从 xUML 模型到 J2EE 应用系统的转换方法研究

朱忠旭,袁兆山,李宏芒

(合肥工业大学 计算机与信息学院,安徽 合肥 230009)

摘 要:Executable UML(xUML)是统一建模语言(UML)的一个剖面(Profile),它可以看作是由传统 UML 加上精确的语义构成。在阐述 xUML 和 J2EE 平台基本概念的基础上,提出了一种从 xUML 业务模型生成基于 J2EE 平台的 Java 代码的方法,对从平台无关模型到应用代码的转换进行了尝试。

关键词:可执行统一建模语言;模型驱动架构;J2EE

中图分类号:TP311 **文献标识码**:A **文章编号**:1673-629X(2007)07-0013-04

Research on Method of Transforming xUML Model to J2EE Applying System

ZHU Zhong-xu, YUAN Zhao-shan, LI Hong-mang

(School of Computer and Information, Hefei University of Technology, Hefei 230009, China)

Abstract: Exectuable UML(xUML) is a profile of UML and it can be seen as classic UML plus accurate action sementics. In this paper discussed the concept of xUML and J2EE platform, then brought forward and attempted a method of code generator from xUML model to Java code running on J2EE platform.

Key words: executable UML; MDA; J2EE

0 引 言

2001 年,对象管理组织(Object Management Group,OMG)提出了模型驱动架构(MDA)。MDA 的核心思想是通过建立与具体的实现技术无关的、完整描述业务功能的平台无关模型(PIM),制定到不同实现平台的映射规则,然后通过这些映射规则及辅助工具将 PIM 转换成与具体实现技术相关的应用模型 PSM,最后将 PSM 转换成可执行的代码。

可执行统一建模语言(Executable UML, xUML)的核心是精确的动作语义[1],通过动作语义可以对类的方法(包括状态无关方法和状态相关方法)以及类之间的关系(类的创建、调用、查询等)进行精确的描述,然后,借助于模型和代码之间的映射规则,实现从 xUML 模型到代码之间的转换。

1 Executable UML

Executable UML(xUML)是 UML 的一个可执行的子集,它的含义可以用图 1 来表示[2]。

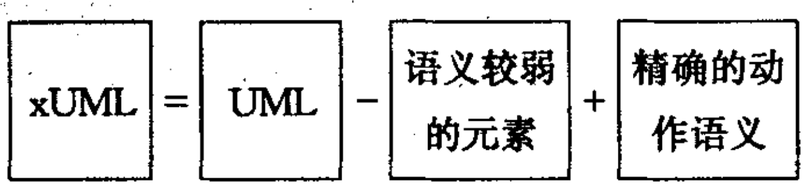

图 1 xUML 的定义

xUML 去掉了 UML 中语义较弱的元素,如组件图和部署图等,而保留了语义较强的元素,包括包图、类图、顺序图、协作图、状态图。

域是由一组独有的类的集合所组成的一个独立的模块单位[3],在 xUML 模型中,用包图表示域。

类是对一组具有相同属性、操作、关系和语义的对象的描述[4],每个类可以通过最多一个状态机,零个或多个操作,以及零个或多个属性进行正式定义。类之间可能存在关联,关联是一种结构关系,它指明一个事物的对象与另一个事物的对象之间的联系[4]。关联用一条连接相同类或不同类的实线表示,除此之外,还有四种用于关联的修饰:名称、角色、多重性和聚合。

顺序图和协作图用于对类之间的交互关系建模。顺序图中强调的是交互的时间顺序,合作图忽略了时间顺序而把重点放在交互的模式和合作的紧密程度上。

状态图用于描述对象的与状态相关的行为,在

收稿日期:2006-10-21

基金项目:合肥工业大学科学研究发展基金项目(050502F)

作者简介:朱忠旭(1974-),男,安徽凤台人,硕士研究生,研究方向为软件工程;袁兆山,教授,研究方向为软件工程、计算机网络。

xUML 中,每个类最多有一个状态图。

动作语义旨在向建模者提供一种在 UML 模型内精确定义行为的方式,它可以描述方法的主题、状态或活动图中状态间转移的输出或与系统状态相关联的活动。动作语义与 UML 模型一起,能够用于建造在高于编程语言的抽象层次上指定问题的完整精确的模型,而且能够支持问题规范形式上的更正校验,使基于模型的高保真模拟和检验成为可能。

