Which are harder? Soft skills or hard skills?

Vreda Pieterse[1] and Marko van Eekelen[2]

[1] University of Pretoria, vreda.pieterse@up.ac.za
[2] Open University of the Netherlands & Radboud University Nijmegen, Marko.vanEekelen@ou.nl

Abstract. This paper describes some technical and employability skills that are essential for our students to succeed in a career in software development. We conducted research aimed at understanding the students' problems when required to develop these skills. We explain our techniques for observing skills gaps. Knowledge about these gaps enables us to intervene and suggest remedial action. We discuss how we create opportunities for our students to enhance their skills, based on our experience and the findings of our research.

Keywords: software development, technical skills, employability skills, soft skills

1 Introduction

Apart from equipping our students with the technical knowledge and practical experience needed to enter the workforce, it is our educational responsibility to create opportunities for them to develop their employability skills.

Employment experts agree that technical skills may secure an interview, but that soft skills may well be a decisive factor in landing and keeping a job. Potential employees are expected not only to have the skills required in the job description of a vacancy, but also to convince their potential employer that they will be able to make progress in an enterprise and contribute successfully to its strategic directions.

Our teaching is aimed at providing a complete learning experience to cover the spectrum of skills required in a career in software development. These skills can roughly be classified in two categories, namely technical skills, often called hard skills and employability skills, commonly referred to as soft skills. Here we mention a broad selection of each of these types of skills relevant to our context and describe how we create opportunities for our students to develop these skills in our final-year software engineering module.

2 Technical skills

A software engineering course should introduce students to common software engineering practices and tools from both a theoretical and a practical perspective

[10]. Every student should aquire the necessary technical fluency skills such as unit testing, pair programming, refactoring and continuous integration. Besides these general skills and those specified in the Computer Science Curricula 2013 report by ACM/IEEE [6], the following should receive attention:

Problem solving Higher cognitive skills such as inference, problem solving and product development are learned through life experiences similar to those for learning social skills. Stokes and Fisher [14] observe that working with constraints is critical to creative achievement. For this reason, we emphasise constraints when presenting the software development tasks in our course.

Configuration management It is common practice to use modern configuration management tools such as git[3] or subversion so that a team of people can be facilitated to work concurrently on the same artefacts, resolving conflicts as needed. We expect our students to use these tools.

Build tools Modern software systems are generally complex and building complex systems requires identifying and configuring the dependencies among a variety of components, which may themselves be developed in different technologies. Activities such as linking, compiling, testing, packaging, deployment and distribution of these systems are complex. It is standard practice in industry to use platform-independent build tools which automate these activities. Examples include `make`, `Apache`, `Ant`, `Maven`, `Gradle` and `npm`.

3 Employability skills

The need for employability skills is emphasised in the Computer Science Curricula 2013 report by ACM/IEEE, where the knowledge areas explicitly include social issues and professional practice as well as project management with all its facets, as part of the software engineering knowledge area [6]. Liebenberg and Pieterse [7] observe that high-value computing skills and capabilities alone are not enough when one has to compete and succeed as a software developer in industry. The following skills are important in all careers:

Communication The success or failure of a software engineering project can often be attributed to the effectiveness of communication among the various stakeholders of the system under development [1]. We monitor our students' communication skills by regularly evaluating all forms of communication, from the quality of their comments in code and git commit messages, the quality and coherence of the documentation of their projects to their presentation skills when demonstrating their projects.

[3] https://git-scm.com/

Management and planning It is well known that software engineering management and planning involve balancing the scope, budget, time-to-market and quality of a software project. The consequences of bad planning may include the failure of the project as well as interpersonal disasters[4]. We encourage the use of software tools for project management, including the use of burn-down charts and Gantt charts.

Teamwork and collaboration Teamwork is not merely the ability to work well as a member of a team. It includes aspects such as getting along with people of different ages, genders, races, religions or political persuasions; defining one's role in a team; identifying the strengths of the other team members; and being able to lead a team effectively [8]. We aim to enhance the teamwork skills of our students by fostering the characteristics of high-performing collaborative teams identified by Cheruvelil *et al.* [3], namely the positive interdependence of team members, effective communication, and individual and group accountability.

Interpersonal relations Interpersonal skills have two components, namely social sensitivity and emotional engagement. Social sensitivity is the capacity to maintain healthy social relationships [16]. Emotional engagement is the level of empathy one has for the other team members and one's devotion to the project as a whole [11]. During the first six weeks of our course, we assigned students to short-lived teams to complete a task. This strategy provides a platform where students are exposed to situations where they could use interpersonal skills in cases where mistakes could be made without having to resolve the harm caused.

