doi: 10.12068/j.issn.1005-3026.2020.01.005

基于语义和结构的UML类图的检索

袁中臣[1,2], 马宗民[1]
(1. 东北大学 软件学院, 辽宁 沈阳 110169; 2. 沈阳工业大学 化工过程自动化学院, 辽宁 辽阳 111003)

摘 要: 以软件重用为背景提出基于语义和结构的UML类图检索. 构建了UML类图的重用模型,定义了存储UML类图的重用库结构. 提出将本体的概念语义距离应用到UML类图的语义相似性度量和使用图表示UML类图的结构进行结构相似性度量. 基于检索流程形式化检索需求,提出了UML类图的检索算法. 基于提出的衡量标准,从语义、结构和混合三种检索类型对提出的算法进行了验证. 实验结果表明,所提出的检索算法在检索质量和检索效率上要优于其他方法.

关 键 词: UML类图;语义;结构;相似性度量;检索
中图分类号: TP 311 **文献标志码**: A **文章编号**: 1005-3026(2020)01-0023-06

Retrieval of UML Class Diagrams Based on Semantics and Structure

YUAN Zhong-chen[1,2], *MA Zong-min*[1]
(1. School of Software, Northeastern University, Shenyang 110169, China; 2. School of Chemical Process Automation, Shenyang University of Technology, Liaoyang 111003, China. Corresponding author: YUAN Zhong-chen, E-mail: yuanzhongchen@163.com)

Abstract: The UML class diagram retrieval based on semantics and structure was proposed under the background of software reuse. The reuse model of UML class diagram was constructed and the reuse repository structure of UML class diagram was defined. The conceptual semantic distance of ontology was applied to the semantic similarity measure of UML class diagram and a graph was used to represent the structure of UML class diagram for the structural similarity measure. The retrieval requirements were formalized based on the retrieval procedure and the retrieval algorithm of UML class diagram was proposed. The proposed algorithm was validated from three retrieval types(semantics, structure and hybrids) according to the proposed criteria. The experimental results showed that the proposed algorithm is superior to other methods in terms of both retrieval quality and efficiency.

Key words: UML class diagram; semantics; structure; similarity measure; retrieval

随着软件复杂性的日益增强,对于软件重用的需求在不断增加. 重用的内容也不再仅限于代码,而是涉及到软件生命周期的各个阶段[1]. 实际上,软件开发过程中产生的任意产品都在重用范围之内,这种重用被视为软件工程师的知识和经验的重用. 因为软件设计对接下来的开发阶段将产生重要的影响,所以软件设计的重用受到了关注. 作为建模工具,UML类图被广泛地应用于设计阶段,已成为软件设计事实上的标准[2],所以UML类图的重用成为研究热点[3-4].

随着语义web的发展,大量的本体被开发. 本体是共享概念的明确而详细的说明,概念通过特定的关系建立联系[5]. 本体分为通用本体和领域本体. 通用本体覆盖了若干领域的概念知识,如WordNet;领域本体是由来自单个领域的概念构成的,如基因组学领域的基因本体(gene ontology,GO)等. 本体为概念之间的相似性度量提供了途径. 例如,在WordNet中提供了基于路

收稿日期: 2018-12-20
基金项目: 国家自然科学基金资助项目(61370075,61772269).
作者简介: 袁中臣(1978-),男,辽宁辽阳人,东北大学博士研究生,沈阳工业大学讲师; 马宗民(1965-),男,山东金乡人,东北大学教授,博士生导师.

径和信息共享两种方法来衡量概念之间的相似性[6]. 相似性度量被广泛应用于分类、聚类和检索等. 在软件产品的重用过程中,检索是核心工作,检索是基于相似性. UML 类图所呈现的信息包含语义和结构,类、属性和操作属于语义范畴;UML 类图的结构通过类之间的关系来表达,类之间的关系包含不同的类型(关联、泛化、聚合、组合、依赖和实现). 所以,UML 类图的相似性度量既包含语义相似性也包括结构相似性.

