智能科学与非线性科学丛书

面向对象技术

蔡希尧 陈 平

西安电子科技大学出版社

面向对象技术

蔡希尧　陈　平

西安电子科技大学出版社
1995

（陕）新登字 010 号

内 容 简 介

本书对于面向对象技术的基本原理、分析和设计方法、主要的应用方面等作了较全面深入的论述。书的内容包括类型系统，面向对象的程序设计，面向对象的开发方法，面向对象数据库，面向对象技术在新一代操作系统、人机交互界面和系统模拟中的应用等，对于面向对象这一新兴技术的主要研究成果和开发应用作了系统的概括和阐述。书的内容新颖，论述得体，有自己的见解和风格。本书适用于计算机、信息系统工程等有关专业的研究生、大学教师和工程技术人员使用。

面向对象技术

蔡希尧　陈　平

责任编辑　徐德源

西安电子科技大学出版社出版发行

西安电子科技大学印刷厂印刷

新华书店经销

开本 787×1092 1/16　印张 22 6/16　字数 534 千字

1993 年 11 月第 1 版　1995 年 7 月第 2 次印刷　印数 2 001—7 000

ISBN 7－5606－0296－7/TP・0103　　定价：17.00 元

前　言

面向对象(Object Orientation)是当前计算机界关心的重点，是90年代软件发展的主流。实际上，面向对象的概念和应用已经超越程序设计和软件开发，扩展到很宽的范围，如数据库系统、交互式界面、应用结构、应用集成平台、分布式系统、网络管理结构、CAD技术、系统模拟、人工智能等方面。一些新的工程概念及其实现，如并发工程、综合集成工程等，也需要面向对象的支持。所以，我们认为：

- 面向对象是程序设计新范型。
- 面向对象是新的方法论。
- 面向对象是一门新技术。

本书是基于这一认识来撰写的。

本书的内容是这样安排的：第一章的标题为“什么是面向对象?”，阐述面向对象的概念和特点，以及作者对面向对象作为范型、方法论和技术的基本观点。其余各章分为三个部分，第一部分是关于程序设计的，包括第二、第三和第四章，讨论“类型系统”这一共同的基础，然后分别对面向对象的顺序程序设计和并发程序设计进行了讨论。第二部分是关于方法论的，讨论面向对象分析、面向对象设计，以及工具和环境，这部分只有一章，即第五章。第三部分是应用，包括两章，其中第六章是面向对象数据库，第七章讨论面向对象技术在新一代操作系统、人机交互界面以及系统模拟等方面的应用。当然，面向对象的内容非常丰富，可以有多种选择和安排。本书现在所选定的格局，一方面是基于作者的理解和所掌握的知识，同时也限于篇幅的约定。现在已经看出，有的内容是应当加强的，如面向对象方法论，但最好是另写专著。把类和继承性放在类型系统中讨论，只反映了现有的发展水平，这就是把类和子类当做类型和子类型看待，这种理解显然是有局限性的。至于应用，更可以有多种选择，我们只在信息系统的应用中涉及更多的几个方面作了介绍，所有这些方面的内容都作了压缩，因为篇幅有限，而所讨论的每一个应用方面都可以写成各自的专著。我们在每一章的最后都列了经过选择的参考文献，以弥补对内容作了压缩所导致的不足。

本书是在作者近年来研究面向对象技术的基础上撰写的。我们的研究工作得到以下的组织和单位的支持：国家自然科学基金、博士点基金、电子科学技术研究院、长城计算机集团公司应用开发事业部、西安电子科技大学软件工程研究所，作者向这些组织和单位表示深切的感谢。作为C++标准化的特邀专家，作者收到不少由ISO工作小组ISO/IEC JTC1/SC22/WG21寄来的资料；校友黄欣经常从美国给我们寄来新的软件，都对我们的研究和写作工作以很好的支持；西安电子科技大学出版社为书的出版创造了优惠的条件，我们在此对他们表示诚挚的谢意。

