A survey on graduates' curriculum-based knowledge gaps in software testing

Lilian P. Scatalon*, Maria Lydia Fioravanti*, Jorge M. Prates*‡, Rogério E. Garcia† and Ellen F. Barbosa*
* University of São Paulo (ICMC-USP), São Carlos, SP – Brazil
† São Paulo State University (FCT-UNESP), Presidente Prudente, SP – Brazil
‡ Mato Grosso do Sul State University (UEMS), Nova Andradina, MS – Brazil
Email: {lilian.scatalon, mlfioravanti, jorgemprates}@usp.br, rogerio.garcia@unesp.br, francine@icmc.usp.br

***Abstract*—This research full paper presents a study on graduates' knowledge gaps in software testing according to industry needs. Several studies indicate that students graduate from computing programs with a knowledge gap in software testing. In this sense, we aimed to investigate in details this broader testing gap, by considering gaps in the level of testing topics. We conducted a survey with Brazilian practitioners in order to collect data (N=90). For each testing topic, knowledge gaps were calculated as the difference between what respondents' learned/practiced in undergraduate courses and what they actually applied in industry after graduating. Results provide evidence on points that could be improved in software testing education. Firstly, for all testing topics there was a negative gap on practice activities. This means that students could benefit from more testing assignments throughout the curriculum. Regarding gaps in concepts, some testing topics presented negative gaps (such as test in web applications) and others positive (such as test in aspect oriented software and mutation analysis). Therefore, results suggest that it is possible to counterbalance them in order to reduce the existing gaps. We also present respondents' opinions about their experience in software testing education and industry practices.**

I. Introduction

Teaching students effectively in any subject is a challenging endeavor by itself. In particular, Software Engineering (SE) present additional challenges that are inherent to the subject. Students can face difficulties to learn about the software development process, since it is an intricate set of activities, performed by people who should communicate effectively with each other, using different tools and techniques.

Besides, Software Engineering presents a landscape that changes quickly, what makes practitioners have to constantly cope with the emergence of new tools, techniques and methods. Given this scenario, it is specially difficult to give students a realistic notion about what it takes to build software.

Lethbridge et al. pointed out a set of challenges that researchers, curriculum designers and instructors should address in order to improve SE education [9]. One challenge that stands out is that SE education should have a closer relation with industry practices.

Radermacher and Walia [13] pointed out several areas in which graduates most frequently do not meet industry needs, ranging from technical skills (such as programming) to personal skills (such as written and oral communication). Software testing was among the main areas presenting knowledge deficiencies, according to their study.

In the same direction, results from a study conducted by Carver and Kraft indicate that senior-level students are not able to effectively and thoroughly test even simple programs [2]. This is a cause for concern, since testing is not an activity restricted only to quality assurance professionals, but all software developers can be involved with some kind of software testing.

Still according to Radermacher and Walia, previous studies in the literature do not provide specific information about which testing topics raise knowledge deficiencies [13]. Therefore, there is a need to investigate in details what are the knowledge gaps in the testing area.

Considering this scenario, in this paper we investigated graduates' knowledge gaps in software testing, looking specifically into testing topics. These topics were taken from the contents of a software testing textbook [3]. We conducted a survey with practitioners asking them about which testing topics were addressed in their undergraduate courses and which they had to apply in industry.

We followed the same definition of knowledge gap given by Lethbridge [8]: a difference between what is learned in education and what is the topic importance. In our study, topic importance is given by industry needs, i.e., by whether a graduate had the need to apply it in industry.

We considered two kinds of knowledge gaps: in concepts (theory) and in practice activities. We chose to address separately gaps in practice activities because software testing is inherently a practical activity, similarly to programming. The teaching of programming usually is filled with many programming assignments to help students develop their programming skills. It would be expected that students learn about software testing through many testing assignments as well.

Results from our survey help to shed light on how to deal with the broader knowledge gap in software testing identified by Radermacher and Walia [13]. Moreover, it identifies "smaller" knowledge gaps, in the lower-level of testing topics. This kind of information is very useful for instructors and curriculum designers that intend to adjust and improve software testing education in computing courses.

