第30卷 第8期 计 算 机 工 程 2004年4月
Vol.30 №8 Computer Engineering April 2004
·基金项目论文· 文章编号：1000—3428(2004)08—0048—03 文献标识码：A 中图分类号：TP311.5

遗产系统的构件化技术

药 锐，赵文耘，张 志
(复旦大学计算机与信息技术系软件工程实验室，上海 200433)

摘 要：遗产系统一直是企业级解决方案需要考虑的重要内容，而接口的处理是遗产系统融入企业级解决方案的关键所在。UML是一种面向对象的建模语言，但是它对接口、构件规格说明及构件建模概念支持甚少。该文通过对UML进行扩展，并且利用企业应用集成(EAI)中提供的应用编程接口(API)，提出一种将遗产系统构件化的方法。

关键词：遗产系统；统一建模语言；企业应用集成；应用程序编程接口；类型模型；映射；构件

Componentized Technology of Legacy System

YAO Rui，ZHAO Wenyun，ZHANG Zhi
(Lab of Software Engineering, Dept.of Computer & Information Technolgy,Fudan University,Shanghai 200433)

【Abstract】 Legacy system is a major issue of the enterprise solution so far. Interface is the bridge between the solution and the legacy system. UML is an object-oriented modeling language, but it provides little support to interface, component specification and component modeling. This paper presents a method to componentize the legacy system by extending the UML and applying the API of EAI.

【Key words】 Legacy system；UML；EAI；API；Type model；Mapping；Component

目前，对遗产系统的处理一般都是尽可能把它集成在现有的软件平台上。如果在现有软件平台上为每一套遗产系统都开发相应的接口，工作量很大，而且一旦用户增加新系统，又要开发新的接口。企业应用集成(EAI) 通过建立底层结构，完成在企业内部的ERP、CRM、SCM、数据库、数据仓库以及其他重要的内部系统之间共享和交换数据的需要。有了EAI，企业就可以将遗产系统和新的Internet解决方案结合在一起。

在基于Internet的企业级解决方案中，构件化是首当其冲要考虑的。然而遗产系统目前只能做到无缝地集成到现有平台中，还没有实现构件化。本文通过研究企业应用集成(EAI)的各种规范和基于构件的软件开发方法，提出了一种利用现有的面向对象建模语言——UML，并对之进行扩展来支持遗产系统构件化的方法。

1 相关理论简介

1.1 UML简介

对象管理组织(OMG)对UML的定义是：统一建模语言(UML)是一个通用的可视化建模语言，用于对软件进行描述、可视化处理、构造和建立软件系统制品的文档。它记录了对必须构造的系统的决定和理解，可用于对系统的理解、设计、浏览、配置、维护和信息控制。UML描述了一个系统的静态结构和动态行为[1]。UML的概念和模型可以分为5个概念域：静态结构，动态行为，实现构造，模型组织及扩展机制。

1.2 EAI简介

EAI是一种在企业中集成应用程序和数据以便达到自动业务处理的规则。EAI[3]利用通用的中间件(middleware)融合了企业已有应用软件、商业封装式软件和新代码3方面的功能。基于中间件的EAI从3个方面降低了应用软件集成的复杂程度：(1)封装应用程序的机制；(2)应用程序共享信息的机制；(3)应用程序协调企业流程的机制。

EAI几乎不需要改变现有的遗产系统，而且很少需要扩展程序或定制接口。EAI通常使用现有应用程序编程接口(API)和数据库。

2 遗产系统的构件化

分3个步骤来进行遗产系统的构件化：(1)寻找构件；(2)定义构件的接口；(3)构件内部的实现。

2.1 寻找构件

遗产系统的不灵活性主要体现在功能模块的实现和模块间的交互交织在一起，这不符合软件工程一直倡导的思想：高内聚低耦合。本文主要从此观点入手，将遗产系统功能的实现和模块间的交互进行分离。

在遗产系统中寻找构件的主要过程如下：

(1)是否所有的遗产系统的资源都可以获得；
(2)对失去的资源，确定是否通过工具从可执行文件中获得；
(3)确定源代码版本和可执行版本之间的关系；
(4)确定遗产系统中使用的编程语言；
(5)获取此编程语言的分析工具；
(6)运用工具对遗产系统进行分析，解释结果；
(7)根据结果判断构件的划分是否可行、边界是否合理。

