Evaluation of Students' Skills in Software Project

Pınar Cihan[1], Oya Kalıpsız[1]

[1]*Yıldız Technical University, İstanbul, Turkey*

***Abstract* – Software project probably is a sector that has witnessed the highest rate of project failure in the world. The industry claims that the software engineering graduates are not able to meet their requirements in software industry. This is surprising to the academia that offers software engineering specialization. This type of situation is creating the barrier between the software industry and academics and the efforts are made to reduce the gap. So it is important identify weaknesses of projects. This paper outlines evaluating questionnaire about the student projects in software development courses student projects. The questions were intended to test students' soft skills and hard skills. This questionnaires performed Software Engineering, System Analysis and Design, and Agile Software Development courses in the Department of Computer Engineering. All of these courses are project based learning. In this study, soft skills and hard skills in software development courses students project evaluated. The purpose of this study identify the students project weaknesses in project based courses to provide better quality education in software development courses, preparing students for professional life, improvement of the success rate of software projects and possibly lead to reduce gap between academy and industry.**

***Keywords* – Software engineering education, Software engineering industry, Quality assurance in education, Soft skills, hard skills.**

1. 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].

Software is present everywhere in today's world and one of the most prominent, the most challenging and the toughest technology in the new area [14]. Software project probably is a sector highest rate of project failure in the world. For this reason many studies work on project soft skills and hard skills [15-18].

Hard skills are the technical skills required within the confines of a domain. Hard skills often described as a science comprises processes, tools and techniques applied to projects [19].

Soft skills are the non-technical skills. Often described as an art [19], Soft skills have been identified as critical for project success. The following soft skills are crucial for successful project: communication, organizational effectiveness, leadership, problem solving and decision-making, team building, flexibility and creativity and trustworthiness.

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 soft skills are not a good [9, 17, 20, 21]. 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, 22]. Collaboration between the Software Engineering Education and Software Engineering Industry, can gain some benefits including the fresh graduates [23-26].

In this study, questionnaire applied 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.

The aim of this study, identify the students project weaknesses in project based courses to provide better quality education in software development courses, preparing students for professional life, improvement of the success rate of software projects and possibly lead to reduce gap between academy and industry.

In the next section we briefly describe software engineering education and software engineering industry (see section II). Finding will be explained (see section III). Conclude the paper with future work (see section IV).

2. Software engineering education and software engineering industry

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 [27].

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 [28].

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 [29]. 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 [30]:

- 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.

Software industry, software education and research institutes are closely associated.
Beckman et. al. summarized below the key benefits of cooperation between universities and industry [31].

Industry benefits:

Industry cooperation and support have many advantages for industry.

- Cost-effective, customized education and training.
- Influence on academic programs.
- Access to university research in new software engineering methods, tools, and technology to improve competitiveness.
- New revenue sources

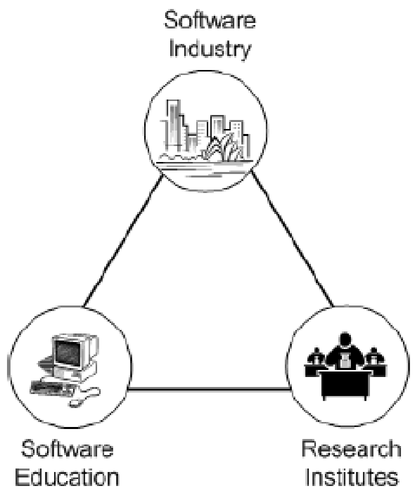

Figure 1. Association with Software industry, software education and research institutes [25].

Academic benefits:

Industry cooperation and support have many advantages for universities.

- Placement of students
- Insight into corporate issues at the applied, practical level
- Research and continuing education opportunities for faculty.
- Special funding from a corporate partner to the cooperating university.

2.1. *Structure of questionnaire*

Two different universities and totally 189 student participated questionnaire. In Software Engineering course at Yıldız Technical University 70 students, in System Analysis and Design course at Yıldız Technical 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 and Design, 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' soft skills, second part observed students' technical skills.

The soft skills 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 hard 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).

2.2. *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. Replace Missing Value used for data preprocessing which is the Weka tool [32]. Then the data were statistically analysis and the results are given below.

3. Findings

3.1. *Evaluation of Soft Skills*

All soft skills questions are joint Software engineering, System Analysis and Design and Agile Software Development courses questionnaire.

Figure 2 shows the 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 [33]. However software engineers are extremely weak in this regard [34]. 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.

