On The Gap Between Software Maintenance Theory and Practitioners' Approaches

Mívian Ferreira
DCC - UFMG
Belo Horizonte, Brazil
mivian.ferreira@dcc.ufmg.br

Mariza Bigonha
DCC - UFMG
Belo Horizonte, Brazil
mariza@dcc.ufmg.br

Kecia A. M. Ferreira
DECOM - CEFETMG
Belo Horizonte, Brazil
kecia@cefetmg.br

Abstract—The way practitioners perform maintenance tasks in practice is little known by researchers. In turn, practitioners are not always up to date with the proposals provided by the research community. This work investigates the gap between software maintenance techniques proposed by the research community and the software maintenance practice. We carried out a survey with 112 practitioners from 92 companies and 12 countries. We concentrate on analyzing if and how practitioners understand and apply the following subjects: bad smells, refactoring, software metrics, and change impact analysis. This study shows that there is a large gap between research approaches and industry practice in those subjects, especially in change impact analysis and software metrics.

Index Terms—software maintenance, survey, software engineering, software metrics, refactoring, bad smells, change impact analysis

I. Introduction

There is a large gap between industry practice and academic proposals, and this gap tends to grow over time [1]. Some initiatives have been taken for decades to bridge this gap, aiming to comprehend how relevant software engineering research is to practitioners [2] [3] [4]. There are difficulties for both sides. Research publications might not be accessible to the industry, and their results might not be of practical use [5]. On the other hand, software engineering researchers may face challenges when collaborating with practitioners, such as differences between culture and time perspective [6].

We focused this work on the gap between research and practice in software maintenance, one of the most critical and most expensive software live cycle activities. The spectrum of software maintenance subjects is vast and comprises activities such as: log of modification requests, change impact analysis, modification of code and other artifacts, re-engineering, reverse engineering, program comprehension, measurement, migration, tests, training, and daily support [7]. This work investigates if and how practitioners apply: change impact analysis, software metrics, bad smell, refactoring. We consider those subjects because our research is mostly concentrated on them.

We surveyed 112 practitioners from 12 countries and 92 companies. In particular, we investigate if practitioners are familiar with the subjects considered in this work, if they usually apply such concepts and techniques, the tools regarding those topics practitioners use in practice, and the main challenges developers face when carrying out software maintenance.

II. Related Work

The problem of the distance between what the software engineering research community produces and what practitioners apply in their practices has gained increasing attention [2] [3]. Murphy-Hill et al. [8] investigated the adoption of refactoring tools by developers. They found that 90% of refactoring is performed manually. Their study relies on data of refactoring Java programs in the Eclipse environment. Although popular, Eclipse is only one out of dozens of development environments used nowadays.[1] We analyze refactoring practice too; however, we did not base our study in any specific language or development environment. Moreover, we considered other three subjects besides refactoring, and we did not gather our data automatically since we performed our analysis by questioning practitioners about their practices.

Our results contradict the assumption that practitioners have widely applied software metrics. Kupiainen et al.'s [9] systematic literature review on industrial studies analyzed the adoption of software metrics in Agile and Lean software development. They found 102 metrics in the primary studies. However, only one metric is for source code, namely violations of static code, defined as the number of violations found in the static code regarding rules from tools like Findbugs, PMD, and Checkstyle. Dozens of source code software metrics have been proposed, some of them widely known by researchers, such as the CK Metrics. However, none of such metrics appeared in the Kupiainen et al.'s study neither were mentioned by the participants of our research.

Palomba et al. [10] investigated the difference between developers' perception and academic concepts of bad smells. Amjed et al. [11] investigated bad smells' popularity in Stack Overflow. Both works reached similar conclusions, namely: the academic community might be perceiving bad smell as a problem that is not considered a real problem by developers. The aim and the approach we used are different from those studies since our work investigated if developers know and apply the concept of bad smell. Our results do not contradict

---

[1]According to the PYPL index, available at https://pypl.github.io/PYPL.html.

the previous ones and empirically show a fundamental reason for the differences in academia and industry perceptions: the bad smell is not a widely known concept among developers.

According to Brodin and Benitti [4], over 70% of software developers work with maintenance. However, they point out that it is still a topic little researched. They studied whether practitioners in the industry use the topics about software maintenance taught in undergraduate courses through a survey. They identified that engineering, reverse engineering, software processes, and software measurement are much considered by academia and rarely used in the industry. In the results reported, Brodin and Benitti state that refactoring is the only topic widely studied by academia that practitioners extensively use. Our results are more precise regarding refactoring since we found that practitioners indeed apply a few refactoring techniques. Moreover, our results indicate that change impact analysis is applied by using mainly naive techniques, such as 'search and replace'.

