PEARSON Addison Wesley

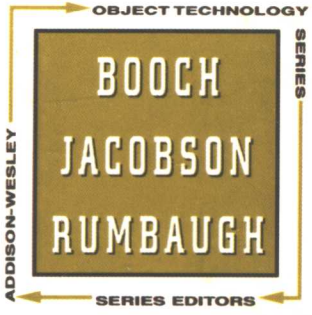
用例驱动的UML对象建模应用
——范例分析

APPLYING USE CASE DRIVEN OBJECT MODELING WITH UML

AN ANNOTATED E-COMMERCE EXAMPLE

DOUG ROSENBERG
KENDALL SCOTT

〔美〕Doug Rosenberg Kendall Scott 著
管 斌 袁国忠 译



人民邮电出版社
POSTS & TELECOM PRESS

用例驱动的 UML 对象建模应用
——范例分析

[美] Doug Rosenberg　Kendall Scott　著

管　斌　袁国忠　译

人民邮电出版社

图书在版编目（CIP）数据

用例驱动的 UML 对象建模应用：范例分析 /（美）罗森堡（Rosenberg，D.），（美）斯克特（Scott，K.）著；管斌，袁国忠译．—北京：人民邮电出版社，2005.5

ISBN 7-115-13395-6

Ⅰ．用… Ⅱ．①罗…②斯…③管…④袁… Ⅲ．面向对象语言，UML—程序设计

Ⅳ．TP312

中国版本图书馆 CIP 数据核字（2005）第 033491 号

版 权 声 明

用例驱动的 UML 对象建模应用——范例分析

◆ 著 [美] Doug Rosenberg Kendall Scott
译 管 斌 袁国忠
责任编辑 俞 彬

◆ 人民邮电出版社出版发行 北京市崇文区夕照寺街 14 号
邮编 100061 电子函件 315@ptpress.com.cn
网址 http://www.ptpress.com.cn

读者热线 010-67132687
北京顺义振华印刷厂印刷
新华书店总店北京发行所经销

◆ 开本：800×1000 1/16
印张：10.5
字数：204 千字 2005 年 5 月第 1 版
印数：1－6 000 册 2005 年 5 月北京第 1 次印刷

著作权合同登记号 图字：01-2002-3742 号

ISBN 7-115-13395-6/TP • 4657

定价：28.00 元

本书如有印装质量问题，请与本社联系 电话：（010）67129223

前 言

理论与实际

在我们编写的第一本书 *Use Case Driven Object Modding with UML*（用例驱动的 UML 对象建模）中指出过：从理论上说，理论和实践之间是没有区别的，而实际上并非如此。在该书中，作者基于自 1993 年来给数以百计的项目小组就 OOAD（Object-Oriented Analysis and Design，面向对象分析和设计）进行培训时获得的经验，试图将 OOAD 建模理论简化为易于学习、极其通用的实用子集。

面市两年后，该书已经重印 5 次。虽然我们的工作得到了读者的极大认可，但也存在一些遗憾的地方。在过去的两年中，我们经常听到的读者反馈意见是："希望有更多的用例和 UML 建模范例"。与此同时，当我们将该书用作主要的培训教材，将理论用于实际的客户项目时，我们发现对模型进行复核至关重要，而很多学员并没有很好地理解这一点。

虽然我们在第一本书中提供了大量的范例，但基于上述原因，我们说服了 Addison Wesley，出版了这本配套练习手册，书中以循序渐进的方式详细地剖析了如何设计一个网上书店，其中说明了常犯的许多错误以及改正这些错误后得到的相关模型。我们之所以选择网上书店作为范例，是由于当前许多 Web 驱动项目都与此相关；同时我们在培训中一直将该范例作为练习，其中包含课堂 UML 模型的丰富资源，还有学生们在建立这些模型时经常犯的错误。

我们收集了一些典型的错误——学生们常犯的错误，并以此为基础编写了本书，同时增加了 3 章有关复核的内容：需求复核、初步设计（preliminary design）复核和关键设

