A Context-based Information Retrieval Technique for Recovering Use-Case-to-Source-Code Trace Links in Embedded Software Systems

Jiale Zhou, Yue Lu, Kristina Lundqvist
School of Innovation, Design and Engineering
Mälardalen University
Västerås, Sweden
{zhou.jiale, yue.lu, kristina.lundqvist}@mdh.se

***Abstract*—Post-requirements traceability is the ability to relate requirements (e.g., use cases) forward to corresponding design documents, source code and test cases by establishing trace links. This ability is becoming ever more crucial within embedded systems development, as a critical activity of testing, verification, validation and certification. However, semi-automatically or fully-automatically generating accurate trace links remains an open research challenge, especially for legacy systems. Vector Space Model (VSM), a notably known Information Retrieval (IR) technique aims to remedy this situation. However, VSM's low-accuracy level in practice is a limitation. The contribution of this paper is an improved VSM-based post-requirements traceability recovery approach using a novel context analysis. Specifically, the analysis method can better utilize context information extracted from use cases to discover relevant source code files. Our approach is evaluated by using three different embedded applications in the domains of industrial automation, automotive and mobile. The evaluation shows that our new approach can achieve better accuracy than VSM, in terms of higher values of three main IR metrics, i.e., recall, precision, and mean average precision, when it handles embedded software applications.**

Keywords—trace link recovery; post-requirements traceability; vector space model (VSM); context analysis; embedded system; functional requirements;

I. INTRODUCTION

Requirements Management (RM) is a critical activity for system development. It should be carried out for all the phases of systems development life cycle (or the software development process in other words), rather than a single phase. RM assumes requirements elicitation, tracking and preservation of integrity, and handles a large amount of software development artifacts (i.e., the artifacts hereafter). The quality of RM is very important for system development, e.g., customers satisfaction, requirements coverage, efficient utilization of resources.

The heart of RM is Requirements Traceability (RT), which is defined as "the ability to describe and follow the life of a requirement, in both a forwards and backwards direction (i.e., from its origins, through its development and specification, to its subsequent deployment and use, and through periods of on-going refinement and iteration in any of these phases)" [1]. RT provides critical support for system developers throughout the entire software development process. Tracing requirements can help to, but is not limited to, perform change impact analysis, risk analysis, criticality analysis, regression testing, and requirements satisfaction assessment. However, in traditional industrial practices, especially for legacy systems [2], trace links are manually established and maintained. Such activities tend to be costly to implement and are therefore perceived as financially non-viable by many companies [1], [3]. To address this problem, many efforts [4], [5], [6], [7], [8], [9], [10] have been devoted to semi-automatic or fully-automatic trace link creation. However, such creation throughout the entire systems development process, remains a challenging issue [11].

Among these efforts, the algebraic model Vector Space Model (VSM) [13] (referred to as the standard VSM hereafter), has been most investigated to build trace links between textual artifacts, such as requirements and source code [14], [15], [16]. After requirements and the target artifacts are preprocessed by e.g., removing stop words, stemming, the obtained term-document matrix is used by the standard VSM, which produces descending-ordered ranked lists of candidate trace links. Such candidate trace links contain the scores which express the similarity between requirements and subsequent artifacts, based on the occurrences of terms. Then, different strategies are applied to prune undesired links, and finally, the resulting candidate trace link lists are vetted by human analysts w.r.t. relevancy to a specific project. Nevertheless, statistical analysis [17] showed that analysts' tracing experience and amount of effort (applied to look for missing links, comfort level with tracing and so on) do not affect the accuracy of the final trace link lists. Rather, the initial accuracy of the candidate trace link lists is the most important factor, impacting the accuracy of the final trace link lists. Our goal in this paper is to tackle the above problem by using a novel VSM-based context analysis, which involves lightweight human intervention in the early phase of the RT recovery process. In doing this, higher-accuracy candidate trace link lists are obtained when compared with the standard VSM, which also dramatically reduces the human efforts involved in the final phase of the RT recovery process.

Note that Use Case (UC) technique has been widely adopted as a Requirement Specification Language (RSL) in the embedded systems development, with the advantage of many benefits it provides [18]. In order to better illustrate our approach, we are particularly interested in establishing trace links between UCs and source code files in this work. In particular, the technical contributions of this paper are two-fold:

1) We introduce the VSM-based context analysis, which consists of three steps. Specifically, the first step is to analyze the constructs of the RSL, in order to obtain the requirement intent and a set of context information. Further, the extracted context information is classified into two groups, i.e., requirements intent-positive and requirements intent-negative (cf. Section III). In the second step, the requirement intent and the intent-positive context information are separately used by the standard VSM as input, which generates two trace link lists. Lastly, the two trace link lists are combined together through a weighted knowledge model, which generates the candidate trace link list.
2) We show that our new approach improves the traceability accuracy of the standard VSM, by obtaining higher values of three main IR metrics, i.e., *recall*, *precision*, and *mean average precision* (MAP) scores. Typically, our case studies are three different embedded applications in the domains of industrial automation, automotive and mobile.

The remainder of the paper is organized as follows. Section II introduces the related work and background theory. Section III firstly gives an overview of our analysis, and then presents different parts of our proposed method together with the implementation of the algorithm in detail. Next, Section IV that describes the evaluation setup, research questions for evaluation, evaluation metrics, improvements of analysis results as well as results validity, and finally, conclusions and future work are drawn in Section V.

