

一、课程介

绍1、课程性

质

《软件工程》是全国高等教育自学考试计算机及应用（独立本科段）的一门专业课。

软件工程是研究软件开发的一门课程，其主要内容包括：软件开发所需要的过程、活动和任务，以及这些活动和任务的组织、实施和管理。

2、指定教材

本课程指定教材为《软件工程》，全国高等教育自学考试指导委员会组编，王立福主编，机械工业出版社出版，2011 年版。

新版教材与 2000 年版相比，无论是内容还是内容的组织，都有了很大的变化。整个知识体系、章节安排、内容选取都不一样，这是考生一定要注意的。新版教材的内容组织特点主要体现在：

基于对软件开发本质的认识，讲解软件工程的两大技术问题：一是开发逻辑，二是开发途径。

开发逻辑涉及软件生存周期过程、软件生存周期模型（有关过程、活动和任务的组织框架）以及项目软件生存周期的规划与监控。

开发途径涉及结构化方法和面向对象方法，以及支持软件评估所需要的软件测试技术等。

3、章节体系

本课程共有 8 章：

第 1 章：回答什么是软件开发的本质

第 2 章：软件需求与软件需求规约

第 3 章：结构化方法

第 4 章：面向对象方法-

UML 第 5 章：面向对象方

法-RUP 第 6 章：软件测试

。

第 7 章：软件生存周期过程及管理

第 8 章：集成化能力成熟度模型

CMMI 二、考情分析

1. 历年真题的分布情况

由于教材刚刚经过改版，新教材刚经过 2011 年 10 月、2012 年 01 月、2012 年 10 月三次考试。通过对 2011 年 10 月、2012 年 01 月这两次真题的分析，各章所占分值的分布情况如下表所示：

年份 章名、题型	2011-	2012-
一、绪论（单项、填空题）	3 分	3 分
二、软件需求与软件需求规约	9	11
三、结构化方法（单、填、简答、综合）	25 分	25 分
四、面向对象方法-UML（单、填、简答）	11 分	11 分
五、面向对象方法-RUP（单、填、简答）	12 分	12 分
六、软件测试（单、填、简答、综合）	25 分	23 分
七、软件生存周期过程及管理（单、填、简）	10 分	10 分
八、集成化能力成熟度模型 CMMI	5	5

从上面的统计数据可以看出：主要的分值分布在第 3 章和第 6 章，分别占到总分的 25% 左右。第 1 章和第 8 章的考核知识点相对较少。

2. 题型分析

本课程的考试题型分为：

- (1) 单项选择题，共 15 小题，每小题 2 分，共 30 分
- (2) 填空题，共 20 个空，每空 1 分，共 20 分
- (3) 简答题，共 6 小题，每小题 5 分，共 30 分
- (4) 综合应用题，共 2 题，每题 10 分，共 20 分

3. 复习方法

(1) 以教学大纲为准绳。自学考试的原则是：考试范围既不超出大纲又不超出教材范围。所以考生一定根据教学大纲规定的考试内容和考核要求，认真学习教材，要全面、系统了解教材中的基本概念、基本知识。

(2) 有的放矢。在学习的过程中，为了达到“事半功倍”，要学会“舍”。要用有限的时间去抓重点，对重点内容要进行深入细致的学习。

(3) 注意学习方法，理论联系实际，注重理解

重视理论联系实际，训练并逐渐提高运用所学理论分析和解决实际案例的能力。考生应当注意在全面系统学习教材的基础上，尽可能多地了解和分析实际案例，以便更深刻地领会教材的内容，提高分析和解决实际问题的能力。

(4) 合理安排时间，抓住学习重点

根据实际情况自己安排，利用平时空余时间观看网络课件，形成基本的了解。接下来认真地做一些练习题，不清楚的地方再回过头去看看书，并注意对不同的知识点进行比较，加深印象。

第一章 绪论

复习建议：

本章内容较少，主要是让大家了解软件工程的提出的背景-软件危机以及软件工程研究的内容。

考试题目类型主要是单项选择题、填空题，题量在 3%~5%之间。

第一节 软件工程概念的提出与发展

1. 软件危机

(1) **速度**：软件的发展水平远远滞后于硬件的发展水平，生产率低下，软件制造仍然是一种人工集约生产方式

(2) **质量**：软件的质量低下，不能满足用户的需求、适应性差

(3) **成本**：软件开发成本居高不下

软件开发的速度、软件制品的质量、软件开发成本是软件工程的三个核心问题。

2. 软件工程的发展

(1) 20 世纪 60~80 年代

瀑布模型；过程化语言；支持工具

(2) 20 世纪 80 年代~今

软件复用技术；软件生产管理；面向对象语言

(3) 近几年

第二节 软件开发的本质

1. 软件

软件=程序+文档

2. 软件开发的本质：“映射”，即实现问题空间的概念和处理逻辑到解空间的概念和处理逻辑之间的映射。

3. 系统建模

运用所掌握的知识，通过抽象，给出系统的一个结构。

4. 模型

模型是一个抽象。模型是在特定意图下所确定的角度和抽象层次上对物理系统的描述，通常包含对该系统边界的描述、对系统内各模型元素以及它们之间关系的语义描述。

5. 系统模型的类型

(1) 概念模型：描述软件是什么

(2) 软件模型：实现概念模型的软件解决方案。包括设计模型、实现模型和部署模型。

第二章 需求获取

复习建议：

正确定义问题，是解决问题的基础。

需求获取是软件开发的第一步，它的工作质量决定了整个软件开发工作的成败，因此本章的内容是考核的重点内容。

考核的题目类型主要有：单项选择题、填空题、简答题，分值在10%左右。

内容以基本概念、基本原理为主。

第一节 需求与需求获取

1. 需求的定义

一个需求是有关一个“要予构造”的陈述，描述了待开发产品/系统功能能力、性能参数或其它性质。

2. 需求的基本性质

-
- (1) 必要的
 - (2) 无歧义的
 - (3) 可测的
 - (4) 可跟踪的
 - (5) 可测量的

3. 需求的分类 ★

- (1) 功能需求，是整个需求的主体。
- (2) 非功能需求：性能需求、外部接口需求、设计约束和质量属性需求。

能够区分哪些是功能需求，哪些是性能需求。

4. 接口需求的类别

- (1) 用户接口
- (2) 硬件接口
- (3) 软件接口
- (4) 通信接口
- (5) 内存约束
- (6) 运行
- (7) 地点需求

5. 设计约束需求

- (1) 法规政策
- (2) 硬件限制
- (3) 与其它应用的接口
- (4) 并发操作
- (5) 审计能力
- (6) 控制功能
- (7) 高级语言要求
- (8) 握手协议
- (9) 应用的关键程度
- (10) 安全和保密

6. 质量属性

- (1) 可靠性
- (2) 存活性

-
- (3) 可维护性
 - (4) 用户友好性

7. 需求发现的技术

- (1) 自悟
- (2) 交谈
- (3) 观察
- (4) 小组会
- (5) 提炼

第二节 需求规约 (SRS)

1. 需求规约的定义 ★

是一个软件/产品/系统所有需求陈述的正式文档，它表达了一个软件/产品/系统的概念模型。

2. 需求规约的基本性质 ★

- (1) 重要性和稳定性程度：对需求进行分级
- (2) 可修改的
- (3) 完整的：没有被遗漏的需求
- (4) 一致的：不存在互斥的需求

3. 需求规约的格式

IEEE 标准 830-1998(IEEE 1998)描述的需求规格说明书模板。

- (1) 非形式化的需求规约
- (2) 半形式化的需求规约
- (3) 形式化的需求规约

9

-
- (1) 需求规约是软件开发组织和用户之间一份事实上的技术合同书，是产品功能及其环境的体现
 - (2) 需求规约是一个管理控制点
 - (3) 对于产品/系统的而设计，需求规约是一个正式的、受控的起始点
 - (4) 需求规约是创建产品验收计划和用户指南的基础

第三章 结构化方法

复习建议：

自顶向下，逐步求精。

本章是整个课程的重点内容，其基本思想、基本原理和基本方法是软件工程理论体系中最经典的内容，考核题型涉及单项选择题、填空题、简答题、综合应用题所有题目类型，占分值25%左右。

建议考生在牢记基本概念、基本原理的基础上，对综合应用题多下工夫，多做练习。

第一节 结构化需求分析

1. 需求分析面临的挑战

- (1) 问题空间理解
- (2) 人与人之间的通信，“有效沟通”
- (3) 需求的变化性

2. 结构化分析中的基本术语及表示方法

- (1) 数据流



- (2) 加工



- (3) 数据存储



- (4) 数据源和数据潭



1. 数据流图 DFD 图 ★

用于建立系统功能模型。

是一种描述数据变换的图形化工具，其中包含的元素可以是数据流、数据存储、加工、数据源和数据潭等。

2. 建模过程（绘制流程图的过程）

自顶向下、功能分解

- (1) 建立系统环境图
- (2) 0 层图：从 0 层图开始对流程图中的要素编号
- (3) 1 层图
- (4)

