中国计算机学会
学术著作丛书

邵维忠 杨芙清 著

面向对象的系统设计

清华大学出版社

中国计算机学会
学术著作丛书

耿卫东 杨芙清 著

面向对象的系统设计

清华大学出版社
北京

内 容 简 介

本书是我社 1998 年 12 月出版的《面向对象的系统分析》的姊妹篇，二者构成了完整的面向对象的分析与设计(OOA&OOD)方法体系。本书主要论述了如何在面向对象的分析(OOA)的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 和 OOD 方法，论述 OOA 和 OOD 的关系。第 2 章介绍本书的 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。

作者长期从事软件工程和面向对象方法等领域的科研、教学及工程实践。本书是他们参阅国内外大量文献，收集工程实践中提出的问题，总结自己多年研究的精心之作。本书内容详实、立论严谨、实例丰富、图文并茂，论述深入浅出。

本书的读者对象是：高等院校计算机软件专业的教师、研究生和高年级本科生，从事软件开发的工程技术人员，以及软件工程和面向对象方法等领域的研究工作者。

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. —北京：清华大学出版社，2003
(中国计算机学会著作丛书)
ISBN 7-302-06185-8

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

中国版本图书馆 CIP 数据核字 (2002) 第 107114 号

出 版 者：清华大学出版社(北京清华大学学研大厦，邮编 100084)
http://www.tup.tsinghua.edu.cn
责任编辑：焦 虹
印 刷 者：北京市清华园胶印厂
发 行 者：新华书店总店北京发行所
开 本：787×1092 1/16 **印张**：14.75 **字数**：337 千字
版 次：2003 年 2 月第 1 版 2003 年 2 月第 1 次印刷
书 号：ISBN 7-302-06185-8/TP·3702
印 数：0001～5000
定 价：29.80 元

前　言

本书是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》[30]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）的基础上继续运用面向对象（OO）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，通过 OOA 建立的系统分析模型（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，使模型成为可实现的，不再存在悬而未决的建模问题。这就是 OOD 所要解决的问题。

尽管 OOD 的出现早于 OOA。但是迄今为止关于 OOD 的理论与技术却远不如 OOA 成熟，内容也不如 OOA 充实，其中主要存在以下问题：

• 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准确、一致的见解。

• 以往大部分论述面向对象的分析与设计（OOA&OOD）的著作都是以论述 OOA 为主，对 OOD 的论述则偏于简略。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（例如 Rumbaugh 的 OMT[16]）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，与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。对于 OOA 和 OOD 之间的关系，特别是对二者之间既存在界限，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。

• 在众多的面向对象分析与设计方法中，Coad/Yourdon 方法是对 OOD 讨论最多的一种，对 OOA 和 OOD 之间关系的处理也较为得当[5,6]。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其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，该方法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，因此一般读者在学习之后仍然不知道如何实施。

本书作为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 给出了更确切的定义；详细地论述了 OOA 和 OOD 之间的关系；对与此相关的一些观点进行了分析和讨论。

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的阶段。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 的分量绝不比 OOA 小。作为一种 OOD 方法，如果只是注意到设计阶段的一部分问题，或者对大量的设计问题只是给出一些原则性的意见，甚至只是简单地罗列出来，那么读者就很难学会怎样实际进行面向对象的设计。目前从事实际系统开发的读者反应最强烈的一个问题就是，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，并且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》[30]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 与 OOD 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了新的进步。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，必然要求 OOD 方法不断向前发展。我们的目标是力求在 OOD 方法中体现计算机科学技术的新进展。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，我们曾经以技术培训、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 阶段遇到某些疑难问题，其中有些问题在以往的著作中找不到现成的答案。他们把这些问题反映给我们，希望我们进行研究和解决。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，本书的很多内容都是针对这些问题开展研究并进行总结提炼的结果。

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 和 OOD 方法，重点介绍了各种方法中的 OOD 部分。这一章的最后对 OOA 与 OOD 之间的关系进行了较深入的讨论。

