Croatian Journal of Education
Vol: 14 (3/2012), pages: 595-624
Review paper
Paper submitted: 18th October 2011
Paper accepted: 26th February 2012

Industry Oriented Advanced Software Engineering Education Curriculum

Alok Mishra and Deepti Mishra
Department of Computer & Software Engineering, Atilim University

Abstract

Software engineering is the fastest-evolving engineering discipline and most of the tasks of software development organizations are diverse in nature. Various studies have shown that there is a wide gap between software industry needs and education for prospective software engineers. It is the responsibility of Software engineering education to prepare SE professionals by providing them with the skills to meet the expectations of the software industry. SE curriculum should correspond to the industry needs, and only then can Universities produce highly skilled professionals, who can meet the needs of software industry. During the last decade, software engineering education (SEE) has been emerging as an independent and mature discipline. Accordingly, various studies are being conducted to provide guidelines for SEE curriculum design. This paper summarizes the need for software industry related courses and discusses the significance of industry oriented software engineering education to meet the educational objectives of all stakeholders. The software industry oriented curriculum for undergraduate and graduate levels is discussed. An industry oriented graduate level (master's level) software engineering course which includes foundational and applied courses to provide effective training for future software engineers is also proposed. This will lead to an increase in their employment prospects in the industrial and allied sectors.

Key words: *Curriculum, Software Engineering Education (SEE), Software Engineering, Software Industry*

Introduction

Software Engineering (SE) is becoming popular and is moving towards maturity. 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 facilitate the student in nurturing the essential knowledge, skills and attitudes, which are actually needed by the industry (Shaw et al., 2005). According to Moreno et al. (2012) aligning SE education with industry is a considerable challenge. Recent studies by Kitchenham et al. (2005), Lethbridge et al. (2007) and Moreno et al. (2012) also identified this issue. It is common to hear the complaints of software engineering companies about the practical knowledge of the students who start working after the completion of their academic programmes (Mishra & Yazici, 2011). While students can have a high level of theoretical knowledge, they often lack the practice of solving real-life industrial problems (ibid). Complaints about software quality are becoming common and this depends on many factors and partially on the university which does not teach essential skills (Jaakkola et al., 2006). Many of the challenges associated with software engineering education are due to our inability to provide students with the '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 function of the quality of the SE education (SEE) (Mishra et al., 2007). We are aware that SE is the fastest-evolving engineering discipline and most of the software development organizations tasks are diverse in nature (Mishra & Yazici, 2011). The pace of changes in SE is substantially higher, broader, and quicker than the impact of other disciplines and the most important feature is its ability to provide tools and methods throughout society, engineering and engineering education (Kral & Zemlicka, 2008). In this context, it is the responsibility of SEE to prepare SE professionals by providing them with skills to meet the expectations of the software industry. Several researchers have already stressed the existing gap between software engineering education and industry needs (Lee & Han, 2008; 2006; Aasheim et al., 2009). To meet the industry requirements, professional curriculum design must be based on the requirements of the software industry. Therefore, in order to effectively fill this gap, it is necessary, on the one hand, to assure that educational programmes provide the knowledge required for the job profiles suggested by industry and also ensure that it is imparted in a manner enabling future professionals to correctly handle the problems that they will face during their professional career (Loftus et al., 2011). The process of industry oriented education must be characterized by the involvement of industry oriented staff and the development and implementation of an education quality assurance with suitable procedures and a strong industry involvement along with key programme learning outcomes (Tao et al., 2010). Further, if the students are well-versed in emerging technologies, in-house training duration will be shorter, which means less time and money for organizations (Mishra et al., 2007). Jaakkola et al. (2006) also supported the claim that SE curriculum should correspond to the industry needs, because only then can universities produce highly skilled professionals who can meet the needs of the software industry. They further argued that the development of curricula should take into account different standards,

frameworks, and recommendations developed by different interest groups. The collaboration between the software industry and higher education departments can lead to synergies for both in accomplishing their objectives (Mishra & Yazici, 2011). Tao et al. (2010) recommended that industry oriented programme should include the following:

1) A good programme concept
   a) a design team with both academic and industrial experts
   b) aimed at market demand
2) Taught programmes, including- Programme documentation, validation process, programme committee, complete quality assurance system and industrial visit/ placements/projects
3) Cyclic optimizing mechanism
   a) fit the industrial demand change
   b) review and feedback on the education process 2 or 3 years.

The objective of this paper is to examine the issues of industry oriented SE education and provide recommendations for graduate (master's) level SE education.

The remainder of the paper is organized as follows: in Section 2, significant issues of industry oriented software engineering education are discussed. In Section 3, recommendations for graduate level SE education are presented. Finally, the paper presents the conclusion in Section 4.

Industry Oriented Software Engineering Education

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 (Barzilay 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 and Swebok, 2008).

Industrial Exposure in SE Education

The projects should be assigned to students with the aim of providing experience in developing a medium sized software engineering project in a small team of four or five students. The teams should take real projects including the innovative ones provided by local businesses to give students industry experience. This will also provide students with an in-depth knowledge of their project domain and students will acquire the confidence and ability to apply that knowledge in practice. Regarding the evaluation of students' projects, Hayes et al. (2003) concluded that a good grading scheme must take into account a range of information, not just rely on the final product. The challenge is to come up with a system to grade the students in such a way that the same grading criteria can be applied to all students although they have worked on different projects (Clark, 2005). Gates et al. (2000) identified the importance of structuring individual accountability to ensure that all members of a team contribute to the project. The

scheme must include an approach for measuring the individual effort for each team member. McGourty et al. (1998) suggested multi-source assessment as it provides critical information from several sources, such as peers, self and faculty members, on student competences and specific behaviour and skills, providing a student with a better understanding of personal strengths and areas in need of development. The Software Engineering project also focuses on giving the students opportunities to communicate effectively.

Industrial practices can be included in SE curriculum in the following ways:

1. Process and product development dimensions can be assigned to teams to discuss issues like software process and quality, software architecture and requirements issues.
2. Small projects/assignments should be given as a part of a software project management course in which students can use different software project management tools to make different charts and diagrams.
3. Class discussions can be related to fundamentals, theory and practice.
4. The industry trend is communicated in various ways through invited talks, mailing lists and technology based forums.

According to Barzilay et al. (2009) the four axes of the course framework provide educators with the ready means of adapting the framework to their needs by adding, removing or changing individual modules, as long as the axis rationale is kept. These four axes are: 1) Fundamentals of SE, 2) Practices and Tools 3) Productization 4) Technology Evolution.

Hilburn et al. (2006) strongly suggested that "Case studies are of special value in problem-based learning, which concentrates on the development of problem-solving skills, and team skills". Group projects, as important educational components, are used for teaching students various teamwork skills (Su et al., 2007). Su et al. (2007) observed, however, that the importance of supplying students with the "real world" teamwork environment is largely ignored in these academic group software development projects. Group projects have been widely adopted in many undergraduate and post graduate courses in software engineering, computer science, and information technology. The major motivation for group projects within a higher level of education is to simulate a "real world" setting, which has been identified as an important characteristic of successful group project courses (Prey, 1995). Therefore, industrial case studies can be helpful in understanding different real life aspects for future software engineers. Industry-relevant case studies and projects along with industry internship should be an integral component of the SEE curriculum (Saiedian, 1999; Wohlin & Regnell, 1999).

Association between the Software Industry and Academia

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 the key areas of SE such as: a) software

development models; b) requirements engineering; c) software architecture and high-level design; d) software processes; e) software quality assurance and management; f) software project management; g) managing people, organizations and working in teams; h) software testing; etc (Mishra & Yazici, 2011) .

