第5章 需求建模方法与技术

第5章 需求建模方法与技术

需求建模主要是根据待开发软件系统的需求利用某种建模方法建立该系统的逻辑模型（也称需求模型或分析模型），以帮助软件开发人员检测软件需求的一致性、完全性、二义性和错误等。

软件需求建模应具备的特点：

（1）提供描述手段；

（2）提供基本步骤。

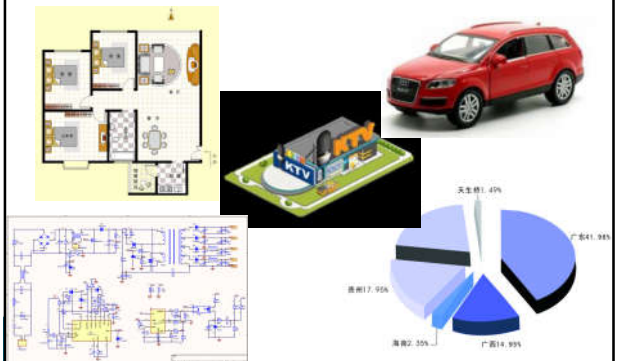
2

第5章 需求建模方法与技术

5.1 什么是模型
5.2 软件工程中的模型
5.3 结构化的需求建模方法
5.4 面向对象的需求建模方法
5.5 基于图形的需求建模技术

3

日常生活中的模型



4

5.1 什么是模型

- 模型的定义
  1. 由某些人根据其目的而对事物进行的抽象描述。
  2. 根据实物、设计图或设想，按比例或其他特征制成的同实物相似的物体。
  3. 当一个数学结构作为某个形式语言（即包括常符号、函数符号、谓词符号的集合）的解释时，称为模型。
  4. 为了理解事物而对事物作出的一种抽象，是对事物的一种无二义性的书面描述。

5

5.1 什么是模型

- 模型的分类
  1. 描述性模型
  2. 规约性模型
  3. 探测性模型

软件工程中的模型有的是描述性模型，有的是规约性模型，但大部分是描述性模型。作为特例，需求模型既是描述性模型（描述问题域），又是规约性模型（软件的需求规格说明）。

6

5.2 软件工程中的模型

- 软件工程中模型的概念
  - 对客观世界的问题领域进行抽象并用某描述方法给予表示的结果称为模型。
- 特点
  - 由于软件工程中多数模型是用于表示问题领域中的元素以及元素间的关系或相互作用等，故在建模过程中应该注意问题域中有什么对象，应该选择什么样的关系或动作，然后用适当的模型给予表示。

7

软件建模的目的

- 模型的作用
  - 有效地帮助人们更好地认识、应用、设计复杂事物
- 软件模型的作用
  - 更好地理解正在开发的系统
- 软件模型的产生
  - “数据结构+算法=程序”模式已不能适应现代软件的复杂度
- 软件建模的目的
  - 帮助我们按照实际情况或按我们需要的样式对系统进行可视化
  - 提供一种详细说明系统的结构或行为的方法
  - 给出一个指导系统构造的模板
  - 对我们所作出的决策进行文档化

8

5.2 软件工程中的模型

- 软件工程中模型的分类
  1. 开发过程模型
  2. 信息流模型
  3. 设计模型
  4. 交互作用模型
  5. 状态迁移模型
  6. 用于构造细节的原理模型
  7. 能力成熟度模型（CMMI）
  8. 其他模型

9

建模方法论的时代背景与要点

方法论	时代背景	建模要点
程序=数据结构+算法	出现于20世纪50-60年代，软件开发主要解决的是科学计算问题，Fortran语言是代表	选择合适的数据结构和算法显然是解决此类问题的关键
结构化分析与设计	出现于20世纪60-70年代，将解决一些与数据处理相关的问题，例如计费等。COBOL、C语言是代表	一是确定有哪些数据，格式是什么，如何存储，因此提供了E/R模型；二是确定数据的加工、处理过程，因此提供了DFD
面向对象分析与设计	出现于20世纪80-90年代，信息系统覆盖了更多业务过程	数据并不再是唯一的视角，业务流程、人的视角越来越重要，因此加入更多这方面的建模工具

10

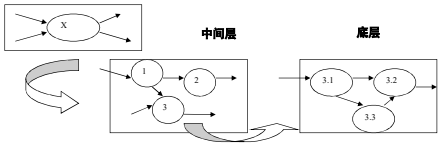
5.3 结构化的需求建模方法

- SA方法的特点
  1. 表达问题时尽可能使用图形符号的方式，这样即使非计算机专业人员也易于理解；
  2. 设计数据流图时只考虑系统必须完成的基本功能，完全不需要考虑如何具体地实现这些功能。

11

5.3.1 SA方法的基本思想

- 基本思想
  - 按照由抽象到具体、逐层分解的方法，确定软件系统内部的数据流、变换（或加工）的关系，并用数据流图给予表示。
- 复杂系统分解示例



12

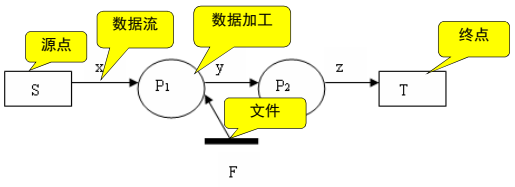
5.3.2 SA方法的描述手段

- 组成
  1. 一套分层的数据流图
  2. 一本词典
  3. 其他补充材料
- 数据流图（DFD：Data Flow Diagram）
  - 描述系统内部处理流程、用于表达软件系统需求模型的一种图形工具，亦即是描述系统中数据流程的图形工具。
  - DFD是一种历史悠久的业务流程建模方法，适用于数据流为主线索的处理过程。

13

5.3.2 SA方法的描述手段

- DFD的简例



14

5.3.2 SA方法的描述手段

- 数据流
  - 数据流是由一组数据项组成的数据，通常用带用带标识的有向孤给予表示。
  - 数据流可以从加工流向加工，或从源点流向加工，或从加工流向终点，或从加工流向文件，或从文件流向从加工。
  - 在数据流的命名中，不能使用缺乏具体含义的词如“数据”、“信息”等当作为数据流名。
  - 不能把控制流作为数据流。

15

5.3.2 SA方法的描述手段

- 加工（变换）
  - 对数据进行的操作或变换就称为加工。
- 加工的命名方法
  - 最高层的加工可以是软件系统的名字；
  - 加工的名字最好由一个谓语动词加上一个宾语组成；
  - 不能使用空洞或含糊的动词作为加工名；
  - 当遇到不能合适命名的加工时，可以考虑将加工分解。

16

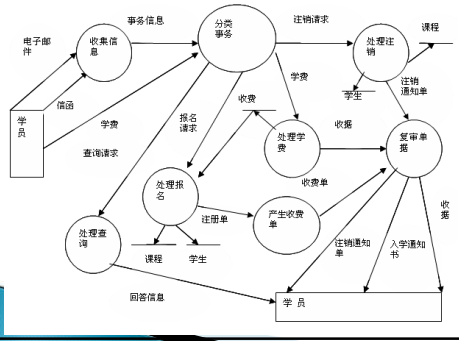
5.3.2 SA方法的描述手段

- 文件
  - 文件是存放数据的逻辑单位，且通常用图形符号“ ”，“ ”和“ ”分别表示加工要写文件，读文件和读写文件。另外，在这个图形符号中还要给出文件名。
- 源点和终点
  - 源点和终点用于表示数据的来源和最终去向，且通常用图形方框给予表示。

17

5.3.2 SA方法的描述手段

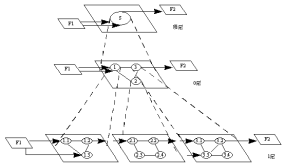
- 例：4.1节中某培训中心管理信息系统



18

5.3.2 SA方法的描述手段

- 分层的DFD
  - 对于大型而又复杂的软件系统，如果用一张DFD说出所有的数据流和加工，整个图就会变得相当复杂和难以理解，而且一张纸也难以写下这样的图。为了控制复杂性，通常可采用分层的方法。
- 分层DFD的组成
  - 顶层、底层和中间层。



19

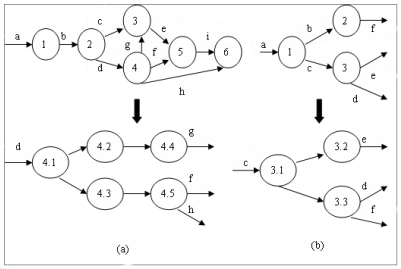
5.3.2 SA方法的描述手段

- 画完整的分层的DFD需注意的几个问题
1. 在画DFD时应区别于流程图
2. DFD的完整性问题
3. DFD的一致性问题
4. 在分层DFD中文件的表示
5. 分解层次的深度

