CASE-BASED REASONONG TECHNIQUES V.S. OTHER TECHNIQUES FOR RETRIEVING COMPONENTS

Bisera Dugalikj
Faculty of Computer Science and Engineering
Skopje, Macedonia

Goran Velinov
Faculty of Computer Science and Engineering
Skopje, Macedonia

ABSTRACT

Component retrieving methods have become essential in the process of finding the adequate component as a result to the constant increasing of the number of components that are presented on the market and the often difficulties in finding and recognizing the right component to match with the requirements of the user. In this paper we intend to identify and compare some of the better practices currently available in the process of retrieving components. Component-based software engineering is an area that has a constant progress and growth and it also has a huge expansion in production of components from many different vendors and not just from component libraries so the number of available components is considerably increasing. As a result, there are many component retrieval methods and each of them has their specific approach to the subject. This research is focused on several component retrieval methods including: traditional information retrieval like free-text-based documents, XML retrieval, pre-enumerated vocabulary method, signature matching method, behavior-based retrieval method, faceted selection and conversational component retrieval method (CCRM).

I. INTRODUCTION

The priorities regarding the process of developing software always were the quality of the software and the time it takes to be developed. As a result to the increasing number of clients that are interested to have software that is more dependable and developed more quickly, component based software engineering is elevated among very important and tempting software development approaches and in most of these cases it presents a very suitable choice.

The process of reusing software components leads to a possibility to withstand with complexity and to hand software with greater performances in much shorter time. It is very specific approach because on the one hand it helps to reduce the complexity of the whole process of software development because it offers complete parts with pre-defined behavior that are ready to be implemented while on the other hand it helps in keeping the complexity of the whole system because these parts often perform specific complex tasks. [1]

Along with the rapid development of components and development of component architecture standards that allows components to integrate without complications, comes the difficulty in retrieving the component that best satisfies the requirements of the user. Now for every user searching for components there are a lot of possible choices that answer the requirements and in order to help users find what they are looking for there are several component retrieval methods and each has a specific approach on the subject.

Although every component retrieval method has different approach, according to some analysis, most of them have the same lack when it comes defining the properties of the component the user needs. Most of these approaches start with the perception that every user is able to fully and precisely describe the component that they need. This may not be problem to some of the users but to a large number of them this demand is not rational and presents quite a difficulty. One specific approach that does not have this problem is conversational component retrieval method (CCRM) since it incorporates knowledge-intensive case-based reasoning technologies and conversational case-based reasoning methods.

The rest of this paper is organized as follows. In section 2, there is an overview on component–based development and its significance in the process of developing software; in section 3 there is a review on current best practices among retrieval methods with a short summary about their advantages and disadvantages; in section 4, CBR methods for retrieving components are presented with a focus on conversational component retrieval model (CCRM). In the end in section 5, there is a brief review on the topic and some directions for future work.

II. CURRENT PROGRESS OF COMPONENT BASED DEVELOPMENT

Organizations choose to use components because of the opportunity they offer to encapsulate the already available functionality, and build new services to support business processes. It was also found very useful when it comes to distributed systems and because of that, this kind of

©2013 Faculty of Computer Science and Engineering

technology stimulates further development of component view of application deployment and integration. [2]

The parts that can be reused vary from simple functions to entire applications systems. For some time it was quite troublesome to reuse the "medium-grained" parts of software since they are considered to have notably more functionality than individual objects while on the other hand they are more specific and usually much smaller than application systems. This problem has relatively been solved with standardization that is being promoted by some large software vendors. [3][4]

Component based development with intercommunication standards is considered to bring the new era of software development. Being a combination of high-flying software engineering principles of object technology on one side and user controls on the other makes it a recommended part since it brings the chance for rewrapping same as a chance for new, modular and plug-in elements with flawless interfaces. It has a special role that makes combination of addressing the development of systems as an assembly of components, development of components like reusable pieces, and system maintaining and upgrading by replacing or customizing components. [5]

Component based development can be explained as a practice of describing, implementing and integrating or composing of independent components that are not tightly conjoined into systems. There are many definitions about components based on standards or characteristic of components. According to some unofficial general agreement components can be considered as separate software units that can be integrated in a software system with other components.

