Supporting Natural Language Queries across the Requirements Engineering Process

Sugandha Lohar

School of Computing
DePaul University, Chicago, IL, 60604, USA
slohar@cs.depaul.edu

Abstract. [**Context and Motivation:**]Software project artifacts such as regulatory codes, requirements, design, code, and test cases facilitate many requirements-related engineering tasks including change impact analysis, safety assessment, and coverage analysis. Unfortunately, users often lack technical expertise in query languages such as SQL or XQuery and therefore have difficulties constructing meaningful queries. The problem can be alleviated when users express their queries using natural language (NL). [**Question/problem:**] NL interfaces depend upon domain models that capture the concepts and terminology of the domain. While prior studies have explored the types of queries, and associated terminology, used by software developers and maintainers, no such systematic study has been conducted in the requirements domain. We therefore do no fully understand the queries, terminology, and user interfaces needed to interactively support NL queries for requirements engineers. [**Principal ideas/results:**] This dissertation proposes a series of empirical studies to discover the information needs, and related queries, pertinent to requirements engineers. A domain model capturing concepts and terminology of the domain will be extracted from the sample queries. The work will also include a series of user-studies focusing on improving query representation techniques for displaying and communicating the interpreted NL Queries back to the requirements engineer. [**Contribution:**]Prior work has developed a prototype NL interface for supporting Software Engineering queries but has not studied its use in practice. The proposed solution will focus on NL Queries in the requirements domain, delivering a domain model to enhance understanding of requirements-related queries, and developing and evaluating query representations which support requirements engineers in the tasks they perform.

Keywords: Software Project Queries, Natural Language Processing, Human Computer Interaction

1 Introduction

Software projects produce a huge amount of data artifacts throughout their development process. This data can be leveraged to analyze and improve the soft-

ware systems by supporting activities such as requirements engineering, regression analysis, predictive analysis, coverage analysis, impact analysis and so on [4]. However, retrieving this data can be challenging as it is often stored in both structured and unstructured formats and geographically distributed. Furthermore, the queries for extracting this data are often extremely complex, requiring multiple joins and negations across varied artifact types. Writing non-trivial queries in structured languages, can be challenging even for experts; thereby, aggravating the data accessibility problem [9, 11].

A Natural Language Interface(NLI) addresses this problem by allowing users to express queries in their own words; however, general purpose NL interfaces have limited information of a particular domain and as a result, they fail to fully understand many complex domain-specific queries [12]. Customizing an interface for a particular domain requires extensive knowledge of the terminology and concepts of the domain. The goal of this work is to deliver a NLI solution for supporting requirements engineering queries. To this end, we will identify the information needs of business analysts and requirements engineers through a series of interviews, construct an appropriate NL domain model, evaluate and customize our current NLI tool, and evaluate the use of the NLI for supporting queries related to Requirements Engineering tasks. The proposed work will address the following research questions:

- **RQ1 :** What kinds of queries are useful for requirements engineers and business analysts?
- **RQ2 :** In what ways does the Natural Language domain model for general Software Engineering need to be modified to accommodate Requirements Engineering queries?
- Controlled natural language queries must be displayed to the user to ensure that the queries have been correctly interpreted. We therefore need to learn, **RQ3 :** What kinds of interactive displays provide effective and sufficient support for Requirements Engineers as they integrate queries into common requirements-related activities?

The remainder of this paper is laid out as follows. Section 2 provides the literature review of NL queries in the software engineering domain and discusses various query mechanisms. Section 3 gives an overview of the prior work on domain model construction and query representations. Finally, Section 4 discusses the proposed work and the final contribution of this research study.

2 Literature Review

This section summarizes the related work on natural language project queries and the query representation techniques.

Software Project Queries : Many researchers have identified the questions asked by software practitioners, however limited studies have been done that focus on the information needs of a requirements engineer. Fritz et al. interviewed

11 software developers and identified 78 questions spanning from multiple artifacts such as source change, change sets, work items, etc., [3]. Begel et al. provided a ranked list of 145 questions that a software engineer seeks from an analyst. The questions were classified based on their roles and activities. Roles included were developer, tester, program manager, customer and data analyst [1]. To identify the queries pertaining to the area of **software evolutions**, Silito et al. derived and categorized 44 questions that are to be answered by a programmer in order to gain sufficient comprehension to maintain source code [13].Other researchers have identified questions related to software development activities like **software testing** [2] or **security analysis** [14].

