# 一种基于 XML 的统一构件匹配技术

曾　一　刘元勇　郭永林
（重庆大学计算机学院　重庆 400044）

**摘　要**　构件检索和匹配是CBSD和软件复用的关键技术。目前构件库所采用的关键字、刻面、规约、行为等匹配技术都存在两个问题：1)构件表示方法各异，理解和共享构件困难；2)每个查询匹配算法侧重点不同，关键字和刻面匹配侧重构件的语义，而规约匹配侧重构件的结构，造成单独使用一种匹配算法或多或少产生一些冗余或无关构件。本文提出一个 XML 统一构件描述模型，该模型利用 XML 模式对构件进行层次化的组织，采用一种最大权匹配算法，从语法和语义两方面计算两个构件间的匹配度，有效地解决了上述问题。通过实验证明，该技术在保证构件查全率的基础上有效地提高了查准率。

**关键词**　软件复用，XML，构件检索，构件匹配

## A United Component Matching Technology Based on XML

ZENG Yi　LIU Yuan-Yong　GUO Yong-Lin
(College of Computer Science & Engineering, Chongqing University, Chongqing 400044)

**Abstract**　Component retrieving and matching are one of key technology of CBSD and software reuse. Component matching technology used now in component repository has two main problems. One is that different expressions of components leads to the difficult understanding and sharing of components, the other is that different arithmetic of matching emphasizes particularly on components which causes some redundancy and irrelated components. An uniform component description model which makes use of XML Schema to organize components hierarchically has been brought forward in the article and the author also introduces a maximal weight matching method for calculating matching degree of components in semantic and syntax. It is testified that it can not only help users to lookup the component, but also improve the precision of the searching.

**Keywords**　Software rqeuse, XML, Component retrieval, Component matching

基于构件的软件开发(CBSD)已经成为软件开发的主要模式之一。其中，作为核心和关键的构件检索匹配技术已成为一个主要的研究热点。目前，很多文献中提及的构件检索匹配技术有关键字匹配、刻面分类匹配、基调匹配、构件规约匹配等。关键字匹配是一种通过一组关键字进行简单匹配的方法，它已被证明缺乏准确性，容易产生大量的冗余构件。刻面分类检索方法是一种基于预定义分类特征的方法，它比单纯的关键字检索匹配能对构件提供一个更好的描述，但是用户必须熟悉分类模式以便有效地检索构件。而且，随着领域知识的不断扩充，它的分类模式难以管理。基调匹配和规约匹配都是基于语法结构的匹配方式，它们通过构件本身或接口的类型决定两个构件是否匹配，容易产生一些无关的构件。另外，上述几种匹配方法采用不同的构件表示模型，在异质构件库间共享构件困难。因此，本文借鉴 Naiyana Tansalarak 和 Kajal T. Claypool 提出的 XCM 模型[1]，提出了一种对构件进行逐层分类描述并涵盖构件交互信息，支持不同构件库间检索和匹配的统一描述模型，并阐述了一种适用于该模型的匹配方法。该方法针对构件不同的特征元采用不同的语义或语法匹配，通过匹配权重计算匹配度，从而定量分析出两个构件的匹配质量。最后通过实验证明该匹配技术在保证构件查全率的基础上有效地提高了构件的查准率。

## 1　基于 XML 的统一构件描述

构件本体是用于制定构件元数据的标准[1]。构件本体描述可以展开为一棵描述树，而 XML 语言本身的树结构特征及其良好的规范性，使得 XML 语言成为构件本体描述的上佳选择。在文[1]中，利用构件本体论提出了 XCM 模型。XCM 模型的主要思想是把一个构件分为 3 个主要的部分：基本信息、特征集和设计模式。基本信息指的是构件的名称、版本信息等，特征集是指属性、方法和事件的集合，而设计模式则描述了一个构件如何用一些预先存在的子构件来构造。最终形成的构件本体框架如图 1 所示。

为了提供对构件匹配质量进行定量的分析，本文采用术语匹配度 M 来描述。依据图 1 的结构，结合一定的算法计算出构件的各个部分特征元的匹配度，然后依据一定的算法得出整个构件的匹配度，从而提供了对构件匹配质量的定量分析策略。

