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。

3、概要设计

在本阶段，开发人员需要将确定的功能需求转换成相应的体系结构。概要设计就是设计软件的结构，明确软件有哪些模块组成，模块的层次以及功能。与此同时，还要设计应用系统的总体数据结构和数据库结构。

4、详细设计

主要任务就是对每个模块完成的功能进行描述，不是编写程序，而是设计出程序的详细规格说明，使程序员可以根据它们写出实际的程序代码。

软件生存周期

5、编码和单元测试

就是把每个模块的结构转换成计算机可接受的程序代码，即写成某种特定程序设计语言表示的源程序清单，并仔细测试编写出每一个模块。

6、综合测试

综合测试阶段的关键任务是通过各种类型的测试（及相应的调试）使软件达到预定的要求。最基本的测试是集成测试和验收测试。

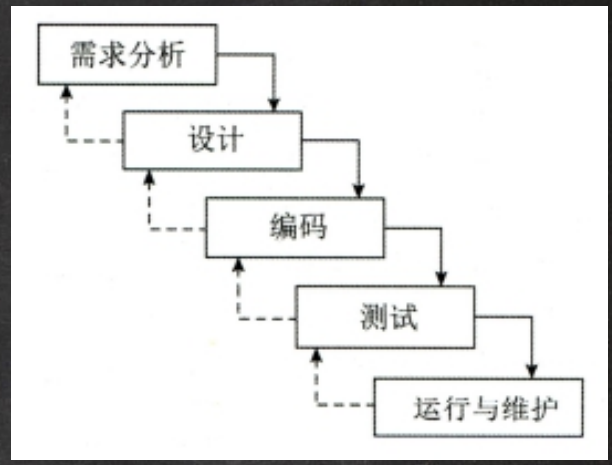
7、维护

维护阶段是软件生存期中时间最长的阶段。软件一旦交付正式投入运行后便进入软件维护阶段。该阶段的关键任务是通过各种必要的维护活动使系统持久地满足用户的需要。每一项维护活动都应该准确地记录下来，作为正式的文档资料加以保存。

软件生存周期模型

1、瀑布模型

瀑布模型是将软件生存周期各个活动规定为依线性顺序连接的若干阶段的模型。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。瀑布模型假设，一个待开发的系统需求是完整的、简明的、一致的，而且可以先于设计和实现完成之前产生。



优点：容易理解，管理成本低；强调开发的阶段性早期计划及需求调查和产品测试。

不足：客户必须能够完整、正确和清晰地表达他们的需要；在开始的两个或三个阶段中，很难评估真正的进度状态；当接近项目结束时，出现了大量的集成和测试工作；直到项目结束之前，都不能演示系统的能力。

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，开发费用超预算。

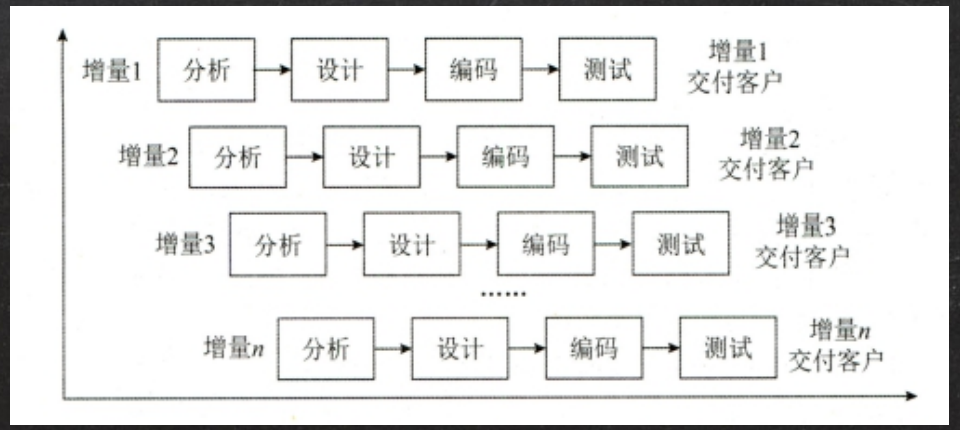
软件生存周期模型

2、增量模型

增量模型融合了瀑布模型的基本成分和原型实现的迭代特征，它假设可以将需求分段为一系列增量产品，每一增量可以分别地开发。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”。增量模型强调每一个增量均发布一个可操作的产品。

