《软件工程基础》期末复习

一 绪论

1、软件工程概述

软件工程这门学科的主要研究内容包括：

（1）**软件开发的理论。**任何一门学科发展到一定程度，都会形成一定的理论。软件开发的理论当然是整个软件开发实践的一些经验总结，并且是从这些经验中抽象出来的关于软件开发的一些基本概念、原则、思想等等。

（2）**软件开发的方法。**在一些公认的理论的指导下，不同的软件开发人员和研究人员，都提出了各种各样的具体方法，这些方法各有优势，也各有不足。但是大多数都已经在实际运用中取得了比较好的效果。

（3）**软件开发的标准。**任何一个产业，要能够实现工业化的、大批量的生产，必需有一定的大家共同承认和共同遵守的标准。在技术领域，标准的来源主要有两种，一种是由各种标准化组织制定的（包括国际的、国家的、行业的），另一种是由在这个产业里居主导地位的生产厂商事实上形成的，当然，这种标准也会根据市场情况逐渐地被标准化组织所采纳。

2、软件的定义和特点

软件是指计算机系统中与硬件相互依存的另一部分，包括程序、数据及相关文档的完整集合。

软件的主要特点为：

（1）软件不是物理实体，而是一种特殊的逻辑实体；

（2）软件的开发和设计过程是以人为主的，是通过设计将人的智力成果转化成软件；

（3）软件设计的结果是没有误差概念；

（4）软件一旦开发成功，它不会随着时间的流逝老化；

（5）软件的开发过程和生产过程是统一在一起的，开发过程的结束也就是生产过程的结束；

（6）软件完成后有一个较长时期的维护工作。

3、软件工程的产生背景：

由于缺乏对软件开发工作的研究，缺乏正确的理论和方法导致的“**软件危机**”。软件危机主要表现为：

（1）软件的质量难以达到用户的要求，并且极易发生故障，可靠性差；
（2）开发一个软件的成本和进度难以预测，也难以控制；
（3）开发出来的软件维护困难；
（4）软件开发工作难以管理。
（5）软件的成本逐年上升。

对于软件系统的认识、开发等形成的错误观念，也是造成“软件危机”的原因：
（1）开发软件就是编程。
（2）只要向用户提交了程序，并且可以运行，软件开发工作即告结束。
（3）软件是灵活的，可以很容易修改。
（4）增加人员可以加快进度。

4、软件工程的定义和基本目标：

“软件工程”的定义（IEEE）：将系统的、规范的可度量的工程化方法应用于软件开发、运行和维护的全过程及上述方法的研究。

软件工程研究的目标：在有限的时间、人力、物力、财力的基础上，生产出符合用户需求的软件产品，提高软件生产率。

软件工程的组成：方法、工具和过程。

5、软件工程的七条基本原理：

任何一门学科，例如数学、物理等学科都有一些基本的原理，实际上就是一些公理。软件工程的 7 条基本原理是：

（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）采用现代程序设计技术；
（5）结果应能清楚地审查；；
（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。

二　软件工程的生命周期方法学

1、　软件生命周期定义和阶段划分：

软件工程认为：任何一个软件，都有产生、发展、消失（灭亡）的过程。这个过程就叫做软件的“生命周期”。软件工程的各种理论和方法都是以软件的生命周期为基础。

软件生命周期的阶段划分为：需求分析（问题定义和可行性分析）、设计（概

要设计、详细设计）、实现（编码）、测试、维护（灭亡）。

2、软件工程生命周期方法学的定义和特点

软件工程的生命周期方法学是指严格按照软件的生命周期，采用分阶段、有计划和控制，以及顺序实施的步骤，以**结构化分析与设计**（Structured Analysis，**SA**； Structured Design，**SD**）或**面向对象分析与设计**（**Object-Oriented Analysis，OOA**；**Object-Oriented Design，OOD**）为主的软件开发过程。

生命周期方法学的特点是：

（1）分阶段计划；

（2）瀑布式开发模式；

（3）阶段性的技术审查和管理复审——基线；

（4）在各个阶段采用结构化技术 / 面向对象技术。

结构化技术是指采取自上而下，逐步求精，单入口，单出口的模块化的分析与设计的模型。

面向对象技术是指在客观世界映射到信息世界时，采用人们认识客观世界的过程中普遍运用的思维方法，直观、自然地描述客观世界中的有关事物——类与对象，并通过抽象性、封装性、继承性、多态性和消息机制等设计类与对象，并建立他们之间关系的分析与设计的模型。