20

5.3.2 SA方法的描述手段

- 一致性问题示例



21

5.3.2 SA方法的描述手段

- 分解层次深度的应验性准则
1. 某个加工的分解最好不超过7～8层，尽量减少分解层次；
2. 分解应根据问题的逻辑特性进行，不能硬性分解；
3. 每个加工被分解为子加工后，子图中的子加工数不要太多，通常为7～10个；
4. 上层可分解快些，下层应该慢些，因为上层比较抽象，易于理解；

22

5.3.2 SA方法的描述手段

5. 分解要均匀，即在一张DFD中，有些已是基本加工，另外一些还要被分解为多层；
6. 分解到什么程度才能到达底层DFD呢？一般来说应满足两个条件：一个是加工能用几句或十几句话就可清楚地描述其含义。另一个是一个加工基本上只有一个输入流和一个输出流。

23

5.3.2 SA方法的描述手段

- 画分层的DFD的步骤
1. 先确定软件系统的输入/出数据流、源点和终点；
2. 将基本系统模型加上源点和终点构成顶层DFD；
3. 画出各层的DFD。

24

5.3.2 SA方法的描述手段

- 画每张DFD时，应遵循的准则
  1. 将所有软件的输入/出数据流用一连串加工连接起来；
  2. 应集中精力找出数据流；
  3. 在找到数据流后，标识该数据流，然后分析该数据流的组成成分及来去方向，并将其与某加工连接；在加工被标识后，再继续寻找其它的数据流；

25

5.3.2 SA方法的描述手段

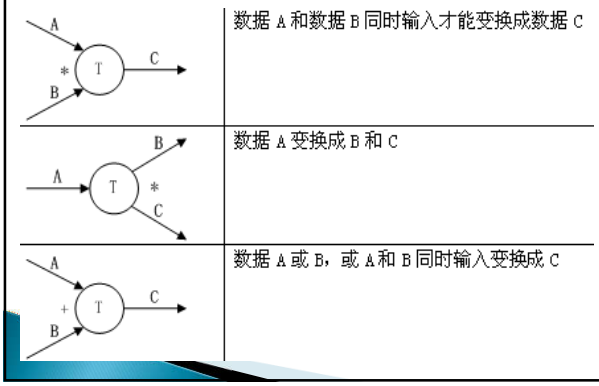
4. 当加工需要用到的共享和暂存数据时，设置文件及其标识；
5. 分析加工的内部，如果加工还比较抽象或其内部还有数据流，则需将该加工进一步分解，直至到达底层图；
6. 为所有的数据流命名,命名的要求见前；
7. 为所有加工命名的编号。

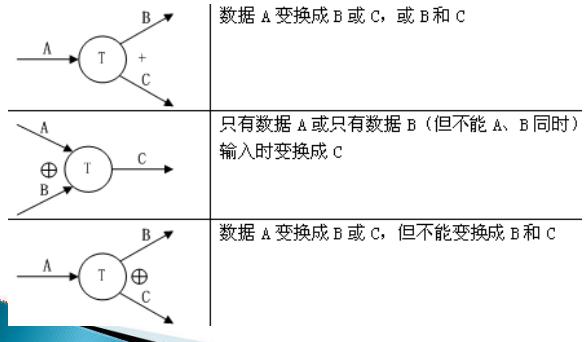
26

5.3.2 SA方法的描述手段

- 在画DFD时还应注意的情况
  1. 画图时只考虑如何描述实际情况，不要急于考虑系统应如何启动，如何工作，如何结束等与时间序列相关的问题；
  2. 画图时可暂不考虑一些例外情况如出错处理等；
  3. 画图的过程是一个重复的过程，一次性成功可能性较小，需要不断地修改和完善。

27





5.3.2 SA方法的描述手段

- 示例
  - 某医院拟开发一个分布式患者监护系统（PMS: Patients Monitoring System）。PMS将用于监视病房中每个患者的重要生理信号（如体温、血压、脉博信号等），并能定时更新和管理患者的病历。此外，当患者的生理信号超过医生规定的安全范围时，系统能立即通知护理人员，并且护理人员在需要时可随时通过系统产生某患者有关报告。

30

5.3.2 SA方法的描述手段

PMS的主要功能为：

① 通过一个病床监视器实现本地监测，以获得患者的生理信号。

② 在护士办公室实现中央监测

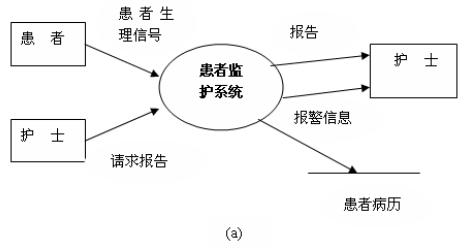
③ 更新和管理患者病历

④ 产生患者情况的报告以及报警信息

31

5.3.2 SA方法的描述手段

- 第0层数据流图

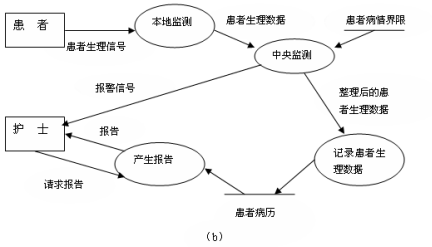


(a)

32

5.3.2 SA方法的描述手段

- 第1层数据流图

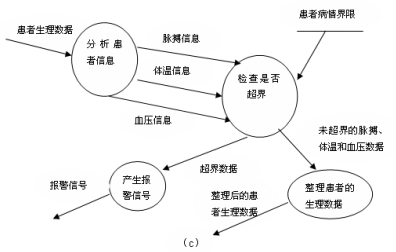


(b)

33

5.3.2 SA方法的描述手段

- 第2层数据流图



(c)

34

5.3.2 SA方法的描述手段

- 数据字典
  - 数据词典是由DFD中所有元素的“严格定义”组成。其作用就是DFD中出现的每个元素提供详细的说明，即DFD中出现的每个数据流名、文件名和加工名都在数据词典中应有一个条目以定义相应的含义。

35

5.3.2 SA方法的描述手段

- 数据词典中的条目类型
  - （1）数据流条目
    - 用于定义数据流，主要说明由哪些数据项组成数据流，且数据流的定义也采用简单的形式符号方式。

订票单＝顾客信息+订票日期+出发日期+航班号+目的地+……

    - 对于复杂的数据流，可采用向顶向下逐步细化的方式定义数据项。

顾客信息＝ 姓名+性别+身份证号+联系电话

36

5.3.2 SA方法的描述手段

- 当数据项由多个更小的数据元素组成时，可利用集合符号“{ }”给予说明。

  选修课程＝ 课程表+ 教师+ 教材

  课程表＝{ 课程名+ 星期 + 上课时间 + 教室 }

- 当某些数据项是几个不同的数据流的公用数据项时，可将它们列为专门的数据项条目。

  教室＝101| 102|.....

37

5.3.2 SA方法的描述手段

- 当所有出现在DFD中的数据流都给予定义后，最后的工作就是对出现在数据流中的数据项进行汇总，然后以表格的形式汇录每一数据项。

标识符	类型	长度	中文名称	来　源	去向
JS	整型	3	教室	教务处	学员

38

5.3.2 SA方法的描述手段

- （2）文件条目
  - 文件条目除说明组成文件的所有数据项（与数据流的说明相同）外，还可说明文件的组成方式。

如：航班表文件＝{ 航班号+ 出发地+ 目的地+ 时间 }

组成方式＝ 按航班号大小排列

39

5.3.2 SA方法的描述手段

- （3）加工条目
  - 加工条目主要描述加工的处理逻辑或“做什么”
  - 加工条目并不描述具体的处理过程，但可以按处理的顺序描述加工应完成的一些功能，而且描述加工的手段通常使用自然语言，或者结构化的人工语言，或者使用判定表或判定树的形式。

    某培训中心管理信息系统中的“处理报名”加工：

    ① 根据报名要求查询收费标准文件，确定相应费用。

    ② 学生注册

    ③ 根据选修课程登录课程统计文件

    ④ 产生注册单等

- 最后，由所有的数据条目、文件条目和加工条目就构成一本数据词典。

40

- 数据字典要求对数据元素（尤其是其结构）的描述要精确、严格和明确

符号	含义	示例
=	包含，由…构成	Name=first_name+last_name
+	指明序列结构	
()	内容可选	Phone_No.=(Area_No.)+Local_No.
[]	内容多选一	Number=[0|1|2|3|4|5|6|7|8|9]
|	分割[]内部的多个选项	
n{}m	循环,最少n次,最多m次	Area_No=3{Number}4
@	数据存储的标识符(关键字)	Student=@ID+Name+...
**	注释	Area_No=3{Number}4**区号为3到4位数字

41

数据字典为每个数据元素组织描述信息

名称	数据元素的原始名称
别名	数据元素的其他名称
使用地点	会使用该数据元素的过程
使用方法	该数据元素扮演的角色（输入流、输出流或者数据存储等）
使用范围	该数据元素存在的范围
描述	对数据元素内容的描述
单位/格式	数据元素的数据类型，可能事先设置的取值