优点：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；运行增量投资，即在项目开始时，可以仅对一个或两个增量投资。

不足：如果没有对用户的变更要求进行规划，那么产生的初始增量可能会造成后来增量的不稳定；如果需求不像早期思考的那样稳定和完整，那么一些增量就可能需要重新开发，重新发布；管理发生的成本、进度和配置的复杂性，可能会超出组织的能力。



软件生存周期模型

3、演化模型

主要针对事先不能完整定义需求的软件开发，是在快速开发一个原型的基础上，根据用户在使用原型的过程中提出的意见和建议对原型进行改进，获得原型的新版本。重复这一过程，最终可得到令用户满意的软件产品。

优点：任何功能一经开发就能进入测试，以便验证是否符合产品的需求，可以帮助引导出高质量的产品要求。

缺点：如果不加控制地让用户接触开发中尚未稳定的功能，可能对开发人员及用户都会产生负面影响。

软件生存周期模型

4、螺旋模型

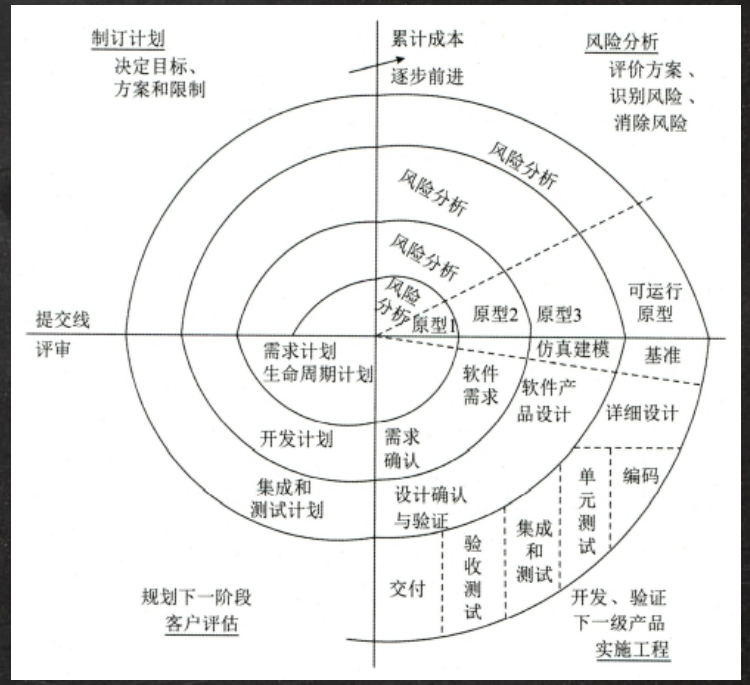
将瀑布模型和演化模型结合起来，加入了两种模型均忽略的风险分析，弥补了两种模型的不足。螺旋模型将开发过程分为几个螺旋周期，每个螺旋周期大致和瀑布模型相符合。

（1）制订计划。确定软件的目标，选定实施方案，明确项目开发的限制条件。

（2）风险分析。分析所选的方案，识别风险，消除风险。

（3）实施工程。实施软件开发，验证阶段产品。

（4）用户评估。评价开发工作，提出修正建议，建立下一个周期的开发计划。



螺旋模型强调风险分析，使得开发人员和用户对每个演化层出现的风险有所了解，继而做出应有的反应。因此特别适用于庞大、复杂并且具有高风险的系统。

软件生存周期模型

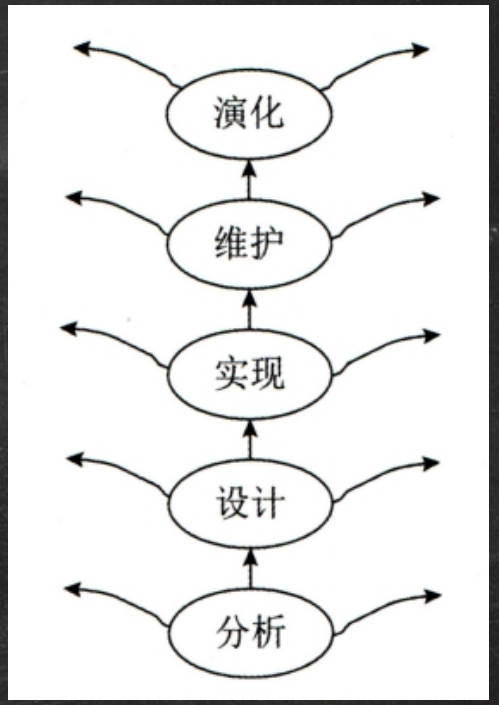
5、喷泉模型

喷泉模型是一种以用户需求为动力，以对象作为驱动的模型，适合于面向对象的开发方法。它克服了瀑布模型不支持软件重用和多项开发活动集成的局限性。喷泉模型使开发过程具有迭代性和无间隙性。

