软件工程 复习笔记

第 1 章 软件工程概论

1．1 软件工程发展史

1． 软件：有三层含义

（1） 个体含义，指计算机中的程序及其文档。

（2） 整体含义，指在特定计算机系统中所有上述个体含义下的软件的总称，即计算机系统中硬件除外的所有成分。

（3） 学科含义，指在研究、开发、维护以及使用前述含义下的软件所涉及的理论、方法、技术所构成的学科。

软件工程：是一类求解软件包的工程。它应用计算机科学、数学及管理科学等原理，借鉴传统工程的原则、方法、创建软件以达到提高质量、降低成本的目的。其中，计算机科学、数学用于构造模型与算法，工程科学用于制定规范、设计范型、评估成本及确定权衡，管理科学用于计划、资源、质量、成本等管理。软件工程是一门指导计算机软件开发和维护的工程学科。

2． 软件开发过程的演化

第一个时期：40-60 年代 个体时期（程序时期）

第二个时期：软件作坊时期（程序+文档）60-70 年代

第三个时期：软件工程时期（70 年代-现代）

一个阶段：软件工程方法时期（结构化方法等）

计算机辅助软件工程技术（CASE）

二个阶段：集成的计算机辅助软件工程时期（ICASE）

3． 软件危机

定义：指在计算机软件开发和维护过程中所遇到的一系列的严重问题。

具体表现在以下几方面：

① 软件开发成本高，成本难以控制

② 研制周期长，软件开发进度难以估计

③ 正确性难保证（软件质量差）

④ 缺乏完整、正确的文档资料

⑤ 软件的维护困难，维护的人员与费用不断增加

⑥ 软件的发展跟不上硬件的发展和用户的要求

软件危机的原因：

① 客观原因：软件需求大，规模大

② 主观原因：软件本身特点

开发过程中的原因

* 用户对软件需求的描述不精确（二义性、遗漏、错误）

* 软件开发人员对用户需求的理解与用户本来的愿望有差异

1．2 软件工程框架（重点内容，仔细读书中相关内容）

* 目标

① 正确性

② 可用性

③ 合算性

软件工程的目标概括为“生产具有正确性、可用性以及开销合宜的产品

*活动

① 需求

问题定义（Specification 软件需求归约，也称规格说明）

需求分析（功能归约）

② 设计

概要设计

详细设计

③ 实现—软件产品

④ 确认 贯穿于整个开发过程（交付产品）

⑤ 支持

软件工程活动是“生产一个最终满足需求且达到工程目标的软件产品所需要的步骤”

*原则

① 开发模型

② 设计方法

③ 支持过程（工程支持）

④ 管理过程

1．3 软件生命（存）周期

指软件产品或软件系统，从生产、投入使用到被淘汰的全过程。

把软件生命周期依次划分为若干阶段，每个阶段有相对独立的任务，逐步完成各个阶段的任务后，软件的生命周期向前推进。

全三大阶段

（一） 软件计划阶段

① 问题定义

“要解决的问题是什么？”

② 可行性研究

在时间和资源的约束条件下，能否完成指定的任务.

可行性研究所研究的内容如下：

（1） 技术可行性

（2） 经济可行性

（3） 法律可行性

（4） 社会可行性

（二） 软件开发阶段

* 需求分析

* 软件设计

* 编码

* 测试

* 运行

（三） 软件维护阶段

1．4 软件开发方法

指软件开发过程中所遵循的方法和步骤，是规则、方法和工具的集成，既支持开发，也支持开发以后的演化过程。

（1） 由一系列步骤组成的设计开发过程

（2） 各个步骤使用的设计表示

（3） 各步骤中使用的工具、技术和方法

第 2 章 软件开发模型

掌握模型的活动、特征、活动间的关系、不同点。

软件开发模型：是软件开发全部过程、活动和任务的框架。

（要求掌握：什么是？特征，图，优缺点，适用范围）

2．1 瀑布模型（waterfall）

将软件生命周期的各项活动规定为按固定次序连接的若干阶段的工作。

特征：P5

① 上一阶段的结果作为本阶段的输入（从上一阶段接受本阶段工作的对象，作为输入）

② 尽可能推迟软件编码时间

③ 为保证软件质量，每个阶段都要实施评审工作，以便及早发现错误，改正错误

优点：

降低软件复杂度，促进软件开发工程化

缺点：

① 缺乏灵活性，特别是无法解决软件需求不明确或不准确的问题

② 后期错误，修正代价高

2．2 演化模型(evolutioary)

针对不能完整定义需求的软件开发，用户可给出待开发系统的核心需求，对开发出的核心系统有效的提出反馈，最终设计和实现软件系统。

图见书第 6 页

原型：

① 抛弃型

② 演示型

③ 样品型

④ 增长式的演化型

特征：减少了软件开发活动的盲目性，减少由于软件需求不明确而给工作带来的风险性

2．3 螺旋模型（spiral）

把瀑布模型和演化模型结合起来，并在此基础上加入风险分析得到的模型。

图见书第 7 页

2．4 喷泉模型（fountain）

图见书第 8 页

特征：迭代和无间隙

迭代：软件开发各个阶段多次重复。

2．5 增量模型（Increaseted）

先开发出整个系统中一个功能（子集），再逐步开发出后续功能，直至整个系统的完成。

重点掌握：软件工程框架　　软件生命（存）周期（概念）

第 3 章 需求分析

关键性阶段：

对于要开发的目标系统在功能上、性能上进行规格说明。

对于目标系统理解、分析、建立系统模型，将需求精确化、完全化、形成一个软件需求规格说明书。

基本任务：准确地定义未来系统的目标，回答“系统必须做什么？”