II. Background

This section firstly describes the related work in Section II-A, and then illustrates the trace link recovery process based on Information Retrieval (IR) techniques in Section II-B, which is followed by an introduction about context-based analysis in Section II-C.

A. Related Work

Many recent studies have explored the feasibility of different IR methods for semi-automatically or fully-automatically recovering trace links between requirements and subsequent artifacts. Deerwester et al. [4] discuss the effectiveness of Latent Semantic Indexing (LSI) for recovering trace links between different kinds of artifacts, and Marcus et al. [19] further the work, showing a promising result over VSM. Abadi et al. [15] present a novel IR technique based on Jensen & Shannon (JS) model. They also compared JS, VSM and LSI for traceability recovery purpose, and concluded that VSM and JS are the best-fits. A similar comparison is also conducted by Oliveto et al. [20], which showed that for building trace links between requirements and source code, the results of JS, VSM and LSI are almost equivalent.

Variants of basic IR methods have been proposed to improve the accuracy of IR-based traceability recovery approaches. Fautsch et al. [16] present four extensions to the classical *tf-idf* VSM model. The basic idea is to retrieve domain specific information. Kong et al. [8] present a VSM enhancement using term location. By utilizing the relationship between words in different textual documents, a better accuracy was achieved. Lucia et al. [9] present the approach

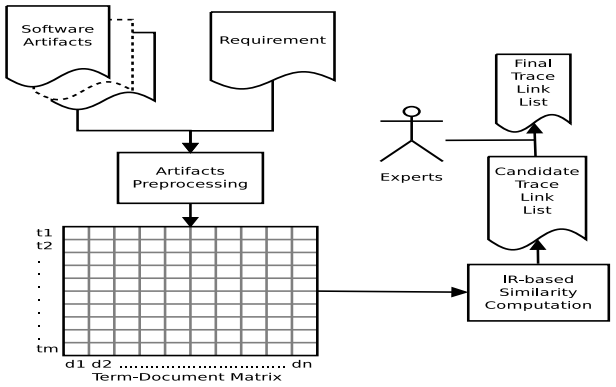

Fig. 1. An IR-based traceability recovery process.

that uses smoothing filter to improve the input in the IR-based traceability recovery process. Specifically, the words that contribute less information but repeatedly occur in the documents, are removed. Cleland-Huang et al. [6] introduce three accuracy enhancement strategies, which are hierarchical modeling, logical clustering of artifacts, and semi-automatic pruning of the probabilistic network. It should be pointed out that our approach is a variant of IR methods, but it is very different from the strategies proposed in the prior work. In their work [6], though context information is obtained from the artifact hierarchy to improve traceability results, it is not the type of "context" defined in our work. In our case, such context information is the title, pre-condition, and post-condition that are extracted from the requirements (i.e., UCs), which can contribute greatly to improve the traceability results.

Some other pieces of work also show us another promising perspective. Asuncion et al. [27] apply Topic Modeling technique, featured by Latent Dirichlet Allocation (LDA) [5] to capture trace links prospectively. Lucia et al. [12] discuss the feasibility of using user feedback analysis to improve the accuracy of the results of traceability recovery tools. Mahmoud et al. [10] propose a semantic relatedness approach that exploits external knowledge sources, e.g., Wikipedia, to identify a set of relevant terms that are used to expand the query, in an attempt to improve traceability results.

B. IR-based Traceability Recovery

The IR-based traceability recovery aims at utilizing IR techniques to compare a set of source artifacts as queries (e.g., requirements), against another set of target artifacts e.g., source code files, and calculate the textual similarities of all possible pairs of artifacts. The textual similarity between two artifacts is based on the occurrences of terms (words) within the artifacts contained in the repository. Pairs with a similarity score lower than a certain threshold (usually defined based on engineers' experience) are filtered out, and the reserved pairs form the candidate trace link list. The ranked list of candidate trace links are then analyzed by software engineers to decide if such links are true positive or not. Typically, an IR-based traceability recovery process follows the steps depicted in Figure 1.

The artifacts have to be preprocessed before they are used to compute similarity scores. The preprocessing of the artifacts includes a text normalization by removing most non-textual tokens (e.g., operators, punctuations) and splitting compound

identifiers into separate words by using the underscore or Camel Case splitting heuristic. Furthermore, common terms, referred to as "stop words" (e.g., articles, prepositions and programming language keywords), which contribute less to the understanding about artifacts, are also discarded by using a stop word filter. Words with the length less than a defined threshold are also pruned out. In addition, stemmer is commonly used to perform a morphological analysis, which reduces the inflected words to their root, e.g., returning verb conjugations and removing plural nouns.

After preprocessing, an artifact (e.g., a UC requirement, a source code file) can be represented as a plain document containing a list of terms (in this paper, we use *documents* and *artifacts* interchangeably). The extracted terms are generally stored in a $m \times N$ matrix (called term-by-document matrix), where m is the number of all the terms that occur in all the documents, and N is the number of documents in the corpus. A generic entry $w_{i,j}$ of the matrix denotes a measure of the relevance of the i_{th} term in the j_{th} document. Based on the term-by-document matrix representation, different IR methods can be used to calculate textual similarities between paired artifacts.