42

名称	telephone number
别名	phone number, number
使用的地点和方法	read-phone-number (input) display-phone-number (output) analyze-long-distance-calls (input)
描述	telephone no. = [local extension | outside no. | 0] local extension = 3{0-9}7 outside no. = [service code | domestic no.] service code = [110 | 120| …] domestic no. = (area code) + local number area code = 3{0-9}4 local number= 7{0-9}8
格式	alphanumeric data

43

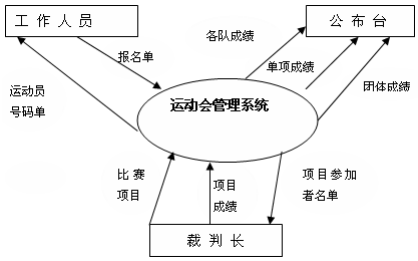
5.3.3 实例说明

- 某学校拟开发一个运动会管理系统。有关运动会的业务流程如下：
  - （1）确定运动会的举办时间和地点，设置哪些项目，报名时间等
  - （2）确定一些限制规定，如每人最多可参加几个项目，每个项目每队最多可由多少人参加，取前几名，打破单项比赛记录后的处理等。
  - （3）由各参加队提供报名单后，需给每个运动员编号，并统计每个项目的参加人数及名单，最后根据每个项目的参加人数等具体情况排出比赛日程。
  - （4）在运动会期间不断接受各项目的比赛成绩，及时公布单项名次，累计团体总分。
  - （5）比赛结束后，公布最终的团体名次。

44

5.3.3 实例说明

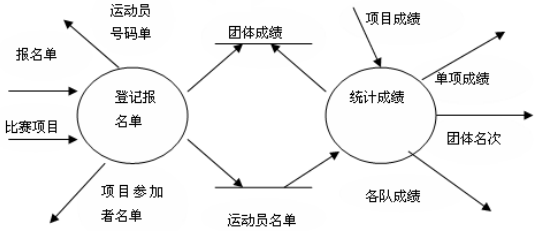
- 第0层数据流图



45

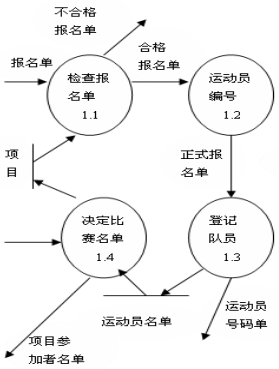
5.3.3 实例说明

- 第1层数据流图



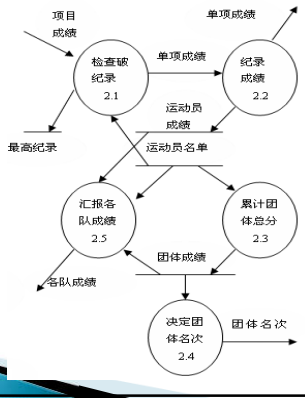
46

- 第2层数据流图



47

- 第2层数据流图



48

5.3.3 实例说明

- 数据字典说明

1. 数据流条目

数据流名	标识符	组成
项目成绩	ZMCJ	项目名+ {运动员号 + 成绩}
单项名次	DZMC	项目名+ {名次+ 运动员号+ 成绩+ 破记录}
单项成绩	DZCJ	项目名+ {运动员号+ 成绩+ 破记录}
各队成绩	GDCJ	队名+ 总分+ {运动员号+ 项目名+ 成绩+ 破记录}

49

5.3.3 实例说明

2. 汇总后的数据项

数据项名	值	位数
项目名	字符串	8
姓名	……	4
运动员号	1~ 4999	4
破记录	1 | 0	1
成绩	0 ~ 999	4
总分	正整数	4

50

5.3.3 实例说明

3. 文件条目

文件名	文件标识	组成	组织
团体成绩	TTCJ	队名 + 总分	按队名拼音顺序递增排列
运动员名单	YDYMD	队名+运动员号+ 姓名+ {项目名}	按运动员号递增排序
运动员成绩	YDYCJ	运动员号+项目名+成绩+{破记录}	同上

51

5.3.3 实例说明

4. 加工条目

加工名　　：记录成绩　编号：2.2　启动条件：收到单项成绩。

加工说明：a、取单项成绩

b、根据项目名、运动员号记录运动员成绩入运动员成绩文件中。

c、处理破记录情况

d、建立单项成绩表。

执行效率：100项/天

52

5.3.3 实例说明

加工名　　汇报各项成绩

编号：2.5　启动条件：接收到“汇报各队成绩”命令。

加工说明：a、查询团体成绩文件

b、为每队建立信息：各队成绩= 队名+部分名+ 成绩+ 破记录

c、根据队名查运动员名单和运动员成绩文件

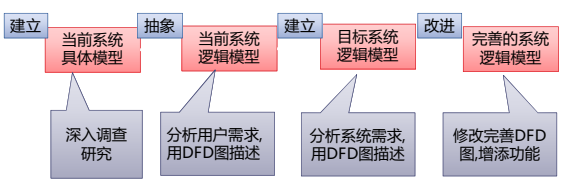
d、为每队填写信息：各队成绩= 运动员姓名+ 项目

e、建立各队成绩表。

执行频率：1次/天

53

5.3.4　SA方法的分析步骤



5.3.4 SA方法的分析步骤

- 将现实中已存在的人工系统称为当前系统，把待开发的计算机系统（主要是指软件系统）称为目标系统。
- 步骤
  - （1）理解和分析当前的现实环境，以获得当前系统的具体模型；
    - 具体模型必须忠实地反映人工系统的实际情况。软件开发人员在获取的需求信息的基础上，利用DFD将现实环境中的人工系统表达出来。在这样的DFD中，会有许多“具体”的东西，如人物、地点、名称和设备等。
  - （2）建立当前系统的逻辑模型；
    - 当前系统的逻辑模型应反映当前系统必须满足的性质，即当前系统“做什么”。此步的作用在于除去具体模型中非本质因素或一些“具体”因素。

55

  - （3）建立目标系统的逻辑模型；
  - 主要是分析目标系统与当前系统在逻辑系统的差别，并建立目标系统的逻辑模型。
  - 其具体步骤如下：
    - 首先，确定当前系统逻辑模型的“改变范围”。此步就是沿着当前系统逻辑模型的底层DFD，逐个检查每个基本加工。如果该加工在目标系统中不能实现或包含“具体”因素时，则这个加工属于改变的范围。这样，当前系统的逻辑模型变成了不需改变和需改变两个部分。当把需改变的部分进行修改后，就可获得目标系统的逻辑模型。
    - 其次，把“改变范围”视为一个加工，并确定此加工的输入/出数据流，当该加工比较抽象时，可将其进行逐层分解，然后画出各层的DFD。
    - 另一种方法是，首先建立目标系统的顶层DFD(0层)和第一层DFD（由若干子系统组成），然后再参照当前系统的逻辑模型，去掉其中所有“具体”因素和细化各子系统，最后可得到目标系统的逻辑模型。

  - （4）进一步完善目标系统的逻辑模型
    - 完善的工作大致为：
      - 至今尚未说明的处理细节，如出错处理、系统的启动和结束方式。
      - 某些需要的输入/出格式或用户界面的说明。
      - 增加性能需求和其他一些约束限制等。

57

练习

- 有一食品订购系统，其功能描述如下：
  - 顾客向系统提交食品订单，系统将食品订单发送给厨房，并将该食品订单信息、以及该食品订单所需的原材料清单信息存储起来。
  - 系统通过比对存储的食品订单信息和原材料信息，生成一份报告，发送给经理。
  - 经理可以通过系统发起原材料订单，原材料订单被发送给供应商，同时系统根据该原材料订单更新存储的原材料信息。
- 请根据上述描述，绘制出该系统的数据流图。

58

面向对象的需求分析

- 面向对象(Object-Oriented,缩写为OO)方法学的出发点和基本原则是尽可能模拟人类习惯的思维方式，使开发软件的方法与过程尽可能接近人类认识世界解决问题的方法与过程。
- 面向对象方法实际上是一整套的软件开发方法，它包括面向对象的分析OOA、面向对象的设计OOD、面向对象的编程OOP、面向对象的测试OOT。

59

5.4 面向对象的需求建模方法

- 面向对象的需求建模方法到目前为止已有许多不同的版本，其中具有代表性的是面向对象建模技术 OMT、面向对象的软件工程 OOSE、面向对象的分析/设计方法 OOAD和统一建模语言 UML等。
- 在这些需求建模方法中，本节将主要介绍基于OMT的需求建模方法，而且重点放在需求建模方面。
- 面向对象的需求建模方法的关键是从获取的信息中识别出问题域中的类和对象，并分析它们之间的关系，最终建立起简洁、精确和易理解的需求模型。

