“十一五”国家重点图书　计算机科学与技术学科前沿丛书
计算机科学与技术学科研究生系列教材(中文版)

面向对象的系统设计

(第2版)

邵维忠　杨芙清　著

清华大学出版社
北　京

内 容 简 介

本书是一本论述面向对象设计方法的专著，其第 1 版于 2003 年由清华大学出版社出版，被国内许多大学用作研究生或高年级本科生教材，并被许多软件开发单位作为工程技术用书。本次再版，根据国内外面向对象领域理论与技术的最新发展做了不少修改。

本书是作者的另一本著作《面向对象的系统分析》(第 2 版)的姊妹篇，二者构成完整的 OOA&D 方法体系。

本书的主要内容是论述如何在面向对象的分析(OOA)基础上进行面向对象的设计(OOD)。全书分为 7 章，第 1 章介绍 OOD 的发展历史、现状和几种典型的 OOA&D 方法，论述 OOA 和 OOD 的关系。第 2 章介绍本书提出的 OOD 方法概貌。第 3～6 章分别介绍 OOD 模型各个组成部分的设计方法。第 7 章介绍统一建模语言(UML)，并分析和讨论其优点与缺点。

读者对象：计算机软件专业的教师、研究生和本科生，软件技术培训教师与学员，计算机软件领域的研究人员和工程技术人员。

关键词：软件工程，面向对象的系统设计，UML

图书在版编目(CIP)数据

面向对象的系统设计/邵维忠，杨芙清著. —2 版. —北京：清华大学出版社，2007.5
(计算机科学与技术学科研究生系列教材)
ISBN 978-7-302-14798-5

Ⅰ. 面… Ⅱ. ①邵… ②杨… Ⅲ. 面向对象语言－程序设计－研究生－教材 Ⅳ. TP312

中国版本图书馆 CIP 数据核字(2007)第 029513 号

责任编辑：焦　虹　顾　冰
责任校对：时翠兰
责任印制：李红英

出版发行：清华大学出版社　　**地　　址**：北京清华大学学研大厦 A 座
http://www.tup.com.cn　　**邮　　编**：100084
c-service@tup.tsinghua.edu.cn
社 总 机：010-62770175　　**邮购热线**：010-62786544
投稿咨询：010-62772015　　**客户服务**：010-62776969

印 刷 者：北京市昌平环球印刷厂
装 订 者：三河市李旗庄少明装订厂
经　　销：全国新华书店
开　　本：185×260　**印　张**：14.25　　**字　数**：340 千字
版　　次：2007 年 5 月第 2 版　　**印　次**：2007 年 5 月第 1 次印刷
印　　数：1～4000
定　　价：29.00 元

“十一五”国家重点图书　计算机科学与技术学科前沿丛书
计算机科学与技术学科研究生系列教材

本书责任编委：钱乐秋

Abstract

This is a book on object-oriented design (OOD). Its first edition published in 2003 by our press had been widely used as a textbook for graduate or undergraduate students and as a technology handbook for software engineer in Chinese universities and software companies. In this edition, a lot of improvement have been made according as the development of the object-oriented technologies. This book is a companion volume to *Object-Oriented Systems analysis*, which is another composition of the same authors published by our press in 2006, and together they constitute an integrated method of Object-Oriented Analysis and Design.

There are seven chapters in this book. The first chapter makes a brief introduction to the OOD, and discusses the relationship between OOA and OOD. The second chapter summarizes the main characteristics of the OOD method suggested in this book. Chapter 3 to 6 expatiates upon the whole process of OOD, presents the modeling activities for every major components with detailed process guide and engineering strategies. Chapter 7 makes an introduction and comment to UML.

Audience: teachers, graduates and undergraduates majoring in computer software; teachers and students in software technology training schools; researcher and engineers in the field of computer software.

Keywords: software engineering, object-oriented design, UML

序

未来的社会是信息化的社会，计算机科学与技术在其中占据了最重要的地位，这对高素质创新型计算机人才的培养提出了迫切的要求。计算机科学与技术已经成为一门基础技术学科，理论性和技术性都很强。与传统的数学、物理和化学等基础学科相比，该学科的教育工作者既要培养学科理论研究和基本系统的开发人才，还要培养应用系统开发人才，甚至是应用人才。从层次上来讲，则需要培养系统的设计、实现、使用与维护等各个层次的人才。这就要求我国的计算机教育按照定位的需要，从知识、能力、素质三个方面进行人才培养。

硕士研究生的教育须突出“研究”，要加强理论基础的教育和科研能力的训练，使学生能够站在一定的高度去分析研究问题、解决问题。硕士研究生要通过课程的学习，进一步提高理论水平，为今后的研究和发展打下坚实的基础；通过相应的研究及学位论文撰写工作来接受全面的科研训练，了解科学研究的艰辛和科研工作者的奉献精神，培养良好的科研作风，锻炼攻关能力，养成协作精神。

