EXPERT'S VOICE® IN .NET

# UML Applied
## A .NET Perspective

# UML 实战教程
## ——面向.NET 开发人员

（美）Martin L. Shoemaker 著
高 猛 朱洁梅 译

清华大学出版社

# UML 实战教程

## ——面向.NET 开发人员

(美)Martin L. Shoemaker 著

高 猛 朱洁梅 译

清华大学出版社

北 京

EISBN：1-59059-087-2
UML Applied：A .NET Perspective
Martin L. Shoemaker


北京市版权局著作权合同登记号　图字：01-2005-5282



**图书在版编目(CIP)数据**

.UML 实战教程—— 面向.NET 开发人员/(美)修马克(shoemaker,M.L.)著；高猛，朱洁梅译. —北京：清华大学出版社，2006. 2
书名原文：UML Applied：A .NET Perspective
ISBN 7-302-11939-2
Ⅰ.U… Ⅱ.①修… ②高… ③朱… Ⅲ.面向对象语言，UML—程序设计—教材 Ⅳ.TP312

中国版本图书馆 CIP 数据核字(2005)第 114689 号

**出 版 者**：清华大学出版社　　**地　　址**：北京清华大学学研大厦
http://www.tup.com.cn　　**邮　　编**：100084
**社 总 机**：010-62770175　　**客户服务**：010-62776969
**组稿编辑**：曹　康
**文稿编辑**：王　黎
**封面设计**：孔祥丰
**版式设计**：孔祥丰
**印 刷 者**：北京密云胶印厂
**装 订 者**：三河市化甲屯小学装订二厂
**发 行 者**：新华书店总店北京发行所
**开　　本**：185×260　**印张**：21.5　**字数**：550 千字
**版　　次**：2006 年 2 月第 1 版　2006 年 2 月第1 次印刷
**书　　号**：ISBN 7-302-11939-2/TP・7741
**印　　数**：1～4000
**定　　价**：39.80 元

# 前　言

**提示：**

*致缺少耐心的读者……如果您没有时间掌握基本原理并且希望立刻开始学习 UML(统一建模语言)符号，则可以直接跳到第 2 章开始学习。*

**每个人的 UML**

我在 UML 中看到的一个重大讽刺(任何软件技术都有这类问题，但 UML 是我知道的最佳的示例)是它那难以置信的神秘性，以及这种神秘性又成为了两种人之间的一堵墙，一种是了解如何使用技术的人们，另一种是可能真正从这种技术中获益，但却没有时间克服这种神秘性的人们。这一点非常讽刺，并且使人感到沮丧，因为这些人通常希望设计技术来帮助他们解决问题。他们期望新的技术能帮助他们工作得更快和更有效率。他们非常繁忙的事实决定了他们没有多余时间学习。这种神秘性——具有新的专门术语和全新的词汇表——极大地超出了技术自身的实际学习曲线。

现在我们有了.NET，最新的 Microsoft 策略，用于为支持网络的世界构建一个常见的开发平台。当我编写这本书时，我很难立刻告诉您这种策略是否将在很长一段时间内获得成功。但是，通过查看过去一些软件领域的变革，我可以确定地预测带来的直接影响将是：仅仅编写代码也会带来更多的压力！我们都知道确实存在压力。我们都知道在自己虚弱时不得不屈服于这种压力。我们也都知道做如下工作时带来的灾难性后果：通过错误的计划表，以错误的方法获得的错误系统；我们希望获得的所有益处正在缓慢流逝。带着类似的压力，我们感觉更加无法研究类似于 UML 的新主题。但是，如果您希望在将来的.NET 开发世界中居于领先地位，关键就在于使用良好设计的产品尽早地进入市场，这种产品可满足良好定义的用户需求。

