Building open learning environment for software engineering students

Alexey Khoroshilov[1], Victor Kuliamin[1], Alexander Petrenko[1], Olga Petrenko[2], and Vladimir Rubanov[1]

1 Institute for System Programming of the Russian Academy of Sciences (ISPRAS), Russia {khoroshilov,kuliamin,petrenko,vrub}@ispras.ru,

2 Moscow Institute for Open Education, Russia
o-l-petrenko@yandex.ru

Abstract. The paper discusses principles of open education and possibilities of implementing these principles for software engineering education on the base of open source software development projects. A framework of practical courses for software engineering students built on these ideas is presented. Experience of building courses on the base of this framework is discussed on the example of “Software Engineering” course provided to students of the System Programming departments of the two Russian top-ranked universities, Moscow State University and Moscow Institute of Physics and Technology.

Keywords. Participatory learning, lifelong learning, open source, software engineering education

1 Introduction

Educational system should match the needs and tendencies of continuously developing society to sustain its evolution. That is the reason of evergrowing demand for innovation education programs, methods, and supporting infrastructure across all over the world. A conceptual base for innovation education in Russia was formed by academician M. A. Lavrentev [1] who states a principle of “sciences — personnel — industry” (triangle of Lavrentev [2,3]). Most of currently applied approaches of innovation education implement this idea in modern conditions – education happens during generation of new knowledge as a result of integration of fundamental science, educational process, and industry.

Traditional system of professional education is mainly based on transition to students of fundamental knowledge helping them to feel themselves with confidence in some area and skills that can be applied in some practical work at once. However, under the conditions of intensive technological evolution, such an approach to education becomes irrelevant because students slowly become able to transform fundamental knowledge to practically applicable one as well as concrete practical knowledge quickly becomes outdated and unclaimed. The main abilities demanded in these circumstances become adaptability, constant knowledge update, decision making unbiased from established patterns, dynamic activity planning, etc.

All these issues are applicable to the area of software engineering, which evolves very quickly. Schematically, the main fields of skills and knowledge of information technology professionals can be presented as in Fig. 1 [4]. So, just subject knowledge is not enough for successful work of a good specialist in the modern society. The Memorandum of European Commission on lifelong learning [5] emphasizes the need in such social skills as acting with confidence, result-oriented focus of personal activities, right balancing of risks and responsibility in decision making, as well as such cognitive skills as ability to learn continuously, adaptability to changing environment, skills in finding right information in various areas, and ability to filter necessary information in the huge informational flow that each active individual in the modern society is subject to.

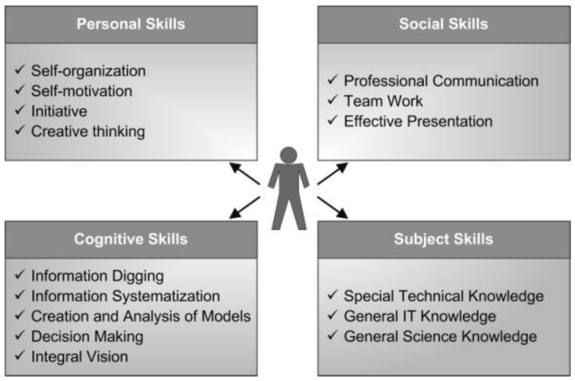

Figure 1. Main skills of modern IT-professionals.

In general some specific arrangements are required to make high school graduates more compliant with labor market requirements. In some high schools teaching Information Technologies undergraduates have few possibilities to work with real-life examples in their domain, so starting their career they immediately face with the issues not covered in the traditional university courses. Such courses are usually focused on scientific and technical aspects of the domain and contain (if any) only rather shallow presentation of organizational and social issues. Potential of many undergraduates is inhibited by lack of knowledge and skills at these areas. One more impediment to their growth is lack of comprehension of relations between theoretical matter they get in university and their practical work, so many of them think sincerely that most of that theory has no real use. This usually demonstrates lack of (and bad training in) an important skill — systematic analysis of routine technical issues, which force to use scientifically approved methods presented in theoretical courses.

The first step to change this situation may comprise in introduction of courses targeted to development of social skills necessary for professional work in the related technical domain. An objective of such a course is nurturing these social skills that help to realize technical skills and knowledge obtained and to resolve non-technical problems met in real professional life.

