Advanced Undergraduate Computing Students' Perception of Software Quality Teaching: a Survey in the Brazilian Paraná State

Guilherme N. Ferrari[1], Thais S. Nepomuceno[1], Claudia H. Santana[1], Carlos D. Luz[1], Gislaine C. Leal[1], Aline M. M. M. Amaral[1], Renato Balancieri[2], Edson OliveiraJr[1]

[1]State University of Maringá
Maringá – PR – Brazil

[2]State University of Paraná
Apucarana – PR – Brazil

{guinetoferrari, thais.nepomuceno1, chsantana}@gmail.com

carlos.danilo.luz@gmail.com, {gclleal, ammmamaral}@uem.br

renato.balancieri@unespar.edu.br, edson@din.uem.br

Abstract. *Aspects of software quality (SQ), such as process and product metrics, and assessment techniques, can be taught to computing students during their undergraduate courses, however, there is no consensus on how. In Brazil, computing courses are structured as the Brazilian Computer Society suggests, still, researchers point out that there are few SQ subjects in these courses. This paper aims to analyze the perception of SQ concepts by advanced undergraduate students in the northwest of the Paraná state. We applied a survey and received ninety-nine answers. Our results show that most SQ concepts are taught, but the students feel they did not learn and are not able to apply them. We discuss and suggest guidelines to improve the understanding of SQ concepts.*

1. Introduction

Software quality management incorporates a set of activities to guide and control the organization with a focus on processes, products, and services. Such management is an essential factor for an organization to maintain a competitive position in the market, in addition to ensuring business success, customer satisfaction, and product acceptance [Laporte et al. 2007, Jenkins 2007, Bourque et al. 2014]. The absence of Software Quality (SQ) has impacting consequences, such as delays in the delivery of new applications, high maintenance and support costs for customers, and long learning curves. [Jones and Bonsignour 2011].

For software users, quality is related to the usability and implementation of required functionalities. As for software developers, the most important quality issue is that the software meets the specifications and provides services as previously contracted [Tian 2005, Kokol 2022]. Therefore, quality is subjectively determined by those who interact with the system, whether to solve their problems through software or are responsible for designing, implementing, and testing the solutions, making judgments about the quality of the product [Gillies 2011, Suali et al. 2019]. Therefore, the use of techniques and tools that ensure and guarantee SQ is essential [Page 2008, Chren et al. 2022].

An important piece of graduates has their first contact with the job market in areas that require SQ knowledge, such as testing and maintenance. Usually, they are not deeply involved in undergraduate programs which offer educational content on how to build high-quality software or how to integrate quality activities during the software development process [Hilburn and Towhidnejad 2002, Garousi et al. 2019]. In addition, there is no consensus on how to educate future software engineers on topics of SQ and software improvement [Dingsøyr et al. 2000, Kappelman et al. 2016].

As SQ is becoming increasingly important, both because of educational requirements and the industry, it should be at the center of the undergraduate curriculum of courses that teach software development. Thus, students are introduced to a variety of SQ activities, enabling them to develop technical knowledge and skills [Hilburn and Towhidnejad 2002, Richardson et al. 2011].

According to [Bettin et al. 2022], there is a low number of specific courses on SQ in undergraduate computing courses, as well as differences between the topics covered in the program and what is expected for the graduates' profile. They claim what is taught is not enough for students to have a proper view of SQ, which might imply a deficit of skills in the software development industry. Thus, investigating such claims is an existing demand in the area.

In this context, this paper is aimed at carrying out an exploratory analysis of the perception of undergraduate Computing students in relation to the knowledge acquired in SQ courses.

The analysis was based on nine knowledge axes defined by the Brazilian Computing Society (SBC) in its educational References for Undergraduate Computing Courses. To fulfill this objective, a survey was carried out with 99 students from the last two years of different undergraduate Computing courses from higher education institutions, public or private, in the state of Paraná, southern Brazil.

Our results provide an understanding of the perception of the undergraduate students regarding their knowledge of SQ acquired during the course, and how they perceive themselves regarding their capability in applying what was learned. This way, we can contribute to the discussions of the curricula on computing courses, highlighting the knowledge axes that should be improved in order to improve the learning for future practitioners.

This paper is organized as follows: Section 2 provides essential concepts on the teaching of software quality and the SBC educational references; Section 3 presents the methodology of our study; Section 4 presents general and research question-based data analysis; Section 5 discusses the obtained results; and Section 6 presents the conclusion and directions for future work.

2. Background

To contextualize the research, we will present in this session the fundamental concepts of teaching software quality and the SBC's Software Quality Formation Reference.

2.1. Teaching of Software Quality

Specific subjects in the computing courses are intrinsically challenging for the teaching process. Students complain about excessive theoretical content, with a high-volume de-

mand and short deadlines. On the other side, teachers report issues regarding the short time to teach a large quantity of content and a lack of integration between other subjects. Added to the subject's challenges, the teachers also face diversity in the level of knowledge of students [Lemos et al. 2019].