According to the observations from Lethbridge's study, the following topics are very important for the software industry; however, generally, students could learn these topics during job learning (Lethbridge, 2000a):

- object-oriented concepts and technologies,
- requirements gathering and analysis,
- analysis and design methods,
- testing verification and quality assurance,
- project management,
- human–computer interaction/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 the curriculum along with the real industrial practice. Another important issue with practice experience are the skills needed to work in groups. In the real software project environment people need to work in groups. Kitchenham et al. (2005) also support several observations mentioned by Lethbridge with respect to the over-emphasis of mathematical topics and the under-emphasis of business topics. However, their findings are different from Lethbridge's with respect to the topics that have a larger knowledge gap (Mishra et al., 2007). SE students should have skills to work individually as well as in 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 (Mishra & Yazici, 2011). Also, it is important that the industry client must be prepared to accept the outcome of a failed project (Hogan et al., 2005). Usually students work on projects in teams and different teams may come up with different solutions. These solutions are discussed by the entire class. Students analyze various solutions and present their approach and during this process, discussions facilitate learning and communication skills (Kornecki et al., 2003). 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 (ibid). Kornecki et al. (2003) further observed that industry is competing for college graduates with the skills to meet the challenges of developing safety-critical software-intensive systems as it is required to get 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 & Regnell (1999) presented strategies for industrially relevant education for software engineers. There are a variety of ways in which the industry and academia can cooperate and many academic institutions have installed departmental or other academic unit-wide industrial advisory boards (Mishra & Yazici, 2011). These groups, consisting of managers and engineers from industries closely associated with the academic organization, are one of the vehicles that provide feedback on an academic programme direction. Including other typical forms of cooperation are occasional short-term projects meeting the current needs of industry and allowing faculty and students to become familiar with the domain and often contribute to the industrial partner's goals (Kornecki et al., 2003). Other modes of cooperation are student summer practices and faculty internship programmes (Mishra & Yazici, 2011). These facilitate understanding of industry needs and allow the university to make appropriate programme revisions (ibid). All these approaches can be effective but do not present a stable result. 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 software engineering (Dey & Sobhan, 2007).

The most recent ACM model curricula (ACM/IEEE, 2005) recognized the different perspectives of academia and industry. They recommended teaching in our computing curricula those technical and non-technical skills required for large software development. It is also suggested to present students "with an appropriate range of applications and case studies that link theory and skills learned in academia to real-world occurrences to illustrate their relevance and utility" (ACM/IEEE, 2005, p. 36).

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 & Ford, 1989). Graduate education is a key element in advancing the state of professional software engineering practice (Ardis et al., 2011). Recently the Integrated Software & Systems Engineering (ISSEC) Curriculum project at Stevens Institute of Technology has developed a set of guidelines for master's degree programmes, titled Graduate Software Engineering 2009 (GSwE2009): Curriculum guidelines for graduate degree programmes in software engineering (Pyster, 2009). Instructional material design and content creation should take into account Bloom's levels of learning (Bloom et al., 1956). Since knowledge is assimilated 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).

Designing the curriculum should take the following issues into consideration (Pinto, 2010):

1. Identify job opportunities and duties involved
2. Identify personality traits (soft-skills) required for the job
3. Identify the competences required to perform the job
4. Identify the knowledge, know-how and current skills required to achieve these competences
5. Identify corresponding subject concepts (theory) to acquire the required knowledge base
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 5, 6 and 7 above.
9. Design assessments that can check a candidate's knowledge at various levels for a given competence.

Although the fundamental conceptual knowledge in IT remains the same, the advances made in the technological area are varied and are changing swiftly (Mishra & Yazici, 2011). Therefore, the skills set that the learner acquires must match the one that the industry currently requires (ibid). The curriculum design must be extensive enough to take this into account (Pinto, 2010). Pinto (2010) suggested that a professionally designed curriculum would require inculcating two main types of competences:

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

Modern SE education is driven by an expectation that the industry's 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 and developing software that is of benefit to a real client is in itself a strong motivator (Tomayko, 1987), as is the use of leading-edge commercial tools in the projects. Within the computer science model curricula designed by the joint IEEE-CS/ACM task force, software engineering is divided into eight core areas and four elective areas:

Core Areas:

- Software Design
- Using APIs
- Software Tools and Environments
- Software Processes
- Software Requirements and Specifications
- Software Validation

- Software Evolution
- Software Project Management
- Elective Areas:
- Component-based Computing
- Formal Methods
- Software Reliability
- Specialized Systems Development

Koska & Romano (1988) recommended that the 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 the 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 by continuous updating of educational programmes and engineering curricula (Waks & Frank, 2000).

Recommendations for Graduate Level SE Education

SE Curriculum Design

Software Engineering deals with the creation and application of engineering fundamentals for systematic and team-based analysis, development, use, evaluation, etc. of large, software-intensive systems as technical product (Horn & Kupries, 2003). The need for highly qualified specialists being capable of mastering software systems and adopting different tools throughout the entire process is the reason for the emerging software engineering as a separate discipline from computer science (Horn & Kupries, 2003).

- The Curriculum should be designed keeping in view the needs of the industry which will provide employment to graduates.
- The structure of the individual courses (Obligatory and Optional) should follow the models suggested by various authorities and should take into account different prerequisites.
- The Curriculum should cover important areas of SE.
- Industrial projects should be an integral part of the Curriculum.
- Students should be able to understand the course and its significance.

The curriculum design is somewhat influenced by commercial companies, who offer universities hardware and software of low prices, often together with ready-made courses where this hardware and/or software is used. IBM offers free course materials, training and curriculum development to universities in its academic initiative programme (IBM, 2006). Similarly, Microsoft has provided financial support for creating an SE Curriculum in specific areas (mainly for US universities) for several years: currently proposing $ 1 million to assist in developing courses in computer

science, business and law that focus on secure computing (which is a recent popular area). Earlier, Microsoft was providing $480,000 to computer gaming curriculum (Microsoft, 2006).

While developing the curriculum, it is important to determine the curriculum objectives and the expected outcomes. Aligning SE education curriculum with industry is a major challenge. Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering (SE, 2004) or GSwE2009 (Graduate Software Engineering) have been developed with the relevance of industrial needs in mind (Moreno et al., 2012). Thus, during the initiation of a course both stakeholders (academia, industry) need to collaborate with each other in setting the curriculum. At later stages of the curriculum development, other interest groups and accreditation bodies are also involved. IEEE (SEEK, 2004) provides the comprehensive and latest guidelines for undergraduate programmes in SE. It includes directions for SE curriculum design, delivery, courses and sequences. Other programme implementation issues such as faculty, students, infrastructure and industry co-ordination, assessment and accreditation have also been discussed briefly (Mishra et al., 2007). A framework "Guide to the Software engineering Body of Knowledge-SWEBOK" (Swebok, 2008) defines the software engineering knowledge that professionals should have after four years of course completion. As a body of knowledge, SWEBOK is well structured and comprehensive. However, several SE researchers raised concerns about some issues of SWEBOK. Recently Pyster et al. (2009) also supported that it reflects software development of the late 1990s and early 2000s and, for instance, it does not really address the highly distributed global software development projects and web-based development and agile methods. The Agile Software Development paradigm has become increasingly popular in the last few years, since it claims lower costs, better productivity, better quality and better business satisfaction (Mishra & Mishra, 2011). Further, Software Verification and Validation (V & V) is one of the most important fields of software engineering for developing high quality software. It is also a new concept gradually becoming part of the curriculum of the universities' software and computer engineering departments. ACM classification and IEEE SE 2004 models provide the tools to position the curriculum contents in the map of "computing science", which is especially important from the research point of view (Jaakkola et al., 2006).

The Computing Curricula 2005 overview report is a further attempt of IEEE and ACM joint task force towards providing guidelines in Computer engineering, computer science, information technology, information systems and a software engineering undergraduate course. Efforts have also been initiated towards a graduate course in SE. Based on rigorous literature review, available recent guidelines for Graduate Software Engineering 2009 (GSwE2009) (Pyster, 2009) and consultation with SE academics, the following tables present a good combination of core, optional and industry oriented projects for graduate level SE education.

Table 1. *Core Software Engineering Courses*

Course Title	Credits
Foundations of Software Engineering	3
Software Requirements Engineering	3
Software Project Management	3
Software Engineering Process	3
Software Quality assurance	3
Software Testing and Validation	3
Software Architecture and Implementation	3

Table 2. *Advanced Software Engineering Course*

Course Title (3 credits each)	Credits:
Analysis and Design of Information System	Choose any three courses of 3 credits each
Advanced Computer Networks	
Software Technology Management	
Embedded and Real-time Software Engineering	
Computer Animation and Virtual Reality	
Human Computer Interaction	
Distributed Systems	
Software Evolution (Maintenance)	
Engineering Economics for Software	
Cyber Ethics and Laws	
Data Mining and Warehousing	
Emerging Trends in Software Engineering	

Table 3. *Industrial project*

Software Industrial project (5-6 months duration)	9 credits

GSwE2009 guidelines strongly recommend that students completing the curriculum must be able to understand and appreciate the importance of negotiation, effective work habits, leadership, and good communication with stakeholders in a real life software development environment.

