# UML 类图向 OWL 本体转换工具的设计与实现

许卓明,顾华建,倪玉燕,朱□琼

(河海大学计算机及信息工程学院,江苏 南京 210098)

**摘要**:介绍了 UML 类图向 OWL 本体转换的方法,给出了基于 J2SE 平台的转换工具(UML2OWL)的设计思想与实现技术.利用 DOM API 解析、提取 UML 类图元素,并根据 UML-OWL 转换规则,将其转换为相应的 OWL 本体元素.工具开发和案例研究表明,该转换方法在转换 UML 类图到 OWL 本体上是可行的,基于该转换方法的自动转换工具是可实现的.

**关键词**:模式转换;UML 类图;OWL 本体;本体开发;语义网

**中图分类号**:TP311 **文献标识码**:A **文章编号**:1000-1980(2007)04-0477-06

本体(ontology)是语义网知识表示的基础[1].尽管 W3C 已于 2004 年推出了标准 Web 本体语言[2],但是,当前本体开发代价过高的现实一直是制约语义网成功与普及的瓶颈[3].从现有知识源获取领域知识、以(半)自动方式学习本体,是本体开发便捷而有效的途径[4].统一建模语言(UML)[5]是软件分析、设计与可视化建模的工业标准.在 UML 模型中,UML 类图(UML class diagram)专门用于从概念上建模应用领域,因此,研究从 UML 类图中提取领域知识并将其转换为 OWL 本体的方法和工具,对 Web 本体开发具有重要意义.文献[6]进行了 UML 类图向 OWL 本体的转换,但采用的是基于 XSLT 转换的方法.这种方法一方面效率较低,另一方面也不易集成到更大的本体工程环境中.本文对基于 Java 编程来实现 UML 类图向 OWL 本体转换的方法进行了探讨,并给出了相应转换工具 UML2OWL 的设计思想与实现技术.

## 1 UML 类图向 OWL 本体的转换方法

### 1.1 UML 类图与 OWL 本体

UML 类图主要用于描述系统中各个对象的类型及对象间存在的各种静态关系,也用于描述类的特性和操作以及对象连接方式的约束[7].本文以提取 UML 类图中包含的领域知识、构造 OWL 本体为目标,侧重于 UML 类图的概念视角(conceptual perspective)[8],因此仅需考虑 UML 类图的以下元素[5]:

a. 类:表示具有共同特性的一组对象(即类的实例),由类名来标识,其特性包括类拥有的属性以及类(实例)参与的关联等.

b. 关联:表示类(实例)之间的二元或多元关系,可由关联名来标识.

c. 关联类:既是关联又是类,因此,关联类可以拥有属性.

d. 属性:表示对象所拥有的静态特性,由属性名来标识,取值于数据类型(由预定义的数据类型名来标识),并附加重数(即类型中多少实例可以被属性所取值)等.

e. 角色:表示参与关联(包括关联类)的类(实例)在关联关系中所起的作用,由角色名来标识,并附加重数(即类的实例通过角色可以参与的关联关系的最大/小次数)等.

f. 泛化:表示父类与子类之间的二元(继承)关系.

g. 泛化组:具有共同父类的一个或多个泛化可以组成一个泛化组.它可以有 2 种约束:不相交,即泛化组的子类之间没有公共实例;覆盖,即泛化组的子类集的外延完全覆盖了父类的外延.

OWL 语言有 3 个表达力递增的子语言[2]:Lite,DL 和 Full.为了既能捕获 UML 类图中尽可能多的领域知

收稿日期:2007-01-15

基金项目:国家自然科学基金资助项目(60573098);江苏省自然科学基金重点资助项目(BK2003001);江苏省高校"青蓝工程"资助项目

作者简介:许卓明(1965—),男,江苏苏州人,教授,博士生导师,主要从事数据库、信息检索、Web 知识表示及语义 Web 技术研究.