III. STUDY DESIGN

This section describes the study's design presented in this paper, detailing the following aspects: the research questions, the construction and validation of the questionnaire, the participants' selection, and the method we applied to analyze the data.

A. Research Questions

The research questions aim to elucidate if and how some of the main concepts and techniques proposed by the research community for software maintenance are applied. This section presents the research questions and how we analyzed the data to answer them.

RQ1. Are developers familiar with the concepts of software metrics, bad smells, refactoring, and change impact analysis?

To answer this question, we calculated the percentage of the participants who answered 'yes' to the questions related to familiarity (Rows 2, 7, 12, and 16 of Table I). We used a yes/no question to split the respondents into two distinct groups [12], once we want to know if (not how) the respondent is familiar with the concepts.

RQ2. Do practitioners apply software metrics, refactoring, bad smells, and change impact analysis in practice?

To answer this research question, we calculated the percentage of the participants who answered 'yes' to the questions described in Rows 4, 8, 13, and 17 of Table I. For the same reasons mentioned in RQ1, we also chose to use a yes/no question in this case.

RQ3. Which are the tools most used by practitioners in software maintenance?

In this research question, we aim to identify which tools are used to support the studied topic's activities. The participants answered the questions associated with RQ3 in a text field. To analyze the data, we read all the answers and tabulated the tools described by the participants.

RQ4. How do practitioners perform change impact analysis?

With this research question, we investigate if and how practitioners analyze the impact of changes they need to perform in software systems. To answer this question, we summarized and reported the answers to the question described in Row 18 of Table I.

RQ5. Which metrics, refactoring techniques, and bad smells practitioners apply in their activities?

To answer this research question, we read all the answers given to the questions described in Rows 5, 9, and 14 of Table I and summarized the data.

RQ6. What are the biggest challenges faced by practitioners when carrying out software maintenance?

To investigate this research question, two authors of this work tabulated and labelled, separately, the answers to the question shown in the first row of Table I. In the presence of different labels for the same answers, these two authors opted for the final label by a consensus between them.

B. Questionnaire Construction

We based the questionnaire on the guideline of Kitchenham and Pfleeger [13]. It is composed of seven sections described as follows. Table I shows the survey questions and answers options.

Term of consent introduced the study's purpose to the participants and requested their endorsement to use the data collected and participants' anonymity assurance.

Participants' characterization. Collect data about the participants' professional lives: (i) name and country of the company where they work; (ii) the position held in the company; (iii) academic background; (iv) years of professional experience; (v) programming languages currently used in their jobs; and (vi) the methodology used in their company's software development process.

Challenges to perform software maintenance. This section contains only one question in which we requested the participants to describe the main challenges and difficulties they face to perform maintenance activities.

Metrics. In this section, we asked the participants if they are familiar with metrics; and consider them useful. We also asked if they use metrics to measure the code quality, the tools they use to do it, and the most common metrics they apply.

Refactoring. About refactoring, we asked if the participants are familiar with the term "refactoring" and if they usually perform code refactoring. We also asked them to name the types of refactoring techniques they apply and their tools to perform such activity.

Bad smells. We asked if the participants are familiar with the term "bad smell", and if they use it to verify the presence of bad smells in the software code. If so, we requested them to list which bad smells they use to search in software systems.

Change impact analysis. We asked the participants: if they notice the need of changing other pieces of code when they carry out a specific change in the code; if they are familiar with the term "change impact analysis"; if they use to search other pieces of code that need to be modified when they make

TABLE I: Questions and response options of the questionnaire.

Subject		Question	Reply Options
Challenges to Perform Maintenance	1	Describe the main difficulties you face when performing maintenance on software.	Open Field
Software Metrics	2	Are you familiar with software metrics concept?	Yes or No
	3	What is your opinion about the use of software metrics to ensure the quality of the source code?	'Very important', 'Important', 'Little important', 'Unnecessary' or 'I don't have background to give an opinion.'
	4	Do you use software metrics to evaluate the quality of the source code at your work?	Yes or No
	5	If you use software metrics to evaluate the quality of the source code at your work, please name them.	Open Field
	6	If you use metrics to evaluate the quality of the source code at your work, which measurement tool(s) do you use?	Open Field
Refactoring	7	Are you familiar with the concept of refactoring ?	Yes or No
	8	Have you ever applied code refactoring at your work?	Yes or No
	9	If you have ever used code refactoring at your work, what kind (s) of refactoring did you use?	Open Field
	10	If you have ever used code refactoring at your work, have you used a tool for this?	Yes or No
	11	If you have ever used code refactoring at your work and have used a tool to do so, which tool (s) did you use?	Open Field
Bad Smell	12	Are you familiar with the concept of bad smell?	Yes or No
	13	When developing or maintaining a system at work, do you usually check bad smells in the source code?	Yes or No
	14	If you answered 'yes' to the previous question, what are the bad smells most commonly detected by you?	Open Field
Change Impact	15	Have you ever noticed whether a change performed in a software system by you had caused the need to make other changes not initially foreseen?	'Never', 'Few times', 'Oftentimes' or 'Always'
	16	Are you familiar with the term "Change Impact Analysis"?	Yes or No
	17	When correcting a bug (error or failure), performing a change or creating a new functionality in the system, do you usually analyze the impact of the change in the rest of software system?	Yes or No
	18	What kind of technique do you apply to analyze parts of the software that need to be modified?	'I explore the code manually and intuitively, not always with prior knowledge about it.', 'I explore the code manually guided by the prior knowledge I have about it.', 'I use a tool for this.', or 'I do not analyze all the parts that need to be modified, I make the modifications as I identify the problems.'
	19	If you use a tool to analyze which parts of the software need to be modified, please name them.	Open Field

