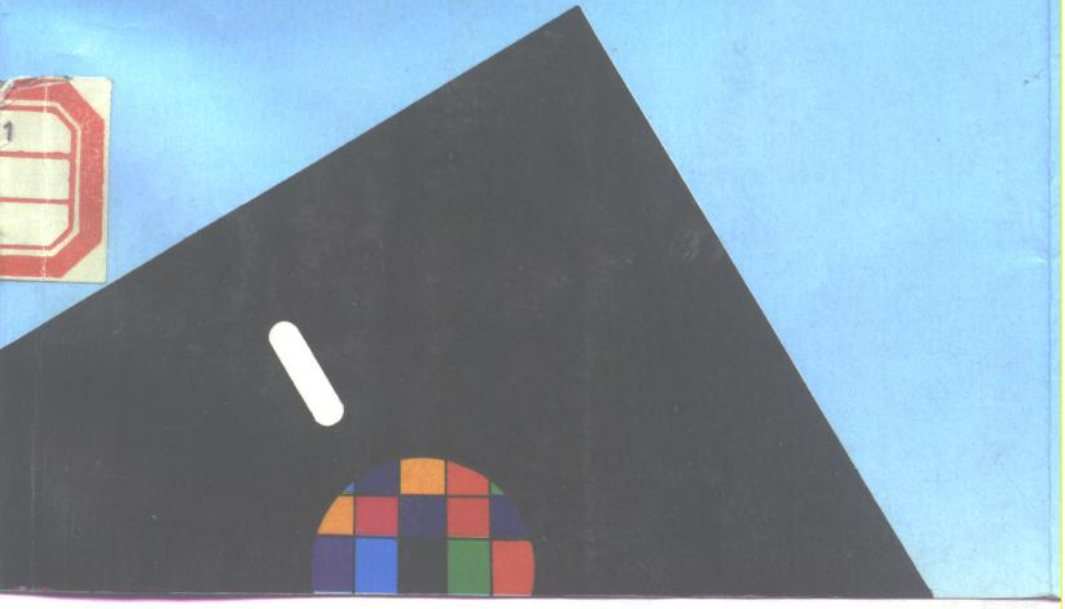
面向对象的分析

[美] PETER COAD　EDWARD YOURDON　著
邵维忠　廖钢城　李　力　译
杨芙清　校



面向对象的分析

[美] PETER COAD　EDWARD YOURDON　著
邵维忠　廖钢城　李力　译
杨芙清　校

北京大学出版社

新登字(京)159号

面向对象的分析

[美] Peter Coad Edward Yourdon 著
邵维忠 廖钢城 李 力 译
责任编辑:邱淑清
*
北京大学出版社出版发行
(北京大学校内)
北京大学印刷厂印刷
新华书店经售
*
850×1168 毫米 32 开本 6.375 印张 154 千字
1992 年 2 月第一版 1992 年 2 月第一次印刷
印数:0001—5000 册
ISBN 7-301-01955-6/O · 271
定价:6.10 元

译 者 序

自从面向对象语言 Smalltalk 及其环境出现以后，十年来面向对象技术的研究已遍布计算机软硬件各个领域：面向对象语言，面向对象程序设计方法学，面向对象操作系统，面向对象数据库，面向对象软件开发环境，面向对象硬件支持，面向对象的计算机体系结构等等。由于面向对象技术在软硬件开发方面呈现出巨大的优越性，人们将其视为解决软件危机的一个很有希望的突破口，从而使面向对象技术的研究和应用成为 90 年代计算机技术研究和应用的一个相当活跃的领域。

《面向对象的分析》(《OBJECT-ORIENTED ANALYSIS》)是一本讲述面向对象系统分析和需求定义的书。该书比较系统地介绍了将对象概念及面向对象的风格运用于系统分析和需求定义的思想，方法及步骤。该书首先以对象及其属性；类属及其成员；整体及其部分这几个基本概念为基础阐明 OOA 的方法论基础，即人类认识客观事物和理解现实世界过程中常用的三个基本法则：(1) 认识客观对象及其属性；(2) 认识对象的整体及其组成部分；(3) 对象类的形成及类的区分。同时剖析了传统的结构化分析方法，指出结构化分析方法的根本不足在于其功能分析和数据分析之间的不一致性和不协调性。在此基础上，提出了 OOA 的基本出发点：系统的数据流和控制流反映系统两个不同方面的本质特性，二者之间既对立又统一。因而，在进行数据分析和功能分析中应该使用一个统一的概念和方法，那就是对象。对象作为功能和数据的有机统一体成为 OOA 模型的最基本的抽象实体。

