第一章

一、什么是软件？

1.满足功能要求和性能的指令或计算机程序集合；

2.处理信息的数据结构；

3.描述程序功能以及程序如何操作和使用所要求的文档；

二、软件的特点：

1)软件是一种逻辑实体，而不是具体的物理实体，因而它具有抽象性。

2)软件是通过人们的智力活动，把知识与技术转换成信息的一种产品，是在研制、开发中被创造出来的

3)在软件运行和使用的期间，没有硬件那样的机械磨损、老化问题

4)软件的开发和运行经常受到计算机系统的限制，对计算机系统有着不同程度的依赖性

5)软件的开发至今尚未完全摆脱手工的开发方式

6）软件的开发费用越来越高，成本相当昂贵。

三、软件危机的概念：软件危机是指计算机软件的开发和维护过程中所遇到的一系列严重的问题。这些问题表现在以下几个方面：

(1)用户对开发出的软件很难满意。

(2)软件产品的质量往往靠不住。

(3)一般软件很难维护。

(4)软件生产效率很低。

(5)软件开发成本越来越大。

(6)软件成本与开发进度难以估计。

(7)软件技术的发展远远满足不了计算机应用的普及与深入的需要。

四、产生软件危机的原因：(1) 开发人员方面，对软件产品缺乏正确认识，没有真正理解软件产品是一个完整的配置组成。造成开发中制定计划盲目、编程草率，不考虑维护工作的必要性。(2) 软件本身方面，对于计算机系统来说，软件是逻辑部件，软件开发过程没有统一的、公认的方法论和规范指导，造成软件维护困难。(3) 尤其是随着软件规模越来越大,复杂程度越来越高,原有软件开发方式效率不高、质量不能保证、成本过高、研制周期不易估计、维护困难等一系列问题更为突出，技术的发展已经远远不能适应社会需求。

五、软件危机的两个主要问题：如何开发软件，以满足对软件日益增长的需求；如何维护数量不断膨胀的已有软件。

六、软件产品的重用性差，同样的软件多次重复开发。

七、软件通常没有适当的文档资料。

八、软件危机的典型表现：

(1) 对软件开发成本和进度的估计常常很不准确。

(2) 用户对“已完成的”软件系统不满意的现象经常发生。

(3) 软件产品的质量往往靠不住。

(4) 软件常常是不可维护的。

(5) 软件通常没有适当的文档资料。

(6) 软件成本在计算机系统总成本中所占的比例逐年上升。

(7) 软件开发生产率提高的速度，远远跟不上计算机应用迅速普及深入的趋势。

九、软件配置的主要包括：程序、文档和数据等成分。

十、软件工程的定义：软件工程是应用计算机科学、数学及管理科学等原理开发软件的工程。它借鉴传统工程的原则、方法，以提高质量，降低成本为目的。

软件工程的本质特性：

a、软件工程关注于大型程序的构造；

b、软件工程的中心课题是控制复杂性；

c、软件经常变化；

d、开发软件的效率非常重要；

e、和谐地合作是开发软件的关键；

f、软件必须有效地支持它的用户；

g、在软件工程领域中通常由具有一种文化背景的人替具有另一种文化背景的人创造产品。

软件生命周期每个阶段的基本任务：

a、问题定义 ：明白要解决的问题是什么；

b、可行性研究:研究问题的范围，探索这个问题是否值得去解，是否有可行的解决办法；

c、需求分析:主要是确定目标系统必须具备哪些功能；

d、总体设计:a、设计出实现目标系统的集中可能的方案； b、设计程序的系统结构，就是确定程序由哪些模块组成以及模块间的关系；

e、详细设计:把解法具体化，设计出程序的详细规格说明；

f、编码和单元测试:写出正确的容易理解、容易维护的程序模块；

g、综合测试: 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。

软件工程是指研究软件生产的一门学科，也就是将完善的工程原理应用于经济地生产既可靠又能在实际机器上有效运行的软件。

软件工程方法学包括 3 个要素：方法、工具、过程。

软件工程定义：软件工程是指导计算机软件开发和维护的一门工程学科。采用工程的概念、原理、技术和方法来开发与维护软件，把经过时间考验而证明正确的管理技术和当前能够得到的最好的技术方法结合起来，以经济地、高效的开发出高质量的软件并有效地维护它，这就是软件工程。

软件工程准则可以概括为 7 条基本原则：

a 用分阶段的生命周期计划严格管理；

b 坚持进行阶段评审

c 实行严格的产品控制

d 采用现代程序设计技术

e 应能清楚地审查结果

f 合理安排软件开发小组的人员

g 承认不断改进软件工程实践的必要性

所谓**基准配置**又称**基线配置**。

通常把在软件生命周期全过程中使用的一整套技术方法的集合称为**方法学**，也成为**范型**

目前**使用得最广泛的软件工程方法学**，分别是**传统方法学**和**面向对象方法学**

传统方法学也称为**生命周期方法学**或**结构化范型**

面向对象方法学的四个要点：a 把对象作为融合了数据及在数据上的操作行为的统一的软件构件 b 把所有对象都划分成类 c 按照父类（或称为基类）与子类（或称为派生类）的关系，把若干个相关类组成一个层次结构的系统（也称为类等级）。d 对象彼此间仅能通过发送消息互相联系。

软件生命周期:由软件定义、软件开发和软件维护三个时期组成。

1．软件定义时期:①问题定义　②可行性研究

2.软件开发时期:①需求分析　②总体设计　③详细设计　④编码　⑤测试

3．软件运行时期 :维护

软件孕育、诞生、成长、成熟、衰亡的生存过程，一般称之为**计算机软件的生存期。**

软件生命周期（概念、三时期，八阶段）

软件生命周期由软件定义、软件开发和运行维护(也称为软件维护)3 个时期组成。

软件定义时期通常进一步划分成 3 个阶段，即问题定义、可行性研究和需求分析。

软件开发时期分为 4 阶段：总体设计、详细设计、编码和单元测试、综合测试

12.**最基本的测试**是**集成测试**和**验收测试**。

13.瀑布模型，快速原型模型，增量模型，螺旋模型，喷泉模型，概念.方法.优缺点.区别。

瀑布模型:将软件生存周期的各项活动规定为依照固定顺序连接的若干阶段工作，形如瀑布流水，最终得到软件产品。(不适合需求模糊的系统)

瀑布模型的优缺点：

a、优点：可强迫开发人员采用规范的方法（例：结构化技术）； 严格地规定每个阶段必须提交的文档； 要求每个阶段交出的所有产品都必须经过质量保证小组的仔细验证。

b、缺点：瀑布模型基本上是一种文档驱动的模型。

增量模型：是瀑布模型的顺序特征与快速原型法德迭代特征相结合的产物。这种模型把软件看成一系列相互联系的增量，在看法过程的各次迭代中，每次完成其中的一个增量。

快速原型模型： 所谓快速原型是快速建立起来的可以在计算机上运行的程序，它所能完成的功能往往是最终产品能完成的功能的一个子集。快速原型模型的第一步是快速建立一个能反映用户主要需求的原型系统，让用户在计算机上试用它，通过实践来了解目标系统的概貌

数据字典:关于数据的信息的集合，也就是对数据流图中包含的所有元素的定义的集合。 数据字典是在数据流程图的基础上，对数据流程图中的各个元素进行详细的定义与描述，起到对数据流程图进行补充说明的作用。 **内容：**①数据流 ②数据流分量 ③数据存储 ④处理

数据字典的作用是:对用户来讲,数据字典为他们提供了数据的明确定义;对系统分析员来讲,数据字典帮助他们比较容易修改已建立的系统逻辑模型。

数据字典的实现：P49

成本效益分析:成本/效益分析的目的是要从经济角度分析开发一个特定的新系统是否可行，从而帮助使用部门负责人正确地做出是否投资与这项开发工程的决定。**几种度量效益的方法：**货币的时间价值、投资回收期、纯收入

所谓**构件**就是功能清晰的模块或子系统

RUP **软件开发的生命周期**是一个二维的生命周期模型

"极限"二字的含义是指把好的开发实践运用到极致

微软过程把软件生命周期划分为成 5 个阶段：规划阶段，设计阶段，开发阶段，稳定阶段，发布阶段。

面向对象方法=对象+类+继承+用消息通信

模块化解决一个复杂问题时自顶向下逐层把软件系统划分成若干模块的过程逐步求精为了能集中精力解决主要问题而尽量推迟对问题细节的考虑。
信息隐蔽在设计和确定模块时，使得一个模块内包含的信息，对于不需要这些信息的模块来说，是不能访问的。
黑盒测试把测试对象看做一个黑盒子，测试人员完全不考虑程序内部的逻辑结构和内部特性，只依据程序的需求规格说明书，检查程序的功能是否符合它的功能说明。
白盒测试把测试对象看做一个透明的盒子，它允许测试人员利用程序内部的逻辑结构及有关信息，(设计或选择测试用例，) 对程序所有逻辑路径进行测试。(通过在不同点检查程序的状态，确定实际的状态是否与预期的状态一致。)
Beta 测试软件的多个用户在实际使用环境下进行的测试,是在开发者无法控制的环境下进行的软件现场应用。
投资回收期使累计的经济效益等于最初投资所需要的时间。
小结：
1、生命周期方法学把软件生命周期划分为若干个相对独立的阶段，每个阶段完成一些确定的任务，交出最终的软件配置的一个或几个成分（文档或程序）。
2、面向对象方法 = 对象 + 类 + 继承 + 用消息通信
　　也就是说，面向对象方法就是既使用对象又使用类和继承等机制，而且对象之间仅能通过传递消息实现彼此通信的方法。
3、软件过程是为了获得高质量的软件产品需要完成的一系列任务的框架，它规定了完成各项任务的工作步骤。

第二章　可行性研究

可行性研究的目的就是用最小的代价在尽可能端的时间内确定问题能否能够解决。
可行性包括：技术可行性，经济可行性，操作可行性。
可行性研究的目的不是解决问题，而是确定问题是否值得去解决。
可行性研究的任务从技术可行性、经济可行性、操作可行性、社会可行性等方面研究问题的可行性。
从 3 个方面研究每种解法的可行性：技术可行性、经济可行性、操作可行性。
（必要时还应该从法律、社会效益等更广泛的方面研究每种解法的可行性。）
可行性研究过程：
1）复查系统规模和目标
2）研究目前正在使用的系统
3）导出新系统的高层逻辑模型
4）进一步定义问题
5）导出和评价供选择的解法
6）推荐行动方针
7）草拟开发计划
8）书写文档提交审查
可行性研究的方法：可行性研究进一步探讨问题定义阶段所确定的问题是否有可行的解。在对问题正确定义的基础，通过分析问题（往往需要研究现在正在使用得系统），导出试探性的解，然后复查并修正问题定义，再次分析问题，改进提

出的解放… …。经过定义问题、分析问题、提出解法的反复过程，最终提出一个符合系统目标的高层次的逻辑模型。然后根据系统的这个逻辑模型设想各种可能的物理系统，并且从技术、经济和操作等各方面分析这些物理系统的可行性。最后，系统分析员提出一个推荐的行动方针，提交用户和客户组织负责人审查批准。

从哪些方面验证软件需求的正确性①一致性　②完整性　③现实性　④有效性

系统流程图是概括地描绘物理系统的传统工具。它的**基本思想是**用图形符号以黑盒子形势描绘组成系统的每个部件（程序，文档，数据库，人工过程等）。系统流程图表达的是数据在系统各部件之间流动的情况，而不是对数据加工处理的控制过程，因此尽管系统流程图的某些符号和程序流程图的符号形式相同，但是它却是物理数据流图而不是程序流程图。

数据流图(DFD)是一种图形化技术，它描绘信息流和数据从输入移动到输出的过程中所经受的变换。在数据流图中没有任何具体的物理部件，它只描绘数据在软件中流动和被处理的逻辑过程。数据流图是系统逻辑功能的图形表示。

用系统流程图描绘一个系统时，系统的功能和实现每个功能的具体方案是混在一起的。

有数据元素组成的数据的方式只有下述 3 种基本类型：顺序（即以确定次序连接两个或多个分量);选择 (即从两个或多个可能的元素中选取一个); 重复(即把指定的分量重复零次或多次)

总体设计的过程

- 设想供选择的方案
- 选取合理的方案
- 推荐最佳方案
- 功能分解
- 设计软件结构
- 设计数据库
- 制定测试计划
- 书写文档
- 审查和复审

软件测试步骤

1、模块测试　2、子系统测试　3、系统测试
4、验收测试　5、平行测试

总体设计书写文档

系统说明、用户手册、测试计划、详细的实现计划、数据库设计结果

信息流 2 种类型：变换流、事务流

软件系统的文档分为：用户文档、系统文档

文档作用:是影响软件可维护性的决定因素.

