基于UML的J2EE组件建模的研究

何静媛[1] 符云清[2] 唐蓉君[1] 鲜晓东[3]
(重庆大学计算机科学与工程学院 重庆400044)[1] (重庆大学网络教育学院 重庆400044)[2]
(重庆大学通信工程学院 重庆400044)[3]

摘 要 本文从J2EE平台的多层分布式体系构架出发,探讨了使用UML(统一建模语言)的标准规范及扩展机制对EJB组件和Web组件的建模思路,并针对UML在分布式系统建模中存在的局限,提出改进方案。

关键词 组件建模,J2EE,EJB,分布式系统

Research on Modeling Component Based on J2EE

HE Jing-Yuan[1] FU Yun-Qing[2] TANG Rong-Jun[1] XIAN Xiao-Dong[3]
(College of Computer Science and Engineering,Chongqing University, Chongqing 400044)[1]
(College of Distance Educatin,Chongqing University,Chongqing 400044)[2]
(College of Communication Engineer,Chongqing University,Chongqing 400044)[3]

Abstract In this article,a modeling mehod for EJB component and Web component by using standard rules and extend mechanism of UML is analyzed. Based on J2EE distributed system, an improved schema is brought forword to solve the limitation of traditional method of UML modeling.

Keywords Compnent modeling, J2EE, EJB, Distributed system

1 引言

对于企业级的分布式应用开发项目而言,系统的建模思路在很大程度上决定了整个系统开发的效率和应用前景。UML作为OMG组织确定的唯一建模语言,统一了Booch,Rumbaugh和Jacobson的面向对象方法的概念、符号和表示模型,对实时系统、分布式系统等都有很强的的表达能力,但对分布式的组件建模还没有一个成熟的理论[4]。本文利用UML的扩展机制,对J2EE的Web组件及EJB组件建模进行探讨和研究,从而提出了分布式系统组件建模的普遍规律和应用规则。

2 J2EE的多层分布式系统框架模型

J2EE是由sun公司发布的用于开发和部署多层结构的、分布式应用系统平台,它为系统开发者提供了一个灵活、可靠的环境,让开发者专注于系统的业务逻辑,而将复杂的安全性、事务、数据库访问等服务交给特定的服务厂商完成[1]。J2EE平台的系统构架如图1所示。

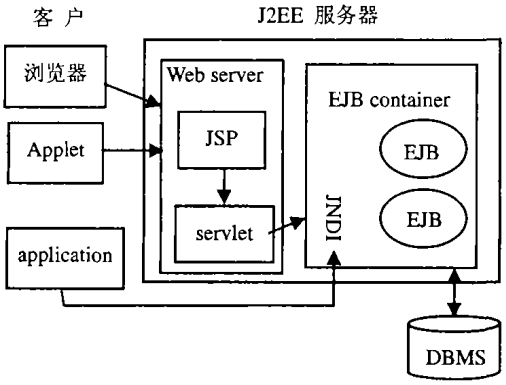

图1 J2EE系统构架

J2EE平台支持的java程序类型有:Applet、Application、Web组件和EJB组件,其中,Applet和Application为客户端,作为分布式应用的第一层(前端);Web组件(包括Servlet和JSP)运行在Web容器内,而EJB(enterprise java bean)组件运行于EJB容器中。Web组件和EJB组件构成第二层,这是应用服务层,所有业务逻辑应该放在该层上,该层是分布式系统的核心;DBMS则是属于第三层(数据存储层)。在多层的分布式系统中,我们应将主要精力放在应用服务层。因此,本文将对该层的Web组件和EJB组件的建模做详细的讨论。

3 UML扩展机制与分布式系统建模

在J2EE分布式系统建模中,当定义结构视图时,仅强调对象的属性和方法是不够的,分布式对象接口应该成为对象模型的一个集成部分。这样,可以将一个接口指派到一个类中,其他进程就可以在该类上进行远程调用。所以,当对于这些分布式的组件建模时,必须将描述业务功能的传统模型和描述分布式的组件模型相结合。但UML的标准表示并不能满足描述分布式组件的模型需求,因此,我们必须要使用UML的扩展机制,为某些模型元素增加新的语义,构造出新的元素[2]。

在一个良定义的系统中,其接口定义应该是规范的,UML的扩展机制用版型《interface》来表示一个接口[3]。调用该接口的类和接口之间使用带箭头的虚线连接,其间是依赖关系,如图2所示。图中的《EJB interface》是笔者对《interface》版型的扩展,用来专指EJB组件内的接口。

在对J2EE系统组件建模时,传统的面向对象的系统建模思路已经难以表达"分布式"的特征,针对J2EE平台的特点,笔者提出了"分开"建模的主张:对Web组件和对EJB组件分别进行建模。在J2EE体系构架中,Web组件和EJB组件在不同的容器内运行,Web组件通过JNDI服务使用EJB对象的业务逻辑。严格意义上,它们是J2EE服务器内部的一种