第 2 章概括地介绍了本书提出的 OOD 方法的概貌，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 模型之间类的演化关系对照表。

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 use case 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 概念表示所用到的界面元素以及它们之间的关系。鉴于在当前的软件开发中已广泛采用可视化编程环境，该章的最后一节给出了在这种实现条件下更清晰、更有效的设计策略。第 5 章主要解决并发系统，特别是分布式系统中控制流（包括进程和线程）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（RDBMS）和面向对象数据库管理系统（OODBMS）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。由于文件系统和 RDBMS 是目前最常用的数据管理系统，所以把它们作为讨论重点。对使用文件系统和 RDBMS 的情况，首先论述了用非 OO 的数据管理系统实现对象存储在理论上是可以接受的，在实践上是合理和可行的，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，并详细地介绍了数据接口部分的设计策略。对使用 OODBMS 的情况也做了适当介绍。

第 7 章介绍统一建模语言 UML，并对它进行了较深入的分析和评论。对 UML 的介绍包括：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。在对上述内容的介绍中，着重解释那些使大部分读者和用户感到困惑和难以理解的概念，或者指出它们的缺陷，目的是排除大部分读者学习和使用

UML 的主要障碍。对 UML 的分析和评论，既肯定它的优点和成绩，也分析它的缺点和问题，目的是使读者了解 UML 中哪些概念是真正有价值的，哪些概念对普通用户用处不大或者在理论上存在问题。

本书所进行的研究得到了国家自然科学基金项目(60073015)和国家 863 高技术研究发展计划项目(2001AA113060 和 2001AA113070)的资助。在写作过程中我们曾就书中的许多思想和观点与南京大学徐家福教授，北京航空航天大学周伯生教授、金茂忠教授，我们的同事王立福教授、梅宏教授、孙家骕教授以及麻志毅、谢冰、张世琨、王千祥等同志进行过多次研究讨论。金茂忠教授还在百忙中抽出时间审阅了书稿并提出了宝贵意见。北大青鸟面向对象开发小组的同志在他们研制的建模工具 JBOO3.0 中为《面向对象的系统分析》[30]和本书所提出的面向对象分析与设计方法提供了支持。江南大学吴鸿雁副教授在北京大学访学期间运用书中的方法和青鸟建模工具开发了两个应用实例并制作了演示图片。我们的许多博士生和硕士生提前阅读了各章的书稿，并将他们发现的问题及时向我们反映。特别是蒋严冰、冯耀东两位同学为本书第 7 章的写作查阅、收集了大量文献资料，并绘制了其中的一部分插图。沈璞、徐松青同志为本书的计算机录入和绘图付出了许多辛勤劳动。在此我们谨向上述单位和个人致以衷心的感谢！在这里，我们还要特别感谢《面向对象的系统分析》[30]的广大读者。许多读者在寄给我们的信函和电子邮件中，或者在网站的论坛中对该书给予高度评价，同时殷切地期待本书早日出版。读者的赞赏和期盼为我们的研究和写作增添了巨大动力。使我们感到惭愧的是，由于日常工作的繁忙以及对书中讨论的许多问题需要开展认真的研究和实践，所以本书的出版比原计划推迟了不少时日。我们希望这份迟交的答卷能够以其内容报答读者的殷切期待。

本书适合作为计算机软件专业研究生或高年级本科生的教材、软件开发者的工程技术用书，以及软件工程和面向对象方法等领域的研究参考资料。我们希望广大读者把在教学、研究和工程实践中使用本书的情况告诉我们，并对本书可能存在的错误和疏漏予以批评指正。我们愿意在今后的工作中继续研究读者提出的新问题，并将多年来在面向对象领域从事科研、教学、培训和工程实践的心得体会与同行们进行交流。

作者联系信息如下：
联系人：邵维忠
地址：北京大学计算机科学技术系(100871)
电话：010－62751790
传真：010－62751792
电子邮件：wzshao@pku.edu.cn

