# Investigating the Identification of Technical Debt Through Code Comment Analysis

Mário André de Freitas Farias[1,2(✉)], José Amâncio Santos[3], Marcos Kalinowski[4], Manoel Mendonça[2], and Rodrigo Oliveira Spínola[5,6]

[1] Federal Institute of Sergipe, Lagarto, Sergipe, Brazil
mario.andre@ifs.edu.br
[2] Federal University of Bahia, Salvador, Bahia, Brazil
manoel.g.mendonca@gmail.com
[3] State University of Feira de Santana, Feira de Santana, Bahia, Brazil
zeamancio@gmail.com
[4] Fluminense Federal University, Rio de Janeiro, Brazil
kalinowski@ic.uff.br
[5] Fraunhofer Project Center at UFBA, Salvador, Bahia, Brazil
rodrigoospinola@gmail.com
[6] Salvador University, Salvador, Bahia, Brazil

**Abstract.** In order to effectively manage technical debt (TD), a set of indicators has been used by automated approaches to identify TD items. However, some debt items may not be directly identified using only metrics collected from the source code. CVM-TD is a model to support the identification of technical debt by considering the developer point of view when identifying TD through code comment analysis. In this paper, we investigate the use of CVM-TD with the purpose of characterizing factors that affect the accuracy of the identification of TD, and the most chosen patterns by participants as decisive to indicate TD items. We performed a controlled experiment investigating the accuracy of CVM-TD and the influence of English skills and developer experience factors. We also investigated if the contextualized vocabulary provided by CVM-TD points to candidate comments that are considered indicators of technical debt by participants. The results indicated that CVM-TD provided promising results considering the accuracy values. English reading skills have an impact on the TD detection process. We could not conclude that the experience level affects this process. We identified a list of the 20 most chosen patterns by participants as decisive to indicate TD items. The results motivate us continuing to explore code comments in the context of TD identification process in order to improve CVM-TD.

**Keywords:** Contextualized vocabulary · Technical debt · Code comment · Controlled experiment

## 1 Introduction

The Technical Debt (TD) metaphor reflects the challenging decisions that developers and managers need to take in order to achieve short-term benefits. These decisions may not cause an immediate impact on the software, but may negatively affect the long-term

© Springer International Publishing AG 2017
S. Hammoudi et al. (Eds.): ICEIS 2016, LNBIP 291, pp. 284–309, 2017.
DOI: 10.1007/978-3-319-62386-3_14

health of a software system or maintenance effort in the future [1]. The metaphor is easy to understand and relevant to both technical and nontechnical practitioners [2, 3]. In this sense, its acceptance and use have increased in software engineering researches.

In order to effectively manage TD, it is necessary to identify TD items[1] in the project [4]. Li *et al.* [5], in a recent systematic review, reported that code quality analysis techniques have frequently been studied to support the identification of TD. Automatic analysis tools have used software metrics extracted from the source code to identify TD items by comparing values of software metrics to predefined thresholds [6]. Although these techniques have shown useful to support the automated identification of some types of debt, they do not cover human factors (e.g., tasks commented as future work) [7, 8]. Thus, large amounts of TD that are undetectable by tools may be left aside. In this sense, pieces of code that need to be refactored to improve the quality of the software may continue unknown. To complement the TD identification with more contextual and qualitative data, human factors and the developers' point of view should be considered [9].

In this context, Potdar and Shihab [8] have focused on code comments aiming to identify TD items. Therefore, they manually read more than 101 K code comments to detect those that indicate a self-admitted TD. These comments were analyzed to identify text patterns that indicate a TD. In the same way, Maldonado and Shihab [10] have read 33 K code comments to identify different types of debt using the indicators proposed by [11]. According to the authors, these patterns can be used to manually identify TD that exists in the project by reading code comments. However, it is hard to perform such a large manual analysis in terms of effort and the process is inherently error prone.

Farias *et al.* [9] presented a Contextualized Vocabulary Model for identifying TD (CVM-TD) based in code comments. CVM-TD uses word classes and code tags to provide a set of TD terms/patterns of comment (a contextualized vocabulary) that may be used to filter comments that need more attention because they may indicate a TD item. CVM-TD was evaluated through an exploratory study on two large open sources projects, jEdit and Lucene, with the goal of characterizing its feasibility to support TD identification from source code comments. The results pointed that the dimensions (e.g. Adjectives, Adverbs, Verbs, Nouns, and Tags) considered by the model are used by developers when writing comments and that CVM-TD can be effectively used to support TD identification activities.

These promising initial outcomes motivated us to evaluate CVM-TD further with other projects. Therefore, in this paper we extend the research of Farias *et al.* [9] with an additional study to analyze the use of CVM-TD and the contextualized vocabulary with the purpose of characterizing its overall accuracy when classifying candidate comments and factors that influence the analysis of the comments to support the identification of TD in terms of accuracy.

We address this research goal by conducting a controlled experiment to investigate the overall CVM-TD accuracy and the influence of the English skills and developer experience factors. We analyzed the factors against the accuracy by observing the

[1] The term "TD item" represents an instance of Technical Debt.

agreement between each participant and an Oracle elaborated by the researchers. We compared the accuracy values for the different factors using statistical tests.

We also analyzed the agreement among the participants. These aspects are decisive to understand and validate the model and the contextualized vocabulary. Our findings indicate that CVM-TD provided promising results considering the accuracy values. The accuracy values of the participants with good reading skills were better than the values of the participants with medium/poor reading skills. We could not conclude that the experience level affects the accuracy when identifying TD items through comment analysis. The results indicate that 89.21% of the TD comments that were chosen by Oracle were also chosen by at least 50% of all participants. We also observed that, on average, the participants identified many comments that were filtered by the vocabulary and selected as a TD indicator by Oracle. This means that the vocabulary proposed by CVM-TD helped the participants to comprehend and identify comments that may point out TD items. We also identified the 20 most chosen patterns by participants as decisive to indicate TD items.

The remainder of this paper is organized as follows. Section 2 presents relevant literature reporting on technical debt identification approaches and the use of comments in source code. Section 3 describes the planning of the controlled experiment. Section 4 presents its operation. The results are presented in Sect. 5. Next, we have a discussion section. Finally, Sect. 7 concludes the paper.

## 2 Background

### 2.1 Code Comments Mining

Comments are an important software artifact which may help to understand software features [12]. Code comments have been used as data source in some research [12, 13].

In [13], the authors analyzed the purpose of work descriptions and code comments aiming to discuss how automated tools can support developers in creating them.

Storey *et al.* [12] analyzed how developers deal with software maintenance tasks by conducting an empirical study investigating how comments may improve processes and tools that are used for managing these tasks.

In fact, comments have been used to describe issues that may require future work, emerging problems, and decisions taken about those problems [13]. These descriptions facilitate human readability and provide additional information that summarizes the developer context [9].

Therefore, code comments are considered a vital documentation used in maintenances. They complete the general documentation of a system and are a conventional way for developers keep documentation and code consistently up to date [14].

Despite of the existence of different syntaxes and types of comments according to the programming language, they are usually divided into two categories: (i) inline comments, which only permit the insertion of one line of comment, and (ii) block comments, which permit the insertion of several lines. Developers write comments in a sublanguage of English using a limited set of verbs and tenses, and personal pronouns are almost not used [15].