面向对象技术的发展日新月异，而我们在短时间内收集不到所需要的资料；在写作过程中不断地由于新的进展，数易其稿而仍感不足，加上实践的经验少、时间短等等，因此，

缪误所在，在所难免；与其他作者以及读者的观点不一，实属必然。作者希望各界人士不吝赐教，以期纠偏补益于来时。

作者识于西安电子科技大学
软件工程研究所
1993 年 1 月

目　录

第 1 章

什么是面向对象？

本章内容

* * *

本书是专门讨论面向对象有关的主要问题的，因此，我们在一开始，先要回答什么是面向对象这个基本问题。为了研究什么是面向对象，先要考察一下为什么我们需要面向对象。

1.1 为什么需要面向对象？

从计算机发明到现在，快半个世纪了，它的作用和影响，尽人皆知。在这将近半个世纪的时间里，人们除了使用计算机以外，不断地加深对计算机的认识，开发它的能力。到现在为止，为人们所认识并被开发的计算机的能力可以概括为 3 个方面：

(1) 计算的能力；

(2) 推理的能力；

(3) 人机交互的能力。

计算的能力是最早被人们所认识，并得到开发。40 多年来，计算机不断地更新换代，计

算的能力不断地得到提高，现在，仍然在继续开发。在计算机发明以后大约 10 年，人们开始探索计算机的推理能力，并逐渐形成了人工智能这一新的学科，走向工业以及其他方面的实际应用。推理能力的开发仍然方兴未艾。计算机和人的交互能力，如果把输入输出包括在内，当然是由来已久了，但是，把人的范围加以扩大，让非专业的普通用户能够和计算机对话，那是 20 世纪 80 年代才开始的事情。1984 年第一个图形用户界面问世以后到目前为止的这段时间，应该讲是计算机和人的交互能力得到充分开发的时期，正是在这一基础之上，计算机辅助设计（CAD）等新兴行业得以蓬勃发展。

计算机的这三种能力现在都还在进一步地进行开发，需要适当的技术支持，才能发展，才能提高。能够同时支持这 3 种能力开发的技术是面向对象。

计算能力的进一步开发，目前主要的发展方面是采用多处理机结构，实现并行计算，而达到并行计算的关键问题是怎样描述系统的并行活动，表达一个系统的实体的特性、通信方式及它们之间的交互作用——竞争和合作。面向对象技术中的对象所具有的特性，很适宜于担任并行计算中实体这一角色。对象是一个自备单元，彼此之间采用统一的消息传递通信方式，这些特点，正是并行计算所希望的。

在人工智能技术中，知识的表示是一个关键问题，目前采用的主要表示方法是框架和语义网络，它们和对象及消息传递构成的表示是等效的。有关人工智能的程序往往大而复杂，采用面向对象的程序设计范型是最合适的。面向对象程序的高度模块化，它所具有的类的继承性，易于构造并行处理等等，对于人工智能的开发，都是很有意义的。

人机交互系统的开发和面向对象的关系更是密切，本世纪 80 年代兴起的图形用户界面，一开始就和对象的概念及其应用联系在一起。图形很自然地可以用对象来描述，用类和继承性来找出各种图形的关系。现代的窗口系统都是事件驱动的，消息传递则是触发事件的基本机制。人机交互系统的进一步发展，仍然依靠面向对象技术的支持。

以上是我们需要面向对象的最基本的理由。也可以从另外的角度来说明什么我们需要面向对象：

(1) 面向对象是一种新的非常有效的程序设计范型，对于提高软件的生产率、可靠性、可重用性等都是很有帮助的，现在大体上已经得到统一认识：面向对象的程序设计是 90 年代程序设计发展的主流。

(2) 面向对象正在为新的信息系统的设计方法论开辟道路，近年来，面向对象分析、面向对象设计得到迅速发展，并开始实际应用，信息系统开发的生命周期模型也正在更新。

(3) 除了程序设计和信息系统的开发以外，面向对象的作用还涉及更加广泛的领域，如前面已经提及的人工智能技术，图形用户接口，以及正在快速发展的 CAD、CAM 技术，计算机集成制造系统（CIMS），交互式系统模拟，网络技术等等，所以，面向对象是正在兴起的影响面广的新技术。