The necessary skills to develop in these courses are:

- writing technical and scientific texts;
- preparation of presentations and performing them in public;

- organization of technical or scientific presentation, in particular, fitting in time and attracting the audience;
- various techniques of information classification and systematic analysis;
- professional argumentation based on scientific methods and knowledge;
- adequate answering on questions;
- posing adequate questions.

A course block intended to develop text creation skills may consider the following topics.

1. Professional communication as a significant component of profession. Specifics of professional communication.
2. Determine the personal communication style (with the help of a test). Using personal communication style in developing professional relations.
3. Verbal and written communications.
4. Characteristics of written communications. How to prepare a written report.
5. Styles of writing. Characteristics of different styles. Practice in preparation of texts written in different styles.
6. Presentations and talks. Time management during talks. Presentation content change techniques.
7. Specifics and requirements for scientific communication and texts. Structure of scientific text.
8. Work with information sources during scientific text preparation.
9. Argumentation and reasoning. How to use arguments in texts and what arguments to use.
10. Practice in scientific text preparation.
11. Scientific presentation. Characteristics and structure of scientific talk. How to answer questions during a talk.
12. How to choose a talk style. Feedback types. Talk evaluation.

However, introduction of courses focused on social skills necessary in technical domains is not sufficient. The entire existing education environment should be modernized in accordance with needs of social development. It is necessary to create such an environment that helps to prepare graduates adapted to the needs of modern industry and markets.

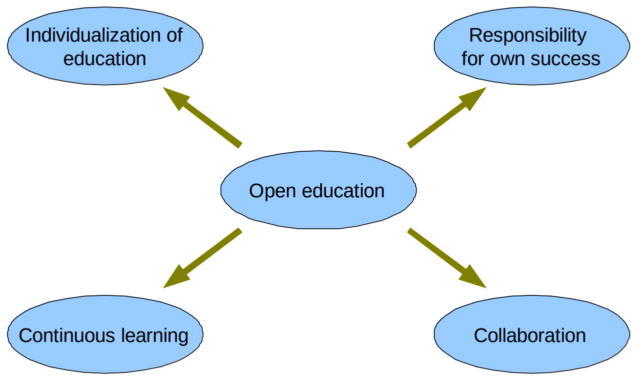
Individualization of education
Responsibility for own success
Open education
Continuous learning
Collaboration

Figure 2. Open education principles.

Acknowledging these principles of continuous lifelong learning stimulates transition of educational systems towards so called *open educational system*. This system is mainly oriented on upbringing independent self-motivated individuals that are able to effectively collaborate with the quickly changing world - individuals that can and want to effectively learn constantly rather than just apply known and established practical skills in some steady work.

In such open educational systems, educational opportunities are open to students - they can see paths of possible education and evolution and thus it becomes possible to discuss with its mentor the means and specific actions for achieving individual goals in the broad space of these opportunities. One of the main characteristics of open education is responsible decision making by the student about the needed educational goals and means to meet them. Students have to realize and take full responsibility in this.

The main principles of open education are [6]:

- individualization of education;
- responsibility for own success;
- collaboration;
- continuous learning.

Transition to the open education may be based on the following principles:

- It is necessary to teach students to be self-dependent in education. Being active is a key thing for this.
- Students must be involved in the mutual personal communication.
- The starting and the target points of each particular part of the educational path should be individual.
- Explicit stage of reflection should be introduced in the education process.

The important general step towards open education is creation of special open learning environment that stimulates students to be actively involved in leading professional societies in the studied field and to actively communicate and collaborate within them. Such societies provide various opportunities in implementing educational goals and students can freely and independently choose whatever fits them best individually.

In this paper we discuss possibilities of building such open learning environment for software engineering students on the base of open source software development projects. We present a framework of practical courses for software engineering students built on these ideas. Experience of using of the framework is discussed by the example of “Software Engineering” course.

2 Software Engineering Education and Open Source Software Development Projects

M. A. Lavrentev once wrote: “There are many ways by which scientific ideas come to industry. The forms of cooperation between science and industry themselves also require scientific approach, creativity, and choice of optimal solutions in each particular case” [7]. Open source software has recently become such a way in the area of software engineering. Benefits of using open source software in high education institutes and especially in the area of software engineering are discussed a lot [8-13]. Traditionally, a possibility to investigate internal software structure is mentioned at the first place as a consequence of source code availability. But it is also important that open source software development projects are also open, i.e. their development process is public and all discussions regarding design and coding decisions, software architecture evolution, and project documentation development are available for investigation too. Thus, open source software development projects provide good material and rich infrastructure for education of software engineering students.