分析的目的是从遗产系统中抽取能展示系统体系结构的信息，而且抽取的信息应该能反映特定领域的商业逻辑。如果无法反映特定领域商业逻辑，那么构件化就无从谈起。

目前遗产系统的分析技术主要包括：

(1)为了得到系统的功能，询问用户、维护人员和系统的设计人员。具体来说，就是要区分出什么是应当保留的，什么是多余的。

(2)使用持久且稳定的数据存储。在所有遗产系统中扮演角色的数据中，很可能存储在数据库中的数据反映特定领域的商业逻辑。

(3)分析程序的调用关系。调用关系可能反映出模块的耦合和内聚以及遗产系统的层次关系。例如，如果一个模块调用许多其他的模块，但并不调用本身，那么该模块很可能是控制模块，没有什么功能；如果一个模块被很多模块所调用，它可能是处理错误或者异常的模块；除了以上两种情况的模块可能会包括商业逻辑。

基金项目：国家“863”计划基金资助项目(2001AA113070)
作者简介：药 锐(1977－)，男，硕士生，研究方向：软件工程；赵文耘，教授；张 志，硕士生
收稿日期：2003-08-05 E-mail：yr_yaorui@msn.com

(4)使用系统提供的界面。通过确定界面的时序和引发的动作，可以得到一个用例。

(5)进行概念和簇分析来得到一致的构件。例如，根据数据元素在过程或程序中的使用把它们进行组合，然后分组成候选类。

本文使用自己开发的代码分析器。通过逆向工程的方法，得到遗产系统的高层模型。然后通过对高层模型的分析，加上对特定领域的理解，找出该遗产系统的构件。

2.2 定义构件的接口

在找到遗产系统的构件后，就可以来着手完成接口定义了。在对接口进行定义前，要先对UML进行扩展。作者通过对UML各种符号的仔细研究，发现目前的UML对构件建模有一些限制，特别是对于非面向对象的构件实现(大部分遗产系统都是使用非面向对象技术)。这些限制包括：(1)为了对接口、构件规格说明和其它关键构件概念建模，用户需要使用UML中的像构造型、命名规范等扩展机制。结果是，这些在UML工具中没有被作为第1级概念来支持，可能会被不同的组织或个人作出不同的解释。(2)UML很难描述一个动态行为来定义多个构件怎样合作完成一个要求的特定行为，必须结合用例图和顺序图(或UML协作图)来描述类之间对象级的交互。(3)UML不支持在所要求的精确程度上定义接口行为。

本文参考国内外关于UML研究的各种分析，提出了通过UML扩展支持遗产系统构件化的方法。这种方法的优点是提供对遗产系统构件化的支持，并提供一个特殊的类型协作的概念来描述接口之间的交互；允许通过前置和后置条件来对接口操作的行为进行严格定义。

2.2.1 扩展一：接口建模

软件工程领域中的对象通过它们的行为来描述，而这些行为的集合描述了一个对象在需要时可以扮演的角色。单个对象可能扮演多种角色，就像在真实世界里，一个对象能扮演多种角色一样。例如，某个人在他和航空公司的交互中，被看作乘客。然而在和宾馆的交互中，却被当作房客。因为他们在不同的交互中执行了不同的行为。

这种行为可以用类型来表示。一个类型定义了对象在特定角色中的外部可见行为。不像类的概念，一个类型并不描述其实现，而是定义了任何实现必须提供的行为。该行为根据任何正确的实现状态以及如何允许状态进行改变来描述。状态可以是简单的属性列表，而通常更多的是一组相关类型来构成该类型的类型模型。

然而，要实现遗产系统的构件化，必须提供精确的接口，只靠上面提供的类型信息去描述接口是不够的，必须对执行行为的操作给出规范说明。图1给出对编辑器接口的详细描述。在图1中，用<<type>>描述任何内部实现的外部抽象。特别地，类型框的中间部分，不像一般认为的是存储数据，而是被解释为抽象属性。类型用一系列操作来描述，而且操作使用类型模型来指定。类型模型定义了实现该类型必须理解的规范术语。

