软件工程学概述

1、什么是软件？特点有哪些 ？

软件是**程序、数据和文档**的完整集合。

特点：

软件是一种逻辑实体。

是人类的智力产品

软件需要长期维护

软件开发过程复杂

软件成本昂贵

软件可以复制

2、 软件危机（超重要）

定义：

软件危机是指在计算机软件的开发和维护过程中所遇到的一系列严重问题。

软件危机包含下述两方面的问题：

1） 如何开发软件，以满足对软件日益增长的需求。

2） 如何维护数量不断膨胀的已有软件。

软件危机的一些典型表现（超重要）

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。

软件危机产生的原因

（1） 客观原因

1. 软件是计算机系统中的逻辑**部件**而不是物理部件
2. 软件规模庞大，而且程序复杂性将随着程序**规模**的增加而呈**指数**上升。

（2） 主观原因

1. 对软件开发和维护有不少糊涂观念，采用了错误的方法和技术。
2. 对用户要求没有完整准确地认识就匆忙着手编写程序。
3. 一个软件从定义、开发、使用和维护，直到最终被废弃，要经历一个漫长的**时期**。
4. 一个软件产品必须由一个完整的**配置**组成，**主要包括程序、文档和数据等成分。**
5. 在软件开发的不同阶段进行**修改**需要付出的代价是很不相同的。
6. **轻视维护**

消除软件危机的方法（考 1）

（1）应该对计算机软件有一个**正确的认识**。

（2）应该充分认识到软件开发是一种组织良好、管理严密、各类人员**协同配合、共同完成**的工程项目。

（3）应该推广使用在实践中总结出来的开发软件的成功的技术和方法,并且研究探索更好更有效的技术和方法,尽快消除在计算机系统早期发展阶段形成的一些错误概念和做法。

（4）应该开发和使用更好的软件工具。

3、软件工程

1.软件工程的介绍

（1）概念（选择）

软件工程是从**管理和技术**两方面研究如何能更好的开发和维护计算机软件的一门新兴学科。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,以经济地开发出高质量的软件并有效地维护它,这就是软件工程。

（3）**本质特征**

1）软件工程关注于大型程序的构造

2）软件工程的**中心课题是控制复杂性**

3）软件经常变化

4）开发软件的效率非常重要

5）和谐的合作是开发软件的关键

6）软件必须有效的支持它的用户

7)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。

2. 软件工程的 7 条基本原理

用分阶段的声明周期计划严格管理

坚持进行阶段评审

进行严格的产品控制

采用现代程序设计技术

结果应能清楚地审查

开发小组的人员应该小而精

承认不断改进软件工程时间的必要性

3.软件工程方法学（方法学也叫做范型）

1）主要包括传统方法学和面向对象方法学

2）软件工程方法学三要素

方法：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：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。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。**

3） 传统方法学（生命周期方法学或结构化范型）

优点：

a 把软件生命周期划分成若干个阶段,每个阶段的任务相对独立，而且比较简单，便于不同人员分工协作，从而降低了整个软件开发工程的困难程度;

b 在软件生命周期的每个阶段都采用科学的管理技术和良好的技术方法，而且在每个阶段结束之前都从技术和管理两个角度进行严格的审查，合格之后才开始下一阶段的工作,这就使软件开发工程的全过程以一种有条不紊的方式进行,保证了软件的质量,特别是提高了软件的可维护性。

