面向对象技术述评

蔡希尧 西安电子科技大学 (710071)

摘 要 面向对象技术已成为信息系统应用和开发的主流技术。本文首先对应用情况做了概要的阐述，然后讨论面向对象的开发方法，并评论发展中存在的争论问题。

关键词 面向对象，软件开发方法，面向对象程序设计，面向对象分析，面向对象设计，面向对象需求

OBJECT-ORIENTED TECHNOLOGY REVIEW

Cai Xiyao

University of Electronic Science and Technology of Xian, 710071

Abstract In contemporary information systems, object orientation is received as the mainstream technology, both in application and development. In this paper, the state of application is reviewed first, then we discuss the state-of-art of object-oriented development methods. At last, some comments on the debating problems are given.

Keywords Object orientation, Software developing methods, OOP, OOA, OOD, OOR

1 概述

"面向对象"这个词来源于程序设计，但它现在显然不仅是一种程序设计范型，而已经发展成为一种信息系统开发的方法论，一种新的信息技术，并且日益得到普遍的重视，逐渐成为信息系统应用和开发的主流技术。我在《面向对象技术》一书中，对此已有较详细的论述[1]。

本文主要讨论面向对象技术在信息系统发展中的作用，面向对象的开发过程和方法，存在的争论问题及评论。

2 面向对象技术在信息系统中的应用

在当前的信息系统中，面向对象技术的应用已遍及各个方面，包括程序设计、图形用户界面、操作系统、数据库系统、网络计算、通信系统、软件重用、系统模拟、虚拟现实、人工智能、计算机辅助设计、计算机集成制造等等。限于篇幅，不能一一列举这么多的应用情况，只就日常工作中关系最密切的操作系统、数据库系统、网络计算、通信系统等四个方面，考察面向对象技术的应用概况，以说明信息系统的发展与面向对象技术的关系。

2.1 操作系统

操作系统是计算机运行的核心部件，目前已经不同程度地引入面向对象技术。

什么是面向对象操作系统，尚无通用的定义。通常把采用微内核、支持分布对象计算、装备有对象登录应用框架、具有多任务功能等特点的操作系统，称为"面向对象的操作系统"(OOOS)。

OOOS 已经有若干产品，如 Windows 中的 OLE；IBM 的 OS/2 Warp 中的 SOM；Apple、IBM、Novell 及 Sun 等联合开发的 OpenDoc 等，多用于台式计算机。Apple 的 Newten 和 General Magia 的 Magia Cap，是用于个人数字助理(PDA)的，采用 ROM 对象存储器。规模更大的如 NetView for AIX4.1，用于 AIX 网络管理平台，增强了分布式管理的能力，它具有事务监测和更好的图形和安全能力，以及对象聚集的特性，能支持 OS/2 和 Windows NT 客户机，并将支持网络管理协议 SNMP2.0。

为面向对象操作系统的开发提出了一些模型，著名的有 Microsoft 公司的 COM(Component Object Model)，OLE 建立在此模型的基础之上；IBM 公司的 SOM(System Object Model)，被用于 Warp 之中。SOM 是语言中立的对象模型，以二进制方式提供类库，跨越进程和机器的对象间通信使用 SOM 的分布式扩展(DSOM)。此外，还有 DEC 公司的 COM(Common Object Model)，它提供很强的接口模型和一个清晰的二进制标准。

互操作性是使用 OOOS 的最大障碍，上述的 COM 和 SOM 等对象模型，是不兼容的。在网络快速普及、分布式对象计算日益受到重视的情况下，面向对象操作系统的互操作性，便成为很重要的急需解决的问题。目前最受注意的是对象管理集团(OMG)所制定的标准 CORBA(Common Object Request Broker Architecture)[2]，已为世界开放系统组织 X/Open 所承认，做为一种试用标准。CORBA 支持客户/服务器工作方式，其核心是"对象请求中介"

收稿日期：1996 年 5 月 29 日 蔡希尧 西安电子科技大学教授，博士生导师，中国电子学会常务理事，陕西省电子学会理事长。目前从事信息系统及软件工程等方面的研究工作。

(ORB),能提供异构环境下不同机器间的应用互操作,无缝地联结多种对象系统。

2.2 数据库系统

数据库是信息系统的核心。目前应用对数据库系统的要求是:

(1)支持复杂的数据类型:如变长数据、非结构化字符串、图形、图象、声音、动画等,传统的数据库对这些信息缺乏描述、操纵和检索的能力。

(2)支持复杂的数据结构:这是工程设计的层次性和设计对象的多样性所需要的。

(3)支持工程事务管理:这些管理的特点是时间长、存在多种协作关系、具有试探性,传统的数据库并发控制策略是以原子性和可串行化为基础的,无法满足这些要求。

(4)支持应用的演化:随着环境、要求、设计人员的思路深化,数据及版本应随之演变。

(5)版本和配置管理:在工程的演化及反复过程中,要有数据和不同设计方案的记录,持久性对象要有版本管理。

(6)支持分布式计算及独立于平台的大型对象存储,以适应大规模的分布式计算的要求。

(7)具有导航式查询和关联式查询以及实时交互的能力。

(8)有新的约束、触发和规则,以支持复杂的数据结构和操作,来保证数据的完整性。现有的关系数据库系统面对这些问题,引进了一些新的技术,如二进制大对象、分布式数据库、用户定义类型和抽象数据类型、动态可伸缩技术等,但只能解决小部分的困难,而继续存在的问题则是:缺乏复杂结构对象的建模能力、不能提供不同层次的抽象、程序设计语言和数据库语言之间存在着严重的断层、触发和约束机制不足等等。这些问题,使用对象技术,可以得到较好的解决。

面向对象数据库和当前广泛使用的关系数据库是互补的,关系数据库系统在事务处理方面仍然是很有效的,面向对象数据库系统的主要应用领域则是:CAD、CAM、CIMS、OA、VLSI 设计、CASE、GIS 等,它能够为这些应用领域提供:复杂的建模能力;大规模的数据管理;可伸缩性事务管理;语义模式的设计;动态模式演化;丰富的约束管理版本管理;计算完整的数据库编程语言。

除了专门设计的面向对象数据库系统之外,已经占领市场的主流关系数据库系统,也都纷纷引入面向对象技术。如 Oracle 的 Form4.5,采用面向对象技术,包括:继承性、封装性、多态性、重用性等面向对象的主要特征。其继承性可在任何一个图形用户界面(GUI)标准环境下,一次性定义一个对象及其行为,此对象可在各种 GUI 中被所有应用程序继承。重用性则允许一次性地建立对象,可在应用开发中重用。又如 Informix 的新产品 NewEra,是一个开放的、图形化的、面向对象的第二代的客户/服务器开发环境,其核心是 NewEra 语言,它是面向对象与结构化语言的结合,还具有 4GL 的功能。NewEra 的强大功能主要来源于类库,其来源有:Informix 标准类库、第三方(Informix 的合作者)类库、开发人员定义的类库。类库包括数据操作类库(DCL)、文件存取类库(FCL)、连接类库(CCL)、可视类库(VCL)、应用服务器类库、应用框架及业务图形等,通过类库可以扩展应用系统。

在数据库查询语言方面,也正往面向对象方向发展。即将推出的新的 SQL-3,将采用面向对象的类型系统,每个对象有唯一的标识,定义了"子类型-超类型"关系以支持重用和多重继承。

在数据库开发工具方面,有 Powersoft 公司所开发的 PowerBuilder,采用面向对象技术,其图形化应用环境,是面向对象/事件驱动的可视化的前端开发工具,易学、易用,界面非常友好,能够支持 Oracle、Sybase、Informix、SQL Server、Ingres、DB2、Foxpro 等流行的数据库系统。通过 Library Manager 提供程序共享和对象重用,显著提高了软件开发效率。PowerBuilder4.0 能支持 OLE2.0;内置 C++ Class Builder,可直接用 C++语言编译。

2.3 网络计算

人们正在议论网络计算时代的到来。1995 年 5 月,SUN 公司推出适用于网络应用的新语言 JAVA;同年 11 月 SUN 公司又推出新的网络计算体系结构 Ultra Computing,重新定义了计算模式;同一个月,在 95'Comdex 会上,IBM 公司总裁 Gestner、Novell 公司总裁 Frankenburg,Microsoft 公司总裁 Gates 等,相继发表演说,共同阐述了网络就是计算的观点,更引起人们对网络计算的重视。

实现网络计算的主要方面,包括语言、计算模式、信息共享、互操作及管理等,面向对象技术起着越来越重要的作用。

JAVA 就是一种面向对象的平台独立的语言,它有效地解决了网络编程问题,可靠性高,安全性好。特别适用于 Internet 的 WWW 环境,名为 HotJava 的 WWW 浏览器,具有实时交互多媒体功能。在句法上 JAVA 与 C++很相似,采用单一继承性和多线程,功能经过优化,比 C++更加安全。JAVA 一经推出,世界上许多公司纷纷响应,表示支持,并争相获取它的使用权。Netscape 公司的 Netscape Navigator2.0 版将支持 Java 程序;Microsoft 公司于 1995 年 12 月宣布获得 Java 的使用权,将在 Windows95 和 Windows NT 上提供支持 Java 语言的浏览程序 Explorer;Novell 与 SUN 达成协议,把 Java 与 Netware 集成使用;Borland 公司今年开发一种基于 Java 的图形工具,称为 Latte;在 VB Script 中将采用 Java。HotJava 浏览器汉化的工作也已经开始。