现有的基于重用的类图检索可以归结为两种策略. 第一种是基于语义的匹配[7],类图之间的相似性是通过语义相似性来衡量. 其中,类图的关系被定义为向量 $\boldsymbol{R}$ = [端点类 1,关系类型,端点类 2],关系的相似性通过向量 $\boldsymbol{R}$ 的差异或距离来度量. 第二种是基于模型查询语言[8],解析类图元素并基于模型查询语言重写类图,并提出模式匹配方法.

第一种策略仅仅考虑了语义,而没有考虑结构. 即使考虑了概念之间关系,本质上也是语义匹配,在仅考虑类图的结构而不考虑语义的检索要求时,这种策略不能很好地工作. 实际上,结构对于一个 UML 类图是非常重要的. 在使用 UML 模型化一个系统时,首要考虑的是系统的结构而不是单个类. 第二种策略虽然考虑了结构,但元素匹配是基于关键字而不是基于语义,没有考虑检索结果的非精确性. 最后,两种方法都没有对检索条件给出一种形式化的定义. 实际上,检索需求经常是复杂的,通常不仅仅是一个简单的类或类图,这就需要一个检索条件的明确表达. 同时,每个检索元素在检索条件中的权重应该被考虑,因为类图中的不同元素在模型化一个系统时的重要程度不同.

本文正是基于语义和结构两个方面实现对类图的检索,定义了语义相似性和结构相似性度量方法. 基于检索流程,形式化地定义了检索表达,并提出了检索算法. 实验验证了所提算法的有效性.

1 UML 类图及其重用模型

1.1 UML 类图

UML 类图是由类和类之间的关系两部分构成. 图 1 是类图的一个样例,表示类“Professor”继承类“Teacher”.

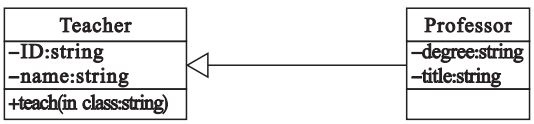

图 1 UML 类图样例

Fig. 1 A sample of UML class diagram

一系列工具能被用来编辑 UML 类图,如 Rose 和 MyEclipse 等. 对象管理组织(object management group,OMG)为所有 UML 模型定义了文档类型定义(document type definition,DTD)标准[9],UML 模型被描述为遵守 DTD 的基于 XML 元数据交换(XML-based metadata interchange,XMI)文档.

1.2 重用模型

UML 类图的重用模型被描述为一个过程,如图 2 所示. 这一过程包含 4 个阶段,分别是类图的检索、候选类图返回、候选类图调整并应用到新项目和新类图被抽取并加入到重用库. 在这几个阶段中,检索是关键,决定着重用质量和效率. 重用库存储大量的可重用类图,重用库的有效管理和组织对于提高检索效率也是非常重要的. 重用库中的类图被分为“域”和“目录”两级,如图 3 所示.

