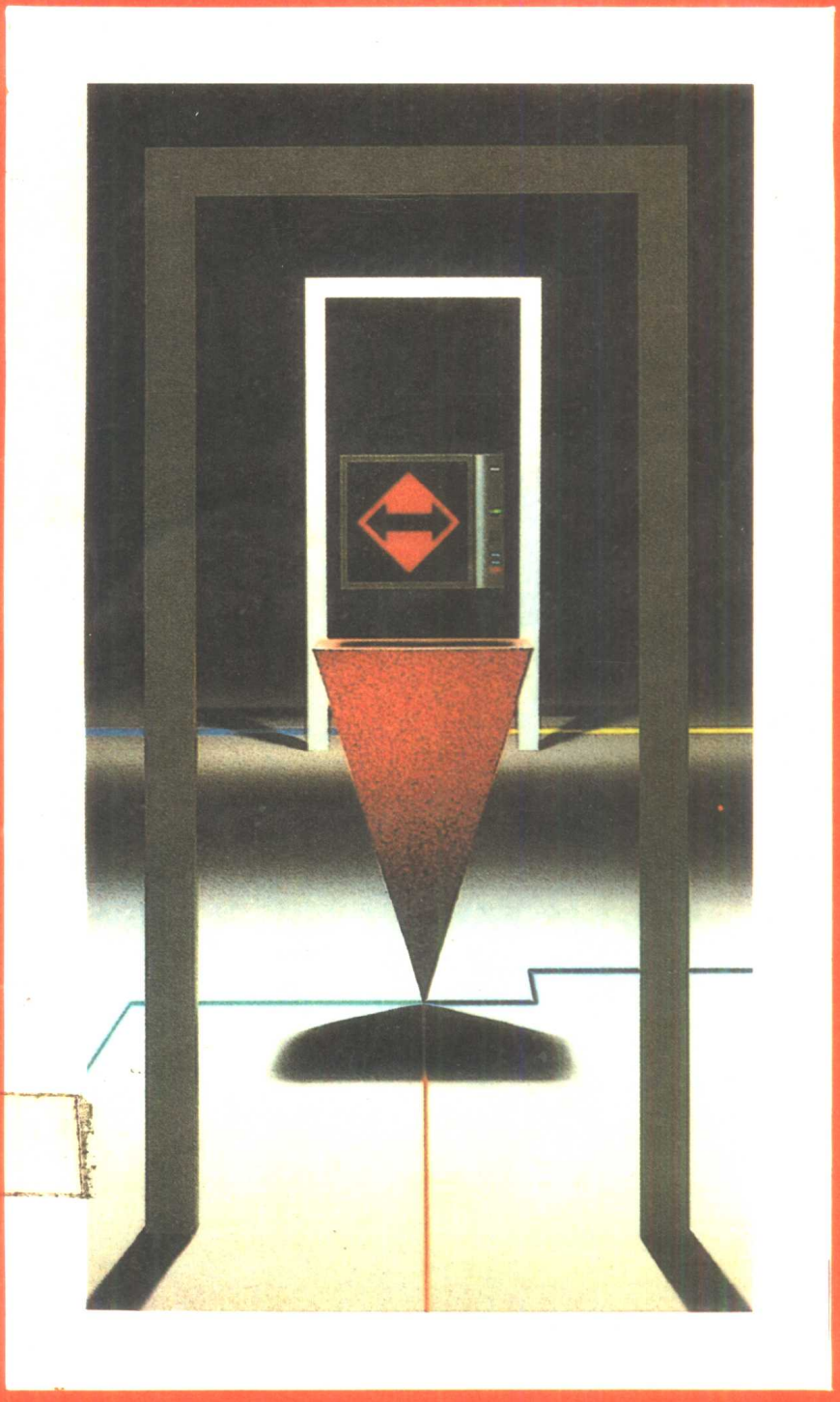
面向对象的软件设计基础

马 茜　盖 刚　张 勇　编



北京科海培训中心

面向对象的软件设计基础

马茜　盖刚　张勇　编写

北京科海培训中心

前 言

自九十年代以来，软件与硬件之间的差距至少有两代处理器之多，并且这种差距还在增大。当软件系统变得更大更复杂时，常规软件工具、技术和概念已不足以应付，从而使软件开发者陷入了困境。但面向对象的程序设计方法能够控制软件的复杂性，并使得所设计的软件质量更好。

在九十年代中，应用程序应满足更为复杂的要求，需使用更为复杂的数据结构，并且还要面对不断增多的用户。因此，软件开发者需要做出很大的努力去建立、扩展和维护庞大而复杂的软件系统；而且由于是面对大量用户的，所以软件又需具有更好的灵活性及易于使用。