Another very common way to understand components is to define them either as conceptually coherent packages of favorable action or as physical, deployable units of software that can perform in well-defined surrounding. Besides these different approaches, component based development focuses on making systems ground on well-defined, independently made pieces. Anyway, this brings the concern regarding the realization in order to develop components as appropriate cohesive functional pieces, while assisting the design and assembly of systems as a combination of recent and earlier developed components. [7] [6]

There is slight distinction in the way components are defined regarding their state. By some definitions components can only be stateless but generally in CBSE it is defined that they can be both stateful and stateless. Stateful components are considered to preserve information from some client's interaction to the next. Otherwise when a component does not carry information within itself, it is called stateless. [8]

Another important thing when it comes to defining the state is the design of the interface that determines precisely if the component is stateful or stateless. For example: when a component interface depends on properties that are set earlier before the component has performed any functions, then the component has an internal state for some period. Later this requires new calls into the component in order to make changes in the properties and to execute the action. On the other hand the component can become more stateless if the interface is designed in the way that a method carries the values necessary to perform the function. This way there is only one call into the component to process the function and there is no information kept within the component while in the previous case there can be as many calls as the component has properties. [9][10]

An important design question concerned by the state of the component is also the scalability of the application. It is related with the implementation details of the component and along with it, the level of statelessness of the component. The sooner the component can be released and the system can get back its resources, the more scalable it is so in case when no information is retained from one method call to next i.e. when the component is stateless, the application scalability is increased. [2][3]

III. COMMON COMPONENT RETRIEVAL METHODS

Most component retrieval methods that are available for the users could be defined from three different perspectives each considering some part of the component retrieval method as presented in Figure1. These three perspectives are: component representation, component query specification that is focused on the requirements of the user, and component retrieval process. [12]

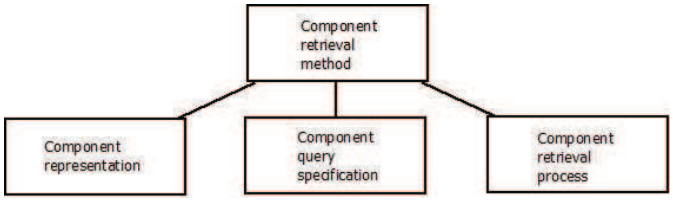

Figure 1: Component retrieval method parts

In cases when components are presented as documents, one of popular methods for retrieving components is XML retrieving method that is guided by the idea to return the most relevant components that answer the query specification. This method is further classified into two sub tasks that are: Content only topics (CO) is simpler version where the user specifies queries as simple text and the search engine returns the most relevant XML components that answer the query concepts and Content and Structure topics (CAS) where the user can define the desired component to be returned by XPath by limiting query concepts to particular XML tags. The primary version of this method has some extensions like a document pivot that scales scores of components by the score of their running article that make significant improvement to

the result. Another extension is to apply Automatic Query Refinement (AQR) algorithms on top of XML component ranking that gives excellent results in the CO track. Since there are many possible combinations of AQR parameters and their variants, the best combination that is to give best results is not found yet. [11]

Another approved component retrieval method that also works with components presented as documents is the free-text-based retrieval method. In this case the query used for retrieval is described using keywords that are checked in all component description documents. The results the user receives are the components that have most matched keywords with the given query. In order to improve the organization and the matching abilities this method implements vector space and indexing technology. Besides many improvements made for this model, until now it has inferior results in accuracy and revocation. The idea to get components that better answer the user's requirements, researchers and practitioners planned to improve the method by using general thesaurus to extend keywords by including also their synonyms and antonyms. Also, in order to get more semantically relevant components by extending initial keywords, they used general domain knowledge. These two improvements improved the retrieval revocation but lowered the retrieval accuracy. [13][14]

Some component retrieval methods use structural information from different perspectives. For example the pre-enumerated vocabulary method uses a set of earlier defined vocabularies to present both components and queries. It brings a form of classification and cataloging that in this case is straightforward and focused only on most significant features of a component. In order to bring positive results, this system must operate within the context of an organizational infrastructure. The system introduces a library system and it includes justification for using classification based on most important features and discusses the need for librarian and organizational support. With this method the user gains on revocation and accuracy in the result but loses when it comes to flexibility in describing the components and specifying component queries. [15]

Behavior-based retrieval method also uses structural information and it is focused on the executable components special characteristics. In this case, queries are represented by a set of input samples and their desired outputs when components take the form of executable codes. The retrieving begins with selecting samples that in the next step are used to execute the components. After the components are executed, those that give the appropriate output are retrieved. The problem with this component retrieval method is that it has low efficiency because it has long execution time and it is designed for executable software components. [16][17]