作　者
2002 年 10 月于北京大学

目　录

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。但这种解释只是粗略的，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。确切地解释该术语的含义需要从历史说起。

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，比仅仅用面向对象的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。对此我们在文献[29]的译者序和文献[30]的第 1 章都曾做过论述。在面向对象的软件开发过程中，应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，进行设计。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设计领域。G. Booch 于 1982 年发表了题为“Object-Oriented Design”的论文[38]，首次使用了“面向对象的设计”这一术语；1986 年，他又发表了题为“Object-Oriented Development”的论文[39]，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，也不单纯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 年发表的论文[37]中提出，用规范的英语描述对一个问题的解释，然后从这段描述中提取对象及其特征。例如，通过名词识别对象，通过动词识别对象的方法(操作、服务)等。这种正文分析方法被后来的许多 OOD 方法所采用。

Booch 方法[39]被看做最早的 OOD 方法。该方法后来又进行了修订，在文献[1]和文献[2]中做了全面论述。在 Booch 方法之后，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)[47]；
- 层次式面向对象的设计(hierarchical object-oriented design，HOOD)[42]；

• 面向对象的结构设计(object-oriented structured design,OOSD)[18]等。

由于面向对象的分析方法在20世纪80年代末期之前尚未出现,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。当时的面向对象编程语言还不像Ada、Modula-2等非OO的语言(虽然带有某些OO特征)那样流行,针对这些非OO编程语言的OOD方法受语言的影响很大,例如将包或模块的概念反映到OOD方法中来。

(3) 从今天的观点看,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、包、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。比如,识别与问题域有关的对象,在现今的面向对象的分析与设计(OOA&D)方法中主要由方法的OOA部分来处理。而早期的OOD方法除主要解决设计问题之外,也在一定程度上涉及现在由OOA解决的对象识别问题。但是识别对象的策略尚不完善,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。所以早期的OOD方法可以看做是现今的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,首先运用面向对象的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。但是,通过OOD确定系统应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,显然不是符合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、它们的内部特征及彼此之间的关系。这是一种崭

新的分析方法，即面向对象的分析。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 年代末起关于 OO 方法的研究从早期的 OOD 延伸到了 OOA。1990 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[1,2]、Coad/Yourdon 方法[5,6]、Firesmith 方法[7]、Jacobson 方法（简称 OOSE）[11]、Martin/Odell 方法[14,15]、Rumbaugh 方法（简称 OMT）[16]、Seidewitz/Stark 方法[18]、Shlaer/Mellor 方法[20,21]、Wirfs/Brock 方法[22] 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，用于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，我们称之为基于 OOA 的 OOD。

与早期的 OOD 相比，基于 OOA 的 OOD 有如下一些特点：

（1）它是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，但二者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）较全面地体现面向对象方法的概念与原则，例如类和对象、它们的内部特征及相互关系，以及封装、继承等。

（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。比如对 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是软件生命周期的两个不同阶段。总的看来，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，对哪些问题必须明确地归属 OOA 或 OOD。结果是各种方法对什么是 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 的介绍大都比较薄弱，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，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与原则、表示法、OOA/OOD 模型、文档规范、过程指导、技术支持等诸多方面，我们讨论的重点是这些方法的 OOA 和 OOD 过程，即二者各自包含哪些主要活动。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[39]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[1]，1994 年他对该书做了一些修订，发表了第二版[2]。

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。

该方法使用的图形文档(表示法)包括以下六种图，即：类图(class diagram)、对象图(object diagram)、状态转换图(state transition diagram)、交互图(interaction diagram)、模块图(module diagram)、进程图(process diagram)。其中类图、对象图、模块图和进程图被称做基本图，一般情况下是不可缺少的；状态转换图和交互图被称做补充图，只在必要情况下使用。在基本图中，类图和对象图既用于分析，又用于设计；模块图和进程图只用于设计，而且是最重要的设计文档。

