Journal of Theoretical and Applied Information Technology
31st January 2018. Vol.96. No 2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

IMPROVING SOFTWARE ARTIFACTS REUSABILITY BASED ON CONTEXT-AWARE REUSE TECHNIQUE

[1]DOOHWAN KIM, [2]JANG-EUI HONG

[1,2]Department of Computer Sciences, Chungbuk National University, Cheongju, 28644, Rep. of Korea

E-mail: [1]dhkim@selab.cbnu.ac.kr, [2]jehong@chungbuk.ac.kr

ABSTRACT

Reusing software artifacts is not limited to only source codes, but also various artifacts which are created in software projects. However, traditional artifacts reuse approach has some problems of difficulties to find an appropriate artifact and to choose suitable terms as search keywords. To reduce the difficulties and enhance reusability of software artifacts, this paper presents some novel techniques based on context-aware reuse; which are context-based automatic keyword generation, ontology-based keyword extension, context-aware retrieval and learning-based result visualization. To realize the techniques, we firstly defined microComponent with the smaller unit (section) of a document as the reusable unit. And then we focus on to how can we retrieve and reuse the microComponents with the concept of context-aware. Our technique can precisely control the reusable unit, and enhance reusability of software artifacts based on the microComponents.

Keywords: *Software Artifact, Reusability, Context-aware Reuse, microComponent, Reuse Framework.*

1. INTRODUCTION

Software artifacts have been recognized as one of the most important factors in successful software development due to provide the visibility of software development [1]. Therefore, the artifacts must be developed systematically to provide high qualities such like consistency, completeness, and traceability. Artifact reuse, especially the reuse of technical document has been known as a strategy to improve the qualities of the document to be created because it allows you to create documents quickly and reliably using already proven artifacts [2].

However, the conventional component-based reuse approach has some problems such that it is difficult to find components that we want to reuse, and also it is not easy to accept without modification to the found components [3,4]. To finding the reusable components based on keywords simply is difficult to find the desired suitable components to reuse in the development of software artifacts. Especially in the conventional reuse of technical document, it is more difficult to find out and to reach the specific contents or figure objects due to file-based reuse approach [5]. Additionally, there are some inconveniences when we refer back and forth several documents to find out the contents to be reused. Therefore, a novel approach for managing reusable documents is required, just as the need to address the existing problems of component-based reuse.

Various techniques for finding reusable artifacts have been proposed; representatively, the search techniques based on keywords, the navigating techniques based on domain facets [6], and the context-based search techniques based on subject word and thesaurus [7,8,9]. The representative studies of the context-based retrieval techniques have been proposed by E. Cruz [7] and S. Nesic [8]. Cruz made to enrich their studies by defining specific contextual information with respect to reusing artifacts, so that the reuse as possible to find more suitable components. Also Nesic [8] had studied on the context-based retrieval technique with intension to reuse technical documents. He assigned tags to every contents of a document and the document was retrieved based on the tags. If a user defines a keyword and the related ontology to retrieve a document, then his search technique allows retrieving the partial content of the document defined by the tag. However, these methods also did not solve the existing problems in the reuse of the documents using file as the search and result unit.

In order to resolve such problems, we propose a retrieval technique based on the content (we called this content as a section) of a document rather than a document in file unit. This paper defines

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

microComponent (mC) which is a retrieval unit for reusable technical documents, and a separated section of the document. Also this paper proposes the component repository architecture to support context-based retrieval based on the mC. Our proposing technique can easily find the content of the document the user wants, directly reach to a specific part of the document, and can support fast and correct understanding for the content due to the small retrieval unit (i.e., section).

This paper is organized as follows. Section 2 surveys the related work for artifact reuse, and Section 3 explains about the microComponent-based reuse framework presented in this paper. Section 4 proposes the techniques of context-based retrieval in our reuse framework. Section 5 verifies the usefulness of our technique by case study. The last section describes the conclusions and future research work.

2. RELATED WORK

Context-aware reuse has been studied by many of researchers steadily in various aspects and subjects [7-18]. The main purpose of these studies is to achieve more efficient, useful and convenient search for artifacts reuse. Among those a lot of studies, we will introduce certain representative studies in this section.

E. Cruz [7] proposed a context based software asset retrieval system. The system has two repositories working together for assets retrieval: The one manages software assets, while another one manages the contextual elements. The authors also classified user's roles to manage and provide contextual information by the role. For example, a keyword can be understood with different context or different meaning for software tester and configuration manager. Even though this system can provide more useful assets for each role of users, their proposing system manages not the contents, but the files as the unit of reusable assets. This make hard to utilize the context information of smaller units which are the components of a file or document (for example sections of a documents or methods from a classes).

