软件工程基础讲义

第一章：软件定义

1.软件（ Software）：计算机系统中与硬件相互依存的另一部分，它是包括程序（Program），数据（Data）及其相关**文档**（ Document）的完整集合。

2.软件的特征：逻辑复杂，开发复杂，成本高，风险大，维护困难。

3.按软件功能分类：系统软件，支撑软件，应用软件。

系统软件：操作系统，数据库管理系统，设备驱动程序，通信处理程序等。

支撑软件：文本编辑程序，文件格式化程序，程序库系统等

应用软件：商业数据处理软件，工程与科学计算软件，计算机辅助设计／制造软件，系统仿真软件，智能嵌入软件，医疗、制药软件，事务管理、办公自动化软件。

按软件规模分类：微型，小型，中型，大型，甚大型，极大型。

按软件工作方式分：实时处理软件，分时软件，交互式软件，批处理软件

4.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

5.软件危机主要是两个问题：1.如何开发软件，以满足对软件的日益增长的需求？2.如何维护数量不断膨胀的已有软件？

5.软件危机的表现：1.成本高，开发成本估计不准确 2.软件质量不高、可靠性差 3.进度难以控制 4.维护非常困难 5.用户不满意 6. 由于软件质量问题导致失败的软件项目非常多。

6.产生软件危机的原因：1.与软件本身的特点有关 2.与软件开发与维护的方法不正确有关.

软件工程学的目的：以较低的成本研制具有较高质量的软件

软件工程技术的两个明显特点：1.强调规范化 2. 强调文档化

软件工程的基本原理(7 条)：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

8.软件工程方法学包含 3 个要素：方法、工具和过程

9.软件生命周期：软件定义（问题定义，可行性研究，需求分析），软件开发（ 总体设计，详细设计，编码和单元测试，集成测试），运行维护 （ 持久满足用户需求）

10.软件过程模型：瀑布模型，快速原型模型，增量模型 ，螺旋模型，喷泉模型。

11.软件过程模型 RUP：初始阶段，细化阶段，构造阶段，移交阶段

第二章：可行性研究的任务

1.可行性研究的五个方案：技术可行性，经济可行性，操作可行性，法律可行性，社会效益

2.可行性研究过程：1.复查系统规模与目标、2.研究目前的系统、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

3.系统流程图：用来描述物理系统的工具。

4.**系统流程图表达**：是数据在系统各部件之间流动的情况，而不是对数据进行加工处理的控制过程。即：系统流程图≠程序流程图。

5.系统流程图的基本思想：用图形符号以黑盒子形式描绘组成系统的每个部件

6.系统流程图元素：处理，输入输出，连接，换页连接，数据流。

7.数据流图：用来描述逻辑系统的工具。

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，即数据流图描绘数据在软件中流动和被处理的逻辑过程。

8.数据流图四种基本符号：数据加工/处理/变换，数据源点或终点（外部实体），数据存储，数据流。

9.数据字典的组成：数据流，数据流分量（数据元素），数据存储，处理。

10.数据元素：顺序，选择，重复，可选。

第三章：需求分析

1.需求分析：软件定义时期的最后一个阶段，

2.需求分析的基本任务：不是确定系统怎样完成它的工作，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。

3.软件需求的组成：业务需求， 用户需求，系统需求。

4.需求分析方法：面向数据流的结构化分析方法（SA），面向对象的分析方法（OOA）等

5.逻辑模型：数据流图(DFD)，数据字典(DD)，实体-关系图(ERD)，状态转换图(STD)