【例题】绘制数据流程图（2008 年 10 月真题）

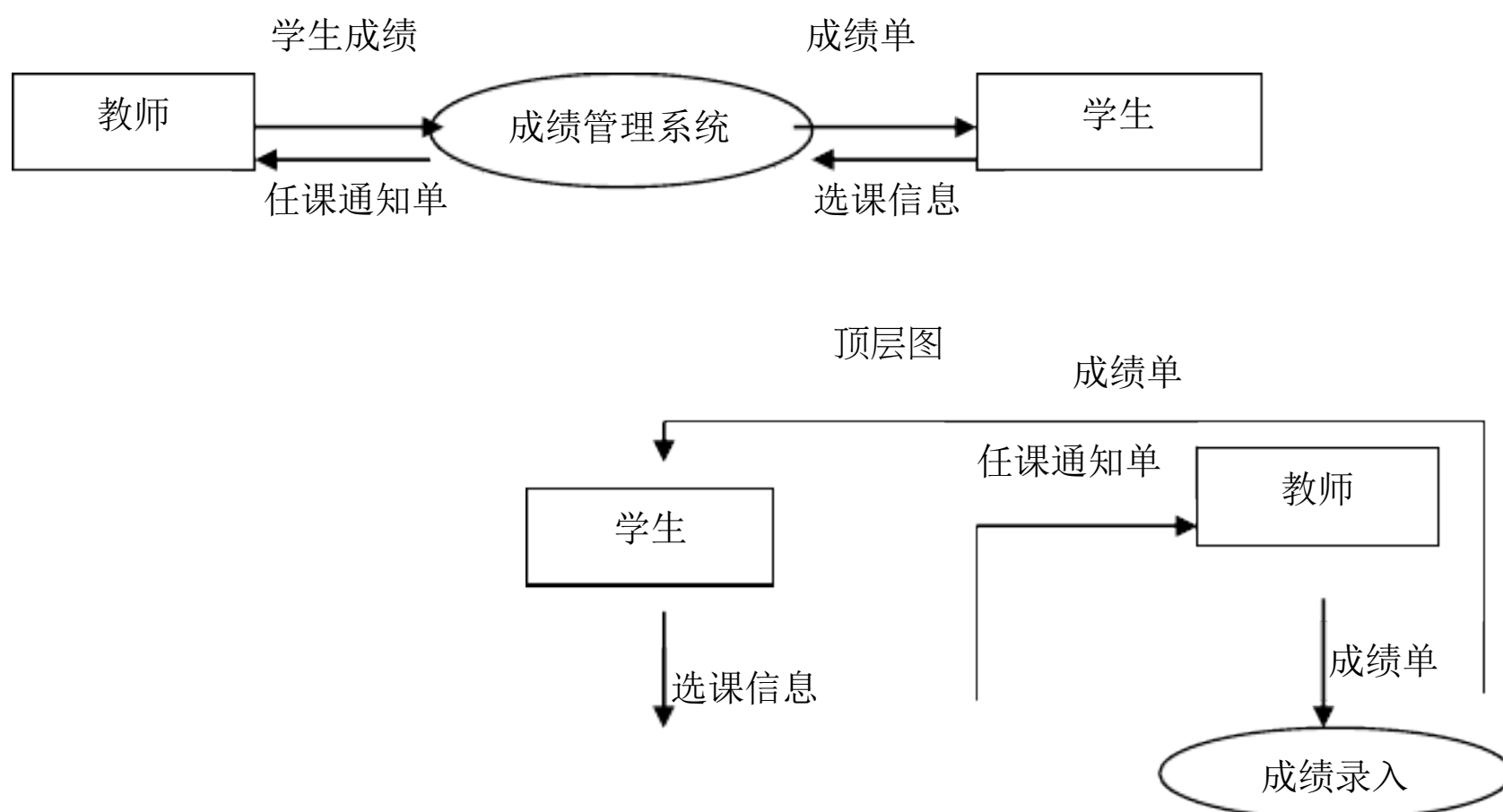
41. 某个学生成绩管理系统的部分功能如下：

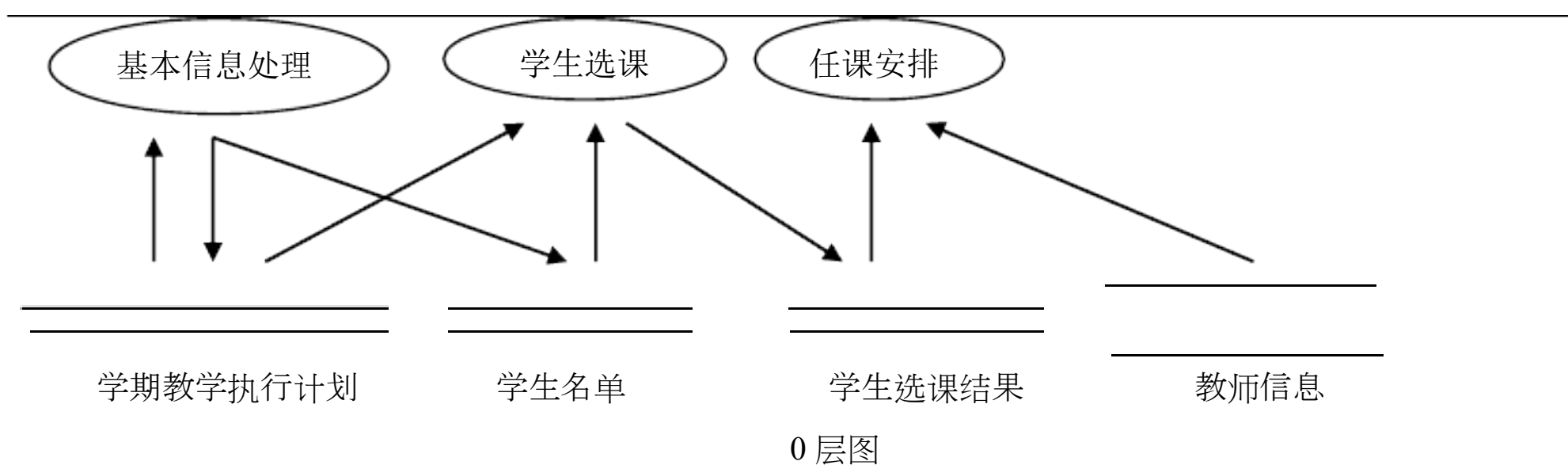
- (1) 基本信息管理：教务管理人员输入或修改学期教学执行计划、学生名单和教师名单；
- (2) 学生选课：学生根据教学执行计划进行选课；
- (3) 分配任课教师：教务管理人员为符合开课条件的课程分配教师，并打印任课通知单给教师；
- (4) 成绩管理：每门课程的教师在考试评分结束后将考试成绩交给教务管理人员，教务管理人员输入、维护成绩，系统可生成成绩单（发给学生）、成绩统计分析表（发给教务管理人员）。

请根据要求画出该问题的分层数据流图（要求画出顶层和 0 层数据流图）。

【解析】

顶层图：只包含数据源/数据潭以及相关的数据流和一个处理。





要注意的问题：

- ① 黑洞(black hole)，即只有输入而没有输出。
- ② 只有输出而没有输入。
- ③ 灰洞(gray hole)，即输入不足以产生输出。灰洞是经常也是不易被察觉的错误。
- ④ 加工处理只用来表示数据的处理和变化，避免将计算机命令作为处理。
- ⑤ 数据流必须起于且/或止于处理，即每一个数据流必须有一个处理与之有关，数据流不能起于数据存贮且止于一个数据源/数据潭或另一个数据存贮；也不能起于某个实体且止于另一个数据源/数据潭或数据存贮。

3. 数据字典

定义数据流程图中所有数据流和数据存储的数据结构。

顺序结构：+

选择结构：|

重复结构：{ }

子界：m..n

4. 加工的描述 ★

(1) 判定表

判断表(Decision Table)也称为决策表，是一个二维表，它说明了每一种条件组合所产生的结果。

该表分为四个象限(quadrants)。

- a) 左上限代表所有的条件
- b) 左下限代表可能的结果
- c) 右上限代表每一种条件的取值(用Y 和N 来表示)
- d) 右下限用X 表示所对应的条件组合所产生的结果

决策规则号		1	2	3	4	5	6	7	8
条件	条件1	Y	Y	Y	Y	N	N	N	N
	条件2	Y	Y	N	N	Y	Y	N	N
	条件3	Y	N	Y	N	Y	N	Y	N
应采取的行动		×	×						
				×	×	×	×		
								×	×

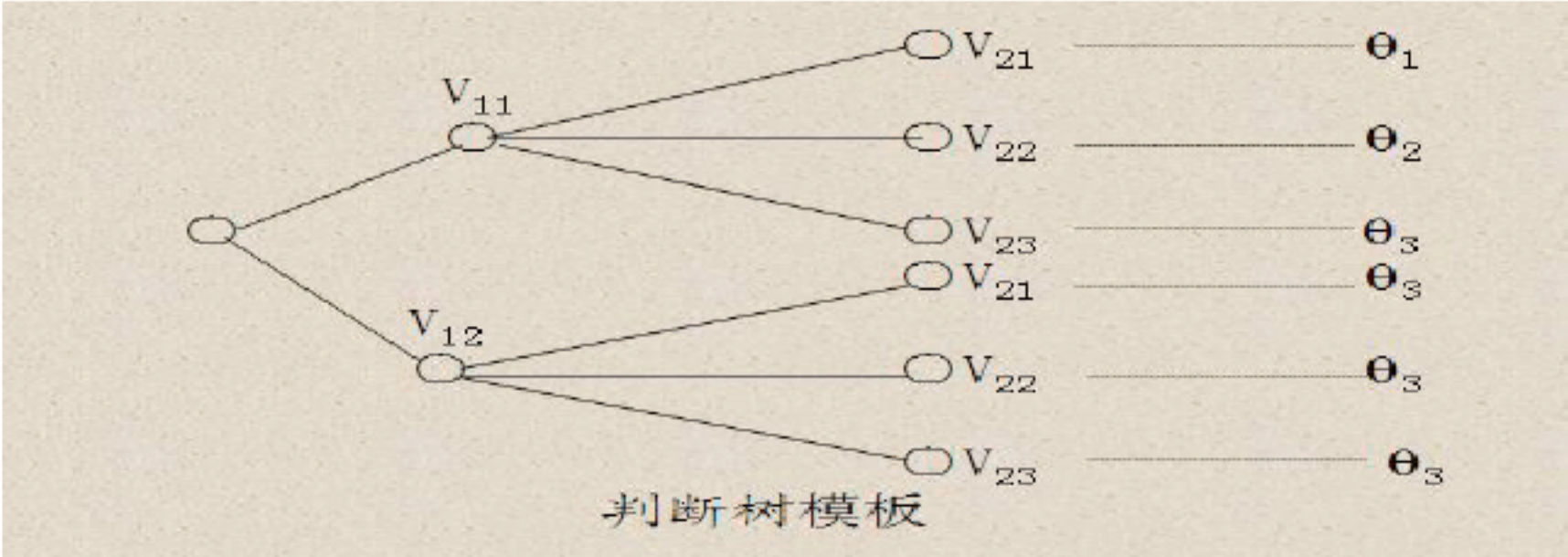
【例题】画出顾客购货的折扣政策的决策表。

销售商在给顾客的折扣时，要考虑付款日期和交易额这两个因素。若付款日期在10天以内(含10天)，则当交易额超过¥10,000时，给予5%的折扣；当交易额在¥5,000到¥10,000之间(含¥5,000)时，给予3%的折扣；当交易额低于¥5,000时，没有折扣。若付款日期超过10天，则无论交易额多少，均不给任何折扣。

【解析】

决策规则号		1	2	3	4
条件	付款日期≤10	Y	Y	Y	N
	交易额≥10000	Y	N	N	—
	交易额<5000	N	N	Y	—
	判定树				
(1)	折扣5%	×			
	折扣3%		×		
	无折扣			×	×

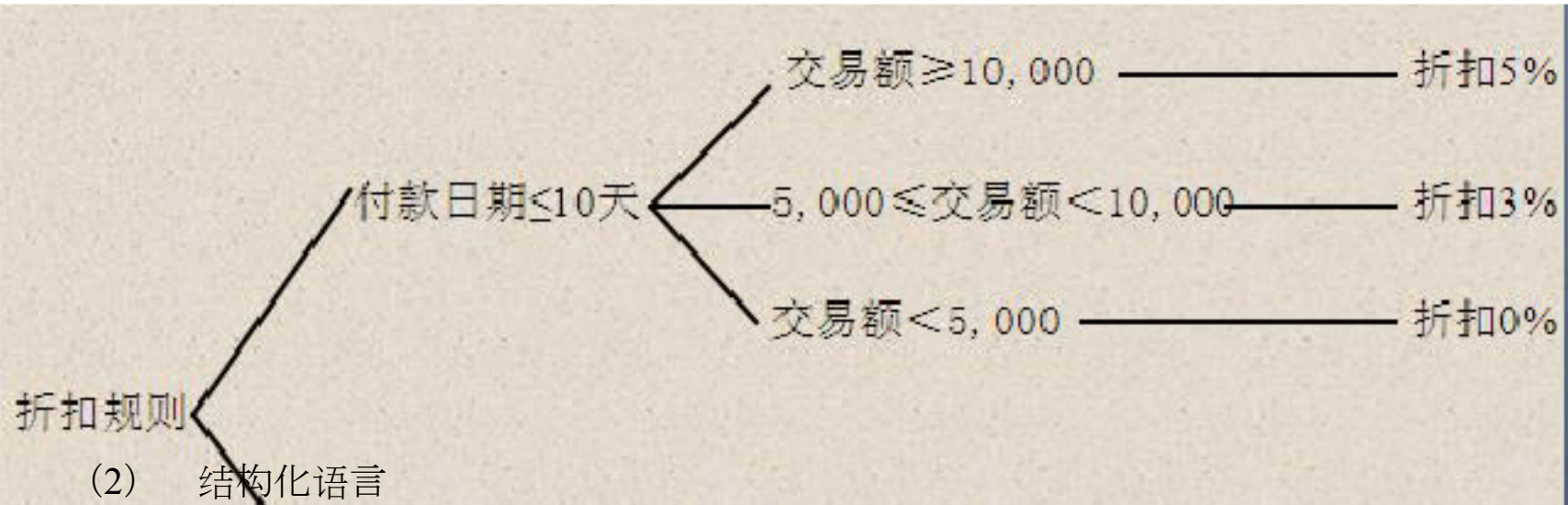
判断树 (Decision Tree)也称为决策树，是用来描述在一组不同的条件下，决策的行动是根据不同条件及其取值来选择的处理过程。业务规则的描述通常可以使用判断树这一过程描述工具。



【例题】画出顾客购货的折扣政策的决策树。

销售商在给顾客的折扣时，要考虑付款日期和交易额这两个因素。若付款日期在 10 天以内(含 10 天)，则当交易额超过¥10,000 时，给予 5%的折扣；当交易额在¥5,000 到¥10,000 之间(含¥5,000)时，给予 3%的折扣；当交易额低于¥5,000 时，没有折扣。若付款日期超过 10 天，则无论交易额多少，均不给任何折扣。

解析：



【例题】用结构化语言表达：顾客购货的折扣政策。

销售商在给顾客的折扣时，要考虑付款日期和交易额这两个因素。若付款日期在 10 天以内(含 10 天)，则当交易额超过¥10,000 时，给予 3%的折扣；当交易额在¥5,000 到¥10,000 之间(含¥5,000)时，给予 2%的折扣；当交易额低于¥5,000 时，没有折扣。若付款日期超过 10 天，则无论交易额多少，均不给任何折扣。