该模型的各个阶段没有明显的界限，开发人员可以同步进行。

优点：可以提高软件项目的开发效率，节省开发时间。

不足：由于喷泉模型在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，不利于项目的管理。此外这种模型要求严格管理文档，使得审核的难度加大。



典型的软件开发方法

1、结构化开发方法

是一种传统的信息系统开发方法，由结构化分析、结构化设计和结构化程序设计构成，它是一种面向数据流的开发方法。其精髓是自顶向下、逐层分解和模块化设计，它的基本原则是功能的分解与抽象。

结构化方法的开发过程一般是先把系统功能视为一个大的模块，再根据系统分析与设计的要求对其进行进一步的模块分解或组合。

结构化方法特别适合于数据处理领域的问题，但是不适合解决规模较大的、比较复杂的项目，且难以适应需求的变化。

2、原型化开发方法

根据用户初步需求，利用系统开发工具，快速地建立一个系统模型展示给用户，在此基础上与用户交流，最终实现用户需求的信息系统快速开发的方法。

原型法的优点主要在于能更有效地确认用户需求。

原型化方法比较适合于用户需求不清、业务理论不确定、需求经常变化的情况。当系统规模不是很大也不太复杂时，采用该方法是比较好的。

典型的软件开发方法

3、面向对象开发方法

面向对象开发方法的基本出发点是尽可能按照人类认识世界的方法和思维方法来分析和解决问题。客观世界是由许多具体的事物、事件、概念和规则组成的，这些均可被看成对象，面向对象方法正是以对象作为最基本的元素，它也是分析问题、解决问题的核心。

使用面向对象的方法构造的系统具有更好的复用性。面向对象的方法使系统的描述及信息模型的表示与客观实体相对应，符合人们的思维习惯，因此可以缩短开发周期。

4、敏捷方法

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，敏捷方法可以使用户能够在开发周期的后期增加或改变需求。

软件项目管理

1、成本估算

2、风险分析

3、进度管理

进度安排的常用图形描述方法有Gantt图（甘特图）和项目计划评审技术（PERT）图。

（1）Gantt图

Gantt图能清晰地描述每个任务从何时开始，到何时结束，任务的进展情况以及各个任务之间的并行性。但是其缺点是不能清晰地反映出各任务之间的依赖关系，难以确定整个项目的关键所在，也不能反映计划中有潜力的部分。

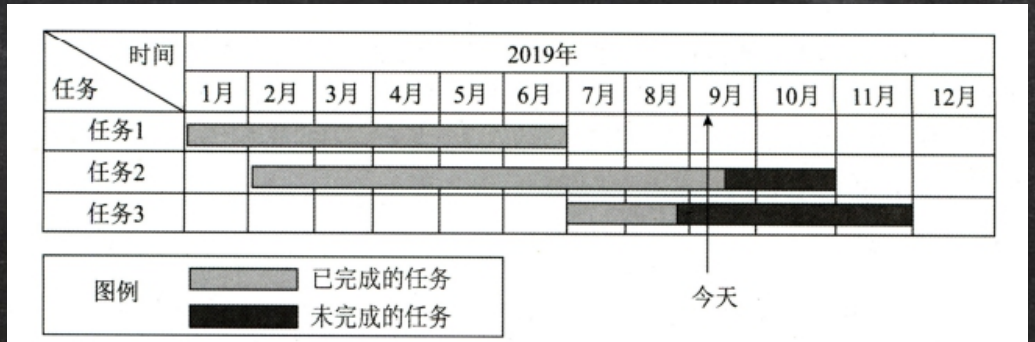


图 10-6 Gantt 图实例

软件项目管理

（2）PERT图

PERT图不仅给出了每个任务的开始时间、结束时间和完成该任务所需的时间，还给出了任务之间的关系，即哪些任务完成后才能开始另外一些任务，以及如期完成整个工程的关键路径。图中的松弛时间则反映了完成某些任务时可以推迟其开始时间或延长其完成所需的时间。但是，PERT图不能反映任务之间的并行关系。