类型框中包括静态部分和动态部分。其中静态部分包括：操作，隐含的规范术语；而动态部分包括：将术语作为属性进行建模，利用模型表示形式化操作规范，通过映射将模型和实现联系起来。

一般来说，对构件的描述必须涉及两个方面：作为类型的构件接口规范和接口实现(类、模块、可执行文件等)。

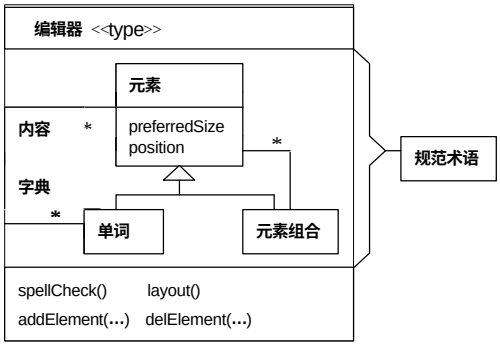

图1 编辑器的类型模型

上面简单说明了类型模型，下面以ListBox为例着重谈一下接口实现。

```
interface awt.ListBox{
  model{
      int count();
      Item itemAt(int pos);
      Boolean selected(Item);
      moveUp( i , j);    }
  void addItem(Item, int);
  Item delItem (int);
  …}
```

上面这段代码中，如果没有黑体部分，那么接口实现中并没有说明表项增加和删除的方向，也没有说明是以0还是1开始，更不知道如果超出容量了怎么办？这说明只有代码无法说明所需的行为。黑体部分正是为了更清楚地说明行为而增加的规范术语。需要强调的是：类型的描述者和类型的实现者必须对类型模型中的规范术语(即上文的黑体部分)有一致的理解。

至此，行为被彻底描述清楚了。下面着重考虑如何由该类型模型产生一个具体实现。本文的方法是在接口和实现之间建立一个映射。只要该实现符合映射关系，就表示它是一个有效的实现，而不管该实现的任何细节。图2展示了该映射关系。该图表明只要在类ListBoxA中实现映射中大括号中的功能，类型ListBox就可以实现了，但是在映射中并不规定使用什么代码去实现该类型等细节。

到此，接口建模描述完毕，可以利用UML扩展一对接口进行定义了。

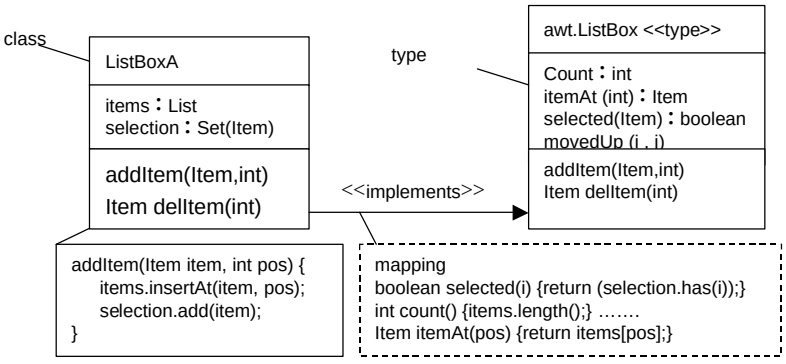

图2 ListBox类型模型和ListBox实现之间的映射关系

2.2.2 接口定义

首先，根据构件扮演的角色定义构件支持的操作，创建一个初始接口类型模型(利用上述描述手段)。值得注意的是：在创建类型模型时，要优先使用遗产系统本身已经实现的操作，也就是说复用遗产系统的功能。如果没有，再自己实现。

在对角色和每一个接口的职责有了深入的理解之后，就可以来完成接口的细节了。主要执行以下3个主要活动：

(1)考虑每一个操作的标记和行为。对于每一个操作，要使用适当的类型定义其参数。操作的行为采用前置条件和后置条件对来描述。每一对前置和后置条件根据初始状态的描述(前置条件)及执行操作后所得到的结果状态(后置条件)定义操作的行为。

