CSECS 2017, pp. 000 - 000

The 13th Annual International Conference on
Computer Science and Education in Computer Science,

June 30-July 03 2017, Albena, Bulgaria

THE CONTENT OF SOFTWARE PRODUCTION COURSE

JULIANA PENEVA, STANISLAV IVANOV

New Bulgarian University, Department of Informatics

Abstract: *Software production is an integral activity that encompasses the majority of computer science and information technologies knowledge and skills such as algorithms, programming, databases, networking etc. Referring to this, an introductory course on software production summarizes and upgrades the learning achievements of students in bachelor programs of computing. This paper focuses on some curriculum issues in software engineering course. The main discussion concerns the choice of the topics and learning activities. The course should expose fundamental concepts and principles that underlie current and emerging methods, tools, and techniques for cost-effective production of high-quality software systems.*

Keywords*: software engineering, curriculum issues*

ACM Classification (CCS 2012) Keywords: *Social and professional topics~ Computer science education*

Introduction

Currently an increasing demand for different kind of computer science and information technologies specialists is observed as a steady trend. Software developers, system analysts, database administrators or information systems managers are among the best jobs, evaluated by expected technology advances and salary over the next ten years. These professions are highly demanding and dynamic ones with an increased level of requirements for knowledge, skills, and expertise. With regard to this computer science educators are expected to design and render proper courses that can not only deliver the required knowledge, understanding, and skills but also stimulate lifelong learning. Moreover in any computing application domain, professionalism, quality, schedule, and cost are critical concerning the development of software systems. Because of this the study of software engineering practices is important in bachelor programs of computer science and information technologies. Software engineering (SE) is concerned with all aspects of software development and the accompanying activities as well as the applied tools and methods from the early stages of requirements engineering to maintaining developed systems.

This paper focuses on some didactical issues to the study of software engineering. The main discussion concerns the choice of the topics and learning activities for a course delivered to students majoring computer science or information technologies. Software Engineering is an introductory survey course on the fundamental concepts and principles that underlie current and emerging methods, tools, and techniques for cost-effective production of high-quality software systems. Definitely this is not a “technology" course, it focuses on the principles of the software production. It emphasizes on some topics that are less familiar to students, such as identifying a development process appropriate to the circumstances, documenting requirements, testing in software development, understanding software evolution, quality assurance, etc. Students acquire skills for teamwork, as well as proper planning and organization of the software development process. We assert that students have to achieve literacy which encompasses the methodological approach in the development, use, maintenance and retire of software applied in organizations. We adhere to ACM 2013 Computer Science Curricula Guidelines [1] and the IEEE/ACM Curriculum

Guidelines for Undergraduate Degree Programs in Software Engineering [2]. We also briefly discuss the relationship of software production courses to other courses in computer science and information technology majors.

The rest of the paper is organized as follows. Section 2 concentrates on the importance of Software Engineering course for students majoring computer science or information technologies. Students have to be acquainted with software development best practices and to be familiar with technological innovations so that participate actively in the future of software production. In Section 3 we present the body of knowledge of the course we deliver. Section 4 deals with some methodological issues - learning activities, examinations and grading. In the conclusion we discuss the disposition of software production courses in the curricula.

The Importance of the Discipline of Software Engineering

Software products facilitate our everyday activities and play a very important role in all aspects of life: banking, communications, manufacturing, transportation, etc. Enormous amount of money is being spent on software development and application domains of different computer programs have become remarkable. Software production is a specific activity that can be analyzed along two features. First of all this is engineering activity and practice that deals with managing pertinent processes and design events. Next, the software as a product has features that complicate the straight application of basic engineering principles. The nature of software is abstract and possesses both static and dynamic properties thus representing a real challenge for description and measurement. At the same time the complexity of the software implies efforts for verification and validation, documentation, and maintenance. The process should provide products of “good” quality but no common measures of quality exist for assessing a software product. In software development the manufacturing cycle focuses more on distribution policies and involves changes through new versions. The difference with conventional engineering processes is obvious.

