Industry partners' reflections on undergraduate software engineering students: An exploratory pilot qualitative study

Bhuvana Gopal
Computer Science and Engineering
University of Nebraska-Lincoln
bhuvana.gopal@unl.edu

Stephen Cooper
Computer Science and Engineering
University of Nebraska-Lincoln
stephen.cooper@unl.edu

Ryan Bockmon
Computer Science and Engineering
University of Nebraska-Lincoln
ryan.bockmon@huskers.unl.edu

Abstract

We conducted an exploratory pilot qualitative study of software engineering industry professionals who worked closely with sophomore software engineering students in the students' semester-long software engineering course project. We traced these industry partners' reflections, suggestions and advice for students based on their close observations and individual interactions with the students during the course project. We used semi-structured interviews, and combined them with researcher memos and reflective researcher journals. We identify, investigate and discuss these observations in detail, and bring the industry perspective to illuminate the strategies and factors that could help students succeed in the software industry. We also discuss the potential implications of our findings in the context of undergraduate software engineering courses.

1. Introduction

Researchers have long been interested in ensuring that computer science education was relevant to industry needs and identifying areas where newly hired graduates required further preparation to become productive at their new jobs (Moore & Streib, 1989). Many functional aspects of software development, including tools, processes and languages have been well-documented and reported through feedback from the industrial community (Begel & Simon, 2008; Brechner, 2003; Lethbridge, 1998). In addition, these are a major component of their new jobs (Ostroff & Kozlowski, 1992; Tomayko & Hazzan, 2004).

Studies have focused on what soft skills software engineers need in order to succeed in the industry, told solely from an industry perspective without involving students (Begel & Simon, 2008; Li, 2016). Other studies present experiences with and design guidelines for software engineering courses with real world clients (Bruegge, Dutoit, Kobylinski, & Teubner, 2000; Bruegge, Krusche, & Alperowitz, 2015; Krusche, Alperowitz, Bruegge, & Wagner, 2014). In both categories of studies, the perspective of the client / industry professionals regarding students' work in the course and how it relates to soft skills required for students to be hired, has often been overlooked and rarely explored.

In this exploratory pilot study, we present the results of a qualitative study where we interviewed industry partners who were closely involved in mentoring sophomore students in a semester-long software engineering course project. We trace the industry partners' reflections on what they observed during the course of the semester, with a focus on what critical soft skills they found important for students to succeed in an industry environment. We identify the soft skills that outstanding students exhibit. We also discuss in detail, based on industry sponsor reflections, how the students who lacked these soft skills could improve on acquiring and honing said skills.

The unique contribution of this work is that our semi-structured interviews were conducted with industry partners who, as part of their sponsorship of the real world project in the course, worked extensively with an entire cohort of software engineering students in a continued, sustained manner throughout the semester. We strive to obtain a specific and contextual understanding of software engineering soft skills as viewed by expert software engineers. We involved these industry professionals in a controlled academic setting, and gleaned valuable insights on student behavior patterns during their first academic software engineering group course project.

2. Related Work

We review the literature related to our work under two categories: Software engineering courses with real world projects, and studies focused on software engineering expertise.

2.1. Studies related to software engineering courses with real world projects

Several software engineering project courses with real clients were influenced by the original ideas of James Tomayko (Tomayko & Hazzan, 2004), in particular with respect to single-project courses with a real client (Bruegge, Cheng, & Shaw, 1991; Bruegge & Coyne, 1994; Bruegge, 1994). Since then there have been several combinations of project setups and parameters, replacing structured analysis with object-oriented modeling (Bruegge, Blythe, Jackson, & Shufelt, 1992), introducing iterative and collaborative design, adding technical writers (Bruegge, Werner, Uzmack, & Kaufer, 1994), adding use case modeling (Coyne, Bruegge, Dutoit, & Rothenberger, 1995), introducing rationale management and issue-based modeling (Dutoit, Bruegge, & Coyne, 1996), and venturing into globally distributed projects (Bruegge et al., 2000), always with one or more industry clients.