任务的承担者：系统分析员

需求分析分两个阶段：

① 需求获取阶段：系统分析员通过学习，并且与用户交互，熟悉要开发系统的领域知识，并且获得用户对要开发系统的要求

活动：需求获取

结果：需求定义

② 需求归约阶段：在获得用户初步需求后进行一致性分析检查，通过与用户协商、解决需求中存在的二义性和不一致性问题，并以一种规范化的形式准确表达用户需求，形成软件需求规格说明书

活动：需求获取　　需求定义

需求分析　　软件需求规格说明（建立系统模型）

需求验证　　软件需求规格说明

3．1 需求获取

1． 概述

* 需求：对系统特征及为了完成用户要求，系统必须做什么的一个描述。

* 目的：清楚理解所要解决的问题，完整地获取用户需求。

* 步骤：

① 获取当前系统的物理模型

② 抽象出当前系统的逻辑模型

③ 建立目标系统的逻辑模型

④ 补充（用户界面、系统启动和结束、出错处理、新的输入输出）

*困难：

① 对于问题空间的理解

② 人与人之间的通信

③ 需求的易变性

④ 表示方法 结构化自然语言+图（表）

2． 需求获取的内容

功能性需求

非功能性需求：性能

3． 应遵循的原则

* 划分：“整体/部分”关系

降低问题空间的复杂性

* 抽象：“一般/特殊”

* 投影：“多角度分析”

4． 需求获取采用的技术和方法

* 系统分析员的素质：

① 概括能力、分析能力、社交活动能力。

② 具有一定的开发计算机软、硬件系统的经验。

③ 能理解用户提出的要求。

④ 善于在用户和软件开发机构之间进行良好的通信，协调。

*采取的手段：

主要活动：

通过学习，请教领域专家，向用户提问，了解要解决的问题，理解用户的需要

①访谈与会议

②观察用户工作流程

③系统分析员与用户成立联合小组

基于数据的系统模型

面向对象的系统模型

3．2 结构化分析（Structure Analysis）方法

1．概述

* 结构化方法：

（1）SA

（2）SD（D-Design）

（3）SD（P-Programe）

* 结构化分析方法

基于数据流的分析方法

根据数据流分析的观点，系统模型的功能是数据变换，将软件系统抽象为一系列的逻辑加工单元，各单元之间通过数据流发生关联，逻辑加工单元接受输入数据流，并把它变换成输出数据流。