Particularly, in Vector Space Model (VSM) [14], given the entire collection of unique terms $T = \{t_1, \ldots, t_m\}$ in a corpus with N documents, the document d_n is represented as a vector $d_n = \{w_{1,d_n}, \ldots, w_{m,d_n}\}$ consisting of m unique terms from the corpus with an assigned weight w_{i,d_n} through a certain weighting scheme. Therefore, the similarity score, denoted as $sim(q, d)$, between the query document q and the target document d is calculated by using the cosine of the angle between their vectors:

$$sim(q,d) = \frac{\sum_{i=1}^{m} w_{i,q} \cdot w_{i,d}}{\sqrt{\sum_{i=1}^{m} w_{i,q}^2 \cdot \sum_{i=1}^{m} w_{i,d}^2}} \quad (1)$$

Next we introduce the *term frequency-inverse document frequency*, i.e., *tf-idf*, which is adopted as the weighting scheme by all the VSM-based approaches (including our method).

$$w_{i,q} = tf_i(q) \cdot idf_i, \quad w_{i,d} = tf_i(d) \cdot idf_i \quad (2)$$

where $tf_i(q)$ and $tf_i(d)$ are measured by the number of times the term t_i occurs in the query document q and the target document d respectively, and idf_i is computed as $log(\frac{N}{df_i})$, where df_i is the number of documents containing the term t_i.

The standard VSM described above has been applied to the requirements traceability recovery process [13]. In our work, the corpus is the entire set of requirements (i.e., UCs) and source code files. In applying the standard VSM, we select a UC (as the query q) and repeatedly calculate the similarity scores between the UC and all the source code files in the corpus. In this way, a descending-ordered ranked list of candidate trace links to the requirement will be generated by VSM. However, the low-level accuracy of the standard VSM in practice is a limitation [21]. From our viewpoint, the main reason is that the standard VSM takes the whole requirement document as its input, regardless of the relevance of the terms associated with the intent of the requirement. Therefore, it is inevitable that the majority of the terms used by VSM are irrelevant ones, hence misleading VSM to produce a very low-accuracy trace link list. Clearly, one possible improvement is to provide VSM with more relevant terms that can better represent the requirement intents, which are obtained through the context-based analysis introduced in the following section.

C. Context-based Analysis

Connolly [22] introduces an important fact about communication: Communication always takes place in a context. Suppose that we are interested in studying an intent of others, which we denote as I. The context consisting of whatever constructs surround I, serves to facilitate the communication between speakers and listeners. Furthermore, the intents of people can be reflected by the context of the conversation between them, based upon their understanding and interpretation. Accordingly, if we regard a pair of a requirement and its subsequent artifacts as "speaker" and "listeners", between which there is a successful communication, then context information surrounding speaker's intent is helpful to recover the pair, i.e., the trace link between the requirement and its subsequent artifacts.

In order to have a better understanding about the idea of using context-based analysis to improve the accuracy of the standard VSM, we give the following example of a little boy buying ice cream: A little boy wanted to eat an ice cream, so he wrote the words "ice cream" on his mother's shopping list. Later on, two actions took place and can be documented as: 1) his mother went to a shop and bought an ice cream; 2) he updated the status of his Twitter with the statement "Ice cream is my favorite dessert!". From the perspective of IR-based traceability recovery, we can consider his writing ice cream on the shopping list as the requirement. The two documented actions can be regarded as the subsequent artifacts, i.e., documents. In this case, both documents are kind of related with the requirement, because all of the requirement and documents share the term "ice cream". Thereby, it is very hard to conclude which document has the higher similarity score with the requirement by using the standard VSM. Nevertheless, if we perform context analysis on the requirement, we can find that the requirement intent is "eat an ice cream" and the requirement context is "shopping list". Obviously, the first document and the requirement share the implicit information "shop". Therefore, the first document is more relevant to the requirement about "eat an ice cream", when the context analysis is applied. By using the above intuitive idea, we propose a novel VSM-based context analysis, which can obtain better traceability results, when compared with the standard VSM.

III. The Proposed VSM-Based Context Analysis Method

A functional requirement is a need that a particular product or process must be able to perform. It can also be regarded as one or a set of *intents*, of which the detailed interpretations are subsequent software development artifacts, e.g., design documents, source code, testing and maintenance documents. Accordingly, the traceability recovery process is about finding different relevant interpretations of such intents. In matters of interpretation, it is very important to understand the context.

Since context not only plays a significant role in influencing the way that the intent is interpreted with a certain level of satisfaction, but also is a construct, helping project experts to improve the accuracy of traceability recovery process. In this work, such useful requirements context is extracted and used in the trace link recovery process.

In the following, we introduce the proposed VSM-based context analysis for the post-requirements traceability recovery process in detail. Firstly, an overview of the analysis is given in Section III-A, which is followed by the description of the context analysis of Use Case (UC) requirements in Section III-B. Next Section III-C describes the weighted knowledge model that we used to combine the two generated trace link lists. In order to better illustrate our approach, in this work, we are particularly focused on establishing trace links between UCs and source code files. Some other interesting types of RSL will be considered as part of our future work.

A. Overview of the VSM-based Context Analysis

The main idea of our proposed VSM-based context analysis is to utilize the context information to enhance the accuracy of the candidate trace link list that will be vetted by experts at the end of the RT recovery process. To be specific, its basic approach is summarized by the following three steps:

1) The first step is to obtain the context information and the requirement intent, by analyzing the constructs of the RSL based upon experts' experience. The extracted context information will be further classified into two groups i.e., intent-positive and intent-negative, according to whether or not such context information can help to communicate and understand the requirement intent. This is the core part of our context analysis.
2) After the context analysis, we employ the standard VSM to generate two ranked trace link lists between the requirement and artifacts, in terms of using the intent-positive context query and the intent query (as input), respectively. In doing this, we will get two candidate trace link lists at the end of this step.
3) Finally, the two generated ranked lists are combined together to form a ranked candidate trace link list containing the recalculated similarity scores. This is done by using a *weighted knowledge model*, and the ranked candidate list will be vetted by experts to produce the final trace link list.