这 3 个方面的问题，将在本章中作进一步的论述。

1.2　面向对象的概念和特点

在面向对象系统中，有以下一些基本的概念及其特点，概要地加以介绍，在以下各章中将有详细的讨论。

1. 对象：

在面向对象的系统中，对象是基本的运行时的实体，它既包括数据（属性），也包括作用于数据的操作（行为）。所以一个对象把属性和行为密封成一个整体。从程序设计者来看，对象是一个程序模块；从用户来看，对象为他们提供了所希望的行为。在对象内的操作通常叫做方法。

2. 消息：

对象之间进行通信的一种构造叫做消息。当一个消息发送给某个对象时，包含要求接收对象去执行某些活动的信息。接收到消息的对象经过解释，然后予以响应。这种通信机制叫做消息传递。发送消息的对象不需要知道接收消息的对象如何对请求予以响应。

3. 类：

一个类定义了一组大体上相似的对象。一个类所包含的方法和数据描述一组对象的共同行为和属性。把一组对象的共同特性加以抽象并存贮在一个类中的能力，是面向对象技术最重要的一点；是否建立了一个丰富的类库，是衡量一个面向对象程序设计语言成熟与否的重要标志。

类是在对象之上的抽象，有了类以后，对象则是类的具体化，是类的实例。类可以有子类，同样也可以有父类，形成层次结构。

4. 继承性：

继承性是父类和子类之间共享数据和方法的机制。这是类之间的一种关系，在定义和实现一个类的时侯，可以在一个已经存在的类的基础之上来进行，把这个已经存在的类所定义的内容做为自己的内容，并加入若干新的内容。图 1.2.1 表示了父类 A 和它的子类 B 之间的继承关系。

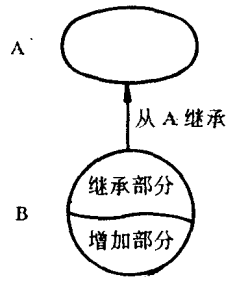

图 1.2.1　类的继承关系

继承性是面向对象程序设计语言不同于其他语言的最主要的特点，是其他语言（如面向过程的语言）所没有的。

图 1.2.1 中的 B 只从一个父类 A 得到继承，叫做“单重继承”。如果一个子类有两个或更多个父类，则称为“多重继承”。

5. 多态：

在收到消息时，对象要予以响应。不同的对象收到同一消息可产生完全不同的结果，这一现象叫做多态。在使用多态的时候，用户可以发送一个通用的消息，而实现的细节则由接收对象自行决定，这样，同一消息就可以调用不同的方法。

多态的实现受到继承性的支持，利用类的继承的层次关系，把具有通用功能的消息存放在高层次，而不同的实现这一功能的行为放在较低层次，在这些低层次上生成的对象能够给通用消息以不同的响应。

6. 动态联编：

联编是把一个过程调用和响应调用而需要执行的码加以结合的过程。在一般的程序设计语言中，联编是在编译的时候进行的，叫做静态联编。动态联编则是在运行时进行的，因此，一个给定义的过程调用和码的结合直到调用发生时才进行的。由于动态联编是在编译

以后才进行的，所以也叫："滞后联编"。

动态联编比较灵活，是面向对象程序设计语言的一个特点（但不是必需的）。动态联编是和类的继承性以及多态相联系的。在程序运行过程中，对象收到消息，但响应消息的方法可能存放在较高的层次之中，于是动态地找到这个方法，并把它和码加以结合。

