计算机科学 1994 Vol. 21 №. 2

面向对象的软件开发

蔡希尧　陈　平　金益民

（西安电子科技大学软件工程研究所　西安 710071）

摘要　面向对象软件开发(OOSD)技术的发展正在加速，应用不断扩大。本文讨论四个方面的内容：首先阐述 OOSD 的特点和优点；然后对 OOSD 的两个主要方面：面向对象分析(OOA)和面向对象设计(OOD)所采用的方法，以"数据驱动"和"责任驱动"为基础，结合自己的研究心得和工作体会作了论述；最后探讨有关 OOSD 的生命周期、标准、和计量方法等问题。

一、引言

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风范，逐渐为计算界所理解和接受，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，去考虑面向对象的软件开发问题，并不断地取得成果，于是，一种新的软件开发的方法论开始形成，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(OOSD)"。

OOSD 由面向对象分析(OOA)、面向对象设计(OOD)和面向对象程序设计(OOP)组成，即：OOSD＝OOA＋OOD＋OOP。开发过程中的顺序则是：OOA→OOD→OOP；但技术的发展过程却是相反的，先有 OOP，然后扩展到 OOD，最后才有 OOA，即：OOP→OOD→OOA。

在 OOSD 的三个成份中，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。所以，应用 OOSD 时，必须有 OOP 和分析及设计(如结构化方法)的基础。

当前，应用最需要的是工业质量标准的软件产品，而开发工业质量标准的软件产品却遇到很大的困难，原因是：

(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)图形用户界面和顾主-服务员系统迅速发展，迫切要求新的开发技术能方便地表示实体并适用于异种结构分布系统。

(4)多变的需求和强烈的竞争要求开发时间短，但质量却要很高，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。

这些困难，用传统的开发方法难以解决，但使用 OOSD 则可以得到缓解，因为：

(1)面向对象的结构是分布的。

(2)系统的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。

(3)从需求阶段到实现阶段，使用相同的概念，开发过程中阶段的改变，不需要方法或风范的转换，可以做到平滑的过渡。

用 OOSD 开发的软件，有更好的可靠性和可重用性，开发的成功率之高是所有开发方法之最。根据美国 1000 家公司的统计[3]，在 12 种开发方法中，OOSD 的成功率居第一，为 91.7%；其次是结构化方法，为 90.2%；CASE 为第 11 位，成功率是 59.7%；最后是 AI/专家系统法，成功率是 55.1%。

蔡希尧　教授，博士生导师，中国电子学会常务理事。陈平　副教授，软件工程所副所长。金益民　教授，软件工程所所长。

用OOP的程序生产率也是最高的。"功能点"(FP，见附录)作为评估软件生产率的准则，已得到普遍承认。(传统的以程序行数来计算生产率，对于抽象程序高的语言来说显然是不公平的。)据IEEE Spectrum(1993年1月)披露，1992年的统计表明，通用的程序设计语言编程的生产率每人月平均为5FPs，而用面向对象程序设计语言(OOPL)编程的生产率则高达15FPs，是平均值的三倍。

OOSD的这些优点，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。

Booch最早系统地使用OOSD方法开发软件[2]，以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[3,4]，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"[5,6]，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，其出发点是对象所封装的状态，某个事件的到来，将触发系统，引起状态的变换，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。

以下讨论OOSD中的两个主要方面：面向对象分析(OOA)和面向对象设计(OOD)技术，讨论以数据驱动和责任驱动两种方法为主展开。

二、面向对象分析

目前使用的OOA，以数据驱动为多。数据驱动法是把抽象数据类型的方法应用于面向对象程序设计，由于类很像抽象数据类型，所以这一应用是直接的。这种分析方法用语义网络模型中的实体、关系、属性等概念，和OOPL中的对象、类结构、封装、继承性等概念结合起来而形成的，信息造型是其基础，用实体-关系图(ERD)描述。信息造型开始用于数据库的设计，描述数据之间或属性之间的关系，ERD则是所用的工具。在加入OOPL的概念以后所形成的OOA方法，比之于ERD有了很大提高，两者的不同之处有：在ERD中，数据没有封装，实体和对它的属性处理不是组合在同一单元之中，没有采用继承性和消息传递来支持模型，而在OOA中，这些方面都得到解决。OOA当然也不同于功能分解或数据流图法，在这些方法中，问题空间的影射是间接的，而在OOA中则是直接的。

