基于本体概念相似度的软件构件检索方法

柯昌博[1,2] 黄志球[2] 肖 甫[1,3]
(南京邮电大学计算机学院/软件学院 南京 210023)[1]
(南京航空航天大学计算机科学与技术学院 南京 210016)[2]
(江苏省无线传感网高技术研究重点实验室 南京 210023)[3]

摘 要 随着软件重用与产品线技术的日趋成熟,基于产品线采用构件快速地开发出软件产品成为了研究的热点,而高效的构件检索方法是此技术能否实施的关键。因此,使用本体Web语言(OWL)描述构件,并将其转化为本体树进行模糊匹配,然后对失配的构件进行重组,并使用KMP算法对查询本体树的相似概念进行修改,从而检索到更精确且满足用户需求的构件。最后,给出了构件查询算法,并在此基础上开发了构件库查询系统原型,并与采用刻面和特征的查询方法进行了比较实验,结果证明了此方法的可行性与有效性。

关键词 语义网,本体,构件,OWL,KMP
中图法分类号 TP311 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2017.12.028

Software Component Retrieval Method Based on Ontology Concept Similarity

KE Chang-bo[1,2] HUANG Zhi-qiu[2] XIAO Fu[1,3]
(School of Computer Science & Technology/School of Software, Nanjing University of Posts and Telecommunications, Nanjing 210023, China)[1]
(College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China)[2]
(Jiangsu High Technology Research Key Laboratory for Wireless Sensor Networks, Nanjing 210023, China)[3]

Abstract With the development of software reuse and technology of product line, how to develop software product quickly with component based on product line has become the focus of research. The key of implementing this technology is the high efficient component retrieval method. In this paper, we described component with ontology Web language and transformed it into ontology tree to fuzzy matching. Then we restructured the mismatching components and revised similarity concept of query ontology trees with KMP algorithm, in order to retrieve more accurate component and satisfying user requirement. At last, we proposed a component retrieval algorithm and developed a prototype of component repository query system accordingly. By comparing with the query method with facet and feature, we proved its feasibility and effectiveness through experiment.

Keywords Semantic web, Ontology, Component, OWL, KMP

1 引言

随着软件工程的快速发展,面向构件和服务的软件开发方法已经成为了广大软件开发人员的主流开发手段。而面向构件的软件开发方法的最大依赖对象是构件库。因此,对构件的查询方法的研究是面向构件的软件开发的研究热点和重点[1]。对软件构件的表示和查询的方法很多,Frakes等人对现有构件的表示和查询进行了分类[2]。Frakes从构件的表示出发,将现有的方法分为3种[3]:1)基于人工智能的方法;2)基于超文本浏览的方法;3)基于库科学/信息科学的方法。目前主流的构件库系统对构件的描述都采用基于特征或者XML的刻面描述方法。但基于特征的方法缺乏对特征模型组织框架的细致研究和说明,在一定程度上导致了特征模型在表现形式上的冗余性和混乱性,也使得领域分析人员在实践中很难有效地进行领域建模活动[4]。刻面描述方法得到了一定的应用,如REBOOT,NOTO都提出了对软件构件的刻面分类方案[5-6]。国内较为知名的两大构件库系统(北大青鸟的构件库系统和上海政府的构件库系统)也是以XML刻面描述为主的。但是,在语义网的条件下,采用刻面的构件描述方法无法描述领域构件的上下文语义,而基于关键字或者刻面描述的查询方法的查全率和查准率十分有限,不能满足软件开发人员的查询需求。文献[2]首先对构件进行编码,然后

到稿日期:2016-11-06 返修日期:2017-02-01 本文受国家自然科学基金项目(61602262),江苏省自然科学基金(BK20150865),江苏省高校自然科学基金(15KJD520001),中国博士后基金(2016M591842),江苏省博士后科研计划(1601198C)资助。
柯昌博(1984-),男,博士,讲师,CCF会员,主要研究领域为基于本体的软件工程、SaaS服务中的隐私增强技术等;黄志球(1965-),男,博士,教授,博士生导师,CCF高级会员,主要研究领域为软件工程、形式化方法、数据仓库;肖 甫(1980-),男,博士,教授,博士生导师,主要研究领域为无线传感网关键技术等。

建立索引进行查询；文献[7]利用通过查询树与描述树之间的匹配而形成的本体概念相似度进行查询；文献[8]利用相似性进行分析；文献[9]产生聚类算法进行查询。以上这些查询算法都参入了布尔查询，也对布尔查询带来的算法的复杂度进行了有效的分析和解决。但布尔查询本身的属性使得查询的时间性能仍然不高。比如：一个布尔查询可以映射为 2^k 次查询匹配计算。然后再对所有查询计算结果进行对比和优选。显然，这种方法的查询计算复杂且容易出错[7]。

