图形化与知识形式企业模型间的相互转换

马志斌,王 刚,任秉银
MA Zhi-bin,WANG Gang,REN Bing-yin

哈尔滨工业大学 机电工程学院,哈尔滨 150001
School of Mechatronics Engineering,Harbin Institute of Technology,Harbin 150001,China
E-mail:mazhi_bin@163.com

MA Zhi-bin,WANG Gang,REN Bing-yin. Transformation between graphic enterprise model and enterprise model in knowledge form. Computer Engineering and Applications,2008,44(14):21-23.

Abstract: To support knowledge sharing among enterprises,and to realize distributed and collaborative modeling,a transformation method between enterprise model in graph form and that in OWL form is proposed. Model ontology is built up. Graphic metamodels correspond to classes and graphic model elements correspond to instances of these classes. Relations between graphic model elements correspond to object properties in OWL ontology. Used as a tool that builds model ontology instances,modeling system transforms the whole graphic model to the model described in OWL form. On the other hand,by parsing the model in OWL form,modeling system generates the graphic model.

Key words: model transformation;knowledge sharing;model ontology;OWL;metamodel

摘 要:为有效地支持生产运营的知识共享和协同建模,提出一种在图形化形式与 OWL 本体形式企业模型间相互转换的方法。建立模型本体,图形化元模型对应于本体中的类,图形化模型元素对应于本体中类的实例。图形化模型元素之间的关联,对应于 OWL 本体中实例之间的对象属性。建模系统则作为一个建立模型本体实例的工具,将整个图形模型转化为 OWL 描述的本体实例。反之,通过解析 OWL 本体形式的模型,建模系统生成相应的图形化模型。

关键词:模型转换;知识共享;模型本体;OWL;元模型
DOI:10.3778/j.issn.1002-8331.2008.14.005 **文章编号**:1002-8331(2008)14-0021-03 **文献标识码**:A **中图分类号**:TP391

1 引言

随着企业内部及企业间的交流与合作的加强,生产运营的知识共享与传输成为亟待解决的问题。大型复杂企业的内部集成以及虚拟企业的异地、分布式建模需求,要求建模工具之间对企业模型的共同理解及方便地传输。图形化建模工具不便于网上的传输和集成,不能很好地解决相互之间的通信和对企业知识、企业模型知识等的共享和一致性问题。因此有必要将图形化模型转换为一种建模工具能共同理解的知识化、形式化的表达形式。为解决这个问题,提出了将图形化企业模型转化为由 OWL(Ontology Web Language)[1] 描述的本体知识形式模型的方法。保证企业知识和模型知识的共享、企业模型的描述及其在工具之间的传输。

2 选择 OWL 作为模型本体描述语言

2.1 使用本体构建企业模型

本体(Ontology)[2] 就是用来描述某个领域甚至更广范围内的概念以及概念之间的联系,使得这些概念和联系在共享的范围内有着明确唯一的定义,达成一种共识,这样人与机器之间就可以进行交流。本体的主要功能是为了知识的重用和共享,通过知识表达的一致性实现信息集成。本体的这些特性适用于企业集成建模特别是异地、分布式企业建模的知识表达和共享需求,从而促使在企业建模领域越来越多地使用本体技术。在企业本体中,典型的有多伦多大学的 TOVE(Toronto Virtual Enterprise)[3] 和爱丁堡大学的 Enterprise[4] 项目等。

2.2 OWL 语言的优势

OWL 处于 W3C 的本体语言栈的最上层,具有强大的语义表达与推理能力、良好的数据存储格式、可扩展性强、高度结构化以及便于网络传输的特点,通过简单灵活的标准格式,提供了一个描述和交换数据的有效手段。另外,进行描述语言选择时,应该向标准看齐,而 OWL 作为 W3C 的标准就用于更广阔的应用空间。