其他知识点

瀑布模型阶段的顺序性和依赖性、质量保证

快速原型模型快速开发工具、 循环、 低成本
增量模型迭代的思路
螺旋模型风险分析迭代过程
喷泉模型以用户需求为动力、适合面向对象的开发方法、开发过程具有迭代性
数据流图基本符号有 4 种：

成本估计技术
1、代码行技术 2、任务分解技术 3、自动估计成本技术
软件**开发过程**是一个**自顶向下，逐步细化**的过程
测试过程是**依相反顺序安排的自底向上，逐步集成**的过程。
软件是计算机系统中与硬件相互依存的另一部分,它包括**程序,数据及其相关文档的完整集合.**
需求分析阶段的文档
- 软件需求说明书
- 数据要求说明书
- 初步的用户手册
- 修改、完善与确定软件开发实施计划

白盒测试技术： 白盒测试时将程序看作是一个透明的盒子，也就是说测试人员完全了解程序的内部结构和处理过程。所以测试时按照程序内部的逻辑测试程序、检验程序中的每条通路是否都能按预定的要求正确工作。白盒测试又称为结构测试。

白盒测试多用于单元测试阶段。逻辑覆盖是主要的白盒测试技术。白盒测试时，确定测试数据应根据程序的内部逻辑和指定的覆盖方式。采用一下几种逻辑覆盖标准：
逻辑覆盖:1.语句覆盖(最弱的,每个可执行语句至少执行一次)
2.判定覆盖.(包含语句覆盖)
3.条件覆盖.(和判定覆盖没有包含关系)
4.判定/条件覆盖 5.条件组合覆盖.(最强)
6.点覆盖. 7.边覆盖. 8.路径覆盖.
满足条件组合覆盖测试用例，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。
控制结构测试:1.基本路径测试(每个循环语句至多执行一次)
2.条件测试.
3.循环测试.
应用:局部数据结构,独立路径,出错处理,单元测试(白盒测试为主,黑盒测试为辅)
黑盒测试技术：黑盒测试时完全不考虑程序内部的结构和处理过程，只按照规格说明书的规定来检查程序是否符合它的功能要求。黑盒测试是在程序接口进行的测试,又称为功能测试。
方法:1.等价划分
等价类:在该子集中,各个输入数据对于揭露程序中的错误都是等效的.
原则:尽可能多的覆盖有效等价类,尽可能少的覆盖无效等价类.
2.边界值分析
3.错误推测(经验)
应用:边界条件,模块接口,集成测试
其中等价类划分和边界值分析法方法最常用。如果两者结合使用，更有可能发现软件中的错误。

系统流程图的定义和作用：

可行性研究对现有系统做概括的物理模型描述，如用图形工具表示则更加直观简洁。系统流程图是描绘物理系统的传统工具,它的基本思想是用图形符号以黑盒子形式描绘系统里面的每个部件(程序、文件、数据库、表格、人工过程等)。系统流程图表达的是部件的信息流程，而不是对信息进行加工处理的控制过程。在可行性研究过程中，利用系统流程图来描述所建议系统的物理模型。

数据流程图的定义和作用：数据流程图有两个特征：抽象性和概括性。

抽象性指的是数据流程图把具体的组织机构、工作场所、物质流都去掉，只剩下信息和数据存储、流动、使用以及加工情况。

概括性则是指数据流程图把系统对各种业务的处理过程联系起来考虑，形成一个总体

数据流程图的组成元素

数据流图可以用来抽象地表示系统或软件。它从信息传递和加工的角度，以图形的方式刻画数据流从输入到输出的移动变换过程，同时可以按自顶向下、逐步分解的方法表示内容不断增加的数据流和功能细节。因此，数据流图既提供了功能建模的机制，也提供了信息流建模的机制，从而可以建立起系统或软件的功能模型。

数据流程图的组成：外部实体（外部实体是指系统之外的人或单位，它们和本系统有信息传递关系）数据流，处理、数据存储。

如何绘制数据流程图

(1)识别系统的输入和输出，画出顶层图

(2)画系统内部的数据流、加工与文件，画出一级细化图

(3)加工的进一步分解，画出二级细化图

(4)其它注意事项

数据流程图的注意点

1）每个处理都必须有流入的数据流和流出的数据流，如果没有，是错误的。（数据守恒）

2）每个数据存储应该有流入的数据流和流出的数据流，如果缺了一种，是 Warning 的；缺两种就错了。

3)、数据流只能在处理与处理、数据存储或者外部实体之间流动。、数据存储到数据存储、外部实提到外部实体、外部实提到数据存储之间的数据流都是错误的。

4)、 一个处理可以细分成多个子处理，分成若干个层次（均匀分解）

5)、良好命名

系统流程图与数据流程图有什么区别?

答：1）系统流程图描述系统物理模型的工具，数据流程图描述系统逻辑模型的工具。

2)系统流程图从系统功能的角度抽象的描述系统的各个部分及其相互之间信息流动的情况。

3）数据流程图从数据传送和加工的角度抽象的描述信息在系统中的流动和数据处理的工作状况。

三、数据流图：

1、组成符号：4中基本图形符号正方形、圆角矩形、开口矩形

2、数据流图的基本要点是描绘“做什么”，而不是考虑“怎么做”。

3、一套分层的的数据流图由顶层、底层、和中间层组成。

4、画分层数据流图基本原则与注意事项：

a.自外向内，自顶向下，逐层细化，完善求精。

b.保持父图与子图的平衡。也就是说，父图中某加工的输入数据流中的数据必须与它的子图的输入数据流在数量和名字上相同。
c.保持数据守恒。也就是说，一个加工所有输出数据流中的数据必须能从该加工的输入数据流中直接获得，或者是通过该加工能产生的数据。
d.加工细节隐藏。根据抽象原则，在画父图时，只需画出加工和加工之间的关系，而不必画出各个加工内部的细节。
e.简化加工间关系。在数据流图中，加工
间的数据流越少，各加工就越相对独立，所以应尽量减少加工间输入输出数据流的数目。
f.均匀分解。应该使一个数据流中的各个加工分解层次大致相同。
g.适当地为数据流、加工、文件、源/宿命名，名字应反映该成分的实际意义，避免空洞的名字。
h.忽略枝节。应集中精力于主要的数据流，
而暂不考虑一些例外情况、出错处理等枝节性问题。
i.表现的是数据流而不是控制流。
j.每个加工必须既有输入数据流，又有输出数据流.在整套数据流图中，每个文件必须既有读文件的数据流又有写文件的数据流,但在某一张子图中可能只有读没有写或者只有写没有读。
小结：一个软件系统，其数据流图往往有多层。如果父图有 N 个加工(Process)，则父图允许有 0～N 张子图，但是每张子图只能对应一张父图。在一张 DFD 图中，任意两个加工之间可以有 0 条或多条名字互不相同的数据流；在画数据流图时，应该注意父图和子图的平衡，即父图中某加工的输入输出数据流必须与其输入输出流在数量和名字上相同。DFD 信息流大致可分为两类：交换流和事务流。

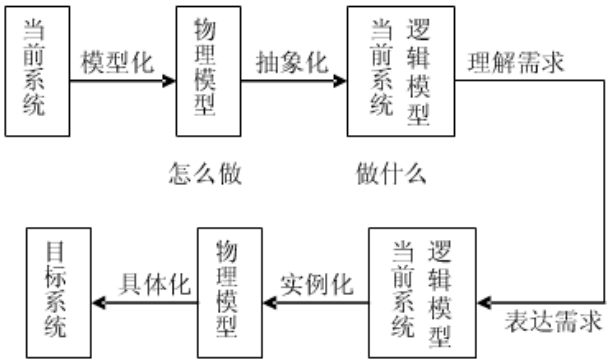
第三章

25.**访谈有两种基本形式**，分别是正式的和非正式的访谈
26.所谓**情景分析**就是对用户将来使用目标系统解决某个具体问题的方法和结果进行分析
27.**结构化分析方法**就是面向数据流自顶向下逐步求精进行需求分析的方法。
28.使用简易的应用规格说明技术分析需求的典型过程：（总结出来）
29.**快速原型**就是快速建立起来的旨在演示目标系统主要功能的可运行的程序。
30.所谓**模型**就是为了理解事物而对事物作出的一种抽象，是对事物的一种无歧义的书面描述。
31.**需求分析过程应该建立 3 种模型**，它们分别是数据模型，功能模型，行为模型.
32.**概念性数据模型**是一种面向问题的数据模型，是按照用户的观点对数据建立的模型
33.数据对象是对软件必须理解的符合信息的抽象。
34.数据对象彼此之间相互连接的方式称为**联系**，也称为关系。**联系**可分为 3 种类型：一对一联系，一对多联系，多对多联系。
35.状态时任何可以被观察到的系统行为模式，一个状态代表系统的一种行为模式。
36.**事件**就是引起系统做动作或（和）转换状态的控制信息。
37.**IPO 图**是输入，处理，输出图的简称。
38.**软件的验证**：一致性，完整性，现实性，有效性
为什么要做需求分析
可行性分析研究阶段已经粗略的描述了用户的需求，甚至还提出了一些可行的方案，但是，许多细节被忽略了，在最终目标系统中是不能忽略、遗漏任何一个微小细节的，所以，可行

性研究不能代替需求分析。

需求分析的方法：需求分析方法由对软件的数据域和功能域的系统分析过程及其表示方法组成，它定义了表示系统逻辑视图和物理视图的方式，大多数的需求分析方法是由数据驱动的，也就是说，这些方法提供了一种表示数据域的机制，分析员根据这种表示，确定软件功能及其特性，最终建立一个待开发软件的抽象模型，即目标系统的逻辑模型。

需求分析的任务：它的基本任务是准确地回答“系统必须做什么？”这个问题。需求分析所要做的工作是深入描述软件的共能和性能，确定软件设计的限制和软件同其它系统元素的接口细节，定义软件的其它有效性需求。需求分析的任务不是确定系统如何完成它的工作，而是确定系统必须完成哪些工作，也就是对目标系统提出完整、准确、清晰、具体的要求。其实现步骤如下图所示：



一般说来需求分析阶段的任务包括下述几方面：

1） 确定对系统的综合需求

对系统的综合需求主要有：系统功能需求、系统性能需求、可靠性和可用性需求、错处理需求、接口需求、约束、逆向需求、将来可能提出的需求:

2） 分析系统的数据需求

就是在理解当前系统“怎样做”的基础上，抽取其“做什么”的本质，明确目标系统要“做什么”，可以导出系统的详细的逻辑模型。具体做法：首先确定目标系统与当前系统的逻辑差别；然后将变化部分看作是新的处理步骤，对功能图（一般为数据流图）及对象图进行调整；最后有外及里对变化的部分进行分析，推断其结构，获得目标系统的逻辑模型。通常用数据流图、数字字典和主要的处理算法描述这个逻辑模型。

3） 导出系统的逻辑模型

4） 修正系统开发计划

在经过需求分析阶段的工作，分析员对目标系统有了更深入更具体的认识，因此可以对系统的成本和进度做出更准确地估计，在此基础上应该对开发计划进行修正。

5）开发原型系统：使用原型系统的主要目的是，使用户通过实践获得关于未来的系统将怎样为他们工作的更直接更具体的概念，从而可以更准确地提出他们的要求。

4、需求分析的步骤：1）调查研究 2）分析与综合 3）书写文档 4）需求分析评审

5、需求分析的原则：

1)、必须能够表达和理解问题的数据域和功能域

2)、按自顶向下、逐层分解问题

3)、要给出系统的逻辑视图和物理视图

6、软件需求的验证：

需求分析阶段的工作结果是开发软件系统的重要基础，大量统计数字表明，软件系统中15%的错误起源于错误的需求。为了提高软件质量，确保软件开发成功，降低软件开发成本，一旦对目标系统提出一组要求之后，必须严格验证这些需求的正确性。一般说来，应该从下述4个方面进行验证：

(1) 一致性所有需求必须是一致的，任何一条需求不能和其他需求互相矛盾。

(2) 完整性需求必须是完整的，规格说明书应该包括用户需要的每一个功能或性能。

(3) 现实性指定的需求应该是用现有的硬件技术和软件技术基本上可以实现的。对硬件技术的进步可以做些预测，对软件技术的进步则很难做出预测，只能从现有技术水平出发判断需求的现实性。

(4) 有效性必须证明需求是正确有效的，确实能解决用户面对的问题。

7、 **状态转换图：**指明了作为外部事件结果的系统行为。为此，状态转换图描绘了系统的各种行为模式(称为“状态”)和在不同状态间转换的方式。状态转换图是行为建模的基础。

需求分析的方法：传统软件工程方法学使用结构化分析技术，完成分析用户需求的工作。需求分析是发现、求精、建模、规格说明和复查的过程。需求分析的第一步是进一步了解用户当前所处的情况，发现用户所面临的问题和对目标系统的基本需求；接下来应该与用户深入交流，对用户的基本需求反复细化逐步求精，以得出对目标系统的完整、准确和具体的需求。具体地说，应该确定系统必须具有的功能、性能、可靠性和可用性，必须实现的出错处理需求、接口需求和逆向需求，必须满足的约束条件及数据需求，并且预测系统的发展前景。