识,又能确保转换成的 OWL 本体具有计算完备性和可判定性,本文选择 DL 子语言作为目标本体语言.DL 子语言有 2 种语法形式[9]:交换语法,即用于在 Web 上交换本体的 RDF/XML 语法;抽象语法,即便于用户评价和理解的框架风格的语法,是交换语法的进一步抽象.这 2 种语法形式具有等价的底层语义.一个用于捕获 UML 类图中领域知识的 OWL DL 本体包含以下元素(不需要个体和个体公理):

a. 类:表示具有共同特性的一组个体(即类的实例),由类描述来定义.类描述可以是一个简单的类标识符(命名类),也可以通过个体枚举、属性限制以及类描述的集合操作来定义(匿名类).

b. 属性:由属性标识符来表示,可分为对象属性和数据类型属性.前者的定义域和值域均是类,用于表示 2 个类(个体)之间的某种关系;后者的定义域是类,值域是数据值域(预定义的 XML Schema 数据类型引用),用于表示类(个体)和数据值之间的属性取值关系.

c. 类公理:用于描述类描述(命名类和匿名类)之间的约束关系,包括包含、等价和不相交关系等.

d. 属性公理:用于描述属性的定义域和值域,属性之间的包含、等价、互逆关系,属性的函数性和逆函数性等全局基约束以及属性的传递性和对称性等逻辑特性.

从以上描述不难看出,虽然 UML 类图与 OWL 本体在语法上不同,但两者存在结构和语义上的对应,且大部分对应是直接的.它们之间主要存在 2 点异构性:一是 UML 类图中的关联(类)可以是多元关系,但 OWL 本体中的对象属性总是二元关系;二是 UML 类图中属性的作用范围仅限于拥有它的类,关联的作用范围仅限于参与关联的诸类组合(属性名和关联名在 UML 类图中不是全局唯一的),但 OWL 本体中所有元素均是“第一类对象”(不管是数据类型属性还是对象属性,它们的标识符均是全局唯一的).

### 1.2 UML-OWL 转换规则

定义转换规则是实现模型间自动转换的关键.UML 类图与 OWL 之间存在结构和语义上的对应是实现从前者到后者自动转换的基础.对它们之间的异构性,可进行必要的处理.具体来说,对 UML 关联(类),不管其表示的是二元关系还是多元关系,均将其进行具体化(reification),即将关联(类)转换为 OWL 类,将类参与关联(类)的角色转换为 OWL 对象属性;对全局不唯一的 UML 属性名和关联名,可进行唯一化处理(如加上其所属的类名).这样,就可以给出以下转换规则:

**规则 1** UML 元素(包括类、关联(类)、属性、数据类型、角色)的名转换为 OWL 元素的标识符(URI 引用中的局部名).特别地,UML 数据类型名转换为 OWL 数据值域(预定义的 XML Schema 数据类型引用),命名不唯一的 UML 属性名和关联名进行 OWL 标识符唯一化处理.

**规则 2** UML 类转换为 OWL 类(不妨称之为实体类).

**规则 3** UML 关联(类)转换为 OWL 类(不妨称之为联系类).

**规则 4** UML 类(包括关联类)拥有的属性及其取值的数据类型和重数,转换为 OWL 数据类型属性,其定义域为此 UML 类所对应的 OWL 类,值域为此 UML 数据类型所对应的 OWL 数据值域;UML 属性的重数转换为 OWL 属性限制中的基数约束.

**规则 5** 表示类参与关联(类)作用的 UML 角色及其重数,转换为 OWL 对象属性,其定义域为此 UML 关联(类)所对应的 OWL 类(联系类),值域为此 UML 类所对应的 OWL 类(实体类);同时,创建与此 OWL 对象属性互逆的对象属性,其属性限制中的基数约束为 UML 角色的重数.

**规则 6** 对 UML 泛化、泛化组及其不相交和覆盖约束进行如下转换:为每个 UML 泛化创建一个 OWL 包含关系类公理,其中子类为 UML 子类所对应的 OWL 类,超类为 UML 父类所对应的 OWL 类;为 UML 泛化组不相交约束创建一组 OWL 不相交关系类公理,用于表示泛化组中诸子类所对应的 OWL 类之间两两不相交;为 UML 泛化组覆盖约束创建一个 OWL 等价关系类公理,用于表示泛化组中共同父类所对应的 OWL 类与诸子类所对应的诸 OWL 类的集合并之间是等价的.