Query Representations : Many techniques have been developed to represent queries in the software engineering domain. Mäder et al. developed the Visual Traceability Modeling Language(VTML) which, assumes an underlying Unified Modeling Language(UML) class diagram representing the traceability strategies of the project [9]. The entities represent the traceable artifacts and the links represent the permitted traces between these artifacts. This class diagram is also generally called a Traceability Information Model (TIM). The queries are presented as a subset of this model along with the associated filter conditions. Maletic and Collard model queries for the artifacts stored in XML format by hiding the low level details of XQuery [10]. Speaking of the more general purpose query representations, Visual SQL by Jaakkola et al. represents SQL features graphically, similar to the entity relationship diagram or a UML diagram [5]. The PICASSO approach by Kim et al. is another general purpose database visual query representation for relational database system System/U [6]. However, these representation techniques either do not support all the query features or require some degree of technical expertise to read/write them.

3 Previous Work

An overview of our preliminary work in creating an elementary level domain model and query representation technique is presented here.

3.1 Building a Domain Model

Natural language project queries were collected from practitioners through a series of different user studies. The queries were then analyzed for extracting the domain terms to construct a domain vocabulary.

Query Collection : In our **pilot study**, we developed a web-collection tool that displayed a set of Traceability Information Models(TIM) [11]. Each participant was asked to issue five useful queries against it. The second study included **project specific scenarios**, which were designed to be used as prompts. An example scenario from the study is, *"The safety officer is worried that an important requirement R136 is not correctly implemented. The developer tells him*

Table 1. Sample Queries collected from Practitioners

#	Natural Language Query	Query Topic
1	How many high level hazards are associated with the security camera?	Safety Analysis
2	Did we develop components which are not backed up by project goals?	Gold Plating
3	Which programmers are more error prone in their code according to the test results?	Personnel
4	Is any of the open fault logs related to an exception case?	Fault Analysis
5	Which use cases have more than 5 requirements associated to them?	Requirements-Coverage Analysis

that it is not only implemented but has also passed the acceptance test. The security officer runs a trace query to confirm this. What is his query?" Participants were asked to address each of the scenarios by creating a NL query. Lastly, we conducted a **live study** at the Conference on Requirements Engineering for Software Quality where we gathered over 300 query samples from almost 50 different participants [8]. The study design was similar to the previous two studies. The participants were also encouraged to create NL queries based on the project they had been working or had previously worked on. Table 1 lists some sample queries gathered from these three studies.

Query Analysis : From the initial parsing of the query, we determined the *project specific terms* and the *question terms*. The *project specific terms* included words or phrases present within the data artifacts while the *question terms* holds the query together and represents specific query functions. For example, "List all", "Show me" or "Provide me a list of" relate to selecting a particular set of data. Similarly, the terms, "How many" or "Count the number of" are associated with aggregation functions. Further, the queries were catalogued according to their related software engineering tasks. The queries shown in Table 1 list sample query topics they were classified into. They were also grouped based on their semantic and syntactical charecteristics related to the query type and required functions. Example query, "Which use cases have the highest risks?" depicts a characteristic of *listing* some set of rows and to perform an *aggregation function*.

3.2 User Evaluation of Query Representation Techniques

We conducted three empirical studies to assess the efficacy of different query representation techniques [7]. The participants recruited in all three studies had a minimum of one year of IT experience and /or had sufficient background in software engineering. There were 39 participants recruited for the entire process.

The initial study included a survey designed to comparatively evaluate the four different query representations, which included SQL, VTML, Reverse Snowflake Joins(SFJ) and the output of the query(QO) as shown in figure 1. SQL is one of the most common structured textual query representation. VTML and SFJ are visual query presentation techniques where VTML specifically represents software data queries and SFJ represents more general purpose database queries. QO can help users to analyze a query by reviewing the data returned.