c 采用生命周期方法学可以大大提高软件开发的成功率和生产率。

4）面向对象方法学（封装，继承、多态）

优点：

降低了软件产品的复杂性

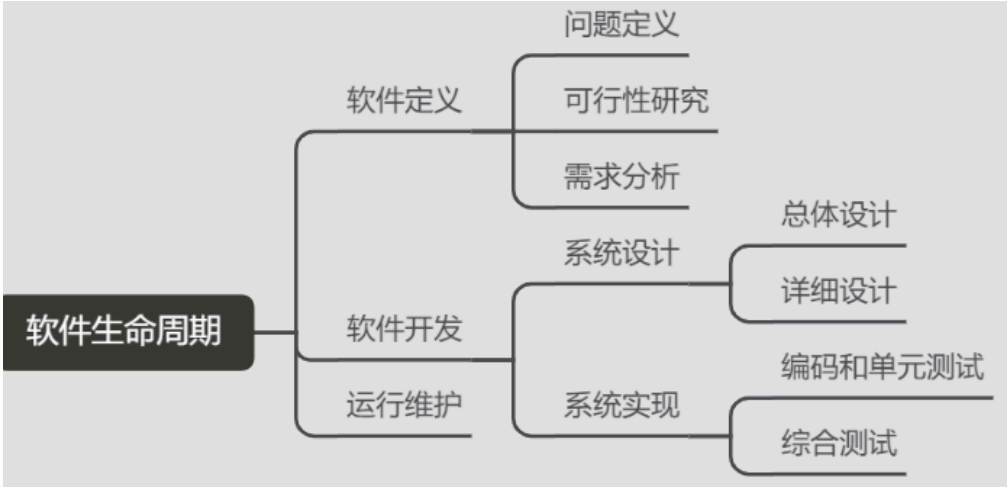
提高了软件的可理解性

简化了软件的开发和维护工作

提高了软件的可重用性

软件生命周期（三个时期，八个阶段）

软件生命周期由软件定义、软件开发和运行维护（软件维护）3 个时期组成，



软件维护

通常有四类维护活动

[1] 改正性维护，即诊断和**改正**正在使用过程中发现的软件错误；

[2] 适应性维护，即修改软件以**适应**环境的变化

[3] 完善性维护，即根据用户的要求改进或扩充软件使它更**完善**

[4] 预防性维护，即修改软件，为将来的维护活动预先做准备

4、软件过程

1. 概念

（1） 定义

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,它规定了完成各项任务的工作步骤。

典型软件过程模型

（1） 瀑布模型

是软件工程中应用得最广泛的过程模型。传统软件工程方法学的软件过程+基本上可以用瀑布模型来描述。

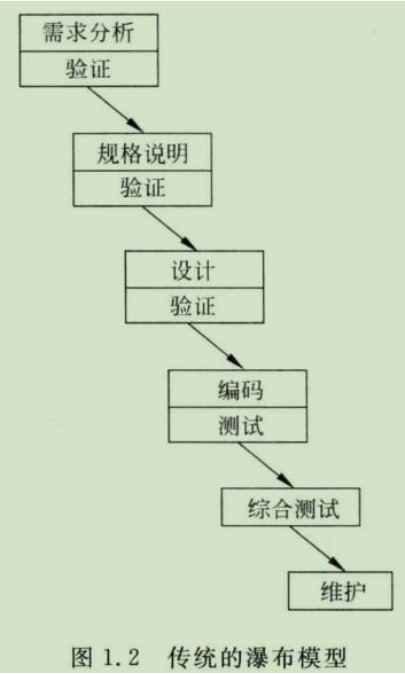


图 1.2　传统的瀑布模型

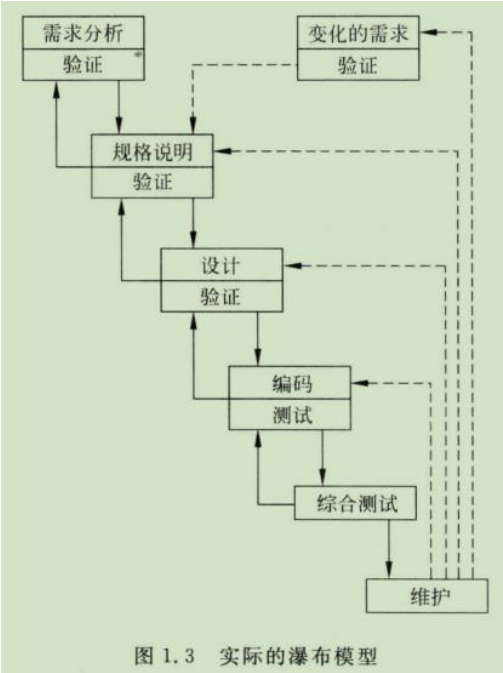


图 1.3　实际的瀑布模型

特点：

1 阶段间具有顺序性和依赖性

① 须等前一阶段的工作完成之后，才能开始后一阶段的工作;②前一阶段的输出文档就是后一阶段的输入文档，因此，只有前一阶段的输出文档正确，后一阶段的工作才能获得正确的结果。