我们虽已进入九十年代，但软件的开发并没有突破性进展。尽管各高等院校和研究机构在软件开发工具和技术方面取得了很大的进展，但这些进展并没有推广到软件开发者的日常开发工作中。当今的许多软件开发过程仍很落后，程序员仍用原始的方式工作。他们编出来的代码缺少一般性、可读性和扩充性。由于软件产品不能适应不断变化的环境（如硬件和操作系统的改进，或用户的要求提高）因而它们的生存期很少能达到十年以上。这个问题不在于硬件，而在于软件落后于硬件的发展。当高性能的计算机越来越多地进入人们的生活中时，缺少高质量软件的问题表现得更为明显。

面向对象技术为软件开发提供了一种新的模型。在这种新模型中，对象和类是构建块，而方法、消息和继承为基本机制。建立一个软件的传统方法是定义一组操作至数据的过程。在本书中讨论的面向对象技术则将程序设计的重点从过程转成对象。一个对象是一个自足模块，它包含数据和作用于数据的过程。对象中的过程有一个新的名字:方法。与常规程序中的被动数据不同，对象可以自行执行，也可以被来自其它对象的消息激活。具有同一用途的对象被组织到一个类中。新的类可以被建立，并且新类可以从已建立的类中继承过程和数据以使程序员能够重用已存在的类—只需设计相异之处。面向对象模型通过利用事先建立的类库，甚至事先建立的特定应用类库或称框架而提供了一种新的抽象层次。

九十年代面向对象模型的实现将使程序员和最终用户都受益非浅。虽然从面向过程的设计转向面向对象的设计似乎是从一个极端转向另一个极端，但面向对象的程序设计的确是一种更为自然的软件设计和模型化方法。若有全面的面向对象的实现，则最终用户可以修改或设计其自己的应用程序，程序员则能够设计更为复杂的应用程序。

若称面向对象软件将是九十年代的一种成熟的计算模型并不过份。当然计算机软件产业总有些爱夸大其辞，在新技术还没有推广之前总喜欢先做许多的承诺。这种过份的热情并非总是不好的。主要的问题是开发者、用户和软件产业行业何时且如何受益于面向对象软件。

具有多年面向对象程序设计经验的软件开发者会明白这个问题的关键部分在于熟悉传统程序设计语言、工具、数据库和传统程序开发规范的程序员是否愿意有所改变。现在，许多人都已经知道这个问题，并在逐渐地克服它。具有面向对象机制的现有工具、语言和数据库使得程序开发者能融合新旧开发技术，从而可受益于面向对象给软件开发带来的新的好处。

在九十年代软件开发的激烈竞争中，面向对象是一种重要的技术，其概念将会为更多的人所接受，并影响到各种技术乃至渗透到下一代的软件构造中。面向对象技术将推进软件开发过程，并将使得更新更好的应用出现。尽管面向对象还没有完全在当今的软件开发工具和应用中实现，但面向对象开发环境的思想已为许多程序开发人员所了解，并且已经在开发周期、程序设计资源以及软件换代方面起到了显著的作用。

面向对象并非是一个新的概念，实际上它已有近二十年的历史。寻其根源可追溯到六十年代的挪威，当时挪威计算中心的 Kristen Nygaard 和 Ole-Johan Dahl 开发了一种称作 Simula67 的语言。Simula67 首次引入了类、协同程序和子类的概念，这很象今天的面向对象语言。

以后，在七十年代中期，Xerox Palo Alto 研究中心(Xerox PARC)的研究人员设计了 Smalltalk 语言，该语言的每个元素都被当作一个对象来实现。Smalltalk 的程序设计环境及其相关的各种方面都是面向对象的。即使是在今天，Smalltalk 仍被认为是最纯的面向对象语言。Simula67 和 Smalltalk 的开发为今天的研究开发工作奠定了基础。Simula67 显示了基于类的程序设计语言的模型化能力以及提出了数据和操作应存放在一起的概念。

但是，面向对象的推广至今仍很慢，这种情况有许多原因。尽管 Simula67 和 Smalltalk 在学院圈子中很著名，但在八十年代以前相对来说它们却难以被主要软件开发团体所接受。例如，Smalltalk 的成果直到 1981 年 8 月因 Byte 杂志的介绍才为外界所知。许多软件开发员了解到 Smalltalk 时，都把它看做是一个窗口系统而不是程序设计技术的一次革命。忽视 Smalltalk 中面向对象程序设计技术的一个典型体现是开发出了许多使用窗口和图符却不允许用户对这些对象进行修改的界面。

八十年代，C 成为很受欢迎的程序设计语言，它不仅可用在微机上，还可用在多种结构的系统和环境中。八十年代早期，AT&T 贝尔实验室的 Bjarne Stroustrup 把 C 语言扩展为支持面向对象程序设计的 C++。有了 C++，程序员可以在其熟悉的语言环境下学习掌握面向对象的程序设计技术，而不必去探究一种新的语言和环境。