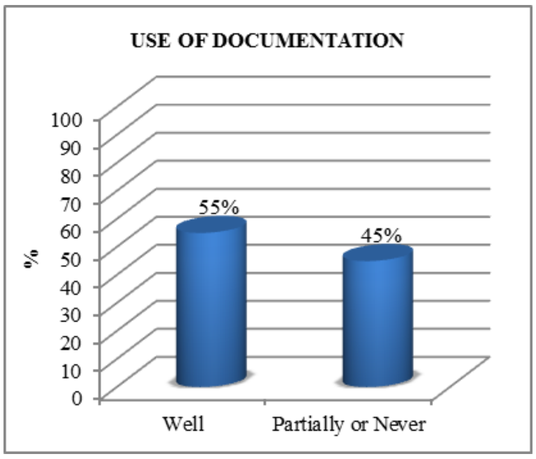

Figure 2. Using of documentation rating [17].

Missing or incorrect documentation in software engineering, can lead to significant deficiency in terms of productivity and quality of software development [35]. The people who work in this area highlight, these abilities as important as at least technical skills in group work [36]. T-test was used to analyze the statistically significant relation

between using of documentation and success of the student project. The t-test revealed that the p≅0.000, $p < 0.05$. So this result shows using of documentation affect success of projects.
Figure 3 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 [20, 21]. In addition, many studies focused on software engineers need to be educated in order to be a good member of a group [33, 37]. T-test was used to analyze the statistically significant relation between team communication and success of the student project. The t-test revealed that the p≅0.000, $p < 0.05$. So this result shows team communication affect success of projects.

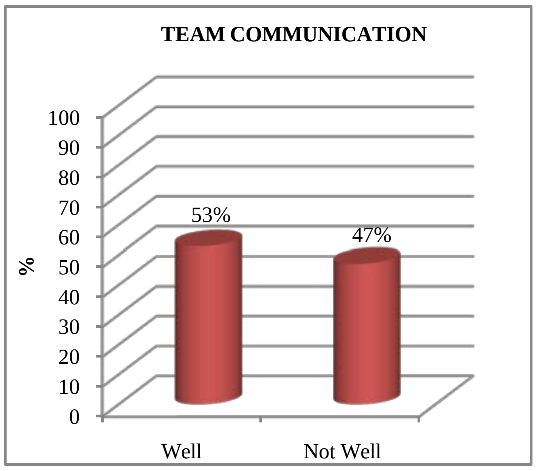

Figure 3. Team communication in the projects [17].

Figure 4 shows the difficulties of project fields in the project from student perspective. The results are analyzed, 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 [20]. 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 [38]. T-test was used to analyze the statistically significant relation between difficulties of the project and success of the student project. The t-test revealed that the p≅0.000, $p < 0.05$. So this result shows difficulties of the project affect success of projects.

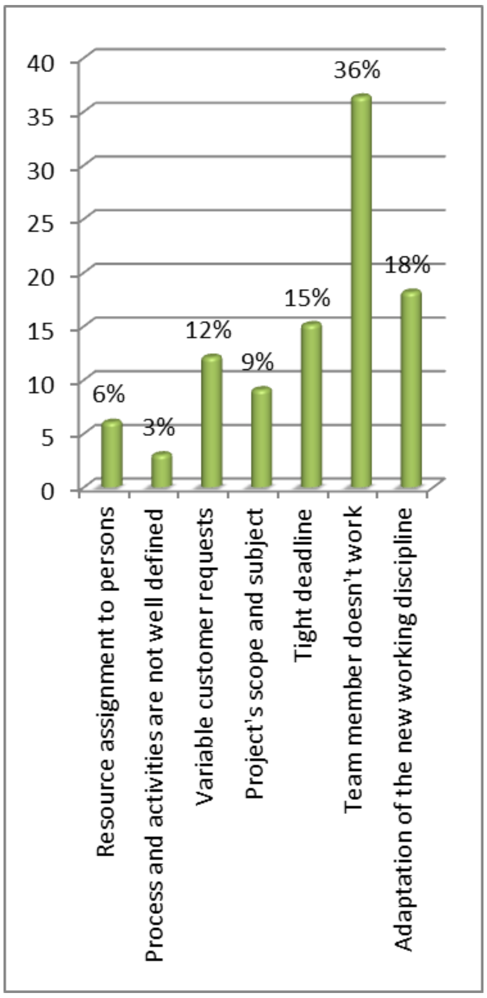

Figure 4. Difficulties of the project rating.

