软件工程复习

填写期末问卷：
https://www.wjx.cn/vm/mBmbndT.aspx#

重庆理工大学
计算机科学与工程学院 李梁
课程号：K3388095
qq:1255214405
liliang@cqut.edu.cn
智慧树课号： K3388095
智慧树网址： www.zhihuishu.com

使用“知到”APP扫码加课

考核方式及分配

课程目标	考核与评价法方式及成绩比例（%）				成绩比例（%）
	平时作业	课堂测试	实验报告	期末考试	
目标1：软件工程问题解决方案			5	10	15
目标2：软件项目评估估算			5	5	10
目标3：可行性研究及方案选优		5	5		10
目标4：软件系统分析设计及测试	10		5	30	45
目标5：软件项目管理	5	5			10
目标6：软件质量标准		5		5	10
合计	15	15	20	50	100

考核方式及分配

- 平时成绩（共计50%）
  - 考勤：3次迟到算1次缺勤，缺勤超过1/4取消考试资格
  - 作业：须以正确的格式和要求时间交作业（15%，5次，每次作业3分）
  - 课堂测试：15%，3次，每次5分
  - 实验成绩（实验指导书中实验）（20%，实验2-3，每个10分）
- 期末理论考试（50%，开卷）

期末考试要点

- 1、选择题（10分，10题）考核课程目标1：运用软件工程原理思想、方法及模型对软件系统和软件应用对象进行分析，寻求制定软件系统的解决方案、思路和方法。基本概念（如软件工程、软件危机、生命周期、开发模型、需求分析、概要设计、详细设计、模块独立性度量标等）
- 2、开放问答题（20分，2题）考核课程目标6：软件工程标准及CMM及CMMI；课程目标1：软件工程解决方案可行性分析及方案优选（软件开发模型）。
- 3、分析设计题（20分，2题）考核课程目标2：应用软件系统的项目成本效益、开发时间、开发风险等估算方法；考核课程目标4：软件测试技术。
- 4、综合题分析设计（50分，2题）：考核课程目标4：软件系统分析设计

期末考试要点

- 4、综合分析设计题（50分，2题）：考核课程目标4：软件系统分析设计及测试
  - ✓根据文字描述，进行需求分析及需求描述
  - ✓针对特定问题提出解决思路（不是解决方案，只需解决问题的方法名称、主要步骤和主要评价指标或理论依据）
  - ✓针对系统部署场景，提出部署硬件方案及核心参数
  - ✓建立分析模型（SA、OO），建立概要设计模型（SA、OO）业务流程图、用例图、SASD法（顶层数据流程图、细化数据字典、软件结构图）、OO法（对象类图、顺序图）。
  - ✓针对系统某核心输入场景，写出测试用例

课程总结

- 第1-2章 软件工程概、软件过程及软件开发模型
- 软件、程序、数据、文档的含义
- 软件危机及其表现、解决方法
- 软件工程的定义与核心思想
- 软件工程的基本原理与原则
- 软件工程方法学三要素
- 软件过程及软件过程模型
- 软件工程的生命周期的概念以及它的各个阶段的内容，各阶段主要任务
- 软件工程的主要开发模型：瀑布模式、增量模型、螺旋模型、原型、面向对象过程模型

课程总结

- 第3-1章 软件需求分析
- 软件需求分析概念、主要过程与步骤、软件需求的类型
- 软件解决方案：目标、性能指标评价、解决方式、可行性分析。
- 启动分析过程：确认利益相关者、识别视点、协同工作、首次提问
- 主要的需求获取技术：会谈、调查表、场景描述、
- 主要的需求描述技术：流程图和用例图

7

课程总结

- 第3-2章 结构化分析建模
- 分析模型概述
- 结构化分析模型概念
- SA概念和SA法主要模型（功能、数据、行为）
- 掌握数据流程图的基本画法，会画数据流程图
- 掌握数据字典的书写格式，会编写数据字典
- 对已明确需求的软件，会设计其主要数据流程图，编写主要数据字典
- 掌握决策树、决策表的绘制，熟悉结构式语言表示法；

8

课程总结

- 第4-1章 软件设计及结构化软件设计
- 系统设计（概要设计、过程设计）的任务和步骤（过程）
- 设计原则：抽象、自顶向下逐步求精、模块化、信息隐藏的、重构概念
- 常见的模块独立性度量标准：耦合（7种）和内聚（7种）
- 软件模块结构中的基本术语和启发式设计策略：深度、宽度、扇出、扇入、调用、判断调用、数据信息、控制信息
- 掌握概要设计（总体设计）绘制工具：层次图和HIPO图、软件结构图
- 掌握SD法设计的过程与步骤
- 会区分变换型、事务型数据流图，并掌握相应的映射方法
- 会使用SD法将DFD转映射成模块结构图

9

课程总结

- 第4-3章 结构化软件设计-详细设计
- 知道详细设计（过程设计）的内容、过程及步骤
- 知道常见的编程语言和工具（C/S和B/S）
- 知道常见的数据库产品
- 知道常见的软件CASE工具
- 知道结构化程序设计的三种基本控制结构

10

课程总结

- 第5-7章 面向对象分析与设计
- 面向对象方法学的概念、特点，与面向过程开发方法的区别；
- 面向对象分析设计的步骤、过程
- 清楚对象、类、实例、消息、属性、方法（服务）、关联、、泛化、组合、依赖的概念
- 知道UML的主要模型及构成的主要图形（用例图、对象类图、顺序图）的画法和事务（类、属性、服务、主要关联、、包、泛化、聚集、用例、事件、状态、消息）
- 对已明确需求的描述，会设计其面向对象的模型（用例图、活动图、对象类图、顺序图）

11

课程总结

- 第8-9章 软件实现及测试
- 知道软件测试的目的、原则、标准和测试步骤
- 知道黑盒法与白盒法测试技术
- 知道单元、集成、系统测试的主要内容，在开发过程中的位置和使用的主要技术
- 会进行逻辑覆盖及基本路径分析及设计测试用例
- 会设计黑合测试中等价类划分、边值测试的测试实例
- 测试用例设计的目的及编制方法
- 了解软件维护的概念、任务及软件维护的分类

12

课程总结

- 第10-13章 软件项目管理
- 了解软件项目管理的内容、过程（组织管理、人员管理、风险管理）
- 基于功能点软件规模估算方法
- 软件项目计划的内容，主要方法（甘特图、工程网络图）
- 第10 14章 软件过程及软件工程标准
- 软件过程的概念、CMM及CMMI基本内容

重点技术

- 需求调查：业务流程图、用例图
- SA法：分层DFD图、数据字典编制（描述）、判断表、判断树
- SD法：软件结构图、HIPO图、DFD映射方法：变换分析与事务分析
- OO：对象模型（对象类图）、动态模型（行为脚本、活动图、顺序图）、功能模型（DFD）
- 测试：使用等价类划分、边值分析设计测试用例

软件工程概述

- 软件：程序+数据+文档
- 软件危机：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。
- 软件工程：软件工程是指导计算机软件开发的维护的一门工程学科，该学科的目的是生产出能按期交付的、在预算范围内的、满足用户需求的、质量合格的软件产品。
  - ✓ 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。
  - ✓ 方法（面向过程、面向对象）、工具和工程
- 软件生命周期：
  - ✓ 可行性研究(软件计划)、需求分析（是什么）、概要设计（做什么）、详细设计、软件实现(编码、单元测试)、软件测试(组装测试、确认测试)、件使用、维护和退役
- 常用软件四种开发模型：瀑布模型、原型模型、增量模型、螺旋模型

软件工程三要素

- 软件工程是把软件看作是一个工程产品，用工程化管理方法
- 软件工程方法学三要素：方法、工具和过程
  - ✓方法：完成软件开发任务的技术方法，回答“如何做”问题。提供一系列软件开发技术。包括完成开发过程中各方面任务的方法并用某种特殊的语言或图形来描述。
  - ✓工具：为运用方法而提供的自动或半自动软件工程支撑环境。将多种工具集成在一起可构成计算机辅助软件工程（CASE）的软件开发支撑系统。
  - ✓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，规定了完成各项任务的工作步骤，回答“何时做”问题。将软件工程方法和工具综合起来，进行软件开发。

软件工程基本目标

- 软件工程基本目标：以工程化思想进行软件开发，生产高质量和高效率的软件。
  - ✓高质量是最大程度地满足客户需求、帮助客户获得成功。
  - ✓高效率就是以最小的成本获得最好的收益，即在规定的时间和规定（资源）预算内，完成规定的功能的软件开发、维护和服务等任务—三个规定

软件开发方法（范型）

- 通常把在软件生命周期全工程中使用的一整套技术方法的集合称为软件开发方法，也称为软件工程开发范型。
- 范型：模型、模式，就是一套实现具体任务的技术、规范的集合。
- 目前使用得最广泛的软件工程方法分别是面向过程的方法学和面向对象方法学。
- 结构化开发方法（面向过程方法）：其技术要么面向行为，要么面向数据
- 面向对象开发方法（范型）：将对象视作一个融合了数据及在其上操作的行为的、统一的软件组件。数据和行为同等重要

