第 35 卷 第 1 期 计 算 机 工 程 2009 年 1 月
Vol.35 No.1 Computer Engineering January 2009
·软件技术与数据库· 文章编号：1000—3428(2009)01—0071—03 文献标识码：A 中图分类号：TP311

面向知识管理系统的本体进化管理框架

柯贤达，王英林
(上海交通大学计算机科学与工程系，上海 200240)

摘 要：如何建立知识管理系统的研究是目前的一个研究热点，其中系统的可重构(可配置)性以及适应环境变化的进化能力是研究中的难点。通过基于企业本体来建立知识管理系统的方法，可以实现系统的可重构性。但是随时间变化，企业领域知识的结构、内容也在不停变化，因而本体进化、知识进化方法及一致性模型的研究成为必须要解决的问题。该文分析了本体进化及版本管理的一些关键问题，讨论了相应策略和算法，给出一个本体进化管理框架并引入到知识管理系统的开发中，并介绍了实际的应用情况和今后的研究方向。
关键词：知识管理系统；本体进化；版本管理；一致性

Framework for Ontology Evolution in Knowledge Management System

KE Xian-da, WANG Ying-lin
(Department of Computer Science and Engineering, Shanghai Jiaotong University, Shanghai 200240)

【Abstract】 How to build knowledge management system is becoming a hotpoint of current research. The reconfiguration and evolution ability to adapt the change of environment are the hard points of research. To get the reconfiguration, the KM system based on the domain ontology can be built. However, as the time goes by, the structure and the content of the domain knowledge change. Hence the ontology evolution, knowledge evolution and the consistent model become the key problems that must be resolved. The key issues of ontology evolution and version management are analyzed. The strategy and the algorithm are introduced. Then a framework for ontology evolution is propsed and this framework is adopted into the knowledge management system.The application of this framework and the directions of future research are introduced.
【Key words】 knowledge management system; ontology evolution; version management; consistent

1 概述

企业的创新活动，离不开知识的管理和运用。为了促进企业的创新活动，近年来很多的企业和研究机构对知识管理的方法进行了许多研究[1]。然而，传统的知识管理系统大多局限于特定的企业知识结构进行开发，这种系统的适应性比较差，往往离开特定的问题背景就无法使用。为了改善系统的适应性，在构建知识管理系统时，引入了本体模型，提出了基于本体的可重构知识管理系统[2]。在该系统中，用户可通过系统所提供的本体表示的基础架构，定制适合自身需要的领域本体，并对非结构化知识建立结构化描述和索引，通过定制企业的本体来实现系统重构。这种建立在本体表示基础上的知识管理建模技术提高了系统对不同企业应用场景的适应性。然而，用户需求也往往会随着时间的推移发生变化，领域本体的结构、内容也在不断发生变化。这引发了一系列的相关问题，主要体现为如何对本体的进化进行管理和版本追溯、如何对数据进行一致性验证等。在当前的本体进化研究中，一些本体建模工具，如KAON，以特定格式日志文件来记录本体的变化操作[3]。Klein提出了基于OWL元数据模型[4-5]，定义了本体变化的描述语言，但并没有深入讨论本体变化的约束模型和算法。本文介绍了本体变化的元数据模型、本体变化的相关语义，讨论了一致性约束算法和策略，提出了一个可扩展的通用本体版本管理框架，并把此框架引入到笔者开发的知识管理工具ReKM中，以某钢铁企业和某飞机设计研究院的应用为实例验证了该框架的有效性。

2 本体变化的管理框架

2.1 本体变化的定义

本体变化操作是指对本体的概念或者实例等进行的一次更改操作。定义如下：

定义 1 本体变化操作 *OntoCh* 是一个三元组：

OntoCh =(*name*, *args*, *rules*)

其中，*name*是本体变化操作类型的标识；*args*是本体变化操作的参数，$args \in (C \cup P \cup I \cup L)^n$, $1 \le n \le 3$。*C*是本体的概念集合，*P*是本体属性集合，*I*是本体的实例集合，*L*是字符串集合，参数可以是表示各种数值的字符串；*rules*指按照变化操作本身的语义，应当遵循的规则的集合。

