Received: December 5, 2018. Revised: June 12, 2019. Accepted: June 26, 2019. Published: July 1, 2019.

Assessing the Human Factors in Software Development Courses Students Project

Pınar Cihan
Dept. of Computer Engineering
Yıldız Teknik University
İstanbul, Turkey
pkaya@yildiz.edu.tr

Oya Kalıpsız
Dept. of Computer Engineering
Yıldız Teknik University
İstanbul, Turkey
kalipsiz@yildiz.edu.tr

***Abstract* —This paper outlines evaluating questionnaire about the student projects in software development courses. This questionnaire performed *Software Engineering*, *System Analysis*, and *Agile Software Development* courses in the Department of Computer Engineering. Two different universities and 189 students attended the questionnaire. We prepared this questionnaire to evaluate students' technical skills and human factor. In this study, the human factors in software development courses students project evaluated, such as documentation, team communication, project difficulties and evaluation of project from students' perspective. The purpose of this study identify the students project difficulties in project based courses to preparing students for professional life and provide better quality education in software development courses.**

Keywords—software engineering education; software development; quality assurance in education; human factors

I. INTRODUCTION

Software engineering education is an engaging issue, and a large number of studies have been conducted in this area [1, 29]. Some studies related to educational programs on software engineering [3, 4]. Some of the studies have taken to the fore the most appropriate way to prepare professional life of software developers [5]. Education of future software engineers was examined [6, 7]. Today teaching a course in software engineering involves doing a project at the end of the course. Some studies are related teaching software engineering project-based method. Such as D. Dahiya studied on "Teaching Software Engineering: A Practical Approach", this paper presents a method to teach Software Engineering using the applied approach that the author designed and successfully used [8]. E. Sventekova et al, studied on "Project-based Teaching, Practice in the Academic Environment", young people should learn these soft skills at the university and increasing student´s interest in participating in research and development, while studying at university and apply the latest knowledge from experience in education [9]. M. Gnatz et al, studied on "A Practical Approach of Teaching Software Engineering", article reports experiences with the concept of a course focusing on providing practical know-how [10]. Dr. Alan R. Peslak studied on "Teaching Software Engineering Through Collaborative Methods", this paper outlines some of collaborative techniques and approaches used in the course [11]. E. O. Navarro et al, studied on "Teaching Software Engineering Using Simulation Games", they developed a pair of games for teach about the software process in software engineering [12]. R. E. K. Stirewalt studied on "Teaching Software Engineering Bottom Up", they developed a new course which incorporates this "bottom up" coverage. Using this method, we are able to instill a higher level of cognitive ability in software-engineering methods than we were able to achieve using the old method [13].

The industry claims that the software engineering graduates are not able to meet their requirements in software industry. In many of the studies observed that; students who graduate from universities inexperienced, communication, collaboration, creative and directing skills namely human factor(soft skills) are not a good [9, 14, 15]. Industries spend time and money for the acquisition of these skills to fresh graduates. For this reason some studies have emphasized the importance of software engineering education, to learn the business environment [10, 16]. Collaboration between the Software Engineering Education and Software Engineering Industry, can gain some benefits including the fresh graduates [17-20].

In this study, we apply questionnaire to students. Questionnaire related to software development courses projects. 189 students of in two different universities participated in these questionnaire and we evaluate result for find difficulties of the projects. We evaluate student projects difficulties because of provide high quality education, training, and educate students.

In the next section we briefly describe software engineering education (see section II). Then software development courses student projects evaluated (see section III). Finding will be explained (see section IV). Conclude the paper with future work (see section V).

II. SOFTWARE ENGINEERING EDUCATION

Software engineering education is traditionally performed with the courses which are Computer Science and Computer Engineering departments of universities.

These programs has a course called "Software Engineering", in some programs in addition to this programs include a variety of software engineering issues courses. In many

universities around the world, was created Software Engineering Undergraduate Programs [21].

Which is an international professional organizations Association for Computing Machinery (ACM) and the IEEE Computer Society (IEEE CS) created "Software Engineering Coordinating Committee" to speed up the ripening process of software engineering in the recent past and started various projects in the field of software engineering [22].

