《COM本质论》作者Don Box又一力作

# Effective COM 中文版

50 Ways to Improve your COM and MTS-based Applications

Don Box Keith Brown
Tim Ewald Chris Sells 著
余蒲澜 译
李明 审

COM本质论》姊妹作品

0条详尽指南，覆盖COM全部核心

OM开发人员必读经典

52

Addison Wesley

中国电力出版社
www.infopower.com.cn

# Effective COM 中文版

TP311.52
26

50 Ways to Improve your COM and MTS-based Applications

Don Box Keith Brown
Tim Ewald Chris Sells 著
余蒲澜 译
李明 审

学技术大学 书馆藏

中国电力出版社

**Effective COM (ISBN 0-201-37968-6)**
**Don Box, Keith Brown, Tim Ewald and Chris Sells.**
**Authorized translation from the English language edition, entitled Effective COM, published by Addison Wesley Longman, Copyright©1999**
**All rights reserved.**
**No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without permission from the Publisher.**
**CHINESE SIMPLIFIED language edition published by China Electric Power Press**
**Copyright©2003**

本书由美国培生集团授权出版
北京市版权局著作权合同登记号 图字：01-2001-1872 号

图书在版编目（CIP）数据

Effective COM 中文版 /（美）伯克斯等著；余蒲澜译. —北京：中国电力出版社，2003
ISBN 7-5083-0847-6
Ⅰ. E... Ⅱ. ①伯...②余... Ⅲ. 软件接口，COM—程序设计 Ⅳ. TP311.52
中国版本图书馆 CIP 数据核字（2003）第 040097 号

**责任编辑：** 闫宏
**书　　名：** Effective COM中文版
**编　　著：** (美) Don Box 等
**翻　　译：** 余蒲澜
**技术审校：** 李明
**出版发行：** 中国电力出版社
地址：北京市三里河路6号 邮政编码：100044
电话：（010）88515918 传真：（010）88518169
**印　　刷：** 汇鑫印务有限公司
**开　　本：** 787×1092 1/16　　**印　　张：** 13.5　　**数：** 200千字
**版　　次：** 2003年8月北京第一版
**印　　次：** 2003年8月第一次印刷
**定　　价：** 29.00 元

# Preface from the Translator

# 译 者 序

COM 是一种开发软件组件的方法。Microsoft 的许多技术，如 ActiveX、DirectX、OLE 以及.NET 平台的核心，都是基于 COM 建立起来的；同时，Microsoft 也大量地使用 COM 组件来编写他们的应用程序和操作系统。COM 所蕴含的概念并不只是在 Windows 操作系统下才有效——COM 并非一个大的 API。实际上，它所代表的面向组件编程与结构化编程及面向对象编程一样，也是一种编程方法，而且有望成为未来的主流。在任何一种操作系统中，开发人员均可以遵循“COM 方法”。可以预见，COM 的应用将会日益广泛。因此，对开发人员而言，掌握 COM 技术就显得尤为重要了。

本书由著名的 COM 大师 Don Box 及其合作者一起编写，是 Addison Wesley 出版社名声显赫的、由 Scott Meyer 开创的“Effective 系列”图书中的重要一员。他们对 COM 的背景和技术内核有着极为深刻的理解。有关 COM 的基础教程，一般来说，只是单纯地介绍一些基本的理论和技巧，而本书面向的则是高层次的 COM 开发人员。作者从不同角度引导我们解决 COM 开发中最常见的一些高深疑难问题，而这些问题的解决，对应用程序的正常运行起着至关重要的作用。为了让我们可以更加高效地运用 COM，本书还提供了具体的指导原则和详尽准确的示例程序。尤为可贵的是，作者把自己在 COM 领域中多年的研究成果无私地奉献给了读者。

本书的结构安排合理。它按不同主题逐步深入地探讨了在 COM 开发中可能遇到的高级问题。全书涉及了 C++到 COM 的转换接口、安全、单元、实现以及事务等多方面的具体内容。针对每一方面，本书都提供了一些详尽实用的指导原则。它不但考虑了 C++的开发人员，而且也兼顾了其他语言的开发人员。相信各层次开发人员在阅读本书之后，在 COM 的开发方面都会受益匪浅。

