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，本钱相当昂贵。

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)软件开发本钱越来越大。

(6)软件本钱与开发进度难以估计。

(7)软件技术的开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。造成开发中制定方案盲目、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、公认的方法论和标准指导，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、本钱过高、研制周期不易估计、维护困难等一系列问题更为突出，技术的开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，同样的软件屡次重复开发。

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) 对软件开发本钱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) 软件本钱在计算机系统总本钱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那么、方法，以提高质量，降低本钱为目的。

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根本任务：

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，是否有可行的解决方法；
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〕使软件到达预定的要求。

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。

软件工程准那么可以概括为 7 条根本原那么：

a 用分阶段的生命周期方案严格管理；
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 成认不断改良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，把假设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.**最根本的测试**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假设干阶段工作，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标准的方法〔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：瀑布模型根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根底上，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拟容易修改已建立的系统逻辑模型。

数据字典的实现：P49

本钱效益分析：本钱/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假设干模块的过程**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假设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）草拟开发方案

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根底，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，改良提出的解放… …。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**根本思想是**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 种根本类型：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屡次)

总体设计的过程

- 设想供选择的方案
- 选取合理的方案
- 推荐最正确方案
- 功能分解
- 设计软件结构
- 设计数据库
- 制定测试方案
- 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、测试方案、详细的实现方案、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、 低本钱
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
数据流图根本符号有 4 种：

本钱估计技术
1、代码行技术 2、任务分解技术 3、自动估计本钱技术
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局部,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方案

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.根本路径测试(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.

原那么: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，如用图形工具表示那么更加直观简洁。系统流程图是描绘物理系统的传统工具，它的根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。

概括性那么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)其它考前须知

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，分成假设干个层次〔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局部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中根本图形符号正方形、圆角矩形、开口矩形

2、数据流图的根本要点是描绘“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、画分层数据流图根本原那么与考前须知：

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。根据抽象原那么，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，防止空洞的名字。

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)，那么父图允许有 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.**访谈有两种根本形式**，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。

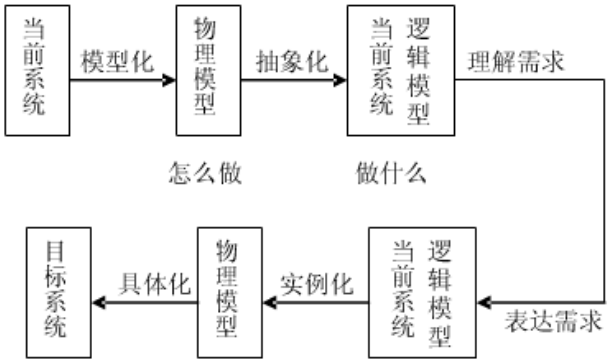
38.**软件的验证**：一致性，完整性，现实性，有效性

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：它的根本任务是准确地答复“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。其实现步骤如以下图所示：



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”的根底上，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异；然后将变化局部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局部进行分析，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） 修正系统开发方案

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，因此可以对系统的本钱和进度做出更准确地估计，在此根底上应该对开发方案进行修正。

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、需求分析的原那么：

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根底，大量统计数字说明，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，降低软件开发本钱，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根本上可以实现的。对硬件技术的进步可以做些预测，对软件技术的进步那么很难做出预测，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根底。

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根本需求；接下来应该与用户深入交流，对用户的根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，并且预测系统的开展前景。

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适宜的记号〔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，那么称为过程内聚。

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，那么称为顺序内聚。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。

56.一个模块的扇入说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拟高，中层扇出比拟少，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.总体设计阶段的根本目的是用比拟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〕的软件比拟容易开发出来；b、独立的模块比拟容易测试和维护。

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。经验说明，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，模块的独立性那么越差。模块间耦合上下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。假设一个模块内各元素(语句之间、程序段之间)联系得越紧密，那么它的内聚性就越高。

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根底。

模块的高内聚、低耦合的原那么称为模块独立原那么，也称为模块设计的原那么。

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**：总体设计的根本目的就是答复“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方案、软件需求分析提供的文档、提出最正确推荐方案，用系统流程图，组成物理元素清单，本钱效益分析，系统的进度方案，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，测试方案，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那么，例如模块划分独立性原那么，信息隐蔽原那么等

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〕、推荐最正确方案

4〕、功能分解

5〕、设计软件

6〕、设计数据库

7〕制定测试方案

8〕、书写文档：系统说明、用户手册、测试方案、详细的实现方案、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那么是，模块化和软件本钱关系〕：

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.