According to Steidl et al. [14], the comments within these categories can be classified in seven different types:

- **Copyright Comments:** notes that include information about the copyright or the license of the source code file. They are usually found at the beginning of each file;
- **Header Comments:** give an overview about the functionality of the class and provide information about the class author, the revision number, or the peer review status.
- **Member Comments:** describe the functionality of a method/field, being located either before or in the same line as the member definition. This type of comment is similar to header comments.
- **Inline Comments:** describe implementation decisions within a method body. This type was named as the same description of the category of comments presented above.
- **Section Comments:** this comments address several methods/fields together belonging to the same functional aspect.
- **Code Comments:** contain commented out code which is source code ignored by the compiler. Often code is temporarily commented out for debugging purposes or potential later reuse.
- **Task Comments:** are a developer note containing a to-do statement, a note about a bug that needs to be fixed, or a remark about an implementation hack.

In this work we are interested only in the **inline comments** and **task comments** types because they can describe the developer's feeling about open tasks in the project and the risk of causing problems if not done in the future.

### 2.2 Using Code Comments to Identify TD

More recently, code comments have been explored with the purpose of identifying TD [8–10, 16].

Potdar and Shihab [8] analyzed code comments to identify text patterns and TD items. They read more than 101 K code comments. Their findings show that 2.4–31.0% of the files in a project contain self-admitted TD. In addition, the most used text patterns were: (i) "is there a problem" with 36 instances, (ii) "hack" with 17 instances, and (iii) "fixme" with 761 instances.

In another TD identification approach, Maldonado and Shihab [10] evolved the work of Potdar and Shihab [8] proposing four simple filtering heuristics to eliminate comments that are not likely to contain technical debt. For that, they read 33 K code comments from source code of five open source projects (Apache Ant, Apache Jmeter, ArgoUML, Columba, and JFreeChart). Their findings showed that self-admitted technical debt can be classified into five main types: design debt, defect debt, documentation debt, requirement debt, and test debt. According to the authors, the most common type of self-admitted TD is design debt (between 42% and 84% of the classified comments).

In the same sense, Bavota and Russo [16] performed a replication of the work of Potdar and Shihab [8]. Their study reported an empirical analysis conducted on 159

software projects to investigate the diffusion and evolution of TD. The authors mined commits and code comments using the patterns proposed by [8] and carried out a qualitative analysis. Their results show that the diffusion of TD in OSS projects is, on average, 51 instances per system, the TD instances have a long survivability, on average, more than 1,000 commits, and have an increasing trend over the projects lifetime.

In an another approach, Farias *et al.* [9] proposed the CVM-TD. CVM-TD is a contextualized structure of terms that focuses on using word classes and code tags to provide a TD vocabulary, aiming to support the detection of different types of debt through code comment analysis. To evaluate the model and quickly analyze developers' comments embedded in source code, the *eXcomment* tool was developed. This tool extracts and filters candidate comments from source code using the contextualized vocabulary provided by CVM-TD.

This research provided preliminary indication that CVM-TD and its contextualized vocabulary can be effectively used to support TD identification (the whole vocabulary can be found at https://goo.gl/TH2ec5). However, the factors that may affect its accurate usage are still unknown. In this work, we focused on characterizing CVM-TD's accuracy and some of these factors.

## 3 Study Planning

### 3.1 Goal of Study and Research Questions

This study aims at investigating the following goal: "Analyze the use of CVM-TD with the purpose of characterizing its overall accuracy and factors affecting the identification of TD through code comment analysis, with respect to accuracy when identifying TD items from the point of view of the researcher in the context of software engineering master students with professional experience analyzing candidate code comments of large software projects". More specifically, we investigated five Research Questions (RQ). The description of these RQs follows.

***RQ1:*** *Do the English reading skills of the participant affect the accuracy when identifying TD through code comment analysis?*

Considering that non-native English speakers are frequently unaware of the most common terms used to define specific parts of code in English [17], this question aims to investigate whether a different familiarity with the English language could impact the identification of TD through code comment analysis. In order to analyze this variable, we split the participants into levels of "good English reading skills" and "medium/poor English reading skills". This question is important to help us to understand the factors that may influence the analysis of comments to identify TD.

***RQ2:*** *Does the experience of the participant affect the accuracy when identifying TD through code comment analysis?*

Experience is an important contextual aspect in the software engineering area [18]. Recent research has studied the impact of experience on software engineering experiments [19]. Some works have found evidence that experience affects the identification of code smells, and that some code smells are better detected by experienced

participants rather than by automatic means [20]. Considering this context, this question aims to discuss the impact of the participants' experience on the identification of TD through code comment analysis. To analyze the variable, we classified the participants into three levels considering their experience with software development: (i) high experience, (ii) medium experience, and (iii) low experience. This question is also important to help us to understand the factors that may influence the analysis of comments to identify TD.

***RQ3:*** *Do participants agree with each other on the choice of comments filtered by CVM-TD that may indicate a TD item?*

With this question, we intend to investigate the contribution of CVM-TD in the TD identification process and how many and what comments had high level of agreement. That is, what comments point out to a TD item. This question will also allow us to analyze the agreement among the participants about the candidate comments that indicate a TD item. We conjecture that high agreement on the choice of comments filtered by CVM-TD evidences its relevance as a support tool on the TD identification.

***RQ4:*** *Does CVM-TD help researchers on select candidate comments that point to technical debt items?*

With this question, we intend to investigate if the contextualized vocabulary provided by CVM-TD points to candidate comments that are considered indicators of technical debt by researchers. This question will also allow us to investigate the contribution of CVM-TD to support the TD identification process.

***RQ5:*** *What were the most chosen code comment patterns by participants as decisive to indicate a TD item?*

In this question, we intend to investigate which patterns were considered more decisive to help participants on identifying comments that report a situation of TD. For each comment marked as yes, the participants highlighted the part of the comment that was decisive for their answer. The answer for this question will allow us to classify the most decisive patterns and closing a feedback cycle expanding the vocabulary by inserting new patterns.

### 3.2 Participants

The participants of the study were selected using convenience sampling [21]. Our sample consists of 21 software engineering master students at the Federal University of Sergipe (Sergipe-Brazil) and 15 software engineering master students at the Salvador University (Bahia-Brazil). We conducted the experiment in the context of the Empirical Software Engineering Course.

In order to classify the profile of the participants and their experience in the software development process, a characterization form was filled by each participant before the experiment. The questions were about professional experiences, English reading skills, and specific technical knowledge such as refactoring and programming languages. The result of the questionnaire showed that participants had a heterogeneous experience level, but all had some experience on software projects.

The participants were classified into three experience levels (high, medium and low) regarding the experience variable and the classification proposed by [18], which is

presented in Table 1. We discarded the category E1 because there were not any undergraduate students as participants. We considered low experience for participants related to the categories E2 and E3. The participants related to the category E4 were considered as having medium experience, and, finally, we considered the participants related to category E5 as having high experience.

**Table 1.** Classification of the experiences of participants.

| Category | Description | Experience levels |
|---|---|---|
| E1 | Undergraduate student with less than three months of recent industrial experience | – |
| E2 | Graduate student with less than three months of recent industrial experience | Low |
| E3 | Academic with less than three months of recent industrial experience | Low |
| E4 | Any person with recent industrial experience between 3 months and two years | Medium |
| E5 | Any person with recent industrial experience for more than two years | High |

When considering the English reading skills, the participants were classified into two levels (good and medium/poor). We had 4 participants with poor English reading skills and 21 participants with medium. Despite these participants have been selected as medium/poor English, they may understand short sentences like code comments in English. Table 2 shows the characterization of the participants.

