面向对象的MIS开发方法DDD*

诸葛海

(中国科学院软件研究所,北京 100080)

摘要 本文重新定义了面向对象的基本概念,提出了有向目录图DDD,为描述对象及其组成、继承、服务和状态转移等关系提供了统一简洁的符号工具. DDD与一组概念、原则、策略、辅助工具以及对象管理系统相结合,构成一种面向对象的MIS开发方法. 该方法已在财政行业MIS群的开发中得到了应用.

关键词 软件工程,软件工程环境,面向对象,管理信息系统.

面向对象(Object-Oriented,简称OO)的概念和技术正在软件工程领域中得到广泛承认和应用,OO方法学(简称OOM)也逐渐成为软件工程方法学的主流. 然而,OOM尚缺乏完整的理论基础、实用的开发规范和相应软件环境的支持. 为此,作者重新定义了OO的基本概念,并在开发MIS的实践中提出了一套简洁实用的OO MIS开发方法,方法的核心是以直观的有向目录图DDD(Directed Directory Diagram)为基础的系统分析与设计方法和支持系统开发全过程的对象管理系统OMS(Object Management System).

1 OO基本概念的定义

基本假设:集合论与抽象代数的基本概念(如集合、关系、映射等)、基本公理和命题在本系统内都成立. 本文试图用最少的概念来定义OO概念体系. 作者认为,对象和抽象是OO的两个核心概念,下面给出两条基本定义.

定义 1.1. 对象是可以用唯一的标识加以区分的实体或概念.

(1)对象用一个或多个属性来描述其内在特征.

〈属性〉::=(〈属性标识〉,〈属性类型〉,〈属性值〉)

(2)任何对象都为其它对象提供一个或多个服务,也可引用一个或多个其它对象的一个或多个服务. 引用对象服务是对象之间通信的唯一方式.

〈提供服务〉::=s(〈服务标识〉,(〈参数部分〉))

〈引用服务〉::=c(〈服务标识〉,(〈参数部分〉))

〈参数部分〉::=((〈参数表〉), R(〈参数表〉)) | (〈参数表〉) 〈空〉

* 本文1993-05-27收到,1993-11-13定稿

本研究受到国家八五科技攻关项目和国家自然科学基金的资助. 作者诸葛海,1963年生,副研究员,主要研究领域为软件工程方法与环境,人工智能理论与应用.

本文通讯联系人:诸葛海,北京 100080,中国科学院软件研究所

〈参数表〉::=〈参数标识,参数类型〉|〈参数表〉,〈参数标识,参数类型〉

其中,R〈参数表〉表示引用服务所要求返回的参数表.

(3)对象的属性值只能通过对其服务的引用来改变.

(4)对象的部分(也可全部)属性和服务称为对象视图.

定义 1.2. 抽象是对象集上的一种关系,也是一种划分方法.将所有已标识的对象按一定的方法(比如,按具有相同属性和提供相同服务来分)分为对象类,给每一类对象以新的标识,并赋以类中对象所具有的那些共同属性和服务,形成新对象,称为该对象类的型(也称类中某对象的型),类中的每个对象称为该型的实例.型的形成过程是一种抽象.对型集进行上述抽象形成型的型(父型).由此下去,对象集在抽象关系下构成一个由型、父型组成的层次结构.

运用以上基本定义可以继续定义一系列概念.

定义 1.3. 子型是对象型集上的关系,设A,B为两对象类的型,如果所有B的实例也可作为A的实例,则称B为A的子型,A为B的父型.

定义 1.4. 继承是对象型集上的关系,具有这种关系的两个对象型其中一个是另一个的父型,它们存在一个相同的视图.利用继承关系可以用现有的型定义新的型.

子型继承父型的属性和服务,父型是子型的抽象.

2 符号工具的定义

一套良好的符号工具是系统分析与设计的基础.结构化系统分析与设计(简称SA/SD)及其变形方法有一定局限性,如(1)图形复杂,不便绘制和生成;(2)文档庞大,导致长的开发周期;(3)分析和设计阶段分别采用两套不同的符号体系,使得分析的结果不便向设计阶段转化;(4)基于功能的方法难以适应领域发生的变化.为了克服这些不足,并与OO概念相衔接,本文提出了有向目录图DDD,作为系统分析和设计的统一符号工具.