高素质创新型计算机人才应具有较强的实践能力，教学与科研相结合是培养实践能力的有效途径。高水平人才的培养是通过被培养者的高水平学术成果来反映的，而高水平的学术成果主要来源于大量高水平的科研。高水平的科研还为教学活动提供了最先进的高新技术平台和创造性的工作环境，使学生得以接触最先进的计算机理论、技术和环境。高水平的科研也为高水平人才的素质教育提供了良好的物质基础。

为提高高等院校的教学质量，教育部最近实施了精品课程建设工程。由于教材是提高教学质量的关键，必须加快教材建设的步伐。为适应学科的快速发展和培养方案的需要，要采取多种措施鼓励从事前沿研究的学者参与教材的编写和更新，在教材中反映学科前沿的研究成果与发展趋势，以高水平的科研促进教材建设。同时应适当引进国外先进的原版教材，确保所有教学环节充分反映计算机学科与产业的前沿研究水平，并与未来的发展趋势相协调。

中国计算机学会教育专业委员会在清华大学出版社的大力支持下，进行了计算机科学与技术学科硕士研究生培养的系统研究。在此基础上组织来自多所全国重点大学的计算机专家和教授们编写和出版了本系列教材。作者们以自己多年来丰富的教学和科研经验为基础，认真研究和结合我国计算机科学与技术学科硕士研究生教育的特点，力图使本系列教材对我国计算机科学与技术学科硕士研究生的教学方法和教学内容的改革起引导作用。本系列教材的系统性和理论性强，学术水平高，反映科技新发展，具有合适的深度和广度。同时本系列教材两种语种（中文、英文）并存，三种版权（本版、外版、合作出版）形式并存，这在系列教材的出版上走出了一条新路。

相信本系列教材的出版,能够对提高我国计算机硕士研究生教材的整体水平,进而对我国大学的计算机科学与技术硕士研究生教育以及培养高素质创新型计算机人才产生积极的促进作用。

陈火旺

陈火旺现任国防科学技术大学教授、中国工程院院士。

第2版前言

本书是一本介绍面向对象设计方法的著作，是我们的前一本著作《面向对象的系统分析》[53]的姊妹篇，二者构成一个完整的面向对象的分析与设计方法体系。

面向对象的设计(OOD)是在面向对象的分析(OOA)基础上继续运用面向对象方法解决软件生命周期中设计阶段的问题，产生一个满足用户需求，并且完全可实现的系统模型，即OOD模型。在面向对象的软件开发中，系统分析建立的OOA模型离实现的要求还有很大的距离，因为还有很多设计问题尚未解决，需要在设计阶段运用OOD方法去解决这些问题，并且把设计结果在OOD模型中表达出来，使模型成为真正可实现的，这就是OOD所要解决的问题。

尽管OOD的出现早于OOA，但是长期以来关于OOD的理论与技术却远不如OOA成熟，其内容也不如OOA充实，主要存在以下问题：

- 对什么是OOD，各种著作和论文缺乏统一认识。主要原因是，早期的(即OOA出现之前的)OOD和现今的(即基于OOA的)OOD在内容上有很大的不同，而迄今大部分文献在讨论OOD时没有清晰地对二者加以区别，所以对OOD的概念、过程以及它应该包含哪些内容没有形成一致的见解。
- 以往大部分关于面向对象分析与设计(OOA&D)的著作都是以论述OOA为主，对OOD的论述则过于简略。对软件生命周期的设计阶段需要解决的大量实际问题缺乏全面、深入的讨论和切实可行的面向对象设计策略。有些方法对面向对象概念在设计阶段的运用基本上局限于对OOA结果进行细化，对许多全局性的设计问题没有给出与面向对象方法密切相关的设计策略，和在结构化设计中所采用的策略没有什么不同。
- 对OOA和OOD的关系，特别是对它们之间的界限和分工缺乏统一认识。一种基本一致的意见是，OOA和OOD属于软件生命周期的两个不同阶段，它们之间的界限不像结构化分析和结构化设计之间那样严格和清晰；但是除了这种原则性的共同见解之外，没有更进一步的一致意见。特别是对OOA和OOD之间既存在界限，同时又允许一定程度的模糊这个问题，大部分著作没有深入地加以讨论，并具体指出哪些建模问题应该明确地属于OOA或者OOD，哪些建模问题允许模糊二者之间的界限。
- 在以往的面向对象分析与设计方法中，Coad-Yourdon方法[8,9]是对OOD讨论最多的一种，对OOA和OOD之间关系的处理也较为得当。但是由于历史条件的限制，一些在20世纪90年代出现的理论与技术没有在有关的方法和著作中得以体现。对有些设计问题，例如一个用面向对象方法开发的系统如何用关系数据库或文件系

统存储其对象的问题,虽然提出了正确的解决方案,但是理论上的论述和技术策略的介绍都不够详细,使一般读者在学习之后仍然不知道如何实施。

本书作为一本在当前科学技术背景下出版的OOD著作,在学习和借鉴前人研究成果的基础上,力求在以下方面取得进步:

