# COMPONENT RETRIEVAL USING CONVERSATIONAL CASE-BASED REASONING

Mingyang Gu1, Agnar Aamodt2 and Xin Tong3
*1,2,3Department of Computer and Information Science, Norwegian University of Science and Technology, Sem Sælands vei 7-9, N-7491, Trondheim, Norway +47 7359 7410 {1Mingyang, 2Agnar, 3Tongxin}@idi.ntnu.no*

Abstract: Component retrieval, about how to locate and identify appropriate components, is one of the major problems in component reuse. It becomes more critical as more reusable components come from component markets instead of from an in-house component library, and the number of available components is dramatically increasing. In this paper, we review the current component retrieval methods and propose our conversational component retrieval model (CCRM). In CCRM, components are represented as cases, a knowledge-intensive case-based reasoning (CBR) method is adopted to explore context-based semantic similarities between users' query and stored components, and a conversational case-based reasoning (CCBR) technology is selected to acquire users' requirements interactively and incrementally.

Key words: Software Component Retrieval, Conversational Case-Based Reasoning, Knowledge-Intensive Case-Based Reasoning, Semantic Similarity Calculation, Incrementally Query Acquisition

## 1. INTRODUCTION

One of the major problems associated with component reuse is component retrieval[1, 2, 3], which is concerned with how to locate and identify appropriate components to satisfy users' requirements. This problem becomes more critical as the emergence of several component architecture standards, such as, CORBA, COM, and EJB. These standards make software components interoperate more easily. Therefore component reuse surpasses the limitation of a single software company, that is, instead of getting components from an in-house component library, users search for desired components from component markets[4] (web-based software component collections provided by vendors or third parties), which separate component users and component vendors from each other.

A large and rapidly increasing number of reusable components put more strict demands on the retrieval efficiency[5]. If it is acceptable for users to look through tens of available components to identify the most appropriate ones, it is intolerable for them to look through hundreds, or thousands of candidate components, to select what they really need.

Several methods have been put forward to address the component retrieval problem. Most of them assume users can define their component query clearly and accurately, which puts too much impractical burden on component users. Based on the analysis of current retrieval methods, we propose a component retrieval model combining knowledge-intensive case-based reasoning technologies and conversational case-based reasoning methods.

Case-Based Reasoning (CBR) is a problem solving method[6]. The main idea underlying CBR is that when facing a new problem, we will search in our memory to find the most similar previous problem, and reuse the old solution to help solve the new problem.

A CBR process can be divided into four phases: retrieve, reuse, revise and retain, as described in[6]. Our research, as reported in this paper, focuses on the retrieve phase.

In the retrieve phase, a new case (new problem description) is compared to the stored cases, and the most similar one (ones) will be retrieved. Partial matching is adopted in the retrieve phase. Note that the CBR notion of partial matching, i.e. the matching of a group of features in order to return a best match, and where each feature typically has its own weight, distinguishes this technology from information retrieval and database access methods in general. Some CBR methods are 'knowledge-poor', which only consider superficial or syntactical similarities between a new case and stored cases, while other systems take both the syntactical similarity and the semantic similarity into account by combining case-specific knowledge and general domain knowledge. The latter approach is referred to as knowledge-intensive CBR[7].

Conversational case-based reasoning (CCBR) is an interactive form of case-based reasoning. It uses a mixed-initiative dialog to guide users to facilitate the case retrieval process through a question-answer sequence[8]. In the traditional CBR process, users are expected to provide a well-defined problem description (a new case), and based on such a description, the CBR system can find the most appropriate case. But usually users can not define their problem clearly and accurately. So instead of letting users guess how to describe their problem, CCBR calculates the most discriminative questions automatically and incrementally, and displays them to users to extract information to facilitate the retrieval process.

CCBR has been probed in several application domains, for instance, the customer support domain[9], and products or services selection in E-Commerce[10]. To our knowledge, current CCBR methods are to a large extent based on superficial feature properties, and there are so far no published results on CCBR applied to software component retrieval. In our research, we combine knowledge-intensive CBR and conversational CBR in an attempt to resolve the component retrieval problem.