3、软件工程生命周期方法学的阶段划分：

（1）需求分析（问题定义、可行性分析）

（2）概要设计

（3）详细设计

（4）实现（编码）

（5）测试

（6）维护

按照软件工程思想，采用生命周期方法学，则上述的某些阶段，可以简化和合并，但是不能够完全取消，或者破坏瀑布模型的从上至下的流程。

三 需求分析

1、问题定义

问题定义阶段的主要任务是：明确要解决的问题究竟是什么。

（1） **问题描述**：系统分析员与用户方的项目负责人交谈，并对现场情况进行调研，然后根据所收集到的资料，描述软件系统开发的目标。一方面交用户审阅，一方面与用户的项目负责人、使用负责人、其他有关人员进行详细的讨论，澄清双方认识上不一致的地方和报告中描述模

糊的内容。

（2） **可行性分析：**对问题定义中确定的开发目标，进行技术可行性、操作可行性、经济可行性和法律可行性等方面的综合分析，形成可行性分析报告。

2、需求分析

需求分析的主要任务是分析出软件开发目标中的所有具体要求，主要包括：

（1）**功能需求：**系统实现什么功能？

（2）**性能需求：**包括速度、存储、数据量、安全性和可靠性等指标。

（3）**领域需求：**包括与领域相关的保密性、可扩展性等。

（4）**其它需求：**如预防性维护需求、版权、政策等。

3、需求建模

（1）问题定义：功能和性能（约束条件），注意问题说明和定义的一致性和完整性；

（2）结构化分析方法

按照结构化的思想，采取从总到分，逐步求精的过程。

按照面向对象的思想，采取类-对象分析、属性、方法、服务、主题层分析。

（3）需求建模：数据建模、功能建模、行为建模

（4）修正开发计划

对在可行性研究阶段制定的开发计划进行仔细的修订，对人员的配备、设备的要求、进度安排、费用分配都进行进一步的研究。

4、需求分析阶段使用的图形工具

（1）数据流图

数据流图（DFD）纯粹是从系统数据流动的角度来描绘系统的逻辑模型。

数据流图的基本符号只有数据的源点/终点、变换数据的处理、数据存储和数据流线四种。强调一点：DFD 中带箭头的线段表示数据的流动，而不是控制的转移。

（2）数据字典

数据字典是表达系统逻辑模型中所有数据元素详细情况的工具，它严格定义整个系统逻辑模型中所涉及到的基本数据、数据结构、文件形式，每一个元素作为一个词条，在词条解释中对该元素的定义、描述、使用特点、与其他数据元素的关系等等都进行了准确详细的表达。

数据字典的形式定义包括：表格、范式（定义式）等形式。

在编辑数据字典的时候，要注意这样几个问题：

- **完整性**：在 DFD 中的数据流名称在 DD 中必定有相应的名称。
- **一致性**：在 DFD 中的数据流名与在 DD 中定义的数据语义必须一致。
- **正确性**：数据定义真实描述用户领域的数据、结构、文件等信息。

四 软件设计基础

1、软件设计的概念和原理

模块的独立性由模块化、抽象和信息隐藏共同体现。

（1）模块化

模块化是结构化设计的基本理论之一。**模块是指具有输入、输出、逻辑功能、内部数据这四个基本属性的指令集合。**模块化的目的就是为了降低软件自身的复杂度，减少开发所需的工作量。在模块数量适中的**某一个值**时，软件开发的总工作量最少，成本也最低。

（2）抽象

抽象是一个方法论的名词，抽象是指从客观事物中抽取它们共同特征，并对其进行概括和总结；对于它们在其他细节上的差异暂时忽略。在软件设计中，采用抽象的方法也就是采用结构化的思想。

（3）信息隐藏和局部化

信息隐藏：每一个模块内部与其他模块无关的数据，其他模块都没有办法访问；每个模块只完成一个相对对立的功能；模块之间交换的仅是为了实现系统功能而必须交换的信息。

局部化：是指应该把可能互相作用的过程和数据尽量放在一个模块内。

信息隐藏和局部化的意义都在于降低模块之间接口的复杂度，提高模块的独立性。

（4）模块独立

模块之间的独立程度的度量方式是模块的耦合和内聚。

1、耦合

耦合是对一个软件结构内不同模块之间互连程度的度量。按照由弱至强的顺序，模块间的耦合关系有这样几种：

- **非直接耦合**：模块之间没有直接的调用关系；
- **数据耦合**：信息交换仅发生在接口上；
- **特征耦合**：交换的信息是一个复合的数据结构；
- **控制耦合**：交换的信息是控制信息；
- **公共耦合**：交换的公共的数据环境；
- **内容耦合**：模块之间的叠加。