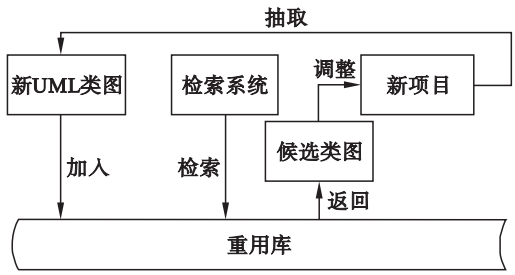

图 2 UML 类图重用模型

Fig. 2 Reuse model of UML class diagrams

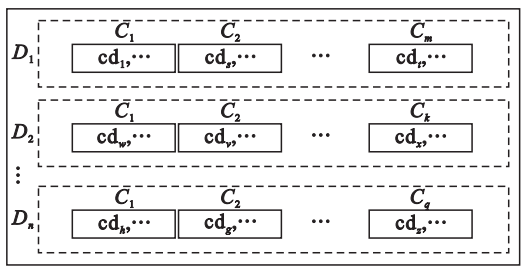

图 3 重用库结构

Fig. 3 Structure of reuse repository

建模不同领域项目的 UML 类图 cd_i 被分类到不同的域,如 $D_1, D_2, \cdots, D_n$. 在一个域内,类图又被划分为多个结构目录,如在 D_1 中的 $C_1, C_2, \cdots, C_m$. 结构相同或者相近的类图被分类到相同的目录. 在每个域内定义一个特征概念集合,在每个目录中定义一个特征结构集合. 特征概念和特征结构的选取是基于其所在域和目录的重用次数和出现频率. $f_c(D_i)$ 和 $f_s(C_j^i)$ 分别标记域 D_i 的特征概念集合和域 D_i 中结构目录 C_j 的特征结构集合.

$$f_c(D_i) = \{C_1^i, C_2^i, \cdots, C_p^i\},$$

$$f_s(C_j^i) = \{S_1^{ij}, S_2^{ij}, \cdots, S_q^{ij}\}.$$

2　相似性度量

2.1　语义相似性

现有的语义相似性能被归结为同一类，就是基于本体来计算两个概念的语义相似性. 本文选择被广泛使用的基于本体路径语义距离来计算 UML 类图概念之间的语义相似性. 在本文中，除了类，属性和方法也被认为在语义范畴内，所以类图之间的语义相似性为

$$\mathrm{SimCS}(\mathrm{cd}_1,\mathrm{cd}_2)=\frac{\sum \alpha\mathrm{Sim}C(C_i,C_j)+\beta\mathrm{Sim}A(A_i,A_j)+\gamma\mathrm{Sim}O(O_i,O_j)}{\min(|\mathrm{cd}_1|,|\mathrm{cd}_2|)}.$$

式中：cd_1 和 cd_2 是两个类图；类 C_i 来自类图 cd_1，A_i 和 O_i 分别是类 C_i 的属性集合和操作集合；同理，类 C_j 来自类图 cd_2；A_j 和 O_j 分别是类 C_j 的属性集合和操作集合；C_i 被匹配到 C_j，A_i 被匹配到 A_j，Q_i 被匹配到 Q_j；Sim$C$，Sim$A$ 和 Sim$O$ 分别表示类相似性、属性相似性和操作相似性. 在计算属性和操作相似性时，除了考虑概念语义还考虑到属性的类型和操作的返回值类型. 类图之间的语义相似性实际上就是类图概念元素之间的相似性. 这里仅计算平均相似性，$|\mathrm{cd}_i|$ 表示类图 cd_i 中包含的类的数目；α,β 和 γ 是权重因子，且 $\alpha+\beta+\gamma=1$.

2.2　结构相似性

由于 UML 类图的半形式化，图和 UML 类图两种模型在结构上的相似性（类对应于顶点，关系对应于边），本文提出采用图来表示 UML 类图的结构，目的是进行结构相似性度量. UML 类图中的类被转换成图的顶点，UML 类图中的关联、泛化、聚合、组合、依赖和实现等关系被转换成边，并分别被标记为 e_1,e_2,e_3,e_4,e_5 和 e_6. 这样类图之间的结构相似性就被转换成对应的两个图的相似性. 两个图之间的结构相似性可以通过最大公共子图（maximum common subgraph，MCS）度量[10]. 公式为

$$\mathrm{SimSS}(g_1,g_2)=\frac{N_{\mathrm{MCS}}}{\min(|g_1|,|g_2|)}.$$

其中：N_{MCS} 表示存在于 MCS 中的边的数目；$|g_i|$ 是表示图 g_i 中边的数目.

当然，这里求解最大公共子图是基于边，而不是顶点，并且边的匹配是基于上面所述的标记. 求解最大公共子图的算法见算法 1.

