doi:10.3969/j.issn.1673-9833.2013.01.016

软件开发本体构建与模块化的应用研究

方 晓，董 辉，孙士新，朱庆友

（亳州职业技术学院 信息工程系，安徽 亳州 236800）

摘 要：从本体概念出发，提出了软件开发本体构建和代码提取流程，对软件本体的模块化进行定义和分析，研究软件开发本体语义和模块间的逻辑关系，最后将本体模块化应用于协同制造企业中。实例结果表明：软件开发本体构建能提高领域知识共享度和信息的集成度，以及提高协同制造企业应用软件开发的自动化程度。

关键词：软件开发本体；模块化；语义；逻辑关系；协同制造企业

中图分类号：TP311 **文献标志码**：A **文章编号**：1673-9833(2013)01-0071-06

The Applied Research of Software Development Ontology Construction and Modularity

Fang Xiao，Dong Hui，Sun Shixin，Zhu Qingyou

（Department of Information Engineering，Bozhou Vocational and Technical College， Bozhou Anhui 236800，China）

Abstract：Starting from the ontology concept, proposes the process of software development ontology construction and code extraction, defines and analyzes the software ontology modularity, studies the logical relationship between the ontology semantics and modules, finally applies the ontology modularization in collaborative manufacturing enterprises. The application results show that the software development ontology construction improves domain knowledge sharing and information integration, as well as the automation of applied software development in collaborative manufacturing enterprise.

Keywords：software development ontology; modular; semantics; logical relationship; collaborative manufacturing enterprises

0 引言

模块化思想在软件工程中已被广泛应用，其优点是重用性和维护性较好[1]。一个大型软件可分成多个相互关系的模块，且每个模块都包含子模块。近年来也有人做过相关的尝试，将模块化思想运用到知识表示和推理领域。W. Farmer 等人[2]提出了运用模块化与数学结构相结合的方法对复杂的问题进行推理验证，并证明了模块化方法在重用和减少建模成本方面的优势。之后，重用、组织知识块而不是白手起家地建设知识库的思想开始被知识工程界广泛采用。S. Mcllraith 等人[2]提出的知识库的模块化对

收稿日期：2012-07-02

基金项目：安徽省高等学校省级优秀青年人才基金资助项目（2012SQRL248），亳州职业技术学院院级课题基金资助项目（BYK1001）

作者简介：方 晓（1981-），男，安徽黄山人，亳州职业技术学院讲师，硕士，主要研究方向为智能信息处理，E-mail：fx8188@163.com

推理也十分有益，即使是对已有的知识系统进行分解，也能提高推理效率。A. Rector[2]提出了一种使用描述逻辑来模块化实现本体的策略，这种方法虽然能够较方便地建立和重用本体知识，但其仍然将整个模型看作是一个单独的本体，对事件用耦合的概念进行描述。

基于以上研究，本文提出了一种将模块化思想运用到软件开发本体构建中的方法，并将其运用到协同制造企业的软件开发中，以提高该软件开发的自动化程度。

1 软件开发本体构建

1.1 本体构建标准

现在的本体系统[3]是将所有概念组织成一个概念网络，概念之间存在着错综复杂的关系。T. R. Gruber [4]于1995年提出了有益于本体构建的5条标准，该标准不但给建模过程提供有力指导，还对已建成模型的评估起到重要作用。这5条标准如下。

1）明确性和客观性。本体用自然语言对所定义术语给出明确的、客观的语义定义，即必须有效地说明所定义术语的含义。

2）完全性。所给出的定义是完整的，完全能表达所描述术语的含义。

3）一致性。由术语得出的推论与术语本身的含义是相容的，不会产生矛盾。

4）最大单调可扩展性。向本体中添加通用或专用术语时，不需要修改其已有的内容，即支持在已有概念基础上定义新术语。

5）最小承诺。对建模对象给出尽可能少的约束。

本体复用问题是制约本体应用的瓶颈。本体复用可以减少本体规模，而在大规模的知识系统中，本体复用是较困难的， 既要根据用户的需求提取出内容，又要形成完整可用的本体模块。这需解决两大问题，即本体模块分解和本体模块组合。

从本体中选取模块有2种方法，一是模块的划分，二是模块的提取。这两种方法的应用场合不同。模块划分是将一个复杂本体划分为若干个独立本体模块；模块提取则从复杂本体中提取用户所需的本体模块。

