International Journal of Computers & Technology
Volume 5, No. 3, May -June, 2013, ISSN 2277-3061

MVICS: a Repository and Search Tool towards Holistic Semantic-Based Precise Component Selection

Xiaodong Liu, Chengpu Li
School of Computing, Edinburgh Napier University, Edinburgh, UK
{x.liu, c.li}@napier.ac.uk

ABSTRACT

Driven by the continuous expansions of software applications and the increases in component varieties and sizes, the so-called component mismatch problem has become a more severe hurdle for component selection and integration. Although many component repositories and search tools have been proposed, so far there is no satisfactory solution which simultaneously achieves the following goals: automated, semantic-based, and precise. This paper presents a novel component repository and associated search tool which implements holistic semantic-based and adaptation-aware component specification and retrieval. The repository and tool is based on a Multiple-View and Interrelated Component Specification ontology model (MVICS), which has a smooth integration with domain related software system ontologies. The MVICS provides a formally defined and ontology-based architecture to specify components automatically in a spectrum of perspectives. The integration enhances the function and application scope of the MVICS model by bringing domain semantics into component specification and retrieval. The repository and search tool contributes to the current state of the art with four unique features: ontology-based component specification mechanism, semantic-based component retrieval method, adaptive component matching, and a comprehensive result component profile. The repository and tool has been widely tested and evaluated via its online version and follow-on survey reports, which concluded that they are effective for avoiding the component mismatch problem and is promising for industrial use.

General Terms

Component-Based Software Engineering, Component Semantic, Component Repository, Component Reuse.

Indexing terms

Component Semantic, Component Repository, Adaptation Assets, Component Result Profile.

Academic Discipline And Sub-Disciplines

Computer Science, Information and Communication Technology (ICT), Software Engineering.

SUBJECT CLASSIFICATION

Software Engineering, Software Reuse, Component-Based Software Engineering.

COVERAGE

Modern Computing , Information and Communication Technology (ICT).

TYPE (METHOD/APPROACH)

Software Design and Development, Knowledge Design, Case Studies and Experiments, Literary Analysis.

1. INTRODUCTION

As a popular software development methodology, the advantages of Component-Based Development (CBD) have been well stated by many publications [5][16][19], such as shortened development life cycle, reduced time-to-market and reduced development costs. However, CBD is still not widely accepted as it should be due to the increasingly severe difficulty in finding perfectly matching components automatically, in particular for large complex software applications and from a huge collection of diverse and often very similar components. Severe mismatches often exist between the user query and result components due to the lack of consideration of full semantics, lack of precise understanding of the semantics and lack of automation support. These problems not only prevent CBD from reaching its full potential, but also hinder the acceptance of many existing component repositories.

To overcome the above problems, several component search tools had engaged a variety of technologies to support better component specification and retrieval, among which several recent research projects attempted to use domain models and ontology in component retrieval [1][9][17][21][22]. Based on an analysis of the existing semantic-based component specification and retrieval approaches (Section 2), it is clear that the ontology in these approaches has a monolithic structure and few relationships to deal with the specification and retrieval of modern components, which narrowed its application scope. In this paper a novel ontology-based approach is developed and then fully realized for holistic and semantic-based component specification and follow-on automatic and precise component retrieval. As the foundation of the proposed approach, a Multiple-View and Interrelated Component Specification ontology model (MVICS) is developed for component specification and repository building. A formal definition of the MVICS model is first presented in the paper, which ensures the rigorousness of the model and the high level of automation of the tool. The MVICS model provides an ontology-based architecture to specify components in a spectrum of perspectives; it integrates the knowledge of Component Based Software Engineering (CBSE) and the domain knowledge of application domains, and supports

International Journal of Computers & Technology
Volume 5, No. 3, May -June, 2013, ISSN 2277-3061

ontology evolution to reflect the continuous developments in CBD and components. Moreover, the integration with a domain related software system ontology model enhances the function and application scope of the MVICS model by bringing more domain semantics into the component specification and retrieval. Based on the MVICS model and the integration, a MVICS based component repository and search tool have been developed and then published as an online system via the project web site. The repository and search tool supports semantic-based component matching and adaptive component matching; they are fully automated, and presents a comprehensive profile of the result components instead of a mere number of relevance. The result of retrieval includes not only the matching components but also accurate relevance rating and unsatisfied discrepancy, which are presented to CBD engineers in the component matching profile.

The rest of the paper is organized as follows: Section 2 discusses related work. Section 3 and Section 4 describe the formal definition of MVICS and its realization in a component repository. Section 5 describes the realization of the MVICS based component search tool. Section 6 presents the case study. Section 7 evaluates the repository and tool via real-life use. Finally, section 8 presents the conclusion and future work.

2. RELATED WORK

In Component Based Development (CBD), the mismatch between the requirements and the selected components has existed as a rather persistent problem, which is getting increasingly severe with the emerging of modern software systems and the evolution of CBD itself. At an early stage, software developers modify the code of accessible components to satisfy the requirements. This approach is known as white-box reuse, which is applicable to local repositories and incurs much cost in making the changes. Thereafter, most local repositories extend to external or even global markets, and components are usually reused “as is”, i.e. without changes to its code. This type of reuse is known as black-box reuse. As the investigation in [1][6][12][14][20], more and more component vendors put their components to the Internet, which gradually forms numerous online component agencies (component repositories), including ComponentSource, Flashline, Buydirect, Brattbery, Findcomponents. The growing component markets raise thirteen conditions [18] which affect CBD. Three of them play a crucial role in connection with overcoming the mismatch problem, including user query formulation, standard specification of components and component search engine with search relevant rating. This paper focuses on the last two issues, which are referred to as component specification and retrieval respectively.

Component search tools were developed on the basis of component description and retrieval approaches. The existing approaches can be classified into two types: traditional and ontology-based. The traditional approaches [11][12][13][23] include keyword searching, faceted classification, signature matching and behavioral matching. Two typical examples of component search tools from traditional approaches are Agora [15] and Zaremski [23].

Traditional approaches are not effective for component selection, suffering from lower recall and precision, i.e., poor completeness and accuracy of components matching [17]. 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 the components [17][20][21][22]. Sugumaran’s tool [17] is developed on the basis of his proposed semantic-based component retrieval approach, which enables a user to execute more intelligent queries by using domain ontology and natural language parsing techniques. With the help of domain ontology, the tool generates three layers of optional user query refinement panels, which are selected by the user to identify the accurate requirement. From this process, the initial query is augmented or revised by exploiting the additional knowledge from domain ontology. The specification of result components and other related components are shown on the search result panel at the end. This is a good start point to use domain ontology models for refining the user query and specifying the component. However, the tool is not mature enough to apply in large-scale domain ontology, because it is not taken into account how to acquire and evaluate the domain ontology. Moreover, the display method of result components is too simple.

