基于刻面描述的构件检索*

王渊峰[1], 张 涌[1], 任洪敏[1], 朱三元[2], 钱乐秋[1]

[1](复旦大学 计算机科学与工程系 软件工程实验室,上海 200433);
[2](上海计算机软件中心,上海 200233)

E-mail: yfwang@fudan.edu.cn; www_yf@sina.com
http://www.fudan.edu.cn

摘要: 随着软件复用实践的深入和软件构件库规模的扩大,对软件构件的表示与检索的研究正受到越来越多的重视.针对基于刻面描述的软件构件,结合模式分析中的树匹配思想,根据构件刻面描述的特点,提出了一种基于树包含(tree inclusion)的构件检索方法,并进行了理论上的分析与实验上的检验.实验结果证明了它的可行性与有效性.

关 键 词: 构件库;构件检索;刻面;树匹配;软件复用

中图法分类号: TP311 **文献标识码**: A

可复用构件的描述和检索是软件复用和构件库研究的一个重点,它伴随着复用实践的深入和新技术的出现而不断取得进展.针对不同的构件描述形式,研究人员已提出了许多相应的检索方法.例如,Andy Podgurski 等人针对构件的行为表示提出的基于构件行为采样的检索[1];Amy Moormann 等人针对构件的形式化规格说明提出的基调(signature)匹配(接口规约)和规约匹配(功能规约)[2].针对传统的构件文献编目描述,许多研究学者还提出了将神经网络[3]、模糊数学[4]、关联传动[5]等方法应用于构件的检索.目前,构件的刻面描述是一种正逐步得到重视与应用的描述方法.例如,REBOOT,NATO 提出的构件分类方法都是基于刻面的.青鸟构件库中的构件也是采用以刻面分类为主、多种分类模式相结合的方法对构件进行分类描述[6].但是,目前对于刻面描述的构件的检索主要采用的还是以传统的数据库检索技术为主,并结合利用同义词词典和刻面术语间的层次结构来实现构件的松弛匹配[6,7].另外,伴随着构件库面向网络的实践,以 XML 作为构件描述的标记语言已经在 Forbes Gibb 等人的项目中得到了一定的实现[8].但是对于 XML 的构件刻面描述文档,用目前的 XML 检索语言来完成构件检索的任务,还存在值得改进之处.首先,构件库的检索与一般的数据库或文献库中的检索不同,构件的检索需要一定的模糊匹配能力,在保证一定的查准率的情况下提高查全率.另外,构件库的检索需要兼顾对查询的构件的不完全描述,对查询的匹配应有一定的张弛能力,不仅要求能给用户返回匹配的结果还要求能返回相应的匹配程度,为用户复用构件提供有意义的参考信息.另外,各构件库的刻面分类方案可能完全不一样,为了查到合适的构件,用户可能需要跨越多个构件库,如何实现跨构件库的构件检索,对用户有效地屏蔽异质构件的刻面描述间的差别,这也是一个亟待解决的有意义的研究课题.

构件的刻面描述可以展开为一棵刻面描述树,于是构件描述与构件查询间的匹配可以转化为这种树形的结构化描述之间的匹配.我们首先借鉴了有关树匹配方面的研究成果,并在此基础上结合构件匹配的具体特征给出了匹配的条件约束以及匹配代价的定义.由于求解树的匹配代价的算法在计算复杂性上还是一个 NP 难

* 收稿日期: 2001-07-10; 修改日期: 2002-02-25

基金项目: 上海市教委重点学科建设资助项目(B990105)

作者简介: 王渊峰(1974－),男,上海人,博士,主要研究领域为软件复用,构件库系统;张涌(1973－),男,河北南皮人,博士,主要研究领域为软件复用,软件测试自动化;任洪敏(1969－),男,重庆人,博士生,主要研究领域为软件复用;朱三元(1936－),男,江苏苏州人,研究员,博士生导师,主要研究领域为软件工程;钱乐秋(1942－),男,江苏吴江人,教授,博士生导师,主要研究领域为软件工程,CASE 工具与环境,软件复用.

