实战OO：部署与实施

徐锋／文

引言

在前面，我们建立了一静一动两个模型：即描述系统环境的“概念模型”和反映系统行为的“用例模型”。并在用例模型的基础上，结合Robustness分析、交互建模等方法对系统进行了更深入的分析与设计，从而将设计元素引入到概念模型中，设计出整个系统的类模型。

这时，我们将拥有一个相对较完整的类模型：其中包括最早发现的，反映客观世界中的事物的实体类（分析类）；为完成人机界面的边界类（通常是界面类）；还有那些为了实现业务逻辑的控制类；以及那些为了使整个构架更合理、系统扩展性更好而引入的设计类。从以上的描述中，我们也显然可以发现一个系统的类模型通常都包括大量相关联的类，类与类之间的关系错综复杂，难以一眼看到全局。

这样的局面会给开发，部署等工作带来一定的困难。因此，我们需要借助另外两个有用的模型来缓解这一问题：即表现系统物理结构的构件图和说明系统部署结构的部署图。下面我们就一起来认识一下它们。

系统物理结构——构件图

仅有类模型的话，会让开发团队陷入到“只见树木，不见森林”的尴尬局面。因此我们需要在此基础上进行整理与融合，这就是构件模型。

什么是构件

根据《UML参考手册》的定义可以得知：“构件是系统中可替换的物理部分，它包装了实现而且遵从并提供一组接口的实现。”通常来说，每个构件可能包含很多的类，并实现很多的接口。构件可以分为三种类型：

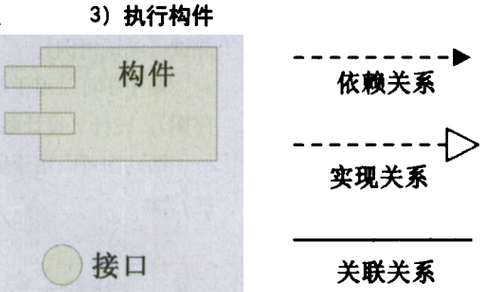
1）实施构件

这类构件是构成一个可执行系统必要和充分的构件。例如动态链接库（DLL）、可执行文件（EXE），另外还包括如COM+、CORBA及Enterprise Java Beans、动态Web页面（如Java Server Page等）也属于实施构件的一部分。