From the above literature analysis, we conclude that existing component repositories and search tools failed to have a sound semantic model as their foundation to reach adequately sufficient level of automation and comprehension of the semantics of components. Consequently, these repositories and tools are doomed to have fatal drawbacks in the delivery of the desired aims. In detail, these limitations include: i) the ontology models in existing repositories are all domain specific, therefore a generic computing-oriented overview is missing, often leaving the component search in a unsystematic style and within too narrow scopes; ii) the domain ontology in existing component repositories is too simple for a holistic specification of components, in particular large and complex ones; iii) the architecture of the domain ontology is monolithic and has few relationships, which limit their semantic expressiveness; iv) the evolution of the domain ontology is not considered; v) existing ontology-based repositories and search tools presumed that the domain ontology in use already exists; the method to access such domain ontology is not mentioned.

3. THE FOUNDATION FOR SEMANTIC COMPREHENSION AND TOOL AUTOMATION

To overcome the above limitations, a formal and machine recognizable semantic model is needed as the foundation for component specification and follow-on automated component search. The semantic model needs a domain-independent structure and meanwhile a domain-friendly interface for integration with a set of common application domains. This section defines the foundation for achieving the targeted holistic semantic-based automatic component retrieval, which includes the Multiple View Interrelated Component Specification ontology model (MVICS), and its linkage with domain models, and the formal definition of both.

International Journal of Computers & Technology
Volume 5, No. 3, May -June, 2013, ISSN 2277-3061

3.1 Multiple View Interrelated Component Specification Ontology Model

The MVICS ontology model has a pyramid architecture, which contains four facets: a function model, an intrinsic model, a context model and a meta-relationship mode. 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 a CBSE knowledge and have extension slots for specific application domains. The first three models (function model, intrinsic model and context model) can be viewed as sub-ontology models, each of which describes one facet of the component specification. The forth (meta-relationship model) is used to store four types of inter-relationships among the classes of the first three models. These relationships represent more of the semantics among each facet in the model, while the architecture of the model is not thereby changed. A detailed description of the MVICS model can be found in [7], as a supplement to the brief introduction that follows (3.1.1-3.1.4).

3.1.1 Intrinsic Model

The intrinsic model specifies the essential information of a component, which does not have to be relevant to the functionality and applicable context of the component, e.g. its name, type, and applicable software engineering phases. In the proposed model, such information is defined as top-level classes including *component name*, *component vender, component price, component version, component date, component type*. The specific attributes of top-level classes constitute their sub-classes and sub-sub-classes, till the most specific units at the bottom level. Among these classes, two types of relationships are used to show the links between the classes in different layers. The *isA* is used to describe superior-subordinate relationship between component types. The *isAttributeof* defines the value set of an attribute of a class in the ontology model.

3.1.2 Function Model

The function model specifies the functionality, domain information and interface description of components. As an ontological model, the top-level classes include *function type*, *component domain* and *interface*. Functions are performed by components which represent fundamental characteristics of software. Specific application domain ontologies can be interfaced with the function model as the sub-classes of the class *component domain*. Class *interface* includes two composite sub-classes *pre-conditions* and *post-conditions*, each are then further composed of a set of methods pre-condition and post-condition, according to the methods/procedures in the component. The way to link classes in the function model is the same as in the intrinsic model.

3.1.3 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*.

3.1.4 Meta-Relationship Model

The 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*.

3.2 Linkage between domain related software system ontology and MVICS

The MVICS model is a component specification ontology model based on the IT specific functions, rather than the application domain related functions and other features. Therefore this MVICS model does not support domain oriented component specification or selection. To extend the semantic-based component search into a specific domain two mechanisms, namely *Association Link* (*AssL*) and *Aggregation Link* (*AggL*), were developed to integrate the domain related software system ontology into MVICS. With such integration, the domain ontology is linked to MVICS effectively and thus extends the application scope of MVICS without changing the architecture of the model. Because the precision of MVICS component search is calculated based on the weights of the search paths, and the impacts of *AssL* and *AggL* in the search path identification are equivalent to the original relationships in the MVICS or the domain ontology, the calculation of search precision would not be affected by the integration. In the function model of MVICS, the class “component domain” is set to interface domain ontology with MVICS. The *AssL* and *AggL* generated from the integration will be stored under the class *component domain*.

The fundamental of the *AssL* and *AggL* mechanisms was first introduced in [8], which are used to link two different kinds of classes of domain ontology to the MVICS. In this paper, these two mechanisms are defined formally and further utilise to automate the repository building and component search.

Those classes in the domain ontology which can be viewed as sub-classes of a MVICS class are named as “*Association class*”. The *Association classes* in the domain ontology represent specific operations, as a specialization of their MVICS super-classes in the relevant domain. *AssL* is used to link these classes with their super-class counterparts in MVICS.

A set of reusable *Aggregation*s is set up in terms of the IT function in MVICS, to represent the group functions of the domain operations. An *Aggregation* here is defined as a set of MVICS classes which work together to perform a larger function. These reusable *Aggregation*s are the function units of MVICS, with minimum intersection of reusable MVICS

International Journal of Computers & Technology
Volume 5, No. 3, May -June, 2013, ISSN 2277-3061

functions. Each *Aggregation* is viewed as a reusable unit oriented to different operations in the specific domain. To link a domain model with MVICS, an *AggL* is defined as a link of a domain class to an *Aggregation* in MVICS.

3.3 Component adaptation model in MVICS

Component adaptation is a common method to change the functionality and quality features of pre-qualified components [2][3][10]. The MVICS model provides a new adaptation model, which records the impact of adaptation in the specification and selection of matching components. This unique feature gives more choices for system developers to opt for the suitable and low cost result components. We name those components whose function and QoS may vary via the application of adaptation assets as "adaptive components". In MVICS, the adaptive components are linked to a class via adaptation method/assets when the component is relevant to that class, after adaptation with that method or asset. Such adaptation methods/assets are defined as classes or instances in the adaptation model of MVICS.

3.4 Formal definition of MVICS model

While the architecture of MVICS is set up and the relevant classes are in place, OWL- DL is used to define the classes, individuals and relationships of the four sub-models of MVICS. With these formal definitions in OWL-DL, automatic semantic extension, automatic ontology validation, and semi-automated ontology evolution can be achieved with the support of an ontology reasoner.

3.4.1 Original MVICS Model Definition

To define the *function model*, the intrinsic *model* and the *context model* in OWL-DL, let $\top$ represent the top class. C_i^n, C_j^n, C_k^n represent classes in the n[th] level of the hierarchical architecture. The details are as follows:

$$C_i^1 \sqsubseteq \top$$

where *i = name, vendor, price, version, date* or *type* in the *intrinsic model*,

where *i = function type, application domain, interface* or *quality attributes* in the *function model*,

where *i = OS, platform, CPU requirements, disk requirements or memory requirements* in the *context model*.

$C_i^n \sqsubseteq C_k^{n-1}$, which defines C_i^n is a subclass of C_k^{n-1}, for example,

$C_{DLL}^2 \sqsubseteq C_{type}^1$, which states class *DLL* is a subclass of class *component type*.

In a model, the subclasses of the same level of one class are mutually disjoint.

If $C_i^n \sqsubseteq C_k^{n-1}$ and $C_j^n \sqsubseteq C_k^{n-1}$, then $C_i^n \sqsubseteq C_j^n = \bot$, for example

If $C_{java}^2 \sqsubseteq C_{type}^1$ and $C_{.NET}^2 \sqsubseteq C_{type}^1$, then $C_{java}^2 \sqcap C_{.NET}^2 = \bot$,

which defines class *java* and class *.NET* are disjoint on condition that they are the same level subclasses of class component type.