值得指出的是,知识表示模型之间理想的转换规则应该是语义保持的.由于 OWL 语言已有基于模型论的形式语义[9],但 UML 的语义是非形式的[5],因此,要形式证明以上转换规则的语义保持性,必须首先将 UML 类图的语义进行形式化.文献[8]对 UML 类图的形式语义进行了研究,因此,类似于“从 ER 模式到 OWL 本体翻译”的语义保持性[10],本文也完全可以形式化证明 UML 类图向 OWL 本体转换的语义保持性.

## 2　转换工具的设计与实现

### 2.1　体系结构

UML2OWL 的体系结构如图 1 所示，主要包含以下 3 个功能模块：

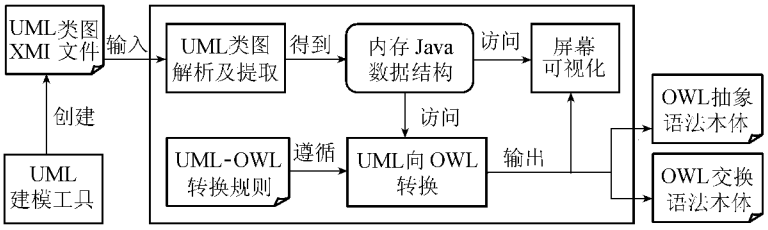


**图 1　UML2OWL 体系结构**

**Fig.1　Architecture of UML2OWL**

**a.** UML 类图解析及提取：读入由 UML 建模工具创建并输出的 UML 类图(XMI 格式文件)，对其进行解析后，从中提取出元素数据，并将其存放于内存 Java 数据结构中.

**b.** UML 向 OWL 转换：访问内存 Java 数据结构，根据 UML-OWL 转换规则，实现从 UML 类图到 OWL 本体的转换，转换结果既向屏幕输出，又生成 OWL 本体文件(抽象语法格式和交换语法格式).

**c.** 屏幕可视化：以内存数据为基础，经必要的格式化处理后，以树形结构显示 UML 类图，以文本形式显示生成的 OWL 本体(抽象语法格式和交换语法格式).

### 2.2　内存数据结构

从 UML 类图中提取出蕴含了领域知识的元素数据并将其存放于内存数据结构中. Java 类设计如图 2 所示.

```
abstract class ElementModel {  //元素抽象模型
    String id = null;              //元素标识
    String name = null;            //元素名
}
class ClassModel extends ElementModel {  //类
    Vector attributeIdVector = null;      //拥有的属性
    Vector roleIdVector = null;           //参与的角色
    Vector parentClassIdVector = null;    //父类
}
class AssociationModel extends ElementModel {  //关联
    Vector roleIdVector = null;                 //所联系的角色
}
class AssociationClassModel extends AssociationModel {  //关联类
    Vector attributeIdVector = null;                     //拥有的属性
}
class AttributeModel extends ElementModel {  //属性
    String objectId = null;                   //所属的(关联)类
    String typeId = null;                     //数据类型
    String maxMultiplicity = null;            //最大重数
    String minMultiplicity = null;            //最小重数
}
    class GeneralSetModel extends ElementModel {  //泛化组
    String parentClassId = null;                   //超类
    Vector childClassIdVector = null;              //子类
    boolean disjoint = false;                      //不相交性
    boolean covering = false;                      //覆盖性
}
class RoleModel extends ElementModel {  //角色
String classId = null;                   //作用的类
String associationId = null;             //参与的关联
String maxMultiplicity = null;           //最大重数
String minMultiplicity = null;           //最小重数
}
```

**图 2　存储 UML 类图元素数据的 Java 类**

**Fig.2　Java classes for storing element data of UML class diagram**

### 2.3　实现技术

本文选择支持 UML 2.0 标准[5]和 XML 元数据交换(XML Metadata Interchange，XMI)2.0 格式[11]的建模工具 MagicDraw UML v11.5 来创建 UML 类图. 图 3 是用该建模工具创建的一个 UML 类图 Production.