The rest of this paper is organized as follows. In section 2, we review current existing retrieval methods, briefly discuss their advantages and disadvantages; in section 3, our conversational component retrieval model (CCRM) is proposed and some examples are illustrated; in section 4, we discuss the current status of using CBR technologies in the component retrieval field, and identify the advantages and limitations of our component retrieval model. In the end, we discuss our results so far, and point to future work (section 5).

## 2. CURRENT COMPONENT RETRIEVAL METHODS

A component retrieval method can be described from three aspects: component representation, component query (users' requirements) specification, and component retrieval process. A popular component retrieval method, named free-text-based retrieval method[11, 12], comes from the information retrieval community. In this method, components are represented as free-text-based documents, while a component query is described using keywords. The retrieval process is to look up the keywords in all component description documents. The components with most matched keywords will be selected. Vector space and indexing technology are used to facilitate documents organizing and matching. This method has low scores on both recall and precision[5]. Researchers and practitioners have proposed to use general thesaurus to extend keywords, by including their synonyms and antonyms, to get more relevant components[13]. In addition, general domain knowledge is also used to extend initial keywords to get more semantically relevant components[5]. However, both of these two improvements increase retrieval recall at the cost of retrieval precision.

Instead of free-text-based component and query descriptions, the following four types of retrieval methods represent components and specify queries using structural information from different perspectives. The pre-enumerated vocabulary method uses a set of pre-defined vocabularies to express both components and queries[14]. In this way, both recall and precision are increased at the cost of the flexibility to describe components and specify component queries. The signature matching method[15] describes both components and queries using signatures which specify the interfaces of components, for instance, the number and the type of input, and output variables. This approach is suitable for components implemented using strongly-typed programming languages. Its weakness is its lack of domain and searching context information. The behavior-based retrieval method[16] is based on the special characteristic of software components being executable. Components take the form of executable codes, and queries are represented by a set of input samples and their desired outputs. The retrieval proceeds by selecting samples, and executing components using the selected samples. The components that satisfy the desired output are retrieved. This method is designed for executable software components and has low efficiency because of long execution time.

The final method we want to mention in this category is faceted selection. This approach predefines a set of dimensions, called facets, which are used to classify components from different perspectives[14]. Users can find their desired components by searching down the stratified categories. This method is getting increasing attention because it takes domain knowledge into account when designing facets. But there exists a design embarrassment: If facets are designed too simple or few, there will be too many components in final categories, which will ask users to select further manually. On the other hand, if facets are designed too complex, it is hard for users to understand them and hard for designers to classify all components into different categories[17, 18]. In addition, the faceted selection method essentially uses the exact matching process. However, it is very hard to get the appropriate components through exact matching because of the universal differences between component requirements and components descriptions[19].

All the retrieval methods mentioned above have one common assumption, that is, users can well define their component queries, and the retrieval system can find one or a few appropriate components according to users' queries. However, this assumption is not always realistic. People often lack clear ideas about what they need while they begin searching for components and usually can not define their queries accurately. They need retrieval system to guide them refining their queries incrementally. Hence, an efficient component retrieval system should be able to support partial matching, select components based on both the syntactical similarity and the semantic similarity, and guide searchers to refine their component query incrementally. Conversational case-based reasoning, extended with knowledge-intensive CBR methods, provides a possibility for satisfying these requirements.

## 3. THE CONVERSATIONAL COMPONENT RETRIEVAL MODEL (CCRM)

### 3.1 CCRM Overview

As illustrated in Fig. 1, our conversational component retrieval model (CCRM) includes six parts: a knowledge base, a new case generating module, a knowledge-intensive CBR module, a component displaying module, a question generating and ranking module, and a question displaying module.