These guidelines further recommend that students demonstrate their accumulated skills and knowledge in capstone experience, which might be a software industrial project, internship/summer training, or a thesis.

According to the recent GSwE2009 guidelines in master's programme the following percentages have been specified as core by CBOK - Master the Core Body of Knowledge:

Ethics and Professionalism (1-2%)
System Engineering (2-3%)
Requirements Engineering (6-8%)
Software Design (9-11%)
Software Construction (1-3%)

Software Testing	(4-6%)
Software Main Concept	(3-4%)
Configuration Management	(2-3%)
Software Eng. Management	(7-9%)
Software Process	(3-4%)
Software Quality	(3-4%)
Non-core Curriculum	(~50%)

Framework for an Industry-Oriented Software Engineering (Undergraduate) Curriculum

O' Leary et al., (2006) surveyed software industry employers in Ireland, the UK and China and found that the ability to learn, the ability to adapt to changing circumstances, and the ability to place developments and emerging technologies in context are the qualities which employers in this industry consider to be the crucial ones. It is also interesting to note a respondent's comment that "throughout their careers, most people will learn any number of new technologies, but it will be much easier to do this if you have a solid knowledge of the fundamentals".

O' Leary et al., (2006) reported on the Emersion industry-oriented software engineering curriculum based on collaboration between Dublin Institute of Technology (DIT), Harbin Institute of Technology (HIT) and University of Wolverhampton (UOW). The Emersion programme has been designed to lead a novice through a four-year programme in order to graduate as an industrially-oriented practitioner. They presented the following framework for the industry-oriented software engineering curriculum:

The **first year** objective is to provide students with foundational programming and problem solving skills, good communication and presentation skills as well as the recognition of the main topics in computing and the role of the software engineer in industry.

During the **second year** it is important to develop the students' abilities in developing the whole systems, thus introducing the problems related to modular and object oriented software development. The significance of design approaches and methodologies is stressed as the key one at this stage.

The **third year** aim is to enhance the students' software development skills to the point where they can develop solutions for reasonably large, industrial types of problems, thus further developing their problem solving and software engineering skills.

The **final year** focuses entirely on preparing the students for the transition into industry. Successful integration into commercial organizations where a graduate would be in a position to become productive within a short time span is the principal objective of this year. In this year students will also complete the work placement programme (industry internship), which runs parallel to their writing of the academic dissertation. By requiring students to complete their dissertation while on the project

placement, with the topic selected and the project supervised (in part) by industrial supervisors, students are presented with an opportunity to begin their transition from the academic to the industrial environment (O' Leary et al., 2006).

A Plan for the Software Industry and Academic Linkage

1. There should be a continuous summer internship programme or one semester industrial project at the software industry research and development facilities.
2. There should be an exchange programme between the faculty and technical personnel from the software industry. The faculty may spend their summer or sabbatical leave at the software industry while the industry personnel may be invited to deliver a series of lectures on their real-life software projects, challenges, recent advances in software, hardware, tools, etc.
3. The establishment of a chair with support from the software industry to encourage industry oriented research and exchange of knowledge among academics and industry personnel.
4. Fellowships, project support, awards and similar financial encouragement for students and faculty should be introduced.
5. There should be a continuous transfer of the research outcomes, solutions and technologies to an industrial partner.
6. Installation of a laboratory with the support of the industrial partner's software engineering areas of interest at the University. This will conduct software engineering related research in areas of mutual interest.
7. Inclusion of courses consistent with the objectives of the industrial partner.

SE Ethics and Professional Practice

Professional practice and continuous education along with the computing-software ethics should also be included in the SE curriculum for both undergraduate and post graduate students. These will provide the information required for prospective software engineers and managers as they will have to manage different types of operations, contracts, outsourcing, etc. in the future (Dey & Sobhan, 2007). The teaching of computing and software ethics should be an integral part of the SE curriculum. Facing ethical dilemmas is an integral part of a software engineer's career and therefore academia should provide such a course and training to prospective SE engineers. The accreditation standards for SE programmes also emphasize these issues. Institutions seeking accreditation in engineering related programmes by ABET are asked to specify how the programmes assure the development of an understanding of the ethical, social, and economic considerations in the professional practice (ABET, 2002). The teaching of SE ethics must focus on the production of software product and should be different from the myriad of abuses that occur via the use of a computer (Towell, 2003). Students should also provide information regarding how to handle confidentiality and conflict of interest situations in SE areas.

Accreditation and SE Programmes

Accreditation is one of the major concerns to judge the merit of a programme. It encourages universities to review their programmes periodically to keep pace with the fast growing requirements changes in business and technology (Dey & Sobhan, 2007). The Accreditation standards for SE programmes also highlight ethical issues. Institutions seeking accreditation in engineering related programmes by the ABET are asked to address the issue of how the programme assures the development of an understanding of the ethical, social, and economic considerations in the professional practice (ABET, 2002). The accreditation process of undergraduate and graduate SE courses is still at its initial stage in many countries. It is expected that the accreditation organizations concerned will provide a vision towards industry oriented courses and projects to all stakeholders in an excellent way in the future.

Conclusion

SE is gradually becoming a mature discipline and increasingly critical in all walks of life. The demand for SE has been increasing but efficient and knowledgeable software engineers are not available. This requires a rigorous in-house training by software development houses. The collaboration between the software industry and academic departments will lead to the synergy of both parties to accomplish their objectives. The university has a research facility but the financial support for that research should come from the industry. This would provide real-life project handling experiences as well as financial incentives for students. The faculty will engage in applied research and make a contribution to the solution of software industry problems. Theoretical research will find a way to appreciate its realization in real software projects. The result of the lab activities can be used in the classrooms for discussions and further enrichment of the curriculum.

There are many reasons for incorporating real, unique industry projects: increased student motivation and confidence; students can explore in-depth ICT areas not covered in the curriculum or the ones covered only in a summarized manner; students can develop increased problem-solving and critical thinking skills, communication skills and business vision (Clark, 2005). It is further suggested that if the students are also allowed to work in teams, the plan must structure individual accountability and derive information from all related sources towards a better understanding of both strengths and areas for further improvement.

References

Aasheim, C.L., Li, L., & Williams, S. (2009). Knowledge and skills requirements for entry level information technology workers: a comparison of industry and academia. *Journal of Information Systems Education* 20, 349–356.

ABET, Applied Science Accreditation (2002). Volume*I*Self-Study Questionnaire 2002-*3rd Edition.* Available at 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.

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

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.

Clark, N. (2005). Evaluating student teams developing unique industry projects. 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, 21-30.

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.

Gates, A. Q., Delgado, N., & Mondragon, O. (2000). A structured approach for managing a practical software engineering course. In *Proceedings of the 30th Annual Frontiers in Education - Volume 01* (October 18 - 21, 2000). FIE. IEEE Computer Society, Washington, DC, T1C/21-T1C/26.

Hayes, J. H., Lethbridge, T. C., & Port, D. (2003). Evaluating individual contribution toward group software engineering projects. In *Proceedings of the 25th international Conference on Software Engineering* (Portland, Oregon, May 03 - 10, 2003). International Conference on Software Engineering. IEEE Computer Society, Washington, DC, 622-627.

Hilburn, T. B., Towhidnejad, M., Nangia, S., Shen, L., & Hilburn, T. (2006). A Case Study Project for Software Engineering Education. 36th ASEE/IEEE Frontiers in Education Conference, San Diego.

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.

IBM. (2006). IBM Academic Initiative. Available at *http://www304.ibm.com/jct09002c/us/en/university/scholars/*

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.

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.

Lee, C.K. & Han, H.J. (2008). Analysis of skills requirement for entry-level programmer/analysis

in fortune 500 companies. *Journal of Information Systems Education* 19, 17–27.

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.

Loftus, C., Thomas, L., & Zander, C. (2011). Can graduating students design: revisited. In:

Proceedings of the 42nd SIGCSE Technical Symposium on Computer Science Education, pp. 105–110.

McGourty, J., Dominick, P., & Reilly, R. R. (1998). Incorporating student peer review and feedback into the assessment process. In *Proceedings of the 28th Annual Frontiers in Education - Volume 01* (November 04 - 07, 1998). FIE. IEEE Computer Society, Washington, DC, 14-18.