在以信息模型为基础的前提下，数据驱动OOA还辅之以其他模型，包括进程模型，状态模型和通信模型。进程模型用内部进程描述每个对象的内部活动，这些内部进程对本对象或系统中的其他对象的属性进行操作。状态模型用状态、事件以及由事件触发所引起的状态转移来描述对象对环境反应的内部机制。通信模型描述系统级的对象之间的交互作用。这四个模型，都和内部的属性和行为有关，信息模型还描述外部关系，通信模型则可以传送对外部的操作。

在OOA阶段，认定对象及彼此间的关系是主要的任务。一种方法是把需求已经明确的问题域用若干个实体和它们之间的关系加以描述，得出ERD，然后把ERD转换成对象-关系图(ORD)。简单的实体就是一个对象；复杂的实体可以分解成几个对象，并且要明确彼此间的关系。从ERD转换成ORD后，数据和作用于它们的操作一起被封装在

一个对象之中，继承则做为对象间的一种关系。

另一种方法是：在需求描述中找出名词、动词和形容词，名词作为对象的候选，动词为方法(即服务，消息的接口也根据动词决定)，而逻辑特性则从形容词导出。

使用责任驱动法时，在OOA阶段认定对象及它们之间的关系，是通过对系统行为的分析而达到的。责任驱动法是按照顾主-服务员模型建立的，顾主和服务员之间的交互作用以契约表，即一个服务员为响应一个顾主的请求所提供的服务来描述，因此，这种方法认为重要的是理解所需要的信息处理特性，并做为一种服务加以对待。当这些服务被理解以后，我们就能够决定哪种类型的实体能够最好地去完成这些服务。所以，找寻对象首先是对系统的行为进行分析，搞清楚系统的行为以后，知道了系统中哪些部分是行为的发动者和参与者，因而可以找到对象。对象提供服务并管理系统中的信息，因此，它们负有某种责任，这是这种方法叫做责任驱动法的由来。

关于行为分析，首先要理解与行为有关的问题，文[7]提出为此要撰写脚本，每一步都采用"发动者-动作-参与者"来表示，并确定参与者的服务；然后考察脚本中的发动者和参与者的属性，建立属性词典。当脚本完成以后，按照它们的前置条件和事后条件进行匹配，把脚本加以连结，给出系统中所进行的动作序列，在此基础上，可以认定动作的发动者和参与者，从而可以找出对象。但是，仅仅是一个发动者，还不能算做一个对象，它必须同时是一个参与者，这是因为一个发动者通常处于系统的作用域之外，或在边界上，它们只是使用作用域内的服务，而参与者则是服务的提供者，是作用域内的对象。

以上三种认定对象的方法，我们认为各有特点，可根据被分析系统的实际情况选择采用。从实体到对象的方法，对于一些复杂的系统来说，容易入手，因为实体容易鉴别，有了实体，经过进一步分析，可以得到对象。从词类识别对象，比较直接，但需要有需求规范(或至少有一个比较明确的文字说明)，在实际工作中，这常常是不容易做到的，只适用于较小的系统。这两种方法都属于数据驱动法，和传统的方法联系较多，易于掌握。但由于方法所认定的对象或类是等同于抽象数据类型的，因此，对象的结构是对象定义的组成部分，因而影响对象的密封性，而结构还可能反映在操作的定义中，这是数据驱动法的缺点。通过行为分析来认定对象，对一些动作明确的系统来说，是方便的，但对动作复杂、参与者众多的系统，用起来并不简单；但是，这种建立在顾主-服务员模型之上的方法，有更好的密封性，因为系统中的顾主只按规范提出请求，而服务员只对请求做出响应，着眼点在于服务员能为顾主做什么，而不是服务员怎样去做，不涉及对象的结构。

