实战OO：开启征程（结尾篇）

徐锋／文

引言

时间过得很快，实战OO自第2期与大家见面以来，已经连载了6期。在这6个月里，我们一起从最初的领域分析（域建模，对系统的上下文更深入的理解）、用例分析（用例建模，完成需求分析工作）开始，然后进行鲁棒性分析（从分析过渡到设计的必由之路）、交互建模（详细设计的要点，并开始加入基础类、运用设计模式）勾勒出完整的解决方案；最后使用构件图、部署图来表示系统的物理结构，更好地完成系统的部署与实施工作。

并且在讲解部署与实施工作之前，我们还一起从软件开发过程的角度对所有的工作进行了一次认真的总结，相信这些内容可以有效地帮助读者从宏观、微观两个角度都建立起较清晰的认识。但这么多东西如何真正有效地应用到日常的实践中去呢？本文将围绕这个话题与大家共同探讨与交流。

另外，在前面的文章中，我们先后接触了UML9大图中的类图、对象图、用例图、交互图（包括顺序图、协作图）、构件图、部署图等7种。其中遗漏了的就是活动图和状态图。对UML有一些了解的人一定会说活动图和状态图都是很重要的呀，为什么没有介绍呢？别急，的确它们是UML中十分有价值的两种图，但它们是细节、是分支，为了让大家从主线条的角度来掌握知识，因此就将它们放到了最后进行介绍。

面向对象的流程图——活动图

活动图的应用非常广泛，它既可用来描述操作（类的方法）的行为，也可以描述用例和对象内部的工作过程。活动图是由状态图变化而来的，它们各自用于不同的目的。活动图依据对象状态的变化来捕获动作（将要执行的工作或活动）与动作的结果。活动图中一个活动结束后将立即进入下一个活动（在状态图中状态的变迁可能需要事件的触发）。

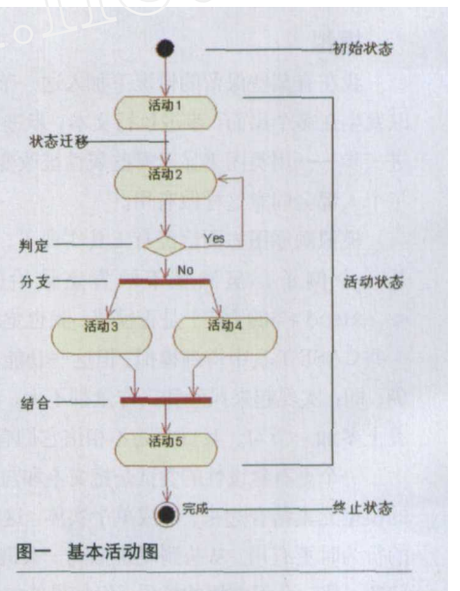
活动图的形式与应用

根据活动图的简繁程度划分，包括两种不同的形式：

1）基本活动图：

活动图包括了初始状态、终止状态，以及中间的活动状态，每个活动之间，也就是一种状态的变迁。在活动图中，还引入了以下几个概念：

- 判定：说明基于某些表达式的选择性路径，在UML中使用菱形表示。
- 分叉与结合：由于活动图建模是经常会遇到并发流，因此在UML中引入了如上图所示的粗线来表示分叉和结合。

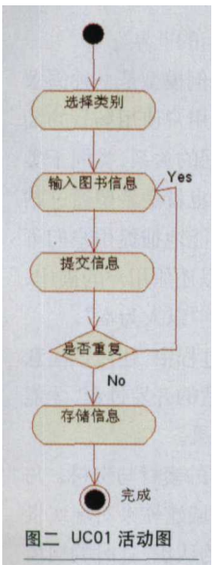


图一 基本活动图

从上面，大家也可以看出，活动图与我们在结构化分析与设计中经常使用的系统流程图十分的相近。

2）实例说明

图二就是“UC01：新增图书信息所对应”的活动图。从图二中我们可以看出，绘制的方法与思路与流程图的制作是十分接近的。它可以将基本事件流、扩展事件流图形化的表示出来，因此对于较复杂的用例而言，我们可以在编写事