计（critical design）复核。

本书的独特之处在于引导读者改正错误。

本书的组织方式

本书共8章，第1章简要地介绍了ICONIX过程。然后以4章的篇幅详细介绍了该过程的4个关键阶段。这些章节的内容如下。

第2、3、5、7章首先从整个过程的角度，分别介绍了域建模、用例建模、健壮性分析和时序图的精髓。每一章将完成网上书店范例中的相关内容，而在每一章的最后将这些内容组合在一起，形成一个概要图。第3章介绍了10个用例，并在第5章、第7章的初步设计和详细设计中分别完成其中的5个用例（第5章和第7章中的用例文本和完整类图也将对第2章中介绍的类图进行描述）。接下来介绍各个阶段的要素。这些内容基本上是《用例驱动的UML对象建模》一书中相应章节的微缩版本，同时增加了一些新的内容。接下来的一节介绍了培训过程中学生常犯的10种错误。本书新增了5个Top 10列表：10种常见的健壮性分析错误、10种常见的时序图错误以及3种复核过程中分别应避免的10种错误。最后一节提供了5个练习，供读者测试自己对本章知识的掌握程度。

各练习的共同点如下：

- 对于单数页，标题名的颜色为黑色，没有底纹。对于域建模和用例练习，标题为“练习X”；而对于健壮性分析和时序图练习，标题为用例名。
- 在单数页中，每页列出了3～4个错误，其中每个错误的旁边有一个编号，指出它违反了哪条规则。
- 在接下来的双数页中，标题名采用反白显示，同时更正了前一页的错误，并对错误进行了解释。

读者在参考正确的做法之前，应尝试对单数页中列出的错误进行更正。

概括地说，第2章列出了书中将介绍的10个用例范例要使用的类，第3章列出了各个用例的片段，第5章和第7章则分别列出了其中5个用例对应的图表。这样安排的想法是，让读者从对用例似懂非懂，到完成每个用例时序图（包括完整的文本）和给出详细设计的要素。

读者可能会问，余下的3章包含什么内容呢？

- 第4章介绍如何进行需求复核，确保用例和域模型协同工作，以满足客户的功

能性需求。

- 第 6 章介绍如何进行初步设计复核（PDR），确保所有用例都有相应的健壮图，域模型包含一组丰富的、对应于原型的属性（以及该模型表示的用例需要的所有对象），同时确保开发小组为进行详细设计做好了准备。
- 第 8 章介绍如何进行关键设计复核（CDR），确保时序图表明的详细设计的实现方式同用例规定的要实现的内容相符，同时确保细节设计足够详细，很容易变成编码。

关于复核的这 3 章都包含概述、细节和 Top 10 列表，但没有提供练习。这些复核工作的共同之处在于，确保模型的各个部分的一致性，符合练习答案的要求。

附录对书店模型进行了总结，读者可从 http://www.iconixsw.com/WorkbookExample.html 下载。附录中包含本书正文中列出的所有图表，同时包含另外 5 个用例的设计细节，让读者能够将这些用例作为练习，并将结果同我们提供的结果进行比较。强烈建议读者这样做。

这非常棒，不是吗？据笔者所知，当前市面上还没有这样的图书，希望本书对读者应用用例驱动的对象建模技术有所帮助。

Doug Rosenberg
dougr@iconixsw.com
http://www.iconixsw.com

Kendall Scott
kendall @ usecasedriven.com
http:///www.usecasedriven.com

目　录

第1章 绪论

ICONIX过程的规模大概在重量级的Rational Unified Process（RUP）和轻量级的极限编程方法（XP）之间。和RUP一样，ICONIX过程也是用例驱动的，但不需要RUP使记录延续到表中带来的大量开销；和XP一样，它相对较小，比较紧凑，但不像XP那样摒弃了分析和设计过程。因此，有助于使用统一建模语言（UML），同时对需求进行跟踪。该过程遵循了Ivar Jacobson的“用例驱动”的思想，能够获得有形、具体、易于理解的用例，开发小组可以使用这些用例来驱动开发工作。

我们采用的方式充分利用了上个世纪90年代初期出现的3种方法。这些方法是由自称为“三人帮”（three amigos）的Ivar Jacobson、Jim Rumbaugh和Grady Booch开发的。基于Doug对这3种方法的分析，我们使用了UML的一个子集。