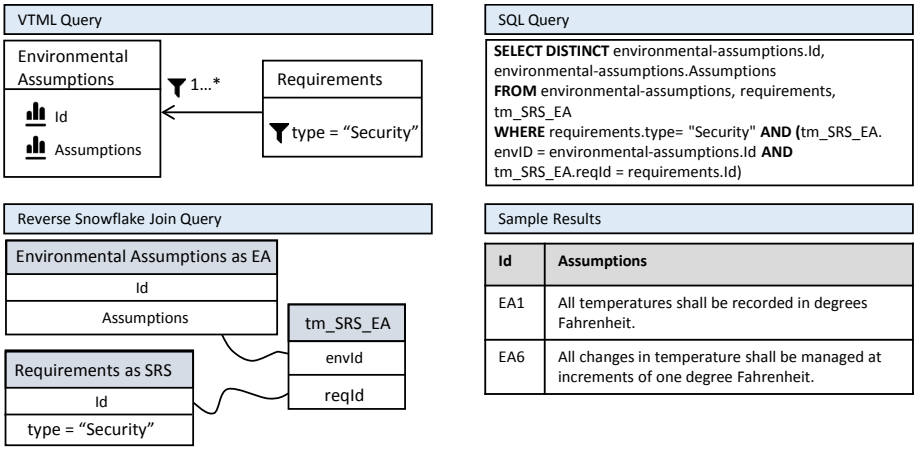

Fig. 1. Four query representations presented to participants.

The multiple choice survey consisted of sixteen queries, four of each kind. For each of them, participants were asked to select the NL query which they believe best matched the given representation of the multiple options provided. The result analysis showed that the VTML representation allowed users to interpret a query more accurately in less time as compared to the other techniques.

We were then interested to know, whether users integrate knowledge from different query representations in order to comprehend the meaning of the query. Another study was designed that included an **eye-tracker** with a usability software which recorded the participants eye-gaze information from the display screen. The study included nine NL queries. Each of these queries were deliberately transformed incorrectly and presented in VTML, SQL and QO. All three representations were displayed on a single screen along with the NL query to be interpreted. The common transformation errors included mapping onto incorrect artifacts or their associated properties, logical errors etc. An example query from the study, *“List any test cases which have failed in the past week”* was represented as *“List any test cases which have failed in the past month”*. The participants were asked to determine whether the NL query matched the given representation and if not then to explain how it differed from the actual query. We asked them to use think-out-loud protocol to answer these questions.

From the result analysis, we observed that even though most users favored a specific representation technique the majority of them leveraged information from multiple sources to analyze the query. Secondly, the users often spent too much time on superfluous details thereby, increasing the total query interpretation time. To address this, we designed another user study which simplified the original query displays by hiding unnecessary details and JOIN paths. The results showed that the time to identify errors in **reduced displays** was significantly decreased in comparison to the original displays.

Results from this work have been integrated into our TiQi tool (see TiqiAnalytics.com) and will be used to support the proposed work.

4 Proposed Work

Previously, we created a domain model that covers broader but more general purpose software domain queries. To customize the domain model supporting requirements related queries, we propose the following:

- **Identifying Requirements Engineer's Tasks and Questions:** We will interview requirements engineers and business analysts to understand the kind of queries they are interested in posing. For this, we plan to use mailing lists such as the RE mailing list, and our network of contacts with professional requirements engineers to recruit at least 10-20 participants. Each participant will be asked to identify requirements-related tasks that they perform, the data they need to support those tasks, and relevant NL queries.
- **Constructing a Requirements-Related Domain Model:** We will create a comprehensive knowledge base to support RE related queries. The knowledge base will initially be populated using our existing existing data mining techniques to mine domain-specific terminology from the text of online requirements textbooks, white-papers, and other requirements-related documents. We will perform the same analysis on the collected queries and augment the domain model accordingly. For example, requirements related terminology that should be recognized by the NL interface includes terms such as *"User Story"*, *"Mitigating Requirement"* or *"Design Constraint"*.
- **Evaluating the use of NL Queries within the Requirements Engineering Context:** Using the generated domain model and the display mechanisms developed in earlier phases of this dissertation work, we will evaluate the use of NL queries for supporting requirements engineering activities. We will recruit 10-12 requirements engineers using the same protocol as used in the proposed work for,"Identifying the Requirements Engineer's Tasks and Questions". Each participant will be presented with a series of identified RE related tasks. Half of the participants will be asked to perform tasks by issuing NL Queries using our NLI, while the others will be given only raw data in a database. For the NL queries we will evaluate the extent to which they are correctly interpreted. Furthermore, we will evaluate the extent to which the use of NL Queries supports and/or augments the completion of tasks.