在以上列举的 6 个概念及其特点中，有的在其他语言中已经采用，并不是面向对象系统中最早定义的。例如对象就是一个很广泛的概念，我们现在所介绍的对象，是在抽象数据类型（ADT）基础之上加以定义的，即使这样，在其他语言中也已经加以采用。消息传递的概念在并发程序设计中普遍使用，是大家熟知的。多态在许多语言中早已使用，算术运算中的加法算符就是多义的，参与加法运算的数据类型可以是多样的；有的语言则设置了类属函数，这些函数的变元类型是不固定的，因而对函数的调用，其结果则随着采用什么类型的变元有关。动态联编在一些解释性语言（如 Lisp）中也已采用。面向对象语言所增加的是类及其继承性，另外则是把以上的这些概念和特点加以扩充和提高，并把它们结合起来，形成了更加强的特有的能力。例如对象虽然在其他语言中已经使用，但和面向对象语言中的对象仍有所不同。在普通的语言中，着重于对象的内部实现，在面向对象的语言中，着重的是对象和外部的通信以及接收到消息时如何响应；对类的概念是在 Simula 语言[1]中最早出现，现在的面向对象程序设计语言被认为发源于 Simula。Smalltalk[2]则把类和子类的共享机制称之为继承性。现在，普遍地把有类的继承性，做为面向对象语言和其他语言区别的标志[3]。当一个语言把对象作为一个语言特征时，被称为"基于对象的语言"。只有当对象属于类，而类的层次可以用继承性增量式地加以定义的语言，才叫作"面向对象的语言"。这一概念的进一步推广，我们可以定义"基于对象的系统"和"面向对象的系统"，等等。还有一种语言，如 CLU，定义了对象和类，每个对象都属于一个类，但类没有继承性。这是处于基于对象和面向对象之间的语言，被称为"基于类的语言"。这样，能够支持对象作为语言特征的程序设计语言，分为三个等级，首先是基于对象，其次是基于类，最后是面向对象。在本书中，我们着重讨论有关面向对象的问题，有时也涉及基于对象的一些问题。

1.3 回顾与展望

回顾历史可以激励现在，规划将来。

在 1.2 节中说过，面向对象发源于 Simula 语言，在这个语言中引入类的概念。但真正的面向对象程序设计是 Smalltalk 语言奠定基础的，"面向对象"这个词也是 Smalltalk 首先采用的。Smalltalk 的主设计师是 Alan Kay。70 年代前期，微型计算机开始问世，Kay 设想在这种个人使用的机器上，用简单的方法能够处理各种形式的信息，他把所设想的计算机系统叫做 Dynabook，在 Xerox 公司的 Palo Alto 研究中心，Kay 和他的研究小组研制了 Dynabook 的第一个型号的硬件，叫做 Alto，同时给出了 Smalltalk 做为软件。

Smalltalk 的很多内容借鉴于 Lisp 语言，而它的核心概念——类，则取自于 Simula。实际上当时的 Smalltalk 离开一个实用的语言还有很大距离，它只是一个程序设计环境，包括四个部分，反映了面向对象的概念。四个组成部分是：(1) 程序设计语言的核：包括语言的语法和语义。(2) 程序设计风格：使用核生成软件系统的方法，这是和面向对象程序设计结合

最密切的部分。(3) 程序设计系统：是对象和类的集合，为编程提供方便。(4) 用户界面模型：提供对象和类结合的样式。

Smalltalk 现在被认为是最纯的面向对象语言，但是在本世纪 70 年代，它没有在 Xerox 的 Palo Alto 研究中心以外得到很好的推广和应用，没有进入计算机社会的主流之中。原因之一是使用这一新的语言和程序设计范型需要特殊的计算平台，这对于商用软件的开发者来说，在经济上是不合算的。经过不断的改进、充实和完善，到了 80 年代① 初期，Xerox 公司终于推出了新的版本 Smalltalk－80[4]，才引起人们的重视。

与此同时，Bell 研究所的 B. Stroustrup 着手在 C 语言的基础之上加以扩展，使之成为一个面向对象的语言，定名为 C＋＋[5]。由于 C 在 80 年代已经成为通用的开发语言，它不仅可用于微机，而且可用于范围很宽的计算结构和环境，因此，在 C 的基础上扩展而成的 C＋＋，虽然不是纯粹的面向对象的语言，却继承了 C 的语言构造，容易学习，不需要特殊的计算平台，这些比 Smalltalk－80 的明显优点，受到计算机界的普遍欢迎，许多 C＋＋商业版本和工具相继问世，原来阻碍面向对象程序设计范型推广使用的性能价格比问题也顺利地得到解决。所以，C＋＋的问世，促进面向对象技术的发展，也使得它唯一有可能成为标准化的面向对象程序设计语言，国际标准化组织 (ISO) 已为此成立了一个工作小组：ISO/IEC JTC1/SC22/WG21。