例如，在某钢铁企业的知识管理系统中，对知识结构建立模型，本体可能会发生如下变化：在 *Root* 下增加一个概念“试样”，用<*AddConcept*(“辊子”, “*Root*”), *AddConceptRule*>表示。其中，“*AddConcept*”是本体变化操作操作类型的标识 *name*；“辊子”、“*Root*”是参数 *args*；*AddConceptRule* 是该操作遵循的规则 *rules*。

对本体的任何一次变化操作，首先均需要检测该操作是否满足规则，以保持本体的一致性。对本体 *o* 的某一变化操

基金项目：国家自然科学基金资助项目“企业全生命周期知识管理系统客户化定制与进化方法”(60773088)
作者介绍：柯贤达(1981－)，男，硕士研究生，主研方向：知识管理系统；王英林，教授、博士生导师
收稿日期：2008-05-13 **E-mail：**kexianda@126.com

作 *ch* 是否可以执行，可用如下函数的求值结果给予表示：

定义 2

$$CanDoOperation(ch,\ o) = \begin{cases} \text{true} & \text{操作可以执行} \\ \text{false} & \text{操作不能执行，回滚} \end{cases}$$

2.2 本体变化表达的元数据模型

2.2.1 模型

当前，已经有一些本体编辑工具如 protégé, KAON 提供了日志文件方式记录本体的变化。但是日志文件格式也存在明显的问题：

(1)不容易管理和扩展，添加新的变化类型时，解析日志文件的软件也必须改动。

(2)没有对版本之间的关系进行明确的表达，即不利于版本回滚，也不能够为访问(如搜索)原版本中(历史的)实例提供便捷的手段。

为解决上述问题，可依据本体变化操作的类似性，用一个类/概念(Class/Conception)来表达本体的某种变化。对本体的某次变化，确定该变化类型对应的类之后，通过增加该类的一个实例来记录具体变化。这种方式用本体语言表达，而不是某种特定格式的文件，具有如下优点：

(1)可以更好地进行定义、分类、继承。

(2)具有通用性，可以扩展和增加新的操作类型。

定义 3 本体$Onto_{change}$的一个类(概念)描述一种本体变化类型，其实例记录一次具体的本体变化操作。定义C_{chg}是$Onto_{change}$的概念集合，定义I_{chg}是实例集合。

$Onto_{change}$这个本体通过有继承层次关系的概念/类(*Concept*/*Class*)来描述各种可能的本体变化类型。如图 1 是描述本体变化的一个示例，该图是基于笔者开发的本体建模工具生成的，展示了类之间继承的层次结构。其中描述本体实例变化的类ModifyInst类继承了Change类的一些基本属性。

图 1 描述本体变化的本体类层次图

2.2.2 概念实例的版本管理建模

现有一些本体编辑工具的目的还仅仅是为了编辑本体而编辑本体，其实它只关心当前的本体。在一个企业知识管理的历史过程中，在版本变化后，如何访问到概念实例(知识实例)的历史内容，是实际知识管理工作面临的问题。针对此问题，扩展建立了一个 ModifyInst 类，描述和记录知识实例的变化。

ModifyInst 类继承 Change 类的 OrderNo 属性，这个属性表示本体变化操作的顺序，为本体建模时的版本恢复的实现提供信息。Class, InstID 2 个属性确定是哪个实例的内容发生变化。Property 标注哪个属性值发生改动。InstVerFrom, InstVerTo 2 个属性描述了本体实例版本之间的关系。属性 OldVal, NewVal 记录该实例中发生了变化的属性的新旧版本内容。对于不同类型的值，OldVal, NewVal 里保存的可能是索引，内容保存在其他地方。这样就能把知识实例的历史内容记录下来，提供访问。Note 属性记录备注信息，比如数据改动原因。还有一些其他描述性的属性，如 Date, Author 等。图 2 记录了“国外新技术研究”知识类别的“摘要”属性的一次更改。

OrderNo	class	InstID	Property	OldVal	NewVal	InstVerFrom	InstVerTo	Date	Note	…
200	员工	1_3	职称	工程师	高工	1	2	2006.7.1	破格	…
202	…研究	35_10	摘要	…	…	1	3	2006.12.5	…	...