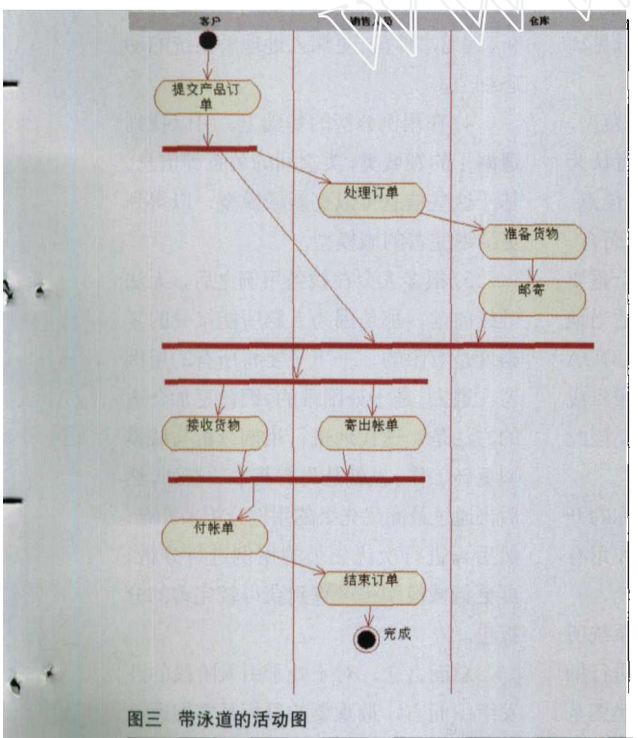


图二 UC01 活动图

件流描述的同时，附上一张活动图，就可以帮助读者更清晰准确地掌握整个用例的事件流程。

3）带泳道的活动图

在前面说明的基本活动图中，虽然能够描述系统发生了什么，但没有说明该项活动由谁来完成。而针对OOP而言，这就意味着活动图没有描述出各个活动由哪个类来完成。要想解决这个问题，可以通过泳道来解决这一问题。它将活动图的逻辑描述与顺序图、协作图的责任描述结合起来。下面我们就一起来看一个使用了泳道的例子（图三）。

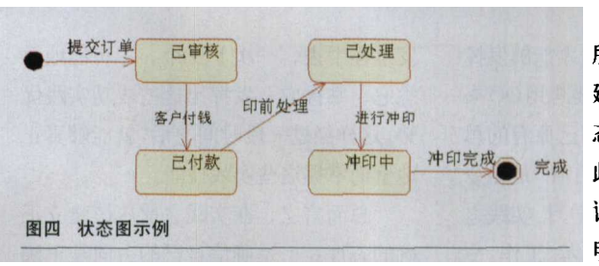


图三 带泳道的活动图

对于带泳道的活动图而言，在不同的阶段可以发挥不同的作用：

1）业务建模阶段

有业务分析经验的读者将容易发现上图与“跨职能流程图”十分接近，说明活动图也是能够有效地应用于业务建模的。在用于业务建模的时候，每一条泳道表示一个职责单位（可以是一个人，也可以是一个部门），该图能够有效地体现出所有职责单位之间的工作职责，业务范围以及之间的交互关系、信息流程。



图四 状态图示例

为了更加有效地利用该图进行业务流程分析，应收集企业的组织结构图、岗位说明等信息，并且有针对性地与他们沟通、调研。每个职责单位泳道中每个活动的具体需求的权威信息应来源于该职责单位，就避免了从其它职责单位获取不准确信息的风险。

2）系统分析设计阶段

到了分析后期以及系统设计期间，这时活动图所针对的就不再是业务流程中的职责单位，而是对象。也就是每一个对象占据一个泳道，而活动则是该对象的成员方法。不过通常在系统分析设计阶段应用带泳道的活动图的情景较少，因为顺序图（也称为时序图）会更好地体现与表达其间的关系。活动图更适合于对其流程进行概述，最常用的场景是通过活动图对用例描述中的事件流部分进行建模。当用例的事件流较复杂、分支较多的时候，一张清晰明了的活动图能够帮助开发人员、涉众更好地理解。

业务状态的迁移——状态图

状态图（State Diagram）用来描述一个特定对象的所有可能状态及其引起状态转移的事件。大多数面向对象技术都用状态图表示单个对象在其生命周期中的行为。一个状态图包括一系列的状态以及状态之间的转移。在我们的例子中，所涉及到的状态变化较少，所以不一定需要进行状态建模。为了能够让大家对状态图有一个感性的认识，在此我们就以数码冲印店的订单状态图为例简单地说明，更多的信息可以参考专门的书籍与文章。