a change in the code; what kinds of techniques they use to perform such analysis; and, finally, if they use any tool to perform change impact analysis.

1) Validation of the Questionnaire: Before sending the questionnaire to the participants, we made a pilot survey to test the questionnaire and identify improvement needs. For this purpose, we sent the questionnaire to two developers from different companies. We identified the following primary need for improvements from the first version. (1) Observing the answers, we notice that the questions we asked before the section "Challenges to Perform Software Maintenance" may had influenced the participants' answers about the main challenges they face in software maintenance. Probably, this happens because we asked about refactoring, dependency analysis, bad smells, and metrics before that section, which may have induced the participants to include problems related to such subjects in their answers. We then decided to put the Section 'Challenges to Perform Software Maintenance' as the first one in the questionnaire. (2) We reformulated the answer options that involve ranges: number of employees and years of professional experience. (3) We divided some long questions into two or more to be more precise and improve the data analysis.

We sent the second version of the questionnaire with such improvements to eight developers in a second round. We made a previous analysis of the responses to ensure that the questionnaire was more precise and able to gather the information we need. Then, we sent the survey to the other participants.

C. Participants Selection

We invited practitioners to answer the survey in two ways: directly and indirectly. Some participants were contacted directly via email, LinkedIn, and Facebook and were chosen by convenience since the authors had their contact. To motivate the practitioners to answer the questionnaire, we chose to send a particular message to each participant explaining the survey's aim and inviting him/her to participate in the research. In this message, we asked them to forward the email to other colleagues, aiming to invite more participants. We also published the questionnaire in our social networks and specialized IT mailing lists.

We received 112 responses. We directly contacted 204 practitioners; out of this amount, 77 answered the questionnaire, achieving a response rate of 37.8%. We reached the remaining 35 practitioners that answered the questionnaire indirectly. As we aimed to have a global assessment of practitioners' perceptions and practices worldwide, we tried to contact professionals from different countries and many companies. The practitioners that answered the questionnaire are from 92 companies and 12 countries.

IV. PARTICIPANTS CHARACTERIZATION

This section describes the characteristics of the participants.
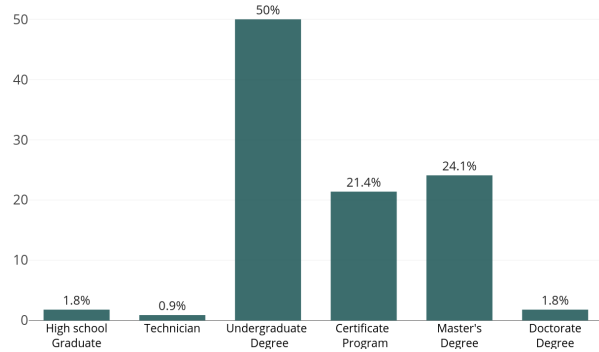
Fig. 1: Distribution of participants' academic background.

Academic Background. The participants with an undergraduate degree, certificate program, or a master's degree correspond to 95.5% of the sample, as shown in Figure 1. Only 4.5% of the participants are high school graduates, technicians, or have a Ph.D. degree, which indicates that the practitioners have a high level of formal education, i.e., they had access to Computer Science's theoretical aspects.
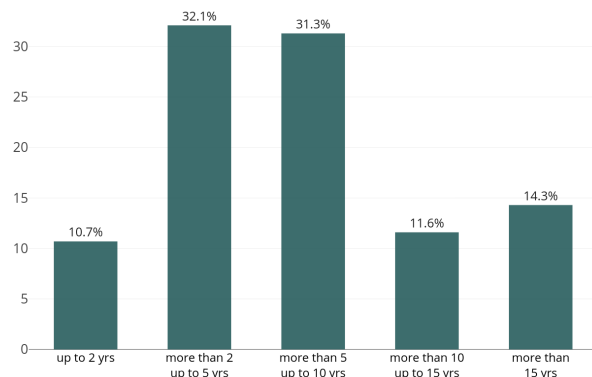

Fig. 2: Distribution of participants' professional experience.

