A Multiple Viewed Interrelated Ontology Model for Holistic Component Specification and Retrieval

Chengpu Li, Xiaodong Liu, and Jessie Kennedy

School of Computing, Edinburgh Napier University, Edinburgh, UK
{c.li,x.liu,j.kennedy}@napier.ac.uk

Abstract. Despite the success that Component-Based Development has achieved so far, component mismatch remains as a major hurdle for wider and smoother component reuse due to the lack of effective and automated approaches to component specification and retrieval. This paper presents a novel ontology-based approach to solve the above problem via holistic, semantic-based and adaptation-aware component specification and retrieval. The Multiple-Viewed and Interrelated Component Specification ontology model (MVICS) provides an ontology based architecture to specify components in a spectrum of perspectives. A semantic-based component retrieval method is then developed and the result of retrieval is presented to CBD engineers in a comprehensive component matching profile. Uniquely, the effect of possible component adaptation is included in the MVICS model and associated component specification and retrieval, which enables a more systematic and holistic view in component specification and selection.

Keywords: component repository, component retrieval, ontology-based component specification, component reuse, adaptation assets, result profile.

1 Introduction

Component-Based Development (CBD) is an approach to developing a software system by assembling and composing already built software components. Numerous advantages of CBD have been identified [3][7][12]. However, at present CBD still fails to reach its full potential due to a few unsolved major hurdles, one of which is the lack of effective and automated methods for holistically and semantically specifying and retrieving components that precisely match users' requirements [8].

The above problem is basically caused by the lack of competent semantic-based component specification/repository and retrieval technologies. Existing approaches failed to specify components at a systematic and complete spectrum of perspectives and utilize such specification in retrieval. Although a few approaches started to use domain model and ontology in component retrieval process, to date it is clear that the ontology in these approaches has too simple and monolithic structure and few relationships to deal with the specification and retrieval of modern components [14]15][15]. Moreover and as part of the consequence, these approaches also failed to rank the found components with accurate relevance rating and clear unsatisfied discrepancy to reuse requirements, all of which provide critical guidelines for user's

D. Ślęzak et al. (Eds.): ASEA 2009, CCIS 59, pp. 50–60, 2009.

decision on component selection and the subsequent component adaptation and integration.

In this paper, a novel ontology-based approach is proposed to achieve holistic and semantic-based component specification and then automatic and precise component retrieval. As a foundation of the approach, a Multiple-Viewed and Interrelated Component Specification ontology model (MVICS) for component specification and repository is first developed. The MVICS model provides an ontology based architecture to specify components in a spectrum of perspectives, it accommodates domain knowledge of CBSE and application domains, and supports ontology evolution to reflect the continuous developments in CBD and components. A semantic-based component retrieval method is then developed based on MVICS model. The results of retrieval include not only the matching components but also accurate relevance rating and unsatisfied discrepancy, which are presented to CBD engineers in a comprehensive component matching profile. Another unique feature of the proposed approach is that the effect of possible component adaptation is included in the MVICS model and associated component specification and retrieval, which enables a more systematic and holistic view in component specification and selection. A prototype tool with an example component repository is built to verify and automate the approach. Extensive user feedbacks have been received based on case studies, which show the approach and tool is effective for the problem.

The reminder of the paper is organized as follows: Section 2 discusses related work with critical analysis. Section 3 introduces the Multiple-Viewed and Interrelated Component Specification ontology model. Section 4 describes the MVICS based holistic component retrieval. Section 5 describes the resultant prototype tool and a case study. Section 6 discusses the results of an initial evaluation of the system from practical use. Finally, section 7 presents the conclusion and future work.

2 Related Work

Existing component description and retrieval approaches can be classified into two types: traditional and ontology-based. The traditional approaches include keyword searching [8], faceted classification [10][13], signature matching [17] and behavioral matching [5]18]. Traditional approaches are not efficient, and suffer from lower recall and precision. Recall is a measure of the completeness of components matching, which can be defined as the proportion of the number of relevant found components to the number of all relevant components in the repository. Precision is a measure of the accuracy of component matching, which can be defined as the ratio of the number of retrieved relevant components to the number of the all retrieved components [13]. The traditional approaches are rather limited in accommodating semantics of user queries and domain knowledge. To solve this problem, ontology is thus introduced to help understand the semantics of components. The typical work include Pahl [11], Sugumaran [14], Liu [8], Yao [15], Yen [4][15].

