Industrial Involvement in Information System Education: Lessons Learned from a Software Quality Course

Stanislav Chren
Faculty of Informatics/Masaryk University
Brno, Czech Republic *chren@mail.muni.cz*

Bruno Rossi
Faculty of Informatics/Masaryk University
Brno, Czech Republic *brossi@mail.muni.cz*

Barbora Buhnova
Faculty of Informatics/Masaryk University
Brno, Czech Republic *buhnova@fi.mun.cz*

Mouzhi Ge
Faculty of Informatics/Masaryk University
Brno, Czech Republic *mouzhi.ge@muni.cz*

Tomas Pitner
Faculty of Informatics/Masaryk University
Brno, Czech Republic *tomp@fi.muni.cz*

Abstract

As Information System (IS) development is closely related to industry and real-world applications, industrial involvement is a critical element in IS education. This paper studies one typical IS course - a Software Quality course, and reflects our experience with involving a mix of industrial experts in building a practical IS course that would increase students' competences in critical thinking about the consequences of the design and quality engineering decisions that they are making during software development. In the course design, the industrial experts are involved in lecturing, hands-on-exercise seminars and final student evaluation. We find that students are showing active course participation with our designed industrial involvement. Furthermore, we summarize lessons learned from the industry involvement, as well as the reflections on the value perceived by the industrial experts involved in the IS education.

Keywords: Industrial involvement, software quality course, information system education, teaching information systems

1. Introduction

Information System (IS) education is developed under the discipline of computing [20]. It has been explicitly recognized that computing knowledge in IS education is mainly developed by computer science, software engineering, and computer engineering [20], where software engineering plays a foundational role in IS development. Therefore, software engineering, as well as its sub-disciplines such as software quality or software architecture, has been considered as a set of critical courses in IS education. However, with the rapid development of the software techniques, university education may lag behind the industry in software development. Thus, Topi [19] proposes to build an industry advisory board for IS education, which serves as a communication channel between software companies and IS educational resources such as university courses.

This idea is widely implemented, e.g. *Industrial Board* at the Faculty of Information Technology, Brno University of Technology, or *Association of Industrial Partners* at the Faculty of Informatics, Masaryk University, Czech Republic. Currently, the Study Program Boards at Czech Universities engage representatives from industry to form link between (some of) their study programs with industrial practice.

Effective learning in IS can be represented by the value of the course content to the students. The value can be further explained by the knowledge obtained during the university education or by the practical usage after the education. Thus, in order to teach software engineering courses in IS, we consider that the perception of the course value to university students can be significantly increased when a balanced mix of both industrial and academic views on the matter is included in the course. One paramount challenge for software engineering courses is the need to have practical involvement of students [13], [16]. Without such involvement, it can be difficult for students to grasp all the implications and complexities of real-world projects. Some experts even propose to postpone frontal theoretical lectures, to support problem-solving learning, from which relevant theory can be explained afterwards, like in the *Extreme Apprenticeship* approach [11].

The intuition of achieving the effective learning in IS education is to drive the lectures by different viewpoints from various industrial experts, in combination with representatives from academia, and trigger discussion on how diverse the topic can be in practice. The goal of such approach is to engage students in critical thinking about software quality from many different perspectives, and hence allow them to predict the consequences of their design decisions. The development of such critical thinking is formulated by our industrial partners as the key skill that distinguishes senior engineers from junior developers. Therefore, within the software engineering courses, it is valuable to blend industrial partners to promote the learning of this skill.

In this paper, we select an essential software engineering course, *Software Quality*, to illustrate the industrial involvement in IS education. We report the learned lessons and experience with maximising such viewpoint balance in developing and teaching a university course on Software Quality, because during discussions with our industrial partners we became to understand that this domain is being governed by many different perspectives that should all be understood and balanced during software development. This is confirmed also by the *Software Engineering Body of Knowledge*, which defines software quality as a pervasive aspect in software engineering, covering also many other areas such as testing, and maintenance [1].

To further enhance the practical skillset of the students, the lectures were accompanied with hands-on-exercise seminars (two hours per week during the whole semester, i.e. 14 weeks), which were offered in two programming languages (Java and C#) and two flavours (academic and industrial—where the industrial version of the seminars was driven and taught solely by industrial experts, while the academic version was governed by our internal team that included industrial experts only during some lessons).