Bruegge et al. (Bruegge et al., 2015) define guidelines for real-world industry projects in classrooms. They called these projects the 6Rs: always look for a real external client who has a real problem to be solved with real data; ask students to work together as a real team in a real project to solve the problem by a real deadline, usually the end of the semester. Vanhanen, Lehtinen and Lassenius (Vanhanen, Lehtinen, & Lassenius, 2012) describe a project course simulating in-vivo software development projects. Gonzales-Morales et al. (González-Morales, De Antonio, & García, 2011) reported the experiences of students in a software engineering course with a real world client project, with specific emphasis on developing students' soft skills. They found that student soft skills can be developed through standalone instruction, embedded in existing courses and based on support programs, and based on formal and informal activities at university levels. Taylor (Taylor, 2016) conducted a questionnaire based survey of 12 industry professionals in South Africa to determine if soft skills were taught adequately at the university level. Seven of the twelve respondents from industry indicated that soft skills were not developed adequately at university.

The studies cited above present experiences designing and implementing software engineering courses with various industry partners and clients. However, they do not report on perspectives of the clients regarding students who participated in those projects.

2.2. Studies related to software engineering expertise and computing education

While there are several studies focused on the importance of industry-relevant topics in the context of computing education, most lack a direct connection between industry perspectives and software engineering courses. Lethbridge (Lethbridge, 1998) surveyed 168 software professionals about the relevance of computer science education topics from the ACM Computing Curricula. Kelley's work examined star performers, including software engineers at HP and Bell Labs (Kelley, 1999). Kelley conducted a seven year study evaluating how productive engineers were before and after they had learned star work strategies by going through our productivity improvement program.

Hewner and Guzdial invested what employers in a small game company look for in new graduates (Hewner & Guzdial, 2010). They interviewed and surveyed over 30 engineers, managers, and artists about qualifications for recent graduates. The authors identified programming skills as well as people skills and suggested differences in expectations between new and senior hires. Begel and Simon (Begel & Simon, 2008) investigated 8 new hires, following them at Microsoft for 4 weeks and examining their daily tasks. The authors found that new hires need to identify "tasks that have an impact", to be "persistent", and to collaborate effectively in a "large-scale software team setting". Li and Ko (Li, 2016) investigated what expert software engineers thought were attributes of great software engineers. They identified attributes that were important for the engineering of software, and how these attributes related to each other.

The above studies show that there is a good body of related work in two important but yet unconnected topics: a) software engineering courses with industry projects, and b) industry perspectives regarding recent graduates and new hires. However, there has been a lack of focus on industry perspectives on students who are currently in the process of learning software engineering. In this pilot study, we seek to give greater context to our understanding of students' software engineering expertise, as viewed by industry professionals, thereby connecting the two perspectives mentioned above. Our pilot study explores industry perspectives on student soft skills with actual, sustained interaction between industry professionals and students throughout a semester-long software engineering course project. These industry sponsors had opportunities to interact closely with every individual student throughout the semester, and observed and recorded student progress periodically at various checkpoints.

2.3. Research Question

We collected, analyzed and synthesized reflections from industry partners who actively worked with a cohort of sophomore software engineering students through a semester-long course project. We sought to remedy the lack of firsthand connection between industry professionals and software engineering students in prior work by trying to understand:

What were the soft skills that expert software professionals deemed valuable and industry-relevant in undergraduate software engineering students?

3. Research Methodology

This research project was determined to be exempt by our University's Institutional Review Board. To answer the research question, we performed an empirical study with 5 semi-structured interviews with architect-level, lead and senior software engineers as well as product managers, with extensive experience in the software industry. We determined that data saturation was achieved based on the richness of the data we gathered as well as guidelines by Creswell (Creswell & Poth, 2016), indicating that anywhere between 3-10 participants are typically required for an exploratory qualitative interview study like ours.