**Table 2.** Distribution of the participants.

| Group | Participants by experience level | | | Participants by English reading level | |
|---|---|---|---|---|---|
| | High | Med | Low | Good | Med/Poor |
| G1 (12) | 4 | 3 | 5 | 1 | 11 |
| G2 (12) | 3 | 5 | 4 | 5 | 7 |
| G3 (12) | 4 | 5 | 3 | 5 | 7 |
| Total (36) | 11 | 13 | 12 | 11 | 25 |

The participants were split into three groups. Each group had 12 participants with approximately the same levels of experience. This strategy provides a balanced experimental design. The design involved each group of participants working on a different set of comments (experimental object) and permits us to use statistical test to study the effect of the investigated variables. We adopted this plan to avoid an excessive number of comments to be analyzed by each participant.

### 3.3 Instrumentation

**Forms.** The experimental package is available at https://goo.gl/DdomGk. We used slides for the training and four forms to perform the experiment:

*Consent Form:* the participants authorize their participation in the experiment and indicate to know the nature of the procedures which they have to follow.

*Characterization Form:* contains questions to gather information about professional experiences, English reading skills, and specific technical knowledge of participants.

*Data collect Form*: contains a list of source code comments. During the experiment, the participants were asked to indicate, for each comment, if it points to a TD item.

*Feedback Form:* in this form, the participants may write their impression on the experiment. We also asked the participants to classify the training and the level of difficulty in performing the study tasks.

**Software Artifact and Candidate Comments.** We gathered and filtered comments from a large and well-known open source software (ArgoUML). The project is written in Java with 1,846 files. In choosing this project, we considered the following criteria: being long-lived (more than ten years), having a satisfactory number of comments (more than 2,000 useful comments).

To be able to extract the candidate comments from the software that may indicate a TD item, we used *eXcomment*. We were only interested in comments that have been intentionally written by developers [9].

Once the comments were extracted, we filtered the comments by using terms that belong to the vocabulary presented in [9]. A comment is returned when it has at least one keyword or expression found in the vocabulary. We will call these comments 'candidate comments'. At the end, the tool returned 353 comments, which were listed in the collect data form in the same order in which they are in the code. This is important because comments that are close to each other can have some relationship.

### 3.4 Analysis Procedure

We considered three perspectives to analyze accuracy:

**(i) Agreement between each participant and the Oracle:** In order to investigate RQ1 and RQ2, we adopted the accuracy measure, which is the proportion of true results (the comments chosen in agreement between each participant and the Oracle) and the total number of cases examined (see Eq. 1).

$$accuracy = (num\,TP + num\,TN)/ (num\,TP + num\,FP + num\,TN + num\,FN) \quad (1)$$

TP represents the case where the participant and the Oracle agree on a TD comment (comment that points to a TD item). FP represents the case where the participant disagrees with the Oracle on the selected TD comment. TN occurs when the participant and the Oracle agree on a comment that does not report a TD item. Finally, a FN happens when the participant does not mark a TD comment in disagreement with the Oracle.

The definition of the Oracle, which represents an important aspect of this analysis process, was performed before carrying out the experiment. We relied on the presence of three specialists in TD. Two of the specialists did, in separate, the indication of the comments that could point out to a TD item. After, the third specialist did a consensus process for the set of the chosen comments. All this process took one week.

**(ii) Agreement among the Participants:** To analyze RQ3, we adopted the Finn coefficient [22]. The Finn coefficient is used to measure the level of agreement among participants. In order to make the comparison of agreement values, we adopted classification levels, as defined by [23], and recently used by [20]: slight, for values between 0.00 and 0.20; fair (between 0.21 and 0.40); moderate (between 0.41 and 0.60); substantial (between 0.61 and 0.80); and almost perfect (between 0.81 and 1.00) agreement.

**(iii) TD Comments selected by Oracle and Participants:** To analyze RQ4, we investigated the candidate comments that point to TD items selected by the Oracle and participants. We also identified and analyzed the false positives (*i.e.*, comments that do not report a TD item).

**(iv) The most chosen Patterns by Participants as decisive to indicate a TD item:** To answer the RQ5, we analyzed the patterns that were chosen by participants as decisive patterns to identify a TD item through code comments analysis.

### 3.5 Pilot Study

Before the experiment, we carried out a pilot study with a computer science Ph.D. student with professional experience. The pilot took 2 h and was carried out in a Lab at the Federal University of Bahia (Bahia-Brazil). We performed the training at first hour, and next the participant performed the experimental task described in the next section. The participant analyzed 83 comments and selected 52 as TD comments.

The pilot was used to better understand the procedure of the study. It helped us to evaluate the use of the data collection form, the necessary time to accomplish the task and, mainly, the number of comments used by each group. Thus, the pilot study was useful to calibrate the time and number of comments analyzed.

## 4 Operation

The experiment was conducted in a classroom at Federal University of Sergipe, and at the Salvador University, following the same procedure.

The operation of the experiment was divided into different sessions. A week prior to the experiment, the participants filled the consent and characterization form. The training

and experiment itself were performed at the same day. For training purposes, we performed a presentation in the first part of the class. The presentation covered the TD concepts and context, as well as the TD indicators [11] and how to perform a qualitative analysis on the code comments. This training took one hour.

After that, a break was taken. Next, each participant analyzed the set of candidate comments, extracted from ArgoUML, in the same room where we performed the training. They filled the data collection form pointing out the initial and end time of the task. For each candidate comment listed in the form, the participants chose "Yes" or "No", and their level of confidence on their answer. Besides, for each comment marked as yes, they should highlight the piece of text that was decisive for giving this answer.

The participants were asked to not discuss their answers with others. When they finished, they filled the feedback questionnaire. A total of three hours were planned for the experiment training and execution, but the participants did not use all of the available time.

### 4.1 Deviations from the Plan

We did not include the data points from participants who did not complete all the experimental sessions in our analysis since we needed all the information (characterization, data collection, and feedback). Thus, we eliminated 4 participants.

Table 3 presents the final distribution of the participants. The value in parentheses indicates the final number of participants in each group. In each of the groups G1 and G3, a participant was excluded because of not filling the value of confidence. In group G2, a participant was excluded because he did not analyze all comments and another was excluded because did not mark the text in the TD comments.

**Table 3.** Final distribution of the participants among groups.

| Group | Participants by experience level | | | Participants by English level | |
|---|---|---|---|---|---|
| | High | Medium | Low | Good | Med/Poor |
| G1 (11) | 4 | 3 | 4 | 1 | 10 |
| G2 (10) | 2 | 5 | 3 | 5 | 5 |
| G3 (11) | 3 | 5 | 3 | 5 | 6 |
| Total (32) | 9 | 13 | 10 | 11 | 21 |

## 5 Results

In this section, we present the results for each RQ.

***RQ1:*** *Do the English reading skills of the participant affect the accuracy when identifying TD through code comment analysis*?

