Improving Information Retrieval-based Concept Location using Contextual Relationships

Tezcan Dilshener
Center for Research in Computing, Department of Computing
The Open University, United Kingdom

Abstract— **For software engineers to find all the relevant program elements implementing a business concept, existing techniques based on information retrieval (IR) fall short in providing adequate solutions. Such techniques usually only consider the conceptual relations based on lexical similarities during concept mapping. However, it is also fundamental to consider the contextual relationships existing within an application's business domain to aid in concept location. As an example, this paper proposes to use domain specific ontological relations during concept mapping and location activities when implementing business requirements.**

Keywords-concept mapping; domain specific ontologies; concept location; contextual relations;

I. RESEARCH PROBLEMS

In software maintenance, prior to implementing business requirements, the designated software engineer has to go through the complex task of locating the relevant program elements implementing the concept at hand. This search activity is called the concept location. To influence the concept location success rate in finding the relevant program elements, researchers have used information retrieval (IR) techniques like Latent Semantic Indexing (LSI) [1]. Such techniques revealed that the relational aspects of the terms extracted from the program identifiers are also needed to obtain more effective results. Further research has looked into term associations within the natural language representation of the application's source code to map the conceptual relations [2]. In addition, these approaches revealed that terms used to represent similar meanings in computer science vocabulary, could have different meanings in the English vocabulary. For example, the term *fire* extracted from the identifier *fireEvent* is used synonymously with the term *notify* extracted from identifier *notifyListener* but the <notify, fire> pair do not commonly relate with one another in English text [3].

Existing information retrieval techniques perform concept location based on the lexical similarities of vocabulary between the search terms and the terms extracted from the source code identifiers. Some concepts though cannot be directly identified by looking at the single identifier names like the concept of *standalone risk*. Multiple words (*n*-grams) put together describe these types of concepts [4]. To catalogue such concepts, it is necessary to consider the conceptual relations that exist within the application's business domain.

Furthermore, current techniques do not consider expanding the abbreviated forms of the terms extracted from the source code identifiers [6]. It is presumed that the identifiers e.g. method names, are made up of single or compound well-formed English words. After splitting them into terms, they are used directly to determine their similarities to other terms or participate in some sort of a part-of-speech tagging mechanism. However, it is common to obtain ambiguous term definitions after identifier splitting. For example, the term *corr* extracted from the identifier *corrAsset* in a financial application represents the concept *correction asset* but when it is extracted from *corrList* it represents a list of *correlation* concepts.

Therefore, formalising the relations between concepts to represent the semantic information existing at a contextual layer is needed during concept mapping. This would improve the effectiveness of the concept location activities by providing fundamental clues. For example, in the case of "standalone risk", the word "standalone" would convey a conceptual meaning only when used together with the word "risk" in a financial application's domain.

In the following sections, a novel research approach is proposed to address the stated problems by utilising the contextual information that exists within an application's business domain captured in domain specific ontologies. The captured context will be used in reverse engineering the embedded knowledge from an application's source code to generate a searchable *corpus* for use in concept location activities. The corpus is the searchable intermediate form of the application's source code.

II. CURRENT RESEARCH

The literature on information retrieval identifies the LSI technique as one of the most commonly used techniques to support search engines [1]. LSI organises the occurrences of artefacts (e.g. terms and documents) in a repository of term by document matrix called the Vector Space Model (VSM). In VSM, the possible term relations are not taken into account when calculating their similarities. So, LSI applies the Single Value Decomposition (SVD) principle to store semantically related terms.

To represent the natural language (NL) relations existing between the terms extracted from an application's source code, Hill *et al.* [2] constructed a NL representation of the source code. This is achieved by applying linguistic phrase structures like nouns and semantic usage information obtained from mining the comments and word similarities. The approach does not consider the structural information, like *caller-callee* or inheritance relationships between the source code entities. In the caller-callee relation, the *caller*

is the method calling the current method and the *calle* is the method being called from the current method.

A promising approach to represent the relational context of an application's domain in software engineering is to capture the knowledge embedded in the project artefacts in domain specific ontologies. Ratiu *et al.* [5] extracted domain ontologies from Java APIs using similarity/path matching, like *isA, hasPart* relations, and validated them against the manually defined concepts. Hayashi *et al.* [6] have demonstrated the use of ontologies in concept location process to recover the traceability link between a natural language sentence and the related source code elements. A shortcoming of Hayashi's method is that the nodes from the ontology graph are directly mapped to the method names. Therefore during concept location the approach fails to detect those classes implementing the related concepts that use different words in the method names than those defined in the ontology graph.