Client/Server 结构。因此单独对两者进行建模，可以将表示逻辑和业务逻辑在多层体系构架中清晰地划分出来。

4 EJB 组件的建模

EJB 组件技术是 J2EE 平台规范的核心，该技术将分布式系统的应用逻辑和业务规则封装在 EJB 对象中，因此 EJB 组件建模的优劣，直接影响着分布式应用的功能。

每一个 EJB 组件是由三方面构成的：EJB 类、Home 接口和 Remote 接口。其中 EJB 类实现业务方法及 EJB 对象的生命周期方法。Home 接口完成对 EJB 对象的创建、定位和删除等操作；Remote 接口作为 EJB 对象的方法访问接口，提供对业务方法的调用。当客户向 EJB 容器提出请求时，必须要通过 JNDI 获得 Home 接口的一个实例，通过调用该实例上的 create()方法，获得 Remote 接口对象，最后，通过它来访问特定 EJB 对象的业务方法。

EJB 对象本身可以分为：Session bean 和 Entity bean 两类。Session bean 代表与客户程序之间的一个短暂会话，通常 Session bean 的方法包括一些与数据库直接存取无关的业务方法；Entity bean 则实质上代表了数据库中的一行数据及作用于该数据的操作。因此，根据 EJB 组件的组成和特色，利用 UML 建模时，本文提出以下观点：

1)类图中的每一个 EJB 类，都应该有相应的 Home 接口和 Remote 接口。如图2所示，它是一个 EJB 组件的内部构成图，图中《EntityBean》是引入的新版型，表示 EJB 中的 Entity bean，同理还可使用版型《SessionBean》表示 Session bean。

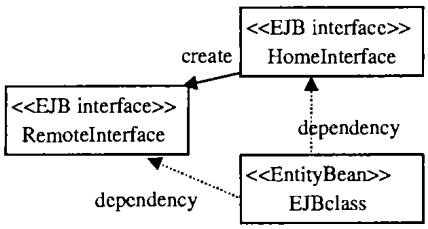

图2 EJB 组件内部构成图

(2)EJB 类有 Session bean 和 Entity bean 两种类型，每种 EJB 类有自己显著的特征，当我们在使用从用例视图中所获得的角色来创建逻辑视图(结构视图)时，可将类分为：控制类、实体类和边界类，将这些类映射到 J2EE 组件应该遵循这样的原则：

a)通常实体类对象需要在存储在数据库中，故可以使用 Entity bean 来实现。

b)控制类实现实体之间信息交流、处理，完成一些涉及到多个实体间的功能序列，故可以使用 Session bean 来实现。

c)边界类是实体之间交换的信息。在 J2EE 中，通常表现为 Web 页面、对话框或者特定的信息类。一般需要在数据库中存储的信息类可作为 Entity bean 实现，并作为 EJB 组件的一员，但通常大部分边界类由页面构成，故不作为 EJB 组件的一部分，而是作为 Web 组件的核心来建模。

5 Web 组件的建模

一个完整的 J2EE 应用是由 J2EE 客户(Applet，Application)、EJB 组件和 Web 组件构成。Web 组件被打包成.war (Web archive)文件，与 EJB 组件的 EJB.jar 文件一起部署，向客户提供强大的功能支持。在 J2EE 服务器内部，实际上 Web 组件通常作为 EJB 组件的客户。Web 组件中的 JSP 或者 Servlet 使用多个 EJB 对象的业务方法，在 EJB 对象之间传递信息，Web 组件的元素充当了边界类角色，故我们在为 Web 组件建模时，常常将边界类映射成为 Web 组件的 JSP 页面。

Web 开发中，常用的是 JSP，它在一个文件中集成了页面的表示逻辑和业务逻辑，所以比 Servlet 方便。因此以下仅讨论使用 JSP 来建模 Web 组件，在模型的结构视图中，一个较好的构建规则如下：

将系统中页面分成两类：服务页面和显示页面，这里的服务页面和显示页面是按照功能划分的，若一个 JSP 可以产生另一个输出页面，则该 JSP 页面认为是服务页面，而被产生的终端页面就是显示页面，它们之间的关系是 produce。静态的 html 页面被认为是显示页面，页面间的超链结用 link 表示。一个 Web 组件的内部构成图如图3，版型《ServerPage》和《DisplayPage》分别代表服务页面和显示页面。在结构图中，JSP 中可以引入 javaBean 来丰富功能，javaBean 实质上就是一个普通的类，它可以使用 EJB 对象的方法，向引入自己的 JSP 页面提供业务功能，使 JSP 页面的表示逻辑和业务逻辑更好地分开。

