Croatian Journal of Education
Vol: 13 (1/2011), page: 188 - 219
Review paper
Paper submitted: 1st June 2011
Paper accepted: 22nd July 2011

AN ASSESSMENT OF THE SOFTWARE ENGINEERING CURRICULUM IN TURKISH UNIVERSITIES : IEEE/ACM GUIDELINES PERSPECTIVE

Alok Mishra, Ali Yazici
Atilim University, Ankara, Turkey

SUMMARY

Software engineering (SE) education has been emerging as an independent and mature discipline. Accordingly, various studies are being done to provide guidelines for SE education curriculum design. This paper presents software engineering education evolvement in Turkey, present SE education scenario in different universities along with the significance of software technology parks in relevance to software engineering education. The objective of this paper is to provide an assessment of SE curriculum in Turkish Universities with respect to IEEE/ACM guidelines given in SEEK (2004). This study will provide a guideline to universities conducting an SE programme at undergraduate level to align their course curriculum with IEEE/ACM guidelines (SEEK, 2004).

Keywords: *Curriculum, Software Engineering Education, Software Engineering, Software Industry, Turkey*

INTRODUCTION

The ability of any large company or government organization to manage its projects and enterprises depends heavily on sophisticated software systems that support its business and technical processes, ranging from logistics to manufacturing systems and to customer relationship management systems (Pyster et al., 2009). Software Engineering (SE) deals with the creation and application of engineering fundamentals for the systematic and team-based analysis, development, use, evaluation, etc. of large, software-intensive systems as technical products (Horn and Kupries, 2003). A study by Horn and Kupries (2003) argues that there is a need for highly qualified specialists capable of mastering complex, software-intensive systems. Software engineering is the branch of computer science that creates practical, cost-effective solutions to computing and information processing problems, preferentially by applying scientific knowledge for developing software systems in the service of mankind. Here development refers to all the activities associated with a software product, from conception through to client negotiation, design, implementation, validation, operation, evolution and other maintenance (Shaw, 2005). SE is a multidimensional field that involves activities in various areas and disciplines such as computer science, project management, system architecture, human factors and technological evolution (Brazilay et al., 2009). Several efforts have been made to map the different dimensions of SE and to design a proper curriculum that addresses them all (SEEK, 2004; SWEBOK, 2008).

Software engineering rests on the following three principal intellectual foundations (Shaw, 2005):

1. Technical Foundation: It is a body of core computer science concepts relating to data structures, algorithms, programing languages and their semantics, analysis, computability, computational models etc.
2. Engineering Knowledge: Technical knowledge is applied through a body of engineering knowledge related to architecture, the process of engineering, tradeoffs and costs, conventionalization and standards, quality and assurance, etc. This provides the approach to design and problem solving that respects the pragmatic issues of the applications.
3. Social and Economic Context: This includes the process of creating and evolving artefacts, as well as issues related to policy, markets, usability

Vol: 13 (1/2011)
page: 188-219

and socio-economic impacts; this provides a basis for shaping the engineered artefacts to be suitable for their intended use.

It is common to hear the complaints of software developing companies about the practical knowledge of the students who start working right after the completion of their academic programmes. While students can have a high level of theoretical knowledge, they often lack the practice of solving real-life industrial problems. Complaints against software quality are widespread, and most depend on several factors including the low quality of teaching at a university, where essential skills are not taught at an acceptable level (Jaakkola et al., 2006). During the last decade, SE education has become widespread and is still growing. In addition, SE is the fastest-evolving engineering discipline, and most of the software development organizations' tasks are diverse in nature. The pace of changes in SE is substantially higher, broader and quicker than the impact of other disciplines and its most important feature is the ability to provide tools and methods for all areas of society, that is, engineering and engineering education (Kral and Zemlica, 2008). In this context, it is the responsibility of SE education to prepare SE professionals for the industry by providing them with skills to meet the expectations of the software industry. Innovations and improvements in the curriculum, instructions and assessment are being directed towards bridging the academia-industry gap by projecting the true nature of software development and facilitating students in nurturing essential knowledge, skills and attitude that are actually needed by the industry (Shaw et al., 2005). Many of the challenges associated with software engineering education are due to our inability to provide students with 'real world', large-scale software development experiences in an academic environment (Su et al., 2007). Therefore, the quality of the SE workforce is a direct reflection of the quality of the SE education (SEE). Studies have shown that there is a wide gap between software industry requirements and education for prospective software engineers (Beckman et al., 1997). Therefore, it is impossible to produce ready-made graduates, who can be absorbed immediately into the industry. As a solution, it is important to provide some in-house training and orientation before assigning them proper work positions. It is also important that graduating students obtain excellent exposure in different areas. Hence, if the students are well-versed in emerging technologies, in-house training duration will be shorter, which means less time

and money spent for organizations (Mishra et al., 2007). Jaakkola et al. (2006) also supported the idea that SE curriculum should correspond to the industry needs, as only then Universities can produce highly skilled professionals who can satisfy the needs of software industry. They further argued that the development of curricula should take into account different standards, frameworks and recommendations developed by interest groups.

The objective of this paper is to provide an assessment of SE curriculum in Turkish Universities with respect to IEEE/ACM guidelines mentioned in SEEK (2004). The remainder of the paper is organized as follows. In section 2, significant issues of software engineering curriculum for industrial relevance are discussed. In section 3 development of SE curriculum is stated. In section 4 SE undergraduate education in Turkey with reference to IEEE/ACM guidelines is presented, followed by curriculum implementation in section 5. Discussion is reported in section 6. Finally, the conclusions are stated in section 7.

SOFTWARE ENGINEERING CURRICULUM AND SOFTWARE INDUSTRY

There are severe problems with the quality of software and the cost of producing it (Ford and Gibbs, 1996), and often the software industry people complain about the knowledge and skills of graduates in some of key areas of SE such as: (i) software development models; (ii) requirements engineering; (iii) software architecture and high-level design; (iv) software processes; (v) software quality assurance and management; (vi) software project management (vii) managing people organizations and working in teams; (viii) software testing and so on.

These areas are directly in compliance with the phases of the software development life-cycle.

According to the observations stated in Lethbridge's study, the following topics are very important for the software industry and students could learn these during job training (Lethbridge, 2000a):

- Object-oriented concepts and technologies
- Requirements gathering and analysis
- Analysis and design methods
- Testing, verification and quality assurance
- Project management

Vol: 13 (1/2011)
page: 188-219

- Human–Computer Interaction (HCI)/user interfaces
- Databases
- Configuration and release management
- Ethics and professionalism
- Technical writing
- Delivering presentations/seminars to an audience
- Leadership skills.

According to Lethbridge (2000b), there is a need to include these subjects in a curriculum along with real industrial practice. Another important issue, along with the practical experience, are the skills needed to work in groups because in a real environment people need to work in groups. Kitchenham et al. (2005) also support several observations of Lethbridge with respect to the over-emphasis of mathematical topics and the under-emphasis of business topics. However, their findings are different from Lethbridge with respect to topics where there is a larger knowledge gap. SE students should have skills to work individually as well as on teams to develop and deliver quality software artefacts (SEEK 2004).

The selection of a suitable industry partner to provide a challenging and motivating project is critical to the success of industry-related projects. This requires that academic coordinating staff be highly motivated to support the project and conduct the preparation for the project prior to the start of each semester. Furthermore, it is important that the industry client be prepared to accept the outcome of a failed project (Hogan et al., 2005). Students usually work on projects in teams, and different teams may come up with different solutions. These solutions are discussed by the entire class with students analyzing various solutions and presenting their approach. During this process, discussions facilitate learning and communication skills. In most cases industry is interested in application-oriented activities that bring more immediate solutions, help in the implementation of new products or improve the bottom line of everyday operations (Kornecki et al., 2003). These authors observed that industry is competing for university graduates with the skills to meet the challenges of developing safety-critical software-intensive systems, as it is a prerequisite for a steady influx of qualified personnel in these areas.