The use of ontologies to represent the domain knowledge and then to utilise them during information retrieval has long been exploited in the biomedical research. The Open Biomedical Ontologies (OBO) consortium[1] promotes the integration of biomedical data through the annotation of multiple bodies using common controlled vocabularies or 'ontologies' [7]. Witte *et al.* [8] applied the ontology usage by integrating information retrieval between biological databases and text research papers within a biological application. In addition, Andreasen *et al.* [9] articulate that ontologies representing specific domains can be used to perform enhanced content-based text search. They argue that searching textual data is progressing towards a semantically oriented form and they present their project where querying over the semantically formalised biomedical text using ontology is achieved. However, the need to validate the approach on large-scale industrial applications is acknowledged.

III. Research Questions

Information retrieval methods expose common challenges across different disciplines when searching for the relevant pieces of information in a large data set. For example, in biomedicine, a neuroscientist attempting to extract information pertaining to the brain from the anatomy data set structure with many relations would only be interested in those relevant to the neuro-anatomical concepts [10]. The neuroscientist can be compared to a software engineer who is attempting to identify program elements implementing a concept in a financial risk application with several lines of code and concept relations. Therefore, to address the problems stated and to cover the gaps presented in the current techniques, there is a need to investigate answers to the following research questions:

How can the use of contextual information captured by domain specific ontologies (1) improve the concept mapping during searchable corpus generation, (2) provide meaningful clues to identify n-gram concepts during the concept location activities.

1. http://www.obofoundry.org
2. http://www.w3.org/TR/owl-features/

IV. Research Method

The research method will involve the definition of a semantic context, the development of a novel concept mapping approach used to generate a searchable corpus and a search engine for concept location.

A. Semantic context

The definition of a semantic context addresses the formal representation of business concepts implemented in an application by using ontologies. First, the previously extracted and validated financial business domain concepts identified in [4] will be further analysed manually to define the associations between each other. Second, this captured domain specific semantic information will be formalised by using the Ontology Web Language (OWL). It is stored as a project artefact to be leveraged during corpus generation and search activities. The OWL[2] is a structural framework for organising and representing information as a set of concepts and their relations within a domain.

During the formalisation of the concepts, only *isA* and *hasPart* relations will be considered [5]. The "isA" relation describes the hierarchical association between a concept and its super or subordinate, the "hasPart" describes the properties of a concept. Figure 1 shows a partial representation of the domain ontology for the "market investment risk" module of the financial application [4]. The two types of risk concepts, "diversified" and "standalone" are associated with the concept "risk" using "isA" relation. The "hasPart" relation is illustrated by the two properties of the concept "index", the concepts *volatility* and *correlation.* In the context of this paper the "isA" relation is going to be used to formalise the *n-gram*, compound concepts, like the concept of "standalone risk".

B. Corpus generation

The corpus will be created in three steps. The first step will extract terms from the program identifiers, split and store them in relational database by using the existing tools as demonstrated in [4]. The second step will develop a new concept mapping approach by extending the existing ones defined in [2] and [6]. In the third step, a static call-graph is going to be generated from the source code of the application to catalogue the structural information, such as caller-callee and inheritance relations amongst the program elements.

The approach presented in [2] extracts nouns, verbs, direct objects and prepositional phrases from method signatures and identifiers to enable contextual searching. The context fails to provide strong clues in detecting all the relevant program elements, especially those ones where the search terms are not used on the declaration of searched elements. For example, Figure 2 illustrates the partial call hierarchy graph of displaying the concepts "correlation" and "covariance" to the user of the financial application. Searching for the term "covariance" using the approach in

[2] detects the methods numbered 2, 4 and 5 but fails to detect the methods numbered 3, 7 and 8. This is because the context of the approach fails to provide relational clues. The approach will be further investigated to determine its extendibility by applying the domain specific ontologies to provide stronger context during concept location.

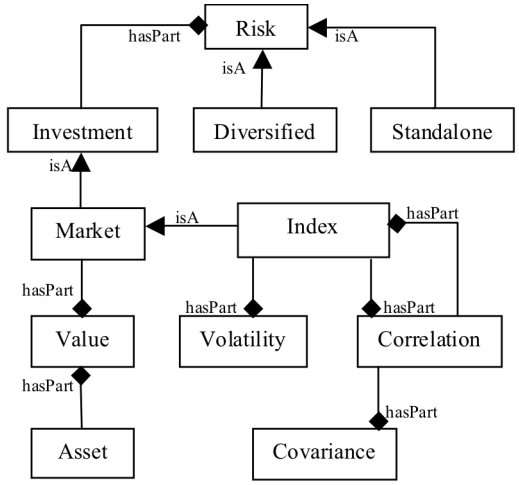

Figure 1. Partial ontology of "Market Investment Risk".