常用CASE工具

- 建模工具：如 Rational Rose / Microsoft Visio / StarUML 等。
- 原型制作工具：Balsamiq Mockups / ExtJS / jQuery / EasyUI /墨刀等。
- 集成开发环境：Visual C++ / Visual Studio.NET / My Eclipse 等。
- 数据库设计工具：PowerDesigner / ERWin 等。
- 代码生成工具：Rational Rose（含 C++/Java 等代码生成模块）等。
- 自动化测试工具：Rational RunNer/LambdaTest/ TestComplete 等平台；JUnit / JMeter 等工具。
- 配置管理工具：Git / GitHub / GitEE 等。
- 项目管理工具：Microsoft Project 等。
- AI编程工具：GitHub Copilot、OpenAI Codex、AI Robocode、AlphaCode（Deepmind lab）、AskCodi、CodeT5、Codota、Polycoder、Ponicode、PyCharm、CodeArts Snap、aiXcoder
- Tabnine（AI assistant for software developers | Tabnine：https://www.tabnine.com/）基于AI的代码补全工具

16:26:51 19

软件生命周期概念

阶段	关键问题	结束标准（任务）
问题定义	问题是什么?	关于规模和目标的报告书
可行性研究	是否可行?	系统的高层逻辑模型； 数据流图；成本/效益分析
需求分析	系统必须做什么?	系统的逻辑模型；数据流图； 数据字典，算法描述
概要设计	任何解决问题?	系统流程图；成本/效益分析 层次图和结构图
详细设计	怎样具体的实现	HIPO或PDL
编码和单元测试	正确的程序模块	源程序清单；单元测试方案和结果
测试	符合要求的软件	综合测试方案和结果； 完整一致的软件配置
运行、维护	持久地满足用户需要	完整准确的维护记录

16:26:51 0

瀑布模型软件开发各阶段的任务表

阶段目标	关键问题	重要活动	阶段成果
问题定义	发现事实，并描述问题是什么？	事实→问题：发现和研究系统面临的问题	《项目建议书》
可行性分析	明确问题是否值得做？回答是否可以做？	问题→建议：系统初步调查与分析	《可行性分析报告》
项目立项	解决项目的立项问题	建议→项目：编制初步需求和立项审批表；立项评审	《项目立项书》
项目计划	解决项目的控制问题	项目→计划：编制项目计划，评审与修订项目计划	《项目计划书》
需求分析	回答系统做什么？明确系统不能做什么？	问题→需求：需求详细调查；问题转化与需求描述；需求评审；需求分析	《系统分析报告》
总体设计	解决框架性如何做问题？	需求→软件架构：需求转化与设计；设计评审与修改；补充调研	《概要设计报告》
详细设计	回答具体细节的怎么做问题	软件架构→软件结构：需求分析与研究；设计评审与修改；补充调研	《详细设计报告》
编码	回答具体做问题	软件结构→程序代码：编码、单元测试，硬件安装与测试，系统集成	《源程序清单》
测试与发布	评价做怎样	代码→软件：系统测试；系统安装与测试；系统发布；系统培训；系统切换	《系统测试报告》 《用户手册》
运行与维护	持久地满足用户需要，尽量延长系统的生命力	软件→答案：构造系统之前发现的问题答案，解决问题	《系统运行日志》 《项目验收报告》 《系统总结报告》

21

软件系统的开发方式

自行开发

□自行开发方式是一种完全依靠用户自身的开发力量，由用户单位自身的员工组成项目组，根据用户单位的特点来开发软件系统

联合开发

□联合开发方式是由用户单位与用户单位外的单位共同组成系统开发小组，针对企业具体的情况和要求，共同完成系统开发任务

外包

□外包开发方式是一种“交钥匙”工程的开发方式，即承包方根据用户单位提出开发的要求，提出软件系统的大体框架和开发所需费用等，当双方签订合同后，将系统开发的任务全部外包给专业软件开发单位。

购买

□具有不同功能的信息系统作为一种商品越来越多，用户单位可以象购买其他物品一样，到市场购买所需要的软件系统，这就是采取购买的方式

22

几种开发方式的比较

		自行开发	联合开发	外包	购买
系统分析与设计能力要求		非常需要	非常需要	不太需要	不需要
编程能力要求		非常需要	不太需要	不太需要	不需要
系统的可维护性		容易	容易	比较困难	困难
程序的可维护性		容易	相当困难	相当困难	困难
开发费用	用于单位外部	小	大	大	小
	用于单位内部	大	中等	小	小
开发风险		大	比较大	比较大	小

23

软件需求分析

- ✓需求分析也称为需求工程，是一个非常重要而有很复杂的，需要交替进行，反复迭代的过程。
- ✓软件需求有功能需求和非功能需求。功能需求描述系统所预期提供的服务，而非功能需求描述与系统不直接相关的一些需求
- ✓领域需求是一种特有的功能需求，反应应用领域的基本问题。
- ✓软件需求规格说明文档描述了系统的数据、功能、行为、性能需求、设计约束、验收标准以及其他于需求相关的信息，它有可能成为客户与开发商之间的合同。
- ✓需求分析过程通过执行初步沟通、需求导出、分析与精化、可行性研究、协商和沟通、规格说明、验证和变更管理八个不同的活动来完成。
- ✓需求获取技术主要包括会谈、调查表和场景技术，用于获取用户需求和系统需求。
- 可行性研究过程：实质上是要进行一次大大压缩和简化了的系统分析和设计过程，也就是在较高层次上以较抽象的方式进行的系统分析和设计过程。

16:26:51 24

可行性研究

- 可行性研究的任务：用最小的代价在尽可能短的时间内确定问题是否能够解决。可行性研究的目的不是解决问题，而是确定问题是否值得解决。
- 经济可行性、技术可行性、操作可行性研究系统是否可行
- 可行性研究的输入是系统的一个框架描述和高层逻辑模型
- 输出是一份需求开发评价报告，对需求工程和系统开发是否值得做的具体建议和意见。
- 三个问题：
  - ✓系统是否符合机构的总体要求？
  - ✓系统是否可在现有技术条件、预算和时间限制内完成？
  - ✓系统能否把已存在的其他系统集成？

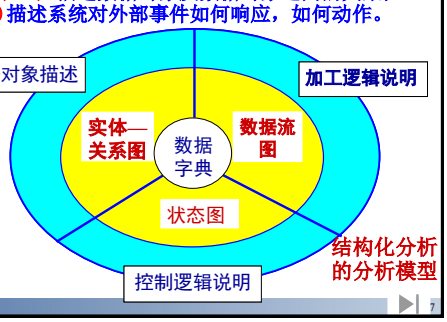
结构化分析概述

结构化分析方法是一种传统的系统建模技术，其过程是创建描述信息内容和数据流的模型，依据功能和行为对系统进行划分，并描述必须建立的系统要素。

- 结构化分析将系统自顶向下逐层分解，达到表达系统的目的，它采用一组过程模型图形化地描述一个系统的逻辑模型。
  - 基于计算机的系统是数据流和一系列的转换构成的
  - 结构化分析模型的组成
    - ✓数据建模和对象描述实体关系图：ERD
    - ✓功能建模和数据流图：DFD
    - ✓行为建模（状态图）：STD
    - ✓数据词典：DD
    - ✓基本加工逻辑说明
  - 在模型的核心是数据词典，它描述了所有的在目标系统中使用的和生成的数据对象。围绕着这个核心的有三种图：ERD、DFD、STD

结构化分析模型

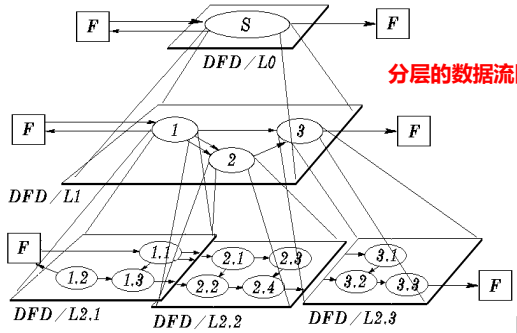
✓数据流图(DFD) 描述数据在系统中如何被传送或变换，以及描述如何对数据流进行变换的功能（子功能）；
✓实体关系图(ERD) 描述数据对象及数据对象之间的关系；
✓状态图(STD)描述系统对外部事件如何响应，如何动作。