3 图形化与本体形式模型转换

建模 的实质就是由建模工具提供元模型,这些元模型是

基金项目:国家高技术研究发展计划(863)(the National High-Tech Research and Development Plan of China under Grant No.2003AA413210)。
作者简介:马志斌(1972-),博士生,主要研究方向:生产运营管理、企业建模、流程分析、知识管理;王刚(1964-),教授,主要研究方向:企业信息化、流程管理、企业建模;任秉银(1966-),博士生导师,主要研究方向:数字化设计与制造、智能结构设计与制造。
收稿日期:2007-12-28 **修回日期**:2008-02-03

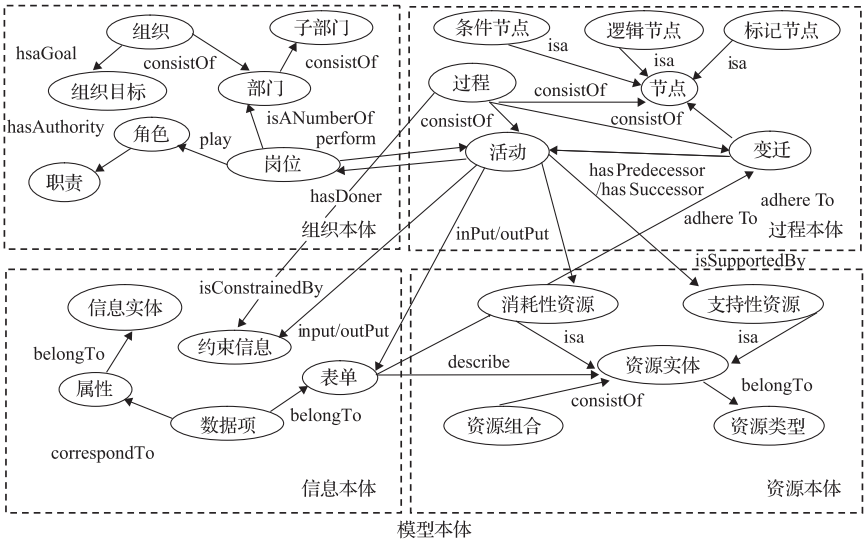

图1 模型本体中主要的类和关系

用来建立企业模型的抽象的模型元素。建模过程的实质,就是用建模工具生成元模型的实例,即模型元素,并建立实例之间的关系。在本体形式模型即模型本体中,“类”(OWL 中的 class)对应于图形化元模型,“关系”(OWL 中的 object property)对应于图形化元模型之间的关系,“实例”(OWL 中的 instance)对应于图形化模型中所建立的各个模型元素。

3.1 本体形式企业模型结构

本文在多视图集成的企业建模[5]工具 IEMS[6]的企业模型理论基础上进行图形化模型与本体知识形式企业模型间的转换。图形化企业模型的过程模型为泳道图形式,如图 4 所示。企业模型包括组织视图、资源视图、信息视图以及作为模型中心的过程视图。由于要用本体形式来表达模型,本文利用描述逻辑[7]对过程和活动进行形式化描述:

过程≡模型元素∩isComposedOf.(活动∪变迁∪节点)∩belongTo.领域∩isConstainedBy.约束信息

活动≡模型元素∩isaPartOf.过程∩belongTo.(领域∩功能层次∩功能类型)∩inPut.(消耗性资源∪表单)outPut.(消耗性资源∪表单)∩isSupportedBy.支持性资源∩isConstainedBy.约束信息

变迁≡模型元素∩isaPartOf.过程∩hasPredecessor.(活动∪过程)∩hasSuccessor.(活动∪过程)

其中,isComposedOf 和 isaPartOf 分别表示“由…组成”和“是…一部分”关系;belongTo 表示“属于”关系;“变迁”是过程/活动间的连接弧,图形化模型中为箭头形式,表示其输送的物料或信息,作为其前驱过程/活动的输出转移到其后继过程/活动作为输入或支持。hasPredecessor 和 hasSuccessor 分别表示变迁与其所连接的前、后活动之间的“从…来”及“到…去”的关系;inPut、outPut、isSupportedBy、isConstrainedBy、hasDoner 分别表示“输入”、“输出”、“由…支持”、“被…约束”和“由…执行”关系,这些属性的值分别是图形化模型中过程的输入、输出、支持、约束和执行岗位;这里的“关系”在 OWL 中,以“对象属性(object property)”形式存在。