UML(有或者没有.NET)的关键是忽略这种神秘性并且正面处理这种学习曲线。毫无畏惧地接近 UML 的人通常就可以快速地理解它，并且只需要很少的指导。毕竟，UML 是统一建模语言的简称，设计它的主要意图在于交流。Three Amigos(Grady Booch、James Rumbaugh 和 Ivar Jacobson)和 OO(面向对象)社团花费了很多精力来建立用于交流的图表(如果使用良好的话)。但是，神秘性和可察觉的复杂性仍然很巨大——在我的书架上有超过 6000 页的 UML 书籍，其中许多书籍我到现在还没有看过——因此繁忙的开发者和设计者继续和许多问题进行斗争，而使用 UML 的面向对象分析和设计已经解决了这些问题。统一建模语言包含许多内容：表达需求、体系结构和系统实现的符号；来自于 Booch、Rumbaugh 和 Jacobson 的思想的综合；以及由 Object Management Group 采纳的标准，仅仅列出了其中一些。但是最重要的是，UML 是用于交流的工具。这是我希望读者留下深刻印象的教训，解除这种神秘性并使其有意义的关键在于：如果您正在进行交流，您就是在很好地使用 UML。

应该交流什么呢？模型：通过主要的画图来表现完整的系统应该看起来像高层次的、抽象的、聚焦的图形。应该有足够详细的信息，从而开发者可以了解他们必须构建什么，但不能过于详细，因为过于详细则项目关系人无法了解由树组成的森林。这些是在充分的抽象层中系统的图形和信息，从而允许来自于用户、分析师、架构师和域专家的有用反馈。但是要记住：重

点不是图形，而是模型。绝对不应该简单地编辑一个图表；应该一直编辑一个底层的模型，图表只是该模型的部分表示。建模将让您尽早捕获代价最大的 Bug，并且尽早地发觉和修正可以使您在 Bug 修复的计划表和费用上节省 50 到 200 的系数(请参见 Steve McConnell 所著的 *Software Project Survival Guide*，第 29 页)。

这将引导我们了解 UML 中的 U：Unified(统一的)。30 年来，真正聪明的人们已经开始思考 OOAD(面向对象分析与设计)和通用设计方面的问题，并且已经提出大量关于这种问题的方法。其中一些思想非常好，一些思想非常杰出，而一些思想则产生了重复——许多都是如此。许多努力都浪费在关于使用云形或矩形这种并不重要的争论上；同时这些思想后面深入的思考者则不断尝试让我们超越这些细微区别的含意，并且带我们进入更深入的过程。因此，Three Amigos 将他们各自最好的思想结合起来，然后大胆地采用来自于这个领域中任何位置的最佳的思想。“Not Invented Here(不使用非本地发明的)”不在他们的词汇表中：如果某个人有关于设计问题的优秀解决方案，他们就采用这种解决方案，并且将其合并到 UML 中。结果是产生一种丰富的建模语言，这种语言可让您研究从简单到复杂的各种设计思想，所有思想都通过良好定义的方法适合于底层的建模。根据两个原因，您可以更快地构建解决方案：语言足够丰富，从而可以解决大多数常见的设计问题，而不需要发明或学习新的设计方法；这种语言可以被广泛的读者所了解，这意味着您将能够更快更有效地进行交流。

### 模型驱动的开发

我非常支持使用统一建模语言(UML)进行面向对象的分析和设计。我在做顾问工作和指导工作中，提倡使用一种过程，我将其称为模型驱动的开发，或者称为 MDD。

- 您有关于问题域、需求、解决方案的体系结构以及解决方案的单独组件的、大量相互关联的模型。
- 所有规范文档引用模型，并且被模型引用。
- 所有的设计和代码都派生自模型。
- 所有的估计和计划表都基于模型的元素。
- 所有的测试计划和测试案例都派生自模型。
- 所有的终端用户文档都根据模型而制定。
- 所有项目人为产物的状态都反映在模型中。

因此，模型成为过程的人为产物，并且系统所有方面资料的中心仓库。我发现，通过启用关于需求、设计、关注方面、解决方案、估计和计划表等方面的交流，持续使用 UML 的模型驱动的开发可帮助开发者构建更好的系统。