UML 标准中虽然加入了动作语义,但却没定义具体的动作语言,而是将动作语言的定义留给了软件厂家。现在已出现了多种为 xUML 设计的动作描述语言,如 MoDAL,BridgePoint,SMALL,ASL 等,动作描述语言使建模者可以为 xUML 的处理行为提供无歧义的、精确的定义。

2 J2EE 平台简介

J2EE 基于分布式多层应用模型,在这种模型中,应用功能在逻辑上分布在联网计算环境中各个级别或各个层上。每一层表示大多数企业应用所共有的相关概念集合。换句话说,J2EE 应用是由软件组件组成的,而这些软件组件根据它们在整个应用结构中所起的作用进行了分组。

J2EE 定义了四个不同的层,分别为客户层、Web 处理层、业务层和企业信息系统层。J2EE 组件在其对应的层上部署,且在对应层的容器中运行。容器提供标准的服务套件,同时使组件能够访问适合于对应层。

系统采用浏览器/服务器结构,分为 Web 表示层、Web 处理层、业务处理层、数据存储层(如图 2 所示)。

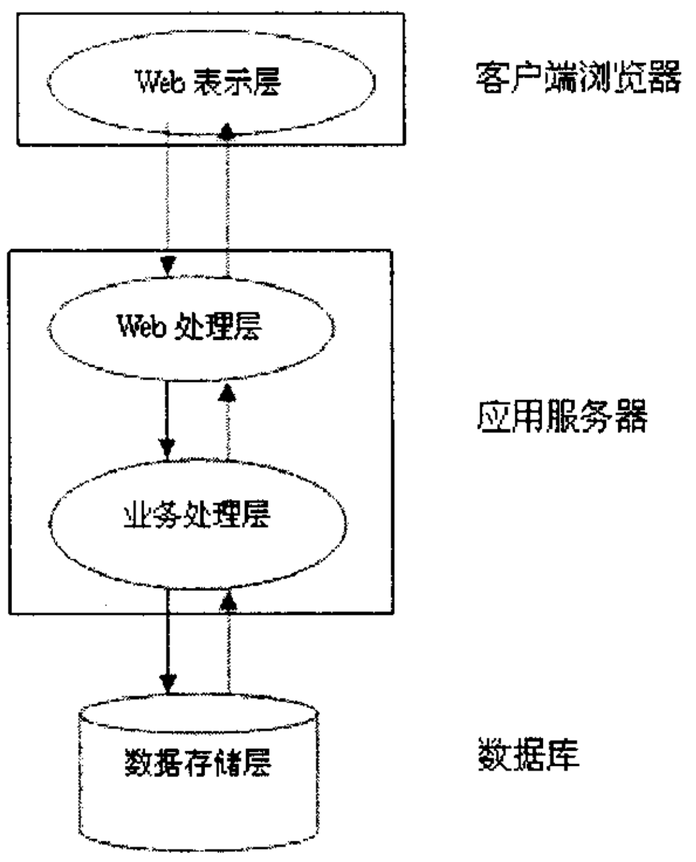

图 2 J2EE 应用系统的体系结构

Web 表示层处理来自 Web 处理层的数据,显示用户界面。用户在用户界面上进行操作,完成各种业务活动的交互部分。Web 表示层对用户交互数据进行简单的处理,并按照统一的格式进行封装,提交给 Web 处理层进行进一步处理。

Web 处理层对来自 Web 表示层的数据进行处理,并将数据交由业务处理层进行处理。对来自业务处理层的数据按照标准的格式发送到 Web 表示层。

业务处理层对 Web 处理层提交的数据进行处理,根据请求的类型将数据保存在数据存储层中或从数据存储层中获取所需的数据。

数据存储层对应于数据库系统,保存业务处理层提交的数据或从存取区中提取所需数据。

在 J2EE 中,服务器端组件主要有 JSP,Servlet 和 EJB,Servlet 用来动态处理客户端的请求以及作相应的应答。EJB 是封装了事务逻辑的服务器端组件,由一些类、接口、描述文件和一些资源文件组成,常用的 EJB 有 Session Bean 和 Entity Bean。Session Bean 主要与商业处理逻辑有关,它通常代表一个商业处理过程。Entity Bean 用成员变量存放来自数据库中的数据,数据库中每一列都对应于 Entity Bean 对象的一个成员变量,数据库中的一条记录则对应于一个 Entity Bean 对象实例。