The remainder of this paper is structured as follows. After related work review in Section 2, Section 3 describes the course content design and our strategy to industrial involvement. The reflections on the course from the industrial perspective are summarised in Section 4, and the conclusion, limitation and future works are presented in Section 5

2. Related Work

There are a number of studies that describe experiences with teaching practical software engineering courses. Very often these courses teach general software engineering practices and principles. Some of these studies recognize the importance of software quality [10], [15]. However, software quality is not their primary focus, i.e. they teach methods which lead to the improvement of software quality, such as team collaboration [18] or testing [7] but they lack comprehensive overview of the complex in the software

quality domain.

There are also several papers devoted to specialised software quality courses [12], [7]. Although these courses are practical, they mostly do not rely on the collaboration with industrial partners. Gotel et al. [5] incorporated the work on real open source projects into the practical parts of the course, but the direct feedback from the industry to the course curriculum is not considered. The industrial involvement in teaching software engineering brings benefits to all interested parties and has been integrated into many courses [8], [14]. There are various approaches on how to involve industry in terms of courses' content. There are courses which involve projects that simulate the industrial environment and assignments but are not done in close cooperation with an industrial partner [6]. Then there are courses, which enable students to work on projects and assignments provided by companies [14].

However, there is a lack of studies that would detail courses in which the industry is directly involved in the course organisation. From the software quality domain, Jaccheri [9] describes a software quality course where the local companies participated in the lecturing process. While this work is in some aspects similar to ours, in our case, the content and design of the course are more industry-driven. That means, our course is fully intertwined with the industry. For example, the selection of lecture topics and content of the practical seminar sessions are designed jointly by the practitioners and university professors. Some practical seminars (the whole semester of 2 hours per week, 14 weeks in row) are conducted and led entirely by the industrial partners. Table 1 summarizes the types of industry involvement according to related works.

Table 1. Teaching methods with industry in software engineering

Reference	Teaching methods
[18]	Team collaboration
[7]	Software testing driven
[12]	Industrial context illustration
[5]	Real-world open source project
[14]	Assignments provided by companies
[9]	Companies participate in the lectures
[8]	Industry as course involvers
This paper	Topic design involving industry and practical seminar by industry

3. Software Quality Course Design

One of the key challenges in defining the course was to select the sub-topics in the IS domain that could be relevant both from the theoretical (lectures) and practical (hands-on-exercise seminar groups) point of view. As this is a software quality course, we take into consideration that some existing courses in the study curricula already introduce the students to several concepts in the software quality area. For example, while no specific software testing course exists at our faculty, most of the concepts are spread across several courses. We also took into consideration the experiences gathered from designing and managing the Software Engineering course at the Faculty of Informatics of Masaryk University, in which recently we looked at the quality over years of UML models produced by students [2]. In the initial run of the Software Quality course, we defined the following areas:

- Describing the concept of Software Quality and the different attributes. This was an introduction for the course, so that the main concepts do not need to be repeated;
- Clean Code, SOLID principles, bad code smells and code refactoring. This was a relevant part of the hands-on-exercise seminar groups to introduce students to the importance of the principles and application of refactoring;

- The role of software architecture. This was a topic promoted strongly by the industrial partners, willing to give their point of view on software architecture and its relevance for software quality;

- Principles of testing. We focused on giving all students the basis to follow the testing part of the course, by providing them with definitions and principles for the application of the testing process. We also focused on the suggestions relevant to testing object oriented software;

- Automated testing and testability. Concepts such as continuous integration and automated testing were considered important to give the students instruments that they could use during the seminar sessions;

- Focus on quality attributes and their conflicts. This lecture focused on the interplay of performance, scalability, reliability, testability, maintainability quality concerns and tactics to address them;

- Testing in Agile development. This lecture focused on giving an overview of testing methodologies within agile development processes;

- Performance engineering and performance testing. Based on the expertise of industrial experts, the lecture focused on giving details of the implications of performance testing and different frameworks that can be used to monitor and improve such quality characteristic;

- Challenges of quality management in cloud applications. Based on the current relevance of cloud applications, the lecture focused on looking at quality attributes for cloud-based software architectures;

- The software quality management process. This lecture was giving the final point on all previous lectures by reviewing different process quality management standards (e.g. ISO/IEC 25000:2014, ISO/IEC 15504 (SPICE), CMMI, various other maturity models);

While these topics do not cover all information relevant to software quality, they are those that were considered the most relevant by the team of both industrial and academic experts, after considering constraints on topics covered in other courses.