COM 技术向来以难于掌握著称，而本书更是众多 COM 书籍中较深的一部。如果读者需要了解 COM 的基本知识和理论，建议循序阅读中国电力出版社出版的《DCOM 入门》和《COM 本质论》。

由于时间仓促，加之译者对 COM 的认识和理解程度有限，译文难免有不妥之处，恳请读者批评指正。

译　者

2003 年

此为试读，需要完整PDF请访问：www.ertongbook.com

# Preface

# 前　言

COM（组件对象模型，Component Object Model）与C++在很多方面是并行发展的。二者的发展都是为了一个共同的目标，即通过对现有编程模型的改进来更好地实现对象重用性和模块性。就C++而言，其先前的模型是C语言的过程型程序设计，C++增加了对基于类的面向对象程序设计的支持。而就COM来说，它以前的模型是C++中基于类的编程模型，COM增加了对基于接口（interface-based）的面向对象程序设计的支持。

随着C++的发展，有关它的经典著作也越来越多。在这些经典著作之中，引人注目的一本是Scott Meyer撰写的《Effective C++》[①]。这本书可能是第一本不以教授读者C++基本技巧和语法为目的的书。该书针对的是一线的C++程序员，并且提供了50条所有C++开发人员在创建基于C++的系统时都应遵从的具体规则。该书的成功一方面因为在此领域有大量应用此种技术工作的开发人员，另一方面也是因为有大量相关经典著作可以借鉴。在该书最初出版时，虽有其他很多介绍性的文章，但是提供帮助的主要是Stroustrup的《The C++ Programming Language》以及Lippman的《The C++ Primer》[②]。

COM编程的发展已经到达了发生重要转变的临界点。虽然微软及其他开发公司大量地采用COM，COM开发人员的数目还是增长缓慢，但是，可以肯定的是，这个数目正逐步接近Windows开发人员的数目。同时，在COM发布5年后，终于有了足够详细的规范，以此为基础可以撰写更高级的书了。这里，谨以本书向Scott Meyer致敬，为他创造性的工

---

① 《Effective C++》已有侯捷所译的中文版出版。它的姐妹篇《More Effective C++》中文版已由中国电力出版社出版。——译者

② 《The C++ Primer》中文版由潘爱民主译，已由中国电力出版社出版。——译者

作，为他的伟大尝试——提供了一本如此大众化的书，使得大多数开发人员都可以很容易地找到通用设计和编码问题的解决方案。

实际上，现有的 COM 图书基本上都假设读者不懂 COM 知识，把大部分精力都集中在基础教学上。本书试图提供一本超越 COM 基本技巧和理论的指南，填补当前 COM 规范的空白。这些具体的指导原则基于作者使用 COM 的实际工作经验，来源于作者最近 4 年来对成千上万 COM 开发人员的培训心得，同时还来源于各种基于 Internet 论坛上的大众智慧。其中最重要的论坛是在 DCOM-request@discuss.microsoft.com 上的 DCOM 邮件列表。

有很多人在本书的编写过程中提供了宝贵的建议，他们是 Saji Abraham、David Chappell、Steve DeLassus、Richard Grimes、Martin Gudgin、Davide Marcato、Ted Neff、Mike Nelson、Peter Partch、Wilf Russell、Ranjiv Sharma、George Shepherd 以及 James Sievert。特别感谢 George Reilly，他以超强的编辑能力改正了作者写作中的错误。任何其余的错误都是作者的责任。你可以通过给 Effectiveerrata@develop.com 发送 Email 来让我们知道这些错误。任何勘误或更新都将被发布在本书的主页上：http://www.develop.com/effectivecom。

本书所提供的有些原则与一般看法或者微软的官方文档不符，这个事实可能会令读者感到迷惑。但我们鼓励你不带偏颇地尝试我们的观点，然后告诉我们你的发现。你可以通过发送 E-mail 给 effectivecom@develop.com 与本书的作者联系。