3.1. Context: Course Project

Data for this study were collected from industry partners collaborating with sophomore students taking a software engineering class in the spring of 2019, in a large R1 university in the USA. This was the first group software engineering project that students were given the opportunity to work with in their curricular progression. The course project lasted 15 weeks. The project was proposed by industry professionals. These industry sponsors worked at the local branch of a large national software company, specializing in financial services. A total of five focal participants – representing the diversity in roles within a software engineering team in the industry- participated in individual interviews. Our participant group had one technical architect, one lead software engineer, two senior software engineers and one senior product manager.

There were 38 students in the class. They were divided into 7 teams of 4 students each, and 2 teams consisting of 5 students each. Teams were selected and composed based on various factors including whether students wanted to assume a technical role or business analysis role, as well as prior internship experience. Students were provided with an spreadsheet of existing skills in the company and were asked to develop a web application that would replace a legacy system at the sponsors' company. This application would keep track of employees' past, current and in progress technical and process skills, and would aid the human resource department in the company identify employee skills appropriate for projects. The application was required to contain functionality for user authentication, skills classification, job requirements posting, skills selection and skills matching, and was required to be hosted on a cloud platform.

3.2. Industry Sponsor - Student Interaction

The industry sponsors provided the problem description for the course project and participated in requirements gathering through face to face user interviews. Sponsors then remained involved at every stage of the project. Sponsor-student interactions included:

- Weekly meetings with each student team.
- Weekly emails clarifying requirements, feature requests etc.
- Continued communication with individual students regarding technology and process related questions through video calls and emails.
- Acclimatizing student teams to certain aspects of their new hire on-boarding processes including company rules and policies regarding legal procedures, since the course project involved elements that involved a process heavy, document rich environment.
- Three formal release meetings during the 15 weeks, each 5 weeks apart, including a large showcase event at the end of the semester.
- Sponsors assessed and ranked every student based on technical skills, process skills as well as work ethic throughout the semester. The sponsors recognized standout individuals and teams at the final showcase event with awards in multiple categories, at the end of the semester-long course project.
- The primary learning outcomes for the course project were twofold: technical, and process oriented. The technical skills we included were software architecture correctness based on SOLID principles, database design and implementation (with basic database security), and software testing (unit testing, integration testing, and continuous integration). Soft skill learning outcomes were based on how well the students did during requirement elicitation, product knowledge, adaptability, communication within the team and with the sponsors, and work ethic.
- After each release meeting, the sponsors met with the instructor and TAs and assessed each student individually based on a rubric that consisted of both technical and soft skills. These periodic assessment data allowed sponsors to gain clarity on each student's individual contribution as well as their collective team effort.
- The final showcase event involved all students, the instructor, teaching assistants (TAs), the sponsors and audience members including other professors and industry professionals. Each student team presented the highlights of their implementation of the project for 10 minutes.

3.3. Data Collection and Analytical Methods - Interviews, coding and qualitative analysis

We conducted one-on-one semi-structured interviews (Weiss, 1995; Wolcott, 2005) and combined them with researcher memos, and reflective researcher journals. We maintained and shared field journals wherein we kept jottings (Emerson, Fretz, & Shaw, 2011) that we turned into expanded field notes. We analyzed these field notes along with transcribed semi-structured interviews. Our analysis of the data employed a parallel approach with reflective collaborative check-ins and theming, consistent with the grounded theory approach in qualitative analysis (Creswell & Poth, 2016).

We began with an initial individual coding of transcripts. We generated and assigned over 200 codes. These codes were then mapped into different visual configurations based on code topic groupings and frequency. We conducted iterative chunking and collaborative theming (Anfara Jr, Brown, & Mangione, 2002). The iterative nature of the analysis process helped us to achieve progressive focusing and to develop conceptual ideas. We used code maps with frequency plots, to guide code grouping and chunking which supported our collaborative integrative theming and theory development (Corbin & Strauss, 2014). The combination of the two approaches helped us to identify and connect the elements we both noted among the emerging themes (Olmanson et al., 2016).

