面向对象的软件开发:思想、特性、方法与风格

胡金柱　（华中师范大学计算机科学系）

提要:本文讨论了面向对象软件开发的基本思想、基本特性、基本步骤和设计的风格.

1.基本思想

OO 方法是以对象作为分析和设计软件系统的主体,同时又把对象作为分析和解决问题的基本单位,使之开发出的软件系统更加自然的贴近于客观世界;OO 是按照人们通常的思维方式去建立问题的模型,并以此作为进行软件开发的基础,从而使得被开发的软件系统更易于理解和阅读,更具有可重用性。

对象是自主的、独立活动的实体,客观世界中的任何事物在一定的前提下都可以成为被认识的对象。客观事物本身具有其自然属性和行为,当对其中的某些属性和行为发生兴趣,并抽取出来加以研究时,客观事物就在这些属性和行为的前提下成为我们所关心的对象。这表明同一客观事物,可以成为不同前提下的对象。例如,函授生可以是某个单位的职工类,也可以划归到某个学校成人教院的学生类,还可以划归到某个专业的学生类,这取决于我们具体研究的问题域。此外,对象本身还具有"粒度"大小问题,这也与具体研究的问题域有关。例如,在研究学校建设时,学校里每个专业都是研究的对象;在研究国家教育布局时,每个学校又成为研究的对象。

软件工程是研究软件开发的专业领域,开发一个软件的目的是为客观世界的一部分建立其在计算机世界中的抽象映射。因此,一个软件的开发过程涉及到两个方面的问题:一是客观世界所对应的专业领域,二是计算机世界中的软件产品。具体要求解决的问题往往只涉及专业领域的一部分,这一部分我们称之为"问题空间",而将问题空间在计算机上的实现称之为"解空间"。所以,从问题空间到解空间的映射过程就是软件开发过程。

OO 开发方法的基本思想是:实现"问题空间"到"解空间"的自然映射,让"解空间"真实而又完整地反映出"问题空间"的各种要求。这个基本思想可以用图 1 加以描述。

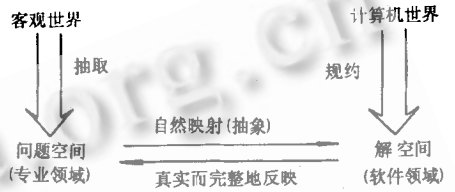

图　1

由图 1 又可以给对象作出如下的定义:对象是指问题空间对象,是问题空间中的实体、实体的属性及其允许的操作的抽象;或者说,对象是具有属性(数据结构)、行为(操作)、标识符和关系的实体。其中关系是指通过消息传递与其他实体的联系,即每个对象都应具有"接收"与"发送"消息的能力。于是可以用如下的四元组来形式化地描述对象:

Obejct＝{Id,DS,OP,MT}

其中:ID 为对象标识符,即对象的名字:

DS 为对象的数据结构,即对象的属性;

OP 为对象的操作,即对象的行为;

MT 为消息传递(Message Transmit),即对象的关系。

2.OO 的基本特性

OO 具有一系列的特性,归纳起来,OO 具有标识的唯一性、分类性、继承性与多态性四个基本特性。

(1) 标识的唯一性(The Sole Identifier)

标识的唯一性是识别问题空间对象的主要标志,问题空间对象的识别主要是指问题空间中实体的识别。而对象的识别方法是目前的主要研究内容之一,目前虽然已有多种识别方法(如借用数据流图 DFD 分解名词与名词短语作为问题空间实体的标识),但其主要的问题是还没有一个公认的、有效而实用的识别方法。

(2) 分类性(Classance)

类即对象类,它是关于对象属性的描述,是一组对象

的抽象,或者说是一组具有一致的属性(数据结构)抽象和行为(操作集)抽象的描述。一个对象类中可以包含有限个或者无限多个对象,但每个对象都应具有唯一标识。例如:

对象类: 学生(Student)
属　性: 学号(Student number)
　　姓名(Name)
　　系别(Department)
　　年级(level)
　　住址(address)
操　作: 注册(Enroll)
　　退学(Drop)
　　年级变更(Change Level)
　　转系(Change Depart)
　　住址变更(Change Address)

将学生对象类的属性实例化,便可得到一个具体的学生对象。例如:

对象: St1
学号: 930074
姓名:张山
年级:93 级
住址:荆州天门
注册:Y
退学:N
年级变更:N
转系:N
住址变更:N

由此可见,对象实例具有对象类的属性和操作特征.