对于良好的结构化设计，应该是：尽量使用数据耦合、少用控制耦合，限制公共耦合范围，坚决不用内容耦合。

2、内聚：

内聚是指一个模块内的处理和数据之间的关联程度。一个模块的内聚方式按照内聚程度的由低到高排列，有以下几种：

- **偶然内聚**：模块内部是松散的关系（如节约内存等）；
- **逻辑内聚**：逻辑上相关，例如都是输入或者输出操作，用参数区别；
- **时间内聚**：在同一时间内上运行，或者在逻辑上有先后顺序；
- **过程内聚**：按照过程描述自上而下组织任务，如“打开 / 关系”
- **通信内聚**：模块内部操作同一数据区域 / 结构；
- **顺序内聚**：同一功能、按顺序执行；
- **功能内聚**：模块是一个整体，完成一个独立的功能。

2、启发式规则

（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、扇出和扇入都应该适中；
（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。
“先让系统正确运行起来，再让系统好起来！”

五、总体设计

1、总体设计的目标

总体设计的目标从根本上来说，就是要确定软件的**总体实现方案**和**总体结构。**
总体设计分为：结构设计、系统数据设计和系统接口/界面设计。

2、总体设计的步骤

（1）设想供选择的方案；
（2）选取合理的方案，要选择**多种方案**：低成本、中等成本和高成本的。成本是综合因素的考虑，包括：技术、资金、人力、物力、时间、空间、管理等。
（3）推荐最佳方案
（4）功能分解
（5）设计软件结构
（6）数据设计
（7）制定测试计划
（8）书写文档

3、图形工具

- **结构图（表示模块间数据的传递）**

结构图是层次图的一种改进。它的线条直接标示在模块之间，而且在调用线旁边使用带箭头的线段来表示模块之间传递的信息。

4、面向数据流图的设计方法(事务流，变换流)

(1) 事务流和变换流的形态；
(2) 变换流的映射方法：自动化边界的划分；
(3) 事务流的映射方法：事务中心的确认；

六、详细设计

1、详细设计的任务、方法

详细设计阶段的主要任务是对每一个模块的内部细节进行具体的设计，实现所有软件的功能。

按照什么顺序去完成每一个模块内部细节的设计。

2、结构化程序设计

（1）基本概念

经典的结构化程序设计仅使用顺序、选择、循环这三种基本控制结构的组合来构成整个软件。

（2）结构化程序设计原则

结构化程序设计的原则有如下几条：

- 使用基本控制结构完成所有软件控制结构的构造。
- 设计软件控制结构的过程应该采用逐步求精技术。
- 所有结构都应该是单入口单出口的。

3、详细设计的工具

（1）程序流程图；
（2）盒图（NS 图）：要能够用盒图描写简单的处理过程；
（3）PAD 图；
（4）过程设计语言（PDL）：要能够阅读简单的 PDL 描述。
（5）判定树、判定表

