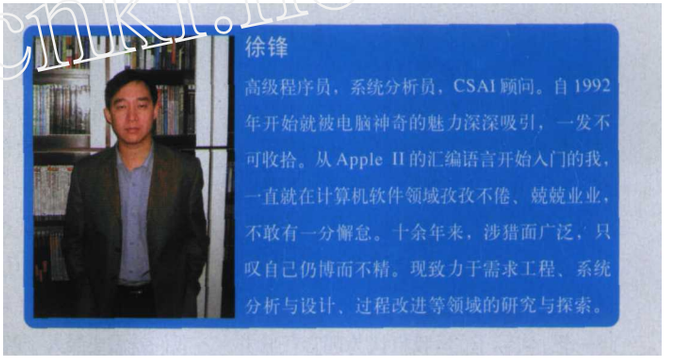
实战OO：过程总结

徐锋／文

引言

在本系列的前面四篇文章中，我们以一个简单的“个人图书管理系统”需求为例，讲解了如何使用OO方法进行分析与设计。其中分别重点阐述了如何建立体现业务知识的域模型、实现需求整合的用例模型，以及如何通过Robustness分析过渡到设计，如何使用交互模型来进行详细的设计等四方面的内容。

但怎么将这四种技术融会贯通地应用到软件开发过程中呢？我们在前面的讲解中，只着重每种技术的局部应用，而且仅是从一个用例着手，逐步深入。为了避免让大家“只见树木，不见森林”，本期我们就在前面的基础上，从软件开发过程这一宏观的角度进行总结，以帮助大家更好地应用于自己的项目实践。

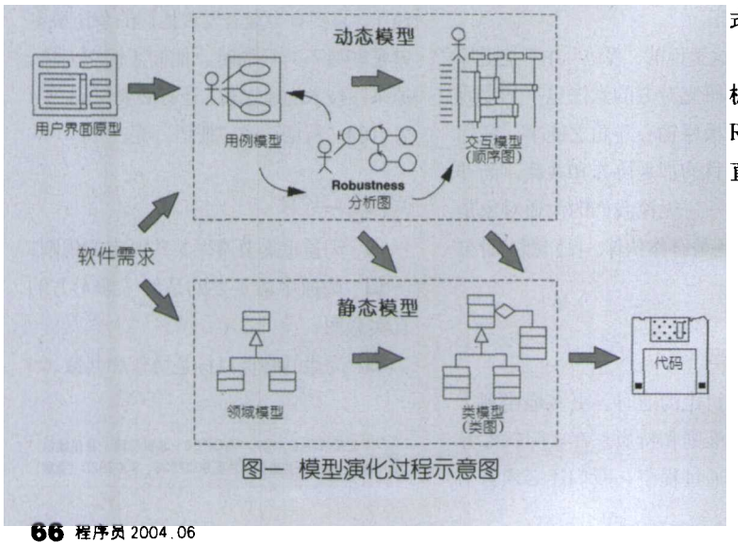


徐锋

高级程序员，系统分析员，CSAI顾问。自1992年开始就被电脑神奇的魅力深深吸引，一发不可收拾。从Apple II的汇编语言开始入门的我，一直就在计算机软件领域孜孜不倦、兢兢业业，不敢有一分懈怠。十余年来，涉猎面广泛，只叹自己仍博而不精。现致力于需求工程、系统分析与设计、过程改进等领域的研究与探索。

模型的演化过程

在UML中，定义了动态和静态两种模型。其中动态模型主要包括用例模型、Robustness分析模型、交互模型；而静态模型则主要包括领域模型与类模型（它们都是以类图的形式进行表示）。那么这些模型的构建顺序是怎么样的呢？它们之间又有着什么样的关系呢？下图就清晰明了地说明了这些问题。