An additional critical element in the success of SE programmes is the involvement and active participation of industry (SEEK, 2004). In this context,

Wohlin and Regnell (1999) presented strategies for industrially relevant education of software engineers. There are a variety of ways that industry and academia can cooperate. Many academic institutions have established departmental or other academic unit-wide industrial advisory boards. These groups, consisting of managers and engineers from industries closely associated with the academic organization, are one of the vehicles that provide feedback about an academic programme's direction (Kornecki et al., 2003). Other typical forms of cooperation are occasional short-term projects meeting the current needs of industry. Such projects allow faculty and students to become familiar with the domain and often contribute to the industrial partner's goals. Other modes of cooperation are student summer practices and faculty internship programmes. These facilitate understanding of industry needs and allow the university to make appropriate programme revisions. All these approaches can be effective but do not develop a permanent solution in this direction. Academic institutions should be able to redesign and implement SE curricula that not only emphasize theoretical and technical aspects of computing, but also focus on the practice of SE (Dey and Sobhan, 2007).

The most recent ACM model curricula (ACM, 2005) recognized the different perspectives of academia and industry. They recommended teaching subjects in SE curricula and providing technical and non-technical skills required for large software development. In this respect, it is also suggested that skills be demonstrated "to an appropriate range of applications and case studies that connect theory and skills learned in academia to real-world occurrences to explicate their relevance and utility" (ACM, 2005).

DEVELOPMENT OF SE CURRICULUM

In 1989, the Software Engineering Institute (SEI) of Carnegie Mellon University published a landmark report on graduate education in software engineering (Ardis and Ford, 1989). Graduate education is a key element in advancing the state of professional software engineering practice (Ardis et al., 2011). Recently Integrated Software & Systems Engineering (ISSEC) Curriculum project at Stevens Institute of Technology has developed a set of guidelines for master's degree programmes, entitled "Graduate Software Engineering 2009 (GSwE2009): Curriculum guidelines for graduate degree programmes in software engineering" (Pyster (Ed.), 2009). Instructional material design and

Vol: 13 (1/2011)
page: 188-219

content creation should take into account Bloom's levels of learning (Bloom et al., 1956). Since knowledge is acquired at different levels, learning modules should be constructed so as to take the learner from an elementary knowledge base of information and concepts through application and analysis of the knowledge, and finally towards synthesis and evaluation (Pinto, 2010).

According to Pinto (2010), the design of a curriculum should take the following into consideration:

1. identify job opportunities and duties involved;
2. identify personality traits (soft-skills) required for the job;
3. identify the competencies required to perform the job;
4. identify the knowledge, know-how and current skills required to achieve these competencies;
5. identify corresponding subject concepts (theory) to acquire the knowledge base required;
6. identify assignments (practice) that enhance the "know-how" of the subject;
7. identify the skill set (the means) required to perform the assignments;
8. identify the reference materials whose subject objectives match the knowledge required in items 5, 6 and 7 above;
9. design assessments that can check a candidate's knowledge at various levels for a given competency.

Although the core fundamental conceptual knowledge in IT remains the same, the advances in the technological area are swift. Therefore, the skill set that the learner attains must be what the industry currently requires. The curriculum design must be extensible enough to take this into account (Pinto, 2010). Pinto (2010) also suggests that a professionally designed curriculum would require inculcating two main types of competencies:

1. Professional competencies that relate to the "knowledge base" required and the ability to use this knowledge in related work area.
2. Personal competencies that represent a set of skills, attitudes and values which enable the professional to work efficiently and adapt to his/her present environment.

Modern SE education is driven by an expectation that industry best practice and state of the art software technologies should be an integral part of the

curriculum, as industrial strength project work is well regarded by potential employers (Hogan et al., 2005). Industry-based projects have students working with a real client who has a significant interest in the results of the project. Developing software that is of benefit to a real client is in itself a strong motivator (Tomayko et al., 1987), as is the use of leading-edge commercial tools in the projects.

Koska and Romano (1988) recommended that college curricula substantively change to emphasize skills that require a systems approach and to create a wide theoretical base and skills that are required by the industry. According to one of the approaches, it is suggested to identify skills and knowledge to be possessed by the graduates and to examine how well the curriculum matches the needs of the industry (Waks, 1995). The rapid pace of technological development must be realized in continuous updating of educational programmes and engineering curricula (Waks and Frank, 2000).

Within the Software Engineering Body of Knowledge report (SWEBOK, 2008) designed by joint IEEE-CS/ACM task force, the software engineering core curriculum comprises of ten knowledge areas, each of which has several knowledge units. These areas and the number of class hours recommended for each is given below:

Knowledge Areas:

- CMP ≡ Computing Essentials (172)
- FND ≡ Mathematical and Engineering Fundamentals (89)
- PRF ≡ Professional Practice (35)
- MAA ≡ Software Modelling and Analysis (53)
- DES ≡ Software Design (45)
- VAV ≡ Software Verification and Validation (42)
- EVL ≡ Software Evolution (10)
- PRO ≡ Software Process (13)
- QUA ≡ Software Quality (16)
- MGT ≡ Software Management (19)

According to this report, an SE curriculum should incorporate a total of 494 class hours for the SE knowledge areas. Different course credit systems are implemented in universities around the world. For example, in Europe, European Credit Transfer System (ECTS) has become a de facto standard which provides a common credit unit system for the students of European Union member countries to enable effective mobility. For the undergraduate

Vol: 13 (1/2011)
page: 188-219

programmes, 30 ECTS units per semester is the standard and unit calculation is based on the amount of student activities and effort during the semester. In the United States and Canada, unit credits are assigned according to the number of face-to-face teaching, laboratory and similar activities. The majority of Turkish universities employ the latter, albeit the fact that, recently, in compliance with the Bologna process, ECTS credits are also mentioned. Consequently, there is a difficulty in matching the hours of SE knowledge areas indicated in the SWEBOK to the actual programmes with varying credit measures.

In the following section, a comparative study will be made to measure the level of conformity of the SE undergraduate programme in Turkey to the suggested number of hours in the SWEBOK.

SE UNDERGRADUATE EDUCATION IN TURKEY

Universities in Turkey have initiated SE undergraduate programmes only in the last decade. Before this the software industry relied heavily on the graduates of the Computer Engineering and other related programmes. So far, in Turkey 11 universities out of 162 (10 foundation/private and 1 government) offer SE undergraduate programmes with a total number of 921 intakes in the 2010-11 Academic Year where the acceptance (by exam) rate was only about 50%. The SE programme in government universities is presently inactive.

On the one hand, the software industry in Turkey is in great need of qualified software engineers, on the other hand, the inclination of the candidates towards the field is surprisingly low. This may be attributed to the low popularity of the field compared to Computer Engineering and the high tution fees at the foundation universities.

Here, the coverage of knowledge areas by the eleven SE undergraduate programmes in Turkey will be investigated. The following observations are made about the programmes.

- The duration of a course is 14 weeks, and therefore, 42 hours in the SWEBOK corresponds to one three-hour credit course.
- Majority of the courses are taught as three hour lectures (three credit units) per week.
- All programmes have a Senior Design Project in the last year.

- First two years of the programmes are similar to the classical engineering programmes.
- Some of the SWEBOK units are taught as technical elective courses.

Table 1. SE Curriculum Compliance

#	University	CMP	FND	PRF	MAA	DES	VAV	EVL	PRO	QUA	MGT	Total
	Equivalent Units (hrs/42)	**4.09**	**2.12**	**0.83**	**1.26**	**1.07**	**1.00**	**0.24**	**0.31**	**0.38**	**0.45**	**11.76**
1	Atılım	3.83 *pe*	1.76 *pe*	0.71 *c*	1.00 *c*	1.07 *c*	1.00 *c*	0.08 *m*	0.15 *m*	>0.38 *c*	>0.45 *c*	10.43
2	Bahçeşehir	3.62 *c*	1.67 *c*	0.63 *m*	0.00	1.07 *c*	1.00 *c*	0.00	0.08 *m*	0.00	>0.45 *c*	8.52
3	Beykent	3.52 *c*	1.90 *c*	0.71 *c*	1.00 *c*	1.07 *c*	0.00	0.08 *m*	0.15 *m*	>0.38 *c*	>0.45 *c*	9.26
4	Işık	3.62 *c*	1.43	0.00	0.25 *m*	0.26 *m*	0.00	0.00	0.05 *m*	0.00	0.11	5.72
5	İstanbul Aydın	3.50 *c*	0.48	0.75 *c*	1.00 *c*	0.54 *m*	0.00	0.00	0.08 *m*	>0.38	>0.45	7.18
6	İzmir Ekonomi	3.10 *ce*	1.83 *pe*	0.75 *c*	0.00	0.54 *m*	1.00 *c*	0.16 *e*	0.15 *m*	0.00	>0.45	7.98
7	İzmir	3.81 *c*	1.81 *pe*	0.30 *c*	1.00 *c*	1.07 *c*	0.50	0.00	0.00	0.19	>0.45	9.13
8	Maltepe	3.81 *pe*	1.64	>0.83	1.00 *e*	1.07 *c*	1.00 *c*	>0.24 *c*	0.05 *m*	>0.38 *e*	>0.45 *c*	10.47
9	Toros	3.33 *c*	1.33	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	4.66
10	Yaşar	3.33 *c*	1.52 *c*	>0.83 *c*	1.00 *e*	1.07 *c*	1.00 *c*	>0.24 *e*	0.05 *m*	>0.38 *c*	>0.45 *c*	9.87
11	Fıratt	*Programme became inactive as of 2011-12 Academic Year*										