客户/服务器计算是网络计算的主导模式,而对象技术是这一计算模式的基础。

在信息共享方面,基于对象的二进制部件库,正在成为主要的方式。以二进制方式提供类库,不需要重新编译或重新连接客户。

网络管理和互操作问题,其解决方案同样依靠对象技术的支持。网络管理的发展方向将基于类库,而且是可剪裁的。OSI(ISO/CCITT)开发了两类网络管理标准:Common Management Information Protocol/Common Management Information Service Elements(CMIP/CMISE),使用 OO 方法于管理对象的存取。OSF 的分布管理环境 DME 基于 CMIP 和 SNMP 以及其他事实上的标准,使用对象技术以

及 OMG 的对象规范,用 C 及 C++语言和标准的 API。前面已经提到的 CORBA,不仅是网络实现互操作的标准,OMG 还把它作为分布式对象计算的标准,并把 TCP/IP 作为 LAN 和 WAN 上移动对象的底层技术支撑。除了 CORBA 以外,OMG 还定义了"公共对象服务规范"(COSS,Common Object Service Specification),包括两个部分:①定义了基本运算的对象集合,包括:对象定义、与事件有关的服务处理、对象的持久性与生命周期;②对象关系、对象的形象化、对象处理中的事务和并发控制。

2.4　通信系统

通信系统软件的共同特点是复杂度高,特别是需要提供多种服务的时候。一个典型的通信服务往往需要几千个对象,它们在数百个硬件对象上运行。所有这些对象以复杂的方式交互作用,而它们的行为是不可预测的。目前,通信软件正在从以文本为基础向着可视的方向转变,使用图形、图标、符号、色彩、菜单、多窗口显示等技术。为了适应这些新技术的要求,更快地增加新的服务,通信软件必须以易懂和易扩展的方法来开发。因此,面向对象技术越来越多地被用到通信软件的开发之中,例如:

- Bellcore 的 SPACE 系统,这是一个智能网络集成的服务生成和服务管理环境;
- 西南 Bell 电话公司的定制网络管理系统;
- NTT 的定购电缆网络管理系统;
- 欧洲的宽带面向对象服务技术(BOOST,Broadband Object-Oriented Service Technology),这个计划是研究面向对象方法在系统建模中的有效性,以及有关服务生成过程的开发技术。

根据 AT&T 公司的"呼叫和数据聚集系统"(CADCS)开发组使用面向对象的五年经验总结表明[3],通信系统的软件使用对象技术可以明显提高编码效率、分析和设计的重用程度,缩短开发周期,简化系统集成,并且是需求生成和使用的有效方法。

交换机是通信系统的核心装备,新一代交换机将以客户/服务器为主要工作模式,使用面向对象技术[4]。

以上的这些事例,足以说明面向对象技术在信息系统中的普及程度和它对今后发展的重要作用了。

3　面向对象的开发方法

八十年代中后期,面向对象程序设计渐趋成熟,做为一种新的程序设计范型,逐渐为计算社会所理解和接受,应用也得到发展,这些成就促使研究者把一部分注意力转向更广、更深的层次,去考虑把面向对象用于软件开发以及信息系统开发的全过程,并不断地取得成果,于是,一种新的开发方法论开始形成,虽然还不完备,但实践证明这一新的方法论有超越八十年代处于全盛的结构化方法论之势,我们称这一方法论为"面向对象软件开发"(扩大一点,则称"面向对象系统开发"),缩写为 OOSD。

当前,信息系统正在加速发展,其趋势是:

(1)应用趋于复杂,规模大。

(2)功能从处理向系统模拟和集成、从集中向分布计算、从文本为主向图形和多媒体转移。

(3)图形用户界面和客户/服务器模式迅速发展,迫切要求有新的能方便地表示实体并适用于异种结构的分布系统的开发技术。

(4)多变的需求和强烈的竞争要求开发时间短,但质量却要求很高,因而矛盾加大。

(5)在大量的硬件平台迅速变化的情况下,应用的可移植性和不同背景的终端用户对易学易懂的要求不断提高。

要满足这些发展要求并克服存在的困难,用传统的开发方法难以解决,而使用 OOSD 则可以使矛盾得到缓解或克服,因为:

①对象与客观世界的实体相似,便于为客观世界建立模型。

②面向对象的结构符合分布计算的要求。

③面向对象的系统结构就是它自身的分类,而这种分类的通用性足够适应任何可能的扩充和修正,在开发过程中结构可以保持一致。

④从需求阶段到实现阶段,使用同一的对象概念,开发过程中阶段的改变,不需要方法或范型的转换,可以做到比较平滑的过渡。

⑤重用和重新工程在信息系统的开发中被提到比较高的地位,面向对象开发能更好地适应重用和重新工程的应用。

面向对象的这些特点,正是它得以发展的原因。

面向对象开发方法包括三个基本的成份:面向对象分析(OOA)、面向对象设计(OOD)和面向对象程序设计(OOP),其中 OOP 是基础,OOA 和 OOD 是应用 OOP 的机制,加上分析和设计的技术(很多都是已有的)而形成的。近年来,对于需求描述也开始采用面向对象技术,简写为 OOR(OOR 常常和 OOA 被合并在一起,记为 OOR/OOA,因此,开发过程还可以看做是三个阶段。这三个开发阶段,以及以后的测试和维护,还可以加入需求跟踪,构成一个完整的面向对象开发过程。