本书共分九章。第一章概述需求分析所面临的挑战，回顾了功能分解，数据流，信息模型化和面向对象这四种常用的控制复杂性

的方法，并进行了总结归纳和比较。第二章以一个面向对象的程序设计语言和环境 Smalltalk 为例向读者展示了面向对象的风格和基本要点。第三章到第七章讲述 OOA 方法五个阶段的任务，过程，方法和工具。第三章着重讨论什么是对象，为什么及怎样标识对象。第四章讨论什么是结构，为什么及怎样标识结构。第五章介绍什么是主题(SUBJECT)，为什么及如何标识主题。第六章讨论什么是属性，为什么及如何定义属性。第七章讨论了什么是服务(SERVICES)，为什么及如何定义服务。第八章集中讨论了 CASE 对 OOA 方法的支持，以及支持 OOA 方法的 CASE 工具所应具备的功能和当前已有的工具对 OOA 支持的程度。最后，第九章探讨了从面向对象的分析阶段进入到面向对象的设计阶段应考虑的问题，以及硬件结构和软件结构对 OOD 工作的影响，并进一步探讨了 OOD 的实现语言：过程化语言，面向程序包(PACKAGE-ORIENTED)的语言，面向对象的程序设计语言(OOPL OBJECT-ORIENTED PROGRAMMING LANGUAGE)，以及面向对象的数据库管理系统(OODBMS)对 OOD 的实现。

本书的中心点是讲述面向对象的系统分析。系统分析是系统设计，系统实现的基础。而分析的关键就在于理解问题空间并将其模型化。在 OOA 中，对象作为现实世界的抽象表示使系统分析员将注意力集中于对问题空间的理解，最终产生一个可见的、可复审的和可管理的稳定的层次模型。

本书原文的两位作者 PETER COAD 和 EDWARD YOURDON 教授都是长期从事软件工程方法研究的专家，其中 EDWARD YOURDON 教授是大家熟悉的提出 YOURDON 结构化方法的著名学者。这本书是他们专门为系统分析员而写的。值得指出的是 OOA 是一个新的方法，需要在实践中不断提炼。因此尚没有可以遵循的规范准则。本书可作为运用 OOA 方法的起点，不断进行扩充和完善。

本书由邵维忠、廖钢城、李力三同志合译，杨芙清教授审核、校

对了全稿。李素兰、张金荣同志为译稿的计算机录入和编排做了大量工作，在此谨表感谢。

译　者

1991 年 5 月于北京大学

目　录

引　言

OOA——面向对象分析是建立在对象及其属性，类属及其成员，整体及其部分这些我们在幼儿园就学过的概念的基础上的，但为什么长期以来一直没有将这些概念运用到分析和信息系统规格说明中？人们猜想可能由于结构化分析正处于全盛期，我们一直忙于追随这个潮流，从而忽视了对其他方法的考虑。

正如大英百科全书指出的：

人类在认识和理解现实世界的过程中普遍运用着三个构造法则：

(1) 区分对象及其属性。例如，区分一棵树和树的大小或空间位置关系。

(2) 区分整体对象及其组成部分。例如，区分一棵树和树枝。

(3) 不同对象类的形成及区分。例如，所有树的类和所有石头的类的形成和区分。

——摘自《大英百科全书》"分类学理论"

OOA 的概念和方法就是建立在这三个常用法则的基础上的。

1984 年，作者之一花了两年半的时间，考察了一个大飞机公司的实时结构化分析的实际应用。观察结果很有趣，但却令人不安。他所研究的一个分析员小组("DFD 组")在项目开始时就使用数据流图产生了一个总体功能分解，并以此作为进一步规格说明的框架。与此同时，第二个分析员小组("数据库组")则首先着眼于系统工作时所需要的信息，然后建立了一个信息模型(也称实体关系图或语义数据模型)。以后，DFD 组继续致力于对基本问题空间的理解(如一个控制器将对飞机的控制移交给另一个控制器时所