本体是一种形式化的共享概念，是解决异构语义与系统的关键因素[10-12]。因此，基于语义或本体的软件构件的描述和查询方法成为目前构件描述和检索方法研究的热点。文献[13]给出了构件描述模型，从功能、环境和质量属性 3 个方面将用户查询的构件与语义构件库中的构件进行匹配；文献[14]提出了基于本体相似度的构件查询算法；文献[15]和文献[16]给出了构件的静态语义描述、基于语义的构件组装方法和软件体系结构的设计；文献[17]提出了一种基于语义的领域构件接口名称的匹配方法；文献[18]提出一种基于语义的方法来提高软件构件的复用率；文献[19]基于领域本体，设计并开发了一种面向格的软件构件检索平台，并将其应用到软件开发中；文献[20]利用描述逻辑和语义 Web 本体语言 OWL 开发了基于本体的软件构件查询平台；文献[21]利用本体的方法支持软件构件的开发和管理，并利用本体获取被查询构件的属性、关系和行为；文献[22]提出了一种基于本体的 UNL 语名到软件构件之间映射的架构，该架构可用于语义信息检索和构件查询。本文在相关工作的基础上提出了基于本体概念相似度的构件检索算法，将构件的 OWL 描述文档转化为本体树，并进行模糊匹配；然后对失配的构件进行重组，在此基础上使用 KMP 算法对查询本体树的相似概念进行修改，从而得到满足用户需求的构件集；最后给出构件查询算法，并通过实验证明此查询方法的可行性与有效性。

2　基本概念与理论

2.1　构件的 OWL 描述和树型表示

采用 OWL 对每一个领域构件进行描述，然后利用 XPath 提取其中的本体概念，但忽略本体中的对象概念的基本属性，如作者、日期等。基于 OWL 的构件描述文档如下所示：

```
〈ComponentDomain rdf:about="hospital"〉
 〈ComponentCode rdf:resource="#001"/〉
  〈hasVersion〉 Beta 〈/hasVersion〉
  〈hasCreatTime〉 2007-11-9 〈/hasCreatTime〉
  〈hasOS〉 Linux 〈/hasOS〉
  〈hasFunction〉 ProcessImage/DataStore 〈/hasFunction〉
  〈hasDevelopLanguage〉Java 〈/hasDevelopLanguage〉
  〈hasVersion〉 Beta 〈/hasVersion〉
 〈/Component〉
 〈ComponentCode rdf:resource="#002"/〉
  〈hasVersion〉 Beta 〈/hasVersion〉
  〈hasCreatTime〉 2007-11-9 〈/hasCreatTime〉
  〈hasOS〉 Win 〈/hasOS〉
  〈hasDataBase〉 Oracle 〈/hasDataBase〉
  〈hasFunction〉Query 〈/hasFunction〉
  〈hasDevelopLanguage〉 VB 〈/hasDevelopLanguage〉
  〈hasVersion〉 Beta 〈/hasVersion〉
 〈/Component〉
〈/ComponentDomain〉
```

将 XPath 提取出的本体概念进行分层分类，从而形成一棵本体树，构件描述文档对应的本体树及本体树转化的二叉树如图 1 所示。

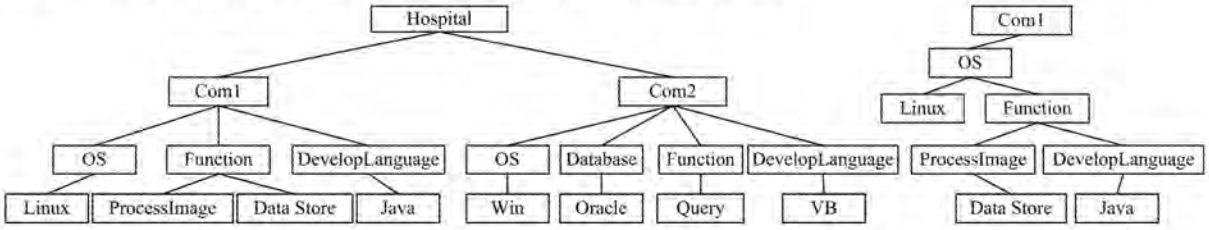

图 1　描述文档对应的本体树及对应的二叉树

2.2　基本概念

定义 1(本体)　本体(Ontology)可以用一个四元组来表示，即 $O=\{C,R,I,A_0\}$，其中，C(Concepts)表示概念的集合，R(Relations)表示概念之间关系的集合，I(Instances)表示实例的集合，A_0(Axioms)表示公理的集合。通常将概念、关系、实例和公理称为本体的元素[23]。