Figure 5 shows the evaluation of the projects from students' perspective. Result show that only 31% of the students were found well their project.

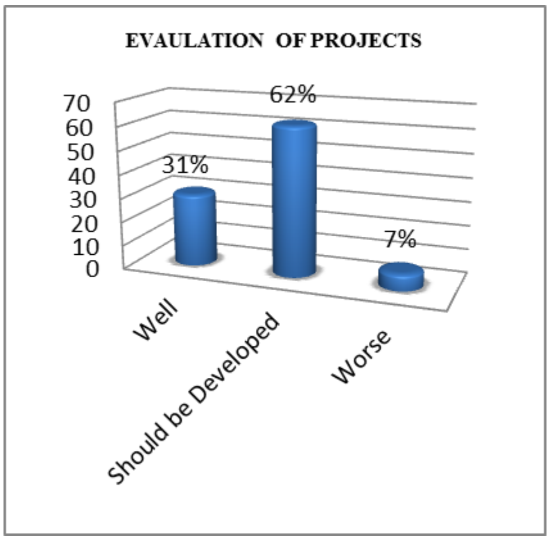

Figure 5. Evaluation of the projects from students' perspective [17].

In 1995, a study conducted by the Standish Group successful projects average is 16% [39]. In 2009, a study conducted by the Standish Group successful projects average is 32% [40]. 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.

3.2. Evaluation of Hard Skills

All course content is different. So each technical skills question are prepared base of course. For this reason, hard skills question are evaluated separately.

Figure 6 shows the percentage of using the Entity relation, Data Flow and Structure Diagrams in System Analysis and Design course projects during the application. Data flow diagrams and entity relation diagrams are main indicators of software engineering. Software engineers use these diagrams to understand, develop and evaluate the information system [41]. Therefore these diagrams applied a large part of the project. After the evaluating of questionnaire; 71% of students use Data flow and entity relation diagrams, 23% of students use these diagrams partially, and 6% of students not use these diagrams in their projects.

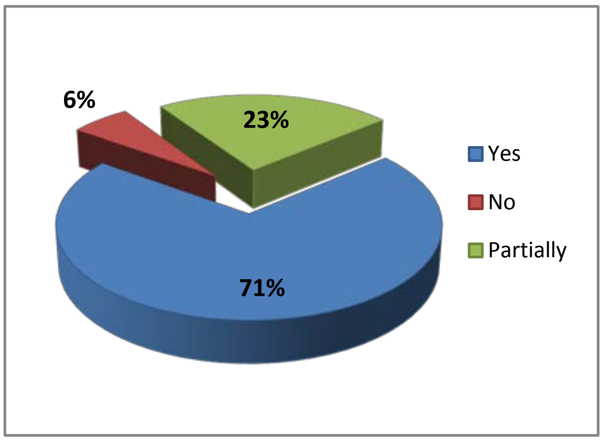

Figure 6. Percentage of using the entity relation, data flow and structure diagrams in system analysis and design course projects during the application.

Figure 7 shows the students' ideas about during the applicability of software given only data flow diagram in System Analysis and Design course project. Although there is no previously knowledge of the application of the system, data flow diagrams show us application of process operations and the flow of information between the resources. 65% of students answered difficult but applicable. This result is consistent.

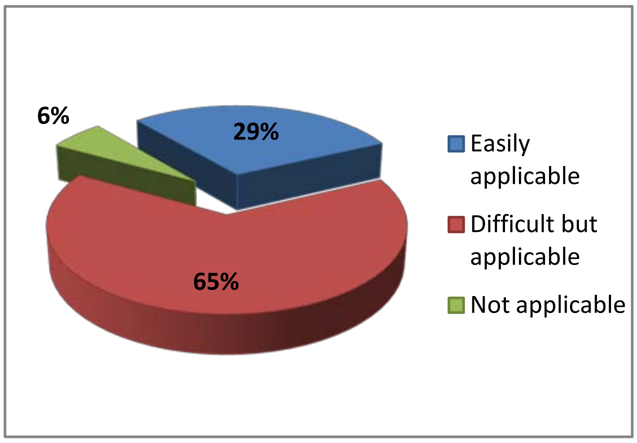

Figure 7. Thoughts about the applicability of the software given only data flow diagram in System Analysis and Design course project.

Figure 8 shows the most appropriate diagram in terms of intelligibility rates for students in software engineering course project. Class diagram is the most basic diagram [42]. Class diagrams are extremely simple structures in their own right. [43]. For this reason intelligibility is more than the other diagrams. In fact, 74% of the students think that the most appropriate diagram is class diagram in terms of intelligibility.