IF 付款日期在 10 日以上
折扣=0

ELSE

IF 交易额>=10000

折扣=3%

ELSE

IF 交易额>=5000

折扣=2%

ELSE

折扣=0

5. 需求验证

- (1) 验证每一个需求满足 5 个性质
- (2) 验证需求规格说明书满足 4 个性质

第二节 结构化设计

分为总体设计和详细设计

1. 总体设计的任务

把系统的功能需求分配到一个特定的软件体系结构中。

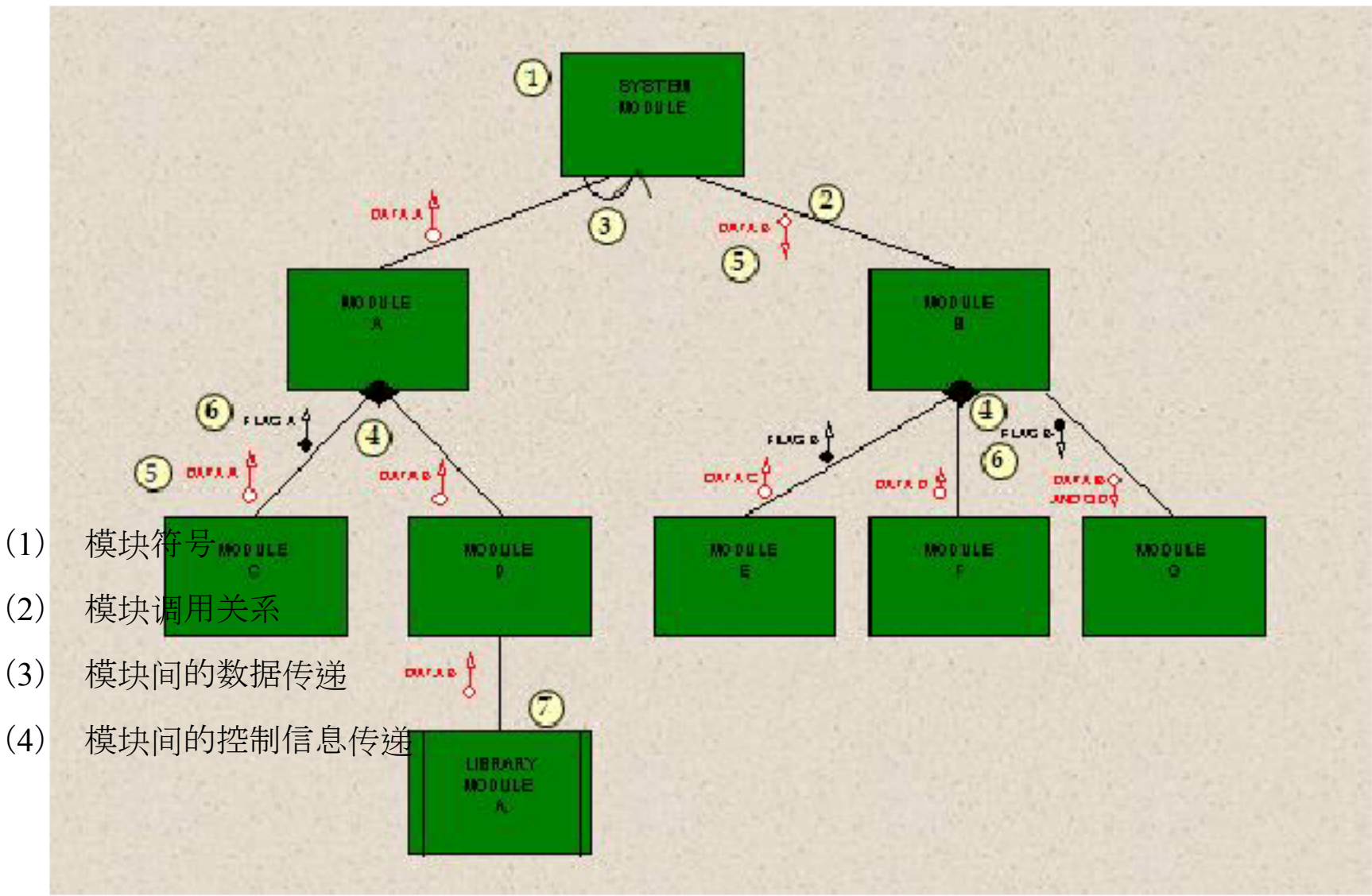
2. 表达软件体系结构的工具

- (1) 模块结构图
- (2) 层次图
- (3) HIPO 图

3. 模块结构图★

结构图(Structure Chart)是对软件总体结构的一种图形描述，它显示了软件的层次结构、组织和通讯。也就是说，在结构图中，显示了软件是由哪些模块组成的，这些模块按照什么样的层次结构组织在一起以及模块之间通过什么接口联系在一起。

结构图也称之为控制结构图、模块结构图或系统结构图。



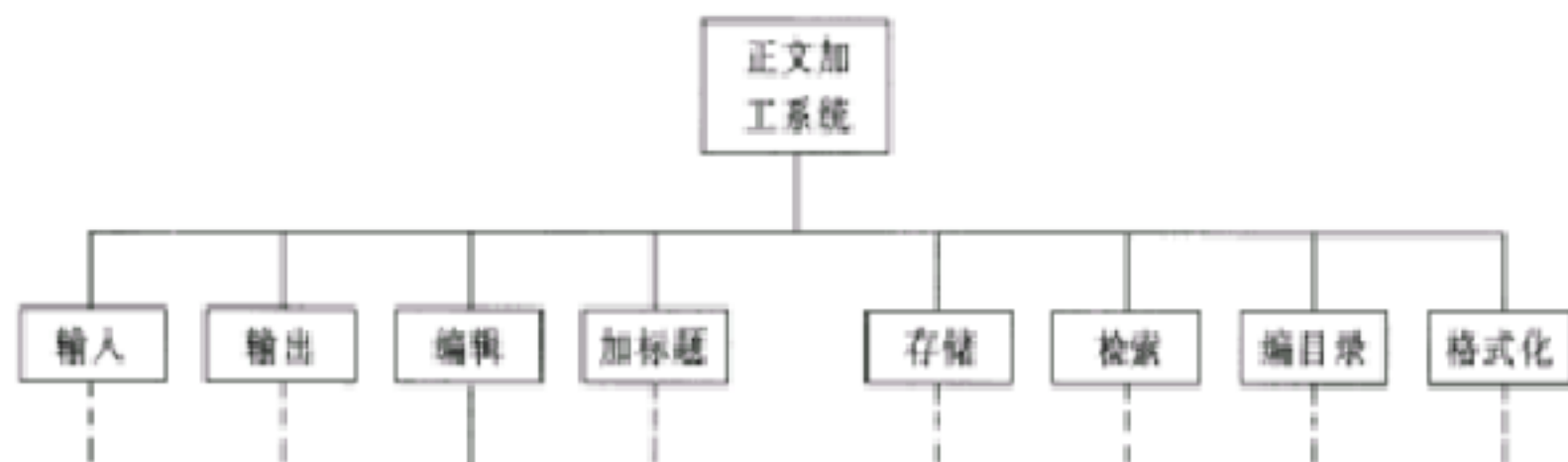
(5) 循环调用结构

(6) 选择调用结构

(7) 数据存储

4. 层次图

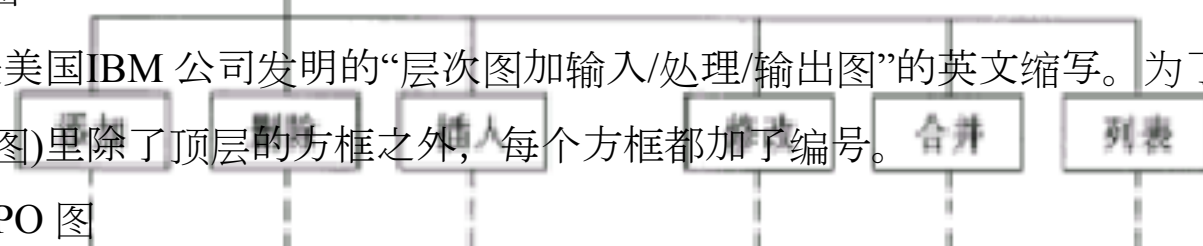
层次图中一个矩形框代表一个模块，框间的连线表示调用关系（位于上方的矩形框所代表的模块调用位于下方的矩形框所代表的模块）。



5. HIPO 图

HIPO 图是美国IBM 公司发明的“层次图加输入/处理/输出图”的英文缩写。为了使HIPO 图具有可追踪性，在H 图(即层次图)里除了顶层的方框之外，每个方框都加了编号。

