# 面向对象软件分析
# 设计与测试

王 晖 郭燕慧 余安萍 编著

科学出版社
www.sciencep.com

# 面向对象软件分析设计与测试

王 晖 郭燕慧 余安萍 编著

科学出版社

北京

## 内 容 简 介

本书主要介绍了面向对象软件工程的理论和实践方法，包括面向对象的分析与设计技术、软件配置管理技术、面向对象的软件测试与度量技术，以及实践过程中典型工具的使用。

本书描述了当前面向对象软件开发过程的基本理论和实用技术，适合作为大学计算机专业高年级学生和研究生学习软件工程的参考书；同时本书提供的面向对象的软件工程项目开发的方法和指南，对具有一定经验的系统分析员、面向对象程序设计人员、软件配置管理人员、软件测试人员和软件开发管理人员的实践活动也具有指导意义。

**图书在版编目(CIP)数据**

面向对象软件分析 设计与测试/王晖，郭燕慧，余安萍编著.—北京：科学出版社，2004

ISBN 7-03-013249-1

Ⅰ.面… Ⅱ.①王… ②郭… ③余… Ⅲ.①软件开发-系统分析 ②软件-测试 Ⅳ.TP311.5

中国版本图书馆 CIP 数据核字(2004)第 028854 号

策划编辑：王淑兰/责任编辑：丁 波

责任印制：吕春珉/封面设计：三函设计

科学出版社 出版

北京东黄城根北街16号

邮政编码：100717

http://www.sciencep.com

双青印刷厂 印刷

科学出版社发行 各地新华书店经销

*

2004 年 5 月第 一 版 开本：787×960 1/16

2004 年 5 月第一次印刷 印张：21 1/2

印数：1—3 500 字数：419 000

**定价：28.00 元**

（如有印装质量问题，我社负责调换〈环伟〉）

# 前　言

在20世纪90年代，面向对象的软件开发技术取得了巨大的发展，大量的软件开发实践中都采用了面向对象的程序设计语言，如C++，Java等，使得软件开发效率和软件产品质量有了显著的提高。到90年代中期，面向对象技术已逐步应用到软件生命周期的各个阶段，形成了一整套从分析、设计到测试和度量的面向对象技术。与此同时，一些软件公司也相继开发出基于面向对象技术的软件工程工具，其中，Rational公司的系列产品由于功能强大、使用方便而得到了广泛应用。Rational的系列产品包括面向对象分析和设计工具Rose、自动化文档生成工具SoDA、软件配置管理工具ClearCase、测试集成管理工具Team Test等。

本书结合Rational公司面向对象系列软件工程工具的具体应用，介绍如何使用面向对象技术来进行软件的需求分析、设计、配置管理以及测试和度量。

本书共分7章，各章的主要内容如下：

第1章主要介绍了面向对象技术的基础知识。以典型C++语言为例，介绍对象、类、消息等面向对象的基本概念，以及封装性、继承性、多态性等面向对象系统的特性。对面向对象技术不了解的初学者通过阅读本章，可以掌握面向对象基本的概念，为后面章节的阅读打下基础。

第2章主要讨论了面向对象的分析与设计技术。从传统系统分析与设计方法入手，介绍数据流分析、E-R模型、结构化的设计、Jacobson方法，并分析了传统分析设计方法存在的不足；对典型的面向对象分析设计方法Booch方法、Coad/Yourdon方法、OMT方法和Jacobson方法进行了介绍，并详细描述了面向对象分析与设计的流程，包括业务过程需求分析、系统对象分析、系统对象设计等。

第3章主要介绍了统一建模语言(UML)。包括用况视图、逻辑视图、组件视图、并发视图和部署视图这5种视图的概念及使用；描述了用况图、类图、对象图、顺序图、协作图、状态图、活动图、组件图和部署图的内容及表示；简要地介绍了UML的通用机制和扩展机制等。

第4章描述了面向对象的分析和设计具体实践，介绍了如何使用Rose实现面向对象的分析和设计，包括如何创建各种视图和图，正向生成代码和从代码逆向生成模型图。本章还介绍了如何使用SoDA输出分析和设计的文档；如何使用文档生成命令建立SoDA模板，并生成符合用户需求格式的文档。

