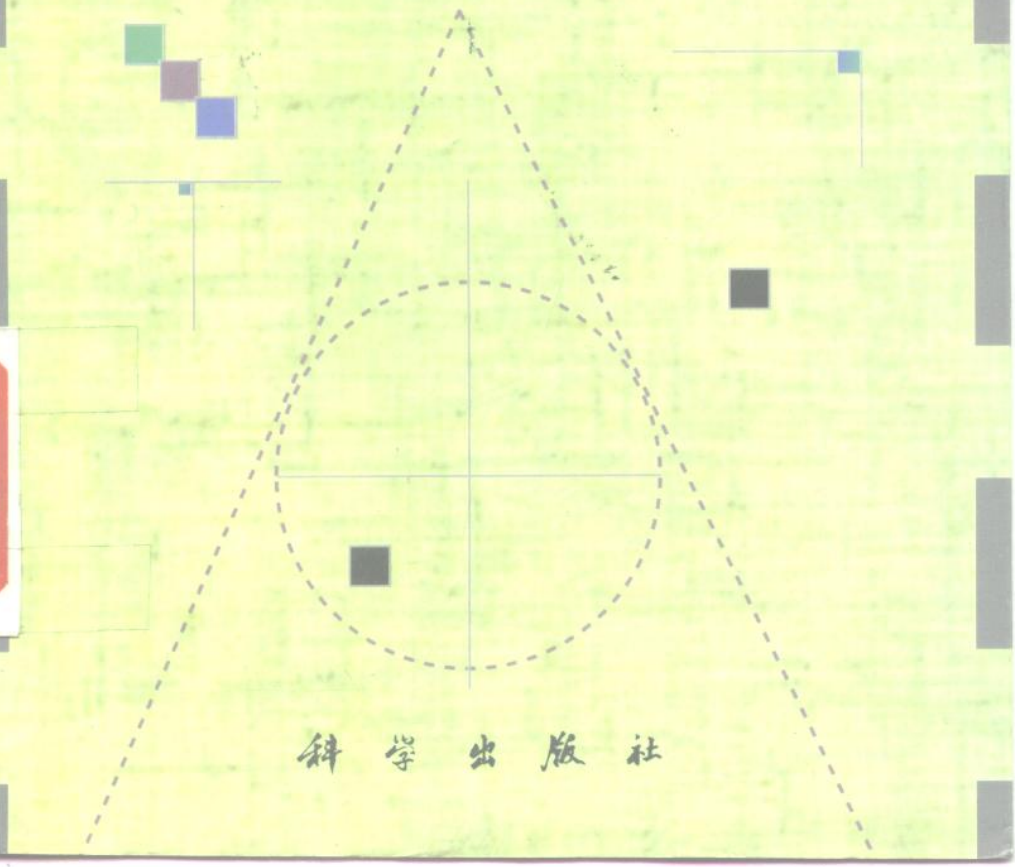
对象技术丛书

# 对象技术导论

冯玉琳 黄涛 倪彬 编著



科学出版社

对象技术丛书

# 对象技术导论

冯玉琳　黄涛　倪彬 编著

科学出版社
1999.2.9 8 8

# 内容简介

面向对象技术正在发展成为当代乃至下一世纪软件发展的主流技术。本书是对象技术的一本入门书，阐述为什么要面向对象，如何面向对象，以及对象技术的现状和未来，使读者通览对象技术的全貌。

全书内容深入浅出，包括对象技术发展的由来、对象技术的基本概念、面向对象的分析和设计、面向对象的程序设计、面向对象的方法论和应用软件开发，以及近几年发展的一些高级对象技术。