With regard to this software engineering is a discipline that applies systematic approach to the development, operation and maintenance of software. It comprises the best techniques to build good small, medium, and large-scale

software systems. Software engineering uses engineering methods, processes, techniques, and measurements and applies tools for managing software development and modeling software artifacts.

Another aspect of the discipline is the nature of software engineering knowledge itself. The study of this discipline involves associated knowledge from mathematics, computer science, traditional engineering and management science along with behavioral skills like leadership and team building. Much attention is paid to the personnel in the software process as it concerns their roles and interactions. Planning, estimation, and version control are managerial activities that appear in any software project nevertheless the chosen process model or size.

So, what should students majoring computer science (CS) or information technologies (IT) know? First of all they need a general knowledge on key topics that are fundamental to all development processes. Next, students are to be able to choose the most appropriate tools, methods, and approaches for a given development environment. Modeling of software processes and the organization of software production are important issues to be covered by an introductory course of software engineering. Students are also expected to work as effective individuals or as team members to design and deliver quality software. So, proper assignments e.g. projects or technical reports that exercise effective students' communications are to be prepared. In addition students are to be aware about the importance of current control procedures, to observe deadlines and to respect individual and team performance evaluations.

Body of Knowledge of the Software Engineering Course

This course is required in the CS and IT core curricula, designed to introduce fundamental concepts and principles of software. It helps understanding issues that are unique to the nature of software and its development. The emphasis is on software process and the various process models, e.g. waterfall, incremental, agile ones. The basic activities related to the software development process are analyzed and different choices of techniques for carrying out any of these key activities are outlined. The course comprises topics concerning some practical aspects of software production such as quality of software, project management, legal and ethical issues.

The description of the educational objectives follows traditionally the Bloom's taxonomy: knowledge, comprehension, application, analysis, synthesis and evaluation [3]. A balance between theory and practice is respected according to ACM/IEEE CC 2013 and IEEE/ACM SE 2014 Curriculum Guidelines. Students will be able to:

- compare and contrast the different process models;
- compare and contrast the activities in each of the phases of the process model;
- create models of software data and processes using object oriented modelling approaches using the UML;
- relate the level of user involvement in a project development to the success of the project;
- organize software production;
- demonstrate skills of software documentation, quality assurance and evaluation, and testing as part of software development;
- work on a team or as effective individuals to deliver quality software;
- appreciate the distinction between software technologies and an engineering approach to software production;
- explain the legal and ethical issues pertaining to software production in terms of intellectual property, privacy and security.

To achieve the educational goals of this course, the following core concepts are to be mastered:

1. Software processes – features, models and types.

   The software process is a key concept as it is concerned with delivering recipes for software production. Various types of software processes such as the process of software development, the process of software management and the process of software configuration are analysed and their features are outlined.

2. Modelling and specification of software processes.

   The life-cycle process model characteristics are presented. Plan-driven and agile development is discussed. The application of

software process models is introduced. Different development methodologies are compared.

3. Requirement engineering.

   The requirements engineering is analysed and functional and non-functional requirements are defined. Different approaches to requirements' elicitation and analysis are discussed. The software requirements documents are explained.

4. Software design and construction.

   Software design focuses on implementation issues for a component or a system. Students become aware about the different design strategies such as structured design (top-down functional decomposition), object-oriented analysis and design, etc. System modelling is discussed. Design notations e.g., class and object diagrams, UML state diagrams are presented. Relationships between requirements and designs are outlined. Software architecture concepts and standard architectures: client-server, n-layer, pipes-and-filters, etc. are introduced.

5. Software verification and validation.

   Software verification and validation uses a variety of techniques to guarantee that a software component or system satisfies its requirements and meets users' expectations. Static approaches and dynamic approaches to verification are summarized. Testing types are presented. Students are expected to differentiate types and levels of testing (unit, integration, systems, and acceptance).