The practical sessions were given within hands-on-exercise seminars focused on Java and C#. Java seminars were held by an academic team, while C seminars, by industry experts to different students divided in groups. The two seminar types—while maintaining relation with the course's main topics—were different in terms of syllabus, with Java seminars mimicking more the content of the lectures, while the C# seminars added a few more topics to take advantage of the expert knowledge present in the team that taught it, mainly on product quality and relevant processes.

The Java seminars instead covered more tools to support the topics (Maven, Git, Junit), Clean Code, SOLID principles, refactoring, TDD with JUnit and Mockito, test plans, issues and Selenium, performance testing and profiling, static code analysis, code reviews, and continuous integration. Furthermore, the Java and C# groups differ in the format of the seminar sessions. The C# groups focus more on the practical demonstrations and hands-on tutorials from industrial experts. In total, there are 12 seminar sessions, each lasting 100 minutes. The Java groups put more emphasis towards independent work of students working on practical assignments in which they can exercise the seminar topics. To be able to finish also more complex tasks, the Java seminar sessions last 200 minutes and there are only 6 sessions during the semester.

Although the attendance at the seminar sessions is required, the attendance at the lectures is not mandatory. In order to motivate the students to attend the lectures, the lecturers were encouraged to engage students in active participation and were allowed to

distribute bonus points among the present students for their participation in the discussions or for completing exercises at the lecture.

On top of that, students are required to complete three large homework assignments consisting of several mandatory and optional tasks. The first assignment is focusing on refactoring, in which students are asked to refactor a legacy code of a simple game. Moreover, students are asked to follow the Clean code and SOLID principles. The second assignment focuses on testing and students need to implement specified tests using Mockito library and various Junit extensions. The third assignment is focusing on the static code analysis with the Checkstyle tool. The goal is to implement a custom check that could detect specified code smells.

At the end of the course, we organised a final colloquium event followed by the written test. During the colloquium, the students were divided into groups, where each group was led by an industrial expert who was responsible for one of the lectures. The groups were assigned with a software quality related topic for discussion. The students were discussing the selected topic, moderated by the industrial partner. The outcome of the discussion was then presented to other groups – where each student from the discussion group was asked for one insight that was most surprising for them. In the next year we consider adding voting for the best insight by the students.

The final grade depended on the total point score accumulated during the course run. The points were awarded for the assignments solved at the seminar sessions, for the activity during the lectures and for the final written test. Currently, the course is offered to the maximum of 70 students. In the future runs, we plan to increase the capacity up to 100 students. Whole course is taught in English.

Over the past decade, Brno, where Masaryk University is located, has grown into a technological hub with very high presence of both established technological leaders (Honeywell, Siemens, IBM, Red Hat, and others) and successful start-ups (Y Soft, Kentico Software, Flowmon Networks, and many more), with very strong link to local universities (with over 85,000 students overall in the city of 400,000). Thanks to these conditions, Masaryk University has a number of established platforms to underline the industrial cooperation, where the most relevant platform for the Faculty of Informatics is the Association of Industrial Partners (AIP), which gathers 32 selected companies with the highest potential of mutually beneficial intensive cooperation.

These were the companies, from which a working group has been established to contribute to the design of the Software Quality course discussed in this paper. Within the first phase, the goals of the course have been established, which were then reflected by our internal team in the design of the course syllabus. The industrial experts were then invited to prepare and give selected lectures and seminars, with the aim to establish 50:50 balance among academia and industry within both lectures and seminars, which has been achieved. Overall, experts from five companies were involved. The brief profiles of the companies is shown in Table 2.

4. Lessons Learned and Reflections from Industry Involvement

At the end of the software quality course, we submitted a questionnaire to the industrial partners. The main goal was to evaluate the industrial involvement in teaching software quality from the industrial practitioners who provide both frontal lectures and practical seminars. The industrial involvement is studied from six aspects, which are the software quality definition, software quality attributes, involvement motivation, teaching reflection, student's required skills, student's missing skills. Each of the aspects is formulated by a question and thus there is a total of six questions. The selection of the six aspects is based the agreement from the five faculty members who are or were teaching the software quality course. The study is conducted in the form of a semi-structured interview to the course industry participants.