第5章主要介绍了软件配置管理技术，包括软件配置管理的基本概念、配置管理过程及配置管理的最佳实践，并给出了利用ClearCase进行软件配置管理的具体操作方法。

第6章描述了面向对象软件测试技术。首先介绍软件测试的基本概念，各种白盒测试和黑盒测试方法，单元测试、集成测试、系统测试、验证和确认测试、回归测试等不同类型的测试，描述了针对确认测试、系统测试和验收测试的软件测试过程，并给出了典型工具PurifyPlus(白盒测试)、TestManager(测试过程管理)、Robot(黑盒测试)、SiteCheck(Web测试)的具体应用。

第7章讨论了面向对象软件度量原理与方法，介绍软件规模度量和复杂性度量的基本概念和基本原理、面向对象软件度量的特点及方法、软件质量度量模型、质量特性及软件质量度量过程。通过对软件的规模度量和复杂性度量，软件工程人员可以了解当前开发的软件规模，估算软件开发进度、合理配置开发资源，控制软件开发过程。通过对软件的质量进行度量，软件工程人员可以了解产品质量特性，对产品做出正确的评价。

本书第1章和第2章由郭燕慧编写，第3章和第4.1节由余安萍编写，第4.2节和第5章～第7章由王晖编写，全书由高振平负责审校。

由于作者学识有限，加之该领域技术不断发展，书中难免存在不妥之处，敬请读者批评指正。

作　者

2004年2月

# 目　录

# 1

# 面向对象技术基础

- 面向对象思想的由来
- 面向对象的基本概念
- 面向对象系统的特性
- 小结

在计算机应用领域中，软件开发的步履维艰与硬件技术发展的日新月异形成了鲜明的对比。软件是一种抽象的逻辑思维的产品，所以软件的研制过程实质上是软件研究人员的“思考”过程。有人说软件就像泥巴一样，你愿意捏成什么样就是什么样。正由于其“容易”修改，因而没有一个软件在投入运行之后就不再进行修改扩充的。软件人员可以按各自的爱好进行工作，没有统一的标准可循，管理人员事前又难以精确地估计项目所需的经费、时间，技术人员在项目完成之前也难以预料系统是否成功；更糟的是，系统失败后又往往无法挽回，除非再从头做起，但由于时间和经费的限制这又是不可能的。所以研制过大型软件系统的工程师们回首往事时，经常把自己所完成的一些工程称为“遗憾工程”。

国外在研制一些大型软件系统时，遇到了许多困难，最著名的例子是 IBM 公司的 OS/360系统和美国空军某后勤系统。这两个系统都花费了数千人几年的努力，历尽艰辛，但结果却令人失望。OS/360 系统负责人 Brooks 生动地描述了研制过程中的困难和混乱：“……像巨兽在泥潭中做垂死地挣扎，挣扎得越猛，就陷得越深，最后没有一个野兽能逃脱被淹没的命运……程序设计就像这样一个泥潭……一批批程序员在泥潭中挣扎……没有料到问题竟会这样的棘手……”，这就是所谓的“软件危机”现象。

造成这种危机的根本原因是我们目前采用的冯·诺依曼机类型的计算机求解问题的方法空间结构与人们认识问题的问题空间结构很不一致。冯·诺依曼机的基本特征是顺序地执行指令，按地址访问线性的存储空间。在这种机器上所能直接接受的是面向过程的语言，这是一种比较难以理解的、与人们的自然语言相距甚远的语言。几十年来，人们为了克服软件危机、控制软件的开发质量和提高软件的生产效率，对软件开发的方法进行了大量深入的研究，提出了用管理工程项目的方法来组织软件项目的开发。到目前为止，最典型的方法是结构化的需求分析、结构化的系统设计方法。这种方法从本质上看，仍具有冯·诺依曼机系统结构的特点，是软件开发人员从开发软件的立场出发而确定的，并不是从人们认识客观世界的过程和方法出发的。为了部分地、有限地缓解软件危机、缩小语言鸿沟，人们提出了面向对象的技术和方法。