DDD的形式定义如下:

〈DDD〉::= △〈对象标识〉〈对象描述〉|△〈对象标识〉

〈对象描述〉::=〈对象描述图〉〈对象描述〉|〈空〉

〈对象描述图〉::= 〈m1〉
　　|〈m2〉
　　├〈m3〉〈DDD〉
　　〈m4〉
　　|〈m2〉
　　├〈m3〉〈DDD〉
　　〈m4〉
　　……
　　|〈m2〉
　　└〈m3〉〈DDD〉

其中:〈m1〉::=||↑　　{↑表示继承关系,|为图的连线}

〈m2〉::= s(服务名(〈参数〉))　{s(服务名(〈参数部分〉))表示提供服务}

| c(服务名(〈参数〉))　{c(服务名(〈参数部分〉))表示引用服务}

| i(〈视图〉)　　{i(〈视图〉)表示继承箭头所指对象的一个视图}

|〈空〉|m2〈标记〉

〈标记〉::= * {* 表示多次重复}

| o {o 表示可选}

| h {h 表示所标记的对象被隐蔽}

| # {# 表示关键对象}

〈m3〉::= ←→ {1:1 关系}

|←→→ {1:m 关系}

|←←→ {m:1 关系}

|←←→→ {m:m 关系}

|— { —为图的连线}

〈m4〉::= ↓ | | {↓ 表示顺序关系或层次结构，| 为图的连线}

加“△”的对象表示已定义的对象，未加“△”的对象表示可以继续用 DDD 逐层定义下去.“{}”中的内容表示注释，在文档的任意部分均可以插入用“{}”括起来的注释. DDD 的定义允许递归. 根据〈mi〉(i=1,2,3,4)的不同取值，DDD 可表达出五种不同形式的关系图.

(1)组成关系图

组成关系图表示一个对象由若干成员对象组成，即表示整体和部分的关系. 在 DDD 中，令：〈m1〉=|，〈m2〉=〈空〉|*|o|h|#，则，DDD 派生为组成关系图. 如图 1 所示. 如果〈m4〉取“↓”，则表示上行的成员对象发生于下行的成员对象之前.

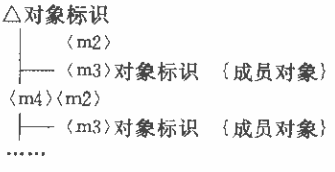


图1 组成关系图

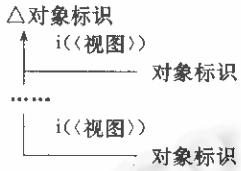


图2 继承关系图示例

(2)继承关系图

继承关系表示一个或多个对象分别继承某个对象的视图. 在继承中引入视图的概念是对以往继承概念的扩展. 在 DDD 中，令：〈m1〉=↑，〈m2〉=i(〈视图〉)，〈m3〉=—，〈m4〉=|，则 DDD 派生为继承关系图，如图 2 所示. 其中，〈视图〉为“↑”所指对象的视图. 表示每个对象可以继承“△对象”的一个视图(各视图可以不同). 在“△对象”中已描述过的内容在其后代(发出箭头的对象)中不必描述，但要指出所继承的视图. 利用继承图可以避免冗余定义，提高可重用性，简化设计.

一个对象同时继承两个以上对象的属性和服务称为多重继承.

原则：为了消除继承冲突以及保持概念和实现上的简单性原则，避免多重继承.

(3)服务图

服务图描述指定对象所提供的服务和引用其它对象的服务. 在 DDD 中，令：〈m1〉=|，〈m2〉=s(服务名(〈参数〉)) | c(服务名(〈参数〉))，〈m3〉=—，〈m4〉=| | ↓，当〈m4〉为↓时，表示相关的服务是有序的，则：DDD 派生为如图 3 所示的服务图.