2）工作产品构件

这类构件主要是开发过程的产物，包括创建实施构件的源代码文件及数据文件。这些构件并不是直接地参与可执行系统，而是用来产生可执行系统的中间工作产品。

3）执行构件



图一 构件图的基本元素

这类构件是作为一个正在执行的系统的结果而被创建的，例如由DLL实例化形成的COM+对象。

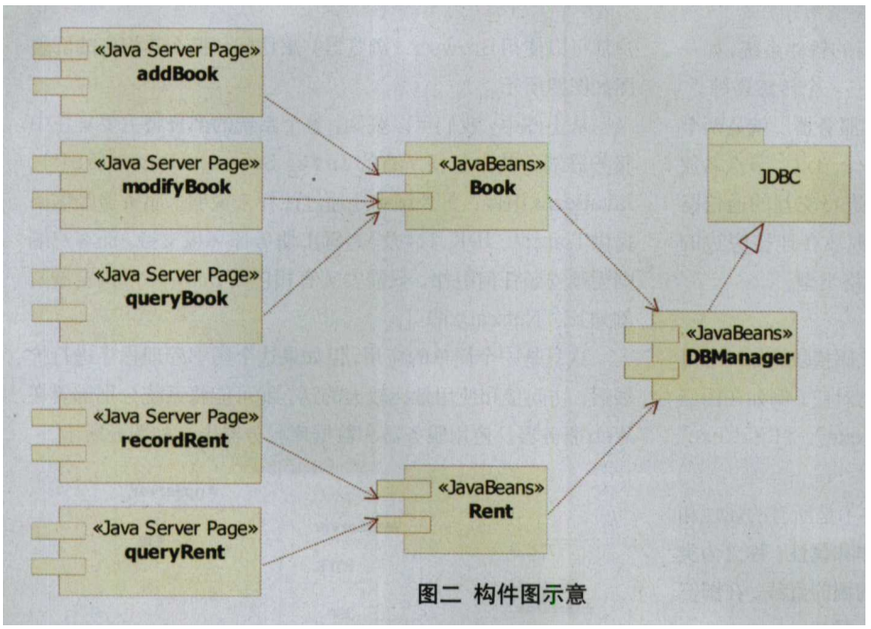
构件图及其作用

构件图是面向对象系统在物理方面进行建模时要用到的两种图之一。它可以有效地显示一组构件，以及它们之间的关系。构件图中通常包括构件、接口以及各种关系。

通常来说，我们可以使用构件图完成以下工作：

- 对源代码进行建模：这样可以清晰地表示出各个不同源程序文件之间的关系。这可以帮助开发团队更好地理解各个源代码文件之间的对应关系，例如表示C++代码里.h和.cpp文件之间的关联与包含关系。
- 对可执行体的发布建模：如图一所示，将清晰地表示出各个可执行文件、DLL文件之间的关系。这可以帮助开发团队更好地理解整个系统中各个执行部分之间的依赖与关联关系。
- 对物理数据库建模：用来表示各种类型的数据库、表之间的关系。这可以帮助开发团队对数据库结构之间的关系有一个更清晰的认识。
- 对可调整的系统建模：例如用于对于应用了负载均衡、故障恢复等技术的系统建模。

在绘制构件图时，应该注意侧重于描述系统的静态实现视图的一个方



图二 构件图示意

面，图形不要过于简化，应该为构件图取一个直观的名称，在绘制时避免产生线的交叉。

使用构件图

那么我们通常在什么情况下使用构件图？又是如何利用构件图来帮助我们工作的呢？最常见的一种情况是，通过构件图对类模型进行有效的整合，使得开发团队知道最终系统是由哪些可执行的构件块组成，让其从类的森林里抽身出来。

例如，图二就是本系列讲座中“个人图书管理系统”的一个构件图的局部示例。

从图二的构件图中，我们可以看得出，整个系统还是根据“用户界面”、“业务逻辑”、“数据访问”的三层结构进行组织的：

■ 用户界面层：主要采用了JSP页面，实现最外层的用户界面。在上图中，我们通过了构造型<<Java Server Page>>来说明构件形式。其大多数都是由边界类组合在一起而成的。

■ 业务逻辑层：由于整个系统并不大，因此对于中间的业务逻辑层在实现上主要采用了JavaBeans技术。我们使用了构造型<<JavaBeans>>来说明其形式。而且大家很容易发现这些构件将主要由实体类组合而成。

■ 数据访问层：为了保证整个系统的稳定性和可扩展性，将所有的数据访问操作封装在一个JavaBeans中（对于更大型的应用，则可能封装成为一个EJB），在本例中，我们仍然以<<JavaBeans>>构造型来描述它。为了能够更清晰地表现出其来源，图中还标明了，它是JDBC的一个实现。

上图显然是对可执行体的发布建模，帮助开发团队能够知道这些一个个可执行块之间的关联关系，从而达到统一认识的目的。

进行编码

关于编码的具体细节，并不是本系列文章所要关心的重点，因此这方面的内容将不作详细的展开描述。在此主要结合构件图与类图模型简单阐述如何过渡到编码工作。

首先，每一个构件都应该包括一系列的类，在进行编码之前应该将这个对应关系明确下来。其次，每一个构件都将是一个相对独立、完整的代码块，可以作为项目管理、测试的一个管理块。第三，有了构件之间的关系，以及类的相应描述，可以通过自动化工具产生代码的框架，然后再手工编写实现代码。