4 Relative difficulty of learning hard and soft skills

We conducted a survey, asking our students to compare the difficulty of learning technical skills with the difficulty of acquiring social skills. There were three options: the Pretoria University Software Engineering class, the Radboud University Software Engineering class [2] or both. To avoid the influence of cultural differences, we chose a single university: Pretoria, since this university had the highest number of students. The students had to answer the following multiple-choice question and then write a sentence or paragraph to explain the reasons for their answer.

At the time, the participants were working in teams on their final capstone projects. Of the 160 students in the class, 107 completed the survey, giving us a response rate of 67%. Five of the responses were incomplete and have not been included in our analysis. We used the explanation that the respondents who selected the middle option to classify them as being either *both are challenging* or *both are easy.*

To visualise the relative ease with which the students mastered the two categories of skills, we classified each type separately in four classes, namely *very easy, easy, challenging* and *very challenging.* When classifying each respondent

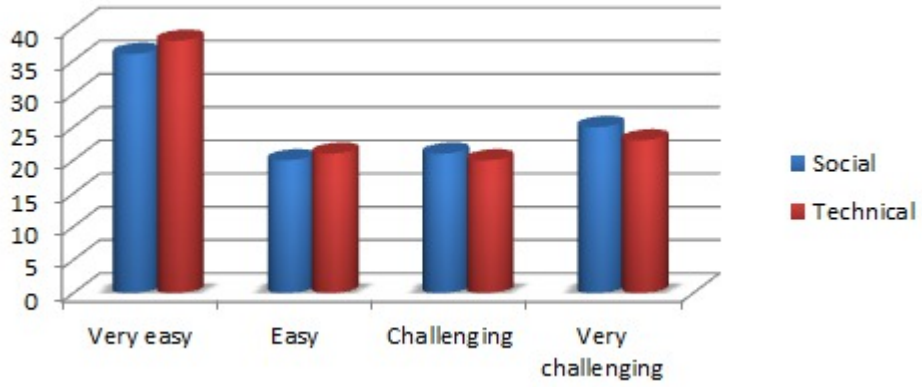
Fig. 1. The question students had to answer

COS301 is about learning skills in a variety of domains. The development of skills, whether technical or social, requires practice. Rate the level of difficulty for you to acquire social skills relevant to work in your team in relation to the difficulty you experience in acquiring technical skills relevant to your project.

- *Acquiring social skills is fun – It comes naturally to me*
- *Acquiring social skills is easier*
- *I find both equally challenging or easy*
- *Acquiring technical skills is easier*
- *Acquiring technical skills is fun – I like solving technical problems*

in terms of their ease of acquiring social skills, students who indicated that both were easy as well as those who indicated that they found social skills much easier were counted as *very easy*; those who responded that social skills were easier were counted as *easy*; those who said technical skills were easier were counted as *challenging*; and the rest were counted as *very challenging*. Similarly the students were classified in terms of their comfort about learning technical skills. Figure 2 shows the results.

Fig. 2. Number of students in each category per type of skill (n=102)



It is clear that the number of students in each of the classes is almost the same for both types of skills, though the number of students who found it very easy to overcome technical obstacles was marginally higher than the number of students who had issues with mastering social skills. Stated differently, those who struggle with social skills are only slightly higher in number than those who struggle with technical skills. The majority of the students stated that they found learning technical skills and acquiring social skills equally easy. A gifted student

who claimed to breeze through the academic programme made the following claim:

> *I am already quite a social person and I really like to think I have a knack for understanding technology at the same time.*

Many students who claimed that they did not have difficulties when social skills were needed, based their argument on the fact that they already had these skills or had the right personality to help them perform teamwork almost effortlessly. The following remark is representative of the comments of these students:

> *I'm a social person. I do well in social situations and am the most sociable guy in the group. So learning to interact with fellow group members wasn't challenging at all to me. I actually enjoyed it.*

Another frequent reason the respondents gave for finding the acquisition of social skills very easy was simply that the other people with whom they had to work were pleasant and accommodating.