算法 1　搜索图 g_1 和 g_2 之间的最大公共子图

输入：图 g_1 和 g_2

输出：最大公共子图（MCS）

```
1. 初始化状态空间 S = NULL;
2. //在 g1 中与 S 相连的下一条边 ek
   while(nextEdge(g1,S,ek)) do
3. begin
4. //ek 是否使得 g1 和 g2 的公共子图尺寸增加
   if(IsFeasible(g1,g2,S,ek)) then
5.    begin //更新公共子图
6.       S = S + ek;//更新状态
7.       if(size(S) > z) then
8.       z = size(S); //更新尺寸
9.    end;
10.   else //退回 S 的上一个状态
11.   backState(S);
12. end;
13. return S;
```

算法执行了一个深度优先搜索. S 是一个状态空间，用于存储图 g_1 和 g_2 之间的公共子图. S 的尺寸会随着更多边的加入而变大直至最后形成最大公共子图.

3　类图检索

3.1　检索流程

UML 类图的检索过程描述如图 4 所示. 用户首先输入检索需求，用户的输入可以是单个类（属性和操作），也可以是类图结构. 解析是从检索要求获取检索元素，任何基于 SAX（simple API for XML）的工具能被用来解析 XMI 文档[11]. 检索需求经常是复杂的，包含着用户的多种意愿，所以将检索需求转换成一个形式化表述，作为检索算法的输入. 检索算法是核心，这里提到的检索应该是非精确的，目的是发现最贴近检索需求的候选类图序列.

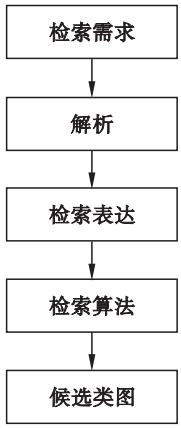

图 4　UML 类图检索流程

Fig. 4　Retrieval procedure of UML class diagrams

3.2 检索表达

在输入检索需求时,经常会有如下几种情况:

1) 希望几个检索元素在候选类图中同时出现. 可以被表达为

$$e_1 \wedge e_2 \wedge e_3 \wedge \cdots \wedge e_m.$$

2) 希望在候选类图中某些元素部分出现或者同时出现. 可以被表述为

$$e_1 \vee e_2 \vee e_3 \vee \cdots \vee e_p.$$

3) 希望有些检索元素在候选类图中不要出现. 可以被表述为

$$!(e_1 \vee e_2 \vee e_3 \vee \cdots \vee e_n).$$

所以,综合上述几种情况,检索需求能被形式化地描述如下:

UCD_QS =

$$(e_1 \wedge e_2 \wedge \cdots \wedge e_{i-1} \wedge (e_i \vee e_i') \wedge (e_{i+1} \vee e_{i+1}') \wedge \cdots \wedge (e_{i+k} \vee e_{i+k}')) \wedge (s_1 \wedge s_2 \wedge \cdots \wedge s_{j-1} \wedge (s_j \vee s_j') \wedge (s_{j+1} \vee s_{j+1}') \wedge \cdots \wedge (s_{j+k} \vee s_{j+k}')) \wedge (!(e''_1 \vee e_2'' \vee \cdots \vee e_p'' \vee s_1'' \vee s_2'' \vee \cdots \vee s_q'')).$$

其中:e_i 和 e_i' 为检索条件中的概念元素,它是类 C_i,也包含类的属性 A_{ij} 或者操作 O_{ij};s_i 和 s_i' 为结构元素;e_p'' 为不能出现在候选类图中的概念元素;s_q'' 为不能出现的结构元素.

3.3 检索算法

一般情况下,在类图的检索中,候选类图不一定完全满足检索需求. 这里设定一个阈值 σ,当检索元素被匹配到重用库中的类图所取得的相似性值大于 σ 时,重用库中的类图即被列为候选类图. 因为不同的元素(概念和结构)在模型化一个系统时具有不同的重要程度,所以不同的检索元素在检索条件中被分别赋予权重,$\boldsymbol{W} = [w_1, w_2, \cdots, w_n]$. 类图的检索算法被描述为算法 2.