3 从 xUML 模型到 J2EE 代码的转换

当前,支持使用 xUML 进行建模的工具,如 i-UML,BridgePoint 等,提供的模型编译器仅支持从 xUML 模型到 C 和 C++ 的代码转换[1],对 Java 和 J2EE 平台的支持还很欠缺。为此,笔者提出了从 xUML 模型到基于 J2EE 平台的应用系统的转换框架,为将基于 xUML 应用于 J2EE 系统的开发提供了一种尝试。

xUML 为开发人员建立精确的业务模型提供了手段,运用 xUML 建立的业务模型完全从系统功能考虑,反映了系统的业务逻辑,它不依赖于实现它的程序语言和操作系统等具体的实现技术和软件环境。因此运用 xUML 建立的系统模型符合平台无关的要求。在一个完整的 xUML 模型中,往往包含了多个域,此外,为了构建完整的 J2EE 应用系统,还要引入各种与实现相关的域,如 J2EE 中的类库等。要使这些模型构成一个可执行的系统,需要做如下的工作:首先,需要将域和域之间的依赖性通过某种方法实现,从而实现各个域的融合;其次,从平台独立的 xUML 业务模型转换成基于 J2EE 平台的 PSM;第三,需要在模型和 J2EE 类库间建立桥接器以调用类库提供的功能;第四,需要定义 J2EE 平台的 PSM 到代码之间的映射。

3.1 域间依赖的实现

在 xUML 中有四种类型的域:应用域、服务域、体系结构域和实现域[1]。对于 xUML 模型来说,各个域

必须联系起来构成单一的组装的元模型,才能一起产生代码。同时,还要使各个域之间保持最大限度的解耦合,以增加域的复用性、降低开发难度以及减少域之间的相互干扰。因此,当域间需要发生功能调用等依赖关系时,在一个域中不允许直接调用另一个域中的操作,而是采用"桥"的方式来建立域之间的联接。桥的实现结构如图 3 所示。