S. Nesic [8] focused on the lack of semantical information and interoperability in document management from traditional desktop system. As a solution and alternation, the author built the semantic document management system having the concept of semantic web. The contextual information of documents is managed by annotating semantical information to the documents. The author also used three distinguished ontologies; document ontology for managing the smaller units of documents by classification, annotation ontology for managing annotation in contextual aspect and change ontology for tracking the change of documents. By using these ontologies, the author makes enable to semantical and contextual retrieval and reuse of documents efficiently. The strength of this study is annotating the semantical information without any modification of original copy. However, the system did not consider the structural information (i.e., contents structure) of documents for contextual information even though the proposed system manages smaller units of the documents for just only classification by semantical information.

A. M. Khattak and colleagues perceived retrieving digital documents to be difficult problem because of their huge size and amount [10]. Also, the authors claimed the inefficiency of keyword-based search in the situation because user often fails to search by keywords when the words are inappropriate, even though they have exactly same meaning. Therefore, the authors provided a solution which is supporting semantic-based search based on ontology. The authors used not only the ontology, but also knowledge base and inference engine to enhance the efficiency and reasonability of search result. The proposed engine showed up the improved precision of search results compared with other previous works. However, they did not concern the structural context of the documents. Moreover, their main target of the retrieval is general documents which are written in only letters. As the result, their system cannot fully support software development activity because there are so many types of artifacts like drawings, diagrams, figures and expression.

There are numerous studies supporting context-based reuse or recommendation like [11, 12]. However, almost existing studies have a little bit difference focus or purpose with our research interests. Our study focus on managing and utilizing a variety of information such as the structural information of documents and the ontology related with keywords, to support context-aware reuse in all aspects of reuse activities.

Journal of Theoretical and Applied Information Technology
31st January 2018. Vol.96. No 2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

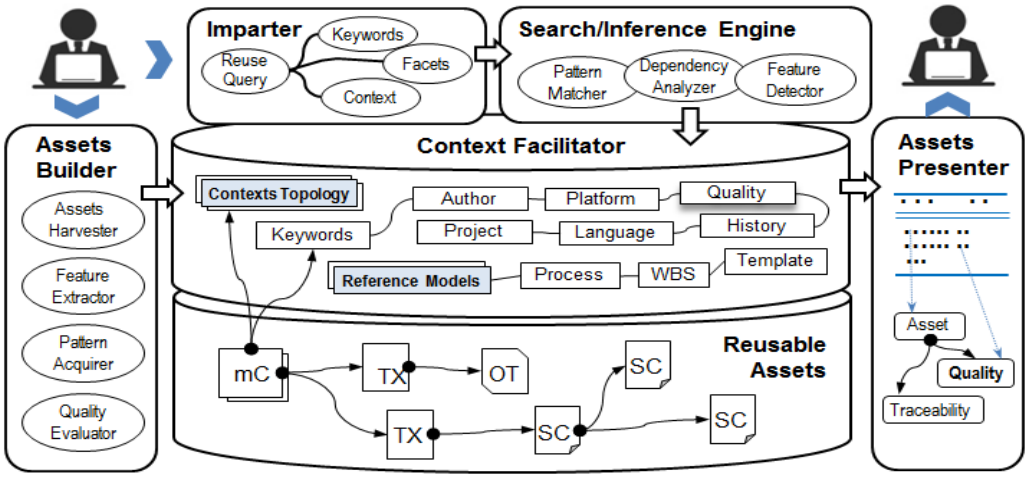


Fig. 1. The conceptual structure of microComponent reuse framework

3. microComponent REUSE FRAMEWORK

3.1 Structure of mCRF

We define the mCRF (microComponent Reuse Framework) to managing, retrieving, and reusing the microComponents. The overall structure of the mCRF is shown in Figure 1.

The mCRF, as shown in Figure 1 represents the overall structure for supporting the reuse of technical documents which are created during software development process. The brief explanations for the functions of mCRF are as follows:

- Assets Builder (AB) has those functions of extracting mCs from reusable document, defining basic information of the extracted mC, and transfers the mCs and their information to Contextual Facilitator.
- Contextual Facilitator (CF) registers the basic information to the context topology catalog, and stores the mCs to reusable asset storage.
- Reuse Imparter (RI) listen to user's request to reuse mC, and automatically generate a query to retrieve the adequate mCs.
- Search Engine (SE) retrieves the context catalog, and selects the candidate mCs according to the requested query.
- Assets Presenter (AP) visualizes the search results in a form of pre-defined styles.

3.2 Meta Information for Reuse

One of the contextual information managed by CF is the catalog of reference models. The reference models include standard document templates which were represented with XML [9]. We extend XML elements to explicitly define the boundary of sections of a document, and also define the document template with the extended XML. The standard documents were based on the specification of the ISO/IEC 12207 [19] and MIL-STD-498: DID (Data Item Description) [20]. An example of the document template, "Software Requirement Description" is shown in Figure 2.

```
<TD title="Software Requirement Description"
version =" " id=" " xmlns:xsi="http://..".>
<mC title="1. Scope">
    <mCsub title="1.1 Document Overview">
</mCsub>
    <mCsub title="1.2 System Ovrview">
    </mCsub >
    <mCsub title="1.3 Terms and Abbreviation">
    </mCsub >
</mC>
    :
<mC title="2. Requirements">
    <mCsub title="2.1 Functional Requirements">
    </mCsub>
    <mCsub title="2.2 Non-Functional Requirements">
    </mCsub>
    <mCsub title="2.3 Interface Requirements">
    </mCsub>
        :
<mC title="5. Other Considering Issues">
</mC>
<mC title="6. Appendix">
</mC>
</TD>
```

Fig 2. An example of standard document template which is represented with extended XML.