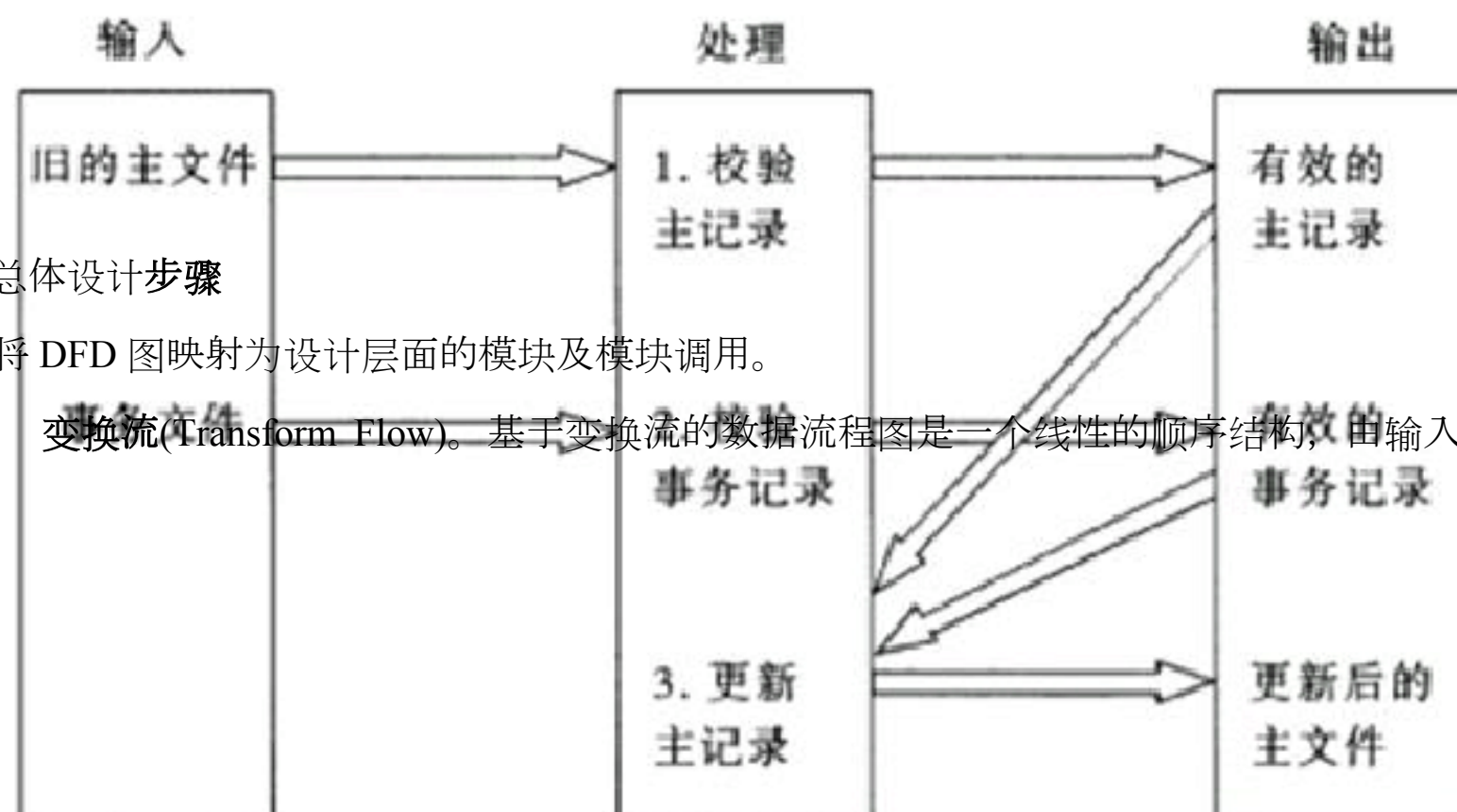
H 图+IPO 图



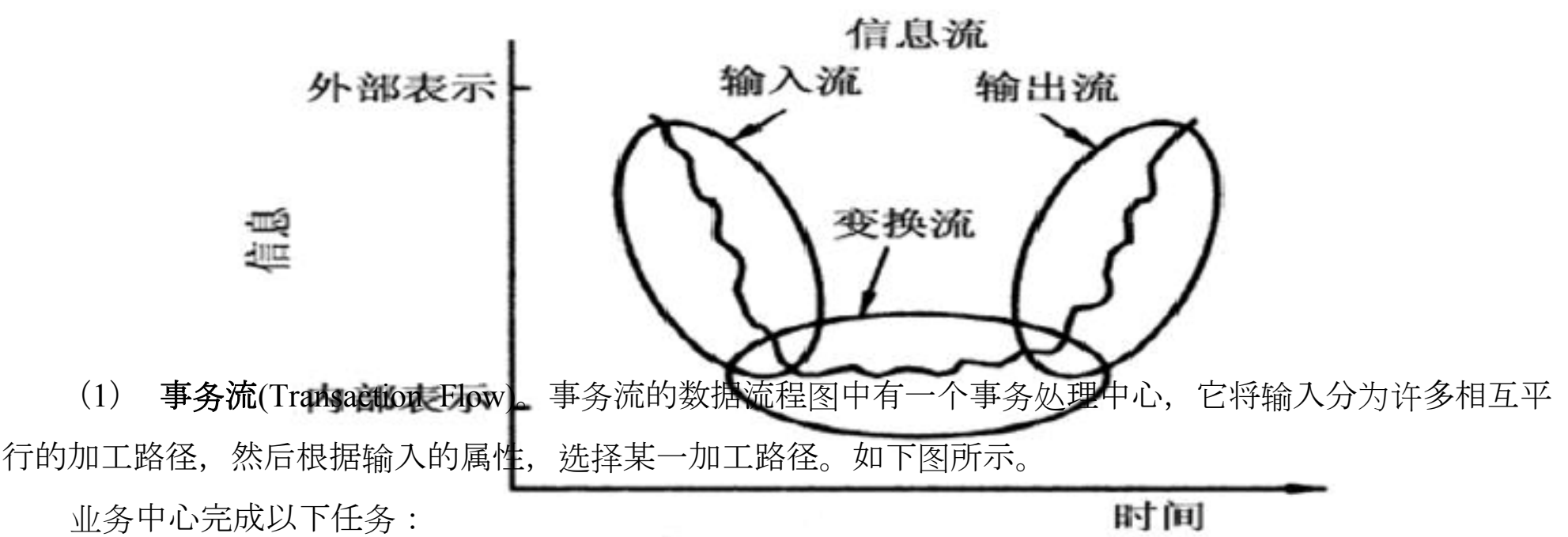
6. 总体设计步骤

将 DFD 图映射为设计层面的模块及模块调用。

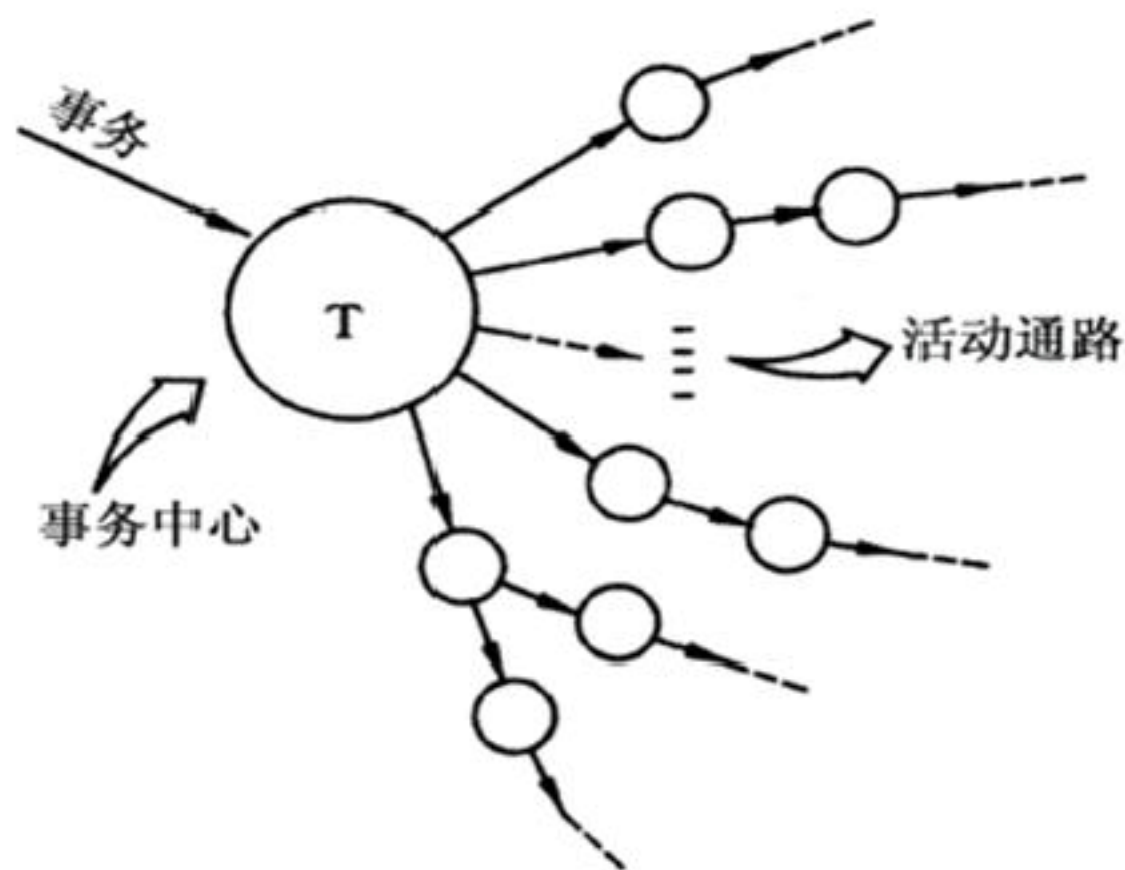
(1) 变换流(Transform Flow)。基于变换流的数据流程图是一个线性的顺序结构，由输入臂、输出臂和

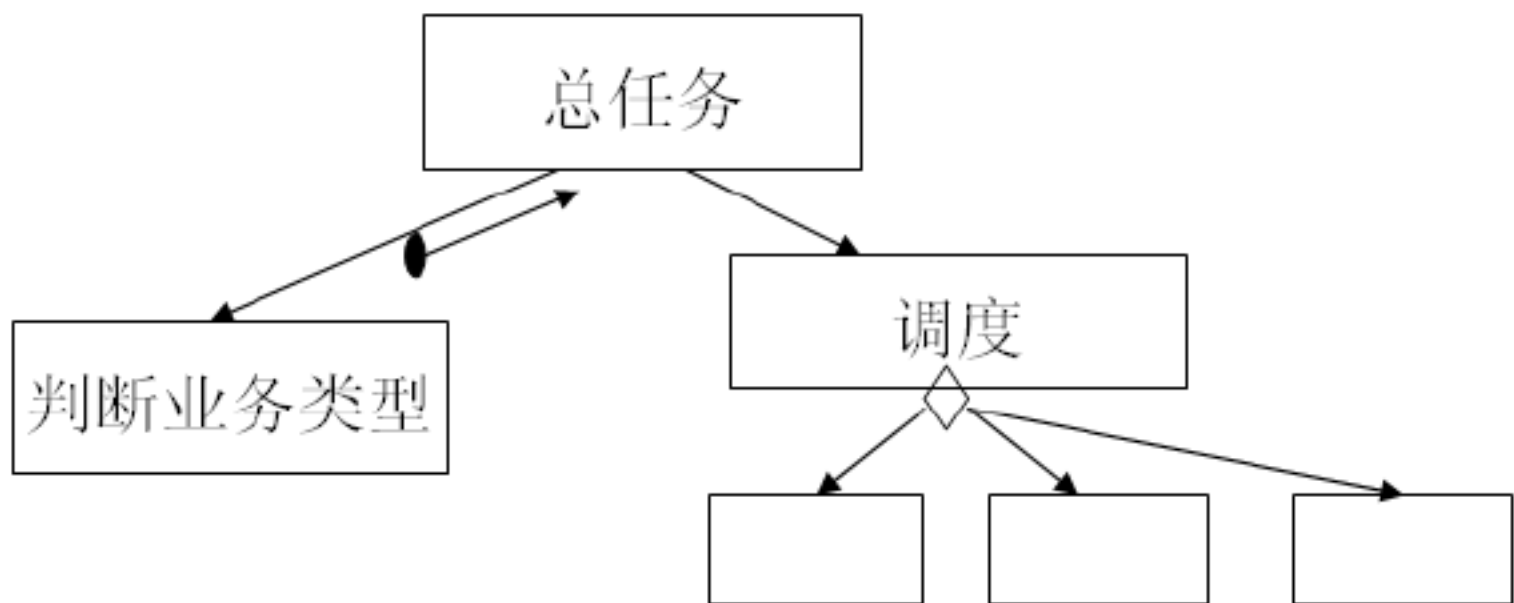


变换中心三部分组成。其中变换中心使系统数据发生本质的变化，输入臂将物理输入变换成逻辑输入，而输出臂则将逻辑输出变换成物理输出。



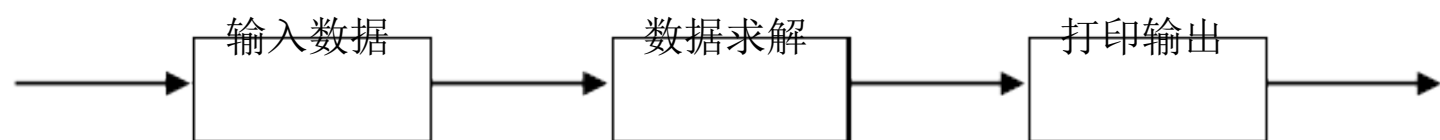
- (1)接收事务(即输入数据)；
- (2)分析每个事务并确定它的类型；
- (3)根据事务的类型选取一条活动通路。