2 推迟实现的观点

3 质量保证的观点

瀑布模型的优点

第一，可强迫开发人员采用规范的方法(例如，结构化技术);

第二，严格地规定了每个阶段必须提交的文档;

第三，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。

第四，对文档的约束，使软件维护变得容易一些，且能降低软件预算。

缺点，起码背下前两条。

a. 开发过程一般不能逆转，否则代价太大;
b. 实际的项目开发很难严格按该模型进行;
c. 客户往往很难清楚地给出所有的需求;
d. 软件的实际情况必须到项目开发的后期客户才能看到。

（2）　快速原型模型

定义：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。

优点：减少由于软件需求不明确带来的开发风险。这种模型适合预先不能确切定义需求的软件系统的开发。

（3） 增量模型

定义：使用增量模型时，把软件产品作为一系列的增量构件来设计、编码、集成和测试，适应需求的变更。

增量模型的优点：

第一， 能在较短时间内向用户提交可完成部分工作的产品

第二， 逐步增加产品功能可以使用户有较充裕的时间学习和适应新产品，从而减少一个全新的软件可能给客户组织带来的冲击。

缺点：增量之间可能有不能集成的风险。

（4） 螺旋模型

风险评估很重要，比较灵活，客户始终参与开发，不会使软件偏离正常方向。

（5） 喷泉模型（面向对象模型）

第一章 可行性研究

可行性研究的任务

可行性研究的目的

可行性研究的目的是，确定客户提出的问题是否有行得通的解决办法。必须分析几种主要的候选解法的利弊，从而判断原定的系统目标和规模是否现实，系统完成后所能带来的效益是否大到值得投资开发这个系统的程度。

从哪些方面研究目标系统的可行性？

（1） 技术可行性 使用现有的技术能否实现这个系统

（2） 经济可行性 这个系统的经济效益能否超过它的开发成本（收益）

（3） 操作可行性 系统的操作方式在这个用户组织内能否行得通（使用）

数据流图

1 概念

（1） 定义

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。

用途：

作为交流信息的工具。

作为分析和设计的工具

数据字典（DD）

数据字典是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，它的作用是在软件分析和设计的过程中给人提供关于数据的描述信息。

意义

数据流图和数据字典共同构成系统的逻辑模型，没有数据字典，数据流图就不严格，然而没有数据流图，数据字典也难以发挥作用。只有数据流图和对数据流图中的每个元素的精确定义放在一起，才能共同构成系统的规格说明。