In order to investigate the impact of the English reading level skills on the TD identification process, we calculated the accuracy values for each participant with

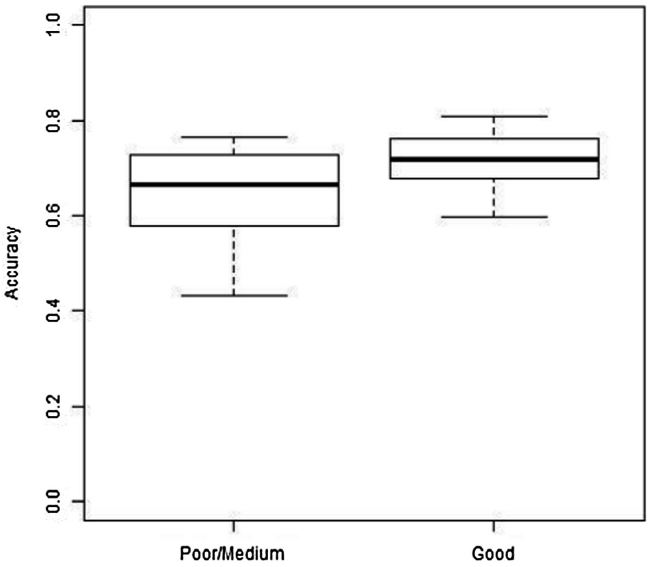


**Fig. 1.** Accuracy value by English reading skills.

respect to the Oracle. Figure 1 shows a box-plot illustrating the accuracy distribution. It is possible to note that the participants with good English reading skills had the lowest dispersion. It indicates that they are more consistent in the identification of TD comments than the participants with medium/poor English reading skills. Moreover, the accuracy values of the participants with good reading skills are higher than the values of the participants with medium/poor reading skills. However, the median accuracy of the participants with medium/poor reading skills is 0.65. This means that the participants with this profile were able to identify comments that were pointed out as an indicator of a TD item by the Oracle.

We also performed a hypothesis test to reinforce the analysis of this variable. To do this, we defined the following null hypothesis:

*H0*: *The English reading skills of the participant do not affect the accuracy with respect to the agreement with the Oracle.*

We ran a normality test, *Shapiro-Wilk*, and identified that the distribution was normal. After that, we ran the *t-test*, a parametric test, to evaluate our hypotheses. We used a typical confidence level of 95% ($\alpha = 0.05$). As shown in Table 4, the p-value calculated ($p = 0.02342$) is lower than the $\alpha$ value. Consequently, we may reject the null hypothesis (*H0*).

**Table 4.** Hypothesis test for analysis of English reading.

| | Shapiro-Wilk (normality test) | | Parametric test |
|---|---|---|---|
| | Good | Medium/Poor | t-test |
| p-value | 0.9505 | 0.9505 | 0.02342 |

We also evaluated our results regarding magnitude, testing the effect size measure. We calculate *Cohen's d* [24] to interpret the size of the difference between the distribution of the groups. We used the classification presented by Cohen [24]: 0 to 0.1: No Effect; 0.2 to 0.4: Small Effect; 0.5 to 0.7: Intermediate Effect; 0.8 and higher: Large Effect.

The magnitude of the result ($d = 0.814$) also confirmed that there is a difference (Large Effect) on the accuracy values with respect to both groups. This evidence reinforces our hypothesis and shows that the results were statistically significant.

In addition, we analyzed the feedback form and we highlighted the main notes at the following (translated to English): (i) *I had some difficulties to understand and decide about complex comments*; (ii) *I had the feeling that I needed to know the software context better*; (iii) *I believe some tips on English comments could help us to interpret the complex comments*. This data is aligned with our finding that indicates that English reading skills may affect the task of analyzing code comments to identify TD in software projects.

***RQ2:*** *Does the experience of participant impact the accuracy when identifying TD through code comment analysis?*

In order to investigate the impact of the experience level on the TD identification process, we calculated the accuracy values for each participant with respect to the Oracle. We show the accuracy distribution by experience level of the participants in Fig. 2. From this figure, it is possible to note that the box-plots have almost the same level of accuracy regarding high, medium and low experience. Considering the medians, the values are very similar. Participants with high and low experience have

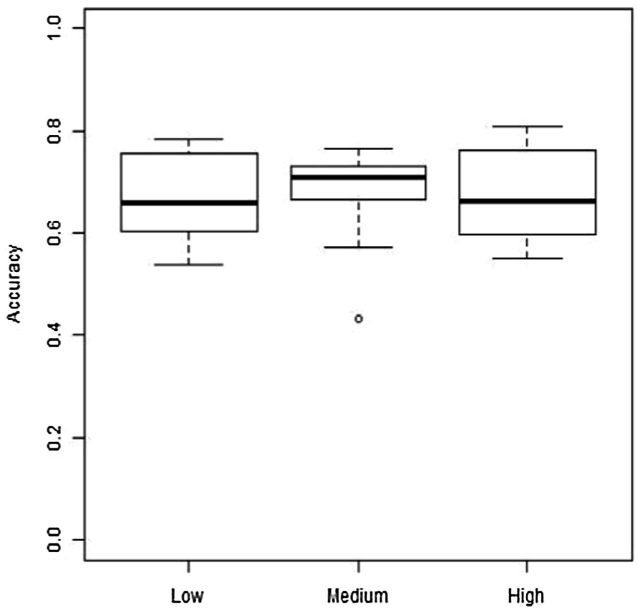


**Fig. 2.** Accuracy by participants' experience.

the same median value (0.66), whereas the median of participants with medium experience is moderately higher (0.71).

We also calculated the variation coefficient. This coefficient measures the variability in each level – that is, how many in a group is near the median. We found the coefficients of 12.91%, 13.17%, and 13.96%, for high, medium and low experience, respectively. According to the distribution presented by Snedecor and Cochran [25], the coefficients are low, showing that the levels of experience have homogeneous values of accuracy.

Finally, we performed a hypothesis test to analyze the experience variable. We defined the following null hypothesis:

*H0*: *The experiences of the participants do not affect his or her accuracy with respect to the agreement with the Oracle.*

After testing normality, we ran Anova, a parametric test to evaluate more than two treatments. The p-value calculated ($p = 0.904$) is bigger than $\alpha$ value. In this sense, we do not have evidences to reject the null hypothesis (*H0*).

From the analysis, we consider that the experience level did not impact the distribution of the accuracy values, i.e., when using CVM-TD, experienced and non-experienced participants show the same accuracy when identifying comments that point out to TD items. A possible interpretation of this result is that CVM-TD can be used by non-experienced participants.

***RQ3:*** *Do participants agree with each other on the choice of comments filtered by CVM-TD that may indicate a TD item?*

Our analysis considered the number of comments per rate of participants that chose the comment. Figure 3 shows the ratios in the *X-axis* and the number of comments in

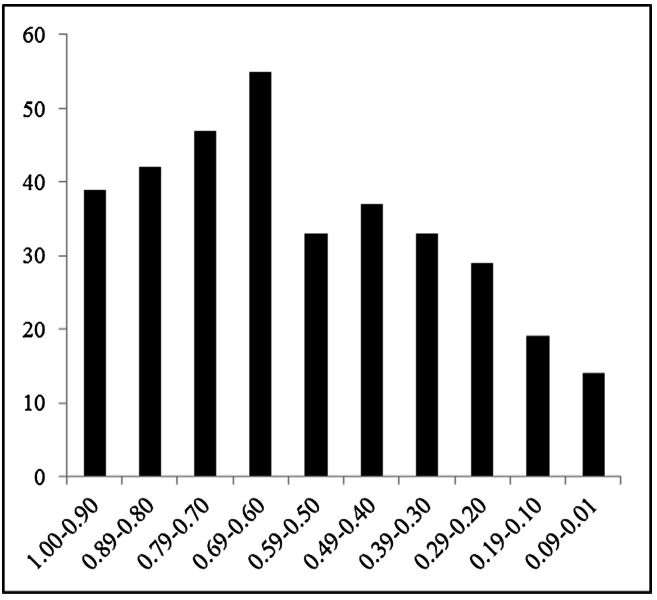


**Fig. 3.** Agreement among TD comments.

each interval in *Y-axis*. The ratio values are the proportion of "number of participants who choose the comment" and the "number of participants in the experiment group". For example, a comment from group G1 (the G1 has 11 participants) that was chosen by 10 participants has ratio = 0.91 (that is, 10/11).