Professional Experience. Most participants, 63.4%, have between two and ten years of professional experience, and 25.9% of the participants are experienced professionals; they have more than ten years of career. Junior practitioners are 10.7% of the participants. Figure 2 shows the distribution of these data. Therefore, most participants (89%) have much practical experience and are likely to know well real software engineering scenarios.

Programming Languages. The programming languages used by the participants are Java, C#, C++, Python, JavaScript, PHP, Scala, Kotlin, TypeScript, ShellScript, Delphi, Swift, Objective C, Golang (Go), Groovy, Pearl, Dart, Ruby, Visual FoxPro, VB.NET, and ASP.NET.

Methodologies. Agile methodologies are the most widely used by developers - being Scrum and XP the most cited of them. Only one participant informed the company uses a Waterfall process.

Companies' Sectors. 71% of the companies are from the IT area. The other companies are in the following areas: trade,

financial, bank, industry, marketing, health, education, and government.

Companies' Characterization. Among the participants, 69.7% work in medium-sized or large companies, and 30.3% work in micro and small companies. Figure 3 presents the distribution of the participants' companies size by the number of employees.
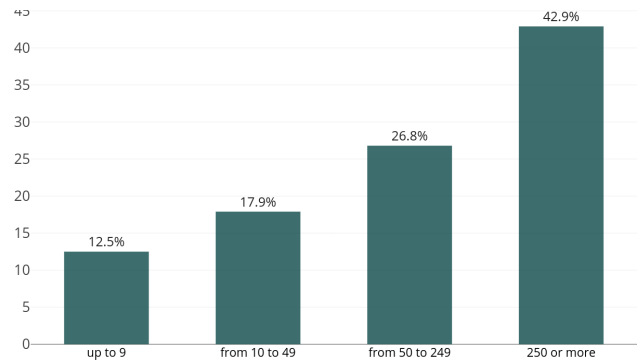

Fig. 3: Distribution of participants' companies size by the number of employees.

V. RESULTS

This section presents the data analysis and the answers to the research questions.

RQ1. Are developers familiar with the concepts of software metrics, bad smells, refactoring, and change impact analysis?

To answer this question, we considered the number of participants who declared to be familiar with the subjects investigated in this paper. Figure 4 shows the percentage of participants who answered 'yes' to the questions regarding the familiarity with software metrics, refactoring, bad smells, and change impact analysis, respectively (see Rows 2, 7, 12, and 16 of Table I).
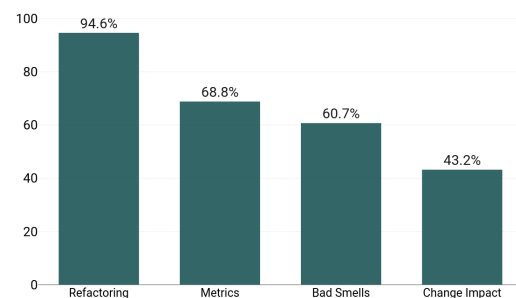

Fig. 4: Percentage of participants who declare to be familiar to the subjects.

Refactoring is the most popular subject among the participants; 94.6% of them claim to be familiar with refactoring. Metrics is the second most popular subject; 68.8% of the participants are familiar with it. Bad Smell is a term known by

60.7% of the participants. Change Impact Analysis is the least familiar term to the practitioners who answered the survey, 43.2%, stated they are familiar with this concept.

RQ2. Do practitioners apply software metrics, refactoring, bad smells, and change impact analysis in practice?

To answer RQ2, we calculated the number of participants that answered 'yes' to the questions regarding the practical application of metrics, refactoring, bad smells, and change impact analysis (Rows 4, 8, 13, and 17 of Table I). Figure 5 shows the results.
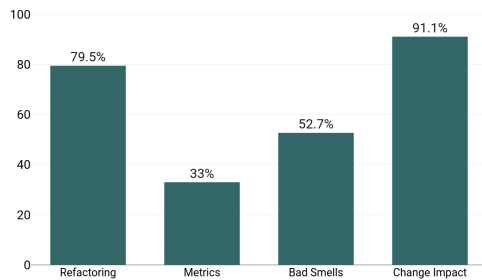
Fig. 5: Percentage of participants who declared to apply refactoring, software metrics, bad smell, and change impact analysis.