(4)状态转移关系图

有些对象适合于用一系列有序状态的转移来描述其内容和演变,状态转移关系图就是用来描述这种关系的.在DDD中,令:〈m1〉=|,〈m2〉=c(服务标识(〈参数〉)),〈m3〉=→,〈m4〉=|,则DDD派生为状态转移关系图,如图4所示.该图表示"△对象"引用服务后转到"→"所指的状态.

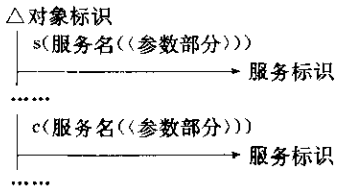


图3 服务图

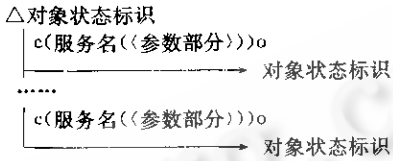


图4 状态转移图

(5)对象属性关系图

对象属性关系图表示指定对象的所有属性型之间的关系.在DDD中,令:〈m1〉=|,〈m2〉=#|〈空〉,〈m3〉=—|←→|←→→,〈m4〉=|,则DDD派生为对象属性关系图,如图5所示.

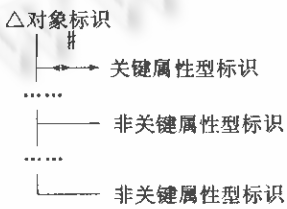


图5 对象属性关系

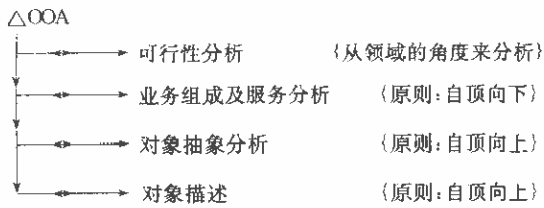


图6 OOA

由于DDD允许嵌套,所以可以根据需要将以上五种(或其中几种)组合描述在同一个DDD中.

3 面向对象的分析方法(OOA)

系统分析的目的在于建立领域的对象模型,其结果应满足以下原则:(1)可以模拟领域;(2)便于交流;(3)图形化;(4)简洁性.

3.1 OOA的内容和步骤

OOA的内容及进行的步骤如图6所示.

工作原则:(1)自顶向下,逐步将领域业务分解为独立的领域业务处理对象,(2)自底向上进行抽象.

第一步:可行性分析(见图7)

第二步:业务组成及服务分析

策略:依据领域现行的管理体系自顶向下按业务进行分解,直到不能再分的简单(业务)对象为止.对于简单对象描述其服务.例如,图8简示了财政业务的组成和服务.

第三步:抽象

抽象是针对一目标在一定抽象级上形成对象型的过程.具体分为两步:对象抽象和

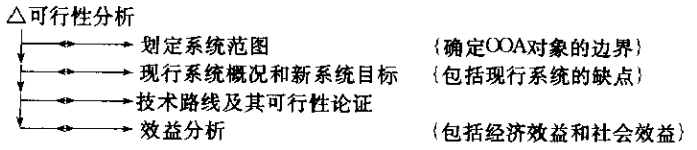


图7 可行性分析

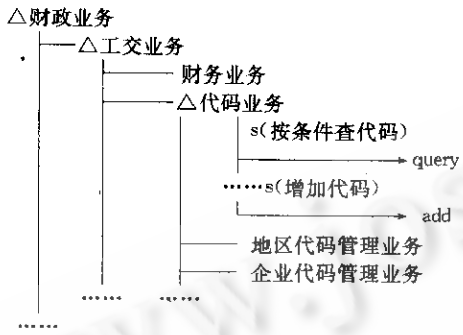


图8 业务组成及服务分析

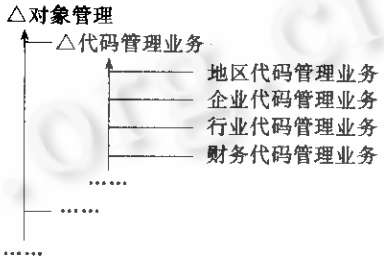


图9 根据服务的型来抽象

型抽象.