Journal of Theoretical and Applied Information Technology
31st January 2018. Vol.96. No 2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

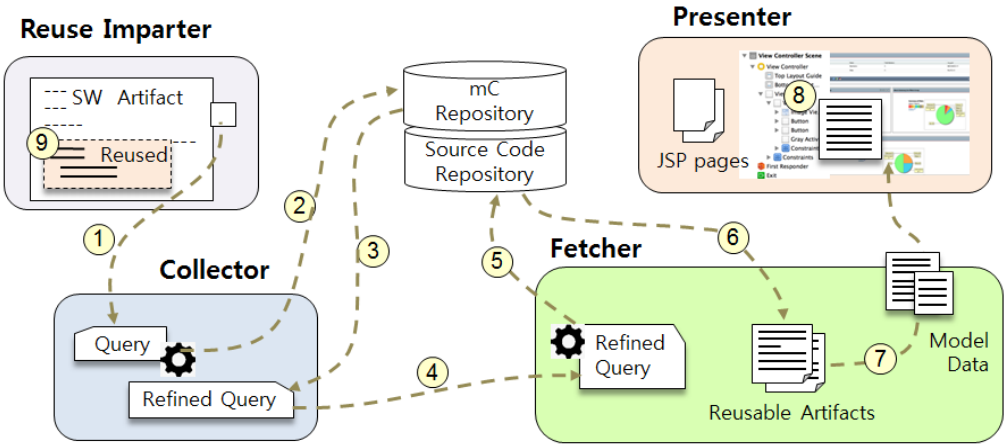


Fig. 3. Imparter-Collector-Fetcher-Presenter Pattern which proposed to support our reuse framework

The XML element <TD> represents a document which contains all contents and the elements <mC> and <mCsub> denotes the section and subsection of a document, respectively. Users can create a technical document using an instance of the standard document template. The completed document is separated into a set of mCs, and then registered to the reusable assets storage by the separated mC. The management information for the mC is given in Table 1.

Table 1. Meta information for mC specification

Class	Attributes
Identification	Identifier, mC name, Type of mC
Project	Project_ID, Developer, Date of creation, Brief description
IntraConnection	Previous_mC, next_mC
InterConnection	Previous_doc, Next_doc
History	Num_Used, Reuse_date, User_review
Quality	[Text mC] Number of pages, Structure type, Template compliance. [Source Code] Halstead Complexity, Cyclomatic complexity, Power consume
Constraints	Where to reuse, Customization, Criteria
Thesaurus	Thesaurus, Ontological link
CPR	Characteristics / Pattern / Recommendation information of mC

The class "IntraConnection" as listed in Table 1 is a pointer that indicates the connection relation between mCs constituting a document; the class "InterConnection" is a pointer to provide the traceability between documents. Also the class "Quality" is the measured quality values for an mC; the "Constraint" class defines the instructions or guidance for mC reuse. The "Thesaurus" is ontology information class to support the context-based retrieval; the "CPR" is a class that defines the parameters of the algorithm used for the mC retrieval. All of the information except “History” as listed in Table 1 is defined or identified at the time point of registering the mCs into asset storage.

4. CONTEXT-AWARE REUSE

4.1 Process View of mCRF Repository

Our mC repository system is configured with the features of mC management, query processing, search engine, and result display in overall user’s perspective. In order to support these features, we consider the process view of assets reuse within mCRF, and develop an architecture pattern, ICFP (Imparter-Collector-Fetcher-Presenter), as shown in Figure 3.

As shown in Figure 3, when the user clicks the reuse request button during the writing of a document using a word processor, the RI recognizes the tag data from the location of the cursor. The tag data is made up with the combination of TD title, mC title and mCsub title. The RI sends the tag data to the Collector (①). The Collector selects the related thesaurus from ontology using the tag data (② and ③) to generate the final query. The Fetcher uses the query (④) to retrieve the appropriate mCs (⑤) form the reusable assets, and creates the list of candidate mCs (⑥).

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

The Fetcher pushes the query results to the Presenter depending on the model data that describes how to represent the mC (⑦), that is to say, the meta-information specified in Table 1. The presenter visualizes the query results to provide the information (model data) to user (⑧). After that, the user can import the mC into the current creating document by clicking the desired mC (⑨).

4.2 Context-Aware Reuse Approach

When trying to reuse of the reusable assets based on the ICFP pattern, this paper utilizes the following three context information in the reuse process.