正如上图所示，状态图包括以下部分：

- 状态：又称为中间状态，用圆解矩形框表示；
- 初始状态：又称为初态，用一个黑色的实心圆圈表示，在一张状态图中只能够有一个初始状态；
- 结束状态：又称为终态，在黑色的实心圆圈外面套上一个空间圆，在一张状态图中可能有多个结束状态；
- 状态转移：用箭头说明状态的转移情况，并用文字说明引发这个状态变化的相应事件是什么。

一个状态也可能被细分为多个子状态，那么如果将这些子状态都描绘出来的话，那这个状态就是复合状态。

状态图适合用于表述在不同用例之间的对象行为，但并不适合于表述包括若干协作的对象行为。通常不会需要对系统中的每一个类绘制相应的状态图，我们可以在进行业务流程、控制对象、用户界面等方面的设计时使用状态图。

如何学以致用

可能洋洋洒洒几万字之后，会博得不少读者的共鸣与认同。但即使是这些有共鸣的读者，可能也只是看看、想想，合上杂志之后，工作依然如旧，总认为：“也许理论是理论，实践是实践，OO还是离实践太远。”因此，笔者认为，要真正将文中理念与方法贯彻到实践中，需要克服恐惧心理，加深对理论知识的理解。为了帮助这些有兴趣实践的读者更好地学以致用，下面就针对几个关键方面做一详细的说明。

建立正确的态度

面向对象、UML不仅仅是名词、方法，而是一种思想，只有充分的理解、掌

握，并且正确、灵活地应用，才能够发挥出它们应有的价值。因此，要想用OO来武装自己，首先就需要对自己原有的思维模式发起冲击。而“挑战习惯”是十分困难的，程序员们在多年的学习、实践之中或多或少都会养成一些思考、工作、交流、学习的习惯，而且这些习惯也曾经有效地帮助他们完成了工作，获得了成功的感觉，一朝需要放弃，谈何容易。

有一点需要十分明确：对于任何一个改进来说，都将面临着抛弃原来的工作习惯，将采用新的方式方法来工作，这通常会产生一个“先抑后扬”的趋势。

- 先抑：由于采用了新的方式方法，将需要更多的学习、思考的时间；还可能会遇到不熟悉的环境与情况，需要更多的时间去解决；而且还可能遇到那些超出自己经验范围之内的问题，以致对整个工作产能带来很大的负面影响，可能造成产能和质量的下降，从而给采用新方法的开发人员（甚至是组织）带来巨大的心理压力。
- 后扬：只有走过“先抑”的那段必然的过渡阶段，才可能将理论进行实践化，将学习和思考放置在具体的工作中，更多的努力也将帮助开发人员和组织深入地理解新方式方法，终于它们发挥了巨大的正面影响，把所有的利益与好处充分地体现出来。

因此，在这种情况下，就需要开发人员能够树立信心，穿过充满艰险的“草地、雪山”，才可能获得“二万五千里长征”的胜利。许多开发人员出于“现有的方法也能够正常的使用，没有必要和自己过不去”的考虑，而没有能够有效地跨越这个心理障碍，因此从一开始就没有建立必胜的信心。

另一方面，开发人员常常秉承“唯美主义”思想，认为在不好与完善之间没有任何中间状态，总是希望自己能够一步到位，将OO技术运用得炉火纯青，使开发效率与结果得到大大的提升。然而，这样的期望通常是不现实的，这就会给使用者带来很大的挫折感，从而影响到实践的效果。而正确的态度则是：将OO与UML的实践，逐一地加到自己日常的开发过程中去，一次1-2个，尽可能地熟悉它，掌握它，发挥出这些成功实践优势。这样经过一段时间之后，就能够真正地全面掌握这些实践技能。

总而言之，在实践之前应该建立正确的态度：一是确信自己的方向是正确的，愿意为这个过程付出相应的代价；二是认识到不可能一蹴而就，克服浮躁的思想，不断地在实践中提高。有了这样的思想基础，成功的机会就会大大提高。

导入OO的最佳点

既然应该采用循序渐进的学习与实践策略，那么，对于初次实践OO的开发人员来说，从哪个环节着手导入最合适呢？也许很多人会说，从OOP（面向对象编程），但笔者不敢苟同。OOP已经出现多年，而且现在所有主流的开发工具均是纯OOP的，但还是出现“类是一个框，什么都往里装”的尴尬局面，说明这不是一个好的起点，容易进入“只见树林、不见森林”的误区。