> *Since we are all friends I find the question above to be one sided since we have already established the social skills in the group to work together. Therefore I would find acquiring social skills to be easier.*

Students who claimed that they found it easy to learn technical skills often admitted that they shied away from the need to interact with people, as is evident from the following remark:

> *I'm a more technical person and find social situations very hard to deal with, so I'd much rather solve technical problems than deal with social problems.*

A number of students explained the aspects of social interaction that contributed to their finding it difficult to master social skills. In contrast to technical knowledge which is more likely to be exact, they pointed out that people were complex and might be inconsistent. Often there are many ways to deal with people and none of them is unconditionally wrong or guaranteed to have the required positive effect. These uncertainties might make it more challenging for these students to collaborate with people than to learn technical aspects, as explained in the following remark:

> *I am not the most social of people and dealing with technical aspects can be frustrating but not as much as dealing with people. People can be quite difficult at times.*

A respondent found it difficult to trust other people and might be tempted to do work on their behalf so as to ensure that the project would not fail not realising that he was denying himself the opportunity to improve his management skills and also denying the other team members an opportunity to gain technical experience.

The following comment by this respondent reveals this:

> *People tend to be unreliable and it is easier to finish a 1 hour bug fix than to wait days or weeks for others to get to it. Ensuring others do their jobs sometimes helps, however it also does put oneself behind on work and that is risky.*

5 Recommendations

This pilot study revealed that, although many students are confident that they are capable of learning the required skills, some individuals may need assistance and encouragement to learn some of the skills. It is often the case that someone who has reached high technical competency may lack social skills, and vice versa. We recommend that a complete learning experience for all our students should be ensured. We describe how we attempt to identify the skills gaps of our students on an individual basis and how we provide opportunities for students to close these gaps. This should serve as inspiration for others to apply similar strategies, appropriate to their situation, to achieve the same goals.

5.1 Uncovering skills gaps

We subscribe to the learning theory of Gibbs [5], supported by Schank [13], that doing is an effective way of learning. We believe that students learn best by resolving their own issues because this increases their sense of accomplishment. It is possible, however, that problems which are not dealt with appropriately at an early stage may grow into bigger problems which may be difficult to resolve at a later stage. For this reason, we observe the team activities closely and are constantly on the alert to signs of underdeveloped skills that often manifest as turmoil in a team.

We instruct our students to complete peer reviews at regular intervals. The main purpose of these peer reviews is to provide a structured opportunity to reflect on teamwork experiences. Such self-reflection may lead an individual to discover personal skills gaps. Apart from serving as a reflection tool, we use these reviews as an instrument to gauge the skill levels of the students. The questions that the students have to answer are intended to guide them to reflect on their own contributions and also on the contributions of the other members. The questions used in our reviews are described by Marshall *et al.* [9].

We analyse the feedback the students provide in their peer ratings, using the procedure described by Pieterse and Thompson [12]. This analysis reveals whether or not there is conflict in a team, which may be an indication of a lack of social skills. Students may report the inability of one or more of their peers to complete certain tasks, in which case the lack of technical skills can be identified.

5.2 Closing skills gaps

Technical skills as well as social skills can be enhanced when students work in teams. When students with varying viewpoints are grouped together for a

project, conflict is likely to arise. When a team deals constructively with this conflict and follows procedure, the act of resolving as well as the resolution itself will probably motivate the members. This in turn can contribute to improved team performance. If a team performs at its peak, the combined achievement could surpass the sum of the achievements of the individual members [15].

The downside of differences in opinion and misunderstanding among members, is that it may decrease motivation. We try to intervene swiftly and with a constructive agenda when we observe signs of destructive conflict. We call our unobtrusive intervention our *chat-walk-chat* strategy [10]. Ideally, from the student's point of view, these chats should seem coincidental, but from the staff member's point of view, they are an active means of seeking opportunities to create a "coincidental" meeting. The use of social media and knowledge of the lecture schedules of the courses for which the students are registered, make it possible to bump into a student on campus and start a conversation aimed at guiding the student to deal with the lurking problem.

When a student complains to staff members, we try to respond openly and as soon as possible. In such a case, we arrange a meeting with all the students involved. The meeting venue is the lecturer's office and the time is agreed individually with the students involved. We try to be as discreet and sympathetic as possible and are careful not to reveal the whistle-blower to the affected parties. We simply state that the issue came to our attention, describe the issue in general terms and then ask all parties, including the whistle-blower, to state their opinion about the truth of our summary of the problem. In most cases the discussion can be steered towards better mutual understanding. Often the whistle-blowers had a greater role in instigating the problem than they may care to admit.