The teaching of Software Engineering (SE) is present in most of the computing undergraduate courses in Brazil and is considered of great importance to developing qualified professionals. These professionals can contribute to the SQ, as well as contribute to solving traditional issues in the software industry [Gibbs 1994, Prikladnicki et al. 2009].

According to [Moser et al. 2021], computing undergraduate courses provide students with theoretical and practical knowledge to prepare them for professional exercise. However, certain processes are often not properly addressed during software teaching, as is the case with SQ, which is often overlooked by the students who do not see it as a priority. However, according to [Gupta and Goel 2019], once the students are exposed to the software industry and its demands, they must acquire this previously neglected knowledge.

SQ has always been present in the curricula of computing undergraduate courses, often with its concepts showing upon different subjects or in one specific subject in the course dedicated to this topic [Chren et al. 2022, Hilburn and Towhidnejad 2002]. Researchers mention that SQ should be approached since the beginning of the students' graduation to integrate the fundamental concepts as early as possible [Scatalon et al. 2017, Gomes et al. 2021].

SQ teaching should provide knowledge that allows the student to implement, measure, assess, and improve the software product throughout its entire life cycle. SQ is a vast knowledge area, its teaching should follow a structured approach, making the student experience close to the software industry reality [Suryn 2003].

2.2. SBC's Software Quality Formation Reference

SBC is a scientific society that brings together Brazilian students, teachers, researchers, and professionals in the area of Computing and Informatics from all over the country. One of its functions is to define national scientific and technological politics. Its goal is to elaborate and provide the reference for the curriculum building of the computing courses in Brazil, aiming at contributing to the formation of computing professionals with social responsibilities [Ribeiro et al. 2019, Zorzo et al. 2017].

At the end of 2016, SBC provided the document entitled "Formation Reference for Computing Undergraduate Courses" (RF-CC-2017), where they presented the references for the formation of undergraduate courses in the area of computing, such as Computer Science, Informatics, Computing Engineering, Information Systems, and other technical degrees. These references are a result of the contribution between the SBC associates from different universities and are material that is used as a consultation for the preparation of these courses' curricula and should be aligned to other national curricula guidelines [Zorzo et al. 2017].

For the Software Engineering course, the SBC (RF-CC-2017) specifies an expected knowledge related to SQ, demanding: "The student should be able to produce high-quality software that conforms to its requirements and satisfies user needs. The achievement of SQ involves models and techniques of the software product and process

quality".

From this specification, the SBC defines nine axes of knowledge for SQ, which are competencies and skills that are expected of the graduates [Zorzo et al. 2017]. These are:

- QA = Quality Attributes: to understand what are the software product quality attributes and their utility;
- PM = Software Product Metrics: To apply software product quality measurements tools;
- AT = Product Assessment Techniques: to apply product assessment techniques;
- RS = Review Techniques and Static Analysis: To apply techniques and procedures of validation and verification (static and dynamic);
- MP = Product Quality Models and Standards: To understand the software product quality models and standards;
- MC = Process Quality Models and Standards: To understand the software process quality models and standards;
- SC = Quality Standardization and Certification: to understand the quality standardization and certification processes;
- CM = Process Metrics: to apply process metrics;
- ST = Software Testing: to apply software testing.

3. Methodology

This section presents the methodology adopted for this study.

3.1. Aim and Research Questions

This paper consists of a characterization of the perception of SQ knowledge of undergraduate students in computing courses, such as Computer Science, Informatics, Production Engineering, Systems Analysis and Development, and Software Engineering. For this, the following research question was elaborated: "***What is the perception of undergraduate advanced computing students in relation to their acquired Software Quality skills?***".

3.2. Audience

The audience of our study is students enrolled in the last two years of undergraduate computing courses from universities in the state of Paraná. Our focus on the last two years of the course is because the first years students may not yet have attended disciplines on SQ.

3.3. Sampling

The data collection was done through the application of an online survey - following the guidelines by [Linåker et al. 2015] - developed on Google Forms. Participation in the survey was optional and anonymous.

The survey was made available for thirty days and sent by email to the undergraduate courses coordinators to be sent to their students. In the end, we collected ninety-nine answers to the survey.

3.4. Design

We decided to perform exploratory research because our aim is to understand the perception of the students regarding SQ concepts.

3.5. Instrument and Evaluation

The structure of the survey is divided into two main sections. The first is a characterization of the participant and its undergraduate course, in which we collect data on the name and duration of the course, the semester in which the student is enrolled, if the university is public or private, the city it is located, and if the respondent has any experience in software development. The universities were not identified because our focus is not to compare the teaching capacities among the institutions, but rather on their students' perception of knowledge.

The second section focuses on the students' perceptions regarding their knowledge and skills on SQ concepts, more specifically the competencies for the SQ knowledge defined by SBC, as we explained previously and discussed by [Bettin et al. 2022].

For each competency, there is a short description of its concepts, followed by three quantitative questions to be answered by a five-point Likert scale [Likert 1932], which aims to verify the level of agreement of the individual with a proposition that expresses something ranging from "totally disagree" to "totally agree" in relation to the SQ. Table 1 presents these questions.