## 2　构件匹配算法

通常将构件匹配分为两类：一类是构件间接口的匹配，另一类是问题域中的虚拟构件和构件库中的实际构件之间的匹配。本文所研究的匹配算法基于后者。

曾　一　副教授，硕士生导师，主要研究方向：软件过程、软件集成环境、软件开发环境、软件工程；**刘元勇**　硕士研究生，主要研究方向：软件工程、软件复用、构件技术；**郭永林**　硕士研究生，主要研究方向：软件工程、软件复用、构件技术。

### 2.1 XML基本元素匹配

XML模式用于匹配XML源文件中的一个元素，一个模式是一个字符串。最通常的模式规定了匹配元素的类型名称。例如，emph模式匹配类型为emph的元素。在本文所描述构件的XML文档结构中，将所有基本元素的模式分为两大类：一类为语法模式，主要表现在构件能够提供的操作、与系统的交互及参数类型等，简写为Syn，例如Type、Style、Status等；另一类为语义模式，主要表现文本或者用户查询在意义上的符合程度，简写为Sem，例如Label、Package、Location等。任何一个XML元素是包含模式和值的二元组〈Pattern，Value〉，则对于给定的两个元素 $E_1=\langle P_1,V_1\rangle$ 和 $E_2=\langle P_2,V_2\rangle$，给出定义：

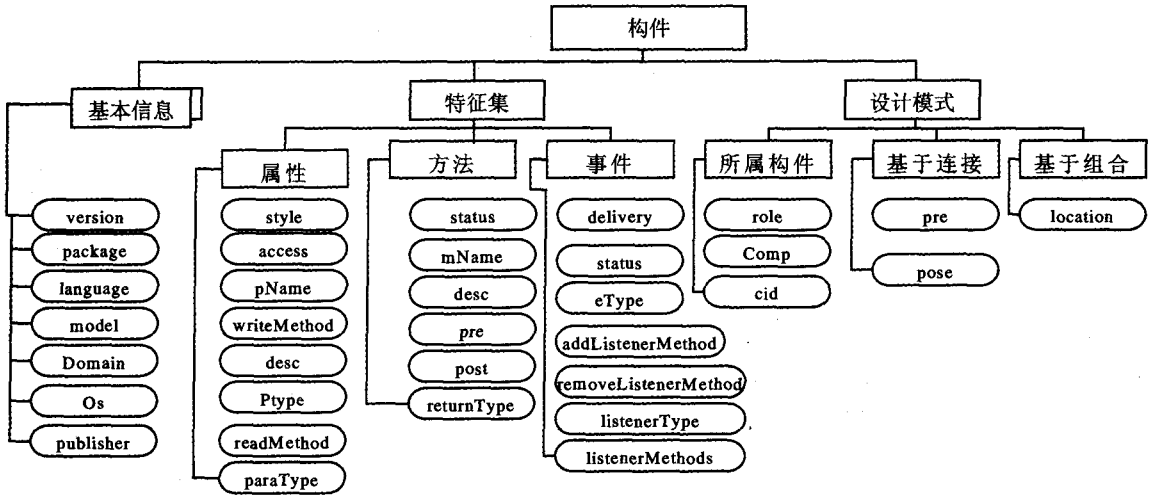


图1 构件XML树状结构图

**定义2.1** 当 $P_1$ 和 $P_2$ 同属于 $Syn$，若满足 $P_1=P_2$ 并且 $V_1=V_2$，则称 $E_1$ 和 $E_2$ 匹配，匹配度 $M_{syn}=1$，否则不匹配，$M_{syn}=0$；

**定义2.2** 当 $P_1$ 和 $P_2$ 同属于 $Sem$，若满足 $P_1=P_2$ 并且 $0<Sim(W_1,W_2)\leqslant 1$，则称 $E_1$ 和 $E_2$ 匹配，匹配度 $M_{sem}$ 为 $Sim(W_1,W_2)$。这里采用基于知网的词汇语义相似性来表示构件语义的匹配度[2]。基于知网的词汇语义相似度以汉语和英语的词语所代表的概念为描述对象，揭示概念与概念之间以及概念所具有的属性之间的关系并定义了 $Sim(W_1,W_2)$ 作为两个元素之间的相似性。

### 2.2 构件基本信息匹配

基本信息包括构件的名称、版本信息、所属包信息、语言、模式、域、操作系统和发布作者等。基本信息的匹配比较简单，所有的Schema都属于语义匹配范畴。

**定义2.3** 基本信息的匹配度

$$M_{info}=\sum_i M_{sem}(W_i,W_{pi})\times Q_i$$

其中 $W_i$ 表示目标构件基本信息的元素，$Q_i$ 表示该元素对应的权重，并满足 $\sum_i Q_i=1$，$W_{pi}$ 表示源构件基本信息与目标构件基本信息中相对应的元素。基本信息元素的权重应该根据构件应用中的领域特点，由用户动态分配。