模块化和软件本钱关系:根据总本钱曲线,每个程序都相应地有一个最适当的模块数目 M,,使得系统的开发本钱最小.

3、**模块设计的准那么：**

(1) 改良软件结构, 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那么，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拟容易开发出来

2)独立的模块比拟容易测试和维护。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、启发规那么：

1）改良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。

设计原那么：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：

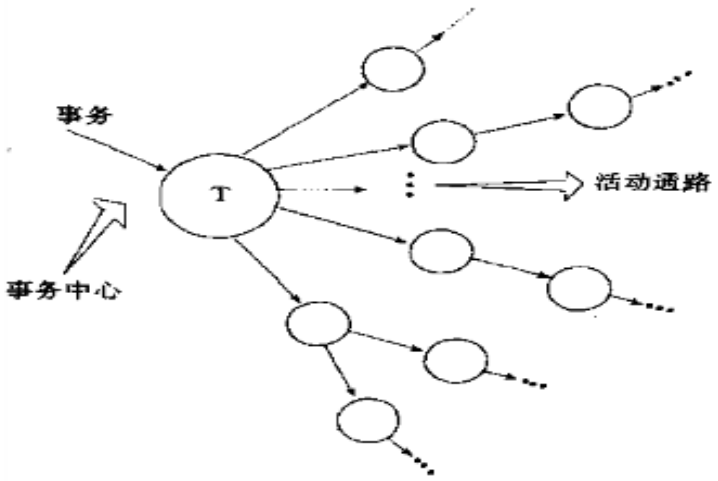
1、面向数据流的设计方法把信息流映射成软件结构，信息流的类型决定了映射的方法。面向数据流的设计要解决的任务，就是上述需求分析的根底上，将 DFD 图映射为软件系统的结构。

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局部组成，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，一个事务中心与假设干条动作路径组成。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 方法规定的一组规那么，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、结构设计的优化规那么：

1〕对模块分割、合并和变动调用关系的指导规那么：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 / 入低扇出原那么

3〕作用域 / 控制域规那么：

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。

总体设计阶段的根本目的是比拟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假设干种合理的物理方案，分析员应该仔细分析比拟这些方案，并且和用户共同选定一个最正确方案。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 种根本结构进行连接，并且每个代码块只有一个入口和一个出口，那么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根本保证，是进行详细设计的逻辑根底。

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：详细设计的原那么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根底上，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方案的重要组成局部，通常包括输入数据，期望输出等内容。

3、**详细设计的原那么：**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，那么称这个程序是结构化的。

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

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，比拟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根本控制结构。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：规定了五种根本图形构件：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 图也设置了五种根本控制结构的图示，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.比拟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”逐步细化的原那么,并以其为共同的根底;

(2) 均服从“程序结构必须适应问题结构”的根本原那么,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那么。

不同点：

(1)

面向数据流的设计以数据流图为根底，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根底，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。 过程设计的原那么和方法：

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种根本循环结构——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 方法的根本步骤是：

(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、比拟 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，设计步骤根本相似。它们的主要差异是：

(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那么有一定的灵活性，而 LCP 设计方法那么更加严密。

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、**软件测试的准那么：**

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方案

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）为了到达最正确的测试效果，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，包括测试方案和测试方案〔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工程选择语言时，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说明未初始化的值；
(4) 对象的局部性：程序中真正需要的那局部才能访问的对象；
(5) 程序模块：控制程序对象的名字；