Once we built the survey, it was validated by two professors who specialized in the SQ subject. We applied a pilot version of the survey on a sample of six invited students to refine the instrument and to eliminate any doubts that might arise during the application. The time required for students to answer the survey was an average of seven minutes, as initially foreseen in the pilot version.

3.6. Data Sharing

All data from our research is available at `https://doi.org/10.5281/zenodo.7429490`.

4. Data Analysis

As we discussed before, the participants of our survey answered six questions aiming at characterizing them, and twenty-seven questions about their skills regarding the nine SQ competencies as presented in Table 1. This subsection is divided into two parts, one for the characterization of the participant and one for illustrating the specific answers of each of the competencies.

4.1. Demographics

Table 2 presents the quantitative and percentage distribution of responses regarding the first six questions characterizing the participants.

The majority of participants are in Software Engineering (32.32%) and Computer Science (20.20%) courses. Most participants are from private institutions (60.61%) and 77.78% of respondents are from Maringá. About 23.23% of the total do not have any experience in software development, while almost half of the participants have about 1 to 3 years of experience.

4.2. Software Quality Competencies

In this section, we present the distribution of the scores given by the participants to each of the competencies. Figure 1 presents the bar graphs for each competency and their scores according to the scale we used. Following, we are going to discuss each graph.

Table 1. Questions regarding the knowledge axis competencies

Competency	Questions
AXIS 1 - QA	QA1 - My course contemplates or addresses concepts of software quality attributes.
	QA2 - I learned about concepts of software quality attributes.
	QA3 - I feel able to apply aspects of software quality attributes.
AXIS 2 - PM	PM1 - My course contemplates or addresses the software product metrics.
	PM2 - I learned about concepts of software product metrics.
	PM3 - I feel able to apply software product metrics.
AXIS 3 - AT	AT1 - My course contemplates or addresses product assessment techniques.
	AT2 - I learned about product assessment techniques.
	AT3 - I feel able to apply product assessment techniques.
AXIS 4 - RS	RS1 - My course contemplates or addresses review techniques and static analysis.
	RS2 - I learned about aspects of review techniques and static analysis.
	RS3 - I feel able to apply review techniques and static analysis.
AXIS 5 - MP	MP1 - My course contemplates or addresses product quality models and standards.
	MP2 - I learned about product quality models and standards.
	MP3 - I feel able to apply product quality models and standards.
AXIS 6 - MC	MC1 - My course contemplates or addresses process quality models and standards.
	MC2 - I learned about process quality models and standards.
	MC3 - I feel able to apply process quality models and standards.
AXIS 7 - SC	SC1 - My course contemplates or addresses quality standardization and certification.
	SC2 - I learned about aspects of quality standardization and certification.
	SC3 - I feel able to apply quality standardization and certification.
AXIS 8 - CM	CM1 - My course contemplates or addresses the process metrics.
	CM2 - I learned about process metrics.
	CM3 - I feel able to apply process metrics.
AXIS 9 - ST	ST1 - My course contemplates or addresses software testing.
	ST2 - I learned about software testing.
	ST3 - I feel able to apply software testing.

***QA** = Quality Attributes, **PM** = Software Product Metrics, **AT** = Product Assessment Techniques*
***RS** = Review Techniques and Static Analysis, **MP** = Product Quality Models and Standards*
***MC** = Process Quality Models and Standards, **SC** = Quality Standardization and Certification*
***CM** = Process Metrics, **ST** = Software Testing*

QA - Quality Attributes

Approximately 76.7% of the participants are positive that the content is seen during the undergraduate course. The majority, about 75.7%, considered having learned the content, but even so, a significant portion, 34.4% feels neutral about being able to apply SQ attributes. The results can be seen in graph (a) of Figure 1.

PM - Software Product Metrics

Table 2. Characterization of the participants of the survey

Participant Characteristic	Count	Percentage
Course		
Software Engineering	32	32.32%
Computer Science	20	20.20%
Analysis and System Development	18	18.18%
Informatics	10	10.10%
Industrial Engineering with an Emphasis in Software	8	8.08%
Information Systems	7	7.07%
Computer Engineering	3	3.03%
Major in Computing	1	1.01%
Institution		
Public	39	39.39%
Private	60	60.61%
City of the Institution		
Maringá	77	77.78%
Apucarana	12	12.12%
Campo Mourão	8	8.08%
Londrina	1	1.01%
Jandaia do Sul	1	1.01%
Software Development Experience		
No experience	23	23.23%
Less than 1 year of experience	20	20.20%
Between 1 and 3 years of experience	47	47.47%
Between 3 and 6 years of experience	9	9.09%
More than 6 years of experience	0	0.00%

We can visualize in graph (b) of Figure 1 that the majority of the participants, about 63.3%, are positive that they have seen the content of this competency, 57.5% agree they have learned it, however, about 65.6%, are neutral or negative about feeling able to apply software product metrics.

AT - Product Assessment Techniques

About 54.5% are positive that the content is seen. About 45.4% consider that they have learned it, but only 22.2% feel able to use product assessment techniques. These results can be seen in graph (c) of Figure 1.