结构化分析的结果是：数据流图

* 数据流图

SA 方法最普遍采用的手段（建立系统模型）

数据流图：一种描述数据变换的图形工具

2．模型表示：

数据流图，数据字典，加工小说明

（1）数据流图：

四个基本成分

加工（动词，用圆圈表示　）：对数据进行处理、接收输入数据，进行处理后

产生输出数据。

数据流（名词 用箭头表示）：表示数据及其流向 通常由一组数据项组成

数据存储（文件 两条平行的线段）：表示信息的静态存储。

数据源和数据潭（短形表示）：

(2)数据字典：以准确的、无二义性的方式定义数据流和文件

数据流条目

数据项条目

文件条目

表示符号：

= 等价于（定义为）

+ 与（顺序结构）

{} 重复（循环结构）

[|] 或（选择结构）

() 任选

m..n 界域（范围）

(3)小说明：描述加工步骤

自然语言

结构化自然语言

判定表：

判定树

3．实施步骤

原则：自顶向下，逐步求精

手段：分解和抽象

① 确定系统边界，画出系统模型

② 自顶向下，画出各层数据流图

顶层

0 层：1、2、3……

父

0 层

子

1 层

③ 定义数据字典

④ 定义小说明

⑤ 汇总前面各步骤的结果

* 注意事项：

（1）平衡规则

① 使用正确符号，符号规范

② 数据字典完整

③ 正确符号

④ 最底层的加工必须写小说明

（叶节点）

⑤ 父图、子图要平衡

（2）控制复杂性规则

①

②

③

输入数据足够。

5． 实例研究

任何系统必须有有效性检查。

3．3 需求验证

重要性

1、 确保软件开发成功

2、 提高质量、降低费用

① 正确性（SRS）

② 无二义性

③ 完整性

④ 可验证性

⑤ 一致性

⑥ 可理解性

⑦ 可修改性

⑧ 可被跟踪性

⑨ 可跟踪性

⑩ 设计无关性

⑪ 注释

3．4 需求分析文档

1．SRS

SRS 重要性（作用）：

① 技术合同

② 开发的基础

③ 测试验收的依据

2．初步测试计划

3．用户系统描述（用户手册）

第 4 章 总体设计（概念设计、初步设计）

4．1 概述

1．软件设计阶段的任务

（1） 任务：确定系统“怎么做“

从 SRS 出发，形成软件的具体设计方案。

（2） 可采用的方法：

结构化设计方法（SD）——数据流

面向对象设计方法（OOD）

面向数据结构的设计方法

（3） 结构化设计

总体设计

详细设计

2．总体设计的主要任务

确定软件系统的整体模块结构

文档：模块结构图（MSD）—（Module Stucture Design）

（1）将系统划分成模块

（2）确定每个模块的功能

（3）确定模块间的调用关系

（4）确定模块间的接口

3．总体设计的表示形式

层次图

HIPO 图

结构图

4．2 结构化设计方法

与 SA 衔接使用

以 DFD 为基础得到软件的模块结构

DFD　　MSD

映射

1．数据流图（DFD）分类

（1） 变换型数据流图

定义：具有较明显的输入、变换（加工）和输出界面的数据流图

（2） 事务型数据流图。

数据沿着输入通路到达一个加工 T。T 将输入数据分解成一串发散的数据流，形成许多的活动路径，并根据输入数据的类型在若干动作序列中选出一个来执行。

事务中心任务：

① 接收输入数据。

② 分析输入数据，确定类型。

③ 根据事务类型选取一活动通路。

2．设计步骤：

（1） 变换设计：

确定逻辑输入

逻辑输出

通用步骤：

A． 复查基本系统模型

B． 复查并精化数据流图

C． 确定数据流图的类型

D． 把数据流图映射到软件结构，设计出模块结构的上层

E． 基于数据流图逐步分解高层模块结构，设计中下层模块（顶层、一层）

F． 对模块结构图求精，得到更为合理的软件结构

G． 描述模块化

D．顶层：设计一个主模块

基功能是控制，协调输入模块、变换模块、输出模块。

一层：

方法一：为每一个逻辑输入设计一个输入模块。其功能是向主模块提供输入数据。

为每一个逻辑输出设计一个输出模块。其功能是将主模块提供的数据输出。

为主加工设计一个变换模块。其功能是将逻辑输入变为逻辑输出。

方法二：设计三个模块

CI：输入控制模块

CT：变换控制模块

CO：输出控制模块

E

方法一：为每个输入模块设计两个下层模块

① 一个用于接收数据

② 另一个用于将这些数据转换为所要求的数据

①②重复直到物理输入为止

为每个输出模块设计两个下层模块

① 一个将调用模块提供的数据转换为所要求的数据格式

② 另一个用于输出数据

①②重复直到物理输出为止

一般每个基本加工设计一个功能模块。

方法二：为数据流图上每个基本加工设计一个模块

具体：在 DFD 上从变换中心的边界开始，从每个逻辑输入（出）出发，逆（顺）数据流方向沿逻辑输入（出）路径向外移动，把输入（出）路径上的每个加工映射成软件结构中 CI（CO）模块下的一个低层模块，直到物理输入（出）为止。最后将变换中心的每个加工映射成 CT 模块控制下的一个低层模块。

例：

（2） 事务设计

首先设计一个顶层模块（总控模块）

功能：接收事务数据，根据事务类型调度相应的处理模块。

下一层包括 2 个分支：

① 接收分支：负责接收数据并根据设计要求的格式实现输入数据的交流（同变换设计）

② 发送分支：包括一个调度模块，控制下层的所有模块

调度模块的下层按数据流图上画出的事务种类数目设计相应的事务处理模块，而每个事务处理模块的下层模块，则按数据流图特征映射成相应结构。

注意：

* 没有明显的事务特征，按转换模型进行

* 机械性遵循映射规则，可能得到一些不必要的模块，如果一个模块功能太复杂，可以分解成两个或多个模块

4．3 好的设计原则

模块（Module）：软件的组成部分，是具有独立功能且可命名的一段程序，所有模块组成 整体，可以满足问题的要求。

准则：

1．模块化（Modularity）:按照一定的原则，将软件划分成若干个模块，每个模

块完成一个特定的功能，然后把这些模块按照某种方法组装成一个软件系统。（可降低复杂度、减少工作量）

令 C(X):问题的复杂程度

E(X):解决 X 问题所需的工作量

规律 1：若两个问题 P1，P2

C（P1）>C(P2),则 E(P1)>E(P2)

规律 2：某问题 P 可以分成 P1，P2，P＝P1＋P2

则 C(P1+P2)>C(P1)+C(P2) 所以：E(P1+P2)>E(P1)+E(P2)

2．抽象（Abstraction）:在解决问题的过程中，集中考虑和当前问题有关的方面，而忽略与当前问题无关的方面。

3．信息隐蔽（Information Hiding）：隐藏功能实现的细节，即在设计和确定一个模块时，使一个模块内包含的停息（过程、数据），如果它不允许外部模块访问的话，则其它模块是不能访问的。