Change impact analysis is the technique most applied by practitioners, 91.1% of the participants affirmed that they perform some extent of change impact analysis when making software code modifications. It is important to note that only 43,2% informed they are familiar with the concept of "change impact analysis". We intentionally used the term "change impact analysis", which is commonly used in academia, when asking if the participant is familiar or not with the concept. However, we did not use this term in the other questions about performing change impact analysis. We constructed the questionnaire in such a way because, in this specific case, we assumed that even not knowing the terminology used by researchers, the participants may use any change impact analysis in practice. This result shows that the industry did not well adopt the researchers' jargon on this topic.

Refactoring is the concept that the participants are most familiar with (94%). Besides, 79.5% of the participants perform code refactoring. *Bad smell* is a more balanced concept in terms of theoretical knowledge and practice: 60.7% declared to know the concept of bad smell, and 52.7% of the participants affirmed to verify bad smells in software code.

Software metrics are the second most popular concept (68,8%). Nevertheless, it is the least applied one. Only 33% of the participants declared to use software metrics. However, 68.8% of the participants consider software metrics essential or very important, contrasting with 9.8% who believe that it is of little importance; 21.4% informed they do not have the background to manifest their opinion about such subject. These results

suggest that when knowing the concept of software metrics, the practitioner is likely to consider their application important but not always actually apply it.

RQ3. Which are the tools most used by practitioners in software maintenance?

We asked participants whether they use tools to collect metrics, perform refactoring, and change impact analysis. In the sequence, we present the results for each of the subjects covered.
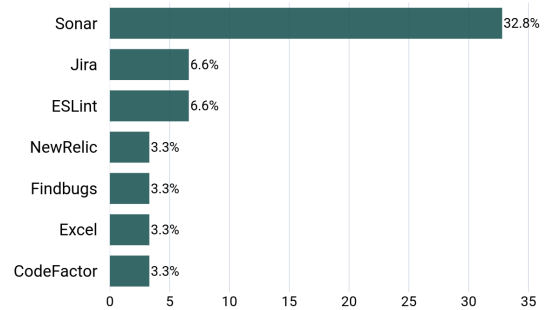
Fig. 6: Tools most used by practitioners to collect metrics.

Software Metrics. Only 33% of the participants affirmed to use a software measurement tool. The participants pointed out 62 tools. Figure 6 shows the citation percentage of the most used tools. SonarQube is the most used tool; 32.8% of practitioners used this platform to collect metrics. About 6.6% of the participants cited ESLint and Jira, and 3.3% of practitioners pointed out CodeFactor, Excel, FindBugs, and NewRelic. Practitioners mentioned other tools just once.
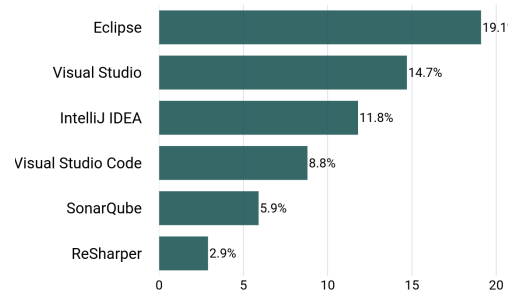
Fig. 7: Tools most used by practitioners to perform refactoring.

Refactoring. Only 36.6% of the participants declared they use tools to perform refactoring. They mentioned 68 tools. It is worthwhile to notice that 79.5% of the participants perform refactoring. Therefore, we may assume that they perform refactoring manually. Figure 7 shows the distribution of the citations of the tools. The most commonly used tools for refactoring are the IDE Visual Studio (37.2%), Eclipse (30.2%), and IntelliJ IDEA (11.8%), or an extension of an IDE, such as ReSharper (4.7%), an extension of the Visual Studio platform. 9.3% of the participants use SonarQube to perform code refactoring. Participants mentioned the other tools just once.

45

Change Impact Analysis. The participants pointed out 13 tools they use to perform this task. Figure 8 shows the percentage of the most used tool of change impact analysis. In this case, IDE is also the most cited tool: Eclipse, IntelliJ IDEA, and Visual Studio. It is essential to mention that 14.3% of the participants mentioned preferring text search ("search and replace") to perform change impact analysis, and this was the second most cited approach. However, none of them is a specific tool for change impact analysis.
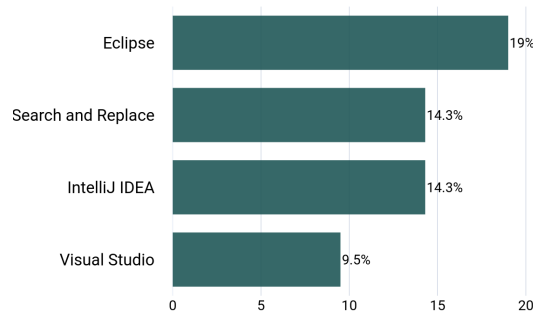

Fig. 8: Tools most used by practitioners to perform change impact analysis.

RQ4. How do practitioners perform change impact analysis?