RS - Review Techniques and Static Analysis

56.5% of the participants are positive about having seen the content. About 34.3% felt neutral regarding having learned, hence, just 26.2% feel able to apply review techniques and static analysis. These results can be seen in graph (d) of Figure 1.

MP - Product Quality Models and Standards

In graph (e) of Figure 1 about 72.1%, are positive about having seen the content. About 56.7% are positive about learning it. Even so, the majority of the answers regarding being able to apply the knowledge remained neutral.

MC - Process Quality Models and Standards

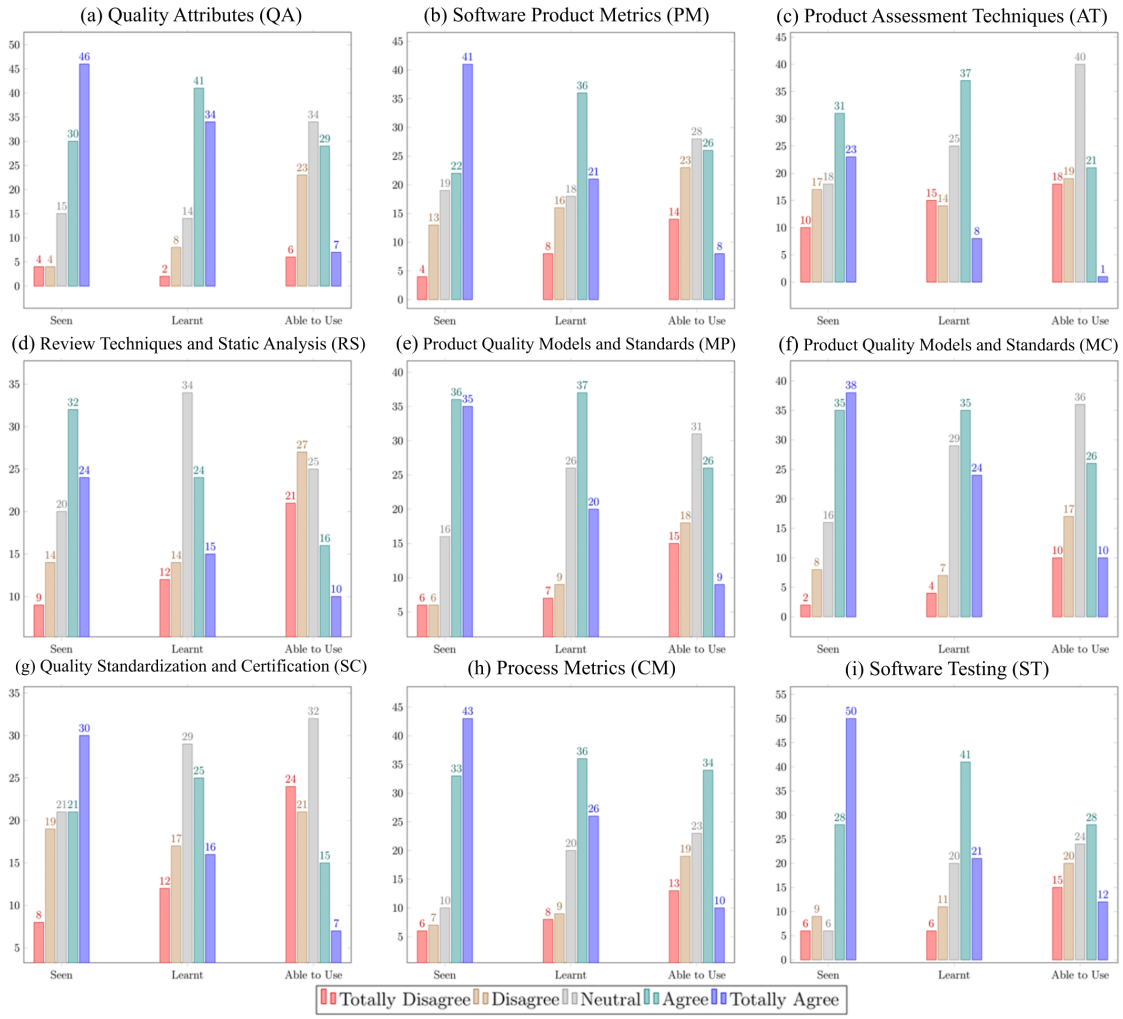


Figure 1. Scores given to the nine software quality competencies

Most participants stated that they had seen the content. Similarly, about 35.4% agreed, and 24.2% strongly agreed about learning it. However, when it comes to feeling able to use it, the response that most prevailed was neutral, about 36.4%, as can be seen in graph (f) of Figure 1.

SC - Quality Standardization and Certification

30.3% totally agreed and 21.2% both agreed and were neutral about having seen the content. Around 29.3%, remained neutral about having learned it, as shown in graph (g) of Figure 1. Almost half of the participants, about 45.4%, do not feel able to apply this knowledge, while about 32.3% remained neutral.

CM - Process Metrics

About 43.5% totally agreed that the content was seen. About 36.4% and 26.3% agreed and totally agreed, respectively, about having learned it. As seen in graph (h) of Figure 1, 34.3% agreed that they feel able to apply the knowledge, while 32% partially or totally disagreed.