4．模块独立性：一个模块同其它模块的独立程度，是评价一个设计好坏的一个重要度量尺度，是保证软件质量的一个关键因素。

＊ 有两个定性的度量指标：

① 内聚（Cohension）：又称为块内联系，指模块内部各成分之间相互关联的程度

追求高内聚

内聚类型：由 A 到 G 内聚性逐渐增强

A、 偶然内聚：一个模块各个成分之间毫无关系

B、 逻辑内聚：将几个逻辑上相关的功能放在同一个模块中

C、 时间内聚：一个模块完成的功能在同一时间执行

D、 过程内聚：一个模块内部的处理成分是相关的，而且必须以特定的次序执行

E、 通信内聚：一个模块的所有成分都集中在同一个数据结构上

F、 顺序内聚：一个模块的各个成分同一个功能密切相关，而且一个成分的输出，作为另外一个成分的输入

G、 功能内聚：模块内的所有成分属于一个整体，完成单一的功能。（内聚最高）

② 耦合（Coupling）

块间联系：对模块之间依赖程度的度量。

* 耦合程度依赖以下因素：

① 一个模块对另外一个模块的调用

② 一个模块向另外一个模块传递数据

③ 一个模块施加于另一个模块的控制

④ 模块之间接口的复杂程度

* 耦合类型（追求低耦合）（从 A 到 F 耦合逐渐降低）

A、 内容耦合：一个模块直接操作或修改另一模块的数据，或者不通过正常入口直接转入　　另一模块

B、 公共耦合：两个或多个模块通过共同引用一个全局数据环境相互作用

C、 控制耦合：模块之间通过传递控制信息相互作用

D、 标记耦合：两个模块之间通过传递公共指针或地址相互作用的耦合

E、 数据耦合：模块之间通过传递数据交换信息

F、 无耦合：模块间无任何关系，独立工作

追求“高内聚，低耦合”方法：

① 设计有独立功能的模块

② 模块间传递数据型的参数

③ 模块间共享信息尽量少

4．4 启发式规则

（模块化设计原则，软件结构优化原则）

目的：改善软件结构，提高软件质量

1．改进软件结构，提高模块独立性

首先设计出软件初始结构，评价该结构，通过模块分解或合并，力求降低耦合提高内聚。

2．模块的规模应该适中

3．模块结构的深度、宽度、扇出和扇入应适中

深度：软件结构中控制的层数

宽度：软件结构中同一层次上最大模块总数

扇入：某一模块有多少直接调用它的上级模块数目（越大越好）

扇出：一个模块直接控制（调用）下级模块的数目。（越少越好，3，4 个为宜，不超过 9 个，“　　原则”）

“顶层扇出较高，中间扇出较小，底层模块高扇入”

4．一个模块的作用域（范围），应处在这个模块的控制域（范围）之内

* 模块的作用域：受该模块内一个判定影响的所有模块的集合

* 模块的控制域：这个模块本身以及所有直接或间接从属于它的模块的集合

例 1：　　例 2 ：

例 3：　　例 4：

注：“◇”表示判定条件

作用域不在控制域内的修改方法：

① 将判定点上移到足够高的位置

② 将那些在作用域内但不在控制域内的模块移到控制域内

例：

作业：某学校对各科学习成绩统计，若平均分良好以上给予奖励，奖励方案如下：

如果学生优秀课程＞＝85％，发给一等奖学金，发给优秀学生奖状。

如果良好，发给二等奖学金，发给给三好学生奖状。

计算奖金：如果“各科成绩”＞＝“良好”并 且“课程成绩优秀率”＞＝85％

　　那么，置“等级标记”＝一等，调“计算一等奖学金“模块

　　否则，置“等级标记”＝二等，调“计算二等奖等金”模块

分发奖状：如果“等级标记”＝一等，则调“分发优秀学生奖状”模块

否则，调“分发三好学生奖状”模块。

问题：作用范围不在控制范围内，请改正。

5．力争降低模块接口的复杂性

6．模块功能应该可以预测

4．5 优化

4．6 设计文档－软件设计说明书（复审）

补充：DFD　MSD（方法一）

例：

DFD：

MSD:

DFD:

作业：

DFD　MSD

第 5 章　详细设计

5．1 概述

1．详细设计（模块设计）

根据得到的模块结构图 MSD，对其中每一个模块给出过程属性的描述，即算法设计。

即确定每个模块的内部特征，即每个模块内部的执行过程（怎样做）

目标：

① 确定怎样具体的实现所要求的系统，提供关于算法的更多的细节。

② 根据详细设计的结果可直接用于编程的程序逻辑结构。

2．与总体设计的不同点

详细设计以总体设计为基础，但不同于总体设计的是：

（1）总体设计阶段：

确定软件的模块结构、接口描述

每个模块是个黑盒子

数据项和数据结构以比较抽象的方式描述

（2）详细设计阶段的根本目标：

针对每一个模块，确定具体的算法（过程）细节

3．详细设计阶段的工作

实现对应总体设计的模块所需要的处理逻辑，主要有：

① 详细的算法

② 数据表示和数据结构

③ 实施的功能和使用的数据间的关系

5．2 结构化程序设计（SP）

详细设计阶段采用的方法是 SP，与 SA，SD 方法衔接。

由于详细设计的目标是给出可以直接用以编码的程序逻辑结构，因而研究良好而规范地表示程序逻辑结构的技术是人们关注的焦点。

E.W.Dijkstra 60 年代提出结构化程序设计概念

1966 年，C.Bohm 和 G.Jacopini 证明：只用三种基本控制结构就能实现任何单入口单出口的程序。

三种基本控制结构（共同特征：严格地只有一个入口和一个出口）

“顺序”、“选择”和“循环”

5．3 详细设计工具

由于是分别考虑每个模块，所以问题地规模已大大缩小；又因为有了结构化程序设计的方法，一般来说，详细设计的难度已不大。关键是用一种合适的表示方式来描述每个模块的执行过程，这种表示方式应该是简明而精确，并由此能直接、机械地导出用编程语言表示的程序。