60

5.4.1 面向对象方法中的一些基本概念

- 对象
  - 对象就是客观实体的抽象，并且是构成概念模型的基本单元。
  - 根据现实中客观实体性质的划分，对象属于如下的几种类型：
    - 人类能感知的物理实体，如房子、汽车、飞机、山川等。
    - 人或者组织，如教师、学生、医生、大学、科研处等。

61

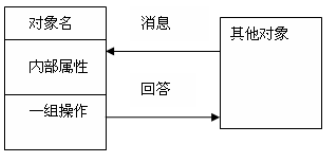
5.4.1 面向对象方法中的一些基本概念

  - 现实中发生的事件，如读书、上课、演出、访问、购买等。
  - 需要说明的概念，如政策、法律、规章制度等。
- 在面向对象的方法中，对象是指封装数据结构及可以施加这些数据构上的操作的封装体。这个封装体，可以唯一标识它的名字，而且向外可提供一组服务。

62

5.4.1 面向对象方法中的一些基本概念

对象图形表示如下：



63

5.4.1 面向对象方法中的一些基本概念

- 类
  - 类是对具有相同性质和操作的一个或多个对象的描述，并且是一组对象的集合。
  - 类与对象的关系是：类给出了该类中所有对象的抽象定义（主要指属性和内部操作两个部分），而对象是符合该定义的一个实例。
  - 在软件开发中，对象是动态的运行实体，它需要分配存储空间来存放属性的实际值和操作代码。
  - 类具有层次性：
    - 一个类上层可有父类，下层可有子类，从而形成类的层次结构。
    - 如:“研究生”类的上层类是学生，下层类可以是“硕士研究生”类等。

64

5.4.1 面向对象方法中的一些基本概念

- 继承
  - 是指能够直接获得已有的性质和特征，而不需要重复定义它们，继承主要是由父类与子类的关系引起的，其中子类除了具有自己的属性和内部操作外，还可继承父类的全部属性和内部操作。
    - 单一继承：指一个子类只允许有一个父类。
    - 多重继承：指一个子类有多个父类。

65

5.4.1 面向对象方法中的一些基本概念

- 消息
  - 消息是系统运行过程中对象之间相互传递的、请求服务的信息。消息实际上就是一段数据结构，通常包括接受消息的对象名、请求的服务名称、输入参数和应答信息等。通过消息实现对象之间的通信。
  - 一个对象对外服务所要求的消息格式称为消息协议。只有满足该协议，对象才能识别和提供服务。

66

5.4.1 面向对象方法中的一些基本概念

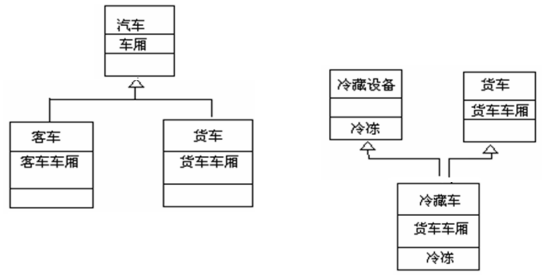
类之间的关系

（1）类之间的泛化关系

这种关系主要是因类之间继承关系而形成的类层次结构，一种存在于一般类别和特殊类别之间的分类关系。在OOM中用“一般 — 特殊”或“Is-a”这一关系来定义这种结构。

泛化关系的表示：用从特殊类指向一般类的空三角箭头表示，多个泛化关系可以用箭头线组成的树来表示，每一个分支指向一个特殊类。

67

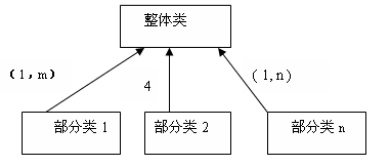


68

5.4.1 面向对象方法中的一些基本概念

（2）类之间组成关系

这种关系是因一个对象是另一个对象的组成部分而形成的结构关系。在OOM中用“整体-部分”或“part-of”这一关系来定义这种结构。

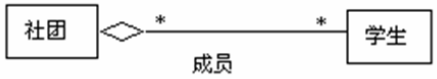


组成关系的表示

69

聚合关系

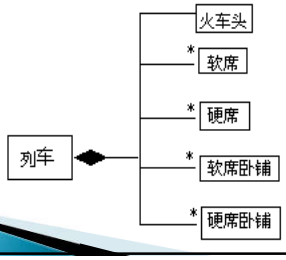
- 是一种弱关联关系，它所描述的“部分”对象不依赖于“整体”对象的。
- 是通过在关联的一端加上一个菱形符号来表示，其中与菱形相连的类“包含”与另一关联端点相连的类。



70

组合关系

- 是一种强关联关系，它所描述的“部分”对象是依赖于“整体”对象的。
- 组合关系有两种表现形式：
  - 在任意给定时刻，一个“部分”对象只能属于一个组合对象；
  - 当一个组合对象被撤销时，所有依赖于这个组合对象的“部分”对象都将被同时撤销。



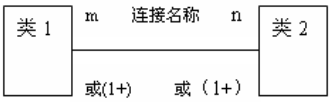
71

5.4.1 面向对象方法中的一些基本概念

对象属性间的静态关系

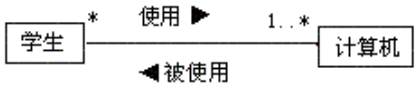
指可以通过对象中的属性形成对象间的一种相关依赖关系。

在OOM中用“实例连接”关系定义对象之间的静态关系。



2017/12/4 72

普通关联



73

限定关联

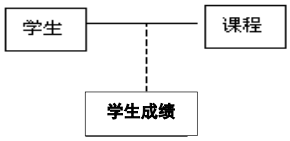
在一对多和多对多的关联关系中，可以用限定词将关联变成一对一的关联，限定词放在关联关系末端的一个小方框中。

74

4.4.1 面向对象方法中的基本概念

关联类

可以用关联类来说明关联的一些附加信息，关联类用一条虚线与关联连接。

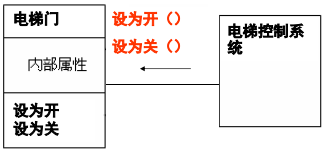


75

5.4.1 面向对象方法中的一些基本概念

对象行为间的动态关系（合作链接）

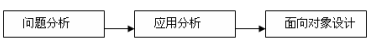
这种动态行为关系主要是由对象间的消息连接而形成的。一个对象可以通过消息向其它对象提出执行动作的要求，动作执行完后的对象通过消息可发送执行的结果等。



76

5.4.2 面向对象的需求分析

根据面向对象的过程模型，面向对象的需求分析从概念上被分为问题分析和应用分析两个方面。



77

5.4.2 面向对象的需求分析

- 问题分析
  - 问题分析的主要任务是收集并确认用户的需求信息，对实际问题进行功能分析和过程分析，从中抽象出问题中的基本概念、属性和操作，然后用泛化、组成和关联结构描述概念实体间的静态关系。最后，将概念实体标识为问题域中的对象类，以及定义对象类之间的静态结构关系和信息连接关系。最终建立关于对象的分析模型。

78

5.4.2 面向对象的需求分析

- 应用分析
  - 应用分析的主要任务是动态描述系统中对象的合法状态序列，并用动态模型表达对象的动态行为、对象之间的消息传递和协同工作的动态信息。
  - 虽然面向对象的概念是相同的，但由于分析工作的出发点、过程和模型的表达不同，故形成了不同的面向对象的分析方法。 如：OMT方法和OOAD方法。

79

5.4.2 面向对象的需求分析

OOAD分析方法的基本思想是：使用基本的结构化原则，结合面向对象的概念，构造一个描述系统功能的需求模型。

OMT分析方法的基本思想是：将面向对象的分析过程视为一个模型的构建过程，即整理获取的需求信息、并逐步分析和建立需求模型的过程。

80

5.4.3 OMT方法的图形描述工具

- OMT方法中的三种需求模型及其描述工具

对象模型

定义了系统的静态结构。主要用来描述类或对象之间的静态关系。

使用的描述工具：类图

动态模型

定义了系统的动态结构。主要用来表达系统动态交互行为中对象的状态受消息影响而发生变化的时序过程。

使用的描述工具：状态转换图和序列图

功能模型

定义系统应该做什么。主要用来描述系统从输入到输出的处理流程。

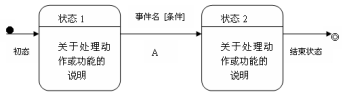
使用的描述工具：数据流图

81

5.4.3 OMT方法的图形描述工具

- 状态转换图

状态转换图（简称状态图）通过描述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（如处理数据）。



82

5.4.3 OMT方法的图形描述工具

一个状态图主要由状态、事件和状态变换组成，其中：

1. 状态：状态是任何可以被观察到的系统行为模式。
2. 事件：事件是在某个特定时刻发生的事情，它能引起系统做动作，并使系统从一个状态转换到另一个状态。
3. 状态转换：由某事件引起的两个状态之间的变化称为状态转换。

