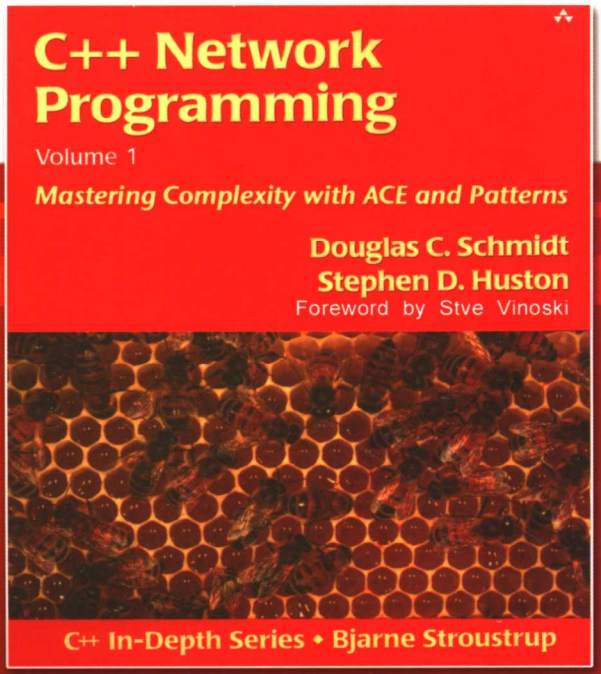
C++网络编程 卷1

运用ACE和模式消除复杂性

C++ Network Programming Volume 1

Mastering Complexity with ACE and Patterns



Douglas C.Schmidt 著
Stephen D.Huston
於春景 译
马维达 审校

华中科技大学出版社

C++网络编程

卷1

C++网络编程 卷1

C++Network Programming Volume 1

Douglas C. Schmidt Stephen D. Huston

图书在版编目(CIP)数据

C++网络编程 卷1/(美) Douglas C. Schmidt Stephen D. Huston 著;於春景 译

武汉:华中科技大学出版社,2003年12月

ISBN 7-5609-3066-2

Ⅰ.C++…

Ⅱ.①D… ②S… ③於…

Ⅲ.C语言-程序设计

Ⅳ.TP312

责任编辑:曾 光

出版发行:华中科技大学出版社 (武昌喻家山 邮编:430074)

http://press.hust.edu.cn

录 排:华中科技大学惠友科技文印中心

印 刷:湖北新华印务有限公司

开 本:787×1092 1/16 印 张:20.75 插 页:2 字 数:308 000

版 次:2003年12月第1版 印 次:2003年12月第1次印刷

ISBN 7-5609-3066-2/TP·517 定 价:35.00元

深入 C++丛书

Bjarne Stroustrup，编者

“我把这封信写得比平时要长，因为我没有时间去把它写得更短。”

——BLAISE PASCAL

对 C++程序员来说，ISO/ANSI C++标准的问世标志着一个新纪元的开始。C++标准为程序设计提供了很多新的便利和可能，但要想在如此众多的信息中挖掘到其中的精髓，作为现实世界中的程序员，我们缺少足够的时间。“深入 C++”丛书就是针对一定的主题，为程序员提供了简明扼要的指导，从而将学习时间和疑惑减至最少。

在这套丛书中，每一本书都呈现一个独立的主题，每本书的技术难度也和相应的主题相适应。本丛书所提供的有效途径，有助于每一位专业人员将自己的程序设计水平提升到更高的层次。这些专著由业内专家撰写，篇幅短小但内容深入。在阅读和参考这些专著时，我们不会被无关的内容所打搅。作为一个系列，这些书相互之间交叉引用；此外，它们也都引用了 Bjarne Stroustrup 的著作《The C++ Programming Language》。

在提高 C++程序设计能力的过程中，日益重要的一点是将重要的知识同华而不实的东西分离开来，然后找出更深入、更全面的内容，从而获得进一步的提高。“深入 C++”丛书所展示的 C++程序设计的工具、概念、技巧和新的方法，将为你提供一个重要的开端。