由“三人帮”编写的《统一建模语言用户手册》一书的第32章指出：对于大多数问题而言，只需使用20%的UML，就可以完成80%的建模工作。但作者没有在该书中告诉读者，应使用哪20%的内容。我们使用的UML子集重点放在完成大部分建模工作所需的一组核心表示法。在本书中，我们也将介绍如何使用UML的其他元素，并在需要时将其加入。

我们常说的一句话是：“从理论上说，理论和实践之间是没有区别的；但实际上并非如此。”实际上，好像总是没有足够的时间来完成建模、分析和设计工作。管理层总是不断敦促开发人员进入编码阶段，人们总是过早地进入编码阶段，因为软件工程的进度常常是以代码的多少来衡量的。我们采用的方法要求最低，效率很高，它将重点放在用例和代码之间的领域。它强调的是，从软件生命周期的什么位置开始，此时必须完成哪些工作，即必须已经有一些用例且需要进行优良的分析和设计。

我们的目标，是确定足以很好地完成软件项目工作所需的、最小的 UML（和建模技术）子集。我们用了八九年的时间，对这里所说的“足够且最小”的定义进行了改进。本书将要介绍的方法被用于成百上千个项目中，被证明在各种行业和项目中都能可靠地工作。

1.1 ICONIX 过程的各个阶段

图 1.1 说明了 ICONIX 过程要回答的关键问题。

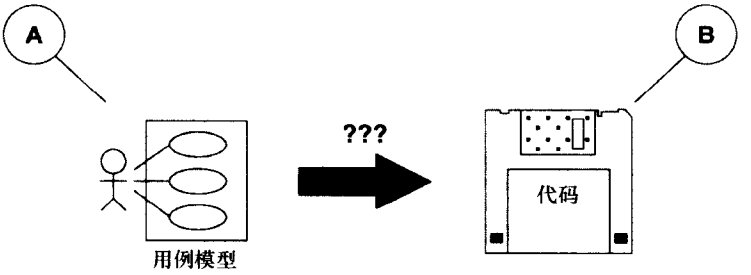


图 1.1　从用例到代码

这里要说明的是，如何在尽可能短的时间内从 A 点直接到达 B 点（实际上，我们不会编写代码，但将深入介绍整个过程，以便读者对其有所了解）。您可以将 A 点看作是这样一种情况——系统必须包含的功能已经确定，因此需要编写一些用例；而将 B 点看作一些完整的、经过测试和调试的代码，它们实际完成用例需要执行的功能。换句话说，代码实现了用例规定的行为。本书将重点介绍如何对“我希望系统以这样的方式完成某些工作”这一模糊、朦胧的状态，进行明确、完整、严格的描述，以构思出优秀、可靠的体系结构和健壮的软件设计方案，并进一步编写出清晰的代码，实现用户要求的行为。

我们将从代码开始，沿过程向前介绍实现目标的步骤。我们将解释为何认为将要介绍的一组步骤是您所需的最少的步骤；同时，在大多数情况下，这些步骤对于缩小用例与代码间的差距也是足够的。图 1.2 说明了我们将做出的 3 种假设：一些原型化工作已经完成；确定了用户界面；已经开始确定系统的一些场景或用例。