限于篇幅,其他模型元素不再用描述逻辑逐一描述,将元模型及其之间关系用模型本体中的类和属性表示,如图 1 所示。模型本体按照视图结构可以划分为过程、组织、信息、资源 4 部分,而这 4 部分本体之间的关系将它们统一起来并实现各视图之间的集成。

3.2 图形化与本体形式模型之间的转换

在建模工具中的一段图形化流程,在模型本体库中建立一个“过程”的实例。在这个过程中的每建立一个图形化模型元素也都在本体库中产生相应的本体实例。图形化元素之间连接及属性页数据项之间的关联,则是本体实例之间的关系。这样,过程的图形化元素及其之间的关联和过程包含的本体实例和关系是一一对应的。图 2 描述了部分图形化元模型与本体中的类的对应关系及部分图形化模型中图形元素之间的连接与本体中属性的对应关系。

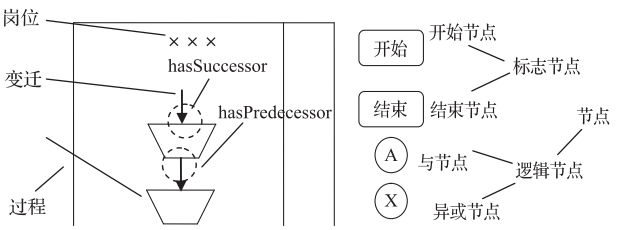

图2 图形化模型与本体模型部分对应关系

以过程视图为例,对于一个过程,由图形化模型向本体形式模型转换的算法如下。

算法 1(模型由图形化向本体形式的转换)

步骤 1 读取图形化过程模型中的图形元素(包括活动、子过程、节点、变迁、泳道),作为本体库中相应元模型类的实例,其名称作为实例的名称。对于没有给出名称的元素,如变迁,按读取的顺序自动赋予实例名称,如“变迁 1”、“变迁 2”等。

步骤 2 对于每个变迁实例,在本体库中,将图形化模型中连接其尾部(无箭头)的图形元素的本体实例作为这个变迁本体实例的属性“hasPredecessor”的属性值。

步骤 3 对于每个变迁实例,在本体库中,将图形化模型中连接其头部(有箭头)的图形元素的本体实例作为这个变迁本体实例的属性“hasSuccessor”的属性值。

步骤 4 在本体库中,将每个泳道名称即岗位的实例,作为图形化模型中位于此泳道中的所有活动的本体实例的属性“hasDoner”的属性值。

步骤 5 将每个图形元素在属性页中的各个数据项,作为本体库中此实例的相应属性的属性值。

步骤 6 将本体库中的整个模型本体实例解析为 OWL 语言表达形式。

对于一个本体模型实例,有本体形式向图形化转换的算法如下:

算法 2(模型由本体形式向图形化的转换)

步骤 1 解析 OWL 文件,将模型包含的相关本体实例(包括类的实例及其之间的属性)读入本体库。

步骤 2 读取本体库中流程中各个岗位的实例,在过程视图中生成泳道,实例名称作为泳道名称。

步骤 3 读取本体库中作为此过程的"isComposedOf"属性的属性值的活动的实例,每个活动在作为其"hasDoner"属性值的岗位实例由步骤 2 所产生的泳道中生成一个活动图形。活动本体实例的名称作为图形化活动元素的名称。

步骤 4 读取本体库每个节点本体实例,随机选择泳道位置,生成图形化节点元素。

步骤 5 读取本体库中每个变迁实例,在过程视图中生成变迁图形元素。这个图形变迁元素的尾部连接变迁本体实例的"hasPredecessor"属性值生成的活动或节点所生成的图形元素,头部连接变迁本体实例的"hasSuccessor"属性值生成的活动或节点所生成的图形元素。