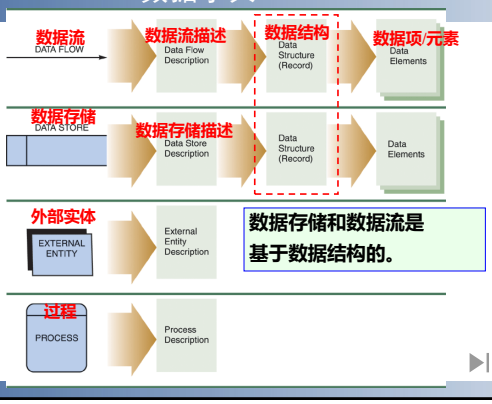
- ERD 用于数据建模
- DFD用于功能建模
- STD用于行为建模

结构化分析的分析模型

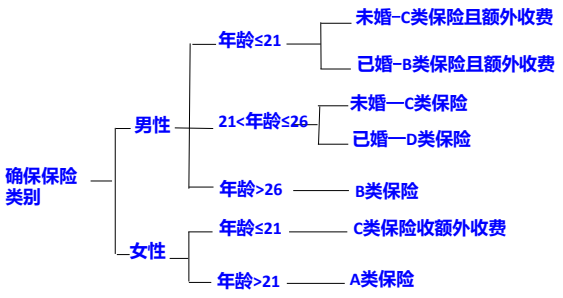
分层数据流图DFD



数据字典



过程描述工具-判定树



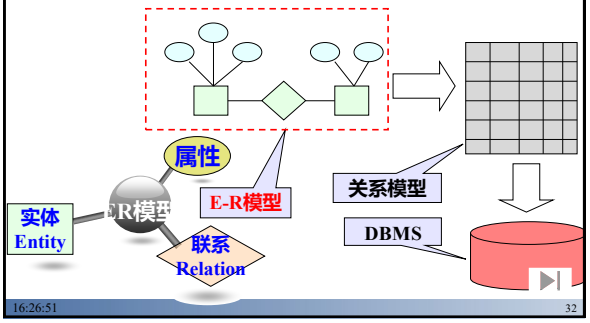
过程描述工具-判定表

条件及行动		1	3	4	5
条件组合	C1:交易额5万以上	Y	Y	Y	N
	C2:无欠款	Y	N	N	—
	C3:交易20年以上	—	Y	N	—
行动	A1:折扣率15%	✓			
	A2:折扣率10%		✓		
	A3:折扣率5%			✓	
	A4:折扣率0%				✓

16:26:51 31

数据建模ER图

数据建模：数据模型包括三种互相关联的信息：数据对象，描述对象的属性，描述对象间相互连接的关系。



16:26:51 32

软件设计概述

- 软件设计分为概要设计、详细设计两个阶段：
  - 概要设计也称总体设计，确定软件的结构以及各组成成分(子系统或模块)之间的相互关系。概要设计通过数据流图来确定系统的结构图，并且对这些结构图进行分析和细化。
  - 概要设计的主要任务是：将系统划分成模块；决定每个模块的功能；决定模块的调用关系；决定模块的界面，即模块间传递的数据。
  - 详细设计就是在概要设计的基础上决定如何具体实现各模块的内部细节，直到对系统中的每个模块给出足够详细的过程描述。在编码实现阶段可以完全按照详细设计的细节过程来映射到代码，最终实现整个系统。一般使用结构化程序设计工具来描述。

16:26:51 33

软件设计概述

- 软件设计是构造系统“怎么做”的模型描述，是对将要实现的软件系统的体系结构、系统的数据、系统模块间的接口，以及所采用的算法给出详尽的描述。
- 结构设计，或称概要设计，或软件总体设计，或高层设计：定义软件组成及各主要成分之间的关系，构造软件系统的整体框
  - ✓ 分析需求规格说明、模块划分，形成具有预定功能的模块组成结构，表示出模块间的控制关系。
- 数据设计：将实体 -关系图中描述的对象和关系，以及数据词典中描述的详细数据内容转化为数据结构的定义。
- 接口设计：接口设计根据数据流图定义软件内部各成份之间、软件与其它协同系统之间及软件与用户之间的交互机制。
- 过程设计，也称模块设计、详细设计，或低层设计：对系统框架、数据结构和界面表示进行细化，对各结构成分所实现的功能，用很接近程序的软件表示形式进行过程性描述。
  - ✓设计模块细节，确定模块所需的算法和数据结构等。

16:26:51

软件模块化设计

- 模块是一个独立命名的，拥有明确定义的输入、输出和特性的程序实体。
- 把一个大型软件系统的全部功能，按照一定的原则合理地划分为若干个模块，每个模块完成一个特定子功能，所有的这些模块以某种结构形式组成一个整体，这就是软件的模块化设计。
- 软件模块化设计可以简化软件的设计和实现，提高软件的可理解性和可测试性，并使软件更容易得到维护。
- 分解、抽象、逐步求精、信息隐蔽和模块独立性，是软件模块化设计的指导思想。
- 模块的独立性是模块化、抽象、信息隐蔽等概念的直接结果，是判断模块化结构是否合理的标准。
  - 模块独立性是指开发具有独立功能而和其他模块没有过多关联的模块。
  - 软件系统中每个模块只涉及软件要求的具体的子功能，和其它的模块的接口是简单的。

35

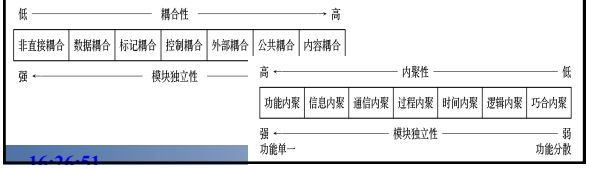
软件模块化

- 软件系统的模块化是指整个软件被划分成若干单独命名和独立访问部分，称之为模块。每个模块完成一个子功能，把全部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。
- 把问题 / 子问题的分解与软件开发中的系统 / 子系统或系统 / 模块对应起来，就能够把一个大而复杂的软件系统划分成易于理解的比较单纯的模块结构。
- 模块化可以使一个复杂的大型程序能被人的智力限制所管理，是软件应该具备的最重要的属性。
- 事实上，每个程序都相应地有一个最适当的模块数目，可使软件系统的开发成本最小。
- 模块划分的目的：①进行功能分解，把复杂的大的功能划分成简单的小的子功能，尽量降低每个模块的成本。②尽量使每个模块间的接口不能太多，太多会使接口成本增加。兼顾二者可取得最佳的划分状态，确保软件总成本最低。

36

模块的独立性

- 模块独立性由两个定性标准度量：
  - 耦合是模块之间的互相连接的紧密程度的度量，模块之间的连接越紧密，联系越多，耦合性就越高，而其独立性就越弱
  - 内聚是模块功能强度(一个模块内部各个元素彼此结合的紧密程度)的度量。一个模块内部各个元素之间的联系越紧密，则它的内聚性就越高，相对地，它与其它模块之间的耦合性就会减低，而模块独立性就越强。
- 模块独立性愈高，则块内联系越强，块间联系越弱。增强模块独立性的方法是尽量做到高内聚、低耦合。



16:26:51

软件设计优化准则

① 划分模块时，尽量做到高内聚、低耦合，保持模块相对独立性。模块划分的准则：“将相关的各部分放在一起，无关的东西不要放在一起。”

② 模块的大小要适中。

③ 模块的接口要简单、清晰、含义明确。便于理解，易于实现、易于测试和维护。

④ 一个模块的作用范围应在其控制范围之内。且判定所在的模块，应与受其影响的模块在层次上尽量靠近。

⑤ 软件结构的深度、宽度、扇入、扇出应适当。

⑥ 力求设计单入口和单出口的模块。避免“病态连接”，以防止内容耦合。

⑦ 设计功能可预测模块的划分，应防止功能过分局限。

16:26:51 38

概要设计

- 概要设计也称总体设计，确定软件的结构以及各组成成分(子系统或模块)之间的相互关系。
- 概要设计的主要任务是：
  - 将系统划分成模块；
  - 决定每个模块的功能；
  - 决定模块的调用关系；
  - 决定模块的界面，即模块间传递的数据。
- 概要设计的实现方式：通过数据流图来确定系统的结构图，并且对这些结构图进行分析和细化。
- 在概要设计阶段，结构化设计主要采用面向数据流的设计方法。

16:26:51 39

详细设计

- 详细设计就是在概要设计的基础上决定如何具体实现各模块的内部细节，直到对系统中的每个模块给出足够详细的过程描述。
- 在编码实现阶段可以完全按照详细设计的细节过程来映射到代码，最终实现整个系统。
- 一般使用结构化程序设计工具来描述

16:26:51 40

面向对象概念

- 面向对象是一种的程序设计方法，或者说它是一种程序设计范型，其基本思想是使用对象、类、继承、封装、消息等基本概念来进行程序设计。
- 面向对象方法学
  - 面向对象分析（OOA，Object-Oriented Analysis）是一种半形式化的规格说明技术。
  - 面向对象设计（OOD）
  - 面向对象实现(OOP)
  - 面向对象测试(OOT)

16:26:51 41

结构化设计方法

- 结构化设计方法：SD设计方法包括一组概念、标准和指导思想。用模块结构图来表达程序模块之间的关系。结构化设计方法有面向数据流的设计方法、Jackson系统开发方法等。
- SD法其目的是降低软件开发费用及维护费用，有利于修正软件及实现新的软件需求。极大地增加了软件复用能力。
- SD设计方法的目标：将软件设计为功能单一的模块，建立系统的模块结构图。它的主要表示方法是一种分层次的结构图。
- SD设计方法的主要思想
  - 认为一个程序、一组程序或一个系统，无非是由一组功能操作来构成的，并进而吸取了结构化分析的“黑盒子”的概念
  - 软件设计者首先必须无视程序、模块或过程的内部情况，而只对它们之间的关系进行分析。将系统看作是，逻辑功能的抽象集合——功能模块的集合。这样软件设计者能够有最大的自由度设计系统结构。