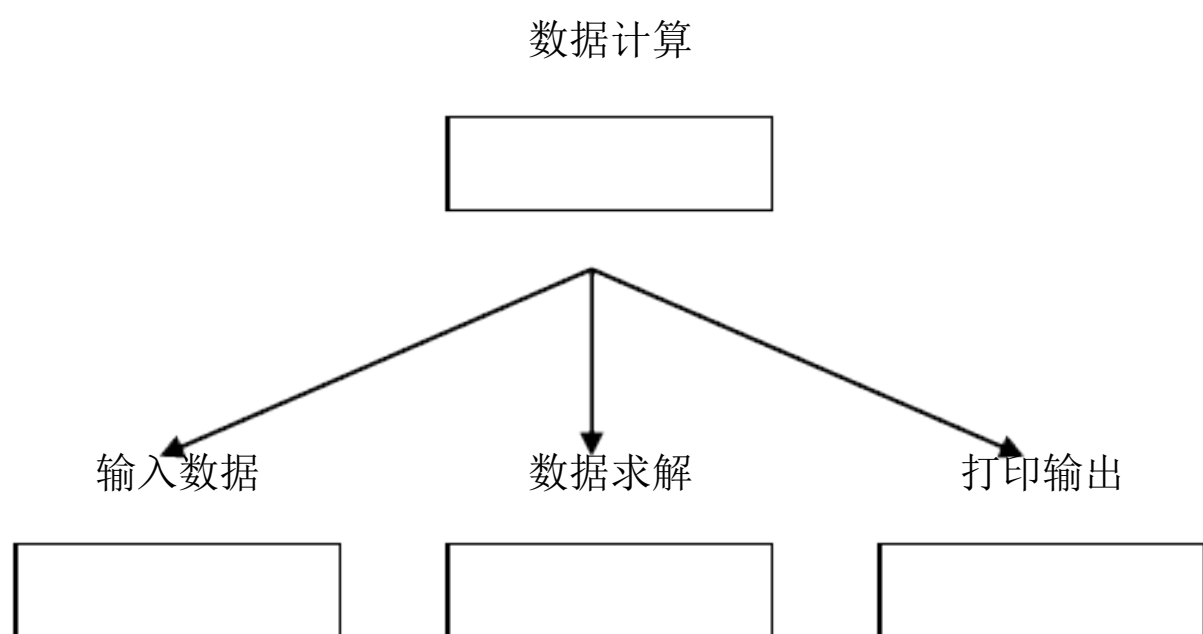
【例题】控制结构图的绘制

根据数据计算的数据流图：



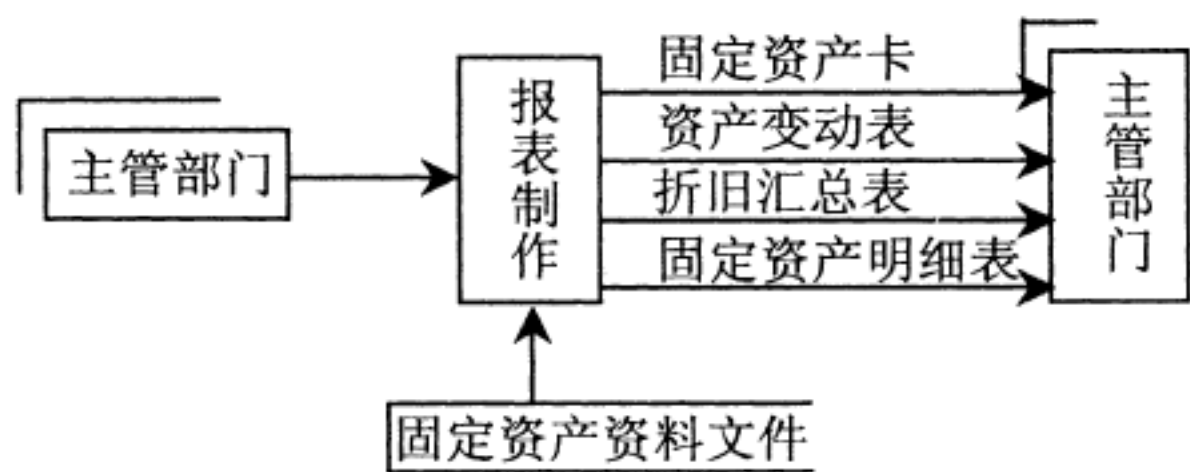
画出以转换为中心的控制结构图。

【解析】这是一个典型的以“转换为中心”结构的分解，可以转化为：



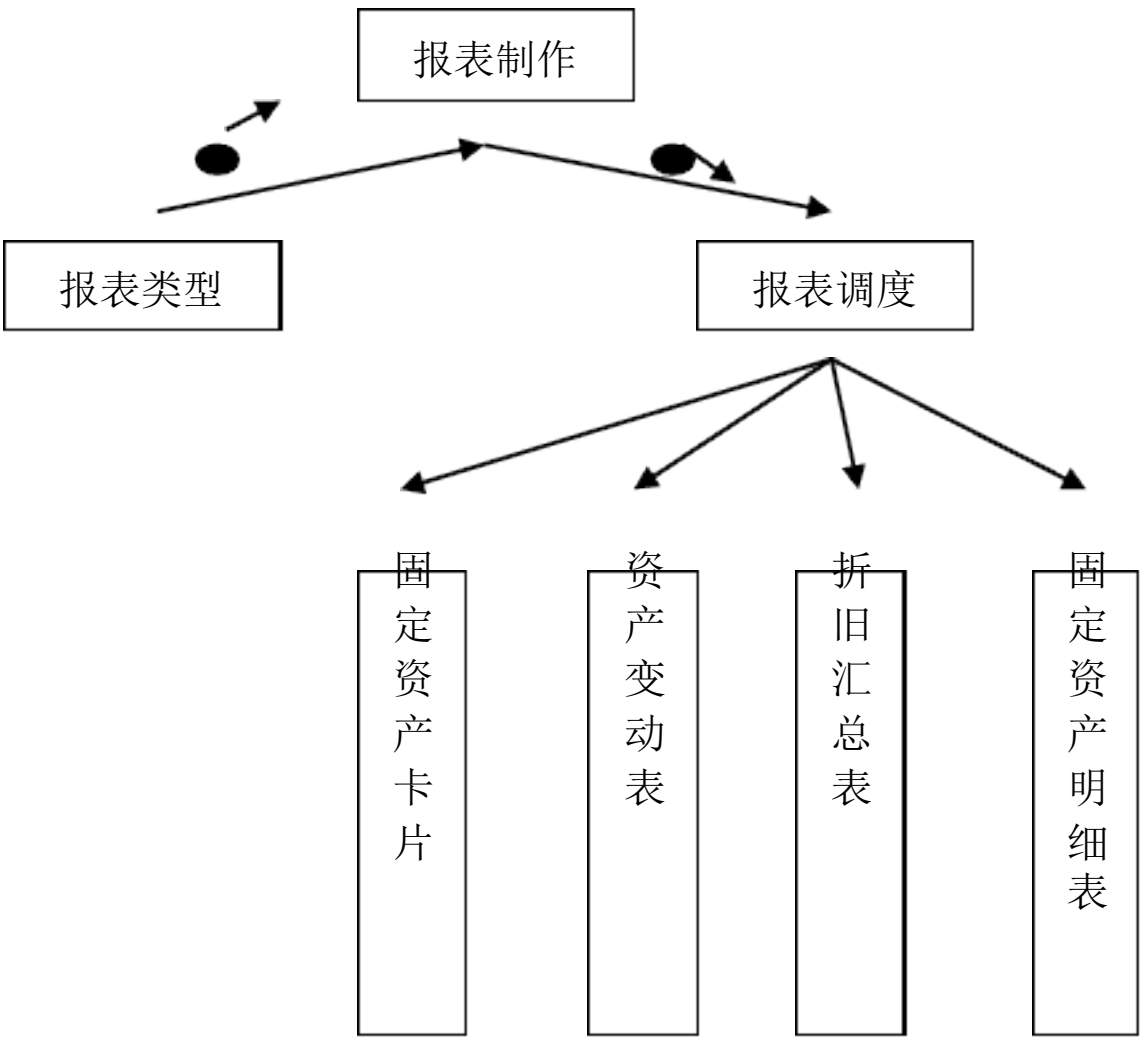
总结：任何处理都可以划分为两种转换类型之一：以转换为中心的分解和以业务为中心结构的分解。

【例题】产生固定资产资料数据流程图如下，做出以业务为中心的控制结构图。



【解析】

这是以业务为中心的处理，根据模板，可以转化为：



1. 模块

执行一个特殊任务的一个过程以及相关的数据结构。模块通常由两部分组成：模块接口和模块体。

2. 模块化

“分而治之”和“抽象”。

把一个待开发的软件分解成若干个简单的、具有高内聚低耦合的模块，这一过程称为模块化。模块化是系统设计基本原理/原则之一。

3. 内聚(Cohesion)

是指一个模块内部个成分之间相互关联程度的度量。也就是说，凝聚是对模块内各处理动作组合强度的一种度量。很显然，一个模块的内聚越大越好。

- (1) 偶然凝聚 可维护性最差
- (2) 逻辑凝聚
- (3) 时间凝聚
- (4) 过程内聚

-
- (5) 通信内聚
 - (6) 顺序凝聚
 - (7) 功能凝聚 可维护性最好

7. 模块耦合

耦合(coupling)是对两个模块之间联接程度的一种度量。模块间的依赖程度越大，则其耦合程度也就越大；反之，模块间的依赖程度越小，则其耦合程度也就越小。

很显然，为了使软件具有较好的可维护性和可修改性，模块间的关联程度即耦合程度应越小越好。因为耦合程度越小，表明模块间的独立程度越大，这样在修改一个模块时，对其它模块的影响程度就越小，从而使模块的修改工作局限于一个最小范围之内。

- (1) 内容耦合
- (2) 公共耦合
- (3) 数据耦合
- (4) 控制耦合
- (5) 标记耦合

原则是：尽量用数据耦合，少用控制耦合，限制公共耦合的范围，避免使用内容耦合。

8. 启发式规则

高内聚、低耦合。

- (1) 改进软件结构，提高软件独立性。模块分解
- (2) 模块规模适中
- (3) 力求深度、宽度、扇出、扇入适中。

深度：表示其控制的层数。

宽度：同一层次上模块总数的最大值。

扇出：一个模块直接控制的下级模块的数目。

扇入：有多少个上级模块直接调用它。

原则：顶层模块扇出比较大，中间层模块扇出较小，底层模块具有较大的扇入。

- (4) 尽量使模块的作用域在其控制域内。

模块的控制域：这个模块本身以及所有直接或间接从属它的模块的集合。

模块的作用域：受该模块内一个判断所影响的所有模块的集合。

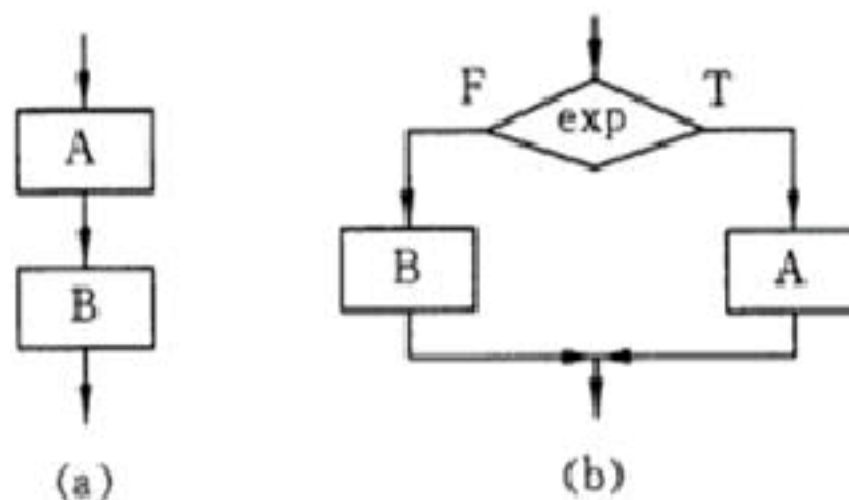
- (5) 尽力降低模块接口的复杂度
- (6) 力求模块功能可以预测

9. 详细设计

具体描述模块结构图中的每一模块，即给出实现模块功能的实施机制，包括一组例程和数据结构。

10. 结构化程序设计方法

一种基于结构的编程方法，即采用顺序结构、选择结构和重复结构进行编程，其中每一结构只允许一个入口和一个出口。



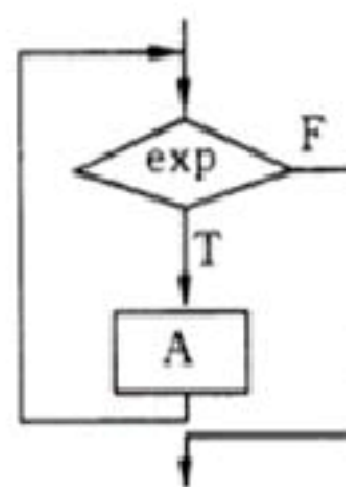
三种基本的控制结构：

(a) 顺序结构，先执行A 再执行B；

(b) IF-THEN-ELSE 型选择(分支)结构；

(c) DO-WHILE 型循环结构

或



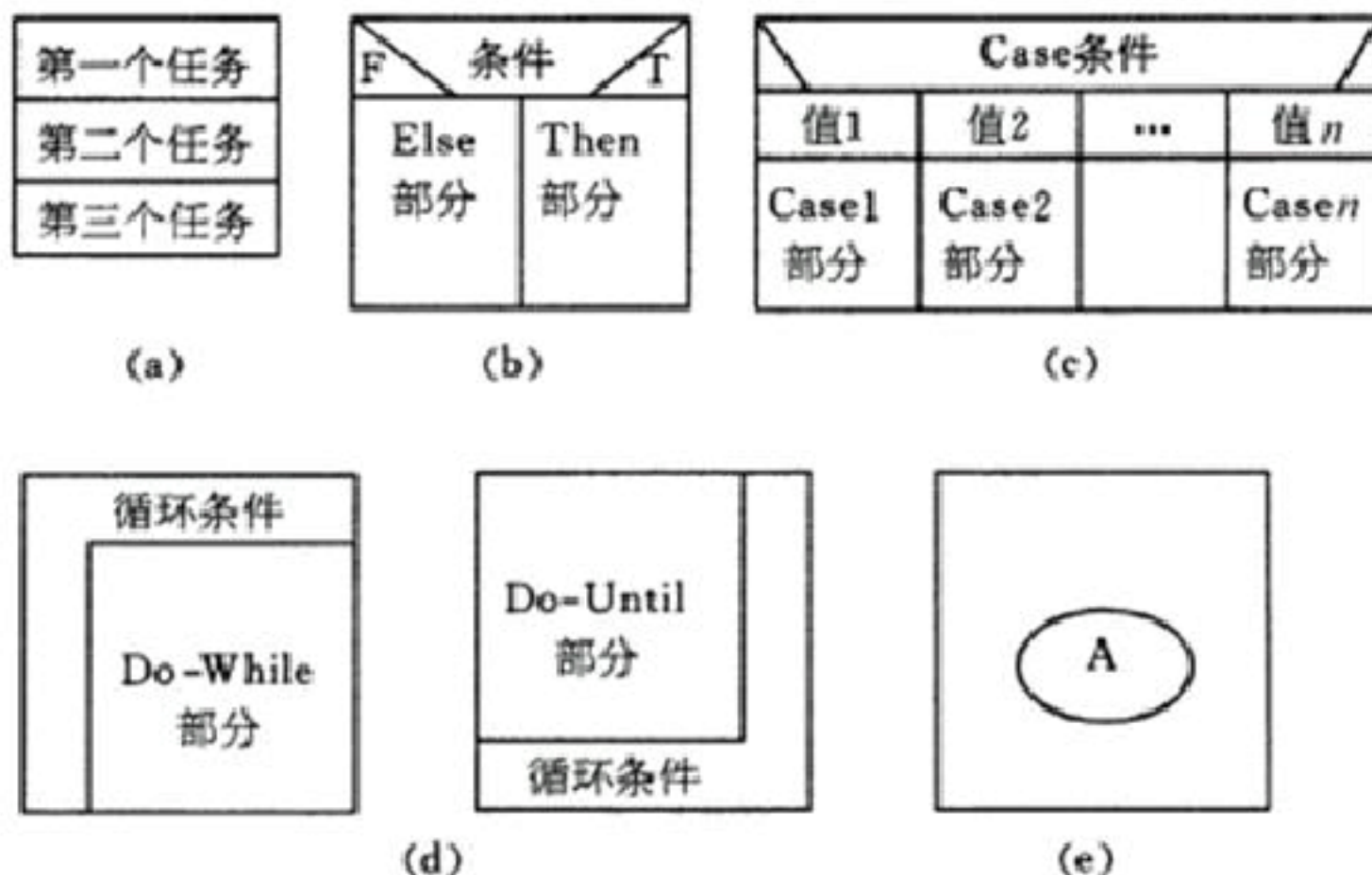
11. 详细设计工具

(1) 程序流程图

程序流程图：程序流程图又称为程序框图，它是历史最悠久使用最广泛的描述过程设计的方法，然而它也是用得最混乱的一种方法。

(2) 盒图 (N-S 图)