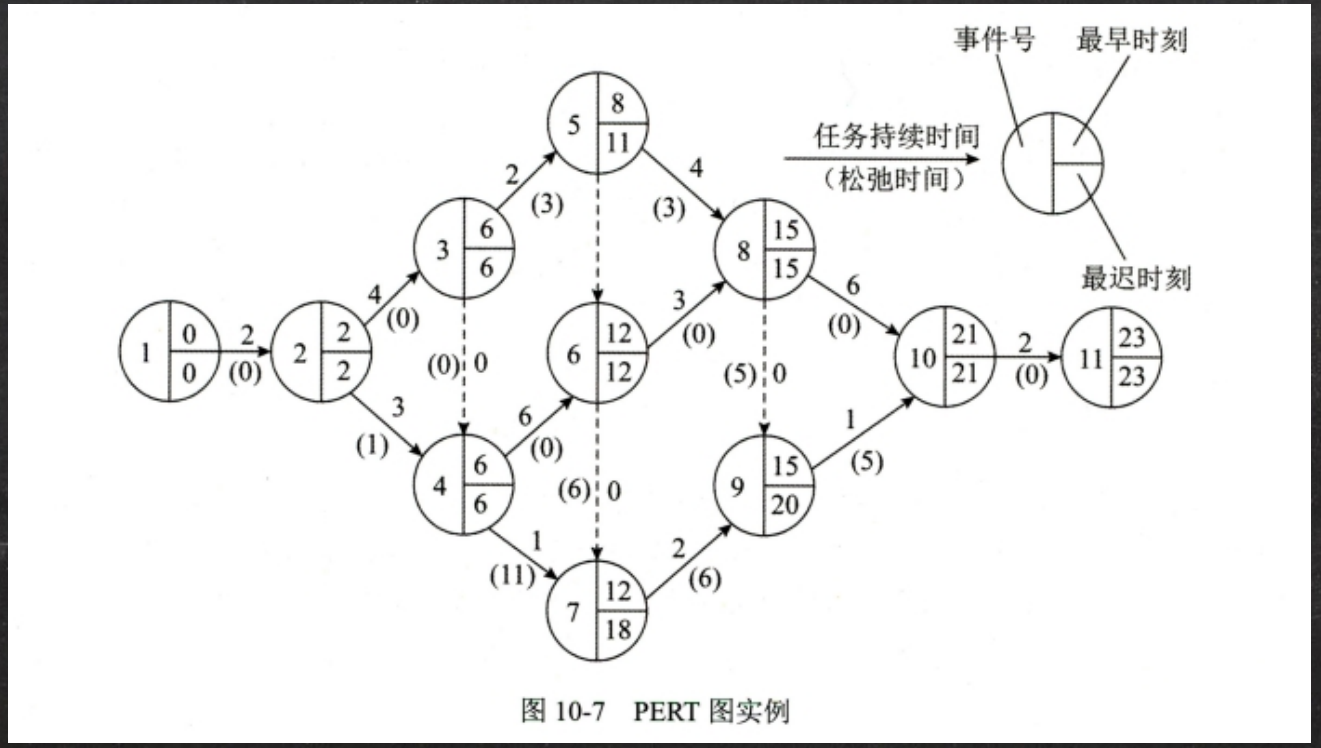


图 10-7 PERT 图实例

软件项目管理

4、人员管理

程序设计小组的组织形式可以有多种，如主程序员组、无主程序员组和层次式程序员组等。

（1）主程序员组。主程序员组由一名主程序员、一名后备程序员、一名资料员和若干名程序员组成。这种组织形式便于集中领导，步调统一，容易按规范办事，但不利于发挥个人的积极性。

（2）无主程序员组。无主程序员组中的成员之间相互平等，工作目标和决策都由全体成员民主讨论，根据需要也可以轮流坐庄。这种组民主气氛比较足，依赖个人的成分少，有利于发挥每个人的积极性。但这种组中交流量大，往往职责不明确，出了问题谁也不负责，而且不利于与外界的联系。

（3）层次式程序员组。层次式组中有一位组长，组长负责全面的工作，他领导若干名高级程序员，每个高级程序员又领导若干名程序员。这种组适合于具有层次结构特点的更大型的软件项目，该项目可分成若干个子项目，每个高级程序员负责一个子项目，然后再对子项目分解，并分配给程序员。