4.2.1 Context-aware query generation

The user does not enter any keyword when he/she requests a reuse. The RI recognizes the structural information of the current document being created, and automatically generates keyword for a user query. There are two major phases to generate the keywords: The first phase is for generating the keywords within a document, and the second phase is for collecting the information of the project characteristics and adding the collected information to the keyword list.

The keyword generated in the first phase is a combination of subsection title, section title, and document title which are tagged. The detail steps to generate a context-aware query for user's reuse request are follows;

1) The RI recognizes the current position of cursor within creating document.
2) The RI finds the XML tag of the element <mCsub>.
3) Adds the title of the element <mCsub> to keyword list.
4) The RI traverse backwardly (i.e., upward) XML tags of current document until it encounters the element <mC> .
5) Adds the title of the element <mC> to keyword list.
6) And the RI continuous the traversal of XML tags of current document until it encounters the element <TD> .
7) Finally adds the title of the element <TD> to keyword list.
8) The RI generates the initial keyword list for the user's query when the encountered tag is <TD>.

Based on the above steps, we can obtain the initial keyword list (i.e., this is a set of tag data collected from document template) like below when this process was applied to the document represented in Figure 2, and the cursor was positioned at section 2.2.

The initial keyword =
{Non-Functional Requirements,
Requirements, Software Requirements
Description}

By the second phase, the initial keyword list will be extended with the project information. This project information comes from the project definition portfolio. In general, the project portfolio is made with project information when a project is created. The detail information of project portfolio is listed in Table 2.

Table 2. The detail information that is defined to create the project portfolio

Information	Description
Project type	It represents the type of a project, which is one of {new development, maintenance (simple modification), upgrades (add new functionality) }
Project Name	It describes the name of a project in short phrase form. Almost project name contains main terms to reflect the functionality of target application.
Project Domain	Domain is a name of category which represents the application areas. For example, intelligence system, embedded system, transaction system, etc.
Project Manager	The name of a person who is responsible for a project.
Project Period	The start and end date of a project

The RI gets the project information form the portfolio. The required information to retrieve appropriate components is project name and project domain. After the second phase, the base keyword list will be formed with several terms, as like below. The below keyword list is an example from generated from our current project entitled to "Context-aware artifacts retrieval system" as one of intelligence systems.

The base keyword = {Non-Functional
Requirements, Requirements,
Software Requirements Description,
Context-aware artifacts retrieval system,

Journal of Theoretical and Applied Information Technology
31st January 2018. Vol.96. No 2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Intelligence systems}

4.2.2 Ontology-based query extension

Using the base keyword list generated by the user request, the Collector automatically generates the final keyword list using the relatum of the elements of the base keyword list. The Collector gathers the thesaurus from the repository, and extends the keyword list to make search query. This extension is intended to increase the precision of the context-based retrieval with zooming in the search scope.

The purpose of using ontology for the keyword extension is making a query with similar or related keywords. Certain keywords can be omitted or unobserved depending on the knowledge or vocabulary of the software engineer (i.e., user). This omission can sometimes fail to obtain the proper search results. Therefore, software engineer who wants to reuse mCs can easily catch other keywords which are having similar or same concept with their own selected keywords. Figure 4 shows the procedure of this keyword extension based on ontology.

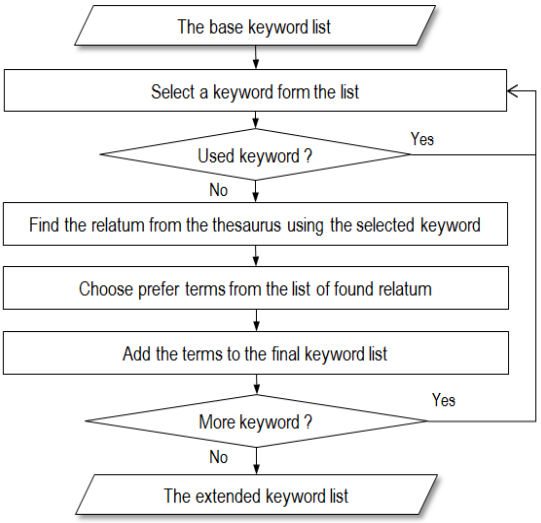


Fig. 4. The procedure to generate the final keyword list using ontology

As shown in Figure 4, the Collector selects a word from the base keyword list at the first step. The Collector finds and lists the relatum from the thesaurus using the selected word. After listing the relatum, software engineer can interfere in generating process of the extended keyword. The engineer can choose the relatum with marking at each term if he/she wants to restrict the extension of the keyword. The concept of this intervention is showed in Figure 5.