(2)接口间的关联，可通过使用不变量来定义。通过把接口的公共状态定义提取为一个整体，这些不变量帮助简化操作行为的描述。

(3)接口类型模型完成后，它和操作参数、不变量以及所定义的前置、后置条件相一致。

下面是使用前置和后置条件对addItem的精确描述：

```
void addItem(Item item, int pos)
 Pre 0<pos and pos<=count+1  // pos必须在一定的范围内
 Post item=itemAt(pos) and selected(item)
       and count=count@pre+1  // count数加一
       and moveUp(pos, count@pre)  // pos后面的项向上移动
```

如上面的代码所示，addItem()操作的参数是item和pos，返回值为空。通过对前置和后置条件的描述，接口被精确地描述出来。

然而，负责实现提供这种行为的系统的人，以及那些负责使用或消费该行为的人将产生大量的反馈信息。通过这些反馈，可以细化这些接口定义以满足从以上这些负责实现的小组收到的要求。这样就完成了接口的精确定义。

构件的内部实现主要包括两个部分：内部更小粒度的构件的协作以及类型模型和遗产系统的实现之间的映射。其中构件内部更小粒度构件的协作需要对UML进行扩展。

2.2.3 扩展二：类型内部构件间的协作

还是以编辑器为例，看一下如何描述其内部的小粒度构件的交互。

首先，必须把编辑器分割为可独立实现的小粒度构件，使它们可以互相协作。本文将编辑器分为拼写检查、布局算法和编辑器核心3个构件。

然后，要对编辑器类型模型中的规范术语根据分割好的构件进行划分。在本例中，只有拼写检查器需要用到字典，可以将字典归于拼写检查器；然而拼写检查器和编辑器核心都需要理解单词并且交换它们；而且在布局管理器中也会用到单词术语，只不过布局管理器只是把它当作有一定顺序的字符集元素而已，必要时可以将它分割布局，所以可以把单词术语归于拼写检查器和编辑器核心构件。