UML2OWL 工具基于 J2SE 1.5.0 平台实现. 本文使用 Java 2 提供的 XML 解析器 DOM API for Java 来解析 UML 类图的 XMI 文件. 解析时，先读取整个 XMI 文件(XML 文档)，在内存将其转换为一棵文档对象树，然后就可以多次遍历此树，访问树中的结点信息并进行进一步处理.

图 4 示意了从 UML 类图到 OWL 本体的元素提取及转换的 Java 实现过程. 图 4(a)是 UML 类图 XMI 文件片断，其中包含了典型的文档对象树结点即 UML 类图元素，包括类、关联、属性、角色、泛化及泛化组；图 4(c)是解析提取后得到的内存 Java 类对象(相应的 Java 数据结构见 2.2 节). 图 4(b)表示各类 UML 类图元素的解析提取过程. 重载函数 *extract*(*nodeType*)根据结点类型提取结点数据后存储于相应的 Java 类中. 如：

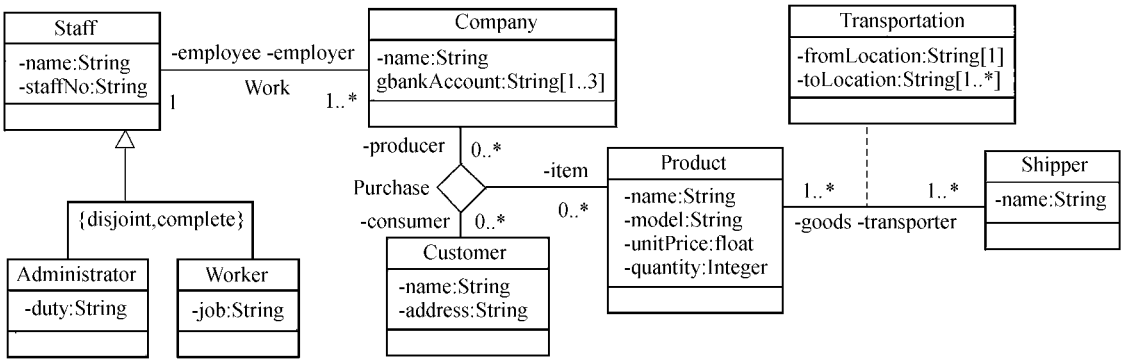


**图3 UML类图**

**Fig.3 UML class diagram**

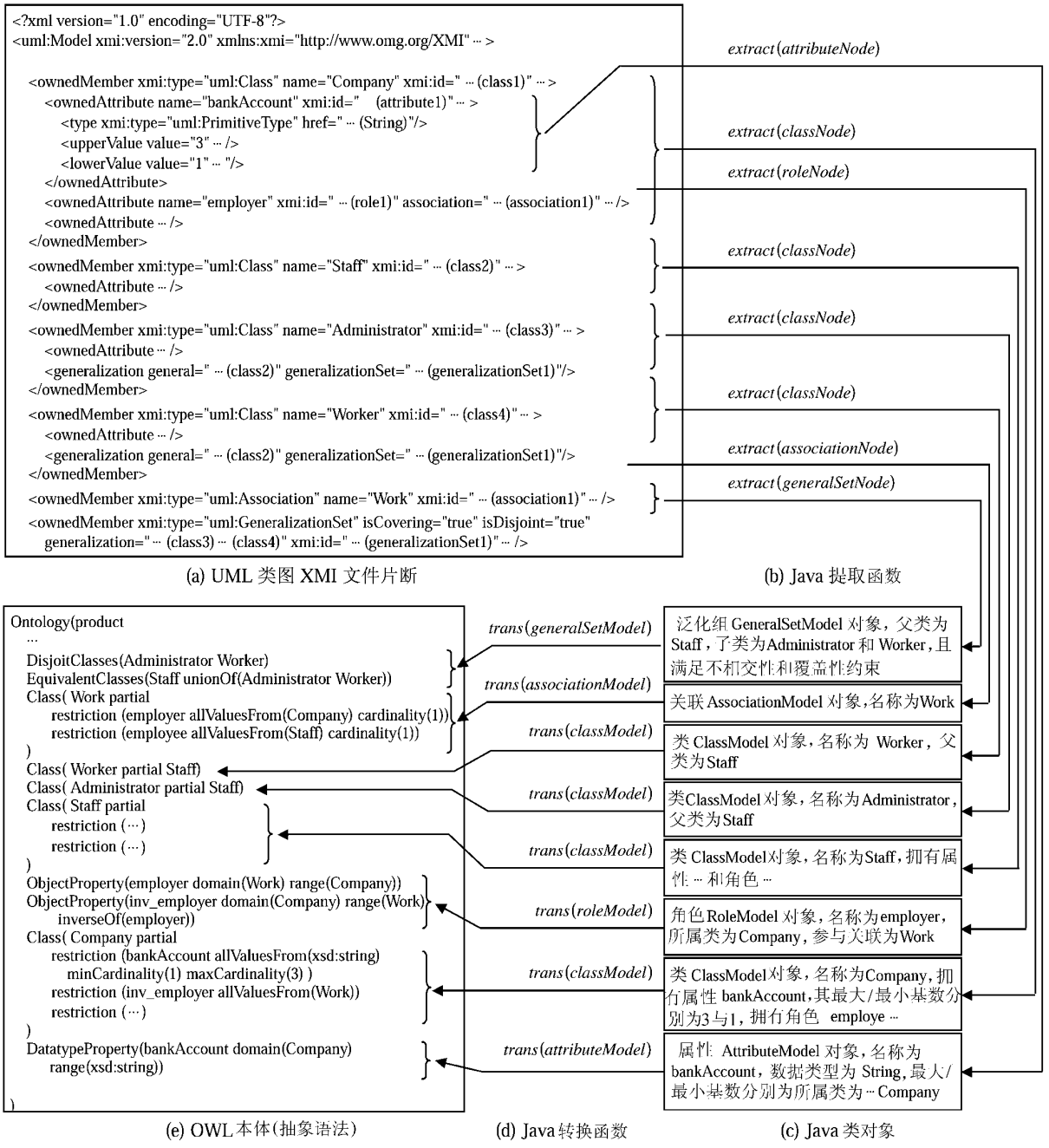


**图4 从UML类图到OWL本体的元素提取及转换过程**

**Fig.4 Process of element extraction and conversion from UML class diagram to OWL ontology**

*extract*(*attributeNode*)提取属性结点数据(属性名为 bankAccount,所属类为 Company,取值类型为 String,最大重数为 3,最小重数为 1)并将其存储到 Java 类对象 AttributeModel(objectId 取值 