步骤 6 读取本体库中各个本体实例的其他属性值,添加到对应图形化元素的属性页中的各个数据项中。

除了活动处于相应的泳道中,由于没有各个图形化元素的具体位置信息,按照算法 2 生成的图形化过程模型其各个图形元素的位置是不规范的,需手工调整图形化元素位置。

限于篇幅,这里对于 OWL 文件在本体库中的存储不再赘述。另外已有成熟的本体管理和生成 OWL 文件的工具,如 protégé。可以在工具与 protégé 软件中做接口,从而利用 protégé 来完成。

4　系统实现及应用示例

图 3 为图形化模型与知识形式模型转换系统的体系结构图。通过用户界面,利用图形化建模模块生成图形化模型,存储到图形化模型库中。图形化模型与本体库中的本体形式模型通过模型转换模块进行相互转换。利用 OWL 本体存储模块实现本体库中的模型与 OWL 文档间的相互转换。

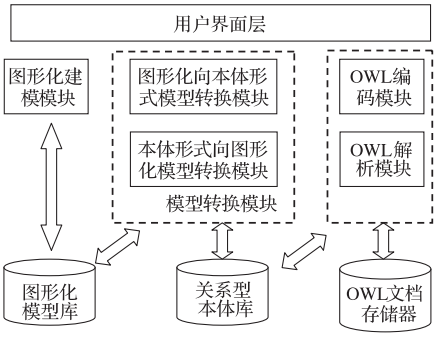

图 3　模型转换系统体系结构

图 4 是建模工具建立的过程"材料出库"的图形化过程模型。由于 OWL 文本较长,故模型只是作为实例说明的简化形式。下面给出系统输出的图 4 中"材料出库"的 OWL 形式模型,由于篇幅关系只给出了模型的片段。

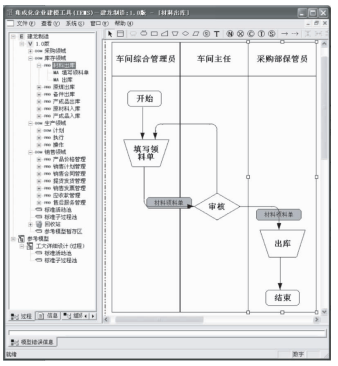

图 4　图形化"材料出库"模型

```
<过程 rdf:ID="材料出库">
  <consistOf>
    <开始节点 rdf:ID="开始节点_1"/>
  </consistOf>
  <consistOf>
    <变迁 rdf:ID="变迁_4">
      <hasSuccessor>
        <人工活动 rdf:ID="出库">
          <inPut>
            <表单 rdf:ID="材料领料单. 审核后">
              <adhereTo>
                <变迁 rdf:ID="变迁_3">
                  <hasSuccessor rdf:resource="#出库"/>
                  <hasPredecessor>
                    <判断活动 rdf:ID="审核">
                      <hasDoner>
                        <岗位 rdf:ID="车间主任">
                          <perform rdf:resource="#审核"/>
                        </岗位>
                      </hasDoner>
                <outPut rdf:resource="#材料领料单. 审核后"/>
                    </判断活动>
                  </hasPredecessor>
                </变迁>
              </adhereTo>
            </表单>
          </inPut>
          <hasDoner>
            <岗位 rdf:ID="采购部保管员">
              <perform rdf:resource="#出库"/>
            </岗位>
          </hasDoner>
        </人工活动>
      </hasSuccessor>
      <hasPredecessor rdf:resource="#审核"/>
    </变迁>
  </consistOf>
  <consistOf rdf:resource="#变迁_3"/>
  <consistOf>
    <结束节点 rdf:ID="结束节点_1"/>
  </consistOf>
  <consistOf rdf:resource="#审核"/>
  <consistOf rdf:resource="#出库"/>
  <consistOf>
    <变迁 rdf:ID="变迁_1">
      <hasPredecessor rdf:resource="#开始节点_1"/>
```

(下转 106 页)