除了 Smalltalk－80 和 C＋＋以外，还有一些其他的一些面向对象的程序设计语言，也在 80 年代相继出现，例如 Objective－C[6]，面向对象的 Pascal[7]，CommonLoops[8]，Flavors[9]，以及这两个语言的后继产品 CLOS。这些面向对象的语言都是已有语言的扩充，但不象 C＋＋那样被普遍采用。新设计的有 Eiffel[10]，它也是纯粹的面向对象语言，特点是全面静态类型化，并支持多重继承，在欧洲计算机界受到重视。此外，还有象 Ada 这样重要的语言，也正在修订，使之具有面向对象的特性，新的版本取名为 Ada9X[11]，预计于 1993 年推出。

面向对象技术在 80 年代兴起有许多原因：主要是：

(1) 微电子技术的迅速发展，使微型计算机的价格不断下降，性能不断提高，能够满足高性能工作站的要求；而工作站则要求有高质量的图形用户界面，丰富的工具，和集成的开发环境的支持，推动了计算机主流往面向对象的方向发展。

(2) 软件的规模不断扩大，复杂的程度日益提高，这就需要多层次的抽象，以满足应用的需要，面向对象程序设计范型恰好具有这一特点，从对象到类到类库，以至于专用的构架，能够支持当前的复杂软件的开发。

(3) 新的工程概念，如并发工程，综合集成工程等的形成，以及这些概念付诸实践所产生的巨大的效益，促使面向对象技术的发展。这些工程的关键成份是专家群体和计算机系统，而后者则包括 CAD，CAM，CAE 等计算机辅助系统和共享信息源系统，都需要面向对象技术的支持。

(4) 软件工业本身的要求。软件的发展速度仍然远远地落后于硬件的发展，不能满足应用的要求，需要有新的软件开发过程模型，需要有新的方法论，要在这些方面取得新的发展和提高，最大的希望寄托于面向对象。

因此，展望将来，面向对象技术是充满活力和希望的，在 90 年代，肯定是计算社会中

① 指 20 世纪 80 年代。以下类同。

的主流。

下一节开始，将对 1.1 节中提出的三个问题：面向对象是新的程序设计范型；是信息系统设计的方法论；是正在兴起的新技术，分别进行论证。

1.4　面向对象是程序设计新范型

一个程序设计范型可以认为是一类程序设计语言的基础，是执行设施的基本集合，或者是关于计算机系统的思考方法。它体现了一类语言的主要特点，这些特点能用以支持应用领域所希望的设计风格。

流行最广泛的程序设计范型是面向过程范型，我们平常所使用的程序设计语言大多属于这一类范型，称它们为面向过程的语言。这一程序设计范型的中心点是设计过程，所以在程序设计时要决定所需要的过程，然后设计过程的算法，找出其最好的。在这类范型的程序中，过程的调用是关键的一环，语言必须提供设施给过程（函数）传送变元和返回的值；如何传送变元，如何区分不同种类的过程（函数、子程序、宏构造等）是这类程序设计中关心的主要问题。

随着程序规模的增大，数据组织成为一个重要的问题。1972 年，Parnas 提出了有名的“信息隐蔽原理”[12]，它的基本思想是：把需求和求解的方法分离；把相关信息——数据结构和算法——集中在一个模块之中，和其他模块隔离，它们不能随便访问这个模块内部的信息。在信息隐蔽原理的指导下，产生了新的程序设计范型，即模块化程序设计，或称信息隐蔽，代表这一范型的语言是 Modula 及其改进的版本 Modula－2。在这种范型中，程序设计的首要问题是划分模块，数据则隐蔽在模块之中。每个模块要有一个接口，模块的表示只能通过接口进行访问，模块在第一次使用以前要初始化。