e ≡ *Offered as an elective course unit* — *pe* ≡ *Offered partly as an elective course unit*
c ≡ *Compulsory course unit* — *m* ≡ *Module in a course unit*
> ≡ *Over emphasized unit*

Based on these observations, the number of hours allocated to each knowledge area is converted to course units (SWEBOK hours/42) and the SWEBOK coverage by the SE programmes in Turkey is summarized in Table 1. The curriculum and the course descriptions of the programmes as provided by the corresponding web pages are investigated. The letters (*c*, *e*, *m*, and *pe*) are

Vol: 13 (1/2011)
page: 188-219

used to denote the way a unit is taught as explained in the annotations in the lower part of the table. The last column of the table displays the total number of course units (over emphasized ">" is taken as "=" in the summation).

Using the table, one can conceivably make the following remarks and suggestions on the SE curriculum of the ten universities in question:

- All programmes seem to cover the majority of the topics in the "Computing Essentials" (CMP) and "Mathematical and Engineering Fundamentals" (FND) knowledge areas.
- In CMP the coverage of the following topics within the "construction technologies" are at an unacceptable level:
  - Middleware
  - Construction methods for distributed software
  - Constructing heterogeneous systems
  - and Performance analysis and tuning.
- In addition, "formal construction methods" of CMP are not covered by any of the programmes.
- In FND, a majority of the programmes offer a compulsory course in "Engineering Economics" to all engineering students. However, such a course should include topics which take into account some economic value considerations over the software life cycle.
- Again, for the FND, some of the "engineering foundations of software" units such as measurement theory, empirical methods and experimental techniques, and systems development issues need to be incorporated.
- In all but two programmes, "Software Evolution" (EVO) is either missing or covered partially. Similarly, "Software Process" (PRO) coverage is at an unacceptable level.
- "Software Management" (MGT) and "Software Quality" (QUA) units are over emphasized since these units are taught as one-semester courses instead of course modules.

CURRICULUM IMPLEMENTATION

An SE curriculum is organized as a chain of courses following the natural ordering inherent in the software development life cycle. An undergraduate SE curriculum usually requires a total of 125-145 credit hours. In addition to the fundamental course units in science and engineering, the curriculum is supported by a set of technical and non-technical electives. In order to fulfill the ABET or similar engineering accreditation criteria, the universities need to consider a balance of credit units in science, mathematics, social science and software engineering.

The SWEBOK units above should be taught in the order shown in the prerequisite chart in Figure 1. The majority of the SE curricula include a final year design project component which involves all the stages of software development life cycle. In the current implementations in Turkey, however, because of time restrictions, software design and implementation phases are overemphasized. Consequently, students usually pay less attention to the study of requirements, project management, documentation and testing phases. Project topics should be offered at an earlier phase of the programme, preferably in the second year of a four year degree programme parallel with the coverage of the software life cycle. On the other hand, in this implementation of the graduation project, some pedagogical problems may arise because of irregular students displaying low performance in some of the courses supporting the software life cycle, hence distracting the progress of the project team.

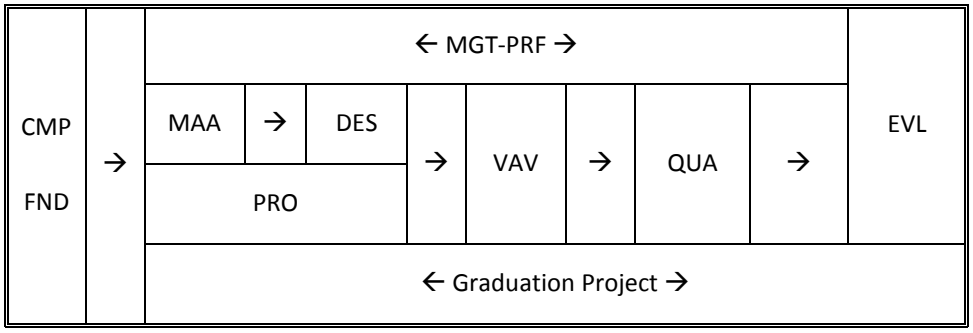

Figure 1. SWEBOK prerequisite chart

Vol: 13 (1/2011)
page: 188-219

DISCUSSION

Curriculum development is naturally affected by the availability of academic staff and their research interests. In Turkish SE programmes, the lack of staff with PhD in SE or related fields is an important factor for full compliance with the IEEE/ACM SWEBOK. Although universities offer SE MSc programmes in the related areas, there are only a few SE PhD programmes. The rich variety of job opportunities in the software sector offered to SE graduates is another factor making the academic path inevitably unfavorable in the industry.

For a more attractive and effective SE curriculum, the following points need to be considered:

- a close collaboration with the technology parks in terms of curriculum development, internship and summer practices;
- inclusion of software developers and CEOs as well as alumni in the curriculum development process;
- partnerships with the leading software companies by incorporating software development and testing tools, and frameworks into the curriculum;
- provision of student involvement in all stages of the software development life cycle by introducing term projects in parallel with the software development phases;
- introduction and close cooperation with the professional NGOs in software engineering such as IEEE, ACM, Software Engineering Institute, and at the national level, YASAD (Software Industrialists Association), TBD (Turkish Informatics Society), TBV (Turkish Informatics Foundation), and so on.

Reasons for the deviation in implementing IEEE/ACM guidelines might be, for example, lack of expert academic staff in specialized areas, like software evolution/maintenance, software architecture, and so on. Further, it is interesting to note that, except for one, all the universities implementing undergraduate education in SE are in the private sector. Flexibility in recruitment and salary packages seem to be significant factors in this regard. Over the last five years, software technology parks have also assisted SE graduates in employment, internship and summer practice in a number of companies. In reviewing the SE curricula, it was observed that there is a need for professional development and ethics related courses in SE education.

Presently, SE education accreditation in Turkey, similar to the one for Computer Engineering education, does not exist, but it is expected that in the future, MUDEK (Association for Evaluation and Accreditation of Engineering Programmes in Turkey) will initiate steps in this direction. In curriculum implementation, for example, specifically in deciding the order (pre-requisite courses) among courses, it is important to note that students should begin with fundamental courses so that they can easily appreciate later courses. For instance, knowledge of software engineering fundamentals, object-oriented concepts, requirements engineering and software quality are important for a software architecture course. In addition, significance of internship, summer practices, senior projects, case studies and industrial reports/white papers should be an integral part of the curriculum. New technologies and tools should be introduced in software labs for all practical oriented courses like “software validation and testing”,“requirements engineering”, “agile software development”, ”software modelling” and the like. Case studies and special lectures by software industry practitioners can be immensely helpful in clarifying concepts along with applications in the courses like “software architecture”, “software project management”, “software process and improvement”, “human-computer interaction”, and so on. Seminars and workshops can enhance the knowledge in ethics, professional development and practice areas. Finally, students should be provided with directions to appreciate software tools from open source in all courses.