面向对象的软件开发最早的倡导者是 Booch[5],采用 Ada 语言加以实现。接着,Goddard 空间飞行中心提出的 GOOD(通用的面向对象软件开发)法,欧洲空间局提出的 HOOD(层次的面向对象设计)法等,都是 Booch 法的扩充,也用 Ada 语言实现。1989 年,由 Wasserman 等人提出的"面向对象系统设计"法[6][7],比 GOOD 和 HOOD 等方法有新的发展,把用于问题的划分和分析的自上而下的结构化技术,与汇集和建造系统的自下而上的技术结合在一起,形成一种混合的开发方法,并加入类和继承性。

现有的大多数 OOSD,都是从对象所封装的数据出发的,被叫做"数据驱动法"。1989 年,提出一种新的方法,称之为"责任驱动法"[8][9],被应用于大型的系统开发,效果很好。这一方法的出发点是:每个对象都是系统的一个部分,对系统负有某种责任,而对象是通过相互合作来履行责任的。

除了数据驱动和责任驱动以外,还有事件驱动法,其出发点是当某个事件发生时,将触发系统,引起对象状态的变换,而事件则在消息的到达时出现。现有的窗口系统,基本上都是采用事件驱动的。

面向对象的软件开发起源于 OOP,从 OOP 发展成 OOD,然后是 OOA,OOR 则是近几年才开始

研究的。但实际开发过程则是从 OOR 开始，其次是 OOA，接着是 OOD 和 OOP。

现有的面向对象技术能很好地描述问题域，因此，需求可以用 OO 加以形式说明，这样更便于与 OOA 的衔接，但现在还缺乏有效的 OOR 技术可用于对外部行为和特性的描述，这是目前正在努力解决的问题。

当需求以 OO 的方式表示，还有利于系统开发过程中可跟踪性的实现。开发过程的可跟踪性是指在系统开发的整个生命周期内正反两个方向上描述和跟踪需求，为系统开发的管理与控制、开发质量的保证等提供了有效的途径，是近年来开始研究的课题。

面向对象分析是对问题的空间的理解和分析，分析阶段通过类或对象的认定，确定类之间（或对象之间）的关系，然后则对它们的属性、所提供的服务和所需要的服务进行描述，并按照它们之间的关系进行组织，得到类（或对象）结构。

由于对问题空间建立模型的方法不同，理解和分析过程不一，因而形成多种 OOA 方法。Coad 和 Yourdon 最早系统地总结了面向对象分析方法[10]，以后又相继提出其他的面向对象分析的方法，如 Jacobson[11]，Martin[12]，Rumbaugh[13]，Shlaer[14]，Wirfs-Brock[15]等人所提出的方法。这些方法各有自己的特点，也有相同之处。

Coad 和 Yourdon 从实体/关系出发，概念简单，方法容易掌握；其缺点是对每个对象的功能和行为注意少。Jacobson 等人所提的方法是以“使用案例”为基础，把使用案例做为功能和行为的抽象，但同样对每个对象的功能和行为注意少，对于关系的建模方法比较特殊，使用案例如何集成缺少论述。Martin 和 Odell 的分析方法对行为的建模概念丰富，缺点是如何从行为建模转移到结构建模缺少指导。Rumbaugh 等人的方法的主要特点是建模的能力强，对结构、功能和行为都能够加以描述，但缺乏明确的集成方法。Shlaer 和 Mellor 方法是基于域及子系统的分析，对结构、功能和行为都能描述，但信息模型是基于关系的，对于复杂对象缺少支持。Wirfs-Brock 方法使用子系统的概念，并重视了合作对象的行为，但对于属性、关系及复杂对象缺乏明显的概念，OOA 和 OOD 的界限不清，忽视了对象的内部行为。所以，可根据不同的应用要求，选择适用的分析方法，或取其所长，混合使用。

面向对象设计是通过类的认定和类层次结构的组织，确定解空间中应当存在的类和类层次结构，并确定外部接口和主要的数据结构。面向对象设计和通常的软件设计一样，划分为两个步骤：概要设计和详细设计。概要设计的主要任务是定义系统是如何工作的，因此，对于工作平台、计算能力、和存储容量等不加限制。在详细设计阶段，要考虑这些问题，并进一步细化概要设计所生成的产品。