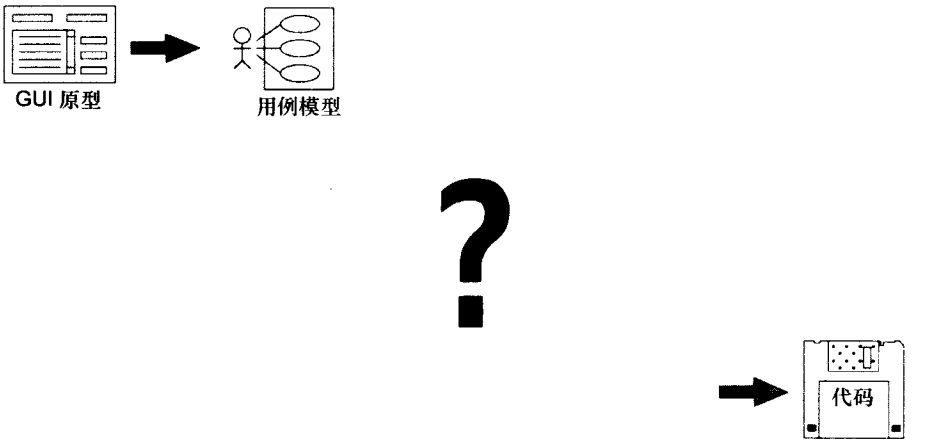


图 1.2　工作起点

这样，我们将开始分析和设计工作。要说明的是如何从这个起点到达代码。开始时，只有一个大大的问号——我们对系统的功能有一个模糊、朦胧的认识。进行编码前，需要填补这一鸿沟。

在面向对象的系统中，代码的结构是由类定义的。因此，编写代码前，必须知道需要的软件类是什么样的。为此，我们需要一个或多个类图来指出系统中的所有类。其中的每个类必须有一组完整的属性，它们是类中包含的数据成员和操作，后者定义了软件函数。换句话说，需要确定所有的软件函数，并确保拥有这些函数完成其工作所需的数据。

我们需要指出这些类如何封装数据和函数，还需要在类图中指出类的组织方式以及它们之间在类图中的关系。我们将使用 UML 类图作为工具来说明这些信息。我们的最终目标是获得一组非常详细的、设计级（design-level）类图。设计级指的是一种详细程度，这种级别的类图可用作系统实际代码的模板——准确指出如何组织代码。

图 1.3 表明，编写代码前必须建立类图，并建立一个这样的设计级图表，即其中的类与源代码中的类一一对应。但现在仍不能直接进入编码阶段。现在，需要根据用例绘制设计级类图，而不是直接从用例进入到编码阶段。

在面向对象的软件开发中，最困难的工作之一是分配行为，这需要就每个要建立的软件函数做出决策。对于每个函数，必须决定要将其放在哪个类中。我们必须分配系统的所有行为——将每个软件函数分配到一组将要设计的类中。

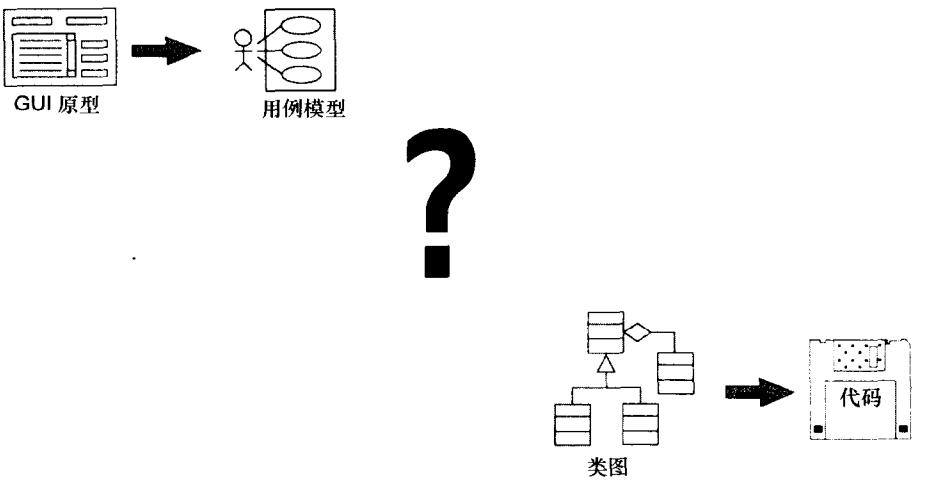


图 1.3　类图规定了代码的结构

在这方面，时序图是非常有用的 UML 图之一。这种图是帮助您做出行为分配决策的理想工具。时序图是基于每个场景的：对于系统中的每个场景，我们都将绘制一个时序图，指出哪个对象负责代码中的哪个函数。时序图指出运行阶段对象实例如何通过传递消息进行通信。每条消息将调用接受该消息的对象的一个软件函数。这也就是这种图成为以可视化方式表示行为分配情况的理想工具的原因。