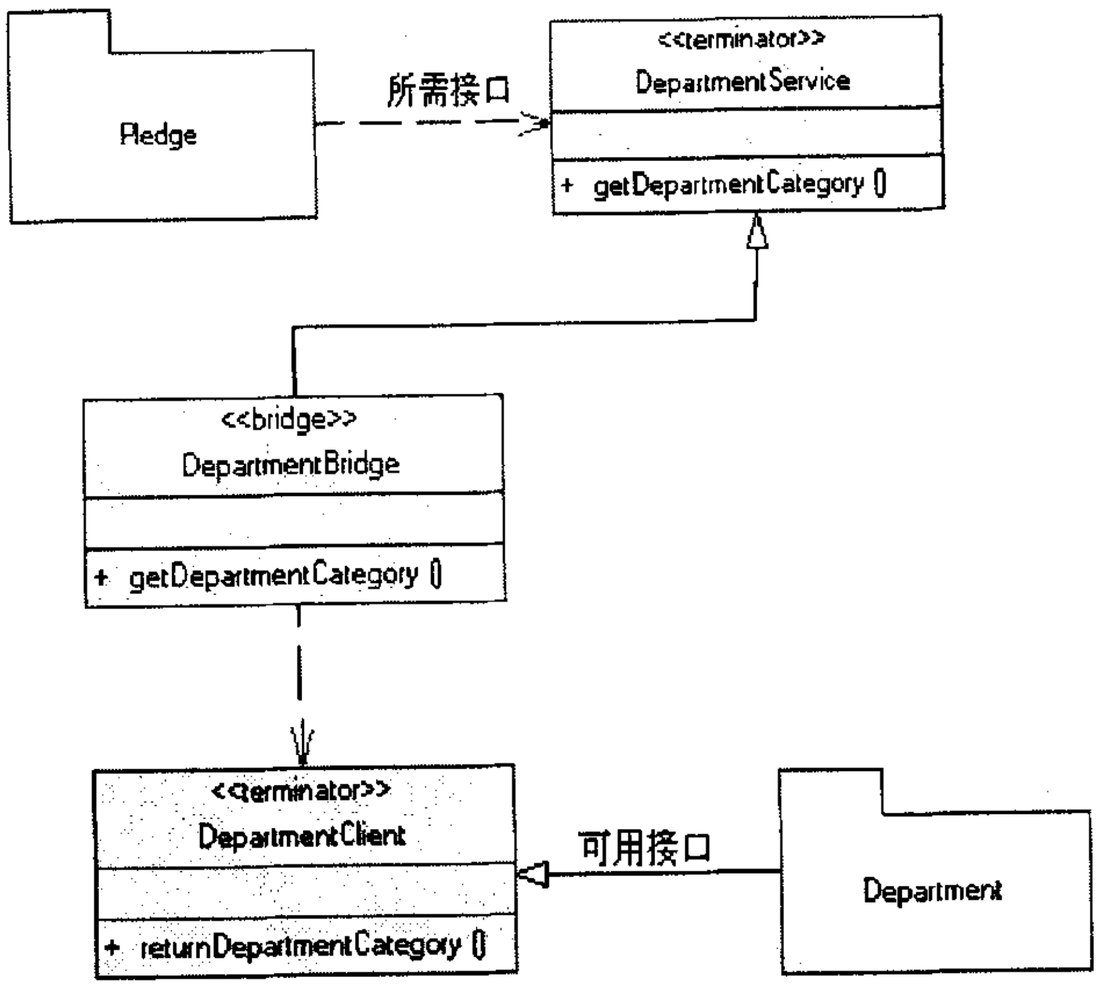

图 3　域之间桥的实现示意图

在图 3 中,Pledge 域中定义了一个所需操作的接口 DepartmentService,Department 域中定义了可用操作接口,该接口包含一个可用的操作 returnDepartmentCategory,可用操作的实现在桥 DepartmentBridge 中使用动作语言进行定义,内容如下:

```
Bridge in Build Set for Domain:Pledge (P)
Required Operations for Terminator:Department (D)
D1:getDepartmentCategory
Contract
Open
Input Parameters
departmentName      Type:Text
Multi Domain Bridge Code
$ USE D
{departmentPar} is departmentType
Departmentpar1 = "Department Name"
Append [Departmentpar1,departmentName] to {departmentPar}
[] = D1::returnDepartmentCategory[{departmentPar}]
$ ENDUSE
```

在桥操作的实现中,$ USE 子句用于指定域上下文,$ USE 子句内语句的执行上下文是 Department 域,而 $ USE 子句外的语句的执行上下文是 Pledge 域。桥 DepartmentBridge 实现域 Pledge 的所需接口时,要借助于 Departmment 域提供的域操作 returnDepartmentCategory,该域操作的方法如下:

```
Domain Scoped Provided Operations
P1::getCategoryName - externally visible
Description
Contract
Open
Input Parameters
departmentPar   Type:departmentParType
Method
[categoryName] = D1:getCategoryName[{departmentPar}]
```

域 Department 通过域操作向桥提供服务,它并不需要知道调用者的信息。当桥要调用域 Department 的服务时,直接调用域操作,而不用调用 Department 域内某个类的操作,因此,使域 Department 和域 Pledge 的耦合度降到最低。

上述桥的实现采用的是动作规约语言(ASL)[5],它们最终将会被转换成运行于 J2EE 平台的 Java 语言。

3.2　从 xUML 业务模型到 J2EE 平台 PSM 的转换

从一个模型生成另一个模型,必须把源模型中的每个元素同目标模型中的一个或多个元素建立联系。xUML 业务模型中的类有三种:边界类、控制类和实体类,将它们记作 InterfaceModel,ControllerModel 和 EntityModel。在 J2EE 平台中,将各种组件分为表示层元素、业务层元素和持久层元素[6],分别记作 WebModel,EjbModel 和 DbModel。WebModel 包括 Servlet,JSP,HTML 和 Form 表单等元素,EjbModel 包括会话 Bean 和实体 Bean,DbModel 是指实现数据持久化的持久化类。由此建立从 xUML 业务模型 PIM 到 J2EE 平台的 PSM 的映射关系如图 4 所示。