The remainder of this paper is organized as follows. In Section II, we discuss similar studies about the comparison between SE education and industry needs. We also discuss similar surveys with practioners about testing practices. In

978-1-5386-1174-6/18/$31.00 ©2018 IEEE

Section III, we discuss the survey design. In Section IV we point out some threats to validity raised by the choices in survey design. Results are presented and discussed in Section V. Finally, we draw conclusions and provide directions of future work in Section VI.

II. Related work

There are several studies comparing what is covered by computing education and what are software industry needs. This kind of investigation is relevant because there is a high demand for qualified software professionals. Attempting to align computing education with industry practices is a good way to address this demand.

Moreno et al. [10] conducted a comparison between curricular guidelines and job profiles. They identified the relationships between recommended computing competences and relevant skills to software professionals. Their results indicate that even curriculum guidelines do not cover all the core knowledge needed by professionals to perform their jobs in industry. This means that, when implemented in specific colleges or universities, these curriculum guidelines would cause knowledge deficiencies in graduates.

Similarly, Radermacher and Walia [13] conducted a systematic literature review looking for knowledge deficiencies reported by previous studies. A knowledge deficiency is defined by them as any knowledge or skill that industry expects an entry-level practitioner to have and she/he lacks it. The same definition applies for academia and graduate students. The authors discussed these deficiencies in the level of Computer Science areas, such as programming, design and testing. Our study investigates in detail one of the areas mentioned by their study: software testing.

In terms of method, Lethbridge [8] conducted a study which is more similar to the one described in this paper. The author conducted a survey with software professionals from several countries to assess the importance of computing topics (data structures, software design and patterns etc) to their career. *Testing, verification, and quality assurance* was among the topics considered more important to respondents. Also, it was among the topics that are more learned in the job, as opposed to learned in formal education. Kitchenham et al. [6] performed a similar survey in the context of UK universities.

Additionally, there are several surveys about software testing practices, but focusing only in industry. They vary in scope, ranging from multiple aspects of testing practices [5, 11] to a single type of test [4, 14] and target practitioners from different nationalities, like Australia in [11] and Canada in [5]. In general, they were all aiming to get a snapshot of current testing practices in industry.

Our survey explored testing practices adopted by practitioners in industry, but we also explored the software testing education delivered to them in undergraduate courses. The idea is to investigate whether the topics addressed in software testing education have been applied by the respondents in their jobs. It was directed to Brazilian practitioners and the questionnaire covered multiple aspects of software testing.

III. Survey design

We followed the guidelines of Kitchenham and Pfleeger to design and execute our survey [7]. The goal was to investigate the following research question:

> *What are the knowledge gaps in testing topics faced by graduates with respect to industry needs?*

In order to answer it, we needed data from two different contexts: software testing education and testing practices in industry. Therefore, our strategy was to collect data about these two contexts from practitioners that are graduates from computing undergraduate programs.

We recruited respondents by sending emails with a link to a web-based questionnaire. We took advantage of mailing lists for graduates from several Brazilian universities and also the Brazilian Computer Society mailing list.

The questionnaire was composed by two sections (see the appendix). In the first section we were seeking to find out about respondents' educational and professional background. Regarding education, we collected respondents' major and asked which computing courses they took that addressed software testing. About the professional profile, we collected their current position in the company, years of experience in software development and the programming languages generally used in their projects.

The second section had the purpose to evaluate the knowledge gaps in software testing. To this end, we collected data about respondents' undergraduate education and industry practice in testing. Then, we compared their responses for these two contexts to obtain the gaps. Still, we considered two kinds of knowledge gaps: in concepts ($\mathbf{gap}_C$) and in practice activities ($\mathbf{gap}_P$). Therefore, questions from the second section presented software testing topics to respondents and asked them to check the following options for each one:

- Industry: if they have applied the testing topic in their job.
- Concepts in education (Education_C): if during their major they have learned about the theory of the testing topic.
- Practice activities in education (Education_P): if during their major they have completed hands-on activities (such as programming/testing assignments) that gave them the opportunity to put the testing topic into practice.

We took the topics from the textbook on software testing of Delamaro et al. [3], which provides a set of testing topics that are usually addressed in computing courses. We also considered the questionnaire used in Garousi and Zhi's survey [5], which helped to organize the topics by characteristics of the testing activity (types of systems under test, testing levels, test types, testing approach in the development process and test case generation techniques).