Between 1998 and 2001 a joint committee of ACM and IEEE-CS developed a set of curriculum guidelines for programs in computer science. This event called Computing Curricula and in this event studies were conducted and reports were prepared for each subject about computer Science, Computer Engineering, Software Engineering and Information Systems fields [23]. Group related software engineering curriculum of the committee defined graduates of an undergraduate software engineering program must be able to

- Working as part of a team to develop software products,
- Identify user requirements and translate them to software requirements,
- Reconcile conflicting project objectives, finding acceptable compromises within limitations of cost, time, knowledge, existing systems, and organizations
- Understand and apply the models, existing theories and techniques for software design, development, implementation, and verification.
- Work effectively in a typical software development environment, be able to leadership when necessary, and better communicate with users.
- Design appropriate solutions in one or more application domains using software engineering approaches that integrate ethical, social, legal, and economic.
- Learn new models, techniques, and technologies as they emerge.

Considering the problems of the future of software engineering education, to consider the preparing of present-day educational programs can be summarized as follows [24]:

- The preparation of the programs will be attractive to students,
- The most effective way to focus on education,
- More active communication with the industry,
- Identification of educational programs for the future,
- Education performed according to the conditions of the present students,
- Acceptance of the need for basic infrastructure for software engineering education,
- Improve the quality and reputation of research in software engineering education.

A. Structure of Questionnaire

Two different universities and totally 189 student participated questionnaire. In *Software Engineering* course at Yıldız Teknik University 70 students, in *System Analysis* course at Yıldız Teknik University 86 students, and in *Agile Software Development* course at Namık Kemal University 33 students are attend these questionnaire. All of these courses are in Computer Engineering Department. System Analysis, Software Engineering, and Agile Software Development courses are respectively, 2, 3, and 4 classrooms. Projects in the courses are carried out one semester. Project teams ranged between 4 and 7 according to courses. Projects topics and team members are determined by the students. Questionnaires consist of 20 questions and two parts. First part observed students' human factor, second part observed students' technical skills.

The human factors in software engineering, such as to document code and its rationale and history, to follow a software methodology, to manage a large project, and to work with others on a software team, are less well supported in university pedagogy.

The technical skills have been driven by advances in professional life, such as, development technics (object-oriented diagrams, structure diagrams, ER diagrams) and methodologies (Agile, Waterfall, Spiral, Extreme Programming).

In this study we analyzed use of documentation, and to work with others on a software team (team communication), projects difficulties, and compare project grade average with project evaluation from student perspective.

B. Pre-processing of Questionnaires and Methods

First of all, the questionnaire papers transferred to the electronic environment. Before evaluating the questionnaire was preprocessing which question is unanswered or marked more than one option. If there are a lot of unanswered question, this questionnaire will be ignore. Unanswered or marked more than one option question is missing data for us. If there is missing data, the highest mean average result written that instead of missing data. Then the data were statistically evaluated and the results are given below.

III. FINDINGS

Figure 1 show, what extent the use of documentation from student in projects. This documentation contains the used process and its reports. Software engineering activities require the ability to prepare a document [25]. However software engineers are extremely weak in this regard [26]. Indeed, analysis of the questionnaire results and reports in the process of using a large extent rate is only is 55%. Use of documentation is very important for this reason this ratio is not sufficient. Because missing or incorrect documentation in software engineering, can lead to significant deficiency in terms of productivity and quality of software development [27]. The people who work in this area highlight, these abilities as important as at least technical skills in group work [28].

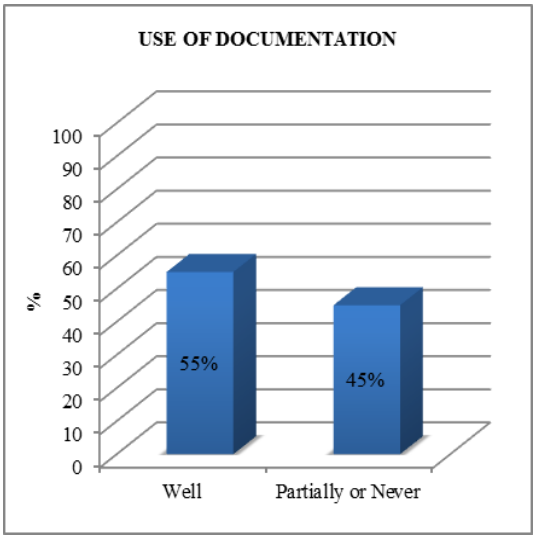

Fig. 1. Using of Documentation Rating

Figure 2 shows the team communication. The results are analyzed; about half of the students do not find a good team communication. This ratio is rather high. Because team communication is an important location in a sector and it is one of the causes of failure. Studies stated that communication and cooperation of fresh graduates does not well, so these skills should be improved [14, 15]. In addition, many studies focused on software engineers need to be educated in order to be a good member of a group [25, 29]. T-test was used to analyzed the statistically significant relation between using of documentation and success of the student project. The t-test revealed that the $p \cong 0.000$, $p < 0.05$. So this result shows using of documentation affect success of projects.