6.物理模型：系统流程图，

7.需求分析的基本思想:“自顶向下，逐步求精”，抽象和分解

8.需求分析;功能模型—数据流图，数据模型—实体-关系图，行为模型—状态转换图

9.**实体-关系图(ERD)**：描述数据对象及数据对象之间的关系

10.**数据流图(DFD)**：描述数据在系统中如何被传送或变换，以及描述如何对数据流进行变换的功能（子功能）

11.**状态转换图(STD)**：描述系统对外部事件如何响应，如何动作

模型的核心是数据字典

12.**实体-联系图(ER)组成：:数据对象（实体）、数据对象的属性及数据对象彼此间相互连接的关系。**

联系：一对一联系，一对多联系，多对多联系。

通常用矩形框代表实体，用连接相关实体的菱形框表示关系；用椭圆形或圆角矩形表示实体(或关系)的属性；并用直线把实体(或关系)与其属性连接起来。

13.数据规范化目的是：1.消除数据冗余，即消除表格中数据的重复；2.消除多义性，使关系中的属性含义清楚、单一；3.使关系的“概念”单一化，让每个数据项只是一个简单的数或字符串，而不是一个组项或重复组；4.方便操作。使数据的插入、删除与修改操作可行并方便；5.使关系模式更灵活，易于实现接近自然语言的查询方式。

14.**状态转换图(简称为状态图)：通过描绘系统的状态及引起系统状态转换的事件，来表示**系统的行为。此外，状态图还指明了作为特定事件的结果，系统将做哪些动作(例如，处理数据)。

15.状态：初态：一个，终态：0 或多个，中间状态

16. 验证软件需求：一致性，完整性，现实性， 有效性。

第四章：总体设计

1.总体设计（概要设计）：将软件需求转化为数据结构和软件的系统结构

2.数据库设计包括三个步骤：模式设计，子模式设计，存储模式设计。

3.软件设计原理：模块化，抽象，逐步求精，信息隐藏与信息局部化，模块独立

4.**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，而且

有一个总体标识符代表它。C、C++和 Java 语言中的 {...} 对过程、函数、子程序和宏等面向对象方法学中的对象是模块，对象内的方法也是模块

模块化是好的软件设计的一个基本准则

5.模块独立的含义：模块完成独立的功能，符合信息隐藏和信息局部化原则，模块间关连和依赖程度尽量小。

6.独立性的度量：耦合、内聚。

7.耦合是对一个软件结构内不同模块之间互连程度的度量。

8.耦合的强弱取决于模块间接口的复杂程度，进入或访问一个模块的点以及通过接口的数据

9.模块间的耦合程度强烈影响系统的可理解性、可测试性、可靠性和可维护性。

耦合性越高，模块独立性越弱

10.耦合强度依赖的因素：

一模块对另一模块的引用

一模块向另一模块传递的数据量

一模块施加到另一模块的控制的数量

模块间接口的复杂程度

11.耦合性由强到弱排列为：内容耦合，公共耦合，特征耦合，控制耦合，数据耦合。

12.原则：尽量使用数据耦合，少用控制耦合，限制公共耦合的范围，完全不用内容耦合。

13.内聚（Cohesion）：标志一个模块内各元素彼此结合的紧密程度。

14.内聚有七种，由弱到强分别为：偶然内聚->逻辑内聚->时间内聚->过程内聚->通信内聚->顺序内聚->功能内聚。

15.深度 = 分层的层数。过大表示分工过细。

16.宽度 = 同一层上模块数的最大值。过大表示系统复杂度大。

17.扇出 = 一个模块直接调用/控制的模块数。

18.扇入 = 直接调用该模块的模块数。

19 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。

20.作用域：受该模块中的一个判定所影响的所有模块的集合。

面向数据流的设计方法：变换流，事务流。

第五章：详细设计

1.详细设计：描述系统的每个程序，包括每个模块和子程序名称、标识符、层次结构系

2.对程序的功能、性能、输入、输出、算法、流程、接口等进行描述

3.程序控制结构：顺序、选择，循环，(多分支，DO While ,DO Until)五种基本控制结构。

4.程序流程图又称为程序框图：是对一个模块的内部执行过程用图形来描述。

5.盒图:只能从上边进入，从下边走出，没有其他的入口和出口，

6.盒图的基本符号:顺序,选择型(If-then-else),多分支选择型(CASE 型),DO-WHILE 循环(先测试循环), DO-UNTIL 循环(后测试循环). 调用子程序.

7.PAD 图:PAD 图中竖线的总条数就是程序中的层次数

8.PAD 图基本符号:顺序,选择,循环,Case 分支,语句标号,定义.