本文方法是基于语义词典 Wordnet 的概念相似度的计算方法。在 Wordnet 中，每个节点 s 表示一个概念，Pantel 和 Lin 等人根据 Wordnet 定义了两个概念的相似度[24]：

$$similarity_d(s_1,s_2)=\frac{2\times\log\ p(s)}{\log\ p(s_1)+\log\ p(s_2)}$$

其中，$p(s)=count(s)/total$ 表示在 Wordnet 中概念节点 s 及其子节点所包含的单词个数在整个词典中所占的比例，$total$ 是 Wordnet 的单词总数，节点 s 是 s_1 和 s_2 的公共祖先节点。

定义 2(本体树)　树是图的一种特例，用 T 表示，$T=(V,E,root(T))$，常省略树根 $root(T)$，即记为：$T=(V,E)$。其中，V 表示树中点的集合，E 表示边的集合。将树中的每一个点的关键字记为节点的概念 $label(v)$，对应本体中的概念。假设构件库中的 OWL 描述文档对应的本体树中的每一个节点的概念都有自己的概念相似度，记为 $Similarity(v)$，且 $Similarity(L)>Similarity(N)$，其中 L 为叶子节点 $Leaf$(简称为叶子)，N 为分支节点 $Node$(简称为节点)。节点的左子树必不为空，即 $v\rightarrow lchild(v)\neq$null，则节点 $Node$ 可以表示为一个三元组：$Node=\{label(N),Similarity(N),label(rchild(N))\}$，其中 $label(N)$ 表示节点对应的本体概念，$Similarity(N)$ 表示概念相似度，$label(rchild(N))$ 表示节点的右孩子所对应的概念。L 叶子的左子树必为空，即 $v\rightarrow lchild(v)=$null，则叶子 $Leaf$ 可以表示为一个三元组：$Leaf=\{label(L),Similarity(L),label(rchild(L))\}$，其中 $label(N)$ 表示叶子对

应的本体概念，$Similarity(N)$表示概念相似度，$label(rchild(N))$表示叶子的兄弟所对应的概念。

定义 3(匹配价值) 设 $T=(V,E,root(T))$为任意的一棵本体树对应的二叉树，则二叉树 T 的匹配价值 $H(M)$为：

$$H(M)=\{\sum_{v\in V}Similarity(v),|V|\}$$

其中，$\sum_{v\in V}Similarity(v)$表示树中所有节点的本体概念相似度之和，$|v|$表示匹配长度，即树中节点的个数。

定义 4(模糊匹配映射) $Q=(V,E,root(Q))$，$D=(V,E,root(D))$为两棵本体树对应的二叉树。其中：

$$H(M(Q))=\{\sum_{v\in V}Similarity(v),|V|\}$$

$$H(M(D))=\{\sum_{w\in W}Similarity(w),|W|\}$$

若$(\sum_{v\in V}Similarity(v)\approx\sum_{w\in W}Similarity(w))\wedge(|V|=|W|)$，则称两棵树为模糊匹配(注："≈"是指在某个阈值范围内成立)。图 2 为模糊匹配图。

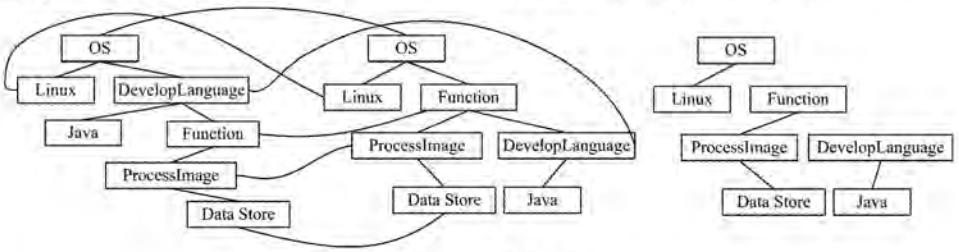

图 2 模糊匹配及子图系列

若两棵树不存在模糊匹配，则将树 D 中的节点($Node$)和连同节点的叶子一起划分成 $|Node|$ 棵子树。同理，将查询树 Q 也划分为若干棵子树。对每棵子树对进行模糊匹配。