需求分析

1、需求分析的任务

需求分析主要是确定待开发软件的功能、性能、数据和界面等要求。具体来说，可有以下五个方面：

（1）确定软件系统的综合要求。

（2）分析软件系统的数据要求。包括基本元素、数据元素之间的逻辑关系、数据量和峰值等。常用的数据描述方法是实体-关系模型（E-R模型）。

（3）导出系统的逻辑模型。在结构化分析方法中可用数据流图来描述；在面向对象分析方法中可用类模型来描述。

（4）修正项目开发计划。

（5）如有必要，可开发一个原型系统。对一些需求不够明确的软件，可以先开发一个原型系统，以验证用户的需求。

2、需求的分类

软件需求包括功能需求、非功能需求和设计约束三方面的内容。

（1）功能需求：所开发的软件必须具备什么样的功能。

（2）非功能需求：是指产品必须具备的属性或品质，如可靠性、性能、响应时间、容错性和扩展性等。

（3）设计约束：也称为限制条件、补充规约，这通常是对解决方案的一些约束说明。

结构化分析方法

结构化分析（Structured Analysis，SA)方法是一种面向数据流的需求分析方法，适用于分析大型数据处理系统，是一种简单、实用的方法，现在已经得到广泛的使用。

结构化分析方法的基本思想是自顶向下、逐层分解。分解和抽象是人们控制问题复杂性的两种基本手段。对于一个复杂的问题，人们很难一下子考虑问题的所有方面和全部细节，通常可以把一个大问题分解成若干个小问题，每个小问题再分解成若干个更小的问题，经过多次逐层分解，每个最底层的问题都是足够简单，容易解决的，于是复杂的问题也就迎刃而解了。这个过程就是分解的过程。

SA方法的分析结果由以下几部分组成：一套分层的数据流图、一本数据字典、一组小说明（也称加工逻辑说明）、补充材料。

面向对象分析方法

1、 面向对象的基本概念

（1）对象：在面向对象的系统中，对象是基本的运行时的实体，它既包括数据（属性），也包括作用于数据的操作（行为）。

（2）消息：对象之间进行通信的一种构造叫作消息。

（3）类：一个类定义了一组大体上相似的对象。类是在对象之上的抽象，对象是类的具体化，是类的实例。

（4）继承：继承是父类和子类之间共享数据和方法的机制。这是类之间的一种关系，在定义和实现一个类的时候，可以在一个已经存在的类的基础上来进行，把这个已经存在的类所定义的内容作为自己的内容，并加入若干新的内容。

（5）多态：在收到消息时，对象要予以响应。不同的对象收到同一消息可以产生完全不同的结果，这一现象叫作多态。

（6）动态绑定和静态绑定：绑定是一个把过程调用和响应调用所需要执行的代码加以结合的过程。在一般的程序设计语言中，绑定是在编译时进行的，叫作静态绑定。动态绑定则是在运行时进行的，因此，一个给定的过程调用和代码的结合直到调用发生时才进行。

面向对象分析方法

UML 2.0中包括13种图。几种常见的图如下：

（1）类图：类图描述一组类、接口、协作和它们之间的关系。

（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的实例的静态快照。

（3）用例图：用例图描述一组用例、参与者与它们之间的关系。

（4）序列图：序列图是一种交互图，交互图展现了一种交互，它由一组对象或参与者以及它们之间可能发送的消息构成。交互图专注于系统的动态视图。序列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，是强调消息的时间次序的交互图。