第五章

42.**总体设计过程通常由两个主要阶段组成：**系统设计阶段，确定系统的具体实现方案；结构设计阶段，确定软件结构。

43.**模块**是由边界元素限定的相邻程序元素（例如，数据说明，可执行的语句）的序列。

44.**抽象**就是抽出事物的本质特性而暂时不考虑它们的细节。

45.**逐步求精定义：**为了能集中精力解决主要问题而尽量推迟对问题细节的考虑。

46.抽象程序对抽象的数据进行某些特定的运算并用某些合适的记号（可能是自然语言）来表示。

47.**信息隐藏，信息隐藏**的原理，

48.**局部化**就是把一些关系密切的软件元素物理的放得彼此靠近。

50.数据耦合是低耦合，控制耦合式中等程度的耦合，最高程度的耦合式内容耦合。

51.如果一个模块完成一组任务，这些任务彼此间即使有关系，关系也是很松散的，就叫做**偶然内聚**。

52.**中内聚主要有两类：**如果一个模块内的处理元素是相关的，而且必须以特定次序执行，则称为过程内聚。

53.**高内聚也有两类：**如果一个模块内的处理元素和同一个功能密切相关，而且这些处理必须顺序执行，则称为顺序内聚。深度表示软件结构中控制的层数，它往往能粗略的标志一个系统的大小和复杂程度。

54.**宽度**是软件结构内同一个层次上的模块总数的最大值。

56.一个模块的扇入表明有多少个上级模块直接调用它。

57.设计的很好的软件结构通常顶层扇出比较高，中层扇出比较少，底层扇入到公共的实用

模块中去（底层模块有高扇入）
58.**模块的作用域应该在控制域之内**（2 种方法）？？？？？
59.面向数据流的设计方法把信息流映射成软件成结构，信息流的类型决定了映射的方法。**信息流**有两种类型：变换流，事务流。
60.总体设计阶段的基本目的是用比较抽象概括的方式确定系统如何完成预定的任务，也就是说，应该确定系统的物理配置方案。并且进而确定组成系统的每个程序的结构。
61.在进行软件结构设计时应该遵循的最主要的原理是**模块独立原理**。

1、**模块的独立性很重要，主要有两条理由**：a、有效的模块化（即具有独立的模块）的软件比较容易开发出来；b、独立的模块比较容易测试和维护。
2、模块的独立程度可以由**两个定性标准度量**，这两个标准分别是内聚和耦合。
a、**耦合**是对一个软件结构内不同模块之间互联程度的度量；（注：尽量使用数据耦合，少用控制耦合和特征耦合，限制公共环境耦合的范围，完全不用内容耦合。）
b、**内聚**标志着一个模块内各个元素彼此结合的紧密程度，它是信息隐藏和局部化概念的自然扩展。
3、**扇出**是一个模块直接控制（调用）的模块数目，扇出过大意味着模块过分复杂，需要控制和协调过多下级模块；扇出过小也不好。经验表明，一个设计得好的典型系统的平均扇出通常是 3 或 4（扇出的上限通常是 5-9）。

5、 衡量模块独立的两个标准是什么？它们各表示什么含义？

答：衡量模块的独立性的标准是两个定性的度量标准：耦合性和内聚性。

(1)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，模块的独立性则越差。模块间耦合高低取决于模块间接口的复杂性、调用的方式及传递的信息。

(2)内聚性。又称块内联系。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。若一个模块内各元素(语句之间、程序段之间)联系得越紧密，则它的内聚性就越高。

耦合性与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时，尽量做到高内聚低耦合，提高模块的独立性，为设计高质量的软件结构奠定基础。

模块的高内聚、低耦合的原则称为模块独立原则，也称为模块设计的原则。

五、总体设计（概要设计）

重点掌握的内容：概要设计的过程和方法
一般掌握的内容：概要设计的文档和评审
考核知识点：

一、总体设计：

1、总体设计的**目的**：总体设计的基本目的就是回答“概括地说，系统应该如何实现?”这个问题，因此，总体设计又称为概要设计或初步设计。1、面向结构设计(SD)2、面向对象设计(OOD)

2、总体设计的**任务**：

1）系统分析员审查软件计划、软件需求分析提供的文档、提出最佳推荐方案，用系统流程图，组成物理元素清单，成本效益分析，系统的进度计划，供专家沈顶峰，审定后进入设计

2）去顶模块结构，划分功能模块，将软件功能需求分配给所划分的最小单元模块。确定模块之间的联系，确定数据结构、文件结构、数据库模式，确定测试方法与策略。
3）编写概要设计说明书，用户手册，测试计划，选用相关的软件工具来描述软件结构，结构图是经常使用的软件描述工具。选择分解功能与划分模块的设计原则，例如模块划分独立性原则，信息隐蔽原则等
3、总体设计过程通常由**两个主要阶段组成**：系统设计阶段，确定系统的具体实现方案；结构设计阶段，确定软件结构。
4、**典型的总体设计过程包括下述9个步骤：**
1)、设想功选择的方案
2)、选取合理的方案
3)、推荐最佳方案
4)、功能分解
5)、设计软件
6)、设计数据库
7）制定测试计划
8)、书写文档：系统说明、用户手册、测试计划、详细的实现计划、数据库设计结果；
9)、审查和复审
二、设计原理分析（模块化，在模块化程序设计中，按功能划分模块的原则是，模块化和软件成本关系）：
模块具有输入和输出(参数传递)、功能、内部数据结构(局部变量)和程序代码**四个特性**
1、**模块化:**就是把程序划分成独立命名且可独立访问的模块,每个模块完成一个子功能,把这些模块集成起来构成一个整体,可以完成指定的功能满足用户的需求.
2、**模块化的根据**:把复杂的问题分解成许多容易解决的小问题,原来的问题也就容易解决了.模块化和软件成本关系:根据总成本曲线,每个程序都相应地有一个最适当的模块数目 M,,使得系统的开发成本最小.
3、**模块设计的准则：**
(1) 改进软件结构, 提高模块独立性:在对初步模块进行合并、分解和移动的分析、精化过程中力求提高模块的内聚，降低藕合。
(2) 模块大小要适中:大约50行语句的代码，过大的模块应分解以提高理解性和可维护性;过小的模块,合并到上级模块中。
(3) 软件结构图的深度、宽度、扇入和扇出要适当。一般模块的调用个数不要超过5个。
(4) 尽量降低模块接口的复杂程度;
(5) 设计单入口、单出口的模块。
(6) 模块的作用域应在控制域之内。
4、**抽象的概念:**抽出事务的本质特性而暂时不考虑它们的细节.
6、 **信息隐蔽：**模块中所包括的信息不允许其它不需这些信息的模块调用
信息局部化：是把一些关系密切的软件元素物理地放得彼此靠近
7、 什么是模块独立性？
答：模块独立性概括了把软件划分为模块时要遵守的准则，也是判断模块构造是不是合理的标准。
8、 **模块独立性：**是软件系统中每个模块只涉及软件要求的具体子功能，而和软件系统中的其它的模块接口是简单的。模块独立的概念是模块化、抽象、信息隐蔽和局部化概念的

直接结果。

9、 为什么模块的独立性很重要？

答：1）有效的模块化的软件比较容易开发出来

2)独立的模块比较容易测试和维护。总之，模块独立是好设计的关键，而设计又是决定软件质量的关键环节。

10、启发规则：

1) 改进软件结构提高模块独立性

2) 模块规模应该适中

3) 深度、宽度、扇出、、和扇入都应适当

深度表示软件结构中控制的层数，它往往能粗略地标志一个系统的大小和复杂程度。

宽度是软件结构内同一个层次上的模块总数的最大值；一般来说，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。

一个模块的扇入是指直接调用该模块的上级模块的个数。

一个模块的扇出是指该模块直接调用的下级模块的个数。

设计原则：低扇出、高扇入。

4) 模块的作用域应该在控制域内

5) 力争降低模块接口的复杂程度

6) 设计单入口和单出口的模块

7) 模块功能应该可以预测

三、概要设计的方法：

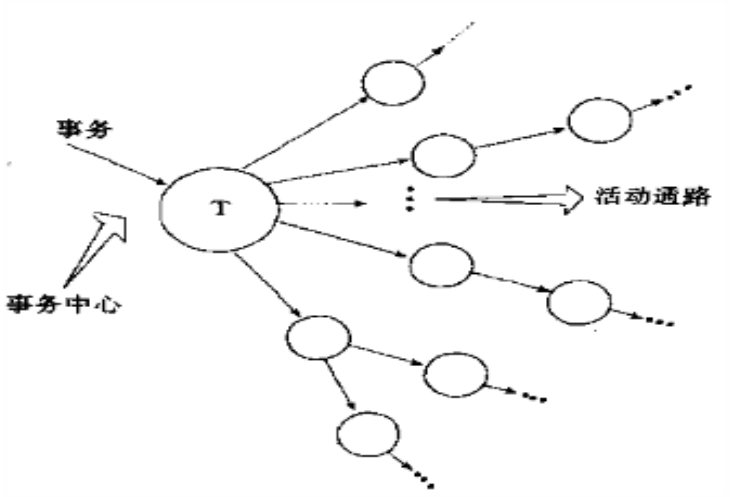
1、面向数据流的设计方法把信息流映射成软件结构，信息流的类型决定了映射的方法。面向数据流的设计要解决的任务，就是上述需求分析的基础上，将 DFD 图映射为软件系统的结构。

2、**数据流图的类型：**交换型结构和事务型结构

交换型结构：由3部分组成，传入路径，变换中心，输出路径

系统的传入流经过变换中心的处理，变换为系统的传出流。

事务型结构：有至少一条接受路径，一个事务中心与若干条动作路径组成。当外部信息沿着接受路径进入系统后，经过事务中心获得某个特定值，就能据此启动某一条动作路径的操作。



四、结构化设计

1、结构化设计方法：是一种面向数据流的设计方法，中心任务就是把用 DFD 图表示的系统分析模型转换为软件结构的设计模型，确定软件的体系结构域接口。

2、结构化方法的步骤：

1）复审 DFD 图，必要时刻再次进行修改或细化：

2）鉴别 DFD 图所表示的软件系统的结构特征，确定它所代表的软件结构是属于变换型还是事务型。

3）按照 SD 方法规定的一组规则，吧 DFD 图转换为初始的 SC 图。

变换型 DFD 图 —**变换映射**→ 初始 SC 图

事务型 DFD 图 —**事务映射**→ 初始 SC 图

3、结构设计的优化规则：

1）对模块分割、合并和变动调用关系的指导规则：以提高模块独立性为首要标准，除此之外，适当考虑模块的大小。

2）保持高扇 / 入低扇出原则

3）作用域 / 控制域规则：

作用域不要超出控制域的范围；

软件系统的判定，其位置离受它控制的模块越近越好。

总体设计阶段的基本目的是比较抽象概括的方式确定系统如何完成预定的任务，也就是说，应该确定系统的物理配置方案，并且进而确定组成系统的每个程序的结构。因此，总体设计阶段主要由两个小阶段组成。首先需要进行系统设计，从数据流图出发设想完成系统功能的若干种合理的物理方案，分析员应该仔细分析比较这些方案，并且和用户共同选定一个最佳方案。然后进行软件结构设计，确定软件由哪些模块组成以及这些模块之间的动态调用关系。层次图和结构

图是绘画软件结构的常用工具。

第六章

63.**结构程序设计的经典定义：**如果一个程序的代码仅仅通过顺序，选择和循环这 3 种基本结构进行连接，并且每个代码块只有一个入口和一个出口，则称这个程序是结构化的。

64.**系统响应时间**指从用户完成某个控制动作（例如，按回车键或单击鼠标），到软件给出预期的响应（输出信息或动作）之间的这段时间。

65.系统响应时间有两个重要的属性：**长度和易变性**。

66.**易变性**指系统响应时间相对于平均响应时间的偏差。

67.一般交互指南涉及信息显示，数据输入和系统整体控制。

68.**过程涉及的工具**:程序流程图，盒图，PAD 图，判定表，判定树

69.PDL 作为一种设计工具有如下一些**特点**（要求看懂伪码）

衡量程序的质量不仅要看它的逻辑是否正确，性能是否满足要求，更主要的是要看它是否容易阅读和理解。

小结：

1、详细设计阶段的关键任务是确定怎样具体的实现用户需要的软件系统，也就是要设计出程序的“蓝图”。除了应该保证软件的可靠性之外，使将来编写出的程序可读性好，容易理解，容易测试，容易修改和维护，是详细设计阶段最重要的目标。结构程序设计技术是实现上述目标的基本保证，是进行详细设计的逻辑基础。

2、在设计人机界面的过程中，必须充分重视并认真处理好系统响应时间，用户帮助设施，出错信息处理和命令交互这 4 个设计问题。

六、 详细设计

重点掌握的内容：详细设计的任务和方法

一般掌握的内容：详细设计的原则和详细设计的规格与评审

考核知识点：

一、详细设计

1、**详细设计目的：**对模块的算法设计和数据结构设计（设计出的处理过程应该尽可能简明易懂）。

2、**详细设计的任务：**详细设计就是在概要设计的结果的基础上，考虑“怎样实现”这个软件系统，直到对系统中个的每个模块给出足够详细的过程描述，主要任务如下：

编写软件的“详细设计说明书”.软件人员要完成的工作:

(1) 为每一个模块确定采用的算法, 选择某种适当的工具表达算法的过程,写出模块的详细过程描述.