本书适于从事计算机软件开发和应用的广大工程技术人员阅读，也可作为大专院校计算机专业高年级学生和研究生的教材或参考书。

**图书在版编目(CIP)数据**

对象技术导论/冯玉琳等编著．—北京：科学出版社，1998.3
(对象技术丛书)
ISBN 7-03-006397-X

Ⅰ．对… Ⅱ．冯… Ⅲ．面向对象程序设计 Ⅳ．TP311.11

中国版本图书馆 CIP 数据核字(97)第 25295 号

科学出版社出版
北京东黄城根北街 16 号
邮政编码：100717

中国科学院印刷厂印刷

新华书店北京发行所发行 各地新华书店经售

*

1998 年 3 月第 一 版 开本：850×1168 1/32
1998 年 3 月第一次印刷 印张： 7
印数：1—3 000 字数：175 000

**定价：12.00 元**

# 《对象技术丛书》前言

国内有关面向对象语言和技术的书籍已经出版不少，为什么还要再出这样一套丛书？道理很简单，这是社会的需要，既是高等学校进行软件工程教学和实践的需要，也是广大从事软件开发和应用的工程技术人员的需要。有人认为，能够用C++编写程序就是掌握了对象技术。这是一种误解。当代软件工程发展正面临着从传统的结构化范型到面向对象范型的转移，这需要有新的语言、新的系统和新的方法学的支持，对象技术就是这种新范型的核心技术。对象技术正在发展成为当代乃至面向下一世纪软件工程发展的主流技术。只有真正深刻理解对象技术的内涵，才能在实践活动中运用自如，进入一个新的境界。本丛书旨在向国内读者全面、深刻地介绍面向对象技术，包括面向对象的语言、方法学、体系结构、技术标准和产品应用等。本丛书部分专题由国内有关对象技术专家撰写，部分专题经IBM授权同意，由该公司的技术资料翻译而成。由于对象技术尚在蓬勃发展阶段，各种对象技术的新系统和新产品不断涌现。本丛书没有预先确定一套完整的书目清单，根据先求实不求全的原则，按照对象技术的发展和国内读者的需要，将陆续选定出版的专题。丛书编委会负责指导和组织专题的编写和翻译工作。

希望这套丛书能对我国从事对象技术教学和研究开发的科技人员有所帮助，并能为我国对象技术的应用和软件产业的进步起到推动作用。一本书、一套书的作用总是很有限的，但是它所激发的社会响应却可能大得多。这正是我们所追求的。

编　者

1997年12月

# 前　言

“面向对象”是一种风范，是观察和分析问题的一种方法论。基于这样的方法论，人们可以用自然的方式认识和模拟现实世界，并由此带来软件制造方式的根本变化。对象技术的发展是近 20 年的事。80 年代的研究工作表明，对象技术对于小规模的程序设计环境是很成功的。到 90 年代，各种基于对象技术的语言和工具系统就如雨后春笋般地涌现，对象技术进入工程实用阶段，成功地开发出许许多多的大规模应用。对象技术已发展成为应用软件开发的基本核心技术。特别是 Internet/Intranet 的发展和 Java 语言的出现，更使得对象技术的应用成为人们关注的焦点。为此，人们常常会提出如下的一些问题：

- 为什么要面向对象？
- 对象技术的最基本概念是什么？
- 如何进行面向对象的分析和设计？
- 如何选择面向对象程序设计语言？
- 与传统软件工程比较，面向对象方法有什么优点？
- 现在有哪些对象技术标准和产品？如何选用？
- 如何进行面向对象的软件系统集成？

为了回答这些问题，本书全面系统地介绍对象技术的概念、内容和发展现状，深入浅出地将对象技术的真谛呈现给读者。本书试图赋予读者足够宽阔的视野，能将这个领域中的基本概念和技术奥秘一览无余。为了进一步深刻理解本书各章内容，建议读者学习时能适当参阅书后所列的参考文献。本书适用于从事计算机软件开发和应用的广大工程技术人员阅读，也可作为大专院校计算机专业高年级学生和研究生的教材或参考书。

全书共分七章。

第1，2章讲解什么是面向对象技术？为什么需要面向对象技术？并从自然的角度出发，运用通俗浅显的例子讲解常用对象技术的基本概念。

第3章概述面向对象分析和设计的基本原则，并介绍若干经典的分析和设计方法。第4章介绍几种主要的面向对象的程序设计语言，阐述面向对象程序设计的一些基本原则和高级技术。第5章是面向对象数据库的一般介绍，讲述面向对象数据库的基本特征。