当今的个人计算机系统已满足高性能工作站、高质量图形显示和具有丰富工具的开发环境的基本要求。面向对象集成环境的引入，尤其是 1984 年 Macintosh 和 1988 年 NeXT 的推出，是推广应用面向对象技术的里程碑。

不断增加的技术复杂性是加速面向对象技术使用的加速器。面向对象为处理技术复杂性提供了一种较好的途径。使用现有开发工具建立应用程序的程序员常认为，若没有抽象的层次概念，则应用程序的开发将会受阻。

软件工业正试图尝试更好的软件开发方法。由于缺乏较好的开发过程，软件工业因此受到阻碍。可以肯定面向对象技术将缩短软件产品的开发周期、提高代码的可扩充性，并使软件产品的适用范围更广。

本书为有兴趣了解面向对象软件特征、面向对象软件对软件工业的影响以及面向对象技术在今后软件工业发展中的地位的读者提供了详细的背景介绍。本书将描述软件开发人员如何使用面向对象工具和技术，以及面向对象技术之长处。

尽管具备计算机的经验知识将会增加对本书所给示例和术语的了解程度，但并不是只有程序员才能阅读本书。对于程序员或计算机专业的学生来说，本书对面向对象进行了总体描述，并可作为有关技术读物的指南。对于那些习惯于计算机程序即是顺序排放的指令

集的过程程序员来说，本书将把他们带入一种新境界。当你读完本书，你可以不是一个面向对象程序设计员，但你将会掌握和了解：

- 从用户和软件开发者两方面来看面向对象所具有的长处；
- 经常用来描述面向对象软件结构的术语和技术的含义；
- 面向对象对基本软件构造块，包括语言、数据库和用户界面等的影响；
- 面向对象对新的和现有应用软件的影响。

面向对象初看上去似乎是一个复杂的主题，但读完本书后你将会发现并非如此。我们知道本书读者将具备各种背景和动机，所以，对于非程序员读者，我们提供了面向对象的总体描述，说明了面向对象是如何应用到整个系统环境中以及如何影响应用程序的；对于程序员，我们给出了面向对象软件设计和结构的基本内容描述。对于所有的读者，我们都描述了面向对象技术将会对软件市场的影响。

为了满足各种用户的需要，我们推荐三种阅读方法：

方法 1：仔细阅读第一部分，然后只阅读第二部分中每一章有关效益的内容，最后再通读第三部分的各章。想对面向对象有个大概了解，并且没有计算机经验知识的读者适合于本方法。该方法着重介绍面向对象的基本术语、优点和应用，它还提供了软件开发者和用户使用面向对象方法的总体描述。

方法 2：从第一部分开始，但可以只是大概浏览一下，然后详细阅读第二部分的各章，最后再阅读第三部分的各章。具有一些程序设计经验并希望研究某些代码的读者适于本方法。该方法强调了面向对象的基本术语、优点和应用，并且还探讨了面向对象对计算机语言、数据库和用户界面的意义。

方法 3：直接从第二部分开始，但可不时回过头来翻阅一下第一部分，然后进入第三部分。那些希望全面了解面向对象技术的一丝不苟的程序员适于本方法。该方法着重于软件综述，面向对象软件开发的描述，以及对应用途径的讨论。

小结

- 面向对象是针对软件设计的传统方法的一种变革。
- 传统方法是把主动过程作用到被动数据。面向对象方法把过程和数据放在一起。
- 面向对象技术可以应用到大多数软件组件中，包括程序设计语言、数据库和界面。
- 使用面向对象技术可使得软件产品易于扩展、维护、使用且灵活性很好。
- 本书的读者可以具有不同的背景和目的，阅读时最好为自己选择一种适当的方法。

目 录

前言 …… 1

第一部分 基本概念 …… 1

第一章 综述 …… 1

1.1 引言 …… 2

1.2 动力 …… 2

1.2.1 多媒体信息 …… 3

1.2.2 最终用户计算 …… 4

1.2.3 分布式处理 …… 5

1.3 未来 …… 5

1.4 近期进展 …… 8

1.4.1 Apple Macintosh …… 9

1.4.2 NeXT …… 10

1.5 九十年代的对象体系结构 …… 11

1.6 小结 …… 12

1.7 要点 …… 12

第二章 面向对象的基本思想 …… 14

2.1 基本机制 …… 15

2.1.1 对象 …… 15

2.1.2 消息与方法 …… 16

2.1.3 类、子类与对象 …… 17

2.1.4 继承 …… 19

2.2 主要概念 …… 20

2.2.1 封装 …… 20

2.2.2 抽象 …… 20

2.2.3 多态性 …… 20

2.2.4 持久性 …… 20

2.3 有关技术术语 …… 20

2.3.1 动态联编 …… 20

2.3.2 可视编程 …… 21

2.3.3 BLOB …… 21

2.4 传统方法与面向对象方法 …… 21

2.5 小结 …… 22
2.6 要点 …… 22

第三章 面向对象方法的效益 …… 23
3.1 复杂性的维护 …… 25
3.1.1 软件开发中的灵活性 …… 25
3.1.2 可重用性 …… 25
3.2 生产率的提高 …… 26
3.2.1 可扩充性与可维护性 …… 26
3.2.2 用户编程 …… 27
3.3 小结 …… 27
3.4 要点 …… 27