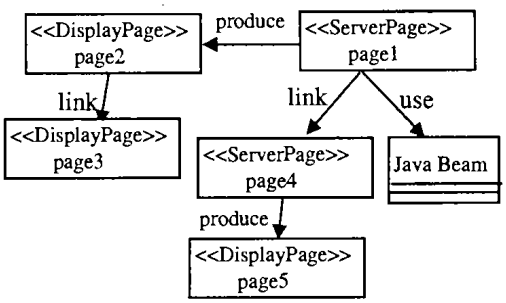

图3

一个系统理论上可以拥有多个 Web 组件和多个 EJB 组件，通常，程序员习惯的开发方式是：采用多个 EJB 组件和尽量少的 Web 组件(一般只一个)组成分布式系统，笔者建议：由于 EJB 属于进程间组件，运行的系统开销大，当业务逻辑需要操作数据库或需要保持某些状态时，可以考虑使用 Entity Bean 和有状态的 Session Bean 来实现，除此之外，要尽量避免使用无状态的 Session Bean，那些无需对数据库进行存取和保持状态的业务逻辑可以使用 javaBean 来实现。JavaBean 被引入到 Web 组件中的 JSP 中，在同一进程内执行，效率比使用 EJB 组件要高许多。

6 J2EE 系统的组件模型

在对分布式环境中的 EJB 组件和 Web 组件分别建模的基础上，可以由它们组成系统的组件图。在系统组件图中引入了版型《EJB Component》和《Web Component》，前者代表 EJB 组件，而后者表示 Web 组件，如果需要区分 EJB 组件的类型，还可以引入其它新的版型分别表示 Entity Bean 组件和 Session Bean 组件。但并不推荐这样做，若使用过多的扩展版型，反而会使系统组件图表达复杂，湮没 UML 标准规范本身的优势。如图4，一个简洁、完整的组件图不但能表达组件之间的关系，而且还能清晰地展示 J2EE 平台上组件的类型和调用层次。

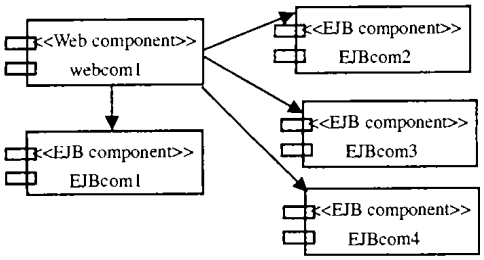

图4 系统组件图

结论 本文利用 UML 统一建模语言的语义及扩展机制，深入探讨了建模技术在基于组件的分布式平台上的实现，针对传统的面向对象的系统建模方法的缺陷，提出了相应的 Web 组件和 EJB 组件的解决方案。并总结出组件建模时应当注意的原则：

(1)针对分布式环境的特点，利用其特定的技术规范，对不同类型的组件分别建模，如 Web 组件与 EJB 组件，在系统组件图中，描述出之间的调用关系。

(2)EJB 组件属于进程间组件，执行开销较进程内组件大，在构建组件模型时，应当尽量衡量利弊，在不影响主要结构的前提下，应避重就轻，尽量以 Java Bean 的方式来替代某些 EJB 组件的功能。

(3)建模组件内结构时，引入了 UML 的扩展机制，在使用组件组装系统时，仍然应该极大限度地利用 UML 传统的优秀建模理论。

多层分布式系统的建模技术还在发展之中，本文仅对组件部分提出了自己的建模思路，但将 UML 的建模思想完全成熟地运用于分布式系统，还有待进一步研究。

参 考 文 献

1 Tremblett P 著，潇湘工作室 译. 即时应用 Enterprise JavaBeans. 北京：人民邮电出版社，2001
2 周秉锋 著. UML 软件建模. 北京：北京大学出版社，2001
3 吕智林，苏德富，郑宇. 基于 UML 的 EJB 组件建模方法的研究. 计算机应用研究，2002(2)
4 Kobryn C. Modeling Components and Frameworks with UML. Communications of ACM，2000，43(10)

(上接第128页)

算每个 $eu(p)$ 时所需的计算代价。在标书中增加 $pem(p)$ 信息造成了少量额外的通讯负载。然而，鉴于整个 MAS 系统环境中所发生的巨大通讯负载量，我们相信，此改进所增加的微量负载几乎是可以忽略不计的。