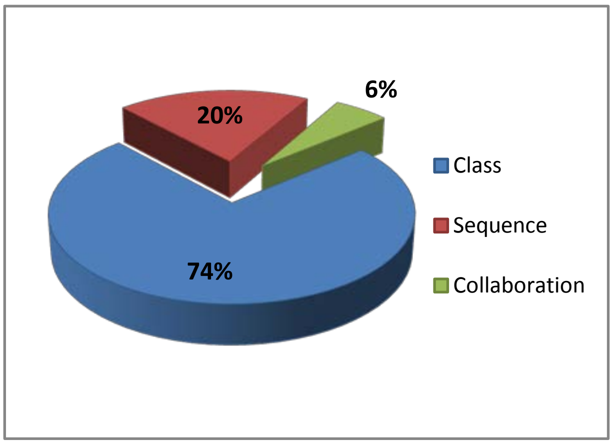

Figure 8. The most appropriate diagram in terms of intelligibility rates for students in software engineering course project.

Figure 9 shows the useful diagram for expression modelling and design in software engineering course project. Class diagram is a central object-oriented method, also class diagram is used to refer modeling and design [44]. Large majority of students (72%) thinks that the most useful diagram is class diagram for expressing modelling and design.

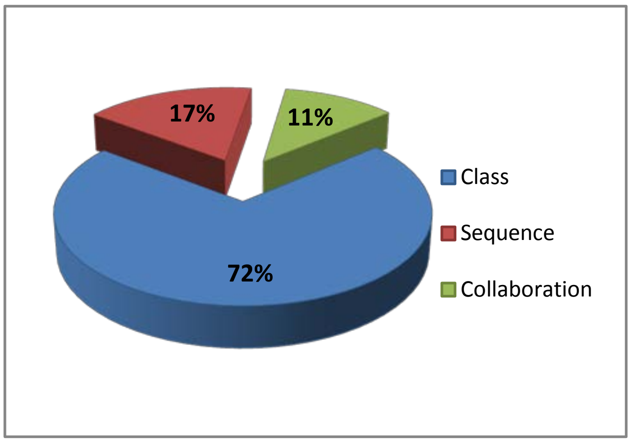

Figure 9. Useful diagram for expression modelling and design in software engineering course project.

Figure 10 shows the created first diagram of students in software processes in Agile Software Development course projects. 76% of the student created firstly use case diagram. Earlier as mentioned, the software engineer firstly draws the use case diagram. In this case the answers are consistent.

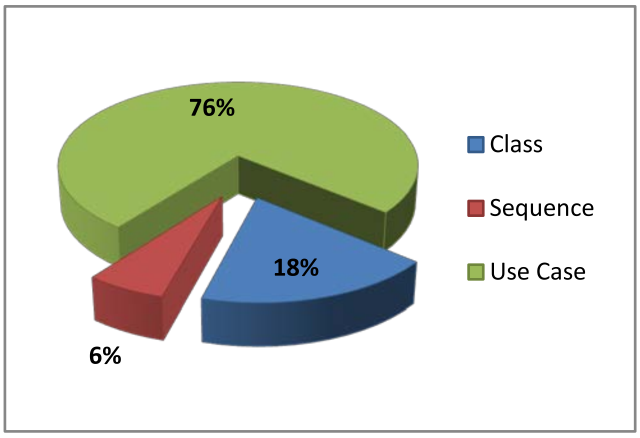

Figure 10. Created first diagram of students in software processes in agile software development course projects.

Figure 11 shows the more dominant difference for students about agile methods compared with the other previous projects.

Agile software development course project is test driven project. For this reason agile software project is different from than software engineering and system analysis courses projects. Indeed 61% of the students think that the dominant difference test.

Figure 12 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 and Design (SAD) course.

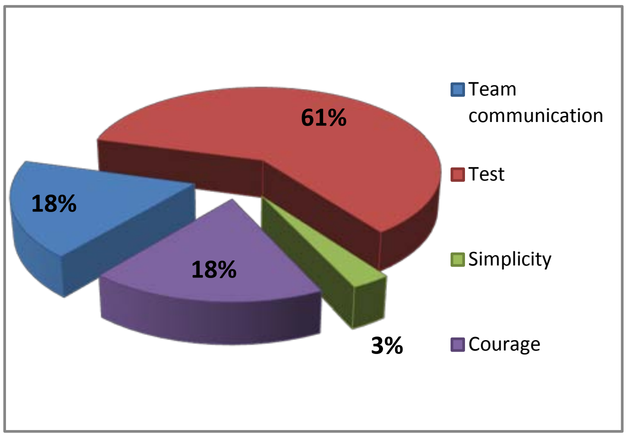

Figure 11. More dominant difference for students about agile methods compared with the other previous projects.