图 2 记录本体实例变化的数据

2.3 本体变化的类型和一致性约束

2.3.1 类型和约束

为了保证变化操作发生后本体的一致性，必须考虑满足相应的约束。每一种类型的本体变化的约束可以用特定的算法来表示。*CanDoOperation*()就是一个这样的检查函数，这个函数的具体表达或者实现，就是约束。定义 4 表达本体变化的类型与对应的约束(算法)之间存在的对应关系。

定义 4 如果某一本体变化类型*t*对应的约束算法为*c*，则这种对应关系可以表达为二元关系*Correspondence*(*t*,*c*)。其中$t \in C_{chg}$, $c \in \{RulesCheckAlgorithm\}$。

RulesCheckAlgorithm 算法用于检测该本体变化操作是否可以执行。每一种类型的本体变化对应一个检测算法。而所有这些约束算法构成的集合称之为 *ChangeCheckAlgorithm Base*。

对每一种类型的本体变化的执行，以及对执行后相应版本等信息的记录，也对应一种特定的算法，定义 5 反映了这种对应关系。

定义 5 如果某一本体变化类型*t*对应的执行算法为*e*，这种对应关系可以表达为二元关系*Correspondence*(*t*,*e*)。其中，$t \in C_{chg}$, $e \in \{oOperationAlgorithm\}$。

DoOperationAlgorithm 是在实现本体变化时候可以调用的算法。而所有这些执行算法构成的集合，称之为 *DoOperationAlgorithmBase*。

在每一个算法中，将执行实际的变化，同时根据本体变化的类型，给 *OntologyOfChange* 本体的类增加实例，通过这个实例来记录领域本体的一次变更。

2.3.2 交互的策略

在执行本体变化的过程中，往往需要用户的参与。这是因为一个本体的元素往往与多个其他元素(此处包括概念的实例)相互关联，因此对一个元素的改变，通常情况下将不可避免地影响到其他元素。这时可以采取让用户参与交互的策略。系统明确告诉用户该操作的后果，允许用户来选择，不管用户做哪种选择，本体的一致性将始终得到保持。

采取用户交互的策略可以提高本体进化的灵活性，从而使知识管理系统在企业生命周期更加具有适应环境变化的适应性。

2.3.3 本体变化约束检测和执行的算法示例

以下的算法描述了删除本体概念(类)变化类型的检测过程中的语义。

```
bool CanDoDeleteConcept(Concept c)
begin
  /*取得 c 在本体中的所有关系*/
  Set rs = GetRelation(c)
  foreach( Relation r in rs)
    /*如果不是父子类关系，有关联关系则不能执行操作*/
    if( IsParentSonRelation(r) is false ) then
        return false
```

```
    end if
  end foreach
  /*递归检查子类是否可以删除*/
  Set ss = GetSubConcept(c);
  foreach( Concept subC in ss)
    if(   CanDoDeleteConcept(subC) is false )
      return false
    end if
  end foreach
  /*如果有实例，告知用户后果，用户交互决定*/
  if( HasInstance(c) ) then
    if ( ! UserAgreeToDelete() ) then
      return false
    end if
  end if
  return true
end
```

在上述算法中，对于删除类的本体变化的操作，进行相关的一致性检查，并采用了用户参与的策略。

2.4 本体进化版本管理框架

通过前面的讨论，下面给出一个面向本体进化、版本管理的系统框架。根据前述描述本体变化的本体，以及本体变化类型与语义一致性检查算法之间的对应，变化类型与执行算法的对应，可把本体版本管理分为 4 个阶段，如图 3 所示。