Requirements Engineers can greatly benefit from increased accessibility to underlying data. This work will contribute an enriched domain model which captures RE terminology and concepts, and a NL interface which provides effective support for requirements engineers to utilize NL Queries to perform common analytic tasks.

5 Acknowledgement

I am very thankful to my advisor Dr. Jane Cleland-Huang and co-advisor Dr. Alexander Rasin for their continuous guidance and immense support for carrying out this research.

References

1. A. Begel and T. Zimmermann. Analyze this! 145 questions for data scientists in software engineering. In *Proceedings of the 36th International Conference on Software Engineering*, pages 12–23. ACM, 2014.
2. S. Breu, R. Premraj, J. Sillito, and T. Zimmermann. Information needs in bug reports: improving cooperation between developers and users. In *Proceedings of the 2010 ACM conference on Computer supported cooperative work*, pages 301–310. ACM, 2010.
3. T. Fritz and G. C. Murphy. Using information fragments to answer the questions developers ask. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 1*, pages 175–184. ACM, 2010.
4. Gotel, O et al.,. Traceability fundamentals. In J. Cleland-Huang, O. Gotel, and A. Zisman, editors, *Software and Systems Traceability*, pages 3–22. Springer, 2012. 10.1007/978-1-4471-2239-51.
5. H. Jaakkola and B. Thalheim. Visual SQL – High-Quality ER-Based Query Treatment. In *Conceptual Modeling for Novel App. Domains*, Lecture Notes in Comp. Science. Springer Berlin / Heidelberg, 2003. 10.1007/978-3-540-39597-3.
6. H.-J. Kim, H. F. Korth, and A. Silberschatz. Picasso: a graphical query language. *Software, Practice and Exp.*, 18:169–203, March 1988.
7. S. Lohar, J. Cleland-Huang, and A. Rasin. Evaluating the interpretation of natural language trace queries. In *Joint Proceedings of REFSQ-2016 Workshops, Research Method Track, and Poster Track co-located with the 22nd International Conference on Requirements Engineering: Foundation for Software Quality (REFSQ 2016), Gothenburg, Sweden, March 14-17, 2016.*, page to appear, 2016.
8. S. Lohar, J. Cleland-Huang, A. Rasin, and P. Mäder. Live study proposal: Collecting natural language trace queries. In *Joint Proceedings of REFSQ-2015 Workshops, Research Method Track, and Poster Track co-located with the 21st International Conference on Requirements Engineering: Foundation for Software Quality (REFSQ 2015), Essen, Germany, March 23, 2015.*, pages 207–210, 2015.
9. P. Mäder and J. Cleland-Huang. A visual language for modeling and executing traceability queries. *Software and System Modeling*, 12(3):537–553, 2013.
10. J. I. Maletic and M. L. Collard. Tql: A query language to support traceability. In *TEFSE '09: Proceedings of the 2009 ICSE Workshop on Traceability in Emerging Forms of Software Engineering*, pages 16–20, Washington, DC, USA, 2009. IEEE Computer Society.
11. P. Pruski, S. Lohar, G. Ott, W. Goss, A. Rasin, and J. Cleland-Huang. Tiqi: Answering unstructured natural language trace queries. *Requirements Engineering.*, 20(3):215–232, 2015.
12. S. P. Shwartz. Problems with domain-independent natural language database access systems. In *Proceedings of the 20th annual meeting on Association for Computational Linguistics*, ACL '82, pages 60–62, Stroudsburg, PA, USA, 1982. Association for Computational Linguistics.
13. J. Sillito, G. C. Murphy, and K. De Volder. Questions programmers ask during software evolution tasks. In *Proceedings of the 14th ACM SIGSOFT international symposium on Foundations of software engineering*, pages 23–34. ACM, 2006.
14. J. Smith, B. Johnson, E. Murphy-Hill, B. Chu, and H. R. Lipford. Questions developers ask while diagnosing potential security vulnerabilities with static analysis. In *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*, pages 248–259. ACM, 2015.