(2) 确定每一模块使用的数据结构.

(3) 确定模块结构的细节,包括对系统外部的接口和用户界面,对系统内部其它模块的接口,以及关于模块输入数据、输出数据及局部数据的全部细节.

(4) 为每一个模块设计出一组测试用例,以便在编码阶段对模块代码(即程序)进行预

定的测试.模块的测试用例是软件测试计划的重要组成部分，通常包括输入数据，期望输出等内容。

3、**详细设计的原则：**过程描述是否易于理解、复审和维护，进而过程描述能够自然装换成代码，并保证详细设计与代码完全一致。

4、详细设计的描述工具应具备什么功能？

答：无论哪类描述工具不仅要具有描述设计过程，如控制流程、处理功能、数据组织及其它方面的细节的能力,而且在编码阶段能够直接将它翻译为用程序设计语言书写的源程序。

二、结构化程序设计

4、结构程序设计：如果一个程序的代码块仅仅通过顺序、选择和循环这3种进本控制结构进行连接，并且每个代码块只有一个入口和一个出口，则称这个程序是结构化的。

5.**结构化程序设计的基本原则：**在详细设计中所有模块都使用单入口、单出口的顺序、选择、循环三种基本控制结构.

四、过程设计

1、描述程序处理过程的工具称为过程设计工具，它们可以分为**图形、表格和语言**3类。

2、**详细设计的方法：**程序流程图、N-S 图、PAD 图

程序流程图:程序流程图又称之为程序框图,它是软件开发者最熟悉的一种算法表达工具。它独立于任何一种程序设计语言，比较直观和清晰地描述过程的控制流程，易于学习掌握。在流程图中只能使用下述的五种基本控制结构。1）顺序型、2）选择型、3）while 型　　4）Until 型循环　　5）多情况型选择

N-S 图：规定了五种基本图形构件：1）顺序型、2）选择型、3）while 重复型　　4）Until 重复型循环　　5）多分支选择型

PAD 图：它是用结构化程序设计思想表现程序逻辑结构的图形工具。PAD 图也设置了五种基本控制结构的图示，并允许递归使用。

判定表：能够清晰地表示复杂的条件组合与应该做的动作之间的关系。

判定树的优点：它的形式简单到不需要任何说明，一眼就可以看出其含义，因此易于掌握和使用。

过程设计语言（PDL）也称伪码，它是正文形式表示数据和处理过程的设计工具。

4.比较面向数据流和面向数据结构两类设计方法的异同?

相同点：

(1)　遵守结构程序设计“由顶向下”逐步细化的原则,并以其为共同的基础；

(2)　均服从“程序结构必须适应问题结构”的基本原则,各自拥有从问题结构(包括数据结构)导出程序结构的一组映射规则。

不同点：

(1)　面向数据流的设计以数据流图为基础，在分析阶段用 DFD 表示软件的逻辑模型，在设计阶段按数据流类型，将数据流图转换为软件结构。面向数据结构的设计以数据结构为基础，从问题的数据结构出发导出它的程序结构。

(2)　面向数据流的设计的最终目标是软件的最终 SC 图,面向数据结构的设计的最终目标是程序的过程性描述。 过程设计的原则和方法：

1) 清晰第一的设计风格：大多数情况下，应该优先考虑程序的清晰度，把效率的考虑放在第二位。(除少数使用特别频繁，或者实时程序)

2) 结构化的控制结构：所有的模块都只使用单入口单出口的3种基本循环结构——顺序、选择、循环

3) Goto 语句不应滥用，但也不必完全禁止

4) 逐步细化实现方法。

五、 面向数据结构的设计方法（Jackson 方法和 Warnier 方法）

1、 **Jackson 设计方法：**Jackson 方法是最著名的面向数据结构的设计方法,而不是面向数据流的设计方法。它是以信息驱动的，是将信息转换成软件的程序结构

2、Jackson 方法的基本步骤是：

(1）分析并确定输入数据和输出数据的逻辑结果，并用 Jackson 图描绘这些数据结构.

（2）找出输入数据结构和输出数据结构中有对应关系的数据单元。

（3）从描绘数据结构的 Jackson 图导出描绘程序结构的 Jackson 图

（4）列出所有操作和条件（包括分支条件和循环结束条件），并且把他们分配到程序结构图的适当位置

（5）用伪代码表示程序

3、比较 Jackson 方法和 LCP 方法的异同?

答：Jackson 与 LCP 设计方法都是以数据结构为出发点，以程序的过程描述为最终目标，设计步骤基本相似。它们的主要差别是：

(1)使用不同的表达工具，其中 LCP 方法中的表达工具 Warnier 图比 Jackson 设计方法中的表达工具 Jackson 图有更大的通用性；

(2)Jackson 方法的步骤和指导原则有一定的灵活性，而 LCP 设计方法则更加严密。

第七章

70. 通常把编码和测试统称为**实现**。

1、所谓**编码**就是把软件设计结果翻译成用某种程序设计语言书写的程序。

71. 编码和单元测试属于软件生命周期的同一个阶段。

72. **编码的标准：**1. 系统用户的要求 2. 可以使用的编译程序 3. 可以得到的软件工具 4. 工程规模 5. 程序员的知识 6. 软件可移植性要求 7. 软件的应用领域

2、 **编码的任务？**

答：使用选定的程序设计语言，把模块的过程性描述翻译为用语言书写的源程序(源代码)。

73. 所谓**程序内部的文档**包括恰当的标识符、适当的注释和程序的视觉组织等。

74. 当多个变量名在一个语句中说明时，应该按字母顺序排列这些变量。

75. **效率主要指**处理机时间和存储器容量两个方面。

76. 在大型计算机中必须考虑操作系统页式调度的特点，一般说来，使用能保持功能域的结构化控制结构，是提高效率的好方法。

77. 二级存储器的输入输出应该以信息组为单位进行。

1、**软件测试定义：**为了发现程序中的错误而执行程序的过程

3、**软件测试的准则：**

1）所有测试都应该能够追溯到用户需求

2）应该远在测试开始之前就制定出测试计划

3）把 Pareto 原理应用到软件测试中

4）应该从“小规模”测试开始，并逐步进行“大规模”测试

5）穷举测试是不可能的。

6）为了达到最佳的测试效果，应该由独立的第三方从事测试工作。

4、**软件测试方法：**第一种方法是黑盒测试（功能测试），第二种方法是白盒测试（结构测试）

黑盒测试：如果已经知道了产品应该具有的功能，可以通过测试来检验是否每个功能都能正常使用。

白盒测试：如果知道产品内部工作过程，可以通过测试来检验产品内部动作是否按照规格说明书的规定正常进行。

5、**软件测试步骤：**1）单元测试(模块测试) 2）.子系统测试 3）. 系统测试 4）. 验收测试（确认测试）5.） 平行运行

6、**测试阶段的信息流：**1）软件配置，包括需求说明书、设计说明书和原程序清单

测试配置，包括测试计划和测试方案（输入数据、输出数据和检测功能）

78.**测试阶段的根本目标**是尽可能地发现并排除软件中潜藏的错误，最终把一个高质量的软件系统交给用户使用。

79.**软件测试的目标：**1.测试是为了发现程序中的错误而执行程序的过程 2.好的测试方案是发现了至今为止发现的错误的测试 3.成功的测试是发现了至今为止尚未发现的错误的测试。

3、对于软件测试而言，黑盒测试法把程序看作一个黑盒子，完全不考虑程序的内部结构和处理过程；白盒测试法与黑盒测试法相反，它的前提是可以把程序看出装在一个透明的白盒子里，测试者完全知道程序的结构和处理算法。

4、在**单元测试期间着重从5个方面对模块进行测试：**

1）模块接口：参数的数目、次序、属性或单位系统与变元是否一致；是否修改了只作输入用的变元；全局变量的定义和用法在各个模块中是否一致。

2）局部数据结构

3）重要执行通路

4）出错处理通路

5）边界条件：边界测试是单元测试中最后的也可能是最重要的任务。

5、**自顶向下测试方法的主要优点**是不需要测试驱动程序，能够在测试阶段的早期实现并验证系统的主要功能，而且能在早期发现上层模块的接口错误。自顶向下测试方法的主要缺点是需要存根测试，可能遇到与此相联系的测试困难，低层关键模块中的错误发现较晚，而且用这种方法在早期不能充分展开人力。

6、Alpha 测试由用户在开发者的场所进行，并且在开发者对用户的“指导”下进行测试。开发者负责记录发现的错误和使用中遇到的问题。 Beta 测试由软件的最终用户们在一个或多个客户场所进行。

7、**软件可靠性的定义：**软件可靠性是程序在给定的时间间隔内，按照规格说明书的规定成功地运行的概率。 软件可用性的定义：软件可用性是程序在给定的时间点，按照规格说明书的规定，成功地运行的概率。

一、编码：

3、**程序设计的特点：**程序设计语言是人鱼计算机交流的媒介。软件工程师应该了解程序设计语言各方面的特点，以及这些特点对软件质量的影响，以便在需要为一个特定的开发项目选择语言时，能作出合理的技术抉择。**其特点表现为九方面：**

(1) 名字说明：程序中使用对象的名字，能为编译程序所检查和识别；

(2) 类型说明：定义对象的类型，确定该对象的使用方式；

(3) 初始化：为变量提供适当的初始值或由系统给变量赋一特殊的表明未初始化的值；

(4) 对象的局部性：程序中真正需要的那部分才能访问的对象；

(5)　程序模块：控制程序对象的名字；
(6)　循环控制结构：如 FOR 语句、WHILE-DO 语句、REPEAT-UNTIL 语句等；
(7)　分支控制结构：如 IF 语句、CASE 语句等；
(8)　异常处理：为程序运行过程中发生的错误和意外事件提供检测和处理上的帮助；
(9)　独立编译：能分别编译各个程序单元。

4、**编码风格：**标准，源程序代码的逻辑简明清晰，易读易懂①程序内部应该有很好的文档②数据说明应该易于理解便于查阅③语句构造应该尽可能简单直观④输入输出风格遵守人机界面设计准则，力求做到数据输入安全方便，信息输出清晰易懂⑤效率能满足用户需求即可，不要牺牲程序的清晰性和可读性来追求不必要的高效率

编码风格又称程序设计风格或编程风格，实际上指编程的原则。表现为五个方面：
源程序文档化：符号名的命名、程序的注释、标准的书写格式
数据说明：数据说明的次序应当规范化。使数据属性容易查找，也有利于测试，排错和维护
语句结构：语句结构力求简单、直接，不能为了片面追求效率而使语句复杂化，可以从以下几点注意：使用标准的控制结构、尽可能使用库函数、程序编写首先应该考虑清晰性、注意使用 goto 语句
输入/输出：输入/输出地方式和格式应当尽可能做到对用户友善，尽可能方便用户的使用。
效率： 程序效率：程序效率是指程序的执行速度及程序占用的存储空间。影响程序效率的因素是多方面的。

三、单元测试

1、**单元测试**（Unit testing ）也称为**模块测试或结构测试**，通常可放在编程阶段(实现阶段)，主要采用逻辑覆盖技术，由程序员对自己编写的模块自行测试，检查模块是否能实现了详细设计说明书中规定的功能和算法。
2、单元测试主要发现编程和详细设计中产生的错误。
3、测试一个模块时需要为该模块编写一个驱动模块和若干个桩（stub）模块。顶层模块测试时不需要驱动模块，底层模块测试时不需要桩模块。
4、在进行单元测试时，常用的方法是白盒测试(采用逻辑覆盖的测试技术)，辅之以黑盒测试。

四、非渐增式测试与渐增式测试有什么区别？渐增式测试如何组装模块？

非渐增式测试与渐增式测试的测试方法有以下**区别：**

㈠非渐增式测试方法把单元测试和集成测试分成两个不同的阶段，前一阶段完成模块的单元测试，后一阶段完成集成测试。而渐增式测试往往把单元测试与集成测试和在一起，同时完成。

㈡非渐增式需要更多的工作量，因为每个模块都需要驱动模块和桩模块，而渐增式利用已测试过的模块作为驱动模块或桩模块，因此工作量较少。

㈢渐增式可以较早的发现接口之间的错误，非渐增式最后组装是才发现。

㈣渐增式有利于排错，发生错误往往和最近加进来的模块有关，而非渐增式发现接口错误推迟到最后，很难判断是哪一部分接口出错。

㈤渐增式比较彻底，已测试的模块和新的模块再测试。

㈥渐增式占用的时间较多，但非渐增式须更多的驱动模块、桩模块也占用一些时间。

㈦非渐增式开始可并行测试所有模块，能充分利用人力，对测试大型软件很有意义。

渐增式测试有以下两种不同的组装模块的方法：

㈠自顶向下组合。该方法只需编写桩模块，其步骤是从顶层模块开始，沿被测程序的软件结构图的控制路径逐步向下测试，从而把各个模块都结合起来，它又有两种组合策略：①深度有先策略：先从软件结构中选择一条主控制路径，把该路径上的模块一个个结合进来进