83

5.4.3 OMT方法的图形描述工具

画状态图的基本步骤：

1. 确定初态；
2. 确定事件（事件可由动作或输入信息等形成），并根据事件以及某些限制条件确定由当前状态转到下一个状态以形成一个状态转换；
3. 重复2的过程，直到最后确定结束状态为止。

84

5.4.3 OMT方法的图形描述工具

- 状态图示例

例1：一个人带着一头狼、一头羊以及一棵青菜，处于河的左岸。有一条小船，每次只能携带人和其余的三者之一。人和他的伴随品都希望渡到河的右岸，而每摆渡一次，人仅能带其中之一。然而，如果人留下狼和羊不论在左岸还是在右岸，狼肯定会吃掉羊。类似地，如果单独留下羊和菜，羊也肯定会吃掉菜。如何才能既渡过河而羊和菜又不被吃掉呢？

对象：

人（M）、狼（W）、羊（G）以及菜（C）。

状态：

用连字号“—”连接子集的对偶表示状态，例如MG-WC，其中连字号左边的符号表示处于左岸的子集；连字号右边的符号表示处于右岸的子集。

初始状态：MWGC-Φ

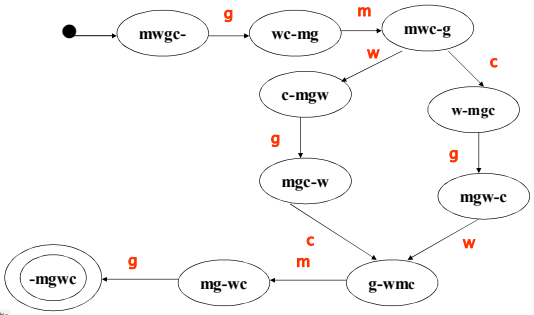
终止状态：Φ-MWGC。

事件：

由人所进行的活动作为系统的事件。

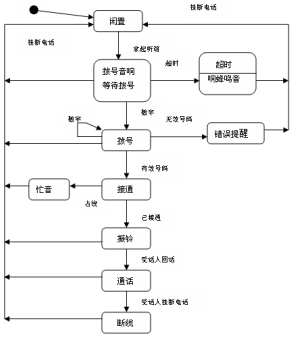
他可以单独过河（输入M）、带着狼过河（输入W）、带羊过河（输入G），或者带菜过河（输入C）。

5.4.3 OMT方法的图形描述工具



5.4.3 OMT方法的图形描述工具

例2：电话系统状态图



5.4.3 OMT方法的图形描述工具

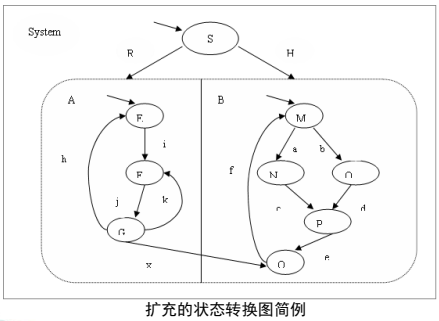
- 扩充的状态转换图

1. 扩充后的状态图仍继续沿用原来状态图符号。
2. 引入超状态（也称抽象状态）的概念，而超状态又可表示为由多个状态组成的状态图（或称子状态图）
3. 基于超状态概念，状态图可表示为层次式的，并且每个超状态可表示一个处理过程。超状态间的关系可用“与”和“或”关系给予表示。

5.4.3 OMT方法的图形描述工具

4. 一个上层状态图的事件可同时引起多个并行状态图（处理过程）工作。此外，并行的状态图之间可互相通过事件使对方发生状态转换。
5. 多个并行的状态图首先处于各自的初始状态，然后各状态图根据事件各自发生状态转换。这些状态图能通过某一事件同时到达上层状态图中的某一状态；也可以由某一子状态图通过某一事件到达上层状态图中的某一状态，从而强制结束并行执行的状态。
6. 每个子状态图都有各自的初始状态，而结束状态的有无可视具体情况设置。

5.4.3 OMT方法的图形描述工具

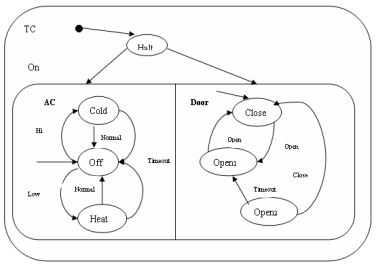


扩充的状态转换图简例

91

5.4.3 OMT方法的图形描述工具

- 扩充的状态转换图示例



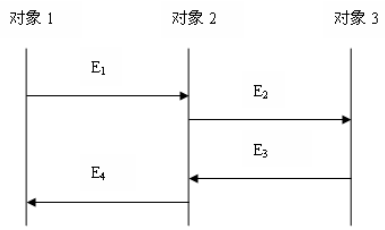
92

5.4.3 OMT方法的图形描述工具

- 序列图
  - 序列图主要用于表达对象与对象之间可能发生的所有事件，以及按事件发生时间的先后顺序列出所有事件的一种图形工具。
  - 序列图用一条竖直线表示一个对象或类，用一条水平的带箭头的直线表示一个事件，箭头方向是从发送事件的对象指向接受事件的对象。事件按产生的时间从上向下逐一列出。箭头之间的距离并不代表两个事件的时间差，带箭头的直线在垂直方向上的相对位置（从上到下）表示事件发生的先后顺序。

93

5.4.3 OMT方法的图形描述工具

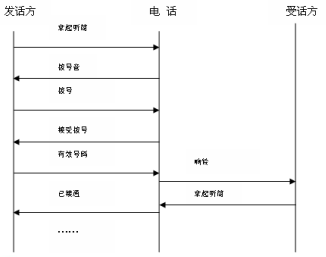


序列图的简例

94

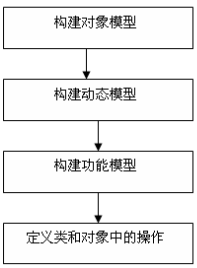
5.4.3 OMT方法的图形描述工具

- 示例：电话系统序列图



95

5.4.4 基于OMT方法的需求建模步骤



96

需求描述：

（1）某银行拟开发一个自动取款机系统，它是一个由自动取款机、中央计算机、分行计算机及柜员终端组成的网络系统。ATM和中央计算机由总行投资购买。总行拥有多台ATM，分别设在全市各主要街道上。分行负责提供分行计算机和柜员终端。柜员终端设在分行营业厅及分行下属的各个储蓄所内。该系统的软件开发成本由各个分行分摊。

（2）银行柜员使用柜员终端处理储户提交的储蓄事务。柜员负责把储户提交的存款或取款事务输进柜员终端，接收储户交来的现金或支票，或付给储户现金。柜员终端与相应的分行计算机通信，分行计算机具体处理针对某个账户的事务并且维护账户。

（3）储户可以用现金或支票开设新账户。储户也可以从自己的账户存款或取款。通常，一个储户可能拥有多个账户。拥有银行账户的储户有权申请领取银行卡。使用银行卡可以通过ATM访问自己的账户、提取现金（即取款），或查询有关自己账户的信息（例如，指定账户的余额）。

97

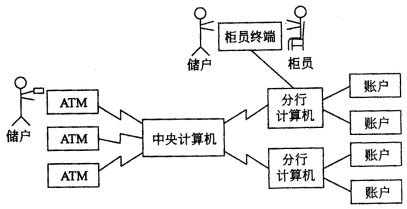
（4）银行卡是一张特制的磁卡，上面有分行代码和卡号。分行代码唯一标识总行下属的一个分行，卡号确定可以访问哪些账户。每张银行卡仅属于一个储户，但同一张卡可能有多个副本。因此，必须考虑同时在若干台ATM上使用同样的银行卡的可能性。也就是说，系统应该能够处理并发的访问。

（5）当用户把银行卡插入ATM之后，ATM就与用户交互，获取有关这次事务的信息，并与中央计算机交换关于事务的信息。首先，ATM要求用户输入密码，接下来ATM把读到的信息以及用户输入的密码传给中央计算机，请求中央计算机核对这些信息并处理这次事务。中央计算机根据卡上的分行代码确定这次事务与分行的对应关系，委托相应的分行计算机验证用户密码。如果用户输入的密码是正确的，ATM就要求用户选择事务类型（取款、查询等）。当用户选择取款时，ATM请求用户输入取款额。最后，ATM从现金出口吐出现金，打印出账单交给用户。