In particular, we included a question asking respondents to mention which testing tools/frameworks they have used in their company. Additionally, at the end of the questionnaire, there was an optional question where respondents could add comments about their experience with software testing education and testing practices in industry.

Regarding how we calculated the knowledge gap for each testing topic, firstly we assigned values for the options Industry, Education$_C$ and Education$_P$. When a respondent had checked the option, the assigned value was one, and zero otherwise. In this way, following the same definition of knowledge gap from Lethbridge et al. [8], we were able to define equations to calculate the knowledge gaps for concepts and practice activities in education, respectively:

gap$_C$ = Education$_C$ - Industry
gap$_P$ = Education$_P$ - Industry

By applying these equations, we got both kinds of knowledge gaps (in concepts and practice activities) for each respondent in each testing topic. Considering the results individually like this, the possible resulting values are the following:

- gap = 0, when there is no knowledge gap. Either the testing topic was addressed in education and used in industry, or it was not addressed nor used.
- gap = -1, when there is a knowledge gap, which is a knowledge deficiency that a graduate faced while doing her/his job. She/he had to apply the testing topic at industry, but has not learned (or practiced) it during the major.
- gap = 1, when there is also a knowledge gap, but it can be considered as a "knowledge abundance", since it was addressed in education, but it was not applied in industry by the graduate.

Under the same reasoning, the overall knowledge gap for a given testing topic t was calculated by the average of gaps in that topic for all respondents:

$$gap_t = \frac{\sum_{s=1}^{N} gap_t si}{N}$$

where $gap_t si$ is the knowledge gap in topic t for the respondent s_i and N is the total number of respondents (90). Then, by applying this equation, we got the values of the average knowledge gaps within the interval $-1 < gap_t < 1$.

IV. Threats to Validity

The choices for survey design and conduction involve threats to validity, as it happens with any empirical study. The first one concerns generalization of results. Our sample of respondents are from Brazilian practitioners and results are limited to represent the educational and industry context from Brazil. Therefore, knowledge gaps calculated in our study are also limited to this context. Nevertheless, it can be a good indicative of points to be adjusted in testing education when seeking to meet industry needs.

Other threat concerns the accuracy of responses. Respondents had to remember about the studied/applied testing topics to answer the questionnaire. They had to inform about events that could have happened many years before. In this sense, knowledge gaps can be influenced by compromised memory, since they may have forgotten about details from undergraduate courses.

Also, the nature of the Software Engineering area itself can have an influence on results, since it presents a quickly changing landscape. In this way, a knowledge gap could simply indicate an outdated technology. This situation can apply specially for recent graduates.

Lastly, we considered knowledge gaps from the industry viewpoint. However, meeting industry expectations is not the only purpose of computing education, which should provide a good theoretical foundation that will always be used indirectly by graduates to learn about new technologies. Furthermore, computing education should develop other kinds of students' abilities besides the technical ones, such as communication, teamwork and ethics [13].

V. Results

In this section we present the survey results. We received 90 responses from graduates in total.

A. Educational profile

Aiming to get an overview of the educational context from where we are assessing the knowledge gaps, respondents were asked to provide their academic major (see Figure 1). Computer Science is the major from over half (63.33%) of the respondents, followed by Information Systems (16.67%) and Computer Engineering (8.89%). Some respondents (11.11%) mentioned other majors, such as Data Processing, Electrical Engineering, and Software Analysis and Development.

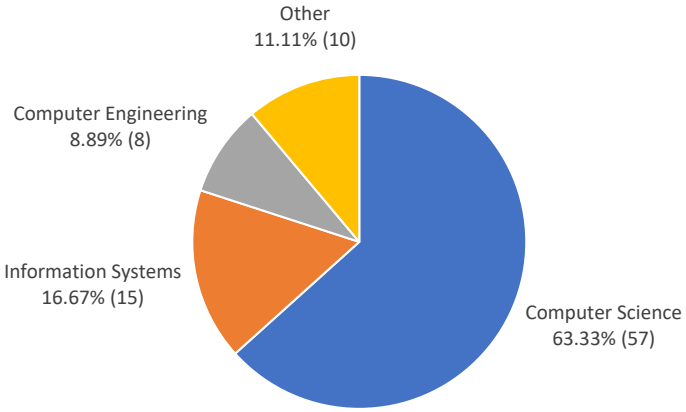

Fig. 1. Respondents' major

Next, in order to understand specifically the context of software testing education, they were asked to inform which courses they took that addressed software testing (see Figure 2). Different computing courses can address this subject, so this was a question allowing multiple answers.