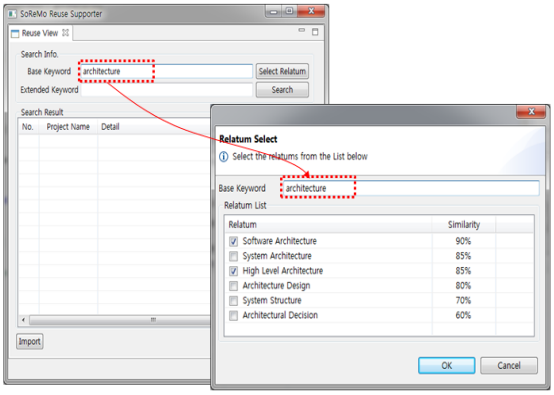


Fig. 5. The sample screen to restrict the relatum scope by user's intervention

Keyword extension is started from a base keyword which is given from the Section 4.2.1. This keyword is key or clue for extension because user wants to retrieve mCs which include one of the base keywords or some other keywords having same or similar concept. The Collector will receive the keyword and send it to ontology for extension. After that, ontology will recommend certain relatum (i.e., other keywords having same concepts). The ontology is built in mC repository from Figure 2. The terms came from the base keyword list can be removed from the extended list when software engineer did not mark to all the relatum. From the above example, the final extended keyword list is as below;

The extended keyword = {Non-Functional Requirements, Quality requirements, Software Requirements Description, Context-aware artifacts retrieval system, Contents-aware retrieval system}

Therefore, the search query will be generated to find the mCs like that "Search mCs from repository with the terms, non-functional or quality requirements in the software requirements description for context-aware artifacts retrieval system or contents-aware retrieval system."

Journal of Theoretical and Applied Information Technology
31st January 2018. Vol.96. No 2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

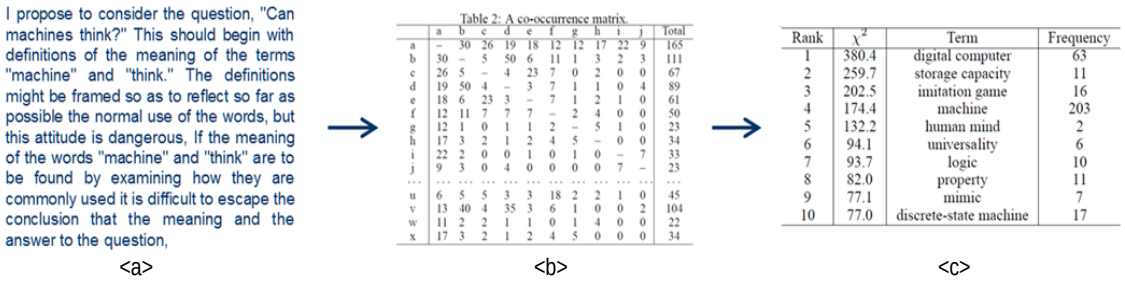


Fig. 6. The process to extract contextual information (i.e., important keywords) from original reusable text; <a> is original text, <b> is Co-occurrence matrix, and <c> is Keyword extract with rank

4.2.3 Context-aware retrieval

Using the created search query, the Fetcher recommends the candidate mCs from reusable assets repository. This recommendation process is composed of two phases; the first phase is a process that extracts the contextual information from stored mCs, and then compares it with the final extended keywords. The second phase is to recommend the candidate mCs to the user by using the comparison result.

At first phase, the Fetcher extracts the keywords from the original text, and then measures the number of times that each keyword has appeared in conjunction with other keywords. Based on the number, it calculates the probability that two keywords emerged together within the text. This probability is used for determining the important keywords compared to the probability of the emergence of the keyword alone.

This keyword extraction process can propose the important keywords with high probability for representing main context of an mC. This method had been suggested in the study of Matsuo et. al. [21], and is used as an approach to discover important keywords from a document, as shown in Figure 6.

The Fetcher recommends a list of candidate mCs as the result of the search query if the important keywords were selected for a single document (i.e., an mC). For this recommendation, the contextual information for an mC and the relationship information among mCs must be controlled in the repository. The recommendation is made through the main two processes. The first one is to recommend the mCs that have exact matching with the context information or high degree of similarity. The second is to determine the rankings of the recommended mCs using the relationship attributes between them. The relationship attribute means the property of an mC that may be used in combination when a particular mC is reused.

The similarity between mCs is determined as the frequency of keyword appearance within mCs both, by comparing the important keywords of the stored mC and the extended keywords used for the query request. The relationship attributes between mCs are achieved by considering the types of relationships, as below:

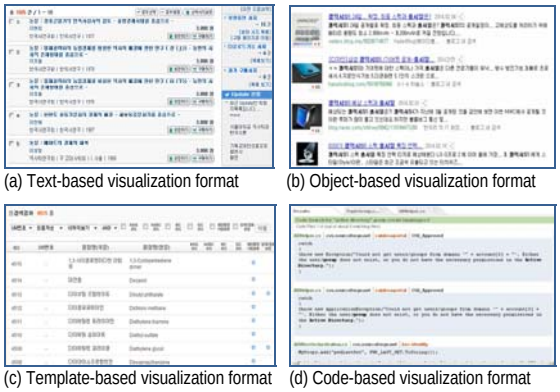
- Dependency Relationship: It refers to the dependency between mCs. The dependency occurs in cases of that an mC directly uses the other mC, or an mC is included in the other mC in the hierarchy structure of a document.
- Familiar Relationship: Although the dependency between mCs does not exist, they can be used in combination with a lot.
- Intersection (similarity) Relationship: It is a relationship that represents the similarity (intersection) of the mCs. This is defined based on the detailed properties of the type rather than just type of mC. If the similarity of two mCs is matched completely, they can be interchangeable.
- Inclusion Relationship: This is an attribute indicating how much an mC includes the contents of the other mC. For example, the mC “A” includes the mC “B” in case of that the “A” has 100% similarity to the “B,” while the “B” has 50% similarity to the “A.”
- Interchangeable Relationship: If two mCs has high contextual similarity, but it does not appear to be reused in both, then the two mCs are interchangeable. This is for the case of having the same meaning expressed in different terms, or for the case of representing with different models for the same content.
- Complementary Relationship: The two mCs are complementary to each other if they are often tending to be more used together than a pre-defined threshold when the two mC have similar context.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

To recommend reusable candidates to user, some of mCs are firstly selected by the similarity analysis. The final candidates of mCs will be listed with filtering out the selected mCs under the consideration of the relationship attributes among them.

4.2.4 Learning-based result visualization

When search results obtained by the Fetcher, the Presenter can visualize the result in a variety of formats. The visualization can be different depending on the type of mC. Also the Presenter learns the user's behaviors for changing the visualization format by the type of mC, to utilize them in the selection of visualization format. For the different types of mC, we defined four kinds of visualization format as shown in Figure 7.



(a) Text-based visualization format (b) Object-based visualization format

(c) Template-based visualization format (d) Code-based visualization format

Fig. 7. Examples of four kinds of visualization formats by mC type

The visualization will be provided one from four formats in Figure 6. Of course, the user can change the visualization format if he/she wants. However, the format will be automatically decided and provided based on two criteria; the type of mC and user preference. In aspect of the type of mC, the mC can be classified into four categories; (a) text, (b) object, (c) reference templates and (d) source codes. The text category means text based contents including minutes, technical reports and documents which are created during software project process. The object category represents figures included within the contents. The figures include JPG/GIF-typed pictures, UML diagrams and vector drawings which are managed in repository. The templates category means reusable assets or reference models which were registered into repository for setting up a project. The examples of the template are waterfall model for software process, WBS (Work Breakdown Structure) for prototyping approach and standard document templates. These templates were defined using mark-up language like XML. The code category means the source codes that are managed in the repository. As shown in Figure 7, each of visualization formats is responsible to each type of mCs. Therefore, the Presenter will provide a specific visualization format based on the type of mC.

However, the type of mCs is not only one criterion to decide the visualization format because certain users prefer to see and check search results with other specific format of visualization. To apply this preference, the Presenter will manage the history about which kind of visualization format should be selected by user, with considering the mC type in search result. Actually that history information will be managed indirectly by giving to the Fetcher. Therefore, the Presenter can learn the information about preference for each user.

5. CASE STUDY

In order to verify the performance of our context-aware reuse technique, we performed the experiment with the goal of checking the usefulness of search results.

5.1 Experimental Design

Experiments were carried out on an ongoing reuse system development project. In this project, we tried to apply our reuse technique proposed in this paper to create RDD (requirements definition description) and SDD (software design description) documents. Table 3 summarizes the time points of reuse trials attempted in the document creation process.

Table 3. Application of context-aware reuse technique

Exp.ID	Doc. Name	Contents for Reuse
E1	RDD	System Overview
E2		Non-Functional Requirements
E3		Interface Requirements
E4		Required Computer Resources
E5	SDD	System overview
E6		Overall Architecture design
E7		Interface Design
E8		Database Design
D9		Traceability Matrix

To check the usefulness of our context-aware reuse technique, the number of mCs managed in the assets repository totally covers 4360 mCs extracted from a total of 450 documents.

5.2 Experimental Scenario

The scenarios for checking the usefulness of our technique are as follows.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

(1) The user first starts to create the document by using the template of the document to create the RDD document.
(2) Simply click the "Reuse" button at a certain position – i.e., a specific subsection - of the document.
(3) A list of candidate components for reuse is presented from the system. The proposed list shows only the candidate components that have match results of at least 70% in the query condition for the search.
(4) The user clicks the link of a candidate component to take its details, which is ranked in the top of the search result, and then decides whether or not to reuse it.
(5) Import the reusable component at the position of the current creating document.

5.3 Experiment Results

We examined the experimental results from two points; Accuracy of search results and superiority of search results.

Accuracy of search results: The context-aware reuse technique presented in this paper provides automatic query generations, contextual information - based retrieval that considers components features and appearance patterns. Using this technique, we examined the ranking of the reused component by user from the search results. The results are summarized in Table 4. For each experiment identifier (E_ID), RFR (Reuse in the First Retrieval Result), ROR (Ranking of Reused Component within the result), and RSR (Reuse in the Second Retrieval Result) are summarized.