98

5.4.4 基于OMT方法的需求建模步骤

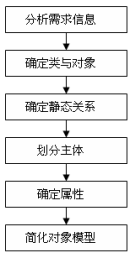
- 示例：ATM系统



99

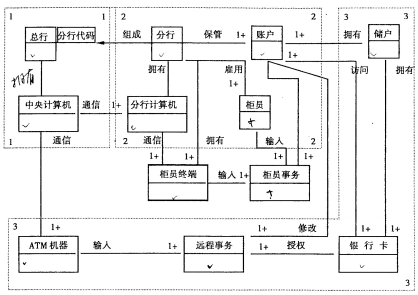
5.4.4 基于OMT方法的需求建模步骤

1. 建立对象模型



100

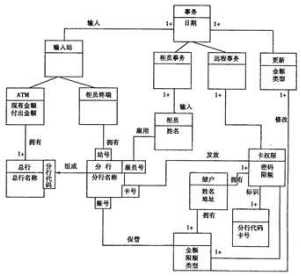
5.4.4 基于OMT方法的需求建模步骤



ATM系统初始类图

101

5.4.4 基于OMT方法的需求建模步骤

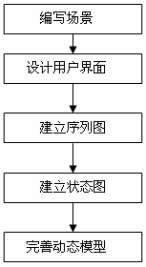


简化后的对象模型

102

5.4.4 基于OMT方法的需求建模步骤

2. 构建动态模型



103

5.4.4 基于OMT方法的需求建模步骤

1. 编写场景

- 场景的编写过程实质上也就是一个分析用户对系统交互行为的需求的过程，因此，在编写场景的过程中应与用户充分交流和讨论。
  - 首先编写正常情况的场景
  - 再考虑特殊情况
  - 最后考虑出错情况

104

5.4.4 基于OMT方法的需求建模步骤

·ATM请储户插卡；储户插入一张银行卡。
·ATM接受该卡并读它上面的分行代码和卡号。
·ATM要求储户输入密码；储户输入自己的密码，如“123456”等数字。
·ATM请求总行验证卡号和密码；总行要求“39”号分行核对储户密码，然后通知ATM说这张卡有效。
·ATM要求储户选择事务类型（取款、存款、转账、查询等）；储户选择“取款”。
·ATM要求储户输入取款额；储户户输入“1000”。
·ATM确认取款额在预先规定的限额内，然后要求总行处理这个事务；总行把请求转给分行，该分行成功地处理完这项事务并返回该账号的新余额。
·ATM吐出现金，请求储户取现金；储户拿起现金。
·ATM问储户是否继续本次事务；储户回答“不”。
·ATM打印账单，退出银行卡，请求储户取卡；储户取走账单和卡。

ATM系统的正常情况场景

105

5.4.4 基于OMT方法的需求建模步骤

·ATM请储户插卡；储户插入一张银行卡。
·ATM接受这张卡并顺序读它上面的数字。
·ATM要求密码；储户误输入“88888”
·ATM请求总行验证输入数字和密码；总行在向有关分行询问之后拒绝这张卡。
·ATM显示“密码错”，并请储户重新输入密码；储户输入“123456”；ATM请总行验证后知道这次输入的密码正确。
·ATM请储户选择事务类型；储户选取“取款”。
·ATM询问取款额；储户改变主意不想取款了，他敲“取消”键。
·ATM退出银行卡，请求储户取卡；储户拿走他的卡。
·ATM请储户插卡。

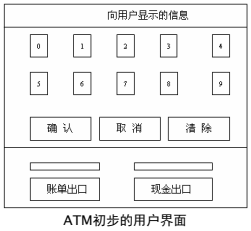
ATM系统的异常情况场景

106

5.4.4 基于OMT方法的需求建模步骤

2.设计用户界面

- 在设计用户界面时，用户界面的细节并不太重要，重要的是在这种界面下的信息交换方式。



ATM初步的用户界面

107

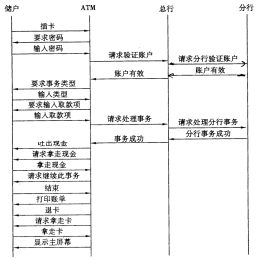
5.4.4 基于OMT方法的需求建模步骤

3. 建立序列图

- 首先应认真分析每个场景的内容
  - 提取所有外部事件信息及异常事件和出错条件的信息
  - 传递信息的对象的动作也可作为事件
  - 事件形成对象与对象之间的交互行为
- 确定了每类事件的发送对象和按受对象之后，就可以利用序列图将事件序列以及事件与对象间的关系清晰和形象地表示出来
- 每个场景对应一张序列图

108

5.4.4 基于OMT方法的需求建模步骤



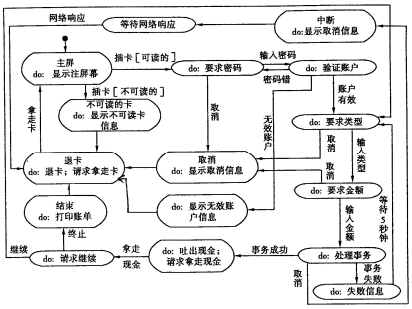
ATM系统正常情况场景的序列图

5.4.4 基于OMT方法的需求建模步骤

4.构建动态模型

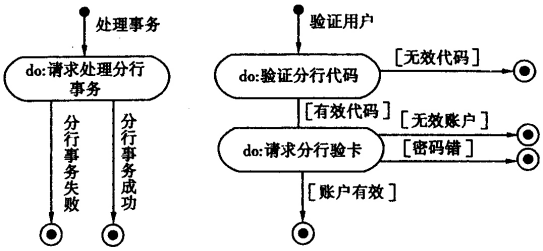
- 状态图适用于表示动态模型，其刻画了事件与对象状态之间的关系。
- 依据序列图构造状态图的过程如下：
  - 分析序列图
  - 正常事件描述之后，还应考虑边界情况下可能发生的事件
  - 根据一张序列图画出对象或类状态图之后，再把其他与该对象或类相关的场景（如异常情况场景）的序列图加入到已画出的状态图中

5.4.4 基于OMT方法的需求建模步骤



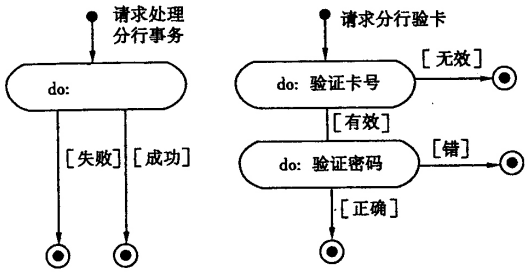
ATM的状态图

5.4.4 基于OMT方法的需求建模步骤



总行类的状态图

5.4.4 基于OMT方法的需求建模步骤



分行类的状态图

5.4.4 基于OMT方法的需求建模步骤

3. 构建功能模型

- 功能模型主要表达系统内部数据流的传递和处理的过程
- 数据流图适用于描述系统的功能模型。

5.4.4 基于OMT方法的需求建模步骤

4. 定义类和对象中的操作

- 定义类和对象中操作的原则：
  - 基本的属性操作
  - 事件的处理操作
  - 完成数据流图中处理框对应的操作
  - 利用继承机制优化服务集合，减少冗余服务

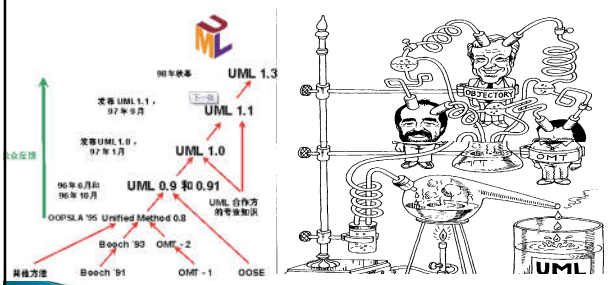
5.5 基于图形的需求建模技术

- 优点
  1. 图形具有直观性、简单性以及可理解性等优点；
  2. 图形能自然地表达客观世界；
  3. 在实体之间，满足传递关系的情况有很多，这可以理解成图中路径探索，并可以研究路径探索的有效算法。
- 存在的问题
  1. 图形的语义往往有时是含糊的；
  2. 在图中不能表示数据定义；
  3. 图形中表示符号的种类有限。

5.5.1 UML概述

- UML（Unified Modeling Language）是综合面向对象分析/设计方法中使用的各种图形描述技术，并试图给出这些图形描述的语法和语义的语言
- 可分为表示对象静态结构和动态结构
- 静态结构：
  - 用例图、类图、构件图等
- 动态结构类：
  - 状态图、活动图、序列图、协作图和配置图等

UML发展历史



UML的准确理解

- Language
  - 只是一种表示方法，本身不包含任何方法论的部分
- Modeling
  - 不是用于编程、而是用于建模；不仅包含软件建模功能，还包括业务建模、流程建模等多种应用领域
- Unified
  - 一方面是OMG组织认可的工业标准；同时也是得到IBM、SUN、Borland等众多大型公司支持的事实标准