One interesting way to analyze these results is by noticing how much of software testing is addressed in entry-level and upper-level courses. Only 20% (18) of the respondents learned about software testing early in the curriculum, during introductory programming courses. In contrast, 82% (74) learned about testing in the Software Engineering course. Moreover, for 50% (45) of the respondents it was the only course that addressed software testing.

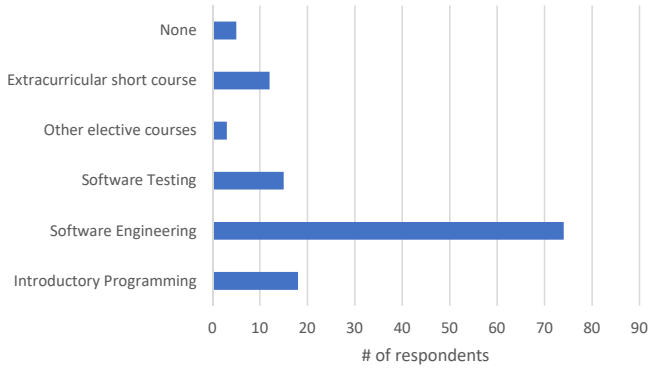

Fig. 2. Courses that addressed Software Testing in respondents major

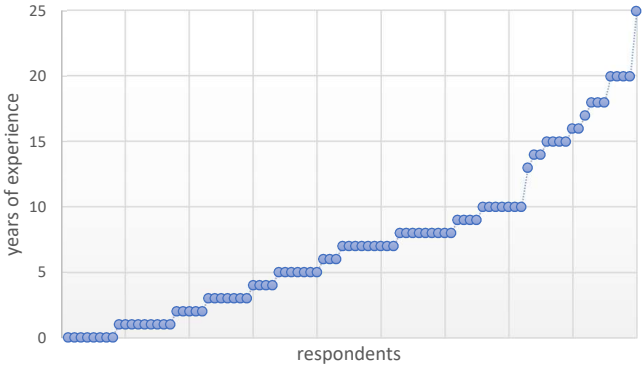

Fig. 4. Respondents' years of experience in industry

Fewer respondents (17% – 15) took a course dedicated to this subject or other elective courses (3% – 3). Some of them (13% – 12) took short courses not included in the curriculum that addressed testing. For 6% (5) of the respondents this subject was not even addressed during the major.

B. Professional profile

Figure 3 shows respondents' current positions. Many of them are software developers (40 respondents). A smaller group work specifically with software quality assurance (QA), as testers (15) or as QA analyst/lead (14). Some of the respondents work in other roles, such as project manager (9), product owner (8) and scrum master (2).

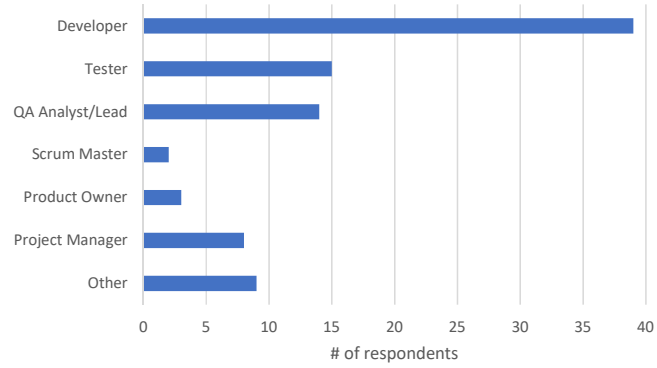

Fig. 3. Respondents' current position in industry

The distribution of work experience in years is given in Figure 4. Most of them (82% – 72) had up to ten years of work experience. The average was 7.32 years, with a standard deviation of 5.91 years and a median of 7 years. There was a significant variability in respondents' experience and this can contribute positively to the study, since professionals in different moments of their career can bring complementary contributions to the results.

We also collected the programming languages used in the projects that respondents are involved, which are showed in Figure 5. Most respondents (71% – 64) mentioned working with Java, followed by other languages like JavaScript, Python and C++. The choice of programming language is important to the software testing activity, since each one has different kinds of existing supporting tools.