3.4. Reliability and Trustworthiness

We utilized intercoder agreement (Creswell & Poth, 2016) based on the use of multiple coders to analyze transcript data. Two of the researchers analyzed the individual codes separately to come up with themes. We report an intercoder agreement of 100% among all themes. To aid in ensuring trustworthiness, participants were invited to give feedback on our interpretations and theory building, commonly known as Member Checking (Lincoln & Guba, 1985). Participants were offered a descriptive summary of our themes and theories, and/or of reading the entire paper, and invited to give open-ended feedback on the extent to which our interpretations and organization resonated with their experiences. All participants agreed to give written feedback. This feedback was used to clarify information and confirm our organization and findings. Our goal with this exercise was to look for agreement / disagreement with their thoughts expressed in their interviews for this study, and we report no discrepancies.

3.5. Data Organization

In the remainder of this paper, we have presented the experiences and voices of our participants in the following data sections, organized around the overarching theme of soft skills that the sponsors observed during their regular and involved interactions with student teams. This way of organizing data was in line with the way in which education researchers position participant attrition and participation (Ketelhut & Schifter, 2011). It also aligned with how industry partners spoke about their experiences. In the data presentations that follow, we forefront participant voices in the text and employ gleanings from student quotations as subheadings. This is in alignment with the recommendations to present direct quotations (Foley, 2002). Sections 4.1 through 4.5 represent the themes that emerged from our data analysis.

4. Participant voices and meaning making: Analysis and Discussion

What makes our study unique is the active and close engagement of students with the industry sponsor throughout the semester. This makes the sponsors uniquely qualified to make comments and suggestions based on fifteen weeks of close interactions with students. The sponsors' voices express concrete, actual issues they found that were far from theoretical or abstract.

4.1. Understand business needs and develop a detailed knowledge of the product

"Typically they don't yet understand our business or customers with the industry and important that they recognize how important is to understand the needs of their clients." - Kevin

"It is important to be able to work directly on the business level, and we observe them to see if they have potential..." - Nancy

"When we have somebody coming in and we recognize very early on that they don't understand our business, it becomes hard for them to succeed." - Briana

"Software is just a tool for solving domain problems." - Bob

"Developers need to understand the priority of business tasks...and the priority of technical tasks in relation to business tasks..." - Kevin

Nancy, Briana, Bob and Kevin all echoed the opinion that the usefulness and hence the professional success of a junior developer in a software company is directly related to how well they can translate business requirements into software. Novice software engineers, such as our students, typically do not have an understanding of what the client needs or of the business needs. Technical tasks, whether it be feature development, bug fixes or technical debt, need to be correlated with a corresponding business task. These thoughts are in alignment with previous research by Li and Ko (Li, 2016) who focused on seasoned software engineers rather than on novices. Li and Ko's findings indicate that software engineers who found success often were knowledgeable about their business and technical domains (Li, 2016).

"Develop detailed knowledge of the product from day one." - Nancy

Students attempt to master immense amounts of material in a short amount of time, all the while trying to take on tasks that have an impact on the team. This struggle leads to an escalation in stress and anxiety, not unlike the situation faced by new hires (Begel & Simon, 2008). Nancy puts this in perspective by setting the focus on the product: a sustained effort to learn the product that the new hire is assigned to work on, is an important step to success. Our students needed to focus on the product a lot more.

"Skipping over important documentation because it's boring or time consuming can be a costly mistake. Take notes." - Cody

"Dig into manuals other source materials for deep understanding before getting started with a new tool or framework." - Kevin

In a real industry project, our students were learning every day, correcting mistakes, hearing terminology they might not have known. The key takeaway from these participant quotes is there is a need for students to constantly learn and improve their domain knowledge as well as their technical skills.