## 读者对象

本书适合使用 COM 和 MTS 的软件开发人员阅读。它既不是一本教程，也不是入门级读物，而是假设读者有一定的 COM 编程经验，并且正为分布式对象计算的复杂性和广泛性而烦恼。与《Essential COM》[①]一书中描述的一样，本书假定读者熟悉 COM 的工作术语。本书主要是为 C++程序员而作，然而，很多主题（如，接口设计、安全、事务）也可供 Visual Basic、Java 以及 Object Pascal 的开发人员参考。

## 本书内容

本书分为 6 章。除了第 1 章介绍“100%纯”C++与 COM 的文化差异外，其他各章都介绍 COM 的核心元素之一。

### 第 1 章　从 C++转变到 COM

C++开发人员在使用 COM 工作时具有最大的灵活性。然而，正是这些开发人员必须进

---

① 这是本书的姐妹篇，由 Don Box 撰写，中文版《COM 本质论》由潘爱民译，已由中国电力出版社出版。——译者

行调整以适应基于 COM 的开发。该章提供的 5 条具体指导原则，使得从纯 C++到基于 COM 开发的转变成为可能。关于 COM/C++方面的讨论包括异常处理、单实例（singleton）以及基于接口的编程。

**第 2 章　接口**

COM 开发最基本的组成部分是接口。设计良好的接口将有助于提高系统效率和可用性，而设计糟糕的接口将使系统脆弱并难以使用。该章提供了 12 条具体的指导原则来帮助 COM 开发人员设计高效、正确并易于使用的接口。关于接口方面的讨论包括往返优化、语义纠正以及一些常见的设计缺陷。

**第 3 章　实现**

不管用来设计 COM 组件的框架以及类库是什么，在 C++中编写 COM 代码更需要理解细节。该章提供 11 条具体指导原则来帮助开发人员编写出高效、正确以及可维护的代码。关于 COM 实现方面的讨论包括引用计数、内存优化以及类型系统错误。

**第 4 章　单元**

COM 最复杂的内容之一就是单元（apartment）的概念。单元在 COM 中用于建立并发模型，在很多操作系统和语言里都没有类似的概念。该章提供了 9 条具体的指导原则，帮助开发人员确保他们的对象在多线程环境里能够运行正常。关于单元方面的讨论包括真实锁定管理、常见列集（marshal）错误以及生存期管理。

**第 5 章　安全**

在 COM 里面存在少数几个比单元更加令人生畏的领域，其中之一就是安全。这是因为很多开发人员天生对安全厌恶；另一部分原因在于那些相当神秘而又不完整的文档困扰着 COM 接口的安全性。该章提供了 5 条具体的指导原则来帮助开发人员掌握 COM 领域的安全解决方案。关于安全方面的讨论包括访问控制、验证和授权。

**第 6 章　事务**

许多出版物都使用大量篇幅讨论 MTS（微软事务服务器，Microsoft Transaction Server），但却很少讨论关于 MTS 所隐含的新的事务编程模型的问题。该章提供了 8 条具体的指导原则，帮助开发人员建立基于 MTS 的更有效、可扩展和正确的系统。讨论的主题包括截取的重要性、基于活动的并发管理，以及将即时激活作为增强可伸缩性的基本机制的危险性。

## 致谢

首先，同时也是最重要的，Chris 要感谢他的妻子 Melissa。他的各种活动，包括本书

的编写，Melissa 都给予了很多支持。

其次，感谢 J. Carter Shankin 及 Addison Wesley 公司的工作人员为我提供了理想的写作环境。

感谢所有的审校者为本书提供深思熟虑的反馈意见。

感谢我所有的学生以及对 DCOM 和 ATL 邮件列表有贡献的成员。本书中所有的真知灼见都来自于大家围绕 COM 互相讨论的问题。

最后，但可不是最不重要的，感谢我的合作者。在这个项目中，他们自始至终都在勤奋地工作。能够和他们一起工作，我倍感荣幸。

Don 希望感谢其他 3 个同伴，因为正是他们丰富了他 COM 以外的生活方式。

感谢我的合作者（这些饥饿的工作狂们）与我分担压力并耐心地等待我完成自己杂乱的那一部分。

非常感谢 Scott Meyers 允许我们采用[①]他那种取得巨大成功的写作格式，他将这种格式应用到自己终身为之奋斗的一门技术中。

感谢我在 DevelopMentor 的所有同事。感谢他们在我还拖延了另一本图书写作项目的同时，能够忍受另外 6 个月的延迟。

感谢 Addison Wesley 的 J. Carter Shanklin，是他创造了一个舒适的工作环境。

感谢那些曾参与这个漫长而有趣话题的众多 DCOM 列表成员。本书在各方面都是对工作在 COM 最前线的编程人员提出的安全漏洞、MTS 技术内幕以及 IDL 挑战性难题的总结。