### 2.3 特征集匹配算法

特征集是属性、方法和事件的集合。同时，属性、方法和事件又包括若干成员。对此，须由构件成员至构件成员集来定量分析构件的匹配质量。

#### 2.3.1 构件成员匹配

一个成员包含若干个基本元素。以属性为例，包括Name、Type、Access和Style等基本元素。若两个属性所有的基本元素都匹配成功，且匹配度为1，则两个属性为精确匹配；若两个属性只是其中部分元素匹配成功或匹配度大于0且小于1，则为近似匹配；否则为不匹配。

对于同一个成员的两个近似匹配来说，如何度量两个元素的匹配质量呢？例如：给定一个属性 $P_1$，

(1)存在属性 $P_2$，$M_{syn}(P_2,P_1)=1$、$M_{sem}(P_2,P_1)=0$；

(2)存在属性 $P_3$，$M_{syn}(P_3,P_1)=0$、$M_{sem}(P_3,P_1)=0.7$。

其中，$M_{syn}$ 表示基本元素的语法匹配度，$M_{sem}$ 表示基本元素的语义匹配度。

可以看出，在属性的语法方面，$P_2$ 比 $P_3$ 更符合匹配条件，但是属性的语义方面，$P_3$ 似乎比 $P_2$ 更加接近 $P_1$。这样，很难界定哪一个匹配更有效。

为了定量分析两个成员近似匹配的匹配质量，本文给出两个成员之间的匹配度 $M_{property}$ 计算公式：

$M_{property}=\prod_i M_{syn}p_i\times(\sum_{j=1}^{n}M_{sem}pj\div n)$，其中 $p_i$ 表示属性中的语法元素；$p_j$ 表示属性中的语义元素；$n$ 表示语义元素个数。表1给出了属性、方法和事件的语义和语法特征元素。

表1 属性、方法和事件的特征元素

| | 属性 | 方法 | 事件 |
|---|---|---|---|
| 语法元素 | pType | Status | Status |
| | Access | Pre | eType |
| | Style | Post | addListenerMethod |
| | ParaType | returnType | removeListenerMethod |
| | ReadMethod | | ListenerMethods |
| | WriteMethod | | ListenerType |
| 语义元素 | pName | desc | delivery |
| | Desc | mName | |

对复用者而言，构件的复用成本应该小于重新开发同等功能构件的代价。为此我们定义一个匹配阈值 $R_{comp}$（$0<R_{comp}<1$），用于区分一个构件成员是否为有效匹配的标准。当计算出的构件成员匹配度 $M>R_{comp}$ 时，我们认为该成员匹配有效。否则，认为该构件匹配不成功，可直接令 $M=0$。

#### 2.3.2 构件成员集的匹配

按照构件成员匹配理论，两个成员集之间存在一个多对多的匹配关系。以属性集为例，若给定两个属性集 $P_1\{p_{11}$，

$p_{12},\cdots,p_{1n}\}$和$P_2\{p_{21},p_{22},\cdots,p_{2m}\}$，对于$P_1$中的任何一个属性$p_{1i}$，在$P_2$中存在多个属性$p_{2j},p_{2k}\cdots$与之匹配，它们的匹配度分别为$M(p_{1i},p_{2j}),M(p_{1i},p_{2k}),\cdots$，对于$P_2$中的元素，同样存在类似的情况。另一方面，从构件匹配角度来说，要求两个构件的成员匹配是单射匹配，也就是$p_{1i}$若匹配$p_{2j}$，则不能再匹配给$P_2$中的其它属性。为此，我们引入二部图的最大权匹配理论来度量两个成员集的匹配质量，见图2、3。

在二部图$G(P,E)$中，顶点集$P$表示属性集，边集$E$中的边的权值表示成员间的匹配度$M_E$。其中，图3是图2的完全匹配，没有标识权值的边的匹配度为0。