毕竟OO是一种新的软件开发范型，对其思想精髓的掌握与理解，笔者认为从OOA开始引入是更为合适的最佳点。具体到本系列文章所讲述的技术而言，主要包括域建模、用例建模这两个重要环节，当然也可以根据实际情况适当地引入鲁棒分析作为补充。首先引入OOA还有一个好处是，OOA所涉及的项目成员相对较少一些，主要是SA参与，因此容易建立共识和管理。

在将域建模、用例建模引入你的开发实践中时，应该对它们的本质、作用有着清晰的认识与理解：

1）域模型：域模型反映的是系统所涉及的客观世界的所有物体，对其进行抽象，以类图的形式展现出来。这个类图是整个系统开发的一个基础，有了这个类图，才能够更加有效地在分析和设计阶段来扩展它。因此，在进行域建模时，并不需要苛求其精确性，因为它只是演化的基础。在开发过程中，还有许多机会来更新与完善它们，在本阶段的关键还是用类的思想来看世界。有一个很恰当的比喻：“域模型是戴上OO眼镜后的世界”。

2）用例模型：用例模型是一种需求合成分析技术，它是用户使用场景的抽象，它与使用场景之间的关系，类同于类与对象之间的关系。也就是希望通过站在用户的角度，尽可能地捕捉用户们可能进行的使用场景，以还原用户的操作，使得软件设计与开发“以人为本”。

另外，在实施的过程中，还应该注意建立正确的认识、合适的开发过程，才能够收到更为良好的效果。

1）首先基于原有的素材与资料，与领域专家一起，通过域建模来对现实世界建立宏观的、感性的认识。这时的域模型不一定很完善，也不要求很完整。

2）有了一个初步的域模型之后，就应该结合需求调研对用户的需求进行分析、合成、整理，编写最初的用例规约。然后就应该基于最初的用例规约进行优先级分析，选出具有较高优先级的用例。

3）然后对优先级较高的用例进行分析，编写事件流，更深入地理解系统的动态细节。

4）在用例分析的基础上，可以找到遗漏了的领域类，类之间的关系等信息，基于这些信息可以更新域模型，以得到更清晰完善的域模型。

5）很多人会在找到用例之后，无法继续向前，那是因为人脑所能接受的复杂度是有限的，一下子要将所有的用例考虑进去，是十分困难的。因此更加合适的方法是，迭代地进行用例分析与域模型更新工作。也就是先有基本域模型，然后再通过最高优先级的用例分析来补充，然后再进行次优先级的用例进行分析，再更新域模型……直到获得较完善的分析包。

总而言之，对于处于引入阶段的开发组织而言，最重要的目标是掌握通过OO的视角来观察与抽象现实世界的方法，并能够有效地利用用例分析技术对需求进行分析、管理、跟踪，有效地利用静、动态两个模型，逐渐地深入。通过本阶段的努力，将能够更好地整理出需求，对开发进度进行更加良性的计划与监控。你可以在这样的基础上，仍然采用传统

的设计与实施方法。虽然这时还不足以体现出成功，而且还会产生许多这样那样的问题，还无法使得编写出来的代码真正“OO”，但却能够对OO精神有了宏观的掌握，并充分享受到用例分析给项目带来的益处。

如何渐入佳境

如果通过体系化的培训、加强信念的宣传，并引入系统化的同级评审实践，那么你的团队将会在1-2个项目完成之后，较熟练地掌握了OOA的思路与实践。这时，就给你的征途打下了坚实的基础。但要知道：引入OOA之后，并不容易马上收到效果，这个时候最重要的是坚持。

当你的团队对OOA已经有了较好的认识与理解之后，就可以开始逐渐过渡到更多人员参与的OOD阶段，这时你已经有了一些“传教士”（SA们）。

那么如何进行OOD的普及实践呢？笔者曾经见过许多组织、开发人员都以掌握Rose、Together等等的工具作为掌握OOD的必由之路，从而陷入了大量CASE工具的细节中去。其实这是一个常见的误区，让开发人员过早地陷入细节，却对宏观的东西缺失过多，甚至形而上学，从而难以继续，不得不中途放弃。