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.

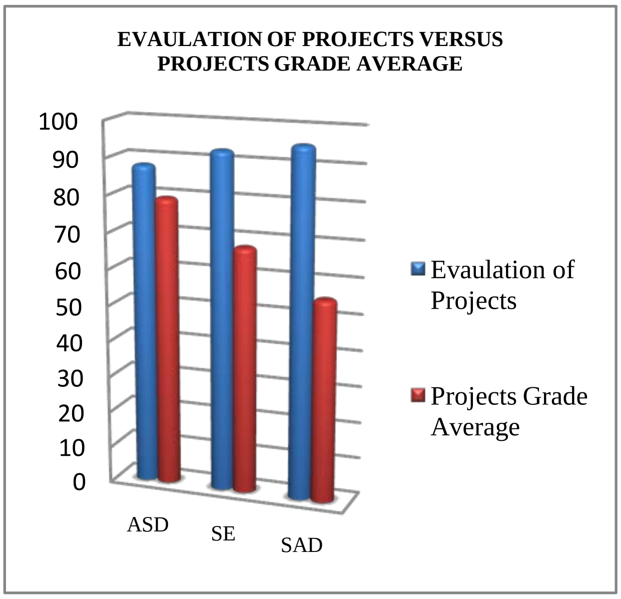

Figure 12. Comparison of the project evaluation from students perspective with grade average of the project result [17].

4. Conclusion and future work

Questionnaire applied to 189 students about courses projects which are the Software Engineering, System Analysis and Design, and Agile Software Development courses in computer engineering department.

In this paper evaluate the students' skills in software project. Result shows that the students' hard skills better than soft skills in software project. Also observed in the results of t-test, soft skills significantly associated with the success of the project. The soft skills deemed important for software projects.

The industry claims that the software engineering graduates are not able to meet their requirements in software engineering industry. For this reason soft skills very important and will be improve at university.

Improving of soft skills lead to, quality of course projects will be increase and students satisfy the requirements of software engineering industry.

In the future, questionnaire applied to industry and compare with the students answers. Also, the questionnaire can be analyzed by data mining techniques.

References

[1]. K. Hashım, N. N. Khaıruddın, *Software Engineering Assessments and Learning Outcomes*, Proceedings of the 8th WSEAS Int. Conference on Software Engineering, (2009).

[2]. K.-M. Mira, *A Method for Designing Software Engineering Educational Programs*, IEEE 25th Conference On Software Engineering Education And Training (CSEE&T), 139-143, (2012).

[3]. J.-G. Schneider, L. Johnston, P. Joyce, *Curriculum development in educating undergraduate software engineers-are students being prepared for the profession?*, ASWEC '05 Proceeding of Conference on Software Engineering, 314- 323, (2005).

[4]. A. Abran, P. Bourque, R. Dupuis, 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, 631-632, (2005).

[6]. P. Ciancarini, *On the Education of Future Software Engineers*, ICSE'05 Proceedings 27th International Conference on Software Engineering, 649-650, (2005).

[7]. T.B. Hilburn, W. S. Humphrey, *The Impending Changes in Software Education*, IEEE Software, Vol. 19, 22-24, (2002).

[8]. Dahiya, D. (2010). Teaching software engineering: a practical approach. *ACM SIGSOFT Software Engineering Notes*, *35*(2), 1-5.

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

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

[11]. A. R. Peslak. (2004). Teaching Software Engineering Through Collaborative Methods, *Issues in Information Systems*, *5*, 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*, Proceedings of the 2004 American Society for Engineering Education Annual Conference & Exposition, (2004).

[14]. R. Mahanti, P. K. Mahanti, *Software Engineering Education From Indian Perspective*, Proceeding of the 8th Conference on Software Engineering Education & Training (CSEET'05), IEEE Computer Society, 111-117, (2005).

[15]. A. Sukhoo, A. Barnard, M. M. Eloff, J. A. Van der Poll, M. Motah, *Accommodating Soft Skills in Software Project Management*, Issues in Informing Science and Information Technology, Vol. 2, 691-704. (2005).