1. 系统地阐述OOA与OOD的理论体系

本书从区别早期的OOD和基于OOA的OOD入手,通过讨论二者在内容和特点上的不同,在概念上澄清了关于什么是OOD的问题。把面向对象的观点运用于整个软件生命周期,在此前提下对什么是OOD给出更确切的定义。详细地论述了OOA和OOD之间的关系。

2. 充实和完善OOD的内容

在面向对象的软件开发中,OOD是软件生命周期中的一个大阶段。在这个阶段中有大量的技术问题需要解决,需要建立一个可实现的系统模型。从工作量和技术难度来看,OOD的分量决不比OOA小。许多运用OO方法进行系统开发的读者所遇到的一个问题是,做完OOA之后不知道OOD到底该做些什么,因此对设计阶段的许多问题不得不继续采用非OO的设计技术去解决。本书的目标是向读者提供一种内容比较完善、策略具体、可操作性强的OOD方法,其中包含了普通应用系统的设计阶段需要解决的大部分问题,包括全局性设计决策和局部的模型细化两个方面的问题。

3. 充分运用OO基本概念解决设计问题

我们在《面向对象的系统分析》中提出,在OOA中应该充分运用面向对象的基本概念(即目前大部分面向对象编程语言能够直接支持的概念)解决各种复杂的建模问题,限制扩充概念的引入。本书依然坚持这一宗旨,没有采用比OOA更多的面向对象建模概念,更没有采用诸如"模块"、"块"等非OO的建模元素。此外,本书对所有的设计问题都是运用面向对象的观点给出设计策略,使读者能够在软件开发中完全采用面向对象的概念和表示法来建立系统的设计模型。这意味着本书运用了尽可能少的建模概念解决了较多的设计问题。建模概念的简练使本书提出的方法更容易学习、掌握和使用,并使得OOA、OOD和OOP在概念上保持高度一致,使模型与实现后的程序具有良好的映射关系。当然,为了做到这一点,本书给出了更强的过程指导,告诉读者如何运用一个精练的OO基本概念集合去解决各种复杂的建模问题。

4. 适应当前计算机科学技术的新发展

迄今在国际上影响较大的大多数OOA&D方法都是在20世纪80年代末和90年代初问世的。在此之后的近十年中,计算机科学技术在很多方面取得新的进步。例如软件复用技术的新发展、可视化编程环境的流行、软件体系结构的研究、网络与分布式系统的普及等。大量的新技术已被广泛地运用到当前的系统开发中。在这种形势下,要求OOD方法做出相应的发展。我们的目标是力求在OOD方法中体现计算机科学技术的新发展。本书的OOD方法是根据当前的技术背景提出的。针对在当前软件开发中被广泛采用的几种主要

技术，分别给出相应的设计策略，并且用 OO 概念表达其设计决策。

5. 解决工程实践中提出的问题

在本书的写作过程中，作者曾经以技术培训、工程指导、项目合作等多种方式与软件企业界的人士进行接触和交流。他们在运用面向对象方法进行软件开发时，常常在 OOD 阶段遇到这样那样的问题。其中有些问题在以往的著作中找不到现成的答案。这些来自工程实践的问题对于 OOD 方法的研究和发展具有很强的促进作用。本书的许多内容是针对这些问题开展研究并进行总结提炼的结果。

本书的第 1 版[51]于 2003 年 2 月出版，数年来已多次重印。这次再版主要有以下变化：

- 按照《面向对象的系统分析》从第 1 版[48]到第 2 版[53]的内容变化，对本书的内容做了相应的修改与调整。
- 根据 UML2.0 的最新版本和我国的“信息技术 软件工程术语”(GB/T 11457—2006)国家标准[45]等标准化文件，进一步规范了全书的术语和模型元素的表示法。
- 在第 7 章对 UML1.x 的介绍和评论中，增加了引导读者阅读文献[53]关于 UML2.0 相关内容的线索。
- 对全书的内容进行了全面的审核和订正，在文字表述方面比第 1 版更为准确和精练。

全书共分 7 章，其内容安排如下：

第 1 章讨论什么是 OOD。通过区分早期的 OOD 和基于 OOA 的 OOD，论述了 OOD 在不同历史时期的不同特点及内容。简要地介绍了几种典型的 OOA&D 方法。对历史上关于 OOA 与 OOD 之间关系的一些传统观点进行了分析和讨论，并从模型驱动的体系结构(MDA)的观点出发论述了应如何处理二者之间的关系。

第 2 章概括地介绍本书提出的 OOD 方法概貌，包括它所采用的概念、表示法、原则、模型及过程。这一章的内容是整个 OOD 方法的总纲。

第 3 章介绍 OOD 模型的核心部分，即问题域部分的设计。内容包括在 OOA 模型基础上针对编程语言、复用支持、硬件性能等实现条件对模型进行修改、补充和调整，完善对象的细节，定义对象实例，建立从 OOA 模型到 OOD 模型之间的映射。