The knowledge base stores both component-specific knowledge (cases) and general domain knowledge (including a domain ontology). The new case generating module can set up a new case based on users' initial query and their later answers to discriminative questions. Given a new case, the knowledge-intensive CBR module calculates the similarities between the new case and stored component cases, and returns the components whose similarities surpass a threshold (the threshold is specified initially and can be adjusted following the execution of the system). The component displaying module displays the candidate components to users, ordered by their similarities. In the question generating and ranking module, possible unknown questions are identified, and an information gain algorithm[20] is used to rank the possible questions according to how much information it can provide if it has been answered. Then general knowledge is used to filter out those questions whose answers can be inferred from the initial query or previously answered questions. These ordered questions are further reordered according to some constraints inferred from general knowledge, for example, people normally prefer to answer the high level questions before answering low level ones. The question displaying module selects the most discriminative question, in order to optimize search towards a meaningful answer.

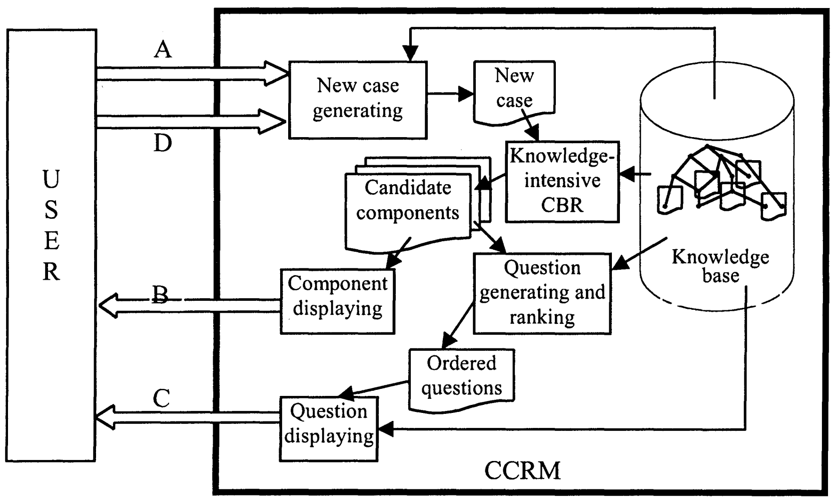


*Figure 1.* The architecture of conversational component retrieval model (CCRM).

Arrows: A, B, C and D, are interactive processes between users and CCRM. A: users input their initial query; B: the system provides users with top matched components; C: the system displays the most discriminative question to users; D: users select a displayed question and provide their answer to the system. Other processes are completed in the system automatically.

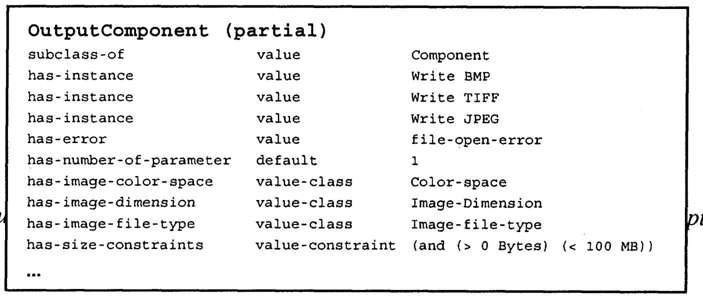
The retrieval process in the CCRM model can be described as the following steps:

1. Users provide their initial query, which takes the form of free-text-based terms.
2. The new case generating module transforms the initial query into a new case. In this step, a general thesaurus and a domain ontology are used to transform the free-text-based initial query into standard terms used in the internal system, and formalize them into a new case.
3. The knowledge-intensive CBR module calculates the similarities between the new case and stored cases through combining both component specific knowledge and general domain knowledge, and the components whose similarities surpass a threshold are returned.
4. If users find their desired component from the displayed candidate components, they can terminate the retrieval process. Otherwise, the conversational process is activated.
5. The question generating and ranking module identifies the unknown questions from the candidate components, and ranks them according to their information gains. Further, the ordered questions are filtered and reordered using general domain knowledge.

6. The question displaying module selects the most discriminative question, and displays it and its meaningful answers to users in a readable format.
7. Users provide the system with their answer to the displayed question. Otherwise, if users can not answer a displayed question, the question displaying module will display the next most discriminative question.
8. The new case generating module combines the previous new case and the newly gained answer to set up a new case.
9. The iterations from 3 to 8 continue until users find their desired component or there are no other discriminative questions left.