出于要有一种不允许违背结构程序设计精神的图形工具的考虑，Nassi 和 Shneiderman 提出了盒图，又称为 N-S 图。

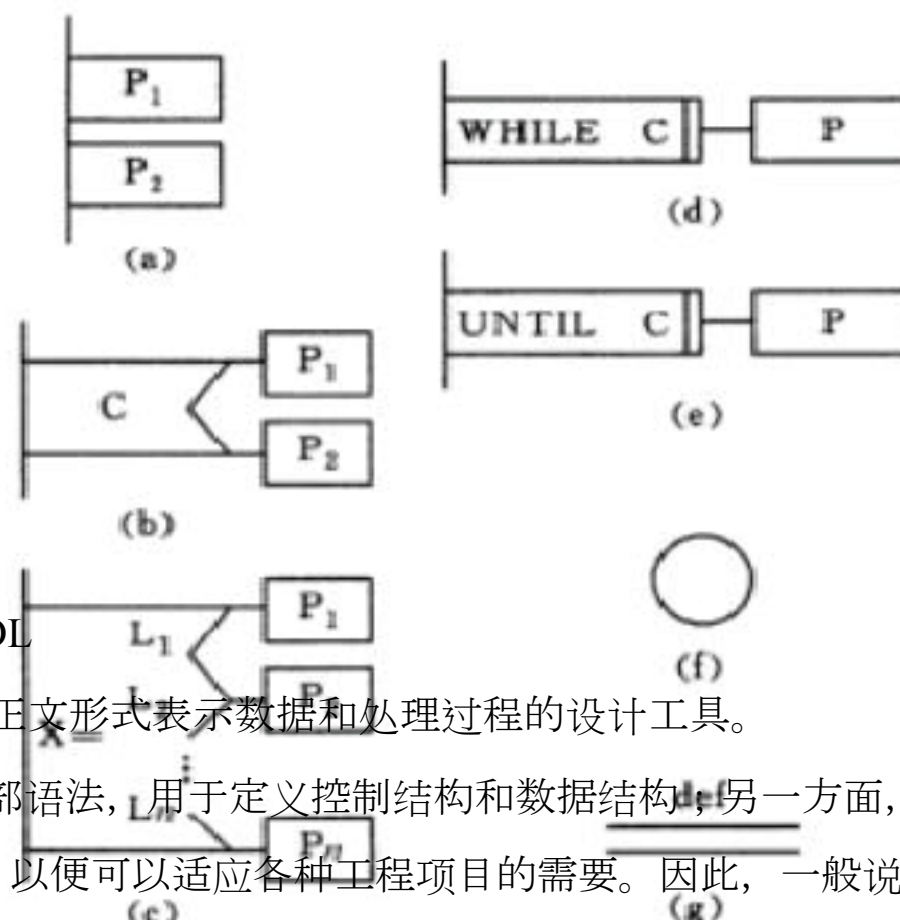


(a) 顺序；(b) IF-THEN-ELSE 型分支；(c) CASE 型多分支；

(d) 循环；(e) 调用子程序A

(3) PAD 图

PAD 是问题分析图(Problem Analysis Diagram)的英文缩写，自 1973 年由日本日立公司发明以后，已得到一定程度的推广。它用二维树形结构的图来表示程序的控制流，将这种图翻译成程序代码比较容易。下图给出PAD图的基本符号。



(4) 类程序设计语言PDL

PDL 也称为伪码，它是用正文形式表示数据和处理过程的设计工具。

PDL 具有严格的关键字外部语法，用于定义控制结构和数据结构；另一方面，PDL 表示实际操作和条件的内部语法通常又是灵活自由的，以便可以适应各种工程项目的需要。因此，一般说来 PDL 是一种“混杂”语言，它使用一种语言（通常是某种自然语言）的词汇，同时却使用另一种语言（某种结构化的程序设计语

言) 的语法。

可以作为注释工具直接插在源程序中间。

12. 设计规约

完整准确地描述满足需求规约所要求的所有功能模块，以及伴随功能模块而出现的非功能机制。设计规约包括概要设计规约和详细设计规约。

(1) 概要设计规约

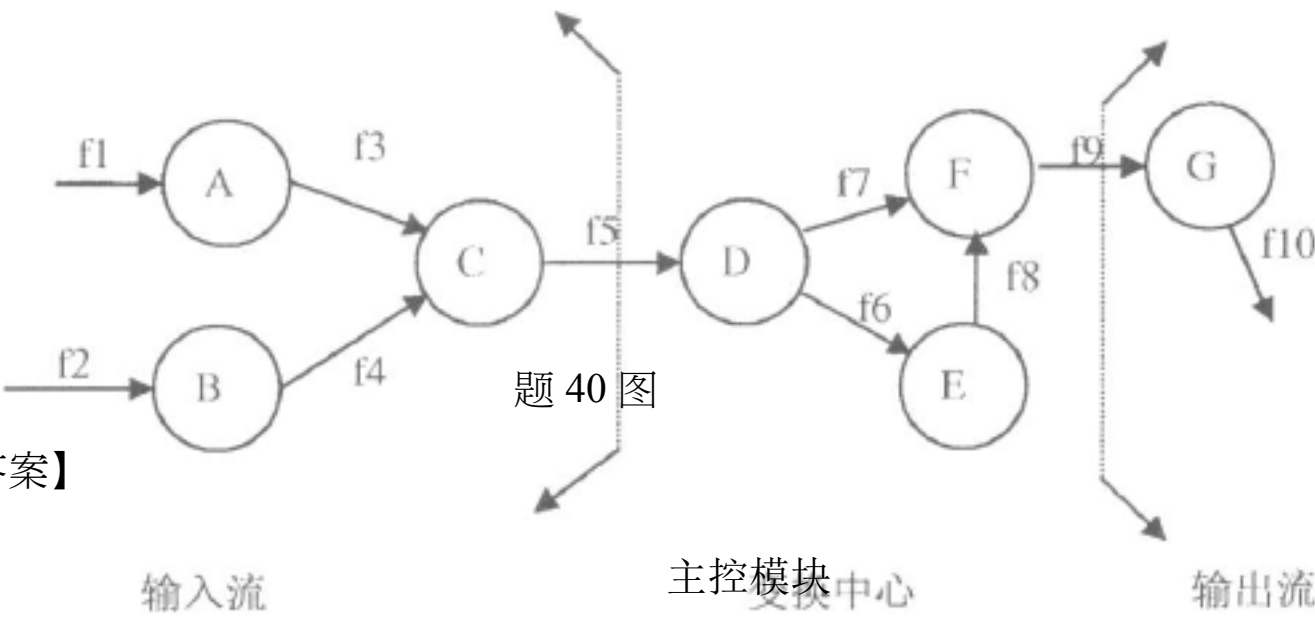
指明高层软件体系结构。

- 系统环境
- 软件模块的结构
- 模块描述
- 文件结构和全局数据文件的逻辑结构
- 测试需求

(2) 详细设计规约

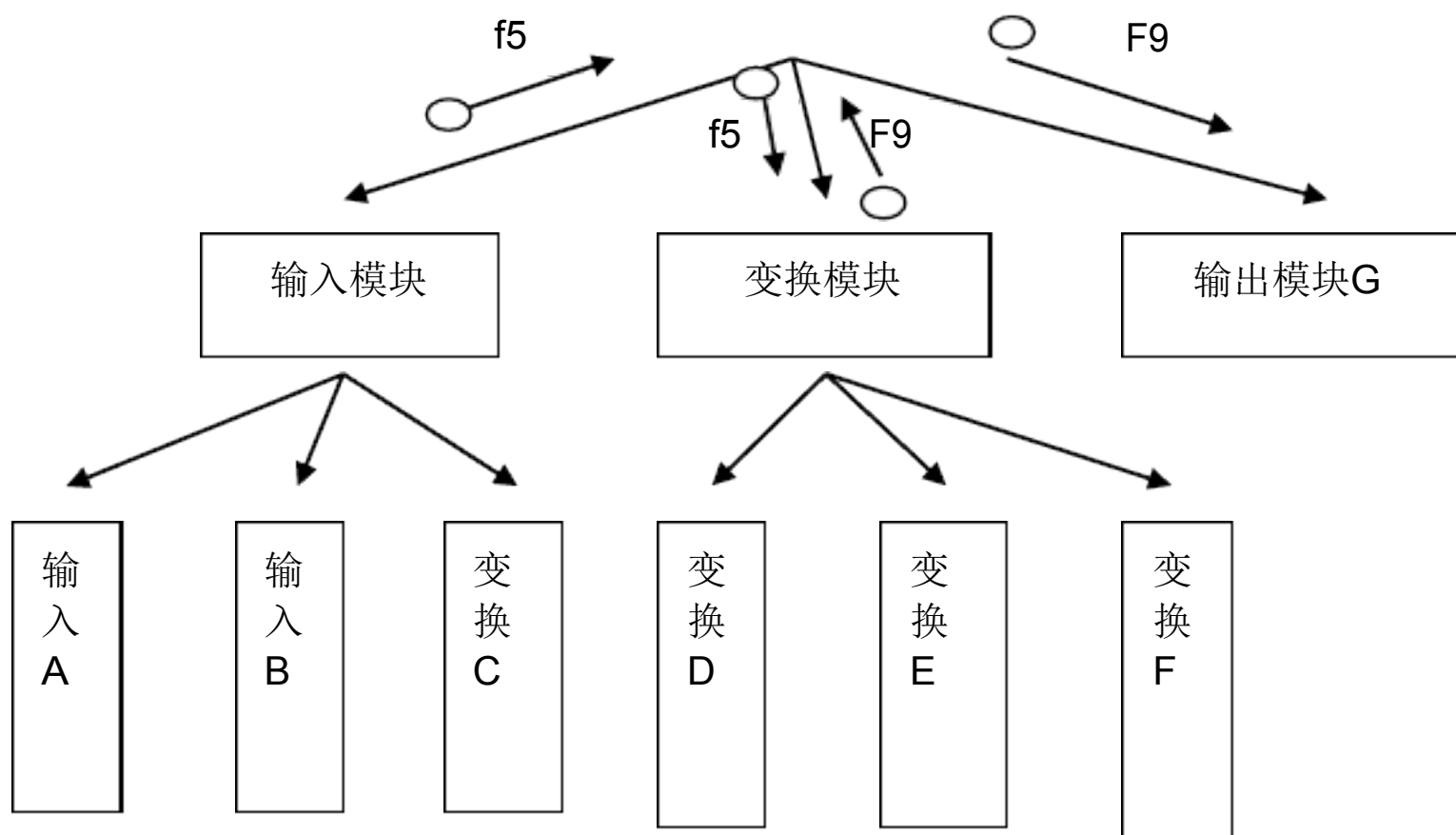
- 各处理过程的算法
- 算法所涉及的全部数据结构的描述

【例题】 根据下列变换型的数据流图，设计出初始软件结构图。



【答案】





【解析】这是一个典型的变换型数据流程图，将其转换为模块控制图时，第一层可以分解为三个模块：输入模块、变换模块、输出模块。每一模块还可以继续分解。

第四章 面向对象方法 UML

复习建议：

以不变应万变。

统一建模语言（Unified Modeling Language, UML）

UML 是目前流行的建模语言，特别是在网站开发中广泛应用。

UML 涉及很多的图，每一种图都有不同的图形符号、作用，在什么情况下用何种图来描述是本章的重点内容。

考核题目类型包括单项选择题、填空题、简答题，分值在 10%~15%之间。需要考生掌握各种 UML 图的作用。

面向对象建模过程的步骤：

- (1) 需求获取
 - a) 建立用况（use case）模型和用况场景
- (2) 需求分析
 - a) 建立活动图和状态图
 - b) 类图（建立域模型）

- (3) 编写需求规格说明书
- (4) 需求验证

第一节 UML 术语表

1. 对象（object）

对象（object）是系统中用来描述客观事物的一个实体。一个对象由一组属性和对这组属性进行操作的一组方法组成。

对象只描述客观事物本质的与系统目标有关的特征。

对象之间通过消息通信，一个对象通过向另一个对象发送消息激活某一个功能。

2. 类

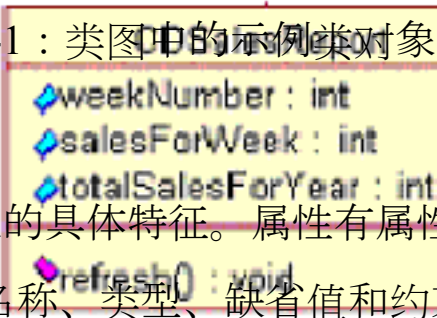
类（Class）是具有相同属性、操作、关系和语义的一组对象的集合，它为属于该类的全部对象提供了同一的抽象描述，其内部包括属性和服务两个主要部分。

类有超类（Superclass）和子类（Subclass）之分。

（相对而言）对象与类的关系犹如程序设计语言中变量和类型的关系。对象是类的实例（Instance）。

类在类图上使用包含三个部分的矩形来描述，如下图4-1 所示。最上面的部分显示类的名称，中间部分包含类的属性，最下面的部分包含类的操作（或者说"方法"）。

图 4-1：类图中的示例类对象



3. 属性

对象或类的属性（attributes）描述了对对象的具体特征。属性有属性名和属性值（或称属性状态）。

每条属性可以包括属性的可见性、属性名称、类型、缺省值和约束特性。

UML 规定类的属性的语法为：

可见性 属性名 ： 类型 = 缺省值 {性质串}

可见性：public (+)、protected (#)、private (-)、包内的 (~)