第6，7章介绍面向对象的高级技术和应用。第6章讲解分布对象计算的一些基本概念，如client/server计算、对象分布、对象总线、框架等；还介绍两种主要的实现分布对象计算的工业标准规范，一种是CORBA对象总线，另一种是COM/OLE对象总线，并对两者作了技术上的比较。第7章讲解基于软件系统结构的集成，软件系统集成是软件工程领域中富于挑战性的一个问题。基于对象总线技术的框架集成是最有发展前途的一种系统集成。

本书第1，2，7章由冯玉琳教授执笔，第3，4章由倪彬博士执笔，第5章由吴胜利博士执笔，第6章由黄涛博士执笔，全书由冯玉琳教授负责修改和定稿。

本书初稿由《对象技术丛书》编委会主编北京大学杨芙清教授、编委会委员北京信息工程学院周锡令教授以及南京大学郑国梁教授审阅，他们提出了许多宝贵意见。本书撰写过程中还得到IBM公司K. S. Ip，Daniel Tkach，Walter Fang，Jack Z. Zhong和Andrew So以及中国科学院软件研究所对象技术中心的许多同志的支持和帮助，在此一并表示深切谢意。

由于对象技术领域的概念和术语很不统一，加之作者阅历有限，书中难免有疏漏谬误之处，敬请读者不吝指教。

作　者

1997年8月于北京

# 目　录

# 第一章　为什么面向对象

面向对象（Object Orientation），简称“OO”，已成为软件工程界广泛使用的词汇。人们经常会讲到面向对象的程序设计语言、面向对象的设计方法学、面向对象数据库、面向对象的用户界面以及面向对象的业务模型等等，似乎面向对象已成为软件工程技术中的时髦和先进的用语。为什么面向对象技术能在如此众多的领域有如此广泛的应用呢？从本质上说来，面向对象是确定动作的主体在“先”，而执行动作在“后”。例如，对于面向对象的用户界面，总是先选定一个界面对象，例如一段正文或一个图标，然后在这个对象上进行操作；对于面向对象的程序设计语言，先指定接受消息的对象，然后才在对象上执行消息指定的操作；对于面向对象的分析和设计，也是先确定系统中的实体对象，然后再确定在这些对象上可能实施的操作。如此的面向对象模式可简单称之为主体-动作模式，先主体，后动作，而不是像传统的面向过程的程序设计语言那样，最关心的是过程，而过程实施的对象是作为过程参数传递的。

面向对象的这种主体-动作模式是与人们对客观世界的认识规律相符合的，从而使得软件工程中如此广泛的领域在对象技术上建立了共同的基础。面向对象是一种风范（Paradigm），是一种观察和分析问题的方法论（Methodology）。对象技术是一种软件系统组织和结构设计的工程技术，它将对象作为软件系统结构的基本组成单元，以主体数据为中心，将数据及其上作用的操作一起封装，以标准的接口规范对外提供服务。

软件工程技术发展的主要目标是提高软件质量和软件生产效率。从汇编语言到高级语言，标志着软件工程技术和软件生产率的一次质的飞跃，促成这次飞跃的决定因素是编译理论和技术的

完善，实现了从高级源码到机器代码变换的自动化。从70年代至今，传统的结构化软件设计的研究，在软件结构和设计方法学上提出了一些基本的原则和方法，例如模块封装、数据抽象、E-R模型、数据流方法等，促进了软件工程技术的发展。但软件工程迫切需要解决的问题，如生产效率低下、软件可扩充和复用能力差、难以维护等，并未从根本上获得解决。