To summarize, although ontology-based technologies have been used in component specification and retrieval, existing approaches have the following limitations: i) ontology in existing approaches has too simple and/or monolithic architecture and few relationships and consequently incapable for a holistic specification of components, in

particular large and complex ones; ii) the gauge of relevance in component retrieval is too simple, inaccurate and only based on incomplete factors; iii) the evolution of the component specification ontology is not considered; iv) the impact of component adaptation is not included as an integral part of component specification and retrieval.

3 Multiple-Viewed Interrelated Component Specification Ontology Model (MVICS)

A holistic ontology model of component specification will provide the foundation for effective semantic reasoning in the component retrieval and improve substantially the precision of component retrieval. The MVICS ontology model has a pyramid architecture, which contains four facets: *function model, intrinsic model, context model* and *meta-relationship model,* as shown in Figure 1. Each of the four models specifies one perspective of a component and as a whole they construct a complete spectrum of semantic-based component specification. All the four models are ontology-based, and are extracted from the analysis of CBSE knowledge and have extension slots for specific application domains.

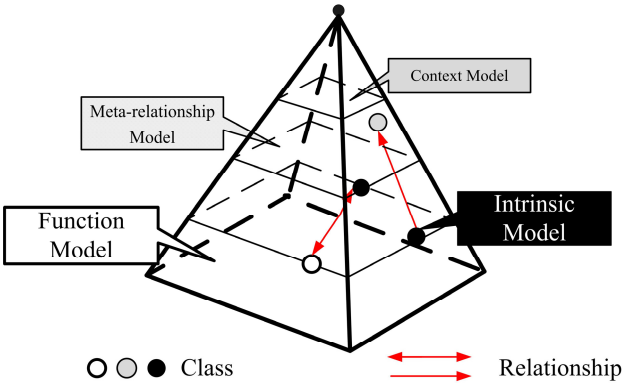

Fig. 1. Multiple-Viewed and Interrelated Component Specification Ontology Model

3.1 Intrinsic Model

The intrinsic model specifies the information of a component which is essential but irrelevant with functionality and quality features of the component, e.g. its name, type, and applicable software engineering phases. In the proposed approach, such information is defined as "intrinsic information" of the component. A taxonomy of the intrinsic information is developed, which includes attributes such as *component name*, *component vender, component price, component version, component type*. These attributes are then further modeled in levels of sub-attributes. The intrinsic attributes are finally modeled as classes in the intrinsic ontology model. Among these classes, two types of relationships are used to show the links between the classes in different layers. *isA* relationship is used to describe super- and sub-class links between component types. *isAttributeof* defines the value set of an attribute of a class in the

ontology model, e.g., *component vender* class is linked with a set of venders under the "*isVenderof*" relationship.

3.2 Function Model

The function model specifies the functionality and quality of service of components. Functions are performed by components which represent fundamental characteristics of software, and a component provides specific functionality or carries out a specific task in a particular business domain. As an ontological model, the top level classes include *function type*, *component domain* and *QoS*. Due to the classes overlap between different domains, the subclasses of the function type are defined in detail and are classified without any overlap, such as *data conversion*, *data entry, data validation* and so forth. The way to link the classes is the same as intrinsic model. *isAttributeof* is used to connect classes which are used to describe a sort of component attribute such as *component function* and *component domain*. Some of these classes link to instances directly, but some of them have large tree type architecture of subclass, sub subclass and so forth. In this sub model, *isA* is used to link the classes and its subclass.

3.3 Context Model

The context model is used to represent the reuse context information of the components, including but not limited to the application environment, hardware and software platform, required resources and possible dependency with other components. The top level classes consist of *operating system, component container, hardware requirement* and *software requirement*. The context model is built in the same way as above two models, i.e., using *isA* to build ontology hierarchies of class *operating system* and class *component container,* and using *isAttributeof* to specify the value set of the attributes of the classes of *hardware requirement* and *software requirement*.