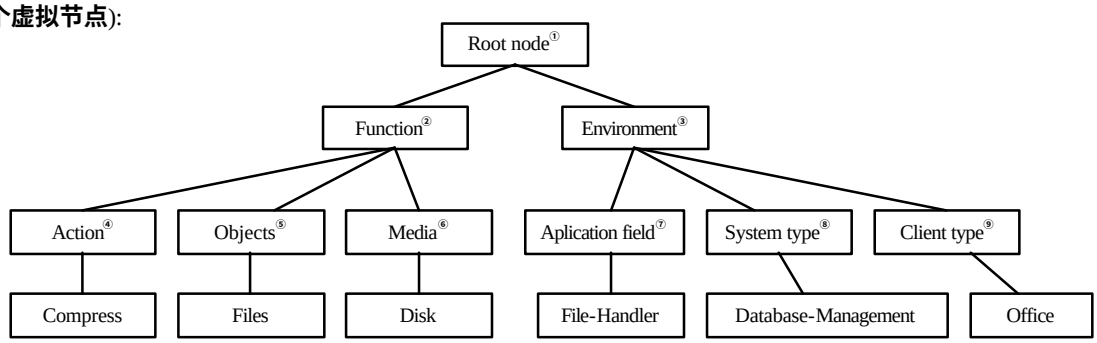
题,在本文中我们首先提出了一般的枚举算法,并对它利用查询树的层次性进行了改进,提出了改进后的枚举算法.最后在理论和实验上对它们的检索性能进行了分析,从一定程度上证明了应用树匹配的思想对刻面描述的构件进行检索这一方法的可行性和有效性.

1 基本概念

1.1 刻面描述及其查询的树建模

一个基于刻面描述的构件库中与检索直接相关的有以下 3 个主要内容: (1) 刻面分类方案;(2) 各个构件的刻面描述集合;(3) 刻面描述术语之间的关系,即术语辞典.

对一个刻面描述方案,我们将其中的刻面、子刻面分别映射为树中对应的父节点、子节点,对采用某个刻面描述方案描述的构件,可以将其相应的刻面描述术语映射为对应的叶子节点.例如:Prieto-Diaz 最早提出来的刻面描述方案为两个主刻面:"功能"和"环境",且每个主刻面分别有 3 个子刻面(作用、对象、媒介)和(应用领域、系统类型、客户类型),在该刻面描述方案下描述的某个构件可以用如图 1 所示的刻面树来表示(其中根节点是一个虚拟节点):



①根节点,②功能,③环境,④作用,⑤对象,⑥媒介,⑦应用领域,⑧系统类型,⑨客户类型.

Fig.1 A facet tree of a component

图 1 一个构件描述的刻面树

对于构件的查询也可以相应地表示为一棵查询树,即将查询中出现的刻面名、子刻面名转化为相应层次的父节点和子节点,将查询的刻面的术语值转化为叶节点,并用一个虚拟的根节点将它们组合为一棵查询树.

于是,构件的检索就转化为查询树与库中构件的刻面描述树之间的匹配.树匹配问题的研究已有许多成果[9,10],与上述匹配直接相关的问题在树匹配中称为标签树的树嵌入(tree embedding)或树包含(tree inclusion)问题,即找到一棵树在另一棵树中的最佳嵌入或包含位置.

1.2 基本概念

结合树匹配的基础知识和构件匹配的具体应用,本文提出以下匹配与匹配代价的概念:

在下面的定义中,设查询树的节点集合为 Q,构件刻面描述树的节点集合为 C.

定义 1(匹配). 如果存在一个映射 $h:Q'\to C$,其中 $Q'\subseteq Q$,满足以下条件,则称存在一个 $Q\to C$ 的匹配 h .

(1) $q_i = q_j \Leftrightarrow h(q_i) = h(q_j), q_i, q_j \in Q'$;

(2) $\text{distance}(label(q_i), label(h(q_j)) \le T$;

其中,$\text{distance}(m,n)\le T$ 表示术语 m,n 具有近似关系, $label(q)$ 表示节点 q 的标签(即术语),T 为由专家设定的阈值.