特别感谢多年以来在 COM 和 Visual C++方面一直提供支持的微软工作人员。

Keith 希望感谢他的家庭能够忍受他工作的日日夜夜，是他们给他的生活带来了无穷的乐趣。

感谢 Don、Tim、Chris 能想到我并且邀请我参加这个重要项目。

感谢 DevelopMentor 的 Mike Abercrombie 和 Don Box 营造了一种独立思考的氛围以及一种建立在诚实和团队协作基础上的业务模式。

感谢那些在 DCOM 邮件列表中参与讨论这些总是很长的话题的人。在 COM 开发人员之间建立文化氛围这点上，这个邮件列表起着令人难以置信的作用。很多丰富的想法从这个文化氛围中涌现了出来，并被写入本书。

感谢 Saji Abraham 以及 Mike Nelson 对 COM 社区所做的贡献。

感谢 Carter，如果你给我们指定本书的最后截稿期限，那么这本书可能会差得多。

---

① 采用（leverage）是在 Windows 开发圈里经常使用的一种委婉说法，它的真正意思显而易见。（这个词大家应该记住，因为在大量英文技术文献中都会使用这一用法，而这是一般字典里查不出来的）——译者

最后，感谢所有参与我的 COM 和安全课程的学生们。你们的评论、问题以及挑战始终驱使着我为进一步揭开真相而努力。

首先，也是最重要的，Tim 希望感谢他的合作者承担了这个项目并且坚持将它完成。与以前一样，先生们，这是一件很愉快的事。

同时，感谢我的朋友和同事 Alan Ewald、Owen Tallman、Fred Tibbitts、Paul Rielly 以及 DevelopMentor 的所有同事、学生，还有倾听我关于 COM 进展的 DCOM 邮件列表的参与者——感谢他们在必要的时候给予的明智的赞许和嬉笑怒骂。

特别感谢 Mike、Don 以及 Lorrie 经历了 DevelopMentor 早期的磨难，而最终出现了一种杰出的思考环境。

当然，还要感谢我的家人——Sarah 让我戴上了 COM 的光环，Steve 和 Kristin 提醒我成功的真谛，Alan 和 Chris 不厌其烦地回答我的问题，Nikke 和 Stephen Downes-Martin 替我接听来自各地的电话。

最后，感谢 J. Cartter Shanklin 以及 Addison Wesley 公司给我们充足的自由。

Chris Sells
俄勒冈州波特兰
1998 年 8 月
http://www.sellsbrothers.com

Don Box
加州 Redondo Beach
1998 年 8 月
http://www.develop.com/dbox

Keith Brown
加州 Rolling Hills Estates
1998 年 8 月
http://www.develop.com/kbrown

Tim Ewald
新罕布什尔州 Nashua
1998 年 8 月
http://www.develop.com/tjewald

# Contents

# 目　录



此为试读，需要完整PDF请访问：www.ertongbook.com

CHAPTER 1

# 第 1 章
# 从 C++转变到 COM

从纯粹的 C++开发转变到 COM 世界似乎会受到很多限制。很多你所知道的和钟爱的语言结构从知识库中消失了，取而代之的是一整套称为属性（attribute）[①]的全新语言结构，这种语言更接近于 C 而不是 C++。C++开发人员的思考方法把重点放在对象的实现上，要使他们的注意力转变到按“通过请求和响应消息来通信”的组件方式思考问题，需要花费相当多的时间。本章将讨论一些重要的观念，这些观念的转变对于帮助你成功地从“100%纯”C++转变到另一种风格的子集（也就是我们将要介绍的 COM）是绝对必需的。