丛书书目

Accelerated C++: Practical Programming by Example, Andrew Koenig and Barbara E. Moo

The Boost Graph Library: User Guide and Reference Manual, Jeremy G. Siek, Lie-Quan Lee, and Andrew Lumsdaine

C++ In-Depth Box Set, Bjarne Stroustrup, Andrei Alexandrescu, Andrew Koenig, Barbara E. Moo, Stanley B. Lippman, and Herb Sutter

C++ Network Programming, Volume1: Mastering Complexity Using ACE and Patterns, Douglas C. Schmidt and Stephen D. Huston

Essential C++, Stanley Lippman

Exceptional C++: 47 Engineering Puzzles, Programming Problems, and Solutions, Herb Sutter

Modern C++ Design: Generic Programming and Design Patterns Applied, Andrei Alexandrescu

More Exceptional C++: 40 New Engineering Puzzles, Programming Problems, and Solutions, Herb Sutter

欲获取更多信息，请访问“深入 C++”丛书网站 http://www.aw.com/cseng/series/indepth/

C++网络编程

卷1

运用ACE和模式消除复杂性

C++ Network Programming

Volume 1

Douglas C. Schmidt　Stephen D.Huston　著

於春景　译

马维达　审校

华中科技大学出版社

中国·武汉

序

为本书作序之时，我正在欧洲畅游。这多亏了欧洲完善的公共交通设施。作为一个美国人，我不禁被这里的基础设施所倾倒。无论在什么地方，只要我一下飞机，就可以方便地找到火车或巴士，享受快速、整洁、安全、准时的服务——或许，更重要的是：将我直接送到目的地。此外，离站和到站的通告用多种语言表示，各种标志和指示也易于理解，即使是我这样一个不懂当地语言的人，也没有感到丝毫不便。

我生活、工作在 Boston。和大多数美国人一样，我几乎完全靠自己的汽车辗转于各个场所。除了偶尔光顾一下 Boston 的地铁系统之外，我总是自己开车外出；因为公共交通设施有太多的限制，它很难将我送到目的地。数百万 Boston 居民以及其他地方的人们也都处于同样的困境。可见，我们的公路系统已经过时，远不能承受今天的交通量。我知道，如果仔细算一算自己一生中因为塞车而浪费的时间，一定会吓一跳。

有趣的是，网络计算系统（networked computing system）和交通系统之间有某些相似之处，其中最显著的一点是：二者的成功都离不开“可伸缩（scalable）”的基础设施。所谓可伸缩的交通系统，不只包括那些显而易见的设施，如火车和铁轨、飞机和机场，还需要有调度、路线选择、维修、检票、监控，而这一切还要能够随着物理上的交通系统的变化而伸缩、变化。类似地，网络计算不仅需要主机和网络——物理上的计算和通信基础设施——还需要软件上的调度、路由选择、分发、配置、转译、验证、授权和监控——只有这样，网络系统才能根据需要调节、变化。

说到基础设施，有这样一个具有讽刺意味的事实：建设好基础设施，这本身已经非常之难；但更难的是，只有让用户越感觉不到它的存在，人们才会认为它越成功。举个例子：瑞士境内的阿尔卑斯山脉地势崎岖，但是，为数不多的设计师、工程人员和建筑工人却凭借他们的技能，为无数的瑞士居民提供了有效的交通系统，从而给他们的日常生活带来了便利。事实上，这套系统如此可靠易用，以至于你很快将它的存在当成是理所当然之事。它因此变得对你透明了。譬如，当你踏上瑞士的铁路之旅时，你的心思只会放在该从哪一个地方到达另一个地方上，而不会关心是什么样的交通工具将你载到那儿。除非你是个游客，否则，你大概不会注意到你正在穿越一个历经数年设计、修建的隧道，也不会注意到火车正攀行在如此陡峭的斜坡上，更不会注意到铁路上还安装了齿轨，以帮助列车爬行。铁轨设施完美地承担了它的职能，所以，你甚至没有注意到它的存在。