## 3.2 Component Representation in CREEK

In CCRM, we adopt a frame-based knowledge representation and reasoning system, CREEK[21], which can unify component-specific cases and general domain knowledge within a single representation system. In CREEK, all knowledge is represented as concepts, and a concept takes the form of a frame-based structure, which consists of a list of slots. A slot acts as a relation from the concept to a value related with another concept. Viewed as a semantic network, a concept (frame) corresponds to a node, and a relation (slot) corresponds to a link between nodes. Slot values have types or roles, referred to as facet. Typical facets include current value, default value, value class, and value constraint. So the knowledge in CREEK is represented in a 4-level structure, frame, slot, facet and value.



**OutputComponent (partial)**

| | | |
|---|---|---|
| subclass-of | value | Component |
| has-instance | value | Write BMP |
| has-instance | value | Write TIFF |
| has-instance | value | Write JPEG |
| has-error | value | file-open-error |
| has-number-of-parameter | default | 1 |
| has-image-color-space | value-class | Color-space |
| has-image-dimension | value-class | Image-Dimension |
| has-image-file-type | value-class | Image-file-type |
| has-size-constraints | value-constraint | (and (> 0 Bytes) (< 100 MB)) |
| ... | | |

*Figu pt.*

Fig. 2 gives, in a frame view, an example to illustrate how a part of an image OutputComponent class is represented in CREEK. Fig. 3 shows, in a network view, a part of the knowledge base for components used in the image processing field. General domain knowledge can be represented as relations between different values. For example, the "extract to" relation from "3D" to "2D" means that 3 dimension images can be extracted to 2 dimension images. Similarly, the two relations "convert to" between "XYZ" and "RGB" mean that images described using "XYZ" color space and images described using "RGB" color space can be converted to each other without losing any information.

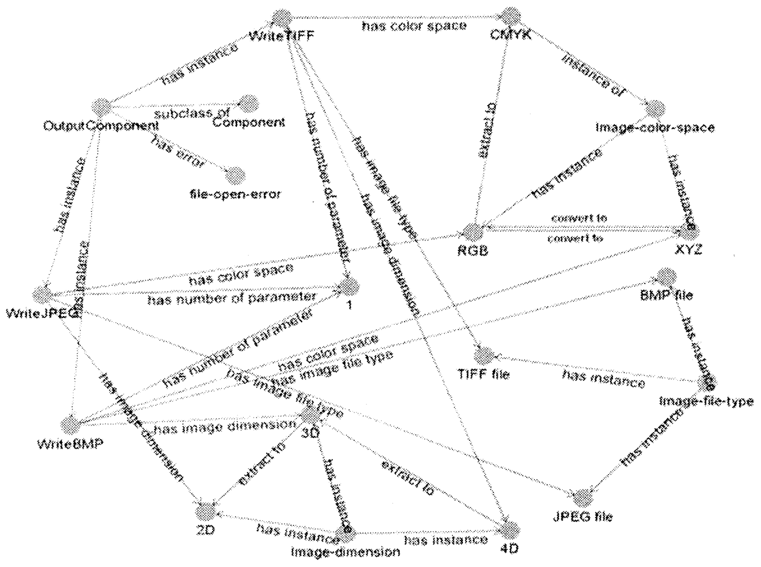


Figure 3. *A part of the knowledge base (implemented in the CREEK system) for components in the image processing field.*

| Component query (partial) | |
|---|---|
| has-number-of-parameter | 1 |
| has-image-color-apace | RGB |
| has-image-dimension | 3D |
| has-image-file-type | BMP file |
| has-error | file-open-error |
| has-file-size-constraints | 5 megabyte |
| ... | |

| Write BMP component | (partial) |
|---|---|
| has-number-of-parameter | 1 |
| has-image-color-space | XYZ |
| has-image-dimension | 2D |
| has-image-file-type | BMP file |
| has-error | file-open-error |
| has-file-size-constraints | (and (> 0 Bytes) (< 100 Megabyte)) |
| ... | |

*Figure 4. The partial frame contents of the component query and the stored component 'Write BMP'.*