可采用下列算法来提取 D 树中的节点 $Node$：

```
p=com→lchild;
while(p→rchild!=null){p=p→rchild;printf("p")}
```

用这种方法可以得到一个节点系列，从而得到子树系列(见图 2)。同时依照同样的分解规则将 Q 树分解成子树。

定义 5(子树的映射)[25] 设 $T_1=(V_1,E_1,root(T_1))$，$T_2=(V_2,E_2,root(T_2))$为两棵本体树对应的二叉树的子树。一个从 T_1 到 T_2 的映射 M 定义为：$M\subseteq V_1\times V_2$，并对所有的(V_1',V_2')，$(V_1'',V_2'')\in M$满足以下条件：

1)$V_1'=V_2'\Leftrightarrow V_1''=V_2''$，表示两棵子树中的节点是一一对应的。

2)$V_1'=ancestor(V_1'')\Leftrightarrow V_2'=ancestor(V_2'')$表示映射间保持祖先后代关系。

映射的定义域为：

$$Domain(M)=\{v_1\in V_1\mid\exists v_2\in V_2:(v_1,v_2)\in M\}\subseteq V_1$$

映射的值域为：

$$Range(M)=\{v_2\in V_2\mid\exists v_1\in V_1:(v_1,v_2)\in M\}\subseteq V_2$$

对于一棵本体树，允许对树中的节点进行插入、删除和改变这 3 种编辑操作。因此，对于其对应的二叉树的子树，也可以进行这 3 种操作。

定义 6(重组代价) 设 $Q=(V,E,root(Q))$，$D=(V,E,root(D))$为两棵本体树对应的二叉树的子树，M 为从 Q 到 D 的映射，则定义 M 的重组代价 $S(M)$为：

$$S(M)=\sum_{(v,w)\in M}S(label(v)\rightarrow label(w))+\sum_{v\in V-Domian(M)}S(label(v)\rightarrow delete)+\sum_{w\in W-Range(M)}S(label(w)\rightarrow label(v))$$

由重组代价的定义可以看出重组代价由 3 部分组成：1)由于 $label(v)\approx label(w)$而修改查询子树的概念的代价，其中 $label(v)\approx label(w)$表示两个标签在语法上不一致，而在语义上是一致的，即 $similarity(v,w)<1$；2)描述子树中没有此节点，而删除查询子树中此节点的代价；3)描述子树中有此节点，而在查询子树中添加此节点的代价。

定义 7(相似概念集) 设集合 $L=\{l_1,l_2,l_3,\cdots,l_n\}$，$n\in N$，对于 $\forall l_i\ \exists l_j(l_j\approx l_i)$且 $l_j,l_i\in L$，则称 L 为相似概念集。例如：l_1=Application domain；l_2=Application area，那么 $l_1\approx l_2$。

2.3 相似概念的匹配

当查询本体树 $Q=(V,E,root(Q))$与描述本体树 $D=(V,E,root(D))$中的某些本体概念相似，即 $label(v)\approx label(w)$，$similarity_d(v,w)<1$时，需要对查询本体树中的概念进行必要的修改，从而获得更为精确的满足用户需求的构件，以提高构件的查准率与查全率。

由于 KMP 算法在回溯比较多的情况下进行字符串的匹配非常有效，因此本文利用 KMP 算法对相似概念进行匹配，从而修改查询本体树中的概念。本体概念的相似通常有以下两种情况：概念表面含义相似，而上下文语义不相似；概念表面含义不相似，而上下文语义相似。举例如下：

1)Application_area 与 Application_domain 表面含义相似，而上下文语义不同，即不是相似概念的情况。通常 Application_area 是指构件的应用环境，即构件所在的硬件环境(单片机还是 PC 机)或者软件环境(操作系统或者数据库)。Application_domain 通常是指应用领域(办工自动化或者图形图像处理等)。但是，在相似概念匹配时也会比较这两个概念，此情况使得匹配时的回溯量比较大。因此，利用 KMP 算法可以很大程度地提高相似概念的匹配效率。

2)Application_area 与 Application_domain 是相似概念，即 $L=\{$Application_area，Application_domain$\}$。在匹配时，其中任何一个概念与 $l_i(l_i\in L)$匹配，回溯量都比较大。因此，利用 KMP 算法可以提高相似概念的匹配效率。在相似概念匹配中，大部分属于这种情况。

根据重组代价的定义可知，当查询树上的某个节点的概念与描述树中对应的概念不匹配时，从相似概念库中提取与查询树中此概念对应的相似概念集，若相似，则替换查询树中的概念。假设概念集所组成的字符串为：$S_1,S_2,S_3,\cdots,S_n$，简

