ISSN: 2278 – 1323
International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)
Volume 1, Issue 7, September 2012

Context Awareness for Effective Software Structure Quality

B.RAGHAVENDRA, Mrs. S.VASUNDRA

***Abstract*- This paper presents an approach that helps developers to maintain source code identifiers and comments dependable with high-level artifact. This approach calculates and shows the textual similarity source code and related artifacts. The assumption is developers are induced to improve the source code lexicon (terms) used in identifiers or comments. The software development environment provides information about the textual similarity between the source code under development and the related high-level artifacts. Proposed approach recommends candidate identifiers build from high-artifacts related to the source code under development. The goal of the experiments is to estimate the similarity of the source and high level artifacts.**

***Keywords*- Software traceability, source code comprehensibility, source code identifier quality, information retrieval, software development environments, empirical software engineering.**

I. INTRODUCTION

The quality of software is assessed by a number of variables. These variables can be divided into external and internal quality criteria. External quality is what a user experiences when running the software in its operational mode. Internal quality refers to aspects that are code-dependent, and that are not visible to the end-user. External quality is critical to the user, while internal quality is meaningful to the developer only.Many Methods and approaches aimed at improving internal software quality problems and at supporting developers in improving software quality in dimension of Maintainability. For instance, previous work correlated source code quality, in terms of its change-Request, Error-Correction, and maintainability, to specific metric profiles [1], [2], [3], or to specific changes Extracted from source code history [4], [5]. From a different perspective, it has been highlighted that source code textual properties, in particular the use of proper identifiers are also an important indicator of software quality in Maintainability Dimension. Marcus et al propose a new cohesion metric (conceptual cohesion), complementary to structural cohesion, that exploits Latent Semantic Indexing (LSI) [12] to compute the overlap of semantic information in a class expressed in terms of textual similarity among methods. Another Approach in which the quality of identifiers and comments and their Correctness with the lexicon of high-level artifacts plays an important role is Information Retrieval (IR)-based traceability recovery [6][7],Such approaches work under the assumption that, if a source code artifact (e.g., a class) is textually similar to a high-level artifact(e.g., a requirement, Models), then it is very likely that there exists a traceability link between them. According to previous studies [8], [9], [10], producing source code with more meaningful identifiers and better comments would improve the code comprehensibility for maintainability of Software. Moreover, the use of IR techniques in traceability recovery to measure the similarity between the text contained in the source code and the domain terms contained in high-level software artifacts suggests that these techniques can also be used to improve identifiers and comments during software development and increase such similarity.

In this paper, we propose an IR-based approach aimed at showing the textual similarity between the source code under development and related high-level artifacts. [13]Our conjecture is that developers are induced to improve the source code lexicon, i.e., terms used in identifiers or comments, if the software development environment provides information about the textual similarity between the source code under development and the related high level artifacts.

In particular, the approach is based on the conjecture—discussed in previous literature [11], [12] and also assumed by traceability recovery approaches—that the similarity between high-level artifacts and the source code is an indicator of the quality of source code comments and identifiers. The suggestion provided to the developer might induce her to take different actions, such as making the source code identifiers more consistent with domain terms or better commenting the source code. To give further support to the developer, the proposed approach also recommends candidate identifiers built from high-level artifacts related to the source code.

II. RELATED WORKS

This section describes relevant background knowledge about terminologies used in the IR domain and relevant works related to our interests. We first present some terminologies currently used in literature, viewed a global context. Afterwards, we present some data smoothing techniques (e.g., document expansion, query expansion) which can be categorized into two main approaches: local context analysis vs. global context analysis. By global context, we mean that concepts or related terms are extracted using a knowledge source or a whole collection independently from the input text (document or query). By local context, related terms or concepts are extracted for a given text (document or query) using statistical properties of the sub-collection (top-ranked documents, k nearest concepts, etc.) related to the corresponding text. Finally, we summarize some related works dealing with search

context for enhancing document/query representations using either a local context (e.g., a sub-collection, top-ranked documents) or global context (e.g., a whole collection, a single terminology or several terminologies)

2.1 Information Retrieval as a global context

Several Domain terminologies have been used by different groups of research in IR, For Example in Bio Medical Domain especially in the context of TREC Genomics. The motivation of TREC Genomics was to support research and development in biomedical IR to drive new experimental research in the area of drug discovery for diseases. Since the commencement of TREC Genomics in 2003, several participants have tried to improve the performance of classical IR approaches by incorporating domain knowledge sources into a conceptual IR model. Generally speaking, conceptual IR model can be viewed a context-sensitive model because conceptual information are extracted within a particular context, e.g., thesaurus, ontology, or related documents, etc. We review in what follows the most termino-ontological resources that have been widely used for indexing biomedical documents.

2.2. Data smoothing techniques in context: query expansion vs. document expansion

In order to close the semantic gap between the user's query and documents in the collection, several research works have been focused on applying data smoothing techniques such as document expansion and query expansion on the original document/query. Theoretically, such techniques allow to enhance the semantics of the document/query by bringing the query closer to the relevant documents in the collection. As stated earlier, semantic information can be detected in a global context (usually from a domain knowledge source or an entire collection) or a local context (usually from a sub collection of related top-ranked documents).