Faceted selection is a method that presents similar but improved version to the pre-enumerated vocabulary method. With this method there facets that are previously defined dimensions that are used to classify components from different perspectives. [15] All users that are looking for components can find what they are need by searching through the defined categories. Important about this method is that it takes domain knowledge into account when designing the facets. Although this method may seem as the most appropriate of the earlier mentioned, it has some omissions when it comes to forming the facets. In cases when facets are designed too complex, it is hard for designers to classify all components into different categories and it is also hard for users to understand them. In cases when facets are designed too simple or few it is also hard for this method to perform because there will be too many components in final categories and in this case users will have to select further manually. One of the major problems with this method is that in most cases it is very hard to get the adequate components with exact matching because of the universal differences between given components descriptions and component requirements that are later defined by the user. [18]

IV. Case-based reasoning methods for retrieving components

A. Conversational Component Retrieval Method

Compared with the previous two component retrieval methods that are based on CBR, conversational component retrieval model (CCRM) has two advantages. In CCRM a special type of knowledge-intensive CBR method called explanation-driven CBR is adopted to explore components' context-based semantic similarities with a query during the retrieval process besides the query refinement using general thesaurus and domain-ontology. This is possible because with CCRM components are chosen based on syntactical and semantic similarities that is a very promising approach.

Users find it easier to form the search query based on their necessary requirements so they can avoid excluding all adequate components. Because of this users usually get a lot of candidate components since there are a large number of available components. Another advantage with CCRM is that requirements defined by the user are acquired interactively and incrementally. With this method an information gain algorithm is used to provide users with the adequate questions to help define the query interactively and incrementally instead of letting them guess the requirements they should specify in the next steps.

The problem with this method is that the knowledge base is assumed to exist initially including component specific cases and general domain knowledge. The beginning step when the initial knowledge base should be designed assigns a lot of work on the knowledge engineering process.

1) Overview on the method

Conversational component retrieval model (CCRM) is consisted of six parts: knowledge base, knowledge-intensive CBR module, new case generating module, component displaying module, question generating and ranking module, and question displaying module.

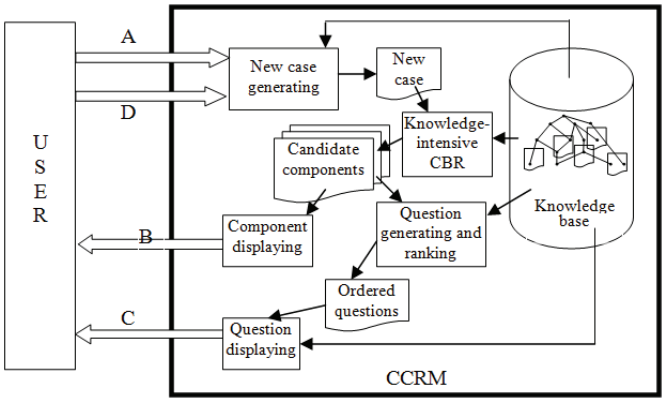

Figure2: Conversational component retrieval model (CCRM) architecture.

Knowledge base is the central part and it collects component-specific knowledge (cases) and general domain knowledge. New cases that come to the system are set up by the new case generating module according to the initial query and the answers to defining questions. Also a threshold is defined at the beginning but it can be adjusted during the execution of the system. When a new case is admitted to the system, the similarities between the new case and stored component cases are calculated by the knowledge-intensive CBR module and it returns the components whose similarities surpass a threshold. Then they are displayed to the candidates ordered by their similarities by the component displaying module. The next part is to identify some unknown questions by the question generating and ranking module and information gain algorithm ranks the possible questions according to how much information it can provide if it has been answered. Questions whose answers can be inferred from the initial query or previously answered questions are filtered out and further reordered according to some constraints inferred from general knowledge. In order to optimize search towards a meaningful answer, the question displaying module chooses the most discriminative question.

As presented in Fig. 3, arrows: A, B, C and D, are interactive processes between users and the system. A – User inputs the beginning query; B – the system returns top matched components; C –system displays question to the user; D - user provides answers to the system.