第二部分 面向对象的软件概况 …… 28
第四章 语言 …… 28
4.1 面向对象语言的历史 …… 30
4.2 面向对象语言的优点 …… 32
4.3 面向对象语言的功能 …… 34
4.3.1 对象和类 …… 35
4.3.2 方法和消息 …… 36
4.3.3 继承 …… 36
4.3.4 多态性 …… 38
4.3.5 动态联编 …… 39
4.3.6 多继承性 …… 39
4.3.7 类库 …… 41
4.3.8 开发工具 …… 43
4.4 面向对象语言 …… 43
4.4.1 C++ …… 44
4.4.2 Objective-C …… 46
4.4.3 面向对象 Pascal …… 48
4.4.4 Smalltalk …… 50
4.4.5 Eiffel …… 52
4.4.6 Common Lisp Object System …… 55
4.5 有关技术问题 …… 56
4.5.1 动态联编与静态联编 …… 57
4.5.2 作为对象的类 …… 57
4.5.3 并发性 …… 58
4.5.4 标准 …… 58
4.6 小结 …… 58

4.7 要点 …… 59

第五章 数据库 …… 60
5.1 面向对象数据库的历史 …… 61
5.2 面向对象数据库的优点 …… 65
5.3 面向对象数据库的功能 …… 67
5.3.1 概述 …… 67
5.3.2 对象 …… 68
5.3.3 方法 …… 70
5.3.4 继承性 …… 70
5.3.5 类库 …… 71
5.3.6 永久性 …… 72
5.3.7 查询 …… 72
5.3.8 完整性 …… 73
5.4 有关技术问题 …… 74
5.4.1 查询优化 …… 74
5.4.2 分布式数据库 …… 74
5.4.3 并发性 …… 75
5.4.4 性能 …… 76
5.4.5 模式修改 …… 76
5.4.6 语言支持 …… 76
5.4.7 标准 …… 77
5.4.8 应用程序 …… 77
5.5 小结 …… 77
5.6 要点 …… 77

第六章 用户界面 …… 79
6.1 面向对象用户界面的历史 …… 80
6.2 面向对象用户界面的优点 …… 81
6.3 面向对象用户界面的功能 …… 83
6.3.1 Windows 和 Presentation Manager …… 85
6.3.2 X Window …… 88
6.3.3 NeXTStep 的界面建立程序 …… 91
6.3.4 Apple Macintosh …… 93
6.4 面向对象用户界面的开发工具 …… 95
6.4.1 Smalltalk …… 95
6.4.2 NewWave …… 96
6.4.3 Caseworks …… 98
6.4.4 CommonView …… 99

6.4.5 NeWS …… 100
6.4.6 Metaphor …… 100
6.5 小结 …… 100
6.6 要点 …… 101

第三部分 面向对象软件的设计 …… 102
第七章 分析和设计 …… 102
7.1 历史 …… 103
7.1.1 结构化系统分析 …… 103
7.1.2 面向对象分析 …… 104
7.2 效益 …… 105
7.3 面向对象设计 …… 106
7.3.1 标识和定义对象 …… 106
7.3.2 定义和组织类 …… 109
7.4 方法学现状 …… 111
7.5 小结 …… 122
7.6 要点 …… 122

第八章 程序设计和维护 …… 123
8.1 过程程序设计和维护 …… 126
8.2 面向对象程序设计和维护 …… 127
8.3 面向对象的项目管理 …… 132
8.4 开发工具和环境 …… 134
8.4.1 浏览器 …… 134
8.4.2 检查器 …… 136
8.4.3 分析器 …… 137
8.4.4 调试器 …… 137
8.4.5 开发环境 …… 139
8.5 小结 …… 140
8.6 要点 …… 140

第九章 应用程序的现状 …… 142
9.1 面向对象应用的效益 …… 143
9.1.1 更大的灵活性 …… 143
9.1.2 透明的集成性 …… 143
9.1.3 使用的方便性 …… 144
9.2 面向对象应用软件的功能 …… 144
9.2.1 从模块到仿真再到实现 …… 144
9.2.2 从集中式计算到分布式计算 …… 145