概要设计的主要工作是：

· 对象行为和对象间交互作用的进一步细化，加入必要的新对象。

· 按照分析阶已经组织成的结构形式（层次结构），对类进一步认定，以得到解空间的结构形式（不再是问题空间的描述），为实现提供支持。

· 对重用的支持。类认定以后，组成类库，用以支持重用。

详细设计是紧接着这概要设计进行的，其目的则是为实现做好准备，而编程所需要的主要是有关对象的描述，因此，加细概要设计阶段所给出的对象描述将是这个阶段的主要工作。在互相通信的对象中，对象属性的类型要取得一致。

在面向对象开发方法中，分析和设计虽然是不同的活动，但两者是密切配合来建立一个问题域模型的，并以构成应用基础的一组实体及其相互关系来描述问题域；在分析阶段开发所获取的信息，不仅是设计阶段的输入，同时是设计阶段的一个完整组成部分。分析得到对象及其相互间的关系，而设计则是解决这些对象及其相互关系的实现问题。因此，所采用的分析方法不同，设计的方法在细节上也有所差异。

OOD 和结构化设计有一个本质的区别：OOD 是一种自底而上的设计方法，而结构化设计则是一种自上而下的方法。这种差别主要反映在对模块的认识上。OOD 将类看作是一种基本的模块，通过继承性加以功能和结构的扩充，来建立模块之间基于共享的联系；通过把能否形成类库中的可重用部件作为设计优劣的标准，来控制模块设计的通用性。因此，OOD 从保证基本模块的通用性和可装配性入手，使得设计结果既满足当前的应用要求，又有利于以后的扩展以及其他设计的重用。结构化设计将目标系统的功能作为划分模块的标准，按照分析时确认的数据流和加工的关系，先使得模块的调用关系与功能的抽象层次相对应，再考虑底层模块的共享可能。由于从高层模块开始就与当前的需求密切相关，并且使数据结构的设计服从于功能实现的要求，所以尽管可以很好地满足当前的要求，但实际上已经为以后的扩充和重用设置了障碍。

设计之后是程序实现。学习并掌握面向对象程序设计，重要的是理解这种程序设计范型的特点以及与传统程序设计的区别。OOP 的基础是对象和类，基本机制是方法、消息和继承性，要掌握面向对象程序设计，必须从这些特点入手。在 OOP 中，对类库的应用是重要的环节，而类库和专用构件的设计也是一种程序开发的任务。近年来，更高层次的“部件库”已经投入实际使用。这种程序设计的新风格，是传统程序设计所没有的。

4 讨论和评价

面向对象开发方法发展到现在，有赞成的，有反对的。这不仅表现在问题的讨论中，也表现在一些计划的实现过程中，例如美国国防部于 1988 年开始，决定修改军用编程语言 Ada，引入面向对象语言机制，定名为 Ada-9X，但对于是否要加入继承性等主要问题上，意见不一，争论了多年，最后还是确定加入继承性，直到 1995 年才推出新版本 Ada-95。这一事例表明，面向对象技术是有生命力的，尽管有不足之处，但在不断地发展成长。

反对者的观点基本上可以归纳为：

(1)使用 OOP 需要对编程人员进行事先培训，并要求他们用一种新的方式思维，这是一些公司和从业人员所不愿意的。

(2)在现代的信息系统中，关系数据库已经普遍

使用,应用 OO 技术将带来麻烦。

(3)现有的 OOP 语言和方法论还不能满足信息系统不断变化的要求。

(4)标准缺乏。这些的确是需要改善和克服的事实。

赞成的人主要是从面向对象技术的特征和机制,以及实现效果来看问题,例如用 OOSD 开发方法,开发阶段的改变,不需要方法的变化,能做到阶段间的平滑过渡。用 OOSD 开发的软件,有更好的可靠性、可重用性和可移植性。根据一些应用的调查结果表明,OOSD 开发的成功率和程序生产率都是所有开发方法中最高的。赞成者还认为,OO 程序可读性好,维护方便,等等。

OOSD 的这些优点,正是使得近年来对它的应用迅速发展,特别是一些大型软件的开发,纷纷采用 OOSD,并不断地有所创新。

但面向对象技术究竟还是年轻,有许多不完善的地方有待改进和提高。

4.1　理论落后于实践

就整体来说,面向对象技术的理论落后于实践是明显的,OOP、OOD、OOA 和 OOR 都缺乏明确而系统的理论。就连"对象"是什么这个最基本的概念上,还存在不一致的定义和模糊的认识。在 OOP 中,对象往往笼统地被解释为一个实体,加以若干约束;有的作者在定义面向对象程序设计时,规定对象必须是抽象数据类型[16]。在信息系统中,对象的定义比之于程序设计中的定义的约束较弱。

作为程序设计所需要的真实或虚拟世界的概念模型的对象,有一些基本的特性:

- 对象有生命期,它们被生成、存在、最后被清除;
- 有自己的结构;
- 是一种可共享的资源;
- 具有状态,而状态是可改变的;
- 具有行为,能为其他对象提供服务;
- 对象有外延,这是识别对象的依据,对象标识就是一种外延。在现实世界中,还有另一类实体,例如数,是永存的,与有始有终的对象这种实体,在意义上有本质的区别。

程序设计语言为支持对象的基本特性,提供相应的构造。对于特性应用的侧重面不同而形成不同的对象定义。程序设计所主要关心的是对象的状态、行为和标识,于是把对象可定义为:

对象＝状态＋行为＋标识

信息系统着重于从用户来观察并使用对象,注意对象的生命周期、状态变换的发生和结果,因而把事件、属性、生命周期以及状态变换后属性的取值作为描述对象的要素,把对象定义为[17]:

对象＝事件＋属性＋生命周期＋事件发生后属性的值

对于对象基本特性应用的侧重面不同,还形成不同的系统。例如对象的生命周期的长短随着应用的不同就有很大的差别,在程序设计中,对象存在的时间决定于它们的作用域,被称之为"暂态对象";数据库中的对象需要长期存在,被称之为"持久对象";这两类对象是各自独立发展起来的,现在,随着对象概念研究的深入和 OO 技术的发展,有可能把它们统一起来。

所以,理论是重要的,只有当我们对于对象的基本特性认识清楚以后,才能从多种不同的应用中了解其实质并加以确切的定义和使用。没有理论,发展就缺乏指导,形式规范就不能形成,严格的描述不能做到。现在急需的是要制定 OO 技术的工业标准,以利推广应用,而理论是制定标准的基础,这是今后一个时期内需要努力解决的基本问题。

4.2　关于 OOP

类型、类和继承性是构成 OOP 的主要成分,但存在一些模糊不清而有待深入研究的问题。类型和类、继承和子类型化,是相同还是有区别?在实际使用的语言中,它们之间常常被看做是等同的。但实际上,类型和类,子类型化和继承,在概念上是有所区别的。

从面向对象的观点来看,一个类型是一组对象,它们共享相同的外部可观察的行为。决定一个对象是否属于某一类型,是对象对于哪些消息并以什么次序予以响应,以及它的变元和结果之间存在的关系。一个类也是一组对象,它们有完全相同的内部结构。类所描述的是它的实例如何被建立,而类型所描述的是它的元素如何被使用。

类型和类的区别反映了子类型化和继承的区别。子类型化是类型间的包含关系,而继承则是类之间对代码共享的一种方式。继承的目的是明确的,但继承关系却形式多样,必须在语义上加以刻画,才能达到正确使用的目的。

类似地,IS-A 和继承是等同的吗?不少人把两者看做是完全等同的,但实际上也是有区别的。IS-A 关系来源于语义网络,并用于数据库的语义数据建模,它是共享属性的实体之间的关系。IS-A 关系可用于面向对象系统,为系统建模减少冗余和复杂度提供了一种方法。面向对象语言中的继承,主要是为了提供码的重用,并做为语言类型系统灵活性的测度。当然,两者之间也有一致的地方。IS-A 可以影射至 OOPL 的继承关系,在一个域的模型和从它导出的程序之间给出更好的可跟踪性;还可以为形式化打下基础。

4.3 关于 OOR、OOA、OOD 及它们之间的转移问题

通常都认为面向对象软件开发的一个特点是开发阶段的改变,不需要范型的改变,过渡是平滑的。与传统的方法比较,面向对象开发在这方面确实有它的优点,但是,这种阶段的改变,是容易的吗?这是一个有争论的问题,有拥护的,有反对的,也有持调和态度的。早期研究面向对象开发的学者,大多持拥护的态度。但是,实践的深入要求我们更全面地来思考这个问题。

需求工程包括两种主要的方面,问题分析与需求规范。问题分析是了解问题域的所有方面,以决定如何最好地解决用户所提出的特定问题。需求规范则是关于待解决问题所需的系统的外部行为及特性(系统所有的输入和输出的描述、以及输入和输出之间的关系描述)的软件需求描述的文档,对于大规模软件的开发,这是很关键的工作。规范必须是不模糊的、一致的、并且尽可能完全。

前面已经说过,现有的面向对象技术能很好地描述问题域,它们有助于分析,但还没有有效的 OOR 技术可用于对外部行为和特性的描述。这是面

向对象技术对于分析和需求的有效程度的不同之处。

任何软件设计技术的目的都是为了把软件需求转变成可行的最佳的软件结构。所谓"最佳设计"是指所设计的软件是可测试的、可维护的、可增强的、可读的、可靠的，并且能够稍加修整就可满足严格的性能或安全要求。面向对象设计的目的是通过对象特性的有效利用，使得可维护性、可重用性、可增强性、可靠性等得到优化。由于 OOD 具有把数据和操作密封在一起的特点，差错少，有了差错也容易发现，在改变时不易引入错误，这些都是优点。