ST - Software Testing

In graph (i) of Figure 1 about 78.8% are positive about having seen it. About 41.4% agreed to have learned the content. However, only 12.1% feel totally able to apply it, while about 20% disagree with the affirmation, and 24% remained neutral.

5. Discussion of Results

According to the data analysis, for each knowledge axis, few students disagreed about having seen the content during their courses.

The Software Testing axis draws our attention due to its level of agreement being higher than 50% of the participants regarding having seen its content. This may have happened due to its content being frequently approached during Software Engineering subjects.

About the data presented in Table 2 regarding the characterization of the participants, we can focus on the level of software development experience to further discuss our findings. We did not specify the type of experience asked, thus we can consider software development in different contexts, such as professional activities or even development from study or research.

We can check that about 23.2% do not have any experience with software development. While around 76.77% of the participants have some level of experience. Of the ninety-nine participants, none said having more than six years of experience in software development.

Aiming at better understanding how this difference in the experience level can influence the student perception, the survey answers were grouped according to the level of experience to be analyzed. Thus, we can calculate the average of the scores given by the students to the twenty-seven questions regarding the nine competencies.

Figure 2 presents three radar charts, one for each question that repeats along the survey. The axes of the chart represent the nine competencies and its scale follows the Likert scale used in our survey. The plotted data on the chart are the average score given by the students grouped by their level of experience, creating four overlapping polygons.

By comparing the three charts in Figure 2 we can see that the level of agreement decreases across the three questions. This can indicate that even though the students agree that the content was seen during their course, they disagree regarding having learned it and feeling able to apply this knowledge. This shows a problem that goes beyond the curriculum structure of the computing courses, it points out that these contents need to be understood and absorbed by the students.

In chart (b) we can see that the green line, representing the higher level of experience, often has higher average scores on competencies like PM, AT, RS, MP, and ST, and, on the other competencies, has lower scores than the other levels of experience. In the SC competence, for example, green and blue lines, representing the two highest levels of experience, is lower than the others, while the red one, which represents no experience, has the highest scores.

Similarly, in chart (c), the green line showed the highest scores on competencies like ST, MP, PM, and QA, while the yellow line, which represents students with less than one year of experience, has the lowest average values.