(3) q_i 是 q_j 祖先节点$\Leftrightarrow h(q_i)$是 $h(q_j)$祖先节点.

对构件匹配定义的 3 个条件作如下的阐述:第 1 个条件要求映射 h 是一个 Q'到 C 的单射,这一点与树包含问题的要求是一致的,但作了微调,即树的包含问题要求是 Q 到 C 的单射,而这里放宽到 Q'到 C 的单射,其中 Q'只要求是 Q 的子集,这意味着允许在匹配中出现查询树中的某些节点找不到匹配节点的情况.这一点改动对增加构件匹配的张弛能力十分重要(虽然会给后继的算法带来更大的难度).第 2 个条件要求所匹配的节点对上的标签(即术语)是近似相等的,而树包含问题中则要求严格相等,这一点改动有益于增强构件匹配的模糊能力.第

3 个条件要求匹配是保持节点间的祖先后辈关系的.这与树包含问题一致,它体现了构件匹配需要保持刻面的层次关系,这使得构件匹配不是单纯的关键字查找,从而具有一定的语义层上的匹配含义.

可以从树之间转化的视角来看待树的匹配,树转化的主要思想是先定义一些编辑操作[11],通过若干个编辑操作可以将一棵树 T_1 转换为另一棵树 T_2.目前比较通用的编辑操作有以下 3 种:(1) 易名操作(change),它表示将某个节点上的标签改变为另一个标签,用 $a\to b$ 表示将节点 a 的标签改变为节点 b 的标签;(2) 删除操作(delete),它表示将某个节点从树中删除,并使得该节点的孩子节点成为该节点父节点的新的孩子节点,用 $a\to\varnothing$ 表示将节点 a 删除($\varnothing$是一个特殊的保留符号,表示空节点);(3) 插入操作(insert),它表示将某个节点插入到树中某一节点下,并使得该节点的部分孩子节点(具体哪一部分对后面的编辑代价的定义并没有影响)成为该插入节点的孩子节点,用 $\varnothing\to a$ 表示将节点 a 插入到树中.对每一个编辑操作 $x\to y, x,y\in$ {标签}$\cup\{\varnothing\}$,赋予一个实数值,称为该编辑操作的编辑代价,记为 $\gamma(x\to y)$.

由定义 1 知道,匹配的实质是一个映射,为了定义符合构件匹配特征的匹配代价,引入下面的映射谱的概念.

定义 2(映射谱). 设 h 为一个从 Q 到 C 的匹配,则 h 的映射谱 $spectrum(h)$ 定义为满足以下条件的节点集合:

$$spectrum(h)=Range(h)\cup\{v\mid\exists v_1,v_2\in Range(h)\quad v_1=ancestor(v)\wedge v=ancestor(v_2)\}.$$

上式中 $Range(h)$ 表示 h 的值域.映射谱的概念是用来描述除映射的值域中的节点以外,还描述了与映射相关的节点.这是为下面定义匹配代价作准备的.

定义 3(匹配代价). 设 h 是一个 $Q\to C$ 的匹配,则 h 的匹配代价定义为

$$\tilde{\alpha}(h)=\sum_{v\in domain(h)}\tilde{\alpha}(v\to h(v))+\sum_{v\in Q-domain(h)}\tilde{\alpha}(v\to\varnothing)+\sum_{w\in spectrum(h)-Range(h)}\tilde{\alpha}(\varnothing\to w),$$

上式中 $domain(h)$ 表示 h 的定义域.

进一步,查询树与构件刻面描述树之间的匹配代价定义为

$$TCost(Q,C)=\min\{g(h)\mid h\text{是一个}Q\to C\text{的匹配}\}.$$

上述定义是在树匹配的基础上结合构件匹配的特征给出的.树匹配中对匹配代价的定义第 3 项中 w 的取值范围是 $C-Range(h)$.而在上面的定义中这一项被修改为 $Spectrum(h)-Range(h)$.这种修改是基于构件匹配中这样的一个事实:各个构件库的刻面描述方案可能各不相同,在计算匹配代价时如果将所有构件刻面描述树中没有被映射到的节点的插入代价都计算在内的话,这必将导致计算到的匹配代价偏大,这是不合适的.通过对上述项的修改,使得计入插入代价的节点只限于与该次匹配有关的节点(映射谱中的),这就避免了节点插入代价的计算偏差,使得对匹配代价的计算更加精确.