4.2. Communicate clearly and early on

"Successful projects ... very important to have the ability to communicate back and forth around getting to know the requirements." - Cody

Cody observed that the students who were successful in implementing the project satisfactorily displayed a strong ability to communicate back and forth with the sponsors, building their knowledge of project requirements. This theme reflects the research by Storey (Storey, Zagalsky, Figueira Filho, Singer, & German, 2016), who explain that developers are becoming increasingly social and rely on their social networks to keep up to date, to find projects to contribute to, and to find others to collaborate with. The rise of the social programmer (Storey, Singer, Cleary, Figueira Filho, & Zagalsky, 2014) and the ways that communities of developers make use of increasingly social tools have led to the emergence of a highly participatory culture of software development. Learning how to effectively communicate with stakeholders of all levels, from fellow developers, managers, analysts and clients in a professional manner is key to honing requirement gathering skills. Even more important than initial communication is the ability to go back and forth on requirements.

"Be transparent with your team every day and communicate clearly." - Bob

"...leverage our team's shared knowledge." - Kevin

Teamwork skills in software engineering hinge on effective communication (Raibulet & Fontana, 2018). Kevin and Bob voice opinions that are consistent with findings about teamwork and communication (Lingard & Berry, 2002). Being able to work effectively in teams is an important learning objective for software engineering students. As shown by participant voices above, communication is not just about learning the domain or the product. Transparency and clarity during communication is very valuable. Such communication helps share one's strengths as well as learn from others, enabling students to leverage and take advantage of the collective knowledge of the team.

"...important to foster an environment where expectations are clear."- Nancy

Effective communication skills also help regroup and practice better boundary setting and problem solving when things go awry or a boundary is broken in the professional environment. Nancy found that some students were unclear on what was expected of them within the team. Actual instruction on and practice opportunities for team based communication skills are necessary in software engineering classes.

4.3. Ask for help

"Ask for help when you need it." - Briana

There is newcomer socialization literature that addresses how learners fit into professional organizations. Much of it is relevant to our students, who were simultaneously introduced to several aspects of the project — writing, testing and reviewing code, release management, data analysis, UI/UX consultation, code maintenance, building product features, and contributing to technical stack long term goals, to name a few. Begel and Simon (Begel & Simon, 2008) find that newcomers are anxious due to their lack of knowledge about the requirements of their role, of the chain of command, and of knowing who in the organization that can help them complete their tasks. They found that newcomers hesitate to ask questions since it reveals to co-workers and managers that one is not knowledgeable.

"When you are stuck, ask someone for help." - Bob

When new developers find themselves in situations that are considerably different from a university capstone experience (typically project based and developing a small greenfield project), several social and communication problems arise (Begel & Simon, 2008). Briana and Bob both reiterate the need for students to ask questions and ask for help when needed. This advice feeds into the continuing narrative we see in our study about transparent and timely communication.

4.4. Work ethic, judgment, attitude and adaptability

"Avoid the freelancer attitude." - Bob

Bob sheds light on an often overlooked aspect of software engineering: motivation. The "freelancer" attitude he describes here refers to the lack of initiative, or drive, to push the product forward. In the software industry, one of the best indicators of curiosity is displaying self motivation.

"Adaptability to the set of... onboarding processes is important to be able to thrive." - Briana

Students that stood out and shone asked for and took on a broad range of tasks. They gauged their limits and pushed themselves to learn as much as they can as quickly as they can. They did not wait for work, as evidenced by their burndown charts and continued interaction with the sponsors: finishing assigned work, checking for quality, and being driven to take on more work got these students noticed and fostered confidence in their abilities and attitude. In short, attitude mattered.