III. Our Context-Sensitive IR Approach

Our context sensitive IR approach relies on twomain steps detailed below:

(1) Conceptual Document Indexing (step1)and (2) Context Sensitive Document Retrieval(step2). We integrate them into a IR process as the combination of the global and local semantic contexts for improving the Source Code Quality in Context Sensitive. The contextual semantic information is detected using domain knowledge sources and statistical information in a sub-collection. The former is referred to as global context while the latter is referred to as local context. Therefore, the contextual semantic information of a given query is revealed by concepts extracted from the global context during document expansion and related terms from a local context during query expansion. Figure 1 depicts the two main step of our IR approach.

Step 1
Conceptual Document Indexing

Input: Collection C, Terminologies T
Output: Index I

1: for all document D in C do
2: # Mono-terminology extraction
3: for all terminology T_i in T do
4: $R(D, T_i) \leftarrow extract(D, T_i)$;
5: end for
6: # Concept fusion
7: $R(D, T_i) \leftarrow \cup\ \hat{}\ n_{i=1} R(D, T_i)$;
8: # Document expansion
9: $D' \leftarrow expand(D, R(D, T_i))$;
10: # Document indexing
11: $I \leftarrow addIndex(D')$;
12: end for
13: return I;

Step 2
Context Sensitive Document Retrieval

Input: Query Q, index I
Output: Result set S

1: # Query normalization:
2: # stemming, stopword removals
3: $Q \leftarrow normalization(Q)$;
4: # First retrieval
5: $S0 \leftarrow search(Q, I)$;
6: # Extract λ best terms from
7: θ top-ranked documents
8: $\Gamma \leftarrow extract(S0, \lambda, \theta)$;
9: # Query expansion
10: $Qe \leftarrow expand(Q, \Gamma)$;
11: # Second retrieval
12: $S \leftarrow search(Qe, I)$;
13: return S;

Fig 1:Algorithm for Conceptual Document Indexing and Context Sensitive Document Retrieval

During the indexing stage, each document in the collection is analyzed to extract the most significant concepts using several terminologies. Our assumption behind multi-terminology based concept extraction is that the more concepts are found in several terminologies, the more they are important in the description of the document since they are well recognizedin several sub domains of medicine. For concept extraction, we adopt MaxMatcher, which is an approximate lookup based on dictionary matching Given a document, Max-Matcher will extract a set of terms or phrases denoting domain concepts as well as their corresponding concept unique

identifiers. However, Max- Matcher does not measure the importance of each concept for describing the semantics of the document, where tk is the constituent1 k of concept cj ; tf(tk) is the number of occurrences of term tk in document D; N is the total number of documents in the collection; nk is the number of documents containing word tk; dl is the document length; avg dl is the average document length; k1, and b are parameters; ℓ is the number of words comprising concept cj.Let C(D) be the set of concepts extracted from document D and C(Ti) be the set of concepts defined in terminology Ti. For each document D, the list of candidate concepts, denoted R(D,T) is extracted using terminology i. We need to find the final set R(D, T) containing the most relevant concepts for document D among the ones identified from several terminologies:

the number of terminologies used for indexing. During the retrieval stage, the top λ terms from the θ top-ranked expanded documents retrieved from the first retrieval stage are used to expand the original user's query.

IV. Experiments And Results

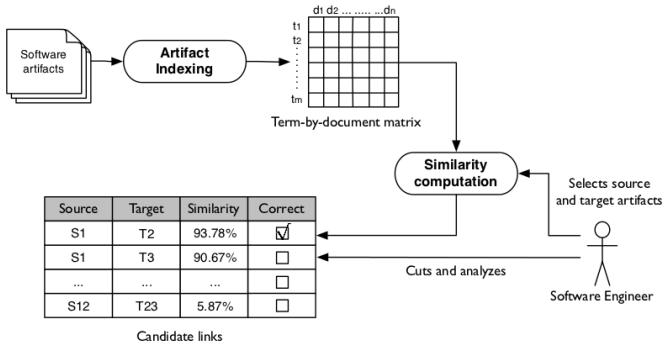

Fig 2: IR Based Traceability Recovery Approach

Information Retrieval is the area of study concerned with searching for documents, for information with in documents, and for metadata about documents, as well as that of searching structured storage, relational databases, and the World Wide Web. IR is the interdisciplinary, based on computer science, library system, information science, physics and statistics. Traceability is the mechanism that allows creating links between and within software artifacts. Traceability linksbetween software artifacts have to be identified and maintained during software development and maintenance.

It is time consuming process for software developers. So, they need methods and tools for handling traceability links. IR-based methods recover traceability links on the basis of the similarity between the text contained in the software artifacts.

