A Representation Model for Reusable Assets To Support User Context

Hend Ben Hadji ° Ho-Jin Choi
School of Engineering, Information and Communications University
{hend.benhadji, hjchoi}@icu.ac.kr

ABSTRACT

In the field of software reuse, several methods for storage and retrieval of assets abound. However, these methods often find their limits; they fail to turn up the suitable reusable assets that satisfy the needs of a particular software system under development. Two problems are the root cause of this situation. One is the lack of accurate semantics for describing software assets. The other is the ignorance of the user query context. In such a context, this paper presents an XML-based asset representation model for describing all kinds of software asset that can be reused within software development process. The proposed model provides semantic metadata for describing assets oriented user context in order to build the foundation for semantic reasoning in the retrieval process.

1. INTRODUCTION

Software reuse has been drawing great attention from researchers since it was proposed about three decades ago. Today every one would agree that the reuse of exiting software is considered as an effective approach to improving the quality, reliability and productivity of software development [1, 2]. However, nowadays, the most challenging tasks in software reuse are to discover suitable reusable assets that fulfill the requirements of a particular software system under development [3]. In order to tackle such a challenge, several methods for storage and retrieval of software assets have been introduced by industry and academia, such as simple keyword, faceted approach, signature matching and behavioral matching. However, most of the proposed methods are still immature and unsatisfying; they often find their shortcomings in turning up the relevant assets to user needs. Mili and Mili [4] point out that successful retrieval method is critically dependent on the appropriate description and representation of both the query and the software asset since the retrieval depends on matching a candidate asset against a query.

Recently, several representation models for describing software assets have been emerged, the most important contributions have been proposed by [5], called Reusable Asset Specification (RAS). This model has improving the retrieval process by not only proposing accurate semantics of describing software assets, but also supporting the description of any kind of reusable assets stored in the repository system, that is any by-product generated during the software process development which can be potentially used. A limitation however with the prior model is that it merely provides the user with the assets that match with his query which is generally seen vague and sometimes confusing. In other words, the user's point of view is usually ignored while defining the metadata of describing assets, such as what he wants to know about the asset, or what the intent of his query is. User generally makes supplementary efforts to find out from the retrieved results the assets that well fit with the usage context and his preferences; this task is very consuming-time and is unfortunately in opposition to the software reuse benefits.

In order to ease the burden of the user, we fervently believe that user context is potentially useful in such a case and warrants to be integrated as a part of the query during the retrieval process. Such a consideration could not be achieved if the metadata of describing assets do not provide matches against user context.

Consequently, we propose in this paper a novel representation model which will extend RAS specification by introducing new features, such as Application Domain, Accessibility, Requirements, and Previous Reuse Experience, and so on. The representation of the proposed model is based on XML language which facilitates its distribution and processing. Our proposed model aims at enhancing the retrieval results and building the foundation for semantic

reasoning in the retrieval process through the incorporation of user context.

The remainder of this paper is structured as follows: Section two outlines a statement of the problem which will be followed by related work (section 3). In Section 4, a brief description of RAS model is presented, followed by some details of our new model. Finally, the paper concludes with a general summary and statement of future work.

2. PROBLEM SATEMENT

Considerable software reuse research has focused on identifying reusable components. Girardi and Ibrahim's ROSA system [6] for retrieving software artifacts uses natural language processing for user queries and software component descriptions. Recently, Sugumaran and Storey [7] proposed a semantic approach. Their approach includes ontology of inter-related domain-terms and definitions to describe components within a software library. A limitation with the prior models is that they rely on the user query which is generally seen vague and sometimes confusing. The user is required to specify more precisely what he actually needs in order to obtain relevant assets. But in many cases, the user is either not capable of doing that, or it drastically reduces the usability of the system. This problem can be overcome by the system's awareness of the user context, such as what is his current task, what kind of working environment does he use, what software asset performances are demanded, is he novice or expert, and so on. The system can then add transparently implicit assumptions of the user to his explicit query.

However, the requirements for applying user context in software retrieval methods calls primarily for extending the metadata of describing software assets, which are currently not geared towards the peculiarities of user context data. Basically, when a new asset is stored in the repository system, the asset developer associates metadata to describe its functionalities and behaviors without taking into account what the user really wants. Therefore, we assume that the extension of the metadata is of paramount need for developing retrieval method capable of discovering suitable software assets required by a given user and a specific user context as well.