图 1.4 表明，当我们继续往后走时，用例和代码之间的差距缩小了。现在，需要根据用例获得时序图。

绘制时序图时，需要将行为分配给类。也就是说要将操作放在软件的类上。使用诸如 Rational Rose 或 GDPro 等可视化建模工具，在时序图上绘制消息箭头时，实际上是将操作分配给了类图上的类。工具强化了这样一个事实，即从时序图开始，行为分配工作就开始了。当您绘制时序图时，类图上的类将被填充操作。

因此，关键在于根据用例获得时序图。在大多数情况下，这并非一个无关紧要的问题，因为用例是系统的需求级视图，而时序图是非常详细的设计视图。这正是我们的方法同当前市面上大多数方法的不同之处。大部分方法都讨论到了用例和时序图，但没有讨论如何跨越模糊的用例和详细程度类似代码的时序图之间的鸿沟。跨越“什么”和“如何”之间的鸿沟是 ICONIX 过程的中心主题。

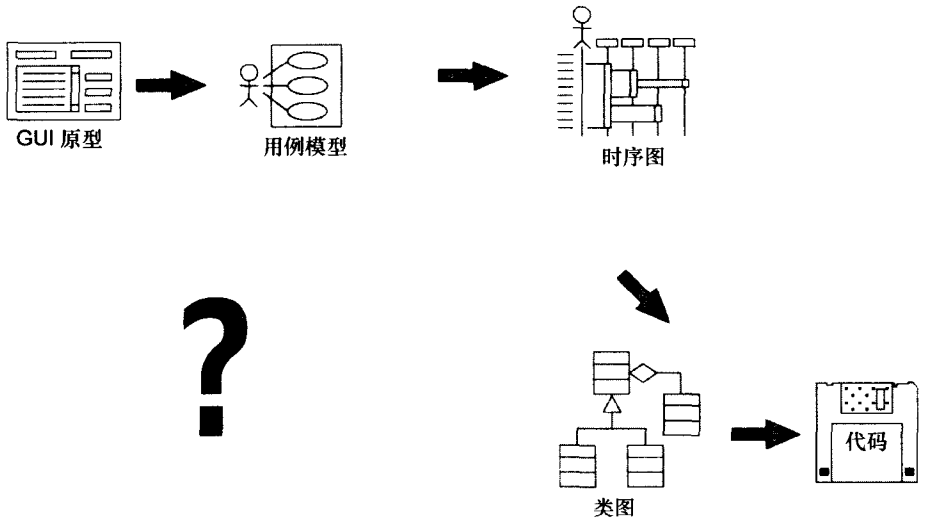


图 1.4　时序图帮助我们将操作（行为）分配给类

现在我们要做的是，使用健壮性图（robustness diagram）来填充模糊、朦胧的用例和非常详细、精确的时序图之间的鸿沟。健壮性图在需求和详细设计之间搭起了一座桥梁，使得从用例得到时序图更为容易。

如果读者阅读过有关 UML 的文献，将发现 UML 最初并没有包含有关健壮性图的全部内容。经过 Ivar Jacobson 的努力，健壮性图已成为 UML 标准的一项附属内容。其原因可追溯到 Booch、Rumbaugh 和 Jacobson 一道将他们的建模方法（而不是相对重要的图表）融合起来的历史和过程。

时序图的最上面是特定场景涉及的一组对象。要绘制时序图，首先必须完成的工作之一是确定场景将涉及哪些对象。当然，如果能够猜测哪些软件函数将在该场景中被执行，也会对我们有所帮助。绘制时序图时，必须考虑将完成所需行为的一组函数分配给场景涉及到的一组对象。

如果明白需要哪些对象，这些对象将执行哪些函数，将有很大的帮助。当您再次这样做时，将比刚开始猜测时准确得多。我们遵循的流程同 Ivar Jacobson 在其 Objectory 研究中描述的流程基本相同，即首先进行猜测（或初步设计），这样做得到的结果称为健壮性图；然后对猜测进行改进，形成详细设计，结果为时序图。对于要构建的每个场景，都需要绘制一个时序图。