In this research question, we investigate whether and how practitioners perform change impact analysis. Figure 9 shows the rank of the participants' answers.
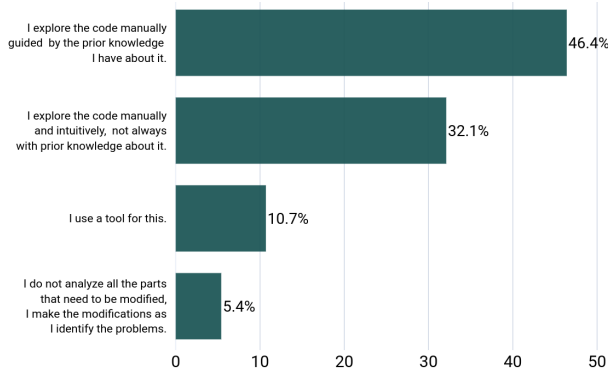

Fig. 9: How practitioners perform change and impact analysis.

Most of the participants, 46.4%, informed that they analyze parts of the code that need to be modified by exploring the code manually and intuitively, guided by prior knowledge about the software source code. Another significant part of the participants, 32.1%, declared that they explore the code manually and intuitively, not always based on prior knowledge about the system. Only 10.7% of the participants pointed out the use of tools to assist in change impact analysis. A small part of the participants, 5.4%, declared that they do not analyze all the elements that need to be modified and make the modifications as they identify the problems. Other ways

of performing change impact analysis correspond to 3.6% of the responses.

RQ5. Which metrics, refactoring techniques, and bad smells practitioners apply in their activities?

We asked the participants to point out the most common software metrics they use, the commonly refactoring techniques performed by them, and the bad smells they use to consider.

Refactoring. The most common refactoring techniques applied by the participants are: *Extract Method* (21.43%), *Rename Method* (13.39%), and *Extract Class* (12.5%).

Metrics. The analysis of RQ1 - *Are developers familiar with the concepts of software metrics, bad smells, refactoring, and change impact analysis?* - showed that software metrics is the second well-known subject. However, the participants just pointed out a few software metrics. The most cited terms regarding software metrics were: *Number of Bugs* (9.9%); *Test Coverage* (8.91%); and *Cyclomatic Complexity* (7.92%). In the last decades the literature proposed many software metrics. Nevertheless, the results of this survey show that they have not been widely applied in the industry. In particular, we noticed that practitioners do not mention the widely known software metrics in academia, such as those proposed by Chidamber and Kemerer [14].

Bad Smell. The most cited bad smells were *Duplicate Code* (23.21%), *Long Method* (19.64%), and *Long Class* (9.82%). This result indicates that developers' mainly assess code structure quality by means of code duplication, method size, and class size.

RQ6. What are the biggest challenges faced by practitioners when carrying out software maintenance?

We asked the participants to write in an open text field, which are the main difficulties they face when performing maintenance on the software systems. The participants indicated several challenges in software maintenance. We reported them in Figure 10.

Lack of documentation is the most cited problem; 22.3% of the participants have this perception. In general, they described that this problem is made more critical due to the high turnover, which is very common in IT companies.

Lack of standard for software development, is cited by 18.8% of the participants and is the second biggest challenge. The participants mentioned that the companies' development patterns are not always followed by developers, making the code difficult to understand and change.

Legacy system is cited by 18% of the participants as a challenge in software maintenance. In this context, the participants refer to the legacy system as "*long-standing codes in which several people have already worked*".

Bad coding practices are cited by 17% of the participants. According to the participants, this problem mainly involves low code readability, poorly structured functions, replicated methods in various parts of the code, and non-explanatory comments.
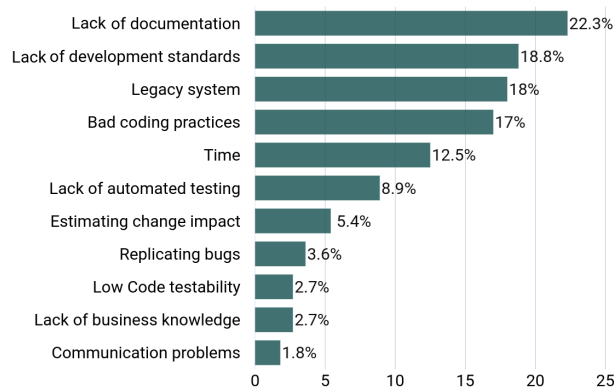
46

Fig. 10: Most important challenges faced by developers in software maintenance.

Time is also mentioned (12.5%). The participants' main issues are the short deadlines for performing maintenance-related activities and little time to understand the code and implement the change.