把程序分解成模块的概念形成以后，人们很快地认识到把模块中有关实现的详细信息应予以局部化，把模块类型化，对一个类型的模块设置足够的操作集，形成了抽象数据类型（ADT）的概念，程序设计的准则把注意力集中决定所需要的类型。ADT 的概念对程序设计的影响很大，新的语言都和这一概念有关，面向对象程序设计也是以 ADT 为重要的基础，对象和类都可以看做是一种抽象数据类型。

面向对象程序设计是在以上的范型之上发展起来的，它的关键在于加入类及其继承性，用类表示通用的特性，子类继承父类的特性，还可以加入新的特性。对象是类的实例。所以，在面向对象的程序设计范型中，首要的任务是决定你所需要的类，每个类必须设置足够的操作集，利用继承机制显式地共享共同的特性。

类及其继承性的设置，克服了 ADT 的类型的通用特性和它的特例之间的特性设有什么区别的缺点，是抽象程度的进一步提高。

要使一个程序设计范型有生命力，能够做到通用，要具备一些基本的特点[13]：

（1）能够在通用的机器上运行。

（2）可以在常用的操作系统的支持之下工作。

（3）程序在运行效率上可以与传统的程序设计范型相匹敌。

（4）能够适用于主要的应用领域。

这些特点说明，新的语言投入使用，新的程序设计范型是否受到欢迎，必须考虑到能

否充分地利用已有的资源和开发环境，因为应用总是和经费开支联系在一起，和人员培训联系在一起，而且不能和传统断然割裂。这也正是为什么开发了多年的纯粹的面向对象语言 smalltalk，不如后起的并非纯粹的 C＋＋那样得到广泛欢迎的原因所在。从“关于计算机系统的思考方法”，“支持应用领域所希望的设计风格”等方面来看，面向对象范型也有许多优于其他范型的优点。用对象为客观事物造型，和人们习惯的思考方法比较一致；面向对象系统的层次分明，层次间的关系明确，这正是应用所希望的。在本节的最后，将进一步阐述面向对象程序设计范型的优点。

人们可能要问：把数据和操作分离，固然会给程序设计带来很多麻烦，但是，在传统的面向过程的语言中，不是有子程序吗？有的语言还带有丰富的子程序库，为程序设计提供了方便。对象、类和类库不是和子程序以及子程序库相似吗？事实上，对象和子程序是有很大区别的，Verify 列举了以下 4 点[14]：

（1）对象反映了更深的抽象。

（2）对象只对发送给它们的经过严格定义的消息做出响应。

（3）对象接收消息后，对系统进行全面控制，直到它自己通过另外消息把控制转移给其他的对象。与此不同，传统的子程序总要把控制权归还给主程序。

（4）对象之间的接口是明确定义的，不会被搅乱，传统的使用子程序的语言没有这样明确的接口，一个随意性较大的 FORTRAN 程序有可能错误地激活子程序深处的码。

对象和面向过程程序语言中子程序的这些不同，正是要从面向过程范型往面向对象范型转移的一个重要原因。

类及其继承性是面向对象程序设计最重要的特征，也是面向对象软件有比较强的重用能力的基础。所以，要掌握面向对象程序设计范型，必须要对类和类库的设计、组织和管理有深入的理解。

类是有相似特性对象的共同特性构成的，对象的特性多种多样，因此类也是多种的，类库是类的集合，并且给出多种类之间的关系描述。构架则是专用的类库，例如图形用户界面，各种 CAD 系统，都有自己的构架。在设计和实现面向对象程序的时候，要用类和类库，才能得到所需要的对象，即类的实例。所以，类库是一种预定义的面向对象的程序库。程序员使用类库中的类，和使用语言中的类型相似，还可以根据应用的需要予以扩充。