3.4 Meta-relationship Model

Meta-relationship model provides a semantic description of the relationships among the classes in different facets (sub-models) of MVICS. Four types of relationships are identified, namely *Matching Propagation Relationship*, *Conditional Matching Propagation Relationship, Matching Negation Relationship* and *Supersedure Relationship*. Let's define a relationship as $A \rightarrow B$, where A and B are classes in different facets of the MVCIS model. The above four relationships are then defined as follows:

Matching Propagation Relationship

$A \xrightarrow{Pro} B$, which reads as the matching propagates from A to B. It means that if A satisfies the requirement of a component search then B and all its subclasses will satisfy the requirement as well. In component retrieval, such a relationship will enable all the components under class B and its subclasses to be part of the result components for a user query that is matched by class A. The impact on the search path of this relationship is given in part a) of Figure 2. When the search engine identifies class A as a match with the user search keyword K_1, it will continue to search for result components in the subclasses of A, and at the same time also identify class B as a match. It

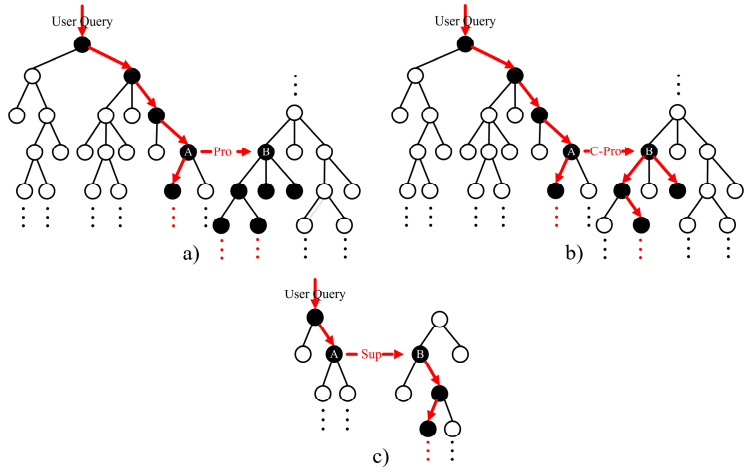

Fig. 2. The impact on search path: a) Matching Propagation Relationship; b) Conditional Matching Propagation Relationship; and c) Supersedure Relationship

would not continue the search in subclasses of *B*, because all the subclasses of *B* are deemed as matching.

Conditional Matching Propagation Relationship

$A \xrightarrow{C\text{–}Pro} B$ (*attri*=*V*), which reads as the matching propagates from *A* to *B* on the condition that value of attribute *attri* is *V*. In MVICS, $A \xrightarrow{C\text{–}Pro} B$ (*attri*=*V*) means that if *A* satisfies the requirement of a component search then *B* and its subclasses may satisfy the requirement if their attribute *attri* has value *V*. In component retrieval, the relationship enables that the components under class *B* are part of the result components for a user query that is matched by class *A,* if their *attri* has value *V*. This relationship will impact on the search path as follows: when the search engine identifies class *A* as a match with a user search keyword K_1, it will continue to search for result components in the subclasses of *A*, and at the same time search *B* and its subclasses on the condition of *attri*=*V*, as shown in b) of Figure 2.

Matching Negation Relationship

$A \xrightarrow{Neg} B$, which reads as the matching with *A* implies not matching with B. In MVICS, this relationship means that if a result component (C_1) is obtained by a keyword matching with class *A*, then C_1 is not the result component obtained by another keyword matching with class *B*. This relationship deals with problems caused by the incompatible requirements in a user query. When user input several keywords, class *A* and class *B*, which are matched with two different keywords respectively, may have Matching Negation Relationship, i.e., a result component can not belong to both classes simultaneously. To tackle this problem, the user query can be treated as two groups of keywords. One group consists of the keyword matched with class *A*, the other group consists of the keyword matched with class *B*.

Supersedure Relationship
$A \xrightarrow{Sup} B$ reads as the matching of *A* is superseded by that of *B*. In MVICS, Supersedure Relationship means that if the content of class *B* has higher priority to the content of class *A*, then the result components obtained by matching *A* will be replaced by the result components obtained by matching *B*. This relationship provides the following impact on the search path, as shown in c) of Figure 2: when the search engine identifies *A* as a match with a user search keyword K_1, it will stop searching in the subclasses of *A*, but turn to search from *B* and its subclasses.

All the above four sub component specification ontology models are defined in OWL. These OWL documents can be seen as the paths that connect user queries and result components.

4 Holistic and Precise Component Retrieval

4.1 Class Weight Calculation Method

Weight of Class (*Wc*) is defined as the foundation for calculating the precision of result components. In each sub-model of MVICS, every class is given a weight to calculate the relevance of each search result. The rules of weight assignment are: i) In one facet, the lower a layer is, the heavier weight its classes have; ii) In different facets, classes at the same depth in the function model are heavier than those in the intrinsic model and the context model. The weight assignment rules are formally defined as follows:

$$Wc = (1+x)^n \tag{1}$$

where *n* is the level of the layer in which the class locates, $x = 0.5$ if the class belongs to the function model, $x = 0.3$ if the class belongs to the intrinsic model, $x = 0.2$ if the class belongs to the context model. The weight of a search path (*Wp*) is the sum of the weight of the classes included in it.