## 实践 1　在定义类之前定义接口（用 IDL 实现）

C++开发人员最基本的条件反射之一，就是从一个“.h”文件开始某个项目的编码阶段。C++开发人员通常正是在这里开始定义数据类型的公共操作及其内部表示形式。当开发一个完全基于 C++的项目时，这是一种完全合理的方法。然而，当开发一个基于 COM 的项目时，这种方法通常会带来痛苦和麻烦。

COM 中最基本的概念是接口和实现的分离。虽然 C++编程语言支持这种方式的编程，但它对于“把接口定义为与实现它们的类分隔的单独实体”，并没有提供太多显式的支持。如果没有这种对接口的显式支持，就很容易模糊接口和实现之间的界限。COM 开发新手常见的毛病就是忘记接口是对某些功能的抽象定义。这意味着 COM 接口的定义不应该泄露实现该接口的某个特定类的实现细节。考虑下面的 C++类的定义：



① 请注意这里的属性（attribute）与一般面向对象文献中指代对象成员数据的属性（以及通过 set 和 get 方法设置的属性，即 property）都是有区别的。——译者

```
class Person {
  long m_nAge;
  long m_nSalary;
  Person *m_pSpouse;
  list <Person*> m_children;
public:
  Person(void);
  void Marry(Person& rspouse);
  void RaiseSalary(long nAmount);
  void Reproduce(void);
  Person *GetSpouse(void) const;
  long GetAge(void) const;
  long GetSalary(void) const;
  const list<Person*>& GetChildren(void) const;
};
```

这是一个完全合理的类定义。然而，如果我们把它当作 COM 接口定义的起点，那么最直接的映射方法应该类似如下：

```
DEFINE_GUID(IID_IPerson, 0x30929828, 0x5F86, 0x11d1,
        0xB3, 0x4E, 0x00, 0x60, 0x97, 0x5E, 0x6A, 0x6A);
DECLARE_INTERFACE_(IPerson, IUnknown) {
  STDMETHOD(QueryInterface)(THIS_ REFIID r,void**p)PURE;
  STDMETHOD_(ULONG, AddRef)(THIS) PURE;
  STDMETHOD_(ULONG, Release)(THIS) PURE;
  STDMETHOD_(void, Marry)(THIS_ IPerson *pSpouse) PURE;
  STDMETHOD_(void, RaiseSalary)(THIS_ long nAmt) PURE;
  STDMETHOD_(void, Reproduce)(THIS) PURE;
  STDMETHOD_(IPerson *, GetSpouse)(THIS) PURE;
  STDMETHOD_(long, GetAge)(THIS) PURE;
  STDMETHOD_(long, GetSalary)(THIS) PURE;
  STDMETHOD_(list<IPerson*> *, GetChildren)(THIS) PURE;
};
```

其中，DECLARE_INTERFACE_宏用于强调这个接口定义必须出现在 C/C++的头文件中。

2

这种接口定义的第一个缺陷就是，它使用 STL（标准模板库，Standard Template Library）列表来返回孩子的集合。虽然这种方法在封闭的单一二进制系统（其中所有组成部分的源代码将被编译和连接成一个不可分割的单元）中是合理的，但是这种技术在 COM 中将是致命的。因为，在 COM 中，组件可能在某一编译器下构造，却被用不同编译器编译的客户代码所使用。在这种情况下，就不可能保证客户和对象在一个 STL 列表中的表示形式是一致的。即使两个实体用同一个厂商所提供的编译器来编译，也不能保证使用的是同一 STL

版本[①]。

与返回STL列表有关的一个更明显的问题是，这个接口有可能泄露底层类的实现细节，即使用一个STL列表来存储孩子的集合。对于前面所给出的实现，这不是一个问题。然而，如果其他类实现者希望实现IPerson接口，他们同样必须把孩子存储在STL列表中，或者在每次调用GetChildern方法时创建一个新的列表。既然STL列表并不是集合的最有效表示形式，那么这个约束就将给所有IPerson实现带来不应有的负担。