事实上，面向对象的系统以及相关的思想几乎与人类的历史一样悠久。人类历史上的所有学科几乎不约而同地采用抽象作为控制问题领域复杂度的一种手段。数千年来，人类一直在用区别物体及其属性、区别整体及其部分、区别不同类型的物体这些基本的方法来认识与把握整个世界。在计算机界，20 世纪六七十年代从事小规模程序设计与人工智能研究的先驱者，几乎各自独立地确立了今天计算机领域中的面向对象概念，这说明面向对象概念有着广泛的思想与应用基础。现在面向对象技术的研究几乎遍及计算机软硬件的各个领域，如程序设计语言、软件开发方法学、操作系统、数据库、软件开发环境、硬件支持等。

面向对象软件开发方法是指将面向对象技术用于开发软件的全过程，这包括面向对象分析（OOA）、面向对象设计（OOD）及面向对象程序设计（OOP）等一系列过程。当前有许多不同的面向对象开发方法，有形式化方法，也有非形式化方法，所有这些方法都是基于对象的概念。

## 1.1 面向对象思想的由来

“面向对象”技术追求的是软件系统对现实世界的直接模拟，尽量将现实世界中的事物直接映射到软件系统的解空间。它希望用户用最小的力气，最大程度地利用软件系统来解决问题。

现实世界中的事物可分为两大部分，即物质和意识，物质表达的是具体的事物，意识描述的是某一个抽象的概念，例如，“自行车”和“这辆白色的自行车”，“这辆白色的自行车”是物质，它是具体的客观存在；“自行车”是意识，它是一个抽象的概念，是对客观存在的事物的一种概括。这些现实世界中的事物可直接映射到面向对象系统的解空间，现实世界中的物质可对应于面向对象系统中的“对象”，现实世界中的意识可对应面向对象系统中的抽象概念——类。自行车在面向对象系统中可用自行车类来表达，一辆白色的自行车在面向对象系统中是一个具体的对象，是自行车类的一个实例，如图 1.1 所示。

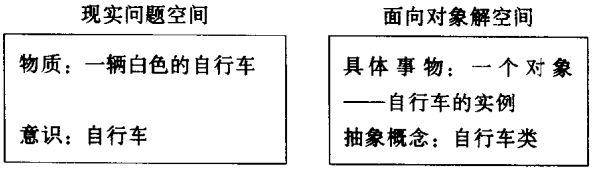


图 1.1 现实世界与面向对象系统之间的对应关系

为了理解面向对象的理论，必须从最基本的概念入手，通过对最基本的概念的掌握，来真正认识面向对象方法的作用。当前C++ 语言已成为工业界普遍接受的面向对象程序设计语言。本章我们将以C++ 语言为例介绍面向对象理论中的几个最基本的概念。

## 1.2 面向对象的基本概念

### 1.2.1 对象

对象是面向对象系统中的核心概念，如果不能正确地认识和定义它，就无法掌握

面向对象的理沦。

1．对象的定义

首先需要搞清楚的第一个问题是，什么是对象？对象具有两方面的含义，即在现实世界中的含义和在计算机世界中的含义。

在我们所生活的现实世界中，“对象”无处不在。在我们身边存在的一切事物都是对象，例如，一粒米、一本书、一个人、一所学校，这些都是对象。除去这些可以触及的事物是对象之外，还有一些无法整体触及的抽象事件，例如，一次演出、一场球赛、一次借书，也都是对象。

一个对象既可以非常简单，又可以非常复杂，复杂的对象往往可以是由若干个简单对象组合而成的。

所有的这些对象，除去都是现实世界中所存在的事物之外，它们都还具有各自不同的特征，例如，一粒米，首先，它是一粒米这样一个客观存在；其次，它还具有颜色、体积、重量；再次，它还可以被食用，可以用来作为原料等。也就是说一粒米是一个具有自身状态和自身功能的客观存在。再例如一个人，首先，他是一个客观实体，具有一个名字标识；其次，他具有性别、年龄、身高、体重等这些体现他自身状态的特征；再次，他还具有一些技能，例如，会说英语、会修电器等。