Company,name 取值 bankAccount,typeId 取值 String,maxMultiplicity 取值 3,minMultiplicity 取值 1)中,其他结点类型数据的提取函数 *extract*(*classNode*),*extract*(*roleNode*),*extract*(*associationNode*),*extract*(*generalSetNode*)具有相似作用.图 4(e)是转换得到的 OWL 本体(抽象语法)文件片断,其中包含 OWL 类、数据类型属性、对象属性、类属性限制、包含及不相交公理等.图 4(d)表示 Java 类对象转换到 OWL 本体的过程.该转换使用重载函数 *trans*(*modelType*)转换不同类型 Java 类对象到相应的 OWL 本体元素.如:*trans*(*attributeModel*)将 Java 类对象 AttributeModel(objectId 取值 Company,name 取值 bankAccount,typeId 取值 String,maxMultiplicity 取值 3,minMultiplicity 取值 1)转换为 OWL 数据类型属性及其公理(数据类型属性 bankAccount,定义域是 Company,值域是 xsd:string)以及关于 OWL 类 Company 的属性限制(属性 bankAccount 全部取值于 xsd:string,最大基数为 3,最小基数为 1).其他重载转换函数 *trans*(*generalSetModel*),*trans*(*associationModel*),*trans*(*roleModel*),*trans*(*classModel*)具有相似作用,分别将相应 Java 类对象转换为相应 OWL 本体元素.