当然，这是有前提的，首先每个人必须熟悉并持续地使用 UML；然后，您的开发过程必须围绕 UML 进行构建。我已经通过实践和耐心开发了一种训练，这种训练将让我把每种开发活动都基于模型。但这一点并不容易，并且可能很难掌握。许多团队看到需要涉及的努力，就会说：“非常好，但我们很忙，因此无法现在就学习所有的内容”。因此，在本书中和我的课程中，我首先使用 UML 五步法，这是一种轻量级过程，但正好够用，可以帮助您使用 UML；而不是迷失在复杂的过程中。

### UML、MDD 和.NET

.NET 不只是一种用于网络系统的常见开发平台。从 UML 的视角来看，.NET 的关键特性在于它是基于组件的，这意味着它本身就很容易重用和扩展——甚至比先前的技术(例如 COM

或 COM+)更容易。使用作为 OOAD 一部分的组件建模可以最好地完成有效的组件重用。此外，您希望将满足良好定义的用户需求的、良好设计的产品尽早投入市场。使用 UML 的 OOAD 可以帮助您实现这个目标。

UML 是否是银弹？当然不是。我们都知道没有这样的东西。良好的代码仍然需要努力、汗水、比萨饼和独创性。良好的分析和设计仍然需要与顾客和域专家的良好管理的会谈和会议。良好的计划仍然需要基于分析的、现实的、可达到的计划表估计，而不是基于展览会的计划表—— 并且管理者愿意支持这种估计。但是，UML 可以在这些活动中插入更好的信息：程序员可以实现的更好的设计，用于在设计期间从顾客处获得更好的反馈和更好的交流，以及计划表估计的更好的输入。然而，您只需要做剩余的困难工作。我知道这样做可以节省时间和开发成本，因为我在自己的项目中已经看到了这一点。通过使用 UML 进行详细的 OOAD 并将结果插入到估计工具中，我就能够确切地告诉顾客我将在何时传递什么；虽然这仍然需要花费很多个晚上和大量的比萨饼，但我尽早传递了主要子系统的第一个版本，在预算之内，没有主要的缺陷，并且比初始的请求具有更多的特性——我们以前做任何大型项目都没有经历过这种结果。现在，我知道 UML 可以帮助我做到这一点，我不会再希望使用其他的方法进行开发。并且，我希望可以帮助您获得相同类型的结果。

这就是本书的目标：消除神秘性，并且介绍紧密集中在统一建模语言上的、面向对象分析和设计过程的功能强大的元素(模型驱动的开发)；然后显示使用 UML 的 OOAD 如何帮助您掌握.NET 开发的复杂性。其中的一些材料来自于一些经验的启发，我在编写和讲授 Richard Shaw 的 UML 训练营讲座时学到了这些经验：不是课程中的经验，而是在讲授课程时学到的经验。虽然本书是全新的材料，但它构建在这些经验之上，我在向全世界学生讲授课程时学到了这些经验。当了解到哪些消息可以传达给学生以及哪些无法达到目标时，我构想了一个更为简单的方法来传达 UML 的原理。我希望能够对 UML 进行轻便直观的介绍，这将缩短我所经历过的学习曲线。我完全确信使用 UML 的 OOAD 过程将帮助您节省在学习曲线上的成本；但是，如果我可以为您减少 6 个月的学习曲线，我将实现我的目标，减少大型软件开发的复杂性。