Table 2. Industrial partner profiles

Name	Employees	Field	Course topic	Participation
Company 1	250+	HW and SW for 2D/3D printing	SW architecture, Clean code & SOLID principles, Refactoring	Lectures, seminars
Company 2	300+	Network management and monitoring	Automated testing	Lecture
Company 3	9000+	Industrial technologies, Energy, Healthcare	Quality and testing in agile	Lecture
Company 4	700+	Operating system, Enterprise SW	Performance testing	Lecture, seminar
Company 5	4000+	Aerospace systems, CPS	Static code analysis	Lecture

As Question 2, 5, and 6 can be detailed to concrete constructs, a quantitative survey is also combined with the interview. Since we suppose that the interview results from industry can reveal more constructive conclusion than students, the interviewee are the industrial experts from the five companies who are involved in the lectures or practical seminars of the software quality course. There are seven interviewees in total.

- Q1. What is the definition of software quality for each industrial participant? *Rationale: there is no univocal definition of software quality, the goal of this question was to get a view about what software quality means for each of the participants, to understand differences about the assumptions in the answers to the other questions.*

- Q2. What are the most important software quality attributes according to the industrial participants? *Rationale: as there is no unique definition of software quality, each industrial participant can have a different consideration about the most important attributes that need to be considered when dealing with software quality.*

- Q3. What are the motivations for industrial participants to take part in the software quality course? *Rationale: we wanted to know about the reasons industrial participants take into account to take part to the teaching process of the software quality course.*

- Q4. What are the lessons learned for industrial participants from taking part in the software quality course? *Rationale: understanding what are the main takeaways by the industrial participants. What are key learning experiences that industrial participants made by taking part to the course.*

- Q5. What are the skills that industrial participants consider relevant for students in the area of software quality? *Rationale: this question was essential to understand what are considered as important skills to be exercised in a software quality course.*

- Q6. What are the skills that the students lack in the area of software quality? *Rationale: similar to the previous question, also this one evaluates the perception of industrial partners about the skills needed. In this case, we are interested in knowing which skills the participants consider as lacking from the side of the students.*

Overall, we got response from all five companies with seven respondents involved in either lecturing, laboratories and/or material preparation. The answers are summarized as follows.

Q1. What is software quality from your point of view? We collected several points of view about question one from the industrial participants. For some of them, quality is all about the people, some others refer to the ISO/IEC 25010 standard about software quality to use as a reference framework for all that is needed in terms of software quality (either product or process). Others take a more *"business needs"* point of view, in the sense that quality should be focused on the final customers and value provided. They emphasize the fact that the *"customer gets the best experience while using the software"*, *"...(final goal should be to build) maintainable software satisfying business needs"*, or *"(develop) a product that does it well in the eyes of all stakeholders on various quality scales"*. However, many of the industrial participants focus more at the product quality level, underlining the importance of *"high maintainability and low number of bugs"* and *"bug free and maintainable software (satisfying business needs)"*. It can be seen that for IS and Software Engineering education, the practitioners may consider the courses from different views, some of which can be quite different from the academic view. For example, the ISO/IEC 25010 standard about software quality can be gradually integrated into IS courses, giving a point of view that can be supported by industrial practices.

Q2. What attributes of software quality are the most important to you? It seems to be an agreement that *reliability* is the most important quality for the participants over *security* and *performance* that come in the second place. *Maintainability* and *scalability* come next, while *testability* seems less important (Fig. 1). Respondents also indicate other attributes as potentially relevant, such as *profitability*, *portability*, *user experience*. The results reflect both the structure of the course and the seminar groups. In this course, security was not covered as it was already discussed in other courses, so there would have been too much overlap with their contents. Considering the number of responses, the overall importance of the individual quality attributes was well balanced. This confirms that we were able to involve experts with different preferences of quality perspectives, which was one of the keys for involving these practitioners. It indicates that in IS education, different industrial experts may have different focus and practical concerns. This can be because of their working environments and teaching preferences. From our experience, it is valuable to involve industrial experts with a range of different expertise to offer a more comprehensive teaching experience to students.