We can note that all or almost all participants have chosen some comments as good indicator of TD, which means that these comments had high level of agreement and CVM-TD filtered comments that may really point out to TD item. Almost 40 comments have ratio intervals between 1 and 0.90. Some examples of such comments are:

```
"NOTE: This is temporary and will go away in a "future"
release (ratio = 1)";
"FIXME: this could be a problem...(ratio = 1) ";
"TODO:Replace the next deprecated call(ratio = 0.90) ".
"TODO: This functionality needs to be moved someplace
useful…(ratio = 0.90) ".
```

The whole set of these comments is available at https://goo.gl/fSaMj9.

On the other hand, considering the agreement among all participants identifying TD comments, we found a low coefficient. We conducted the Finn test to analyze the agreement in each group, considering all comments. Table 5 presents the agreement coefficient values. The level of agreement was 'slight' and 'fair' according to [23] classification.

**Table 5.** Finn agreement test.

| | Finn | p-value | Classification levels |
|---|---|---|---|
| Group 1 | 0.151 | 3.23e−05 | Slight |
| Group 2 | 0.188 | 5.74e−07 | Slight |
| Group 3 | 0.265 | 8.34e−12 | Fair |

***RQ4:*** *Does CVM-TD help researchers on select candidate comments that point to technical debt items?*

We analyzed the candidate comments identified by the Oracle as TD comments. Table 6 shows the number of comments identified by the Oracle. We observed that almost 60% of comments filtered by terms that belong to the vocabulary (candidate comments) proposed by Farias *et al.* [9] were identified as good indicators of TD by the Oracle.

**Table 6.** TD comments identified by the Oracle.

| | Group 1 | Group 2 | Group 3 |
|---|---|---|---|
| Number of candidate comments | 123 | 124 | 106 |
| Number of TD comments | 68 (55.28%) | 83 (66.94%) | 58 (54.72%) |

Regarding the average accuracy of all participants against the Oracle, the overall value is 0.673. On average, this shows that the participants archived good accuracy values. We also analyzed the dispersion of the set of values. The standard deviation (SD) is 0.087 which is considered a low value. In particular, the low SD indicates that the values do not spread too much around the average. That is, it indicates low dispersion of the accuracy values from the average. The box-blot in

Figure 4 represents the distribution of accuracy values for all participants. This figure shows that the values have a homogeneous distribution of accuracy.

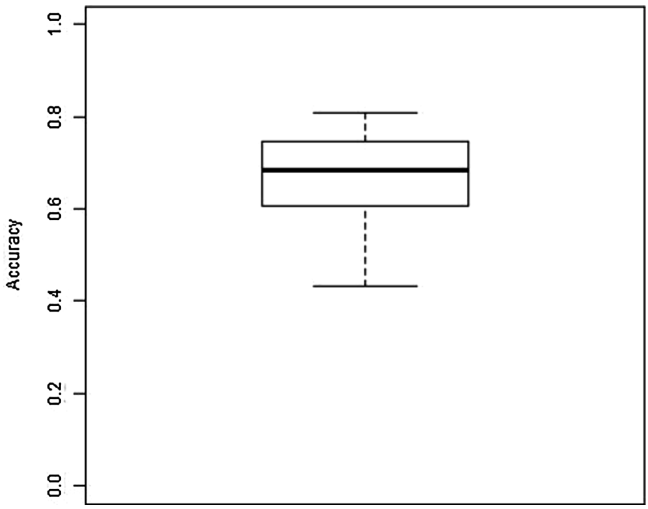


**Fig. 4.** The distribution of the accuracy values.

We can also see the accuracy distribution by each group of the participants in Fig. 5. From this figure, it is possible to note that the box-plots have almost the same level of accuracy. Considering the median, the values are very similar.

Next, we analyzed the number of TD comments chosen by participant and Oracle. Table 7 shows the TD comments identified by the Oracle and also by at least 50% of all participants in each group. This means that 66 comments were chosen by the Oracle and by at least 5.5 participants of the group 1. For group 2, 72 comments were chosen by Oracle and by at least 5 participants. While for group 3, 51 comments were chosen by Oracle and by at least 5.5 participants. The achieved results indicate that 89.21% of the TD comments that were chosen by Oracle were also chosen by at least 50% of all participants. The results indicate that the vocabulary helps the participants to identify comments that can point out to TD items.

We also analyzed the number of comments per rate of participants that chose the comments indicated by Oracle as a TD comment. Figure 6 shows the ratios in the *X-axis* and the number of comments in each interval in *Y-axis*. The ratio values are the proportion of "*number of participants who chooses a comment indicated as TD comment by Oracle*" and the "*number of participants in the experiment group*". For

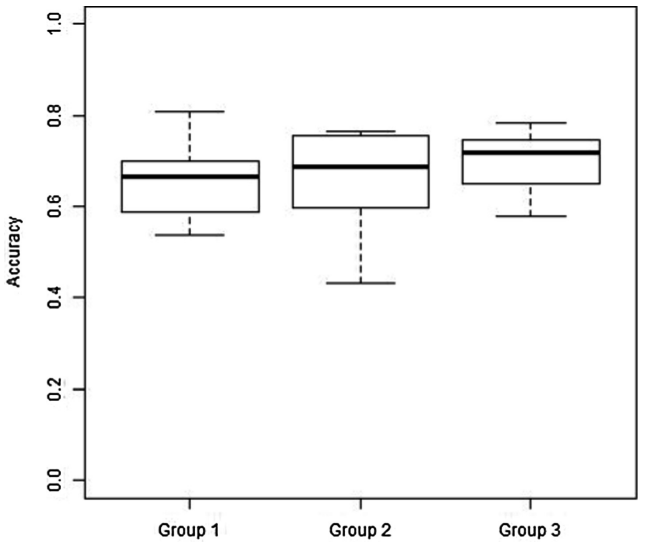


**Fig. 5.** Accuracy values by groups.

**Table 7.** TD comments identified by the Oracle and participants.

| | Group 1 | Group 2 | Group 3 |
|---|---|---|---|
| Number of candidate comments | 123 | 124 | 106 |
| Number of TD comments | 71 (57.72%) | 83 (66.94%) | 58 (54.72%) |
| Number of TD comments chosen by participants and Oracle | 66 (92.95%) | 72 (86.74%) | 51 (87.93%) |

example, a comment from group G1 indicated as TD comment by the Oracle (G1 has 11 participants) that was chosen by 11 participants has ratio = 1.00 (that is, 11/11).

When looking at the number of comments reporting TD, it is possible to note that, on average, the participants identified many comments that were filtered by the vocabulary and selected as TD indicators by Oracle. This means that the vocabulary proposed by CVM-TD helped the participants to comprehend and identify comments that may point out TD items. More than 170 comments have ratio intervals equal or higher than 0.5.