可以将计算机硬件制造和软件制造作一对比。随着微电子技术的突飞猛进，硬件制造已经是高度工业化和集成化。计算机硬件是由若干预先制造的标准组件和部件经过工业生产线组装而成。对这些零部件，我们事先并不知道它们的最终使用地点和具体用途，而是预先按照一定的标准界面接口来精心设计并成批生产入库，一旦整机系统制造需要时就直接投入使用，而不必临时针对每个具体的应用情况一切从头设计和生产。预制的零部件资源愈丰富，硬件制造的水平和生产效率就愈高。这是一种资源集约生产方式。相比之下，软件制造的情况远非如此，软件生产目前基本上仍然是人工集约生产方式，主要依赖于软件人员的知识和能力。软件应用系统的需求各种各样，世界上成千上万的软件工程师花费了大量人力物力编写了各种各样的软件，可是随着应用需求的扩大和变化，按传统方式编写的这些软件很难再利用，软件生产的方式和效率远远赶不上信息化社会发展的需要，世界上对软件人员的需求一直处于紧张状态。90年代开始的对象技术的发展和工程化，为使软件生产从人工集约型向资源集约型发展带来了希望。基于对象技术，人们开始努力研究开发像硬件零部件那样的一些标准软组件、软部件，应用系统的开发也可像硬件组装一样，可以按照一定的规范，组装这些预制好的软件零部件，从而使软件制造进入一个全新的境界。当然，软件模块组装较之硬件模块组装要更复杂，也更难实现。软件是对数据、计算、接口等复杂思维形态的表达，软件模块与应用环境关系密不可分。对象技术对软件模块组件化的设计和复用带来了工程现实的可能性。

## §1.1 模 块 化

在计算机软件中，模块化的概念已经采用了20多年，软件被划分成若干可单独命名和编址的部分，它们被称作模块，这些模块相互连接组成满足应用需求的软件系统。

模块化是软件对付复杂问题所应具备的关键特性，也是使得软件能够被有效地管理维护所应具备的特性。

模块的基本特征是抽象和实现信息隐藏。模块分模块界面和模块体两部分。模块界面是模块对外的“窗口”，模块对外的联系和相互作用只能通过这个“窗口”可以见到的模块属性和操作进行。模块体是模块的具体实现细节，对外是不可见的。抽象和信息隐藏从两个不同侧面说明了模块化设计的基本特征，使得一个复杂的软件系统可以通过定义一组相对独立的模块来实现，这些独立的模块彼此之间仅仅交换那些为了完成系统功能所必须交换的信息。

对象技术按照主体-动作方式工作，对象模块是以主体数据为中心进行设计，而以数据上的操作作为界面。由于数据结构通常表示软件系统中相对稳定的属性特征，围绕数据结构而设计的模块界面接口简单，模块独立性好。当模块内部实现发生变化而代码修改时，只要对外接口操作的功能不变，就不会给软件系统带来影响。

模块化设计的准则有三：

1. 可分解性

可分解性是对付大规模复杂软件系统的最重要特性。一个复杂的应用系统按照一定的方法论可以分解为若干不太复杂的子系统，子系统可再分为若干更简单的部分。如此过程继续下去。这些自顶而下的分解都是在子系统或模块的抽象层次上进行的，直至分解到最后的足够简单的模块。可以将它们分配给不同的开发人员去实现。这种可分解性如图1.1所示。

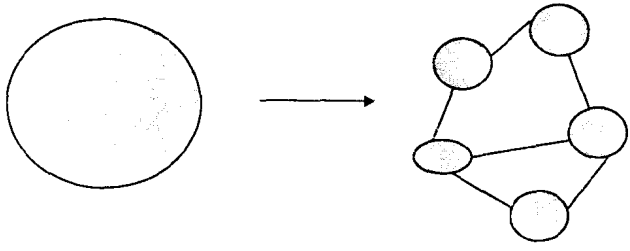


图 1.1　模块分解

2. 可组合性

可分解性是指软件系统从应用需求出发的模块分解，而可组合性正好相反，是指将已有的模块单元组装而成满足应用需求的系统。如图 1.2 所示。

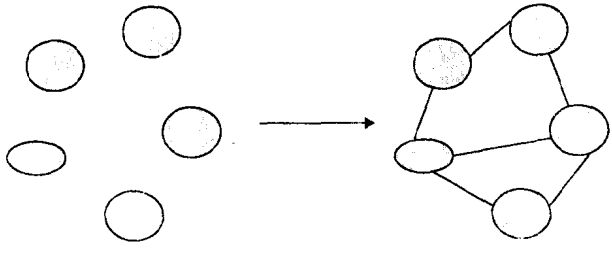


图 1.2　模块组合

显然这种可组合性是与软件复用概念紧密结合在一起的。这些模块单元应该符合软件复用的要求，整个软件构造组合的过程是软件理解和复用的过程。

3. 可理解性

模块的可理解性不仅对于软件系统设计而且对于软件维护都是非常重要的，一个不可理解的软件是无法使用的。

传统的结构化设计的方法很难满足模块化的以上各项要求。与面向过程软件系统结构相比较，面向对象软件系统结构已发生质的变化，对象模块具有完整的语义特征，易于理解；具有较高的通用性和可适应性，易于扩充和修改；具有规范的界面接口，易