4. 类的操作

通常也被称为功能，但是它们被约束在类的内部，只能作用到该类的对象上。操作名、返回类型和参数表组成操作界面。

UML 规定操作的语法为：

可见性 操作名 （参数表）：返回类型 {性质串}

例如：+取客户地址（客户名:字符串）：字符串

5. 接口

接口是操作的一个集合，其中每个操作描述了类、构件或子系统的一个服务。

- (1) 采用具有分栏和关键字<interface>的矩形符号来表示
- (2) 采用小圆圈和半圆圈来表示

6. 协作

协作是一个交互，涉及交互的三要素：交互各方、交互方式以及交互内容。

7. 用况（use case）/用况

对一组动作序列的描述，系统执行这些动作应产生对特定参与者有值的、可观察的结果。

8. 主动类

至少具有一个进程或线程的类。能够启动系统的控制活动，并且其对象的行为通常与其它元素行为并发的。

表示方法：两条竖线。

9. 构件

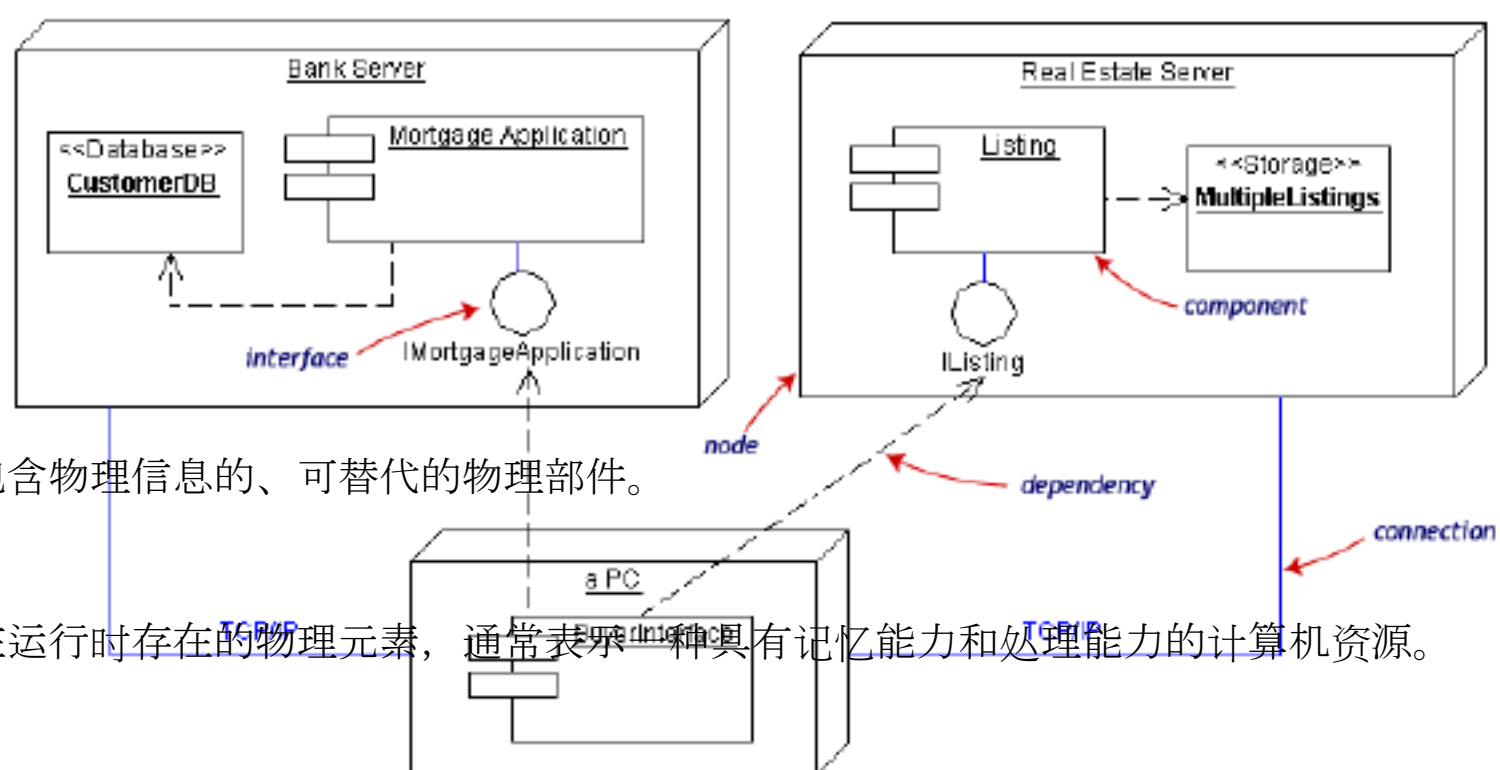
系统设计中的一种模块化部件，通过外部接口隐藏了它的内部实现。

10. 制品

系统中包含物理信息的、可替代的物理部件。

11. 节点

节点是在运行时存在的物理元素，通常表示一种具有记忆能力和处理能力的计算机资源。



12. 关联 (Association)

关联反映了类和类之间的静态关系。关联在模型中，特别是在永久业务对象模型中是最基本的关系。

链 (link) 是对象之间具有特定语义关系的抽象。

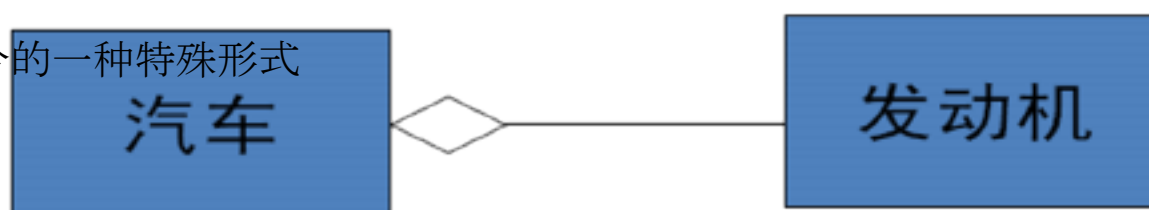
- (1) 关联名
- (2) 导航
- (3) 角色
- (4) 可见性

(5) 多重性：多重性 (Multiplicity) 定义了与一个对象/类相联系的对象/类出现一次，该对象/类可能出现的最小和最大的数目。

(6) 限定符

(7) 聚合：一个类是另一类的一部分。

(8) 组合：是聚合的一种特殊形式



(9) 关联类

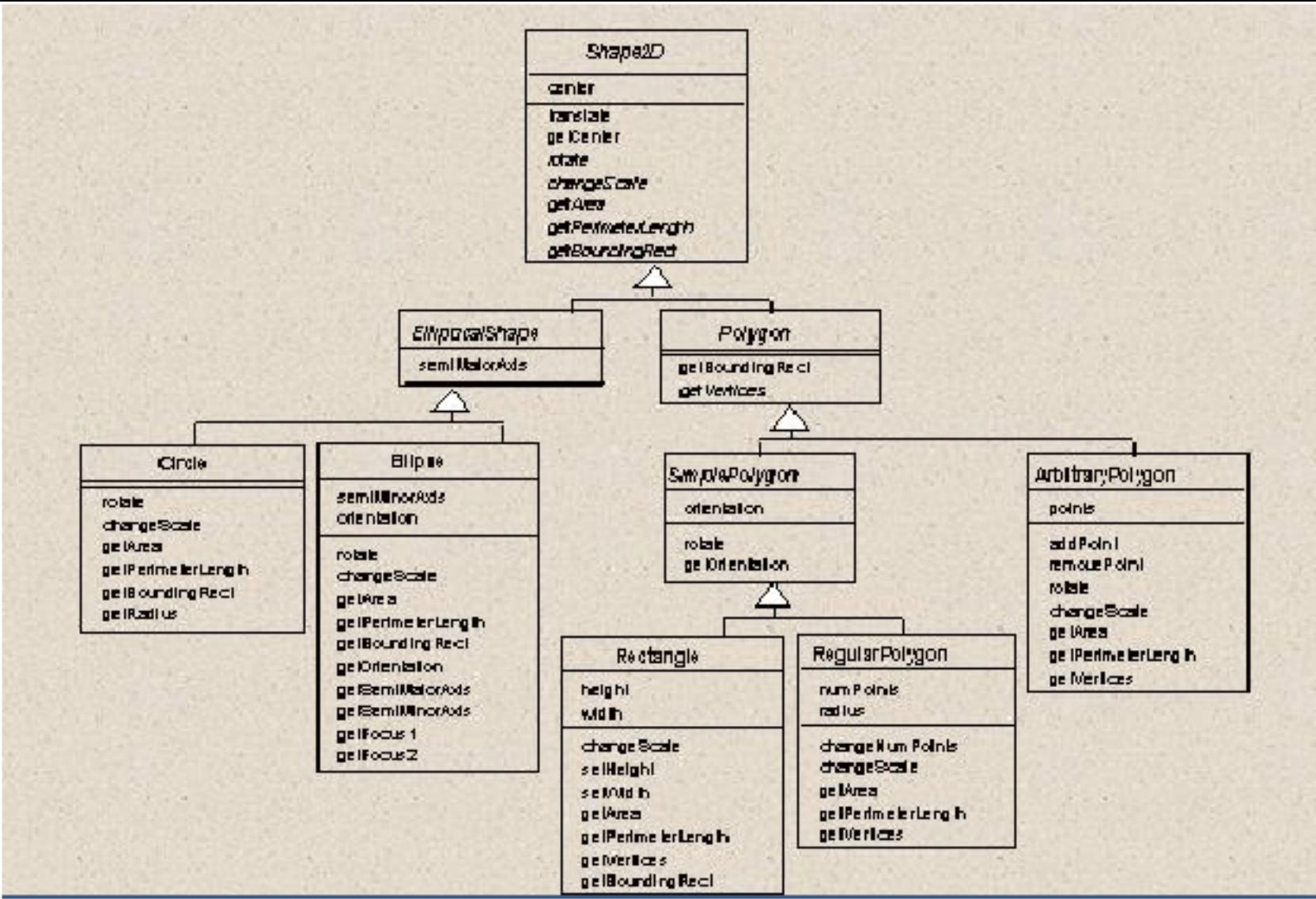
(10) 约束



13. 泛化/继承

继承：特殊类（子类）的对象拥有其一般类（超类）的全部属性与服务，称作特殊类对一般类的继承 (Inheritance)。利用继承 (inheritance)，子类可以继承父类的属性和方法。子类／父类也可分别叫做特殊类／一般类、子类／超类、派生类／基类等。

继承反映了类之间的一种联系或结构：一般-特殊结构，也称分类结构 (Classification Structure)，是由一组具有继承关系的类所组成的结构。仅由一些单继承关系的类形成的结构又称作层次结构 (Hierarchy Structure)；由一些存在多继承关系的类形成的结构又称作网格结构 (Lattice Structure)。



1. 多态性 (Polymorphism)