Lack of automated testing (8.9%), difficulty in *Replicating Bugs* (3.6%), and *Low Code Testability* (2.7%) are also challenges raised by the participants. The participants described that the absence of unit, regression, and integration tests negatively affects software code regarding automated testing. The participants associate the difficulty in code testability with the maintenance of monolithic and poorly structured systems.

Estimating change impact is pointed out as a challenge by 5.4% of the participants. According to them, not knowing how a change in the code will impact the software system raises the system's "fear" of side effects.

Lack of business knowledge (2.7%) and *Communication problems* (1.8%) also appeared as challenges for software maintenance. The participants reported that when the customer does not have a solid knowledge of the business, they demand more system changes. Practitioners also pointed out that the problems of communication with customers and coworkers negatively impact software maintenance.

VI. DISCUSSION

The results of this study lead to some insights, which we discuss in this section.

S*oftware metrics are not fully applied in practice.*: The source code software metrics mostly considered in the literature, are not applied in practice. Some possible causes for that should be investigated: lack of proper tools, lack of thresholds for the metrics, lack of knowledge of the source code metrics by developers, or the metrics proposed in the literature are not of practical use. The most used metrics are the number of bugs, test coverage, and cyclomatic complexity.

R*efactoring is a popular concept, but only the simple refactoring techniques.*: The refactoring techniques performed in practice are simple and provided by IDE: *Extract Method*, *Rename Class*, and *Extract Class*. It is essential to note that 93.7% of participants indicated that their companies use agile methodologies or a mix of agile methodologies. Among those methodologies, the participants indicated XP (eXtreme Programming), refactoring as the primary practice. Therefore, the popularity of the agile methodologies might be the cause of the popularity of refactoring. Moreover, agile methodologies emerged from the industry and not academia, which may also explain its popularity. It is essential to investigating the reasons why developers do not commonly apply other refactoring types.

T*he bad smell issue.*: Although bad smell is known by 60.7% of the respondents, only 52.7% apply the concept in practice. Previous work have pointed out two main possible reasons for that: what is considered as a potential design problem by researchers might not be a real problem in practice; bad smells detection is not of practical use. We add two hypotheses point out by the participants: developers do not apply bad smells because they do not know the concept deeply; lack of proper tools and strategies for bad smell detection.

C*hange impact analysis is not adequately performed in practice.*: The results concerning change impact analysis showed that even not knowing the term used in the literature, it is the most applied technique by the practitioners. A possible cause is that change impact analysis is an intrinsic and indispensable activity in software maintenance. Nevertheless, practitioners do not use the proper tools to perform it. Most developers (78.5%) perform change impact analysis manually, guided or not by the previous knowledge they have about the code analyzed, as described in the results. The participants reported they still use inadequate mechanisms such as "search and replace" to perform change impact analysis. Additionally, the participants pointed out change impact analysis as a significant challenge in software maintenance. These results indicate the need for proper techniques to aid change impact analysis.

D*ifficulties source code maintenance.*: The main challenges in performing code maintenance are lack of documentation, lack of standard software development, legacy systems, and bad coding practices. The research community well knows such problems. However, this result indicates that the proposed solutions for such problems were not enough to overcome such challenges in practice despite the effort made to solve them.

C*hallenges for academia and industry:* The results indicate that the gap between academia and industry in software maintenance is large. For instance, refactoring and bad smell are related concepts first published in 1999 [15]. The knowledge of refactoring has spread, but the knowledge of bad smell has not. Moreover, practitioners know only simple refactoring techniques. Object-oriented software metrics have been discussed since the early '90s but were not adopted by the practitioners, although they consider software measurement necessary. Change impact analysis is an intrinsic and challenging task of software maintenance and has been studied by researchers. However, the industry does not apply any efficient automatic tool in this task. We may raise some non-exclusive hypotheses for that: some techniques proposed in academia

is not useful in practice; the lack of proper methods, tools, and recommending systems for detecting bad smells, change impact analysis, and software measurement may be a reason why such techniques are not well applied in practice; the software engineer education may be outdated; the way the academic proposals are published does not reach practitioners. The results emphasize that the research community should strive to define more proper techniques and tools to aid software maintenance practice. On the other hand, the industry should invest more in updating the knowledge on the field.

VII. THREATS TO VALIDITY

We based the survey data on the responses collected by a questionnaire. Therefore, the data are susceptible to the participants' interpretation. To mitigate this issue, we ran a pilot questionnaire with accurate data for our analysis. Based on the answers collected in this initial round, we constructed a final survey to remove ambiguities and biases.

Our study relies on the responses given by 112 participants. We just considered exclusively participants that are professionals of software development and maintenance. Besides, the sample used in this paper is composed of participants from 92 companies and 12 countries, with a wide range of years of professional experience, with all kinds of academic backgrounds, and using a large number of programming languages.