The approach presented in [6] makes use of relational information defined in ontology and in call graphs. The relevance of the called element is determined based on whether the call to the method is also captured on the ontology graph. Therefore it fails to detect those relations that are formalised at hierarchical class level. For example, in Fig. 2, searching for the term "covariance" would fail to detect the method numbers 7 and 8 because in Fig. 1, there is no direct ontology relation presenting a call from the method 5 to 7 and 8. The relation to the concepts "index" and "volatility" is established through the concept "correlation". This is illustrated in Fig. 2 where the class *ReaderCorrelation* references the classes *ReaderIndex* and *ReaderVolatility* to access their methods 7 and 8.

Also just following the call graph is not enough because the method 6 is not of interest for the search. Since classes implement the concepts, the ontology relations at class level need to be considered during a search. If the search starts at a method level then it won't find what is looked for if none of the methods in the class has the search terms in its name. For example, when searching for the concept "market value", the approach of [6] would not detect a method called *writeMVAssetCalc()*, called from the method *actionCalc()* in the class *MarketHelperCalcMVDetail* because none of the search words appear on any of the method names. The approach in [6] will be further investigated to determine the applicability of ontology relations at hierarchical levels.

Additionally, there are many-to-many relations between the *hard words* extracted from the source code and the *terms* describing the concepts [4]. For example, the concept "standalone" is represented by the hard words *std, stand* or *alone*. Such many-to-many relations between the hard words and the terms will be mapped by utilising the generated OWL and stored in the corpus. This <hard_word, term> mapping will include all the hard words referring to a concept and the relation of that concept to other concepts. This type of mapping is to allow the detection of the n-gram concepts during a search activity. For example, hard words "std", "stand" and "alone" will map to concept "standalone" which in turn will map to concept "risk" defined in OWL to represent the n-gram concept "standalone risk".

C. *Search engine*

When searching for the term "covariance", the proposed approach will expand the query term with the related ontology concepts "correlation", "index" and "volatility" as defined in Fig. 1. The methods 1-5, 7 and 8 will be detected by hard word similarities. The search will also evaluate the call-graph relations catalogued in the corpus to filter out those classes that do not have any direct call references from the selected classes. For example, the classes without a direct call containing the hard words "correlation", "index" and "volatility" will be filtered. The method 1 is not called directly from the class "ReaderCorrelation" and will be omitted from the results.

Subsequently, the ranking of the results will be established based on the frequency of the terms in the source code and in the text documentation [4].

Finally, the results will be presented by grouping the extracted method signatures based on the call hierarchy distance identified in the call-graph to facilitate relevance. For example the methods 3, 7 and 8 are to be grouped under 5. Similarly, the n-gram concepts will be searched by expanding the query term with their "isA" related concept from OWL and adding the related hard words from the corpus. For example, the query term "standalone" will include the query term "risk" and the hard words "std, stand and alone" to cater for "standalone risk".

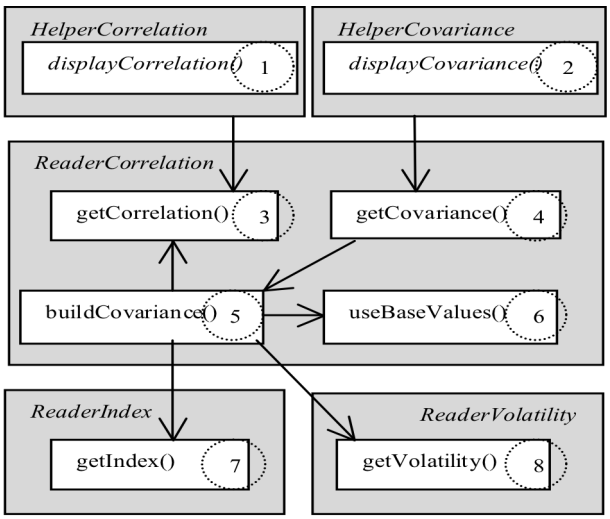

Figure 2. Partial call graph of displaying "Covariance".

In addition, we will conduct searches over the corpus using existing IR methods like LSI and cluster analysis (CA). In LSI, the corpus artefacts will be indexed and in CA, they will be grouped by applying an optimisation algorithm. Comparing the results against our gold standards [4], we hope the results will demonstrate the benefit of leveraging the contextual information captured by our approach in existing IR methods.

D. Validation

This research will be conducted using the financial applications at our industrial partner, a global financial IT solutions provider located in southern Germany. The approaches defined in [2] and [6] will be investigated to determine how well they perform when the relations between the source code entities are also considered using the domain specific ontologies. This will be measured by precision and recall. Comparing the results obtained from the generated corpus against the manually identified classes and concepts will assess the validity of the new concept mapping approach. The hypothesis is that the combination of the methods complemented by contextual information using domain specific ontologies will improve precision and recall results obtained in [4]. Although, the applicability of the research approach is demonstrated in the financial domain, the generalisability in other industries is only dependent on the definition of domain specific ontologies for that industry before it can be adopted.