Moreover open source software development projects can be a good basis for building participatory learning environment in the area of software engineering. Such projects can effectively and easily enable the following tools and educational means for almost any specific technical subject.

- *Rich educational materials*: feature requests from users and ideas of developers, requirement documents, the executable software itself, specifications of its architecture, implementation source code, lists of already fixed and still standing issues and features elaborated at the development and maturity phases of the product lifecycle, detailed argumentation in favor of or against some particular solutions and approaches in the context of real system.
- *Ready infrastructure for practical classes*: configured information systems for version control and project collaboration with possibility to communicate with real developers of the product.
- Possibility to create *strong motivation* for students by involving them in the practical activities in real life development projects, especially if such projects have high social importance and prestige.
- Help to students in creation of personal portfolio and *finding a job* in future by possibility to demonstrate results of their work evaluated by expert community to potential employers.

Documents and source code of one or several projects can be used as a ready-to-use education material. But more important is the diversity of open-source software projects that provides students a wide variety of choice options for thinking and finding individual learning paths within the general software engineering educational plan. Also, open source projects provide wide variety of choices of technical aspects such as programming languages, development technologies and methodologies, even within a specific topic, because there are usually a number of different competing projects that use different technical approaches for implementing the same functionality or a product of the same kind. This variety and freedom of choice demonstrate real life complexity to students and improve their motivation and responsibility.

At the same time, using open source software development projects for education is not a trivial task. Members of open source software projects are not always happy to be a training ground for a crowd of students. They would like to see future colleagues (even beginners) but not short term students. This fact has to be given proper weight in organizing learning environments on the base of open source software projects. For example, communications between students and open source developers should not be a mandatory element of an educational process. It may be an optional element available for most motivated students. In this case some introduction into professional communications and open source development culture should be provided for students.

Another conclusion is that solely open source projects are not enough to build an effective learning environment. Work with open source projects should be supplemented by a substantial set of activities provided by an educational organization. These activities should help students to dive into open source development environment by:

- explaining general models of open software development and hidden details of open source development culture;
- answering any kind of questions related to the project chosen;
- helping to start communication with open source community;
- advising useful informational sources and transferring patterns and best practices of work with them.

It is also important to stimulate discussions between students on their experience in investigation of open source projects and to provide general theoretical materials constituting a background of practical world of open source projects.

3 Practical Software Engineering Courses Framework

ISP RAS developed a framework of practical courses for software engineering students based on open source software development projects and principles of open education. The key elements of the framework are as follows:

- participation in individual open source project;
- personal mentors;
- theoretical lectures;
- colloquiums devoted to joint discussion of students results.

Instead of classic practical classes based on pre-selected model examples and tasks we introduced a requirement for students to participate in at least one public open source project that they can choose by their own decision. A possibility to choose the project on their own is very important to improve students' motivation as it was discussed above. Some courses may impose limitations on the projects so that the project can be used to cover required material in the domain under learning. For example, learning basic software engineering principles or methods of analysis and development of huge and complex systems can be supported well by projects of rather big size. Learning architecture styles and design patterns requires projects explicitly and adequately using such styles and patterns, while a majority of open source software cannot be considered as satisfying to this requirement. Learning algorithms may require spending some time in search for open projects and libraries that actually use some sophisticated algorithms.

We assign to each student a personal mentor who keeps in close touch with the student, answers his/her personal questions, monitors student activities, evaluate their effectiveness for learning the chosen domain, and advises useful informational sources and rules of work with them. One mentor can serve no more than 4 students.

In parallel to a practical course a corresponding theoretical course should be provided. It is recommended to keep small time interval between providing theoretical material and using the material in practice. But it is also possible (and usually helpful) to have anticipatory practical tasks, theoretical material for which is provided between a start of the task and the colloquium, where results of the task are presented. Theoretical lectures can be conducted in classic style, but also may incorporate elements of active learning and participatory education.