内容：

数据字典由下列四类元素组成

a 数据流

b 数据流分量，即数据元素

c 数据存储

d 处理

数据字典的用途：

作为分析阶段的工具

数据字典改进分析员和用户之间的通信

数据字典是开发数据库的第一步，而且是很有价值的一步。

优点：全面准确地定义数据。缺点：不够形象直观。

第二章 需求分析

需求分析的任务

对目标系统提出完整，准确，清晰，具体的要求。

需求分析的原因（背）

为了开发出真正满足用户需求的软件产品，需求分析是软件开发工作获得成功的前提条件，不能满足用户需求的产品只会令用户失望，给开发者带来烦恼。

需求分析阶段用到的一些图形工具（背）

实体联系图、状态转换图、层次方框图、Warnier 图、IPO 图

导出系统的逻辑模型

通常用数据流图（DFD）实体联系图（E-R 图），状态转换图，数据字典（DD）和主要的处理算法描述这个逻辑模型。

与用户沟通获取需求的方法（背）

访谈

面向数据流自顶向下求精

简易的应用规格说明技术

快速建立软件原型。

面向数据流自顶向下求精

（1） 定义

结构化分析方法是**面向数据流**自顶向下逐步求精进行需求分析的方法

3.4 实体联系图（ER）

数据对象

特点：

a. 可以由一组属性来定义的实体都可以被认为是数据对象

B．数据对象彼此间是有关联的

C．数据对象只封装了数据而没有对施加于数据上的操作的引用

ER 图的优点：

比较接近人的习惯思维方式

使用简单的图形符号表达，便于用户理解

3.6 状态转换图

状态图中定故意的状态主要有：初态、终态和中间状态，一张状态图中只能有一个初态，而终态可以有 0 至多个。

3．7 其他图形工具

层次方框图

Warnier 图

IPO 图（输入、处理、输出图简称）：方便的描绘输入数据、对数据的处理和输出数据之间的关系。

3.8 验证软件需求

从以下四个方面验证软件需求的正确性

一致性、完整性、现实性、有效性。（背）

助记：一碗咸油

第三章 总体设计

一、概念

定义：基本目的就是回答“系统应该如何实现”这个问题，总体设计又称为概要设计或者初步设计。

二、设计过程

总体设计过程通常由系统设计阶段和结构设计阶段组成

五、描绘软件结构的图形工具（总体设计用到的工具）

（1）层次图和 HIPO 图（层次图用来描绘软件的层次结构）

（2）结构图

三、设计过程应该遵循的原理

1. 模块化

（1） 模块

模块是由边界元素限定的相邻程序元素的序列，而有一个总体标识符代表它。模块是构成程序的基本构件。过程、函数、子程序和宏等，都可作为模块。面向对象方法学中的对象是模块，对象内的方法也是模块，模块是构成程序的基本构件。

（2） 优点：易读性、可靠性、可修改性、可管理性

2.抽象

3.逐步求精

4 信息隐藏和局部化

5.模块独立

模块独立的概念是模块化、抽象、信息隐藏、和局部化概念的直接结果。

模块的独立性可以用两个定性标准度量：耦合和内聚

耦合（数据耦合，控制耦合，特征耦合，公共环境耦合，内容耦合）

定义：是对一个软件结构内不同模块之间互联程度的度量。

设计原则：低耦合，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。

（3） 内聚（背）

定义：一个模块内各个元素彼此结合的紧密程度，

低内聚：偶然内聚、逻辑内聚、时间内聚

中内聚：通信内聚、过程内聚

高内聚：顺序内聚、功能内聚

软件设计的目标是：高内聚，低耦合。

四、启发规则（背）

（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）设计单入口单出口的模块

（7）模块功能应该可以预测

六、面向数据流的设计方法（变换分析、事物分析、设计优化）

1、概念

软件工程的需求分析阶段，信息流是一个关键考虑，通常用数据流图描绘信息在系统中加工和流动的情况。

信息流包括：**变换流、事物流**

第四章 详细设计

一、详细设计概述

二、详细设计用到的工具：流程图、盒图、pad 图、判定表、判定树、

1 目标

2 工作任务

3 关键技术：详细设计的结果基本上决定了最终的程序代码的质量。结构程序设计技术是实现详细设计目标的关键技术，也是详细设计的逻辑基础。

三、结构程序设计

1. 控制结构

（1） 基本的控制结构：顺序、选择、循环

（2） 扩展的控制结构：do until 、do case

（3） 修正的控制结构：leave 和 break

（4） 分类

①经典的结构程序设计
只允许使用顺序、IF THEN ELSE 型分支和 DO WHILE 型循环这三种基本控制结构。
②扩展的结构程序设计
除了基本控制结构之外，还允许使用 DO CASE 型多分支结构和 DO UNTIL 型循环结构。
③修正的结构程序设计
除了扩展的结构程序设计之外，再允许使用 LEAVE（或 BREAK）结构。

四、过程设计的工具（背）

1. 定义

描述程序处理过程的工具称为过程设计的工具，它们可以分为图形，表格和语言三类。

2. 分类

（1）程序流程图

优点：对控制流程的描绘很直观，便于初学者掌握。

缺点：

a. 程序流程图本质上不是逐步求精的好工具，它诱使程序员过早地考虑程序的控制流程，而不去考虑程序的全局结构。
b. 程序流程图中用箭头代表控制流，因此程序员不受任何约束，可以完全不顾结构程序设计的精神，随意转移控制。
c. 程序流程图不易表示数据结构。