（5）通信图：通信图也是一种交互图，它强调收发消息的对象或参与者的结构组织。序列图强调的是时序，通信图强调的是对象之间的组织结构(关系)。

（6）状态图：状态图描述一个状态机，它由状态、转换、事件和活动组成。状态图关注系统的动态视图。

面向对象分析方法

（7）活动图：活动图是一种特殊的状态图，它展现了在系统内从一个活动到另一个活动的流程。活动图专注于系统的动态视图，它强调对象间的控制流程。

（8）构件图：构件图描述一组构件之间的组织和依赖。构件图专注于系统的静态实现视图。它与类图相关，通常把构件映射为一个或多个类、接口或协作。

（9）部署图：部署图描述了运行时的处理节点以及其中的构件的配置。部署图给出了体系结构的静态实施视图。它与构件图相关，通常一个节点包含一个或多个构件。

系统设计

系统设计的主要目的就是为系统制定蓝图，在各种技术和实施方法中权衡利弊，精心设计，合理使用各种资源，最终勾画出新系统的详细设计方案。

系统设计分为概要设计和详细设计。

1、概要设计的基本任务

（1）设计软件系统总体结构

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

（3）编写概要设计文档

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、性能等要求进行评审。

系统设计

2、详细设计的基本任务

详细设计阶段的根本目标是确定应该怎样具体地实现所要求的系统。

（1）对每个模块进行详细的算法设计。

（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。

（5）编写详细设计说明书。

（6）评审. 对处理过程的算法和数据库的物理结构都要评审。

系统设计的基本原理

1、抽象

抽象是一种技术，重点说明一个实体的本质方面，而忽略或者掩盖不很重要或非本质的方面。

2、模块化

在软件的体系结构中，模块是可组合、分解和更换的单元。模块化是指将一个待开发的软件分解成若干个小的简单部分——模块，每个模块可独立地开发、测试，最后组装成完整的程序。模块化的目的是使程序的结构清晰，容易阅读、理解、测试和修改。

3、信息隐蔽

信息隐蔽是开发整体程序结构时使用的法则，即将每个程序的成分隐蔽或封闭在一个单一的设计模块中，定义每一个模块时尽可能少地显露其内部的处理。信息隐蔽原则对提高软件的可修改性、可测试性和可移植性都有重要的作用。

系统设计的基本原理

4、模块独立

模块独立是指每个模块完成一个相对独立的特定子功能，并且与其他模块之间的联系简单。衡量模块独立程度的标准有两个：耦合性和内聚性。

（1）耦合

耦合性是指模块之间联系的紧密程度。耦合性越高，则模块的独立性越差。

1）无直接耦合：指两个模块间没有直接的关系，它们分别从属于不同模块的控制与调用，它们之间不传递任何信息。

2）数据耦合：指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。

3）标记耦合：指两个模块之间传递的是数据结构，如高级语言中的数据组名、记录名、文件名等这些名字即为标记。

4：控制耦合：指一个模块调用另一个模块时，传递的是控制变量，被调模块通过该控制变量的值有选择地执行块内的某一功能。

5）公共耦合：指通过一个公共数据环境相互作用的那些模块之间的耦合。

6）内容耦合：是耦合性程度最高的。当一个模块直接使用另一个模块的内部数据，或通过非正常入口而转入另一个模块内部，这种模块之间的耦合为内容耦合。

系统设计的基本原理

（2）内聚

内聚是指模块内部各元素之间联系的紧密程度，例如一个完成多个功能的模块的内聚度就比完成单一功能的模块的内聚度低。内聚度越低，模块的独立性越差。

1）偶然内聚：指一个模块内的各个处理元素之间没有任何联系。

2）逻辑内聚：指模块内执行几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。

3）时间内聚：把需要同时执行的动作组合在一起形成的模块为时间内聚模块。

4）通信内聚：指模块内所有处理元素都在同一个数据结构上操作，或者指各处理使用相同的输入数据或产生相同的输出数据。

5）顺序内聚：指一个模块中各个处理元素都密切相关于同一功能且必须顺序执行，前一功能元素的输出就是下一功能元素的输入。

6）功能内聚：这是最强的内聚，指模块内所有元素共同完成一个功能，缺一不可。

将软件系统划分模块时，尽量做到高内聚、低耦合，提高模块的独立性。

系统测试基础知识

系统测试的目的是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。测试的目的就是希望能以最少的人力和时间发现潜在的各种错误和缺陷。

在进行信息系统测试时应遵循以下基本原则：

（1）应尽早并不断地进行测试。测试不是在应用系统开发完之后才进行的。

（2）测试工作应避免由原开发软件的人或小组承担。

（3）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输出结果，将实际输出结果与预期结果相比较就能发现测试对象是否正确。