另外值得一提的是,构件刻面描述树中节点的编辑代价一般由构件库管理员或专家赋值,再由系统进行归一化处理.而查询树中的节点的编辑代价,则一般由查询用户赋值(默认赋值为 1),再由系统进行归一化处理.

2 算 法

2.1 算 法

在树匹配领域中树包含问题已被证明为 NP 难问题[11].虽然本文下面提出的对查询树与构件刻面描述树之间匹配代价计算的算法的时间复杂度是指数级别的,但通过具体分析以及实验验证,可以证明改进后枚举算法的有效性和可用性.

为了描述改进后的枚举算法,先引入以下的两个概念:匹配点集 M 和合法匹配集 LM.

定义 4(匹配点集 M). 对查询树中节点 v 的匹配点集 $M(v)$ 是符合以下条件的一个集簇:

$$M(v)=\{\{w\}\mid w\in C\wedge\text{distance}(label(v),label(w))\le T\}\cup\{\{\varnothing\}\}.$$

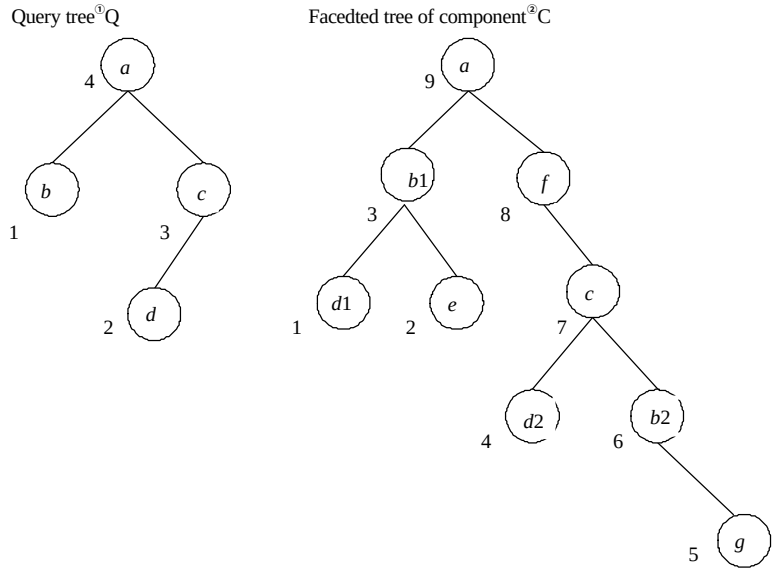
定义 5(合法匹配集 LM). 对查询树中节点集 V 的合法匹配集 $LM(V)$ 是符合以下条件的一个集簇:

$$LM(V)=\{C'\mid C'\subseteq C,\text{存在一个}Q\text{到}C\text{的匹配}h,\text{并且}Domian(h)\subseteq Q[V]\wedge Range(h)=C'\}.$$

上式中 $Q[V]$ 表示以 V 中节点为根节点的查询子树集中的所有节点.

对算法的介绍通过图 2 中的例子来具体阐述.为举例方便,用字母(加数字)来表示节点上的标签,字母相同、

数字不同的标签表示它们具有相似关系.每一个节点的旁边是它的后续编号,用来惟一标示该节点.



①查询树,②构件刻面描述树.