于构造组装。面向对象技术的一大贡献在于改变了传统软件系统的模块化结构特性。而这种结构特性与软件工程技术发展的目标要求是正相适应的。

## §1.2 软件复用

软件复用并不是现在才为人们所认识的，A. Turing 在 40 年代就发明了子程序（Subroutine）的概念。当我们每天在计算机上工作的时候，操作系统和数据库本身就是使用最频繁的软件复用的例子。几十年来，应用系统成千上万，许多应用程序包，如 Ada 的 Package 软件，也已经获得广泛的再应用。应该说，软件复用是软件工程技术发展所必不可少的，而且已经被广泛使用。

但是，对象技术使软件复用更臻于完善和规范，对象封装和继承可以很好地支持软件复用。对象封装允许应用开发者将对象模块视作黑匣子，通过界面去理解和操作对象，而不去关心实现细节；对象继承容许对象实现复用具有相同特性的其它对象的代码，而不要去重复开发。基于对象的统一的语义模型，对象技术可提供统一的机制（如对象总线）将对象模块组装在一起，极大地复用已有的对象，满足各种应用需求。除了各种可复用的公共对象模块外，信息化社会的进步还要求人们开发满足各种应用需求的领域对象。随着对象技术的发展和各种领域对象的出现，今后的应用软件开发将面临新的局面。软件人员结构亦将发生新的变化。软件开发人员将分化为两部分，一部分是软组件/部件制作人员，另一部分是应用系统分析和装配人员。应用开发人员结构的演变如图 1.3 所示。

软件复用可以发生在软件开发的不同阶段，可以在设计级，也可以在源码或二进制代码级。为实现软件复用，除技术因素以外，还需要解决许多非技术的因素，如心理的和法律的因素。

在应用开发中，为了找到可能复用的软件对象，需要有非常好的浏览机制。同时，需要一种自然的方式描述软件对象的语义

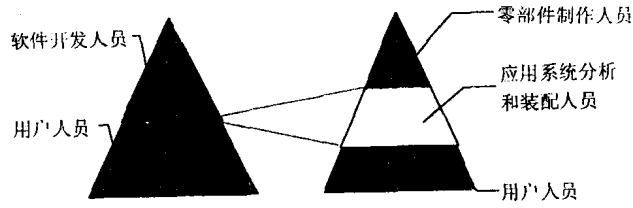



图 1.3 软件人员结构演变

约束，这种语义约束必须是足够明确到不致会误用该对象。

一个可复用的软件对象要满足功能、性能、复杂性等多种要求。这些要求在设计阶段就要充分考虑到，在设计阶段不考虑可复用性而生产出来的软件对象组件或部件是很难被再应用的。软件对象组件应尽可能通用和标准化。为了使对象组件组装在一起能够协同工作，我们需要一种类似硬件总线式的公共软件总线。对于各种共享复用的对象组件，采用延迟绑定的动态连接方式运行，这样可使不同应用软件能方便地复用那些已有的公共对象组件资源。

当然，应该指出，为了实现软件的可复用性，软件开发人员在开发这些可复用组件时就必须付出许多额外的努力，而在应用这些可复用组件时，又要花销额外的代价去发现和理解这些复用组件，这与传统方式下软件开发人员为特定的任务做特定的设计，显然是不同的。需要有新的知识和经验的积累。对象技术的发展为此提供了一种规范的工作模式，但并不能代替所有的这种努力。

## §1.3 软件维护

独立软件开发商和软件用户，常常抱怨软件维护的工作量和开销实在太大。不管是何种形式的软件维护，总是要求一部分代码要修改，而另一部分代码保持不动，却必须保证软件系统能可靠正确地运行。可以说，对象技术的所有方面，封装、继承、多态性、延迟绑定等，不仅支持了软件复用，而且使软件维护工作