16:26:51 42

结构化设计方法

（1）复查基本系统模型，并精化系统数据流图

（2）分析数据流类型，确定数据流具有变换流特征，还是事务流特征

①如果是变换流特征，确定输入流和输出流的边界（也分别称为最高输入/输出抽象点），输入流边界和输出流边界之间就是变换流，也称为“变换中心”。变换流加工处理的是某些形式的内部数据

②如果是事务流特征，则可确定一个接收分支和一个发送分支。其中发送分支包含一个“事务中心”和各个事务动作流

（3）采用自顶向下、逐步求精的方式完成模块分解，确定相应的软件组成结构

（4）根据模块独立性原理和运用设计度量标准，对导出的软件进行结构优化

面向对象方法

- 面向对象是一种的程序设计方法，或者说它是一种程序设计范型，其基本思想是使用对象、类、属性、方法（操作）、继承、封装、多态、消息等基本概念来进行程序设计。
- 面向对象的思想已经涉及到软件开发的各个方面，包括面向对象的分析、设计、编程和测试等。
- 面向对象建模技术所建立的四种模型，即用例模型、逻辑模型、交互模型和部署模型，分别从四个不同侧面描述了所要开发的系统。
- UML是一种基于面向对象的可视化建模语言。其提供了五种模型视图，包括用例模型视图、结构模型、行为模型视图、实现模型视图和部署模型视图，也称为UML的4+1模型视图。

面向对象模型

- 面向对象方法各阶段模型是统一的：在需求分析阶段建立面向对象模型，在设计阶段精化这些模型，在编码阶段依据这些模型使用面向对象的编程语言开发系统。
- 面向对象分析：是面向对象开发过程中的第一步。对复杂系统进行“抽象”的工作。面向对象分析是在一个系统的开发过程中进行了系统业务调查以后，按照面向对象的思想来分析问题。强调在系统调查资料的基础上，针对OO方法所需要的素材进行归类分析和整理。OOA是一种以从问题域（极有可能来自系统调查的资料）词汇中发现类和对象的概念来考察需求的分析方法。OOA的关键是识别出问题域内的类与对象，并分析它们相互间的关系，最终建立起问题域的简洁、精确、可理解的正确模型。在面向对象分析中，主要包括静态模型（对象模型）、动态模型及功能模型。

面向对象模型

- 面向对象设计：根据已建立的面向对象分析模型，进一步的运用面向对象技术进行系统软件设计。可以分成两个时期：
  - 系统设计时期：设计系统的整体结构和选择解决问题的高级策略；主要目标是表示基于软、硬件体系结构的总体结构设计。
  - 对象设计时期：将问题从问题域的概念转换到计算机领域的概念。对象设计又称详细设计或底层设计，着重于对象及其相互之间交互的描述，即对对象的属性、方法、状态和关系的描述
- 面向对象编程：用面向对象的编程语言，将OOD模型中的各个成分编写成为代码。
  - OOA->OOD->OOP无缝连接和平滑过渡提高了开发的效率和质量。
  - OOP以类对象为中心，把客观实体的功能及数据封装起来，搭建系统，使软件重用在面向对象开发中成为自然的开发模式。
  - 在选择程序设计语言时，除了考虑语言本身的特点和优点以外，还需要考虑对语言支持的可视化编辑开发环境（IDE）的好坏，能够使用的类库支持等。

面向对象模型-常用模型

- 面向对象建模技术的常用模型
  - ✓功能（用例）模型：指明系统应该“做什么”即系统功能。它直接反映用户对目标系统的需求。
  - ✓动态（交互）模型：描述系统中对象的交互及其行为，在规定的何种状态下，接受什么事件的触发而“做什么”，它表示瞬间的、行为化的系统“控制”性质。
  - ✓逻辑（对象）模型：描述系统的逻辑组成，包括对象模型、类模型和包模型，定义“做什么”的对象组成关系。它可表达系统的数据或对数据的处理，它是数据流和语义数据模型的结合。
  - ✓实现模型：描绘系统实现的构件组成和依赖关系。
  - ✓部署模型：描述系统的物理组成，即系统结点（硬件）、连接关系（网络、协议、带宽）和构件（程序）部署在哪些节点上

功能模型

- ✓功能模型表示变化的系统的“功能”性质，指明系统应该“做什么”。功能模型更直接地反映了用户对目标系统的需求
- ✓面向对象是以用例驱动的。用例是站在用户的角度描述用户的交互过程，有助于软件开发人员更深入地理解问题域，改进和完善分析和设计。从开发者看，用例就是一种功能。
- ✓用例图建立起来的系统模型称为用例模型。用例模型描述的是外部行为者所理解的系统功能。
- ✓用例图描述了作为一个外部的观察者的视角对系统的印象。强调这个系统是什么而不是这个系统怎么工作。
- ✓用例图与情节紧紧相关的。情节scenario是指当某个人与系统进行互动时发生的情况。
- ✓功能模型也可以用数据流图表示。

逻辑模型

- 逻辑模型描述系统的逻辑组成：对象模型、类模型、包模型
  - ✓对象模型：描述客观世界实体对象及对象彼此间的关系，描述系统的静态结构。
- ✓类模型：是对象模型的抽象（一类对象的抽象 ），以及是对象模型的静态表示。
- ✓包模型：对类模型的封装，形成层的概念，描述整个系统的组成和逻辑架构。
- 对象模型是面向对象方法中最基本、最重要的模型，它为其他模型奠定了基础。对象模型是一个类（包括其属性和行为 ）、对象（类的实例）、类和（或）对象之间关系的定义集
- 对象模型还必须表示类/对象之间的结构关系。类/对象之间的关系一般可概括为关联、归纳/继承（泛化）、组合（聚集）三类。

UML模型结构

UML视图和图

主要的域	视图	图	主要概念
静态结构	静态视图	类图	类、关联、泛化、依赖关系、实现、接口
	用例视图	用例图	用例、参与者、关联、扩展、包括、用例泛化
	实现视图	构件图	构件、接口、依赖关系、实现
	部署视图	部署图	节点、构件、依赖关系、位置
动态	状态视图	状态图	状态、事件、转换、动作、
行为	活动视图	活动图	状态、活动、完成转换、分叉、结合
	交互视图	顺序图	交互、对象、消息、激活
		协作图	协作、交互、协作角色、消息
模型管理	模型管理视图	类图	包、子系统、模型
扩展机制	所有	所有	约束、构造型、标记值

向对象分析与设计过程

1、系统调查和需求分析：用例建模：用例图

2、分析问题的性质：抽象出对象及其行为、结构、属性、方法等，即OOA ，领域建模：对象类图、顺序图、状态图

3、整理问题：对分析的结果作进一步的抽象、归类、整理，并最终以范式的形式将它们确定下来，即OOD

4、程序实现：用面向对象的程序设计语言将上一步整理的范式直接映射为应用软件，即OOP

- 面向对象分析与设计是一个动态迭代的过程：通过用例模型抽取系统的功能；据业务功能和领域概念得到系统所涉及的概念，进而得到类和对象，以及构建对象模型和类模型；基于系统的行为分析系统类或对象的交互行为，得到类或对象的行为和事件，并构建系统的交互模型。

面向对象分析过程

- 面向对象分析阶段的主要任务是获取用户的需求，并构建系统初步的逻辑模型。
- 用例建模：获取用户的需求，构建用例模型。
  - 识别外部用户
  - 场景分析
  - 构建活动图
  - 构建用例图
- 领域建模：目的是建立系统的概念模型。
  - 抽取领域对象
  - 构建领域模型
  - 构建初步的交互模型

面向对象分析

- 面向对象分析需要将真实世界进行抽象，通过问题的叙述，将真实世界系统加以描述。分析的目的是为了构造一个系统属性和系统行为的模型，该模型是根据对象和对象之间的关系、动态控制和功能转移来确定的。
- 面向对象分析模型：在面向对象分析中，主要包括以下3个独立的模型。
  - ①功能模型：以用例模型为基础，描述系统应具有的功能用于实现用户的日常需要。由用例和场景表示。
  - ②对象模型（静态模型）：对用例模型进行分析，把系统分解成互相协作的分析类，通过类图/对象图描述对象/对象的属性/对象间的关系，是系统的静态模型。对象模型是最基本、最重要、最核心的。
  - ③动态模型：描述系统的动态行为，通过顺序图/协作图描述对象的交互，以揭示对象间如何协作来完成每个具体的用例，单个对象的状态变化/动态行为可以通过状态图来表达。

面向对象分析

- 面向对象分析的步骤：

①需求获取及分析问题域，建立用例模型。（第2节）

②发现和定义对象和类。建立静态模型（类图）（第3-4节）

③识别对象的内部特征。

④识别对象的外部联系。

⑤识别对象之间的交互。建立动态模型（顺序图、状态图第5节）

用例模型

- 用例驱动分析过程

①绘制用例图 ②用例描述 ③开发活动图 ④开发泳道图

- 用例：用例是在一个系统中所进行的一连串的活动场景，该活动要能够满足系统外部的执行者对系统的预期。就是用户对于产品或系统的某一个完整的预期。从另一个角度来说，用例也代表着一个具体的业务场景。
- 用例图建立的系统模型称为用例模型。用例模型描述的是外部行为者所理解的系统功能。系统被看作是一个提供用例的黑盒子，它内部如何工作、用例如何实现，对用例模型并不重要
- 用例图用来图示化描述用户的需求，即用户希望系统具备的完成一定功能的动作，通俗理解用例就是软件的功能模块，这些模块之间的协作调用关系。