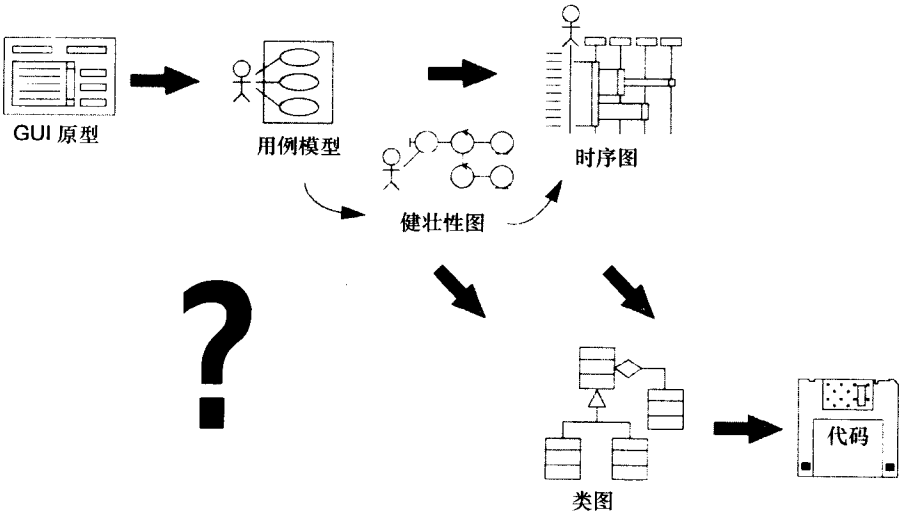


图 1.5　健壮性图在需求和详细设计之间架起了一座桥梁

图 1.5 表明，我们将把健壮性图加入到我们的 UML 子集中。最初的 UML 规范对健壮性图进行了描述，但其定义位于一个名为 Objectory Process-Specific Extensions 的附加文档中。在过去的十年中，我们发现，如果没有这种技术，从用例到时序图的工作将很难完成。使用健壮性图有助于避免项目小组根据用例直接进入到软件设计时常犯的错误。如果您遵循这一步骤，将使这个过程和您的项目变得容易许多。健壮性分析并非我们的发明，但我们将尽力确保它不被人们忘却。事实证明，健壮性分析对于跨越需求和设计之间的鸿沟非常有帮助。

健壮性分析在系统需要做什么和如何真正完成这些任务之间架设了一座桥梁。架设这座桥梁时，将同时完成多项不同的工作。首先，将发现首次猜测系统中有哪些对象时遗漏的对象；同时，在跟踪健壮性图上的数据流时，还可将属性加入到类中。当我们绘制健壮性图时，将完成的另一项重要的工作是更新并改进用例文本。

现在，仍然还有一个问号。这个问号与发现首次猜测时遗漏的对象有关。这意味着我们需要在某一点进行首次猜测。

在帮助讲解如何成功地编写用例时，我们使用一句神奇的格言：在对象模型上下文中对系统使用情况进行描述。这首先意味着，本书讨论的不是编写模糊、抽象、不明确的用例（它们没有包含足够的细节，无法根据它们获得软件设计），而是要编写非常明显、准确而清楚的用例。当我们讨论用例时，心中有一个非常明确的目标：我们要根据它们来驱动软件设计。很多有关用例的书籍采用完全不同的观点，更多地将用例用作一种探

测抽象需求的技术。我们的方法之所以不同源自我们的目标不同。请记住，我们的任务是帮助您从用例到达代码。

我们将首先介绍域模型（domain model），它是一种主抽象词汇，即问题空间（问题域）中最重要的名词。在术语域模型中，“域”来自问题域（problem domain）。例如，如果我们的问题域是电子商务，则将有一个诸如产品目录或订购单这样的域对象。我们将这些属于问题空间的名词称为域对象（domain object）；在分析和设计工作的开始，我们将创建一个域模型——在一个大型 UML 类图中展示所有的域对象。