We can discuss that these high average scores from the more experienced stu-

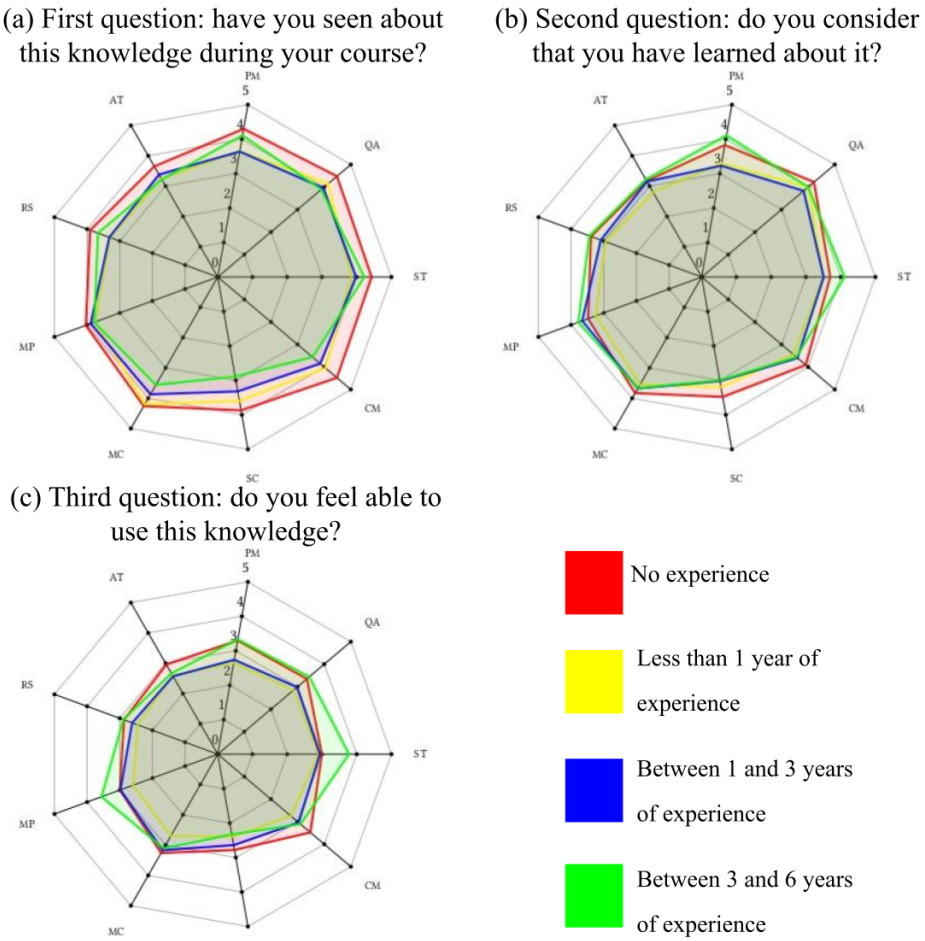


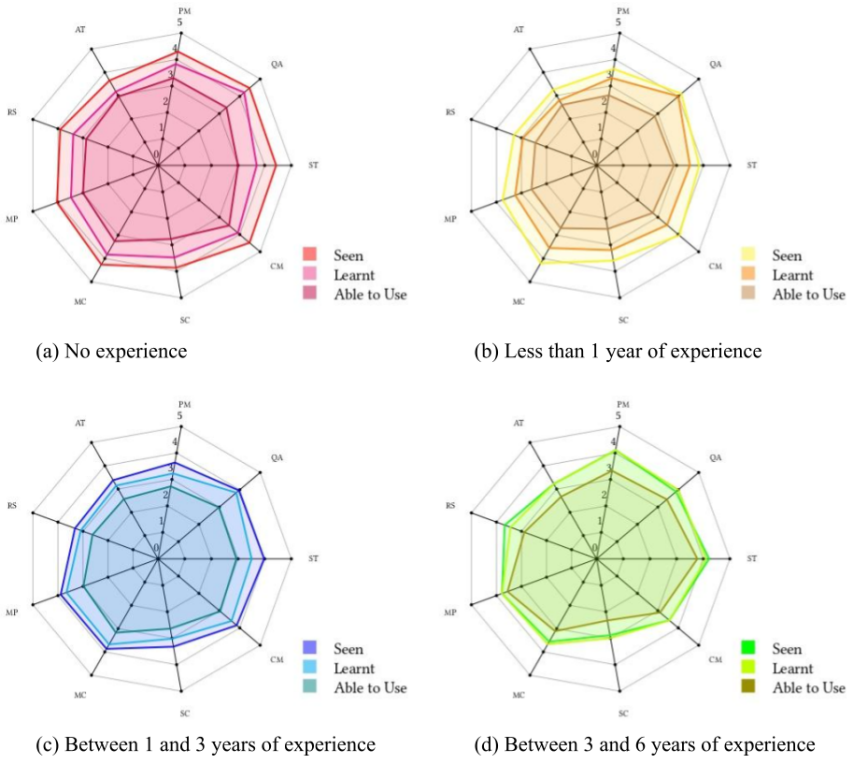
Figure 2. Average scores of the questions according to experience level

dents to competencies like PM, AT, RS, MP, and ST, could indicate that having experience with development practices can contribute to learning. While competencies like SC, where higher experience had lower scores than the rest, can indicate that the practice was different than what was learned or was not enough for them to feel able to apply their knowledge, something that has not yet happened to students with little to no experience.

Therefore, competencies like QA, PM, AT, RS, MP, and ST showed better performance among experienced students. These competencies can be enhanced when the student is practicing software development. While the SC competency showed opposite results, which can signify that what was taught is different or outdated from what is applied in the practice.

To visualize even better the difference of scores given to the three questions for each group of the level of experience, we built Figure 3, which presents four radar charts for the four categories of experience levels. In this case, the data plotted creates three polygons for the three asked questions for each of the nine competencies.

Comparing the three charts, we can notice how the lines representing the scores for each question behave. Figure 3 shows that the lines are more spaced in the first charts and this distance between them is reduced along the charts. It is possible to notice that the more experience in software development the student has, the more they can affirm they have learned what was seen during the course.



(a) No experience

(b) Less than 1 year of experience

(c) Between 1 and 3 years of experience

(d) Between 3 and 6 years of experience

Figure 3. Graphs for the averages for each of the respondent's experience levels

Also, in some competencies, such as MP and TS, experienced students can affirm that they feel able to apply the knowledge almost on the same level that they affirm that they have seen and learned about it during their course. Thus, the experience with software development can contribute to the student's perception regarding their level of understanding and capacity in applying the knowledge of the competencies in SQ.

Another point that we can discuss is regarding the high number of neutral answers to the survey. Some examples are shown in the previous section, like graph (c) of Figure 1, where around 40% of the participants remained neutral regarding knowing how to apply product assessment techniques, or in graph (g) of Figure 1, where both the answers to "learnt" and "able to use" were mostly neutral to the Quality Standardization and Certification competency.

Aiming at analyzing the answers on the extreme ends of the scale, we built Table 3, presenting the percentage of answers regarding each knowledge axis not considering the option "neutral". The answers "agree" and "totally agree" were grouped, as well as "disagree" and "totally disagree". Thus, it is possible to summarize the student's perceptions into positive or negative values.

Regarding having their content addressed during the course, the axis with the most positive scores were QA (93.08%), MC (87.95%), and MP (85.55%). While the ones with the most negative scores were SC (34.62%), AT (33.33%), and RS (29.11%).

Regarding having learned its content, the QA showed the better positive score (88.23%), followed by MC (84.29%), and, third, CM and ST tied (78.48%). On the other hand, the ones with the most negative score was SC (41.43%), RS (40%), and AT (39.18%).