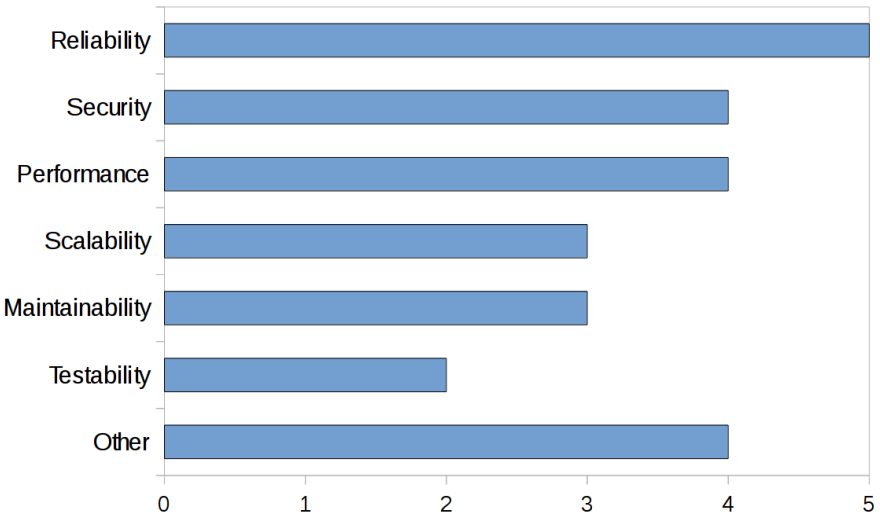

Figure 1. Attributes of software quality that are the most important

Q3. What motivated you to participate in the course? The main answer from industrial participants was to *share practical experience*, followed by *getting in touch with students to understand their interests. Building personal experience in teaching and promotion of the company* were less relevant. Even less relevant were to *build the perception in the students about what it means to cover a specific role in the quality process*, and to *share working opportunities with students*. We observe that we have

bilateral benefits for the industrial involvement. On the one hand, the practitioners would like to share their experience with students, on the other hand, the practitioners may also involve the students in their companies. For IS education, it can be seen that from the industrial perspective, involving in university education is highly motivating. The industrial participants can play an active role in the IS education, instead of becoming just guest lecturers. This can create a sort of feedback loop, in which their involvement gives rise to new topics more in synchronization with industrial needs.

Q4. Are there any lessons learned from your involvement in the course? Industrial participants reported about many aspects that were learned by taking part to the course. *"Talking more about practical examples that are the most interesting for students"*, or *"...to always have a backup plan"*, plus to note *"...how many views we have on the SOLID and some programming techniques"* were some of the main points. Others reported that *"software development uses many different tools, it is not possible to show all of them during one course"* and that *"practical, experience-based examples and simple ways to try something are always better than a theory"*. The results indicate that it can be easier to use practical examples to approach the students in IS education, though, of course, theory must still be present to support the teaching outcome from the course. The main point is that the practical examples can be obtained from industries, to make them as realistic as possible in the context of the future working environment of students. In this sense, one comment was that examples seen during the course are too limited and typical represent *"green field"* projects, while students face different challenges in industry by having to deal with large and legacy systems that need to be updated and maintained. For some IS/Software Engineering course such as IT management, it mostly starts from theories and then applications, and some management framework might not even be used in practice. We propose that practical examples from industrial experts can be introduced at the beginning of the courses to motivate the students and allow students to see the real-world use cases. Afterwards, the students can proactively think about the application scenario of the theoretical frameworks – seeing them better located in the practical context. Of course, there is a long debate whether practice should follow theory, vice-versa, or alternative ways of involving students [4, 17]. The industrial participants seem to indicate that for a Software Quality course, a more practical focus is the key to give students a better learning outcome. Furthermore, regarding what benefits the collaboration brought, many respondents reported that it *"increased the awareness of students about the company"*. This aspect was not considered as one of the main goals by the external industrial collaborators, but it was one result that was appreciated by the management of the companies involved.

Q5. What skills/knowledge are essential for the students to have in terms of software quality? Industrial participants voted on a scale from *0 (not relevant)* to *5 (highly relevant)* based on a list of 16 skills taken from the course's content. Ordered by the median of the answered values (Table 3), we can report that *Continuous Integration / Delivery* was considered the most essential skill for students, followed by *refactoring* and *automated testing*. We consider the answer by industrial participants was due to the importance of these aspects in nowadays software development context, as they are a main part of the DevOps movement for the automation of software development and delivery [3], a key aspect of modern software development practices.