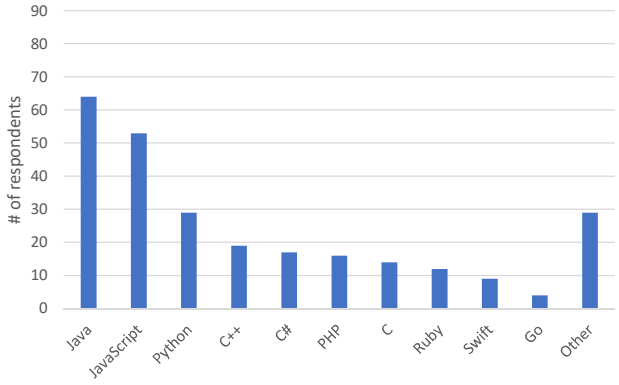

Fig. 5. Programming languages used in respondents' projects

C. Knowledge gaps on software testing

Table I presents the knowledge gaps for testing topics, considering the average among all respondents. Similarly to individual gaps, when the value is negative, it means there is a knowledge deficiency in that topic, either in terms of concepts ($\mathbf{gap}_C$) or practice activities ($\mathbf{gap}_P$). When the value is positive (highlighted in bold), it means there is a knowledge abundance in that topic.

It is possible to note that all gaps related to practice activities ($\mathbf{gap}_P$) are negative, indicating that there is a lack of practice in software testing education. The values of knowledge gaps in concepts ($\mathbf{gap}_C$) allow to assess if the coverage of testing topics is adequate. A negative value suggests that the corresponding testing topic is being underemphasized. Similarly, a positive value indicates an overemphasized testing topic.

Most testing topics present knowledge deficiencies (negative gaps/values). But focusing on the higher absolute values, we highlight a significant knowledge deficiency in the topic of Web applications (for both concepts and practice), for practice

TABLE I
KNOWLEDGE GAPS ON SOFTWARE TESTING

Testing topic	gap_C	gap_P
Types of systems under test		
Web applications	-0.53	-0.67
Mobile applications	-0.36	-0.46
Object oriented software	-0.16	-0.50
Aspect oriented software	**0.07**	-0.08
Concurrent programs	-0.02	-0.20
Testing levels		
Unit testing	-0.18	-0.56
Integration testing	-0.44	-0.82
System testing	-0.40	-0.71
Regression testing	-0.40	-0.66
Test types		
Functionality testing	-0.37	-0.69
Performance testing	-0.48	-0.69
GUI testing	-0.30	-0.51
Usability testing	-0.09	-0.36
Security testing	-0.16	-0.41
User acceptance testing	-0.24	-0.51
Testing approach in the development process		
Test-driven (first) development (TDD)	-0.19	-0.42
Test-last development	-0.26	-0.44
Test case generation techniques		
Client requirements/user stories	-0.27	-0.58
Category partitioning	-0.07	-0.32
Boundary value analysis	-0.13	-0.37
Cause-effect graph	**0.17**	-0.13
Finite state machine	**0.29**	-0.02
Control flow graph	**0.19**	-0.04
Data flow analysis	**0.18**	-0.18
Mutation analysis	**0.20**	-0.06

in all testing levels (unit, integration, system and regression testing) and practice of using client requirements/user stories as a test case generation technique.

On the other hand, knowledge abundance occurs only for concepts in the following testing topics: test of aspect oriented software and most test case generation techniques (cause-effect graph, finite state machine, control flow graph, data flow analysis and mutation analysis). Therefore, the results suggest that these particular topics have not been used much in practice, at least in the respondents' companies.

Additionally, the positive knowledge gaps in test techniques may be due to the fact that some of them work better with a mature set of requirements, which is often not true in software development. In the same direction, it is possible to note a higher demand for functional testing (category partitioning and boundary value analysis) and writing test cases from requirements/user stories.

It is interesting to point out that many respondents indicated behavior-driven development (BDD) as an approach to undertake testing during the development process [12]. We do not have the knowledge gap for this particular testing topic, since it was not included in the textbook contents.