利用面向对象技术进行软件开发，程序代码的最基本单元是类，其在计算机的存储是二进制代码形式，而其表现给用户的是文件形式，如doc文件形式、html形式、xml文件形式或其它文件形式。

1.2 本体形成

从上述软件本体构建标准可知，软件开发本体构建包括了7个过程，即代码提取模块、规则化模块、本体形成模块、本体逻辑组合、本体检测、本体实例化和本体评价，其流程如图1所示。先从代码数据库中提取所需的代码文件，再将这些文件发送至代码提取模块，将代码从文件中提取出来。通过规则化模块将代码中的注释和URL等属性去掉，为之后的概念和属性提取做准备，再根据用户的目标化程度来判断是否要再次对代码进行规则化。利用本体形成模块提取核心的概念属性，形成本体概念空间和本体的层次关系。本体逻辑组合是将本体模块按需要进行组合、封装。通过本体检测合格后，进行本体实例化。最后进行本体评价，如果符合标准，就进行文档化处理；否则，转至本体形成模块。

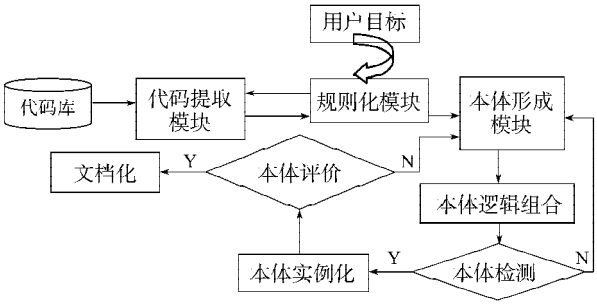

图1 软件开发本体构建流程

Fig. 1 The process of software development ontology construction

1.3 软件代码提取

面向对象的软件开发知识的概念提取，是以软件工程知识为背景语料集，数据源采用面向对象技术知识非结构化数据文档集和已构建的软件开发知识关系数据库。软件知识概念提取框架见图2。

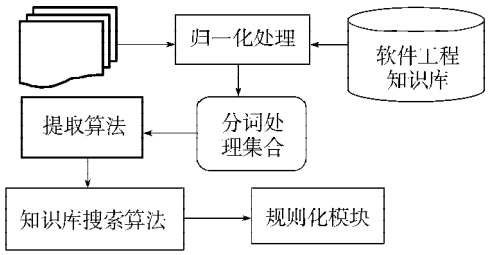

图2 软件代码提取流程

Fig. 2 The process of software code extracting

软件代码提取流程从获取内容上可划分为3个部分。

1）规则化

规则化主要是对非结构化知识数据源中的分词、词性标注、注释、URL、关键字词进行删除，标点及特殊符号进行剔除，为概念提取提供归一化的知识数据源。

2）概念提取

概念提取是在数据源规则化处理的基础上，对归一化的知识源进行概念提取。概念提取有两种方法：基于软件工程知识库的概念搜索算法和基于改进TF-IDF（term frequency-inverse document frequency）算法的概念提取。

3）结果评判

使用准确率和召回率作为评价概念提取结果。

$$准确率=\frac{提取正确的概念数目}{所有的概念数目}，$$

$$召回率=\frac{提取正确的概念数目}{所有提取的概念数目（包含正确和不正确的）}。$$

静态行为获取代码技术往往受到如下因素的限制[5]：1）版权保护使得获取程序的源代码比较困难；2）程序的行为往往依赖于操作系统版本、环境设置、用户输入等因素；3）具有“自修改”特性的程序逐渐增多；4）很多软件使用了抗逆向分析技术，如代码混淆、加密和变形、多态等。动态获取技术可以克服限制静态建模的因素 1, 3 和 4，但其仍然受制于因素 2，从实际应用来看，动态方法比静态方法要简单、实用。因此，本文采取动态方法获取代码。

2　软件本体模块化

2.1　软件本体形式表述

领域本体是一种特定领域知识的专用本体，是对领域知识的概念以及相互关系，领域活动，该领域所具有的特质和规律的逻辑描述。而软件本体是对软件领域知识的形式化描述，由对象、属性、关系以及各个子本体所组成。本体是知识管理的逻辑表达。为了更好地管理和运用领域知识，对本体的内部结构划分是必要的，而本体的内部关系为本体划分提供了可行性。