## 3.3 Knowledge-Intensive Similarity Calculation

In CCRM, we use an explanation-driven similarity calculating method[7], which can be divided into three steps, ACTIVATE, EXPLAIN and FOCUS. ACTIVATE determines what knowledge in the knowledge base is involved in the retrieval process, and calculates the similarities between the new case and activated stored cases based on a rather syntactical or superficial similarity measuring. The output of the ACTIVATE step is a set of components whose similarity values surpass a certain threshold. EXPLAIN is used to evaluate the similarities between the new case and stored cases, selected in the ACTIVATE step, based on general domain knowledge. The evaluation task concerns justifying that the well-matched slots are relevant to the problem goal, and "explaining away" the mismatched slots that are unimportant. According to evaluation results, similarity values are adjusted. For instance, if one mismatched slot is evaluated as important for the problem goal, the total similarity value of the involved component is reduced. Otherwise, the similarity value is increased or keeps unchanged.

In the example shown in Fig. 4, there are two mismatched slots, "has-image-color-space" and "has-image-dimension", between the component query and the stored component. With the domain knowledge that "RGB convert to XYZ" and "XYZ convert to RGB", we can explain that "since the source image using RGB color space can be converted to an image using XYZ color space and vice versa, it is possible to use this stored component to realize the required task", and the similarity value generated in the ACTIVATE step can be kept unchanged or increased. On the contrary, there is not any explanation path from 2D to 3D, which means it is impossible for images with 2 dimensions to be converted to images with 3 dimensions, so the mismatch on the "has-image-dimension" slot can not be explain as unimportant and the similarity value of this stored component is reduced.

## 3.4 Question Selecting and Ranking

There are at least two requirements on the mixed-initiative question-answer interaction in conversational CBR. First, displayed questions should be easy to understand. Second, the selected question should be the most informative or discriminative one.

As to the first requirement, we predefine a question and its possible answers to each slot. For example, on the slot "has-image-file-type", we predefine a question that "what type of images do you want to deal with in this component?" and the possible answers, "BMP", "TIFF", "JPEG", or "Text". All the slots that appear in the candidate components, returned by the knowledge-intensive CBR module, but not in the new case are identified and transformed into unknown questions. Whether or not a possible answer is displayed to users in the conversational process depends on whether this answer appears in the candidate components.

As to the second requirement, "selecting the most informative question", we adopt the information gain metric[20] to quantitatively measure the information one slot (question) can provide (if we know the value of this slot).

The core concept in information gain is entropy. Given a collection S, its entropy value in state m can be calculated using the following formula:

$$Entropy\ (S_m) = \sum^{c} -p_i \log_2 p_i$$

The number c means how many sub-groups the collection can be divided into, and $p_i$ means the proportion of the ith sub-group. If we can not classify a collection of components into sub-groups, its entropy is 0 (c=1, $p_1$=1). After we acquire information on slot n, the collection can be classified into different sub-groups according to their various values on slot n, and the collection's entropy is increased. Information gain of slot n is defined as:

$$InformationGain(slot\_n) = Entropy(S_{have-\inf ormation-about-slot\_n}) - Entropy(S_{have-no-\inf ormation-about-slot\_n})$$

Different slots have different information gain. The larger the information gain one slot has, the more information it can provide if we know the value for this slot. That is, to find the most informative question is to find the slot with the largest information gain.

For instance, there is a candidate component collection with the number of 100, and there are two unknown slots in the new case, "has-image-file-type" and "has-image-color-space". According to the different values of the slot "has-image-file-type", appearing in the candidate components, "BMP", "TIFF" and "JPEG", the collection can be divided into three sub-groups with the numbers, 30, 30 and 40 respectively. According to the different values on "has-image-color-space", "RGB" and "XYZ", the collection can be divided into two sub-groups with the numbers, 30 and 70 respectively. In this case, the information gains of these two slots are calculated using the above formulae:

$InformationGain(S_{has-image-file-type}) = 1.5711 \quad InformationGain(S_{has-image-color-spacee}) = 0.8814$

So the question based on the slot "has-image-file-type" is more informative than that of "has-image-color-space". The question, "what type of images do you want to deal with in this component?" is displayed to users with three possible answers, "BMP", "TIFF", or "JPEG".