(3) 继承性(Inheritance)

继承性是对具有层次关系的类的属性和操作共享的一种描述方式。在 OO 开发过程中,我们可以先粗略地定义一个类,然后再将其细分为多个子类。父类是子类的高层对象类,子类继承父类的所有属性和操作。例如:

对象类: 研究生(Graduate)
父　类: 学生(Student)
属　性: 学位(Program)
　　导师(Advisor)
　　状态(Occupation)
操　作: 导师变更(Change Advisor)
　　学位变更(Change Program)
　　状态变更(Change Occupation)

其中,状态属性在研究生对象类中可以分为: 公费、自费、委培、在职、跟读等多个状态中的一种。

"研究生"对象类作为"学生对象类"的子类,继承了父类的所有属性和操作,同时还可以定义自己所特有的属性和操作。

(4) 多态性(Polymorphism)

多态性是指相关、而又不同的类,它们的行为共享同一个操作。例如,"改变"可以是一种操作,在"学生类"中可以用来改变学生的"系别"、改变"年级"、改变"住址"。而在"职工类"中,它又可用来改变职工的"职称"、"职务"、"工资"、"住址"等等。而在"图书类"中可改变图书印刷的版次,图书的销售价格等。同一操作可以是多个不同类的共享行为,但每个类中操作的具体实现叫做"方法",所以具有多态性的"操作子"可有多种实现方法。

OO 的上述四个基本特性揭示了 OO 的实质,对它们的了解与认识有利于更好地将它们正确地运用于软件开发过程中,开发出真正的 OO 软件。

3.OO 软件开发过程

OO 开发方法是一种建立在对客观世界进行抽取,得到问题空间的基础上,在计算机世界中映射空间的新的软件开发思维方式。但软件生存周期是客观存在的规律,即 OO 开发方法并没有改变前期规划、需求分析、设计、实现与测试这一软件开发规律。所以,OO 开发方法还比较集中于研究和实施 OO 需求分析方法、OO 初步(系统)设计方法和 OO 程序设计方法。

(1) OO 需求分析方法

OO 需求分析方法就是在软件生存周期的需求分析阶段中,运用 OO 方法的基本思想来完成对问题空间和用户功能需求的"理解"、"表达"和"验证"工作。具体而言,就是以问题空间的对象为主体和线索来把握和描述问题空间,以对象之间的消息传递关系来把握和描述"用户"的功能需求。

用户的功能需求是用户提出的、关于要求开发的软件系统在功能方面的各种需求。传统的需求分析方法是完全面向功能的,它所现解、表达和验证的内容仅仅是用户和各种功能要求,它以功能为基础,又以功能分解为阶段性的结果。这就是传统的面向功能分解的需求分析方

法与 OO 需求分析方法的主要区别。

OO 需求分析可以采用自顶向下的方法逐层分解建立系统模型，也可以自底向上地从已有定义的基本对象类出发逐步构造新的对象类。软件规格说明基于这种以对象概念模型形成，以模型描述为基本部分，再加上界面要求，性能限制等其他方面的要求说明。

(2) OO 系统设计方法

设计阶段的主要工作是将对象及其相互关系模型化，并建立分类关系。具体而言就是针对需求说明产生其形式化的描述，即功能规格说明，并逐步细化，直到可以用某种计算机语言实现为止。

OO 系统设计方法基本上是与实现语言无关的方法，无论用什么实现语言，只要软件系统结构以数据为中心进行组织，遵循模块化的一般准则，并具有数据抽象、属性继承等模块特征，都应认为是面向对象设计的。