The less relevant aspects were the *"software quality management process"*, *"risk-based testing"*, *"conflicts between quality attributes"*, *"static code analysis"*, *"cloud quality management"*. While the less interest on some of these aspects can be justified due to the more management point of view, we found surprising the low position of static code analysis, usually an activity that is quite relevant in the teaching part of the course, but might be less relevant for the industrial participants.

Due the importance of this question, we have further plotted the variation of the essential skills in software quality in Figure 2. In this specific course, the required essential skills from industrial view can vary in a large scale (variation is from 1 to 5). In a university education, academics may plan to design a comprehensive syllabus and give the students a better overview of the taught topics. For industry, it is mostly deliverable-

driven, thus, essential skills from industrial view are usually prioritized. In the IS education, we may take into account that what skills industry considers important can influence the focus of the industrial lectures. It is however also important to let the students to have wide view on certain IS topic and deep understanding on some aspects of this IS topic.

Table 3. What skills/knowledge are essential for the students to have in terms of software quality?

Rank	Skill	Median	Mean
1	Continuous integration/delivery	5.00	4.00
2	Refactoring	4.00	3.00
3	Automated testing	3.00	3.40
4	Clean code / SOLID / GRASP principles	2.00	2.67
5	Software measurement and metrics	2.00	2.20
6	Code reviews	2.00	2.20
7	Test case specification	2.00	2.17
8	Functional testing	2.00	2.17
9	Performance testing	2.00	2.00
10	Acceptance testing	2.00	2.00
11	KISS / YAGNI principles	1.50	2.17
12	Software quality management process	1.00	1.67
13	Risk-based testing	1.00	1.40
14	Conflicts between quality attributes	1.00	1.40
15	Static code analysis	1.00	1.00
16	Cloud quality management	1.00	0.80

Q6. Are there any skills/knowledge that current (post)graduate students lack in terms of software quality? Industrial participants voted on a scale from *0 (not relevant)* to *5 (highly relevant)* on the same list of 16 skills taken from the course's content. This time they were asked to evaluate the lack of skills of students according to their industrial experience. Ordered by the median of the answered values (Table 4), *automated testing, clean code / SOLID / GRASP principles, continuous integration / delivery* seem to be the skills that are mostly missing in the area of software quality.

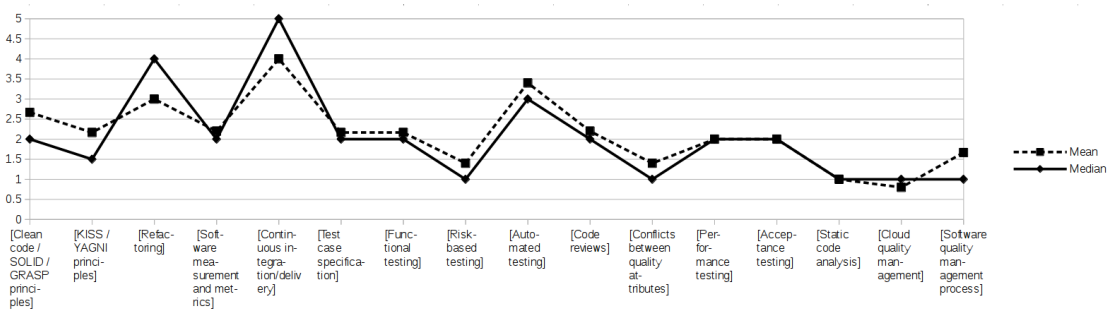

Figure 2. Variation of important attributes in software quality

The results of Q6 are plotted in Figure 3. We can observe that the variation from Q6 is not as big as results from Q5. Together with result in Figure 3 and 3, it can be interpreted that the industry clearly knows what they want but they may not deeply understand the students and their intended education. It reflects that the industry may consider that it is always good for the students to learn something. However, when the industry considers the essential skills from the students, they will have a clear priority, which depends on the concrete IS topics. In IS education, when we intend to involve the industrial opinions in the curriculum design, it is more effective to survey the direct requirements from industry rather not what is missing the in the current IS education.

Table 4. Are there any skills/knowledge that current post graduate students lack in the software quality?