第 4～6 章分别介绍 OOD 模型的三个外围组成部分。每个外围部分是针对一个方面的实现条件来设计的，其作用是处理问题域部分与这些实现机制的接口，并且隔离它们的变化对整个系统的影响。第 4 章实际上既包括人机交互部分的需求分析，也包括该部分的设计。在分析方面给出的主要策略是：从用况提取人机交互需求，然后对交互过程进行细化，进而确定命令的组织结构；在设计方面讨论的主要内容是：根据给定的实现条件选择实现人机交互的界面元素，并且在模型中用 OO 概念表示所有的界面元素以及它们之间的关系。第 5 章主要解决并发系统和分布式系统中控制流的设计问题。首先讨论了系统总体方案、软件体系结构、系统的并发性等相关技术问题，然后讨论如何按选定的软件体系结构风格确定系统的分布方案，进而识别系统中的控制流。最后给出用面向对象概念表示系统中进程和线程的设计策略。第 6 章讨论如何解决对象的永久存储问题。首先介绍文件系统、关系数据库管理系统和面向对象数据库管理系统这三种最主要的数据管理系统的原理、功能和技术特点，然后针对不同的数据管理系统给出相应的设计策略。对使用文件系统和关系数

据库管理系统的情况,首先论述了用非OO的数据管理系统实现对象存储的理论依据,进而给出把应用系统中的对象映射到文件或者关系数据库的存储方案,并详细地介绍数据接口部分的设计策略。

第7章介绍统一建模语言UML,并进行分析和评论。包括UML的背景及演化历史,语言体系结构及定义方式,UML规范中的主要文献,各种图、建模元素及表示法。着重解释那些使读者和用户感到困惑和难以理解的概念,目的是排除读者在学习和使用UML时的主要障碍。第2版保留了这一章中针对UML1.x中的介绍和评论,同时介绍了从UML1.x到UML2.0的演化背景,并给出引导读者阅读文献[53]中关于UML2.0相关内容的线索。

本书的研究与写作得到国家自然科学基金项目"元模型理论与建模方法研究"(60473064)和国家重点基础研究发展计划(973计划)课题"虚拟计算环境的程序设计方法学"(2005CB321805)的支持。在写作过程中,我们曾就书中的许多学术思想与南京大学徐家福教授,北京航空航天大学金茂忠教授,北京大学王立福、梅宏、孙家骕、麻志毅、谢冰、王千祥、张世琨等同事,以及其他许多专家和应用领域的朋友进行了讨论和印证,从中受益良多。北京大学软件研究所的OO建模工具课题组所研制的建模工具集JBOO4.0为本书和文献[53]中的OOA与OOD方法提供了技术支持。我们的许多博士生和硕士生为本书的写作查阅、收集了大量文献资料,并绘制了其中的一部分插图。在此,我们谨向上述单位和个人致以衷心的感谢! 最后,我们恳切希望各位读者对本书可能存在的错误与疏漏给予批评指正。

作者联系信息
联系人:邵维忠
通信地址:北京大学信息科学技术学院; 邮政编码:100871
电话:010-62751790; 传真:010-62751792
电子邮件:wzshao@pku.edu.cn

作者
2006年12月于北京大学

第1章 什么是OOD

顾名思义，面向对象的设计(OOD)就是运用面向对象方法进行系统设计。然而这只是一种很粗略的解释。在不同的历史背景和不同的场合下，OOD的目标、内容和方法有许多不同，确切地解释其含义须从历史说起。

1.1 早期的OOD

在软件生命周期的各个阶段全面地运用面向对象方法进行系统开发，是人们长期以来努力追求的目标。特别是，从分析与设计阶段就开始运用面向对象方法，比仅仅用面向对象编程语言来编程更为重要，并且更能从根本上发挥面向对象方法与技术的优势。对此我们曾在《面向对象的设计》[47]的译者序和《面向对象的系统分析》[48,53]的第1章中进行了论述。在面向对象的软件开发中，通常是首先进行面向对象的分析(OOA)，其次进行面向对象的设计(OOD)，然后进行面向对象的编程(OOP)和测试(OOT)。但是面向对象方法的发展历程，却是首先开始于OOP，然后发展到OOD、OOA和OOT，即OOD的出现早于OOA。

20世纪80年代初发布的Smalltalk-80是第一个比较完善的面向对象编程语言(OOPL)。随后涌现的一大批OOPL标志着OO方法走向了实用。此时人们认识到面向对象的软件开发不仅仅是编程问题，在采用OOPL所提供的类、对象等元素和封装、继承等机制来编写程序之前，必须先用面向对象的观点进行构思和设计。这样才能明确程序中到底应该定义哪些类和对象，并明确它们的内部特征和相互关系。

这一认识促使人们将面向对象的思想向前推进了一步——从单纯地解决编程问题推进到从设计阶段就面向对象运用这一思想。G. Booch于1982年发表了题为“Object-Oriented Design”的论文[2]，首次使用了“面向对象的设计”的术语；1986年，他又发表了题为“Object-Oriented Development”的论文[3]，较完整地阐述了他的OOD思想。当时，“面向对象的设计”和“面向对象的开发”都用OOD作为缩写，并且在内容上也没有根本区别——称“开发”时，也是主要讨论设计问题；称“设计”时，也不是单纯地针对软件生命周期的设计阶段，还涉及一些本应属于分析的工作。