称概念集串；查询树的概念所对应的字符串为：P_1，P_2，P_3，…，P_n，简称查询子串。若此时应与查询子串中第 $k(k<j)$ 个字符继续比较，则查询子串中前 $k-1$ 个字符的子串必须满足关系式(1)，且不可能存在 $k'>k$ 满足关系式(1)：

$$p_1, p_2, \cdots, p_{k-1} = s_{i-k+1}, s_{i-k+2}, \cdots, s_{i-1} \tag{1}$$

而已经得到的“部分匹配”的结果是：

$$p_{j-k+1}, p_{j-k+2}, \cdots, p_{j-1} = s_{i-k+1}, s_{i-k+2}, \cdots, s_{i-1} \tag{2}$$

由式(1)和式(2)可得等式(3)：

$$p_1, p_2, \cdots, p_{k-1} = p_{j-k+1}, p_{j-k+2}, \cdots, p_{j-1} \tag{3}$$

定义 8 ($next[j]$)　若令 $next[j]=k$，则 $next[j]$ 表明当查询串中第 j 个字符与概念集串中相应字符“失配”时，在查询串中需要重新与概念集串中该字符进行比较的字符的位置。由此可得查询串的 $next[j]$ 函数的定义：

$$next[j]=\begin{cases} 0, & j=1 \\ \max\{k \mid 1<k<j \text{ and } p_1, p_2, \cdots, p_{k-1} = p_{j-k+1}, p_{j-k+2}, \cdots, p_{j-1}\}, & \text{this set is not empty} \\ 1, & \text{others} \end{cases}$$

根据这个定义可以得到下列查询串中的 $next[j]$ 函数值：

j	1	2	3	4	5	6	7	8
querystring	*a*	*b*	*a*	*a*	*b*	*c*	*a*	*c*
$next[j]$	0	1	1	2	2	3	1	2

在求得查询串的 $next[j]$ 的函数值之后，可按如下方法进行匹配：

假设指针 i 和指针 j 分别指向概念集串和查询串中正准备比较的字符，令 i 的初值为 pos，j 的初值为 1。若在匹配的过程中 $S_i=P_j$，则 i 和 j 分别加 1；否则，i 不变，而 j 退回到 $next[j]$ 的位置再比较。若再比较时两值相等，则指针各自加 1；否则 j 再退回到下一个 $next[j]$ 的位置。依次类推，直到出现下列两种情况：

1) j 退到某个 $next[j]$ 值($next[next[\cdots next[j]\cdots]]$)时，字符比较相等，指针各自加 1，继续进行匹配；

2) j 退到值为 0 时，需要将查询串继续向右滑动一个位置，即从概念集串的下一个字符 S_{i+1} 起和查询串重新开始匹配。

下面给出一个匹配过程的例子(只有两趟匹配)：

	↓$i=2$
第一趟	*sls*：a c a b a a b a a b a c a
	qs：a b
	↑$j=2$ $next[2]=1$
	↓$i=2$
第二趟	*sls*：a c a b a a b a a b a c a
	qs：a
	↑$j=2$ $next[1]=0$