### 本书的读者

本书主要面向下面 4 种核心读者：

- .NET 开发者：和.NET 相关的 UML 的最基本原理仅仅是：很少有开发者了解统一建模语言，或者不了解如何将其应用于.NET 开发。因此，本书的第一部分是一份指南，用于将 UML 介绍给.NET 开发者。我在职业生涯中大部分都是作为“开发者”这个角色；很明显我的观点来源于开发者这个角色。如同我将在整本书中指出的一样，如果我可避免让一个开发者早上 3 点困在仓库中束手无策，那么我将获得成功。在“开发者”下面，我包括下面的角色：分析员、架构师、设计者、编码员和维护编码员。同时，我承认这些角色在大多数组织中都存在或多或少的重叠。如果您的组织中也存在这种重叠，我鼓励您仍然考虑单独的角色，将其作为在正确时间内完成正确任务的方法。在分析期间就研究代码问题是一种错误，这种错误在每个项目以及每个组织中都会发生。
- 管理者：我承认：我和 Scott Adams 一样说了许多管理者的玩笑。但是，这些玩笑有一个潜在的事实：良好的、有效的软件管理是让开发者获得成功的关键。此外，管理者通常比开发者具有更为困难的工作，并且一般是所有软件开发中最为困难的工作。他们需要帮助遇到挫折的开发者、上层管理人员和顾客；并且由于某种原因，他们不得

不调解总是不断变动的过程。使他们的工作更为容易也就是使每个人的工作更为容易，我希望可以在这方面有所帮助。

- 测试人员：测试人员应该很高兴地知道：这位开发者相信他们的角色是 OOAD 过程成功与否的关键；如果他们没有提供对过程的连续评估，则不会产生任何过程。他们每发现一个 Bug，顾客就会少发现一个 Bug，并且开发者也不会因为这个 Bug 而在早上 3 点困在仓库中束手无策。我希望帮助他们的开发者和管理者分享这种观点，并且更早和更彻底地结合测试。
- 文件编制者：任何编写这么长的一本书的人都对文件编制者抱有同情心：可以很容易地判断何时产生了可用的代码；但最彻底的评审和最仔细的重写工作也无法保证成功的书面交流。当他们不得不将复杂的技术概念翻译为有用的终端用户指示时，他们的工作只会更加困难。如同我将在本书中强调的一样，交流是 UML 的目标，这包括技术文档和用户文档中的交流。我希望较早的和更为彻底的涉及文档的 OOAD 过程将使文件编制者的工作更为容易，从而使他们的终端用户的工作更为容易。

本书只提供对常见 OO 概念的最简单介绍：对象、类、实例、方法、属性、关系、接口和相关主题。如果需要了解更多的 OO 知识，读者应该参考列在附录 C 参考书目中的相关书籍。

**有关案例分析**

选择一个用于讲授 UML 的示例并不是一件容易的任务：这个主题覆盖了非常广泛的活动范围；潜在的读者背景范围很广泛。在设计用于 UML 训练营的练习时，我的目标是找到所有学生都熟悉的“通用”示例；然而，我发现这些示例并不是非常通用。它们可用于解释广泛的概念，但在我需要解释过程或符号中的丰富细节时，这些示例就显得无能为力。对于这些主题，我根据自己的经验或曾经参考过的项目绘制特定的示例。作为结果的练习具有不同程度上的成功：相比于其他的内容，学生们更加理解某些内容(根据他们自己过去的经验)；但是，我收到的最常见的反馈是，最有用的练习是和一个大型框架紧密配合的练习。当学生了解到他们在一个练习中的决定如何引起另一个练习中的更多工作时，他们就会了解整个的过程，而不仅仅是单独的片断。他们的反馈是清楚的：从始至终地进行一个项目。这种方法并不是没有风险：作为示例的一个项目意味着一种观点，一些人对其比较熟悉，而另一些人则对其比较陌生。我可能发现一些读者无法理解特殊的问题域，因此无法从示例中学到什么。

如果您无法理解某些问题，则我就无法与您进行交流。UML 训练营给我 5 天时间来和 30 位学生交流思想(事实上没有时间来回顾某些主题，如果这些主题在经过两次解释之后还是不清楚的话)，但是在这里我拥有整本书的空间；读者也可以随时回顾某些解释。

因此，我已经定义了一个假设客户的假定项目，将其用作本书的案例分析。这个案例分析是犬舍管理系统，在附录 A 的规范中对其进行了详细描述。如果您需要详细地了解这个示例，我鼓励您阅读背景资料。