Even so, it is probably a topic that should be more addressed in software testing education, specially considering its relation with other topics that presented significant negative gaps (functionality testing and the use of client requirements/user stories to generate test cases). Many BDD-related tools were also mentioned by respondents (Section V-D).

D. Supporting tools

Since there was a high number of different tools (83 in total), results are given in Figure 6 sorted by categories. There was a prominence of Web application testing tools (such as Selenium and JMeter) and XUnit frameworks (such as JUnit and unittest), mentioned by, respectively, 47.8% (43) and 42.2% (38) of respondents.

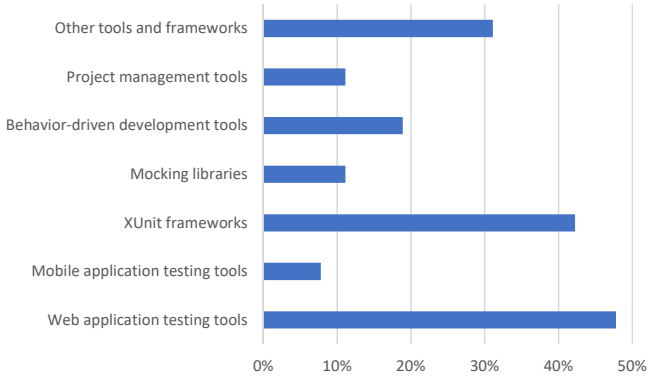

Fig. 6. Tools used in respondents' projects

E. Respondents' experiences

The last survey question took free-text answers about respondents' experiences with testing practices in industry and the software testing education delivered to them during the major. There was a response rate of 27% (24) in this question. We identified four main points in their responses:

1) **Lack of testing practice activities in computing courses**. Many practitioners reported learning in undergraduate courses about the importance of why we should test software and the basics of testing concepts. They recognized that a sound knowledge in testing fundamentals indeed help in becoming a good tester.
However, they complained about software testing education being too much theoretical, with a lack of practical scenarios to show students how the concepts should be applied and how software testing would have an impact in the medium and long term.
As a result, students graduate with little actual testing skills, or depending on how the curriculum is designed, none whatsoever. Some responses are given next, in a free translation to English:

"During my major, the subject of software testing was not addressed in a detailed way. It was restricted to only high-level concepts in the software engineering course. Almost all the learning I had about testing took place in industry"
"I faced difficulties in adapting myself to industry needs, since it usually required experience with BDD or TDD, but I had never done any testing in practice"
"When I started working as a QA analyst, I remember I had the feeling: 'I never saw any of this during my major, except learning the existence of these types of tests' "
"It would be very useful if students were encouraged to submit their assignments with tests, at least in upper-level courses"

2) **Distance between software testing education and real-world practices**. Some respondents reported a good coverage of testing concepts in the major and some did not. This variation may be due to differences on how computing curricula are designed in different universities. But, similarly to what was advocated by Lethbridge et al. [9], there was an agreement that testing education is distant from real-world practices:
"Unfortunately the testing activities explored in academia still are distant to most of what is applicable in industry"
"Undergraduate courses gave me a good notion of industry practices, but they evolved quickly"
"I seldom applied in my job what I learned from my major, because it is distant of industry reality in general"
"The concepts I learned during my major were good, but I think there was a lack of applying them in real situations."
3) **Software testing culture in industry**. It is interesting to notice the variation in the software testing culture from each company. While some respondents reported learning about testing in industry, thereby implying a good testing culture in their company, some of them explicitly tell about a poor testing culture:
"Many companies and development teams do not value at all the testing activity. Only technically strong teams seem to encourage it."
"In my experience, software testing was more present in the major than in the company where I work, which only cares about system testing."
4) **Factors that lead to design ineffective test suites**. Respondents pointed out how other development phases, such as analysis and design, are crucial to software testing. Additionally, they mentioned the importance to develop a tester mindset, which is only possible through practice:
"Testing requires a lot of practice in order to be really effective and not highly coupled with the application design. There is also a high deficiency in relation to software design, which should ease the testing in the first place"
"Requirements analysis is critical to create effective test cases and this depends on the experience of the test analyst. Test techniques help, but without a mature set of requirements, even with a good application of testing techniques and criteria, the constructed tests will be poor"
"I took a course that addressed well unit testing. But regarding functional testing and what concerns developing a tester mindset, it was a very superficial approach."