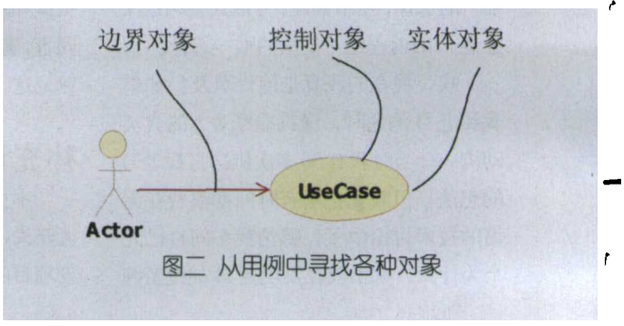


图一 模型演化过程示意图

结合上图，以及我们前面几期的内容，可以总结出模型演变的过程：

1）首先从需求开始，产生领域模型、用例模型这一静一动两个关键的模型。其中领域模型的构建主要是利用“名词动词法”从需求描述中提取相应的领域类，然后再加入主要的属性、识别它们之间的关联关系；对于涉及领域较复杂的系统需求，则应在领域建模前进行业务分析，建立业务模型。而用例模型则是在需求的基础上进行合并，在这个过程中可以进行用户界面原型的构建，从而强化开发人员与用户之间的沟通，发现使用场景；对于用户界面原型的构建，可以采用可视化语言来快速生成、窗口导航图描述，甚至可以使用纸和笔绘制草图的方式实现。

2）接着在用例模型的基础上，进行Robustness分析，通过识别实体对象、控制对象、边界对象，绘制Robustness图，进一步细化，过渡到设计阶段。图二中直观地说明了从用例图中识别各种对象的技巧。对于流



图二 从用例中寻找各种对象

程较复杂的用例，在描述其事件流时，还可以借助活动图来进行表示（关于活动图更多的细节信息，我们将在后面的文章中详细地阐释）。

3）在Robustness分析的基础上，将会发现在领域建模阶段遗漏了的对象，并引入一些与设计更相关的对象。因此，就需要将这些对象进行抽象化，更新在领域模型中，并生成初始的类模型（用类图表示），类模型就是在领域模型的基础上加入了设计阶段的内容。

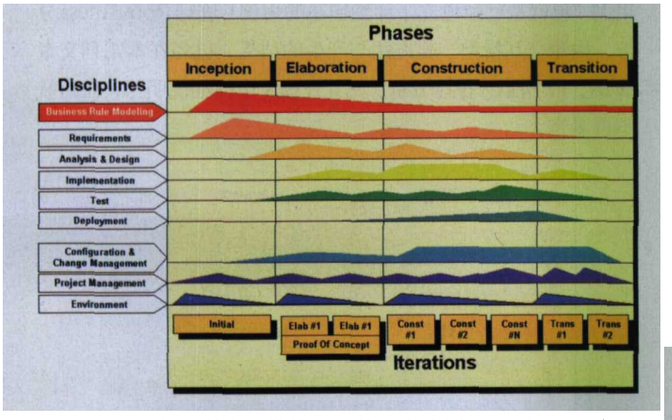
4）在Robustness分析的基础上，就可以对每个用例进行交互建模，通过交互建模能够捕获更多设计细节，包括类的成员方法以及一些更细化的成员属性。从而对类模型进行更新，得到较完善的类模型。

5）最后在较完善的类模型的基础上，进行质量评审，引入基础类，通过设计模式优化结构，以生成高内聚、内耦合，扩展性好的设计方案。

现代软件开发过程

在现代软件开发过程理论中，不管是强调文档的重型方法论，还是强调人与协作的敏捷方法论，都有一个共同的特点，那就是都是一种迭代式的、增量开发过程。

1）RUP：Rational公司（已被IBM公司收于账下）在软件过程的拳头产品，也是重型方法论的典型代表。它的核心理念就是："用例驱动、以体系结构为中心，迭代、增量的软件开发过程"。它将整个软件开发过程分成了四个阶段：初始、精化、构建、交付，而每个阶段又可以细分为多个迭代。在初始阶段的主要任务是理解与分析需求，生成用例模型框架，对优先级较高的用例进行细化；而精化阶段则是完成部分优先级最高的用例开发，并完善出所有的用例模型；在构建阶段分多次迭代，逐渐完成不同优先级的用例开发；而在交付阶段则是进行各种功能、性能测试，对其进行产品化、部署，完成整个系统的开发工作。



2）XP：极限编程，是敏捷软件开发联盟的典型代表。它以最大化发挥人的能量为核心目标，以"小步快走"的逻辑指导开发，其12个最佳实践中的"每日构建"、"用户故事"、"计划游戏"等都充分体现了其迭代式开发的特点。XP中的"用户故事"的作用与RUP中的用例是十分接近的，只不过"用户故事"更加简单、自然一些而已。