The key idea behind such methods is that most of the softwaredocumentation is text based or contains textual descriptions,and that programmers use meaningful domain terms to define source code identifiers. IR methods, includes probabilistic model, Vector Space Models and Latent Semantic Indexing, have also been used to recover traceability between requirements, requirements and design artifacts, and requirements and design documents. In particular, if the source code does not have high similarity with the related high-level artifacts, the quality of source code identifiers or comments is likely to be poor, and this can potentially affect source code understandability and maintainability.

Information about the similarity level between the source code under development and related high level artifacts shows as table by the similarity tab. The first column of the table contains a check box that indicates whether the artifact has to be selected or not and traced onto the source code under development. The second column contains the description of the high-level artifacts, and the third column shows the similarity between the high level artifact and the source code being written. The third column represents the similarity level for the selected high level artifacts. Similarity represents textual similarity between the source code under development and related high level artifacts. Similarity will be low if there is meaningless identifiers and non related identifiers. Similarity will be high if there meaningful identifiers and related identifiers. Low similarity represents the comprehensibility of source code is complex where as high similarity represents the comprehensibility of source code is good. Source code comprehensibility influences the source code quality.

V. RESULTS

INPUT: SOURCE CODE and ARTIFACTS
OUTPUT: SIMILARITY IN CONTEXT SENSITIVE
Select Artifact is D:\LexicalCodeAnalysis\artifact1.txt and source similarity scores are:

AccountAdmin.java:22.727272%
AccountAdmin1.java:13.63636363%
AdministratorForm:0.0%
AdministratorForm1.java:0.0%
campusAdmin.java:18.727272%
campusAdmin1.java:27.727272%

IV. Conclusion

The paper proposed an IR approach to help developers improve the source code quality in context-sensitive. In particular, IR approach computes and shows to developers the textual similarity between source code and related high-level artifacts.

References

[1] Andrea De Lucia, Senior Member, IEEE, Massimiliano Di Penta, Member, IEEE, and Rocco Oliveto, Member, IEEE "Improving Source Code Lexicon via Traceability and Information Retrieval" IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 37, NO. 2, MARCH/APRIL 2011.

[2] V.R. Basili, L.C. Briand, and W.L. Melo, "A Validation of Object-Oriented Design Metrics as Quality Indicators," IEEE Trans. Software Eng., vol. 22, no. 10, pp. 751-761, Oct. 1996.

[3] T. Gyimo´thy, R. Ferenc, and I. Siket, "Empirical Validation of Object-Oriented Metrics on Open Source Software for Fault Prediction," IEEE Trans. Software Eng., vol. 31, no. 10, pp. 897-910, Oct. 2005.

[4] T. Zimmermann, R. Premraj, and A. Zeller, "Predicting Defects for Eclipse," Proc. Third ICSE Int'l Workshop Predictor Models in Software Eng., 2007.

[5] S. Kim, E.J. Whitehead, Jr., and Y. Zhang, "Classifying Software Changes: Clean or Buggy?" IEEE Trans. Software Eng., vol. 34, no. 2, pp. 181-196, Mar./Apr. 2008.

[6] S. Kim, T. Zimmermann, E.J. Whitehead Jr., and A. Zeller, "Predicting Faults from Cached History," Proc. 29th Int'l Conf.Software Eng., pp. 489-498, 2007.

[7] S. Deerwester, S.T. Dumais, G.W. Furnas, T.K. Landauer, and R.Harshman, "Indexing by Latent Semantic Analysis," J. Am. Soc. For Information Science, vol. 41, no. 6, pp. 391-407, 1990.

[8] A. Marcus and J.I. Maletic, "Recovering Documentation-to-Source-Code Traceability Links Using Latent Semantic Indexing," Proc. 25th Int'l Conf. Software Eng., pp. 125-135, 2003.

[9] A. Takang, P. Grubb, and R. Macredie, "The Effects of Comments and Identifier Names on Program Comprehensibility: An Experiential Study," J. Program Languages, vol. 4, no. 3, pp. 143-167, 1996.

[10] D. Lawrie, C. Morrell, H. Feild, and D. Binkley, "Effective Identifier Names for Comprehension and Memory," Innovations in Systems and Software Eng., vol. 3, no. 4, pp. 303-318, 2007.

[11] D. Lawrie, C. Morrell, H. Feild, and D. Binkley, "What's in a Name? A Study of Identifiers," Proc. 14th IEEE Int'l Conf. Program Comprehension, pp. 3-12, 2006

[12] A. De Lucia, F. Fasano, R. Oliveto, and G. Tortora, "Recovering Traceability Links in Software Artefact Management Systems Using Information Retrieval Methods," ACM Trans. Software Eng. and Methodology, vol. 16, no. 4, 2007

[13] D. Poshyvanyk and A. Marcus, "Using Traceability Links to Assess and Maintain the Quality of Software Documentation,"Proc. Int'l Symp. Grand Challenges in Traceability, pp. 27-30, 2007.

First Author B.Raghavendra, JNTU Anantapur, M.Tech., CSE (SE)

Second Author Mrs. Vasundra, Associate Professor, Dept. of CSE, JNTUA College of Engineering, Anantapur.