Among these classes, *isA* relationship is used to describe super- and sub-class links between component type. *isAttributeof* defines the value set of an attribute of a class in the ontology model, e.g., *component vendor* class is linked with a set of *vendor attribute* classes (*vendor name, vendor address* .etc) under the "*isAttributeof*" relationship. The relationships are defined as follows:

For the *isA* relationship, we assert that the range of the relationship is the respective class C_i^n:

$C_i^n \equiv \forall isA\,.\, C_i^{n+1}$, for example,

$C_{type}^1 \equiv \forall isA\,.\, C_{DLL}^2$, defines that the relationship *isA* links the class *DLL* to the class *component type*.

For the *isAttributeof* relationship, we assert that the range of the relationship is the respective attribute class $C_{attribute}^{n+1}$:

$C_i^n \equiv \forall\ isAttributeof\,.\, C_{attribute}^{n+1}$, for example,

$C_{vender}^1 \equiv \forall isattributeof\,.\, C_{vendername}^2$, defines that the relationship *isAttributeof* links the class *vendor name* to class *component vendor*.

International Journal of Computers & Technology
Volume 5, No. 3, May -June, 2013, ISSN 2277-3061

The 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 $C_A \longrightarrow C_B$, where C_A and C_B are classes in different facets of the MVCIS model. To define these relationships in DL, we create a transitive rule R_{match}, which defines a matching relationship from C_A to C_B. It means that if C_A matches the requirement of a component search then C_B will match the requirement as well. The OWL-DL definition of first three relationships is as follows:

Matching Propagation Relationship

Relationship definitions: $C_A \equiv \sqcap \forall R_{match} . C_B$

Role assertions: $< C_A . C_B > : \forall R_{match}$

Conditional Matching Propagation Relationship

Relationship definitions: $C_A \equiv \sqcap \exists R_{match} . C_B$,

if $C_B \sqsubseteq C_V$, where C_V defines that the classes have value *V*

Role assertions: $< C_A . C_B > : \exists R_{match}$

Matching Negation Relationship

Relationship definitions: $C_A \equiv \neg \forall R_{match} . C_B$

Role assertions: $< C_A . C_B > : \neg \forall R_{match}$

Supersedure Relationship

Relationship definitions: $C_A \equiv R_{match} . C_B$ if and only if $C_A \sqsubseteq C_B$

Role assertions: $< C_A . C_B > : R_{match}$ if and only if $C_A \sqsubseteq C_B$

3.4.2 Domain Ontology Definition

The method to define the domain ontology in OWL-DL is the same as the Original MVICS ontology, except that the classes located in the same level are not disjoint in the domain ontology.

$C_i^1 \sqsubseteq \top$

$C_i^n \sqsubseteq C_k^{n-1}$, defines that C_i^n is a subclass of C_k^{n-1}, for example,

$C_{FI}^2 \sqsubseteq C_{PE}^1$, states that class *Fund Investment,* is a subclass of *Private Equity*.

Among these classes, the *hasA* relationship is used to describe super- and sub-class links between classes in the adjacent levels. The relationship is defined as follows:

For the *hasA* relationship, we assert that the range of the relationship is the respective class C_i^n:

$C_i^n \equiv \forall hasA . C_i^{n+1}$, for example,

$C_{PE}^1 \equiv \forall hasA . C_{FI}^2$, defines that the relationship *hasA* links the class *Private Equity* to the class *Fund Investment*.

3.4.3 Linkage Definition

The linkage between the domain ontology and the MVICS are established by *Association Class*, *Aggregation*, *AssL* and *AggL*. Because the *Association Class* is a kind of domain ontology class, the definition of the *Association Class* is the same as the domain ontology classes.

To define *AssL* in OWL-DL, let C_i^n represents classes in the nth level of the hierarchical architecture in the MVICS model. Let C_d^m represent classes in the mth level of the hierarchical architecture in the domain ontology. The details are as follows:

For the *AssL* relationship, we assert that the range of the relationship is the respective class C_i^n:

$C_i^n \equiv \forall isA . C_d^m$,

International Journal of Computers & Technology
Volume 5, No. 3, May -June, 2013, ISSN 2277-3061

for example,

$C_{DT}^{3} \equiv \forall isA\,.\,C_{MT}^{3}$, defines that the relationship *AssL* links the class *Money Transfer* in the domain ontology to the class *Data Transfer* in the MVICS.

To define *Aggregation* and *AggL* in OWL-DL, let C_i^n represents classes in the n[th] level of the hierarchical architecture in the MVICS model. Let C_d^m represent classes in the m[th] level of the hierarchical architecture in the domain ontology. Let C_A represent an *Aggregation* in the MVICS model. The details are as follows:

$$C_A \sqsubseteq C_{i_1}^n \sqcup C_{i_2}^n \sqcup C_{i_3}^{n+1} \sqcup \ldots\ldots$$

For *AggL* relationship, we assert that the range of the relationship is the respective class C_d^m :

$$C_d^m \sqsubseteq \forall isA\,.\,C_{i_1}^n \sqcup isA\,.\,C_{i_2}^n \sqcup isA\,.\,C_{i_3}^{n+1} \ldots\ldots$$

4. MVICS BASED COMPONENT REPOSITORY

The MVICS-based repository is more versatile in terms of refining the users query, supporting component specification and managing the repository. Being registered into a repository, a component will be specified into a MVICS format form. The specification of this component will be linked, with the help of the form, to the relevant classes of each sub-model in the MVICS. Such a linkage reveals the semantic information of the originally syntax-based component specification, through either the relationship between classes within one facet, or the interrelationship between different facets. Besides, the domain semantics are represented with the linkage of MVICS to domain models.

All the contents of the MVICS model, including classes (MVICS classes and domain ontology classes), relationships, interrelationships, *AssL*s, *AggL*s, *Aggregation*, adaptive classes and relevant component information, are saved into four types of OWL files. When a user searches for components, the user's keywords will be searched in the OWL file to locate the matched classes in the MVICS. All the components relevant to the matched classes are identified as result components for this particular search. According the facets of the MVICS and the domain ontology model, the OWL files are categorised into four types: Original MVICS OWL file, Linkage OWL file, Adaptive OWL file and Domain Ontology OWL file. An ontology tool Protégé is used to create and edit these OWL files (http://protege.stanford.edu/)

Original MVICS OWL file are generated by editing the contents of the function, intrinsic, context and meta-relationship models of the MVICS model. The *Association Class*, *Aggregation*, *AssL* and *AggL* are applied to integrate domain ontology with the MVICS. Such contents are saved in the linkage ontology OWL file. The adaptive OWL file consists of the OWL format specification of the adaptive methods/assets classes, the relationship between the adaptive classes and the MVICS class, and the adaptive suggestions (represented as the attributes of the adaptive class). The connected domain ontology, its classes and superior-subordinate relationship are saved in the domain ontology OWL file.