算法 2 类图的检索算法

输入:检索表达 UCD_QS,$\boldsymbol{W}$ 和重用库 $\boldsymbol{L}$

输出:候选类图序列

1. //初始化目标类图所在域、目录和候选类图序列
   d = NULL; c = NULL; l = {NULL};
2. //获取检索元素
   classElements = getClass(UCD_QS); //概念元素
   noClassElements = getNoClass(UCD_QS);
   strucElements = getStructures(UCD_QS);//结构元素
   noStrucElements = getNoStructures(UCD_QS);
3. //赋予权重
   assignWeight (classElements, strucElements, W);
4. //候选类图所在的域
   if(classElements! = NULL) then
5. for each D_i in $\boldsymbol{L}$ do
6. begin
7. FC_i = getFeatClass(D_i); //获取特征概念
8. $SimD_i$ = SimCS(classElements, FC_i); //域相似性
9. end;
10. $SimD_x$ = max(Sim$D$);
11. $d = D_x$;
12. //候选类图所在的目录
    if(strucElements! = NULL) then
13. for each C_i in d do
14. begin
15. FS_i = getFeatStrucs(C_i); //获取特征结构
16. $SimC_i$ = SimSS(strucElements, FS_i); //目录相似性
17. end;
18. $SimC_y$ = max(Sim$C$);
19. $c = C_y$;
20. //计算类图相似性,并插入候选序列
    for each cd_k in c do
21. if(μ * SimCS(classElements, cd_k) + (1 − μ) * SimSS(strucElements, cd_k) > σ) then
22. insertCD(cd_k, l);
23. //删除不期望包含元素的类图
    for each e_i from noClassElements and each s_i from noStrucElements do
24. for each cd_q in l do
25. if(e_i or s_i) in cd_q then
26. deleteCD(cd_q, l);
27. return l; //返回候选类图序列

在检索算法中,基于定义的特征概念和特征结构,通过相似性计算分别决定候选类图所在的域和目录,这样能过滤掉大量的类图,大大提高检索的效率. 检索元素被匹配到已经确定目录内的每个目标类图并计算相似性,当相似性的值高于指定阈值 σ,目标类图被插入候选序列. 最后,从候选序列中删除不期望的元素所在类图. μ 为相似性权重,在应用中可以根据检索需求进行调整.

4 实 验

使用 Java 语言实现了本文所提出的检索算法,并且在 PC 机(Win 10, Intel Core i7, RAM

8GB)上执行. 实验中使用的类图来自 3 个领域，分别为教育、医疗和房地产，每个类图中类的数目在 15 ~ 25 之间. 所有类图均来自公司已经开发的项目，每个领域的类图、目录、特征概念以及每个目录包含的特征结构数量如表 1 所示.

表 1　重用库中 UML 类图

Table 1　UML class diagrams in reuse repository

领域	类图	目录	特征概念	特征结构
教育	100	4	15	4,5,5,4
医疗	100	5	18	4,4,4,3,3
房地产	100	4	16	5,4,3,4

本文做三种检索类型(概念、结构和混合)操作，如表 2 所示. 每种检索类型操作基于不同领域元素被执行 3 次，检索条件的设定主要从检索元素数量和分布考虑.

表 2　检索条件设定

Table 2　Retrieval conditions setting

检索类型	概念	结构
Q_1	8(3)	0
Q_2	0	5(2)
Q_3	6(2)	3(1)

注：括号中数字表示不允许出现在候选列表中的概念或结构的数量.

提出的检索算法和其他方法(语义检索[6]和模型语言检索[8])进行比较，本文提出的检索方法被称为二级检索，主要从检索质量和检索效率两个方面进行比较. 参数设置 $\alpha = 0.5$, $\beta = 0.25$, $\gamma = 0.25$, $\sigma = 0.5$. 在三种检索方法中，相似性计算权重 μ 分别被设定为 1.0,0 和 0.5. 检索元素权重设定为平均权重.