4.2 Retrieval Algorithm

Based on MVICS, a search algorithm was developed. This algorithm accepts composite search conditions with multiple keywords linked with logic connectors. It recognizes the keywords with ‘and’ as a group of requirements, which are then searched together. Those keywords linked with ‘or’ are considered as two different search requirements, which are then searched one after the other. To correspond with the MVICS model in which component specifications are classified into three aspects, the keywords of a user query are also divided into three groups: Function Keywords (*FK*), Intrinsic Keywords (*IK*), and Context Keywords (*CK*).

The search engine will then search the three groups of keywords in the MVICS OWL documents one by one, even though the value of keywords in any group may be ‘Null’. Meanwhile, it will record the search path of every keyword from the result class to top class and then calculate the path weight by summing up every class weight in this path. The search engine will record the components that link to the result class.

4.3 Precision Calculation Method

After retrieval, a set of records is obtained for each keyword, which includes the result component name, the search path and its weight. The match precision of a result component (*Pc*) is calculated with the following unified formula:

$$Pc = \frac{\sum_{r=1}^{a} WpFK_r}{\sum_{t=1}^{i} WpFK_t} \times 0.5 + \frac{\sum_{r=1}^{b} WpIK_r}{\sum_{t=1}^{j} WpIK_t} \times 0.3 + \frac{\sum_{r=1}^{d} WpCK_r}{\sum_{t=1}^{n} WpCK_t} \times 0.2 \quad (2)$$

The numerators in the formula represent the path weight of the result components that partially match with the keywords in each facet, and the denominator represents the path weight of those perfectly matched.

4.4 Adaptive Component Matching

Component adaptation is a popular means to alter the functionality and quality features of selected components [1][2]. The proposed approach accommodates the impact of adaptation in the specification and selection of matching components. This unique feature will allow a more systematic and holistic view in component specification and selection. We call those components whose function and QoS may vary via the application of adaptation assets "adaptive components". In MVICS, the adaptive components are linked to a class via an adaptation method or assets if the component becomes relevant to that class after adaptation with that method or asset. These adaptation methods and assets are defined as classes or instances in MVICS. The retrieval path is then recorded as an adaptive path, in contrast to the direct path, i.e. without adaptation.

The specification of adaptive components in MVICS and the retrieval algorithm take into account the adaptation effects, the adaptation methods/assets, and the effort associated with the adaptation.

4.5 Search Result Profile

In contrast to most existing approaches, which present to the user the name and precision of the result component, our approach provides a holistic profile of the result component to help the user make the best decision in component selection.

The profile shows the matching result in each sub ontology model and the corresponding adaptation information. The profile consists of: *i)* the result component name; *ii)* the overall precision of the component match, including the precision with component adaptation, and the precision without adaptation; *iii)* the match results in sub models: function model, intrinsic model, and context model; *iv)* the associated adaptation method or asset and its incurred effort.

5 The Prototype Tool and Case Study

A prototype tool with an example component repository is built to verify and automate the approach. In this tool, the MVICS ontology is implemented in OWL. The function, intrinsic and context sub-models are implemented in three different OWL files. The relationships in the meta-relationship model are implemented as links between classes in the above OWL files.

The tool has a simple user interface (Figure 3 a)), where the user can fill the query into text area, and the keywords of the user query are classified by different facets of MVCIS in background process of the tool. The search engine will search the OWL documents first and then connect to the components. The search result is shown in the component match profile which we mentioned before. Furthermore, this tool also provides SQL database search by filling component names directly or clicking the component names in the match profile.

The tool and approach have been applied to a case study. As an example from the case study, a user wants to search for a component with the following requirements:

Function: *File Transfer and Encryption*
Component Type: *.NET Class, and WPF*
Component Platform: *Window XP and Window Vista*
Component Container: *Microsoft Visual Basic 2008, IBM VisualAge, and Oracle JDeveloper*

The search engine searched the keywords one by one in function, intrinsic and context sub-model of MVICS. In the same time, the relationships between the sub-models add more semantics, e.g., the class *IBM VisualAge* in the Context model has a Matching Propagation Relationship with *Java Class* and *C++ Class*, which are subclasses of *Component Type* in the intrinsic model. This implies that components which run on IBM VisualAge should have a component type of Java or C++. This indicates that the result component obtained while the user query is matched. This indicates that the result component obtained while the user query is matched class *Java class* or *C++ class* are also the result components when the user search keyword is *IBM VisualAge*.