16:26:51 55

类图建立步骤

建立类图的一般步骤：

① 研究分析问题领域，确定系统需求；

② 确定类，明确类的含义和职责，确定类的属性和操作；

③ 确定类之间的关系。关联，泛化，聚集，组合，依赖；

④ 调整和细化类及其关系，解决重复和冲突；

⑤ 绘制类图，并增加相应说明。

13:07:42 56

绘制类图建议

- 不要试图使用所有的符号。从简单的开始，例如，类、关联、属性和继承等概念。有些符号仅用于特殊的场合和方法中，只有当需要时才去使用。
- 根据项目开发的不同阶段，用正确的观点来画类图。分析阶段，画概念层类图；软件设计时，画逻辑层类图；考察某个特定的实现技术时，应画实现层类图。
- 不要为每个事物都画一个模型，应该把精力放在关键的领域。最好只画几张较为关键的图，经常使用并不断更新修改。
- 使用类图的最大危险是过早地陷入实现细节。应该将重点放在概念层和说明层。
- 模型和模型中的元素是否有清楚的目的和职责(系统功能最终是分配到每个类的操作上实现的，这个机制叫职责分配)。
- 模型和模型元素的大小是否适中。过于复杂的模型和模型元素是很难生存的，应将其分解成几个相互合作的部分。

57

行为建模

- 功能模型指明了系统应该“做什么”，而动态模型则明确规定了“什么时候做”。即在何种状态下、接受了什么事件的触发，来确定对象的可能事件的顺序。
- 动态模型显示了软件如何对外部事件或激励做出响应。
- UML动态模型使用交互视图：描述了执行系统功能的各个角色之间相互传递消息的顺序关系。顺序图和协作图就是交互视图的一种形式。状态图主要用于描述对象具有的各种状态、状态之间的转换过程以及触发状态转换的各种事件和条件。
  1、评估所有的用例，以使得完成理解系统内的交互序列。
  2、识别驱动交互序列的事件，并理解这些事件如何和具体的类相互关联。
  3、为每个用例生产序列（顺序图、协作图）。
  4、创建系统状态图。
  5、评估行为模型以验证准确性和一致性。

16:26:51 58

面向对象设计

OOA与OOD的界限

OOA：运用面向对象方法，对问题域和系统责任进行分析和理解，找出所需的对象，定义对象的属性和操作以及对象之间的关系，建立一个符合问题域，满足用户需求的OOA模型。

OOA不考虑与系统具体实现有关的因素，而将其留给OOD去处理。

OOD：根据实现条件对OOA模型作某些必要的修改和调整。针对具体实现条件，建立人机界面、数据存储和控制驱动等模型。

“OOA→OOD”就是一个逐步建立和扩充对象(类)模型的过程（逐步精化模型，对OOA模型补充细节的过程）。

- OOA到OOD实际上是一个逐渐扩充模型的过程
  - ✓扩充：主要是增加各种组成部分
  - ✓OOA识别和定义的类/对象，是一些直接反映问题空间和系统任务的。而OOD识别和定义的类/对象则是附加的，反映需求的一种实现

16:26:51 59

面向对象设计

分析模型	设计模型
概念模型，回避了实现问题；	物理模型，是实现蓝图 ；
对设计是通用的；	针对特定的实现；
对类型有3种构造型；	对类型有任意数量的构造型（依赖于实现语言）；
不太形式化 ；	比较形式化；
开发费用较低；	开发费用较高；
层数少；	层数多；
勾画系统的设计轮廓；	进行系统设计；
主要通过研讨会等方式创建；	设计模型和实现模型需双向开发；
可能不需要在整个生命周期内都做维护；	在整个生命周期内都应该维护

16:26:51 60

面向对象设计

传统结构化方法	面向对象方法(UML建模工具为例)
需求模型	输入L 处理P，输出O的视角，面向功能的文档(用户需求规格说明书)需求变化，其功能变化，所以系统的基础不稳固	从用户和整体角度出发。使用系统抽象出用例图、活动图，获取需求；如需求变化，对象的性质相对功能稳定，系统基础稳定
分析模型	面向过程的数据流图DFD、实体—关系图ERD、数据字典DD表示分析模型；功能分解,数据和功能过程分开	把问题作为一组相互作用的实体，显式表示实体间的关系 数据模型和功能模型一致 类、对象图表示分析模型，状态、顺序、协作、活动图细化说明
设计模型	功能模块（SC图），模块之间的连接/调用是模块的附属形式	类和对象实现，类/对象的关联、聚集、继承等连接、连接规范和约束作为显式定义
实施模型	体系结构设计	组件图、配置图
测试模型	根据文档进行单元测试、集成测试，确认测试	单元测试采用类图，集成测试用实现图和交互图，确认测试采用用例图

传统方法	OO方法
系统是处理的集合	系统是交互对象的集合
处理数据与实体的交互	处理对象之间的交互
接受输入产生输出	对象发送消息和响应

编码概述

- **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**
  - ✓作为软件工程过程的一个阶段，编码是对设计的进一步具体化，因此，程序的质量主要取决于软件设计的质量。
  - ✓但是，所选用的程序设计语言的特点及编码风格也将对程序的可靠性、可读性、可测试性和可维护性产生深远的影响。
- **正如任何产品在交付使用之前都必须经过严格的检验过程一样，由于软件开发的复杂性和困难性，软件产品在交付使用之前尤其应该经过严格的检验过程。**
  - ✓目前，软件测试仍然是保证软件质量的主要途径，它是对软件需求规格说明、软件设计和编码的最后复审。

软件测试概述

(1) 测试是程序的执行过程，目的在于发现缺陷，不是证明程序中无错误；

(2) 一个好的测试用例在于能发现至今未发现的缺陷；

(3) 一个成功的测试是发现了至今未发现的缺陷的测试。

(4) 测试并不仅仅是为了找出错误。通过分析错误产生的原因和错误的发生趋势，以便及时改进。

软件测试的实践：尽早测试、连续测试、自动化测试

- ✓以最少的时间和人力，系统地找出软件中潜在的各种错误和缺陷。
- ✓没有发现错误的测试也是有价值的，完整的测试是评定软件质量的一种方法。测试数据为可靠性分析提供依据。
- ✓测试不能表明软件中不存在错误，只能说明软件中存在错误。

软件测试的原则

软件测试的原则

（1）测试用例应由测试数据和对应的预期输出结果这两部分组成。

（2）测试用例包括合理的输入条件和不合理的输入条件。

（3）检查程序是否做了它应该做的和不应该做的。

（4）努力以最少的测试用例来发现最大量的可能错误。

（5）应该远在测试开始之前就制定出测试计划。严格执行测试计划，排除测试的随意性。

（6）测试计划、测试报告必须作为文档长期保存。

（7）不要试图穷举测试。

对具有多重选择和循环嵌套的程序，不同的路径数目可能是天文数字。

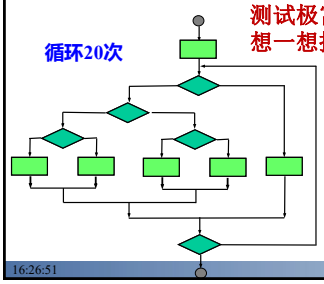
软件测试的原则

软件测试的原则

（7）不要试图穷举测试。对具有多重选择和循环嵌套的程序，不同的路径数目可能是天文数字。因此，测试只能证明程序中有错误，而不能证明程序中没有错误。

测试极富创造性和挑战性——想一想批改别人写的作文吧！

循环20次



执行路径数达 5^{20} 条，假定一条路径进行测试需要1毫秒，一年工作365×24小时，把所有路径测试完需3170年。

软件测试的原则

软件测试的原则

（8）Pareto原理：大多数情况下都存在“关键的少数”群体，也被称为80/20理论。木桶理论：强调改进我们木板最短的那一块. 只有短木板变为长木板，桶装水才会更多。

测试发现的错误中80%很有可能是由程序中20%的模块造成的，即错误出现的“群集性”现象，找到这些模块并进行彻底测试。测试后程序中残存的错误数目与该程序中已发现的错误数目成正比。

（9）应该避免测试自己的程序，测试是找错误，是破坏性的，让人测试自己的代码，是折磨人的。人们对自己的作品印象深刻、熟悉，难以想象哪里会出错。独立的第三方从事测试工作，会更客观、更有效。

软件测试的技术和分类

软件测试类型

测试对象角度	单元测试、集成测试、系统测试、验收测试
测试技术角度	黑盒测试（功能测试）、白盒测试（结构测试）
程序执行角度	静态测试、动态测试
人工干预角度	手工测试、自动化测试