Microsoft (2006). Computer game production curriculum 2004 RFP awards. Available at http//research.microsoft.com/ur/us/fundingopps/Gaming_curriculumRFP_awards.aspx.

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

Mishra, A. & Yazici, A. (2011). An Assessment of the Software Engineering Curriculum in Turkish Universities : IEEE/ACM Guidelines Perspective, *Croatian Journal of Education*, Vol 13 (1), 188 – 219.

Mishra, D. and Mishra, A. (2011). Complex Software Project Development: Agile Methods Adoption, *Journal of Software Maintenance and Evolution: Research and Practice*, 23(8), 549-564.

Moreno,A.M., Sanchez-Segura, M-I, Medina-Dominguez, F., & Carvajal, L. (2012). Balancing software engineering education and industrial needs. *J. Syst. Softw.* 85, 7 (July 2012), 1607-1620.

O'Leary, C., Lawless, D., Gordon, D., Haifeng, L., & Bechkoum, K. (2006). Developing a Software Engineering Curriculum for the Emerging Software Industry in China. In *Proceedings of the 19th Conference on Software Engineering Education & Training* (April 19 - 21, 2006). CSEET. IEEE Computer Society, Washington, DC, 115-122. DOI= http://dx.doi.org/10.1109/CSEET.2006.16

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.

Prey, J. C. (1995). Cooperative learning in an undergraduate computer science curriculum. Proceedings of the Frontiers in Education Conference, Proceedings., 3c2.11-3c2.14 vol.2, November 01-04, 1995.

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. 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.

Saiedian, H. (1999). Software engineering education and training for the next millennium. *J. Syst. Softw.*, 1999, 49, 113–115.

SEEK (2004). Curriculum guidelines or undergraduate degree programs in software engineering. 2008 (Online) Available at: http\\sites.computer.org/ccse/.

Shaw, M., Herbsleb, J. D., & Ozkaya, I. (2005, May). Deciding what to design: Closing a gap in software engineering education. Invited paper for Education and Training Track presented at the 27th International Conference on Software Engineering (ICSE 2005), St. Louis, MO.

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.

Tao, W., Ya-ping, C., Feng, N., & Xi-Qian, G. (2010). Study on the Industry Oriented Education for Computing of Ireland and Its Applications, CIT 2010, IEEE Computer Society, 2005-2009.

Towell, E. (2003). Teaching Ethics in the Software Engineering Curriculum. In *Proceedings of the 16th Conference on Software Engineering Education and Training* (March 20 - 22, 2003). CSEET. IEEE Computer Society, Washington, DC, 150.

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, Incek 06836, Ankara, Turkey
alok@atilim.edu.tr

Deepti Mishra
Department of Computer & Software Engineering, Atilim University, Incek 06836, Ankara, Turkey
deepti@atilim.edu.tr

Kurikul za praktično visoko obrazovanje programskih inženjera

Sažetak

Programsko inženjerstvo je najbrže rastuća disciplina inženjerstva i većina zadataka kojima se bave institucije zadužene za razvoj računalnih programa različite su prirode. Razna istraživanja pokazala su da postoji široki jaz između potreba industrije računalnih programa s jedne strane i obrazovanja budućih programskih inženjera s druge strane. Dužnost je obrazovanja u području programskog inženjerstva dobro pripremiti stručne programske inženjere na način da im se omogući uvježbavanje vještina koje odgovaraju potrebama industrije računalnih programa. Kurikul za programsko inženjerstvo trebao bi odgovarati potrebama industrije i tek će tada sveučilišta biti u mogućnosti osposobljavati visoko kvalitetne stručnjake koji mogu zadovoljiti potrebe industrije. Tijekom posljednjeg desetljeća obrazovanje u području programskog inženjerstva javlja se kao samostalna i zrela disciplina. Shodno tomu, provode se razna istraživanja da bi se osmislile smjernice za izradu kurikula za programsko inženjerstvo. Ovaj rad predstavlja potrebu za kolegijima usko povezanima s industrijom računalnih programa i raspravlja o važnosti praktičnog obrazovanja programskih inženjera da bi se udovoljilo obrazovnim potrebama svih sudionika u tom procesu. Raspravlja se o kurikulu s bitnim elementima praktične nastave na dodiplomskom i diplomskom stupnju obrazovanja. Diplomski studij (magistarski studij) programskog inženjerstva koji uključuje stručnu praksu također se predlaže jer integrira i osnovne teorijske i praktične predmete kojima se postiže uspješno obučavanje programskih inženjera. To će dovesti do boljih izgleda za njihovo zapošljavanje u industrijskim i sličnim sektorima.

***Ključne riječi**: kurikul, obrazovanje u području programskog inženjerstva, programsko inženjerstvo, industrija računalnih programa*

Uvod

Programsko inženjerstvo postaje sve popularnije i polako postiže svoju zrelost. Inovacije i poboljšanja kurikula, upute i procjenjivanje znanja kreću se k premošćivanju jaza između sveučilišta i industrije tako što prikazuju pravu prirodu razvoja računalnog

programiranja i olakšavaju studentima razvijanje osnovnog znanja, vještina i stavova koji su zaista potrebni industriji (Shaw i sur., 2005). Prema Moreno i sur. (2012) usklađivanje programskog inženjerstva i industrije predstavlja značajan izazov. U istraživanjima Kitchenhama i sur. (2005), Lethbridgea i sur. (2007) i Morena i sur. (2012) također su prepoznati isti problemi. Često se mogu čuti pritužbe kompanija koje se bave programskim inženjerstvom na praktično znanje studenata koji počinju raditi nakon završetka akademskog studija (Mishra i Yazici, 2011). Iako studenti imaju visoki stupanj teoretskog znanja, često im nedostaje prakse u rješavanju stvarnih, svakodnevnih problema industrije (isto). Pritužbe na kvalitetu programa su sve učestalije, što ovisi o mnogobrojnim faktorima, ali djelomično i o sveučilištu koje ne poučava studente ključnim vještinama (Jaakkola i sur., 2006). Mnogi izazovi obrazovanja u području programskog inženjerstva postoje zbog naše nemogućnosti da studentima pružimo stvarno, opsežno iskustvo u razvoju programiranja u akademskom okružju (Su i sur., 2007). Stoga je kvaliteta rada programskih inženjera direktna funkcija kvalitete obrazovanja u području programskog inženjerstva (Mishra i sur., 2007). Svjesni smo da je programsko inženjerstvo najbrže rastuća disciplina inženjerstva te je većina zadataka institucija koje se bave programskim inženjerstvom različite prirode (Mishra i Yazici, 2011). Brzina promjena u programskom inženjerstvu je znatno veća, šira i brža nego utjecaj ostalih disciplina, a najvažnija mu je odlika sposobnost da pruži alate i metode cijelom društvu, inženjerstvu i obrazovanju inženjera (Kral i Zemlicka, 2008). U tom kontekstu dužnost je obrazovanja u području programskog inženjerstva pripremiti stručne programske inženjere tako što će im pružiti usvajanje vještina koje industrija računalnih programa treba i očekuje. Nekolicina istraživača već je naglasila postojanje jaza između obrazovanja u području programskog inženjerstva i potreba industrije (Lee i Han, 2008; 2006; Aasheim i sur., 2009). Kako bi se udovoljilo potrebama industrije, stručni kurikul mora se temeljiti na potrebama industrije računalnih programa. Stoga, da bi se učinkovito premostio jaz između obrazovanja i potreba industrije, potrebno je, s jedne strane, osigurati obrazovne programe koji će pružati znanja neophodna za profile zanimanja koje predlaže industrija i također osigurati da se programi primjenjuju tako da osiguraju budućim stručnjacima potrebne vještine pomoću kojih će uspješno rješavati poteškoće na koje će nailaziti u svom poslu (Loftus i sur. 2011). Sastavni dio procesa praktičnog obrazovanja moraju biti i profesori koji posjeduju određena praktična znanja, te osiguranje kvalitete obrazovanja pomoću odgovarajućih postupaka i značajne uključenosti industrije uz ključne ishode obrazovanja (Tao i sur. 2010). Nadalje, ukoliko su studenti dobro upoznati s novim tehnologijama, praktična će obuka u ustanovi biti kraća, što znači manje vremena i novca za organizacije (Mishra i sur., 2007). Jaakkola i sur. (2006) se također slažu da bi kurikul za obrazovanje programskih inženjera trebao odgovarati potrebama industrije, jer samo u tom slučaju sveučilišta mogu obrazovati visoko kvalificirane stručnjake koji mogu odgovoriti na potrebe industrije računalnih programa. Nadalje tvrde da bi izrada kurikula trebala uzeti

u obzir različite standarde, okvire i preporuke raznih interesnih grupa. Suradnja industrije računalnih programa i odsjeka obrazovnih institucija može dovesti do sinergije za obje strane pri postizanju ciljeva (Mishra i Yazici, 2011). Tao i sur. (2010) preporučuju sljedeće elemente koje bi praktični programi trebali uključivati:

1) Dobar koncept programa
   a) Dizajnerski tim sastavljen od znanstvenika i praktičara
   b) Usmjerenost prema potražnji tržišta
2) Programi koji se poučavaju, uključujući dokumentaciju programa, postupak validacije, programski odbor, potpun sustav osiguranja kvalitete i posjete/praksa/projekti u suradnji s industrijom
3) Mehanizam cikličkog optimiziranja
   a) Usklađen s promjenama u potrebama u industriji
   b) Pregled i povratna informacija o obrazovnom procesu 2 ili 3 godine.