（2）盒图（N-S 图）

特点

功能域明确

不可能任意转移控制

很容易确定局部和全程数据的作用域

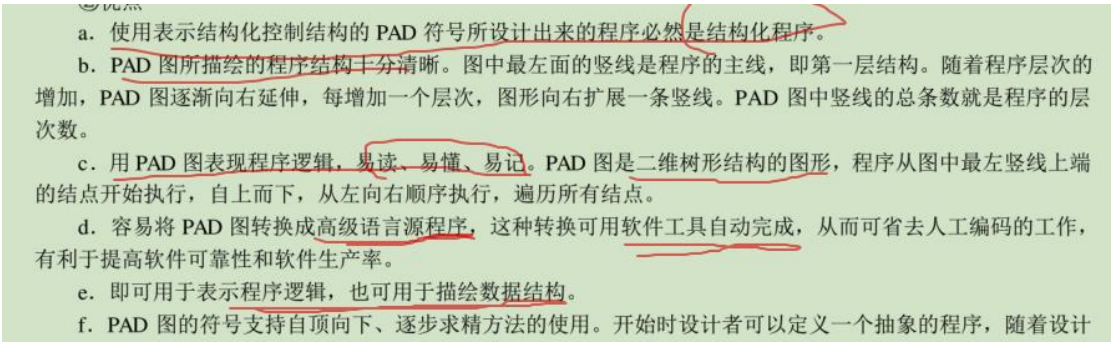
很容易表现嵌套关系，也可以表示模块的层次结构。

优点：

③优点
盒图没有箭头，因此不允许随意转移控制。坚持使用盒图作为详细设计的工具，可以使程序员逐步养成用结构化的方式思考问题和解决问题的习惯。

（3）Pad 图（问题分析图）

优点：



a. 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。

b. PAD 图所描绘的程序结构十分清晰。图中最左面的竖线是程序的主线，即第一层结构。随着程序层次的增加，PAD 图逐渐向右延伸，每增加一个层次，图形向右扩展一条竖线。PAD 图中竖线的总条数就是程序的层次数。

c. 用 PAD 图表现程序逻辑，易读、易懂、易记。PAD 图是二维树形结构的图形，程序从图中最左竖线上端的结点开始执行，自上而下，从左向右顺序执行，遍历所有结点。

d. 容易将 PAD 图转换成高级语言源程序，这种转换可用软件工具自动完成，从而可省去人工编码的工作，有利于提高软件可靠性和软件生产率。

e. 即可用于表示程序逻辑，也可用于描绘数据结构。

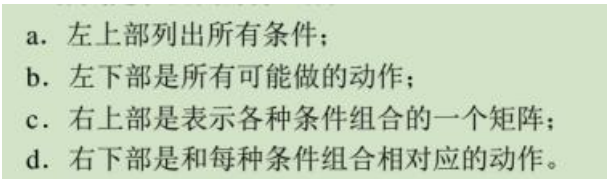
f. PAD 图的符号支持自顶向下、逐步求精方法的使用。开始时设计者可以定义一个抽象的程序，随着设计

（4）判定表

适用性

算法中包含多重嵌套的条件选择时，判定表能够清晰地表示复杂的条件组合与应做的动作之间的对应关系

由四部分组成：



a. 左上部列出所有条件；
b. 左下部是所有可能做的动作；
c. 右上部是表示各种条件组合的一个矩阵；
d. 右下部是和每种条件组合相对应的动作。

（5）判定树

五、面向数据结构的设计方法

最终目标：得出对程序处理过程的描述。

适用性：适合在详细设计阶段使用

Jackson 图

Jackson 方法和 warnier 方法是最著名的两个面向数据结构的设计方法，Jackson 图中数据元素彼此间的逻辑关系可分为顺序结构，选择结构和重复结构三类。