16:26:51 67

白盒测试法

语句覆盖相当于遍历图中的每个顶点，但不一定遍历了每条边。
判定覆盖相当于从起点到终点的一条路径，遍历图中的每个顶点和每条边（但不一定经过了所有路径）。
条件覆盖不能保证覆盖所有路径。
判定/条件覆盖不能保证覆盖所有路径（是n个条件的n-n组合）
条件组合覆盖选取合适的组合，能够保证覆盖所有路径。不合适的组合依然不能保证覆盖所有路径。
基本路径测试：设计足够的测试用例，覆盖程序所有可能的路径。

发现错误的能力：弱 ↓ 强

语句覆盖	每条语句至少执行一次
判定覆盖	每一判定的每个分支至少执行一次
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
判定/条件覆盖	同时满足判定覆盖和条件覆盖的要求
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

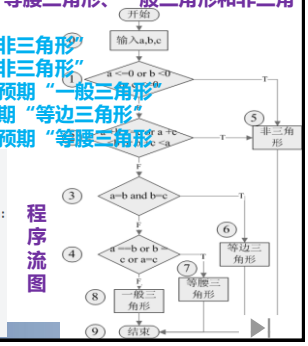
16:26:51 68

白盒测试法-基本路径测试

给定三个整数a、b、c，它们表示三角形的三条边长，请判断这个三角形的类型，包括等边三角形、等腰三角形、一般三角形和非三角形四种类型。
0-1-5-9：a=0, b=1, c=2, 预期“非三角形”
0-1-5-9：a=1, b=1, c=2, 预期“非三角形”
0-1-2-3-4-8-9：a=1, b=2, c=3, 预期“一般三角形”
0-1-2-3-6-9：a=1, b=1, c=1, 预期“等边三角形”
0-1-2-3-4-7-9：a=2, b=2, c=3, 预期“等腰三角形”

```
def triangle(a, b, c):
    if a <= 0 or b <= 0 or c <= 0:
        return "非三角形"
    if a + b <= c or a + c <= b or b + c <= a:
        return "非三角形"
    if a == b and b == c:
        return "等边三角形"
    if a == b or b == c or a == c:
        return "等腰三角形"
    return "一般三角形"
```

程序流图



白盒测试法-基本路径测试

思考：其它测试策略
1. 边界值测试：包括等于、小于和大于边界值的情况。例如，对于边长范围为1-100的三角形分类函数，可以测试输入值为1、100和101的情况。
2. 异常值测试：包括非数值、负数、零和空值等异常情况。例如，可以测试输入值为“bc”、“1”、“0”和空的情况。
3. 分支覆盖测试：覆盖所有可能的分支情况。例如，可以测试三个边长相等、两个边长相等、三个边长不相等且形成直角三角形等情况。
4. 条件覆盖测试：覆盖所有可能的条件组合情况。例如，可以测试两条边之和小于第三条边、两条边之和等于第三条边、两条边之和大于第三条边且三边相等等情况。
5. 路径覆盖测试：覆盖所有可能的执行路径情况。例如，可以测试输入三个边长相等、两个边长相等、三个边长不相等且形成等腰三角形、三个边长不相等且形成直角三角形等情况。

16:26:51 70

黑盒测试技术

- 黑盒测试是根据程序组件的规格说明测试软件功能的方法，所以也称为功能测试。被测对象作为一个黑盒子，它的功能行为只能通过研究其输入和输出来确定，又称为输入/输出接口测试
- 设计黑盒测试用例的原则
  - ✓对于有输入的所有功能，既要用有效的输入来测试，也要用无效的输入来测试。
  - ✓经过菜单调用的所有功能都应该被测试，包括通过同一个菜单调用的组合功能也要测试。
  - ✓设计的测试用例数量，能够达到合理测试所需的“最少”（减少测试成本）。
  - ✓设计的测试用例，不仅能够告知有没有错误，而且能够告知某些类型的错误存在或不存在（提高测试效率）

16:26:51 71

等价类划分测试用例设计案例

QQ账号为5-11位自然数，请用等价类划分方法设计测试用例。
有效等价类：5-11位自然数 无效等价类：小于5位，大于11位，非自然数

用例编号	覆盖等价类	输入	预期结果
LG_ST_001	1、2	12345678	系统提示输入正确
LG_ST_002	3	1234	系统提示用户名应为5-11位自然数
LG_ST_003	4	123456789101	系统提示用户名应为5-11位自然数
LG_ST_004	5	-123456	系统提示用户名应为5-11位自然数
LG_ST_005	6	1.23456	系统提示用户名应为5-11位自然数
LG_ST_006	7	123456a	系统提示用户名应为5-11位自然数
LG_ST_007	8	123456%	系统提示用户名应为5-11位自然数
LG_ST_008	9	1234567小强	系统提示用户名应为5-11位自然数
LG_ST_009	10	空	系统提示用户名应为5-11位自然数

72

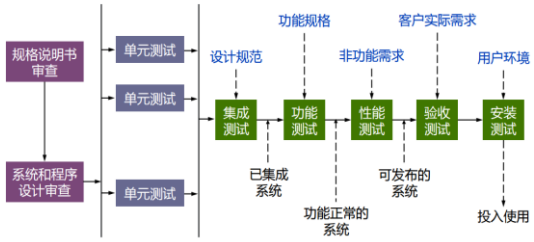
白盒测试和黑盒测试的方法对比

		白盒测试	黑盒测试
测试规划		根据程序的内部结构，如语句的控制结构，模块间的控制结构以及内部数据结构等进行测试	根据用户的规格说明，即针对命令、信息、报表等用户界面及体现它们的输入数据与输出数据之间的对应关系，特别是针对功能进行测试
特点	优点	能够对程序内部的特定部分进行覆盖测试	能站在用户的立场上进行测试
	缺点	无法检验程序的外部特性 无法对未实现规格说明的程序内部欠缺部分进行测试	不能测试程序内部特定部分 如果规格说明有误，则无法发现
方法举例		**逻辑覆盖：**语句覆盖 判定覆盖 条件覆盖 判定-条件覆盖 **基本路径覆盖** **循环路径测试**	等价类划分 边界值分析 因果图法测试

16:26:51 73

软件测试过程

软件产品在交付使用之前一般要经过单元测试、集成测试、确认测试和系统测试4个阶段，下图描述了整个测试过程。



16:26:51 74

软件过程

- **概念：软件过程是为了开发出软件产品，或者是为了完成软件工程项目而需要完成的有关软件工程的活动**
  - ✓通常使用生命周期模型简洁地描述软件过程
  - ✓每项活动可分为一系列工程任务，科学的软件过程是一组适合软件项目特点的任务集合，任务集合包括一组软件工程任务、、里程碑、应交付的产品。

软件过程框架 重庆时代百货销售系统

- □框架是实现整个软件开发活动的基础，软件过程框架定义了若干小的框架活动，与过程有关的角色、职责的定义以及实现也都离不开框架的支持。实际上就是软件过程的工程模板
- □**框架中的普遍性活动：沟通、计划、建模、构建、部署**
- ✓组织及管理框架：实现过程活动涉及到的角色与职责
- ✓技术及工具框架：实现过程活动自动化及需要的设备与工具

16:26:51

软件过程

- 软件过程提高了软件工程活动的稳定性、可控性和有组织性，过程受到严格的约束，保证软件活动有序进行。
- 从软件工程师的观点来看，产品依赖过程，其就是过程定义的一系列活动和任务的结果，即要交付的软件。
- 软件团队根据产品的特征以及自身特点选择特定的软件过程来开发产品。
  - ✓**软件产品越复杂，其开发周期也越长，开发成本越高。团队就要选择重型软件过程，如螺旋模型或者统一过程模型。**
  - ✓**当产品较为简单或需求比较稳定时，开发周期也比较短，开发人员也比较少，可采用轻型软件过程，如增量方法、极限编程方法或瀑布模型。**

76

软件过程评估

软件过程评估：评价软件过程中的各种活动（管理、技术方法、控制、计划等）是否满足软件工程成功所需的基本过程标准要求。

作用：检验和识别软件过程风险，促进软件过程改进，提高软件开发能力。

- **过程评估**
  - ✓**CMMI/CMM**
  - **软件能力成熟度模型/集成**
  - **卡内基梅隆大学2001年9月**
  - ✓**ISO9001:2000**
  - ✓**PSP（个人软件过程）**
  - ✓**TSP（团队软件过程）**
  - ✓**SPI（软件过程改进）**

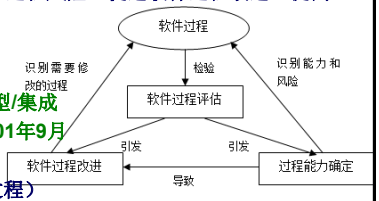


图 2-1 软件过程评估

77

软件过程评估

CMM：分析或诊断软件或软件团队的相对成熟度

CMMI：在CMM基础上集成了工程方面的评价指标，包括启动、诊断、建立、执行和学习，主要用于软件企业。

ISO9001:2000：质量保证体系，用于产品、系统或服务的整体质量评估，采取“计划-实施-检查-行动”循环，将其应用于软件项目的质量管理环节，可直接应用于软件团队和软件企业。