[16]. A. Baki, S.H. Hamzah, C.M. Mat Isa, R. Md Sem, M.A. Yahaya, F. Mohamad Yusof, A.H. Hassan, S. Abd Rahim, *Improvement of Students' Soft-Skills Through University-Industry Collaborations*, Proceedings of the 7th WSEAS International Conference on on Education and Educational Technologies (EDU'08), (2008).

[17]. P. Cihan, O. Kalıpsız, *Assessing the Human Factors in Software Development Courses Students Project*, International Conferance on Education and Educational Technologies, (2013).

[18]. R. A Rashid, R. Abdullah, A. Zaharim, H. A. Ghulman, M. S. Masodi, *Engineering Students Performance Evaluation of Generic Skills Measurement: ESPEGS Model*, 5th WSEAS / IASME International Conference on Engineering Education (EE'08), (2008).

[19].K. Belzer, Project management: Still more art than science, http://www.pmforum.org/library/papers/BusinessSuccess.htm

[20]. A. Begel, B. Simon, *Struggles of New College Graduates in their First Software Development Job*, SIGCSE'08, (2008).

[21]. A. Begel, B. Simon, *Novice Software Developers, All Over Again*, ICER'08, (2008).

[22]. Yeh, R.T.(2002).Educating future software engineers. *Education, IEEE Transactions on, 45*(1), 2-3.

[23]. I. Garcia, C. Pacheco, 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, (2010).

[24]. S. Karunasekera, K. Bedse, *Preparing Software Engineering Graduates for an Industry Career*, Proceedings of 20th Conference on Software Engineering Education & Training (CSEET'07), IEEE Computer Society, 97-106. (2007).

[25]. 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), 248-254, (2009).

[26]. Mahmood, Z. (2007). A framework for software engineering education: a group projects approach. *International Journal of Education and Information Technologies–Powered GoogleDoc Journals*, *1*(3), 153-156.

[27]. Computing Curricula - Software Engineering Volume, http://sites.computer.org/ccse/

[28]. Abran, A., Bourque, P., Dupuis, R., & Moore, J. W. (2001). *Guide to the software engineering body of knowledge-SWEBOK*. IEEE Press.

[29]. Computing Curricula.(2001). http://www.computer.org/education/cc2001/

[30]. G.W. Hislop, *Software Engineering Education: Past, Present and Future*, in Software Engineering Effective Teaching and Learning Approaches and Practices, Information Science, (2009).

[31]. Beckman, K., Coulter, N., Khajenoori, S., & Mead, N. R. (1997). Collaborations: closing the industry-academia gap. *IEEE software*, *14*(6), 49-57.

[32]. Data Mining Software in Java, http://www.cs.waikato.ac.nz/ml/weka/

[33]. 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 , 110-119, (1999).

[34]. Blake, M. B. (2003). A student-enacted simulation approach to software engineering education. *Education, IEEE Transactions on*, *46*(1), 124-132.

[35]. M. Cataldo, M. Bass, J. D. Herbsleb, L. Bass, *Managing Complexity in Collaborative Software Development: On the Limits of Modularity*, CSCW Workshop '06, 17, (2006).

[36]. Rosca, D. (2005). Multidisciplinary and active/collaborative approaches in teaching requirements engineering. *European journal of engineering education,30*(1), 121-128.

[37]. V. M. Teles, 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), 158-165, (2003).

[38]. Grinter, R. E. (1995, August). Using a configuration management tool to coordinate software development. In *Proceedings of conference on Organizational computing systems* (pp. 168-177). ACM.

[39]. Standish Group, International, The CHAOS report (1995).

[40]. Standish Group, International, The CHAOS report (2009).

[41]. Song, I. Y., & Froehlich, K. (1994). Entity-relationship modeling. *Potentials, IEEE, 13*(5), 29-34.

[42]. Yazılım Geliştirmede Sistem Modelleme, http://web.itu.edu.tr/~kanoglu/crs-iscpm-systemmodeling.pdf

[43]. O. Kalıpsız, A. Buharalı, G. Biricik, Sistem Analizi ve Tasarımı, Papatya Yayıncılık, İstanbul, (2011).

[44]. Y. Kılıçaslan, *Nesneye yönelik analiz ve tasarıma giriş*, yilmazkilicaslan.trakya.edu.tr/teaching/uml_giris.ppt

Corresponding author: Pınar Cihan
Institution: Yıldız Technical University, İstanbul, Turkey
E-mail: pkaya@yildiz.edu.tr