In case of the identifier E7 in Table 4, the search result shows 30 or more reuse candidates as the initial result, but the user tries re-search with changed query condition after confirming about the details of ten candidates only. This is because the user cannot find appropriate design content that matches the interface requirements defined in the target project from the first search result as well as because the user hastily decides that the search result is not appropriate.

Table 4. Ranking of reused components within search results

E_ID	RFR	ROR	RSR	ROR
E1	Y	2	-	-
E2	Y	1	-	-
E3	Y	3	-	-
E4	Y	1	-	-
E5	Y	3	-	-
E6	Y	4	-	-
E7	N	-	Y	15
E8	Y	2	-	-
E9	Y	1	-	-

Superiority of search results: In order to verify the superiority of the search results, we performed a comparison between the existing user-selected keyword-based search technique and our context-aware retrieval technique. At this time, the keyword selected by the user is a combination of the title of the subsection in which the cursor is located and the name of the target system. And the result is provided in unit of files. Figure 8 shows the comparison result for the ranking of the components reused by users within the search results provided by the two techniques.

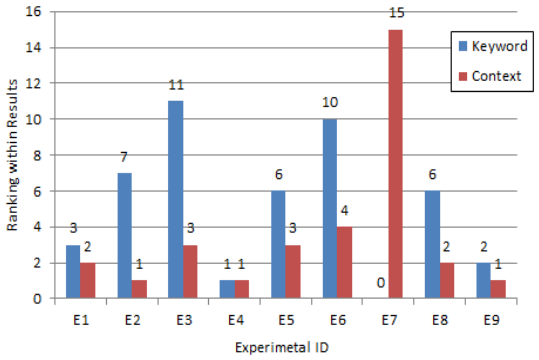


Fig. 8. Reusability Comparison between simple keyword search and context-aware search results

As shown in Figure 8, it can be seen that the context-based retrieval technique is very good for presenting the reusable components desired by the user as search results. In the cases of E1 and E4, there is no significant difference between the search keywords selected by user and the automatically generated search condition. The value of user's keyword search in E7 is 0 means that any component was not reused from the search result.

5.4 Threats to Validity

In contrast to conventional search, our technique is more applicable for system development project through R&D (Research and Development) approach. For R&D-based system development, reusable components should be able to provide specific information that users want, and should have more content and/or technical similarity than conceptual similarity.

From this point of view, there is a need for precise provisioning of the granularity of reusable

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

components. Our research result is sufficient to support this demand. Nonetheless, we investigate the following issues that could be considered in the experimental process.

The one issue is on the response time to present search results. The user-selected keyword-based search technique may be superior to the context-aware retrieval in terms of response time to present search results. However, the rate of response time to the search results depends on the size of search space. Therefore, our context-aware retrieval is surely superior when considering the time until the final decision to reuse a component.

The other issue is to the ranking of the candidate components listed in the search results. Is the rank on search results meaningful? The user may not click on the first component listed in the search result. Sometimes users look through the search results as a whole, and click on any component they think about, regardless of the listed order of candidate components. Nevertheless, it is important to arrange the candidate component that best matches the search criteria at the top of the search results because the user looks through the list from top to bottom.

Additionally, we can also consider language-dependency as one of reuse barriers. In general search engines, the language of retrieval results may depend upon the language of input keywords. If the keywords were written in English, the results will be represented with English. If it is Korean, the results will be Korean. However, one of the ways to solve the linguistic barrier can be somewhat complicated, but it can be solved by connecting the lexicons between different languages with a set of ontologies, or by using an automatic translation system.

6. CONCLUSION

In this paper, we propose a novel technique for context-aware reuse approach. In order to develop the technique, we define a reusable unit as mC (microComponent) which is defined with a section of a document, and also propose a framework to support the context-based retrieval for the mCs. Our proposed technique can improve the usefulness for the retrieval by suggesting the pattern, ICFP for the mC repository in our reuse framework, especially in the perspective of the users.

Now, we finished the comparative analysis of our techniques with performing some case studies. And we have confidence to the superiority of our proposed technique when comparing ours with conventional keywords-based retrieval techniques.

7. FUTURE RESEARCH DIRECTIONS

Even though our proposed technique can give better retrieval results than conventional techniques for reusable components, our mCRF may have some improving issues to support more efficient context-aware reuse.

The first further work is to enhance the context-aware retrievals with developing new technique at different aspects like dynamic recommendation and learning-based retrieval. This is to reflect the users' behaviors for looking through the retrieval result. If a user clicks a reusable candidate from the result, and the candidate was unused or ignored, then the list of the final extended keywords can be modified dynamically with deleting the corresponding relatum from the list. This is possible to provide more suitable results to users concurrently by performing background retrieval (i.e., hidden to the user) of reusable candidates.