3）FDD：特征驱动开发，是已被Borland公司收于帐下的著名建模工具厂商Together公司创始人Peter Coad所创。其通过"特征"来制定开发计划，也是每日构建为核心，强调按特征分步开发与交付。

而且，这些现代软件开发过程还有一个十分相似的共同点，那就是都应用了软件建模技术，RUP有Rose作为支持，FDD有Together作为支持，就连最疯狂的XP也相应地提出了"敏捷建模"思想，如此迷信代码功效的它，也还是认为有必要用纸和笔来构建关键的软件模型。

而大家看到前面关于模型演化过程的描述时，可能会产生一个误解：整个演化过程似乎是瀑布式的，一个模型完成后再演化到下一个模型中去。其实不然，模型的演化应该安排到迭代中去。下面我们就结合"个人图书管理系统"的例子来体会。

结合实例理解开发过程

相对来说，本系列文章中所选用例子"个人图书管理系统"规模较小，因而在整个开发过程中，所需的迭代次数较少。不过，迭代的思想也可以有效地贯彻其中。

第一次迭代

在本次迭代中主要的目的是捕获需求，构建初始的领域模型和用例模型框架。另外，还将对整个开发制订初步的开发计划。根据本阶段的成果，详细地制订下一次迭代的开发计划。

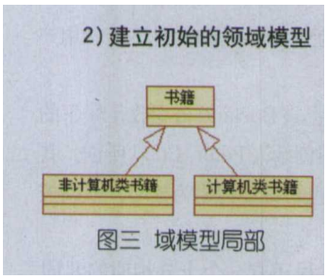
1）捕获需求

千里之行，始于足下。对于软件系统的开发来说，这个"足下"就是需求的捕获工作。关于需求的捕获不是本系列文章的重点，在此就不详细论述，只列出简要的工作步骤。

- 首先应该与项目提出者与决策者沟通，了解系统的开发动机，从中得到"系统需要解决什么问题"，并将这个问题的答案作为整个开发的指导思想，以保证开发出来的结果能够满足项目的真实需要。
- 了解原始系统（手工处理的流程也应视为原始系统）的流程，找到相应的角色，分别针对性地提出问题，总结成为需求描述。
- 这时的需求是零散的，因此应该将其整理成为一个特征，并将其进行编号：

```
FEAT01. 新增书籍信息
FEAT02. 修改已有的书籍信息
FEAT03. 书籍信息按计算机类、非计算机类分别建档
……（详见系列文章II：用例建模）
```

在需求捕获时，可以使用的方法很多，如：用户座谈、问卷调查、现场观摩等。



图三 域模型局部

在捕获需求的同时，就应该逐步利用捕获的信息，构建领域模型。领域模型的产生的过程应该是，发现类，识别关联关系，然后通过与客户之间的沟通来验证其正确性。例如当我们需要验证域模型中以下部分的正确性时，就可以询问："你确认书籍分为计算机类和非计算类，而不在继续细分吗？"

在构建域模型的同时，应对领域专用的术语，特别是容易产生歧义的术语建立相应的词汇表。与此同时，还可以利用数据字典技术对其内容进行一些定义。

词汇表：

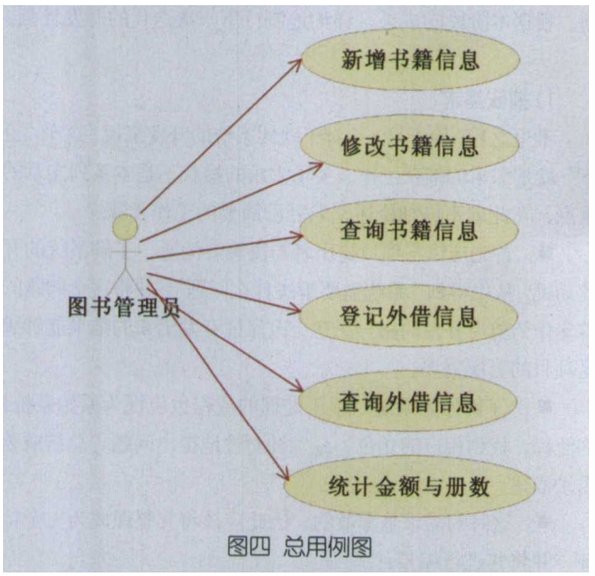
非计算机类书籍：除了计算机类以外的书籍，包括经济、体育、文学等其它所有类。它们的编号规则不相同。