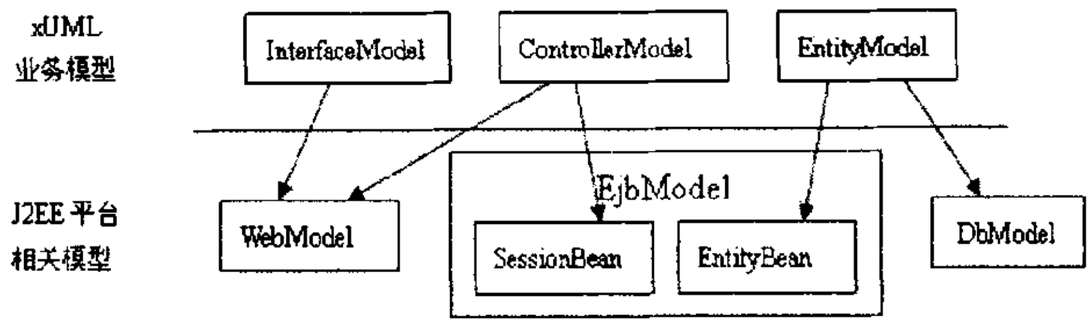

图 4　从 xUML 业务模型到 J2EE 平台相关模型

从 xUML 模型到 J2EE 平台 PSM 的转换包括五个方面:

1)从 EntityModel 到 DbModel 的转换。此时直接把 EntityModel 中的类映射到 DbModel 中的 DbTable 类,把 EntityModel 类的属性映射为 DbTable 类的属性,并根据业务情况确定关键字。

2)从 EntityModel 到 EjbModel 中的 Entity Bean 的映射。此时直接把 EntityModel 中的类映射到 Entity Bean 中的相应类,把 EntityModel 中的类的属性和方法映射为相应类的属性和方法,并根据业务情况确定关键字。

3) 从 ControllerModel 到 EjbModel 中的 Session Bean 的映射。此时直接把 ControllerModel 中的类映射为 Session Bean 中的相应类,并把 ContrllerModel 中类的方法映射为相应类的方法。

4) 从 ControllerModel 到 WebModel 的映射。此时把 ControllerModel 的类映射为 WebModel 中的 HttpServlet 类。

5) 从 InterfaceModel 到 WebModel 的映射。此时把 InterfaceModel 中的类映射为 WebModel 中的 JSP 文件、HTML 文件或 Form 表单。

3.3 Java 代码的生成

在从 PSM 到 Java 代码的转换阶段,把 DbModel 转换成 J2EE 应用系统的后台数据库,这通过根据 DbModel 中的类生成创建数据库表的 SQL 语句来实现。如对于带有属性 kid, name, sex 的类 Person,将生成如下的 SQL 语名:

```
CREATE TABLE PERSON(KID INT NOT NULL, NAME VARCHAR(20), SEX VARCHAR(4))
```

EjbModel 对应 J2EE 架构的持久层,在进行到代码的转换时,将为 EjbModel 中的每个 Entity Bean 和 Session Bean 生成相应的 Java 类代码。每个 Entity Bean 和 Session Bean 都由一个 Home 接口、一个 Remote 接口和一个实现接口的 Bean 类三个 Java 文件组成。

在生成 Entity Bean 类时,将需要在 Home 接口中加入 Create()和 findByPrimaryKey()方法;在 Remote 接口中加入 Entity Bean 的所有方法;在实现接口的 Bean 类中,除了把 Entity Bean 中的所有属性和方法加入其中外,还需要加入 ejbCreate, ejbPostCreate, ejbLoad, ejbStore 等实体 Bean 的缺省方法。

同样,在生成 Session Bean 类时,需要在 Home 接口中加入 Create()方法;在 Remote 接口中加入 Session Bean 的所有方法;在实现接口的 Bean 类中,除了 Session Bean 中的所有属性和方法外,再加入 ejbCreate, ejbRemove, ejbActivate, ejbPassivate 和 setSessionContext 方法。