5. MVICS BASED COMPONENT SEARCH TOOL

5.1 Tool architecture

To exert the power of the MVICS model in expressing semantics and process automation in component search, a MVICS based component search tool was developed. It implements the complete process of component search, starting from filling the initial query and ending up with receiving the result component profile; the whole process is accomplished automatically. The tool consists of a set of modules, which impact on component search, including Dynamic Class Weight Assignment, Search Precision Calculator, Search Time Recorder and so forth. The general system architecture of the MVICS component search tool is shown in Figure 1. It contains four functional parts: Users Query Refinement, Ontologies and Component Repository, Component Search and Result Display. The Ontologies and Component Repository is identified as the core of the component search process and they controls or supports the key subsystems in other parts of the tool. As the foundation of the search tool, the MVICS OWL files and the corresponding component specification in the Ontologies and Component Repository part have been introduced in previous sections. Other functional parts of the search tool will be presented in accordance with the flow of the component search.

International Journal of Computers & Technology
Volume 5, No. 3, May -June, 2013, ISSN 2277-3061

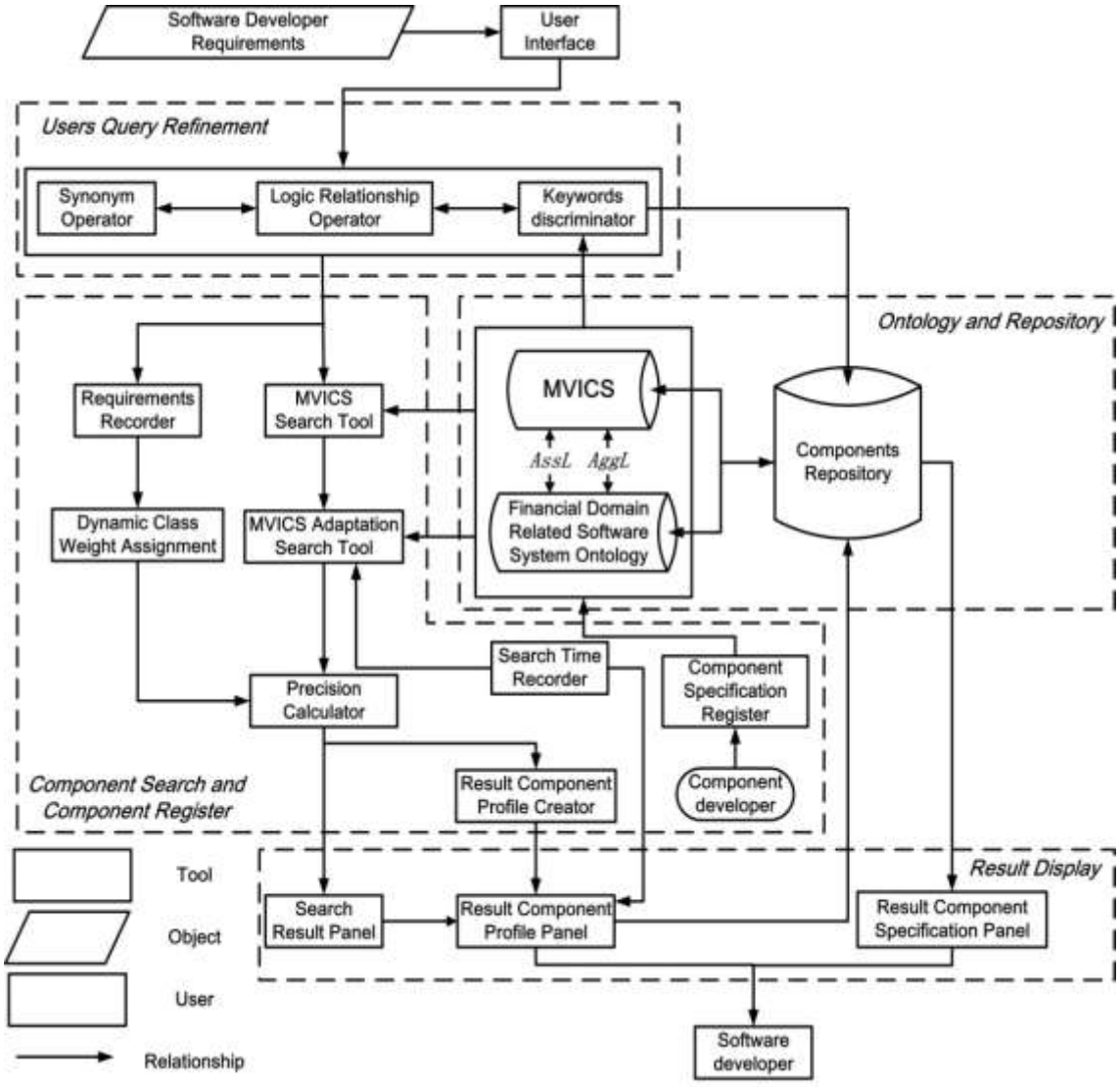


Figure 1: The architecture of MVICS Component Search Tool

5.2 MVICS Component Search

The MVICS component search is based on the MVICS model and the linkage with domain ontology model. It focuses on retrieving the relevant components from the repository according to the refined user keywords. The MVICS based component search comprises two stages: original MVICS component search and adaptive component search. The MVICS component search will identify the matched classes in the MVICS, *AssL* and *AggL* OWL files with the refined user keywords, and will retrieve the result components via the matched classes. As a unique feature of the search tool, the adaptive component search, searches more suitable result components in the adaptive OWL file. It provides not only the matched adaptation assets/methods, but also their suggested effort. The adaptive search results will give user more options during the system development.

5.2.1 Refined User Keywords Parser

Prior to the component search, the refined user keywords will be further processed by the Refined User Keywords Parser (RUK Parser). The parser first parses the keywords base on the sub-models of MVICS. The parser classifies the keywords into three groups, including function keywords (domain related keywords belongs to function keywords), intrinsic keywords and context keywords. In the second step, the parser generates several scratch storages for the component search, according to the numbers of the keywords. The scratch storages deposit the temporary search data generated during the searching process.

5.2.2 MVICS Component Search Path

The MVICS component search tool searches the classified keywords in the function, intrinsic and context models of the MVICS model, adaptive model and the domain ontology model respectively. The components related to the matched classes in each model are identified as the result components. The search also assigns a precision to illustrate the

International Journal of Computers & Technology
Volume 5, No. 3, May -June, 2013, ISSN 2277-3061

relevance rating for each result component, which is calculated on the basis of search paths obtained during the search. Four types of search paths are identified according to the location of the matched classes in the ontologies.

The first type is the original MVICS search path, which is generated by analysing the matched class in the function, intrinsic and context models of the MVICS. It starts from the matched class located in the sub-model of MVICS, and ends with its top-level class in the corresponding sub model. The second and third types of search path are called the domain *Association Link* (*AssL*) search path and domain *Aggregation Link* (*AggL*) search path, which are connected with the domain specific keywords. In the MVICS based component search, the *AssL* and *AggL* are used to generate the domain related keywords search paths. According to the definition, the *AssL* is used to link the domain class (*Association class*) with their super-class counterparts in MVICS. When the user keyword is matched with an *Association class*, a domain *AssL* search path is generated starting from this *Association class*, linking with its super-class in the MVICS, by *AssL*, and ending with the top-class in the corresponding sub-model. The domain *AggL* search path is composed of the matched domain ontology class and the *Aggregation* linked with it by the *AggL* in the MVICS model. The domain *AggL* search path can be counted as the set of original MVICS search paths, which are obtained by the original MVICS classes located in the *Aggregation*.