通过上面的这些举例我们可以对“对象”下一个定义，它具有如下特性。

① 有一个名字以区别于其他对象。

② 有一个状态用来描述它的某些特征。

③ 有一组操作，每一个操作决定对象的一种功能或行为。

④ 对象的操作可分为两类：一类是自身所承受的操作，一类是施加于其他对象的操作。

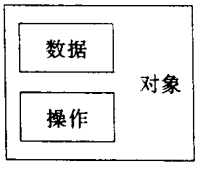


图 1.2　对象的定义

对象是由描述其自身状态特征的数据及可以对这些数据施加的操作结合在一起所构成的独立实体，如图 1.2 所示。

例如，有一个人名叫王东，性别男，身高 1.80m，体重 68kg，可以修电器，可以教计算机课，下面我们来描述这个对象。

对象名：王东

对象的状态：

性别：男

身高：1.80m

体重：68kg

对象的功能（可做的操作）：

回答身高
回答体重 } 均属于自身所承受的操作
回答性别

修理电器
教计算机课 } 属于施加于其他对象的操作

在计算机世界中可以把对象看成是存储器中一个可标识的区域，它能保存固定数目的数值（或数值的集合）。

2．对象的状态

一个对象之所以能在现实世界中独立存在，是因为它具有自身的状态——即自身所具有的那些特征。由于这些状态的存在，使其能对自身及对外界对象施加操作。当然，一个对象的这些状态并不是完全用来直接为外界服务的，但它们是对象为外界提供服务的基础。

例如，一个人的五脏六腑是他的内部状态，作为人这个对象还具有一些如身高、体重等外部状态。所有这些状态中有一些并不是用来直接向外界提供服务的，如那些内部状态，但它们体现了人的身体状况，若没有好的身体，人是不能很好地为外界服务的，因此，这些内部状态是基础。

在面向对象系统中，一个对象的状态是通过“域”来描述的，也称为私有存储单元，这些私有存储单元只能由它自己的操作来进行处理。在C++中用成员变量来存放一个对象的状态。

3．对象的特性

对象的特性主要体现在以下几方面。

（1）数据的封装性

封装的概念与将集成电路芯片用陶瓷封起来颇为相似，芯片的内部电路是不可见的，使用者只需了解引脚的电气参数与提供的功能，就可很好地使用该芯片，而将不同芯片的引脚接在一起，就可以方便地组装成某一产品。封装和数据隐藏与芯片有着异曲同工之妙。

（2）以数据为中心

一个对象包括两个要素：一是数据，二是所要进行的操作。其中，操作总是围绕着数据来进行的，是根据对该对象的数据所做的处理来设置的，不设置与其数据无关的操作，并且操作的结果往往与当时的数据状态有关。

(3) 对象是处理的主体

与传统的数据不同，对象本身是进行处理的主体，由于对象的封装性，不能从其外部直接加工它的私有数据，而是要通过它的公有接口向对象发送消息，请求它执行其某个操作，处理它的私有数据。

(4) 易维护性

由于对象的功能被封装、隐藏在对象内部，所以修改、完善功能及其实现的细节都被局限于该对象的内部，不会波及到外部，这就使得对象和整个系统变得非常容易维护。

### 1.2.2 类

在日常生活中，我们经常听到“类”这个术语，它是对一组客观对象的抽象，它将该组对象所具有的共同特征（包括结构特征和行为特征）集中起来，以说明该组对象的能力和性质，例如，“人类”这个词就抽象了所有人的共同之处。

在面向对象方法学中，把一组具有相同数据结构和相同操作的对象的集合称为类。类代表了某一批对象的共性和特性，类还可以描述如何创建这个类的新对象。在一个类中，每个对象都是类的实例，它们可以使用类中提供的函数，而类本身又是这些对象的抽象，例如，在C++语言中，可以首先声明一个“类”类型，然后用它定义若干个同类型的变量，每个变量就是一个对象。类是一种用来定义对象的抽象数据类型，它可以被视为是产生对象的模板。