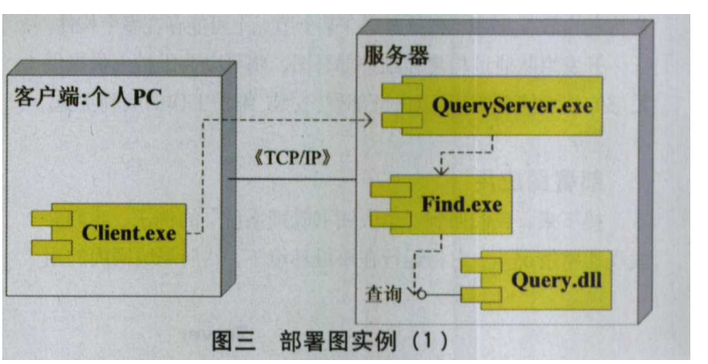
通过构件模型和类模型，可以有效地实现分工与协作，类模型设计的水平也决定了是否能够成功地进行项目工作分解。另外，如果能够较成功地完成这一点，也将大大提高外包管理能力。

系统部署结构——部署图

我们通过构件图将能够理解系统的物理组成结构，但是它并没有办法体现出这些物理组成部分是如何反映在计算机硬件系统之上的。而部署图正是用来弥补这个不足的，它的关注点就在于系统如何部署。

部署图初步

部署图，也称为实施图。和构件图一样，是面向对象系统在物理方面建模的两种图之一。构件图相对来说，是用于说明构件之间的逻辑关系，而部署图则是在此基础上更近一步，描述系统硬件的物理拓扑结构以及在此结构上执行的软件。部署图可以显示出计算节点的拓扑结构和通信路径、节点上运行的软件构件，常常用于帮助理解分布式系统。例如图三就是一个远程查询系统的部署图示例。



图三 部署图实例（1）

很显然，这样的图示将可以使系统的安装和部署工作变得更加简单。从上面我们也可以看出，在部署图中包括以下一些关键的组成部分：

1）节点和连接

节点（Node）是存在于运行时并代表一项计算资源的物理元素，一般至少拥有一些内存，而且通常具有处理能力。

最常见的节点是一个物理设备以及其上运行的软件系统，如一台 Unix 主机、一个 PC 终端、一台打印机、一个传感器等。

如图三所示，“客户端：个人 PC”和“服务器”就是两个节点。在 UML 中，使用一个立方体表示一个节点，节点名放在左上角。节点之间的连线表示系统之间进行交互的通信路径，在 UML 中被称为连接。通信类型则放在连接旁边的“<<>>”之间，表示所用的通信协议或网络类型。

2）构件和接口

在部署图中，构件代表可执行的物理代码模块，如一个可执行程序。逻辑上它可以与类图中的包或类对应。例如在图三中，“服务器”节点中包含“QueryServer.exe”、“Find.exe”和“Query.dll”3 个构件。

在面向对象方法中，类和构件等元素并不是所有的属性和操作都对外可见。它们对外提供了可见操作和属性，称之为类和构件的接口。界面可以表示为一头是小圆圈的直线。在图三中，“Query.dll”构件提供了一个“查询”接口。

部署图的作用

部署图通常可以用于以下情况的建模工作：

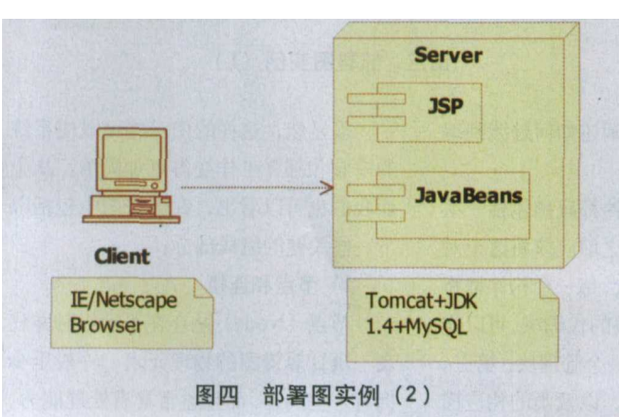
■ 对处理器和设备建模：这通常包括对单机式、嵌入式、客户/服务器式和分布式系统的拓扑结构的处理器和设备进行建模。其实也就是将系统的工作范围很明确地表示出来。通常采用以下策略：1）识别系统中的计算元素，并为每个元素建模为节点；2）如果这些元素代表一般的处理器和设备，则使用相应的构造型；3）与类建模一样，可以对节点进行属性和操作的设置。