程序的开发需要类库，一个信息系统也希望在自己内部组织类库，以支持系统的开发和应用。

建造和维护一个类库的工作，统称为“类的管理”，它包括通常的数据库管理的一些内容，如数据的造型，访问方法和权限等，还有一些是为类的管理要做的新的工作。例如当需求改变或设计更新时，类也有所更动，叫作类的改造。开发人员需要帮助以找到一个类来重用，叫做类的包装。还有类库的安全性等，也属于管理的任务。

处理类之间的关系是类的组织的任务。软件和信息系统的开发需要掌握类之间关系，这首先是为了重用。当然，单个类是可以重用的，但有时候一个类和另一个类相依，重用将是一组类，这就需要知道这种相依的关系。其次，类之间的关系有助于对类的选择和使用。第三，类的关系可以用来检测不一致性和不完全性，例如有类但没有它的父类，则系统是不完全的。

在类库中，类的关系可以分为两种。一种是结构的关系，如继承关系（Subclass of），实

例关系（Instance of）等，这些关系都可以从源码中导出。另一种关系是不能从源码中导出的，而是由外部的机构显式地加以定义的。

类库的基础是类，但利用一个面向对象程序设计语言去开发一个可以在一定的领域中重用的类，是不容易的，往往要经历多次的反复。这是因为[15]：（1）经验表明，稳定的可重用的类必须经过反复的测试和改进。（2）很难在预先定义的分类学中安排类。（3）用户的需要很少是稳定的，新增加的约束和功能必须经常地集成到现有的应用去。（4）当共享类的程序员队伍不是来自共同的、标准的水平时，重用软件将引起复杂的集成问题。

在选择类库时，有两个主要的方面要加以考虑，第一是库包含的范围，它是通用的还是专用的？第二是库是语言的卖主所提供的软件包的一部分（如 Eiffel 库），还是对许多系统都可用的独立产品（如 Classix 库，可用于多种 C＋＋语言产品）。此外，为了便于重用，类库必须是高质量的，易学易用，使用的效果应当比从头开始进行设计的程序要好。

面向对象程序设计范型的使用历史很短，对于类库，虽然已有不少商品问世，但语言本身还没有国际标准，类库的构造也还没有公认为的准则，但已有一些很好的经验总结和建议。以下是 Korsn 和 McGregor 提出的有关类库应具备的特性和设计规范[16]，只将其中的项目列举如下，详细的论述可参考原文。

1. 类库的特性

・完全性：要覆盖全部概念。

・一致性：库的每个方面遵循一致的方法，如所有类的文档用一种标准格式。

・易学。

・易用。

・效率高。

・可扩展。

・可集成：可以把不同厂商的库结合起来成为一个软件库；在同一应用中能使用不同库的类。

・直观的。

・健全的。

・可以得到支持的。

2. 准则

共提出 23 条，并分成 7 个方面，以这些准则来保证上述类库特征的实现。

（1）域：每个类库都 有一覆盖的域。

・完全性：能提供一个完全的通用的模型，覆盖所需要的概念。

・抽象：环绕少数关键的抽象来设计类库。

・标准：设计要按域的标准知识建立模型。

（2）库的结构：由类之间的关系形成。

・继承结构。

・纯正：库应设计成类的网络，没有自由于网外的项。

・耦合：类之间的耦合是低层次的。

・异常：库应提供一种一致的和易懂的方法以处理差错和异常。

（3）类的设计：

·偏函数：是指有前提的函数。对每一偏函数，要设置一个“检查函数”，以检查前提是否满足。

·抽象的完整性：库的用户不可能破坏类所表示的抽象。

·完全的类的接口。

(4) 实现：

·高效率。

·一致性：库中的项目定义和命名必须一致。

·类属性：可能的话，提供参量化的类属性的类。

·完全的实现：例如一个过程能够做某种任务，总会有一个相应的实现。

(5) 文档编制：

·组织：文档的组织要反映库的结构。

·综述：包括内容和结构。

·侧重面：为每一级别的用户提供一种单独的文档。

·索引：至少包括类名（按字母排序），继承的层次和关键设施。

·形式规划。

(6) 工具：

·存取工具：帮助用户往库中增加新的类。

(7) 环境：

·支持：商品化的类库应当受到支持。

·升级：采用新的范型，设计和实现，更新已有的库使 其升级。

这些特点和准则，可以作为类库建造的参考。从这些特点和准则可以看出，类的设计和类库的建立是很不容易的，需要有专门的队伍。因此，在面向对象的程序设计范型之下，程序员队伍可能要分成两种，他们都以类做为工作对象，一个队伍主要是设计类和类库，另一个队伍则是使用类，来设计应用程序。