屏幕可视化时,显示 UML 类图(转换源)和 OWL 本体(转换目标).UML 类图元素显示为一棵树,如图 5 左边所示.本文采用深度优先算法构建树模型(Java 对象),树节根结点为以 XMI 文件名命名的空结点(以 表示),根结点的子结点为 UML 顶层类(以 表示)、关联(以 表示)及关联类(以 表示)等元素,类结点的子结点为其属性(以 表示)及参与关联的角色(以 表示).关联类结点的子结点为其属性.转换得到的 OWL 本体(抽象语法与交换语法)均可视化直接用文本框来显示,如图 5 右边所示.

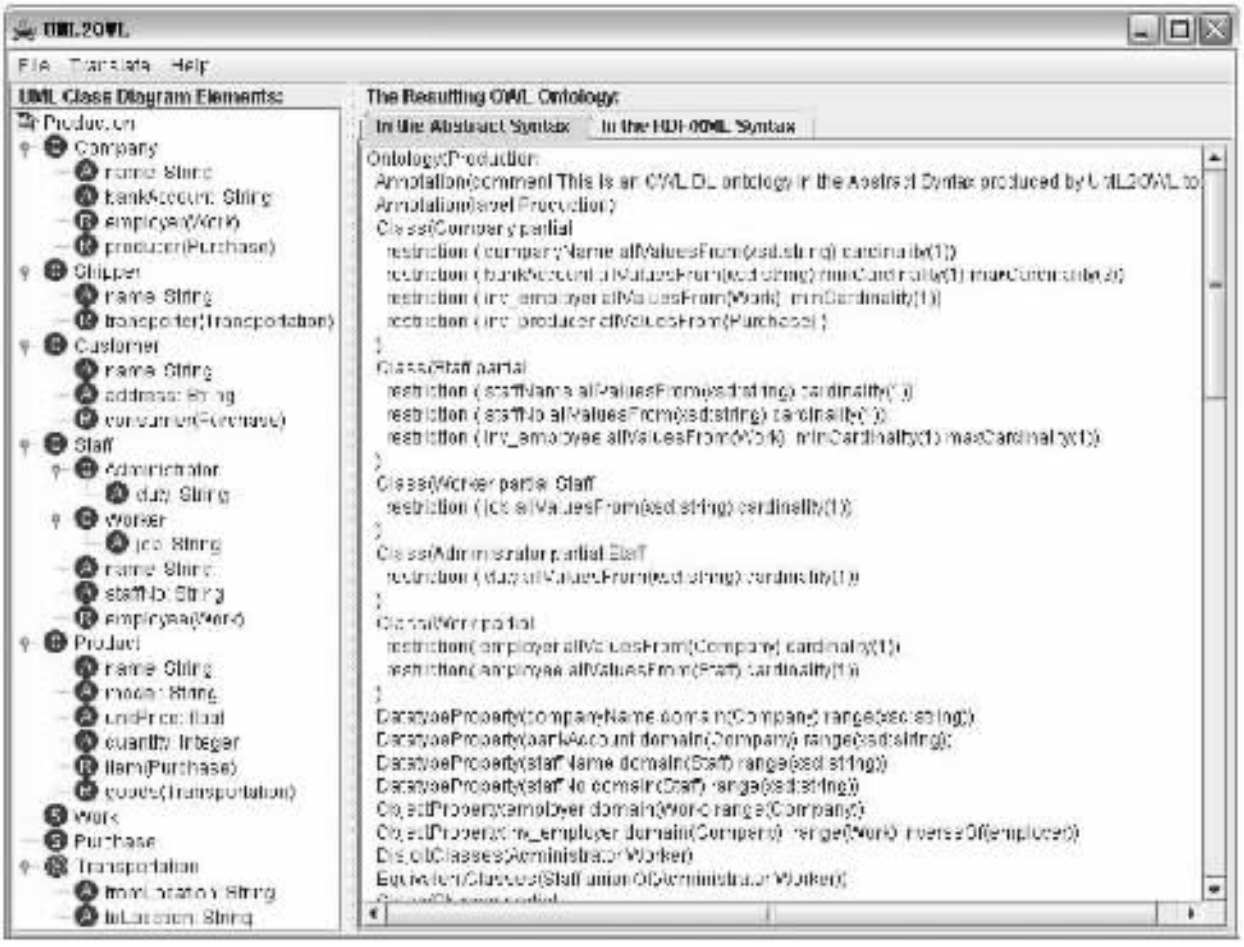


**图 5　UML2OWL 工具截屏**

**Fig.5　Screenshot of UML2OWL tool**

## 3　实　　例

笔者利用已实现的 UML2OWL 工具进行了案例研究,验证了转换方法的可行性以及自动转换的可实现

性.图5的UML2OWL工具截屏给出了一个小型案例:以图3中UML类图Production(XMI文件)为输入,运行UML2OWL后,可输出正确的OWL本体.图5左边是UML类图元素的可视化,右边是转换得到的OWL本体(抽象语法).

## 4 结 语

确定各种知识表示模型之间的对应性和异构性并提供相互转换的工具,对语义网的成功和普及至关重要[3].本文在简要介绍UML类图向OWL本体转换方法的基础上,着重给出了一个基于Java的转换工具UML2OWL的设计思想与实现技术.工具开发和案例研究表明,本文的转换方法是可行的,机器自动转换是可实现的.

## 参考文献:

[1] BERNERS-LEE T,HENDLER J,LASSILA O,et al.The semantic Web [J].Scientific American,2001,284(5):34-43.
[2] DEAN M,SCHREIBER G.OWL Web ontology language reference[EB/OL].(2004-02-10)[2007-01-15].http://www.w3.org/TR/owl-ref/.
[3] SHADBOLT N,BERNERS-LEE T,HALL W.The semantic Web revisited[J].IEEE Intelligent Systems,2006,21(3):96-101.
[4] MAEDCHE A,STAAB S.Ontology learning for the semantic Web[J].IEEE Intelligent Systems,2001,16(2):72-79.
[5] Object Management Group.Unified modeling language superstructure specification,V2.0[EB/OL].(2005-04-04)[2007-01-15].http://www.omg.org/docs/formal/05-07-04.pdf.
[6] NA H S,CHOI O H,LIM J E.A method for building domain ontologies based on the transformation of UML models[C]//Proceedings of 4th International Conf on Software Engineering Research,Management and Applications.[S.l.]:IEEE Computer Society,2006:332-338.