需求技术的目的是优化可理解性，其次则是增加正确性、一致性、完全性等。设计则是为了得到最佳的功能和性能。系统需求规范对最佳的功能和性能是没有什么直接帮助的。因此，OOR 中对象的定义和 OOD 中对象的定义有所区别。OOR 所定义的对象是为了表示某种现实世界的实体，其目的是为了优化可理解性，而 OOD 是为了优化功能和性能，以及可维护性。在 OOR 中，对象的选择原则很简单：这个实体是问题域中的一个重要部分吗？如果是重要的则选用，如果不是则不选用。在 OOD 中，选择的准则是：我们能建造它吗？选用它是否能有效地使用数据抽象等特性？此外，在分析阶段，对于对象所提供的服务不必详细描述，在需求规范中也是如此；但在设计阶段，对象所提供的服务必须有详细的描述。这是面向对象技术在开发的不同阶段的要求不同之处。

学者们研究面向对象开发的出发点也不一样，有的是从分析导出需求规范，有的是从概要设计导出需求规范，有的是从问题域开始定义对象和类，有的是从实体/关系图引出对象，有的甚至是从传统的结构化设计出发转移到面向对象，再加上前面所述的不同开发阶段的不同要求，因此，出现了不同的观点[18]：

(1)认为需求和设计中的对象是一样的，分析的模型可以用于设计和实现(Jacobson and Embley 等人)。

(2)认为从分析到设计，是逐渐扩充模型的过程，而扩充主要是加入某些成分，所以转移是容易的，分析中的对象就是设计时的对象，但设计时要予以细化(Coad and Yourdon 等人)

(3)认为早期所认定的类和对象可用于整个设计阶段，分析的结果就是设计的开始要用的输入，设计主要是做加细的工作(Booch)。

(4)认为设计可以与需求有区别，但相信设计所得到的子系统(类的集合、聚集、操作、事件和约束等)是对象的组合，所以，设计的概貌和需求的概貌相似，不需要引入新的对象(Rumbaugh 等人)。

(5)不强调需求中对象的作用，认为这样会限制设计的自由，面向对象程序设计主要是通过反复加细来开发并提高可维护性(Cox and Meyer 等人)。

(6)认为描述外部系统行为的需求文档是在 OOA 之后，OOD 之前需要的，当需求规范有了以后，关键是如何分离复杂的问题。对于一个复杂系统，需求和设计都是困难的，不能使两者同时进行。有关类、技术、契约等的文档将提交给设计，但设计者可以在设计中自由地加以改变。在 OOA 之后，设计仍然是困难的，在得到最佳的设计以前，需要反复的思考，反复地提出问题并解决问题(Cherry、Lorenz、Shlaer and Mellor 以及 Wirfs-Brock 等人)。

4.4 面向对象软件的维护问题

软件维护是软件开发和使用过程中开销最大的环节，面向对象的拥护者认为采用对象技术对维护是大有好处的，但事实上也应当从好坏两个方面来看问题。

面向对象软件以下面两种方式减少了维护过程的困难程度和成本：

(1)节省维护人员花在理解现有程序上的时间。在面向对象的开发环境和工具的支持下，对已有代码的查阅是一种有效而系统的过程。

(2)程序所具有的模块化结构减少了维护上的困难。由于对象的封装性，唯一能够访问和改变数据的方式就是向对象发送消息，这样，在发现数据不正常的时候，就简化了查找原因的过程：只需要检查传给对象的消息，就可以鉴别引发问题的原因。另外，在维护时如果需要加入新的类，也不必担心是否对已有的类会有副作用。

但是，复杂的程序相依性和面向对象程序所具有的继承性、动态联编、多态性等等，都给维护带来新的问题和困难。

相依性是指程序中一个实体被改动时，如果对另一个实体带来影响，则称这两个实体是相依的。在面向对象系统中，相依性分为：类和类的相依、类和方法的相依、类和消息的相依、类和变量的相依、方法和变量的相依、方法和消息的相依，OOP 的维护环境必须提供设施以浏览这些相依关系。

继承性使得对于一个类 C 的完全描述必须在观察 C 的同时，要观察 C 的每一个父类。由于不同的类是在程序的不同的地方描述的，而且可能分布在不同的文件之中，因此，程序员不能在单一的地方得到一个类的完全描述。需要花很多的时间去找寻这种描述。

多数维护工具依赖于程序内部相依度的跟踪，而一些 OO 语言允许动态联编而使跟踪复杂化。多态性使得一个变量可以用于任何类的一个对象，而当一个消息要求一个方法执行时，变量的类型则由对象的类于运行时来决定。在动态联编中，实际被调用的方法实现依赖于对象的类，而不同的实现则建立了不同的依赖关系，静态分析常常不能准确地辨别程序中的依赖关系。有时候，即使对于一条语句，也要通过多个类来追踪一个方法调用链以找到作业是在什么地方完成的，所以，找出关系链是重要的，但动态联编使得找寻这种链变得很复杂。这又给维护带来困难。