R. J. Abbott在1983年提出一种通过正文分析来发现对象的方法，即用规范的英语对问题进行陈述，然后从正文中提取对象及其特征[1]。例如，通过名词识别对象，通过动词识别对象的操作以及通过定语成分识别对象的属性等。这种正文分析方法被后来的许多OOD方法采用。

Booch 方法[3]被视为最早的 OOD 方法。在此之后,从 1986 年到 20 世纪 90 年代初,相继出现了一批 OOD 方法,其中有:

- 通用面向对象的开发(general object-oriented development,GOOD)[34];
- 层次式面向对象的设计(hierarchical object-oriented design,HOOD)[16];
- 面向对象的结构设计(object-oriented structured design,OOSD)[39]。

由于面向对象的分析方法在 20 世纪 80 年代末尚未出现,所以当时提出的 OOD 方法都受到了这种历史背景的限制,其影响延续到 20 世纪 90 年代初期。在这种背景下出现的 OOD 方法具有以下特点:

(1) 不是在面向对象的分析基础上进行面向对象的设计,而是基于结构化分析。例如早期的 Booch 方法和 GOOD 都是从结构化分析所产生的数据流图(DFD)开始进行面向对象的设计,HOOD 是自顶向下地进行对象分解,其中要用到一些结构化分析和结构化设计的图形表示(如 DFD),并将所得到的结构化概念映射为对象及其外部接口。OOSD 也部分地基于结构化方法,是一种结构化和面向对象相结合的方法,为熟悉结构化设计的开发者提供了逐渐过渡到 OO 方法的措施。

(2) 大部分方法是针对特定编程语言的。例如早期的 Booch 方法是针对 Ada 和 Modula-2 的,GOOD 和 HOOD 都是针对 Ada 的。只有少数方法(例如 OOSD)是独立于语言的。针对特定编程语言的 OOD 方法在很大程度上受到语言的影响,例如将包或模块的概念反映到 OOD 方法中来。

(3) 从现今的观点看,早期出现的 OOD 方法大部分不是纯 OO 的。一方面对一些重要的 OO 概念缺少支持,例如 Booch 方法、GOOD 和 HOOD 都不能表示类之间的继承;另一方面却很注重对某些非 OO 概念(例如数据流、模块和异常处理等)的表示。

(4) 名曰"面向对象的设计",实际上不只是针对设计问题。早期的许多 OOD 方法往往在不同程度上包括了 OOA 阶段的工作,例如识别问题域中的对象。但是其识别对象的策略很不完善,特别是没有哪种方法强调从考察问题域中的事物入手来确定系统中的对象。因此,早期的 OOD 方法可以看成是现今的 OOA&D 方法的雏形,而不是只对应其中的 OOD 部分。原因是 OOA 方法在当时尚未形成,编程之前所有的建模问题(包括分析与设计)都属于新生的 OOD 方法所考虑的范围。对分析问题表达能力和处理策略的不足使之在很大程度上借助和依赖结构化分析。

1.2 基于 OOA 的 OOD

OOD 的出现是面向对象方法与技术的进步——在编程之前,首先运用面向对象方法建立一个系统设计模型,比单纯地用 OOPL 编程更能保证系统的 OO 风格。然而,在 OOD 中用来确定系统设立哪些对象类的依据是什么?这一根本问题尚未真正解决。从结构化分析的结果 DFD 来发现对象,显然不是运用 OO 原则的最好途径。面向对象方法的基本思想之一就是从问题域中客观存在的事物出发来识别对象并建立由这些对象所构成的系统,而不是先把问题域抽象为某种已经很难辨认事物本来面貌的非 OO 的软件概念(如 DFD),然后再通过它们来识别对象。识别对象最准确、最可靠的途径是以面向对象的观点直接地对问题域进行分析与研究,以系统责任作为当前目标对问题域中的事物进行抽象,从而确定系统

中的对象、类以及它们的内部特征和彼此之间的关系。这是一种崭新的分析方法，即面向对象的分析(OOA)。与 OOD 相比，OOA 对于建立良好的系统总体结构并保持系统模型与问题域的良好映射具有更为关键的作用。

因此，从 20 世纪 80 年代末开始，关于 OO 方法的研究从早期的 OOD 延伸到了 OOA。1989 年以后相继出现了一批以解决分析问题为重点，同时也考虑设计问题的面向对象的分析与设计方法，例如 Booch 方法[4,5]、Coad/Yourdon 方法[8,9]、Firesmith 方法[12]、Jacobson 方法(简称 OOSE)[17]、Martin/Odell 方法[22,23]、Rumbaugh 方法(简称 OMT)[32]、Seidewitz/Stark 方法[35]、Shlaer/Mellor 方法[37,38]、Wirfs/Brock 方法[40]等。在这些 OOA&D 方法中，OOD 的含义发生了变化：它不再像早期的 OOD 方法那样以独立的、自成体系的面貌出现，而是作为方法的一部分，与 OOA 共同构成一种 OOA&D 方法。OOA 一般是方法的主体部分，解决软件生命周期中分析阶段的建模问题(建立系统的分析模型)；OOD 是基于 OOA 的，是 OOA 的延续，解决设计阶段的问题(在分析模型基础上建立设计模型)。这是现今的 OOD 思想，本书称之为基于 OOA 的 OOD。