由于详细设计的任务：给出软件模块的内部过程描述（模块内部的算法设计），我们只关心算法的表现形式。

* 表示方法分类

① 图形：（程序流程图、IPO 图、PAD 图、WLD 图、N－S 图、判定树）

② 表格：（判定表）

③ 语言：（伪码 psudo-code）

* 要求：无二义性描述

即能指明控制流程、处理功能、数据组织以及其他方面的细节，从而在编码阶段能把设计描述直接翻译成程序代码。

1．程序流程图 P60

历史最悠久，使用广泛

用的混乱

优点：直观、形象化、易于理解、易于进行复审，便于初学者掌握

主要符号（三种基本结构，通过嵌套方式可以导出更复杂的逻辑结构）

2．N－S 图（盒图）：一定是结构化的

主要符号：

3．PAD 图（问题分析图）

二维树形结构图

主要符号：

主要优点：

（1）

（2）

（3）

（4）

（5）

（6）

PAD 图是面向高级语言的，FORTRAN，COBOL，PASCAL，C 都有相应的图形符号，所以将 PAD 图转换成对应的高级语言程序比较容易

将下面的程序结构图转换为 PAD 图：

4． 类程序设计语言（PDL）

(Program Design Language)—又称伪码

用正文的形式表示数据结构和处理过程的设计方法

是一种混合语言

既有：严格的关键字、外部语法（用于定义控制结构和数据结构）

C，PASCAL，借用某种结构化的程序设计语言的语法控制框架

又有：使用一种语言的词汇（灵活自由地表示实际操作判定条件）

某种自然语言，英语/汉语

使用一种语言的词汇，同时又使用另一种语言的语法

特点：（书 P63）

优点：（书 P68）

缺点：不如图形工具形象直观

（例如，描述复杂的条件组合与动作间的对应关系时，不如判定表或判定树清晰简单）

5．判定树

6．判定表

① 程序流程图优缺点

② N－S 图

③ PAD 图特点　　要求掌握

④ PDL 语言（伪码）　　相互转换

⑤ 判定树

⑥ 判定表

将非结构化的程序流程图转化为结构化的程序流程图

转换方法：

（1）编号

a． 从程序入口外向下遇到的第一个 p 节点或 d 节点，标号 1

b． 程序的出口处标号为 0

c． 对于关键性的 p 节点或 d 节点（即该节点上面是一个 IC 节点（即多入口或

多出口的地方）或 dc 节点）标号，从 2 开始顺序编号

（2）引入一个状态变量 I

I＝0~N (0,1,2,…,N　N 为最大标号数)　（下例中 N＝3）

（3）

① 在流程图入口处增加一个 p 节点（处理框），执行 I：＝1 操作，然后把整个流程图变成一个 N 个分支的分支结构

② 第 I 个分支第一个节点即是标以 I 的 p 节点或 d 节点，然后照画其余的图，当遇到一个标以号码 N 的节点时，增加执行 I：＝N 操作的 p 节点，并立即结束该分支

③ 在上面分支结构的外面增加一个 Repeat 循环

例：

5．4 详细设计文档

详细设计说明书（包括模块开发卷宗）

详细设计说明书编写的目的：尽可能详细地说明软件所包含的程序中，各成分的设计考虑，以利于程序员编制程序

详细设计文档的主要内容：

① 对程序进行整体描述，包括该程序的各项功能和性能

② 详细说明程序的运行环境

③ 说明程序运行过程，包括程序的装入、启动、停机、恢复

④ 详细描述程序的输入、输出和所用的数据环境，主要用图表的形式描述程序的逻辑流程，并加以叙述

（逻辑流程详细描述程序的处理过程，对每个程序功能列出程序运行说明，描述包括算法逻辑、数据操作与逻辑判断处理在内的一切处理，还应详细解释用于转移的测试条件，标出条件和由程序进行出错处理的方法）

有相应的规范

5．5 补充：编码

1．概述

编码是在详细设计阶段的基础上完成的

（1）主要任务：

根据详细设计阶段给出的程序逻辑结构的描述，选择某种语言，按照编码规范，编写出高质量的、具有一致性、易移植性、易维护性、高效率的程序代码。前面软件工程的各个步骤，都是为了一个最终目标：