**注意：**

虽然我已经包括了一些需要亲自实践的练习，但大多数练习都是无限制的，并且允许对项目进行选择。对于大多数练习，您将只需要纸和铅笔。您可能也会在手边找到一些便笺纸(Post-it notes)，用来绘制图表，然后重新安排图表。如果您有一个偏爱的 UML 工具，我鼓励您使用它；但是，不要让这种工具的缺点妨碍您学习 UML。

因为 UML 是关于交流的，如果单独地执行某些练习，您将只会获得很有限的益处。UML 本身就是功能强大的；但是，UML 的真正功能只有在您和其他人共享您的设计并获得他们的反馈时才会体现出来。我鼓励您和一个合作伙伴或整个的团队一起评审您的解决方案。良好的设计需要来自其他人的评审。

我并不鼓励您接受犬舍管理系统中的解决方案，不能确定地将其认为是“正确的”。从带有缺陷的设计中，您可以学习到和良好的设计中一样多的内容——可能甚至更多，因为您必须应用自己所学到的内容，从而确认和修正这些缺陷。在确认我的错误的过程中，您将慢慢学会使用 UML。错误总是会发生，假装没有发生错误是愚蠢的行为。聪明的做法是评审这些错误并修正它们。

—— Martin L.Shoemaker
Hopkins,MI

**离题话(这是第 1 个)**

正如我的 UML Bootcamp 学生会证明的，我在讲课时喜欢说些离题话。我骄傲的是我最后总能从离题话中回到我离开的那一点，无论是直接回来还是通过类比。

而且，我允许我的学生把我推进这些离题话中而没有真正的反抗。我回答他们的问题时，不仅仅是将罐装的答案脱口而出。如果他们问了一个指向编程细节的不可思议的问题——或者甚至于完全脱离了编程轨道——我也会在它引向的任何地方沿问题展开。如果我们努力寻找的话，有时可能引向的地方正好是要讲述的相关要点。

在本书中我重新创作了最好的和最相关的离题话，我希望这些小隐喻会发出一些微弱的光，照亮开发过程，帮助您从不同的角度思考问题。我认为真正的设计是一个杂乱的过程，组织起来的整洁程度也有限。跳出结构外面来思考通常是解决困难的问题的钥匙。

不过为了保护那些只要原始材料不要隐喻的读者，我仔细地将离题话框了起来。类似这样离题话是正文中的枝节问题：哲学观点、相关问题或者一个思想的深入探索。如果您觉得文中的信息已经够清楚了，那您完全可以跳过它们。

# 目　录

## 第 I 部分　UML 和 UML 五步法：都是关于交流的

## 第 II 部分 案例分析：应用于.NET 解决方案的 UML

## 第 III 部分 隐藏在代码后面的内容

## 第 IV 部分 附 录

# 第I部分　UML和UML五步法：都是关于交流的

本书的第I部分将介绍OOAD（Object-Oriented Analysis and Design，面向对象的分析和设计），以及UML在OOAD中扮演的角色。没有建模语言，建模过程也无多大用处；而建模语言若没有应用到建模过程中，则它仅仅只是一个理论性的抽象。因此，我们将同时介绍这两方面的内容。

第1章将重点介绍一些基本概念，第2章将更详细地介绍UML五步法的细节、一个小型的、以模型为中心的迭代过程，以及用于进一步开发的UML语言核心部分。您将学习到不同的UML图，以了解它们的目的，以及如何在实际的上下文中使用这些UML图。

有关语言和过程的规则理论上都很完善，但您也需要了解一些常识和一些行之有效的建议来帮助您学会建模，这是我们将在第3章中重点介绍的内容。本章中介绍的准则将提高您创建的UML图的表达水平。

最后，因为本书的重点是学习如何将UML OOAD应用到.NET项目中，所以本书中第I部分的最后一章将介绍UML中一些.NET核心概念的表示。这将帮助您理解案例分析中的图，我们将在第II部分中重点介绍UML图。