Rank	Skill	Median	Mean
1	Automated testing	3.00	3.25
2	Clean code / SOLID / GRASP principles	3.00	2.67
3	Continuous integration/delivery	2.50	3.00
4	Test case specification	2.00	2.33
5	Performance testing	2.00	2.25
6	Code reviews	2.00	1.67
7	Refactoring	2.00	1.33
8	Software quality management process	1.00	1.75
9	Acceptance testing	1.00	1.67
10	KISS / YAGNI principles	1.00	1.00
11	Functional testing	1.00	1.00
12	Risk-based testing	1.00	1.00
13	Static code analysis	1.00	1.00
14	Software measurement and metrics	0.50	1.50
15	Conflicts between quality attributes	0.50	1.25
16	Cloud quality management	0.00	0.67

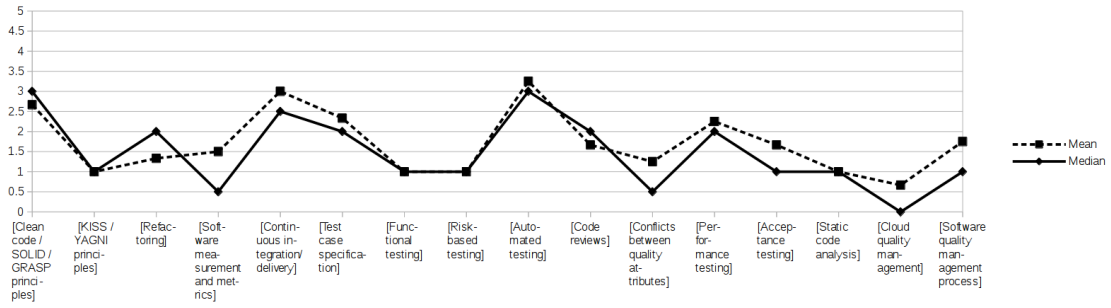

Figure 3. Variation of lack of skills in software quality

Based on the questionnaire and qualitative interpretations, we found that the practitioners may consider topics the IS courses from various perspectives. It is important to let the students understand the different industrial thinking. Since different industrial experts may have different practical foci, it is valuable to involve more industrial experts from different companies into the IS courses. From the industrial side, they are highly motivated to get involved in the university courses. The IS education may offer more opportunities for industrial involvement. Furthermore, we propose practical example is a good entry point for the students to learn the theoretical knowledge. Also, the real-world use cases and applications can provoke the student's learning interests to IS courses. Finally, when we design the IS courses, it is more effective to ask the requirements from industry, which can better catch what is important and needed from industry.

4.1. Threats to Validity

There are several threats to validity we need to report for this research article. The first one is about external validity and generalization of the results. We cannot claim results hold for any industrial context, but they can be considered as representative of the local context. The whole sample of industrial participants was based on a set of most representative companies in the area with the most interested employees in concepts such as software quality. Sample selection might be considered as a form of selection bias, as

the involved participants are highly interested in the course topics, and not randomly selected within companies. However, for the goals of the survey, such selection can be considered as irrelevant, as the scope was the evaluation of the specificity of a software quality course, so the selection strategy was based on getting industry experts highly experienced in the topic.

About internal validity, for practical reasons the survey was submitted at the end of the software quality course. While some questions could only be answered at the end, the answer to others (like Q1, about the definition of software quality from each participant) might have been influenced by the respondents' participation to the course. As such, all answers to questions have to be considered as ex-post answers after one iteration of the course.

5. Conclusions

In this paper, we have studied a typical IS course in software engineering: software quality. This paper has presented our reflections, interpretations and lesson learned on defining, preparing and teaching a Software Quality course in very close cooperation with industrial partners, who were involved not only in lecturing (which is a common strategy), but also in definition of the course syllabus, student's involvement (in active participation in lectures and final colloquium event), hands-on-exercise seminars and final student evaluation. These findings are expected to be not only useful in other IS courses, but also be a valuable inspiration for other academic teams that would like to take advantage of the practical software engineering knowledge available within local industry. Furthermore, since in our study that many anonymous students refer to this course as the best course they ever attended, we consider this course evaluation result as a success from the course design, which can also validate the implications of the findings in this paper

One of the limitations in this work is the representativeness of the samples in industry. Although we have considered the diversity of the selection of different companies, the five companies involved in the study are limited to a regional sample. Further, the interviewees from the companies may not represent the whole opinion from their companies. Thus, the replies to the questionnaires may miss addressing specific aspects from comprehensive feedback. As future work, we plan to integrate the industrial involvement into more IS courses such as IS Management or IT Services Management. Based on the pilot survey study in this paper, we intend to propose a systematic method of how to involve industrial partners in different IS courses, where different experiences can be shared and integrated into the entire IS curriculum.