各类中由 PSM 映射过来的方法,由于在 xUML 模型中已用动作语言进行了准确的定义,它们可由动作语言的元模型和原型语言生成对应的 Java 代码。同时,在 xUML 模型中的每个类都有至多一个状态机,用于表达类对象的状态相关行为,可以用如下方法将状态机整合到拥有状态机的类中:

1)在类中建立一个状态转移表,包括事件名称、当前状态、下一状态等。并将每一个事件对应的状态信息加入到状态转移表中。

2)在类中加入一个当前状态的属性和一个私有函数 GetNextState。

3)为每一个状态增加一个私有函数,封装其对应的过程,输入参数是一个事件类。

4)为每一个输入事件建立一个公有函数,将事件的参数依次填入此函数的参数表中。

5)在输入事件所在函数中增加如下内容:调用 GetNextState 函数,取得下一个状态,如果返回一个有效状态,则将当前状态设置为返回状态,并将输入参数打包成事件类,传入当前状态所对应的过程。

WebModel 对应 J2EE 平台的 Web 层。转换时,将为 WebModel 中的 HttpServlet 类生成相应的代码,除了 PSM 中对应的方法外,还要为 Servlet 类加上 Init(), doPost(), doGet(), Destroy()等方法。同时,对 WebModel 中的每个类生成一个 JSP 文件,在每个 JSP 中,对每个属性都生成一个包含一个名字的头部,并生成一个 useBean 元素,用来访问对应的 Entity Bean 的远程接口;生成使用 EJB 数据对象管理器来获得数据对象的集合并在集合进行迭代的代码,在每个迭代中生成一个 HTML 行为并为每个属性生成一个 getProperty 元素;生成一个名为 MainMenu 的 JSP,对每个组件在 MainMenu 中生成一个指向该组件的 URL,并生成一个 HTML 文件用于启动主菜单。

4 结束语

用 xUML 对业务逻辑进行精确建模,然后将精确的业务模型转换成适用于具体平台的代码,是实现 MDA 的途径之一,但当前将该方法应用于实际开发还有不少困难,主要是支持工具较少,同时对业务模型进行状态建模的复杂性太大,因此当前 xUML 主要应用于嵌入系统的开发中[7]。文中提出了从 xUML 业务模型到基于 J2EE 平台的应用系统的转换方法,对将基于 xUML 的 MDA 方法应用于 J2EE 应用系统的开发进行了探索。下一步工作是将此方法应用于实际项目的开发中,在实践中对此方法进行检验、细化和发展。

参考文献:

[1] Mellor S J, Balcer M J. Executable UML: A Foundation for Model-Driven Architecture[M]. [s. l.]: Addison Wesley, 2002.

[2] Raistrick C, Francis P, Wright J. Model Driven Architecture with Executable UML[M]. Cambridge: Cambridge University Press, 2004.

(下转第 20 页)

3 算法性能分析

为了分析算法的性能，引用文献[6]给出引理 4。

引理 4 极大独立集节点个数 $|MIS| \leqslant 4opt + 1$，其中，MIS 是极大独立集，opt 是最优极小连通支配集的节点个数。

定理 3 CB－MCDS 算法生成的极小连通支配集个数 $|CB-MCDS| \leqslant 8opt + 1$。

证明：由引理 4 得 $|V_1| \leqslant 4opt + 1$，再根据推论 1 需 $|V_1| - 1$ 个网关节点得 $|CB-MCDS| = 2|V_1| - 1$，所以，$|CB-MCDS| \leqslant 8opt + 1$。定理 3 得证。

定理 4 CB－MCDS 算法的时间复杂度是 $O(\triangle)$，消息复杂度为 $O(n\triangle)$。其中，$\triangle$ 为节点的最大度数，n 为总的节点数。

证明：首先证明时间复杂度。节点的时间复杂度主要取决于节点搜索邻居节点的颜色信息(2.1 节算法的行 6 ～ 11)，因而，一个节点在最坏情况下的时间复杂度为 $O(\triangle)$。