Figure 2 shows the detailed work flow of our approach. Note that our approach involves lightweight human intervention in the early phase of the RT recovery process, which is very different from the traditional way that heavyweight human intervention is often involved in the last stage of the RT recovery process. In doing this, we provide the standard VSM with more accurate information to generate better candidate trace link lists. As a result, the final trace link lists can be significantly improved as well as the pertaining human efforts demanded in the final phase of the RT recovery process can be dramatically reduced.

B. Context Analysis of Use Cases

Requirement context consists of whatever constructs surround requirement intent, which are relevant to the requirement

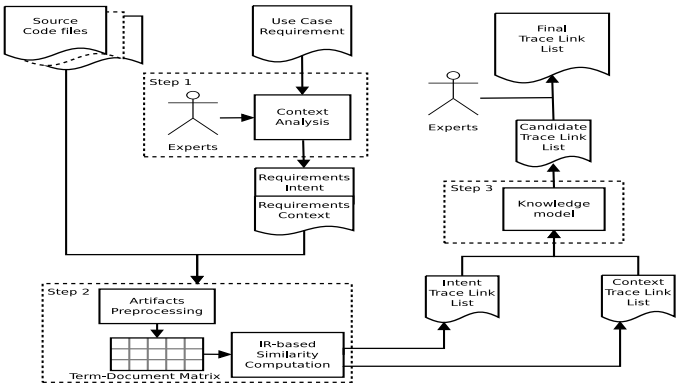

Fig. 2. The work flow about our proposed VSM-based context analysis approach.

interpretation on the subsequent artifacts. Moreover, there are various constructs which are contained by different RSLs, introducing great diversity. In order to avoid the case that the constructs of requirements context become too intractable to be processed in practice, it is essential to restrict the range. Therefore, our context analysis defines the range of context and intent constructs for a given RSL, based around a set of proposed criteria and definitions using experts' experience. Additionally, since the results of our context analysis can be reused for the projects using the same RSL, we consider the human efforts involved in our approach can still be regarded as light-weighted.

The criterion, which we apply to judge if a construct belongs to context or intent, is given below:

Criterion 1. *A construct is considered as the requirement intent if it is used by the system developers to implement the functionalities of a system described by the requirement; it belongs to context otherwise.*

In this work, we choose UC technique as an example, to examine our approach. In general, although different projects or companies may use different structured UC requirements, the constructs of the UCs as in the widely accepted industrial practice [18], can be expressed by Definition 1.

Definition 1. *Use Case constructs of our interest include the title, pre-conditions, flow of events and post-conditions of the use case.*

It is interesting to stress that our current research only focuses on the functional requirements which describe the functionalities of a system. The above definition should thereby be adapted when non-functional requirements are considered.

According to Criterion 1, we give the following definitions of UC intent and UC context used in our work.

Definition 2. *Intent of a Use Case requirement refers to its flow of events.*

Definition 3. *Context of a Use Case requirement refers to its title, pre-condition, and post-condition.*

Further, our definitions are given based around the following train of thoughts:

1) The title of a UC is traditionally named as an active verb phrase. Although its information is not rich enough for system developers to implement the functional requirement, it still can help to provide extra information for traceability recovery process. Therefore, the title construct is defined as context information.
2) The initiation of a UC occurs whenever the pre-conditions are met, and the post-conditions, on the other hand, describe what data need to be stored in the UC. Therefore, they belong to context.
3) The flow of events is the main part of a UC, which defines the relevant functional requirements with all the details. Hence it is considered as the intent of a UC.

However, from the traceability perspective, not all the context information is helpful for the interpretation of requirement intent, we thereby give the following definition of

Criterion 2. *If the construct aims at describing the functionalities of a system, i.e., the requirement intent, the construct is defined as intent-positive. If the construct aims at describing the constraints, extensions, meta information, etc., the construct is defined as intent-negative.*

Based on Criterion 2, the pre-conditions and post-conditions should be classified as intent-negative constructs, since they do not aim at describing the functions of a system. The title of UC is usually associated with some information about a certain functional requirement, it is thereby regarded as intent-positive construct.

C. The Weighted Knowledge Model

Gethers et al.present the weighted knowledge model in [23], which is used in our work. Next we introduce its basic idea in our context: The two trace link lists generated by using requirements context query and requirements intent query (i.e., the context query and the intent query hereafter) are viewed as two knowledge sources, both of which can contribute greatly to address trace link recovery between the requirement and a set of target artifacts. Since the two trace link lists express their judgments from different perspectives (i.e., as either requirements context or requirements intent), their pertaining weights should be considered when they are combined together to obtain a more accurate candidate trace link list.