[7] FOWLER M.UML distilled:a brief guide to the standard object modeling language[M].3rd ed.Boston:Addison-Wesley,2003.
[8] BERARDI D,CALVANESE D,GIACOMO G D.Reasoning on UML class diagrams[J].Artificial Intelligence,2005,168(1/2):70-118.
[9] PATEL-SCHNEIDER P F,HAYES P,HORROCKS I,et al.OWL Web ontology language semantics and abstract syntax[EB/OL].(2004-02-10)[2007-01-15].http://www.w3.org/TR/owl-absyn/.
[10] 许卓明,董逸生,陆阳.从ER模式到OWL DL本体的语义保持的翻译[J].计算机学报,2006,29(10):1786-1796.
[11] Object Management Group.MOF 2.0/XMI mapping specification,V2.1[EB/OL].(2005-09-01)[2007-01-15].http://www.omg.org/docs/formal/05-09-01.pdf.

# Design and implementation of a UML-to-OWL conversion tool

**XU Zhuo-ming, GU Hua-jian, NI Yu-yan, ZHU Min-qiong**

(*College of Computer and Information Engineering, Hohai University, Nanjing* 210098, *China*)

**Abstract**: A conversion approach from UML class diagram to OWL ontology was briefly introduced. The design ideas and implementation techniques for J2SE platform-based conversion tool UML2OWL were presented. DOM APIs were adopted as tools for analysis and extraction of UML class diagram elements, and then the extracted elements were converted to the corresponding elements of OWL ontology according to the proposed UML-OWL conversion rules. Tool development and case study show that the conversion approach is feasible, and the automatic conversion tool based on the approach is realizable.

**Key words**: schema conversion; UML class diagram; OWL ontology; ontology development; Semantic Web