定义 1　一个领域 D 上的本体 O 形式化定义为一个五元组，即

$$O=\{C, A^C, R, R^A, \sigma\}，\tag{1}$$

式中：C 是领域本体所定义的概念集合；

A^C 是基于 C 中概念的属性集合；

R 是不同概念之间的关系集合；

R^A 是不同属性的关系集合；

σ 是函数，定义为 $\sigma \div C \times C \to R$。

定义 2　软件本体的形式化定义为

$$SO=\{SC, SA^C, SR, SR^A, S\sigma\}。\tag{2}$$

设通用本体 $O=Map\{O_1, O_2, \cdots\}$。以 I 为化分标准，则 $O=\{IO_1, IO_2, \cdots | I\}$。

设 O 有 n 个模块，则 $O=Map\{O_1, O_2, \cdots, O_n, R', \sigma\}$，且满足

$$C=\{C_1 \cup C_2 \cup \cdots \cup C_n\},\tag{3}$$

$$R=\{R_1 \cup R_2 \cup \cdots \cup R_n \cup R'\},\tag{4}$$

$$\sigma=\{\sigma_1 \cup \sigma_2 \cup \cdots \cup \sigma_n \cup \sigma'\},\tag{5}$$

式中：R' 是所有模块具有的概念间关系集合；

σ' 是跨模块的约束。

依据通用本体的划分定义，软件本体的划分定义如下。

设 SO 中有 n 个模块，则 $SO=Map\{SO_1, SO_2, \cdots, SO_n, SR', S\sigma\}$，且满足

$$SC=\{SC_1 \cup SC_2 \cup \cdots \cup SC_n\},\tag{6}$$

$$SR=\{SR_1 \cup SR_2 \cup \cdots \cup SR_n \cup SR'\},\tag{7}$$

$$S\sigma=\{S\sigma_1 \cup S\sigma_2 \cup \cdots \cup S\sigma_n \cup S\sigma'\},\tag{8}$$

式中：SR' 是所有模块具有的概念间关系；

$S\sigma'$ 是跨模块的约束。

为解决网络划分度量的问题，纽曼等人提出了著名的模块度（modularity）的概念[4]。依据此思想，将软件本体中的各模块作为一个节点，软件本体就形成了一个软件网络的概念。度量软件网络方法，即软件本体的模块度为

$$Q=\frac{1}{2m}\sum_{ij}\left(A_{ij}-\frac{k_i k_j}{2m}\right)\delta\left(c_i, c_j\right)，\tag{9}$$

式中：m 表示软件网络中边的总数；

邻接矩阵 $\boldsymbol{A}$ 的元素 A_{ij} 表示连接模块 i 和模块 j 的边数；

$k_i=\sum_j A_{ij}$ 表示模块 i 的度；

c_i 表示模块 i 所属的网络，当且仅当 $c_i=c_j$ 时，$\delta\left(c_i, c_j\right)=1$。

2.2　模块划分准则

领域本体是一个多学科领域知识的结合体，如动画设计领域就涉及美术、设计、软件、影视等许多学科。这些学科领域知识之间关系错综复杂，如何将整个知识混合的本体体系划分成有机的本体模块是需要解决的关键问题。模块划分准则如下：

1）模块的划分要简单、易懂。建立本体系统的目的一方面是使计算机能够更好地理解处理信息，但最基本的还是要让用户来理解；另一方面是利于系统的共享和重用，也有利于系统的维护和扩展。

2）划分的模块应具有知识的逻辑相对独立性。知识逻辑相对独立就能降低模块间的耦合度，减少模块间的通信和相互依赖关系。这样使得整个本体系

统 O 中跨模块的概念间关系的集合 R' 和跨模块的约束集合 σ' 的作用范围都尽可能小。

3）每个模块的内部知识具有高内聚性。高内聚性表现在模块的内部知识依赖程度高，知识的紧密度强，且不可分割。

根据以上准则，软件开发本体系统可划分为软件工程模块、程序语言模块、算法模块、应用领域知识模块、用户模块和硬件平台模块。这些模块本身都是较为独立的知识系统，其中软件工程模块包含了大量的软件开发技术知识；而程序语言模块关系到软件的实现，及对计算机的控制和指挥；应用领域知识模块是为软件在特殊环境和背景中的应用提供知识；算法模块关系到软件质量；显然，用户模块和软件工程模块相对独立，不需要调用其它模块知识。软件本体模块间的调用关系见图3。