概括起来，对于面向对象这一新的程序设计范型，我们可以说它具有一个范型所必备的条件，富有生命力，还有其他范型所缺乏或不具备的特点，如建立在类及其继承性基础之上的重用能力；可以应付复杂的大型的软件开发；便于扩展与维护，抽象程度高，因而具有较高的生产率。可以肯定地说，这种新的范型得到广泛的理解和使用以后，必将有力地推动软件开发的新的进展。

1.5　面向对象是新的方法论

80 年代，面向对象程序设计的语言趋于成熟，作为一种新的程序设计范型开始为计算社会所关注，并且为更多的人所理解和接受，这一成就促使研究者把一部分注意力转向更广、更深的层次，在面向对象分析（OOA）和面向对象设计（OOD），以及面向对象的系统开发过程等方面不断地取得进展，一种新的信息系统开发的方法论——面向对象的方法论，已经渐露头角，虽然尚不完备，但却有超越 80 年代处于全盛的结构化程序设计这一公认的方法论，引人注目。

1.5.1 分析方法

分析是信息系统开发的主要环节之一，是系统设计的前提。什么是分析？DeMarco 在他的有名的《结构化分析和系统规范》一书中是这样定义的[17]：

“分析是在采取某种动作以前，对一个问题的研究”。

Coad 和 Yourdon 在《面向对象分析》一书中，对分析的定义是[18]：

“分析是对一个问题空间的研究，导出外部可观察的行为的一个规范——一个所需要的，完全的，一致的和可行的陈述”。

第二个定义把所研究的问题扩展至问题空间，并指出分析将得出什么结果，更符合于我们现在的需要，因为在一个对象组成的系统中，它的行为特征是需要特别强调的。

现在，我们先简要地回顾一下流行的分析方法，然后对面向对象分析做一简要的介绍，以便比较。

1. 结构化分析法

结构化分析法是认识到建造一个大的、复杂的系统时，可以把子程序作为一种抽象机制的启发下加以提炼而成的。所以，结构化的分析过程是一个功能分解的过程，把一个系统看作是功能的集合，可以分解。这是早期的结构化分析法的基本思想，是目前在信息系统的开发中使用最广泛的方法。

功能分解法的着眼点在于一个信息系统需要什么样的加工方法和过程，以过程抽象来对待系统的需求。这一做法是很直接的，但使得程序员很少去注意数据结构，重点是在操作。对于一个系统来说，结构是相对稳定的，而行为则是相对不稳定的。功能分解法把基点放在不稳定的行为上，难以适应系统的变化，这是很大的缺点。

结构化分析法中另一重要的方法是数据流图法。用图去描述一个系统是一种很好的方法，但用于分析，必须克服两个重要的问题。第一是被分析的系统很复杂，如何用图形清楚地描述一个系统？第二是被描述的系统往往不是顺序的进程，而是互相有关系的许多非同步的进程所组成，怎样加以表示？

通常，采用 4 种类型的图元素来组成一个数据流图，它们是：

- 提供数据的源和使用数据的纳。
- 数据流：是数据从系统的一个部分到达另一个部分的通路。
- 变换：对数据进行处理，把数据从一种形式变为另一种形式。
- 数据存贮：数据暂时保存的设施。

数据流图是现用的结构化分析方法的核心，有的作者在定义结构化分析时，以它作为主要的内容，如 DeMarco 对结构化分析的定义是[17]：

“结构化分析是使用以下的工具：

数据流图

数据词典

结构化英语

判定表

判定树