本书针对网络计算系统，探讨了一种基础设施软件（infrastructure software），通常称为*中间件（middleware）*。之所以称之为“中间件”，是因为它就像是“沙漏的腰（waist in the hourglass）”，位于操作系统和网络之上、应用程序之下。中间件具有多种形态（shapes）、规模（sizes）和能力（capabilities）：从 J2EE 应用服务器、异步消息处理系统、CORBA ORB，到小型嵌入式系统中的 Socket 监控软件。中间件必须支持日益繁多的应用程序、操作系统、网络协议、编程语言和数据格式。如果没有中间件，要想应对网络计算系统中日益增长的多样性（diversity）和异种性（heterogeneity），将十分麻烦，而且容易出错且代价昂贵。

中间件种类繁多，要解决的问题也各式各样，但在解决各种复杂难题时，不同类型的中间件往往采用相同的模式（patterns）和共同的抽象（abstractions）。例如，如果去窥探一个具有可伸缩性和灵活性的应用程序服务器、消息系统或 CORBA ORB 的内部，就会发现，它们采用了类似的技术来完成诸如连接管理、并发、同步、事件多路分离、事件处理程序分发、错误记录、监控等任务。如同瑞士铁路系统的用户数量远远大于设计和建造铁路的工程人员的数量那样，一套成功的中间件的用户数量也远远大于设计和构造中间件的开发人员的数量。如果要设计、构造或使用中间件，那么，只有熟悉、理解和运用这些常见模式和抽象，才会获得成功。

虽然很多人都知道，中间件必须具有可伸缩性和灵活性，但很少有人能够高效地提供这种中间件。然而，Doug Schmidt 和 Steve Huston 在本书介绍的 ADAPTIVE 通信环境（ACE）中却做到了这一点。ACE 是一套应用广泛的 C++工具包，它汇集了很多常用的模式和抽象，这些模式和抽象在各种极为成功的中间件和网络程序中都有广泛的应用。ACE 已经成为许多网络计算系统的基础：从实时的航空电子应用到 CORBA ORB，以及对主机“端对端（peer-to-peer）”通信的支持。

和其他所有优秀的中间件一样，ACE 也将异种环境下各式各样的复杂性隐藏在底层；和其他众多基础设施中间件不同的是：ACE 能够在程序需要的任何地方提供最大的灵活性，但不会损及系统的性能或可伸缩性。作为一名长期从事中间件开发工作的设计师，我清楚地知道，要想在同一个软件包中兼顾性能和灵活性，难度该有多大！

但从某种意义上说，ACE 在灵活性和性能上的优越表现没有出乎我的意料。和 Doug 的长期合作让我清楚地知道，他无愧于这个领域的先锋。当今各种具有可伸缩性、高性能和灵活性的中间件，都明显地留下了他的痕迹，并深受他的影响。Steve 是一位天才的 C++开发者和作家，他的工作使 ACE 多年来取得了长足进步。Doug 和 Steve 联手，为所有设计、构造、甚至使用中间件的人奉献了一部“必读”之作。万维网（World Wide Web）和互连嵌入式系统（interconnected embedded systems）的日益普及，意味着网络计算系统的数量、规模和重要性将持续增长。只要理解了 Doug 和 Steve 在书中讲述的重要模式、技术、class（类）和经验教训，我们就有希望创建出完全透明、高效、可靠的中间件基础设施。

Steve Vinoski
总设计师兼副总裁，Platform Technologies
IONA Technologies
2001 年 9 月

译　　序

计算机网络的普及，给软件开发者带来更大的挑战。硬件设备、软件环境的多样性（variety）和异种性（heterogeneity），使得网络程序设计的复杂性（complexity）大大提高。应用程序作为软件功能的最终实现者，不应当（有时甚至不可能）去直接应对这所有的复杂性。因此，一套专门处理“多平台”差异和编程复杂性的中间件（middleware），对网络程序开发具有重要意义。ACE 就是这样一套优秀的中间件。

本书是 ACE 的缔造者撰写的一部网络中间件专著。它以 ACE 的设计为核心，深入探讨了网络中间件开发中面临的各种挑战，以及设计上的各种选择，并为你展示了 ACE 针对这些问题的解决之道。