References

1. Bourque, P., Fairley, R. E., et al.: Guide to the software engineering body of knowledge, Version 3.0. IEEE Computer Society Press (2014)
2. Chren, S., Buhnova, B., Macak, M., Daubner, L., and Rossi, B.: Mistakes in uml diagrams: Analysis of student projects in a software engineering course. In Proceedings of the 41st International Conference on Software Engineering: Software Engineering Education and Training, Montreal, Canada (2019)
3. Ebert, C., Gallardo, G., Hernantes, J., and Serrano, N.: IEEE Software, 33(3), pp. 94–100 (2016)
4. Gannod, G., Burge, J., and Helmick, M.: Using the inverted classroom to teach software engineering. In Proceedings of the ACM/IEEE 30th International Conference on Software Engineering, pp. 777–786. IEEE (2008)
5. Gotel, O., Scharff, C., and Wildenberg, A.: Teaching software quality assurance by encouraging student contributions to an open source web-based system for the assessment of programming assignments. In ACM SIGCSE Bulletin, volume 40, pp. 214–218. ACM. (2008)
6. Hayes, J. H.: Energizing software engineering education through real-world projects as

experimental studies. In Proceedings of 15th Conference on Software Engineering Education and Training, 2002, pp. 192–206 (2002)

7. Hilburn, T. B. and Townhidnejad, M.: Software quality: A curriculum postscript? In Proceedings of the 31st SIGCSE Technical Symposium on Computer Science Education, pp. 167–171. ACM (2000).
8. Jaccheri, L. and Morasca, S.: On the importance of dialogue with industry about software engineering education. In Proceedings of the 2006 International Workshop on Summit on Software Engineering Education, pp. 5–8. ACM (2006)
9. Jaccheri, M. L.: Software quality and software process improvement course based on interaction with the local software industry. Computer Applications in Engineering Education, 9(4), pp. 265–272 (2001)
10. Jazayeri, M.: The education of a software engineer. In Proceedings of the 19th IEEE International Conference on Automated Software Engineering, pp .18– 27. IEEE (2004)
11. Keijonen, H., Kurhila, J., and Vihavainen, A.: Carry-on effect in extreme apprenticeship. In 2013 IEEE Frontiers in Education Conference, pp. 1150–1155. IEEE (2013)
12. LaPorte, C. Y., April, A., and Bencherif, K.: Teaching software quality assurance in an undergraduate software engineering program. Software Quality Professional, 9(3), (2007)
13. Liebenberg, J., Huisman, M., and Mentz, E.: The relevance of software development education for students. IEEE Transactions on Education, 58(4), pp. 242–248, (2015)
14. Reichlmay, T. J.: Collaborating with industry: Strategies for an undergraduate software engineering program. In Proceedings of the 2006 International Workshop on Summit on Software Engineering Education, pp. 13–16. ACM (2006)
15. Richardson, I., Reid, L., Seidman, S. B., Pattinson, B., and Delaney, Y.: Educating software engineers of the future: Software quality research through problem-based learning. In Proceedings of 24th IEEE-CS Conference on Software Engineering Education and Training, pp. 91–100, (2011)
16. Shaw, M.: Software engineering education: a roadmap. In Proceedings of the conference on The future of Software Engineering, pp. 371–380. ACM, (2000)
17. Strayer, J. F.: How learning in an inverted classroom influences cooperation, innovation and task orientation. Learning environments research, 15(2), pp.171–193, (2012)
18. Sussy, B. O., Calvo-Manzano, J. A., Gonzalo, C., and Tomás, S. F.: Teaching team software process in graduate courses to increase productivity and improve software quality. In 2008 32nd Annual IEEE International Computer Software and Applications Conference, pp. 440–446, (2008)
19. Topi, H.: Is education: Proposing an industry advisory board for is education. ACM Inroads, 9(1), pp.17–18, (2018)
20. Topi, H.: Reflections on the current state and future of information systems education. Journal of Information Systems Education, 30(1), pp.1–9, (2019)