发生的详细情况)。相应的,数据库组也获得了对空运控制的深入理解。然而,结果却不相吻合;更糟的是他们之间互相矛盾。原则上,这两个模型应该设法统一起来,但迫于进度和预算的压力,在没有解决分歧的情况下,两个结果都被送交概要设计,并希望那时能解决上述分歧。但遗憾的是数据库组被看成令人讨厌的,甚至当成制造麻烦的人。这个重大的分歧及其错误的解决方法使人们付出了代价,并影响了他们的前途。

1987 年和 1988 年,同一个作者又在一个联邦政府机构和一个州政府机构的项目开发中看到了相同的情况。DFD 组先行一步,在时间和权力上都领先。数据库组获得了大量的对分析至关重要的认识,但却常常被忽视。而且,数据库组和他们的头头都被视为找麻烦的人。

实践中,经常看到,不同的处理及数据模型所采用的符号和策略相互分离,这就使 DFD 组和 DB 组永远难以统一。由于存在这种分歧,我们开始了方法(符号和策略)的研究和开发。这个方法将有助于分析员首先获得对问题空间的最必要的理解,然后,在基于确实理解的结构上增加处理需求。我们将这种方法运用于实践中,不断地进行提炼,使其成为一个系统的方法。最后,作者之一向全美国的高级专业人员,以及加拿大、以色列、意大利、联邦德国、瑞典和英国的高级专业人员提出了这个方法,并从他们获得了很有价值的反馈和启示。在 OOA 运用于实际项目时,客户和研讨会的与会人员都对这个方法的发展作出了显著的贡献。

0.1 组织

本书(我们相信它是第一本论述 OOA 的书)共分九章。

第一、二章是基础部分。"改善分析"这一章讨论了系统分析所面临的挑战,然后回顾了控制复杂性的一些规则。它总结了四种流行的分析方法:功能分解,数据流,信息模型和面向对象。第二章"体会对象观点",探讨了一个完全面向对象的程序设计语言和环

境，以此说明 OOA 使用的一些基本要点。

第三章到第七章论述 OOA 方法的五大步骤：标识对象，标识结构，定义属性，定义连接和定义服务。每一章都组织成“什么是”，“为什么”，“如何定义”和要点（简要的总结）几个部分。

第八章“为 OOA 选择 CASE”描述 CASE 对 OOA 的支持，指出了需要什么和当前已有的支持。

最后，第九章，转入面向对象设计（OOD），讨论设计上的考虑和 OOD 过程中 OOA 模型所发生的变化。

附录，“OOA 与 DOD-STD-2167A 的对照”叙述了在美国国防部 DOD-STD-2167A 防务系统软件开发工作中是如何运用 OOA 的。

0.2 为什么需要 OOA？

使用 OOA 有七个主要理由：

1. 它使我们能在人类概括客观事物的三个基本方法（对象及其属性，分类结构和组装结构）的框架上定义和交流系统需求。

2. 它首先集中于问题空间的理解——用户所处的环境和应用领域，以及所要开发的系统的本质的理解。

3. 它将对象的属性和作用于属性的服务视为一个固有的整体，而不是像其他的分析方法那样孤立地或不完整地处理属性和服务。这样，OOA 就将数据和处理模型结合成一个完整的模型。

4. 它允许使用自包含（self-contained）分块来分析和说明系统（使一个对象和其它对象之间依赖性最小），这就导致了一个更加稳定的模型。

5. 通过显式地表示共性而给我们带来好处。

6. 它为分析（做什么）和设计（怎么做）提供了一个一致而强有力的基本表示方法。

7. 它能适应系统族，也能适应不断进行的实际调整——例如，一个完整的系统对照一个实现部分较少的系统。

0.3 读者

本书的对象是实际中的系统分析员，他们每天都要应付现实的系统开发项目。假设读者具备计算机技术和系统分析概念的基本知识，并希望多数读者有一些分析工具（如数据流图和实体关系图）的使用经验。管理人员、测试人员、标准审核人员（standard bearer）和用户都可以阅读本书，并将从改善系统分析的整个方法中获益。