CONCLUSION

SE is progressively becoming a mature discipline and increasingly critical in all walks of life. The demand for software engineering is increasing, so there is an increasing demand for efficient and knowledgeable software engineers. The collaboration between the software industry and higher education departments can lead to synergies for both in accomplishing their objectives. In this study, all SE courses at undergraduate level in Turkish universities have been assessed with respect to IEEE/ACM guidelines provided by SEEK (2004). However, as with other studies, there are some limitations in our study as data have mostly been collected from the universities' websites. Although it has been examined in detailed, there might be a possibility of error in judgement while taking into account partial coverage of the course content,

Vol: 13 (1/2011)
page: 188-219

for example, considering only one unit in the module instead of two. We believe this study will provide a guideline to universities conducting SE courses at the undergraduate level to align their course curriculum with IEEE/ACM guidelines (SEEK, 2004). The development of the social, technical and ethical skills of teamwork and accountability can also build on and add considerable value to the key technical skills in software engineering education to produce graduates with strong career credentials.

REFERENCES

ABET, Applied Science Accreditation (2002). Volume I Self-Study Questionnaire 2002-*3 Edition.* Available http://www.abet.org/ info_prgs_rac.html

ACM/IEEE Joint Task Force on Computing Curricula: 2005 - overview report (2005). Available at http://www.acm.org/education/curricula.html, retrieved March 23, 2007.

Ardis, M. and Ford, G.A. (1989) SEI Report on Graduate Software Engineering Education. In *Proceedings of the SEI Conference on Software Engineering Education*, Norman E. Gibbs (Ed.). Springer-Verlag, London, UK, 208-249.

Ardis, M., Bourque, P., Hilburn, T., Lasfer, K., Lucero, S., McDonald, J., Pyster, A., Shaw, M. (2011) Advancing Software Engineering Professional Education, *IEEE Software*, IEEE Computer Society Digital Library. IEEE Computer Society, <http://doi.ieeecomputersociety.org/10.1109/MS.2010.133

Beckman, K., & Coulter, N., Khajenouri, S. & Mead, N. (1997). Collaborations: closing the industry–academy gap. IEEE Software, 14(6), 49–57.

Bloom, B.S. & Hastings, J.T., Madaus, G.F. (1956). Taxonomy of Educational Objectives: The Classification of Educational Goals. Handbook 1, Cognitive Domain. New York, McCay, 1956.

Brazilay, O, Hazzan, O, & Yehudai, A. (2009) A Multidimensional software engineering course, IEEE Transactions on Education, 52(3), 413-424.

Dey, S.K., & Sobhan, M.A. (2007). *Guidelines for Preparing Standard Software*. *Engineering Curriculum*: Bangladesh and Global. Perspective, IEEE Computer Society.

Ford, G., & Gibbs, N.E., (1996). A Mature Profession of Software Engineering. Technical Report CMU/SEI-96-TR-004, ESC-TR-96-004, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, abstract and downloading point at http://www.sei.cmu.edu/products/publications/96.reports/96.tr.004.html.

Hogan, J. M., Smith, G., & Thomas, R. (2005). Tight spirals and industry clients: the modern SE education experience. In *Proceedings of the 7th Australasian Conference on Computing Education - Volume 42* (Newcastle, New South Wales, Australia). A. Young and D. Tolhurst, Eds. ACM International Conference Proceeding Series, vol. 106. Australian Computer Society, Darlinghurst, Australia, 217-222.

Horn, E., & Kupries, M. (2003). A Study Program for Professional Software Engineering. in proceedings of the 16th Conference on Software Engineering Education and Training (CSEET' 03), IEEE Computer Society.

Jaakkola, H., Henno, J., & Rudas, I.J. (2006). IT Curriculum as a. Complex Emerging Process. IEEE Computer Society.

Kitchenham, B., Budgen, D., Bereton, P. & Woodall, P. (2005) An investigation of software engineering curricula. Journal of System and Software, 74, 325–335.

Kornecki, A. J., Khajenoori, S., Gluch, D., & Kameli, N. (2003). On a Partnership between Software Industry and Academia. In Proceedings of the 16th Conference on Software Engineering Education and Training (March 20 - 22, 2003). CSEET. IEEE Computer Society, Washington, DC, 60.

Koska, D.K., & Romano, J.D. (1988). Countdown to the Future: The Manufacturing Engineer in the 21st Century. Detroit, MI: Soc., Manufact. Eng.

Vol: 13 (1/2011)
page: 188-219

Kral, J., & Zemlicka, M. (2008). Engineering Education - A Great Challenge to Software Engineering. In Proceedings of the Seventh IEEE/ACIS international Conference on Computer and information Science (ICIS 2008) (May 14 - 16, 2008). International Conference on Information Systems. IEEE Computer Society, Washington, DC, 488-495.DOI= http://dx.doi.org/10.1109/ ICIS.2008.116

Lethbridge, T.C. (2000a). What knowledge is important to a software professional? IEEE Computer, 33(5), 44–50.

Lethbridge, T.C. (2000b). Priorities for the education and training of software engineers. Journal of System and Software, 53(1), 53–71.

Mishra, A., Cagiltay, N.E., & Kilic, O. (2007). Software Engineering Education: Some Important Dimensions. European Journal of Engineering Education, 32(3), 349-361.

Pinto, Y. (2010). A strategy, implementation and results of a flexible competency based curriculum. *ACM Inroads* 1, 2 (Jun. 2010), 54-61. DOI= http://doi.acm.org/10.1145/1805724.1805739

Pyster, A. (Ed.) (2009) Graduate Software Engineering 2009 (GSwE2009) Curriculum Guidelines for Graduate Degree Programs in Software Engineering, Integrated Software & Systems Engineering Curriculum Project, Stevens Institute of Technology, September 30, 2009. Available at www.gswe2009.org

Pyster, A., Turner, R., Henry, D., Lasfer, K., Bernstein, L. (2009) Master's Degrees in Software Engineering: An Analysis of 28 University Programs. *IEEE Software,* 26, 5 (September 2009), 94-101.

SEEK (2004). Curriculum guidelines or undergraduate degree programs in software engineering. 2008 (Online) availble at: http\\sites.computer.org/ccse/.
Shaw, M, Herbsleb, J. D., & Ozkaya, I. (2005). *Deciding What to Design: Closing a Gap in Software Engineering Education.* Invited paper for Education and Training Track of 27th Int"l Conf on Software Engineering(ICSE 2005), May 2005.

Shaw, M. (eds.) (2005) Software Engineering for the 21st Century: A basis for rethinking the curriculum – CMU-ISRI-05-108 available at www.cs.cmu.edu/ ~Compose/SEprinciples-pub-rev2.pdf

Su, H., Jodis, S., & Zhang, H. (2007). Providing an integrated software development environment for undergraduate software engineering courses. *J. Comput. Small Coll.* 23, 2 (Dec. 2007), 143-149.

SWEBOK (2008). Software Engineering body of knowledge. Available at http://www.swebok.org/index.html.

Tomayko, J.E. (1987). Teaching a Project-Intensive Introduction to Software Engineering. Technical Report SEI-SR-87-1, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA.

Waks, S. (1995). Curriculum Design-From an Art Toward Science. Hamburg, Germany: Tempus.

Waks, S., & Frank, M. (2000). Engineering Curriculum versus Industry Needs – A Case Study. IEEE Transactions on Education, 43 (4), 349-352.

Wohlin, C., & Regnell, B. (1999). Strategies for industrial relevance in software engineering education. Journal of System and Software, 49, 125–134.

Alok Mishra
Department of Computer & Software Engineering, Atilim University, Kızılcaşar Mahallesi, 06836 Incek Gölbaşı – Ankara, Turkey, dralokmishra@gmail.com

Ali Yazici
Atilim University, Kızılcaşar Mahallesi, 06836 Incek Gölbaşı - Ankara, Turkey, aliyazici@atilim.edu.tr

Croatian Journal of Education
Hrvatski časopis za odgoj i obrazovanje

Vol: 13 (1/2011)
page: 188-219

EVALUACIJA NASTAVNOG PLANA I PROGRAMA PROGRAMSKOG INŽENJERSTVA NA TURSKIM SVEUČILIŠTIMA S OBZIROM NA IEEE/ACM SMJERNICE

SAŽETAK

Obrazovanje programskih inženjera (PI) razvija se kao samostalna i zrela znanstvena disciplina. Sukladno tome, provode se različita istraživanja u cilju određivanja smjernica za izradu nastavnih planova i programa za obrazovanje PI. Ovaj rad predstavlja razvoj obrazovanja programskih inženjera u Turskoj, trenutnu situaciju u obrazovanju PI na različitim sveučilištima, kao i značenje tehnoloških parkova za njihovo obrazovanje. Cilj ovoga rada je dati evaluaciju nastavnoga plana i programa za Programsko inženjerstvo na turskim sveučilištima s obzirom na IEEE/ACM smjernice navedene u dokumentu SEEK (2004). Ova će studija pružiti smjernice sveučilištima prilikom provođenja programa PI na dodiplomskoj razini kako bi se uskladio plan i program kolegija s IEEE/ACM smjernicama (SEEK, 2004).