在对象及其相互关系明确以后，需要对这些对象及关系、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，得到对象结构。表示的方法可用表格、图形或文字，分析者可自行选择。

对象间的关系除了继承关系以外，主要是服务关系，也叫做契约关系。对象加以组织后的结构主要是层次关系。当两个或更多个对象共有某些服务时，可生成新的对象以提供共有的服务；当两个或多个对象共有某些属性时，可生成一个新的对象以提供共有的属性，这两种做法，在分析工作中叫做"抽象"。有时候，一个对象被赋予多种责任(以提供的服务来表示)时，可把它们分解成多个对象，同一种类的责任由一个独立的对象来承担。这些都是对象认定以后要做的细化工作。

在分析阶段结束时，要给出产品，并进行评估，通过以后，做为下一阶段的输入。产品以工程开发所规定的文档形式给出。以下是一个空中交通管制系统在OOA阶段结束时的产品[8]，包括(1)实体-关系图；(2)对象-关系图；(3)对象描述，是一种表格；(4)对象互

相参照表;(5)继承图;(6)系统行为说明;(7)系统行为图;(8)顾主-服务员图,可资参考。

三、面向对象设计

分析的结果必须影射到设计。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。

设计阶段划分为两个步骤: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:(1)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。由于系统的行为很大程度上体现在对象间的交互作用上,因此给出交互作用明确且完全的定义是这个阶段的重要工作。(2)类的认定。在分析阶段已经把对象组织成一定的结构形式(层次结构),在此基础上对类加以认定,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。在分布处理系统中,由于继承不易实现,常常只使用对象,对类不一定加以定义。(3)对重用的支持。类认定以后,组成类库,用以支持重用。方法要尽可能放在类库的高层次中,这样,能够共享这些方法的子类就越多,重用也就得到更好的支持。在应用时,在类库中选择所需要的类,实例化以后得到对象。

针对某种应用,把若干个对象或类及其关系组成一个子类库,叫做构架(Framework),这是重用的实际基础。在特定的范围内,当应用和某一构架的特征相匹配时,是整个构架而不是其中的个别或部分内容被使用,所以,构架是OOD最理想的设计目标。

在类库中,类是以继承关系联系起来的,但构架中的类除了继承关系外,还有其他关系,这些关系和应用密切相关,这是类库和构架的区别,因此,在OOD中,即要设计类库,还要设计构架,而不是从类库中简单地划分出来一部分就可作为构架。目前,不少公司已在市场上推出类库和构架商品。

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,例如,一个对象中的方法,哪些是公用的,哪些是私有的,内部的处理如何进行,需要哪些系统调用等,都要详细地加以确定。在互相通信的对象中,对象属性的类型要取得一致。由于对象的加细和某些变动,因此相应地将有类的某些变动。

硬件和软件平台要在详细设计阶段予以确定。概要设计和详细设计阶段同样要给出经过评估的产品。有时候,在详细设计阶段结束时还要求给出伪码。

在采用责任驱动法时,在OOD阶段,对象的加细同样是一项主要的工作,此外,需要对责任进行再分配,加细后的责任的形式化,生成层次,以及对象间合作的细致剪裁。类似于构架,在OOD阶段,按照应用的要求,需要把若干个对象组合成一个子系统,在子系统中的对象互相合作以履行某种责任,而这些责任是子系统以外的其他对象需要子系统承担的。这样的子系统显然便于重用。子系统的组合要注意提高内聚性,为子系统以外的对象直接提供服务的对象数目量应减少到最低限度。