Cilj ovoga rada je istražiti probleme praktičnog obrazovanja u području programskog inženjerstva i pružiti preporuke za diplomski (magistarski) stupanj obrazovanja u području programskog inženjerstva.

Ostatak rada podijeljen je na sljedeći način: u 2. poglavlju raspravlja se o važnim pitanjima i problemima praktičnog obrazovanja u području programskog inženjerstva. U 3. poglavlju daju se preporuke za diplomski studij programskog inženjerstva. Na kraju se daje zaključak u poglavlju 4.

Problemi praktičnog obrazovanja programskih inženjera

Programsko inženjerstvo je višedimenzionalno područje koje uključuje aktivnosti u raznim područjima i disciplinama kao što su informacijska znanost, upravljanje projektima, arhitektura informacijskih sustava, ljudski faktori te razvoj tehnologije (Barzilay i sur., 2009). Do sada je bilo nekoliko pokušaja određivanja dimenzija programskog inženjerstva i stvaranja kvalitetnog kurikula koji ih sve uzima u obzir (SEEK, 2004 i Swebok, 2008).

Praktično iskustvo u obrazovanju programskih inženjera

Studentima bi se, u četveročlanim ili peteročlanim timovima, trebali zadavati projekti s ciljem stjecanja iskustva u razvoju srednje opsežnih programerskih projekata. Timovi bi trebali dobiti stvarne projekte, uključujući i one inovativne koje bi omogućile lokalne firme da bi studentima pružile pravo iskustvo rada u njihovoj grani industrije. To bi također omogućilo studentima detaljno znanje domene njihovog projekta te bi studenti stekli pouzdanje i sposobnost da praktično primjene stečeno teorijsko znanje. Što se tiče ocjenjivanja projekata dodijeljenih studentima Hayes i sur. (2003) dolaze do zaključka da bi dobro razrađen plan ocjenjivanja morao uzeti u obzir čitav niz čimbenika, a ne samo procjenu završnog proizvoda. Izazov je kako osmisliti sustav ocjenjivanja studenata na takav način da se isti kriteriji ocjenjivanja mogu primijeniti

na sve studente iako su radili na različitim projektima (Clark, 2005). Gates i sur. (2000) uvidjeli su važnost uvođenja pojedinačne odgovornosti da bi bili sigurni da su svi članovi tima doprinijeli razvoju projekta. Taj plan ocjenjivanja mora uključiti i pristup za određivanje uloženog truda svakog pojedinca u timu. McGourty i sur. (1998) predložili su višestruko ocjenjivanje jer ono pruža kritičke informacije od strane nekoliko izvora kao što su kolege i članovi sveučilišne zajednice. Te informacije odnose se na kompetencije, određen obrazac ponašanja i vještine studenata, pružajući studentu bolje razumijevanje njegovih osobnih snaga i područja na kojima bi trebao poraditi. Projekt programskog inženjerstva također je usmjeren k pružanju prilike studentima da nauče uspješno komunicirati.

Praktični rad se u kurikul obrazovanja programskih inženjera može uključiti na sljedeće načine:

1. Aspekti procesa i razvoja proizvoda mogu se dodijeliti timovima koji će raspravljati o pitanjima kao što su proces i kvaliteta programa, programska arhitektura i zahtjevi.
2. Manji projekti/zadaci trebali bi se zadavati kao dio kolegija o upravljanju programskim projektima u kojemu bi studenti koristili raznovrsne alate upravljanja programskim projektima da bi izradili razne tablice i dijagrame.
3. Diskusije unutar veće grupe studenata bile bi usmjerene na osnove, teoriju i praksu.
4. Trendovi u industriji predstavljaju se organiziranjem predavanja, preko mailing-lista, te foruma o tehnologiji.

Prema Barzilay i sur. (2009) četiri okosnice okvira kolegija pružaju nastavnom osoblju već gotove načine prilagođavanja okvira svojim potrebama dodavanjem, uklanjanjem ili mijenjanjem pojedinih modula, sve dok se omjer okosnica ne mijenja. Ove četiri okosnice su: 1) osnove programskog inženjerstva; 2) Praksa i alati 3) Produktizacija i 4) Razvoj tehnologije.

Hilburn i sur. (2006) čvrsto su uvjereni da su „analize slučaja od posebne važnosti u problemskom učenju koje je usredotočeno na razvoj vještina za rješavanje problema, kao i razvoj vještina za dobar timski rad." Grupni projekti se, kao važne obrazovne komponente, koriste za poučavanje studenata raznim vještinama potrebnima za rad u timu (Su i sur., 2007).

Su i sur. (2007) su, međutim, primijetili da se važnost upoznavanja studenata sa stvarnim radnim okružjem timskog rada često zanemaruje u akademskim projektima programiranja. Grupni projekti se uvelike prihvaćaju u mnogim dodiplomskim i poslijediplomskim kolegijima u području programskog inženjerstva, informacijske znanosti i informacijske tehnologije. Glavna motivacija u grupnim projektima na visokom stupnju obrazovanja je simulacija stvarnog radnog okružja, za što je utvrđeno da je glavno obilježje kolegija koji podrazumijevaju grupne projekte (Prey, 1995). Stoga analize slučajeva iz industrije za buduće programske inženjere mogu biti od velike pomoći pri razumijevanju raznih stvarnih aspekata života. Takve analize slučajeva i

takvi projekti u kombinaciji sa stažiranjem u struci trebali bi biti sastavni dio kurikula u području obrazovanja programskih inženjera (Saiedian, 1999; Wohlin i Regnell, 1999).

Povezivanje industrije računalnih programa i sveučilišta

Postoje ozbiljni problemi s kvalitetom programa i troškovima koji nastaju njihovom proizvodnjom (Ford i Gibbs, 1996) te se zaposlenici industrije računalnih programa žale na znanje i vještine apsolvenata u nekim ključnim područjima programskog inženjerstva kao što su: a) modeli razvoja programa; b) inženjering zahtjeva; c) programska arhitektura i napredni dizajn; d) programski procesi; e) osiguravanje kvalitete programskog proizvoda i upravljanje istim; f) upravljanje programskim projektima; g) upravljanje ljudima, organizacijama i timskim radom; h) testiranje programa itd.

Prema opažanjima opisanima u Lethbridgevom istraživanju, sljedeće teme su jako važne za programsku industriju, iako bi ih, općenito govoreći, studenti mogli naučiti tijekom stručne prakse (Lethbridge, 2000a):

- Objektno-orijentirani pojmovi i tehnologije;
- Prikupljanje i analiza zahtjeva;
- Metode analize i dizajna;
- Ovjera testiranja i osiguravanje kvalitete;
- Upravljanje projektima;
- Interakcija čovjek-računalo/korisničko sučelje;
- Baze podataka;
- Upravljanje konfiguracijom i izdanjem;
- Etika i profesionalizam;
- Tehničko pisanje;
- Održavanje prezentacija/seminara slušateljstvu;
- Vještine vodstva.