把对软件的描述翻译成一种计算机可以“识别”的形式——计算机程序

（2）两点对编码的要求：

a. 编程的方法应简单易用

使用户也成为编程人员

b. 大力提高编程的生产率

利用工具或环境，考虑软件复用问题

使用程序设计环境和各种代码自动生成工具，使用软件复用技术

2．程序语言的选择与编码风格

（1）程序的质量涉及的因素

* 软件设计
* 与所选语言的特性有关
* 与编码风格有关

（2）程序语言的选择

* 语言类别
* 应用领域

（3）编码风格

① 写清楚 ② 直接说明用意 ③ 用库函数 ④ 变量名 ⑤ 避免不必要的转移 ⑥ 三种结构⑦ 避免用空 else ⑧ 每个模块做一件事 ⑨ 加注释

第 9 章　软件测试

9．1 软件测试目标与软件测试过程模型

1．软件测试目标

（1）定义：IEEEE 标准

使用人工或自动手段运行或测定某个系统的过程，其目的是检验该软件（系统）是否满足规定的需求，或是清楚地了解预期与实际结果之间的差异。

简述为：按照特定规程，发现软件错误的过程。

“软件测试只能发现错误，不能证明软件没有错误“

（2）测试目标

① 测试是程序的执行过程，目的在于发现错误

② 好的测试方案在于发现至今尚未发现的错误

③ 一次成功的测试是发现了至今尚未发现的错误的测试

2．软件测试与软件调试

软件测试：发现软件中有错误的迹象，并不知道错误发生的位置和发生错误的原因。

软件调试：在测试以后，纠正错误所做的工作（找到错误并纠正错误所做的工作）

① 确定错误的确切位置

② 修改错误

* 两者的区别：书 P132

* 常用的软件调试辅助方法：

① 对计算机的工作过程进行模仿或跟踪

② 设置语句

③ 逐层分块调试

3．软件测试过程模型

软件测试是一个有规则的过程：

包括测试设计、测试执行、测试结果比较

* 软件测试过程模型图，书 P133

* 错误分类：

① 结构错误

② 数据错误

③ 编程错误

④ 接口错误

4．软件测试的原则

① 设计测试用例时，要给出测试的预期结果

② 开发小组与测试小组分立

③ 要设计非法输入的测试用例

④ 在对程序进行修改之后，要进行回归测试

回归测试：为了保证对软件所做的修改没有引入新的错误，而重复过去已经进行过的测试

⑤ 在进行测试时，要集中测试容易出错的程序段

⑥ 除了检查程序是否做了应该做的工作，还要检查程序是否做了不应该做的工作

9．2 软件测试技术

（1）概述

两大类：

* 白盒（箱）测试：测试软件的内部活动是否符合设计要求，测试人员需要了解软件内部结构，对其所有逻辑路径进行测试。

* 黑盒（箱）测试：测试软件的功能是否达到预期的要求。测试主要着眼于软件的外部特性，看输入是否能正确接收且能否正确地输出结果。

功能、输入、输出

测试用例设计

* 测试用例：以发现错误为目标的用于进行软件测试的输入数据，以及与之对应的预期输出结果。

* 设计目标：确定一组最可能发现某个错误或某类错误的测试数据（即发现错误有较多的概率）

逻辑覆盖测试技术

是一种白盒测试法，考虑程序内部逻辑覆盖程度。

① 控制流程图：是程序控制结构的图形表示

② 路径：是一串指令，有一个入口，一个出口

③ 语句覆盖：选择足够的测试用例，使程序中的每条语句至少执行一次

④ 判定（分支）覆盖：设计足够的测试用例，使得判定中每个分支至少执行一次

⑤ 条件覆盖：设计足够的测试用例，使得判定中的各个条件的所有可能（不要求条件组合）情况都至少执行一次

⑥ 路径覆盖：执行所有可能的穿过程序的控制流程路径

* 根据流程设计测试用例

例：设计测试用例

黑盒测试法及测试用例设计

（1） 等价类划分法

* 首先，划分等价类，根据每一个输入的条件，找出两个（合理的等价类和不合理的等价类）或更多的等价类

* 为每一个等价类设计测试用例

例：

① 如果某个输入条件说明了输入值的范围，则可划分出一个合理的等价类和两个不合理的等价类

分数：10－100 1 个合理的等价类

＜10＞100 2 个不合理的等价类

② 某个输入条件说明了输入数据的个数，则可划分为一个合理的等价类和两个不合理的等价类

选课：1－3 门 1 个合理的等价类

＜1 门

＞3 门 2 个不合理的等价

类

③ 一个输入条件说明了必须成立的的条件，划分为一个合理等价类和一个不合理等价类

文件名的第一个字符必须是字母：是字母 1 个合理等价类

不是字母 1 个不合理等价类

设计测试用例的步骤：

1） 为每个等价类编号

2） 设计一个新的测试用例，它能包括尽可能多的尚未包括的合理的等价类，重复 2）步，直到包括了所有合理等价类

3）设计一个新的测试用例，它能包括且仅包括一个尚未包括的不合理的等价类，重复 3）步，直到包括了所有不合理等价类

（2） 边界值分析法

经验表明：程序往往在处理边界情况时发生错误

测试用例设计：

正好等于，大于或小于这类的数据

9．3 测试步骤

单元测试（模块测试）、集成测试、有效性测试、系统测试

1．单元测试

目的：检查模块是否能独立地正确执行

测试依据：详细设计文档

（1） 考虑问题： 模块接口

数据结构

“重要的“执行路径

错误处理

边界值情况

（2）实施

驱动模块的作用是模拟 H 的调用模块，它接收不同测试用例的数据，并把这些数据传送给测试模块 H，最后打印或显示结果

支持模块的作用是模拟被测试模块的下属模块