行测试，以便完成一个特定的子功能，接着再结合其它需要优先考虑的路径。②宽度有先策略：逐层结合直接下属的所有模块。

㈡自低向上结合。该方法仅需编写驱动模块。其步骤为：①把底层模块组合成实现一个个特定子功能的族。②为每一个族编写一个驱动模块，以协调测试用例的输入和测试结果的输出。③对模块族进行测试。④按软件结构图依次向上扩展，用实际模块替换驱动模块，形成一个个更大的族。⑤重复②至④步，直至软件系统全部测试完毕。

五、集成测试：

1、**集成测试**（integration testing）也称为组装测试，在单元测试的基础之上，把所有的模块组装成一个系统进行测试。主要测试设计阶段产生的错误，集成测试计划应该在概要设计阶段制定。

2、**非渐增式集成测试**

首先将每个模块分别进行单元测试，再把所有的模块组装成一个完整的系统进行测试。目前在进行集成测试时已普遍采用渐增式集成。

3、**渐增式集成测试**

又可以分为自顶向下集成和自底向上集成。自顶向下集成先测试上层模块，再测试下层模块，由于测试下层模块时上层模块已经测试过，所以不必要另外编写驱动模块。自底向上集成，先测试下层模块，再测试上层模块。

顶层模块测试时不需要驱动模块，底层模块测试时不需要桩模块。

软件的集成测试最好由不属于该软件开发组的软件设计人员承担，以提高集成测试的效果。

六、**确认测试（验收测试）**

1） 确认测试的目标是验证软件的有效性

2） 在系统验收测试中，验证测试是在模拟的环境中进行强度测试的基础上进行，主要依据软件需求说明书检测软件的功能，性能及其他特征是否与用户的要求一致，而确认测试是在一个实际环境中使用真实数据运行系统。

3） 确认测试计划应该在需求分析阶段制定。

4） Alpha 测试

由用户在开发者的场所进行，并且在开发者的指导下进行测试。开发者负责纪录发现的错误和使用中遇到的问题，也就是说 Alpha 测试是在受控的环境中进行的。

5） Beta 测试是在一个或多个用户的现场由该软件的最终用户实施的，开发者通常不在现场，用户负责记录发现的错误和使用中遇到的问题并把这些问题报告给开发者。也就是说，Beta 测试是在受控的环境中进行的。经过确认测试之后的软件通常就可以交付使用了。

结构化的软件测试：软件测试在程序员对每一个模块的编码之后先做程序测试，再做单元测试，然后再进行集成（综合或组装）测试，系统测试，验收（确认）测试，平行测试，人工测试，其中单元测试的一部分已在编码阶段就开始了。

测试：就是用已知的输入在已知环境中动态地执行系统（或系统的“部件”）。如果测试结果和预期结果不一致，则很可能是发现了系统中的错误。

软件测试：软件测试是对软件计划、软件设计、软件编码进行查错和纠错的活动（包括代码执行活动与人工活动）。

程序测试：是对编码阶段的语法错、语义错、运行错进行查找的代码执行活动。找出编码中错误的代码执行活动称程序测试。纠正编码中的错误的执行活动称程序调试。程序测试的目的是查找编码错与纠正编码错，保证算法的正确实现。

测试的原则：

（1）测试前要认定被测试软件有错，不要认为软件没有错。

（2）要预先确定被测试软件的测试结果。

（3）要尽量避免测试自己编写的程序。

（4）测试要兼顾合理输入与不合理输入数据。

（5）测试要以软件需求规格说明书为标准。

（6）要明确找到的新错与已找到的旧错成正比。

（7）测试是相对的，不能穷尽所有的测试，要据人力物力安排测试，并选择好测试用例与测试方法。

（8）测试用例留作测试报告与以后的反复测试用，重新验证纠错的程序是否有错。

测试方法:按照测试过程是否在实际应用环境中来分，有静态分析与动态测试。测试方法有分析方法（包括静态分析法与白盒法）与非分析方法（称黑盒法）。

静态分析技术：不执行被测软件，可对需求分析说明书、软件设计说明书、源程序做结构检查、流程分析、符号执行来找出软件错误。

动态测试技术：当把程序作为一个函数，输入的全体称为函数的定义域，输出的全体称为函数的值域，函数则描述了输入的定义域与输出值域的关系。这样动态测试的算法可归纳为：

①选取定义域中的有效值，或定义域外无效值。

②对已选取值决定预期的结果。

③用选取值执行程序。

④观察程序行为，记录执行结果。

⑤将④的结果与②的结果相比较，不吻合则程序有错。

动态测试既可以采用白盒法对模块进行逻辑结构的测试，又可以用黑盒法做功能结构的测试、接口的测试，都是以执行程序并分析执行结果来查错的。

白盒法:是通过分析程序内部的逻辑与执行路线来设计测试用例，进行测试的方法，白盒法也称逻辑驱动方法。白盒法的具体设计程序测试用例的方法有：语句覆盖、分支（判定）覆盖、条件覆盖、路径覆盖（或条件组合覆盖），主要目的是提高测试的覆盖率。

黑盒法:是功能驱动方法，仅根据 I/O 数据条件来设计测试用例，而不管程序的内部结构与路径如何。黑盒法的具体设计程序测试用例的方法有：等价类划分法，边界值分析法，错误推测法，主要目的是设法以最少测试数据子集来尽可能多的测试软件程序的错误。

软件测试的步骤：

单元测试：单元测试也称模块测试、逻辑测试、结构测试，测试的方法一般采用白盒法，以路径覆盖为最佳测试准则。

集成测试：单元测试之后便进入组装测试。尽管模拟了驱动模块和存根模块进行单元测试，由于测试不能穷尽，单元测试又会引入新错误，单元测试后肯定会有隐藏错误，组装不可能一次成功，必须经测试后才能成功。集成测试分为增式组装测试和非增式组装测试，所谓非增式组装，按照结构图一次性将各单元模块组装起来。所谓增式组装是指按照结构图自顶向下或自底向上逐渐安装。

确认测试：确认测试也称合格测试或称验收测试。组装后已成为完整的软件包，消除了接口的错误。确认测试主要由使用用户参加测试，检验软件规格说明的技术标准的符合程度，是保证软件质量的最后关键环节。

系统测试：一般的系统测试除了确认测试外还要做如下几个方面的系统测试

1） 恢复测试：通过系统的修复能力，检测重新初始化，数据恢复，重新启动，检验点设置机构是否正确，以及人工干预的平均恢复时间是否在允许范围内。

2） 安全测试：设计测试用例，突破软件安全保护的机构安全保密措施，检验系统是否安全保密的漏洞。

3） 强度测试：

性能测试：设计测试用例并记录软件运行性能，与性能要求比较，看是否达到性能要求规格。这项测试常常与强度测试项结合进行。

小结：

1、大型软件的测试应该分阶段地进行，通常至少分为单元测试、集成测试盒验收 3 个基本阶段。

2、设计白盒测试方案的技术主要有，逻辑覆盖和控制结构测试；设计黑盒测试方案的技术主要有，等价划分、边界值分析和错误推测。

第八章

81.**软件工程的主要目标**就是要提高软件的可维护性，减少软件维护所需要的工作量，降低软件系统的总成本。

82.所谓**软件维护**就是在软件应经交付使用之后，为了改正错误或满足新的需要而修改软件的过程。

83.**四种维护的定义：**1.改正性维护 2.适应性维护 3.完善性维护 4.预防性维护。P189

84.用于维护工作的劳动可以分成**生产性活动和非生产性活动**。

85.每个维护要求都通过维护管理员转交给熟悉该产品的系统管理员去评价。

86.适应性维护和完善性维护的要求沿着相同的事件流通路前进。

87.当发生恶性的软件问题时，就出现所谓的“救火”维护要求。

88.**.评价维护活动：**1.每次程序运行平均失效的次数 2、用于每一类维护活动的总人时数。3.平均每个程序、每种语言、每种维护类型所作的程序变动数 4.维护过程中增加或删除一种源语句平均花费的人时数 5.维护每种语言平均花费的人时数 6.一张维护要求表的平均周转时间 7.不同维护类型所占的百分比。

89.**可以把软件的可维护性定性的定义为：**维护人员理解，改正，改动或改进这个软件的难易程度。

90.**决定软件可维护性的因素：**1.可理解性 可测试性 可修改性 可移植性 可重用性

91.所谓**重用**是指同一事物不做修改或稍加改动就在不同环境中多次重复使用。

92.软件系统的文档可以分为用户文档和系统文档两类。用户文档主要描述系统功能和使用方法，并不关心这些功能是怎样实现的；系统文档描述系统发设计、实现和测试等各方面的内容预防性维护方法是由 Miller 提出来的，他把这种方法定义为“把今天的方法学应用到昨天的系统上，以支持明天的需求。”

2、**软件维护分为一下几类：**

1）改正性维护：纠正开发期间未发现的遗留错误，即在程序使用期间发现的程序错误进行诊断和改正的过程；

2）适应性维护：是为了和变化的环境适当地配置而进行的修改软件的活动，是既必要又经常的维护活动。是软件适应新的运行环境而进行的工作；

3）完善性维护：满足用户在使用过程中提出增加新的功能或修改已有功能，以满足用户日益增长的需要而进行的工作； 该维护活动通常占软件维护工作的大部分；

4）**预防性维护：**为了改善未来的可维护性或可靠性而修改软件的工作。 用于维护工作的劳动可以分成生产性活动（如：分析评价，修改设计和编写程序代码）和非生产性活动（如：理解程序代码的功能，解释数据结构、接口特点和性能限度）。

3、软件维护过程本质上是修改和压缩了的软件定义和开发过程，而且事实上远在提出一项维护要求之前，与软件维护有关的工作已经开始了。

4、**软件维护过程：**1）维护组织；2）维护报告；3）维护的事务流（维护事务流中最后一件事件是复审）。P193图8-1 ；4）保存维护记录；5）评价维护活动

5、**决定软件的可维护性的因素：**1）可理解性；2）可测试性；3)可修改性；4）可移植性；5）可重用性；

6、**软件再工程过程：**1）存库目录分析；2）文档重构；3）逆向工程；4）代码重构；5）数据重构；6）正向工程

7、为什么软件难维护？

答：因为结构化维护与非结构化维护差别巨大;维护的代价高昂;维护的问题很多.

第九章　面向对象方法学引论

1、**为什么提出面向对象的方法：**随着大型软件系统的出现，在中小型软件系统取得成功的传统的软件工程方法面临巨大的危机：

1、软件生产率无法满足市场需求

2、软件复用率不高

3、软件维护困难

4、软件往往不能真正满足用户需求

2、**面向对象的基本概念:**对象是由描述该对象属性的数据以及可以对这些数据施加的所有操作封装在一起构成的统一体. 一、面向对象方法的出发点和基本原则：是尽可能模拟人类习惯的思维方式，使开发软件的方法与过程尽可能接近人类认识世界解决问题的方法与过程，也就是使描述问题的问题空间（也称问题域）与现实解法的解空间（也称求解域）在结构上尽可能一致。

二、面向对象方法具有以下四点要素：

1)、认为客观世界是有各种对象组成的，任何事物都是对象，复杂的对象可以有比较简单的对象以某种方式组合而成。按照这种观点，可以认为整个世界就是一个最复杂的对象。

2)、把所有对象方法都划分成各种对象类，每个对象都定义了一组数据和一组方法。数据用于表示对象的静态属性，是对象的状态信息。因此，每当建立该对象类的一个新实例时，就按照类中队数据的定义为这个新对象生成一组专用的数据，以便描述该对象独特的属性值。

3)、按照父类（或称为派生类）与父类（或称为基类）的关系，把若干个对象类组成一个层次结构的系统（也称为类等级）.