Ključne riječi: *nastavni plan i program, obrazovanje programskih inženjera, programsko inženjerstvo, softverska industrija, Turska*

UVOD

Sposobnost bilo koje velike tvrtke ili vladine organizacije da upravlja vlastitim projektima i poduzećima uvelike ovisi o sofisticiranim softverskim sustavima koji podržavaju poslovne i tehničke procese, od logističkih do proizvodnih sustava i sustava upravljanja odnosima s klijentima (Pyster i sur., 2009). Programsko inženjerstvo (PI) bavi se stvaranjem i primjenom inženjerskih osnova za sustavnu i timsku analizu, razvoj, uporabu, vrednovanje, itd. velikih, programski intenzivnih sustava kao tehničkih proizvoda (Horn i Kupries, 2003). Prema istraživanju autora Horn i Kupries (2003), postoji značajna potreba za visoko kvalificiranim stručnjacima osposobljenima za

svladavanje složenih, programski intenzivnih sustava. Programsko inženjerstvo je grana računalne znanosti koja stvara praktična, troškovno učinkovita rješenja za probleme u računarstvu i obradi informacija, naravno, uz primjenu znanstvenih spoznaja za razvoj programskih sustava u službi čovječanstva. Ovdje se pojam *razvoj* odnosi na sve aktivnosti vezane uz softverski proizvod, od ideje do pregovora s klijentom, izrade, primjene, provjere valjanosti, izvršenja, razvoja i ostalih postupaka vezanih uz održavanje (Shaw, 2005). Programsko inženjerstvo je višedimenzionalno znanstveno polje koje uključuje aktivnosti u različitim područjima i disciplinama kao što su računalne znanosti, projektni menadžment, arhitektura sustava, ljudski faktor i tehnološka evolucija (Brazilay i sur., 2009). Brojni su napori učinjeni kako bi se uočile različite dimenzije programskog inženjerstva i izradio odgovarajući nastavni plan i program koji bi se bavio svim dimenzijama PI (SEEK, 2004; SWEBOK, 2008).

Programsko inženjerstvo počiva na sljedećim trima glavnim intelektualnim osnovama (Shaw, 2005):

1. Tehnička osnova: To je skup ključnih koncepata računalne znanosti koji se odnose na strukturu podataka, algoritme, programske jezike i njihovu semantiku, analizu, izračunljivost, računske modele i sl.
2. Znanja o inženjerstvu: Tehničko se znanje primjenjuje kroz brojna znanja o inženjerstvu vezana uz arhitekturu, proces inženjerstva, balansiranje i troškove, konvencionalizaciju i standarde, kvalitetu i jamstvo, itd. Ovo omogućuje pristup izradi i rješavanju problema koji uzima u obzir pragmatična pitanja vezana uz aplikacije.
3. Društveni i gospodarski kontekst: Uključuje proces stvaranja i razvijanja proizvoda, kao i pitanja vezana uz politiku, tržište, upotrebljivost i društveno-ekonomske utjecaje; pruža osnovu za oblikovanje projektiranih proizvoda kako bi bili prikladni za njihovu predviđenu namjenu.

Česte su pritužbe tvrtki koje se bave razvojem programskih paketa na praktična znanja studenata koji počinju raditi odmah po završetku studija. Studenti uglavnom imaju visoku razinu teorijskog znanja, ali često im nedostaje iskustva u rješavanju stvarnih problema u industriji. Pritužbe na kvalitetu programskih paketa široko su rasprostranjene, a većina ovisi o nekoliko čimbenika, uključujući lošu kvalitetu nastave na sveučilištu, gdje niti poučavanje ključnih vještina nije na prihvatljivoj razini (Jaakkola i sur., 2006). Tijekom

Vol: 13 (1/2011)
page: 188-219

posljednjeg desetljeća, obrazovanje programskih inženjera postalo je široko rasprostranjeno i još se uvijek širi. Nadalje, PI je najbrže rastuća inženjerska disciplina, a većina zadataka organizacija za razvoj programskih paketa po prirodi je raznovrsna. Promjene unutar PI znatno su veće, sveobuhvatnije i brže od utjecaja drugih disciplina, a njegova je najvažnija osobina sposobnost pružanja alata i metoda primjenjivih u svim područjima društva, odnosno, inženjerstva i obrazovanja inženjera (Kral i Zemlica, 2008). U tom kontekstu, obrazovanje PI ima dužnost pripremiti stručnjake-programske inženjere za potrebe industrije pružajući im vještine potrebne da bi se zadovoljila očekivanja softverske industrije. Inovacije i poboljšanja u nastavnom planu i programu, poučavanje i evaluacija usmjereni su k premošćivanju jaza između akademskog svijeta i industrije prikazivanjem prave prirode razvoja programskih paketa i pružanjem pomoći studentima razvijanjem osnovnih znanja, vještina i stavova koji su zapravo potrebni industriji (Shaw i sur., 2005). Mnogi izazovi povezani s obrazovanjem programskih inženjera rezultat su naše nesposobnosti da unutar obrazovnih institucija studentima omogućimo praktična iskustva u 'stvarnome svijetu', razvojem velikih programskih paketa (Su i sur., 2007). Dakle, kvaliteta programskih inženjera izravni je odraz kvalitete obrazovanja PI. Istraživanja su pokazala da postoji veliki jaz između potreba softverske industrije i obrazovanja budućih programskih inženjera (Beckman i sur. 1997). Dakle, nemoguće je producirati gotove diplomante, koji se neposredno po završetku obrazovanja mogu izravno uključiti u industriju. Rješenje bi moglo biti u omogućavanju praktične izobrazbe unutar organizacije, prije nego im se dodijele odgovarajuća radna mjesta. Također je važno omogućiti diplomantima usvajanje kvalitetnih praktičnih znanja iz različitih područja. Dakle, ako su studenti dobro upućeni u nove tehnologije, vremenski period izobrazbe na radnome mjestu može biti kraći, što znači da će tvrtke morati uložiti manje vremena i novca (Mishra i sur., 2007). Jaakkola i sur. (2006) također podržavaju ideju prema kojoj bi nastavni plan i program programskog inženjerstva trebao biti u skladu s potrebama industrije, jer tek tada sveučilišta mogu proizvesti visoko kvalificirane stručnjake koji će zadovoljiti potrebe softverske industrije. Nadalje, autori smatraju kako pri razvoju nastavnih planova i programa treba uzeti u obzir različite standarde, okvire i preporuke interesnih skupina.

Cilj ovoga rada je pružiti evaluaciju nastavnoga plana i programa programskog inženjerstva na turskim sveučilištima s obzirom na IEEE/ACM smjernice navedene u dokumentu SEEK (2004). Ostatak rada organiziran je na

način kako slijedi. U poglavlju 2 raspravlja se o značajnim pitanjima vezanim uz nastavni plan i program programskog inženjerstva s obzirom na značenje za industriju. U poglavlju 3 prikazan je razvoj nastavnog plana i programa programskog inženjerstva. U poglavlju 4 predstavljen je dodiplomski studij u Turskoj s obzirom na IEEE/ACM smjernice, a primjena nastavnog plana i programa opisana je u poglavlju 5. Rasprava o temi prikazana je u poglavlju 6. Konačno, zaključci su predstavljeni u poglavlju 7.