3　基于 OWL 的构件匹配

3.1　匹配算法流程及描述

先将本体树转化为对应的二叉树，采用模糊匹配，如果匹配，则算法结束；否则：

1)当第一次匹配时(即当 $n=1$ 时)，按节点($Node$)将本体树分解成 $|Node|$ 棵子树，然后进行匹配，若匹配成功，则结束。

2)当有两次以上的匹配时(即当 $n\geqslant 2$ 时)，根据重组代价进行重组，并利用 KMP 算法进行相似概念的修改，直到所有的子树匹配结束。

算法的具体流程如图 3 所示。

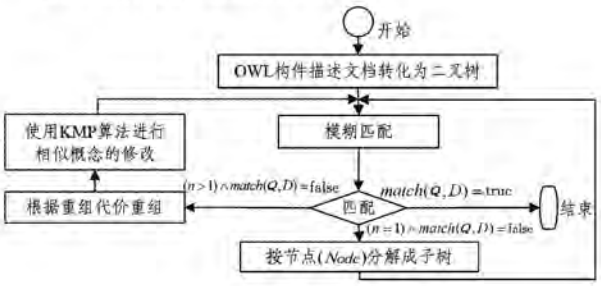

图 3　算法流程

算法 1　匹配算法

```
1. Input:Component:OWL
2. Output:Component
3. n=0;/* n is matching count */
4. pr_i=Xpath (C_i,pr_i);
5. {pr_1,pr_2,…,pr_i,…,pr_n}→ OntologyTree;
6. OntologyTree → BinaryTree;
7. while (n ≤ 1) do
8.   Fuzzy matching;
9.   if(!(Σ_{v∈V} Similarity(v) ≈ Σ_{w∈W} Similarity(w)) ∧ (|v|=|w|))) do
10.      BinaryTree → {ST_1,ST_2,…,ST_n}
11.      n++;
12.    end if
13.  else break;
14. end while
15. while(n>1) do
16.   Σ_{(v,w)∈M} S(label(v)→label(w));
17.   KMP (l_i,l_j);
18.   Σ_{v∈V-Domian(M)} S(label(v)→delete);
19.   Σ_{w∈W-Range(M)} S(label(w)→label(v));
20. end while
```

3.2　匹配算法的复杂度分析

由算法的流程图可知，整个算法只有一个循环，而此循环最多进行两次，因此，算法的时间复杂度为多项式级的。每一步的时间复杂度如下：

1)本体树转化为二叉树是树的遍历和创建二叉树的过程，其时间复杂度为 $O(n)$；

2)模糊匹配是两棵树的遍历，即 Q 树和 D 树的遍历，其时间复杂度为 $O(n)$；

3)按节点分解成子树并进行模糊匹配的时间复杂度为 $O(\sum_{i=1}^{Q.|Node|}\sum_{j=1}^{D.|Node|}(|Q.Leaf_i|+|D.Leaf_j|)+2)$，等价于 $O(n)$，取决于构件及构件库的规模；

4)重组代价的时间复杂度为 $O(|V|\times|W|+\sum_{j=1}^{|V|}w_jh_j+\sum_{r=1}^{|V|}w_rh_r)$,主要取决于构件的规模大小。

由此可知,算法总的时间复杂度为 $O(n+\sum_{i=1}^{|V|}w_ih_i)$。

注:$|V|$为二叉树节点的个数;h_j 为第 j 节点在二叉树上的层次数;节点的权 $w_j=cp_j$,$j=1,2,3,\cdots,n$,其中 p_j 为节点所对应的本体概念相似度的值,c 为常量。

3.3 构件的匹配模型

为了提高构件复用的质量和模糊查询的能力,在原有的构件库的基础上增加 OWL 描述文档库和相似概念库。构件的匹配模型分为应用层和核心层两层(见图 4)。

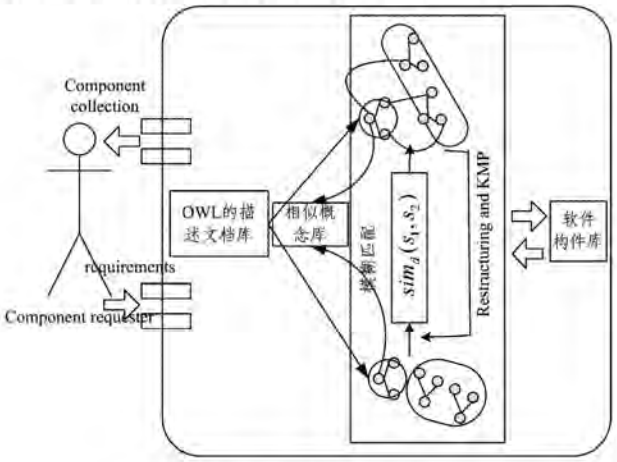

图 4 构件的查询模型

1)应用层:由构件的 OWL 描述文档库、相似概念库和软件构件库 3 个部分组成。

2)核心层:主要是构件的匹配,首先从构件的 OWL 描述文档库中提取构件的描述文档,并将其转化为本体树进行模糊匹配。在此过程中,如果匹配不成功,则进行构件的重组,并利用 KMP 算法对相似概念进行修改,以获得更为精确的满足用户需求的构件集。然后,构建文档与之对应的构件自动映射,使用户从构件库中检索到相应的构件。

4 实验与结果分析

在本文中,实验环境为 LENOVO 台式机 Pentium D925 CPU,512M 内存;操作系统为 ubuntu Linux 10.10;实验平台为 Jdk 1.6.0_37,hadoop1.1.1,juno eclipse。依据上文的构件查询模型,使用 eclipse 开发了一个基于本体的构件查询原型,并采用 Protégé 本体建模工具构建了领域构件本体。实验共采用了 120 个构件,并将其存放在构件库中,其中大部分构件是从上海政府构件库中提取的自由构件[26]。对每个构件分别使用 OWL 和刻面描述方式进行描述。将本体的元素作为概念库。实验结果主要以查准率和查全率作为对比参数。查准率(Precision):$P=\frac{NRRC}{TNC}$,指匹配结果中的正确的匹配数与总构件数的比值。查全率(Recall):$R=\frac{NRRC}{NRC}$,指匹配结果中的正确的匹配数与识别到的构件数的比值。其中,$NRRC$ 表示识别到的正确匹配结果,TNC 表示构件的总数,NRC 表示未识别到的构件数。具体实验结果数据如表 1 所列。