SPICE标准（软件过程改进和能力测定）：定义软件过程评估的一系列要求，帮助软件企业对其软件过程的改变进行计划、制定以及实施，建立客观的评价体系，评估软件过程的有效性

个人软件过程（PSP）：建立个体过程基线，进行个人度量。

团队软件过程(TSP)：采用循环递增的开发策略，整个软件生产过程由多个循环出现的开发周期组成，每个开发周期划分出若干个相对独立的阶段。具体方法：计划评审、设计和编码标准、设计和代码评审方法、缺陷评审、质量分析

16:26:51

能力成熟度模型-评估模型

- CMM为软件企业的过程能力提供一个阶梯式的改进框架，它基于过去所有软件工程过程改进的成果，吸取以往软件工程的经验教训，提供一个基于过程改进的框架。
- CMM是指导软件开发组织或项目逐步改进其软件能力成熟度的一个指南。
- CMM只回答“做什么？”，具体地“如何做？”由开发组织自己定，并不提供做这些改进的具体措施。使软件过程是一个可管理、可测量和可改进的过程。
- 在CMM中把软件过程从无序到有序的进化关键过程(18个)分成5个等级（初始级、可重复级、已定义级、已管理级（已控制级）、优化级），形成5个逐层提高的等级。共52个目标，316个关键实践
- CMM引导软件开发组织不断识别出其软件过程的缺陷，并指出应该做哪些改进。

能力成熟度模型-CMM五个级别

级别	CMM级别的特征
1级，初始级	软件过程无序，有时甚至混乱，成功归功于个人的努力
2级，可重复级	在项目内部已建立健全的项目管理，可重复已有的成功经验
3级，已定义级	在组织内部已定义并实施全组织统一的软件标准过程，人员自觉遵守
4级，已管理级	软件过程和产品质量已得以有计划和定量的管理和控制，可度量和控制
5级，优化级	已建立使软件过程和产品质量得以可持续改进的架构

从无序到有序、从特殊到一般、从定性管理到定量管理、最终达到动态优化。

能力成熟度模型-CMMI

✓CMMI (Capability Maturity Model Integration)：即能力成熟度模型集成，是CMM模型的最新版本。

✓将各种能力成熟模型集成到一个框架中去，建立一种从集成产品与过程发展，从工程角度健全的系统开发原则的过程改进，消除不同模型之间的不一致与重复

✓软件集成：各种硬件（计算能力、存储能力、网络交换能力、信息安全）、系统软件、应用软件、工具软件、存储、运行环境（机房电力、温度、清洁、防雷、网络通讯）等集成到一起的工程。

✓主要集成部分：系统工程（SE）、软件工程（SW）、集成产品和过程开发（IPPD）、供应商外包管理（SS）

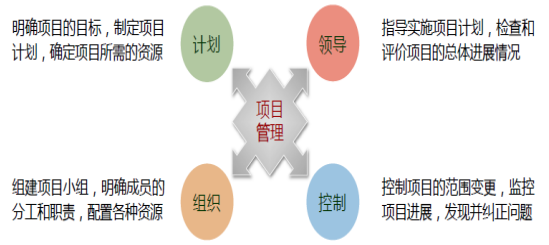
能力成熟度模型-CMMI

CMMI认证可以通过以下几个方面提高软件开发质量：

1. 流程规范化：CMMI认证要求组织建立和维护一套有效的流程和实践。通过规范化的流程，可以确保软件开发过程中的每个环节都得到适当的关注和处理，减少错误和风险的发生。
2. 质量管理：CMMI认证要求组织建立有效的质量管理体系，包括度量和分析、问题管理、缺陷管理等。通过质量管理的手段，可以对软件开发过程中的质量进行监控和控制，及时发现和解决问题，提高软件的质量和稳定性。
3. 风险管理：CMMI认证要求组织建立风险管理机制，包括风险识别、评估、规划和控制。通过风险管理，可以在软件开发过程中及时识别和评估潜在的风险，制定相应的风险规划和控制措施，降低风险对软件质量的影响。
4. 持续改进：CMMI认证要求组织建立持续改进的文化和机制，能够不断寻求和实施改进措施。通过持续改进，可以不断优化软件开发过程和实践，提高开发效率和质量，降低错误和缺陷的发生。

软件项目管理

软件项目管理是为了使软件项目能够按照预定的成本、进度、质量顺利完成，而对成本、人员、进度、质量和风险进行控制和管理的活动。

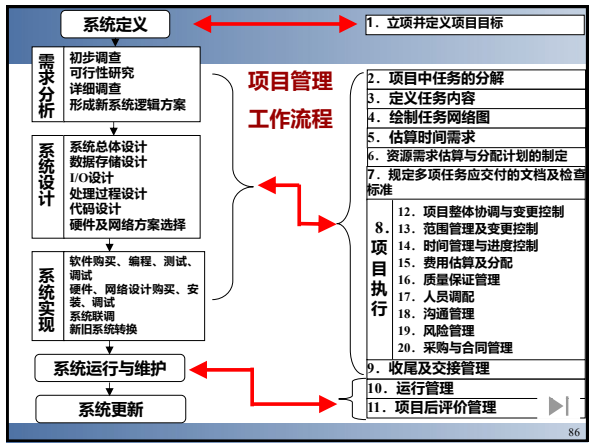


软件项目管理

- 所谓管理就是通过计划、组织和控制等一系列活动，合理地配置和使用各种资源，以达到既定目标的过程。
- 项目管理在一定资源条件，如时间、资金、人力、设备、材料、能源、动力等的约束下，为有效地达到项目的既定目标（如项目竣工时计划达到的质量、投资、进度），按照项目的内在规律和程序，对项目的全过程进行有效的计划、组织、协调、领导和控制的系统管理活动。
- 软件项目管理的对象是软件工程项目。它所涉及的范围覆盖了整个软件工程过程。
- 为使软件项目开发获得成功，关键问题是必须对软件项目的工作范围、可能风险、需要资源(人、硬件／软件)、要实现的任务、经历的里程碑、花费工作量(成本)、进度安排等做到心中有数。

软件项目任务

- 软件项目管理的任务
  - ✓软件开发管理的任务：制定文档；预计需要的资源；费用估算；安排工作任务和计划；定期做评审；质量保证管理；开发总结报告；处理意外情况等。
  - ✓软件测试管理的任务：制定测试计划；测试分析并报告；编制用户手册。
  - ✓项目运行管理的任务：人员的组织与管理；设备和资料管理；财政预算与支出管理；作业时间管理。
  - ✓项目后评价管理的任务：技术水平与先进性评价；经济与社会效益分析；系统的内在质量评价；系统的推广使用价值评价；系统的不足之处与改进意见等。



管理中的基本概念

1、风险分析

- 每当新建一个系统时，总是存在某些不确定性。
  - ✓ 用户要求是否能确切地被理解？
  - ✓ 在项目最后结束之前要求实现的功能能否建立？
  - ✓ 是否存在目前仍未发现的技术难题？
  - ✓ 在项目出现严重误期时是否会发生一些变更
- 风险分析对于软件项目管理是决定性的，然而现在还有许多项目不考虑风险就着手进行。

管理中的基本概念

2、进度安排

- ✓ 预先对进度如何计划？
- ✓ 工作怎样就位？
- ✓ 如何识别定义好的任务？
- ✓ 管理人员对结束时间如何掌握 ？
- ✓ 如何识别和监控关键路径以确保结束？
- ✓ 对进展如何度量？
- ✓ 如何建立分隔任务的里程碑。

- 软件项目的进度安排与任一个工程项目的进度安排基本相同。首先识别一组项目任务，再建立任务之间的相互关联，然后估算各个任务的工作量，分配人力和其它资源，制定进度时序。

管理中的基本概念

3、追踪和控制

- 一旦建立了开发进度安排，就可以开始着手追踪和控制活动。
- 由项目管理人员负责追踪在进度安排中标明的每一个任务。
- 如果任务实际完成日期滞后于进度安排，则管理人员可以使用一种自动的项目进度安排工具来确定在项目的中间里程碑上进度误期所造成的影响。
- 还可对资源重新定向及对任务重新安排
- （做为最坏的结果）可以修改交付日期以调整已经暴露的问题。用这种方式可以较好地控制软件的开发。

甘特图

- 甘特图（Gantt Chart）：描绘任务的分解情况，每个任务的工作量、开始时间和结束时间、各个任务间的依赖关系
  - ✓甘特图是历史悠久、应用广泛的制定进度计划的工具，也称横道图，是一种最直观的进度计划方法。
  - ✓采用直线线条在时间坐标上表示出单项工程内容进度，线段的起点和终点分别对应子任务的开工时间和完成时间，线段的长度表示完成任务所需的时间。
  - ✓从该图上可以很清楚地看出各子任务在时间上的对比关系。
  - ✓在甘特图中，每一任务完成的标准，不是以能否继续下一阶段任务为标准，而是以必须交付应交付的文档与通过评审为标准。因此在甘特图中，文档编制与评审是软件开发进度的里程碑。 重庆时代百货销售系统