Prema Lethbridgeu (2000b) postoji potreba da se svi ovi predmeti uključe u kurikul zajedno s pravom stručnom praksom. Drugo važno pitanje koje se tiče praktičnog iskustva su vještine potrebne za grupni rad. U stvarnom okruženju izrade programskog projekta ljudi moraju raditi u grupama. Kitchenham i sur. (2005) također se slažu s nekoliko Lethbridgeovih zapažanja koja se tiču prevelikog naglašavanja matematičkih tema a nedovoljnog bavljenja poslovnim temama. Međutim, njihova opažanja razlikuju se od Lethbridgeovih u temama u kojima je vidljiv veći raskorak u znanju (Mishra i sur., 2007). Studenti programskog inženjerstva trebali bi posjedovati vještine samostalnog rada jednako kao i vještine timskog rada da bi razvijali i proizvodili kvalitetne programske artefakte (Kurikul obrazovanja programskih inženjera, 2004).

Izbor odgovarajućeg industrijskog partnera koji će omogućiti intrigantan i motivirajući projekt od presudne je važnosti za industrijske projekte (Mishra i Yazici, 2011). Također je važno napomenuti da klijent mora biti spreman prihvatiti i rezultate propalog projekta (Hogan i sur., 2005). Obično studenti na projektima rade u timovima

pa različiti timovi mogu doći do različitih rješenja. O tim rješenjima raspravlja cijela grupa te studenti analiziraju razna rješenja i predstavljaju svoj pristup. Tijekom ovoga procesa razgovori olakšavaju učenje i razvijaju komunikacijske vještine (Kornecki i sur., 2003). U većini slučajeva industrija je zainteresirana za aktivnosti primjene znanja koje donose izravna rješenja, pomažu u provođenju novih proizvoda ili poboljšavaju najvažnije čimbenike svakodnevnog rada (isto). Kornecki i sur.(2003) su nadalje opazili da se industrija otima za apsolvente koji posjeduju vještine odgovaranja na izazove razvijanja sigurnosno kritičnih sustava programske arhitekture jer se zahtijeva stalan priljev kvalificiranog osoblja u ovim područjima.

Još jedan element od presudne važnosti za uspjeh plana programskog inženjerstva je uključivanje i aktivno sudjelovanje industrije (Kurikul obrazovanja programskih inženjera, 2004). U ovom kontekstu, Wohlin i Regnell (1999) predstavili su strategije obrazovanja programskih inženjera koje su jako bitne za industriju. Postoje mnogi načini na koje sveučilište i industrija mogu surađivati. Mnoge akademske institucije su imenovale industrijske savjetodavne odbore u nekim odjelima ili u cijelom sveučilištu (Mishra i Yazici, 2011). Ti odbori se sastoje od direktora i inženjera iz industrije usko povezane s akademskom organizacijom, te su jedno od sredstava koja daje povratnu informaciju o tome kako se akademski program odvija uključujući i ostale uobičajene oblike suradnje kao što su povremeni kratki projekti koji odgovaraju potrebama industrije. Takvi projekti omogućuju fakultetu i studentima da se upoznaju sa svojim područjem i često pridonesu ostvarenju ciljeva industrijskog partnera (Kornecki i sur., 2003). Dodatni oblici suradnje su ljetna stručna praksa studenata kao i provođenje fakultetskog programa stažiranja (Mishra i Yazici, 2011). Oni omogućuju bolje razumijevanje potreba industrije i omogućuju fakultetu korigiranje nastavnih programa (isto). Svi ovi pristupi mogu biti učinkoviti ali ne predstavljaju trajno rješenje. Akademske institucije trebale bi biti sposobne preurediti i provesti kurikul programskog inženjerstva koji ne samo da naglašava teoretske i tehničke aspekte računarstava, nego se također fokusira na praktični dio programskog inženjerstva (Dey i Sobhan, 2007).

Najnoviji ACM model kurikula (ACM/IEEE, 2005) uočava različite perspektive sveučilišta i industrije. Ti kurikuli preporučuju podučavanje onih tehničkih i netehničkih vještina u kurikulima računarstva koje su potrebne za razvoj velikih programa. Također se predlaže da bi studentima trebalo pružiti „prikladan niz primjena znanja i analiza slučajeva koji povezuju teoriju i vještine usvojene na sveučilištu sa događanjima i pojavama u stvarnom svijetu da bi se zorno predočila njihova važnost i korisnost“ (ACM/IEEE, 2005, str. 36).

Razvoj kurikula za obrazovanja programskih inženjera

Godine 1989. Institut za programsko inženjerstvo (the Software Engineering Institute) Sveučilišta Carnegie Mellon objavio je relevantno izviješće o diplomskom obrazovanju u području programskog inženjerstva (Ardis i Ford, 1989). Diplomski

studij ključan je element u poboljšanju statusa stručne prakse u programskom inženjerstvu (Ardis i sur., 2011). Nedavno je projekt razvoja kurikula Integriranog programskog i inženjerstva sustava na Stevens tehnološkom institutu (Stevens Institute of Technology) predstavio niz smjernica za magistarske studije pod nazivom Diplomski studij programskog inženjerstva 2009 (GSwE2009): smjernice kurikula za diplomski stupanj obrazovanja u području programskog inženjerstva (Pyster, 2009). Izrada instrukcijskih materijala i određivanje obrazovnog sadržaja trebali bi uzeti u obzir Bloomove stupnjeve učenja (Bloom i sur., 1965). Budući da se znanje usvaja na različitim stupnjevima, moduli učenja trebali bi se izrađivati tako da vode učenika od osnovne razine znanja o informacijama i pojmovima do primjene i analiza znanja, te konačno do sinteze i procjene znanja (Pinto, 2010).

Izrada kurikula treba razmotriti sljedeća pitanja (Pinto, 2010):

1. Odrediti izglede za zapošljavanje i pripadajuće dužnosti.
2. Odrediti osobne karakteristike (soft-skills) koje se traže za taj posao.
3. Odrediti kompetencije koje su potrebne za obavljanje posla.
4. Odrediti znanje, praktično znanje i vještine koje su trenutno potrebne da bi se ovladalo tim kompetencijama.
5. Odrediti odgovarajuće predmete/kolegije (teoriju) koji su potrebni da bi se usvojilo traženo osnovno znanje.
6. Odrediti zadatke (praksu) koji povećavaju stupanj praktičnog znanja.
7. Odrediti skup vještina (sredstva) koje su potrebne da bi se zadaci uspješno izvršili.
8. Odrediti literaturu čiji obrazovni ciljevi odgovaraju znanju koje se traži u pitanjima 5., 6. i 7.
9. Izraditi zadatke koji mogu provjeriti znanje kandidata na različitim stupnjevima pojedinih kompetencija.

Iako temeljno konceptualno znanje u informacijskog tehnologiji ostaje isto, ostvareni napredak u području tehnologije raznolik je i brzo se mijenja (Mishra i Yazici, 2011). Stoga, skup vještina koje učenik usvaja mora odgovarati onima koje industrija trenutno koristi i treba (isto). Izrada kurikula mora biti dovoljno sveobuhvatna da ovu činjenicu uzme u obzir (Pinto, 2010). Pinto (2010) je predložio da bi stručno izrađen kurikul zahtijevao uključivanje dviju glavnih vrsta kompetencija:

1. Stručne kompetencije koje se odnose na „osnovno znanje" koje se traži i sposobnost da se to znanje koristi u srodnom području rada.
2. Osobne kompetencije koje predstavljaju skup vještina, stavova i vrijednosti i koje omogućavaju stručnjaku da učinkovito obavlja svoj rad i da se prilagodi svojoj trenutnoj okolini.

Moderno obrazovanje programskih inženjera vođeno je očekivanjima da će najbolja praksa u industriji i najnovije i najbolje programske tehnologije postati sastavni dio kurikula jer potencijalni poslodavci blagonaklono gledaju na praktični projektni rad u industriji (Hogan i sur., 2005). Projekti koji se praktično provode u industriji

podrazumijevaju da studenti surađuju sa stvarnim klijentom koji ima značajan interes za rezultate projekta, a razvoj programa koji su korisni stvarnim klijentima je sam po sebi jak motivirajući faktor (Tomayko, 1987), kao što je i uporaba vodećih komercijalnih alata u projektima. U modelima informatičkih kurikula koje su izradili zajedničke radne skupine IEEE-CS/ACM, programsko inženjerstvo dijeli se na osam ključnih i četiri izborna područja:

Ključna područja:
- Izrada programa
- Uporaba aplikacijskog programskog sučelja
- Programski alati i okolina
- Programski procesi
- Programski zahtjevi i specifikacije
- Provjera ispravnosti programa
- Razvoj programa
- Upravljanje programskim projektima

Izborna područja:
- Komponentno programiranje
- Formalne metode
- Pouzdanost programa
- Razvoj specijaliziranih sustava

Koska i Romano (1988) su dali preporuku da bi se sveučilišni kurikuli trebali znatno promijeniti i staviti naglasak na vještine koje zahtijevaju sustavni pristup i stvaraju široku teoretsku osnovu te vještine koje su potrebne industriji. Prema jednom od tih pristupa, preporuča se određivanje vještina i znanja koje bi apsolventi trebali posjedovati te bi se trebalo ispitati u kolikoj mjeri kurikul odgovara potrebama industrije (Waks, 1995). Brz tempo tehnološkog napretka mora se ostvarivati stalnim dorađivanjem obrazovnih programa i inženjerskih kurikula (Waks i Frank, 2000).