The last type is the adaptive search path, which is achieved during the adaptive component search. In MVICS, the adaptive components are linked to a class via adaptation method/assets if the component becomes relevant to that class after adaptation with that method or asset. The retrieval path is then recorded as an adaptive component search path, in contrast to the first three types which are obtained during original MVICS component search.

5.2.3 Adaptive Component Search Scratch Storage and Adaptive Suggestion Processing

After the adaptive component search, a scratch storage of each adaptive search path will be generated to keep the path for further precision calculation and adaptation suggestion. The searched keywords with their matched adaptation methods/assets are saved in the corresponding storages. In addition, an adaptation suggestion processing will give every matched methods/assets an effort suggestion. The effort suggestions are classified into three levels, which indicate as Strong, Medium and Weak. For each available adaptation methods/assets, the effort suggestion information is defined as attributes of the relevant classes, which are stored in the MVICS model. An adaptation suggestion processing will invoke these attributes and save them in the adaptive search result storage for displaying to the user.

5.2.4 Result Component Oriented Data Conversion

After the component search, the search results will be processed by the Result Component Oriented Data Converter. The data converter will implement three tasks. The first task is to map the four types of component search paths (original MVICS search path, *AssL* search path, *AggL* search path and adaptive search path) into Function Keyword search path (pFK), Intrinsic Keyword search path(pIK) and Context Keyword search path(pCK), based on whether the matched classes in the component search paths belong to the function, the intrinsic or the context models of the MVICS. The second task is data conversion. The original search results and the adaptive search results in the scratch storage are saved according to users keywords. In order to calculate the precision of each result component, the keyword oriented search results (original and adaptive) should be converted into result component oriented by the data converter. The converted data, which includes the result component name and related search path (pFK, pIK and pCK) are stored in a new scratch storage. The third task is the records of the information of matched keywords, unmatched keywords and adaptive information for each result component.

5.2.5 Precision Calculation

Based on the converted data, the match precision of a result component (*Pc*) is calculated with the following unified formula as mentioned in [7].

$$Pc = \frac{\sum_{r=1}^{a} WpFK_r}{\sum_{t=1}^{i} WpFK_t} \times X_f + \frac{\sum_{r=1}^{b} WpIK_r}{\sum_{t=1}^{j} WpIK_t} \times X_i + \frac{\sum_{r=1}^{d} WpCK_r}{\sum_{t=1}^{n} WpCK_t} \times X_c$$

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. X is fiducial weights, $X = 0.5$ for a class in the function model, $X = 0.3$ for one in intrinsic model, $X = 0.2$ for class in the context model. The yield value of the X for each sub model is given based on our experience, and it will be updated dynamically by the dynamic fiducial weight assignment.

5.2.6 Dynamic Fiducial Weight Assignment

The MVICS component search tool is based on the tree structure of the MVICS model, and calculates the precision of the result component by substituting the values of the search path weights and the fiducial weights (X) into the formula. The fiducial weights (X) of the classes in each model are given as tentative values, which are adjustable via the dynamic fiducial weight assignment mechanism by analysing the user keywords. Each group of the keywords will be recorded by the Requirements Recorder after clicking the search button on the UI. The keywords and their respective sub-model of the MVICS are stored in an XML document. According to the collected data, the fiducial weights will be updated dynamically

International Journal of Computers & Technology
Volume 5, No. 3, May -June, 2013, ISSN 2277-3061

after every 100 groups of user keywords are obtained. The rules of dynamic fiducial class weight assignment are: the more frequently the keywords are used in a facet, the heavier fiducial weight of this facet is [8].

5.3 Result Component Profile

A Result Component Profile is proposed to present a comprehensive view of the search result to the user. This task is fulfilled by the result component profile creator. The creator collates and arranges the search result and the result component precision according to the result component profile format. It carries out two functions: firstly, for each result component, its relevant search paths are arranged by the facets MVICS based component search result, domain related component search and adaptive component search; and secondly, the creator will calculate the percentages of the matched keywords amongst all searched keywords in every facet.

Three panels are developed to show the results. In the UI (Figure 2), the black panel at the right hand side shows the summary of the search results, which comprises the result component names and their precisions. The result components are arranged in the descending order of their precisions. When two result components happen to have the same value of precision, the criterion of cost performance ratio will be taken for the ranking, i.e. the result component with the lower cost will be listed in the front. When the user clicked the result component name, the result component profile appears. In contrast to most existing component search tools, which only present to the user the name and precision of the result component, our tool 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-model of MVICS, the result in domain ontology and the corresponding adaptation information [7]. By clicking the component name in the profile, the complete specification of the component will be presented.