"Always deliver a high quality UX. Products live and die on their UX. Simply meeting acceptance criteria is not enough." - Cody

"Do not skip testing." - Kevin

Cody's statement could be extended beyond the User Experience (UX) paradigm, but in essence, it explains how important UX and software quality are to the success of a software product. Two of the nine teams failed to focus on testing. One lost focus on the UI/UX. Students and potential new hires need to remember that it is simply not sufficient to develop to specifications. Software testing is important, and a necessary skill to help learn the nuances of the product being built.

"Know when to take on technical debt and when to pay it off." - Kevin

Technical debt (also known as design debt or code debt, but can be also related to other technical endeavors) is a concept in software development that reflects the implied cost of additional rework caused by choosing an easy (limited) solution now instead of using a better approach that would take longer (*The Engineer's Complete Guide to Technical Debt, year = 2020, url = https://www.stepsize.com/blog/complete-guide-to-technical-debt, lastchecked = 05.14.2021*, n.d.). An often overlooked aspect of software engineering instruction, judgment is an important factor is helping software engineering students and potential new hires succeed in the industry. Kevin observed, based on periodic scrutiny of the teams' product and release backlogs, that some teams took on technical debt tasks too early, or too late, indicating that student judgment on timing these tasks needed honing.

4.5. Curiosity and continued learning

"Welcome the opportunity to work on something new and challenging." - Nancy

Every project that software engineers work on has the potential to be different: different frameworks, different setup, different goals, different team members, different business needs, etc. Being able to learn and adapt to new and challenging environments is a skill that sets apart students from one another. Nancy noticed that the students who she and her colleagues identified as the strongest welcomed new challenges, and delivered "nice-to-have" features in addition to those in the "must-have" list.

"Stay sharp and current." - Briana

"....attend collaborative meetups." - Cody

Being curious leads to creative solutions, new interests and best practices. Software engineering is a dynamic discipline with continuous technological and process improvements to suit business needs. Students who wish to succeed in industry need to learn how to stay up to date with emerging technologies and processes.

Our overall findings of this section complement the work by Begel and Simon (Begel & Simon, 2008) in illuminating what soft skills to focus on while working on real world software engineering projects, and how to succeed as a new hire in a company. Our findings also reiterate and expand upon those by Taylor (Taylor, 2016) who found that communication, flexibility, self management and teamwork were rated highest by the industry professionals in their study.

5. Threats to validity

As with any empirical study, there are various internal and external threats to validity that our results are subject to. While our analysis was systematic, other researchers may glean different themes, attributes and definitions than ours from the same raw data. Our sampling method included 5 interviews (2 women, 3 men) that yielded rich insights. Typically, for data saturation in exploratory qualitative interview such as ours, the recommendation is 3-10 participants (Creswell & Poth, 2016). However, it was a small sample in a single location in the Midwest of the US, and this could lead to under-representation of minorities. Nonetheless, as we interviewed sponsors from a successful financial company whose software was widely used by the public, we deem our findings to be valuable and relevant.

6. Conclusion and Future Work

Software engineering in the industry requires an ability to deftly combine skills pertaining to coding, business analysis, team work and communication, and staying current with emerging technologies at the same time. In our study, our participants were able to explain what they observed when students lacked certain skills related to software engineering.

The contribution of this exploratory pilot study is that we have gleaned valuable, industry-relevant information from one-on-one interviews with industry professionals based on their semester-long, close interactions with students. This gives them a unique perspective on each individual student's progress throughout the course project. Our key result themes indicate that outstanding students:

- Worked hard to gain a clear and strong understanding of what business needs their software aims to fulfill. They strive to obtain detailed knowledge about the product they are asked to help develop.
- Were curious, and exhibit the drive to stay current and sharp with technology and process advancements. They are adaptable to onboarding requirements.
- Asked for help when they are stuck.
- Displayed a strong work ethic, and avoid the freelancer attitude.
- Were transparent with their team and stakeholders, and communicate with clarity in a timely manner.
- Were adaptable and flexible with onboarding processes.