责任的再分配和合作的剪裁也是为了增加内聚,降低耦合。一组相关的责任被组合成顾主-服务员的契约形式,构成类,每个类所支持的不同契约数应降到最少;同样,和一个类合作的其他类的数目也应降到最少。类按继承关系组成层次,共享的责任放在尽可能高的层次。此外,可利用多态以增加方法的重

用程度。

四、关于 OOSD 开发的生命周期、标准和计量问题

1. 面向对象分析(OOA)和面向对象设计(OOD)的发展，以及在信息系统开发中越来越多的应用，促使了在新的方法之下展开有关开发生命周期的研究，但现在还没有一致的见解。对 OOSD 的一些明显的优点，认识是一致的，例如从 OOA 到 OOD 的转换，要比传统的开发方法来得平稳和一致；OOA、OOD、以及 OOP，有可能使用同一面向对象语言，为开发和维护带来很大方便。但是，这些优点和问题反映到开发过程上，将会带来什么影响？生命周期会有什么变化？

有的研究者为面向对象的系统开发，提出五个阶段[8]，即分析、系统设计、软件设计、实现、测试，和传统的生命周期基本一致，只是每个阶段的实际内容和所用方法有所不同。有的研究者认为面向对象方法论能否在当前条件下进入商品环境，在于是否在分析、设计、实现的过程中，全部采用面向对象技术，还是传统的功能分解法和面向对象的方法混合使用[8]。全过程都用面向对象方法，是研究 OOSD 的最终目的，但目前不少大的工程项目，考虑到技术的熟练和可靠，仍然采用自顶向下的功能分解；就设计的辅助工具来说，传统方法也比较丰富和成熟；此外，还有人员的培训问题。基于这些考虑，开发的三大阶段，可以有三种不同的组合：

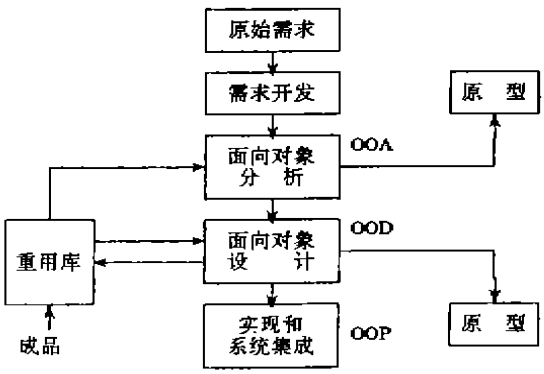
(1)FOO：开始用功能分析，而设计和实现用面向对象方法。

(2)OOF：前两个阶段用面向对象的方法，但用标准的面向过程语言来实现。

(3)OOO：全过程采用面向对象方法。

第一种方法是早期采用面向对象技术的开发方法，结构化方法仍然起很重要的作用；第二种方法已经把分析和设计这两个与系统关系密切的阶段用了面向对象技术，只有实现采用面向过程语言；这两种开发过程都和老的方法关系密切。全部过程采用面向对象方法，可能还要进一步积累经验，完善工具，逐步推广普及，在过渡阶段，混合的方法仍将是有生命力的。

我们在一个工程任务中应用 OOSD 方法的初步经验是：不能割断历史，渐进的方法是比较稳妥的，在使用 OO 方法中，结构化方法的一些方面仍然可以使用。由于工程任务的复杂性，需求往往是不完备的、不确定的、多变的，因此开发人员有责任担负起需求开发的工作，这样，在生命周期中，要加上这一环节。大工程的开发周期长，需要早期反馈分析和设计信息，最好是在不同阶段给出不同的原型，这也应包括在生命周期之内。此外，重用库在开发过程中除了使用以外，实际上存在着充实和改进的问题，这是 OOSD 的一个特点，也是开发过程中的一个环节。这样，OOSD 的生命周期可如下图所示：