4)、对象彼此之间仅能通过传递信息相互联系。

三、面向对象方法学的优点：

1）与人类习惯的思维方法一致

2）稳定性好

3）可重用性好

4）交易开发大型软件产品

5）可维护性好

1、由于以下因素的存在，使得用面向对象方法所开发的软件可维护性好：

1)、面向对象的软件稳定性比较好

2)、面向对象的软件比较容易修改

3)、面向对象的软件比较容易理解

4)、易于测试和调试

四、概念：

1)、对象：它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，这个封装体有可以唯一地标识它的名字，而且向外界提供一组服务（即公有的操作）。

对象的特点：以数据为中心；对象是主动的；实现了数据封装；本质上具有并行性；模块独立性好

2）类：用于表示某些对象的共同特征（属性和操作），对象是类的实例。

（类是支持继承的抽象数据类型）

3)实例：就是由某个特定的类所描述的一个具体的对象。

4）消息：消息传递时对象与外界相互关系的唯一途径。对象可以向其它对象发送消息以请求服务，也可以响应其他对象传来的消息，完成自身固有的某些操作，从而服务于其他对象。

一个消息有3部分组成：　接收消息的对象；消息选择符（也称消息名）；零个或多个变元。

5)方法：方法就是对象所能执行的操作，也就是类中所定义的服务。方法描述了对象执行操作的算法，响应消息的方法。

6）属性：属性就是类中定义的数据，它是对客观世界实体所具有的性质的抽象。

7）封装：封装也就是信息隐藏，通过封装对外界隐藏了对象的实现细节。

对象具有封装性的条件：有一个清晰的边界；有确定的接口（即协议）；受保护的内部实现。

8）继承：是现实世界中遗传关系的直接模拟。可用来表示类之间的内在联系以及对属性和操作的共享。子类可以沿用父类的某些特征，同时子类也可以具有自己独立的属性和特征。

9）多态性：

10）重载：

函数重载是指在同一作用域内的若干个参数特征不同的函数可以使用相同的函数名字；

运算符重载是指同一个运算符可以施加不同类型的操作数上面。

五、 面向对象建模：

1、 模型定义:模型，就是为了理解事物而对事物作出的一种抽象，是对事物的一种无歧义的书面描述。

2、 对象模型（描述系统数据结构的对象模型）：**对象模型表示静态的、结构化的系统的“数据”性质。它是对模拟客观世界实体的对象以及对象彼此间的关系的映射，描述了系统的静态结构。**

3、 **通常，使用 UML 提供的类图来建立对象模型。在 UML 中术语“类”的实际含义是，“一个类及属于该类的对象”。**

4、 **类图的基本符号**

1）：定义类：UML 中类的图形符号为长方形，用两条横线把长方形分成上、中、下3个区域（下面两个区域可省略），3个区域分别放类的名字、属性和服务，如图所示。

2）类名是一类对象的名字。为类命名时应该遵守以下几条准则：

(1) 使用标准术语。(2) 使用具有确切含义的名词。(3) 必要时用名词短语作名字。名字应该是富于描述性的、简洁的而且无二义性的。

3)、定义属性

UML 描述属性的语法格式如下：可见性 属性名：类型名=初值{性质串}

4）. 定义服务

服务也就是操作，UML 描述操作的语法格式如下：

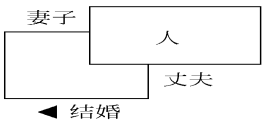
可见性 操作名（参数表）： 返回值类型{性质串}

5、 **表示关系的符号**

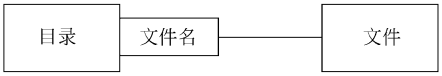
1）关联：表示两个类的对象之间存在某种语义上的联系

（1） 普通关联：

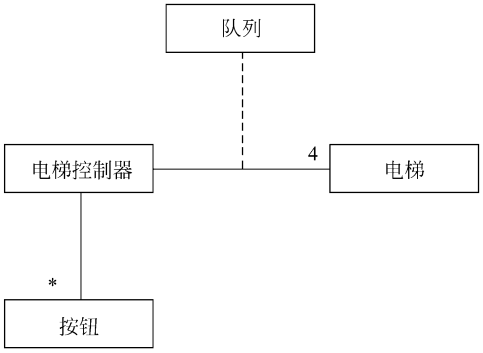
（2） 关联的角色：在任何关联中都会涉及到参与此关联的对象所扮演的角色（即起的作用），在某些情况下显式标明角色名有助于别人理解类图。如：



（3） 限定关联：限定关联通常用在一对多或多对多的关联关系中，可以把模型中的重数从一对多变成一对一，或从多对多简化成多对一。在类图中把限定词放在关联关系末端的一个小方框内。



（4） 关联类:如：



3） 聚集：

聚集也称为聚合，是关联的特例。聚集表示类与类之间的关系是整体与部分的关系。4). 泛化：UML 中的泛化关系就是通常所说的继承关系，它是通用元素和具体元素之间的一种分类关系。

5） 依赖和细化

（1） 依赖关系：

依赖关系描述两个模型元素（类、用例等）之间的语义连接关系： 其中一个模型元素是独立的，另一个模型元素不是独立的，它依赖于独立的模型元素，如果独立的模型元素改变了，将影响依赖于它的模型元素。

（2） 细化关系

当对同一个事物在不同抽象层次上描述时，这些描述之间具有细化关系。

重点：对象模型实现 P217，要求会作图。 1、动态模型（描述系统控制结构的动态模型）：它是基于事件共享而相互关联的一组状态图的集合（联系 P65页，状态转换图）。

2、 功能模型（描述系统功能的功能模型）：**功能模型表示变化的系统的“功能”性质，它指明了系统应该“做什么”，因此更直接地反映了用户对目标系统的需求。通常，功能模型由一组数据流图组成。**

重点：功能模型实现 P225，要求会作图。例子：自动取款机

3、面向对象建模的三种模型之间的关系？

（P228）答：在面向对象方法学中，对象模型是最基本最重要的，它为其他两种模型奠定基。

1）针对每个类建立的动态模型，描述了类实例的生命周期或运行周期。

2）状态转换驱使行为发生，这些行为在数据流图中被映射成处理，在用例图中被映射成用例，它们同时与类图中的服务相对应。

3）功能模型中的处理（或用例）对应于对象模型中的类所提供的服务。通常，复杂的处理（或用例）对应于复杂对象提供的服务，简单的处理（或用例）对应于更基本的对象提供的服务。有时一个处理（或用例）对应多个服务，也有一个服务对应多个处理（或用例）。

4）数据流图中的数据存储，以及数据的源点/终点，通常是对象模型中的对象。

5）数据流图中的数据流，往往是对象模型中对象的属性值，也可能是整个对象。

6）用例中的行为者，可能是对象模型中的对象。

7）功能模型中的处理（或用例）可能产生动态模型中的事件。

8）对象模型描述了数据流图中的数据流、数据存储以及数据源点/终点的结构。

总之，功能模型指明了系统应该”做什么”;动态模型明确规定了什么时候(即在何种状态下接受了什么事件的触发)做;对象模型则定义了做事情的实体.

4、用例图：一幅用例图包含的模型元素有系统、行为者、用例及用例之间的关系。

1） 用例之间的关系：

（1） 扩展关系

向一个用例中添加一些动作后构成了另一个用例，这两个用例之间的关系就是扩展关系，后者继承前者的一些行为，通常把后者称为扩展用例。

（2） 使用关系

当一个用例使用另一个用例时，这两个用例之间就构成了使用关系。

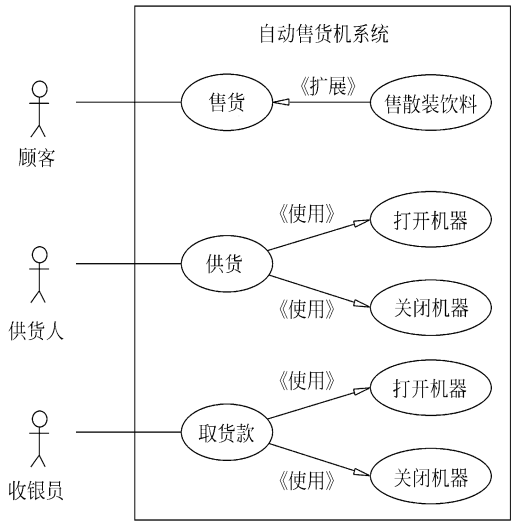


图 9.18 含扩展和使用关系的用例图　第十章：面向对象分析。

一、开发软件，分析的过程都是系统需求提取的过程，它要求系统分析员通过与用户及相关领域的专家交流，充分理解用户的需求和相关领域的背景知识，最终把这种理解制作成文档资料的过程，但由于问题的复杂性，该理解过程并不能达到理想的成程度，因此需要进一步去验证说明规格文档的正确性、完整性、有效性。

1、分析工作主要内容： 理解、表达、验证（其中理解和验证的过程是反复进行的，有事还需要原型系统作为辅助工具）。

2、分析过程得出的结果就是软件需求规格说明文档，相对于面向对象，由对象模型、动态模型、功能模型组成。

二、面向对象分析的基本过程

１、面向对象分析，就是抽取和整理用户需求并建立问题域精确模型的过程。

1) 基本过程流程:

（１）从分析陈述用户需求的文件开始（可由用户单方，也可由系统人员与用户配合写出需求陈述）。

（２）需求陈述通常是不完整、不准确的、还是非正式的。因此，需求陈述不是一成不变， 应通过分析，补充和改正其内容。

（３）系统人员深入分析理解用户的需求，抽象出系统的本质属性，并用模型准确表示出来。

（４）建模过程中，分析员应该认真向领域专家学习，还应仔细研究以前针对相同的或类似问题域进行面向对象分析得出的结果。

2).三个子模型与五个层次

a.三个子模型：

⑴ 静态结构（对象模型）：被包含于下面两种模型中。
⑵ 交互次序（动态模型）：适用于问题涉及交互作用和时序时（如用户界面及过程控制）。
⑶ 数据变换（功能模型）：用于解决运算量大的问题（如高级语言的编译）。

b. 五个层次：

⑴ 主题层
⑵ 类与对象层
⑶ 结构层
⑷ 属性层
⑸ 服务层

与之对应的面向对象分析过程中建立的对象模型为：

⑴ 找出类与对象
⑵ 识别结构
⑶ 识别主题
⑷ 定义属性
⑸ 定义服务

综合以上知，面向对象分析可按下列顺序进行：

⑴ 寻找类与对象
⑵ 识别结构
⑶ 识别主题
⑷ 定义属性
⑸ 建立动态模型
⑹ 建立功能模型
⑺ 建立服务

2、 需求陈述

其内容包括：

⑴ 问题范围
⑵ 功能需求
⑶ 性能需求
⑷ 应用环境
⑸ 假设条件

三、建立对象模型

六、面向对象分析的过程，首要的工作是建立问题域的对象模型，然后再建立另外两个

模型。

七、建立对象模型时的主要信息来源：

⑴ 需求陈述
⑵ 应用领域的专业知识以及关于客观世界的常识

3、建立对象模型工作步骤：

⑴ 确定对象类和关联，对于大型复杂问题还要进一步划分
⑵ 给类和关联增添属性，进一步描述他们
⑶ 利用适当的继承关系进一步合并和组织类

1)、确定类与对象

（1）找出候选对象与类

对象定义：对象是对问题域中有意义的事物的抽象，可是物理实体，也可是抽象概念。

其中，大多数客观事物包括：

a. 可感知的事物
b.人或组织的角色
c.应该记忆的事物
d.两个或是两个以上对象的相互作用，通常具有交易或是接触的性质
e.需要说明的概念

2)、筛选出正确的类和文件

筛选时删除不正确或不必要的类和对象时依据标准：

⑴ 冗余
⑵ 属性
⑶ 无关
⑷ 笼统
⑸ 操作
⑹ 实现

3)、确定关联

（1）初步确定关联

a.直接提取动词短语得出的关联
b.需求陈述中隐含的关联
c.根据问题域知识得出的关联

（2）筛选

筛选时依据的标准删除候选的关联：

a.已删去的类之间的关联
b.与问题无关的或应在实现阶段考虑的关联
c.瞬时事件
d.三元关联
e.派生关联

（3）进一步完善改进遵循的方面：

a.正名
b.分解
c.补充
d.标明重数

4、确定属性

属性定义：它是对象的性质，藉助于属性人们能对类与对象的结构有更深入的认识。

包含的过程：分析和选择

1）分析

分析过程应该首先找到最重要的属性，且属性跟问题域和目标的任务有关。

2）选择

认真考察初步确定下来的属性，删掉不正确的或不必要的属性，通常有下列几种常见的情况：

a.误把对象当作是属性
b.误把关联类的属性当作是一般对象的属性
c.把限定误当成属性
d.误把内部状态当成属性
e.过于细化
f.存在不一致的属性

5.确定属性

1）确定了类中应该定义的属性后就可以利用继承基质共享公共性质，而继承关系的建立实质其实就是知识的抽取过程，它反映出一定的深度的领域知识，因此应该有专家的配合。

2）建立继承关系的两种常见的方式

a自底向上：抽象出现有类的共同性质泛化出父类，该过程实质就是模拟了人类的思维过程。
b自顶向下：把现有类细化成更具体的子类，该过程模拟了人类演绎思维过程。

6.反复修改

四、建立动态模型

建立动态模型的步骤：

a.编写典型交互行为的脚本
b.从脚本中提取事件，确定触发每个事件的动作对象以及接受事件的目标对象
c.排列事件发生次序，确定每个对象可能有的状态及状态间的转换关系，并用状态图描绘
d.比较各个对象的状态图，检查它们之间的一致性，确保事件之间的匹配

1）编写脚本

(1)应注意的内容：

a.编写正常的脚本
b.考虑特殊情况
c.考虑特殊情况

(2)编写脚本目的：保证不遗漏重要交互步骤，有助于确保整个交互过程的正确性和清晰性。

(3)编写脚本的范围：没有固定的模式，可包含系统发生的全部事件，也可只有某个特殊的事件，总之，是由便写脚本的目的决定。

2）设想用户界面

（1）软件人员应该快速的建立一个用户界面的原型，以公用户试用并给出适当的评价，因为用户使用系统时首先接触到的就是用户界面，因此，界面很重要。

3）画事件跟踪图

目的：进一步明确事件与事件对象的关系

4）画状态图

作用：描绘事件与对象状态的关系。

5）审查动态模型

五、建立功能模型

1）画出基本模型图：基本模型图由若干个数据源点，终点，及一个处理框组成，该处理框代表了系统的加工，变化数据的整体功能。

2）画出功能数据图：把基本系统模型中单一的处理框分解成若干个子处理框，用以描述系统的加　工，交换数据的基本功能，就得到功能级数据流图。

3）描述处理框功能

注意：它的目的是描述每个处理框的功能，而不是实现功能的算法；它可是说明性的，也可是过程性的。

1）定义服务

1）常规行为

2）从事件导出的操作

3）与数据流图中处理框对应的操作

注意：数据流图中每个处理框都与一个对象（可以是对歌对象）上操作相对应。

4）利用继承减少冗余的操作

第十一章 面向对象设计

本章首先讲述为获得优秀设计结果应该遵守的准则,然后具体讲述面向对象设计的任务

和方法.