NASTAVNI PLAN I PROGRAM PROGRAMSKOG INŽENJERSTVA I SOFTVERSKA INDUSTRIJA

Postoje ozbiljni problemi vezani uz kvalitetu programskih paketa i troškove proizvodnje (Ford i Gibbs, 1996), a predstavnici softverske industrije često se žale na znanja i vještine diplomanata u nekim ključnim područjima programskog inženjerstva, kao što su: (i) modeli razvoja programskih paketa, (ii) inženjerstvo zahtjeva, (iii) arhitektura softvera i dizajn više razine, (iv) softverski procesi, (v) osiguranje i upravljanja kvalitetom softvera, (vi) upravljanje softverskim projektima, (vii) upravljanje organizacijama i timski rad, (viii) testiranje programskih paketa, i tako dalje.
Navedena su područja izravno u skladu s fazama razvojnog ciklusa i trajanja programskog paketa.

Prema zapažanjima iznesenima u studiji Lethbridgea, sljedeće su teme vrlo važne za softversku industriju i studenti bi ih mogli naučiti tijekom radnog osposobljavanja (Lethbridge, 2000a):

- Objektno usmjereni koncepti i tehnologije
- Prikupljanje i analiza zahtjeva
- Metode analize i dizajna
- Testiranje, verifikacija i osiguranje kvalitete
- Upravljanje projektima
- Interakcija čovjek-računalo (HCI) / korisničko sučelje
- Baze podataka
- Upravljanje konfiguracijom i distribucijom
- Etika i profesionalizam
- Tehničko pisanje
- Prezentacije/seminari za publiku
- Vještine vođenja.

Vol: 13 (1/2011)
page: 188-219

Prema Lethbridge (2000b) postoji potreba za uključivanjem ovih predmeta u nastavni plan i program zajedno sa stvarnom praksom u industriji. Drugo važno pitanje, zajedno s praktičnim iskustvom, su vještine potrebne za rad u skupinama budući da u stvarnom okruženju ljudi moraju raditi u skupinama. Kitchenham i sur. (2005) također podržavaju nekoliko zapažanja koje iznosi Lethbridge u odnosu na prenaglašavanje matematičkih tema uz nedovoljan naglasak na poslovnim temama. Međutim, njihova se saznanja razlikuju od Lethbridgea s obzirom na teme kod kojih su veće razlike u znanju. Studenti programskog inženjerstva trebali bi posjedovati vještine potrebne pri pojedinačnom ali i timskom radu na razvoju i isporuci kvalitetnih programskih paketa (SEEK 2004).

Odabir prikladnog partnera u industriji s ciljem osiguranja izazovnog i motivirajućeg projekta presudan je za uspjeh projekata vezanih uz industriju, što od akademskog osoblja zaduženog za koordinaciju zahtijeva visoku motiviranost za pružanje potpore projektu i provođenje pripreme projekta prije početka svakog semestra. Nadalje, važno je da klijent iz industrije mora biti spreman prihvatiti i ishod neuspjelog projekta (Hogan i sur., 2005). Studenti obično rade na projektima kao tim, stoga različiti timovi mogu doći do različitih rješenja. Dobivena rješenja se raspravljaju unutar cijele skupine na način da studenti analiziraju različita rješenja i predstavljaju svoj pristup. Tijekom ovog procesa rasprave olakšavaju učenje i razvoj komunikacijskih vještina. U većini slučajeva industrija je zainteresirana za aktivnosti usmjerene na primjenu koje donose neposredna rješenja, pomažu u primjeni novih proizvoda ili poboljšavaju konačan ishod svakodnevnih aktivnosti (Kornecki i sur., 2003). Ovi su autori primijetili da se poduzeća natječu za diplomante koji posjeduju vještine za suočavanje s izazovima razvoja sigurnosno-kritičnih programski intenzivnih sustava, budući da je stalan priliv kvalificiranog osoblja u tim područjima neophodan.

Dodatni ključni element za uspjeh programa programskog inženjerstva je uključivanje i aktivno sudjelovanje industrije (SEEK, 2004). U tom su kontekstu Wohlin i Regnell (1999) predstavili strategije za obrazovanje programskih inženjera relevantne za industriju. Postoje razni načini ostvarivanja suradnje između industrije i akademske zajednice. Mnoge akademske institucije osnovale su unutar svojih odsjeka, ili na neki drugi način, akademske savjetodavne odbore vezane uz industriju. Te skupine, sastavljene od

menadžera i inženjera iz industrije koji su usko povezani s akademskom organizacijom, jedan su od načina dobivanja povratne informacije o potrebama studijskog programa (Kornecki i sur., 2003). Ostali karakteristični oblici suradnje su povremeni kratkoročni projekti koji udovoljavaju trenutnim potrebama industrije. Takvi projekti omogućuju fakultetu i studentima upoznavanje s ovim područjem i često pridonose ostvarenju ciljeva partnera u industriji. Ostali načini suradnje su ljetna praksa za studente i programi stažiranja na fakultetu. Na taj se način olakšava razumijevanje potreba industrije i omogućuje sveučilištu uvođenje odgovarajućih promjena programa. Svi ti pristupi mogu biti učinkoviti, ali ne pružaju trajno rješenje u tom smjeru. Akademskim ustanovama treba omogućiti preoblikovanje i provedbu programa PI koji ne samo da naglašavaju teorijske i tehničke aspekte računarstva, već se i usredotočuju na praksu PI (Dey i Sobhan, 2007).

Najnoviji primjer nastavnih planova i programa ACM (2005) prepoznaje različite perspektive akademske zajednice i industrije. ACM preporučuje nastavne predmete za planove i programe PI te razvijanje tehničkih i netehničkih vještina potrebnih za razvoj velikih programskih paketa. U tom se smislu preporučuje prikaz vještina "na odgovarajućem rasponu aplikacija i studija slučaja koje povezuju teoriju i vještine naučene na fakultetu sa pojavama u stvarnome svijetu u svrhu pojašnjenja njihove relevantnosti i korisnosti" (ACM, 2005).

RAZVOJ PLANA I PROGRAMA PROGRAMSKOG INŽENJERSTVA

Godine 1989. *Software Engineering Institute* (SEI) s *Carnegie Mellon* Sveučilišta objavio je značajno izvješće o diplomskom studiju programskog inženjerstva (Ardis i Ford, 1989). Diplomski studij je ključni element za unapređivanje stručne prakse programskog inženjerstva (Ardis i sur., 2011). Nedavno je projekt „Plan i program integriranog programskog i sistemskog inženjerstva“ (ISSEC) na *Stevens Institute of Technology* razvio niz smjernica za program magistarskog studija pod nazivom "Diplomski studij programskog inženjerstva 2009 (GSwE2009): Smjernice za nastavni plan i program diplomskog studija programskog inženjerstva" (Pyster (ur.), 2009). Izrada nastavnih materijala i odabir sadržaja treba uzeti u obzir Bloomove razine učenja (Bloom i sur., 1956). Budući da se znanje usvaja na različitim razinama,

Vol: 13 (1/2011)
page: 188-219

moduli bi trebali biti organizirani na način da se studenta vodi od osnovnih znanja i koncepata kroz primjenu i analizu znanja te konačno prema sintezi i evaluaciji (Pinto, 2010).

Prema Pinto (2010), izrada nastavnog plana i programa treba uzeti u obzir sljedeće:

1. utvrditi prilike za zapošljavanje i pripadajuće obveze
2. utvrditi osobine ličnosti (generičke vještine) potrebne za posao
3. utvrditi kompetencije potrebne za obavljanje posla
4. utvrditi teorijska i praktična znanja, kao i trenutne vještine potrebne za postizanje kompetencija
5. utvrditi odgovarajuće predmetne pojmove (teoriju) važne za stjecanje potrebnih temeljnih znanja
6. utvrditi zadatke (vježbe) koji poboljšavaju praktična znanja vezana uz predmet
7. utvrditi vještine (sredstva) potrebne za obavljanje zadataka
8. utvrditi referentni materijal čiji se predmetni ciljevi podudaraju sa znanjima potrebnima u točkama 5, 6 i 7
9. pripremiti testiranje kojim se može provjeriti znanje kandidata na različitim razinama za određenu kompetenciju.