表 1 构件检索性能比较

Method of Component Retrieval	Total Number of Component(TNC)	Number of Retrieval Component(NRC)	Number of Retrieval Relevant Component(NRRC)	Precision(P)/%	Recall(R)/%
Keyword-based	120	162	57	47.5	35.1
Facet-based	120	154	86	71.6	55.8
Ontology-based	120	131	114	95.0	87.0

本文采用了 132 个关键字并使用了包含“功能”“应用领域”“层次”“使用环境”和“表示方法”5 个刻面,共 82 个术语进行对比实验。由实验数据可知,基于本体的构件查询方法在查准率和查全率上明显高于基于关键字的查询方法和基于刻面的查询方法。在查准率方面,所提方法高于基于刻面的 23.4%,高于基于关键字的 47.5%;在查全率方面,所提方法高于基于刻面的 31.2%,高于基于关键字的 51.9%。

结束语 本文提出了一种基于本体的构件检索算法。该算法通过模糊匹配提高了构件的模糊查询的能力,利用本体树转化为二叉树,明确了节点与叶子的区别,并通过节点分解二叉树,从而缩减了算法的规模,使其易于理解,同时给出了相应的匹配算法。该匹配算法实现了与布尔无关的执行过程,使算法复杂度由原来的指数级降低到多项式级,提高了算法的效率、构件查询的查准率和查全率;同时用 KMP 算法对相似概念进行匹配并对失配构件进行重组。最后,给出了构件库查询模型,并开发了构件库查询原型,实验证明基于本体描述的构件的查全率和查准率有很大提高。下一步将进一步研究基于 OWL 的构件描述文档的相似度,并将本体映射应用到构件的查询中,从而更大地提高构件的查询效率。

参 考 文 献

[1] VYATKIN V. Software engineering in industrial automation: State-of-the-art review[J]. IEEE Transactions on Industrial Informatics,2013,9(3):1234-1249.

[2] ZHANG L,CHEN L C,PAN L H,et al. Study on Tags Representation of Components and Tags Based Components Retrieval [J]. Journal of Chinese Computer Systems,2013,34(5):1076-1079. (in Chinese)
张雷,陈立潮,潘理虎,等. 构件的标识表示与检索方法研究[J]. 小型微型计算机系统,2013,34(5):1076-1079.

[3] WANG Y L,ZENG G S. Trust evaluation method for component reuse based on component use dependency relation[J]. Journal of Computer Applications,2015,35(12):3524-3529. (in Chinese)
王燕玲,曾国荪. 基于构件使用依赖关系的构件复用可信度计

算方法[J]. 计算机应用,2015,35(12):3524-3529.

[4] ZHANG W,MEI H. A Feature-Oriented Domain Model and Its Modeling Process[J]. Journal of Software,2003,14(8):1345-1356. (in Chinese)
张伟,梅宏. 一种面向特征领域模型及其建模过程[J]. 软件学报,2003,14(8):1345-1356.

[5] GAO M,JIN C Q,WEI N Q. Real-time and personalized recommendation on microblogging systems[J]. Chinese Journal of Computers,2014,37(4):963-975.

[6] JUNG D W,KIM W H,WILLIAMS D R. Reprogram or reboot: small molecule approaches for the production of induced pluripotent stem cells and direct cell reprogramming[J]. ACS Chemical Biology,2013,9(1):80-95.

[7] XU R Z,QIAN L Q,CHENG J P,et al. Research on Matching Algorithm for XML-Based Software Component Query[J]. Journal of Software,2003,14(7):1195-1202. (in Chinese)
徐如志,钱乐秋,程建平,等. 基于 XML 的软件构件查询匹配算法研究[J]. 软件学报,2003,14(7):1195-1202.

[8] XIE B H,CHEN L C,ZHANG L,et al. ATE Expression of Software Component and Its Clustering Method[J]. Computer Engineering,2012,38(11):42-44,47. (in Chinese)
谢斌红,陈立潮,张雷,等. 软件构件的 ATE 表示及其聚类方法[J]. 计算机工程,2012,38(11):42-44,47.