(1)对象抽象:(a)将对象的属性抽象为属性型;(b)将对象的服务抽象为服务型,例如,将"按地区查代码"和"按行业查代码"等服务抽象为"根据条件查代码"服务型.

(2)型抽象:设 O_i 和 O_j 为两个对象,O_i 提供服务 $S_{i1},\cdots,S_{in}$,O_j 提供服务 $S_{j1},\cdots,S_{jm}$,S_{ik} 和 $S_{jk'}$ 分别为 O_i 和 O_j 的第 k 个和第 k' 个服务,$T(S_{ik})$,$T(S_{jk})$ 分别为相应的型,$T(S_{ik})=T(S_{jk'})$,$1\leqslant k,k'\leqslant\min(n,m)$.

策略:(1)如果 $m=n$,则形成 O_{ij},其服务与 O_i 及 O_j 的服务相同,否则进行(2);

(2)抽取:形成新的 O_{ij},O_{ij} 的属性和服务为两者的交集,并将 O_i 和 O_j 作为 O_{ij} 的子型;

例如,根据所提供的服务类型相同,将"财务指标管理"从财务业务中分离出来和各类型代码业务抽象为"代码管理业务".最后将"代码管理业务"等抽象为"对象管理"的子型.在进行抽象的同时建立如图 9 所示的继承图.

第四步:合并优化

不受领域现行管理体系的限制,根据服务的相关性来组织对象.

原则:(1)尽可能减少引用服务联结的复杂性,参数尽可能少;(2)尽可能运用继承关系;(3)不含无用属性和服务.

第五步:对象描述

在继承关系中描述每个对象自身特有的属性和服务,通过继承关系可得到的属性和服务不必重新描述,但要指出被继承对象的标识和视图,从而提高软件设计的可重用性.

3.2 对象字典及其辅助工具

对象字典(Object Dictionary,简称:OD) 存放和管理对象及关系,定义如下:

〈对象字典〉::=(〈对象表〉,〈服务表〉,〈属性表〉,〈参数表〉)

〈对象表〉::=(〈对象标识〉,〈对象名〉,〈对象型〉,〈服务链〉,〈组成链〉,〈属性链〉,〈状态

转移链〉)

〈服务表〉::=(〈服务标识〉,〈服务名〉,〈服务型〉,〈参数链〉)

〈属性表〉::=(〈属性名〉,〈属性型〉,〈值型〉,〈型长〉,〈初值〉,〈缺省值〉)

〈参数表〉::=(〈参数名〉,〈参数型〉)

〈值型〉::=〈整型〉|〈实型〉|〈字符型〉|……

……

对于给定的领域,建立、维护和确认OD的过程实际上就是OOA的过程,OD应保持一致性和无冗余性.

OD辅助工具的主要功能是提供建立OD的方便界面,进行一致性检查,消除冗余,辅助分析,生成5种形式的DDD.由于DDD是结构化的,消除了线条的交叉.所以给图的生成带来很多方便.同时也增强了图的可读性,简化了文档.

4 面向对象的设计OOD

正如SA/SD方法学有结构化程序设计方法和语言的支持一样,DDD方法的OOD不仅应与其OOA相衔接,而且更需要一种实现手段和实现方法学的支持.

4.1 OOD的内容和步骤

OOD根据OOA的结果设计出相应的软件,文献[1]论述了从领域结构向软件结构映射的一种方法.OOD的内容和步骤用图10来描述.