本书适合 ACE 的使用者或者（潜在的）ACE 的开发者阅读。本书对 ACE 的能力、ACE 要解决的问题，以及如何通过 ACE 提高网络程序的可伸缩性和可移植性等议题作了详尽的论述。对 ACE 使用者来说，本书是迄今为止最为权威、详尽、系统的 ACE 专著。

但是，本书并不只针对 ACE 的使用者，任何一位网络程序开发者——无论应用程序开发者或中间件开发者——都能从本书受益。作者在书中就网络编程领域存在的复杂性作了全面、细致的阐述，对各种设计方案的利弊作了深入的剖析，并给出了 ACE 的解决之道。这些内容，是 ACE 的开发者十多年来经验的积累，任何一位网络开发者都能从中汲取养分，增添功力。

对 C++程序员和模式爱好者来说，本书也不容错过。ACE 运用了 C++面向对象设计技术，也使用了模板、traits（特征类）等高级语言特性和其他 C++技术；此外，还运用了大量的设计模式。长期以来，高级 C++特性和技术在实际软件开发中的运用、

设计模式的实践性等问题，给不少编程爱好者带来很多困惑和疑问。本书通过现实世界的产品级代码，就这一问题作出了清晰而明确的回答。

虽然本书并不是一本网络编程、C++或设计模式方面的教程，但在阅读本书之前，必须对这三方面的知识有所了解。此外，本书具有很强的实践性，它是作者十多年 ACE 实践经验的总结。就我看来，这也是“深入 C++”丛书中最具实践性的专著之一。

最后要说明的是，《C++网络编程》共两卷，本书是第一卷。这使得我在翻译过程中能有机会同第二卷的译者马维达先生进行多次深入的探讨。我十分高兴能和他共同交流学习。我也期望我们的工作能给各位读者带来帮助。

於春景，2003/04/22

深圳蛇口，海上世界

关于本书

过去 10 年，并发式“面向对象”网络编程（concurrent object-oriented network programming）已经成为一种有效的应用软件开发范式（paradigm）。在这些应用程序中，相互协作的对象可以：

1. 在一个进程或计算机中相互关联；
2. 分布在一组通过网络（如嵌入式互连系统、局域网（LAN）、企业内部网、互联网）相连的计算机中。

如果对象是分布（distributed）存在的，那么，组成这些对象的各种实体（entity）之间就要能够有效地通信和协调。而且，当应用程序在其生命周期内发生变化时，也必须始终这样。对象的布局、现有网络设施、平台并发机制的选择都允许有一定程度的自由，这种自由带来了强大的功能，同时也带来了挑战。

如果设计合理的话，并发式“面向对象”网络编程将提供强大的功能，从而大大提高应用程序的灵活性。例如，根据项目的需求和现有资源，可以使用：

- 实时（real-time）、嵌入式（embedded）或手持（handheld）式系统；
- 个人或膝上型电脑；
- 各式各样、不同规模的 UNIX 或 Linux 系统；
- “大型”主机甚至超级计算机。

然而，当你在多个操作系统（OS）平台上开发和移植网络应用程序时，将面临错综复杂的挑战。这些复杂性的表现形式各异：网络协议不兼容，在不同软、硬件平台上具有不同 API 和语义的组件库，由于 OS 本身的（native）进程间通信（IPC）机制

和并发机制的局限性造成的“*偶发复杂性（accidental complexity）*”。为减小这些问题的影响，ADAPTIVE Communication Environment（ACE）提供了一套“面向对象”工具包。此工具包可运行在大量硬件和 OS 平台上，包括 Win32 和 UNIX 的大多数版本，以及很多实时及嵌入式操作系统。

可能有人告诉过你：一些“事实上”或“官方”的 OS 标准（如 POSIX、UNIX98 或 Win32），是所有程序员的保护伞，它们可以让你的应用程序免受“可移植性”难题的困扰。不幸的是，有这样一句格言：“标准的最大好处是，它们提供了大量的选择”[Tan96]。这一格言在今天比在 10 年前更适用。今天，数十种不同的 OS 平台正应用于商业、学校和政府项目中；每一个新版本和变体的出现，都会使这个排列数增加。