In this paper, a new approach is presented that is capable of representing and efficiently dealing with the user context. The proposed approach will be built upon RAS model, discussed in the following. We choose RAS specification for two reasons. First, it is the first and the only project that include accurate description of all kinds of software assets, unlike other research which are concentrating on components form (executables or code source). Second, the consideration of many forms of asset will fully require the incorporation of user context.

3. RELATED WORK

Although much software reuse research has focused on storage and retrieval techniques, less has deal with context-awareness.

Ye and Fischer [8] are the first to propose a context-aware search and retrieval tool, called CodeBroker. They use a proactive approach to formulate user query. CodeBroker is integrated into the development environment so that the system can find components relevant to the task. CodeBroker runs continuously as a background process in Emacs and deliver suitable components whenever a standard Java is entered in the environment. Thus the comment is considered as a query and the system will respond and show a list of the candidate components. The system will analyze the preceding java comment with signature to find matching components. However, the effectiveness of this approach may be limited by the need to write suitable comments which most developers lack the skill of writing goods comments [9]. Moreover, the use of context is almost similar from one situation to other; the candidate asset is always java code source and the environment tool used has to be Emacs.

Holmes and Murphey [9] proposed a search engine, called *Strathcona* that uses the concept of structural context, based on java source code. Strathcona resembles the CodeBroetrieval system. The tool extracts the structural context of the code on which a developer is working when the developer requests examples. In contrast to CodeBrocker, Strathcona can apply to any code and any framework irrespective of coding conventions since all source code incorporates structure. However, the context is still not understood in his proper meaning.

Therefore, we assume that the use of context could be more effective for retrieval methods if the situation change from user to another, such as his role, the environment used, and so on.

4. USER CONTEXT

It is true that modeling software asset enables a better understanding of reusable assets; however, it is not sufficient and enough to enhancing the retrieval process. Recently, researchers have come to the conclusion that context information in combination with query is important while searching for specific information [10]. Likewise, we fervently believe user context could play an important role in refining the retrieval process. Thus, combining the user context with his query makes user's requirements more clear

and understandable by the retrieval process. The context will make the system understand and adapt the retrieval process and thereby providing the user with the accurate assets.

In this way, user context should include all information that might represent the situation of a given user and influence the retrieval results. *Dey and Abowd [11] define context as "a piece of information that can be used to characterize the situation of participant in an interaction". Similarly, Luo and Hsieh [12] define context as "any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and application themselves*".

The concept of Context has often been interwoven and used in different fields [13]. Information that constitute context change from ubiquitous domain or artificial intelligence domain, to software engineering domain, and therefore the information tend to become subjective and very domain specific.

In this work, we propose the following categories of user context elements that may refine the retrieval results:

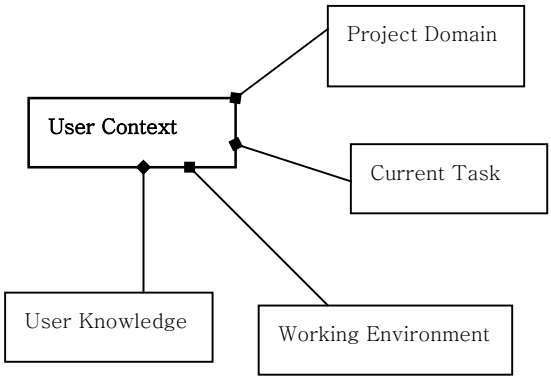

Figure 1. User Context elements

- Project domain, identifies the project in which the user is involved, as well as the application domain of a particular software system under development;
- Current task, defines the current role of user (analyst, designer, programmer, or validator) and the goal within software development process;
- Working Environment, specifies the environment tool used to perform his task (Integrated Development Environment, Editor, UML modeling, etc.), as well as the development language used (this information is important if the target asset is a program code);
- Knowledge, includes information about the competencies and skills of user in software engineering area, whether he is novice or expert.

Note that our focus, in this paper, is not to investigate how to acquire, represent and use user context, but to identify the relevant user context information for the purpose of suggesting an asset representation model whose metadata could support matches against user context.

5. REPRESENTATION MODEL FOR REUSABLE ASSETS TO SUPPORT USER CONTEXT

In this section, we describe briefly the standard RAS and then we introduce the new features of our new model.