The names of the result components and their precisions are displayed in the right pane of the interface, as shown in a) of Figure 3. When a result component is highlighted, its search result profile will pop up, as shown in b) of Figure 3. The upper part of the profile illustrates the result component name and the overall precision of the component search. The first and the second number indicate the precision after the component adaptation, more than one adaptation path (AP_i) is possible.

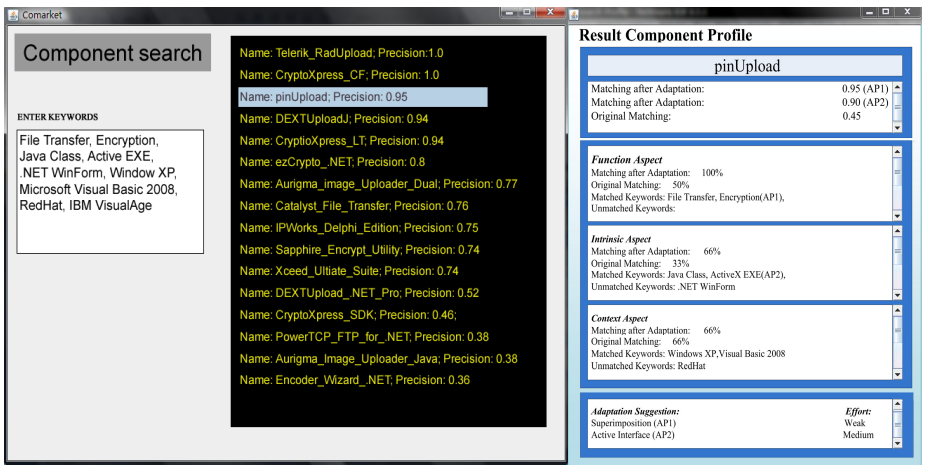

Fig. 3. a) The UI of the MVICS prototype tool b) The Result Component Profile

The third number (0.45) indicates the original match precision. The three output areas in the middle indicate the match results in Function, Intrinsic and Context model. The text area at the bottom of the profile shows the adaptation method(s) or asset(s) used in the component search and their efforts to apply. By clicking the component name in the profile, the complete specification of the component will be presented.

6 Validation

To test the validity of the approach, a project website was built. The prototype tool was transformed to a web application and published on the site. 300 components (acutely component specifications) were selected from several component sale websites, e.g. Componentsources, Componentplanet and Allfreeware with possible adaptation assets developed, and then were populated into a corresponding component repository. Software engineers, researchers and amateurs are able to use the application and comment on it via a questionnaire. The above users followed the following steps to evaluate the MVICS tool against traditional component retrieval approaches:

1) Proposing requirements based on the exiting component specifications and selecting suitable result component (R_1) manually.
2) Using the MVCIS based prototype tool to search the same requirements and receive a set of search results (R_2).
3) Using the SQL database search tool which is supported by traditional approaches to search the requirements again and record another set of results (R_3).
4) Comparing R_2 and R_3 with R_1 respectively, and then fill out a questionnaire regarding how well each search performed according the four criteria: Recall (R), Precision (P), Result Display (RD), and Adaptation Suggestion (AS).

Recall and *precision*, as motioned in section 2, are crucial dimensions to judge the effectiveness of component retrieval. The *result display* is to indicate the degree of user satisfaction with the completeness, clearness and usefulness of the display of the result components. The criteria *adaptation suggestion* is used to estimate the degree of usefulness and user acceptance of the found adaptation suggestion.

Up to present, 69 users have tested the tool in practice. The results of these component retrieval experiments are analyzed and shown in Figure 4. The MVICS based search tool improves recall, precision, result display, and adaptation suggestion effectively at a rather large extend, in particular on the criteria of result display and adaptation suggestion.