Iako ključna temeljna konceptualna znanja iz IT ostaju ista, napredak postignut u području tehnologije izrazito je brz. Stoga, vještine koje studenti razvijaju moraju odgovarati trenutnim potrebama industrije. Izrada nastavnoga plana i programa mora biti dovoljno fleksibilna kako bi napredak u tehnologiji bio uzet u obzir (Pinto, 2010). Pinto (2010) također naglašava da profesionalno izrađen plan i program zahtijeva uključivanje dviju osnovnih vrsta kompetencija:

1. Stručne kompetencije koje se odnose na potrebnu "bazu znanja" i sposobnost za korištenje ovog znanja unutar odgovarajućeg radnog područja.
2. Osobne kompetencije koje predstavljaju skup vještina, stavova i vrijednosti, a koje omogućuju stručnjacima da rade učinkovito i prilagode se svojoj okolini.

Suvremeno obrazovanje PI usmjerava se prema očekivanju da bi najbolja praktična znanja industrije i posljednja dostignuća vezana uz softversku

tehnologiju trebala biti sastavni dio nastavnog plana i programa, budući da potencijalni poslodavci pozitivno procjenjuju projektni rad na jačanju industrije (Hogan i sur., 2005). Projekti vezani uz industriju omogućuju studentima rad s pravim klijentom kojega zanimaju rezultati projekta. Razvoj programskih paketa koji su od koristi pravom klijentu sam po sebi je snažan motivacijski čimbenik (Tomayko i sur., 1987), kao i korištenje naprednih komercijalnih alata u projektima.

Koska i Romano (1988) preporučili su značajne promjene u nastavnom planu i programu studija kako bi se naglasile vještine koje zahtijevaju sustavni pristup i stvorila široka teoretska osnova te razvile vještine koje su potrebne industriji. Prema jednom od pristupa, preporučuje se identificiranje vještina i znanja potrebnih diplomantima i provjeravanje u kolikoj mjeri nastavni plan i program odgovara potrebama industrije (Waks, 1995). Brzi tempo tehnološkog razvoja mora biti realiziran kroz kontinuirano ažuriranje svih obrazovnih programa, pa tako i plana i programa inženjerstva (Waks i Frank, 2000).

Zajednička radna skupina IEEE-CS/ACM izradila je Izvješće o temeljnim znanjima programskog inženjerstva (SWEBOK, 2008) unutar kojeg se temeljni plan i program programskog inženjerstva sastoji od deset područja znanja od kojih svaki sadrži nekoliko obrazovnih jedinica. Područja i preporučeni broj sati nastave dani su u nastavku:

Područja znanja:

- CMP ≡ Osnove računarstva (172)
- FND ≡ Osnove matematike i inženjerstva (89)
- PRF ≡ Stručna praksa (35)
- MAA ≡ Modeliranje i analiza programskog paketa(53)
- DES ≡ Izrada programskog paketa (45)
- VAV ≡ Provjera i vrednovanje programskog paketa(42)
- EVL ≡ Razvoj programskog paketa (10)
- PRO ≡ Softverski proces (13)
- QUA ≡ Kvaliteta programskog paketa (16)
- MGT ≡ Upravljanje programskim paketom(19).

Prema ovom izvješću, nastavni plan i program programskog inženjerstva trebao bi uključivati ukupno 494 sati nastave iz područja znanja programskog inženjerstva. Različiti sustavi bodovanja kolegija provode se na

Vol: 13 (1/2011)
page: 188-219

sveučilištima diljem svijeta. Na primjer, u Europi je europski sustav prijenosa bodova (ECTS) postao de facto standard koji omogućuje zajednički sustav bodovnih jedinica za studente zemalja članica Europske unije kako bi se omogućila učinkovita mobilnost. Za preddiplomske programe 30 ECTS bodova po semestru je standard i izračun bodova temelji se na količini studentskog rada i uloženog truda tijekom semestra. U Sjedinjenim Američkim Državama i Kanadi bodovi se dodjeljuju prema broju sati izravne nastave, laboratorijskih vježbi i sličnih aktivnosti. Većina turskih sveučilišta koristi potonji, iako je činjenica da se odnedavno, u skladu s Bolonjskim procesom, razmatraju i ECTS bodovi. Prema tome, postoje poteškoće pri usklađivanju broja sati područja znanja PI navedenih u SWEBOK-u i stvarnih programa s različitim brojem bodova.

U sljedećem će poglavlju biti prikazana poredbena studija kojom će se mjeriti stupanj sukladnosti programa preddiplomskog studija programskog inženjerstva u Turskoj s predloženim brojem sati nastave u SWEBOK-u.

PREDDIPLOMSKI STUDIJ PROGRAMSKOG INŽENJERSTVA U TURSKOJ

Sveučilišta u Turskoj pokrenula su preddiplomske programe PI tek u posljednjem desetljeću. Prije toga softverska se industrija uvelike oslanjala na diplomante računalnog inženjerstva i drugih srodnih programa. Do sada u Turskoj 11 sveučilišta od ukupno 162 (10 privatnih i 1 državno sveučilište) nude preddiplomske programe PI s ukupnim brojem upisanih studenata od 921 u akademskoj godini 2010-11, s upisom (temeljem ispitnih rezultata) od samo oko 50 %. Program PI na državnim sveučilištima trenutno nije aktivan.
S jedne strane, softverskoj industriji u Turskoj uvelike su potrebni kvalificirani programski inženjeri, dok je s druge strane zanimanje kandidata za ovo znanstveno polje iznenađujuće malo. To se može pripisati niskoj popularnosti polja u odnosu na računalno inženjerstvo i visoke školarine na privatnim sveučilištima.

U ovom će se poglavlju obraditi pokrivenost područja znanja na jedanaest preddiplomskih programa PI u Turskoj. U nastavku su neka zapažanja o programima.

- Trajanje nastave je 14 tjedana, i stoga 42 sata u SWEBOK-u odgovaraju jednom kolegiju koji se izvodi tri sata tjedno.

- Većina kolegija poučava se kao tri sata predavanja (tri boda) tjedno.
- Svi programi imaju „projekt za studente završne godine“ u posljednjoj godini studija.
- Prve dvije godine programa slične su klasičnim inženjerskim programima.
- Neke jedinice SWEBOK-a poučavaju se kao tehnički izborni predmeti.

Temeljem opisanih zapažanja, broj sati dodijeljen svakom području znanja pretvoren je u bodove (SWEBOK sati/42), a sažetak zastupljenosti preporuka SWEBOK-a u programima PI u Turskoj prikazan je u Tablici 1. Istraživanje je uključivalo uvid u nastavni plan i program i opis programa studija prikazan na odgovarajućim web stranicama pojedinih sveučilišta. Slova (*c, e, m*, i *pe*) se koriste za označavanje načina poučavanja određenih kolegija prema komentaru u donjem dijelu tablice. Posljednji stupac tablice prikazuje ukupan broj bodova (prenaglašen ">" se uzima kao "=" u konačnom zbroju).

Tablica 1.

Temeljem podataka prikazanih u tablici, mogu se navesti sljedeće primjedbe i prijedlozi vezani uz nastavni plan i program PI prikazanih deset sveučilišta:

- Čini se da svi programi pokrivaju većinu tema u područjima znanja "Osnove računarstva“ (CMP) i "Osnove matematike i inženjerstva" (FND).
- U „Osnovama računarstva“ pokrivenost sljedećih tema unutar "konstrukcijskih tehnologija" na neprihvatljivoj je razini:
  - *Middleware*
  - Konstrukcijske metode za distribuirane programske pakete
  - Konstrukcija heterogenih sustava
  - Analiza izvedbe i podešavanje.
- Osim toga „formalne konstrukcijske metode" „Osnova računarstva“ (CMP) nisu pokrivene ni jednim programom.
- U „Osnovama matematike i inženjerstva" (FND) većina studijskih programa nudi obvezni predmet „Inženjerska ekonomija“ svim studentima inženjerstva. Međutim, takav kolegij treba sadržavati teme

Vol: 13 (1/2011)
page: 188-219

koje uzimaju u obzir neka razmatranja o ekonomskoj vrijednosti razvojnog ciklusa i trajanja programskog paketa.

- Također, za „Osnove matematike i inženjerstva" (FND), neke jedinice "inženjerskih osnova programskih paketa", kao što su teorija mjerenja, empirijske metode i eksperimentalne tehnike te pitanja razvoja sustava trebaju biti uključene u program.
- U svim, osim u dva programa, "Razvoj programskog paketa" (EVO) nedostaje ili je pokriven djelomično. Slično tome pokrivenost "Softverskog procesa" (PRO) nije na prihvatljivoj razini.
- "Upravljanje programskim paketom" (MGT) i "Kvaliteta programskog paketa" (QUA) prenaglašeni su jer se poučavaju kao jednosemestralni kolegiji umjesto kao moduli.