5.1. RAS

RAS [5] is a standard for packaging reusable software assets, which was adopted by Object Management Group (OMG). RAS defines a reusable software asset as any cohesive collection of artifacts that solve a specific problem or set of problems encountered in the software development life cycle. Artifacts are any by-products from the software development lifecycle, such as requirements documents, models, source code files, deployment descriptors, test cases or scripts, etc. Every reusable asset must contain at a minimum one manifest file and at least one artifact to be considered a valid reusable asset. The manifest file is an XML document that validates against one of the known RAS XML Schemas, and passes an additional set of semantic constraints described in the profile document. An asset package is the collection of artifact files plus a manifest. Although RAS include all kinds of software with their associated description, it is not fully appropriate to deal with user context and solve what the user needs.

5.2 OUR MODEL

Based on [5, 3] and user context information, seven dimensions have been proposed:
- *GenInfo,* provides general information about the asset;
- *Dependency,* describes the relation with other assets and provides the degree of dependency;
- *Classification,* defines the application domain in which the asset might operate as well as the function(s) or service(s) provided;
- *Non Functional Requirements,* indicates the prerequisites necessary to make the correct deployment and usage of the asset;
- *Resources,* refers to as useful resources to help the user to understand the assets.
- *Accessibility* is concerned by the usability of the asset;
- *Experience Reuse* provides information about previous

experiences of the reuse of assets, as well as some advices in order to use properly the asset; each dimension, or sub-class, represents some characteristics provided by the asset. For better understanding of the proposed model, we present each dimension iteratively to analyze relevant properties of each one.

The two first features are based on RAS Specification, the third and fourth features come from [14], while the last ones are our contribution.

5.2.1 GenInfo Dimension

This class has three attributes. (1) *Variability* element enlightens the user the modification level of the implementation. Assets whose internals can not be seen like black-box assets have a low level of variability; whereas white-box assets whose internals can be modified have a high level of variability. There are also difference in variability between glass-box assets and gray-box assets. Glass-box assets expose implementation details, however they can not been modified. Gray-box assets expose and allow modification only to a subset of the asset. (2) *Articulation* element describes the degree of completeness of the asset in providing the implementation solution. That is, the phase of software development process. Assets which specify the solution and do not provide the implementation have a low degree of articulation such as requirements, design patterns, and use cases; whereas, assets which provide the complete solution along with supporting documents, have a grater degree of articulation. (3) *Technology* element specifies whether the asset is standardized or conform to some model that constrains their deployment, such as CCM, COM+, JavaBeans, EJB, and .NET

5.2.2 Relation & Dependency Dimension

This class has two sub-classes: *Composition* and *Dependency*. The *Composition* element identifies the artifacts, or so called multi-phased asset, that compose an asset. These artifacts are created by asset developers during the software development process, such as requirements, designs, test suite. Such artifacts can help greatly the user to find a solution to his software system under development. Whereas, *Dependency* element refer to as the relationships with other assets stored in the repository system, called related-asset. These relationships could be an aggregation, similarity, dependency or parent [5]. The relation & Dependency dimension is necessary to scale asset reuse to larger-grained, coarse grained, or fine-grained, and then to provide user with the level of the complexity of the asset reuse.

5.2.3 Classification Dimension

Given that the reuse success is directly related to finding the suitable asset [3], it is essential to carry out an effective asset classification mechanism. Here, we provide hierarchical classification based on the Application domains. The asset developer can choose from the predefined hierarchical classification the relevant domain(s) that suit best with the asset characteristics and behaviors. The classification mechanism is seen as a key factor in retrieving the relevant assets from a large repository system since it provides the user with the scope of reuse, the functionalities that the asset is supposed to do and under what circumstances.

5.2.4 Non-Functional Requirements Dimension

For better usage of the asset, non-functional requirements should be defined explicitly to the user. *The non-functional requirements* are grouped into sub-categories, mandatory and optional elements. The mandatory requirements provide information about operating system, required storage space and processor family. While optional requirements allow the developer to specify specific requirements, such as memory space, required tools and so on.

5.2.5. Accessibility.

This feature is also important for user because it is concerned by the usability of the asset; it informs the user about the skills and knowledge required to handle efficiently the asset. This feature supports the context matcher against User knowledge to provide the suitable asset that he goes with his skills.