（4）在设计测试用例时，不仅要设计有效合理的输入条件，也要包含不合理、失效的输入条件。

（5）在测试程序时，不仅要检验程序是否做了该做的事，还要检验程序是否做了不该做的事。

（6）严格按照测试计划来进行，避免测试的随意性。

（7）妥善保存测试计划、测试用例，作为软件文档的组成部分，为维护提供方便。

（8）测试例子都是精心设计出来的，可以为重新测试或追加测试提供方便。

系统测试基础知识

测试的类型：软件测试可分为单元测试、集成测试、确认测试、系统测试和回归测试等类别。

（1）单元测试：单元测试也称为模块测试。单元测试侧重于模块中的内部处理逻辑结构和数据结构。

（2）集成测试：集成测试的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。

（3）确认测试：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。

（4）系统测试：系统测试的对象是完整的、集成的计算机系统，系统测试的目的是在真实系统工作环境下，验证完整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。技术依据是用户需求或开发合同。

（5）回归测试：回归测试的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。

软件测试方法

1、黑盒测试法：

黑盒测试也称为功能测试，在完全不考虑软件的内部结构和特性的情况下，测试软件的外部特性。主要用于集成测试、确认测试和系统测试中。常用的黑盒测试技术有：等价类划分、边界值分析、错误推测和因果图等。

2、白盒测试法

白盒测试也称为结构测试，主要用于软件单元测试中。它的主要思想是将程序看作是一个透明的白盒，测试人员完全清楚程序的结构和处理算法，按照程序内部逻辑结构设计测试用例。

白盒测试常用的技术是逻辑覆盖，循环覆盖和基本路径测试。主要的逻辑覆盖标准有语句覆盖、判定覆盖、条件覆盖、条件/判定覆盖、 条件组合覆盖、路径覆盖等。

系统维护

系统的可维护性可定义为：维护人员理解、改正、改动和改进这个软件的难易程度。

1、系统可维护性的评价指标：

1）可理解性：是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。

2）可测试性：诊断和测试的容易程度取决于易理解的程度。

3）可修改性：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、软件维护的一般内容：

（1）正确性维护：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：是指使应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：这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。这些功能对完善系统功能是非常必要的。

（4）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。

【13年第15题】

“软件产品必须能够在3秒内对用户请求作出响应”属于软件需求中的（　　）。

A.功能需求　　B.非功能需求　　C.设计约束　　D.逻辑需求

【13年第19题】

某项目为了修正一个错误而进行了修改。错误修正后，还需要进行（　　）以发现这一修正是否引起原本正确运行的代码出错。

A.单元测试　　B.接受测试　　C.安装测试　　D.回归测试

【14年第18题】

以下关于进度管理工具Gantt图的叙述中，不正确的是（　　）。

A. 能清晰地表达每个任务的开始时间、结束时间和持续时间
B. 能清晰地表达任务之间的并行关系
C. 不能清晰地确定任务之间的依赖关系
D. 能清晰地确定影响进度的关键任务

【14年第19题】

项目复杂性、规模和结构的不确定性属于（　　）风险。

A.项目　　B.技术　　C.经济　　D.商业

【15年第16题】

在（　　）设计阶段选择适当的解决方案，将系统分解为若干个子系统，建立整个系统的体系结构。

A.概要　　B.详细　　C.结构化　　D.面向对象

【15年第27题】

某公司计划开发一个产品，技术含量很高，与客户相关的风险也很多，则最适于采用（ ）开发过程模型。

A.瀑布 B.原型 C.增量 D.螺旋

【16年第17、18题】

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。

(17) A.数据对象之间的关系，用于对数据建模

B.数据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.系统对外部事件如何响应，如何动作，用于对行为建模

D.数据流图中的各个组成部分

(18) A.读者 B.图书 C.借书证 D.借阅

【16年第19题】

软件开发过程中，需求分析阶段的输出不包括（ ）。

A.数据流图 B.实体联系图 C.数据字典 D.软件体系结构图

【17年第17、18题】

在采用结构化开发方法进行软件开发时，设计阶段接口设计主要依据需求分析阶段的（ ）。接口设计的任务主要是（ ）。

(17) A.数据流图 B.E-R图 C.状态-迁移图 D.加工规格说明

(18) A.定义软件的主要结构元素及其之间的关系

B.确定软件涉及的文件系统的结构及数据库的表结构

C.描述软件与外部环境之间的交互关系，软件内模块之间的调用关系

D.确定软件各个模块内部的算法和数据结构

【17年第19题】

在进行软件开发时，采用无主程序员的开发小组，成员之间相互平等；而主程序员负责制的开发小组，由一个主程序员和若干成员组成，成员之间没有沟通。在一个由8名开发人员构成的小组中，无主程序员组和主程序员组的沟通路径分别是（ ）。