9.判定表:左上部列出所有的条件,左下部是所有可能的操作,右上部是各种条件的组合矩阵,右下部是每种条件组合对应的动作

第六章：软件实现

1.实现：编码和测试

2 编码：把软件设计结果翻译成用某种程序设计语言书写的程序

3.程序设计语言：机器语言，汇编语言，高级语言

4.程序内部的文档包括：恰当的标识符，适当的注释，程序的视觉组织 。

5.符号名即标识符;包括模块名、变量名、常量名、标号名、子程序名、数据区名以及缓冲区名等。

6.程序的注释:程序员与日后的程序读者之间通信的重要手段

7.注释分为序言性注释和功能性注释.

8.软件测试是保证软件质量的关键步骤，是对软件规格说明、设计和编码的最后复审，其工件量约占总工作量 40%以上（对于人命关天的情况，测试相当于其它部分总成本的 3－5 倍）。

8.软件测试方法:静态测试方法,,动态测试方法

9.静态测试方法:人工测试方法,计算机辅助静态分析方法

10.动态测试方法:白盒测试方法,黑盒测试方法.

11.黑盒测试法又称功能测试: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

12.白盒测试法又称为结构测试：把程序看成装在一个透明的白盒子，测试者完全知道程序的结构和处理算法

13.软件测试步骤：1.模块测试又称（单元测试），2.子系统测试，3.系统测试称为集成测试，4.验收测试也称为确认测试，5.平行运行

14.单元测试主要使用白盒测试技术。

15.单元测试重点：模块接口，局部数据结构 ，重要的执行通路，出错处理通路，边界条件。

16.集成测试方法：非渐增式测试方法，渐增式测试方法

17.渐增式测试策略：可使用深度优先的策略，或宽度优先的策略

18.回归测试：是指重新执行已经做过的测试的某个子集，以保证修改变化没有带来非预期的副作用。

19.白盒测试技术：逻辑覆盖

20.逻辑覆盖是以程序内部的逻辑结构为基础的设计测试用例的技术。

21.逻辑覆盖：语句覆盖， 判定覆盖 ，条件覆盖，判定－条件覆盖， 条件组合覆盖，

21.路径覆盖，点覆盖= 语句覆盖，边覆盖=判定覆盖，路径覆盖 与条件组合覆盖。

22.语句覆盖：每条语句至少执行一次

23.判定覆盖 ：每一判定的每个分支至少执行一次

24.条件覆盖：每一判定中的每个条件，分别按“真”、“假”至少各执行一次

25.判定－条件覆盖：同时满足判定覆盖和条件覆盖的要求

26.条件组合覆盖：求出判定中所有条件的各种可能组合值，每一可能的条件组合至少执行一次。

27.路径覆盖：每条可能的路径都至少执行一次，若图中有环，则每个环至少经过一次