Formally, such a combination is obtained through two steps. At the first step, the two set of similarity scores of two trace link lists are normalized by using a standard normal distribution, as expressed by Equation 3:

$$sim_{l_i}(q,d)_n = \frac{sim_{l_i}(q,d) - mean(sim_{l_i}(q,D))}{stdev(sim_{l_i}(q,D))} \quad (3)$$

where q represents the query, D represents a set of related artifacts, $d \in D$, $sim_{l_i}(q,d)_n$ is the normalized similarity score of $sim_{l_i}(q,d)$ (where l_i is one of the trace link lists), and $sim_{l_i}(q,D)$ is a set of similarity scores in l_i. The functions $mean()$ and $stdev()$ return the mean and standard deviation of the similarity scores of two trace link lists respectively. Note that such normalization is required to guarantee that the two different sets of similarity scores are commensurable.

At the second step, the normalized scores are combined by using the following weighted knowledge model, as shown in Equation 4:

$$sim(cq,iq,d)_c = \lambda \times sim_{l_i}(cq,d)_n + (1-\lambda) \times sim_{l_j}(iq,d)_n \quad (4)$$

where cq and iq represent the context query and intent query, $\lambda \in [0,1]$ expresses the confidence in each query. The higher the value the higher confidence gives by the technique. In our evaluation, we find the value of λ to be 0.3 (as the weight of context query), which usually produces good combined similarity scores.

In a nutshell, our context analysis in practice provides the standard VSM with both requirements intent query and requirements context query. Typically, such a combination contains more enhanced semantics, which can accurately represent the intent of the requirement. As a result, the accuracy of the standard VSM toward recovering true trace links can be improved.

D. Algorithm Description

The precise description of the algorithm using pseudo-code is outlined in Algorithm 1, which takes four parameters and returns a ranked list of candidate trace links at the end of its execution.

Parameters:
D: list - the collection of source code files D
uc: string - the UC requirement uc
iq: string - the intent query iq of the use case uc
cq: string - the context query cq of the use case uc
Returns:
$LIST_{vsm-ca}$: list - the ranked candidate trace link list between the UC requirement uc and the source code files D, containing the combined similarity scores.

Algorithm 1 $VSM-CA(D, uc, iq, cq)$

```
1: m ← 0, D' ← ∅
2: D ← d_1, d_2, ..., d_{n-1}, d_n
3: cq ← context(uc)
4: iq ← intent(uc)
5: for all d_i ∈ D such that 1 ≤ i ≤ n do
6:     sim_{cq,d_i} ← sim(cq, d_i)
7:     sim_{iq,d_i} ← sim(iq, d_i)
8:     if sim_{cq,d_i} ≠ 0 or sim_{iq,d_i} ≠ 0 then
9:         m ← m + 1
10:        D' ← D' ⋃ {d_i}
11:    end if
12: end for
13: for all d'_i ∈ D' such that 1 ≤ i ≤ m do
14:    sim_{cq,d'_i} ← sim_{l_1}(cq, d'_i)_n
15:    sim_{iq,d'_i} ← sim_{l_2}(iq, d'_i)_n
16:    sim(uc, d'_i) ← sim(cq, iq, d'_i)_c
17:    if sim(uc, d'_i) ≥ threshold then
18:        LIST_{vsm-ca} ← LIST_{vsm-ca} ⋃ {sim(uc, d'_i)}
19:    end if
20: end for
21: return LIST_{vsm-ca}
```

IV. Empirical Evaluation

This section describes the evaluation carried out to assess the improvement given by our VSM-based context analysis over the standard VSM, comprising six parts. Section IV-A introduces the evaluation setup, and Section IV-B formulates

TABLE I. CHARACTERISTICS OF THE THREE DIFFERENT EMBEDDED SOFTWARE APPLICATIONS USED IN OUR EVALUATION.

System	KLOC	UCs	Source code files	True links
iRobot	2706	21	20	45
iTruck	952	24	14	37
iSudoku	9285	18	54	51

our research questions for evaluation. Our evaluation metrics and the corresponding results are presented in Section IV-C and Section IV-D respectively. Finally, we summarize the evaluation by highlighting some interesting observations in Section IV-E, before we give our view of validity of results in regard to some possible threats in Section IV-F.

A. Definitions and Context

The goal of the evaluation is to provide the evidence that our VSM-based context analysis can obtain better results over the standard VSM, when it is used for trace link recovery in embedded applications.

The context of our evaluation is represented by three different embedded software applications developed at Mälardalen University in Sweden, which are: 1) one industrial robotic control system *iRobot* and, 2) one truck navigation system *iTruck* and, 3) one embedded mobile application *iSudoku*. Specifically, iRobot is a C program, which models a robotic control application containing complicated timing behavior, and it has been designed and evaluated in [24]. iTruck is also a C program, which is developed for modeling a truck navigation system by using SaveComp Component Model (SaveCCM) [25]. iSudoku is a Sudoku game application developed in Java for the Android platform. In addition, all the three applications have different number of UCs, source code files and true trace links, as shown in Table I. One example of the UCs in iSudoku, with its title, pre-condition, flow of events and post-condition, is shown by Figure 3. Moreover, all the relevant files of the three case studies, e.g., UCs, source code files, are available upon request.

For implementation, we use the Lucene library [26], which is the well-known VSM with tf-idf weighting scheme, and has been considered as the default IR model by many pieces of work [15], [16], [27]. Our testbed is running Mac OS X, version 10.6.8, and the computer is equipped with the Intel Core Duo CPU i7 processor, 4GB RAM and a 256KB L2 Cache. The processor has four cores and one frequency level: 2.2 GHz.

```
List Puzzles on Screen Use Case
  Pre-conditions:
    The game is initiated.
  Flow of events:
    The application loads all the puzzles stored in the
    local database.
    The puzzles and their corresponding folders can be
    listed on the screen one by one.
  Post-conditions:
    List variables is initiated.
```

Fig. 3. An example shows one use case in iSudoku.