A.32和8　　B.32和7　　C.28和8　　D.28和73

【18年第23、24题】

在下列机制中，（ ）是指过程调用和响应调用所需执行的代码在运行时加以结合；而（ ）是过程调用和响应调用所需执行的代码在编译时加以结合。

A. 消息传递　　B.类型检查　　C.静态绑定　　D.动态绑定

A. 消息传递　　B.类型检查　　C.静态绑定　　D.动态绑定

【18年第25题】

耦合是模块之间的相对独立性（互相连接的紧密程度）的度量。耦合程度不取决于（ ）

A. 调用模块的方式
B. 各个模块之间接口的复杂程度
C. 通过接口的信息类型
D. 模块提供的功能数

【18年第26题】

以下关于软件可靠性测试的叙述中，错误的是（ ）。

A. 软件可靠性测试的目的是评估软件系统的可靠性
B. 软件可靠性测试前应先确定软件可靠性的目标
C. 应平衡地考虑对软件开发进度和成本的影响
D. 应选用适用于所有软件的标准的可靠性测试模型

【19年第25题】

以下关于系统原型的叙述中，不正确的是（ ）

A. 可以帮助导出系统需求并验证需求的有效性
B. 可以用来探索特殊的软件解决方案
C. 可以用来指导代码优化
D. 可以用来支持用户界面设计

【19年第26题】

已知模块A给模块B传递数据结构X，则这两个模块的耦合类型为（ ）

A. 数据耦合　　B. 公共耦合　　C. 外部耦合　　D. 标记耦合

【19年第27题】

以下关于软件测试的叙述中，正确的是（ ）

A. 软件测试的目的是为了证明软件是正确的
B. 软件测试是为了发现软件中的错误
C. 软件测试是在软件实现之后开始，在软件交付之前完成
D. 如果对软件进行了充分的测试，那么交付时软件就不存在问题了

【19年第28题】

数据流图建模应遵循（ ）的原则

A. 自顶向下、从具体到抽象　　B. 自顶向下、从抽象到具体
C. 自底向上、从具体到抽象　　D. 自底向上、从抽象到具体

【20年第25题】

传统过程模型中，（　　）首先引入了风险管理。

A.瀑布模型　　B. 螺旋模型　　C. V模型　　D. 原型化模型

【20年第26题】

以下有关测试的说法中，错误的是（　）

A. 测试证明了程序的正确性

B. 测试无法执行穷举测试，只能做选择测试

C. 测试工作需要制定测试计划，按计划执行测试工作

D.白盒测试方法用于单元测试环节

【20年第27~28题】

在软件设计中通常用耦合度和内聚度作为衡量模块独立程度的标准，耦合程度最高的是（　）耦合；内聚程度最高的是（　）内聚。

（27）A. 数据　　B. 内容　　C. 标记　　D. 公共

（28）A. 顺序　　B. 功能　　C. 时间　　D. 逻辑

【21年第25题】

ISO软件质量模型由3个层次组成，分别是质量特性，质量子特性和量度指标。例如（　）质量子特性属于可靠性质量特性。

A.依从性　　B.成熟性　　C.易操作性　　D.易安装性

【21年第26题】

在UML图中，（　）展现了一组对象以及它们之间的关系，描述了类实例的静态快照。

A.类图　　B.对象图　　C.序列图　　D.状态图

【21年第27题】

（ B ）强调风险分析，比较适用于庞大、复杂且高风险的系统。

A.瀑布模型　　B. 螺旋模型　　C. V 模型　　D. 原型化模型

【21年第28题】

软件能力成熟度模型（CMM）是对软件组织进化阶段的描述，分为5个成熟度级别，其中在（　）级别，说明该组织已经建立了基本的项目管理过程来跟踪成本和进度。

A.可重复级　　B.已定义级　　C.已管理级　　D.优化级

【22年第25题】

软件过程模型中，（ ）首次引入风险管理。

A.螺旋模型 B.瀑布模型 C.V模型 D.原型化模型

【22年第26题】

某软件需求“发送消息需要在1秒钟内得到响应”，该需求属于（ ）。

A.功能需求 B.非功能需求 C.设计约束 D.过程约束

【22年第27题】

数据流图设计中，（ ）描述输入数据流到输出数据流之间的转换。

A.外部系统 B.数据存储 C.加工 D.用户

【22年第28题】

在UML图中，（ ）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。

A.类图 B.对象图 C.序列图 D.状态图