为什么要用UML

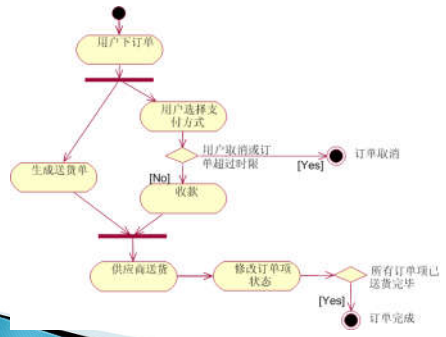
- UML作为一种统一、标准化的建模语言，能为参与软件设计和开发的人提供一种公共的语言，使他们能够基于共同的模型来理解业务、需求，理解软件和架构如何构造
- 作为应用面广泛的建模语言，不仅可用于软件系统建模，还可以用于业务流程、业务知识、数据库、嵌入式等多个领域，让不同的人们可以基于相同的语言沟通；不同领域模型可以通过相同的机制进行互换和迁移。

需求阶段所需要使用的UML图

使用频率	图名	功能	关注要点
主体	活动图、交互图	说明业务流程，以及业务活动的步骤	事
	类图	说明业务实体之间的关系，体现结构规则	物
	用例图	说明角色和使用场景之间的关系	人
辅助	构件图	说明主题域划分以及它们之间的服务接口	接口
	部署图	描述系统的部署环境，体现设计约束	设计约束

12
1

简单活动图



活动图的主要元素

- 初始节点和活动终点：用一个实心圆表示初始节点，用一个圆圈内加一个实心圆来表示活动终点
- 活动节点：是活动图中最主要的元素之一，它用来表示一个活动

用户下订单 — 文字描述

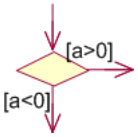
len=a.length+1 — 表达式

dispatch(aOrder) — 消息

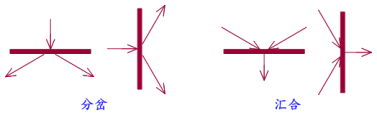
- 转换：当一个活动结束时，控制流就会马上传递给下一个活动节点，在活动图中称之为“转换”，用一条带箭头的直线来表示

活动图的主要元素

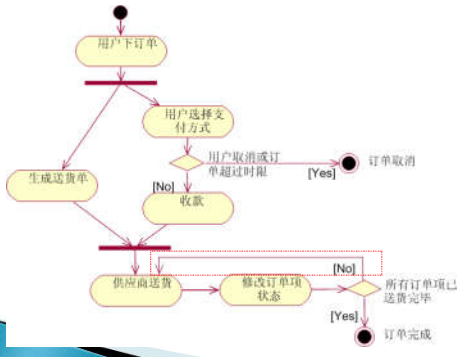
- 分支与监护条件：分支是用菱形表示的，它有一个进入转换（箭头从外指向分支符号），一个或多个离开转换（箭头从分支符号指向外）。而每个离开转换上都会有一个监护条件，用来表示满足什么条件的时候执行该转换。



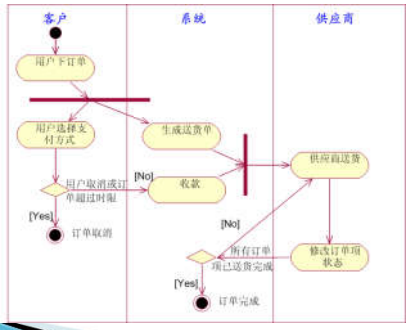
- 分岔与汇合：



修改后的简单活动图



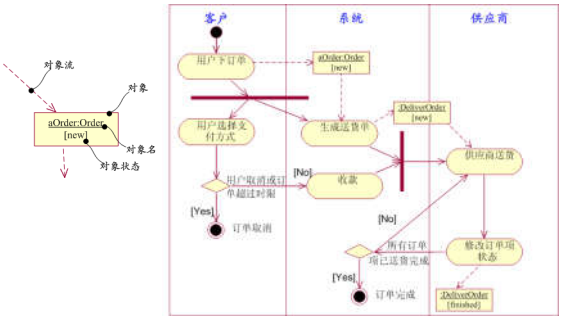
带泳道的活动图



带对象流的活动图

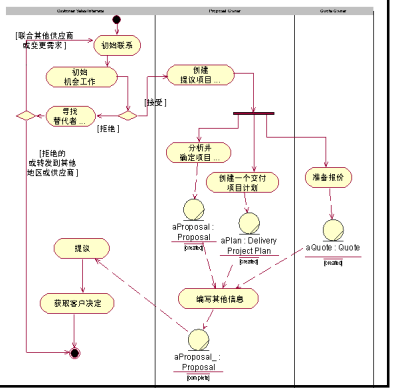
- 使用对象流
  - 对象流用于显示如何在工作流程中创建并使用业务实体。
  - 对象流符号不仅仅表示对象本身的存在，而且还表明它所处的特定状态。同一个对象可被大量的、改变该对象状态的连续活动所控制。此后，该对象就可在活动图中多次出现，而且每次出现时都表示其生命期中的不同状态。该对象在每一点所处的状态都可置于括号内，并附加到其类名称之后。

带对象流的活动图



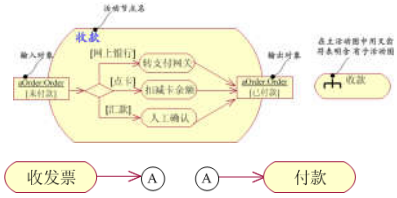
带对象流的活动图

- 一个对象流状态可能表现为一个对象流（转移）的目标和多个对象流（转移）的来源。



复杂活动图

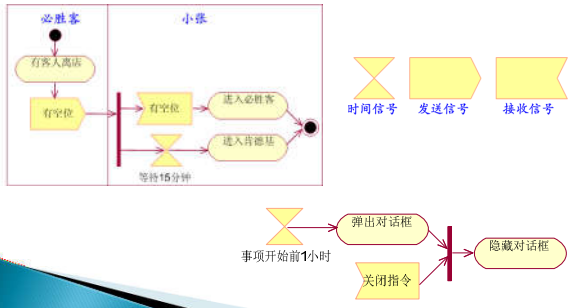
- 辅助活动图：



- 汇合描述：当汇合的所有入流均到汇合点时，就将执行汇合点指向的活动节点。但是有些时候，你希望对其做一些约束，这时就可以借助汇合描述来完成。汇合描述实际上是一个约束，其格式就是“{约束条件}”。

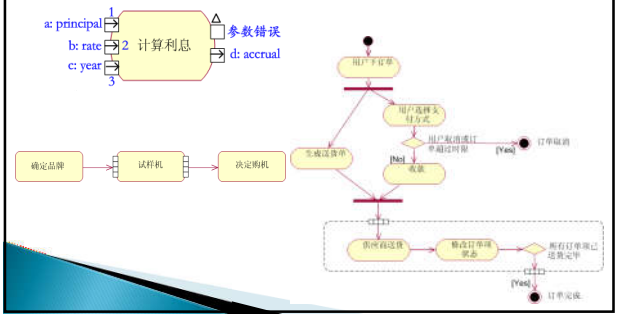
复杂活动图

- 发送信号与接收信号：



复杂活动图

- 引脚：表示活动节点的相应参数



练习

- 某教学系统操作员登录过程是：启动该系统，系统给出登录窗口，在登录窗口中需要输入用户名和密码，如果用户名或密码有误，则系统提示错误，操作员重新输入，若连续3次用户名或密码均没有输入正确，则系统拒绝登录。如果输入正确，则进入系统。用活动图描述操作员的登录过程。
- 请根据上述需求描述绘制带泳道的活动图

用例图

识别参与者

- 已有的上下文关系图（表示系统范围）及其他相关模型：它们描述了系统与外部系统的边界，从这些图中可以寻找出与系统有交互关系的外部实体。
- 项目相关人员分析：对项目的相关人员进行分析，就能够决定出哪些人将会与系统进行交互。
- 书面的规格说明和其他项目文档（如会谈备忘录等）
- 需求研讨会和联合应用开发会议的记录：这些会议的参与者通常是很重要的，因为他们在组织中所代表的角色就是可能与系统发生交互的参与者。
- 当前过程和系统的培训指南及用户手册：这些东西中经常会有潜在参与者。

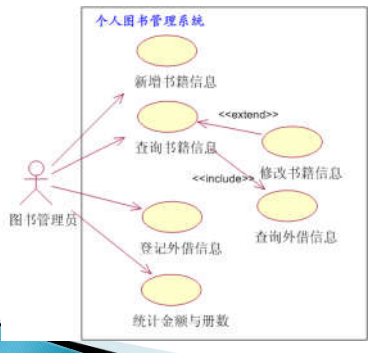
记录需求—特性表

编号	说明
FEAT01	新增书籍信息
FEAT02	修改已有的书籍信息
FEAT03	书籍信息按计算机类、非计算机类分别建档
FEAT04	录入新书时能够自动按规则生成书号
FEAT05	计算机类与非计算机类书籍采用不同的书号规则
FEAT06	录入新书时如果重名将自动提示
FEAT07	按书名、作者、类别、出版社等关键字组合查询书籍
FEAT08	列出所有书籍信息
FEAT09	记录外借情况
FEAT10	外借状态能够自动反应在书籍信息中
FEAT11	按人、按书查询外借情况
FEAT12	列出所有的外借情况
FEAT13	按特定时间段统计购买金额、册数
FEAT14	所有查询、列表、统计功能应可以单独对计算机类或非计算机类进行