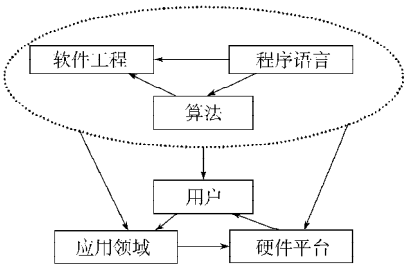

图3 软件本体模块间调用关系

Fig. 3 Transfer relationship between software ontology modules

将软件开发本体模块化与软件开发思想相结合，软件开发本体模块化的特征有：

1）封装性。隐藏模块内部的实现细节，对外仅提供模块的接口。

2）内聚性。具有密切关系的部分（如概念、功能等）应被划分到同一个模块之中。

3）耦合性。软件开发本体模块之间的关系应尽量简化，即模块对外提供的接口要较简单、清晰，且功能较全。

4）独立性。软件开发本体模块之间的低耦合性，即要求各个软件开发本体模块的高内聚性。高独立性意味着各个模块可以独立的改进或扩展，即通过修改模块的某一部分来改进或扩展模块，而不必修改系统内的其他模块。在封装模块时，应努力提高模块的独立性。

5）重用性。某系统开发的模块可不经修改或仅经简单修改就可用于其它系统开发中。直接重用成功的模块或者在其基础上进行简单修改后重用的，可极大地减少系统开发所需的人力、物力和时间投入。

6）标准性。标准化是重用性的深化。模块的接口符合开放式的标准，这意味着模块可以被方便地装配成更大的模块或整个系统，甚至可以“即插即用”，因此，可满足多个系统要求的标准化模块能显著地降低系统开发和生产的成本。

2.3 软件本体模块的语义

各软件本体模块之间存在语法和语义联系。在软件本体模块设计过程中，要考虑语法层面和语义层面上的模块化设计原则。语义模块化划分应遵循以下几个原则：

1）软件本体间上下文的语义环境。每一个本体在上下文语义环境中有各自的位置，这使得本体中概念及概念间的关系只能依据环境和情景才能有正确解释。从语义学上分析，软件本体在上下文语义环境中隐含包含关系（subsum ption），且不同本体具有不同的包含关系。如果将不同的本体组合在一起或复用，隐含包含关系之间就会产生逻辑冲突。在进行本体模块划分时，必须充分考虑这种隐含包含关系之间的相容性[2]。

2）上下文语义环境中本体可部分复用性。语法环境中本体的可部分复用性是静态的，即外部本体中有部分复用的模块。静态复用只能是在外部本体中已存在所需复用模块的设计模式，然而一个本体中不可能包含全部的复用模块，使之能够满足各种各样可能的复用需求。因此，通过语义环境下的复用，可实现动态复用模式，即根据复用需求，从外部本体中找到相关模块进行上下文语义环境下的动态复用。

3）上下文语义环境下的封装性。通过语义对本体的模块化封装，封装的接口与外部联系，将接口与模块内部信息分开，模块内部的信息对外部本体是不可见的（见图4）。模块内部的信息被隐藏，解决了本体内部信息的安全性和稳定性，保证了模块的相对独立性，且能更好地对本体模块进行维护，也为大规模的本体系统平台开发提供方便[2]。

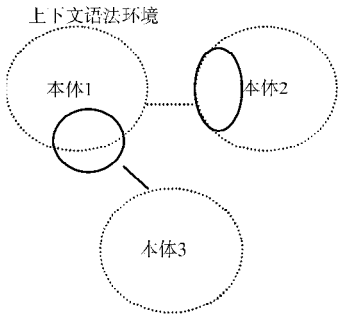

图4 软件本体语义

Fig. 4 The software ontology semantics

本体的结构比较复杂，可将其划分为3个层次：

子模块层、核心模块层和实例层，如图 5 所示。

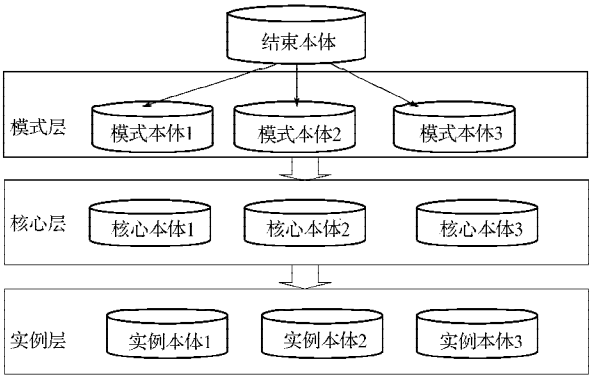

图5　软件本体结构

Fig. 5　The software ontology structure

2.4　软件本体模块的逻辑关系

定义 3　软件开发领域空间用二元组<D, W>表示，其中，D 为软件工程领域，W 是软件工程领域中“事务”的所有可能状态的最大集合。

定义 4　在软件工程领域空间<D, W>上，定义软件本体模块的内涵关系$\delta_{ij}: w \to 2^{D^n}$，表示从 W 到 D 上 n 元外延关系的函数集合。