There is much scope for future work in this domain. In future work, we anticipate exploring what the discovery of these soft skills means for project based software engineering education. Now that we have identified valuable soft skills that students need to learn during their coursework in software engineering, how can we facilitate student learning of these skills? The results from this study offer a good starting point to examine, revise and restructure learning outcomes and assessment mechanisms for project based software engineering courses, to better focus on soft skills deemed valuable by industry. We would also like to expand this exploratory pilot study across different sponsor and student samples from different courses and industry domains. In addition, based on the results of this study, we intend to investigate if process oriented pedagogies such as Process Oriented Guided Inquiry-Based Learning (POGIL) (Kussmaul, 2014) can help improve learning outcomes concerning the soft skills students need to succeed in the software industry.

7. References

Anfara Jr, V. A., Brown, K. M., & Mangione, T. L. (2002). Qualitative analysis on stage: Making the research process more public. *Educational researcher*, *31*(7), 28–38.

Begel, A., & Simon, B. (2008). Novice software developers, all over again. In *Proceedings of the fourth international workshop on computing education research* (p. 3–14). New York, NY, USA: Association for Computing Machinery. Retrieved from `https://doi.org/10.1145/1404520.1404522` doi: 10.1145/1404520.1404522

Brechner, E. (2003). Things they would not teach me of in college: what microsoft developers learn later. In *Companion of the 18th annual acm sigplan conference on object-oriented programming, systems, languages, and applications* (pp. 134–136).

Bruegge, B. (1994). From toy system to real system development: Improvements in software engineering education. In *Software engineering im unterricht der hochschulen seuh'94* (pp. 62–72). Springer.

Bruegge, B., Blythe, J., Jackson, J., & Shufelt, J. (1992). Object-oriented system modeling with omt. *ACM SIGPLAN Notices*, *27*(10), 359–376.

Bruegge, B., Cheng, J., & Shaw, M. (1991). *A software engineering project course with a real client* (Tech. Rep.). Carnegie-Mellon Univ Pittsburgh PA Software Engineering Inst.

Bruegge, B., & Coyne, R. F. (1994). Teaching iterative and collaborative design: Lessons and directions. In *Conference on software engineering education* (pp. 411–427).

Bruegge, B., Dutoit, A. H., Kobylinski, R., & Teubner, G. (2000). Transatlantic project courses in a university environment. In *Proceedings seventh asia-pacific software engeering conference. apsec 2000* (pp. 30–37).

Bruegge, B., Krusche, S., & Alperowitz, L. (2015). Software engineering project courses with industrial clients. *ACM Transactions on Computing Education (TOCE)*, *15*(4), 1–31.

Bruegge, B., Werner, M., Uzmack, J., & Kaufer, D. (1994). Fostering co-development between software engineers and technical writers. In *Proceedings software education conference (srig-et'94)* (pp. 4–11).

Corbin, J., & Strauss, A. (2014). *Basics of qualitative research: Techniques and procedures for developing grounded theory*. Sage publications.

Coyne, R. F., Bruegge, B., Dutoit, A. H., & Rothenberger, D. (1995). Teaching more comprehensive model-based software engineering: experience with objectory's use case approach. In *Conference on software engineering education* (pp. 339–374).

Creswell, J. W., & Poth, C. N. (2016). *Qualitative inquiry and research design: Choosing among five approaches*. Sage publications.

Dutoit, A. H., Bruegge, B., & Coyne, R. F. (1996). Using an issue-based model in a team-based software engineering course. In *Proceedings 1996 international conference software engineering: Education and practice* (pp. 130–137).

Emerson, R. M., Fretz, R. I., & Shaw, L. L. (2011). *Writing ethnographic fieldnotes*. University of Chicago Press.

The Engineer's Complete Guide to Technical Debt, year = 2020, url = https://www.stepsize.com/blog/complete-guide-to-technical-debt, lastchecked = 05.14.2021. (n.d.).

Foley, D. E. (2002). Critical ethnography: The reflexive turn. *International Journal of Qualitative Studies in Education*, *15*(4), 469–490.