Fig.2 Example for method

图 2 算法示例图

一般枚举方法分为以下 3 个步骤:第 1 步,找到查询树中每个节点 v 的 $M(v)$.例如:图 2 例中有 $M(1)=\{\{\varnothing\},\{3\},\{6\}\}$,$M(2)=\{\{\varnothing\},\{1\},\{4\}\}$,$M(3)=\{\{\varnothing\},\{7\}\}$,$M(4)=\{\{\varnothing\},\{9\}\}$.第 2 步,计算 $M(1)\times M(2)\times M(3)\times M(4)$,再遍历笛卡尔积得到的集簇中的每一个元素(每一个元素对应一种 Q到 C 的单射),遍历时验证该单射是否符合匹配条件以及对符合匹配条件的每个单射(它们的集合就是根节点的 LM)计算相应的匹配代价(按定义 3).第 3 步,返回第 2 步中计算到的最小的匹配代价即为所求的查询树到该构件刻面描述树的匹配代价.

一般枚举方法的时间复杂度主要由第 2 步中的遍历操作的验证次数(n_c)以及对匹配代价计算次数(n_m)决定.对一般枚举方法有 $n_c=\prod_{v\in Q}|M(v)|-\sum_{v\in Q}|M(v)|+(|Q|-1)$ 成立.上例中的验证次数为$|M(1)|\cdot|M(2)|\cdot|M(3)|\cdot|M(4)|-(|M(1)|+|M(2)|+|M(3)|+|M(4)|)+(|Q|-1)=36-10+3=29$ 种.其中减去的 10,表示$\{\varnothing\},\{3\},\{6\},\{\varnothing\},\{1\},\{4\},\{\varnothing\},\{7\},\{\varnothing\},\{9\}$这 10 个与$\{\varnothing\}$笛卡尔积的结果,因为它们对匹配条件的符合是直接的,故不需要验证.加 3 是因为减 10 中有 3 个$\{\varnothing\}$重复减了.

从上面的分析知道,一般枚举的方法没有利用查询树的层次结构,使得枚举比较盲目,在下面提出的改进算法中,利用了查询树的层次结构,由各查询子树的匹配自底向上地逐步构造出最后整个查询树的匹配,这种枚举方法通过自底向上的逐层的过滤机制,可以较为有效地规避一些盲目的枚举.

改进枚举方法需要用到以下两个算子:

(1) $\otimes: LM\otimes LM\to LM$

$$LM(v_1)\otimes LM(v_2)=\{A\cup B\,|\,A\in LM(v_1),B\in LM(v_2),\text{并且}\forall\, a\in A,b\in B,a\text{和}b\text{之间没有祖先后代关系}\}.$$

(2) $\oplus: LM\oplus M\to LM$

$$LM(v_1)\oplus M(v_2)=\{A\cup B\,|\,A\in LM(v_1),B\in M(v_2),\text{并且}\forall a\in A,b\in B,b\text{是}a\text{的祖先节点}\}.$$

算子 $\otimes$ 的优先级高于算子 $\oplus$.这两个算子的算法实现都是先进行笛卡尔积,再通过验证剔除得到的笛卡尔积中不符合条件的元素.

改进后的枚举算法分为以下 3 个步骤:第 1 步,找到查询树中每个节点 v 的 $M(v)$.第 2 步,按后序顺序(这种顺序可以保证在遍历任一节点时该节点的儿子节点已被遍历)遍历查询树中的节点,在遍历节点 v 的时候计算

$LM(v)=M(v)\oplus(\underset{v_i\text{是}v\text{的儿子节点}}{\otimes}LM(v_i))$.就图中的例子来说,这一步的具体过程如下所示:

遍历节点 1 时:$M(1)=\{\{\varnothing\},\{3\},\{6\}\}$;$LM(1)=\{\{\varnothing\},\{3\},\{6\}\}$;叶子节点的 LM 等于 M;这一步 $n_c=0$.

遍历节点 2 时:$M(2)=\{\{\varnothing\},\{1\},\{4\}\}$;$LM(2)=\{\{\varnothing\},\{1\},\{4\}\}$;这一步 $n_c=0$.

遍历节点 3 时:$M(3)=\{\varnothing,7\}$; $LM(3)=M(3)\oplus LM(2)=\{\{\varnothing\},\{7\},\{1\},\{4\},\{7,4\}\}$.$\{7,1\}$被从 $M(3)\times LM(2)$中剔除,因为节点 7 不是 1 的祖先节点.这一步,$n_c=2\cdot 3-(2+3)+1=2$.