&11.1 面向对象设计的准则

1. 模块化
2. 抽象(过程、数据、规格说明、参数化抽象)
3. 信息隐藏（信息隐藏通过对象的封装性实现）
4. 弱耦合
5. 强内聚
6. 可重用

&11.2 启发规则

1. 设计结果应该清晰易懂(用词一致;使用已有的协议;减少消息模式的数目;避免模糊的定义)
2. 一般_特殊结构的深度应适当
3. 设计简单的类(避免包含过多的属性;有明确的定义;尽量简化对象之间的合作关系;不要提供太多服务)
4. 使用简单的协议(一般来说,消息中的参数不要超过 3 个)
5. 使用简单的服务
6. 把设计变动减至最小

&11.3 软件重用

11.3.1 概述

1.重用(知识重用;方法和标准重用;软件成分的重用)

2.软件成分的重用级别(代码重用;涉及结果重用;分析结果重用)

3.典型的可重用软件成分(项目计划;成本估计;体系结构;需求模型和规格说明;设计;源代码;用户文档和技术文档;用户界面;数据;测试用例)

11.3.2 类构件

1.可重用软件应具备的特点:模块独立性强,具有高度可塑性,接口清晰简明可靠.

2.类软件的重用:实力重用;继承重用;多态重用.

11.3.3 软件重用的效益

1.质量； 2.生产率； 3.成本:C=Cs-Cr-Cd

&11.4 系统分解

典型的面向对象设计模型,逻辑上由 4 部分组成:人机交互部分,问题域部分,任务管理部分,数据管理部分.

1. 子系统之间的两种交互方式:客户-供应商关系， 平等伙伴关系.
2. 组织系统的两种方案:层次组织， 块状组织.
3. 设计系统的拓扑结构

&11.5 设计问题域子系统

在面向对象设计过程中,可能面向对象分析所得出的问题域模型做的补充或修改:调整需求； 重用已有的类； 把问题域类组合在一起； 增添一般化类以建立协议； 调整继承层次.

&11.6 设计人机交互子系统

1. 分类用户(从不同角度进行分类)
2. 描述用户(了解每类用户的情况)
3. 设计命令层次
4. 设计人机交互类

&11.7 设计任务管理子系统

1. 分析并发性

2. 设计任务管理子系统:①确定事件驱动型子系统；②确定时钟驱动型任务；③确定优先任务；④确定关键人物；⑤确定协调任务；⑥尽量减少任务数；⑦确定资源需求.

&11.8 设计数据管理子系统

11.8.1 选择数据存储管理模式:文件管理系统，关系数据库管理系统，面向对象数据库管理系统.

11.8.2 设计数据管理子系统:

1.设计数据格式

⑴文件系统；

⑵关系数据库管理系统；

⑶面向对象数据库管理系统.

2.设计相应的服务

&11.9 设计类中的服务

11.9.1 确定类中应有的服务

11.9.2 设计实现服务的方法

1.设计实现服务的算法

考虑因素:⑴算法复杂度 ⑵容易理解与容易实现 ⑶易修改

2.选择数据结构

3.定义内部类和内部操作

&11.10 设计关联

关联的遍历

实现单向关联

实现双向关联

关联对象的实现

&11.11 设计优化

一. 确定优先级

二. 提高效率的几项技术

1. 增加冗余关联以提高访问效率
2. 调整查询次序
3. 保留派生属性

三. 调整继承关系

1. 抽象与具体
2. 为提高继承程度而修改类定义
3. 利用委托实现行为共享

第十二章 面向对象实现

一、 面向对象实现主要包括两项工作：把面向对象设计结果翻译成用某种程序语言书写的面向对象程序；测试并调试面向对象的程序。

二、 面向对象语言的优点

到底选择面向对象语言还是非面向对象语言，关键不在于语言功能强弱。从原理上说，使用任何一种通用语言都可以实现面向对象概念。当然，使用面向对象语言，实现面向对象概念，远比使用非面向对象语言方便，但是，方便性也并不是决定选择何种语言的关键因素。选择编程语言的关键因素，是语言的一致的表达能力、可重用性及可维护性。从面向对象观点来看，能够更完整、更准确地表达问题域语义

的面向对象语言的语法是非常重要的，因为这会带来下述几个重要优点：一致的表示方法、可重用性、可维护性。

三、 面向对象语言的技术特点

1） 支持类与对象概念的机制
2） 实现整体-部分（即聚集）结构的机制
3） 实现一般-特殊（即泛化）结构的机制
4） 实现属性和服务的机制
5） 类型检查：弱类型和强类型
6） 类库
7） 效率
8） 持久保存对象
9） 参数化类
10） 开发环境

四、 选择面向对象语言

开发人员在选择面向对象语言时，还应该着重考虑以下一些因素：

1）将来能否占主导地位
2）可重用性
3）类库和开发环境
4）其它因素：对用户学习面向对象分析、设计和编码技术所能提供的培训服务；在使用这个面向对象语言期间能提供的技术支持；能提供给开发人员使用的开发工具、开发平台、发行平台；对机器性能和内存的需求；集成已有软件的容易程度等。

五、 程序设计风格

良好的程序设计风格对面向对象实现来说尤为重要，不仅能明显减少维护和扩充的开销，而且有助于在新项目中重用已有的程序代码。

良好的面向对象程序设计风格，既包括传统的程序设计风格准则，也包括为适应面向对象方法所特有的概念而必须遵循一些新准则：

1、 提高可重用性：1）提高方法的内聚；2）减小方法的规模；3）保持方法的一致性；4）把策略与实现分开：策略方法应该检查系统运行状态，并处理出错情况，它们并不直接完成计算或实现复杂的算法；实现方法仅仅针对具体数据完成特定处理，通常用于实现复杂的算法。5）全面覆盖；6）尽量不使用全局信息；7）利用继承机制
2、 提高可扩充性：1）封装实现策略；2）不要用一个方法遍历多条关键链；3）避免使用多分支语句；4）精心确定公有方法
3、 提高健壮性：所谓健壮性就是在硬件故障、输入的数据无效或操作错误等意外环境下，系统能做出适当响应的程度。为提高健壮性应该遵守以下几条准则：1）预防用户的操作错误；2）检查参数的合法性；3）不要预先确定限制条件；4）先测试后优化

六、 测试策略：测试软件的经典策略是，从“小型测试”开始，逐步过渡到“大型测试”。用软件测试的专业术语描述，就是从单元测试开始，逐步进入集成测试，最后进行确认和系统测试。

1、 面向对象的单元测试：在测试面向对象软件时，不呢不过再孤立地测试单个操作，而应该把操作作为类的一部分来测试。
2、 面向对象的集成测试：分为基于线程的测试和基于使用的测试两种
3、 面向对象的确认测试：和传统的确认测试一样，面向对象的确认测试也集中检

查用户可见的动作和用户可识别的输出。对于面向对象的软件来说，主要还是根据动态模型和描述系统行为的脚本来设计确认测试用例。

七、 设计测试用例

1、 测试类的方法：对面向对象的软件来说，小型测试着重测试单个类和类中封装的方法。测试单个类的方法主要有随机测试、划分测试和基于故障的测试等3种。

2、 集成测试方法：1）对类测试；2）从动态模型导出测试用例

❤软件工程的基本原理：1.用分阶段的生命周期计划严格管理 2.坚持进行阶段评审 3.实行严格的产品控制 4.采用现代的程序设计技术 5.结果应能清楚的审查 6.开发小组的人员应该少而精 7.承认不断改进软件工程实践的必要性

❤软件工程方法学包含三个要素：方法，工具和过程其中方法是完成软件开发的各项任务的技术方法，回答“怎样做”的问题，工具是为运用方法而提供的自动的或半自动的软件工程支撑环境，过程是为了获得高质量的软件所需要完成的一系列任务的框架，它规定了完成各项任务的工作步骤

❤软件生命周期：1.问题定义（要解决的问题是什么）2.可行性研究（对于上一个阶段所确定的问题有行得通的解决办法吗）3.需求分析（为了解这个问题，目标系统必须做什么）4.总体设计（概括地说，应该怎样实现目标系统）5.详细设计（应该怎样具体地实现这个系统呢）6.编码和单元测试 7.综合测试 8.软件维护（改正性维护，适应性维护，完善性维护，预防性维护）

❤快速原型模型，增量模型（渐增模型），螺旋模型喷泉模型

❤可行性研究过程的步骤：1.复查系统规模和目标 2.研究目前正在使用的系统 3.导出新系统的高层逻辑模型 4.进一步定义问题 5.导出和评价供选择的解法 6.推荐行动方针 7.草拟开发计划 8.书写文档提交审查

❤数据流图和数据字典共同构成系统的逻辑模型

❤数据字典由下列四类元素的定义组成 1.数据流 2.数据流分量（即数据元素）3.数据存储 4.处理

❤定义数据的方法：1.顺序 2.选择 3.重复 4.可选

❤成本估计三种估算技术：1.代码行技术 2.任务分解技术 3.自动估计成本技术

❤成本/效益分析的方法：1.货币的时间价值 2.投资回收期 3.纯收入 4.投资回收率

❤软件需求正确性的验证①一致性②完整性③现实性④有效性

❤对系统的综合需求：1.功能需求 2.性能需求 3.可靠性和可用性需求 4.出错处理需求 5.接口需求 6.约束 7.逆向需求 8.将来可能提出的要求

❤与用户沟通获取需求的方法：访谈，面向数据流自顶向下求精，简易的应用规格说明书，快速建立软件原型

❤分析建模：数据模型——E-R 图 ，功能模型——数据流图，行为模型——状态转换图从哪些方面验证软件需求的正确性：1.一致性 2.完整性 3.现实性 4.有效性

❤验证软件需求的方法：1.验证需求的一致性 2.验证需求的现实性 3.验证需求的完整性和有效性

❤总体设计过程通常由两个主要阶段组成：系统设计阶段，确定系统的具体实现方案；结构设计阶段，确定软件结构
❤总体设计过程步骤：1.设想供选择的方案 2.选取合理的方案 3.推荐最佳方案 4.功能分解 5.设计软件结构 6.设计数据库 7.制定测试计划 8.书写文档（①系统说明②用户手册③测试计划④详细的实现计划⑤数据库设计结果）9.审查和复审
❤模块对立的概念是模块化，抽象，信息隐藏和局部化概念的直接结果
❤开发具有独立性功能而且和其他模块之间没有过多的相互作用的模块，就可以做到模块独立
❤耦合（低→高）是对一个软件结构内不同模块之间互连程度的度量。耦合强弱取决于模块间借口的复杂程度，进入或访问一个模块的点，以及通过接口的数据
❤耦合包括①数据耦合（低耦合）两个模块彼此间通过参数交换信息，而且交换的信息仅仅是数据②控制耦合：传递的信息中有控制信息（尽管有时这种控制信息以数据的形式出现），③公共环境耦合：两个或多个模块通过一个公共数据环境相互作用，④特征耦合（标记耦合）：只使用一部分所传递的参数。
❤启发规则：1.改进软件结构提高模块独立性 2.模块规模应该适中 3.深度、宽度、扇出和扇入都应适当 4.模块的作用域应该在控制域之内 5.力争降低模块接口的复杂程度 6.设计单入单出口的模块 7.模块功能应该可以预测
❤描绘软件结构的图形工具：层次图和 HIPO 图、结构图
❤最高程度的耦合是内容耦合。如果出现下列情况之一，两个模块间就发生了内容耦合：1.一个模块访问另一个模块的内部数据 2.一个模块不通过正常入口而转到另一个模块的内部 3.两个模块有一部分程序代码重叠（只可能出现在汇编程序中）4.一个模块有多个入口(这意味着一个模块有集中功能)
内聚（高→低）标志着一个模块各个元素彼此结合的紧密程度，它是信息隐藏和局部化概念的自然扩展。理想内聚的模块制作一件事情
❤耦合使用应采用的措施：尽量使用数据耦合，少使用控制耦合好特征耦合，限制公共环境耦合的范围，完全不用内容耦合
❤内聚和耦合是密切相关的，模块内的高内聚往往意味着模块间的松耦合
❤内聚包括①功能内聚：模块内所有元素属于同一整体且完成单一功能。②顺序内聚：模块内元素与功能密切相关且顺序执行处理。③通信内聚：模块中多有元素使用相同输入、产生相同输出。④过程内聚：模块内的处理元素是相关的且以特定次序执行。⑤时间内聚：模块包含的任务必须在同一段时间内执行。⑥逻辑内聚：模块完成的任务在逻辑上属相同或相似的一类。⑦偶然内聚：模块所完成的任务间关系松散或无关系
❤面向数据流的设计方法的目标是给出设计软件结构的一个系统化的途径
❤信息流有下述两种类型：1.变换流 2.事物流
❤需求分析的任务①确定对系统的综合要求②分析系统的数据要求③导出系统的逻辑模型④修正系统开发计划
❤软件工程的 7 条基本原理①用分阶段的生命周期计划严格管理②坚持进行阶段评审③实行严格的产品控制④此阿勇现代程序设计技术⑤结果应能清楚的审查⑥开发小组人员应该少而精⑦承认不断改进软件工程实践的必要性
❤软件维护的定义：所谓软件维护就是在软件交付使用之后，为了改正错误或满足新的需要而修改软件的过程。
❤改正性维护：把诊断和改正错误的过程称为改正性维护。
❤适应性维护：为了和变化了的环境适当的配合而进行的修改软件的活动。
❤完善性维护：为了满足用户在软件使用过程中提出增加新功能或修改已有功能的要求而进