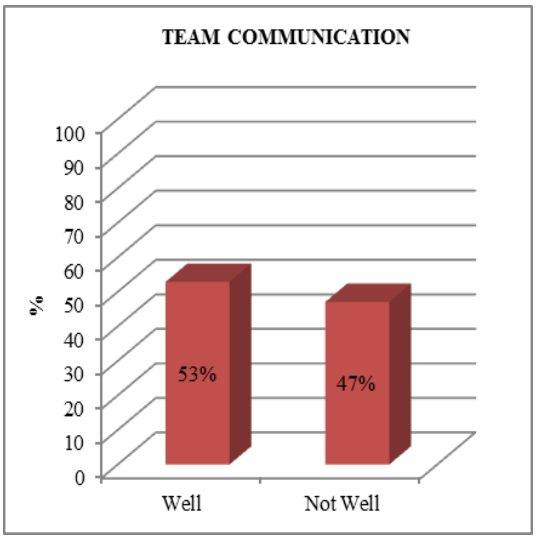

Fig. 2. Team Communication in the Projects

Figure 3 shows that, the most difficulty fields in the project from student perspective. The results are analyzed, almost half of the students have difficulty due to tight deadline and member of the team does not work enough. Such a result come out is fairly consistent. Because the projects are carried out in a semester, and during this period there are projects and exams in the other courses. According to the study, new college graduates in their first software development job and see that, they should fix and bug the "right" way, even if they do not have time for it was observed [14]. In addition, they want to use a the new technology in an organization the difficulty is not adapt to new technology, problem due to project deadlines got tight [30]. T-test was used to analyzed the statistically significant relation between team communication and success of the student project. The t-test revealed that the $p \cong 0.000$, $p < 0.05$. So this result shows team communication affect success of projects.

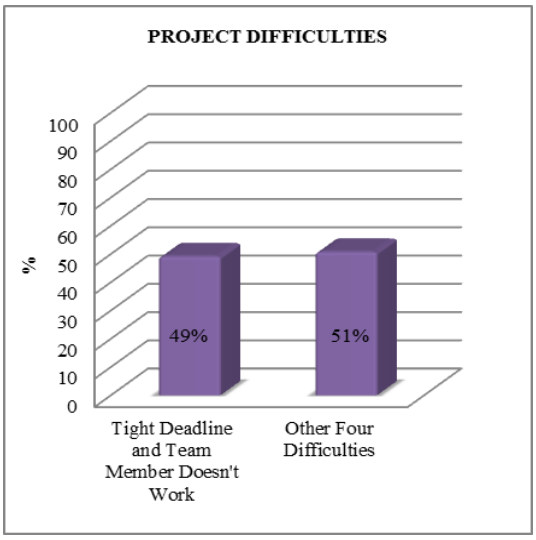

Fig. 3. Difficulties of the Project Rating

Figure 4 shows, evaluation of the projects from students' perspective. Result show that only 31% of the students were found well their project. In 1995, a study conducted by the Standish Group successful projects average is 16% [31]. In 2009, a study conducted by the Standish Group successful projects average is 32% [32]. In 2012, a study conducted by our successful projects average is 31%. Considering the results obtained by other, this low successful rate is not surprising. Because tight deadline, member of the team does not work enough, and team work issues are cause satisfactory product is not obtained at the end of the process. During software development, relationships with each other developer in the community affect the quality of the whole software system. T-test was used to analyzed the statistically significant relation between difficulties of the project and success of the student project. The t-test revealed that the $p \cong 0.000$, $p < 0.05$. So this result shows difficulties of the project affect success of projects.