在健壮性图中，我们也将使用边界对象（boundary object）。边界对象包括诸如系统屏幕之类的东西。在用例文本中，我们将显式地引用域对象和边界对象。我们将编写诸如用户如何与屏幕交互以及屏幕如何同域对象交互等内容。域对象常常同一个数据库相关联，而后者可能位于系统 OO 部分的后面。如果我们遵循在对象模型上下文中描述系统如何被使用的指导原则，则用例文本将更具体、更明确。

在域建模过程中，我们需要确定最重要的抽象集，它们描述了任务要创建的系统的问题空间或问题域。为此，我们必须遵循 Jim Rumbaugh 创建的方法——对象建模技术（OMT），它详细地描述了一些有助于建立域模型的技术。

我们的方法同您可能遇到的其他面向用例的方法的区别之一是，我们坚持在整个过程的开始就进行域建模。编写用例时，将使用域模型中的名词集。这样将域模型作为词汇表，能够明确地定义一组可在用例文本中引用的术语。事实证明，这种方法很有帮助，尤其是当您在小组环境中工作，其中有多组人员需要描述系统不同部分中的场景时。如果能就系统中重要的名词达成一致，则可消除用例模型中的不明确性。例如，这让您能够知道定购单是什么，line item 是什么，购物车是什么。所有这些东西从一开始就是明确的，因为我们在编写用例之前定义了一个术语表。

用 UML 的术语来说，域模型基本上是一个类图，因此它是一种同设计级类图相同的图。通常，域模型中将省略大量细节，具体地说，我们不在类上列出其属性和操作。域模型更像一个全局总结级类图。事实上，这是对类图的首次猜测，完全侧重于要创建的系统的问题域。我们对类图进行首次猜测，然后完成用例的所有细节，并对系统视图进行改进。在完成场景的过程中，首次猜测的类图将演变成系统的详细静态模型。

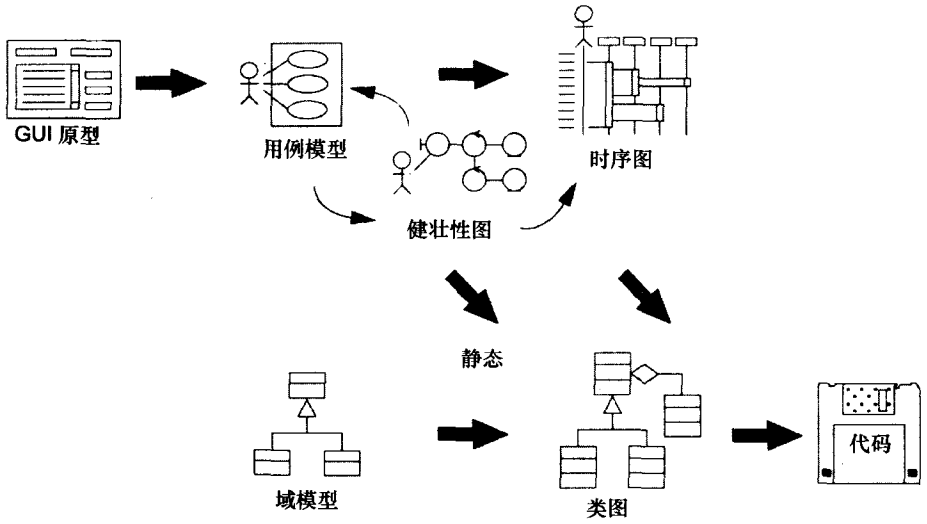


图 1.6　通过名称引用域对象消除了用例中的二义性

从图 1.6 所示可知，现在我们得到了一个非常完整的视图，其中没有任何鸿沟，它将帮助我们从左边的用例和原型到达右边的设计级类图和源代码。

我们使用的方法效率非常高。这里只使用了 4 种不同的 UML 图。UML 是由 9 种不同的图构成的，这是其中的 4 种。通常，对于大多数项目而言，大多数时候只需使用不超过一半的 UML 内容就可完成大部分工作。在学习使用 UML 进行建模时，将重点放在这些图的核心子集上，将极大地避免您的学习走弯路。