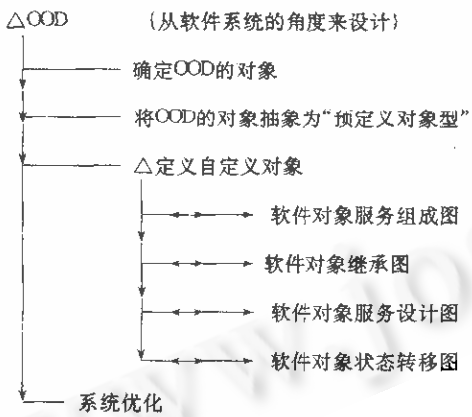


图10 OOD的内容和步骤

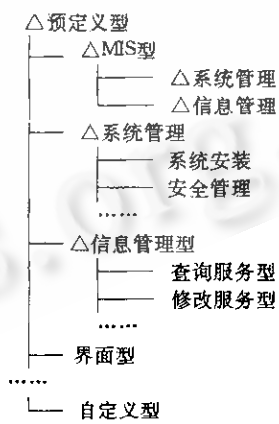


图11 预定义型

4.2 确定OOD的对象

设DO为领域对象类,DS为领域服务类,SO为系统对象类,SS为系统服务类,确定OOD对象分f1,f2两个映射来完成:f1:DO→SO,f2:DS→SS.

在映射过程中要明确哪些对象及其服务是软件系统来完成的,哪些是软件系统不能完成要由人工或其它系统来完成的.那些由软件系统来完成的对象就是OOD的对象.从这一步起,思维的角度要从"领域"转向"软件系统".

4.3 设计中的抽象

抽象是人类的智能行为，它始终贯穿于 OOA/D 之中，抽象能力与人的知识和机制有关，是因人而异的，所以不同的设计人员设计出的软件系统存在差异. 文献[2]有关于抽象的详细论述. 一种 MIS 开发方法应包含策略和工具以帮助进行抽象.

(1)OOD 的抽象目标

在 DDD 方法中定义了包括抽象数据类型在内的“预定义型”，它给出对象抽象的目标以辅助设计者进行抽象. 图 11 是一部分预定义型.

(2)抽象策略

OOA 中的抽象方法和 OOD 的抽象方法是抽象方法的两个实例，换言之，OOA 的抽象方法完全可以在 OOD 中得到重新使用.

4.4 软件服务组成

OOA 的组成及服务图经抽象后将其映射为预定义型和自定义型两部分，例如，CZMIS 的软件服务组成用图 12 描述.

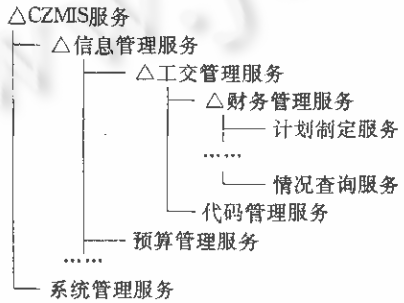


图12 CZMIS的服务组成图

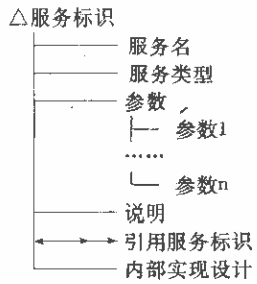


图13 服务描述

策略：(1)先描述 MIS 型，描述提供和引用的服务；

(2)选择 OOA 的组成及服务图中未考察过的对象；如果领域服务可抽象为预定义型，则运用该预定义型，否则为自定义型，给出对象及服务的设计；

(3)如果 f1(DO)和 f2(DS)中的每个对象都已在软件服务组成图中描述过则结束，否则继续进行(2).

4.5 软件对象继承图

软件对象的继承包括两层含义，一是通过 OOD 对象反映出的 OOA 所描述的领域对象和服务继承关系，另一是 OOD 对象本身为了提高软件的可重用性而形成的继承关系. 二者有一部分是一致的，但也有一部分不同. 比如，有时为了提高效率而要增加冗余度.

软件对象继承图按照对象型、子型和实例的继承关系分类描述. 例如，“代码管理”继承“对象管理”的服务，“企业代码管理”又继承“代码管理”的服务，要查找某企业的代码只要引用对象管理的条件查询服务即可.

4.6 软件服务设计

对于自定义型的对象给出其服务的设计，如图 13 所示.