如同程序设计有多种不同的语言和风格一样，在面向对象开发的方法上有这样多不同的观点和有待解决的问题是不足为奇的，而那么多的学者能聚集到这个领域来从事研究，争论问题，各抒己见，共谋解决方案，这不正是说明了面向对象技术的重要吗？

5 结束语

面向对象技术从面向对象程序设计开始，最早的面向对象语言 Simula 是 60 年代由挪威人 Dahl 和

面向对象的管理信息系统分析与设计

陶宏才 朱 鹰 潘启敬 西南交通大学计算机与通信工程学院 (610031)

摘 要 本文对面向对象方法(OO)应用于 MIS 系统开发中的诸多问题进行了较详尽的讨论,提出了 OO 对象在管理应用领域存在管理生命周期(MLC)的新思想,并将该思想应用到面向对象的 MIS 系统分析与设计中,取得了较好的效果。

关键词 面向对象,管理信息系统,管理生命周期,关系型 DBMS,对象模型

THE OBJECT-ORIENTED MIS SYSTEM ANALYSIS AND DESIGN

Tao Hongcai, Zhu Ying and Pan Qijing

College of Computer & Comm. Engi., Southwest Jiaotong Univ., Chengdu 610031

Abstract This paper discusses in detail the problems of the Object-Oriented Approach (OO) applied to MIS development, proposses a new idea that OO object has the management life cycle (MLC) in management application field, and applies the idea to OO MIS system analysis and design.

Keywords Object-oriented, MIS, Management life cycle, RDBMS, Object model

1 前言

目前,在 MIS 系统的分析与设计中,有二种方法可以利用,即:结构化系统分析与设计方法(SSA & SSD)和面向对象的分析与设计方法(OOA & OOD)。其中,SSA & SSD 是一种面向过程的方法,强调功能抽象与模块性,在国内大部分 MIS 系统的开发中用得较多且成熟,但这种方法要求现实系统的业务管理规范、处理数据齐全、用户能全面完整地描述其业务需求,并且还要求整个系统较为稳定。不过,所有这些由于多种原因(如:系统业务重构、系统内外环境的变化以及用户素质等)往往不易满足,这是其一;其二,SSA & SSD 方法从系统调查、系统分

收稿日期:1996 年 5 月 27 日 陶宏才 讲师,硕士,长期从事计算机网络与 MIS 系统教学及科研工作。 朱鹰 西南交通大学博士研究生 主要研究方向:计算机软件 潘启敬 教授,四川省计算机学会副理事长,西南网络与信息系统专委会主任,博士导师。主要研究方向:计算机网络,铁路电气化自动化,管理信息系统数据库。

Nygaard 首先提出[19],它出现在结构化程序设计之前。由于超前太早,当时没有被计算社会所接受,但在 Simula 以后的许多研究工作中,吸取了它的丰富思想,包括抽象数据类型(ADT),人工智能中的框架(Frame),语义数据模型,和基于能力(Capability-based)的计算机系统,而这些领域的成就,又反过来促进了 OOP 及对象技术的发展。

OOP 在发展过程中,形成了两种不同的学派:

斯堪地那维亚学派 斯堪地那维亚是 OO 技术的发源地,而 Simula 的设计目标是为了离散事件模拟,所强调的是 OOP 和系统建模之间的对应。这个学派认为 OOP 程序是一种物理模型,主要是模拟世界的真实或虚构部分的行为。

美国学派 强调数据抽象和程序重用,因此,把 OOP 看做是提高程序设计生产率的一种途径。这个学派的观点所注意的是实际应用,目前流行较广,大部分 OO 的应用系统都是由此演生出来的。

OO 技术虽然已经成为信息系统的主流技术,但正如本文已经指出的,有待继续研究和发展的内容仍然很多,包括基本概念、基础理论、开发过程和方法、标准和度量、集成和维护等等,要解决这么多问题,可能需要结合这两个学派的特点和长处,以求更快地取得成功。

参考文献

[1]蔡希尧,陈 平. 面向对象技术. 西安:西安电子科技大学出版社,1993:

[2]OMG Document Number 93. xx. yy, The Common Object Request Broker: Architecture and Specification, Revision 1.2, Draft 29 Dec. 1993

[3]Kamath, Y. H., Smilan, R. E. and Smith, J. G.. Reaping Benefits with Object-Oriented Technology, AT&T Technical Journal, 1993; 72(5)

[4]Ebert, I. and Richards, P.. Technology's Role in Switching Evolution, IEEE Communications Magazine, Jan. 1993; 31(1)

[5]Booch, G.. Object-Oriented Development, IEEE Trans. Feb. 1986; SE-12(2)

(参考文献共 19 篇,因版面有限,其余存编辑部。)