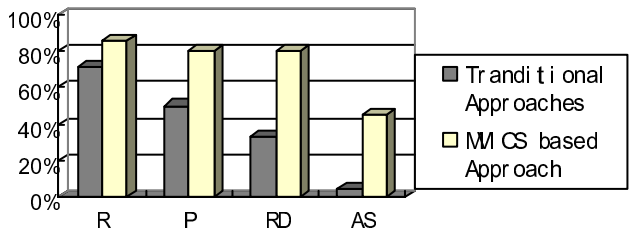

Fig. 4. The level of satisfaction of MVICS prototype tool and traditional search tools

7 Conclusions

The objectives of the research are to develop an ontology-based approach to solving the component mismatch problem via holistic, semantic-based and adaptation-aware component specification and retrieval. Our literature investigation has shown that the proposed approach has novel contributions to the research area and similar work has not been done.

The MVICS ontology model has a novel architecture. It gets rid of the over-complication problem in traditional monolithic ontology, because it has a set of highly coherent and relatively loosely coupled sub-models. The inter-relationships among the classes in different sub-models ensure a holistic view in component specification and selection, and improve the retrieval precision and efficiency. Another contribution is that search result is presented in a profile which consists of a spectrum of elements instead of simply the components and their relevance. Unlike existing approaches, in the MVICS approach component adaptation is considered as an integral part of component specification and selection. Available adaptation assets and methods such as wrappers and aspects are defined in MVICS. During component selection, appropriate adaptation assets/methods will be selected or suggested against the unsatisfied discrepancy.

Our case studies and user feedbacks have shown that the approach and the tool are promising in their ability and capability to solve the identified drawbacks in component specification and selection. In the future, we could improve the MVICS approach by extending MVICS to popular application domains. i.e., add more domain specific attributes to improve its capability, and by developing a mechanism for MVICS model evolution.

References

1. Bosch, J.: Superimposition: A Component Adaptation Technique. Information and Software Technology 41(5) (1999)
2. Bracciali, A., Brogi, A., Canal, C.: A formal approach to component adaptation. Journal of Systems and Software 74(1), 45–54 (2005)
3. Due, R.: The Economics of Component-Based Development. Information Systems Management 17(1) (2000)
4. Gao, T., MA, H., Yen, I.-L., Khan, L., Bastani, F.: A Repository for Component-Based Embedded Software Development. International Journal of Software Engineering and Knowledge Engineering 16(4), 523–552 (2006)
5. Hall, J.: Generalized Behavior-Based Retrieval. In: Proceedings of the Fifteenth International Conference on Software Engineering, pp. 371–380 (1993)
6. Han, J.: A Comprehensive Interface Definition Framework for Software Components. In: Proceedings of Asia-Pacific Software Engineering Conference APSEC 1998, p. 110 (1998)
7. Kim, Y., Stohr, E.A.: Software Reuse: Survey and Research Directions. Journal of Management Information Systems 14(4), 113–147 (1998)
8. Liu, Q., Jin, X., Long, Y.: Research on Ontology-based Representation and Retrieval of Components. In: 8th ACIS International Conference, vol. 1, pp. 494–499 (2007)

9. Mili, A., Mili, R., Mittermeir, R.: Storing and Retrieving Software Components: A Refinement- Based System. IEEE Transactions on Software Engineering 23(7), 445–460 (1997)
10. Ostertag, E., Hendler, J., Prieto-Diaz, R., Braum, C.: Computing Similarity in a Reuse Library System: An AI-based Approach. ACM Transactions on Software Engineering and Methodology 1(3), 205–228 (1992)
11. Pahl, C.: An ontology for software component matching. International J. Software Tools Technology Transfer 9, 169–178 (2006)
12. Patrizio, A.: The new developer portals. Information Week (799) (August 2000)
13. Prieto-Diaz, R., Freeman, P.: Classifying Software for Reuse. IEEE Software 4(1), 6–16 (1987)
14. Sugumaran, V., Storey, V.: A Semantic-Based Approach to Component Retrieval. The Database for Advances in Information Systems, Volna 34(3) (2003)
15. Yao, H., Letha, E.: Towards A Semantic-based Approach for Software Reusable Component Classification and Retrieval. ACMSE 2004 (2004)
16. Yen, I., Goluguri, J., et al.: A Component-based Approach for Embedded Software Development. In: Proceedings of the 5th IEEE International Symposium on Object-Oriented Real-Time Distributed Computing. ISORC, p. 402 (2002)
17. Zaremski, A.M., Wing, M.: Signature Matching: A Key to Reuse. Software Engineering Notes 18(5), 182–190 (1993)
18. Zaremski, A.M., Wing, J.M.: Specification Matching of Software Components. Software Engineering Notes 20(4), 6–17 (1995)