Jackson 图的优点

（1） 便于表示层次结构，而且是对结构进行自顶向下分解的有利工具

（2） 形象直观可读性好

（3） 既能表示数据结构也能表示程序结构

缺点：

表示选择或者重复结构时，选择条件或循环条件不能直接在图上表示出来，图的表达能力差，不易直接把图翻译成程序。

第五章 实现

实现包括编码和测试两个阶段。

一、编码

1 定义

编码是把软件设计结果翻译成用程序设计语言书写的程序，是对设计的进一步具体化，程序的质量主要取决于软件设计的质量。

效率主要是处理机时间和存储器容量两个方面（也就是时间和空间效率）

注意：软件测试中设计测试用例主要由输进数据和预期输出结果两部分组成。

二、软件设计基础

1 软件测试的目标（背）

（1） 测试是为了发现程序中的错误而执行程序的过程

（2） 好的测试方案极可能发现迄今为止尚未发现的错误

（3） 成功的测试是发现了至今为止尚未发现的错误的测试

（4） 软件测试的根本目标是尽可能多的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

注意：测试只能查找出程序中的错误，不能证明程序中没有错误。

2 软件测试准则（背）

所有测试都应该能追溯到用户需求。

应该远在测试之前就制定出测试计划。

应该从“小规模测试”开始，逐步进行“大规模测试”。

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。

3 测试方法（背）

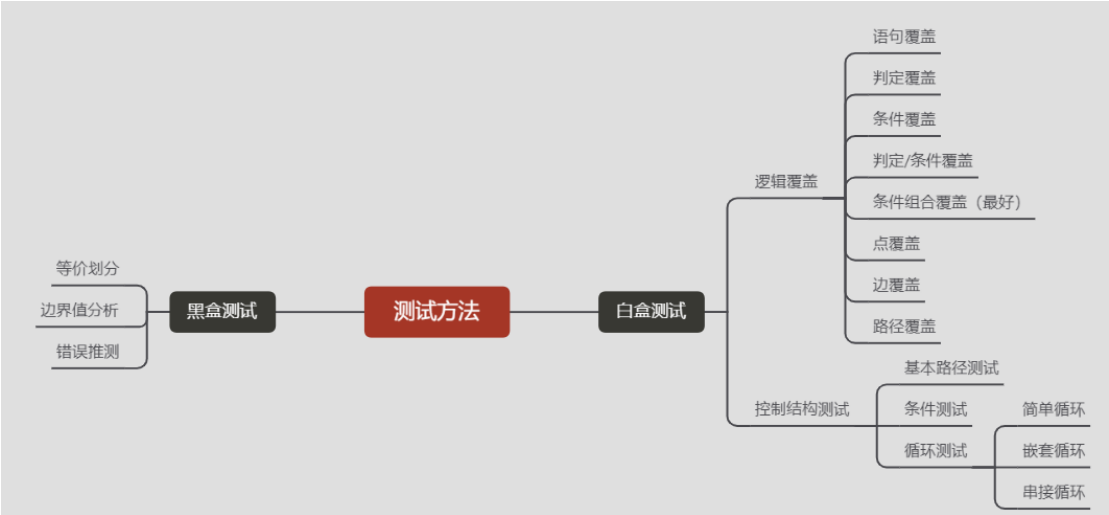
（1） 白盒测试（逻辑覆盖、控制结构测试）

白盒测试的前提是可以把程序看成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。这种方法按照程序内部的逻辑测试程序，检测程序中的主要执行通路是否都能按预定要求正确工作。

（2） 黑盒测试（等价类划分、边界值分析、错误推测）

黑盒测试（功能测试）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。黑盒测试是在程序接口进行的测试，只检查程序功能是否能按照规格说明书的规定正常使用，程序是否能接收输入数据并产生正确的输出信息，程序运行过程中能否保持外部信息的完整性。