B. Research Questions

In this study, we aim at addressing the following research question:

- Can our VSM-based context analysis approach obtain better quality trace link lists, compared with the standard VSM?

To answer this question, we plan to use three well-known IR metrics for results comparison (to be introduced in the following section).

C. Metrics

There are many different measures for evaluating the accuracy of IR methods. In this work, we use three well-known IR metrics, i.e., recall, precision and mean average precision (MAP) [21]. Specifically, *recall* shows the ratio of the number of relevant documents retrieved by the method over the total number of relevant documents, and 100% recall means that all relevant documents were retrieved. *Precision* is the fraction of the relevant documents retrieved over the total number of the retrieved documents, and 100% precision means that all the retrieved documents are relevant ones, though there could be some relevant links that were not discovered. Recall and precision can be expressed by Equation 5 and Equation 6 as follows:

$$recall = \frac{|D_{rel} \cap D_{ret}|}{|D_{rel}|} \tag{5}$$

where D_{rel} represents the collection of source code files that are relevant to a UC, and D_{ret} is the collection of the retrieved source code files using a certain IR technique.

$$precision = \frac{|D_{rel} \cap D_{ret}|}{|D_{ret}|} \tag{6}$$

where D_{rel} represents the collection of source code files which are relevant to a UC, and D_{ret} is the collection of the retrieved source code files.

Another evaluation metric MAP, as one of the most frequently used IR measures, considers the rank of the retrieved trace links. The higher the MAP score is, the better quality of the retrieved ranked list of trace links is, in terms of requirements relevance. In particular, given a collection of UCs as queries Q, and a set of related source code files as documents D_a, the MAP score of the ranked list L of retrieved documents for the given query q, is defined as below:

$$MAP = \frac{1}{|Q|} \sum_{q=1}^{Q} \frac{1}{|D_a|} \sum_{d=1}^{D_a} SCORE_{rank}(d, L) \tag{7}$$

where $SCORE_{rank}(d, L)$ is the ranking score of the document d in the list L. The higher the rank of d is, the larger ranking score the document has.

D. Analysis of Results

In this section, We investigate whether the accuracy of our VSM-based context analysis approach is superior to that of the standard VSM approach.

1) Improvement of Recall and Precision Scores given by our VSM-based Context Analysis: *Recall* and *precision* frequently exist in a state of mutual tension. For example, 100% recall can be achieved simply by returning all possible links. This may result in a very low level of precision, which is not so useful in practice [21]. When choosing any traceability recovery approach, the end-users should consider which one between *recall* or *precision* is preferred. For example, for safety critical projects, *recall* will probably be more important, since the end-users will not want to run the risk of missing any true link. On the other hand, in most cases, a non-safety critical project with a short time-to-market may prefer to favor *precision* [21], i.e., the end-users will expect that the traceability recovery approach can obtain more true links when they just have time to check part of the candidate trace link list. It should be pointed out that the level of recall at 90% is a common choice at which the precision scores are compared to show improvements [6], [23]. Figure 4 provides the precision/recall curves achieved by our approach and the standard VSM. As shown by the figure, we have one case (i.e., iRobot) where our approach outperforms for all levels of recall, and there are two other cases (i.e., iSudoku and iTruck) where the precision scores of our approach are much better than that acquired by the standard VSM approach when the level of recall is lower than 90%, which means that our approach would be more suitable for the embedded systems which prefer *precision* rather than *recall*.

2) Improvement of MAP Scores given by our VSM-based Context Analysis: Table II shows the improvement of MAP scores given by our method over the standard VSM, for all the three case studies. It is also interesting to stress that we only show the average of the MAP scores of all the UCs for a certain case study in the table, for the sake of space. Moreover, we also present such improvements in terms of percentages (i.e., $\frac{MAP_{vsm-ca}-MAP_{vsm}}{MAP_{vsm}}$, where $vsm\text{-}ca$ is our VSM-based context analysis) in Column *Imprv. %* in Table II. As shown in the table, the most significant improvement achieved by our approach is 28.0%, comparing the standard VSM.

TABLE II. OUR PROPOSED VSM-BASED CONTEXT ANALYSIS CAN RETRIEVE HIGHER MAP SCORES, COMPARING THE STANDARD VSM.

System	AVG of MAP VSM	AVG of MAP VSM-CA	Imprv. %
iRobot	0.717	0.733	2.23%
iTruck	0.660	0.762	15.4%
iSudoku	0.547	0.700	28.0%

E. Experiments Summary

Summarizing the above observations, our evaluation results have confirmed the following points:

1) Our proposed approach can help to recover more accurate trace links than the standard VSM, in the sense of obtaining higher precision scores corresponding to certain levels of recall.
2) Our proposed approach can help to recover more accurate trace links than the standard VSM, in the sense of obtaining higher MAP scores.
3) The computing time required by the trails of our approach, on average took only a few minutes to compute. This is an important step toward handling real life-scale requirements traceability problems.