To identify the main challenges the participants face when performing software maintenance, we asked them to describe their difficulties in an open text field. The answers to this question were manually categorized. Therefore, they are subject to interpretation by those who performed this categorization. To mitigate this threat to validity, we standardized the labels used in the categorization, i.e., we identified the keywords mentioned in the participants' responses, such as documentation, legacy system, readability, and standard, among others. Besides, the label assigned to each answer was made separately by two authors of this paper. After this, we compared the classifications to obtain the final classification of each answer. In case of divergence in the categorization, both authors analyzed it to obtain a consensus.

VIII. CONCLUSION

We surveyed software practitioners to investigate whether and how software maintenance techniques have been applied in practice. In particular, we investigated the usage of the following concepts and techniques: software metrics, refactoring, bad smells, and change analysis impact. For this purpose, we surveyed 112 software development practitioners from 92 companies and 12 countries. The results showed that change impact analysis is the most applied technique among the ones considered in this work. However, there is a lack of proper tool support to perform change impact analysis. Although refactoring is a widespread technique, few refactoring techniques have been applied in practice. Moreover, refactoring is mostly provided by IDE. Bad smells and software metrics are the less known and applied concepts.

This study also revealed that participants considered the lack of system documentation, lack of development patterns, and legacy software as the leading software maintenance challenges. The results indicate that software maintenance demands even more community effort to develop and provide proper tools and methods for software maintenance, especially in change impact analysis and software measurement.

We intend to perform two additional analyses with this survey's data: the differences in how practitioners approach software maintenance depending on their academic background and professional experience. We also envision the following main future work: replicate this study with other software engineering techniques, investigate how agile methodologies have been applied in practice, and detail the reasons why software metrics have been timidly applied in the industry.

REFERENCES

[1] D. Parnas, "Software engineering - missing in action: A personal perspective," *Computer*, vol. 44, no. 10, pp. 54–58, Oct 2011.

[2] D. Lo, N. Nagappan, and T. Zimmermann, "How practitioners perceive the relevance of software engineering research," in *10th Joint Meeting on Foundations of Software Engineering (FSE)*, 2015, pp. 415–425.

[3] J. C. Carver, O. Dieste, N. A. Kraft, D. Lo, and T. Zimmermann, "How practitioners perceive the relevance of esem research," in *10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, 2016, pp. 56:1–56:10.

[4] A. S. Bordin and F. B. V. Benitti, "Software maintenance: What do we teach and what does the industry practice?" in *32th Brazilian Symposium on Software Engineering (SBES)*, 2018, pp. 270–279.

[5] B. G. Bern, "From theory to practice: Experiences of industry-academia collaboration from a practitioner," ser. SER&IP '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 22–23.

[6] P. Runeson, "It takes two to tango – an experience report on industry – academia collaboration," in *Proceedings of the 2012 IEEE Fifth International Conference on Software Testing, Verification and Validation*, 2012, p. 872–877.

[7] I. C. Society, P. Bourque, and R. E. Fairley, *Guide to the Software Engineering Body of Knowledge (SWEBOK)*. IEEE Computer Society Press, 2004.

[8] E. Murphy-Hill, C. Parnin, and A. P. Black, "How we refactor, and how we know it," *IEEE Trans. Softw. Eng.*, vol. 38, no. 1, p. 5–18, Jan. 2012.

[9] E. Kupiainen, M. V. Mäntylä, and J. Itkonen, "Using metrics in agile and lean software development – a systematic literature review of industrial studies," *Information and Software Technology*, vol. 62, pp. 143 – 163, 2015.

[10] F. Palomba, G. Bavota, M. Di Penta, R. Oliveto, and A. De Lucia, "Do they really smell bad? a study on developers' perception of bad code smells," in *2014 IEEE International Conference on Software Maintenance and Evolution*, 2014, pp. 101–110.

[11] A. Tahir, A. Yamashita, S. Licorish, J. Dietrich, and S. Counsell, "Can you tell me if it smells? a study on how developers discuss code smells and anti-patterns in stack overflow," in *Proceedings of the 22nd International Conference on Evaluation and Assessment in Software Engineering 2018*, 2018, p. 68–78.

[12] S. Canada, *Survey Methods and Practices*. Statistics Canada, 2010, vol. Catalogue no. 12-587-X.

[13] B. A. Kitchenham and S. L. Pfleeger, *Personal Opinion Surveys*. London: Springer London, 2008, ch. 3, pp. 63–92.

[14] S. R. Chidamber and C. F. Kemerer, "A metrics suite for object oriented design," *IEEE Transactions on Software Engineering*, vol. 20, no. 6, pp. 476–493, June 1994.

[15] M. Fowler and K. Beck, *Refactoring: Improving the Design of Existing Code*, second edition ed. USA: Addison-Wesley Longman Publishing Co., Inc., 2018.