衡量检索质量可以从查准率 P、查全率 R 和二者调和平均值 F 三个指标来度量[12]. 这里设定 A 表示正确地出现在查询结果中候选类图的数量；B 表示错误地出现在查询结果中候选类图的数量；C 表示应该出现但没有出现在检索结果中的类图数量. P,R 和 F 计算公式为

$$P = \frac{A}{A + B},$$

$$R = \frac{A}{A + C},$$

$$F = \frac{2PR}{P + R}.$$

平均查准率 P、查全率 R 和调和平均值 F 的对比分别如图 5，图 6 和图 7 所示. 可以看出，三种检索方法查准率差别不大，如图 5 所示. 但查全率出现较大差别，特别是结构和混合检索，如图 6 所示. 综合来看，对于概念检索(Q_1)，三种检索方法呈现的结果差别不大；而对于结构检索(Q_2)，本文提出算法和模型语言检索得出的结果要明显优于语义检索；对于混合检索(Q_3)，本文提出的算法的检索结果要优于其他两种方法，如图 7 所示. 原因是本文的检索算法同时考虑了语义和结构，而其他两种方法存在片面性.

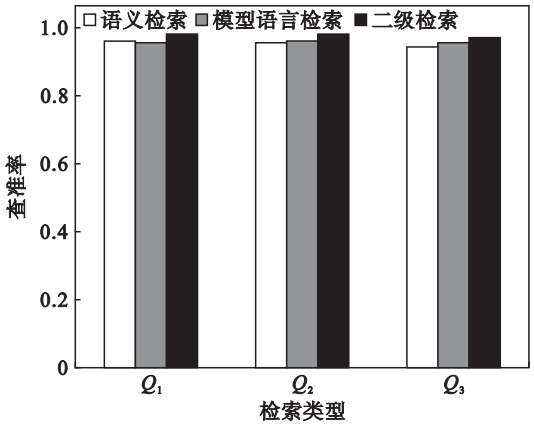

图 5　查准率比较

Fig. 5　Comparison of precision ratios

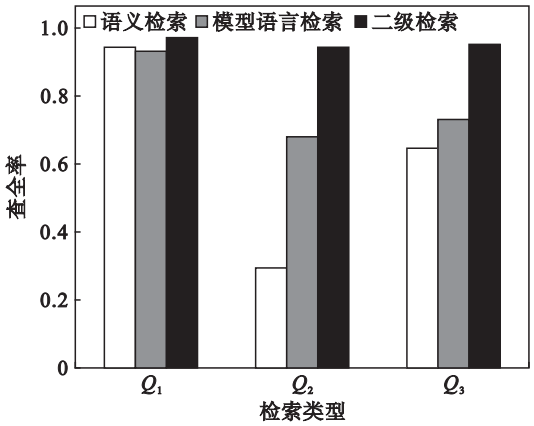

图 6　查全率比较

Fig. 6　Comparison of recalls

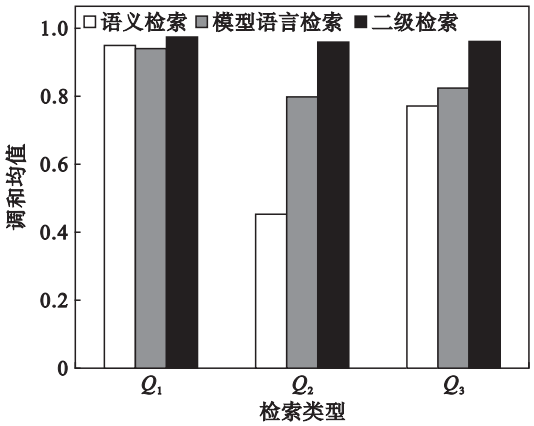

图 7　调和均值比较

Fig. 7　Comparison of harmonic means

检索效率是指执行检索的响应时间. 三种检索方法分别在不同检索类型数据上执行检索所需的响应时间如图 8 所示.