七、编码

编码风格

八、软件测试

1、 软件测试的基本概念

（1）测试的定义和目标

测试是为了发现程序中的错误而执行程序的过程。

测试的目标，是为了发现错误，而不是为了证明软件是正确的。

（2）测试方法：

1、黑盒测试

不考虑程序的内部结构和处理过程，只对它的输入输出进行测试，就是“黑盒”测试。实际上，黑盒测试完成的是对模块功能的测试。

2、白盒测试

如果测试是按照程序的内部结构或者运行的逻辑顺序来进行，这样的测试就是白盒测试。

无论是黑盒法还是白盒法，要想做到穷举测试都是不可能的。

（1）测试策略

软件测试的测试策略和基本步骤是：

1、单元测试；

2、集成测试；

3、系统测试；

4、验收测试；

（2）测试原则

1、一般不测试自己设计的程序。

2、要对测试输入有预期输出。

3、要对合法输入和非法输入都进行测试。

4、测试数据应作为 SCI 纳入文档管理。

6、程序中的错误率和已经发现的错误的比例成正比。

2、单元测试的任务、步骤和方法

单元测试的测试对象是模块，主要任务是检查每一个模块是否能够按照预定的设计要求进行工作，完成预定的功能。

单元测试包含于编码阶段之中，一般按照以下的步骤来进行：

（1）代码审查：静态测试、代码走查；

（2）动态测试：根据测试用例，动态运行程序。

（3）测试软件测试：

单元测试一般以白盒测试为主，之后的其它测试以黑盒测试为主。

3、测试策略

综合测试包括集成测试和系统测试，它的任务主要是测试软件结构上的问题，包括模块之间的接口和整体的功能。

综合测试的方法主要区别在于集成子系统的方法，有非渐增式测试和渐增式测试。渐增式测试其实是把单元测试和组装测试或系统测试结合在一起来进行，是一种循序渐进的方法。使用渐增式测试方法，在如何组合模块的过程上，又有两种不同的策略，自顶向下和自底向上，分别需要设计桩模块和驱动模块。

4、白盒测试与黑盒测试

（1）基于白盒法的测试数据设计（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、路径覆盖。

（2）基于黑盒法的测试数据设计

1、等价类划分：至少有两大类，无效等价类和有效等价类。
2、边界值分析；
3、错误推测；

九、面向对象分析（OOA）

1、OO 基本概念

（1） 面向对象的基本概念，包括：类、对象、属性、方法、消息。
（2） 面向对象的三大特征：封装性、继承性、多态性。
（3） 类-&-对象之间的关系：关联关系、依赖关系、聚合关系、泛化关系、实现关系、消息连接

2、OOA 的过程

（1） 建立用例（功能）模型；
（2） 确定类－&－对象；
（3） 确定类属性；
（4） 确定类服务；
（5） 确定类结构；
（6） 确定主题；
（7） 建立动态模型。

静态（对象）模型

3、UML 的 OOA 建模

面向对象的分析方法，就是采用基于 UML 的图形工具，将现实世界的问题转换为分析模型。

（1） UML 的建模图：
1．用例建模图形工具：用例图。
2．静态建模图形工具：类图（类对象图）、包图、构建图（组件图）、配置图。
3．动态建模图形工具：顺序图（序列图）、协作图（合作图）、活动图（泳道图）、状态图。

（2） UML 的建模过程：
1．用“用例图”、场景描述用户需求；
2．用包图和类图描述系统软件总体框架结构；
3．用顺序图（序列图）、协作图（合作图）、活动图（泳道图）、状态图描述用户操作过程。

十、面向对象设计（OOD）

1、基本概念

面向对象的设计方法，就是采用基于 UML 的图形工具，将分析模型转换为设计模型。

2、OOD 的主要过程

（1） 系统设计：分层、分块、抽象；
（2） 类-对象设计：精化类的属性和方法、精化类间关系；
（3） 数据设计：实体类与关系数据库的映射关系；
（4） 消息设计：消息的类型和发送；
（5） 人机交互设计：界面类、接口类的设计。

结构化设计准则仍然有用：1）模块化　　2）抽象　3）信息隐藏　　4）弱耦合 5）强内聚　　6）可重用

十一、软件维护

1、基本概念

（1）维护的定义和分类：

在软件已经交付使用之后，为了改正错误或满足新的需要而修改软件的过程，称作维护。

1、纠错性维护；

2、适应性维护；

3、完善性维护；

4、预防性维护。

（2）可维护性：

可维护性是指维护人员理解、改正、改动和改进需要进行维护的软件的难易程度。

影响软件可维护性的主要因素有以下这样三种：

1、可理解性；

2、可测试性；

3、可修改性。

2、维护的过程

（1）人员组织：

以维护管理员为核心的人员组织方式。

（2）维护的事件流

（3）维护阶段的文档：

维护阶段自身的文档主要是指维护记录。包括软件问题报告和软件修改报告。

3、维护的副作用

（1）修改代码的副作用；

（2）修改数据的副作用；

（3）修改文档的副作用。

十二、软件项目管理

1. 项目管理的基本概念：过程、软件过程、软件质量、软件配置、软件配置项、

项目质量管理、配置管理、人员组织管理、CMM

2. 规模度量：代码行技术；功能点；项目进度管理、甘特图、工程网络图