VI. Conclusions and future work

In this paper we presented an investigation about graduates' knowledge gaps in software testing, considering industry needs. We conducted a survey with software professionals, collecting data about the testing education delivered to them and about testing practices they have applied in industry.

We considered knowledge gaps in two aspects: in concepts and in practice activities. This distinction allowed us to assess knowledge gaps in terms of how the teaching of theory and practice in software testing has been addressed in computing undergraduate programs. Additionally, the knowledge gaps were represented by values that range from -1 to 1. This raises two kinds of knowledge gaps: knowledge deficiency (negative gap) and knowledge abundance (positive gap). They can indicate, respectively, when topics are being underemphasized and overemphasized.

Ideally, all gaps should be close to zero in order to reflect a good software testing education according to industry needs. However, there are clear limitations about time and resources that do not allow to address every topic of a given subject in depth. Even so, considering that results show positive and negative gaps, there is room to counterbalance them.

In general, results indicated a deficiency for all testing topics in practice activities. In particular, there were also negative gaps in topics such as test of web applications, functionality testing and test case generation from client requirements/user stories. On the other hand, there were some positive gaps on topics like test on aspect oriented software and some test case generation techniques (cause-effect graph, finite state machine, control flow graph, data flow analysis and mutation analysis). Therefore, these topics could be considered in order to make adjustments in software testing education, when seeking to reduce graduates' knowledge gaps.

We also collected comments on respondents' experience with software testing education and industry testing practices. In summary, from the educational point of view, they reported a lack of testing practice activities in computing courses (as results about knowledge gaps have also suggested) and a distance of testing education from real-world practices. This distance was also reported in the broader context of Software Engineering by Lethbridge et al. [9].

From the industry point of view, they reported about testing culture in the companies, which not always encourage practitioners to test software, and they pointed out some factors that can lead to ineffective test suites, such as changing requirements, bad software design and the lack of a tester

mindset. Therefore, these factors should also be considered when trying to improve software testing education.

As short-term future work, we intend to investigate how introductory programming courses can help to reduce the identified knowledge gaps. Introductory courses provide an adequate context to encourage the use of testing practices, since students are constantly working on programming assignments [1, 15]. The idea is to deal with negative knowledge gaps in practice activities, which were present for all testing topics.

ACKNOWLEDGMENTS

We would like to thank the Brazilian funding agencies – CAPES (Procad 071/2013), CNPq and FAPESP (grant 2014/06656-8). We also thank the practitioners that participated in the survey and the reviewers for their comments.

REFERENCES

[1] E. Barbosa, M. Silva, C. Corte, and J. Maldonado. Integrated teaching of programming foundations and software testing. In *Annual Frontiers in Education Conference (FIE)*, pages S1H–5–S1H–10, Oct 2008.

[2] J. C. Carver and N. A. Kraft. Evaluating the testing ability of senior-level computer science students. In *24th IEEE-CS Conference on Software Engineering Education and Training (CSEE&T)*, pages 169–178, May 2011.

[3] M. E. Delamaro, J. C. Maldonado, and M. Jino. *Introducao ao teste de software (Introduction to software testing)*. Elsevier, 2nd edition edition, 2016.

[4] E. Engström and P. Runeson. A qualitative survey of regression testing practices. In *Proceedings of the 11th International Conference on Product-Focused Software Process Improvement (PROFES'10)*, pages 3–16, Berlin, Heidelberg, 2010. Springer-Verlag.

[5] V. Garousi and J. Zhi. A survey of software testing practices in Canada. *Journal of Systems and Software*, 86(5):1354–1376, May 2013.

[6] B. Kitchenham, D. Budgen, P. Brereton, and P. Woodall. An investigation of software engineering curricula. *Journal of Systems and Software*, 74(3):325 – 335, 2005.

[7] B. A. Kitchenham and S. L. Pfleeger. *Personal Opinion Surveys*, pages 63–92. Springer London, London, 2008.

[8] T. C. Lethbridge. What knowledge is important to a software professional? *Computer*, 33(5):44–50, 2000.

[9] T. C. Lethbridge, J. Diaz-Herrera, R. J. J. LeBlanc, and J. B. Thompson. Improving software practice through education: Challenges and future trends. In *Future of Software Engineering (FOSE '07)*, pages 12–28, May 2007.