Table 3. Participants answers for each axis not considering the "neutral" option

Axis	Seen		Learnt		Able to Use	
	Disagree	**Agree**	**Disagree**	**Agree**	**Disagree**	**Agree**
1 - QA	6.92%	93.08%	11.77%	88.23%	44.61%	55.38%
2 - PM	21.25%	78.75%	29.63%	70.37%	44.61%	47.88%
3 - AT	33.33%	66.67%	39.18%	60.82%	62.71%	37.29%
4 - RS	29.11%	70.89%	40.00%	60.00%	64.86%	35.14%
5 - MP	14.45%	85.55%	21.91%	78.09%	48.52%	51.48%
6 - MC	12.05%	87.95%	15.71%	84.29%	42.86%	57.14%
7 - SC	34.62%	65.38%	41.43%	58.57%	67.16%	32.84%
8 - CM	14.61%	85.39%	21.52%	78.48%	42.10%	57.90%
9 - ST	16.13%	83.87%	21.52%	78.48%	46.67%	53.33%

Lastly, regarding being able to apply the knowledge, the higher positive score was seen in CM (57.90%), followed by MC (57.14%), and QA (55.38%). While the axis with the lowest scores were SC (67.16%), RS (64.86%), and AT (62.71%).

The majority agreed that the content was seen during the course. However, according to [Richardson et al. 2011], the presence of a topic in a curriculum does not provide any guidance on how it should be taught. Therefore, we created three guidelines to help and improve the teaching of SQ, allowing students to understand the concepts and also develop the social and emotional skills - soft skills - that are desired in the software industry.

G.1: Integrate the content of the Software Quality competencies.

We noticed a difference between students with little experience in software development to those with some years of practice. We believe that aiming at relating what is taught in the classroom to what is applied during software development practices can help students who do not yet have experience also acquire adequate knowledge about these concepts. Also, all the contents of the SQ axis could be addressed together, since one complements the other.

G.2: Promote learning through practice based on case studies, problems, experiences, and real or fictional projects.

By developing activities that relate what is taught to what is practiced in the software industry, we hope that the student can develop leadership, good communication skills, and strategic thinking, among other skills desired.

G.3: Review the content of the curriculum of the computing undergraduate courses regarding SQ.

It is necessary to guarantee that what is offered in the curriculum of undergraduate courses is updated, following the orientations of the Ministry of Education and the guidelines of SBC.

5.1. Validity Evaluation

Some limitations and threats to validity regard the application of the survey. We noticed a low volume of participation by universities and students. This may have happened due to the limited time that the questionnaire was available. We understand that the number of participants can influence the results, however, we had to work with these values due to a limited time for the research. Hence, we suggest a replication with a larger sample of students. Moreover, there were a lot of neutral answers, which may indicate uncertainty in how to respond or fear of giving an opinion. We suggest the use of a scale with an even value of answers, which can contribute to reducing this neutrality and mitigating the threat.

6. Conclusion

Our paper presents the results acquired through an exploratory analysis of the perception of undergraduate students of computing courses in the universities of Paraná regarding the knowledge of SQ competencies.

Our results allowed us to understand that, even when the students are presented with the content of the SQ competencies, they often still say that did not learn it and do not feel able of applying it. This condition could be a reflection of the teaching method during the undergraduate course. Thus, the present study fills a gap discussed by [Bettin et al. 2022] and identifies the need to check whether the content defined in the course curriculum is actually being addressed.

Thinking about this issue, we created some guidelines to improve the teaching of SQ in universities, aiming to support a better understanding of these concepts so that the students can develop the skills and capabilities expected in industry practitioners.

With our paper, one is able to analyze how these competencies of SQ, defined by SBC, are being approached and how learning of them is being perceived by advanced undergraduate students. We hope to point to viable paths to improve the teaching and development of the students, and to provide capable professionals for the market.

For future work, we believe that the results could benefit from a larger sample of students, which could allow a deeper and more insightful analysis of the difference between the perception of understanding for each experience level. Also, we believe that this work can be expanded with more detailed questions regarding the students' perception of learning, elaborating even further for each subject on what are the topics that they feel positive about having learned, narrowing in order to identify the most critical lessons for the competencies.

We also suggest (i) the application of this research to universities of other states in Brazil; (ii) developing research about the method of teaching SQ beyond the student perception; and (iii) analyzing the graduated student regarding their skills acquired during the course.

Acknowledgements

Edson OliveiraJr thanks CNPq/Brazil (Grant #311503/2022-5). Authors thank CAPES/PROAP/Brazil, support Grant #88881.753469/2022-01.

References

Bettin, G., Herculani, J., Shigenaga, M., Leal, G., Balancieri, R., OliveiraJr, E., Colanzi, T., and Amaral, A. (2022). Teaching of software quality in public higher-level universities of paraná-brasil: an exploratory study. In *Proceedings of the 30th Workshop on Computing Education*, pages 286–297, Porto Alegre, RS, Brasil. SBC. in Portuguese.