González-Morales, D., De Antonio, L. M. M., & García, J. L. R. (2011). Teaching “soft” skills in software engineering. In *2011 ieee global engineering education conference (educon)* (pp. 630–637).

Hewner, M., & Guzdial, M. (2010). What game developers look for in a new graduate: interviews and surveys at one game company. In *Proceedings of the 41st acm technical symposium on computer science education* (pp. 275–279).

Kelley, R. E. (1999). How to be a star engineer. *Ieee speCtruM*, *36*(10), 51–58.

Ketelhut, D. J., & Schifter, C. C. (2011). Teachers and game-based learning: Improving understanding of how to increase efficacy of adoption. *Computers & Education*, *56*(2), 539–546.

Krusche, S., Alperowitz, L., Bruegge, B., & Wagner, M. O. (2014). Rugby: an agile process model based on continuous delivery. In *Proceedings of the 1st international workshop on rapid continuous software engineering* (pp. 42–50).

Kussmaul, C. (2014). Guiding students to discover concepts and develop process skills with pogil. In *Proceedings of the 15th annual conference on information technology education* (pp. 159–160).

Lethbridge, T. C. (1998). A survey of the relevance of computer science and software engineering education. In *Proceedings 11th conference on software engineering education* (pp. 56–66).

Li, P. L. (2016). *What makes a great software engineer* (Unpublished doctoral dissertation).

Lincoln, Y. S., & Guba, E. G. (1985). *Naturalistic inquiry. newberry park.* Ca: Sage.

Lingard, R., & Berry, E. (2002). Teaching teamwork skills in software engineering based on an understanding of factors affecting group performance. In *32nd annual frontiers in education* (Vol. 3, pp. S3G–S3G).

Moore, F. L., & Streib, J. T. (1989). Identifying the gaps between education and training. In *Proceedings of the twentieth sigcse technical symposium on computer science education* (pp. 52–55).

Olmanson, J., Kennett, K., Magnifico, A., McCarthey, S., Searsmith, D., Cope, B., & Kalantzis, M. (2016). Visualizing revision: Leveraging student-generated between-draft diagramming data in support of academic writing development. *Technology, Knowledge and Learning*, *21*(1), 99–123.

Ostroff, C., & Kozlowski, S. W. (1992). Organizational socialization as a learning process: The role of information acquisition. *Personnel psychology*, *45*(4), 849–874.

Raibulet, C., & Fontana, F. A. (2018). Collaborative and teamwork software development in an undergraduate software engineering course. *Journal of Systems and Software*, *144*, 409–422.

Storey, M.-A., Singer, L., Cleary, B., Figueira Filho, F., & Zagalsky, A. (2014). The (r) evolution of social media in software engineering. In *Future of software engineering proceedings* (pp. 100–116).

Storey, M.-A., Zagalsky, A., Figueira Filho, F., Singer, L., & German, D. M. (2016). How social and communication channels shape and challenge a participatory culture in software development. *IEEE Transactions on Software Engineering*, *43*(2), 185–204.

Taylor, E. (2016). Investigating the perception of stakeholders on soft skills development of students: Evidence from south africa. *Interdisciplinary journal of e-skills and lifelong learning*, *12*(1), 1–18.

Tomayko, J. E., & Hazzan, O. (2004). *Human aspects of software engineering*. Firewall Media.

Vanhanen, J., Lehtinen, T. O., & Lassenius, C. (2012). Teaching real-world software engineering through a capstone project course with industrial customers. In *2012 first international workshop on software engineering education based on real-world experiences (edurex)* (pp. 29–32).

Weiss, R. S. (1995). *Learning from strangers: The art and method of qualitative interview studies*. Simon and Schuster.

Wolcott, H. (2005). *The art of fieldwork. walnut creek, calif.* AltaMira Press/A Division of Rowman & Littlefield Publishers, Inc.