6 Conclusion

Our research revealed that many of our students were confident that they could master the required skills. The students who stated that some of the skills might be difficult to acquire have also succeeded in identifying the reasons for finding it difficult and proposed actions they would take or had taken to overcome their difficulties. This is evidence that our teaching strategies have been successful and that the students are generally appreciative of our efforts.

In our presentation of the software engineering module, we aim to create optimal opportunities for students to learn by doing tasks on their own, and to develop the required skills through experiential learning, without smothering or policing them.

In future work, we also intend to use the available data to check whether there is a correlation between how students have participated and performed and how hard they found the acquisition of the skills (both soft and hard). Furthermore, it is the intention to enhance this study with a comparison between a Dutch university and a South-African university, including the possibility of cultural influences too.

References

1. Bostrom, R.P.: Successful application of communication techniques to improve the systems development process. Information & Management 16(5), 279 – 295 (1989)
2. Buisman, A.L.D., van Eekelen, M.C.J.D.: Gamification in educational software development. In: Proceedings of the Computer Science Education Research Conference. pp. 9–20. CSERC '14, ACM, New York, NY, USA (2014), http://doi.acm.org/10.1145/2691352.2691353
3. Cheruvelil, K.S., Soranno, P.A., Weathers, K.C., Hanson, P.C., Goring, S.J., Filstrup, C.T., Read, E.K.: Creating and maintaining high-performing collaborative research teams: the importance of diversity and interpersonal skills. Frontiers in Ecology and the Environment 12(1), 31–38 (2014), http://0-dx.doi.org.innopac.up.ac.za/10.1890/130001
4. Ferrucci, F., Harman, M., Ren, J., Sarro, F.: Not going to take this anymore: Multi-objective overtime planning for software engineering projects. In: Proceedings of the 2013 International Conference on Software Engineering. pp. 462–471. ICSE '13, IEEE Press, Piscataway, NJ, USA (2013), http://0-dl.acm.org.innopac.up.ac.za/citation.cfm?id=2486788.2486849
5. Gibbs, G.: Learning by Doing: A Guide to Teaching and Learning Methods. Far Eastern University Publications, Manila (1988)
6. Joint Task Force on Computing Curricula ACM/IEEE: Computer science curricula 2013: Curriculum guidelines for undergraduate degree programs in computer science. http://dl.acm.org/citation.cfm?id=2534860 (January 2013)
7. Liebenberg, J., Pieterse, V.: Career goals of software development professionals and software development students. In: Proceedings of the Computer Science Education Research Conference. p. ?? CSERC '16, ACM, New York, NY, USA (2016)
8. Marock, C.: Grappling with youth employability in South Africa. Tech. rep., Human Sciences Research Council, Pretoria (2008)
9. Marshall, L., Pieterse, V., Thompson, L., Venter, D.M.: Exploration of participation in student software engineering teams. ACM Transactions on Computing Education (TOCE) 16(2), 5:1–5:38 (Feb 2016), http://doi.acm.org/10.1145/2791396
10. Omeleze, S., Pieterse, V., Solms, F.: Teaching modular software development and integration. In: 6^{th} Annual International Conference on Computer Science Education: Innovation & Technology. pp. 178 – 197. GSTF (2015)
11. Parker, J.N., Hackett, E.J.: Hot spots and hot moments in scientific collaborations and social movements. American Sociological Review 77(1), 21–44 (2012)
12. Pieterse, V., Thompson, L.: Investigating the applicability of Belbin Roles on participatory levels in IT student teams. In: Proceedings of the 44^{th} annual conference of the Southern African Computer Lecturers' Association (SACLA). pp. 161 – 169. SACLA Organising Committee, University of the Witwatersrand, Johannesburg (2015)
13. Schank, R.C.: What we learn when we learn by doing. Tech. Rep. Technical Report No. 60, Northwestern University, Institute for Learning Sciences (1995)
14. Stokes, P.D., Fisher, D.: Selection, constraints, and creativity case studies: Max Beckmann and Philip Guston. Creativity Research Journal 17, 283 – 291 (2005)
15. Tziner, A., Eden, D.: Effects of crew composition on crew performance: Does the whole equal the sum of its parts? Journal of Applied Psychology 70(1), 85 – 93 (1985)
16. Woolley, A.W., Chabris, C.F., Pentland, A., Hashmi, N., Malone, T.W.: Evidence for a collective intelligence factor in the performance of human groups. Science 330(6004), 686–688 (2010)