Bourque, P., Fairley, R. E., and Society, I. C. (2014). *Guide to the Software Engineering Body of Knowledge (SWEBOK(R)): Version 3.0*. IEEE Computer Society Press, Washington, DC, USA, 3rd edition.

Chren, S., Macák, M., Rossi, B., and Buhnova, B. (2022). Evaluating code improvements in software quality course projects. In *Proceedings of the International Conference on Evaluation and Assessment in Software Engineering 2022*, EASE '22, page 160–169, New York, NY, USA. Association for Computing Machinery.

Dingsøyr, T., Jaccheri, M. L., and Wang, A. I. (2000). Teaching software process improvement through a case study. *Computer Applications in Engineering Education*, 8(3-4):229–234.

Garousi, V., Giray, G., Tuzun, E., Catal, C., and Felderer, M. (2019). Closing the gap between software engineering education and industrial needs. *IEEE software*, 37(2):68–77.

Gibbs, W. W. (1994). Software's chronic crisis. *Scientific american*, 271(3):86–95.

Gillies, A. (2011). *Software quality: theory and management*. Lulu. com.

Gomes, P. H., Garcia, R. E., Eler, D. M., Correia, R. C., and Junior, C. O. (2021). Software quality as a subsidy for teaching programming. In *2021 IEEE Frontiers in Education Conference (FIE)*, pages 1–9.

Gupta, R. and Goel, S. (2019). Infusing software quality concepts in computer science engineering courses. *2019 Twelfth International Conference on Contemporary Computing (IC3)*, pages 1–7.

Hilburn, Y. and Towhidnejad, M. (2002). Software quality across the curriculum. In *32nd Annual Frontiers in Education*, volume 3, pages S1G–S1G. IEEE.

Jenkins, M. (2007). Experience in teaching software quality management at the graduate level. In *2007 Annual Conference & Exposition*, pages 12–711.

Jones, C. and Bonsignour, O. (2011). *The economics of software quality*. Addison-Wesley Professional.

Kappelman, L., Jones, M. C., Johnson, V., McLean, E. R., and Boonme, K. (2016). Skills for success at different stages of an it professional's career. *Commun. ACM*, 59(8):64–70.

Kokol, P. (2022). Software quality: How much does it matter? *Electronics*, 11(16).

Laporte, C. Y., April, A., and Bencherif, K. (2007). Teaching software quality assurance in an undergraduate software engineering program. *Software Quality Professional*, 9(3):4.

Lemos, W., Cunha, J., and Saraiva, J. (2019). Teaching software engineering in an information systems course: An analysis of problems and solutions in the perspective of teachers and students. In *Proceedings of the 27th Workshop on Computing Education*, pages 305–318, Porto Alegre, RS, Brasil. SBC. in Portuguese.

Likert, R. (1932). A technique for the measurement of attitudes. *Archives of psychology*.

Linåker, J., Sulaman, S., Höst, M., and de Mello, R. (2015). Guidelines for conducting surveys in software engineering. Technical Report 1.1, Lund University, Sweden.

Moser, G., Vallon, R., Bernhart, M., and Grechenig, T. (2021). Teaching software quality assurance with gamification and continuous feedback techniques. In *2021 IEEE Global Engineering Education Conference (EDUCON)*, pages 505–509.

Page, T. (2008). Ensuring software quality in engineering environments. *i-Manager's Journal on Software Engineering*, 2(4):1.

Prikladnicki, R., Albuquerque, A. B., von Wangenheim, C. G., and Cabral, R. (2009). Teaching software engineering: challenges, teaching strategies and lessons learned. In *Software Engineering Education Forum*, pages 1–8. in Portuguese.

Ribeiro, K., Azevedo, J., Maciel, C., and Bim, S. (2019). A gender analysis based on data from brazilian computer society. In *Proceedings of the 13th Women in Information Technology*, pages 159–163, Porto Alegre, RS, Brasil. SBC. in Portuguese.

Richardson, I., Reid, L., Seidman, S. B., Pattinson, B., and Delaney, Y. (2011). Educating software engineers of the future: Software quality research through problem-based learning. In *2011 24th IEEE-CS Conference on Software Engineering Education and Training (CSEE&T)*, pages 91–100. IEEE.

Scatalon, L. P., Barbosa, E. F., and Garcia, R. E. (2017). Challenges to integrate software testing into introductory programming courses. In *2017 IEEE Frontiers in Education Conference (FIE)*, pages 1–9.

Suali, A., Fauzi, S., Nasir, M., Sobri, W., and Raharjana, I. (2019). Software quality measurement in software engineering project: A systematic literature review. *Journal of Theoretical and Applied Information Technology*, 97(3):918–929.

Suryn, W. (2003). Thoughts on teaching software quality engineering. In *Proceedings of 8th Annual INSPIRE Conference*.

Tian, J. (2005). *Software quality engineering: testing, quality assurance, and quantifiable improvement*. John Wiley & Sons.

Zorzo, A., Nunes, D., Matos, E., Steinmacher, I., Leite, J., Araujo, R., Correia, R., and Martins, S. (2017). *References for Formation of Computing Undergraduate Courses*. Brazilian Computing Society (SBC). in Portuguese.