……

3）建立用例模型框架

当一个需求捕获周期完成之后，将对系统的需求有了基本的了解，就可以动手建立用例模型了。在最初的迭代中，必须完成的是一个用例模型的框架。结合本例，在本阶段将完成以下工作：

■ 合并在需求捕获时整理的特征（详见系列文章II：用例建模），绘制出总用例图，并对用例进行编号，以便管理。



图四 总用例图

■ 决定用例的优先级：根据协商，认为最高优先级是UC01：新增书籍信息，UC03：查询书籍信息；次优先级是UC02：修改书籍信息，UC06：统计金额与册数；而UC04：登记外借信息，UC05：查询外借信息的优先级最低。

■ 整理出最高优先级的两个用例UC01、UC03的详细用例描述，而对于其它的用例只需编写出用例描述的框架（通常只需包括用例名称、简要说明、优先级）。

在进行这个工作时，需要注意以下几点：

■ 可以借助"用户界面原型"来编制用例描述，通过用户界面原型的制作也能够更好地将用户融入到用例分析过程中来，以保证用例分析的准确性。而原型制作的方法可以很多，笔者常用的方法就是铅笔＋白纸，一边与用户沟通，一会绘制界面的外观，然后将结果扫描到电脑中保存起来。当然对于一些相对来说重要的界面，也可以使用可视化开发工具（如VB、Delphi）快速绘制，还可以借助Visio这样的工具来绘制。

■ 在用例分析的过程中，经常会发现在领域建模阶段中遗漏的信息。每当发现时，就应该及时地更新领域模型。

■ 还有些时候，在用例分析过程中，会发现无法清晰地判断出用户的使用场景，这时应该及时地询问用户，而不应该主观臆断，去做"业务决策"。这时将重新使用需求捕获的一些方法进行，而且还可以通过与客户的详细交谈，来细化这些场景，以保证其正确性。

4）制订开发计划

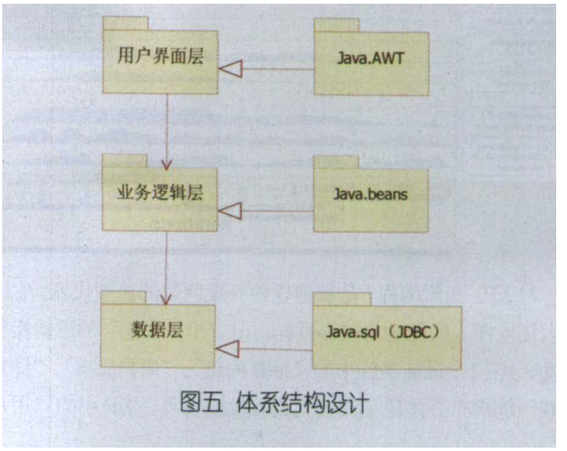
有了这些工作成果，就可以制定初步的开发计划，并且对下一次迭代制作出详细的开发计划。鉴于这项内容不是本系列文章的关注点，在此不详述。

第二次迭代

在这次迭代中，主要的目的是对最高优先级的用例进行分析、设计，并在这个基础上搭建初步的体系结构，通过引入基础类、应用设计模式获得详细的类模型，并在此基础上完成这些用例的开发。最后还应对其它的用户进行进一步的分析，获得详细的用例描述。

1）关键用例的Robustness分析与交互建模

接下来，就应该在用例模型的基础上，对于优先级最高的用例（也就是已经建立了详细描述的用例）进行Robustness分析，构建交互模型。其方法与产生的结果，已经在本系列文章的III、IV两篇（鲁棒分析、交互建模）中详细说明了，在此就不再多言。