Colloquiums devoted to joint discussion of students results are the main form of students work control. Students make presentations of their results to the classmates and mentors and are asked any questions by the audience. Mentors ensure that the students can make correspondence between the theoretical information taken from the course with specific practical aspects of their projects by proper questions. Such organization of learning helps to overcome a gap between pure theory and practice and in that way to increase efficiency of the lectures. In addition, the joint colloquiums help to enable inter student experience sharing.

The main stages of a practical course include the following.

- *Choice of an open-source development project*. The project must have a public infrastructure and be active, that is it should have on-going developments or active maintenance. Students may also join any open-source project, which is performed in ISP RAS itself. Students evaluate various projects by themselves, mentors just check if the final choice meets all the necessary requirements from the domain learning viewpoint.

- *Acquaintance with the project*. Students study their selected projects on their own with the focus on those aspects important for their personal educational path and for the specific courses at the faculty. At the end of this phase, students make *presentations* of their projects to the classmates and mentors where they can be asked any questions by the audience to demonstrate mastering of the project information and understanding of design solutions made in it. Additionally, students are asked to prepare written reports with a predefined structure.
- *Practical tasks*. Mentors prepare a number of tasks for students with some weight points assigned to each task. Such tasks may include analysis, refinement, including formal modeling and specification, and documenting of requirements, extracting design patterns used, modeling and analysis of anticipated software characteristics after possible change of design decisions, preparation of design documentation or user/administrator documentation, implementing new modules, code refactoring, resolving bugs in existing modules, design and executing of tests with reporting of issues found and providing possible solutions for them, providing test documentation, porting the project or its part on some unsupported platform, implementing automated installation, and so on.

  Students may choose a number of tasks to perform to reach a defined target in terms of these points. Students provide regular presentations of their work similarly to the presentations at the acquaintance phase. Mentors ensure that the students can relate adequately the theoretical information from the course with specific practical aspects of their projects, thus increasing the efficiency of lectures.

In preparation of the practical tasks it is important to consider the additional opportunities provided by open source software development projects that can be useful for educational process, such as communication with project team and domain experts on professional topics, providing clear argumentation for requirements, design and test decisions.

Mentors encourage students to directly collaborate with the real project team and use feedback of the team as indicators of successful direction of their work. If it is possible, feedback of open source project community should be used to evaluate results of students' work. This improves students’ motivation as the feedback from external parties is perceived by students as more objective than the one provided by teachers and group-mates. But feedback of community should not be a primary tool of evaluation, in particular, because of existing discrepancies, first, between specific project objectives and goals of learning process and, second, between open source community discussion and communication culture and students’ expectations and habits.

Freedom of task choice also improves significantly students’ motivation and attitude to his/her education [10,12]. It enables individual choice of depth of education and allows forming knowledge and skills at his/her own depending on his/her preferences and needs.

4 A Case Study

An example of a practical course built on base of the framework presented is the course “Software Engineering”. The course is provided to students of the System

Programming departments of the two leading Russian universities — Moscow State University and Moscow Institute of Physics and Technology. In parallel to the first two stages of the practice the theoretical course “Software Engineering — Component Approach” [14] is conducted. In addition, before start of the practice an extra lecture “Open Source Software” is provided to introduce students with basic principles of open source software development and open projects organization.

Initially the only limit on choice of open source project was the requirement to choose a project with an open development process. After analysis of the first experience we decided to limit choice by mature and big enough (25000-30000 lines of source code) projects only. General software engineering education requires that the project chosen has issues specific for complex software development, which usually correlate with project code size.

Students have two months to choose a project and to acquaint themselves with it — to read and analyze project documents, to comprehend the main decisions made, and to understand code-related issues. After that we organize the first colloquium, where students present their projects and answer questions from some predefined list. The key idea of the list is to encourage students to find in the project instances of ideas, techniques, patterns, and approaches discussed in the theoretical course. The questionnaire used currently consists of the following 5 main areas:

- project as a whole;
- requirements;
- architecture and design;
- quality assurance;
- personal participation.

In addition, during acquaintance with the project students are asked to prepare one of project documents (project concept description, requirements sketch, architecture and design outline, test plan, etc.) according to the given template.

The next stage is actual practical work. For this course we have prepared about 30 practical tasks covering various areas of software engineering:

- modeling of business domain;
- requirements management;
- software architecture;
- software quality assurance and control;
- debugging and bug fixing;
- feature planning;
- user documentation;
- user support;
- complex tasks.