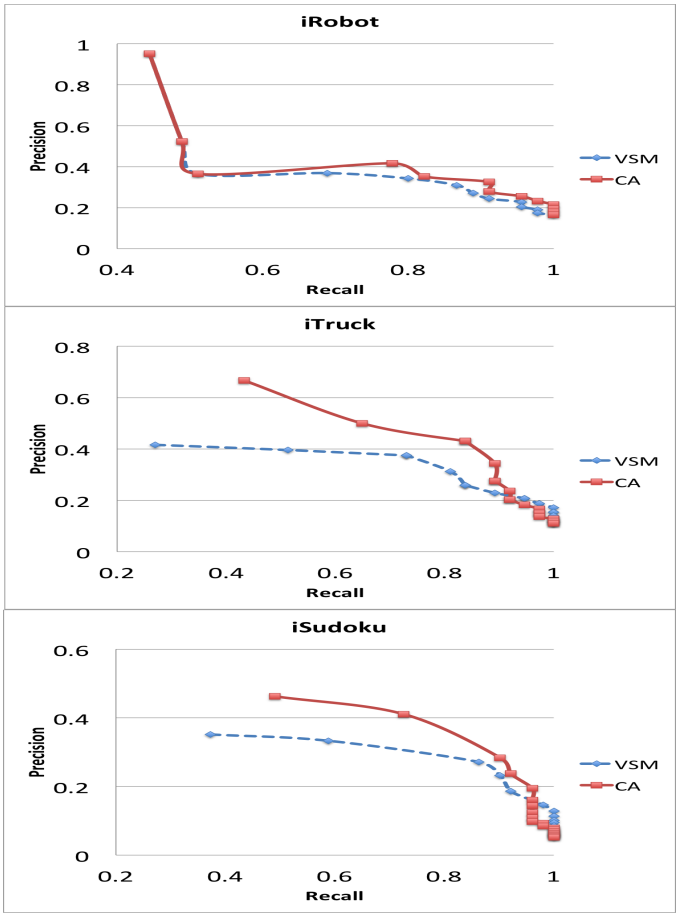

Fig. 4. The precision/recall curves of our approach and the standard VSM approach, in the order of iRobot, iTruck and iSudoku from top to bottom. In addition, the curves in dashed lines are for the standard VSM.

F. Threats to Validity

In this section, we discuss the threats that can impact on the validity of our evaluation, from the following four perspectives [31]. The first category is *construct validity*, concerning the degree to which the study metrics accurately measure the concepts. The metrics used in our evaluation, i.e., *recall*, *precision* and *MAP*, have been widely adopted for assessing the traceability accuracy of IR methods. Therefore, we believe that they can sufficiently quantify the accuracy of two compared IR methods. The second category is *internal validity*, referring to the extent to which a treatment changes what is measured in the experiment. The internal validity of our experiment can only be affected by the chosen value of the parameter λ in the employed knowledge model. We choose the value of λ based on our empirical evidence. In the future, we will obtain the optimal value of λ by using some advanced techniques, such as optimization and machine learning. The third category is *external validity*, related to the extent to which we can generalize the study results. The reason is that different systems with various requirements and subsequent artifacts may lead to different results. In order to reduce the threats to the external validity, we have chosen three embedded software applications in different domains. Last but not least, the fourth category *conclusion validity* concerns if our evaluation observations can be supported by

some valid statistical techniques as evidences. In this stage, we just visualize our results to illustrate our improvement. In the future, this will be done by using certain non-parametric or parametric statistical tests, such as Wilcoxon signed-rank test and ANOVA.

V. Conclusions and Future Work

In this paper, we have proposed a new Vector Space Model (VSM)-based approach for post-requirements traceability recovery, which uses a novel context analysis. Specifically, our approach utilizes context information featured by requirement context and requirement intent, to build trace links between use cases and relevant source code files, through a weighted knowledge model. We have evaluated the approach by using three different embedded applications in industrial automation, automotive and mobile. The experiment results have shown that our approach can obtain better quality candidate trace link lists, in terms of higher scores of three main IR metrics, i.e., *recall*, *precision*, and *MAP*, comparing the standard VSM approach.

For future work, we will improve the evaluation part by providing some statistical evidence with statistical hypothesis test, which can be conducted by performing, e.g., Wilcoxon signed-rank test with Monte Carlo permutation. Moreover, we will consider to determine the optimal value of the parameters in the weighting schema for combining two ranked trace link lists in the analysis. We also plan to apply some user feedback technique to our context analysis, which would also result in some interesting discoveries. The investigation of using other automated information retrieval methods, instead of VSM, together with our context analysis, as well as the completion of some extensive evaluation by using more datasets are also highly appreciated on our good side.

Acknowledgements

This work was partially supported by the Swedish Research Council (VR), Strål Säkerhets Myndigheten (SSM) and School of Innovation, Design and Engineering, Mälardalen University.

References

[1] O. Gotel and A. Finkelstein, "An analysis of the requirements traceability problem," in *Proceedings of RE'94*, pp. 94–101, 1994.

[2] J. Kraft, Y. Lu, C. Norström, and A. Wall, "A metaheuristic approach for best effort timing analysis targeting complex legacy real-time systems," in *Proceedings of RTAS'08*, pp. 258–269, 2008.

[3] B. Ramesh and M. Jarke, "Toward reference models for requirements traceability," *IEEE TSE*, vol. 27, no. 1, pp. 58–93, Jan. 2001.

[4] S. Deerwester, S. Dumais, T. Landauer, G. Furnas, and L. Beck, "Improving information retrieval with latent semantic indexing," in *Annual Meeting of the American Society for Info. Science 25*, 1988.