模块间最基本的关系是部分与整体关系，该关系有包含、部分属于、映射属于等，记为$X \subset Y$，表示 X 是 Y 的部分[3]。部分与整体的关系包括水平关系和垂直关系。

公理 1　部分与整体的关系是非传递、不对称和反自反。

公理2　若X, Y, Z分别是3个模块，则$(\forall Z(Z \subset Y \to Z \subset X)) \to X = Y$ 是 X 等同 Y 的必备条件。

公理 3　$((\forall X = Y) \cap (X \subset Y)) \to (X \subseteq Y)$ 是 X 包含于 Y 的必备条件。

3　应用实例

软件本体的应用主要有 3 个问题：1）软件本体的代表性，是知识集成和共享的标准格式；2）软件本体的建设，是指定的概念和知识之间的关系；3）软件本体映射，是指在用户、关键字、局部和全局范围内的概念在本体库中的本体映射。根据文献[6]所提的在网络化协同制造环境下基于本体的知识集成框架业务流程，本文对制造企业业务进行了模块化设计。

3.1　软件本体在业务流程中的模块化

协同制造业务流程有 3 个主要阶段：产品设计、工艺规划设计和产品制造。每一个阶段和活动涉及到产品不同的隐性和显性知识，而这些知识分布在不同的协作企业中，例如，企业的产品设计是在美国，而规划设计的企业在英格兰和德国，而同一时间，生产产品的企业在中国、越南和巴西。这些企业都交换各种产品的知识和信息，软件本体构建的业务流程可分为产品设计模块、工业规划模块、产品制造模块（见图 6）。

1）产品设计与工艺设计模块。在产品设计和工艺规划设计的协作企业中，关于产品配置和制造之间的评价结果进行信息交换。

2）工艺设计与制造企业模块。在计划过程中，工艺规划设计和制造企业之间进行信息交换。该信息主要是工艺规划的详情，包括工艺路线和机器，设置路线和装置，及其相关参数。

3）产品设计与制造企业模块。制造企业和产品设计之间的信息交换主要有原材料质量、模型、产品质量等参数。

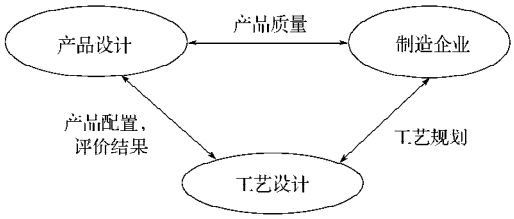

图6　协同制造企业信息交换

Fig. 6　The information exchange of collaborative manufacturing enterprises

3.2　软件本体在协同商务中的模块化

一个领域本体可全面代表一个领域的知识，一个集成的本体可代表所有资料库的知识。通过搜索基于集成本体，用户将得到更全面的、准确的知识。本文基于本体知识集成的框架见图 7，其包括 7 个基本模块{DE, LO, GO, OI, OS, U, KP}。

1）DE 域名企业模块。为占支配地位的企业建立所需知识，并转换成作为本地产品的知识本体。

2）LO 本地本体模块。依据占主导地位的协作企业所建模式，建立一个可共享的本地本体，其负责本地企业的过程知识的整合，然后，将一个或多个分布式当地的本体集成到活动节点作为一个全球性的产品知识本体。

3）GO 全局本体模块。通过使用知识一体化机制，使分散的本地本体产品知识融入占主导地位的企业的全局本体中。因此，合作企业与其它分享自己产品知识的合作企业的知识工作者通过全局本体分享知识。

4）OI 本体集成模块。OI 本体集成方法是在互联网上提供一个公共集成服务器。该集成方法包括 2 个步骤：本体映射和本体合并。第一步是相似度计算，

排序，过滤，并链接本体架构；第二步是关系合并，垂直、水平的概念合并。

5）OS本体搜索模块。在搜寻过程中，根据用户的要求，可以查询综合知识。知识检索的主要步骤如下：查询编辑，索引概念和知识输出。知识输出是通过计算搜索所得知识的相似性，确定最合适的知识。所有的知识存储在设计或制造企业的知识宝库中。