## 4. RELATED RESEARCH AND DISCUSSION

Software is used to resolve practical problems, and software components are existing solutions to previous problems, so component reuse can be described as "trying to use the solutions to previous similar problems to help solve the current problem". Therefore, it is very natural to use CBR methods to support component reuse. In fact, various types of CBR methods have been explored and found useful for component reuse.

Object Reuse Assistant (ORA)[2] is a hybrid framework to use CBR to locate appropriate components in an object-oriented software library (small-talk component library). In this framework, both small-talk classes and small-talk methods take the form of stored cases. The concepts in small-talk, for instance, c-class, c-method and c-data-spec, and their instantiated objects are connected together as a conceptual hierarchy. Though the conceptual hierarchy can be seen as a representation method combining case-specific knowledge and general knowledge, the retrieval process is knowledge-poor (a new case is compared with stored cases based on how many attributes two cases have in common).

IBROW[22] is an automated software application configuration project. Users' tasks (queries) can be decomposed into sub-tasks by matched task decomposers, and sub-tasks can be decomposed further. Tasks or subtasks can finally be solved by matched stored components. Both task decomposers and components are referred to as PSMs (problem solving methods). The output is an application configuration composed of stored components, which satisfies users' query. CBR is used at two levels in IBROW. The high level is called constructive adaptation. In this level, PSMs take the form of cases, which are represented using feature terms, and a knowledge-poor matching method (term subsumption) is adopted when searching the possibly applied PSMs. At the low level, CBR is used as a heuristic algorithm to realize the best-first searching strategy. Previously solved configurations are stored as cases, and represented as feature terms. For each intermediate state, the newly added PSM is considered. The stored configurations in which the same PSM appears as a part are identified, and the similarities between each of these configurations and the new problem are calculated. The most similar configuration is selected, and its similarity value is taken as the heuristic value to the involved intermediate state. As the ORA system, IBROW uses a knowledge-poor retrieval process and only supports tentative and manual interactions between users and the system.

Compared with these two CBR-based component retrieval systems, our proposed conversational component retrieval model (CCRM) has two advantages:

The first is that components are selected based on both their syntactical similarities and semantic similarities. Selecting components based on their semantic similarities with users' query rather than only on syntactical similarities is a promising research topic. However, the existing research concerned with this topic mainly use domain knowledge to refine users' queries before the searching process[5, 17, 23]. In CCRM, besides the query refinement using general thesaurus and domain-ontology, a special type of knowledge-intensive CBR method, explanation-driven CBR, is adopted to explore components' context-based semantic similarities with a query during the retrieval process.

The second is that users' requirements are acquired interactively and incrementally. Normally, component users prefer to provide their initial query only based on their necessary requirements in order to avoid excluding possibly appropriate components. Because of the looseness of the initial query and the large number of available components, users usually still get numerous candidate components. In CCRM, instead of letting users guess and try what requirements they should specify further, an information gain algorithm is used to provide users with the most discriminative questions to refine their query interactively and incrementally.

A limitation of our method is its dependence on knowledge engineering. The knowledge base combining both component specific cases and general domain knowledge is assumed to exist initially. The construction of this initial knowledge base puts a significant workload on the knowledge engineering process.

## 5. FUTURE WORK

The evaluation of CCRM is in process. The knowledge-intensive similarity measuring process has been realized in the CREEK system, and the conversational process is being added. We are building a knowledge base for the components existing in the DynamicImager system, a visual and dynamic image processing experimentation environment, in which there are about 200 different image operating components.

Our current research focus is to use the knowledge-intensive method to facilitate the discriminative question selection. Though the information gain algorithm can select the most discriminative question automatically and incrementally, it is knowledge-poor essentially. We plan to use knowledge-intensive methods, especially the explanation-driven method, to remove the candidate slots (questions) whose values can be inferred from users' initial query or previously answered questions, and to adjust the priorities between slots which represent semantic relations, such as, abstraction, causality, dependency and part-of relations. The hypothesis is that this will help to identify the most informative question, shorten dialog length, and reduce users' cognitive workload.