合并需求获得用例

特性	用例
FEAT01.新增书籍信息 FEAT03.书籍信息按计算机类、非计算机类分别建档 FEAT04.录入新书时能够自动按规则生成书号 FEAT05.计算机类与非计算机类书籍采用不同的书号规则 FEAT06.录入新书时如果重名将自动提示	UC01.新增书籍信息
FEAT02.修改已有的书籍信息	UC02.修改书籍信息
FEAT07.按书名、作者、类别、出版社等关键字组合查询书籍 FEAT08.列出所有书籍信息 FEAT14.所有查询、列表、统计功能应可以单独对计算机类或非计算机类进行	UC03.查询书籍信息
FEAT09.记录外借情况 FEAT10.外借状态能够自动反应在书籍信息中	UC04.登记外借信息
FEAT11.按人、按书查询外借情况 FEAT12.列出所有的外借情况 FEAT14.所有查询、列表、统计功能应可以单独对计算机类或非计算机类进行	UC05.查询外借信息
FEAT13.按特定时间段统计购买金额、册数 FEAT14.所有查询、列表、统计功能应可以单独对计算机类或非计算机类进行	UC06.统计金额和册数

绘制用例图



细化用例描述—搭框架

- 1.用例名称：新增书籍信息（UC01）
- 2.简要说明：录入新购书籍信息，并自动存储建档。
- 3.事件流：
- 3.1基本事件流
- 3.2扩展事件流
- 3.3非功能需求
- 3.4前置条件：用户进入图书管理系统。
- 3.5后置条件：完成新书信息的存储建档。
- 3.6扩展点：无
- 3.7优先级：最高（满意度 5，不满意度5）

细化用例描述—填内容

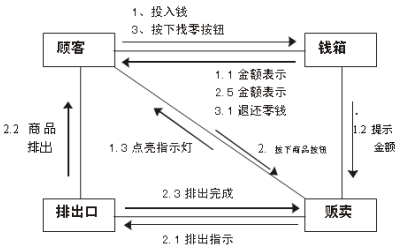
- ……
- 3.事件流：
- 3.1 基本事件流
- 1）图书管理员向系统发出“新增书籍信息”请求；
- 2）系统要求图书管理员选择要新增的书籍是计算机类还是非计算机类；
- 3）图书管理员做出选择后，显示相应界面，让图书管理员输入信息，并自动根据书号规则生成书号；
- 4）图书管理员输入书籍的相关信息，包括：书名、作者、出版社、ISBN号、开本、页数、定价、是否有CDROM；
- 5）系统确认输入的信息中书名未有重名；
- 6）系统将所输入的信息存储建档。
- 3.2 扩展事件流
- 5a）如果输入的书名有重名现象，则显示出重名的书籍，并要求图书管理选择修改书名或取消输入；
- 5a1）图书管理员选择取消输入，则结束用例，不做存储建档工作；
- 5a2）图书管理员选择修改书名后，转到5）
- 4.非功能需求：无特殊要求

5.5.3 协作图（通信图）

- 协作图用于表示对象间消息往来。虽然序列图在某种定义上也能表示对象的协作动作，但能明确描述对象间的协作关系的还是协作图。
- 协作图便于描述对象间有什么样的协作关系，其不需要像一个序列图只能对应于一个场景一样，可以将多个场景中的协作关系一次性地全部描述出来。

5.5.3 协作图（通信图）

- 示例：自动贩卖机的协作图



协作图与时序图

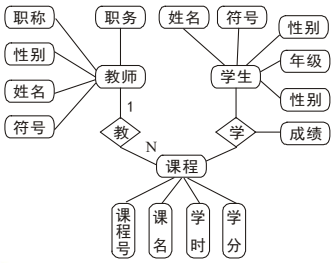
- 共同点
  - 描述对象之间的交互
  - 时序图、协作图
- 区别
  - 时序图：强调顺序，能够反映消息的先后顺序
  - 协作图：强调连接，能够反映消息的上下文

5.5.4 数据库建模 实体关联图

- 实体关联图亦称ER图（Entity-relationship diagram）或称实体联系图，主要用于描述系统的数据关系。
- 组成
  - 实体、实体间的关联和属性。
- 实体间的关联
  - 一对一关联、一对多关联和多对多关联。

5.5.4 实体关联图

- 示例：某校学生管理系统的ER图



14
5

使用UML对数据库建模

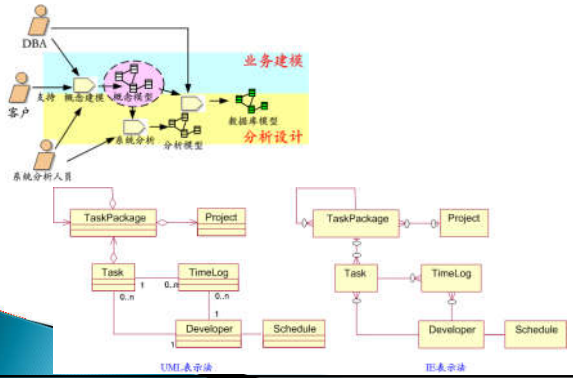
数据库模型与类模型

项目	数据库建模	类建模
核心工作	识别数据实体(data entity)	识别类(在领域建模时主要是实体类)
数据	将数据属性分配给数据实体	将属性和操作分配给类
关联	多重性、继承、组合、聚合	多重性、继承、组合、聚合
目标	关注于数据	关注于对象

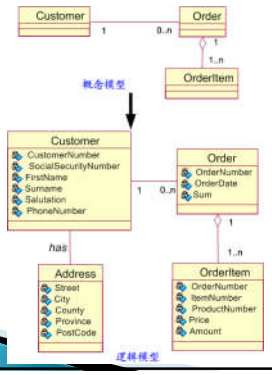
数据模型的三种形式

- 概念模型：典型的概念模型只对领域中的实体和主要领域实体之间的关系建模，与技术细节无关
- 逻辑数据模型（LDM）：LDM不仅揭示了数据实体、数据实体之间的关系，还描述了这些数据实体的数据属性；不过它仍然是与设计细节无关的
- 物理数据模型（PDM）：用来表现数据库的内部schema设计的，它揭示了数据表、表中的数据列，以及数据表之间的关系

概念模型建模



逻辑模型建模

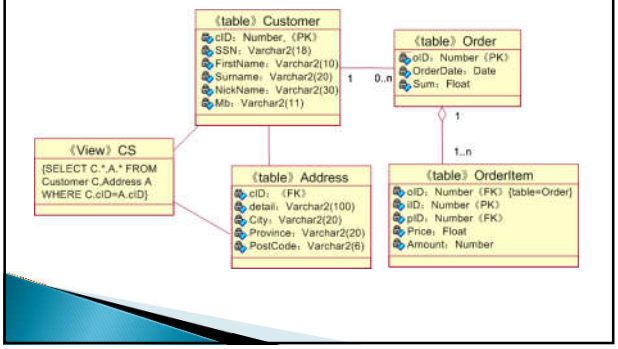


物理模型

- 对数据库表、视图等实体建模：数据库表《Table》、视图《View》、索引《Index》、存储过程《Stored Procedures》、触发器《Trigger》
- 对数据列建模

键类别	构造型	说明
备选键	《AK》	标识该列为备选键，也称为次要键
自生成	《Auto Generated》	该列的值由数据库生成
候选键	《CK》	该列是实体的候选键的一部分
外键	《FK》	该列为数据库表外键的一部分
自然键	《Natural》	该列为数据库自然键的一部分
主键	《PK》	该列为数据库主键的一部分，主键决定唯一列
代理键	《Surrogate》	该列为数据库表中一个代理键

物理模型实例


《table》Customer
cID：Number，《PK》
SSN：Varchar2(18)
FirstName：Varchar2(10)
Surname：Varchar2(20)
NickName：Varchar2(30)
Mb：Varchar2(11)
1
0..n
《table》Order
oID：Number《PK》
OrderDate：Date
Sum：Float
1
1..n
《View》CS
{SELECT C.*,A.* FROM Customer C,Address A WHERE C.cID=A.cID}
《table》Address
cID：《FK》
detail：Varchar2(100)
City：Varchar2(20)
Province：Varchar2(20)
PostCode：Varchar2(6)
《table》OrderItem
oID：Number《FK》{table=Order}
iID：Number《PK》
pID：Number《FK》
Price：Float
Amount：Number