6）U模块。U模块是协同制造环境的主题模块表示用户或用户组，Web服务，应用程序或代理中的一个子系统。

7）KP模块。KP模块是分布式域企业中的知识信息库存储和领域知识表示。

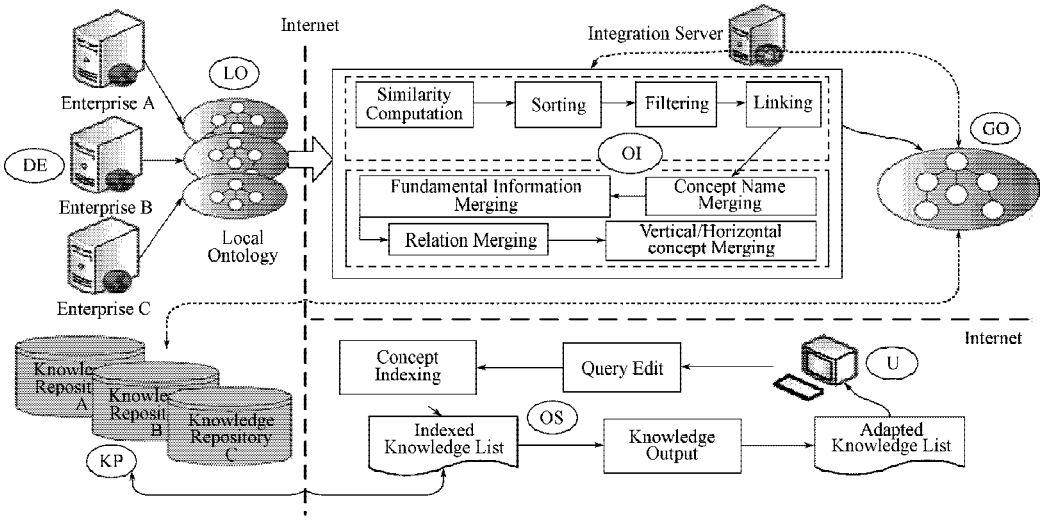

图7 知识集成框架

Fig. 7 The knowledge integrating frame

4 结语

本文从本体出发，定义了软件本体，并提出了软件本体构建框架和软件本体模块化，最后将软件本体模块化应用于协同制造企业。应用实例结果表明：本文所提软件本体模块化促进了制造企业领域知识的共享，加大协同制造企业软件开发中的代码生成，提高软件模块测试质量和软件应用质量。

通过对软件本体的分析及软件行为的研究，找到软件全局测试用例，是本课题组下一步的研究工作。

参考文献：

[1] 龚前伟，蒋翠清. 基于ε-Connections的模块化本体建模[J]. 计算机技术与发展，2010，20(9)：52-56.
Gong Qianwei，Jiang Cuiqing. Study on Modular Ontology Modeling Based on ε-Connections[J]. Computer Technology and Development，2010，20(9)：52-56.

[2] 林松涛. 模块化本体建设研究[D]. 北京：北京邮电大学，2006.
Lin Songtao. Research on the Construction of Modular Ontology[D]. Beijing：Beijing University of Posts and Telecommunications，2006.

[3] 郭文丽，张晓林. 本体模块化：特征、描述与方法研究[J]. 图书馆杂志，2008，27(9)：50-55.
GuoWenli，Zhang Xiaolin. Ontology Modularization：Features, Description and Approaches[J]. Library Journal，2008，27(9)：50-55.

[4] Gruber T R. Towards Principles for the Design of Ontologies Used for Knowledge Sharing[J]. Journal of Human-Computer Studies，1995，43(5/6)：907-928.

[5] 傅建明，陶 芬，王 丹，等. 基于对象的软件行为模型[J]. 软件学报，2011，22(11)：2716-2728.
Fu Jianming，Tao Fen，Wang Dan，et al. Software Behavior Model Based on System Objects[J]. Journal of Software，2011，22(11)：2716-2728.

[6] Jiang Y，Peng G，Liu W. Research on Ontology-Based Integration of Product Knowledge for Collaborative Manufacturing[J]. International Journal of Advanced Manufacturing Technology，2010，49(9/10/11/12)：1209-1222.

[7] Newman M E J，Girvan M. Finding and Evaluating Community Structure in Networks[J]. Phys. Rev. E，2004，69(2)：026113.

（责任编辑：邓 彬）