对于消息复杂度，这里要分 3 步考虑。第一，2.1 节算法的行 17 和 18 需要向邻居发送消息 $O(\triangle)$；第二，2.2 节算法的行 5 所发送的消息也为 $O(\triangle)$。最后，在算法优化部分，叶子节点状态的变化仅需向一个邻居节点发送通告消息。因而，节点的消息复杂度为 $O(\triangle)$。整个网络的消息复杂度为 $O(n\triangle)$。所以，定理 4 得证。

表 2 是 CB－MCDS 同参考文献[4～6]中的几个典型算法的比较，从表中可以看出，CB－MCDS 算法能以更少的时间和消息复杂度代价，保证分布式生成的极小连通支配集节点个数为常数量级 8，这主要是因为 CB－MCDS 算法使每个节点仅需要同其一跳邻居进行信息交流就能决定自己的颜色。表 2 中的 m 表示网络中的边数。

4 总 结

提出了一种新的分布式的极小连通支配集算法 CB－MCDS，算法的特色在于求解簇头的过程中就逐步确立了网关节点，加快了自组网虚拟主干网的构造。理论分析验证了 CB－MCDS 算法的正确性，得出了算法的时间和消息复杂度等性能，通过同已有的典型算法相比较，CB－MCDS 算法开销小，仅需要很少的本地邻居消息传递，在很小的时间复杂度下，就可以高效地求解出尽可能少的主干节点形成虚拟主干网，其性能明显优于其它算法。

表 2 算法性能比较

	Ratio	Time	Message	Information
文献[4]	$O(n)$	$O(n^2)$	$O(n)$	1 跳邻居信息
文献[5]	$O(n)$	$O(n\triangle^3)$	$\Theta(m)$	2 跳邻居信息
文献[6]	8opt－2	$O(n)$	$O(n\log n)$	类似全局拓扑
CB－MCDS	8opt＋1	$O(\triangle)$	$O(n\triangle)$	1 跳邻居信息

参考文献：

[1] Ram R, Jason R. A brief overview of mobile Ad Hoc networks: challenges and directions[J]. IEEE Communications Magazine, 2002, 40(5): 20－22.

[2] Clark B N, Colbourn C J, Johnson D S. Unit disk graphs[J]. Discrete Mathematics, 1990, 86: 165－177.

[3] Das B, Bharghavan V. Routing in ad－hoc networks using minimum connected dominating sets[C]//In: Proc IEEE International Conference on Communications (ICC'97). Montreal, Quebec, Canada: [s. n.], 1997.

[4] Lin C R, Gerla M. Adaptive clustering for mobile wireless networks[J]. IEEE Journal on Selected Areas in Communications, 1997, 15(7): 1265－1275.

[5] Wu J. Extended Dominating－Set－Based Routing in Ad Hoc Wireless Networks with Unidirectional Links[J]. IEEE Trans Parallel and Distributed Systems, 2002, 9(3): 189－200.

[6] Wan P J, Alzoubi K M, Frieder O. Distributed construction of connected dominating sets in wireless Ad Hoc networks[C]// Proceedings IEEE INFOCOM 2002. New York, USA: [s. n.], 2002: 23－27.

[7] 殷剑宏，吴开亚. 图论及其算法[M]. 合肥：中国科学技术大学出版社，2005.

(上接第 16 页)

[3] 刘建宾，李建忠，余楚迎. 模型驱动体系结构及 xUML 规范在其语境中的探讨[J]. 汕头大学学报，2004，19(4)：58－61.

[4] Booch G, RumBaugh J, Jacobson I. The Unified Modeling Language User Guide[M]. Second Edition. [s. l.]: Addison Wesley, 2005.

[5] Carter K. UML ASL Reference Guide for ASL Language Level 2.5 Revision D[EB/OL]. 2003. http://www.kc.com.

[6] Bacvanski V, Graff P. From Declarative Business Models to J2EE Applications: An MDA Based Approach [EB/OL]. 2001. http://www.inferdata.com.

[7] 孙 强，曹跃龙，张振华. 下一代 UML 执行和转译技术——xtUML[J]. 计算机工程，2004，30(8)：90－91.