V. Research Phases

This research will be accomplished in two phases. First, in addition to the proposed approach, the manually created ontologies will be used to aid abbreviation expansion. The absence of domain knowledge creates challenges when choosing from multiple abbreviation expansion possibilities [11]. These will be addressed by enhancing the concepts with *hasAttribute* ontology relations [5]. For example, the concepts "*correction asset*" and "correlation" will have *hasAttribute* relation to term "corr". So if the hard word "corr" is in a class representing the concept "asset" then it will be expanded as "correction" instead of "correlation".

In the second phase, based on the experience gained in mapping ontology to actual code and performing ontology guided concept location, the knowledge will be used to develop a novel approach to semi-automatically extract onotologies from the code. The approach will be validated first by extracting the domain specific ontologies from the financial application and comparing them against the manually extracted ones. Then, the technique will be used to extract the concepts from another application with similar domain concepts and will be validated by business analysts. Finally, Open Source Software like JDraw[3] with widely known domain concepts will be used to demonstrate the generalisability of the approach. The hypothesis is that by following the call-graph and utilising "isA", "hasPart" and "hasAtrribute" relationships will result in an improved approach to semi-automatically recover a first draft of domain ontologies from the code.

So far, we have extracted the hard words from the source code and identified the concepts from the user guide [4]. We have also partially established their ontological relationships and obtained a static call-graph. Next, we will navigate the call-graphs to discover additional program elements and evaluate their relevance to improving precision.

VI. Contributions

The importance of ontologies in reverse engineering has already been recognised, yet current approaches fall short of efficiently utilising the business domain ontologies illustrated in Fig.1. To achieve an accurate recovery of knowledge; there is a research opportunity to consider the domain specific ontologies during concept mapping and location activities.

The main contribution of this research will be the development of a novel concept mapping approach by extending the methods defined in [2] and [6] with domain specific business ontologies. The approach will be utilised in reverse engineering source code to generate a corpus used in concept location activities to improve the detection of the program elements implementing *n*-gram concepts.

References

[1] A. Kuhn, S. Ducasse, and T. Girba, "Semantic clustering: Identifying topics in source code," *Information and Software Technology*, vol. 49, no. 3, pp. 230-243, Mar. 2007.

[2] E. Hill, L. Pollock, and K. Vijay-Shanker, "Automatically capturing source code context of NL-queries for software maintenance and reuse," in *Proc. 31st IEEE ICSE*, 2009, pp. 232-242.

[3] G. Sridhara *et al.*, "Identifying Word Relations in Software: A Comparative Study of Semantic Similarity Tools," *2008,* in *Proc. 16th IEEE Int'l Conf. on Program Comprehension*, pp. 123-132.

[4] T. Dilshener and M. Wermelinger "Relating Developers' Concepts and Artefact Vocabulary in a Financial Software Module," in *Proc. 27th IEEE Int'l Conference on Software Maintenance*, 2011.

[5] D. Ratiu, M. Feilkas, and J. Jurjens, "Extracting Domain Ontologies from Domain Specific *API*s.," *IEEE*, 2008, pp. 203-212.

[6] S. Hayashi, T. Yoshikawa, and M. Saeki, "Sentence-to-Code Traceability Recovery with Domain Ontologies," in *Proc. 2010 Asia Pacific Software Engineering Conference*, 2010, pp. 385–394.

[7] B. Smith *et al.,* "The OBO Foundry: coordinated evolution of ontologies to support biomedical data integration," *Nat Biotechnol.* 2007;25:1251–1255. doi: 10.1038/nbt1346.

[8] R. Witte *et al.*, Combining Biological Databases and Text Mining to support New Bioinformatics Applications. *10th Int'l Conference on Applications of Natural Language to Information Systems (NLDB 2005)*, Springer LNCS 3513, 2005, pp. 310-321,

[9] T. Andreasen *et al.,* "*SIABO - Semantic Information Access through Biomedical Ontologies*". In KEOD. Springer, October 2009.

[10] J.F. Brinkley, L.T. Detwiler, J.H. Gennari, C. Rosse, D. Suciu, "A framework for using reference ontologies as a foundation for the semantic web," in *Proc. AMIA Fall Symposium*, 2006, pp. 95–100.

[11] E. Hill *et al.*, "Amap: Automatically Mining Abbreviation Expansions in Programs to Enhance Software Maintenance Tools," in *Information Sciences*, 2008, p. 79--88.

3. http://jdraw.sourceforge.net/index.php?page=6