最后，要对分割后的构件接口进行描述。本例中，编辑器核心构件需要提供两个接口，然而其它两个构件间不需要接口。

按照以上步骤生成的协作图如图3所示。在该协作图中，椭圆代表构件提供的接口，箭头上方为编辑器核心构件接口提供的行为。

2.2.4 类型模型和遗产系统之间的映射

首先要说明的是：虽然在寻找构件时是以遗产系统的源代码和其它一些资料来着手提炼模型，寻找其在特定领域的

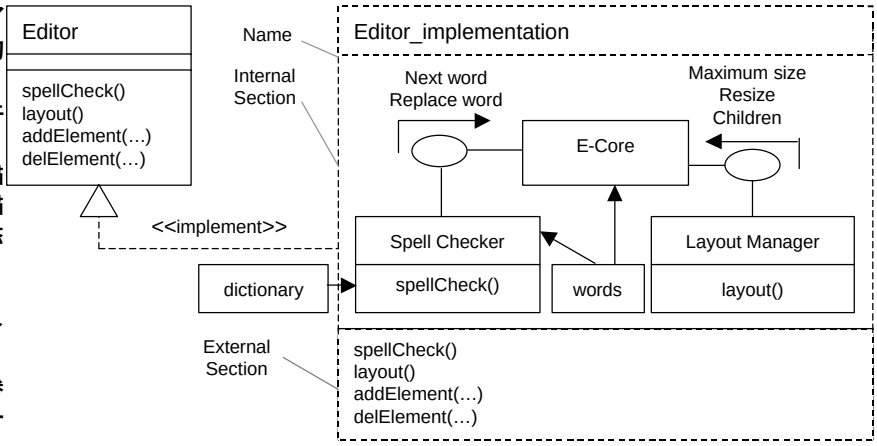

图3 编辑器构件协作图

一些商业逻辑的，但是在本阶段本文使用经过企业应用集成包装后的系统作为映射的基础。这样做有以下几点好处：

(1)屏蔽了遗产系统使用的编程语言的限制。

(2)EAI包装过的接口使用了诸如Java的RMI、CORBA的IIOP以及Microsoft的DCOM等标准机制，使得遗产系统的应用接口不再是私有的。

(3)开发工具可以对使用标准接口的应用进行支持。

(4)EAI包装后的接口描述是使用XML的。XML是一种标准的中间语言，它可以用来以人可读的形式向现有系统或从现有系统传递信息。

(5)EAI包装后的系统体现了一定的商业逻辑，其提供的API和服务可以作为特定领域构件化的基础。

然后，就可以进行映射了，其具体步骤主要包括：

(1) 针对某个遗产系统，将EAI提供的API和服务分为3大类型：商业服务，数据服务，对象。其中，商业服务提供在遗产系统中存在的商业逻辑；数据服务直接提供对数据库的访问；对象是由商业服务和数据服务绑定而来的。

(2) 将类型模型中的属性和接口操作找出来，准备实现属性和协作与EAI提供的接口之间的映射。

(3) 对类型模型中的每个和数据相关的属性，在EAI的数据服务中寻找相关的实现，然后进行匹配；如果需要多个数据服务，则要考虑顺序。

(4) 对类型模型中的每个和商业逻辑相关的操作，根据接口的精确定义pre和post，对接口参数按照步骤(3)处理，对操作本身，寻找相关的商业服务进行匹配；如果需要多个商业服务，则要考虑并发和并行。

(5)将步骤(3)和(4)的结果进行合并，得到此类型模型和包装系统的映射。

(6)将映射关系与类型模型和EAI包装过的系统进行比较，以确认其有效性。

待每个类型模型对应的映射关系确定后，就在抽象层和具体实现层之间建立了可跟踪关系。由于EAI的接口信息是用XML来描述的，因此可以保证此映射关系是与具体实现无关的。开发人员只要根据类型模型和该映射关系，就可以重现本系统。同时，构件化的过程也进入包装阶段。

3 结束语

遗产系统本身有一些有价值的应用和数据信息可供在动态集成时重用，而基于构件的体系结构可能被设计得更加可靠，而且可以对部署环境的变化和用户需求的变化作出快速响应。通过对遗产系统和当前构件技术的实际分析，利用UML扩展来描述遗产系统的构件化，并结合EAI提供的服务

(下转第135页)

定为内部用户，否定则为外部用户。 IP地址和主机名由Request对象的ServerVaribles环境变量获得（关于防止用户采取IP地址欺骗的问题，作者将另文详述）。其次该用户与管理员会话以请求授权，对局域网用户管理员提供所有角色和权限供用户选择，对广域网用户管理员提供广域网角色和所拥有的权限供用户选择，得到授权后将注册用户所拥有的角色及权限等信息写入用户信息数据库中，同时在Oracle用户管理中同步建立该用户的信息，然后返回主页。

2.3 菜单动态生成及访问控制

首先根据合法用户登录的该用户有关信息在RBAC数据库中查询出处于系统当前层的所有该用户有权执行的属于超级链接的应用程序文件的URL地址和其中文描述，以及其在系统中所处的应用层次，其次根据查询结果在页面中动态生成超级链接菜单。

另外为防止非法用户直接以HTTP方式在浏览器请求执行某页面，在每个应用系统程序文件中还需包含具有以下功能的代码：判断该用户是否登录并建立会话，如已建立会话，就检查该用户是否有该页面的执行权限。方法是先取得欲请求执行页面的路径，然后结合该用户保存在Application变量中的用户名、口令、用户属性等用户信息，在RBAC数据库中查询其是否可访问该页面，有则显示，否则释放该变量并终止程序。

2.4 RBAC授权管理

在RBAC管理系统模块中，所具有的功能如图4所示。用户管理包括用户和密码管理，增、删、改用户名和密码；角色管理包括角色的增、删、改；访问许可权限管理包括系统资源的增、删、改，菜单模块的增、删、改；而针对用户的角色授予、管理和针对角色的权限授予、管理则在用户/角色关系维护模块和角色权限关系维护模块中。