9.2.3 从正文和图形到多媒体 …… 145
9.2.4 从类似对象到面向对象 …… 146
9.3 面向对象应用的例子 …… 149
9.3.1 计算机辅助软件工程 …… 149
9.3.2 计算机辅助教学 …… 150
9.3.3 计算机集成制造 …… 151
9.3.4 计算机辅助印刷 …… 151
9.3.5 可视程序设计环境 …… 152
9.4 小结 …… 154
9.5 要点 …… 154

第四部分 附录 …… 155
A. 术语 …… 156
B. 缩写 …… 162

第一部分　基本概念

第一章　综述

本章给出九十年代面向对象软件的总体描述。我们将指出推动面向对象技术普及的动力，这些动力包括对多媒体信息、最终用户计算和分布式系统的需求。我们还将讨论面向对象语言、数据库、用户界面、操作系统和应用程序的进展。

1.1 引言

经过几年的较为沉寂，面向对象技术已开始为许多软件开发者和用户所接受。许多软件都同时出现面向对象倾向，其中包括程序设计语言、用户界面、数据库和操作系统。尽管面向对象程序并非是万能的，但它确实能帮助控制软件开发中不断增加的复杂性和费用。九十年代的面向对象与七十年代的结构化程序设计的目的是相同的，即改进软件的结构、维护和使用。面向对象技术将从根本上改变程序员的工作方法，加快产生下一代应用程序的速度。面向对象技术还能增强应用程序的功能，使最终用户能通过各种计算平台对现有数据类型和扩展数据类型进行访问。

研究机构使用面向对象系统的历史已有十五年之多。面向对象不象许多其它新技术那样仅仅局限在研究机构中，它已为大多数软件领域所接受。在商用软件公司中，成千上万的程序员已经在使用面向对象语言和程序设计工具。

面向对象技术的长处已经越来越为人们所发现。随着对 Pascal、C 和 COBOL 等流行语言的面向对象的扩展，面向对象语言的标准已开始逐渐形成。

面向对象的开发工具已开始用来简化面向对象的设计、编码和调试。用于构造面向对象用户界面的开发环境也已投入使用。这些用户界面环境使得交互性应用程序可从预定义类库中快速生成，从而大大加快了开发速度。

并非九十年代将出现的面向对象体系结构的所有成分都能够在目前就为人们所使用。真正应用上的成功要依赖系统软件、语言、工具、数据库和预定义对象库(类库)的良好配合。目前，还没有一个面向对象的数据库得到广泛的承认。在通常使用的计算机体系结构中，还不能支持面向对象的操作系统。尽管程序员已能使用面向对象程序设计环境，但很少有工具能为最终用户提供编程的能力。图 1.1 给出了对面向对象的展望。

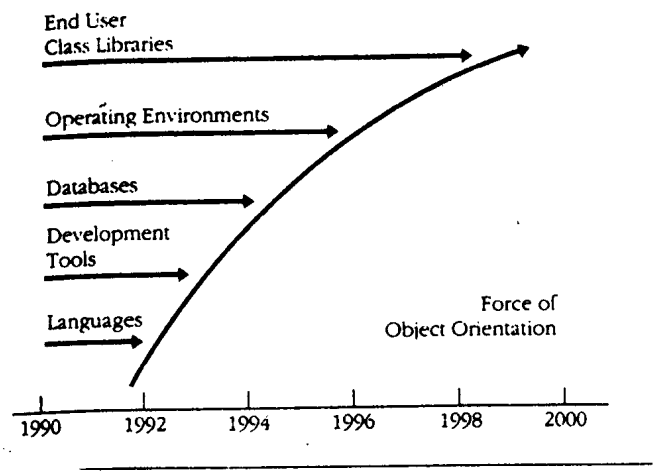


图 1.1　面向对象展望

1.2 动力

面向对象的程序设计和面向对象系统的开发工具及体系结构正在迅速地改变程序设计领域。软件界接受面向对象技术的原因有许多。不断增加的复杂性、多样性和相互关联性

是九十年代大多数软件系统的共同特征。企业间的网络系统需要处理许多不同类型的信息，这些信息来自网络中的各个节点。人们对支持多媒体信息的开发工具、系统环境、应用程序、最终用户计算以及分布式处理的需要是面向对象技术发展的主要动力。

1.2.1 多媒体信息

现今软件系统的复杂性正在不断地增加。这不仅表现在功能上，而且也表现在数据类型上。未来的计算机系统不仅需要处理文本和数据，而且还需要处理图象、声音和视频信息。如图 1.2 所示，软件开发过去面对的问题是改善性能和提高功能，但将来面对的将是控制不断增长的复杂性。