过去 20 年里，我们开发过很多“跨平台”的并发式网络系统，因而，我们可以明确地告诉你：在不同的时期，OS 供应商使用的标准也往往不同。何况，标准也是不断变化和发展的。你很可能工作在多个平台上，而这些平台是在不同的时期、以不同的方式去实现标准的。因而，直接针对 OS API 编程会导致以下两个问题。

1. **容易出错。**因为，用 C 写成的、OS 本身具有的 API 一般都缺乏类型安全（type safe）、可移植（portable）、可重入（reentrant）、可扩充（extensible）的系统函数接口和函数库。例如，在广泛使用的 Socket API（将在第 2 章讲述）中，通信端点是通过“弱类型化（weakly typed）”的“整数”或“指针”型 I/O 句柄（handle）来标识的，这无疑增加了程序运行时产生微妙错误的可能性。

2. **助长“不恰当”的设计技术的运用。**因为，很多用 OS API 写成的网络应用程序都是基于“算法”设计（algorithm design），而不是“面向对象”设计（object-oriented design）的。算法设计需要根据特定的功能需求来分解程序的结构，而功能需求并非一成不变——随着时间的变化，它很可能也会发生变化。因而，这种设计范式（paradigm）导致的软件结构将无法扩充，也无法根据不断变化的应用需求作出快速响应[Boo94]。

在这个经济动荡、缺乏管制、全球激烈竞争的年代，如果完全通过原始的（native）OS API 和算法设计技术来开发应用程序，其昂贵的开销和时间上的浪费将让人无法忍受。

如果你多年来一直从事网络软件系统的开发，或许早已将其中的一些问题视作家常便饭。但是，还有更好的选择！在本书中，我们将向你介绍，C++和 ACE 如何为你提供“面向对象”能力，从而让你尽可能地避开很多陷阱和避免犯错误，同时还不失对标准——甚至，某些“和平台相关”的特性——的支持。随着时间的推移，“面向对象”设计体现出的稳定性将远胜于“算法”设计；因而，在开发多种类型的网络应用程序时，应该优先选用“面向对象”设计。

不必惊讶，这些灵活性也需要代价：可能需要学习一些新的概念、方法、模式（pattern）、工具和开发技术。根据个人经验的不同，你的学习曲线可能是平坦的，也有可能在一开始就显得很陡峭。但重要的是，“面向对象”设计范式（paradigm）为你提供了一套成熟的技术，可以让你应对网络应用程序开发中的很多挑战。本书给出了一系列具体的示例，向你展示如何运用“面向对象”技术开发和应用 ACE 工具包中的类。你可以运用相同的技术和 ACE 类来简化自己的应用程序。

目标读者

本书奉献给“一线”程序员或高年级学生，因为在运用 C++和“面向对象”设计技术进行并发网络编程时，他们需要了解其中的战略、战术。我们将讲述重要的设计空间（dimension）、模式和原则——要想快速方便地开发灵活高效的并发式网络程序，就需要了解它们。我们提供了大量的 C++代码范例，让你强化对设计概念的理解，还具体演示如何尽快使用 ACE 中的核心类。我们还将你带到“幕后”，让你理解 ACE 工具包中的 IPC 和并发机制是如何设计的，以及为什么要那样设计。这些内容将帮助你增强设计技能，在“面向对象”网络程序中更高效地运用 C++和*模式*。

本书并非是讲述“面向对象”开发、模式、UML、C++、系统程序设计或网络技术的详尽教程。所以我们假设，对于以下主题，本书读者应该有一定程度的了解。

- **面向对象设计及编程技术**。例如，框架（framework）[Joh97, FJS99b, FJS99a]、模式（pattern）[GHJV95, BMR+96, SSRB00]、模块化（modularity）[Mey97]、信息隐藏（Information hiding）[Par72]、建模（modeling）[Boo94]。