我们将从域模型开始介绍，它是一个分析级类图，是对系统的静态结构首次进行猜测的结果。然后，我们将不断地对该模型进行修订，并添加细节，最终得到详细设计。图 1.6 中下半部分的类图是对如何组织代码的静态描述，而用例则是对运行阶段行为的动态描述。

我们将对静态模型进行首次猜测，然后将大部分时间用于研究各个用例。每对用例研究一次，我们都将在类图中添加一些新的细节。完成系统需要支持的所有场景，添加使这些场景发生的所有细节，并对所做的工作进行多次复核后，将得到一个能够满足需求的设计方案。这时就可以开始编写代码了。

图 1.7 描述了整个 ICONIX 过程。在《用例驱动的 UML 对象建模》一书中，每章的第一页都包含这个图。它由两部分组成：上半部分是动态模型，对行为进行了描述；下半部分是静态模型，对结构进行了描述。

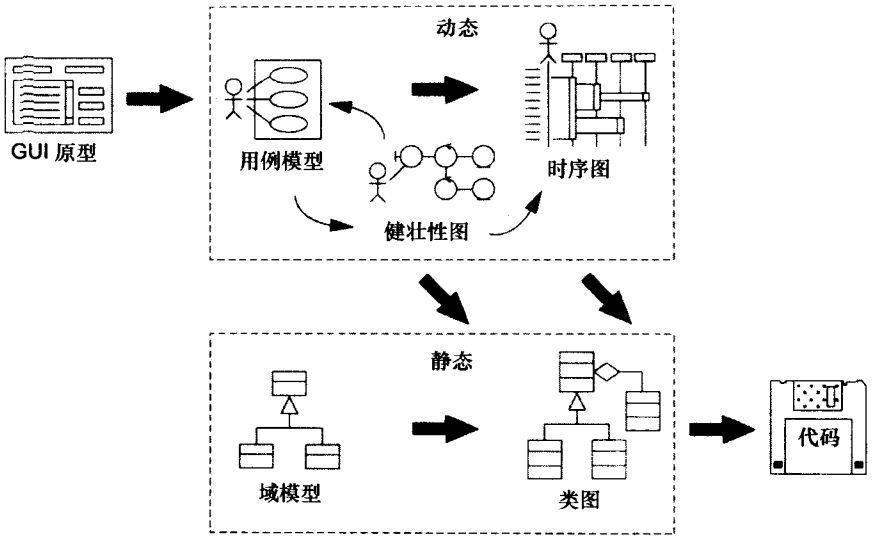


图 1.7　ICONIX 过程——一种高效的 UML 建模方法

我们可以从一些原型或一些屏幕图开始。然后，在用户确认这些内容是正确的之后，便可以开始确定用例图中的用例，用例图指出了系统将完成的所有场景。再后编写用例文本，并在健壮性分析过程中对用例文本进行改进。在初步设计阶段，用例文本必须正确、稳定，然后才能进行详细设计，这至关重要。详细设计是在时序图中完成的。

许多人抱怨需求总是在不断地变化。有些人将此作为过早开始编码的借口。我敢打赌，这些人中的大多数从来没有进行过健壮性分析，而健壮性分析对于确保需求稳定很有帮助。

将动态模型的探索分为 3 步，我们获得了两次对行为描述进行复核的机会。当我们对其进行第二次复核时，很可能对要求的行为有详细的了解，且需求已相当稳定，因此可以开始根据它进行设计。

从图 1.7 所示的静态部分可知，我们首先完全根据问题空间的描述，对对象进行快速的猜测。然后长时间地不断进行改进，这是通过对系统的运行阶段的动态行为进行分析驱动的。我们考虑场景如何进行工作的细节，再基于对场景更进一步的理解，对类图进行更新。然后，回过头来，就系统应该具备什么样的行为做进一步的考虑。

接下来，我们对软件结构做相应的改进。我们的方法 80%是从 Ivar Jacobson 的工作衍生而来的，它以一种非常自然的方式沿用例边界对系统进行分解，然后使用用例分析的结果驱动对象建模工作的前进，使之足够详细，可用来指导代码的编写工作。