The practical work takes about 4 months. Results of the tasks are evaluated by mentors and discussed on joint colloquiums. If the results are good enough, we encourage students to publish them for review by open source project team. And as a result the students get constructive critique as well as positive feedback that are used to finalize the tasks.

To date, we have provided the practical course for two years as an optional element complementing the theoretical course “Software Engineering – Component Approach”. The total number of students completed the practical course is about 20. At the first presentations, students are usually in almost complete confusion because at the first years at their universities they get used to the traditional system of fixed information to learn with subsequent exams so that they are always directed what to do. Only after a while, after working in the more flexible and open environment, they start realizing the real diversity of the problems in projects and the importance of proactive position of participants to solve them. Eventually this helps to understand and learn all the principles of software engineering at the significantly higher level.

Our experience confirms that possibility to choose domain of open source project really improve students' motivation in performing consequent tasks. Also an interesting conclusion is that open source projects can be used for demonstration both best practices of software development and common mistakes and decisions that should be avoided.

The experience of this case study demonstrates that many learning objectives can actually be reached in learning process based on open software development projects. Information digging and systematization, integral vision, effective presentation skills are improved. On the other side noticeable improvements in social skills, decision making, dynamic activity planning require introduction of additional patterns into organization of learning process. This is issue for further development of the approach presented. Perhaps, learning all the necessary skills in single practical course is an idealistic goal, and more realistic method is to build a sequence of 3-4 courses developing those skills gradually.

5 Conclusions

Open source software development projects provide a good basis for building open learning environments. They give both a rich set of education materials and examples of implementation of abstract principles and methods, and possibility to take part in a real-life activity, communicating with recognized experts in the domain of learning. However, it is not a simple task to build an adequate and productive collaboration between students and open source community.

We presented a framework for building practical courses for software engineering students on the base of open source projects. And our experience demonstrates positive effects on all the main fields of skills and knowledge of information technology professionals mentioned in Fig. 1. In particular, feedback of students shows improvement of motivation and independence in decision making.

References

[1] http://computer-museum.ru/english/galglory_en/Lavrentev.htm

[2] S. A. Khristianovich, M. A. Lavrentev, S. A. Lebedev. Actual tasks of scientific work organization. Pravda, 14.02.1956.

[3] N. L. Dobretsov. “Triangle of Lavrentev”: the principles of science organization in Siberia. Bulletin of the Russian Academy of Sciences. vol. 71, # 5, 2001, pp. 428–436.

[4] A. K. Petrenko, O. L. Petrenko, V. V. Kuliamin. Research Organizations in IT Education. ISP RAS Proceedings, 15:41-50, 2008.

[5] Commission of the European Communities. A Memorandum on Lifelong Learning. Brussels, 2000.
URL: http://ec.europa.eu/education/policies/lll/life/.

[6] C. Wedemeyer. Characteristics of open learning systems. In Open Learning Systems, Washington, National Association of Educational Broadcasters, 1974.

[7] M. A. Lavrentev. Highways of Siberian science. Izvestia, 13.02.1971.

[8] K. J. O’Hara, J. S. Kay. Open source software and computer science education. J. Comput. Small Coll., 18(3):1–7, 2003.

[9] Allen, E.; Cartwright, R.; Reis, C. Production programming in the classroom. Proc. of the 34-th SIGCSE technical symposium on Computer science education, Reno, Nevada, USA, pp. 89-93, 2003.

[10] D. Carrington, S.-K. Kim. Teaching software design with open source software. Proc. of 33-rd ASEE/IEEE Frontiers in Education Conf., pp. 9-14, November 2003.

[11] C. P. Fuhrman. Appreciation of software design concerns via open-source tools and projects. Proc. of 10-th Workshop on Pedagogies and Tools for the Teaching and Learning of Object Oriented Concepts, at ECOOP 2006, Nantes, France, July 2006.

[12] M. Pedroni, T. Bay, M. Oriol, A. Pedroni. Open source projects in programming courses. ACM SIGCSE Bulletin, 39(1):454-458, March 2007.

[13] http://www.flosscom.net.

[14] V. V. Kuliamin. Software Engineering. Component-based Approach. Moscow, INTUIT-Binom, 2007.