4.7 用户界面设计

用户界面也是一类对象，其设计大多是针对特定窗口的，界面的状态通过对其提供的服务的引用而演变. 窗口中的主要成份是菜单 M(menu)和表格 F(form)，表格包括：屏幕表

格、屏幕数组和报表.用户界面UI设计定义如下:

〈UI〉::=M(〈对象名〉:〈服务表〉,W(〈x〉,〈y〉,〈Attr〉))
　　| M(〈对象名〉:〈服务表〉) | F(〈FormList〉, W(〈x〉,〈y〉,〈Attr〉))
　　| 〈Message〉|〈OS〉|〈UI〉|〈空〉

〈服务表〉::={〈服务标识〉(〈服务名〉,〈说明信息〉)}

〈FormList〉::={(〈Item〉,〈Type〉,〈Length〉),} | (〈FormList〉,〈Num〉)

〈Message〉::=〈注释〉|〈出错信息〉|〈空〉

……

下面给出一个界面状态转移图的例子:

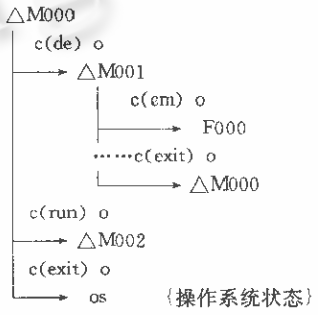
M000:(MIS开发环境:De—开发维护,Run—运行,Exit—退出,W(4,60,attr))

M001:(开发维护:Cm—对象管理,Pr—生成,Exit—退出)

……

F000:(对象管理服务:Add—增加,Del—删除,Ch—修改,Sel—查询,Exit—退出)

……



构成界面的基本构件都作为预定义型,所以在设计界面时,只要定义所需型的实例和自定义型即可.界面定义完备后可由辅助工具生成这些实例.

5　对象管理系统OMS

支持DDD方法的理想实现手段是一个标准化的OO平台OMS.

(1)OMS的组成

OMS是所有关于对象信息的集中存贮和管理.OMS用统一的方式管理所有对象,包括:存贮机制、方便的引用机制、对象属性和对象间关系的管理.支持OOA/D的各种辅助工具也作为OMS中的对象来统一管理.OMS的组成如图14图示.

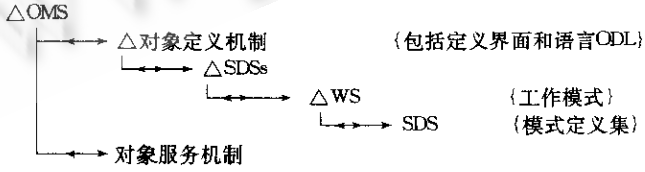


图14　OMS的组成

(2)OOMIS

从OO的观点来看,OOMIS可看作OMS的一个子型.OOMIS继承OMS的结构和服务,即可通过继承来构造OOMIS.在传统方法中颇为棘手的软件结构设计的问题,在DDD方法下变得十分简单(见图15).

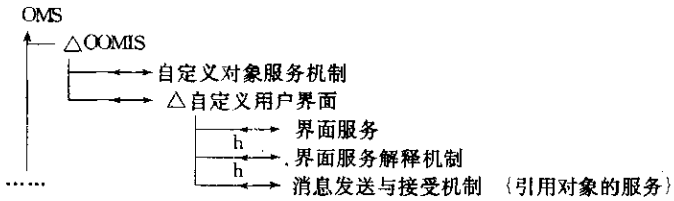


图15 OOMIS结构

(3)属性型、链型

OMS 对象的型规定了其所有实例对象的特点. 所有对象共享共同祖先型 OBJECT 所拥有的最小共同属性集.

属性型定义了所有其实例的特性,包括名、值型、初值和缺省值. 属性型的值型可以是整型,布尔型,串型或日期型. 它们的初始值分别为:0,false,null 和一参考日期.

链是从一初始对象到一目标对象的单向联系. 链可以建立由一个对象到另一个对象的查找. 链型规定了所有它的实例的特点,它有以下特性:(1)标识;(2)出度;(3)一组源对象型;(4)一组目标对象型;(5)如果其出度为"多",则需有一组关键属性型来描述;(6)一组属性型(可能为空). 对于出度为多的链型,关键属性型提供一种命名链的方法. 链型可定义为关系型的一部分. 关键字定义为关键属性的列表,它可区别所有从给定源出发的链.