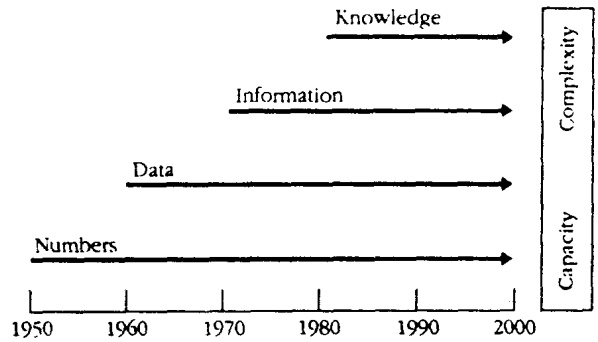


图 1.2　不断增加的软件复杂性

编程现在已超越了面向记录，而是要能够操作更丰富更复杂的数据类型。许多应用领域如计算机集成制造(CIM)，计算机辅助设计(CAD)，计算机辅助软件工程(CASE)和计算机辅助出版(CAP)都已加强了现今程序设计和软件系统结构的能力，这是由于它们对仿真、真实世界表示、操作和各种数据类型间复杂的相互关系的需求所致。图 1.3 给出了一个典型系统中数据组成的变化。

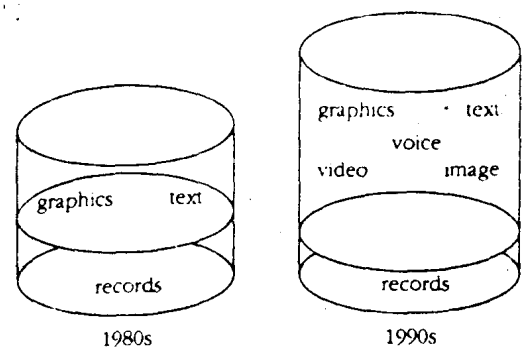


图 1.3　九十年代软件体系结构中面向记录所占比例

许多其它的通用应用领域也开始要求能够支持复杂的数据类型。九十年代的个人计算

机硬件将会经历提供集成多媒体支持能力的变化。到 1992 年，应出现具有标准多媒体支持的个人计算机，其特征可能包括数字信号处理(DSP)以能对声音或音乐应用程序中的数据加以操作、集成化的 CD／ROM 设备、活动视频、扩充内存和增强型彩色图形。早在 1989 年，随着 NeXT 机器的出现，上述许多特征都已进入了市场。表 1.1 给出了九十年代多媒体个人计算机的典型结构。

表 1.1　基本多媒体计算机配置

	1990	1992	1994
内存	＞1MB	＞4MB	＞8MB
图形	512＊512 象素	1024＊1024 象素	1024＊1024 象素
	4 位颜色	8 位颜色	32 位颜色
		有限的活动视频	全活动全屏幕数字视频
数字信号处理	无数字信号处理	带声音数据的处理	带声音数据的处理
存贮	非集成的 CD／ROM	集成的 CD／ROM	集成的 CD／ROM
图形用户界面	窗口界面	窗口界面	面向对象窗口界面
工具	有限的多媒体工具	超级卡片书写系统	集成的面向对象多媒体工具包

这些多媒体计算机不仅预示系统需要能够处理多种数据类型，而且还预示着需要有支持多媒体系统的专门开发者和工具。

九十年代的计算机将能产生一种声音-视觉环境。这种能力将使得用户可以感知应用中的实体。面向对象的 CAD 应用已将这种模拟能力加到系统中，从而使得带有特殊眼镜和手套的用户能够操作三维对象。

正如我们将在以后各章中详细描述的那样，面向对象提供了对各个工作小组建立其各自的对象的支持。它允许不同的对象对同一个请求或消息作出不同的反应。

1.2.2　最终用户计算

不仅应用程序在不断地变得复杂，而且用户对控制应用程序做什么的要求以及用来建立可维护应用框架扩展的原型工具和开发规范的要求也越来越强烈。这些方面已经取得了一些进展。例如，改进的原型工具（如第四代语言），图形程序设计环境（如 Apple 公司的 Hypercard），宏语言（如电子表格程序）都在一定程度上满足了用户的要求。

几乎所有的软件公司都知道用户对大规模系统的要求是无止境的。庞大的软件系统还伴随着硬件及操作环境的种类繁多。若一直用以前的软件开发方法，则程序员的数量永远也满足不了九十年代成千上万的用户需求。

首先，对于软件开发者和用户来说，面向对象的程序设计为基于别人的工作而建立实际的应用程序提供了机会。在面向对象开发环境中开发用户界面的速度将会较快，因为开发者和用户都是用已存在的对象来构造应用程序的。为一个应用程序建立的对象可作为别的应用程序的构造块。当一个软件开发者建立了一个字处理器后，该字处理器就可被重用