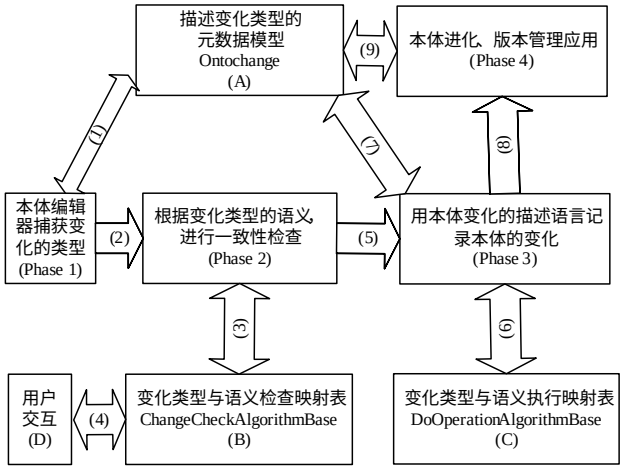

图 3　本体进化、版本管理框架

各阶段说明如下：

(1)本体编辑器捕获本体变化，根据描述本体变化类型的本体，确定本体变化的类型c, $c \in C_{chg}$。

(2)确定本体变化类型后，进行一致性检查，根据变化类型与语义检查映射，调用对应的算法进行一致性约束检查。

(3)如果检查通过，调用相应的算法，执行本体变化操作。并增加变化的类型 c 的实例数据，描述此次本体变化，储存版本管理的信息。这个过程中，用户的参与会影响结果。

(4)$Onto_{change}$的实例数据记录了领域本体变化的描述信息，根据这些信息，可以查询回溯等应用。

3　应用实例

笔者开发了基于本体可重构的知识管理系统。该系统主要包括本体建模、流程管理、知识访问门户等模块，已经成功地应用于某生物制药企业，某钢铁企业和某飞机设计研究院等多个领域，充分体现了系统的可重构性。引入本体进化版本管理框架后，本体建模工具能灵活地改变领域本体的结构内容，提供一定的版本回溯能力，同时保持数据访问的一致性。在知识的获取和评价流程中，版本管理更好地控制了知识的进化。

图 4 所示的示例是在某飞机设计研究院的一个知识推荐的流程中，可以方便地显示知识实例的原来版本。“ 国外新技术研究 ”的属性“ 摘要 ”被更改过，但因改动前后属性的不同版本的值以及相关的信息保存在前文提及的 ModifyInst 类的实例中，所以应用系统能得到知识实例的历史数据。

图 4　知识实例多版本应用

4　结束语

本文介绍并扩充了对本体变化进行表示的元数据模型，具有扩展性和通用性，讨论了本体变化的语义类型和一致性约束，最后给出了一个系统实现的框架。将框架引入笔者先前开发的基于本体的知识管理系统中，更好地解决了企业中本体结构的不断变化和知识对象的不断进化的问题。

在本体和对象的进化过程中，还存在大量的研究内容。例如，在多个本体版本共存下，对各个版本下的知识实例的检索将有特殊性；在分布式环境下，如果允许多个本体建模人员对本体进行编辑，则版本管理和一致性检测将更加复杂。今后将针对上述问题，进行更加深入的研究。

参考文献

[1] Gallupe B. Knowledge Management Systems: Surveying the Landscape[J]. International Journal of Management Reviews, 2001, 3(1): 61-77.

[2] Wang Yinglin, Hu Tao, Zhang Shensheng. Ontology-based Reconfigurable Case-based Reasoning System for Knowledge Integration Systems[C]//Proc. of International Conference on Systems, Man and Cybernetics. [S. l.]: IEEE Press, 2003: 4878-4883.

[3] Stojanovic L. Methods and Tools for Ontology Evolution[D]. Karlsruhe, Germany: Karlsruhe University, 2004.

[4] Klein M, Noy N F. A Component-based Framework for Ontology Evolution[C]//Proc. of Workshop on Ontologies and Distributed Systems. Acapulco, Mexico: [s. n.], 2003.

[5] Klein M. Change Management for Distributed Ontologies[D]. Amsterdam, Netherlands: Virje University, 2004.