The rest of the retrieval process is done by the system and it can be defined in several parts. First the user presents the initial query in free-text-based terms. Then the initial query is turned into a new case by the new case generating module. The initial query is turned into standard terms used in the internal system that are formed as a new case. Then the similarities between the new case and stored cases are calculated through combining component specific knowledge, general domain knowledge, and then the components whose similarities are over the threshold are returned to the user. If the user does not find the adequate component, the conversational process is activated. In this step, the question generating and ranking module chooses questions from the candidate components, and orders them by the information they provide. The questions are filtered and reordered using general domain knowledge and the question displaying module selects the most adequate question, and displays it with an answer in a readable format to the user. If the user cannot answer the question, the next adequate question is displayed. A user provides the system with an answer to the displayed question ant the new case generating module sets up a new case by combining the previous new case and the answer. The process of asking questions continues until the user finds adequate component or the system runs out of questions.

V. REVIEW ON THE TOPIC AND FUTURE DEVELOPMENT

It is very natural to use CBR techniques in retrieving components since both CBR and component-based development have the same basic idea, to help and improve software reuse. The current research is focused on improving CCRM by facilitating the question selection. The current available version of the information gain algorithm is knowledge-poor and it lacks knowledge-intensive methods, especially the explanation-driven method, to remove the candidate questions that can be answered by the initial query or previously answered questions, and to adjust the priorities between slots which represent semantic relations, like abstraction, causality, dependency and part-of relations. With further development of this technique is expected it to help identify the most informative question, shorten dialog length, and reduce users' cognitive workload and in that way this method can give significant results when searching for components.

REFERENCES

[1] Ian Sommerville, "Software Engineering, 8th edition", (2006)

[2] Leonardo Mariani, Mauro Pezze, David Willmor, "Generation of Integration Tests for Self-Testing Components"

[3] Alex Homer, "Components and Web Application Architecture"

[4] Alan W. Brown, "Large-Scale, Component-Based Development" (2000)

[5] Alan Cameron Wills, "Component Based Development"

[6] M. Madiajagan, B. Vijayakumar, "Interoperability in Component Based Software development In: World Academy of Science, Engineering and Technology" (2006)

[7] He Jifeng, Xiaoshan Li, Zhiming Liu, “Component-Based Software Engineering - the Need to Link Methods and their Theories” (2005)

[8] George T. Heineman, Joseph Loyall, Richard Schantz, “Component Technology and QoS Management”

[9] Ivica Crnkovic, “Component-based Software Engineering — New Challenges in Software Development”, John Wiley & Sons, Ltd. (2002)

[10] Ivica Crnkovic, Stig Larsson, Michel Chaudron, “Component-based Development Process and Component Lifecycle”, Journal of Computing and Information Technology - CIT 13, 4, 321-327 (2005)

[11] Yosi Mass, Matan Mandelbrod,“Component ranking and Automatic Query Refinement for XML Retrieval”, IBM Research Lab, Haifa 31905, Israel

[12] Mingyang Gu, Agnar Aamodt and Xin Tong, “Component retrieval using conversational case-based reasoning”, Department of Computer and Information Science, Norwegian University of Science and Technology, Sem Sælands vei 7-9, N-7491, Trondheim, Norway

[13] Kiduk Yang, “Combining text- and link-based retrieval methods for Web IR”, School of Information and Library Science, University of North Carolina, Chapel Hill, North Carolina, U.S.A.

[14] Abraham Bernstein, Mark Klein, “Towards High Precision Service Retrieval”, New York University, New York, NY 10012, USA

[15] Prieto-Daz, R., “Implementing faceted classification for software reuse. Communications of the ACM”

[16] Zaremski, A.M. and J.M. Wing, “Signature matching: a tool for using software libraries. ACM Transactions on Software Engineering Methodology

[17] I.F. Cruz, “The Emerging Semantic Web (Frontiers in Artificial Intelligence and Applications, 75)”, 2002

[18] Sugumaran, V. and V.C. Storey, “A Semantic-Based Approach to Component Retrieval. The DATA BASE for Advances in Information Systems” 2003

[19] Carmen Fernández-Chamizo, Pedro A. González-Calero, Mercedes Gómez-Albarrán and Luis Hernández-Yáñez, “Supporting Object Reuse through Case-Based Reasoning”

[20] IBROW, Website http://www.swi.psy.uva.nl/projects/ibrow/home.html

[21] Janet L. Kolodner, “An Introduction to Case-Based Reasoning”, College of Computing, Georgia Institute of Technology, Atlanta, USA.