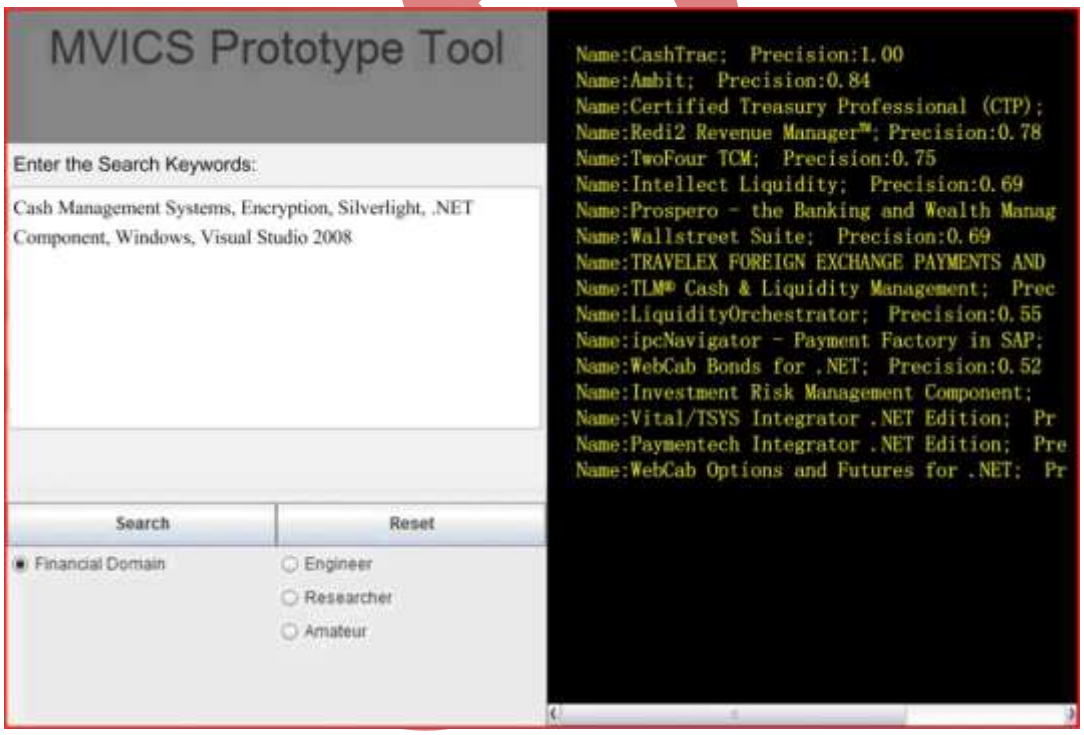


Figure 2: The main interface of the prototype tool

6. CASE STUDY

To exemplify the use of the MVICS component search tool, several search scenarios with corresponding search results are offered on the project website. Users can opt to test some of these given scenarios, or construct their own for the testing. Here we take a financial domain scenario of developing an encrypted Cash Management Systems with Useful Interface to illustrate the function and process of the MVICS repository and its linkage with domain related software system ontology. A financial domain related software system ontology was built, by immigrating existing financial operation ontology. Following the proposed domain ontology immigration method, a financial operation ontology was retrieved from protégé ontology library with help of google filetype search [4]. And then, the selected financial operation ontology was updated by recording and adding the financial operations with the characteristics that maximize the use of CBD approach and the types of the components on the basis of the MVICS format component specification in the repository. Each class in this ontology represents one software system or module that carries out a financial operation. Superior-subordinate relationships have been used to describe the affiliations of the functions of these systems or modules. The top-level classes include systems of Asset Management, Payments & Transfers, Risk Management and so forth. Their subordinates and the subsequent sub-subordinates and so forth constitute their sub-classes and sub-sub-classes, till the most specific function units at the bottom level. Finally, the relevant *AssL*s, *Aggregation*s and *AggL*s are developed by the domain expert.

International Journal of Computers & Technology
Volume 5, No. 3, May -June, 2013, ISSN 2277-3061

After the user query refinement, the scenario is specified as the following requirements:

Function: *Cash Management Systems, Encryption*

Component Type: *Silverlight, .NET Component*

Component Platform: *Windows*

Component Container: *Visual Studio 2008*

User may fill in the relevant keywords into the text area of the search UI and clicks the financial domain option button, and leaves user-oriented option buttons blank for original precision calculation (Figure 2). Thereafter, the search engine first identifies keywords belongs to the models of MVICS. Meanwhile, the information of the keywords and their related models are recorded by the requirement recorder for future refinement. The search engine searches the keywords one by one in the Original MVICS OWL file, the Financial domain OWL file and the Adaptive OWL file respectively.

During the search process, the relationships between the sub-models provide more semantics: for instance, the matched class *silverlight* in the intrinsic model has also a Matching Propagation Relationship with class *multimedia, graphics* and *animations* in the function model. With the semantics meaning of the interrelationship, the *silverlight* type components should have the function *multimedia, graphics* and *animations.* Therefore, these functions are taken into account for the search as supplement. In the context model, the matched class *visual studio 2008* specifies that the OS requirements should be beyond windows XP or above by the Conditional Matching Propagation Relationship. Besides, with the support by the Supersedure Relationship, the search tool will identify the components as the result if the edition of their container is beyond the 2008, e.g. *visual studio 2010*.

The keywords "*Encryption, Silverlight, .NET Component, Windows* and *Visual Studio 2008*" are matched with the classes of MVICS, and their relevant components are identified as the result components first. Afterwards, the search tool continues to search the unmatched keywords "*Cash Management Systems*" in the financial domain OWL files in the same way.

After searching the MVICS and financial domain ontology OWL files, the tool searches available adaptive methods/assets in the adaptive OWL files. All the components relevant to the matched classes are identified as the result components of the search, and the precision for each result component is calculated on the basis of the retrieved search paths by the Precision Calculator. The findings offer the user more options to develop the cash management software systems.

The names of the result components and their precisions are displayed in accordance with the order of precision high to low, on the right panel of the UI (Figure 2). When a result component is highlighted, its result component profile will pop up, which provides a comprehensive summary of the search information for each result component as shown in Figure 3. The profile shows the following information: the result component name, the overall precision of the component search, the match results in the function, intrinsic and context model, the match result in the financial domain ontology and the available adaptation method(s) or asset(s) with their efforts to apply.