Preporuke za diplomski stupanj obrazovanja u području programskog inženjerstva

Izrada kurikula za obrazovanje programskih inženjera

Programsko inženjerstvo bavi se stvaranjem i primjenom osnova inženjerstva u analizama sustava i analizama timova, u razvoju, uporabi, procjeni itd. velikih programskih sustava kao tehničkih proizvoda (Horn i Kupries, 2003). Potreba za visoko kvalificiranim stručnjacima koji su sposobni ovladati programskim sustavima i koji su sposobni primjenjivati različite alate tijekom cijelog procesa je razlog tomu što se programsko inženjerstvo pojavljuje kao zasebna disciplina informacijske znanosti (Horn i Kupries, 2003).

• Kurikul bi trebao biti izrađen imajući na umu potrebe industrije koje će apsolventima osigurati zapošljavanje.

- Struktura pojedinih kolegija (obveznih i izbornih) trebala bi biti u skladu s modelima koje su predložila razna mjerodavna tijela i trebala bi uzeti u obzir različite preduvjete.
- Kurikul bi trebao obuhvaćati važna područja programskog inženjerstva.
- Industrijski projekti trebali bi biti sastavni dio kurikula.
- Studenti bi trebali biti sposobni razumjeti kolegij i njegovu važnost.

Izrada kurikula djelomično je određena i utjecajem komercijalnih kompanija koje sveučilištima nude hardver i programe po niskim cijenama, što često uključuje i već gotove, priređene kolegije gdje se taj hardver ili programi koriste. IBM sveučilištima nudi besplatne nastavne materijale za kolegije, obuku i izradu kurikula u svojem obrazovnom programu akademske inicijative (IBM, 2006). Slično njima, Microsoft već nekoliko godina pruža financijsku potporu za izradu kurikula za obrazovanje programskih inženjera u pojedinim područjima (pretežno u američkim sveučilištima), a trenutno predlažu potporu od 1 milijun dolara za pomoć u osmišljavanju kolegija u područjima informacijske znanosti, poslovne znanosti i prava koji se bave sigurnim radom na računalima (što je u posljednje vrijeme jako popularno područje). Ranije je Microsoft davao potporu od 480,000 dolara za kurikul o računalnim igricama (Microsoft, 2006).

Pri izradi kurikula važno je odrediti njegove ciljeve i očekivane rezultate. Usklađivanje kurikula za obrazovanje programskih inženjera s potrebama industrije predstavlja velik izazov. Smjernice kurikula za diplomski stupanj obrazovanja u području programskog inženjerstva (Programsko inženjerstvo, 2004) ili GSwE2009 (Diplomski studij programskog inženjerstva) razvijeni su imajući u vidu potrebe industrije (Moreno i sur., 2012). Stoga, tijekom uvođenja kolegija oba sudionika (sveučilište i industrija) trebaju međusobno surađivati u postavljanju kurikula. U kasnijim fazama izrade kurikula uključuju se i druge interesne skupine i ovlaštena tijela. IEEE (SEEK, 2004) pruža opsežne i najnovije smjernice za dodiplomske studije programskog inženjerstva. Također su uključene i upute za izradu kurikula u području programskog inženjerstva, za predaju, kolegije i logičan redoslijed. Ostala pitanja provedbe obrazovnog programa kao što su fakultet, studenti, infrastruktura, koordinacija s industrijom, procjena i ovlasti također su kratko spomenuti (Mishra i sur., 2007). Okvir „Vodič kroz temeljna znanja programskog inženjerstva – SWEBOK" (Swebok, 2008) definira znanje o programskom inženjerstvu koje bi stručnjaci nakon završenog četverogodišnjeg obrazovanja trebali posjedovati. Kao prikaz temeljnog znanja, SWEBOK je dobro koncipiran i sveobuhvatan. Međutim, nekoliko stručnjaka koji su se bavili istraživačkim radom u programskom inženjerstvu izrazili su zabrinutost o nekim pitanjima u SWEBOK-u. Nedavno je Pyster i sur. (2009) također izjavio da SWEBOK odražava razvoj programiranja kasnih 1990.-ih i ranih 2000.-ih te se, npr., ne bavi jako razgranatim svjetskim projektima programskog razvoja, kao ni razvojem mreže i agilnim metodama. Paradigma agilnog postupka razvoja programske mreže postala je izuzetno popularna posljednjih nekoliko godina, budući da tvrdi da omogućuje

niže troškove, bolju produktivnost, bolju kvalitetu i veće poslovno zadovoljstvo (Mishra i Mishra, 2011). Nadalje verifikacija i validacija računalnih programa (V & V) jedno je od najvažnijih polja programskog inženjerstva za razvoj kvalitetnih računalnih programa. Također je i novi koncept koji postupno postaje dio kurikula sveučilišnih odjela programskog i računalnog inženjerstva. Na sveučilištima postoji značajna potreba za strožim poučavanjem verifikacije i validacije računalnih programa studentima na odsjecima programskog i računalnog inženjerstva (Mishra et al., 2012). ACM-ova podjela i modeli IEEE (Institute of Electrical and Electronics Engineers) SE 2004 pružaju alate za uključivanje kurikulskih sadržaja u kartu „računarske znanosti" , što je iznimno važno s istraživačke točke gledišta (Jaakkola i sur., 2006).

Opće izvješće o Kurikulu računarstva 2005 je daljnji pokušaj zajedničke radne skupine IEEE-a i ACM-a u pružanju smjernica za računarski inženjering, informacijsku znanost, informacijsku tehnologiju, informacijske sustave i dodiplomski studij programskog inženjerstva. Također su uloženi napori k organiziranju diplomskog studija programskog inženjerstva. Na temelju stroge prosudbe literature, dostupnih novijih smjernica za Diplomski studij programskog inženjerstva 2009 (GSwE2009) (Pyster 2009) i dogovora sa stručnjacima programskog inženjerstva, sljedeće tablice prikazuju dobru kombinaciju osnovnih, izbornih i praktičnih industrijskih projekata diplomskog obrazovanja u području programskog inženjerstva.

Tablica 1.

Tablica 2.

Tablica 3.

GSwE2009 smjernice izričito napominju da bi studenti koji završavaju nastavu morali biti sposobni razumjeti i prepoznati važnost pregovaranja, uspješnih radnih navika, vodstva i dobre komunikacije sa sudionicima u stvarnom radnom okruženju pri izradi programa.

Ove smjernice nadalje preporučuju da bi studenti trebali pokazati stečene vještine i znanja u praksi kao kruni svojega obrazovanja, a ta praksa može biti praktični programski projekt, stažiranje/ljetna praksa, ili disertacija.

Prema novijim GSwE2009 smjernicama za magistarski studij, CBOK navodi sljedeće postotke kao osnovna znanja u područjima:

Etika i stručnost	(1-2%)
Inženjering sustava	(2-3%)
Inženjering zahtjeva	(6-8%)
Programski dizajn	(9-11%)
Izrada programa	(1-3%)
Testiranje programa	(4-6%)
Glavni koncept programa	(3-4%)
Upravljanje konfiguracijama	(2-3%)

Upravljanje programskim inženjerstvom	(7-9%)
Programski procesi	(3-4%)
Kvaliteta programa	(3-4%)
Ne-osnovni kurikul	(~50%)

Okvir za (dodiplomski) kurikul za praktično obrazovanje programskih inženjera

O'Leary i sur. (2006) radili su istraživanje s poslodavcima u industriji računalnih programa u Irskoj, Ujedinjenom Kraljevstvu i Kini i saznali da su sposobnost za učenje, sposobnost prilagodbe promjenjivim uvjetima te sposobnost da se dostignuća i nove tehnologije smjeste u kontekst kvalitete koje poslodavci u ovoj grani industrije smatraju ključnima. Također je zanimljivo skrenuti pažnju na komentar jednog ispitanika da „tijekom svoje karijere većina ljudi će naučiti bilo kakve nove tehnologije, ali će im to učenje biti puno lakše ukoliko imaju dobro osnovno znanje."