注意：测试的早期阶段主要使用白盒测试，测试过程的后期阶段使用黑盒测试



4 测试步骤

主要分为三个阶段，模块测试（单元测试）、集成测试、验收测试（确认测试）（背）

（1） 模块测试

模块测试（单元测试）是把每一个模块作为一个单独的实体来测试，检验其正确性。目的是保证每一个模块作为一个单元能正确运行。模块测试所发现的是编码和详细设计的错误。

测试重点：

（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

（2） 集成测试

集成测试是把模块装配在一起形成完整的软件包，在装配的同时进行测试。

包括两种测试方法：非渐增式方法和渐增式方法。

非渐增式测试：先分别测试每个模块，再把所有模块按设计要求放在一起结合成所要

的程序。

渐增式测试：把下一个要测试的模块同已经测试好的那些模块结合起来进行测试，测试完以后再把下一个应该测试的模块结合起来进行测试，每次增加一个模块。渐增式测试同时完成单元测试和集成测试。

（1） 非渐增式测试的缺点：

把所有模块放在一起，测试者面对的情况十分复杂。

在庞大的程序中诊断定位一个错误非常困难。

一旦改正一个错误之后，又会遇到新的错误，没有穷尽。

（2） 渐增式测试的优点：

把程序划分成小段来构造和测试，比较容易定位和改正错误。

对接口可以进行更彻底的测试。

可以使用系统化的测试方法

（3） 确认测试（验收测试）（alpha 和 beta 测试）

把软件系统作为单一的实体进行测试，它是在用户积极参与下进行的，而且主要使用实际数据进行测试。验收测试的目的是验证系统确实能够满足用户的需要，验收测试发现的是系统需求说明书中的错误。

（1） 验证

保证软件正确的实现了某个特定要求的一系列活动

（2） 确认

为了保证软件确实满足了用户需求而进行的一系列活动

（3） 软件有效性

如果软件的功能和性能确实如用户所期待的那样，软件就是有效的。

2. 确认测试通常使用黑盒测试法。

（4） 回归测试

（1） 定义

回归测试是指重新执行已经做过的测试的某个子集，它可以保证由于调试或其他原因引起的变化，不会导致非预期的软件行为或额外错误。

alpha 和 beta 测试

（1）Alpha 测试

是用户在开发者的场所进行，并且在开发者在对用户的“指导下”进行测试，并且

开发者负责记录发现的错误和遇到的问题。Alpha 测试是在受控的环境中进行的。

（2）Beta 测试

软件的最终用户们在一个或多个客户场所进行。开发者通常不在 beta 测试的现场，即 beta 测试是软件在开发者不能控制的环境中的“真实”应用。

第六章 维护

一、软件工程的目的

提高软件的可维护性，降低维护的代价

二、软件维护

定义：在软件交付使用后，为了改正错误或者满足需要而修改软件的过程。

分类：

改正性维护

适应性维护

完善性维护

预防性维护

三、软件维护的特点

结构化维护与非结构化维护差别巨大

维护的代价高昂

维护存在的问题

四、软件维护的过程

本质上是修改和压缩了软件定义和开发过程

五、软件的可维护性

定义：维护人员理解、改正、改动或改进这个软件的难易程度。提高可维护性是支配软件工程方法学所有步骤的关键目标。

决定软件可维护性的因素

可理解性

可测试性

可修改性

可移植性

可重用性

第七章 面向对象方法学

1、什么是面向对象方法学？有哪些优点？

面向对象方法学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解决问题的方法与过程，使描述问题的问题空间与实现解法的解空间在结构上尽可能一致的方法学科。

优点：

与人类习惯的思维方法一致。

面向对象软件稳定性好

面向对象软件可重用性好

较易开发大型软件产品

可维护性好

2、什么是对象？与传统的数据有何异同？

对象是对问题域中某个实体的抽象

相对于传统数据结构的静态被处理，对象既有静态的属性，也有动态的行为，是进行处理的主体。

3、什么是类？

类是具有相同数据结构和相同操作的一组组相似对象的定义，即类是对具有相同属性和行为的一个或多个对象的描述，包括对怎样创建该类的新对象的说明。

4、什么是继承？

能够直接获得已有的性质和特征，而不必重复定义他们。在面向对象的软件技术中，继承是子类自动地共享基类中定义的数据和方法的机制。