与数据类型有关的最后一个缺点是必须利用每个方法的返回结果。每个IPerson方法都是使用STDMETHOD_宏来定义的，它允许接口设计者显式指明方法的实际结果类型。因此，下面的方法定义：

```
STDMETHOD_(long, GetAge)(THIS) PURE;
```

经C预处理器处理后，将被扩展为以下代码：

```
virtual long __stdcall GetAge(void) = 0;
```

这种方法存在的问题是，它不能返回HRESULT值。在实践2中会详细讲到，COM通过重载方法返回的实际结果来指示通信的失败；而这种方法不返回HRESULT值，所以COM无法通知客户可能发生的通信错误。

如果开发人员使用COM的IDL（接口定义语言，Interface Definition Language）来定义接口，那么上面所说的问题就不会出现。了解IDL后，你会发现它是很吸引人的。那么为什么我们必须掌握另一门语言呢？存在这种疑惑是合理的。大多数25岁以上的开发人员都至少已经掌握了一种编程语言，他们不愿意再学习另一种编写条件和循环语句的语法。这种不情愿是可以理解的，因为，每10年所诞生的新编程语言都承诺将会极大地提高程序员的生产力，然而，伴随而来的往往是奇形怪状的语法。



必须着重指出的是，COM的IDL不是一种编程语言，对大多数人来说它并不是全新的。IDL没有用来编写可执行语句的结构。IDL是一种基于属性的声明性语言，它只有用来定义与COM兼容的数据类型的结构。同时，IDL从C语言那里继承了语法，这些语法对于现在很多使用COM的C、C++和Java开发人员来说都很熟悉。IDL语法是非常简单的，以至于使用其他语言（例如Visual Basic和Object Pascal）的开发人员，也可以在一星期之内轻松地掌握IDL的基础知识。的确，将来的一些COM+版本可能会用一种更为集成的方案来代替当前的IDL编译器。然而，即使MIDL.EXE被扔到回收站，IDL所使用的基于属性的技术也会依然存在，其前提是IDL仍被保留。

IDL直接从C语言中继承了定义枚举、结构和联合的语法。下面的代码片断就是完全

① 很多开发人员不愿使用编译器的STL实现，他们更喜欢来自第三方供应商的高性能版本。

合法的 IDL：

```
enum COLOR { RED, GREEN, BLUE };
struct ColorPoint {
  enum COLOR color; /* color */
  long x;            /* horiz. coord. */
  long y;            /* vert. coord. */
};
struct NODE {
  struct ColorPoint value; /* value of node */
  struct NODE       *pNext; /* pointer to next node */
};
```

这个 IDL 代码片断也是合法的 C 程序。应当注意的是，虽然它也是合法的 C++程序，但 IDL 采用 C 而不是 C++的惯例来区分类型命名空间与标签（tag）命名空间。这意味着：

```
struct NODE {
  ColorPoint value; // Not legal IDL
  NODE       *pNext; // Not legal IDL
};
```

不是合法的 IDL，因为名字 ColorPoint 和 NODE 必须用关键字 struct 来限定作用域。

IDL 在 C 语言上增加的最主要的扩展，就是既能定义 COM 接口，又能定义 COM 类。下面的 IDL 代码片段比先前给出的基于 DECLARE_INTERFACE_的定义更可取：

```
[ uuid(30929828-5F86-11d1-B34E-0060975E6A6A), object ]
interface IPerson : IUnknown {
  import "unknwn.idl";
  void Marry([in] IPerson *pSpouse);
  void RaiseSalary([in] long nAmount);
  void Reproduce(void);
  IPerson * GetSpouse(void);
  long GetAge(void);
  long GetSalary(void);
  list<IPerson*> * GetChildren(void);
}
```

当这个接口定义被提交给 IDL 编译器时，先前所指出的问题将会导致 IDL 编译器产生错误消息。这个错误消息将通知接口设计者他们的方法是错误的。

例如，由于 IDL 编译器假设接口将要跨进程和主机边界使用，因此将确保所有的方法都返回 HRESULT。然而，我们可以通过使用[local]属性来取消这种检查（以及远程代码的生成）：

此为试读，需要完整PDF请访问：www.ertongbook.com