与早期的 OOD 相比，基于 OOA 的 OOD 有如下一些特点：

(1) 在面向对象的分析基础上，继续运用面向对象方法进行系统设计，一般不依赖结构化分析的结果。

(2) 与相应的 OOA 方法共同构成一个完整的 OOA&D 方法体系。在各种方法中，OOA 和 OOD 采用一致的概念与原则，分别解决软件生命周期分析阶段和设计阶段的问题，有不同的目标及策略。

(3) 较全面地体现了面向对象方法的概念与原则，例如类、对象、属性、操作、封装、继承、聚合、关联和消息等。

(4) 大多数方法是独立于编程语言的，即通过面向对象的分析与设计所得到的系统模型可以由不同的编程语言实现。

在当前流行的各种面向对象的分析与设计方法中，对什么是 OOA 和什么是 OOD 的解释虽有许多共同之处，但也有不少差异。例如对 OOA 中的“分析”(analysis)一词，有的方法指的是“需求分析”(requirements analysis)，有的方法指的是“系统分析”(systems analysis)，有的方法包括需求分析和系统分析并对二者加以区别，有的则包括两方面的内容而不加区别。又如对 OOA 和 OOD 的关系，有的方法倾向于将二者融合在一起，有的方法则较明确地指出它们属于软件生命周期的两个不同阶段。总起来看，各种 OO 方法流派趋于一致的意见是，OO 方法不像传统的方法那样强调各阶段之间的严格界限。这种意见是正确的，但是没有深入讨论对哪些问题可以模糊 OOA 与 OOD 的界限，哪些问题必须明确地归属 OOA 或 OOD。结果是，各种方法对于什么是 OOA、什么是 OOD 以及二者之间的关系各有各的解释，很不统一。

虽然 OOD 的提出早于 OOA，但是迄今各种关于 OOA&D 的著作对 OOD 的介绍大都不够深入，一般是以介绍 OOA 为主，仅用很少的篇幅讨论 OOD。各种方法对 OOD 到底该做哪些工作没有统一的认识，而且大都不够完整。只有 Coad/Yourdon 方法对 OOD 论述较多，并且给出了较好的 OOD 模型框架，但是对其模型中的各部分给出的设计策略仍过于简略。另一方面，20 世纪 90 年代以来出现了许多与系统实现有关的新技术和新理论，例如可视化编程环境、客户-服务器技术、分布式对象技术、构件技术、设计模式、软件体系结构等。

OOD方法如何与这些新技术相互融合和衔接,是工程实践者迫切要求解决的问题,这是对OOD方法提出的新挑战。

以下简单地考察几种OOA&D方法。构成各种方法的要素包括建模概念、原则、表示法、OOA/OOD模型框架、文档规范、过程指导、技术支持等诸多方面。讨论的重点是各种方法的OOA和OOD过程,目的是使读者大致地了解在各种方法中OOD主要做哪些事。

1.2.1 Booch方法

G. Booch的OOA&D方法是从他早期的OOD思想[3]发展而来的。他的第一本全面论述面向对象的分析与设计的著作于1991年出版[4],1994年对该书做了一些修订,发表了第2版[5]。

1. 概念、模型与表示法

Booch方法所采用的对象模型(object model)要素是:封装、模块化、层次类型、并发。重要的模型概念是类和对象,类和对象的特征(属性和操作),类及对象之间的关系(关联、继承、聚合、使用、实例、元类)。这些概念在分析和设计中都是一致的,但是在设计阶段引入了模块、模块间的依赖关系、进程等概念。

该方法使用了6种模型图,即类图(class diagram)、对象图(object diagram)、状态转移图(state transition diagram)、交互图(interaction diagram)、模块图(module diagram)和进程图(process diagram)。其中类图、对象图、模块图和进程图称为基本图,通常是不可缺少的;状态转移图和交互图称为补充图,只在必要时使用。在基本图中,类图和对象图既用于分析,也用于设计;模块图和进程图只用于设计,而且是最重要的设计文档。