2．集成测试（整体测试）

在单元测试的基础上，考虑将模块集成为系统的过程中可能出现的问题，最终构成所要求的软件结构

集成测试的两种方式：

① 自顶向下组装模块

② 自底向上组装模块

（1） 自顶向下集成测试

从主控模块开始，沿控制层次向下，或先深度，或先宽度地逐一将模块组合起来，形成与之相符的软件结构

优点：与承接模块相关联的问题可能被尽早的测试；不需要设计驱动模块

缺点：要设计承接模块

（2） 自底向上集成测试

从软件结构最底的一层开始，逐层向上地组合模块并测试

优点：比较容易设计测试用例；不需设计承接模块

缺点：直到最后一个模块加进来，才能看到一个完整软件结构

3．有效性测试

其测试目标：是发现软件实现的功能与需求规格说明书不一致的错误

在测试之前，要做配置复审

4．系统测试

软件是计算机系统的一个组成部分，把软件与其它系统的组成元素结合起来，并进行一系列系统集成测试和有效性测试

9．4 补充：软件维护

1．维护工作

工作量大： 排除错误

增加新功能（新版本）、性能调优

老版本

2．维护的内容（分类）

（1）校正性维护（改正性维护）：针对软件诊断和改正错误的过程

（2）适应性维护：为了适应新的变化而对软件的修改过程

（3）完善性维护：为了满足或部分满足用户提出的新功能而进行的开发

（4）预防性维护：为进一步改进软件的易维护性和可靠性，或者进一步改进，提供

更好的基础而对软件进行的修改

3．导致软件维护困难的因素

易维护性：软件开发方法，合适的文档

开发条件

4．维护的副作用：引进新错误

第 10 章 软件过程

软件过程的定义：是软件生命周期中的一系列相关过程

过程：是活动的集合

活动：是任务的集合

任务：是将输入变换为输出的操作

研究内容：软件生产和管理

软件过程的几种观点：工程观点、系统观点、管理观点、运行观点、用户观点

* 软件过程分类：

按照人员的工作内容分类：1

基本过程、支持过程、组织过程

剪裁过程1

10．1 基本过程

指那些与软件生产直接相关的过程

包括：获取过程---获取者---为得到软件系统或软件产品所进行的一系列活动（合同）

供应过程---供应者---为获取者提供软件产品的一系列活动

开发过程---软件开发者---软件开发者所进行的一系列活动

运行过程---用户和操作人员---在用户的业务环境中为使软件产品投入运行所 进行的一系列有关的活动

维护过程---软件维护人员---软件维护人员所从事的一系列活动

10．2 支持过程

支持过程是有关各方按他们的支持目标所从事的一系列相关过程

目的：提高软件产品的质量

文档过程：是一个记录某一过程或活动所产生信息的过程

配置管理过程：

质量保证过程：为使软件产品符合所规定的需求，并按所制定的计划完成，提供适当保证的过程。

验证过程：

确认过程：

联合评审过程：

审计过程：

问题解决过程：

10．3 组织过程

是指那些与软件生产组织有关的过程

管理过程：---管理者---在软件生命周期过程中，管理者所从事的一系列活动

基础设施过程：建立、维护任何其它过程所需的基础设施的过程

改进过程：建立、评估、度量、控制和改进软件生命周期过程的过程

培训过程：为软件产品提供人员培训的过程

10．4 剪裁过程

目的：为了有效地实施软件过程，应针对特定领域的软件工程，对选定的过程模型和标准进行剪裁，以形成这一工程的模型及标准，形成该工程的各个软件过程和活动。

剪裁过程：是一类软件过程，是对软件过程和活动实施剪裁的过程

第 12 章　计算机辅助软件工程 CASE

目标：提高开发效率、提高软件产品质量

12．1 CASE 综述

1．　CASE 综述

CASE 是一组工具和方法的集合，可以辅助软件生命周期各阶段进行软件开发

2．　CASE 技术分类

（1）　从软件开发和管理角度分

* 支持软件开发过程本身的技术

* 支持软件开发过程管理的技术

（2）　从 CASE 系统产生方式分

元 CASE 技术---分成 CASE 系统的生成器所采用的技术

3． CASE 工具

软件工具：可用于辅助或支持计算机软件的开发、运行、维护、模拟、移植或管理的程序系统

CASE 工具：是一种特殊的软件工具，用于辅助开发、测试、分析（或维护）另一个计算机程序和文档（狭义）

除了操作系统之外的所有软件工具的总称（广义）

4． CASE 系统分类

（1） 支持单个过程任务的工具

（2） 工作台：支持某一过程或某些活动（工具集）

（3） 环境：支持软件过程的所有活动或大部分（工作台集合）

* 集成化环境：对数据、控制、表示集成提供基本功支持

Wesserman 五级模型

平台集成、数据集成、表示集成、控制集成、过程集成

11．2 CASE 工作台

一个 CASE 工作台是一组工具集，支持设计、实现、测试等特定的软件开发阶段，能支持大多数软件过程活动

1． 分类

① 程序设计工具台

② 分析和设计工具台

③ 测试工作台

④ 交叉开发工作台

⑤ 配置管理工作台

⑥ 文档工作台

⑦ 项目管理工作台

2． 开放式和封闭式工作台

11．3 软件工程环境定义

（书 P197）包含有一组软件工具，这些工具按照一定的方法或模型组织起来的，这些工具支持整个软件生存周期的各个阶段或部分阶段

软件工程读书笔记

软件工程过程

(一) 规格说明

(二) 软件开发

(三) 软件确认