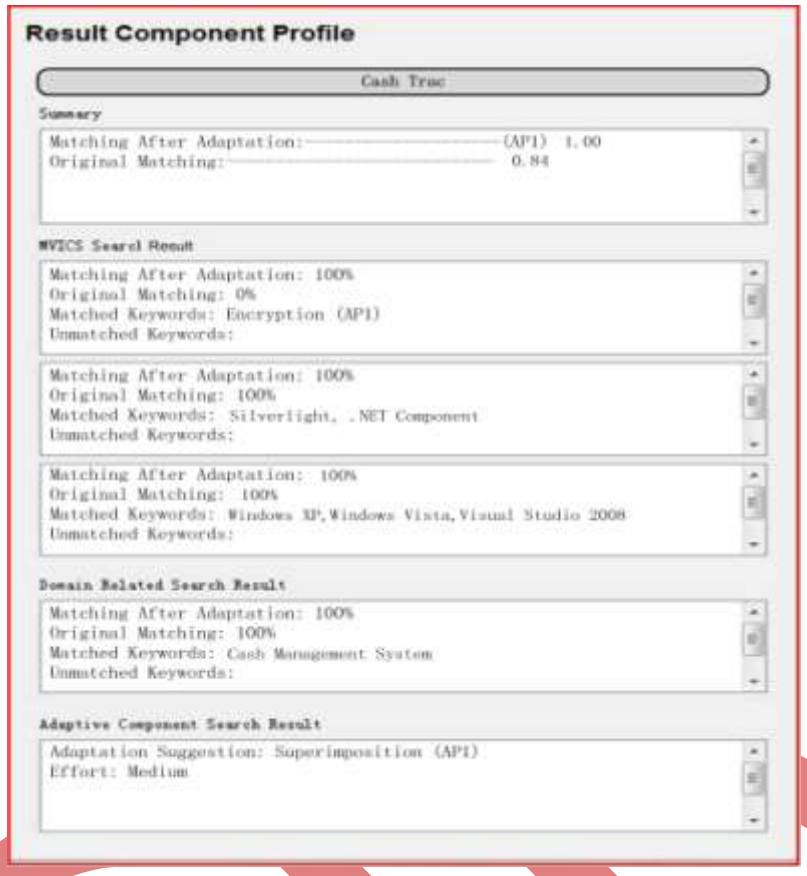


Figure 3: The Result Component Profile

7. EVALUATION

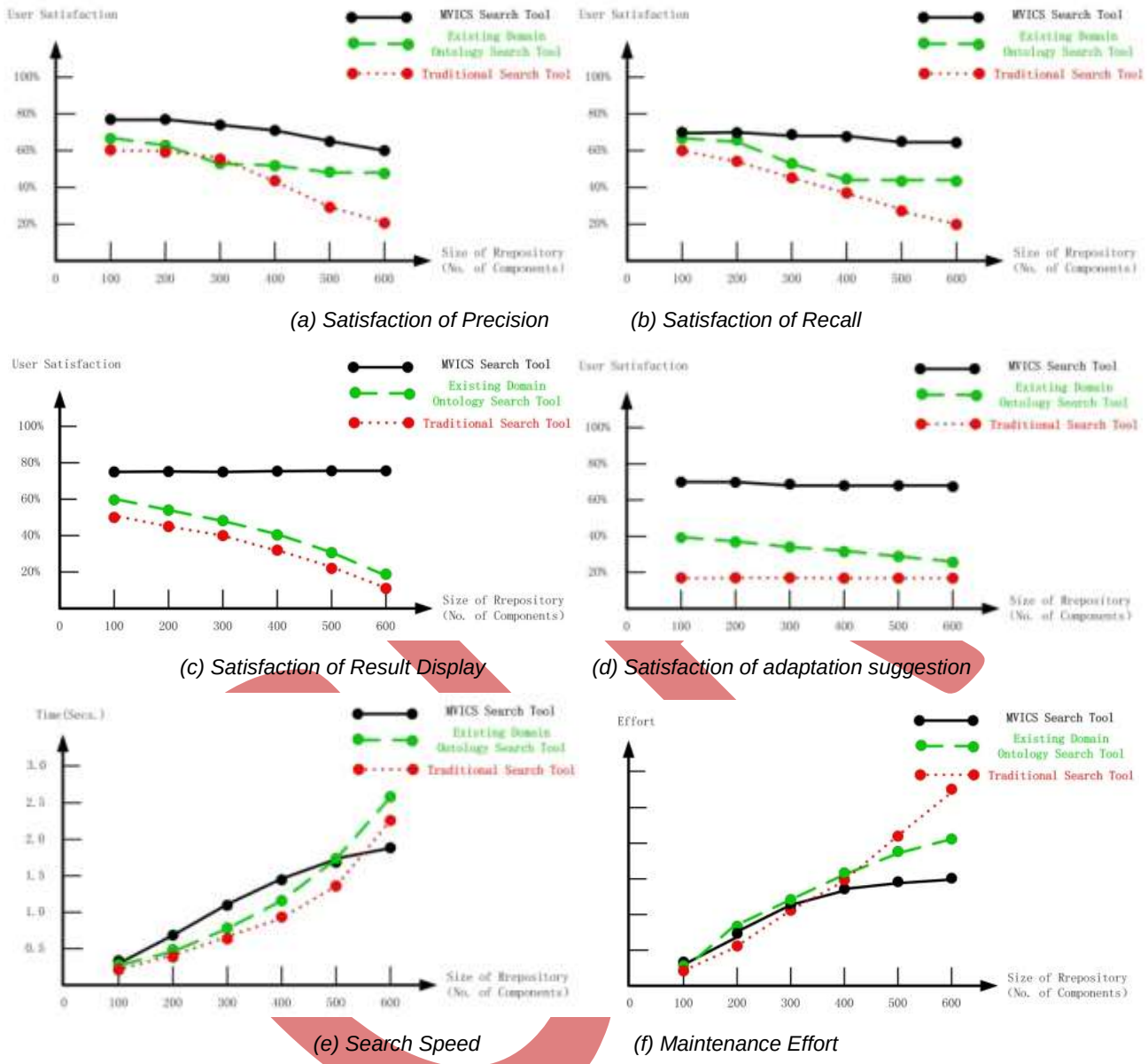
To help users understand and evaluate the MVICS component search tool, a project website is built. Besides the MVICS component search tool, a SQL database search tool and a domain ontology-based component search tool are presented on the website for the comparison test. The SQL database search tool implements the traditional keyword search approach with the support of facet classification and behavior matching approach. The domain ontology-based component search tool implements the existing domain ontology-based approach. It uses the same financial domain ontology to refine the user requirements and specify the components. Software engineers, researchers and amateurs are able to use the applications with the same testing scenarios and to comment on the tool and the search result via a questionnaire.

To date, 125 users have tested the tool in practice, among whom 43% from Europe, 33% from Asia, 17% from North America and 7% from the rest of the world. In the self-appraisal to scale their own software engineering experiences from level 1 to 5 (the level 1 indicates the user has more than 10 years software engineering experience, and then 8 years, 5years, 2 year until the level 5 indicates the user has no experience), 15% opt for scale 1, 7% for 2, 44% for 3, 23% for 4 and 11% for 5. The information of test participants' professional background is shown in Table 1. The results of these component retrieval experiments are analyzed and shown in Figure 4. This figure evidenced that the MVICS search tool improves the recall, precision, result display, and adaptation suggestion effectively to a larger extent, in particular on the criteria of result display and adaptation suggestion, with the result profile and adaptation suggestion as new mechanisms developed in MVICS in contrast to the existing component search tool.

Occupation	Software Researcher	Software Engineer	Amateur
No. of respondents	59	39	27
Years of experience (Average)	7	5	3

International Journal of Computers & Technology
Volume 5, No. 3, May -June, 2013, ISSN 2277-3061

Table 1. Information of evaluation participants



(a) Satisfaction of Precision *(b) Satisfaction of Recall*

(c) Satisfaction of Result Display *(d) Satisfaction of adaptation suggestion*

(e) Search Speed *(f) Maintenance Effort*

Figure 4: The level of satisfaction of the MVICS component search tool, existing ontological domain specific search tools and traditional search tools

The MVICS component search tool improves the search precision and recall, particularly for large repositories as shown in Figure 4a and 4b. Such improvements come from its semantic foundation, i.e., the formalized MVICS ontology model and its linkage with the domain ontology, as well as the successful automation in the repository and search tool. MVICS represents component specification in a multi-faceted and hierarchical structure. In addition, the interrelationships in the MVICS model and the linkages with the domain offer more semantic meaning among the classes of the ontologies (MVICS and domain). In this case, more search paths are retrieved during the search process. This should lead to further improve recall and precision. Comparatively, with lower description capability and less relationships in their ontology, other existing domain ontology search tools offered lower satisfaction in *precision* and *recall* than the MVICS based tool, especially when deal with large repositories.

The *result display* is to indicate the degree of user satisfaction with the display of the result components in terms of the completeness, clearness and usefulness. With the proposed result component profile, the search results can be shown more effectively in the MVICS component search tool (Figure 4c).The criteria *adaptation suggestion* is used to estimate the degree of usefulness and user acceptance of the found adaptation suggestion. With the unique feature of the proposed adaptation component search, the MVICS offers a remarkably improved satisfaction in the aspect of adaptation suggestion (Figure 4d). Besides the improvement of search accuracy, the MVICS tool also takes into account other related properties, such as search speed. To measure the speed, the tool has a Search Time Recorder device, which is used to record the time consumed during component search. Figure 4e shows the comparison of the search speeds of different search tools in repositories of different sizes. The existing domain ontology search tool and the traditional search tool are

International Journal of Computers & Technology
Volume 5, No. 3, May -June, 2013, ISSN 2277-3061

faster than MVICS tool when the size of a repository is less than 500 components; however, the MVICS tool is faster when the repository is in large-scale, when the number of components beyond 500 in this case study. This is because the MVICS tool searches classes in the ontology along its multi-faceted and hierarchical structure. The other reason for the speed loss in the MVICS component search tool is due to more semantic processing, i.e., the semantic-based precision calculation, the adaptive component search and the data collection for a whole profile of result components.