## REFERENCES

1. Mili, A., R. Mili, and R.T. Mittermeir, (1998), A survey of software reuse libraries. Annals of Software Engineering, 5(0): p. 349 - 414.
2. Fernández-Chamizo, C., et al. (1996), Supporting Object Reuse through Case-Based Reasoning. European Workshop on Case-Based Reasoning. Lausanne, Switzerland.
3. Iribarne, L., J.M. Troya, and A. Vallecillo. (2002), Selecting software components with multiple interfaces. 28th Euromicro Conference.
4. Ravichandran, T. and M.A. Rothenberger, (2003), Software reuse strategies and component markets. Communications of the ACM, 46(8): p. 109 - 114.
5. Klein, M. and A. Bernstein. (2001), Searching for Services on the Semantic Web Using Process Ontologies. The First Semantic Web Working Symposium. Stanford, CA, USA.
6. Aamodt, A. and E. Plaza, (1994), Case-Based Reasoning: Foundational Issue, Methodological Variations, and System Approaches. AI Communications, 7(1): p. 39-59.
7. Aamodt, A. (1994), Explanation-driven case-based reasoning. Topics in Case-based reasoning: Springer Verlag.
8. Aha, D.W. and L.A. Breslow, (2001), Conversational Case-Based Reasoning. Applied Intelligence, 14(1): p. 9-32.
9. Muňoz-Avila, H., et al., (1999), HICAP: An Interactive Case-Based Planning Architecture and its Application to Noncombatant Evacuation Operations.
10. Shimazu, H., (2002), ExpertClerk: A Conversational Case-Based Reasoning Tool for Developing Salesclerk Agents in E-Commerce Webshops. Artificial Intelligence Review, 18(3-4): p. 223 - 244.
11. Frakes, W.B. and B.A. Nejmeh, (1987), Software reuse through information retrieval. ACM SIGIR Forum, 21(1-2): p. 30-36.
12. Helm, R. and Y.S. Maarek. (1991), Integrating information retrieval and domain specific approaches for browsing and retrieval in object-oriented class libraries. Conference on

Object Oriented Programming Systems Languages and Applications. Phoenix, Arizona, United States.

13. Magnini, B., (1999), Use of a lexical knowledge base for information access systems. Journal of Theoretical & Applied Issues in Specialized Communication, 5(2): p. 203 - 228.
14. Prieto-Daz, R., (1991), Implementing faceted classification for software reuse. Communications of the ACM, 34(5): p. 89 - 97.
15. Zaremski, A.M. and J.M. Wing, (1995), Signature matching: a tool for using software libraries. ACM Transactions on Software Engineering Methodology, 4(2): p. 146 -170.
16. Park, Y., (2000), Software retrieval by samples using concept analysis. Systems & Software, 54(3): p. 179-183.
17. Sugumaran, V. and V.C. Storey, (2003), A Semantic-Based Approach to Component Retrieval. The DATA BASE for Advances in Information Systems, 34(3): p. 8-24.
18. Vitharana, P., F.M. Zahedi, and H. Jain, (2003), Design, retrieval, and assembly in component-based software development. Communications of the ACM, 46(11): p. 97-102.
19. Redondo, R.P.D., et al. (2002), Approximate Retrieval of Incomplete and Formal Specifications Applied to Vertical Reuse. International Conference on Software Maintenance. Montreal, Quebec, Canada.
20. Quanlan, J.R., (1986), Induction of decision trees. Machine Learning, 1(1): p. 81-106.
21. Aamodt, A. (1994), A Knowledge Representation System for Integration of General and Case-Specific Knowledge. International Conference on Tools with Artificial Intelligence. New Orleans.
22. IBROW, (2004), Website. http://www.swi.psy.uva.nl/projects/ibrow/home.html.
23. Bernstein, A. and M. Klein. (2002), Towards High-Precision Service Retrieval. International Semantic Web Conference. Sardinia Italy.