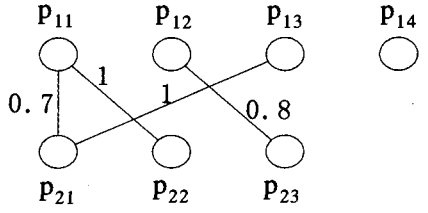


图2　构件属性集表示的二部图

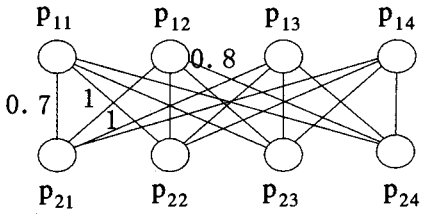


图3　构件属性集表示的完全二部图

按照二部图的理论，可以求出其最大权匹配，以此作为成员集间的匹配度，从而就可以求出其构件特征集的匹配度。

**定义2.4**　在二部图中，$\forall p\in P(G)$定义一个标记$l(p)\geqslant 0$满足$\forall p_{1i}\in P_1,\forall p_{2j}\in P_2$，恒有$l(p_{1i})+l(p_{2j})\geqslant w(p_{1i},p_{2j})$，则称$l(p)$是$G(P,E)$的可容顶点标。其中，$P_1$、$P_2$是构件的两个属性集，$w(p_{1i},p_{2j})$是特征元素$p_{1i},p_{2j}$的匹配度。

**定理1**　设$l(p)$是$G$的一个可容顶点标记，若$G$的等子图$G_l$有一个理想匹配$M$，则$M$必定是$G$的最大权匹配[3]。

由定理1，可以考虑使用Kuhn-munkras算法判断并求出成员集的理想匹配和最大权匹配。下面采用Kuhn-munkras算法来求成员集的最大权匹配。

Kuhn-munkras算法思路：开始时任选取一个可容顶点标记$l$，定出$G_l$用匈牙利算法求$G_l$的理想匹配，若在$G_l$中求出一个理想匹配，则它便是$G$的最大权匹配；否则，匈牙利算法得到的一个非理想匹配$M'$和一棵$M'$交错树$H$，$H$不含$M'$可增长路径。这时可以改进$l$成为一个新的可容顶点标记$\bar{l}$，使得在$G_l$中仍包含$M'$和$H$中的边，且$H$可在$G_l$中继续生长。这种改进顶点标记的做法可以继续下去，直到一个理想匹配在等子图中求到为止。

Kuhn-munkras算法求成员集最大权匹配的步骤：

$A_{G_l}(p_{1i})$：$G_l$中$p_{1i}$的邻接点集

$S$：$G_l$中$M$交错树里属于$P_1$的属性集

$T$：$G_l$中$M$交错树里属于$P_2$的属性集

$f(p_2)$：$p_2$的父亲

$N_{G_l}(S)$：$S$在$G_l$中的邻域

输入：完备二部图的边权矩阵$W=[w_{ij}]_{n\times n}$

(1)给出初始标号

$\forall i,l(p_{1i})\leftarrow\max(w_{ij}),\forall j,l(p_{2j})\leftarrow 0$

(2)求出邻点集

$\forall i,A_{G_l}(p_{1i})=\{p_{2j}\mid p_{2j}\in P_2,\ l(p_{1i})+l(p_{2j})=w_{ij}\}$

以及$G_l$中的一个匹配$M$。

(3)若$M$已经渗透$P_1$的所有顶点，则$M$即是$G$的最大权匹配，停止。否则，取$P_1$中的非渗透点$u$，$S\leftarrow\{u\}$，$T\leftarrow\Phi$。

(4)若$N_{G_l}(S)=T$，计算

$a_l=\min\{l(p_{1i})+l(p_{2j})-w_{ij}\}$

$$\bar{l}(p)=\begin{cases}l(p)-a_l, & p\in S\\ l(p)+a_l, & p\in T\\ l(p), & \text{其他}\end{cases}$$

$l\leftarrow\bar{l}$，$A_{G_l}(p_{1i})\leftarrow A_{G_{\bar{l}}}(p_{1i})$，转(5)。否则，即$N_{G_l}(S)\neq T$，转下一步。

(5)找到$p_1\in S,p_2\notin A_{G_l}(p_1)-T,f(p_2)\leftarrow p_1$。若$p_2$是$M$的渗透点，且$p_2z\in M$，则$S\leftarrow S\cup\{z\},T\leftarrow T\cup\{p_2\}$，转(4)。若$p_2$是$M$的非渗透点，则可按父子关系、匹配关系，从$p_2$回溯，直到起始点，从而得到一条$M$可增长路径$P(u,p_2)$，$M\leftarrow M\oplus E(P)$转(3)。