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。

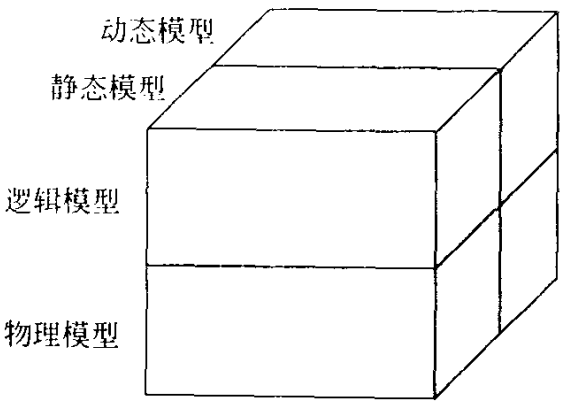


图 1.1 从两个侧面组织系统模型

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。他建议采用的 OO 开发过程分为微过程(micro process)和宏过程(macro process)。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。微过程包括以下四个步骤：

(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

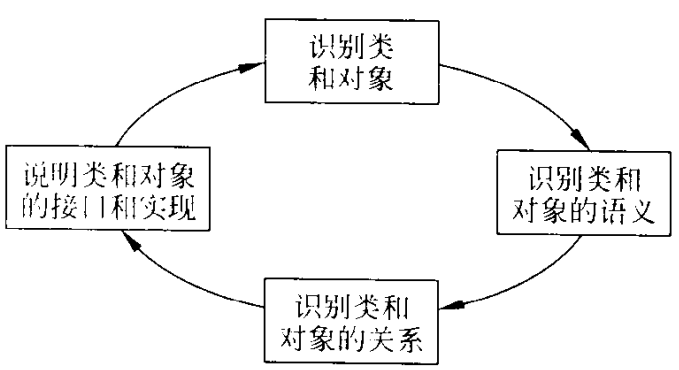
抽象，确定什么是应该关心的，什么是不必关心的；在设计阶段主要是创造，即建立对解元素的抽象；在实现阶段也是创造，即建立构成高层抽象的低层抽象，并发现已有抽象的共性以简化系统结构。

(2) 认别类及对象的语义。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。

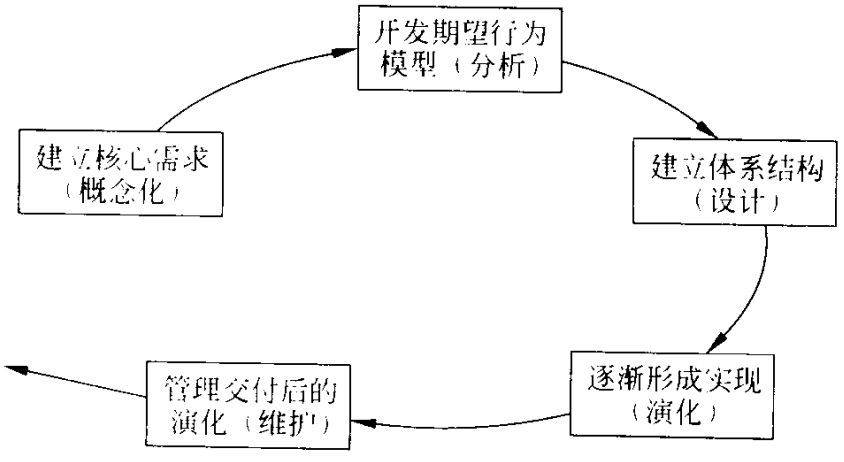
(3) 识别类及对象之间的关系，包括继承、聚合、关联、可导航性等。

(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。该图示有意将分析与设计的界限模糊化，并表明这些微过程步骤是在宏过程的控制下周而复始地进行的。



(a) 微过程



(b) 宏过程

图 1.2 Booch 方法的 OO 开发过程

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 Booch 的著作[2]。以下只对其中的分析与设

计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[15]："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，设计是创造一种可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；然后是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，用 use case、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，则开发这些对象类的状态转换图。
- 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："设计的目的是为渐进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之语义)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