6. Software evolution and documentation.

   The characteristics of maintainable software and the importance of proper software documentation are discussed. Program evolution dynamics is analysed.

7. Quality assurance in software production.

   The role of quality assurance activities in the software process is described. Quality and quality attributes for software are defined.

Quality management systems and process improvement models are presented (CMMI, ISO9000).

8. Legal and ethical issues.

   Legal and ethical issues pertaining to software production in terms of intellectual property, privacy and security are briefly summarized.

Learning Activities Organization

The students are required to attend all the classes nevertheless it appears not obligatory. It is expected that they will be present on every meeting of the course. The learning content is delivered through the Learning Management Systems Moodle implemented at New Bulgarian University to facilitate both regular and distance education. The students are expected to have acquaintance with the content prior to classes and to show active participation during the lectures and the practical seminars. Students are also supposed to take part in different panel discussions. The topic is to be chosen from the learning content. Students are expected to develop a PowerPoint slide show which should demonstrate a high quality design and have at least 10 slides.

The use of lectures supplemented by laboratory sessions, video lessons, and so on is the most common approach to teach an introductory course. However students learn best at the application level when they participate in development experiences e.g. projects. Software projects require students to form a team and to develop a software product. This way the learners can pass through all activities of a chosen software process model. Moreover, much of software engineering is based on effective communications among team members and end users. Projects appear to be quite useful for students to exercise their interpersonal communication skills and to experience new technologies within a practical environment. Besides some soft skills like time management, problem solving and team work, important personal traits could be built like orderliness, persistence, collegiality, patience, work ethics etc.

Student will profit by applying technical knowledge in practice. They should be able to prepare on demand different kinds of technical reports. With respect to this students are required to prepare a ten to fifteen page paper on a topic approved by the lecturer. The assignment is to be prepared and posted on Moodle by the end of the semester.

Exams consist of multiple choice and short essay questions as well problems that require applying specific techniques. The grade comprises the evaluation of the current learning performance.

Conclusions

IT programs should produce graduates who apply information technologies in a wide range of settings. An introduction software engineering course for CS and IT specialists should focus on the technical and developmental aspects, but on the organizational and managerial views as well. This implies from the twofold role these specialists play: designers and developers, but managers of their own work. Organizations of all kinds are dependent on networked computing infrastructure and IT people. Because of this, the elements of software engineering are applicable to developing software in all areas of computing.

In this paper we discuss the importance of the discipline Software Engineering and we propose body of knowledge and proper learning activities for a one semester introductory course. Our experience shows that the course discussed above is a good prerequisite for advanced courses concerning software development. It provides a foundation for other software-oriented knowledge areas. This predetermines its position in the upper level of bachelor programs.

Bibliography

[1] ACM/IEEE CS Joint Task Force on Computing Curricula. “Computer Science Curricula 2013,” ACM Press and IEEE CS Press, 2013,

DOI: http://dx.doi.org/10.1145/2534860, accessed on May 15, 2017.

[2] IEEE/ACM Software Engineering 2014 Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering, a Volume of the Computing Curricula Series, 23 February 2015.

https:// www.acm.org/ education/se2014.pdf, accessed on May 15, 2017.

[3] Bloom, B. S., Englehart, M. D., Furst, E. J., Hill, W. H., & Krathwohl, D. R. (Eds.) A Taxonomy of Educational Objectives: Handbook I. Cognitive Domain. New York: David McKay, 1956.

Authors' Information

Juliana Peneva, PhD, Assoc. Prof, New Bulgarian University, 21 Montevideo St. 1618 Sofia, Bulgaria, jpeneva@nbu.bg.

Major Fields of Scientific Research: database systems, software engineering, information systems, e-learning

Stanislav Ivanov, PhD, Assoc. Prof, New Bulgarian University, 21 Montevideo St. 1618 Sofia, Bulgaria, sivanov@nbu.bg

Major Fields of Scientific Research: object-oriented modeling, programming