该方案最可能的弱点在于：当多个发起者 $I(p)$ 评价 p 的方案时，其中每个 i 都具有不同的 $DoA(p)$ 值。最先的发起者具有最高的 $DoA(p)=1$，后面依次是 $1/2,\cdots,1/|I(p)|$。而当 p 发送标书到最后一个发起者时，前面发起者的 $DoA(p)$ 值也应同时降到 $1/|I(p)|$。另外，如考虑时间因素，虽然早来的发起者确实应该得到更高的 DoA(p)值，但若 p 在很短的时间间隔内发出多个标书，以如上方式降低各 $DoA(p)$ 值就显得不够明智了。下面的特例有助于对该问题的理解。设 p 以非常小的时间间隔发出两份标书依次给 i_1，i_2，那么 i_1 的 $DoA(p)$ 值为1，而 i_2 的值却一下子降到了1/2，这显然存在一定的不合理性，因为 i_2 的值至少应该再高一些，如0.8、0.9或者更高。要解决该问题，我们可能需要引入时间参数，但要把时间对 $DoA(p)$ 的影响进行量化处理，是非常有挑战性的研究课题。

另外，本文中除可用度以外，我们假设标书评价标准仅参考所需付出的代价，但在实际应用中，评价指标可能会完全不同。它可能是时间消耗、经济成本等一些数量越低越好的指标，也可以是服务质量、收益等越高越好的指标。在更复杂的条件下，还可能需要把几项指标综合起来考虑。因此在不同领域，需要良好的机制把它们转换成单一的效用数值，这是与领域密切相关的问题。为了便于阐明思想，本文简单地令 $u(p)=1/cst(p)$，事实上，也可采用其它转换函数，如 $u(p)=1/\sqrt{cst(p)}$ 或 $\lambda/cst(p)$，$\lambda\in R$ 等。对于某特定领域，效用转换函数须能较确切地反映各方案的优劣程度，为便于理解这个要求，请看如下反例：我们知道，考试分数100优于70的程度远高于70优于69的程度。设有函数 u：$u(100)=5$，$u(70)=3.5$，$u(69)=1$，可以看到，u 对于反映上述分数优劣程度的差别无能为力。在实际应用中，我们的改进方案在很大程度上还依赖于一个恰当的效用转换函数。

结论与展望 本文根据最大期望效用原理，在 CNCP 引入可用度概念，使发起者在评价参与者优劣时，以更加理性的方式做出决策。这在很大程度上降低了发起者的任务得不到执行的风险，结合在参与者一方增设 cfp 阈值(另文详述)的概念，更可减少大量不必要的计算消耗。

本文进一步研究包括如何确定合适的效用转换函数及考虑带时间因素的 agent 可用度问题。另外，如下两方面的工作也有待进一步开展：

1)在文[5]中，CNCP 的设计者通过修正 CNCP 的一个缺陷而使之成为 HCNCP。这个缺陷发生在应用 CNCP 到 HMAS 中(Holonic Multi-Agent System，HMAS)时，参阅文[5,8]。我们相信，经过适当的调整，可用度概念同样可帮助提高 HMAS 的性能。

2)另外，针对 FIPA 新公布的 CNP 扩展协议 FIPA Iterated Contract Net Protocol(ICNP)[9]允许多轮重复竞价的情况，需要对可用度机制做进一步探索，以期找到更好的解决方案。

参 考 文 献

1 Ktting B，Maurer F. A Concept for Supporting the Formation of Virtual Corporations through Negotiation. In：IEEE Post-Proc. of the 8th Intl. Workshops on Enabling Technologies：Infrastructures for Collaborative Enterprises，Stanford (USA)，1999
2 Smith R G. The contract net protocol：High level communication and control in a distributed problem solver. IEEE Transactions on Computers，1980，C-29(12)：1104～1113
3 Sandholm T，Lesser V R. Advantages of a leveled commitment contracting protocol. In AAAI，1996. 126～133
4 FIPA. FIPA Contract Net Interaction Protocol Specification. FIPA TC Comunication. 2002/12/03. http://www.fipa.org/specs/pesspecs.tar.gz
5 Knabe T，Schillo M，Fisher K. Improvements to the FIPA Contract Net Protocol for Performance Increase and Cascading Applications. http://citeseer.nj.nec.com/542858.html
6 Schillo M，Fischer K，Kray C. The Contract-Net with Confirmation Protocol：A solution to a fundamental problem of DAI. DFKI Technical Memo. 15，2000
7 Russell S，Norvig P. Artificial Intelligence：A Modern Approach，Prentice Hall Series in Artificial Intelligence. Englewood Cliffs，New Jersey，1995
8 Schillo M，Fley B，Florian M，et al. Self-Organization in Multiagent Systems：From Agent Interaction to Agent Organization. citeseer.nj.nec.com/schillo02selforganization.ht-ml，2002
9 FIPA. FIPA Iterated Contract Net Interaction Protocol Specification. FIPA TC Communication. 2002/12/03. http://www.fipa.org/specs/pesspecs.tar.gz