In order to perform a deep analysis, we investigated a sample of these comments. From the analysis, let's consider the following comments:

*Example comment #1 (ratio 1.00):*

```
/* Install the trap to "eat" SecurityExceptions. *
NOTE: This is temporary and will go away in a "future"
release */
```

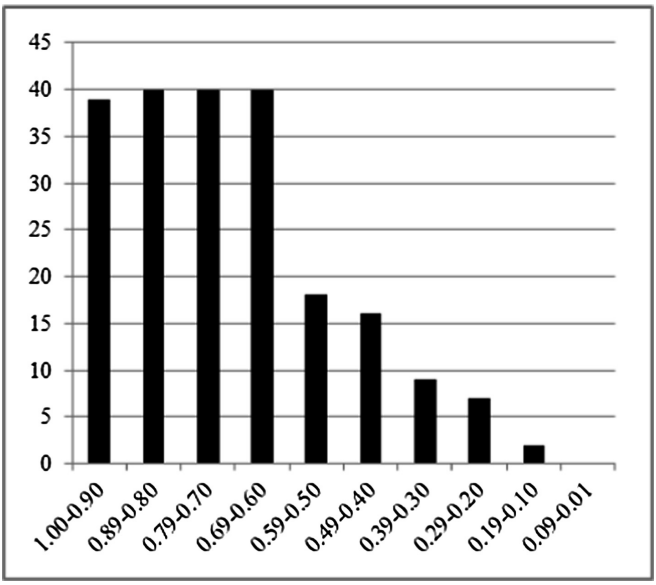


**Fig. 6.** Number of comments per rate of participants.

The above comment (#1) mentions issues in source code and highlights that a specific part of code is temporary and need to be removed in a future release. The comment has three patterns: "*the trap"*, "*NOTE: This is temporary",* and "*future release"*. In other words, it was indicated as a comment reporting a TD item by Oracle and all participants.

*Example comment #2 (ratio 0.92):*

```
/* TODO: Why is this here? Who is calling this? @see
java.beans.VetoableChangeListener#vetoableChange(java.b
eans.PropertyChangeEvent) */
```

*Example comment #3 (ratio 0.64):*

```
// This is somewhat inconsistent with the design of the
constructor // that receives the root object by
argument. If this is okay // then there may be no need
for a constructor with that argument.
```

In comments #2 and #3, the developers describe situations correlated with violation of principles of good design, inadequate location of a method, and inconsistence with the design. We can see this context through the patterns *"Why is this here?"*, "*Who is calling this*?", and *"inconsistent with the design"*. Besides these patterns, we can note the tag *TODO* reporting the need to revisit this code in future.

*Example comment #4 (ratio 0.91):*

```
/* TODO: needs documenting, why synchronized? */
```

*Example comment #5 (ratio 0.82):*

```
/* TODO: As currently coded, this actually returns all
BehavioralFeatures which are owned by Classifiers
contained in the given namespace, which is slightly
different than what's documented.  It will not include
any BehavioralFeatures which are part of the Namespace,
but which don't have an owner. */
```

Comments #4 and #5 are examples that we consider as documentation debt. In the above comment, the developers describe the necessity for documentation and show their worries about the missing or inadequate documentation. Maybe this TD item could not be identified using only code-based metrics because the necessity of documentation cannot be measured only analyzing the source code.

*Example comment #6 (ratio 0.82):*

```
// The following debug line is now the single most
memory consuming // line in the whole of ArgoUML. It
allocates approximately 18% of // all memory allocated.
// Suggestions for solutions: // Check if there is a
LOG.debug(String, String) method that can // be used
instead. // Use two calls. // For now I (Linus) just
comment it out.
```

Comment #6 reports strategies that caused a very large processing consuming in the build process of the projects. The patterns *"most memory consuming"* and *"memory allocated"* may refer to build related issues that make a task harder, and more time/processing consuming unnecessarily.

*Example comment #7 (ratio 0.91):*

```
/* TODO: Replace the next deprecated call. This case is
complicated * by the use of parameters. All other Figs
work differently. */
```

In comment #7, the developers describe situations correlated with bad coding practices, deprecated code, and difficult to be maintained in the future. The patterns *"TODO: replace"*, *"deprecated",* and *"case is complicated"* refer to the problems found in the source code which can negatively affect the quality of the code making it more difficult to be maintained in the future. They are examples of what we consider as code debt.

*Example comment #8 (ratio 0.91):*

```
/* TODO: The copy function is not yet completely
implemented - so we will have some exceptions here and
there.*/.
```

Comment #8 describes the lack of synchronism between optimal requirements specification and what is currently implemented. The pattern *"is not yet completely implemented"* indicates a requirement that are only partially implemented.

Some of these comments have patterns which report more clearly a situation related with the description of TD items, such as, *"TODO: this is temporary"*, *"inconsistent with the design"*, and *"TODO: needs documenting"*. We analyzed some of the top comments having ratio higher than 0.8. Table 8 lists the patterns identified by participants in these comments. Besides the fact that these comments have patterns that we considered to clearly identify TD, we can also see that many of them have more than one pattern reporting a TD context. This also helps indicating that the patterns can be used to filter comments describing a situation of TD.

**Table 8.** List of patterns identified by participants in these comments.

| ID comment | Ratio | Identified patterns |
|---|---|---|
| 17 | 1.00 | the trap |
| | | NOTE: This is temporary |
| | | "Future" release |
| 945 | 1.00 | This is a bug |
| 1005 | 1.00 | TODO: This is probably not the right location |
| 1064 | 1.00 | FIXME: this could be a problem |
| 26 | 0.91 | TODO: |
| | | Need to be in their own files? |
| | | TODO: Why is this here? |
| 44 | 0.91 | Who is calling this? |
| | | TODO: needs documenting |
| 68 | 0.91 | A temporary solution |
| 122 | 0.91 | Should be fixed |
| 318 | 0.91 | TODO: split into |
| 485 | 0.91 | Problems directly in this class |
| | | Critic problem |
| | | TODO: Replace |
| 521 | 0.91 | Deprecated |
| | | This case is complicated |
| | | Is there no better way? |
| 549 | 0.91 | TODO: implement this |
| 835 | 0.91 | TODO: ?? |
| 1227 | 0.91 | Code is not used |
| | | Why is it here? |
| | | Causes dependency |

Although the vocabulary allowed filtering good comments reporting TD, it also returned false positives. The Oracle and participants identified 141 comments, which were returned by the vocabulary, as comments that do not report a situation of TD, that is, they are false positives. This shows that 39.94% of the comments identified by our approach may have been misclassified by patterns of the vocabulary (353 comments were returned in total).

Some examples of these false positives follow:

*Example comment #9:*

```
// NOTE: This is package scope to force callers to use
ResourceLoaderWrapper
```

*Example comment #10:*

```
// Note that this will not preserve empty lines // in
the body
```

The *eXcomment* selected these candidate comments (#9 and #10) because of presence of the *"NOTE:"* but the comment only gives an overview of the information about the code implementation.

*Example comment #11:*

```
/* Now we have to see if any state in any statemachine
of * classifier is named [name]. If so, then we only
have to link the state to c. */
```

The pattern "*have to*" was interpreted as an open task verb reporting a task to be done but it only composes an expression describing the functionality of the code.

*Example comment #12:*

```
/* This is used in the todo panel, when "By Poster" is
chosen for a     * manually created todo item.*/
```

The pattern "*TODO*" made our tool and vocabulary selecting the comment as a candidate comment, while the term was used only to describe the name of a component.

*Example comment #13:*

```
/* This next line fixes issue 4276: */
```

The pattern "*fixes*" may suggest something to be fixed in the project but, in this case, the comment describes the opposite, a piece of the code fixing an issue.