- **面向对象表示法及处理方式**。例如，Unified Modeling Language（UML，统一建模语言）[RJB98]、eXtreme Programming（极限编程）[Bec00]、Rational Unified Process（RUP，Rational 统一过程）[JBR99]。

- **C++语言的基本特性**。例如，类（class）、*继承（inheritance）*、*动态绑定（dynamic biding）*、*参数化类型（parameterized type）*[Bja00]。

- **核心系统编程机制**。例如，事件多路分离、进程和线程管理、虚拟内存、UNIX[Ste98, Ste99, Ste92, Lew95, KSS96, But97]和 Win32[Ric97, Sol98, JO99]平台上都存在的 IPC 机制和 API。

- **网络术语和概念**。例如，TCP/IP[Ste93]、远程操作请求[Obj01]、客户/服务器架构[CS92]。

我们鼓励你广泛使用参考资料，针对想深入学习的主题去追根溯源。

本书也不是一部针对 ACE 程序员的使用手册，也就是说，我们没有对 ACE 中的每一个类（class）、每一个方法（method）进行说明。如果想达到那样详细的程度，可以查看完整的 ACE 在线文档，这些文档用 Doxygen [Dim01] 生成，由 http://ace.ece.uci.edu/Doxygen 和 http://www.riverace.com/docs 提供。相反，本书讲述的重点是：

- 设计成功的面向对象的网络应用程序和中间件需要了解的重要概念、模式和 C++特性；

- 最常用的 ACE TCP/IP 及并发式 wrapper facade 类背后的设计动机及其基本应用。

结构和内容

本书讲述的内容是：如何用 C++和中间件来解决网络应用程序开发中相关的重要课题。首先回顾主流 OS 平台本身具有的核心 OS 机制，并阐释如何在 ACE 中运用 C++和模式，将这些机制封装到类库（class library）wrapper facade 之中，从而提高应用程序的可移植性和健壮性。本书用到的主要例子是一个网络日志服务程序，其功能是：通过 TCP/IP，将日志记录从客户程序传输到一个日志服务器。我们将这个服务程序作为贯穿全书的示例，从而具体演示：

- C++和 ACE 如何有助于实现高效、可预测和可伸缩的网络应用程序；
- 开发并发式“面向对象”网络程序时，在设计和实现上要考虑到哪些因素和方案。

本书分为 11 章，其主要内容简介如下。

- **简介**——第 0 章是有关 C++网络编程的基本介绍。首先，勾勒出问题空间（problem space），提出应用程序运行在单进程、单线程的情况下可能面临的课题。然后，引入中间件层（middleware layer）的分类（taxonomy）概念，并讲述如何运用主机设施中间件和 ACE 工具包来解决常见的网络编程问题。
- **第 1 篇**——第 1~4 章概要介绍通信设计上的选择方案，讲述 ACE 中高效编写 OS IPC 机制时使用的面向对象技术。运用这些技术所得到的类构成了本书中的网络日志服务例程的第一个版本的基础。
- **第 2 篇**——第 5~10 章概要介绍并发设计上的选择方案，并讲述 ACE 中高效编写 OS 并发机制时使用的面向对象技术。

在整个第 1、2 篇中，我们针对那个网络日志服务程序，提供了一系列逐步复杂化的实现，以演示如何将 ACE IPC 和并发 wrapper facade 运用于实践。

附录 A 根据 ACE IPC 和并发 wrapper facade，总结了其底层的“类设计和实现”的原则。附录 B 介绍 ACE 的由来，以及作为开放源码，ACE 在过去 10 年的演变，此

外还描绘了它在未来的远景和发展方向。本书结尾提供了一个技术术语表（包括书中以斜体字形式出现的术语）、一个供你进一步阅读的参考文献的详细列表，以及一个综合性的主题索引。