模块化方法是控制大型软件设计复杂性的关键技术。传统的模块分解方法是面向功能分解的，它以功能实现过程为基础，而以数据作为模块之间的传输界面。OO 模块方法则是以数据抽象为基础，以“信息隐藏”为准则，以数据操作为模块的界面。

数据抽象和信息隐藏是模块化设计的基本原则，OO 分解方法充分体现了这一原则。OO 分解的模块是由数据及其相关的操作组成的；各模块之间是通过模块界面定义的操作发生联系。所以它们具有很高的聚合度和较低的耦合度，即提高了模块的独立性、易理解性、易测试性、易维护性。

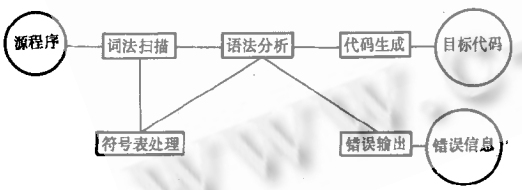

图 2　面向功能分解图

下面以编译过程为例，分别给出它的面向功能分解图(见图 2)和面向对象(OO)分解图(见图 3)。在图 2 中，每个方框表示信息功能变换，方框模块之间通过数据通讯而发生联系；在图 3 中，椭圆框表示数据对象，对象模块之间则通过对象操作而发生联系。

(3) OO 程序设计方法

程序设计是系统设计过程的延伸，是将设计阶段所确定的求解问题的策略，采用某种计算语言具体地编写代码，使之能在计算机世界中具体实现解空间。采用 OO 系统设计的策略是与具体语言无关的，它既可以采用 OO 程序设计语言来实现，例如 C++就是支持 OO 的程序设计语言；又可以采用非 OO 程序设计语言来实现，例如 FORTRAN、C、Ada 等都是非 OO 语言。

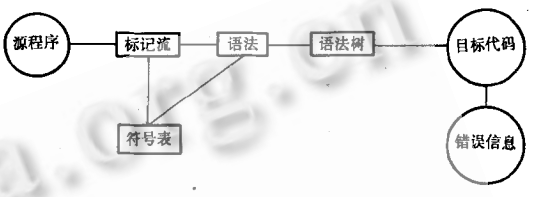

图 3　OO 分解图

在采用 OO 设计方法确定了问题求解策略之后，采用 OO 程序设计语言进行程序设计的基本步骤如下：

①分解确定在问题空间和解空间的全部对象及其属性；

②确定应施加于每个对象的操作，即对象固有的处理能力。

③分析对象间的联系，确定对象间彼此应传递的信息；

④在以上得出的结果基础上，设计对象的消息模式，消息模式和处理能力共同构成对象的外特性；

⑤分析各个对象的外特性，将具有相同外特性的对象归为一类，从而确定所需要的类；

⑥确定类间的继承关系，将各对象的公共性质放到较上层的类中描述，通过继承共享对公共性质的描述；

⑦设计每个关于对象外特性的性质；

⑧设计每个类的内部实现(数据结构和方法)；

⑨创建所需对象类的实例，实现对象间应有的联系(发消息)。

使用非 OO 程序设计语言实现 OO 程序设计的基本步骤与 OO 程序设计语言实现 OO 程序设计的基本步骤是一致的。但使用非 OO 语言设计的 OO 程序必须将 OO 概念映射到目标语言中，而在 OO 语言中，只需编译器自动执行这类映射。具体步骤如下：

(1)将类翻译成数据结构；

(2)将参数传送给方法；

(3)为对象分配存储;
(4)实现数据结构中的继承;
(5)实现方法;
(6)实现联系;
(7)处理并发性;
(8)封装类的内部细节。

4.OO 程序设计风格

程序设计风格是指程序员设计程序时所采用的特有方法和作风。设计风格与具体某种程序设计语言的表达能力无关,任何一种程序设计语言都能具有一种程序设计风格。例如非结构化的过程式程序设计语言,也能体现结构化程序设计风格,只要程序员精心地设计单入口单出口的控制结构,严格按照程序的三种基本控制结构设计程序,不乱用 GOTO 语言,也可以设计出结构化的程序。又如 C++语言是一种多风格的程序设计语言,它既是完善的过程式程序设计语言,又是面向对象的程序设计语言。