由以上算法可以计算出最大匹配权值，得出特征集的最大匹配度。

### 2.4　构件设计模式的匹配算法

根据前面介绍，基于XML的构件描述的匹配有两类设计模式：一种是基于连接的设计模式，另一种是基于组装的设计模式。可以将设计模式的匹配算法分为两个方面：一个是子构件匹配，一个是组合信息匹配。

根据子构件的XML描述信息，求得子构件所属路径。它的匹配度将转移为各个子构件的匹配度，一直递归到不含设计模式的简单构件为止，从而由前述构件的基本信息匹配度和特征集匹配度共同得出。

构件组合信息匹配则可分为语法匹配和构件间关系匹配，由二者的共同特征求出其设计模式的匹配度。

#### 2.4.1　语法匹配和构件间的关系匹配

语法匹配仍然按照前述语法匹配的方法求出其可否匹配。描述构件语法的元素见表2。

表2　设计模式中的语法元素

| | 基于连接 | | 基于组装 | |
|---|---|---|---|---|
| 语法元素 | Pre | True/False | Location | South/Center/Nouth/East/West/SouthEast/SouthWest/NouthEast/NouthWest; |
| | Post | True/False | | |
| | maxOccure | | Role | Master/Support/Client |
| | Role | Master/Support/Client | | |

构件间的关系通过方法来驱动，可以定义5种操作符(operator)来描述构件间的基本关系，分别为与操作、顺序操作、或操作、连接操作、循环操作[4]。在设计模式的匹配算法中判断构件间的关系是否一致再进行计算匹配度。

#### 2.4.2　设计模式的匹配算法

对于设计模式的匹配，从Role的Master出发，首先计算Master构件的匹配度，然后根据该构件与其他构件的关系以及Support和Client构件的前件和后件，循环计算各个支持构件的最大匹配度，最后计算出整个设计模式的匹配度。算法描述如下：

```
long MatchDesign(Comp comp1, Comp comp2){
  long M_design=0//定义设计模式的匹配度
  long M_mcomp=0 //定义核心构件的匹配度为0
```

```
long Mxcomp =0//定义支持构件和客户构件的匹配度为 0
long Mtemp =0;
comp curcomp//当前构件
if (compl. children. Role= =Master and comp2. children. Role==Master){//匹配核心构件
  Mmcomp =Mcomp (compl. children, comp2. children);//递归调用计算核心构件的匹配度
  Curcompl=compl. children;
  Curcomp2= comp2. children;
  If(Mmcomp >0){
    Do While(curcompl. next ! =null and curcomp2. next ! =null)
    {
      If(curcompl. operator = = curcomp2. operator and //构件间的组合关系是否匹配
        curcompl. Role = = curcomp2. Role and
        curcompl. Pre = = curcomp2. Pre and
        curcompl. Post = = curcomp2. Post)
        {
          Mtemp =Mcomp (curcompl, curcomp2); //递归调用计算支持构件的匹配度
          Mxcomp = Mxcomp +Mtemp ;
        }
        Curcompl=curcompl. next;
        Curcomp2=curcomp2. next;
    }
    Mxcomp =Mxcomp /compl. children. length();
  }
}
  Mdesign = Mmcomp * (Weight (Mmcomp )) + Mxcomp * (Weight (Mxcomp ));
  return Mdesign ;
}
long Mcomp (Comp compl, Comp comp2)
{
 long comp. Mfeature =MatchFeature(compl,comp2); //获取构件特征匹配度
 long comp. Minfo =MatchInfo(compl,comp2);//获取构件信息匹配度
 long comp. Mdesign =0
 if (compl. children. count()>=1)//该构件是一个复合构件
    long comp. Mdesign =MatchDesign(compl,comp2);//获取构件设计模式匹配度
 return comp. Mfeature ×Wfeature +comp. Mdesign ×Wdesign +comp. Minfo ×(1-Wfeature -Wdesign );
}
```

算法采用递归方法，符合构件的设计模式思路，由此可以较好地计算出设计模式的匹配度 $M_{design}$。

## 3 构件匹配度和相关实验数据

通过以上算法，得出构件信息、构件特征和设计模式的匹配度 $M_{info}$，$M_{feature}$，$M_{design}$，并由此计算出基于 XML 描述的构件匹配度，作为对构件匹配质量的有效评价。