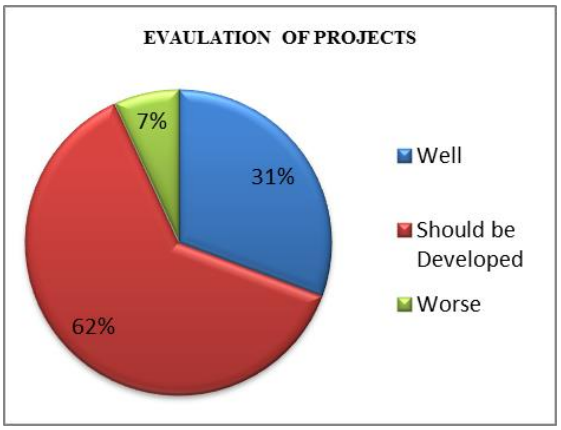

Fig. 4. Evaluation of the Projects from Students' Perspective

Figure 5 shows the comparison of the project evaluation from students' perspective with grade average of project result. Students evaluated their project at the end of the process. 88% of the student found their project well or should be developed in *Agile Software Development (ASD)* course, 93% of the student found their project well or should be developed in *Software Engineering (SE)* course, and 95% of the student found their project well or should be developed in *System Analysis (SA)* course. And their projects average grades are respectively 55%, 68%, and 79%. Analyzing show us at the end of the process the percentage of successfully project while increasing, the projects average grades are decrease. Considering the results obtained by other, this low successful rate is not surprising. Students are not giving importance to the documentation they work result oriented. So they give missing or not well reports. Therefore the students could not get enough feedback from instructor, and they think their project is successful but end of the semester they see project' grade and its surprising for student because results are not as they hoped.

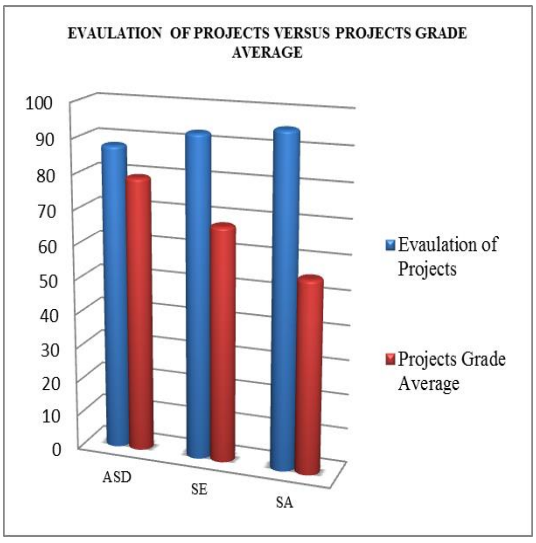

Fig. 5. Comparison of the Project Evaluation from Students Perspective with Grade Average of the Project result.

IV. CONCLUSION AND FUTURE WORK

We have applied questionnaire to 189 students about courses projects which are within in the scope of in *Software Engineering*, *System Analysis*, and *Agile Software Development* courses in computer engineering department.

In order to evaluate the results of this questionnaire, 55% of the students give importance to the documentation, 57% of the students do not reach well team communication, 49% of the students were forced to tight deadline and teammate doesn't work. Evaluation of the product obtained by the students at the end of the project, only 31% of students find well. All of these results showed that Human Factors in Software Development Courses Students Project are not good enough. Also observed in the results of t-test, human factors significantly associated with the success of the project.

The industry claims that the software engineering graduates are not able to meet their requirements in software industry. For this reason if project difficulties will decrease, quality of the course education will be increase and after improve these difficulties students satisfy the requirements of software engineering industry.

In the future, technical skills can be analyzed or questionnaire was applied to more students and then after the result can be reviewed. Also, the questionnaire can be analyzed by data mining techniques.

REFERENCES

[1] K. Hashım, and N. N. Khaıruddın, "Software Engineering Assessments and Learning Outcomes", Proceedings of the 8th WSEAS Int. Conference on Software Engineering, Parallel And Distributed Systems, February 21-23 2009.

[2] K.-M. Mira "A Method for Designing Software Engineering Educational Programs", IEEE 25th Conference On Software Engineering Education And Training (CSEE&T), Nanjing, Peoples R China, April 17-19 2012, pp. 139-143.

[3] J.-G. Schneider, L. Johnston, and P. Joyce, "Curriculum development in educating undergraduate software engineers-are students being prepared for the profession?", Software Engineering Conference, 2005. ASWEC '05 Proceeding of Conference on Software Engineering, Australian, 29 March-1 April 2005, pp. 314- 323.

[4] A. Abran, P. Bourque, R. Dupuis, and L. L. Tripp, "Guide to the Software Engineering Body of Knowledge", IEEE, Los Alamitos, California, 2004.

[5] Z. Aizamil, "Towards an Effective Software Engineering Course Project", ICSE'05 Proceedings 27th International Conference on Software Engineering, St. Louis, Missouri, USA, May 15-21, 2005, pp. 631-632.

[6] P. Ciancarini, "On the Education of Future Software Engineers", ICSE'05 Proceedings 27th International Conference on Software Engineering, St. Louis, Missouri, USA, May 15-21, 2005, pp. 649- 650.

[7] T.B. Hilburn, and W. S. Humphrey, "The Impending Changes in Software Education.", IEEE Software, vol. 19, Sep/Oct2002, pp. 22-24.

[8] D. Dahiya, "Teaching Software Engineering : A Practical Approach", ACM SIGSOFT Software Engineering Notes, vol. 35, March 2010, pp. 1-5.