是指一般类中定义的属性或服务被特殊类继承之后，可以具有不同的数据类型或表现出不同的行为。这使得同一属性或服务名在一般类及其各个特殊类中具有不同的语义。

多态是指用同一界面形式表示不同对象类中的不同实现的能力。

多态性的实现基于两个基本原理：封装和泛化。

多态性实现的方法：

- (1) 泛化
- (2) 定义一个抽象类——接口类

2. 细化

细化是类目之间的语义关系，其中一个类目规约了保证另一类目执行的契约。用空心三角形的虚线表示。

3. 依赖

依赖是一种使用关系，用于描述一个类目使用另一类目的信息和服务。用有向虚线段表示。

4. 包

包是模型元素的一个分组，一个包本身可以被嵌套在其它包中，并且可以含有子包和其它类型的模型元素。

图形化工具。

图的类别：

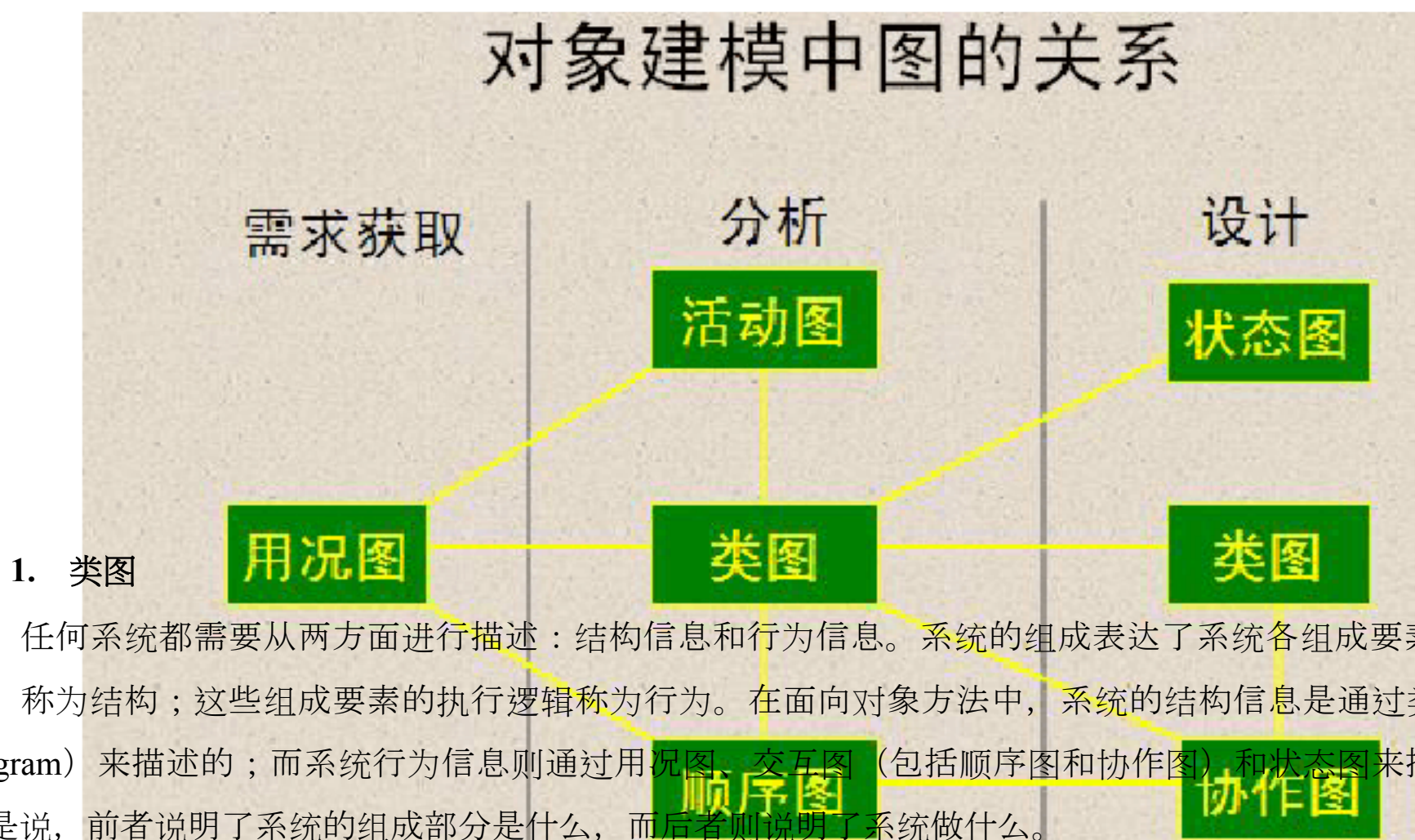
（一）结构图

- （1）对象结构建模—类图和对象图
- （2）应用结构建模—包图、构件图、部署图、组合结构图

（二）行为图

对象交互建模—顺序图、协作图（通信图、交互综述图、定时图）、状态图（状态机）

对象行为建模—用况图、活动图



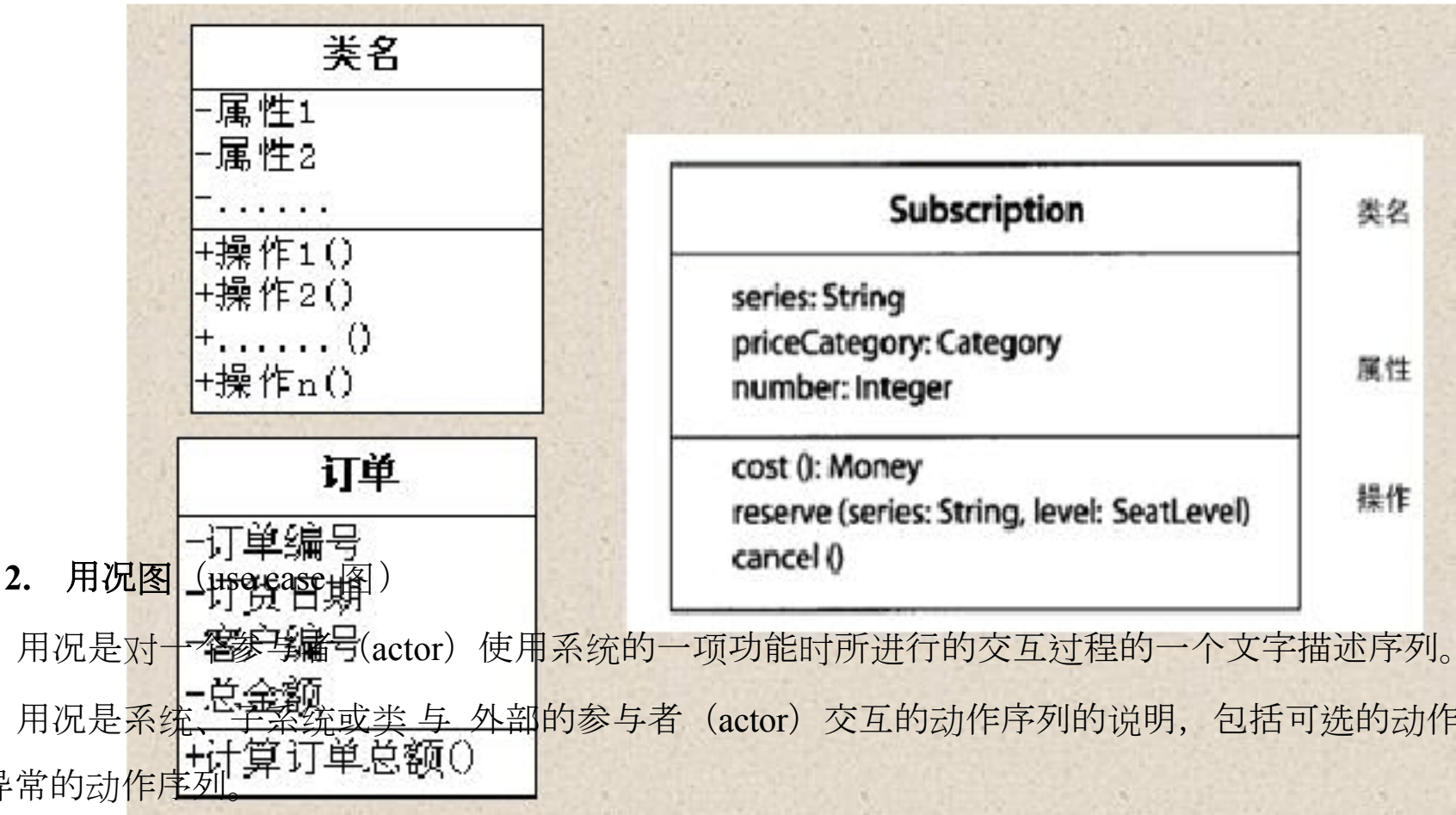
类图（class diagram）表达了系统的静态结构信息，即系统是由哪些类组成的，这些类之间的关系是什么。类图显示系统各个部分以及怎样将它们组装起来;但却不能模拟组装后系统的工作情况。

构造类图的三个关键问题是：

- （1）系统中有哪些需要关心的类？
- （2）这些类是如何描述的？
- （3）这些类之间的联系是什么？

创建一个系统的类图，要涉及 4 方面的工作：

- (1) 模型化待建系统中的概念，形成类图中基本元素
- (2) 模型化待建系统中的各种关系，形成该系统的初始关系。
- (3) 模型化系统中的协作，给出该系统的最终类图。
- (4) 模型化逻辑数据库模式

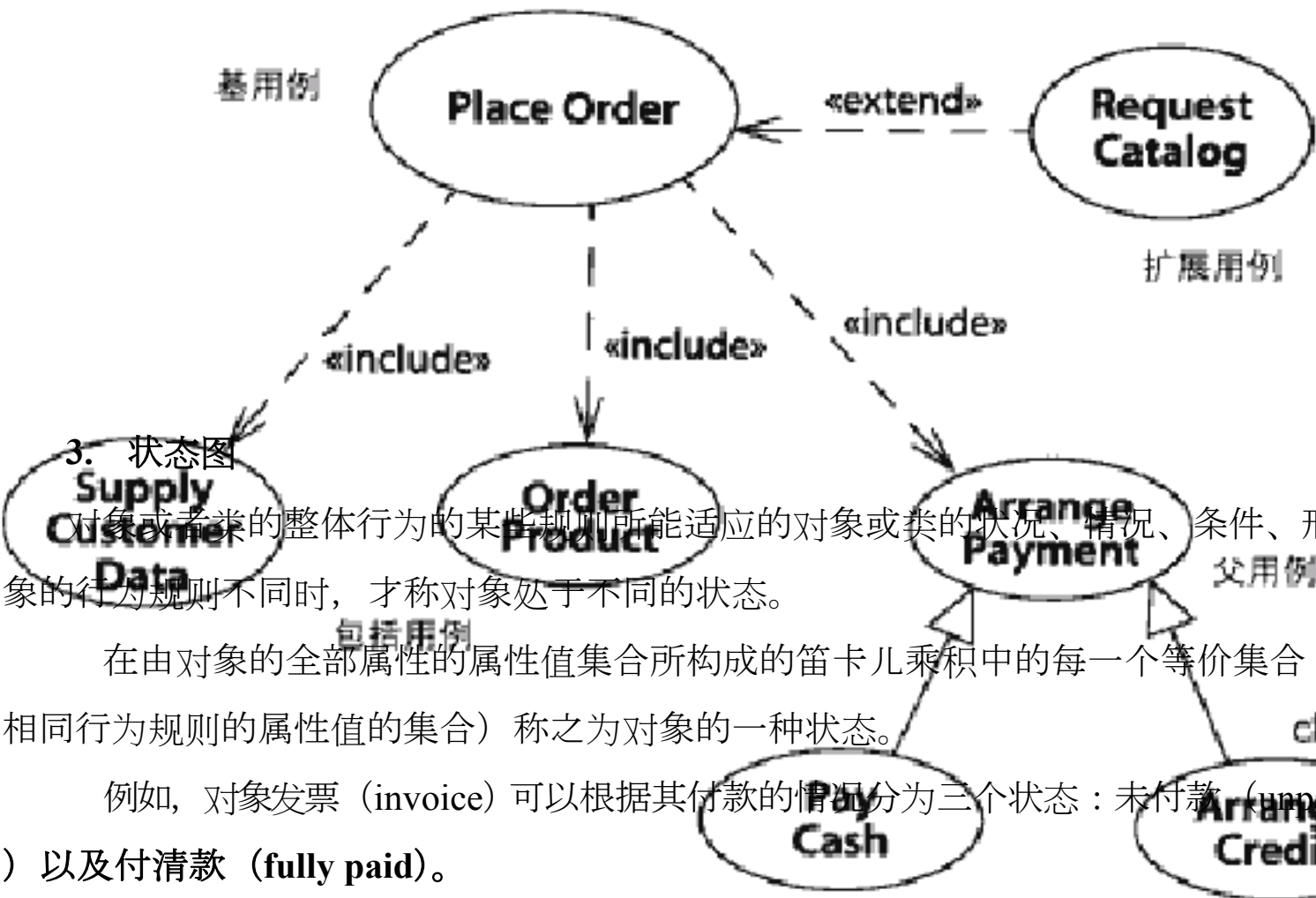


用况图 (Use Case Diagram) 是指反映活动者，系统边界所封闭的用况，及活动者与用况之间，用况与用况之间关系的一种图。

6 个模型元素：

- (1) 主题
- (2) 用况
- (3) 参与者：
 - 系统用户: 是最常见的一种角色。是直接使用系统的人。
 - 另一个系统：如DSS 可作为MIS 的一个活动者。补货系统可作为定单处理系统的活动者。
 - 时间：当经过一定时间触发系统中的某个事件时，时间就成了角色。例如定期的某些业务处理工作。
- (4) 关联
- (5) 泛化

关 系	功 能	表 示 法
关联	参与者与其参与执行的用例之间的通信路径	——
扩展	在基础用例上插入基础用例不能说明的扩展部分	«extend» - - - - ➔
用例泛化	用例之间的一般和特殊关系，其中特殊用例继承了 一般用例的特性并增加了新的特性	➡
包括	在基础用例之上插入附加行为，并且具有明确的描述	«include» - - - - ➔



3. 状态图

对象或者类的整体行为的某些规则所能适应的对象或类的状况、情况、条件、形式或生存周期。仅当对象的行为规则不同时，才称对象处于不同的状态。

在由对象的全部属性的属性值集合所构成的笛卡儿乘积中的每一个等价集合（即，使对象的服务呈现相同行为规则的属性值的集合）称之为对象的一种状态。

例如，对象发票（invoice）可以根据其付款的情况分为三个状态：未付款（unpaid）、部分付款（partly paid）以及付清款（fully paid）。

状态图（state chart diagram）使用状态、事件和转换来记录对象在其生命周期中所历经的状态序列。