甘特图

关键日期表：列出一些关键活动和进行的日期。

- 绘制简单，结构清晰
- 不能反映活动之间的并、串、层叠关系，优化困难

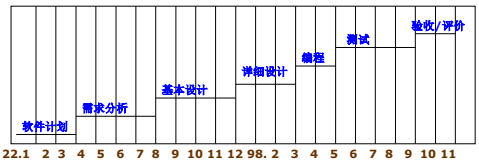
开始日期	结束日期	活动名称	负责人	参与人
2017-1-8	2017-2-8	软件计划	王总关	张晶里，刘祝
2017-2-9	2017-3-10	需求分析	王总关	张晶里，刘祝，徐告收
2017-3-11	2017-4-20	总体设计	张晶里	刘祝，徐告收
2017-4-21	2017-5-20	详细设计	刘祝	张晶里，徐告收

甘特图

重庆时代百货销售系统

甘特图示例：

计划安排的工作
计划安排的工作的开始日期
计划安排的工作的完成日期
在特定的时间内安排的工作量X
工作的目前进度

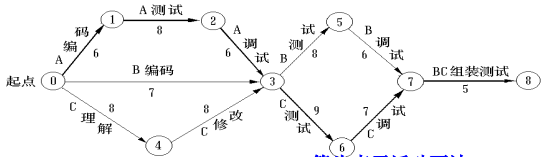


优点：它具有简单、醒目和便于编制等特点。能够动态反映软件项目开发进展的情况。

缺点：难以反映多个任务之间存在的复杂的逻辑关系。

工程网络图

- **工程网络图（PERT）：**用网状图表安排与控制各项活动的方法，由美国海军于20世纪50年代发明。确定完成整个工程至少需要多少时间，哪些子工程是影响工程进度的关键。一般适应于工作步骤密切相关、错综复杂的工程项目的计划管理。
  - ✓图中用箭头表示作业(活动)，用圆圈表示事件（任务）AON
  - ✓事件（任务）仅明确定义的时间点，并不消耗时间和资源
  - ✓作业（活动）通常既消耗资源又需要持续一定时间。



AOA (Activity On Arrow)箭头表示活动画法

软件项目度量概述

1、**软件度量：**对软件产品、开发过程或资源简单属性的定量描述。为了有效地定量的进行项目管理。

2、**软件项目估算（事前）：**对就需要的人力（人月）、项目持续时间（年份或月份）、成本（万元）做出的定量估算。

3、**测量（事后）：**事后或实时状态下采用某种测量方法测量软件的软件使用的人力、项目持续时间及成本。

4、软件项目估算是制定软件项目计划的基础，有以下共性：

估算必然存在一定程度的不确定性（预测未来）；

估算要随着项目的进展不断地进行调整和更新；

可将软件项目被分解为可单独进行估算的小块；

已完项目的度量数据（时间、规模、人力、成本、文档、错误等）是将来同类项目估算的依据和有效的经验。

管理人员一般使用多种估算技术，相互交叉检查结果。

软件项目度量概述

5、软件度量的方式

- **直接度量：**软件规模（代码行数LOC，模块数，功能点数 FP），执行速度，存储量大小，时间周期中所报告的差错数；
- **间接度量：**功能性、复杂性、效率、可靠性、可维护性
- 软件工程过程的直接度量包括所投入的成本和工作量。
- 软件产品的直接度量包括产生的代码行数（LOC）、执行速度、存储量大小、在某种时间周期中所报告的缺陷数。很易度量
- 软件产品的间接度量包括功能性、复杂性、效率、可靠性、可维护性和许多其它的质量特性。 很难度量
- 功能性、效率、可维护性等质量特性却很难用直接度量判明，只有通过间接度量才能推断。

软件项目估算概述

6、软件估算的内容：

- 工作产品规模估计：
  - ✓功能点个数：*N* 个功能点/人月；
  - ✓性能点个数：*N* 个性能点/人月；
  - ✓代码行数：*N* 行代码/人月；
  - ✓实体个数：*N* 个实体/人月；
  - ✓需求个数：*N* 个需求数/人月；
  - ✓文档页数：*N* 页文档/人月。
- 工作量及成本估计：直接劳务费；管理费；差旅费；计算机使用费；其他招待费和公关费。
- 关键资源的量化估计：软件工作产品的规模；运行处理的负载；通信量。

软件项目估算概述

7、常用的估算方法

(1)基于代码行的成本估算方法

(2)基于功能点（用例点、对象点）的成本估算方法

(3)任务分解成本估算

(4)经验统计估算模型

①参数方程

②动态多变量参数模型

③COCOMO模型（constructive Cost Model）

功能点法FP

- 功能点度量方法（Function Points）：不依赖开发语言
  - ✓使用软件的功能测量为单位度量软件规模的间接度量方法。
  - ✓考虑软件的“功能性”和“实用性”，而不是对 LOC计数
- FP ＝UFP×(0.65＋ 0.01×SUM (Fi))
  UFP：加权的功能项总计数　Fi：复杂性调节值
- UFP=a1×输入+a2×输出+a3×查询 +a4×文件+a5×接口
  ai(1≤i≤5)是信息域特性系数（简单、平均、复杂）

信息域参数	计数		加权因数 简单	中间	复杂		加权计数
用户输入数		×	3	4	6	=	
用户输出数		×	4	5	7	=	
用户查询数		×	3	4	6	=	
文　件　数		×	7	10	15	=	
外部接口数		×	5	7	10	=	
总　计　数							

团队管理的基本概念

- 团队的定义：团队是一定数量的个体成员组织的集合。包括自己组织的人、供应商、分包商、客户的人等，为一个共同的目标工作，分工合作，愉快的合作，最终开发出来高质量的产品。
- 团队不是一群人临时聚集在一起，各自完成任务就走人(work group)
- 团队管理的特点：项目组织全体成员的管理和项目组织自身的管理；最大限度发掘个人和团队的能力；是项目管理中最为根本的一项管理。
- 团队管理的特点：要针对临时性；着重团队性；适应项目生命期
- 团队管理的内容：
  ①项目经理确定、任务和职业道德
  ②项目组织形式的确定
  ③项目成员的确定
  ④项目团队的建设
  ⑤沟通管理

风险管理

风险管理：在项目中不断对风险进行识别、评估、制定策略、监控风险的过程。以便最大限度满足项目的目标。

风险管理的层次：危机管理-救火模式；风险缓解；着力预防；消灭根源

风险管理的意义：

- 有效地控制项目的成本、进度、产品需求。
- 可以阻止意外的发生，增加项目成功的可能性。
- 可以防止问题的出现，即使出现，也可以降低程度。
- 可以将精力更多地放到项目的及时提交上。
- 风险管理相当于一份项目保险，是一个保险投资。

风险管理

风险类型

- 预测角度：已知风险－Known known、可预测风险-Known unknown、不可预测风险-unknown unknown。
- 范围角度：项目风险、技术风险、商业风险。

项目风险：项目风险是指潜在的预算、进度、个人（包括人员和组织）、资源、用户和需求方面的问题。项目的复杂性、规模的不确定性和结构的不确定性也是构成项目风险的因素。

技术风险：技术风险是指潜在的设计、实现、接口、检验和维护方面的问题。规格说明的多义性、技术上的不确定性、技术陈旧也是技术风险因素。

商业风险：市场风险、策略风险、管理风险、预算风险

风险的基本性质：风险的客观性、风险的不确定性、风险的不利性、风险的可变性、风险的相对性、风险同利益的对称性。

任务	可能的风险	产生的阶段	产生的原因	避免的措施	发生后的处理
制定设计阶段的规范和标准	时间风险	项目准备	需制定的规范和标准较多，而同时需完成其他工作，使得可使用的时间和资源有限		
开发环境确认	资源风险	系统设计	由于设备未到位导致延误开发		
管理系统设计	技术风险	系统设计	基于TeMIP平台开发SDH专网管理系统对于公司乃至国内都是全新的课题，由于技术的掌握程度和经验的欠缺	在系统设计前请TeMIP专家进行相关培训	该换成其他的技术实现
对功能规格和系统设计的调整	时间风险	α0版本开发	评测结果对功能规格和系统设计影响较大		
α0版本开发	时间风险	α0版本开发	由于学习曲线过长延误时间		
系统测试	资源风险	α0版本开发	开发人员与SQA人员对工作站和服务器使用的争夺		
MD现场调试	资源风险	α1版本开发	由于设备问题延误现场调试		

风险分析表

排序	输入	风险事件	可能性	影响	风险值	采取的措施
1	客户的SOW	需求不明确，增加需求，导致需求蔓延。	70%	50%	35%	1.采取加班的方法 2.修改计划，去掉一些任务 3.与客户商量延长时间
2	合同	进度要求紧，资金不足。	30%	50%	15%	请一些学生做一些辅助性的工作，降低成本，加快进度
3	WBS	供货商、外包商有质量问题。	20%	50%	10%	多选择几个备份的外包商和供应商
4	历史项目信息	开发人员流失。	15%	60%	9%	1.多沟通了解开发人员动态 2.控制好项目中的文档 3.从其它项目组借调人员 4.招聘有相关经验的人员

103