O'Leary i sur. (2006) izvijestili su o kurikulu uranjanja u praktično obrazovanje programskih inženjera koji se zasniva na suradnji između Tehnološkog instituta u Dublinu (DIT), Harbin tehnološkog instituta (HIT) i Sveučilišta u Wolverhamptonu (UOW). Program uranjanja osmišljen je tako da vodi pripravnika kroz četverogodišnji obrazovni program da bi diplomirao kao stručnjak u praksi. Predstavili su sljedeći okvir za kurikul praktičnog obrazovanja programskih inženjera:

Cilj **prve godine** obrazovanja je upoznati studente s osnovama programiranja i vještinama potrebnima za rješavanje problema, dobrim komunikacijskim i prezentacijskim vještinama kao i uočavanjem glavnih tema u računarstvu i ulogom programskog inženjera u industriji.

Tijekom **druge godine** važno je razvijati sposobnosti studenata da izgrađuju čitave sustave te se tako upoznaju s problemima vezanima za modularno i objektno orijentirano programiranje. Važnost pristupa i važnost metoda dizajniranja na ovome stupnju naglašene su kao ključne.

Cilj na **trećoj godini** je razviti kod studenata vještine izgradnje računalnih programa do točke u kojoj oni mogu pronaći rješenja za prilično velike, industrijske vrste problema te tako dalje razvijati svoje vještine rješavanja problema i svoje programerske vještine.

Zadnja godina usredotočuje se potpuno na pripremanje studenata za prijelaz u industriju. Uspješna integracija u komercijalne organizacije gdje je apsolvent u mogućnosti postati poslovno produktivan u kratkom vremenskom roku je glavni cilj ove godine. U ovoj godini studenti će također završiti stručnu praksu/stažiranje koja se zbiva paralelno s pisanjem diplomske radnje. Kroz to što se od njih traži da završe svoju radnju dok odrađuju stručnu praksu koja je dio projekta, o temi koju su odabrali i kroz projekt kojega djelomično nadziru zaposlenici u industriji, studentima se pruža mogućnost da započnu svoj prijelaz iz akademskog u industrijsko okružje (O'Leary i sur., 2006).

Plan za suradnju industrije računalnih programa i sveučilišta

1. Trebao bi postojati stalan program stažiranja tijekom ljeta ili jednosemestralni praktični projekt u istraživačkim objektima industrije računalnih programa.
2. Trebao bi postojati program razmjene fakultetskog i tehničkog osoblja industrije računalnih programa. Fakultetski djelatnici mogli bi provesti ljeto ili studijsku godinu u industriji računalnih programa, dok bi industrijsko osoblje moglo biti pozvano da održi niz predavanja o stvarnim programskim projektima, izazovima, novijem napretku u programima, hardveru, alatima i slično.
3. Osnivanje uprave uz potporu industrije računalnih programa da bi se potaknula praktična istraživanja i razmjena znanja između akademskog i industrijskog osoblja.
4. Trebalo bi uvesti stipendije, podršku projektima, nagrade i slične financijske poticaje za studente i fakultetsko osoblje.
5. Trebala bi postojati stalna razmjena rezultata istraživanja, rješenja i tehnologija s industrijskim partnerom.
6. Osnivanje laboratorija na sveučilištu uz podršku industrijskog partnera i njegovih područja od interesa unutar programskog inženjerstva. To će pomoći provođenju istraživanja u području programskog inženjerstva u područjima od zajedničkog interesa.
7. Uvođenje kolegija koji su u skladu s ciljevima industrijskog partnera.

Etika programskog inženjerstva i stručna praksa

Stručna praksa i kontinuirano obrazovanje zajedno s računalnom i programerskom etikom također bi trebali biti uključeni u kurikul programskog inženjerstva i na dodiplomskom i na diplomskom stupnju. To bi pružilo razne informacije budućim programskim inženjerima i menadžerima jer će oni u budućnosti morati rukovoditi raznim operacijama, ugovorima i uporabom vanjskih resursa (Dey i Sobhan, 2007). Poučavanje računalne i programerske etike trebalo bi biti sastavni dio kurikula programskog inženjerstva. Nailaženje na etičke nedoumice je nezaobilazni dio karijere jednog programskog inženjera te bi stoga sveučilište trebalo pružiti takav kolegij i obuku budućim programskim inženjerima. Standardi službenog odobrenja obrazovnih programa u području programskog inženjerstva također naglašavaju ove aspekte. Institucije koje traže službeno odobrenje inženjerskih obrazovnih programa kod Akreditacijskog odbora za inženjerstvo i tehnologiju (Accreditation Board for Engineering and Technology) trebaju potanko objasniti kako njihovi obrazovni programi omogućavaju bolje razumijevanje etičkih, društvenih i ekonomskih okolnosti u stručnoj praksi (ABET, 2002).

Podučavanje o etici unutar programskog inženjerstva mora se fokusirati na proizvodnju programskog proizvoda i trebalo bi se razlikovati od mnoštva zloporaba koje se događaju pri korištenju računala (Towell, 2003). Studenti bi također trebali znati kako postupati s povjerljivim podatcima i situacijama sukoba interesa u području programskog inženjerstva.

Službeno odobrenje i obrazovni programi u području programskog inženjerstva

Službeno odobrenje je jedan od najvećih problema pri procjeni uspjeha obrazovnog programa. Ono potiče sveučilišta da s vremena na vrijeme revidiraju svoje obrazovne programe da bi išli u korak s brzo rastućim promjenama zahtjeva u poslovnom svijetu i tehnologiji (Dey i Sobhan, 2007). Standardi službenog odobrenja obrazovnih programa u području programskog inženjerstva također naglašavaju etičke aspekte. Institucije koje traže službeno odobrenje inženjerskih obrazovnih programa kod Akreditacijskog odbora za inženjerstvo i tehnologiju trebaju detaljno objasniti kako njihovi obrazovni programi omogućavaju bolje razumijevanje etičkih, društvenih i ekonomskih okolnosti u stručnoj praksi (ABET, 2002). Proces službenog odobrenja dodiplomskih i diplomskih kolegija programskog inženjerstva je još u povojima u mnogim državama. Očekuje se da će organizacije koje daju službena odobrenja dati viziju odlične provedbe praktične nastave i projekata u budućnosti za sve njezine sudionike.

Zaključak

Programsko inženjerstvo postaje zrela disciplina važna u svim sferama života. Potreba za programskim inženjerstvom u porastu je, ali nema dovoljno uspješnih programskih inženjera s velikim znanjem. To zahtijeva strogu obuku unutar programerskih kuća kada se programski inženjeri zaposle. Suradnja između industrije računalnih programa i sveučilišnih odsjeka dovest će do sinergije obiju strana da postignu svoje ciljeve. Sveučilište ima objekte za istraživanje, ali bi financijska potpora za istraživanja trebala doći od industrije. To bi omogućilo stvarna iskustva rukovođenja projektima ako i financijske poticaje za studente. Fakultet će se uključiti u primijenjena istraživanja i dati svoj doprinos rješavanju problema industrije računalnih programa. Teorijska istraživanja će naći put do svojeg oživljavanja u stvarnim programskim projektima. Rezultat laboratorijskih radnji koristit će se u učionicama za razgovore i daljnje obogaćivanje kurikula.

Postoji mnogo razloga za uključivanje stvarnih, jedinstvenih industrijskih projekata: veća motivacija i samopouzdanje studenata; studenti mogu detaljno istraživati područja informacijsko-komunikacijske tehnologije koja nisu uključena u kurikul ili su samo usputno spomenuta; studenti će razviti vještine rješavanja problema i kritičkog razmišljanja, komunikacijske vještine i poslovnu viziju (Clark, 2005). Dalje se predlaže da, ukoliko je studentima također dopušteno raditi u timovima, plan mora razraditi pojedinačnu odgovornost i izvući informacije iz svih srodnih izvora da bi se postiglo bolje razumijevanje jakih područja, ali i onih na čijemu poboljšanju treba poraditi.