### 3.1 匹配度 Mcomp

匹配度 *Mcomp* 定义了源构件到目标构件的整体匹配程度。根据用户对构件的不同用途要求，设计出由用户动态分配权重的构件匹配度量方法：

$$Mcomp=M_{feature}\times W_{feature}+M_{design}\times W_{sesign}\times W_{info}\times(1-W_{feature}-W_{design})$$

其中，$M_{feature}$ 表示特征集的匹配度，$W_{feature}$ 表示它对应的匹配权重；$M_{design}$ 表示设计模式的匹配度，$W_{design}$ 表示它对应的匹配权重；$M_{info}$ 表示基本信息的匹配度。

### 3.2 可适用度 Acomp 和冗余度 Dcomp

可适用度定量分析了目标构件中得到匹配的特征集占整个特征集的比重：

$$Acomp=\left(\frac{N(Method')}{N(Method)}+\frac{N(Property')}{N(Property)}+\frac{N(Event')}{N(Event)}\right)\div 3$$

其中 $N(Method)$ 表示目标构件中属性的个数，$N(Method')$ 表示匹配成功的个数。

冗余度 *Dcomp* 定量分析了源构件中未匹配的特征集占整个特征集的比重：

$$Dcomp=\left[\left(1-\frac{N(Method')}{N(Method'')}\right)+\left(1-\frac{N(Property')}{N(Property'')}\right)+\left(1-\frac{N(Event')}{N(Event'')}\right)\right]\div 3$$

其中 $N(Method'')$ 表示源构件中属性成员的个数。

### 3.3 实验数据

在实验过程中，使用了一个 SliderFieldPanel 构件，它是一个自定义的组件，由 Javax. swing. JSlide、Javax. swing. Jpanel、Javax. swing. TextField 和 Javax. swing. box 等 4 个构件组合而成。同时，由 XCM 表示模型而计算出了几种构件对于 SliderFieldPanel 的匹配度，见表 3 和图 4。

表 3 构件的各个特征元素的匹配度

| Facet \ Comp | JPanel | JTextField | JSlider | FieldPanel |
|---|---|---|---|---|
| genInfo | 0.73 | 0.71 | 0.75 | 0.98 |
| Feature | 0.288 | 0.307 | 0.189 | 0.779 |
| Design | 0 | 0 | 0 | 0.775 |

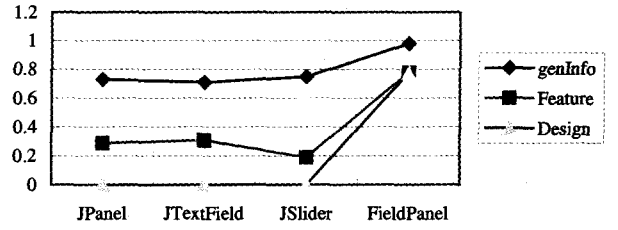


图 4 SliderFieldPanel 构件的匹配计算结果

由图 4 可以看出其在保持查全率的基础上，很好地提高了对构件的查准率，提供了良好的量化机制。

**结论** 在文[5]中，提出了 J2EE 平台下 Java 构件库系统 HSCL(HS-Sysway Component Library)系统。HSCL 是结合语义分类检索和语法匹配的经验模型，按照 MVC 结合构件粒度分类，有效地解决了构件理解的困难，但对于量化构件的匹配质量没有有效的策略。本文通过对构件本体描述的剖析，提出并详细分析了一种只需提供源码接口便可自动描述的构件模型和基于该模型的匹配算法。该方法与以往检索方法相比，不仅规范了不同构件模型描述上的差异，提高了异构构件库的互访性，同时减少了描述成本，还在基于查全率的基础上有效提高了构件的查准率。

## 参 考 文 献


1 Tansalarak N, Claypool K T. XCM: A Component Ontology. In: ACM Conference on Object-Oriented Programming, Systems, Languages and Applications. Vancouver, British Columbia, Canada, 2004. 10

2 刘群，李素建．基于《知网》的词汇语义相似度计算．见：第三届汉语词汇语义学研究会．中国：台北，2002

3 Bondy J A, Murty U S R. Graph theory with Applications. The Macmillan press LTD, 1976

4 Tansalarak N, Claypool K T. XCompose: An XML-Based Component Composition Framework. In: Third International Workshop on Compositional Languages in Conjunction with 17th European Conference on Object-Oriented Programming (ECOOP). Darmstadt, Germany, 2003

5 曾一，郭永林，等．基于 J2EE 平台的 Java 构件库的研究和实现．计算机科学，2006(5)