This result indicates that the vocabulary and *eXcomment* need to be better calibrated to reduce the number of returned comments that do not report a situation of TD. With

this in mind, we intend to perform a complementary investigation with the purpose of identifying the patterns that are more decisive to detect TD items.

***RQ5:*** *What were the most chosen code comment patterns by participants as decisive to indicate a TD item?*

Table 9 shows the top 20 selected patterns. Column "Patterns" lists the patterns identified by participants, *"# of occurrence"* summarizes the number of times that a participant highlighted the patterns as important, and *"# of Participants"* summarizes the total of participants who have chosen the pattern as important to identify a TD item.

**Table 9.** Top 20 patterns more chosen by participants as decisive to indicate a TD item.

| Patterns | # of occurrences | # of Participants |
|---|---|---|
| TODO | 899 | 26 |
| Need to | 48 | 22 |
| Remove | 54 | 19 |
| ? | 49 | 19 |
| Replace | 19 | 14 |
| Is this needed? | 15 | 14 |
| Must be | 22 | 13 |
| Should be | 20 | 12 |
| Does not work! | 18 | 12 |
| Critic | 21 | 11 |
| Check | 14 | 11 |
| Fix | 22 | 11 |
| Why? | 21 | 10 |
| Cyclic dependency | 10 | 10 |
| This is a bug | 10 | 10 |
| Note | 13 | 9 |
| Move | 13 | 9 |
| Have to | 12 | 9 |
| Temporary | 9 | 9 |
| Probably redundant | 9 | 9 |

The most selected pattern was *'Todo"*. It was chosen by 26 of 32 participants (81.25%), followed by "*need to"* selected by 22 participants (65.63%), "*remove"*, and "*Todo: Replace"* by 16 participants (50%). From the table, we can also see that several comment patterns have the tag "*Todo",* such as *"Todo: This does not work!"* and *"Todo: Is this needed/correct?"*. The pattern *"Todo"* was identified as a decisive pattern in 899 cases by the participants. This highlights the importance of "*Todo*" for composing important patterns to identify TD through code comment analysis. Besides the pattern "*Todo*", the patterns *"need to", "remove", "?", "replace", "Is this needed?", "must be", "should be", and "does not work"* were also chosen as decisive patterns by the participants. The complete list is available at https://goo.gl/ZDxTmn.

Even though these results show that there are decisive patterns to identify a situation of TD, further investigations are needed for analyzing how much each pattern is decisive to identify TD. In this way, we intend to perform another experiment to widely analyze and classify all patterns by considering the importance level of the patterns to point out to a TD item.

## 6 Discussion

Our results suggest that the English reading level of the participants may impact the identification of TD through comment analysis. Participants with good English reading skills had accuracy values better than participants who have medium/poor English reading skills. On the other hand, participants with poor/medium English profile were able to identify a good amount of TD comments filtered by the contextualized vocabulary.

We also observed in the feedback analysis that some participants had difficulties to understand and interpret complex comments, and tips might help them with this task. We conjecture that some tips may support participants to make decision on the TD identification process. For instance, highlight the TD terms or patterns of comment from the contextualized vocabulary into the comments.

Considering the impact of experience on TD identification, we could not conclude that the experience level affects the accuracy. This can indicate that comments selected by the vocabulary may be understood by an experienced or non-experienced observer. This reinforces the idea that the TD metaphor aids discussion by providing a familiar framework and vocabulary that may be easily understood [26, 27].

Considering the agreement among participants identifying TD comments, the results revealed some comments pointed out as good indicator of TD, with high level of agreement. It may evidence the contribution of CVM-TD as a support tool on the TD identification. However, in general, the level of agreement between participants was considered low. We believe that this occurred due to the large amount of comments to be analyzed, and the amount of comments selected by the contextualized vocabulary that does not indicate a TD item. In this way, the level of agreement might rise whether the vocabulary is more accurate.

We noted that a high number of comments filtered by the CVM-TD was considered to indicate TD items. In view of the contribution of the CVM-TD to support TD identification, the results indicate that many comments that were chosen by Oracle were also chosen by participants as comments that may point out to a TD item. This means that the vocabulary proposed by CVM-TD helped the participants to comprehend and identify comments that may reporting a situation of TD.

These results provide preliminary indications that CVM-TD and the contextualized vocabulary can be considered an important support tool to identify TD item through code comments analysis. Different from code metrics-based tools, code comments analysis allow us to consider human factors to explore developers' point of view and complement the TD identification with more contextual and qualitative data. Both approaches may contribute with each other to make the automated tools more efficient.

The last aspect we analyzed was the patterns mostly chosen by participants as decisive to indicate a TD item. We identified a list of the top 20 patterns. We could note that there are patterns that are more decisive than others, but we need to perform further investigations on the patterns considering their importance level to identify a TD item.

### 6.1 Threats to Validity

We followed the checklist provided by Wohlin et al. [28] to discuss the relevant threats to this controlled experiment.

**Construct Validity.** To minimize the mono-method bias, we used an accuracy and agreement test to provide an indication of the TD identification through comment analysis. We selected the researchers that composed the Oracle. In order to mitigate the biased judgment on the Oracle, its definition was performed by three different researchers with knowledge in TD. Two of them selected the TD comments, and the third researcher did a consensus to decrease the bias. Finally, to reduce social threats due to evaluation apprehension, participants were not evaluated.

Another thread involves the definition of the proposed vocabulary. It is possible that the set of patterns and combinations used by our model and vocabulary are simply too many to be studied. An alternative would be to limit the studies to specific contexts and software domains. Another point is related to the imprecision in the filtering of candidate comments. Our results might be affected by false positives and false negatives returned by the vocabulary and the automatic filtering heuristic. To reduce this threat in next experiment, we intent to calibrate the vocabulary and *eXcomment* to reduce the number of comments that do not report a situation of TD.

**Internal Validity.** The first internal threat we have to consider is subject selection since we have chosen all participants through a convenience sample. We minimized this threat organizing the participants in different treatment groups divided by experience level.

Another threat is that participants might be affected negatively by boredom and tiredness. In order to mitigate this threat, we performed a pilot study to calibrate the time and amount of comments to be analyzed. To avoid the communication among participants, two researchers observed the operation of the experiment at all times. A further validity threat is the instrumentation, which is the effect caused by artifacts used for the experiment. Each group had a specific set of comments, but all participants used the same data collection form format. To investigate the impact of this threat in our results, we analyzed the average accuracy in each group. Group G1 has average value equal to 0.65. For group G2, the average value is equal to 0.66, and group G3 is equal to 0.69. From these data, it is possible to note that groups have almost the same level of average accuracy. It means that this threat did not affect the results.

**External Validity.** This threat relates to the generalization of the findings and their applicability to industrial practices. This threat is always present in experiments with students as participants. Our selected samples contained participants with different levels of experience. All participants have some professional experience in the software

development process. It is an important aspect in mitigating the threat. A further threat is the usage of software that may not be representative for industrial practice. We used software adopted in the practice of software development as an experimental object to mitigate the threat.

**Conclusion Validity.** To avoid the violation of assumptions, we used normality test, Shapiro-Wilk, and a parametric test, the t-test, for data analysis. To reduce the impact of reliability of treatment implementation, we followed the same experimental setup on both cases.