[5] D. Blei, A. Ng, and M. Jordan, "Latent dirichlet allocation," *Journal of Machine Learning Research*, no. 3, pp. 993–1022, 2003.

[6] J. Cleland-Huang, R. Settimi, C. Duan, and X. Zou, "Utilizing supporting evidence to improve dynamic requirements traceability," in *Proceedings of RE'05*, pp. 135–144, 2005.

[7] N. Ali, Y.-G. Gueheneuc, and G. Antoniol, "Trust-based requirements traceability," in *Proceedings of ICPC'11*, 2011, pp. 111–120.

[8] W. Kong and J. Hayes, "Proximity-based traceability: an empirical validation using ranked retrieval and set-based measures," in *Proceedings of EmpiRE'11*, pp. 45–52, 2011.

[9] A. D. Lucia, M. Penta, R. Oliveto, A. Panichella, and S. Panichella, "Improving ir-based traceability recovery using smoothing filters," in *Proceedings of ICPC'11*, vol. 0, pp. 21–30, 2011.

[10] A. Mahmoud, N. Niu, and S. Xu, "A semantic relatedness approach for traceability link recovery." in *Proceedings of ICPC'12*, pp.183–192, 2012.

[11] O. Gotel, J. Cleland-Huang, J. Hayes, A. Zisman, A. Egyed, P. Grünbacher, A. Dekhtyar, G. Antoniol, and J. Maletic, *The grand challenge of traceability (v1.0)*. Springer, pp. 343–412, 2012.

[12] A. D. Lucia, R. Oliveto, P. Sgueglia, "Incremental approach and user feedbacks: a silver bullet for traceability recovery," in *Proceedings of ICSM'06*, pp. 299–309, 2006.

[13] G. Antoniol, G. Canfora, G. Casazza, A. D. Lucia, and E. Merlo, "Recovering traceability links between code and documentation," *IEEE TSE*, vol. 28, no. 10, pp. 970–983, 2002.

[14] G. Salton, A. Wong, and C. Yang, "A vector space model for automatic indexing," *Communications of the ACM*, vol.18,no.11, pp. 613–620, 1975.

[15] A. Abadi, M. Nisenson, and Y. Simionovici, "A traceability technique for specifications," in *Proceedings of ICPC'08*, pp. 103–112, 2008.

[16] C. Fautsch and J. Savoy, "Adapting the tf-idf vector-space model to domain specific information retrieval," in *Proceedings of SAC'10*, pp. 1708–1712, 2010.

[17] A. Dekhtyar, O. Dekhtyar, J. Holden, J. Hayes, D. Cuddeback, and W. Kong, "On human analyst performance in assisted requirements tracing: statistical analysis," in *Proceedings of RE'11*, pp. 111–120, 2011.

[18] E. Nasr, J. McDermid, and G. Bernat, "Eliciting and specifying requirements with use cases for embedded systems," in *Proceedings of WORDS'02*, IEEE Computer Society, 2002.

[19] A. Marcus and J. I. Maletic, "Recovering documentation-to-source-code traceability links using latent semantic indexing," in *Proceedings of ICSE'03*, pp. 125–135, 2003.

[20] R. Oliveto, M. Gethers, D. Poshyvanyk, and A. D. Lucia, "On the equivalence of information retrieval methods for automated traceability link recovery," in *Proceedings of ICPC'10*, pp. 68–71, 2010.

[21] C.-H. Jane, G. Orlena, and Z. Andrea, *Software and systems traceability*. Springer, 2012.

[22] J. H. Connolly, "Context in functional discourse grammar," *Alfa : Revista de Linguística*, vol. 51, pp. 11–33, 2009.

[23] M. Gethers, R. Oliveto, D. Poshyvanyk, and A. D. Lucia, "On integrating orthogonal information retrieval methods to improve traceability recovery," in *Proceedings of ICSM'11*, pp. 133–142, Sept. 2011.

[24] Y. Lu, T. Nolte, I. Bate, and L. Cucu-Grosjean, "A statistical response-time analysis of real-time embedded systems," in *Proceedings of RTSS'12*, pp. 351–362, Dec. 2012.

[25] M. Åkerholm, J. Carlson, J. Fredriksson, H. Hansson, J. Håkansson, A. Möller, P. Pettersson, and M. Tivoli, "The save approach to component-based development of vehicular systems," *Journal of Systems and Software*, vol. 80, no. 5, pp. 655–667, May 2007.

[26] E. Hatcher, O. Gospodnetic, and M. McCandless, *Lucene in Action*, 2nd ed., 2010.

[27] H. U. Asuncion, A. U. Asuncion, and R. N. Taylor, "Software traceability with topic modeling," in *Proceedings of ICSE'10*, pp. 95–104, 2010.

[28] D. Moore, G. Mccabe, and B. Craig, *Introduction to the practice of statistics*, 6th ed. New York, W. H. Freeman and Company, 2009.

[29] S. Stigler, "Fisher and the 5% Level," *Journal of CHANCE*, vol. 21, no. 4, p. 12, 2008.

[30] XLSTAT, http://www.xlstat.com/en/, 2013-04-17.

[31] B. Kitchenham, L. Pickard, and S. L. Pfleeger, "Case studies for method and tool evaluation," *IEEE Softw.*, vol. 12, no. 4, pp. 52–62, Jul. 1995.