针对这种情况，笔者认为，在引入OOD之时，也应该采用循序渐进，逐步深入的方法。也就是说，与用例分析的深入同步，一开始对最高优先级的用例进行分析与设计，从鲁棒分析开始，然后进行交互建模，借助活动图、状态图等其它手段深入地进行细节设计，从而得出一个系统类图，开发出第一个版本的软件；然后再对次优先的用例进行同样的分析与设计，更新系统的类图，局部进行调整与重构，开发出第二个版本的软件……直到系统开发完成。

到了这个阶段，再让项目组队中的设计、开发人员开始学习各种框架、库，完成基本设计；再学习设计模式，优化设计，就能够获得较好的效果。而在这个过程中，阅读框架、库的类图，也是很好的学习方式。一方面以OO的视角重新审视这些基础资源，能够更好理解OO，更好地结合；另一方面也可以提高设计人员的设计水平。

在这个阶段中，也应该在加强培训的同时，引入设计评审，通过正式的、非正式的交流将这些信息与技能强化下来。通过这样的过程，开发组织将能够真正用OO武装起来，体现出其价值。

过程定义与改进很重要

软件行业一直都在为其它产业的信息化在做贡献，但是回过头来，许多我们向传统企业鼓吹的思想和观点却没有在自己企业中运用。有许多例子中体现出来的管理哲学是那样的浅显，但却长期不能够被我们自身掌握、运用。例如，在实施ERP的过程中，我们会让企业进行BPR（企业流程重组），对于规范度较不成熟的企业还会对其进行BPD（企业流程定义），因为信息系统只是起到固化流程的作用。但软件企业却长期对自身的流程—软件过程没有足够重视。再如，在企业引入AutoCAD提升设计效率时，只是让优秀的设计师能够更轻松更省时，并不能够让蹩脚的设计师成为优秀的设计师。但在软件企业却一直认为跟踪最先进的技术，就能够使团队能力大幅提升。

这些理念都告诉我们，我们也应该先进行流程的定义与重组，我们也应该将知识水平的培育作为更大的重点，而非仅是简单地引入CASE工具。

我们的流程定义与重组，就是软件过程定义和改进。应该在项目团队内、企业组织中，制订适合OO开发的软件过程，并且由简到繁，由粗到细，帮助每一个开发人员真正的理解与融入。另外，还应该对大家的分析与设计技能进行有计划的培训。总之，优秀且适合的人加上符合实际情况的过程，才是成功之道。

结言

到此为此，本系列文章到止就结束了。在这个系列中，笔者尝试从宏观的角度，将对OO和UML如何与实践结合的认识与大家分享，希望能够对大家有所帮助。在文章的结束之前，我还想再次提醒大家：OO技术已经出现了20多年，各种成功的实践告诉我们它是软件开发的必然发展趋势；UML也已经出现了近10年，已经成为了软件建模的产业标准；而用例分析技术更是已有近40年的历史。对于这些强大的工具，我们没有理由忽视，更没有理由在没有深入运用之前妄加批评。更加务实的态度是在工作中实践，在实践中总结，在总结出升华，才能够真正利用巨人的肩膀让我们站得更高、看得更远。

正所谓千里之行，始于足下。光有理论上的认知与理解，是无法对产能有所提高的。要想真正地享用这些技术给我们带来了益处，唯一的办法就是实践、实践、再实践。面对信息时代的迅速发展，软件开发速度的重要性早已超越软件的运行速度，如何广征博引，博采各领域之众长，提高软件开发管理理念，是整个产业最重要的问题。在UML建模、MDA、产生式编程等一系列新技术、新概念迎面而来的今天，如何快速地迎浪而上，已是摆在每一个中国软件人面前的重要课题。只有努力地实践最先进的理念与方法，全面提升成熟度，才是振兴我国软件产业的必由之路。少一分批评，多一分学习；少一分浮躁，多一分专注；在实践中总结、创新、超越才是真正的发展之道。

鉴于杂志的特点，笔者无法过细地阐述更多的技术实现细节，加上自身水平的限制，一定还存在许多不尽人意的地方，也希望诸位读者能够与我多多交流。笔者正在着手编写《实战OO》一书，希望能够通过它为各位带去更多的帮助。也希望倾听到大家的建议与意见。欢迎大家与我联系：fjxufeng@msn.com。

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。