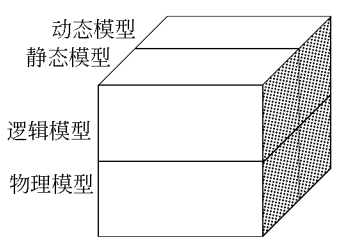


图1.1 从两个侧面组织系统模型

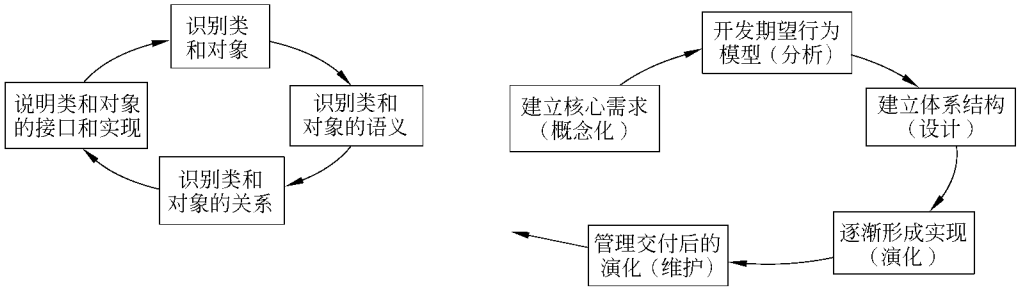
从总体上看,由上述各种图形成的系统模型可以从两个侧面来刻画,如图1.1所示。一个侧面着眼于模型的不同的抽象层次,分为逻辑模型和物理模型;另一个侧面着眼于模型中表达的静态建模信息和动态建模信息之间的区别,分为静态模型和动态模型。这两个侧面是正交的,即无论逻辑模型还是物理模型都包含静态和动态两种模型信息;反之,无论静态模型还是动态模型都包括逻辑的和物理的两个抽象层次。

2. 宏过程与微过程

Booch认为,面向对象的分析与设计过程不能像一本烹调手册那样简单地加以描绘,而应该是一种渐进的、反复进行的过程。他提出的OO开发过程分为微过程(micro process)和宏过程(macro process),如图1.2所示。

微过程是由宏过程的剧本和产品驱动的,是开发者在宏过程的各个阶段(或者说在不同的抽象级别上)反复进行的日常活动,包括以下4个步骤:

(1) 在给定的抽象级别上识别类和对象。在分析中主要是发现,即发现对问题域的抽象,确定什么是应该关心的,什么是不必关心的;在设计阶段主要是创造,即建立对于解元素的抽象;在实现阶段也是创造,即建立构成高层抽象的低层抽象,并且发现已有抽象的共性



(a) 微过程　　(b) 宏过程

图 1.2　Booch 方法的 OO 开发过程

以简化系统结构。

(2) 识别类及对象的语义。目的是在上述各阶段的抽象级别上建立类和对象的行为和服务。

(3) 识别类及对象之间的关系，包括继承、聚合、关联、可导航性等。

(4) 说明类及对象的接口和实现。主要是选择数据结构和算法。

上述微过程步骤如图 1.2(a)所示。该图有意将分析与设计的界限模糊化，并表明这些微过程步骤将在宏过程的控制下周而复始地进行。

宏过程是 Booch 方法对整个软件生命周期的过程布局，如图 1.2(b)所示。宏过程的每个步骤对应着软件生命周期中的一个大的开发阶段，其内部是由前面所述的微过程构成的。用 Booch 的话来说就是："宏过程是微过程的控制框架"。宏过程的步骤如下：

(1) 概念化　建立系统的核心需求。

(2) 分析　开发一个系统行为模型。

(3) 设计　创造一个可实现的系统结构。

(4) 演化　通过不断地细化而逐步实现系统。

(5) 维护　管理系统交付之后的演化。

关于上述步骤的详细解释，读者可参阅文献[5]。以下只对其中的分析与设计过程做进一步的介绍和引述。

3. 分析与设计过程

(1) 分析

在说明分析的目的时，Booch 引用了 S. Mellor 的论述[24]："分析的目的是提供对问题的描述。这种描述必须是完整的、一致的、可读的、可由有关各方复审的并可实际测试的。"用 Booch 本人的话说就是："分析的目的是提供一个系统行为模型。"Booch 强调，分析的焦点是行为，而不是形式。他认为在这一阶段追求类的设计、表示或其他技术决策是不合适的。分析必须产生关于系统"做什么"的陈述，而不是解决"怎么做"的问题。任何关于"怎么做"的陈述只有当它有助于揭示系统行为时才是有用的。在这一点上分析和设计很不相同。分析是通过识别形成问题域词汇的类和对象(以及它们的作用、责任和协作)为现实世界建模，设计是创造一种能够提供分析模型所需行为的人工制品。Booch 强调在分析中要识别

"功能点"(function point)。功能点是最终用户的业务功能,是从系统外部可观察、可测度的系统行为。对分析结果的表示,首先是用CRC卡(一种表示类、类的责任和类之间关系的描述机制,就像一张图书索引卡片),建立一个描述系统行为的剧本;然后用对象图和类图描述该剧本的语义,并表现类之间的关系。分析阶段包含的活动如下:

- 识别系统的基本功能点。如果可能的话,将它们组织到功能相关行为的类簇中。
- 对每个值得考虑的功能点,通过用况、行为分析和CRC卡技术编排一个剧本。当每个剧本的语义都清楚时,就用对象图为之建档,以表示发起或提供行为并合作完成剧本活动的那些对象。
- 如果需要,则生成二级的剧本,以表示异常条件下的行为。
- 当某些对象的生命周期对一个剧本很重要时,则开发这些对象的状态转移图。
- 整理剧本之间的模式,并以更抽象、更一般的剧本来表示这些模式,或者用类图的方式来表现关键抽象之间的关系。
- 对演化中的数据字典进行更新,使之包括为每个剧本识别的类和对象。