(4)模式

定义 5.3.1. 模式是一个对象定义,它提供给系统一系列服务. 例如,(1)当一对象创建时,初始化属性;(2)针对 OMS 数据空间的语义特性保证服务的正确性;(3)在解释请求时给出某些信息.

定义 5.3.2. 模式定义集(SDS)是相关定义的,完整、一致、自恰的集合. SDS 是 OMS 对象并可以根据 OMS 范围内唯一的名来访问. SDS 是参与构造工作模式的基本元素. 一个给定型的定义也可属于几个 SDS. 定义之间可存在联系. SDS 将定义之间的联系约束于同一个 SDS 中.

定义 5.3.3. 工作模式(Working Schema,简称 WS)是全部模式的部分视图,允许同时有多个工作模式存在. WS 提供引用者所关心的那部分信息.

定义 5.3.4. 型的定义构成了全部 OMS 模式,并将这些模式组成较小的 SDS. 每个 SDS 都作为一个 OMS 对象并支持对全局 OMS 的部分视图描述.

对模式的修改总是通过对 SDS 的更新来进行的,从 SDS 到实际模式的映射由 OMS 模式管理机制自动进行. 工作模式由一个或多个 SDS 组成.

定义 5.3.5. 对象定义语言 ODL 是一个模式描述机制,它的解释器可以建立在模式定义机制之上负责将 ODL 语句转换为 SDS 上的服务.

每个定义被创建时,都有一个系统范围内唯一的标识,根据命名规则将型定义组合到 SDS 中. 只要相连的型的实例存在,定义的标识就是有效的,且可通过引用 OMS 服务查询.

支持 OMS 的底层环境是建立在 UNIX 之上的基本机制和分布式机制,这些机制负责进行网络管理、进程间通信、对象分布管理、进程分布管理等等.

6 结　论

本文从实用的观点提出一种OO MIS开发方法.它具有以下优点:(1)简洁,用DDD所提供的符号工具构成的图形没有交叉线,便于在有限的纸张上表现出更多的内容,给图的生成、打印、显示带来了方便;(2)采用所定义的OO概念体系,并将其贯穿于系统开发的全过程,简化了系统分析、设计与实现;(3)分析阶段和设计阶段采用统一的一套符号系统,弥补了以往SA/SD分别采用两套符号体系及其自身图形符号并非结构化的不足;(4)表达能力强,可以表达多种关系结构,甚至还可用来书写用户使用说明书,从而使整个开发过程使用统一的一套符号系统;(5)与OMS相衔接将使DDD方法贯穿于分析、设计、实现的全过程.分析设计文档及其辅助工具也作为OMS的对象来统一管理.

通过我们近期开发的财政管理信息系统群CZ—MISs及其开发环境MISDE的实践表明该方法是实用的.目前正将其应用于企业MIS群的开发中,使其在实践中得到完善和发展.

致谢　感谢中国科学院软件研究所仲萃豪研究员给予作者研究工作的支持和帮助.

参考文献

1　诸葛海.信息系统与类比方法论.杭州:浙江大学出版社,1992.
2　Zhuge Hai. Research on object analogical reasoning. Chinese Journal of Advanced Software Research, 1994,1(2): 166—177.

DDD——AN OBJECT—ORIENTED METHODOLOGY FOR MIS DEVELOPMENT

Zhuge Hai

(*Institute of Software, The Chinese Academady of Sciences, Beijing* 100080)

Abstract　The paper first redefines the basic concepts of object—oriented. Then, a directed directory diagram (DDD) is proposed to serve as a unified tool for describing objects and their relations such as composition, inheritance, service and state transformation. Therefore, an object—oriented MIS development methodology is formed by corporating concepts, principles, strategies, assistant tools and object management system. The method had been applied to the finance MISs development.

Key words　Software engineering, software engineering environment, object—oriented, management information system.