PROVEDBA NASTAVNOG PLANA I PROGRAMA

Nastavni plan i program PI organiziran je kao niz kolegija koji prate prirodni slijed svojstven razvojnom ciklusu programskog paketa. Nastavni plan i program dodiplomskog studija PI obično zahtijeva ukupno 125-145 sati. Osim temeljnih predmeta iz prirodnih znanosti i inženjerstva, nastavni plan i program uključuje i niz tehničkih i netehničkih izbornih kolegija. Da bi se zadovoljili kriteriji ABET-a ili slični kriteriji za akreditaciju inženjerstva, sveučilišta trebaju uzeti u obzir usklađenost bodovne vrijednosti u prirodnim znanostima, matematici, društvenim znanostima i programskom inženjerstvu.

Spomenute SWEBOK nastavne jedinice trebale bi se poučavati redoslijedom koji je prikazan u preduvjetima na slici 1. Većina nastavnih planova i programa PI obuhvaća, kao sastavni dio završne godine, izradu projekta koji uključuje sve razvojne faze i trajanje programskog paketa. U aktualnim su programima u Turskoj, međutim, zbog ograničenja vremena prenaglašene faza izrade programskih paketa i primjene. Stoga studenti obično posvećuju manje pozornosti fazama proučavanja zahtjeva, upravljanja projektima, izrade dokumentacije i testiranja. Teme projekata trebale bi biti ponuđene u ranijoj fazi programa, po mogućnosti u drugoj godini četverogodišnjeg studijskog programa paralelno s pokrivanjem razvojnog ciklusa i trajanja programskog paketa. S druge strane, u ovoj provedbi završnog (diplomskog) projekta, neki pedagoški problemi mogu nastati zbog neredovitih studenata koji pokazuju niske rezultate iz nekih kolegija vezanih uz razvojni

ciklus i trajanje programskog paketa, čime se otežava napredak projektnog tima.

Slika 1.

RASPRAVA

Razvoj nastavnog plana i programa naravno ovisi o dostupnom akademskom kadru i njihovim istraživačkim interesima. U turskom programu PI nedostatak osoblja s doktoratom iz PI ili srodnih područja važan je čimbenik za potpunu usklađenost s dokumentom IEEE/ACM SWEBOK. Iako sveučilišta nude magistarske programe PI u sklopu srodnih područja, postoji samo nekoliko doktorskih programa PI. Bogati izbor mogućnosti zapošljavanja, koji se nudi diplomantima PI, još je jedan od razloga zašto je akademski put neizbježno nezanimljiv ovoj struci.

Kako bi nastavni plan i program PI bio atraktivniji i učinkovitiji, sljedeće točke treba uzeti u obzir:

- uska suradnja s tehnološkim parkovima u smislu razvoja nastavnog plana i programa te stažiranja i ljetne prakse
- uključivanje razvojnih inženjera za softver i izvršnih direktora, kao i bivših studenata, u proces razvoja nastavnog plana i programa
- partnerstva s vodećim softverskim tvrtkama putem uključivanja razvoja programskih paketa, alata za testiranje i okvira u nastavni plan i program
- omogućavanje studentima da se uključe u sve faze razvojnog ciklusa i trajanja programskog paketa uvođenjem semestralnih projekata paralelno s fazama razvoja programskog paketa
- upoznavanje i blisku suradnju s profesionalnim nevladinim udrugama vezanim uz programsko inženjerstvo kao što su IEEE, ACM, *Software Engineering Institute*, a na nacionalnoj razini YASAD (*Software Industrialists Association*), TBD (Tursko informatičko društvo), TBV (Turska informatička zaklada) i tako dalje.

Razlozi za odstupanja u primjeni IEEE/ACM smjernica mogu biti, na primjer, nedostatak stručnog nastavnog kadra u specijaliziranim područjima kao što su razvoj programskog paketa/održavanje, arhitektura softvera i tako

Vol: 13 (1/2011)
page: 188-219

dalje. Nadalje, zanimljivo je napomenuti da su, osim jednog, sva sveučilišta koja provode dodiplomsko obrazovanje PI u privatnom sektoru. Čini se da su značajni faktori u tom pogledu fleksibilnost pri zapošljavanju te plaće i beneficije. Tijekom posljednjih pet godina tehnološki parkovi su također pomogli diplomantima PI pri zapošljavanju, stažiranju i odrađivanju ljetne prakse u nekoliko tvrtki. Prilikom revidiranja nastavnih planova i programa PI primijećeno je da u obrazovanju programskih inženjera postoji potreba za profesionalnim razvojem i kolegijima vezanim uz etiku.

Trenutno u Turskoj ne postoji akreditacija obrazovnih programa PI, slična onoj za Računalno inženjerstvo, ali se očekuje da će u budućnosti MUDEK (Udruga za ocjenu i akreditaciju inženjerskih programa u Turskoj) pokrenuti određene postupke u tom smjeru. Prilikom provedbe nastavnog plana i programa, na primjer, posebice pri određivanju redoslijeda (uvjetnih) kolegija važno je naglasiti da bi studenti trebali početi s temeljnim kolegijima, tako da mogu lakše pratiti nastavu na zahtjevnijim kolegijima. Na primjer, poznavanje osnova programskog inženjerstva, objektno usmjerenih koncepata, inženjerstva zahtjeva i kvalitete programskog paketa važno je za kolegij arhitektura softvera. Značaj stažiranja, ljetne prakse, projekata tijekom završne godine studija, studija slučaja i industrijskih izvješća/bijele knjige trebali bi biti sastavni dio nastavnog plana i programa. Nove tehnologije i alate treba uvesti kao dio vježbi za sve praktično orijentirane kolegije kao što su "provjera valjanosti i testiranje programskog paketa", "inženjerstvo zahtjeva", "prilagodljivi razvoj softvera", "modeliranje programskog paketa" i slično. Studije slučaja i posebna predavanja ljudi zaposlenih u softverskoj industriji mogu neizmjerno pomoći u razjašnjavanju pojmova zajedno s primjenama u kolegijima kao što su " arhitektura softvera", "upravljanje softverskim projektima", "softverski procesi i poboljšanje", "interakcija čovjek-računalo" i tako dalje. Seminari i radionice mogu unaprijediti znanje etike, omogućiti profesionalni razvoj i usvajanje praktičnih znanja. Na kraju, studente bi trebalo uputiti da uvažavaju i usvajaju softverske alate iz otvorenog koda u svim kolegijima.

ZAKLJUČAK

Programsko inženjerstvo postupno postaje zrela znanstvena disciplina, sve značajnija u svim područjima života. Potražnja za programskim inženjerstvom je u porastu, a time i potražnja za učinkovitim i obrazovanim

programskim inženjerima. Suradnja između softverske industrije i odgovarajućih sveučilišnih odsjeka može ostvariti sinergiju obaju partnera u ostvarenju njihovih ciljeva. U ovom su radu svi preddiplomski PI studiji na turskim sveučilištima ocijenjeni s obzirom na IEEE/ACM smjernice predstavljene u dokumentu SEEK (2004). Međutim, kao i kod drugih istraživanja, postoje neka ograničenja u našoj studiji, ponajprije jer su podatci uglavnom prikupljeni s web stranica sveučilišta. Iako su podatci detaljno analizirani, postoji mogućnost pogrješke u procjeni pri razmatranju djelomične pokrivenosti sadržaja kolegija, na primjer, analiziranjem samo jedne jedinice u modulu umjesto dvije. Vjerujemo da će ovo istraživanje pružiti smjernice sveučilištima koja imaju preddiplomski studij PI na koji način svoje nastavne planove i programe mogu uskladiti s IEEE/ACM smjernicama (SEEK, 2004). Razvoj društvenih, tehničkih i etičkih vještina prilikom timskog rada i odgovornost mogu se također dalje izgrađivati i dodati znatnu vrijednost ključnim tehničkim vještinama u obrazovanju programskih inženjera kako bi diplomanti po završetku studija bili kvalitetni stručnjaci.