(四) 软件演进

二、软件生存期

(一) 计划时期

1. 问题定义：目标及范围说明书

2. 可行性研究：可行性研究报告

(二) 运行时期

1. 需求分析：需求分析说明书

2. 设计：设计文档

3. 编码：程序

4. 测试：测试报告

(三) 维护时期

1. 运行及维护：目标与范围说明书

三、软件生存期模型

(一) 瀑布模型

线形生存期各阶段的模型

(二) 循环模型

回溯，维护阶段经理的循环（瀑布循环）

(三) 增量模型

非整体开发模型：分瀑布的渐增和圆形的快速模型

(四) 螺旋模型

1. 将瀑布和增量结合，加入风险分析形成螺旋模型

2. 每个螺旋周期分

a) 确定目标、方案和限制条件

b) 评估方案、标示风险和解决风险

c) 开发确认产品

d) 计划下一周期工作

四、软件开发方法

(一) 结构化方法

1. 面向功能或面向数据

2. 在设计、分析、编程三阶段都利用结构化

3. 两类典型结构：变换性、事务型

(二) 面向数据结构的软件开发方法

1. Ｊａｃｋｓｏｎ方法把问题分解为顺序选择和重复

2. 三种基本结构形式表示的各部分的层次结构

3. 这一方法从目标的输入、输出数据结构入手，到处程序邝杰结构，在补充其它细节，得到完整的程序结构图。

4. 它也可以与其他方法相结合，用于模块的详细设计

(三) 面向问题的分析法

1. 考虑输入、输出数据结构，直到系统分解，在系统分析下逐步综合

a) 从输入、输出数据结构到处基本处理框；

b) 分析处理框间的先后关系

c) 按先后关系逐步综合处理框

(四) 原型化方法

1. 需求不清、业务理论不确定，规模不大时采用

a) 确定需求

b) 开发原型

c) 征求意见

d) 修改完工

(五) 面向对象的软件开发方法

是自底向上，和自顶向下相结合的一种方法，通过对象模型的建立，能真正基于用户的需求，且可维护性大大改善。

(六) 可视化开发方法

五、软件工程的基本目标

(一) 在确定的时间内成功开发高质量的软件系统。

六、软件开发工具和开发环境

(一) 软件工具由工具、工具接口和用户接口三部分构成。

七 、需求分析

(一) 需求获取：与用户交流

(二) 需求建模：抽象描述

(三) 形成需求规格：精确形式描述

(四) 需求验证：以需求规格说明为输入，通过符号执行，模拟或快速原型，分析需求规格的正确性和可行性

(五) 需求管理

八、软件需求的三个层次

(一) 业务需求：客户对系统的高层次目标要求

(二) 用户需求：用户使用产品必须要完成的任务

(三) 功能需求：开发人员必须要完成的软件功能，使用户完成任务实现业务需求。

九、Ｂ／Ｓ需求分析

(一) 需求分析及变更管理

(二) 项目模型及业务流程分析

(三) 系统分析及软件建模

(四) 界面设计、交互设计及程序开发

(五) 系统测试及文档编写

(六) 客户培训、技术支持及售后服务

(七) 注意事项：

1. 畅所欲言
2. 分析潜在需求
3. 自然语言描述
4. 利用图表表现需求
5. 确认需求分析各角色的阅读需要。

a) 项目经理

b) 系统分析员

c) 开发经理

d) 交互设计师

e) 测试人员

f) 文档人员

g) 客户代表

6. 建立需求变更日至
7. 本阶段重点工作角色：客户代表，业务员，项目经理

十、编写需求文档

(一) 需求开发的成果

1. 项目视图、范围文档
2. 使用实例文档
3. 功能需求文档
4. 非功能需求文档：质量属性和外部接口

(二) 编写需求规格说明的方法

1. 结构化的语言：文本
2. 图形化模型：过程、状态、数据、逻辑、对象
3. 形式化规格说明：数学逻辑语言

(三) 技巧：

1. 序列号

2. 层次化编号

3. 无不确定语句（模糊，比较）

4. 五复合需求（要分开）

十一、 软件文档分类

(一) 文档编制要求：

1. 有针对性

2. &nb sp; 精确性

3. 清晰性

4. 完整性

5. 灵活性

6. 可追溯性

(二) 1 3 种文档：

1. 可行性分析报告

说明项目在技术、经济上、社会因素上的可行性，说明达到目标可选的方案，并说明理由

2. 项目开发计划

个部分工作的负责人员，开发进度，经费预算，软硬件资源

3. 软件需求说明书

功能，性能，用户界面，运行环境

4. 概要设计说明书

功能分配，模块化粪，程序总体结构，输入输出及接口设计，运行设计，数据结构设计，出错处理设计。

5. 详细设计说明书

每一模块的实现：算法，逻辑，流程

6. 用户操作手册

描述软件，如何使用

7. 测试计划

测试内容，进度，条件，人员，测试李子的选取原则，测试结果允许的偏差范围。

8. 测试分析报告

测试计划执行得说明，结果的分析，测试的结论意见

9. 开发进度月表

进度计划于实际比较，阶段成果，问题与解决，下月打算

10. 项目开发总结报告

开发完成后，于实施计划对照，总结进度，成果，资源利用，成本和投入的人力，评价，经验教训。

11. 软件维护手册

软件基本说明，支持软件的说明，维护的说明

12. 软件问题报告

软件问题的时间，发现人，状态，问题所述模块

13. 软件修改报告

问题，修改考虑，影响。

十二、 测试

(一) 界面，样式，错字，语句不通，相混或相近

(二) 功能

(三) 需求

(四) 性能

(五) 其它：多系统融合，少模块