Regarding the effort of maintenance, it is observed that the ontology-based search tools (MVICS and existing domain ontology-based approach) are easier to manage in medium size repository (Figure 4f). Again this advantage comes from that the MVICS component search tool uses ontology formally defined in OWL DL, which makes it possible for the automatic validation through ontology reasoners.

8. CONCLUSION

The presented work provides a novel and effective solution to avoid the component mismatch problem with precise semantic-based and adaptation-aware component selection. Its key contributions lie at: i) the formalized MVICS model as the basis for semantic comprehension and process automation; ii) the integration of domain knowledge into MVICS model via the ontological interface of the developed relationships; ii) the automation of the retrieval process and repository building with the MVICS component repository and the component search tool. Our literature investigation has shown that similar work has not been done prior to the MVICS project. Our user testing and evaluation have also shown that the repository and tool is suitable for industrial use with their delivery of much better functionalities and performance.

The MVICS based repository and search tool provide an integral mechanism for component selection, including component specification, component retrieval, user query refinement, component registration, and online repository management. As the foundation of the search tool, the MVICS ontology model and its linkage with domain ontology not only solves the shortage of description capability of the ontology used in the existing semantic-based component search tools, but also with its formal definition in OWL-DL guarantees the adequately good reasoning capability for component search and ontology evaluation. Moreover, the MVICS gets rid of the over-complication problem in traditional monolithic ontology by a set of coupled sub-models of high coherency as well as relative flexibility. The inter-relationships among the classes in different sub-models also ensure a holistic view in component specification and selection, and thus improve the search precision. On the retrieval side, the MVICS tool supports dynamic and user group oriented retrieval by adjusting the fiducial facet weights, which further decreases the mismatch between the result components and the user query. The use of *AssL* and *AggL* improves the functionality and application scope of component retrieval and provides a practicable way to integrate in domain related system ontology. The adaptive component matching and the search result profile are novel concepts in component search tools; they make the MVICS approach “holistic” for the component retrieval and specification.

For future work, we will further refine the MVICS model, extend the ontology linkage method to more relationships, and refine the formulae for weight assignment and component precision calculation on the basis of a wider and more industrial extent of test results and user feedbacks.

ACKNOWLEDGEMENTS

This work was sponsored by a research grant from Agilent Technologies Foundations.

REFERENCES

[1] Alnusair, A., and Zhao, T. Component Search and Reuse: An Ontology-based Approach. The 11th IEEE International Conference on Information Reuse and Integration (IRI-2010). IEEE Computer Society Press, pp, 258-261.

[2] Bosch, J.: “Superimposition: A Component Adaptation Technique”, Information and Software Technology, 1999, 41(5).

[3] Bracciali, A., Brogi, A., and Canal, C.: “A formal approach to component adaptation”, Journal of Systems and Software. 2005, 74(1), pp. 45-54.

[4] DuCharme, B.: “Googling for XML”, http://www.xml.com/pub/a/2004/02/11/ googlexml.html, February 11, 2004,

[5] Due, R.: “The Economics of Component-Based Development”, Information Systems Management, 2000, 17(1), pp. 92-95.

[6] Guo, J. and Luqi.: “A survey of software reuse repositories”, IEEE International Conference and Workshop on the Engineering of Computer Based Systems, 2000, Edinburgh, UK.

[7] Li, C., Liu, X., and Kennedy, J.: “A Holistic Semantic Based Adaptation-Aware Approach to Component Specification and Retrieval”, ASEA’09. Springer LNCS. 2009

[8] Li, C., Liu, X., Kennedy, J.: “Achieve Semantic-based Precise Component Selection via an Ontology Model Interlinking Application Domain and MVICS”, International Conference on Software Engineering and Knowledge Engineering, SEKE 2010.

[9] Liu, Q., Jin, X., Long, Y.: “Research on Ontology-based Representation and Retrieval of Components”. 8th ACIS International Conference, Volume 1, Page(s): 494 – 499, 2007.

International Journal of Computers & Technology
Volume 5, No. 3, May -June, 2013, ISSN 2277-3061

[10] Liu, X., Feng, Y., and Kerridge, J.: "Generative Aspect-Oriented Component Adaptation", IET Software, April 2008, volume 2, no. 2.

[11] Mili, A., Mili, R., and Mittermeir, R.: "Storing and Retrieving Software Components: A Refinement-Based System", IEEE Transactions on Software Engineering, 1997, 23(7), pp. 445-460.

[12] Mili, A., Mili, R., and Mittermeir, T.: "A survey of software reuse libraries", Annals of Software Engineering, 1998, pp. 349-414.

[13] Ostertag, E., Hendler, J., Prieto-Diaz, R., and Braum, C.: "Computing Similarity in a Reuse Library System: An AI-based Approach," ACM Transactions on Software Engineering and Methodology, 1992, 1(3), pp. 205 – 228.

[14] Peng, Y., Peng, C., Huang, J., and Huang, K. " An Ontology-Driven Paradigm for Component Representation and Retrieval". CIT '09. Ninth IEEE International Conference on Computer and Information Technology, 2009.

[15] Seacord, R., Hissam, S., and Wallnau. K.: "Agora: A search engine for software components", Technical report, Carnegie Mellon University/SEI, 1998.

[16] Sprott, D.: "Componentizing the enterprise application packages", Communications of the ACM, 43(4), 2000, pp. 63-69.

[17] Sugumaran, V., and Storey, V.: "A Semantic-Based Approach to Component Retrieval", The Database for Advances in Information Systems, 2003, 34(3).

[18] Traas, V., and Hillegersberg, J.: "The software component market on the internet current status and conditions for growth", Software Engineering Notes, 25(1), 2000.

[19] Vitharana, P.: "Risks and challenges of component-based software development", Communications of the ACM, 2003, 46(8), 67– 72.

[20] Yan, W., Rousselot, F., and Zanni-Merk, C.: "Component Retrieval Based on Ontology and Graph Patterns Matching". Journal of Information & Computational Science, 7(4), 2010, 893–900.

[21] Yao, H., and Letha, E.: "Towards A Semantic-based Approach for Software Reusable Component Classification and Retrieval", ACM Southeast Conference, ACMCE 2004.

[22] Yen, I., Goluguri, J. et. al.: "A Component-based Approach for Embedded Software Development", IEEE International Symposium on Object-Oriented Real-Time Distributed Computing. ISORC 2002, pp. 0402.

[23] Zaremski, M., and Wing, M.: "Specification Matching of Software Components", Software Engineering Notes, 1995, 20(4), pp. 6–17.

Author' biography with Photo

Dr. Xiaodong Liu is a reader and the director of Centre for Information & Software Systems, in the School of Computing, Edinburgh Napier University, UK. As an active researcher, his research focuses on context-aware services, service evolution, mobile clouds, pervasive computing, software reuse, and component-based software engineering. He has led 6 externally funded projects, and published over 60 papers in established international journals and conferences. He is the editorial board member of 3 international journals and editor of 2 research books. He is a member of IEEE Computer Society and British Computer Society.

Dr. Chengpu Li received the PhD degree in Computer Science from Edinburgh Napier University in 2012. He is currently a research scientist in China Mobile. His research interests include component semantics and repository, ubiquitous and pervasive computing and service-based systems. He has published in several well-known international journals and conferences.