■ 对构件的分布建模：这通常是用来可视化地指定其构件的位置与协作关系。图三所示的例子就是一个对构件的分布建模的例子。通常采用以下策略：1）将所有有意义的构件，分配到一个特定的节点上；2）充分考虑在某个节点上可能存在多个构件。

开发组队通过构建和维护部署图，将可以为维护人员提供足够的技术信息支持，以保证部署、安装、维护工作的顺利实施。

部署图应用

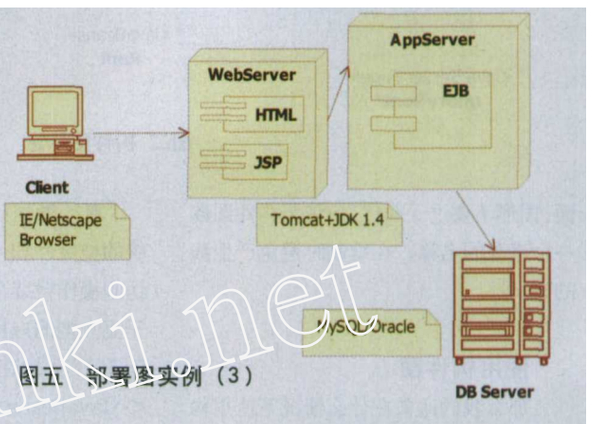
接下来，我们回到“个人图书管理系统”的例子，我们会发现通常情况下，它将运行在单机环境下，然后局域网内的用户都可以使用 Browser（浏览器）来访问，那么其相应的部署图如图四所示。



图四　部署图实例（2）

从上图中，我们可以获知，整个系统的部署将主要集在中服务器端，其中包括所有的 Java Server Page（JSP）、JavaBeans 组件；另外图中还通过注释来说明，服务器端需要提供 Tomcat、JDK 1.4 及 MySQL 服务库环境支持。而客户端则无须安装任何组件，只需安装有相应的 Browser（浏览器），例如 IE、Netscape 即可。

这只是一个简单的应用，但如果这个图书管理程序进行扩展后，访问量和使用量比较大的话，就可能将系统分别部署在 Web 服务器、应用服务器和数据库服务器上，如图五所示。



图五　部署图实例（3）

在上面的部署方案中，我们将静态页面（HTML 文件）、动态页面（Java Server Page）存放在 Web Server 中；将业务逻辑对象放置在专门的应用服务器中，由于这时已经是在不同的机器上，需要使用远程的对象访问技术，因此在此需要使用 Enterprise Java Beans（EJB）技术。

而图中使用注释标记符标示出来的各个节点所需的软件、硬件配置将会为安装、部署和维护人员提供足够的信息。

结束语

这一次我们一起探讨了如何借用 UML 中的两种用于描述物理结构的构件图和部署图来帮助开发团队、部署团队更好地对系统的全景建立统一的认识。不仅能够为开发团队提供好处，其最大的作用在于给今后的部署人员提供了大量有价值的信息。

而它们之间的区别在于关注点：构件图相对而言，更加重视这些物理模块之间的关系，而部署图则更加重视它们的具体位置。

讲到这里，我们已经完成了一次从需求分析到系统设计，最后过渡到具体实现的旅行。但许多读者看到这里，一定会问活动图、状态图都去哪了？不重要吗？不是的，这两个模型在开发过程中有着十分重要的价值，笔者还没有谈到的原因是因为它们相对来说更注重细节，而前面的主框架中并无需过多的依赖于它们。鉴于它们巨大的作用，我们将在下一期专门描述这两个模型的具体应用方法，下个月再见。