相关材料

本书讲述的重点是如何通过特定的C++特性、模式和ACE来消除复杂性。这套丛书的第二卷——《C++ Network Programming：Systematic Reuse with ACE and Framework》[SH]——扩展了这些内容，对ACE提供的“面向对象”网络编程框架（framework）进行了说明。这些框架具体化了本书呈现的ACE wrapper facade类的常见使用模式（common usage pattern），支持更广泛、更深层次的系统复用（systematic reuse）。本书介绍的ACE wrapper facade类和卷2介绍的ACE框架类的区别是，ACE wrapper facade类很少有虚方法（virtual method），但ACE框架类中大部分方法都是虚方法。

本书基于ACE 5.2版，此版本发布于2001年10月。ACE软件和书中提供的所有示例程序都是开放源码，可以在http://ace.ece.uci.edu和http://www.riverace.com下载到它们。这些网站还提供了大量和ACE相关的其他资料，如教程、技术论文，以及一篇概括性的文档，此文档就本书没有提到的其他IPC和同步机制ACE wrapper facade作了介绍。我们提倡大家去下载一份ACE的副本，这样，就可以随着我们，将实际的ACE类和框架完完整整看个通透，并在阅读本书的过程中运行相应的示例程序。此外，还有已经编译好了的ACE版本，你可以向http://riverace.com购买，价格微不足道。

如果想更多地了解ACE，或者想报告你在本书中发现的任何错误，可以订阅ACE邮件列表ace-users@cs.wustl.edu。可以发送电子邮件至ace-users-request@cs.wustl.edu，向Majordomo列表服务器订阅邮件。在电子邮件正文中，请包含下面的命令（邮件主题省略）：

```
subscribe ace-users [emailaddress@domain]
```

只是当邮件信息中的“From”地址不是你想订阅的地址时，才需要提供emailaddress@domain。

发送给ACE邮件列表的帖子也会被同时转发至USENET新闻组comp.soft-sys.ace；发送给 ACE 邮件列表的所有帖子的历史档案则存储在 http://groups.yahoo.com/group/ace-users 上。

致谢

诚挚的谢意献给以下审阅者：Christopher Allen、Tomer Amiaz、Alain Decamps、Don Hinton、Susan Liebeskind、Dennis Mancl、Patrick Rabau、Eamonn Saunders、Johnny Willemsen。他们审阅了全部文稿并提供了广泛的意见，使本书在形式和内容上都有了实质性的提高。当然，书中遗留的任何问题，责任在我们。

来自世界各地的其他很多 ACE 用户对本书初稿提供了反馈意见，他们是 Mark Apple、Shahzad Aslam-Mir、Kevin Bailey、Barry Benowitz、Emmanuel Croze、Yasir Faiz、Gillmer Derge、Iain Hanson、Brad Hoskins、Bob Huston、Christopher Kohlhoff、Serge Kolgan、Andy Marchewka、Jeff McNiel、Phil Mesnier、Arturo Montes、Aaron Nielsen、Jeff Parsons、Pim Philipse、Yaron Pinto、Stephane Pion、Nick Pratt、Paul Rubel、Shourya Sarcar、Leo Stutzmann、Tommy Svensson、Alain Totouom、Roger Tragin、Reuven Yagel。

我们要感谢Washington University, St. Louis和University of California, Irvine的所有Doc group 成员——无论是过去的还是现在的成员，以及 Object Computing Inc.和Riverace Corporation的成员，他们都为开发、改进和优化本书讲述的许多ACE功能作出了贡献。他们包括：Everett Anderson、Alex Arulanthu、Shawn Atkins、John Aughey、Darrell Brunsch、Luther Baker、Don Busch、Chris Cleeland、Angelo Corsaro、Chad Elliot、Sergio Flores-Gaitan、Chris Gill、Pradeep Gore、Andy Gokhale、Priyanka Gontla、Myrna Harbibson、Tim Harrison、Shawn Hannan、John Heitmann、Joe Hoffert、James Hu、Frank Hunleth、Prashant Jain、Vishal Kachroo、Ray Klefstad、Kitty Krishnakumar、Yamuna Krishnamurthy、Michael Kircher、Fred Kuhns、David Levine、Chanaka Liyanaarachchi、