遍历节点 4 时:$M(4)=\{0,9\}$;先计算 $LM(3)\otimes LM(1)=LM(3,1)=\{\{\varnothing\},\{3\},\{6\},\{7\},\{1\},\{4\},\{7,4\},\{3,7\},\{3,4\},\{3,7,4\},\{6,1\},\{6,4\}\}$.这一步,$n_c=3\cdot 5-(3+5)+1=7$.再计算 $LM(4)=LM(3,1)\oplus M(4)=\{\{\varnothing\},\{9\},\{3\},\{6\},\{7\},\{1\},\{4\},\{7,4\},\{3,7\},\{3,4\},\{3,7,4\},\{6,1\},\{6,4\},\{9,3\},\{9,6\},\{9,7\},\{9,1\},\{9,4\},\{9,7,4\},\{9,3,7\},\{9,3,4\},\{9,3,7,4\},\{9,6,1\},\{9,6,4\}\}$.这一步,$n_c=2\cdot 12-(2+12)+1=11$.

第 3 步,依次计算查询树根节点的 LM 中每个单射的匹配代价,其中最小的即为所求的查询树到该构件刻面描述树的匹配代价.总结上例中计算过程的总的 $n_c=0+0+0+2+8+11=21$;$n_m=|LM(4)|=24$.该例中改进枚举算法的 n_c 比一般枚举算法的 n_c 减少了 27.6%.

2.2 算法分析

改进后的枚举算法的有效性从理论上可以分析如下:一般枚举方法可以看成是计算各个节点的 M 的笛卡尔积的过程.而改进后的枚举方法可以看成是各个节点的 M 的 $\otimes$ 或 $\oplus$ 算子的连接过程.例如上例中就是 $(M(1)\otimes(M(2)\oplus M(3)))\oplus M(4)$.由 $\otimes$ 和 $\oplus$ 算子的定义知道:$|LM\otimes LM|\le|LM\times LM|$ 和 $|LM\oplus M|\le|LM\times M|$ (因为 $LM\otimes LM\subseteq LM\times LM$ 和 $LM\oplus M\subseteq LM\times M$)所以在改进枚举中需要验证的情况只是一般枚举方法的子集.故改进后的枚举方法在时间上更有效.

改进后的枚举算法的有效性的实验分析如下:在我们设计的原型系统中,对 VC 运行库、MFC 库和 STL 中的 110 个构件(包括函数、类、模板)进行了实验,表 1 是两种方法的时间性能的一个简单比较,从中进一步说明了该匹配算法,尤其是改进后的算法在实践应用中的可用性与有效性.

通过实验还可知道,刻面的正交性越强,检索的效率越高,因为这时各个 M 中的元素个数越少.

Table 1 Runtime comparison among the two methods ($|Q|\le 10$, number of components=为 110,CPU=PⅢ667)

表 1 两种算法的时间性能比较(实验中$|Q|\le 10$,构件数为 110,CPU=PⅢ667)

Algorithms①	Mean time④(s)	Variance⑤(s^2)	Query times in experiment⑥
General enumerative algorithms②	0.406	0.012 14	68
Improved enumerative algorithms③	0.342	0.001 85	76

①方法,②一般的枚举算法,③改进的枚举算法,④平均时间,⑤方差,⑥实验的查询次数.

在该原型系统中,对构件的刻面描述树的构造是在构件入库时完成的,并将它存放进相应的数据库中,使得以后无须每次检索都重复构造,这样做使得给系统带来的代价最小,也不会给用户带来额外的负担.

3 总结和进一步的工作

本文针对基于刻面描述的构件,借鉴树匹配中的思想,结合构件刻面描述以及构件匹配的具体特征,提出了一种基于树包含的新的构件匹配方法,并给出了相应的算法.该方法可以在一定程度上解决异质刻面描述的构件间的匹配问题,并可以计算匹配代价以反馈给用户有益的复用建议.