于建立一个电子邮件编辑器。同样，翻滚条、菜单和对话对象可重用于不同的用户界面。当一个开发者建立了一个面向对象的声音驱动器时，它也可重用于声音邮件。面向对象的体系结构能够简化对应用程序的集成，从而减少了需要用来建立计算环境的基本应用程序(对象)的数量。只要有几个核心的面向对象的应用程序就可以了：用来建立应用程序的用户调色板的功能是很强的。

很受欢迎的用户程序设计工具，如 Apple 的 Hypercard(可认为是类似对象的而不是纯面向对象的)表明了若提供适当的环境，非程序员也可以建立应用系统。除了 Hypercard 外，完全面向对象的应用程序也显示了这一特征。一个应用程序要承担所有事务的观念将被取代，将来的应用程序将由模块组成，其功能将通过面向对象系统的基于消息的机制来体现。

1.2.3 分布式处理

对于分布式体系结构，用户要求具有联结性、相互可操作性和穿越系统软件、数据格式和应用程序的通信工具。并且由于工作站逐渐减少了对共享处理器的依赖，不断变得复杂的组织结构和应用领域也使得数据共享变得更加重要了。

在九十年代，局域网(LAN)将逐渐丧失其独有的优势而将向广域网(WAN)发展。局域网主要结构的能力也将不断发展，包括支持多媒体数据传输的光纤分布式数据界面。

这些网络还将被更强壮的结构所支持。综合业务数字网络(ISDN)提供端对端联接以能处理各种形式的数字通信业务、数据交换、电子邮件、传真和可视电话。

九十年代还要面对的挑战是要如何提供非协作操作程序间的通信——不仅是在定义完备的环境中，而且包括通过各种计算环境构成的适用于各种应用领域的网络。

1.3 未来

面向对象技术能使复杂系统的开发变得清晰和灵活。当今的应用程序不能提供一致性和灵活性，而它们又是一个高效的编制环境所需要的。面向对象软件能解决这个问题，因为它可提供远比大家所熟悉的窗口界面要多的功能。应用程序的相互操作性将远远超过现今使用的剪裁板。面向对象技术提供的环境能实现应用程序间的通信以及很容易地在分布式的各种结构中移动。

在不久的将来，面向对象将使下列三种人员受益：用户、一般商用软件程序员和系统软件开发者。系统软件开发者将受益最多，他们将利用这种开发方法和发展中的工具来实现越来越复杂的软件。面向对象技术也将开始影响一般商用软件程序员和用户。图 1.4 给出了面向对象所产生的影响。

有些惯性将会继续存在。程序员的自我中心、对传统方法的熟知以及新软件和硬件购置的经济性考虑都在维护着手工式软件。用面向对象方法建立软件的过程对于习惯使用自己的工具和风格的程序员来说是一个挑战。这只是影响到手工式软件开发人员的变化之一。另外，程序设计员还将被分为对象生产者和对象消费者。对象生产者将提供专门领域中的对象库，对象消费者将汇编已有对象库，加上不多的一些自己建立的对象以建立实际的应用程序。

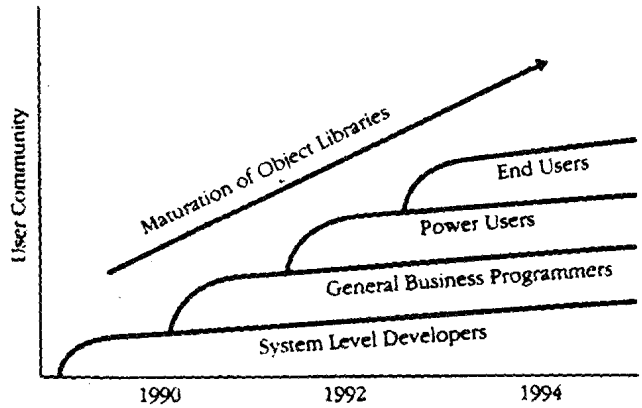


图 1.4　对象库的成熟程度与用户的关系

当前软件规范可能很快就会过时，对许多用户来说很熟悉的图形含义将被保留，但被动的图符将为主动图符所替代。这些主动图符可被认为是智能代理，它们执行象路由电子邮件或填充时间表这样的功能。在许多应用中，这种代理的工作过程是看不见的。将来的对象库将使得应用程序能很容易地从现有构造块中建立起来。当今库中的许多按钮操作、过程和文本都是静态的，并且是不可修改的。象 Apple 公司的 Hypercard 和 Lotus 的 Macros 这样的系统的用户可编程性尽管与将来的发展相比还有一些局限，但已使我们看到在适当的环境下用户可建立他们自己的集成化应用程序的前景。