## 7 Conclusion and Future Work

In this paper, we performed a controlled experiment to evaluate the CVM-TD aiming to characterizing its overall accuracy and factors that may affect the identification of TD through code comment analysis. Our results indicate that: (i) English reading skills affect the participants' accuracy; (ii) we could not conclude that the experience level impacts on understanding of comments to support the TD identification; (iii) concerning the agreement among participants, although we found low agreement coefficients between participants, some comments have been indicated with a high level of agreement; (iv) CVM-TD provided promising results concerning to the identification of comments as good indicator of TD by participants. The results indicate that 89.21% of the TD comments that were chosen by the Oracle were also chosen by at least 50% of all participants. The results indicate that the vocabulary helps the participants to identify comments that can point out to TD items.

Although the vocabulary has been used to filter good comments reporting TD, it also returned false positives. We identified that 39.94% of the comments automatically identified by our approach may have been misclassified by patterns of the vocabulary. This result indicates that the vocabulary and *eXcomment* need to be better calibrated to reduce the number of returned comments that do not report a situation of TD.

The results motivate us to continue exploring code comments in the context of the TD identification process to improve CVM-TD and the *eXcomment*. We believe that the vocabulary and the study findings on how code comment analysis supports the TD identification represent a significant contribution to the research community.

Future works include to: (i) perform further investigations for analyzing how much each pattern is decisive to identify TD, (ii) develop some features in *eXcomment* associated with the CVM-TD to support the interpretation of comments, such as "usage of weights and color scale to indicate the comments with more importance in TD context, and highlight the TD terms or patterns of comment into the comments", and (iii) evaluate the use of CVM-TD in projects in the industry.

**Acknowledgements.** This work was partially supported by CNPq Universal 2014 grant 458261/2014-9. The authors also would like to thank Methanias Colaço and André Batista for their support in the execution step of the experiment.

## References


1. Izurieta, C., Vetrò, A., Zazworka, N., Cai, Y., Seaman, C., Shull, F.: Organizing the technical debt landscape. In: 3rd International Workshop on Managing Technical Debt, MTD 2012 – Proceedings, pp. 23–26 (2012)
2. Ernst, N.A., Bellomo, S., Ozkaya, I., Nord, R.L., Gorton, I.: Measure it? Manage it? Ignore it? Software Practitioners and Technical Debt. In: 10th Joint Meeting on Foundations of Software Engineering - ESEC/FSE 2015, pp. 50–60 (2015)
3. Alves, N.S.R., Mendes, T.S., Mendonça, M.G., Spínola, R.O., Shull, F., Seaman, C.: Identification and management of technical debt: a systematic mapping study. Inf. Softw. Technol. **70**, 100–121 (2016)
4. Guo, Y., Spínola, R.O., Seaman, C.: Exploring the costs of technical debt management – a case study. Empir. Softw. Eng. **1**, 1–24 (2014)
5. Li, Z., Liang, P., Avgeriou, P., Guelfi, N.: A systematic mapping study on technical debt and its management. J. Syst. Softw. **101**, 193–220 (2014)
6. Mendes, T.S., Almeida, D.A., Alves, N.S.R., Spínola, R.O., Mendonça, M.: VisMinerTD - an open source tool to support the monitoring of the technical debt evolution using software visualization. In: 17th International Conference on Enterprise Information Systems (2015)
7. Zazworka, N., Spínola, R.O., Vetro', A., Shull, F., Seaman, C.: A case study on effectively identifying technical debt. In: Proceedings of the 17th International Conference on Evaluation and Assessment in Software Engineering - EASE 2013, pp. 42–47. ACM Press, New York (2013)
8. Potdar, A., Shihab, E.: An exploratory study on self-admitted technical debt. In: IEEE International Conference on Software Maintenance and Evolution, pp. 91–100 (2014)
9. Farias, M.A.F., Silva, A.B., Mendonça, M.G., Spínola, R.O.: A contextualized vocabulary model for identifying technical debt on code comments. In: 7th International Workshop on Managing Technical Debt, pp. 25–32 (2015)
10. Maldonado, E.S., Shihab, E.: Detecting and quantifying different types of self-admitted technical debt. In: 7th International Workshop on Managing Technical Debt, pp. 9–15 (2015)
11. Alves, N.S.R., Ribeiro, L.F., Caires, V., Mendes, T.S., Spínola, R.O.: Towards an ontology of terms on technical debt. In: Sixth International Workshop on Managing Technical Debt (MTD), pp. 1–7 (2014)
12. Storey, M., Ryall, J., Bull, R.I., Myers, D., Singer, J.: TODO or to bug : exploring how task annotations play a role in the work practices of software developers. In: ICSE: International Conference on Software Engineering, pp. 251–260 (2008)
13. Maalej, W., Happel, H.-J.: Can development work describe itself? In: 7th IEEE Working Conference on Mining Software Repositories (MSR), pp. 191–200 (2010)
14. Steidl, D., Hummel, B., Juergens, E.: Quality analysis of source code comments. In: 21st International Conference on Program Comprehension (ICPC), pp. 83–92. IEEE (2013)
15. Etzkorn, L.H., Davis, C.G., Bowen, L.L.: The language of comments in computer software: a sublanguage of English. J. Pragmat. **33**, 1731–1756 (2001)
16. Bavota, G., Russo, B.: A large-scale empirical study on self-admitted technical debt. In: 13th Working Conference on Mining Software Repositories – MSR, pp. 315–326 (2016)
17. Lemos, O.A. de Paula, A.C., Zanichelli, F.C., Lopes, C.V.: Thesaurus-based automatic query expansion for interface-driven code search categories and subject descriptors. In: 11th Working Conference on Mining Software Repositories – MSR, pp. 212–221 (2014)

18. Host, M., Wohlin, C., Thelin, T.: Experimental context classification: incentives and experience of subjects. In: Proceedings of 27th International Conference on Software Engineering, ICSE 2005, pp. 470–478 (2005)
19. Salman, I., Misirli, A.T., Juristo, N.: Are students representatives of professionals in software engineering experiments? In: Proceedings of the 37th International Conference on Software Engineering (2015)
20. Santos, J.A.M., Mendonça, M.G., Pereira, C.: The problem of conceptualization in god class detection: agreement, strategies and decision drivers. J. Softw. Eng. Res. Dev. **2**, 1–33 (2014)
21. Shull, F., Singer, J., Sjoberg, D.: Guide to Advanced Empirical Software Engineering. Springer, London (2008). doi:10.1007/978-1-84800-044-5
22. Finn, R.H.: A note on estimating the reliability of categorical data. Educ. Psychol. Measur. **30**(1), 71–76 (1970). doi:10.1177/001316447003000106. ISBN 0013-1644
23. Landis, J.R., Koch, G.G.: The measurement of observer agreement for categorical data. Biometrics **33**, 159–174 (1977)
24. Cohen, J.: Statistical Power Analysis for the Behavioral Sciences, 2nd edn. Lawrence Earlbaum Associates, Hillsdale (1988). http://www.worldcat.org/isbn/-0805802835
25. Snedecor, G.W., Cochran, W.G.: Statistical Methods, 6th edn. Iowa State University Press, Ames (1967)
26. Spínola, R., Zazworka, N., Seaman, C., Shull, F.: Investigating technical debt folklore. In: 5th International Workshop on Managing Technical Debt, pp. 1–7 (2013)
27. Kruchten, P., Nord, R.L., Ozkaya, I.: Technical debt: from metaphor to theory and practice. IEEE Softw. **29**(6), 18–21 (2012)
28. Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B., Wesslén, A.: Experimentation in Software Engineering: An Introduction. Kluwer Academic Publishers, Norwell (2000)