OO 程序设计风格把程序看作是传递消息的对象集合,而程序设计就是定义对象,建立对象间的通讯关系,运行程序就是将对象集的初始状态转换成终止状态,输入数据就是建立初始的一部分,程序的输出就是取出终态数据。输入、输出操作也是对象间的通讯结果,是程序中的对象向输入输出设备(如显示器或打印机等)对象发消息的结果。

封装、继承、类与实例都是 OO 程序设计(简称 OOP)风格的重要特征。封装即"信息隐藏",它是将数据以及处理这些数据所需的各种操作连接在一起,构成一个整体的技术。从封装的角度后,对象是封装的数据和操作。从程序设计角度看,OO 程序就是对象及其通讯。如果一个程序只有对象,则有许多对象是具有相同数据和允许相同操作的,这样就组成类。编写程序时只写一个类对象,给一组初值生成一个对象;程序一开始运行首先对象发消息,生成一系列实例对象,然后实例对象互相发消息完成该程序各项计算任务;程序在运行过程中,还可以根据上下需要临时生成新的对象。

这种类一实例机制大大增强了 OO 程序的表达能力,体现了抽象性:程序设计在类一级上进行,而程序运行则是在实例对象一级上进行。结合类的继承性,把凡是调试无误的类放在系统的类库中,每次只在原有基础上派生少量的子类即很完善的大类库,今后在遇到新的开发任务时,一般问题就可以不再编程序了,只需指定几个子类对象生成所需的实例对象,就可以完成程序的功能了。这就体现了 OO 程序的高度可重用性。

5.结束语

如果一个软件是用 OO 设计方法设计的,并且是用 OO 程序设计语言实现的,那么在分析阶段所识别的对象及分类关系就能在设计阶段得到保留并被丰富,而且可以直接用代码去实现,因而不再存在象功能分解软件开发过程中各阶段的断层。

用自顶向下地功能分解方法的观点来看,OO 开发方法是首先识别、设计和实现底层的基本对象类,然后才去解决具体应用任务。

自顶向下与自底向上是两种极端的方法。在软件开发过程中,无论是面向功能的分解法还是 OO 分解法,都不可能,也不应该绝对采用自顶向下或者自下而上的进行。在构造系统时不可能不去考虑底层的可设施性,也不可能在没有应用背景的情况去构造设施。它们二者的区别是:自顶向下法重视的是功能,而自下而上法重视的是对象。重视功能易于较快地得到早期的系统结构,但这是一种短期行为,从长远的观点来看,这种行为是灾难性的:当功能发生变化时,设计者就必须重新设计。重视对象是为了构造可重用的软件集成块(简称软件 IC),使得所开发的系统具有较好的易修改和易扩充性。

OOP 风格与传统程序设计风格具有很大的区别,它的高度动态性使初学者难以有一个静态的"设计程序的概念"。从方法学的角度来看,OOP 开发方法是先进行问题空间的分解,找出映射解空间的对象类及其对象关系,一直到设计系统、实现各个类的全过程是自顶向下,逐步求精的过程。然而严格讲,OOP 只是在分析、设计出系统的对象类体系结构之后,再从可重用的类库中找出相应的软件 IC,赋给所需的参数、传递消息,这个过程是自下而上的集成过程。一般情况下则是自上而下与自下而上相结合的、交互式的设计风格。到目前为止,OO 软件开发方法还是发展中的方法,虽然有许多基本概念是趋于成熟,但还有许多技术和理论尚待软件工作者进一步地研究、充实、提高和完善。本文所讨论的一些内容仍有待于进一步地发展。

参考文献(略)