图五 体系结构设计

2）体系结构设计

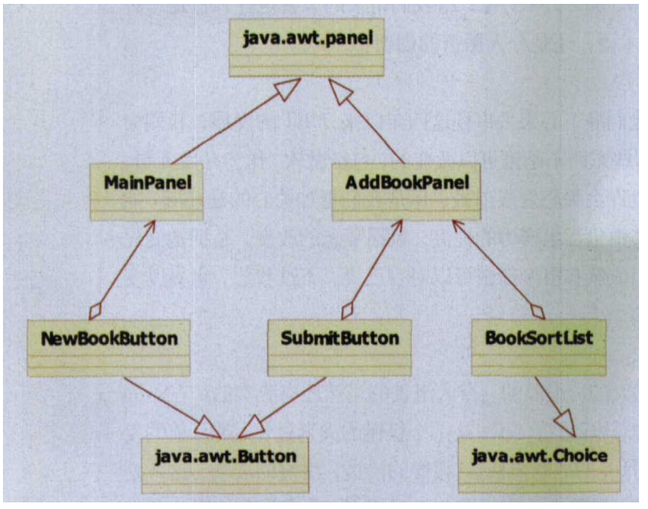
当对主要的用例进行分析、设计之后，就可以进行初步的体系结构设计。有许多体系结构模式可以套用，例如MVC、层次、管道等等。其中层次结构最为常用，在本例中就可以采用。体系结构图，通常使用包图来描述，如上页图5。

在本例中，我们将采用J2SE进行系统开发，因此，这三层分别使用Java.AWT、Java.beans和Java.sql（即JDBC）来实现。我们可以发现，在Robustness分析中识别的边界类就应位于用户界面层；而控制类和实体类都应该处于业务逻辑层；而对于每一个需要存储的实体类，在数据层还应该有一个相对应的类，来负责存储，以实现业务逻辑与留存机制的分离。

另外，值得注意的是，在进行体系结构设计时，应该充分考虑系统待开发的其它功能，以及今后的扩展性，以保证体系结构能够适应这些变化。

3）建立类模型

在图五中所绘制的是一个体系结构的宏观结构，而在每一个层中，都应该包括一个类图，把交互模型中相应的类引入进来。例如用户界面层的类模型就应该如下图所示：



图六 用户界面层部分类模型示意图（未加入属性与方法）

从这里也可以发现，用户界面层的类，来自于Robustness分析；领域模型中的类则应该放置于业务逻辑层；而数据层的类则主要是对业务逻辑层中需要留存的实体类。

4）关键用例的开发与测试

当拥有了细化的类模型之后，就可以进行开发了，本阶段结束后，一个可以运行版本就能交付试用了。值得一提的是，大家可以借助Rose XDE、Together等工具，利用类模型产生部分代码，以提高整个开发的效率。还可以借助JUnit进行自动化的单元测试，提高代码质量。

5）完善用例模型

在交付了第一个版本之后，将接下来进行对其它的用例进行分析，构建详细的描述。并根据这些成果，制订下一次迭代的开发计划。

后面的迭代

而接下来的主要任务是完成所有用例的分析、设计与开发。由于本例中的系统较小，因此可以再分为2次迭代，分别完成次优先级用例（包括UC02：修改书籍信息，UC06：统计金额与册数）和最低优化级的用例（包括UC04：登记外借信息，UC05：查询外借信息）的分析、设计与开发工作。

在这两次迭代中，应根据用例的实际情况，对体系结构进行适当的调整、优化，更新类模型，完成开发与测试工作。其过程类同于第二次迭代，在此就不详述。不过要注意的是，每一次迭代之后，都应该交付一个可以运行的中间版本。

最后的迭代

当完成最后一个用例的开发之后，还应该花一次迭代对其进行产品化和部署，这也是一个较关键的环节。

1）整体测试

虽然在前面，我们可以采用每日构建的方式，“小步快走”的节奏走来。但在最后的一个阶段，还是应该安排验收式的功能测试和性能测试。你可以借助一些JMeter、TestFactory之类的工具进行各种功能、性能、压力测试，以达到考验系统的目的。

2）部署与安装

在开发过程中，应该生成相应的部署图，以便安装维护人员进行系统的部署。

3）产品化

在最后的迭代中，还可对其进行一些产品化包装，例如用户帮助文档的完善、小缺陷的修改、界面与性能的调优等。

结言

在本期中，我们站到了一个更加宏观的角度来审视了这次“OO之旅”，从软件开发过程的角度来总结了前几篇文章中提到的建模技术，以便帮助读者能够在应用于实践时更有章法。也希望看到这里的读者，能够开始在你的项目中去尝试，充分发挥出OO技术的价值。笔者认为，如何更好地在软件开发过程中，正确地应用建模技术、面向对象软件开发方法，提高软件开发质量和开发效率是最重要的，而不应该“形而上学”，单独、片面地应用UML，而导致脱离软件开发的本质。到这里，我们的旅行也快到达了终点站。