本文提出的构件检索算法可以作为构件检索的有效方法之一.在实用时,尤其是在大规模的构件库的检索时,如果和其他构件检索方法结合使用,可以使它的算法搜索空间大大缩小,从而进一步提高检索的效率与质量(查准率与查全率).另外,对于树包含的算法目前还只考虑了枚举算法,为了进一步提高它的效率,我们正在实验模拟退火和遗传算法等方法来提高它的搜索效率.并且,为了更加有效地使用和推广该检索算法,还需要提供友好的查询树编辑界面及高效的检索平台,这些都是我们下一步工作的方向.

References:

[1] Podgurski, A., Pierce, L. Retrieving reusable software by sampling behavior. ACM Transactions on Software Engineering and Methodology, 1993,2(3):286~303.

[2] Zaremski, A.M. Signature and specification matching [Ph.D. Thesis]. School of Computer Science Carnegie Mellon University, 1996.

[3] Merkl, D., Tjoa, A.M., Kappel, G. Learning the semantic similarity of reusable software components. In: Frakes, W.B., ed. Proceedings of the 3rd International Conference on Software Reuse (ICSR'94). Rio de Janeiro: IEEE Computer Society Press, 1994. 33~41.

[4] Damiani, E., Fugini, M.G., Bellettini, C. A hierarchy-aware approach to faceted classification of objected-oriented components. ACM Transactions on Software Engineering and Methodology, 1999,8(3):215~262.

[5] Henninger, S. Supporting the process of satisfying information needs with reusable software libraries: an empirical study. In: Samadzadeh, M.H., Mansour, K.Z., eds. Proceedings of the 17th International Conference on Software Engineering on Symposium on Software Reusability. Seattle, WA: ACM Press, 1995. 267~270.

[6] Chang, Ji-chuan, Li,Ke-qin, Guo, Li-feng, *et al*. Representing and retrieving reusable software components in JB (JadeBird) system. Electronica Journal, 2000,28(8):20~24 (in Chinese).

[7] Sorumgard, L.S., Sindre, G., Stokke, F. Experiences from application of a faceted classification scheme. In: Proceedings of the 2nd International Conference on Software Reuse (REUSE'93). IEEE Computer Society Press, 1993. 24~26.

[8] Gibb, F., McCartan, C., O'Donnell, R., *et al*. The integration of information retrieval techniques within a software reuse environment. Journal of Information Science, 2000,26(4):520~539.

[9] Wang, J.T.L., Shapiro, B.A., Shasha, D., *et al*. An algorithm for finding the largest approxmately common substructures of two trees. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1998,20(8):889~895.

[10] Schlieder, T. ApproXQL: design and implementation of an approximate pattern matching language for XML. Technical Report, B 01-02, Freie Universität Berlin, 2001.

[11] Tai, Kuo-chung. The tree-tree correction problem. Journal of the Association for Computing Machinery, 1979,26,(3):422~433.

附中文参考文献:

[6] **常继传,李克勤,郭立峰,等.青鸟系统中可复用软件构件的表示与查询.电子学报**,2000,28(8):20~24.

Retrieving Components Based on Faceted Classification*

WANG Yuan-feng[1], ZHANG Yong[1], REN Hong-min[1], ZHU San-yuan[2], QIAN Le-qiu[1]

[1](*Software Engineering Laboratory, Department of Computer Science and Engineering, Fudan University, Shanghai* 200433, *China*);

[2](*Software Institute of Shanghai, Shanghai* 200233, *China*)

E-mail: yfwang@fudan.edu.cn

http://www.fudan.edu.cn

Abstract: With the deepening of reuse practice and scaling up of the component repository, the research of representing and retrieving software components gains more attention in software engineering research. A method based on tree inclusion is proposed to retrieve reusable components classified in faceted scheme, which combines the theory of tree matching and the feature of faceted classification scheme. The analysis and the experimental results demonstrate the feasibility and effectiveness of this method.

Key words: component repository; component retrieval; facet; trees matching; software reuse

* Received July 10, 2001; accepted February 25, 2002

Supported by the Development Foundation for Key Disciplines of Shanghai Education Commission of China under Grant No.B990105