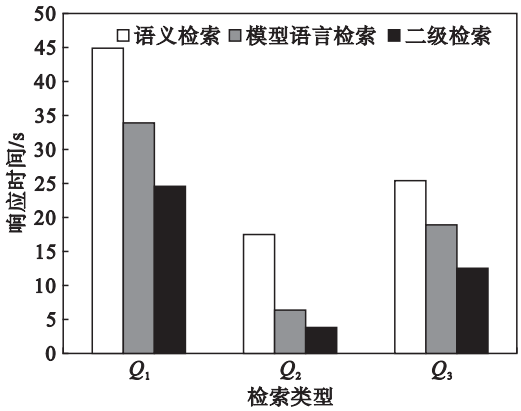

图 8 检索时间比较

Fig. 8 Comparison of retrieval time

可见,对于每种检索类型,本文算法的检索响应时间要少于其他两种方法. 类图检索的响应时间主要取决检索元素和重用库中目标类图匹配的次数. 本文提出的算法的匹配次数决定于重用库中域的数、候选类图所在域包含的结构目录数以及候选类图所在结构目录包含的类图数,所以匹配次数要远远少于其他两种方法. 关于重用库中类图分类的这部分内容将会在接下来的工作中讨论.

所以,无论从检索质量还是从检索效率上,本文提出的算法都是可行的并且要优于其他两种方法. 由于特征概念和特征结构的引入,算法对数据量越大的存储库检索表现越好.

5 结 语

本文基于软件重用从语义和结构两个方面提出对 UML 类图的检索. 通过本体和图的引入,分别提出了类图之间语义相似性和结构相似性度量方法. 基于检索流程,提出了类图的检索算法. 实验结果表明,本文提出的算法在语义、结构和混合三种检索类型都获得较好的检索质量,特别是对结构和混合检索,与其他检索方法相比具有明显的优势. 同时,本文提出的算法的检索效率也是可以接受的,并且优于其他两种方法.

参考文献:

[1] Frakes W B, Kang K. Software reuse research: status and future[J]. *IEEE Transactions on Software Engineering*, 2005, 31(7): 529 – 536.

[2] Medvidovic N, Rosenblum D S, Redmiles D F, et al. Modeling software architectures in the unified modeling language[J]. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 2002, 11(1): 2 – 57.

[3] Adamu A, Zainon W M N W. A review of UML model retrieval approaches[J]. *Indian Journal of Science and Technology*, 2016, 9(46): 1 – 8.

[4] Salami H O, Ahmed M A. UML artifacts reuse: state of the art[J]. *The International Journal of Soft Computing and Software Engineering (JSCSE)*, 2013, 3(3): 115 – 122.

[5] Antoniou G, Van Harmelen F. Web ontology language: OWL[M]//Handbook on ontologies. Berlin: Springer, 2004: 67 – 92.

[6] Meng L, Huang R, Gu J. A review of semantic similarity measures in wordnet[J]. *International Journal of Hybrid Information Technology*, 2013, 6(1): 1 – 12.

[7] Robles K, Fraga A, Morato J, et al. Towards an ontology-based retrieval of UML class diagrams[J]. *Information and Software Technology*, 2012, 54(1): 72 – 86.

[8] Zhang X, Chen H, Zhang T. An UML model query method based on structure pattern matching[C]//International Conference on Trustworthy Computing and Services. Berlin: Springer, 2012: 506 – 513.

[9] Routledge N, Bird L, Goodchild A. UML and XML schema[C]//Australian Computer Science Communications. Melbourne: Australian Computer Society Inc, 2002: 157 – 166.

[10] Raymond J W, Gardiner E J, Willett P. Rascal: calculation of graph similarity using maximum common edge subgraphs[J]. *The Computer Journal*, 2002, 45(6): 631 – 644.

[11] Grose T J, Doney G C, Brodsky S A. Mastering XMI: Java programming with XMI, XML and UML[M]. New York: John Wiley & Sons, 2002.

[12] Yang Y. An evaluation of statistical approaches to text categorization[J]. *Information Retrieval*, 1999, 1(1/2): 69 – 90.