Other research direction is on the application of visualization technique toward big data analytics. Lots of data scientists would like to gain useful information from big data. However, sometimes the information gaining from big data is often not used by them even though the data used for the analysis was appropriate and valuable population. Therefore, it is important to provide informatively visualization format to data scientist for the retrieval results from big data.

ACKNOWLEDGEMENT

This article is a revised and expanded version of a paper entitled "A microComponent-based Reuse technique for Reusing Software Artifacts" presented at The 11th Asia Pacific International Conference on Information Science and Technology (APIC-IST 2016) held on June 26 – 29, 2016 at Hokkaido, Japan. This research was supported by a research grant from National Research Foundation, funded by the Ministry of Science, ICT and Future Planning, Korea (NRF-2014M3C4A7030505). Corresponding author: Jang-Eui Hong

REFERENCES:

[1] B.W. Weide, W.F. Ogden, S.H. Zweben, "Advance in Computers: Reusable Software Components," *Academic Press*, 1991, pp. 1-65.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

[2] William W. Agresti, "Software Reuse: Developers' Experiences and Perceptions," *Journal of Software Engineering and Applications*, 2011, pp. 48-58.

[3] William B. Frakes and Kyo Kang, "Software Reuse Research: Status and Future," *IEEE Transaction on Software Engineering*, Vol.31, No.7, 2005, pp. 529-536.

[4] Douglas C. Schmidt, "Why Software Reuse has Failed and How to Make It Work for You," *C++ Report magazine*, 1999.

[5] E. Guerrieri, "Software Document Reuse with XML," *Proceedings of the ICSR*, 1998, pp. 246-254.

[6] M. J. Henry, et. al., "MultiFacet: A Faceted Interface for Browsing Large Multimedia Collections," *IEEE International Symposium on Multimedia*, 2013, pp. 347-350.

[7] E. Cruz, et.al., "Modeling Context in Software Reuse," International Workshop on Modeling and Reasoning in Context, Computer Science Research Report, Vol. 112, 2007, pp. 89-102

[8] Sasa Nesic, "Semantic Document Model to Enhance Data and Knowledge Interoperability," *Annals of Information Systems*, Vol. 6, 2009, pp. 135-160.

[9] ISO/IEC 29500-1:2008, "Information technology - Document description and processing languages - Office Open XML File Formats - Part 1: Fundamentals and Markup Language Reference," 2008.

[10] A. M. Khattak, N. Ahmad, J. Mustafa, et. al., "Context-Aware Search in Dynamic Repositories of Digital Documents," *in Proceedings on IEEE 16th International Conference on Computational Science and Engineering (CSE) 2013*, 2013, pp. 338-345.

[11] Lars Heinemann, "Facilitating Reuse in Model-Based Development with Context-Dependent Element Recommendations," *in Proceedings on RSSE 2012*, 2012, pp. 16-20.

[12] R. Holmes and G. C. Murphy, "Using structural context to recommend source code examples," *in Proceedings of 27th International Conference on Software Engineering*, 2005, pp. 117-125.

[13] G. Aravanis, A. Bucur and M. Pechenizkiy, "Hippocrates: A Context-aware, Collaboration Enabling Search Tool," *in Proceedings on IEEE 28th International Symposium on Computer-Based Medical System (CBMS)*, 2015, pp. 320-325.

[14] B. Bislimovska, A. Bozzon, M. Brambilla, and P. Fraternali, "Content-based search of model repositories with graph matching techniques," *in Proceedings of the 3rd International Workshop on Search-Driven Development: User, Infrastructure, Tools, and Evaluation*, 2011, pp. 5-8.

[15] X. Zhu, X. Pan, and S. Wang, "Approaches to Context-Based Knowledge Share and Reuse," *in Proceedings on 4th International Conference on Fuzzy Systems and Knowledge Discovery*, 2007, pp. 741-746.

[16] K. S. Mule and A. Waghmare, "Context based information retrieval based on ontological concepts," *in Proceedings of International Conference on Information Processing*, 2015, pp. 491-495.

[17] J. Cordeiro, B. Antunes and P. Gomes, "Context-based recommendation to support problem solving in software development," *in Proceedings of 2012 Third International Workshop on Recommendation Systems for Software Engineering (RSSE)*, 2012, pp. 85-89.

[18] R. N. Deborah and S. Chitrakala, "A context-aware approach based web service recommendation," *in Proceedings of 2016 2nd International Conference on Advances in Electrical, Electronics, Information, Communication and Bio-Information (AEEICB)*, 2016, pp. 188-192.

[19] ISO/IEC 12207:2008, "Software Life Cycle Processes," 2008.

[20] MIL-STD-498, "Software Development and Documentation," 1994.

[21] Y. Matsuo and M. Ishizuka, "Keyword Extraction from a Single Document Using Word Co-Occurrence Statistical Information," *International Journal on Artificial Intelligence Tools*, vol. 13, no. 1, 2004, pp. 157–169.