0.4 中心和历史

尽管在后面的章节中我们会更加明确，但这里应强调一下本书所讨论的是面向对象分析而不是面向对象程序设计（OOP）或面向对象设计（OOD），系统分析员必须首先理解手中的问题域。如果不首先对空运控制究竟是什么进行研究、表达并验证对它的理解，就直接开始写空运控制功能需求，其意义就不大了，更谈不上考虑设计结构或编写代码了。对象作为现实世界的抽象集中体现对重要的问题空间的理解。最终，由此产生一个实在的、可复审的、可管理的层次模型集，即由 OOA 五个主要步骤所产生的主题（subject）、对象、结构、属性、连接和服务。

我们可以争辩说这个观点尽管还不太流行，但一直就是很重要的。若果真如此，那么为什么本书还要讨论 OOA？为什么说对象范型（Object paradigm）最终已经趋于成熟？为什么要在现在讨论？

面向对象程序设计最初是在 60 年代后期，由使用 SIMULA 语言的人提出讨论的。到 70 年代，它就成为在 Xerox PARC 开发的 Smalltalk 语言的一个重要组成部分。与此同时，其他方面则随着如 COBOL 和 FORTRAN 等语言的发展而蹒跚地前进，并使用功能分解方法（我们将在第一章中详细讨论）解决设计和规格说明的问题。如果有的话，也很少集中讨论面向对象设计，更不用说面向对象分析。

在过去的十年里，发生了四个变化，这些变化现在成了我们跨入 90 年代的关键因素：

- 面向对象方法的基本概念经历了十年时间走向成熟，人们的注意力逐渐从编码转向设计和分析。提出功能分解的人们花了十年的时间从结构化程序设计转到结构化设计和结构化分析。面向对象的发展经历同样的过程是不奇怪的。
- 系统构造技术变得更强有力，不幸的是系统分析的思考方式仍然受到如何设计以满足其需求这种先入之见的影响；而有关设计的思想又受到关于如何编码的先入之见的影响；而编码的思想又在很大程度上依赖于现有的程序设计语言。因此，当我们选择汇编语言 AUTOCODER 或 FORTRAN 时，就很难考虑结构化的程序设计，这样也就难以考虑结构化设计和分析；当 PASCAL，PL/1，ALGOL，FORTRAN-77，结构化 BASIC 和新版本的 COBOL 相继出现以后，事情就容易多了。类似地，当可选择的语言仅仅是 COBOL，FORTRAN 或普通 C 语言时，也很难考虑面向对象风格的编程；有 C＋＋，Objective-C，Smalltalk 和 Ada 之后，就变得容易了。
- 今天我们所要建造的系统已不同于十年或二十年以前的了。在任何方面，它们都变得更庞大和更复杂；它们还更加不稳定和容易经常变化。在后面的章节中我们将论证分析和设计的面向对象方法可能产生更稳定的系统。而且，与 60 年代和 70 年代的面向正文的批处理系统相比，我们发现现今的联机交互式系统把更多的注意力放在用户界面上。有些观察家，如 Sun 公司的 Bill Joy 论证当今系统高达 75％的代码是与用户界面有关的，例如：操纵窗口，下拉式菜单，图形符，鼠标移动等等。我们的经验是：对这样的系统，面向对象方法（从分析到设计到编程）

是处理这种面向用户的系统的一种更自然的方法。

- 许多机构认识到今天所建造的系统比 70 年代和 80 年代建造的系统更加面向数据。对功能复杂性关注已不如以前了，而建立数据模型变得更加重要。

0.5 OOA 未来的发展

OOA 是一个很新的方法；它将在实践中不断进化。因此，我们恳请读者，不要在计算机会议上来找我们，告诉我们说你正在“遵循 Coad/Yourdon OOA 标准”开发软件。正相反，希望以本书为运用 OOA 的起点，跟踪和扩充这个方法以适应你们的组织或项目的需要。