[10] A. M. Moreno, M.-I. Sanchez-Segura, F. Medina-Dominguez, and L. Carvajal. Balancing software engineering education and industrial needs. *Journal of Systems and Software*, 85(7):1607 – 1620, 2012. Software Ecosystems.

[11] S. P. Ng, T. Murnane, K. Reed, D. Grant, and T. Y. Chen. A preliminary survey on software testing practices in Australia. In *Proceedings of the 2004 Australian Software Engineering Conference*, ASWEC '04, pages 116–, Washington, DC, USA, 2004. IEEE Computer Society.

[12] D. North. Introducing BDD, 2006. Available at https://dannorth.net/introducing-bdd.

[13] A. Radermacher and G. Walia. Gaps between industry expectations and the abilities of graduates. In *Proceeding of the 44th ACM Technical Symposium on Computer Science Education (SIGCSE '13)*, pages 525–530, New York, NY, USA, 2013. ACM.

[14] P. Runeson. A survey of unit testing practices. *IEEE Software*, 23(4):22–29, July 2006.

[15] J. L. Whalley and A. Philpott. A unit testing approach to building novice programmers' skills and confidence. In *Proceedings of the Thirteenth Australasian Computing Education Conference (ACE '11)*, pages 113–118, Darlinghurst, Australia, Australia, 2011. Australian Computer Society, Inc.

APPENDIX

Survey questionnaire

1) What is your current position in the company?

2) How many years of work experience do you have in software development?

3) What is your university degree in?

- ☐ computer science
- ☐ computer engineering
- ☐ information systems
- ☐ software engineering
- ☐ other:

4) What computing courses addressed software testing in your major?

- ☐ introductory programming courses
- ☐ software engineering course
- ☐ software testing course
- ☐ extracurricular short course
- ☐ other:

5) Which programming languages are generally used in your projects?

- ☐ Java
- ☐ C
- ☐ C++
- ☐ Python
- ☐ C#
- ☐ PHP
- ☐ JavaScript
- ☐ Ruby
- ☐ Perl
- ☐ Swift
- ☐ other:

The remainder of the questionnaire addresses elements that characterize the testing activity. In questions 6 to 10, for each element, please analyze the testing approaches listed in each line and check the following columns:

- Applied in industry, if you have applied the testing approach in your job.
- Concept addressed in major, if you have learned about the theory of this testing approach during your major.
- Practice activities in major, if you have completed hands-on activities (such as programming/testing assignments) that gave you the opportunity to put the testing approach into practice.

6) Types of systems under test

	Applied in industry	Concepts addressed in major	Practice activities in major
Web applications	☐	☐	☐
Mobile applications	☐	☐	☐
Object oriented software	☐	☐	☐
Aspect oriented software	☐	☐	☐
Concurrent programs	☐	☐	☐

Other types of systems under test:

7) Testing levels

	Applied in industry	Concepts addressed in major	Practice activities in major
Unit testing	☐	☐	☐
Integration testing	☐	☐	☐
System testing	☐	☐	☐
Regression testing	☐	☐	☐

Other testing levels:

8) Test types

	Applied in industry	Concepts addressed in major	Practice activities in major
Functionality testing	☐	☐	☐
Performance testing	☐	☐	☐
GUI testing	☐	☐	☐
Usability testing	☐	☐	☐
Security testing	☐	☐	☐
User acceptance testing	☐	☐	☐

Other test types:

9) Testing approach in the development process

	Applied in industry	Concepts addressed in major	Practice activities in major
Test-driven (first) development (TDD)	☐	☐	☐
Test-last development	☐	☐	☐

Other testing approaches:

10) Test case generation techniques

	Applied in industry	Concepts addressed in major	Practice activities in major
Client requirements/ user stories	☐	☐	☐
Category partitioning	☐	☐	☐
Boundary value analysis	☐	☐	☐
Cause-effect graph	☐	☐	☐
Finite state machine	☐	☐	☐
Control flow graph	☐	☐	☐
Data flow analysis	☐	☐	☐
Mutation analysis	☐	☐	☐

Other test techniques:

11) Which testing tools/frameworks do you use in your company?

12) If you wish, please comment other aspects about your experience with learning software testing in your major and industry testing practices.