[9] E. Sventekova, and T. Lovecek, "Project-based Teaching, Practice in the Academic Environment", Proceedingd of the 11th WSEAS International Conference on Education and Educational Technology, May 11-13 2012.

[10] M. Gnatz, L. Kof, F. Prilmeier, and T. Seifert "A Practical Approach to Teaching Software Engineering", 16th Conference on SE Education and Teaching (CSEET) 2003. CSEET'03 Proceedings of the 16th Conference on Software Engineering Education and Training, 20-22 March 2003, p. 120.

[11] A. R. Peslak, "Teaching Software Engineering Through Collaborative Methods", Issues in Information Systems, vol. 5, 2004, pp. 247-253.
[12] E. O. Navarro, A. Baker , A. van der Hoek "Teaching Software Engineering Using Simulation Games", International Conference on Software Engineering 2004.
[13] R. E. K. Stirewalt, "Teaching Software Engineering Bottom Up", American Society for Engineering Education Annual Conference 2004, Proceedings of the 2004 American Society for Engineering Education Annual Conference & Exposition, 2004.
[14] A. Begel, and B. Simon, "Struggles of New College Graduates in their First Software Development Job", SIGCSE'08, Portland, Oregon, USA, March 12-15 2008.
[15] A. Begel, and B. Simon, "Novice Software Developers, All Over Again", ICER'08, Sydney, Australia, September 6-7 2008.
[16] R. T. Yeh, "Educating Future Software Engineers", IEEE Transactions on Education, vol. 45, Feb 2002, pp. 2-3.
[17] I. Garcia, C. Pacheco, and N. Coronel, "Learn from Practice: Defining an Alternative Model for Software Engineering Education in Mexican Universities for Reducing the breach between Industry and Academia", Proceedings of the International Conference on Applied Computer Science, September 15-17 2010.
[18] S. Karunasekera, and K. Bedse, "Preparing Software Engineering Graduates for an Industry Career", Proceedings of 20th Conference on Software Engineering Education & Training (CSEET'07), IEEE Computer Society, July 2007, pp. 97-106.
[19] G. V. B. Subrahmanyam, "A Dynamic Framework for Software Engineering Education Curriculum to Reduce the Gap between the Software Organizations and Software Educational Institutions", Proceeding of 22nd Conference on Software Engineering Education and Training (CSEET), Feb. 2009, pp. 248-254.
[20] Z. Mahmood, "A Framework for Software Engineering Education : A Group Projects Approach", International Journal of Education and Information Technologies – Powered GoogleDoc Journals, vol. 1, March 2007, pp. 153-156.
[21] "Computing Curricula" - Software Engineering Volume", http://sites.computer.org/ccse/
[22] A . Abran, and J. Moore, "Guide to the Software Engineering Body of Knowledge SWEBOK". IEEE Computer Society Press, 2001.
[23] "Computing Curricula", http://www.computer.org/education/cc2001/
[24] G.W. Hislop, "Software Engineering Education: Past, Present and Future", in Software Engineering Effective Teaching and Learning Approaches and Practices, Information Science, 2009.
[25] M. R. Heil, "Preparing Technical Communicators for Future Workplaces: A Model that Integrates Teaming, Professional Communication Skills, and a Software Development Process" SIGDOC'99 Proceedings of the 17th annual international conference on Computer documentation , 1999, pp. 110-119.
[26] M. B. Blake, "A Student-Enacted Simulation Approach to Software Engineering Education", IEEE Transactions On Educatıon, vol. 46, February 2003.
[27] M. Cataldo, M. Bass, J. D. Herbsleb, and L. Bass, "Managing Complexity in Collaborative Software Development: On the Limits of Modularity", CSCW Workshop '06, Banff, AB, November 2006, p.17.
[28] D. Rosca, "Multidisciplinary and Active/Collaborative Approaches in Teaching Requirements Engineering", European Journal of Engineering Education, vol. 30, March 2005, pp. 121–128.
[29] V. M. Teles, and C. E. T de Oliveira "Reviewing the Curriculum of Software Engineering Undergraduate Courses to Incorporate Communication and Interpersonal Skills Teaching", Proceedings of the 16th Conference on Software Engineering Education and Training (CSEET'03), 20-22 March 2003, pp. 158-165.
[30] R. E. Ginter, "Using a Configuration Management Tool to Coordinate Software Development", Proceedings of conference on Organizational computing systems, Milpitas, California, United States, 1995, p. 175.
[31] Standish Group, International, The CHAOS report 1995.
[32] Standish Group, International, The CHAOS report 2009.