[9] MUHSIN B,SAMPATH A,GRUBER T. Systems and methods for storing,analyzing,retrieving and displaying streaming medical data:U. S. Patent 9,142,117[P]. 2015-9-22.

[10] ARCH-INT N,ARCH-INT S. Semantic ontology mapping for interoperability of learning resource systems using a rule-based reasoning approach[J]. Expert Systems with Applications,2013,40(18):7428-7443.

[11] LI J,TANG J,YI L,et al. RiMOM:A Dynamic Multistrategy Ontology Alignment Framework[J]. IEEE Transactions on Knowledge and Data Engineering,2009,21(8):1-12.

[12] AMAGASA T,ZHANG F,SAKUMA J,et al. A scheme for privacy-preserving ontology mapping[C]// Proceedings of the 18th International Database Engineering & Applications Symposium. ACM,2014:87-95.

[13] LU J J,SONG P Z. On Component Retrieval Method Based on the Combination of Facets Description and Domain Ontology[J]. Computer Applications and Software,2013,30(8):36-38. (in Chinese)
陆敬筠,宋培钟. 领域本体和刻面描述相结合的构件检索研究[J]. 计算机应用与软件,2013,30(8):36-38.

[14] LI W J,YUAN C A,LIAO W Z. Research on Software Component Query Algorithms Based on Ontology Similarity[J]. Computer Engineering and Science,2010,32(8):153-154. (in Chinese)
李文敬,元昌安,廖伟志. 基于本体相似度的构件查询算法研究[J]. 计算机工程与科学,2010,32(8):153-154.

[15] SHEN G H,ZHANG W,HUANG Z Q,et al. Description Logic Based Feature Modeling and Verification[J]. Journal of Computer Research and Development,2013,50(7):1501-1512. (in Chinese)
沈国华,张伟,黄志球,等. 基于描述逻辑的特征语义建模及验证[J]. 计算机研究与发展,2013,50(7):1501-1512.

[16] PENG X,ZHAO W Y,LIU Y M. Feature Model and Component Semantics Based Conceptual Architecture Design[J]. Journal of Software,2006,17(6):1307-1309. (in Chinese)
彭鑫,赵文耘,刘奕明. 基于特征模型和构件语义的概念体系结构设计[J]. 软件学报,2006,17(6):1307-1309.

[17] ZHANG Z,ZUO C,WANG Y G,et al. Domain component interface identifier matching based on semantic[J]. Journal on Communications,2007,28(5):73-76. (in Chinese)
张正,左春,王裕国,等. 基于语义的领域构件接口名称匹配方法[J]. 通信学报,2007,28(5):73-76.

[18] RODRÍGUEZ-GARCÍA M,VALENCIA-GARCÍA R,GARCÍA-SÁNCHEZ F,et al. Ontology-based annotation and retrieval of services in the cloud[J]. Knowledge-Based Systems,2014,56(3):15-25.

[19] MA Y,HE K,LIU W,et al. A grid-oriented platform for software component repository based on domain ontology[C]// IEEE International Conference on Services Computing (SCC 2007). IEEE,2007:628-635.

[20] PAHL C. An ontology for software component matching[J]. International Journal on Software Tools for Technology Transfer (STTT),2007,9(2):169-178.

[21] OBERLE D,EBERHART A,STAAB S,et al. Developing and managing software components in an ontology-based application server[C]// Proceedings of the 5th ACM/IFIP/USENIX international conference on Middleware. Springer-Verlag New York,Inc.,2004:459-477.

[22] LINHALIS F,DE MATTOS FORTES R P,DE ABEU MOREIRA D. OntoMap:an ontology-based architecture to perform the semantic mapping between an interlingua and software components[J]. Knowledge and Information Systems,2010,22(3):319-345.

[23] ZHONG Q,LI J Z,TANG J,et al. Data Field Based Large Scale Ontology Mapping[J]. Chinese Journal of Computers,2010,33(6):955-958. (in Chinese)
仲茜,李涓子,唐杰,等. 基于数据场的大规模本体映射[J]. 计算机学报,2010,33(6):955-958.

[24] WU W,LI H,WANG H,et al. Probase:a probabilistic taxonomy for text understanding[C]// Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data. ACM,2012:481-492.

[25] MA Y,LIU L,LU K,et al. A graph derivation based approach for measuring and comparing structural semantics of ontologies[J]. IEEE Transactions on Knowledge and Data Engineering,2014,26(5):1039-1052.

[26] Shanghai Component Repository [EB/OL]. http://www.sstc.org.cn.