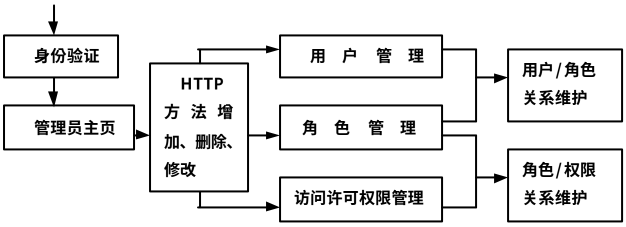

图4 RBAC管理系统

由图4可知，当用RBAC管理系统删除某用户和角色时，为保证数据完整性，用户－角色关系表中该用户的相关数据和角色－功能资源关系表中该角色的相关数据也应该同时被删除。可以在程序中实现，也可以在数据库中建触发器实现。

为使RBAC管理系统和Oracle用户管理系统一致并相互无缝集成，一方面，当在RBAC管理系统中增加、修改、删除某一用户或某一角色时，必须执行相应的SQL语句，以在Oracle用户管理中同样增加、修改、删除该用户或该角色的权限。类似地，在用户/角色关系维护和角色/权限关系维护中，为用户授予角色或为角色授予权限时，也必须执行相应的SQL语句，以在Oracle用户管理中同步为该用户授予角色或为该角色授予数据操作权限。另一方面，通过查询数据字典视图中Oracle用户管理包含的用户信息，并在RBAC管理模块中用程序适时、动态地显示这些用户信息，可以将Oracle系统管理员在Oracle用户管理中增加、修改、删除某用户的操作结果动态显示并集成到RBAC管理模块中。这样不仅真正做到了RBAC管理系统和Oracle用户管理系统的一致，而且在应用系统中只须为RBAC系统管理员提供RBAC管理系统的操作界面，无须进入Oracle用户管理操作，就能够完全实现用户、角色、权限的管理以及授权。

3 结束语

该RBAC方案能有效地实现对用户的访问控制，访问控制的内容由应用系统所服务的企业或部门具体确定，访问控制的对象包括系统资源和数据库实体对象。同时还能根据用户的RBAC权限自动生成用户界面。由于采取Oracle数据库的用户身份认证，因此非法用户试图破解口令的可能性大大降低。将RBAC管理系统和Oracle用户管理系统集成到一起，实现了应用安全和数据安全的统一。该方案在某国防开发系统应用中得到广泛采纳并获得了较好的效果，深受用户好评。

参考文献

1 Ferraiolo D F, Barkley J F, Kubn D R. A Role Based Access Control Model and Reference Implementation Within a Corporate Intranet[C]. ACM Transactions on Information Systems Security, 1999, (2)

2 Ye Xijun, Xu Yong, Wu Guoxin. Implementation Technique Based on RBAC Used in Web[J]. Computer Engineering, 2002, 28 (1): 167-169

3 Luo Xueping, Zheng Yili, Xu Guoding. An Exetended Role-based Access Control Model[J]. Computer Engineering, 2001,27 (6): 106-110

4 Yi Xiaochao, Zhang Shaolian, Mao Bin, et al. The Research and Development of Access Control Technology[J]. Computer Science, 2001, (7): 26-28

5 Fu Lianxu, Luo Fei, Wen Shaochun, et al. The Database Design of User Security Management Based on the Integration Technology Between Windows NT and MIS System[J]. Computer Engineering and Applications, 2002,38 (9): 214-217

☆☆☆

(上接第50页)

接口，提出一种将遗产系统构件化的方法。本文的创新点就在于利用通过EAI包装后的遗产系统提供的标准接口将遗产系统构件化。当然，也有不足之处。今后将在此基础上，着重研究如何选取某个工程领域中有代表性的遗产系统，以及采用更加形式化的手段对接口进行描述。

参考文献

1 Rumbaugh J,Jacobson I,Booch G.UML参考手册.北京:机械工业出版社,2001

2 Pressman R S.软件工程实践者的研究方法.北京:机械工业出版社, 1999

3 Ruh W A, Maginnis F X.企业应用集成.北京:机械工业出版社, 2003

4 Brown A B.大规模基于构件的软件开发.北京:机械工业出版社, 2003

5 Cummins F A.Enterprise Application:An Architecture for Enterprise Application and Systems Integration. John Wiley & Sons, Inc.,2001