在这个图中，主要考虑 OOSD 的三个方面，即 OOA、OOD 和 OOP，所以系统集成以后的测试、验证和确认，以及使用、维护等常规内容，均未画出。

2. OOSD 现在还没有规范和标准，但 ISO 已组织了专家组在制定 OOPL 的标准。

已经使用OOSD的系统，如一些用Ada语言实现的与军用有关的系统，部分地可以用美国国防部颁布的《国防系统软件开发标准DOD-STD-2167A》。按照这个标准，需求分析中的几个方面，可直接使用OOA方法。Ada的新版本Ada-9X即将公布，这个新版本把面向对象语言的主要特征引入Ada，这样，有关OOSD的标准也将会得到加强。

3. 关于软件生产率的计量标准，对OOSD来说，用传统的原码行数(SLOC)来说，显然是不公平的，应采用"功能点"(FP)来计量为妥，越来越多的软件开发项目，现在都转向采用FP为计量标准，我们将在附录中简要地介绍FP的要点。

五、小结

本文是根据我们的理解和初步的经验写成的，OOSD虽然已经显示了它的优势，受到越来越多的重视和采用，但究竟还是发展中的新方法，有待充实、提高、和完善，这一方面需要理论指导，同时需要实践经验的积累。目前，总的说来，面向对象技术的理论落后于实践，从OOPL开始，就缺乏形式描述的工具，而OOSD也没有形成一致的认识、通用的开发过程、和可实用的规范，这些都是今后需要加快研究解决的问题。

附录　功能点的概念

功能点分析法是Allan Albrechet于1979年提出来的。方法分两个步骤：第一步决定信息处理的规模，第二步决定技术复杂度因数，分别说明如下：

第一步：统计所设计系统的五个分量的规模：(1)外部输入的数目。(2)外部输出的数目。(3)逻辑的内部文件数(用户可构思的，不是物理文件)。(4)外部接口文件数(应用所存取的文件，而不是保存的)。(5)外部查询(所支持的在线询问类型)。规模分为三级：低、中、高，取决于被处理的文件数和每个文件所占的域的数目。加权系数如下：

	低	中	高
外部输入	3	4	6
外部输出	4	5	7
逻辑内部文件	7	10	15
外部接口文件	5	7	10
外部查询	3	4	6

未加调整的功能点(UFP)，或称功能计数(FC)，为：

$$UFP=\sum_{i=1}^{5}\sum_{j=1}^{3}W_{ij}*X_{ij}$$

式中，W_{ij}是权系数，X_{ij}是规模的值。

第二步：决定技术规模因数：有14个系统特征参数，在[0,5]区间取值，它们是：

(1)数据通信。　(8)在线更新。
(2)分布式功能。　(9)复杂处理。
(3)性能。　(10)重用率。
(4)使用频繁的配置。　(11)组装难易度。
(5)事务处理率。　(12)操作难易度。
(6)在线数据入口。　(13)位置的数目。
(7)终端用户效率。　(14)对变化的适应程度。

把这些值用C_i表示，用以下公式计算值调整因数VAF：

$$VAF=0.65+0.01\sum_{i=1}^{14}C_i$$

式中，$0\leqslant C_i\leqslant 5$。$C_i$的取值按以下情况决定：

$C_i=0$，不存在或无影响。　$C_i=3$，影响较大。
$C_i=1$，影响不大。　$C_i=4$，影响大。
$C_i=2$，影响中等。　$C_i=5$，自始至终都有影响。

最后得到功能点(FP)为：

$$FP=(UFP)*(VAF)$$

为了更好地推广和完善FP，1986年成立了"国际功能点用户集团(IFPUG)"。目前，不少大公司在软件开发中都使用功能点做为计量标准，并在使用中发现一些需要改进的地方，如FP的计算带有一定的主观性，需要训练有素的工作人员，数据的收集花钱较多等，并已经提出一些改进的简化计算方法。

(参考文献共9篇略)