(2) 设计

关于设计,Booch说:"设计的目的是为增量式的实现创建一个系统结构,并制定系统的不同元素都必须采用的共同策略。"设计阶段包括系统结构设计、策略设计和发布设计等活动。

① 系统结构设计

- 考虑来自分析结果的功能点簇,将它们划到系统结构的一些层次和部分中。若一组功能建立在另一组功能之上,则划归不同的层次;若在同一抽象层次上协作完成某一行为,则划到该层的不同部分。
- 通过创建可实行的系统发布(一个发布满足若干由分析产生的剧本的语义)使系统结构生效。
- 提交该系统结构并评估它的强弱,识别其风险。

② 策略设计

- 针对给定的问题域,列出系统结构的不同元素必须采用的共同策略。有些策略是独立于领域的,如存储管理、出错处理等;有些策略是针对领域的,如实时系统中的控制策略和管理信息系统中的事务处理和数据库等。
- 为每项公共策略开发一个描述其语义的剧本。
- 为每项策略建档,并公布其版本。

③ 发布设计

- 将有关的功能点分配到一系列系统发布中,这些发布的最后交付就是系统产品的体现,此外还包括调整目标、调度以及任务计划等工作。

4. 评论

G. Booch是一位思想很活跃的OO方法学家。他的著作[4,5]不仅是他早期OOD思想的发展和完善,而且又继续提出了一些新的思想。但是他的著作对开发过程的介绍不很清晰。文献[5]专门讨论过程的第6章(本节关于该方法的过程介绍就是从其中摘录的)中,对宏过程的介绍没有进一步阐述在分析和设计中应如何实际执行其微过程步骤。比如,在从

宏过程的角度介绍分析和设计时，虽然都列出了一些操作步骤，但是没有清晰地说明这些步骤和微过程的步骤（识别类和对象、识别类和对象的语义、识别类和对象的关系、说明类和对象的接口与实现）之间是一种什么关系。或许这正是作者的本意——面向对象的分析与设计不能像一本烹调手册那样通过一些清晰的、可以逐条操作的步骤来描绘，但是对从事实际系统开发的读者而言，他们更希望看到的是可操作性较强的过程指导。

Booch 方法把类图和对象图作为两种不同的模型图，并主张在分析和设计中既使用类图，又使用对象图。这种做法在其他 OO 方法中是不多见的（虽然它也被目前的 UML 采纳）。这种观点在理论上是否合理，在实践上是否必要，都值得商榷。本书将在第 7 章评论 UML 时讨论这一问题。

1.2.2　Coad/Yourdon 方法

P. Coad 和 E. Yourdon 于 1990 年发表了他们的 OOA 著作[7]，次年再版[8]，并发表了与之配套的 OOD 著作[9]（文献[7]和[9]的中文译本为[46]和[47]）。

1. 概念、原则及表示法

Coad/Yourdon 方法强调 OOA 与 OOD 采用完全一致的概念和表示法，使分析与设计之间不需要表示法的转换。该方法所有的概念都是现有的 OO 编程语言能够直接支持的，从而可使概念上的一致性贯穿于整个软件生命周期。用于 OOA 和 OOD 的概念是对象、类、属性、操作、整体-部分结构、一般-特殊结构、实例连接、消息连接、主题。强调的原则是抽象、封装、继承、关联、消息通信、通用的组织方法、粒度控制、行为分类。

表示法也很简练，模型图和图形表示符号的种类都很少。其 OOA 著作的第 2 版对表示法做了一些修改：一是将整体-部分结构的集中式表示符号改为分散式，使画图方便一些；二是将整体-部分结构和实例连接的多重性改用数字和字母表示，这比原先采用不同的端点形状更加清晰；三是将没有对象实例的类（抽象类）和带有对象实例的类（称为“类及对象”）用不同的表示符号加以区别，但是这种做法带来了一些问题，I. Graham 曾对此提出批评意见[15]。

2. 面向对象的分析——OOA 模型与分析过程

Coad/Yourdon 方法对分析的定义是：“分析：一种研究问题域的活动，它将产生一个关于外部可见行为的规格说明，一个关于需要什么的完整、一致并且可实行的陈述，一个关于功能以及可量化的操作特征（可靠性、可用性、性能）的范围说明。”OOA 的目标是为应用系统建立一个反映问题域和系统责任、独立于具体实现条件的 OOA 模型。强调独立于实现，是为了加强 OOA 结果的可复用性，使一个 OOA 模型可以针对不同的实现条件进行设计和实现，从而可产生多个系统版本，或者形成系统家族。OOA 模型由 5 个层次叠加而成，如图 1.3(a)所示。

这 5 个层次是：

- 类及对象层　给出直接反映问题域和系统责任的类及对象。
- 结构层　描述类及对象之间的结构关系，包括一般-特殊结构和整体-部分结构。
- 主题层　将关系较密切的类及对象组织在一起作为一个主题，整个系统由若干主题构成，使读者能够从不同的粒度阅读系统模型。