将来的人机界面将是让用户把所有可见的元素(图符)看作对象，这些对象将直接由鼠标和键盘操作，并完全为其内部的对象系统所支持。这种应用程序设计如同卡耐基·梅隆大学的 Andrew 系统一样，主要是作为研究室的原型。当前准备一个复合文档由多个步骤组成：打开专用应用程序，将其剪切到剪裁板，关闭该应用程序，打开另一个应用程序，再把剪裁板上的内容剪贴上去。一个图形可以通过修改它所基于的电子表格、重复剪切和剪贴而得到更改。有了面向对象的复合文件，则可以插入图形。对一张电子表格的修改将自动反映到字处理文件中保留的图形上。面向对象文件系统将支持各应用程序并提供对对象的透明管理。

图 1.5 给出了可用于早期类似面向对象系统中对象管理的例子。在该例子中，用户用一个字处理器编辑一份稿件。为了修改一张嵌入图表，用户只需选择编辑图表。在相应的画图应用程序中开一个窗口，在该应用程序中对图形对象的操作改变了该对象在两个应用程序中的特性。当前的此类商品，如 HP 的 NewWave，提供了这类相互可操作性。但在程序间的移动对用户来说仍是不透明的。

图 1.6 给出了同一个应用程序，但它具有一个面向对象文件系统所提供的较高的应用透明性。一个对象，如本例中的 DrawApp 对象，携带着其功能。用户不再需要启动和退出程序。当需要时，相应功能便出现。并且，如在前面的例子中，对图表的修改也将被自动地保存在它出现的其它地方。

一个面向对象系统的可视对象将支持这种直接的修改类型，并且将在很大程度上与应用程序相独立。这是由传统系统到面向对象系统迈进的主要一步。当今，程序员或用户必

须进入及退出文件才能获得这种相互可操作性。用户必须打开文件，选择信息，或者还要进行一些修改，然后才能把相应信息传送到下一个应用程序中。

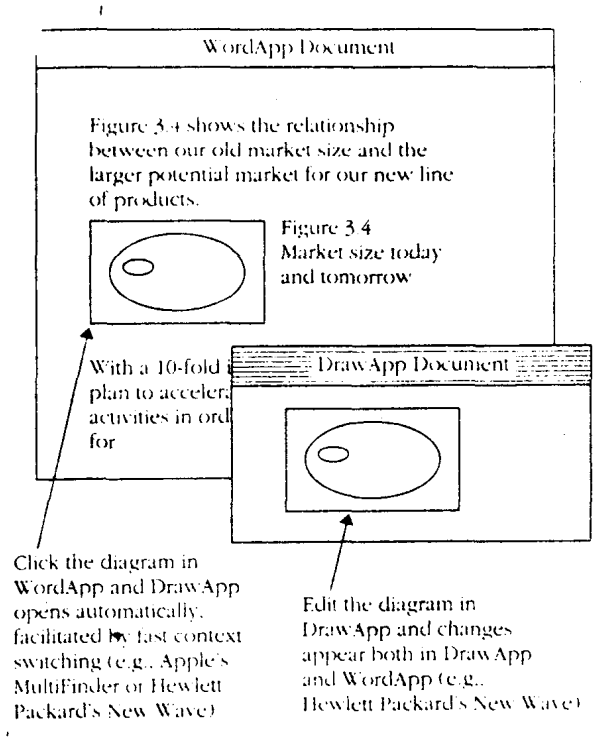


图 1.5　发展中的相互可操作性

在一个面向对象系统中，复杂的数据结构被当作由内部操作环境来自动管理的单个对象。用户可以连接、组合和操作复杂的数据，如文本、图形、映象和活动视频。并且，还提供了一种用来浏览或查看复杂对象而不是实际运行应用程序的机制。在我们的例子中，字处理程序只可为不太一般的任务所访问。封装过程对提问来说是不可见的。除此之外，面向对象应用框架鼓励用户把对象看作是用户自己可以进行编程的软件机器人。

与所有的系统软件构成一样，标准是很重要的。尽管实际上已有一些标准进入了语言领域，但由于开发方法与系统结构的很大变化，标准的设置上还存在很大的问题。面向对象标准可能还会导致软件工业结构的改变。软件工业的统治者将会是能够控制用于对象消息通信的应用程序界面(API)的公司。

随着标准的建立，在软件工业本身中也需要建起一个里程碑。关键是要建立面向对象的体系结构标准。应用投资商对面向对象结构的特定实现是不会满足的。软件工业中的竞争者必须能够共享标准以提供对象间的相互可操作性。

商用软件开发者将必须接受这些标准（或许包括嵌入到所有人对象库中的主要可视对象）。对当前应用程序间通信的复杂性的解决将决定于对这些标准的接受。由于这些原因，语言和数据库的强壮性都需要不断改进以完全实现面向对象。

面向对象系统组成、开发工具和标准的实现的意义远超过了它们本身。将来的用户只需利用所需的对象来灵活地设计自己的应用程序。用户建立的对象可以进行交易。有了对