行的维护。
❤预防性维护:为了改进未来的可维护性或可靠性或为了给未来的改进奠定更好的基础而修改软件。
❤软件维护的特点（1）结构化维护和非结构化维护差别巨大①非结构化维护②结构化维护（2）维护的代价高昂（3）维护的问题很多
❤软件维护过程①维护组织②维护报告③维护的事件流④保存维护记录⑤评价维护活动
❤决定软件可维护性的因素①可理解性②可测试性③可修改性④可移植性⑤可重用性
❤文档分为用户文档和系统文档。是影响软件可维护性的决定因素
❤文档需要满足的要求：必须描述如何使用这个系统；必须描述怎样管理和安装这个系统；必须描述系统需求和设计；必须描述系统的实现和设计；
❤可维护性复审的目的：保证软件配置的所有成分是完整的、一致的和可理解的。便于修改和管理已经编目归档的。
❤软件再工程过程:①库存目录分析②文档重构③逆向工程④代码重构⑤数据重构⑥正向工程;
❤详细设计的根本目标：确定怎样具体的实现所要求的系统。任务：设计出程序的蓝图
❤基本控制结构：顺序，选择，循环
最基本的控制结构：顺序，循环
DO-WHILE 先选择
DO-UNTIL 后选择
❤结构程序设计的经典定义：如果一个程序的代码块仅仅通过顺序、选择和循环这三种基本控制结构进行连接，并每个代码块只有一个入口和一个出口，则称这个程序是结构化的。
❤设计问题①系统响应时间②用户帮助设施③出错信息处理④命令交互
❤设计过程：用户界面设计是一个迭代的过程，通常先创建设计模型，再利用原型实现这个设计模型，并由用户试用和评估，然后根据用户意见进行修改。
❤设计指南①一般加护指南：信息显示、数据输入、系统整体控制②信息显示指南③数据输入指南
❤程序流程图的缺点①程序流程图本质上不是逐步求精的好工具,它诱使程序员过早的考虑程序的控制流程，而不去考虑程序的全局结构②用箭头代表控制流，因此程序员不受任何约束，可以完全不顾结构层序设计的精神，随意转移控制③不易表示数据结构
❤盒图：DO-WHILE 小方框在右下角
❤PAD 图的优点①使用表示结构化控制结构的 PAD 符号所设计出来的程序必然是结构化程序②程序结构清析③易读、易懂、易记④利于提高软件可靠性课生产率⑤可用于描绘数据结构⑥PAD 图的符号支持自定向下、逐步求精方法的使用。
❤过程设计的工具①程序流程图②盒图③PAD 图④判定表⑤判定树⑥过程设计语言
❤数据元素彼此间的逻辑关系：顺序、选择和重复
❤jackson 方法的步骤①分析并确定输入输出数据的逻辑结构，并用 jackson 图描绘②找出输入输出数据结构中友对应关系的单元③从数据 jackson 图导出程序 jackson 图④列出所有操作和条件，并把他们分配到程序结构图的适当位置⑤用伪码表示程序。
❤计算环形复杂度①流图中的区域数②边的条数-节点数③判定节点数+1
❤Halstead 方法计算程序复杂度
实际长度 N=N1（运算符出现总次数）+N2（操作数出现总次数）
预测长度 H=n1（不同运算符个数）log2n1+n2（不同操作数个数）log2n2
程序中包含错误的个数
E=Nlog2（n1+n2）/3000

❤对象有哪些特点①以数据为中心②对象是主动的③实现了数据封装④本质上具有并行性⑤模块独立性好

✿**实现**：编码和测试的统称

✿**编码**：软件设计结果翻译成用某种程设计语言书写的程序

✿**选择程序设计语言**：①系统用户的要求②可以使用的编译程序③可以得到的软件工具④工程规模⑤程序员的知识⑥软件可移植性要求⑦软件的应用领域

✿**从三个方面讨论效率**①程序运行时间②存储器效率③输入输出的效率

✿**测试**：为了发现程序中的错误而执行程序的过程。

✿**测试的规则**（G.Myers 给出的）：①为了发现程序中的错误而执行程序的过程②好的测试方案是极可能发现迄今为止尚未发现的错误的测试方案③成功的测试是为了至今为止尚未发现的错误的测试

✿**软件测试准则**：①所有测试都应该能追溯到用户需求②应该远在测试开始之前就制定出测试计划③把 Pareto 原理应用到软件测试中④应该从“小规模”测试开始，并逐步进行“大规模”测试⑤穷举测试是不可能的⑥为了达到最佳的测试效果，应该由独立的第三方从事测试工作

✿软件测试步骤：①模块测试（单元测试）：编码和详细设计②子系统测试：模块接口③系统测试：软件设计，需求说明④验收测试（确认测试）：需求说明书⑤平行运行：新旧系统同时运行

✿测试阶段的信息流有两类：①软件配置，包括需求说明书，设计说明书，源程序清单等②测试配置，包括测试计划和测试方案

✿测试方案：所谓测试方案不仅仅是测试时使用的输入数据，还应该包括每组输入数据预定要检验的功能，以及每组输入数据预期应该得到的正确输出

✿单元测试：集中检测软件设计的最小单元——模块

✿单元测试从 5 个方面进行测试：①模块接口②局部数据结构③重要的执行通路④出错处理通路⑤边界条件

✿代码审查：由审查小组正式进行

✿审查小组最好由 4 人组成：①组长②程序的设计者③程序的编写者④程序的测试者

✿驱动程序：“主程序”，接受测试数据，把这些数据传送给被测试的模块并且印出有关结果

✿存根程序：“虚拟子程序”，使用被它代替的模块的接口，可能做最少量的数据操作，印出对入口的检验或者操作结果，并且把控制归还给调用它的模块

✿集成测试：是测试和组装软件的系统化技术

✿集成策略：自顶向下和自底向上

✿自顶向下集成步骤：①对主控制模块进行测试，测试时用存根程序代替所有直接附属于主控制模块的模块②根据选定的结合策略，每次用一个世纪模块代换一个存根程序③在结合进一个模块的同时进行测试④可能需要进行回归测试

✿自底向上集成步骤：①把低层模块组合成实现某个特定的软件自功能的族②写一个驱动程序，协调测试数据的输入和输出③对由模块组成起来形成更大的子功能族

✿确认测试（验收测试）的目标：验证软件的有效性

✿验证：保证软件正确的实现了某个特定要求的一系列活动

✿自顶向下测试方法的优点：不需要测试驱动程序，能够在测试阶段的早期实现并验证系统的主要功能，而且能在早起发现上层模块的借口错误。缺点：需要存根程序，可能遇到与此相联系的测试困难，低层关键模块中的错误发现的较晚，而且用这种方法在早期不能充分展开人力

✿回归测试：重新执行已经做过的测试的某个子集，以保证修改没有带来的非预期的副作用

✿确认：保证软件确实满足了用户需求而进行一系列活动
✿软件有效性：如果软件的功能和性能如同用户所合理期待的那样，软件就是有效的
✿确认测试的重要内容：复查软件配置
✿确认测试通常使用黑盒测试法
✿Alpha 测试由用户在开发者的场所进行，是受控的环境中进行的。Beta 测试由软件的最终用户们在一个或多个场所进行。
✿逻辑覆盖：是对一系列测试过程的总称，这组测试过程逐渐进行越来越完整的通路测试。
✿逻辑覆盖：①语句覆盖②判定覆盖③条件覆盖④判定/条件覆盖⑤条件组合覆盖⑥点覆盖⑦边覆盖⑧路径覆盖
✿常用的控制结构测试（白盒测试技术）：①基本路径测试②条件测试③循环测试
✿基本路径测试：是 Tom McCabe 提出的一种白盒测试技术。首先计算程序的环形复杂度，并用该复杂度为指南定义执行路径的基本集合，从该基本集合导出的测试用例可以保证程序中的每条语句至少执行一次，而且每个条件在执行时都将分别去真，假两种值
✿基本路径测试步骤：①根据过程设计结果画出相应的流图②计算流图的环形复杂度③确定线性独立路径的基本集合④设计可强制执行基本集合中每条路径的测试用例
✿条件测试的优点：①容易度量条件的测试覆盖率②程序内条件的测试覆盖率可指导附加测试的设计
✿黑盒测试力图发现以下错误：①功能不正确或遗漏了功能②界面错误③数据结构错误或外部数据库访问错误④性能错误⑤初始化和终止错误
✿循环测试方法：①简单循环②嵌套循环③串接循环
✿白盒测试在测试过程的早期阶段进行，黑盒测试主要用于后期。
✿黑盒测试设计方案时应考虑：①怎样测试功能的有效性？②哪些类型的输入可构成好测试用例？③系统是否对特定的输入值特别敏感？④怎样划定数据类的边界？⑤系统能够承受什么样的数据率和数据量？⑥数据的特定组合将对系统运行产生什么影响？
✿应用黑盒测试能够设计出满足以下标准的测试用例：①所设计出的测试用例能够减少未达到合理测试所需要设计的测试用例的总数②所设计出的测试用例能够告诉人们，是否存在某些类型的错误，而不是仅仅指出与特定测试相关的错误是否存在
✿黑盒测试方法：①等价划分②边界值分值③错误推测
✿等价划分是一种黑盒测试技术，这种技术把程序的输入或划分成若干个数据类，据此导出测试用例。
❤等价划分的步骤①设计一个新的测试方案以尽可能多的覆盖尚未被覆盖的有效等价类，重复这一步骤直到所有有效等价类都被覆盖为止②设计一个新的测试方案，使他覆盖一个而且只覆盖一个尚未被覆盖的无效等价类，重复这一步骤直到所有无效等价类都被覆盖为止
❤可以划分出的等价类：有效输出的等价类①数字串（最高位不是 0）②最高位是 0 的数字串③最高位左邻是负号的数字串；无效输出的等价类①左部填充的字符既不是 0 也不是空格②最高位数字右面由数字和空格混合组成③最高位数字右面由数字和其它字符混合组成⑤负号与最高位数字之间有空格；合法输出的等价类①在计算机能表示的最小负数和 0 之间的负整数②0③在 0 和计算机能表示的最大正整数之间的正整数；非法输出的等价类①比计算机能表示的最小负整数还小的负整数②比计算机能表示的最大正整数还大的正整数
❤调试的结果①找到了问题的原因并把问题改正和排除掉②没找出问题的原因
❤调试也诚纠错，是在测试发现错误之后排除错误的过程。是把症状和原因联系起来的尚未被人深入认知的智力过程。
❤调试的途径①蛮干法②回溯法③原因排除法
❤软件可靠性：程序在给定的时间间隔内，按照规格说明书地规定成功的运行的效率

❤软件的可用性：程序在给定的时间点，按照规格说明书的规定，成功地运行的概率
❤估算平均无故障时间的方法①符号②基本假定③估算平均无故障时间④估计错误总数的方法（①植入错误发②分别测试法）
❤错误：由开发人员曹成的软件差错
❤故障：由错误引起的软件的不正确行为
❤测试过程的步骤①模块测试②子系统测试③系统测试④验收测试⑤平行运行
❤单元测试期间从 5 方面对模块测试①模块接口②局部数据接个③重要的执行通路④出错处理通络⑤边界条件