5.2.6. Resources Dimension

It is believed that good asset documentation would help the user to use properly the retrieved asset; the documentation generally contains useful information about asset usage. However, even though the documentation seems be useful for efficient reuse, it is not enough in its self for success reuse. Therefore, we argue that references such as tutorials, additional resources could help the user for better understanding.

5.2.7 Experience Reuse Dimension

In most cases, users are interested in getting information about the previous experiences of the asset reuse. In other words, users who have never used the asset before paid a great attention to the points of view of others before start using the retrieved asset. In such a context, our approach presents information about the previous experiences of asset reuse as well as some advices in order to use properly the asset.

Our approach aims at assessing the user in finding the relevant asset which fit with his context, help him for better understanding, and guide him for proper reuse. By achieving these goals we will ease the burden of user. Our future work will focus on developing a retrieval methods based upon the proposed model.

6. CONCLUSION AND FUTURE WORK

Our claim in this paper is that user context is rarely used in existing retrieval process. Mostly, they rely on user query, which is generally seen vague and sometimes confusing. User generally makes supplementary efforts to find out from the retrieval results the assets that well fit his context. In order to ease the burden of the user, we fervently believe that user context warrants to be considered as a part of the query while discovering the candidate asset. For that reason, we believe that the primary step towards intelligent retrieval is to provide assets with accurate semantic supporting the matches against user query and context as well.

In such a context, this paper presents an XML-based asset representation model for describing all kinds of software asset that can be reused within software development process. The proposed model provides semantic metadata for describing assets oriented user context in order to build the foundation for semantic reasoning in the retrieval process.

ACKNOWLEDGEMENT

This research was supported by the MKE (Ministry of Knowledge Economy), Korea, under the ITRC (Information Technology Research Center) support program supervised by the IITA (Institute of Information Technology Advancement) (IITA-2008-(C1090-0801-0032)).

REFERENCES

[1] C. W. Krueger, Software reuse, ACM Computing Surveys (CSUR), v.24 n.2, p.131-183, June 1992.

[2] P. Vitharana, and *al*., Knowledge-based repository scheme for storing and retrieving business components: a theoretical design and an empirical analysis, IEEE Transactions on Software Engineering, pp. 649-664, 2003.

[3] G.Elias, and *al.*, X-ARM: An Asset Representation Model for Component Repository Systems, SAC'06, April 23-27, 2006, Dijon, France.

[4] A. Mili, R. Mili, and R. Mittermeir, Reuse-based Software Engineering: Techniques, Organizations and Controls, ISBN 0-471-39819-5, 2001.

[5] OMG, Reusable Asset Specification- version 2.2, 2005, http://www.omg.org/docs/ptc/05-04-02.pdf.

[6] F. Ali, and W. Du, Toward reuse of object-oriented software design models, Information & Software Technology, pp. 499-517, 2004.

[7] V. Sugumaran, and V.C Storey, A semantic-based approach to component retrieval, SIGMIS Database, pp. 8-24, 2003.

[8] Y. Ye and G. Fisher, Context-aware browsing of large component repositories, IEEE Computer Society, pp. 99-106, 2001.

[9] R. Holmes and G. C. Murphey, Using structural context to recommend source examples, ACM, pp. 117-125. 2005.

[10] V.C. Storey, V. Sugumaran, "The Role of User Profiles in Context-Aware Query Processing for the semantic Web, Springer, Berlin, pp.51-63, 2004.

[11] G..D. Abowd,, A.K Dey,. A. Wood, Applying Dynamic Integration as a Software Infrastructure for Context-Aware Computing, Technical Report GIT-GVU-97-18, GVU Center, Georgia Institute of Technology, September 1997.

[12] D. Luo and C. Hseih, A Context-Aware Approach Enhancing XML Semantics Integration, IEEE Conference on System Sciences, 2008.

[13] P. Vajirkar, S. Singh, Y. Lee, Context-Aware Data Mining Framework for Wireless Medical Application, *Lecture Notes in Computer Science* (LNCS), Volume 2736, Springer-Verlag, ISBN 3-540-40806-1, pp. 381 – 391, 2003.

[14] S. Park and al., Extending Reusable Asset Specification to improve Software Reuse, SAC'07, March 11-15, 2007.