28.黑盒测试着重测试软件功能。

29.黑盒测试技术：等价类划分，边界值分析法，错误推测法

30.等价类：有效等价类和无效等价类

31.边界值分析法，应该选取刚好等于、稍小于和稍大于等价类边界值的数据作为测试

数据

32.调试途径—调试策略：蛮干法，回溯法，原因排除法--

33.原因排除法-包括：对分查找法、归纳法、演绎法

34.软件可靠性：可靠性，可用性，正确性

35.可靠性和可用性的区别是：可靠性是在 0 到 t 时间间隔内，系统没有失效的概率。而可用性是在 t 时刻，系统正常运行的概率。

第八章：软件维护

1.软件维护的定义：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。

2.软件维护的原因：在运行中发现软件错误和设计缺陷，这些错误和缺陷在测试阶段未能发现。

3.软件维护的类型：改正性维护，适应性维护，完善性维护，预防性维护

4.软件维护的内容：程序维护，数据维护，硬件维护

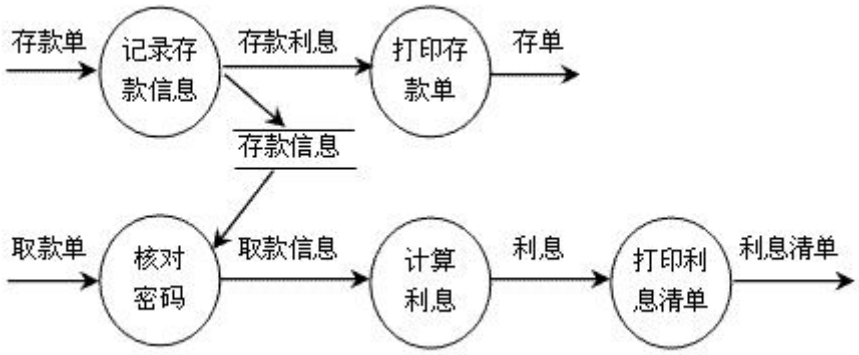
5.软件维护的特点：结构化维护与非结构化，维护的代价分（有形代价和无形代价），维护的问题。

6.软件维护过程：建立维护组织，维护报告，维护的事件流，保存维护记录，评价维护活动，

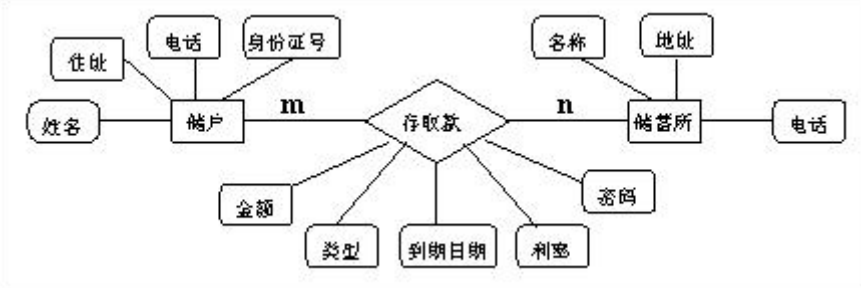
7.软件的可维护性：1.决定软件可维护性的因素（可理解性，可测试性，可修改性，可移植性， 可重用性）2.文档 —— 影响可维护性的决定因素，比代码更重要。3.复审

2、某银行计算机储蓄系统的工作流程大致如下：储户填写的存款单或取款单由业务员键入系统，如果是存款则系统记录存款人的姓名、住址（或电话号码）、身份证号码、存款类型、存款日期、到期日期、利率及密码（可选）等信息，并印出存款单给储户；如果是取款而且存款时留有密码，则系统首先核对储户密码，若密码正确或存款时未留密码，则系统计算利息并印出利息清单给储户。请用数据流图描绘本系统的功能。并画出系统的 E-R 图。

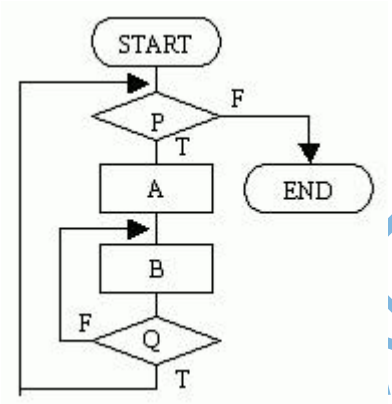
DFD 图：



ER 图：

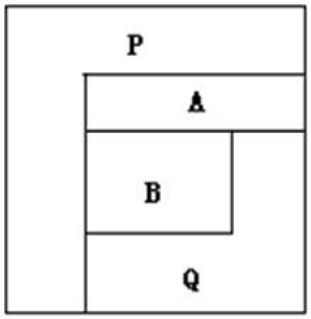


3、试用 N-S 图和 PAD 表示下面程序流程图，并计算它们的 McCabe 复杂性度量.（基本路径测试法的 环复杂度）。（10%）

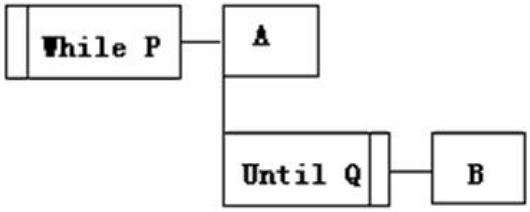


解答：

N-S 图：



PAD 图：



McCabe 复杂性为 3。