我们希望以后进一步发展 OOA，目前正在考虑的一些问题有：

1. 用 OOA 描述 OOA 本身。
2. 为所有引用非隐含服务的消息增加显式参数。
3. 增加不需要响应的消息。
4. 允许属性本身就是对象。这要权衡对象嵌套所带来的好处和可看到的复杂性增加之弊端。
5. 增加对风险的识别与分析的特殊准则，以及面向对象开发的管理方法。
6. 将 OOA 和 OOD 作为一个完整方法的两个方面进行研究。
7. 增加和区分对象和对象的收集。

第一章　改善分析

1.1　分析面临的挑战

系统分析使那些像被海妖的歌声诱入迷境的人们又振奋又恼怒。分析的困难是什么？面临的挑战是什么？我们认为主要有三个困难使系统分析员在各种项目中陷入苦恼：问题空间的理解，人与人之间的通讯和不断的变化。

1.1.1　问题空间

分析员面临的最大问题之一是研究应用领域并有所发现。这就是理解"问题空间"的挑战。我们经常以请教者的身份极度地体验到这个问题：我们可能陷入到某个项目一星期，一个月，甚至有时长达一年。大多数情况下，我们根本不可能在用户业务和应用方面以行家自居，我们需要把握和深入理解问题空间，而且必须尽可能快地做到这一点。当然，对于多数系统分析员来说，情况不会太严重。但是，即使你碰巧既是分析员同时又是一个业务方面的行家，你仍然需要用来和同组的其他人有效地交流专业知识的工具。例如，如果你在雷达系统工作了二十多年，对问题空间可能相当熟悉；当你在为另一个雷达系统说明需求时，你的首要问题可能是与其他的雷达专家交流，同时还要与那些尚不能区分雷达系统和葡萄柚的项目成员交流。

分析员必须考虑他们工作的问题空间。例如，考虑空运控制问题。分析员必须投入到相应的问题领域中，甚至比那些整天从事空运控制工作却没有全面考虑的人更要体察入微。另一个例子，考虑一个业务系统，该系统保存有关汽车注册和发照的信息。该系统的分析员需要研究和熟悉所有的细节，包括由于特殊利益集团的要

求而产生的规则的许多例外。

上述讨论描述了一个称职的分析员所要完成的主要工作。它远不止是写出一些可观测的功能需求。诚然，一个分析员必须详细说明处理需求，简洁地装订使得同伴们能阅读和理解他所说明的那些需求。但问题空间的理解却是系统分析真正的难点。

如果一个分析员简单地认为自己具有用户应用领域的知识，那么他很可能满足于某个导致模糊的需求的想法。作者之一最近参加了一个大的空运控制项目，在该项目中，需求分析员甚至在做了软件需求的详细说明两年之后仍然在捕捉空运控制的基本概念。这种情况是不该发生的。分析员必须理解问题空间并建立它的模型，尤其对大的复杂的系统；有了这样的理解，就能简单明了地用文字详细说明可度量的需求①。

1.1.2 通讯

分析的挑战还包括通讯。分析员在整个分析过程中都需要通讯。为了从用户获得问题空间和需求，分析员必须与用户通讯。他要考虑通讯问题而且要自行推敲。他还要和同事互相切磋。最后，还必须将他对问题空间的理解及由此得出的需求反馈给用户，检验他对需求的理解。他还可能要帮助用户放弃一些受财力和进度限制而难以达到的需求。

“软件工程”的一个可笑的讽刺是：虽然从字面上让人想像它是一些公式和算法，是“过得硬的”科学方法，但实际上软件工程是非常面向人(people-oriented)的工作。最近，作者之一在芝加哥的一个会议上发言时，有人问到 OOA(或其他的软件方法)是否是软件开发成功的关键。回答是什么呢？我们说，是的，有一个一致的技术方法是非常有价值的。但软件方法只是在一定程度上，即帮助人们互相通讯时有作用。如果一个软件工程方法的应用只是产生了

① 无论其方法如何，分析员在一个项目的开始不大可能充分理解问题空间。分析就是一个不断学习用户业务及其详情的过程。