为了产生一种对象，应该提供建立实例的操作，例如，C++语言中，利用构造函数在声明一个对象时建立实例，另外，当某些对象不再使用了，应把它们的存储空间释放，以备其他的对象使用，即撤销实例。如C++语言的析构函数给出了一个 delete 操作，就可以用来释放一个对象所用的空间。

1. *类与实例的关系*

类的概念在前面已经给出，组成类的对象均为此类的实例。

类与实例之间的关系可以看成是抽象与具体的关系，例如，我们看到一个苹果，就会说：“这是一个苹果”，把它变成我们面向对象系统的语言来叙述：“这是一个苹果类的实例”，再例如，王东是一个学生，学生是一个类，而王东作为一个具体的对象，是学生类的一个实例。

类是多个实例的综合抽象，而实例又是类的个体实物。

对于同一类的不同实例，它们具有如下特点：

① 相同的操作集合。

② 相同的属性集合。

③ 不同的对象名。

2. 类的确定与描述

类的确定是采用归纳的方法，对所遇到的对象进行分析，归纳出共同特征来确定一个类，例如，我们可以通过对许多马的归纳总结，抽象出各种马的共同特征，从而得到马类。

在实际工作中，单纯地描述一个独立的对象如何工作是没有任何意义的，一般是描述一个类的工作。在C++程序中定义类非常方便，下面是用C++语言对马类的描述。

```
class horse {
            int color;
            char *sort;      } 结构特征
public:
        void move ();
        void shift ();       } 行为特性

};
```

### 1.2.3 消息

客观世界中各式各样的对象，并不是孤立存在的，它们之间是有联系的。不同的对象间通过发送消息来传递信息完成通信。正是对象间的这种相互作用、相互联系，才构成了世间各种不同的系统。

1. 消息的定义

什么是消息，我们通过一个具体的例子来叙述，例如，我们在前面举的例子，一个对象为人，名字叫王东，他是一个电器工程师，他可以修理电器，也可以讲计算机课。除去这些向他人提供的服务外，他也要接收其他对象的服务，如吃饭、穿衣、娱乐等。他要吃饭但不可能自己去种地，他要穿衣但不可能自己去织布，他就要请求他人来帮助解决这些问题，这里的“请求”便是一个人与其他人进行交互的手段。同样，他什么时候修电器，什么时候讲课，需要在得到其他对象的请求后才进行。因此，在面向对象技术中这个“请求”本身就是发送的“消息”。在日常生活中除去“请求”以外，还有“命令”，如上级对下级的命令等，这些“命令”也是一种“消息”。

因此，消息是对象之间相互请求或相互协作的途径，是要求某个对象执行其中某

个功能操作的规格说明。

通常，我们把发送消息的对象称为发送者，接收消息的对象称为接收者。对象间的联系只能通过传送消息来进行。对象也只有在收到消息时，才被激活，被激活后的对象代码将“知道”如何去操作它的私有数据，去完成所发送/接收的消息要求的功能。

消息具有如下3个性质。

① 同一对象可接收不同形式的多个消息，产生不同的响应。

② 相同形式的消息可以送给不同对象，所做出的响应可以是截然不同的。

③ 消息的发送可以不考虑具体的接收者，对象可以响应消息，也可以对消息不予理会，对消息的响应并不是必须的。

2．公有消息和私有消息

在面向对象系统中，消息分为两类，即公有消息和私有消息。

到底哪些消息是公有消息，哪些是私有消息呢？这需要有一个明确的规定。

若有一批消息同属于一个对象，其中有一部分是由外界对象直接向它发送的，称之为公有消息；还有一部分则是它自己向本身发送的，这些消息是不对外开放的，外界不必了解它，称之为私有消息。

外界对象向此对象发送消息时只能发送公有消息，而不能发送私有消息，私有消息是由其自身发送的。

例如，在C++中有一个对象类的定义如下。

```
class person {
        private:
                char      name [20];
                int       age;
                char      add [40];
                char      sex [10];
                void      printName ( ) {...};
                void      printage ( ) {...};
                void      printadd ( ) {...};
                void      printsex ( ) {...};
                public:
                void      print ( ) {
                          printName ( );
                          printage ( );
                          printadd ( );
```