Electronic Communications of the EASST
Volume 20 (2009)

Proceedings of the Third International Workshop on Foundations and Techniques for Open Source Software Certification (OpenCert 2009)

Open Source Certification and Educational Process

Alexey Khoroshilov

8 pages

Guest Editors: Luis Barbosa, Antonio Cerone, Siraj Shaikh
Managing Editors: Tiziana Margaria, Julia Padberg, Gabriele Taentzer
ECEASST Home Page: http://www.easst.org/eceasst/ ISSN 1863-2122

Open Source Certification and Educational Process

Alexey Khoroshilov[1]

[1] khoroshilov@ispras.ru, http://www.linuxtesting.org/
Institute for System Programming of the Russian Academy of Sciences, Moscow, Russia

Abstract: This paper discusses possibilities to benefit software engineering and computer sciences educational process from involving students in open source software certification activities. On the other hand the open source certification community can take advantages of this involvement if it cooperates with educational one. The situation is considered in the context of Russian higher institutes of education but the conclusions can be applicable to other institutes as well.

Keywords: Open Source Software, Software Engineering Education, Verification

1 Introduction

Open source software certification is an essential factor for success of open source within government and business organizations. At the moment, such certification is mostly carried out by commercial companies that offer open source based solutions to end users.

There is repeatedly discussed need for community based certification process founded on rigorous, mathematically based, methods in software analysis and engineering by community and at the OpenCert workshop [Ope07, Ope08]. There were proposed approaches [CS08, KM08, PB08b] that defined some feasible ways to enable the process. Technically the initial steps are rather clear. But actually nothing is happened mostly owning to organizational question. There were considered the following possibilities for the process organization:

- to start pure community project;
- to involve teachers and students in the project;
- to obtain funding from government or non-profit foundations.

The community is not sufficiently interested in the process for various reasons [PB08a]. Government and non-profit foundations have not yet shown interest in the activities.

Although it should be noted the massive program of the Linux Foundation [LFT] to develop a serious test suite for conformance testing against the Linux Standard Base (LSB) [LSB] standard, in which there has been developed more than 160000 tests for more then 26600 interfaces. Under this project three different technologies are used to develop tests of different quality grades. And for the core functionality of the LSB model based testing technology is used, which provides the most thoroughly tests developed with involvement of mathematical methods.

Within the paper there is considered the possibility of attracting students to the organization of open source software certification in order to bring benefits to all parties: the students will get necessary skills and experience which can be useful in the future, while the open source

certification community will be able to start moving forward and to get a basis for more active development.

In the next section, we will describe the features of the educational process in higher educational institutions of Russia, to understand what opportunities exist for integration of open source certification into the educational process. Then we will discuss what challenges the open source certification face. And finally, we will consider how to mutually benefit each other and what concrete steps can be done.

2 Educational Process in Russia

In higher educational institutions of Russia the most common types of educational activities are as follows:

- mandatory and special courses;
- practical training;
- special seminars;
- course and diploma work.

Mandatory and special courses consist of lectures, seminars and workshops. Lectures are the basis of theoretical training for students. Their purpose is to provide a foundation of scientific knowledge on a subject, reveal problems, state of and prospects for progress in specific areas of science and technology. A course of lectures is a systematic problem statement, which consistently presents material on a subject.

Lectures are usually accompanied by various kinds of seminars, practical workshops and supplemented by homework. These exercises can be aimed at a thorough study of the most important topics of the course or to obtain practical skills on the subject.

Special courses differ from mandatory ones as they are not mandatory to visit by students. Students have opportunity to choose between different special courses, so it is important to make special courses interesting and attractive.

Courses usually last one or two semesters and are concluded by assessment test on a five-point scale. Assessment of practical work may be a part of assessment of the course or as an independent test. Lectures are usually provided by one or a couple of lecturers. Seminars and workshops are conducted for groups of 15-25 students by a trainer.

The purpose of practical training is to consolidate and extend the knowledge gained by students in the learning process, to acquire knowledge, skills and practice necessary for profession. Duration of practice may vary. For example, the most practical trainings at Computer Sciences Department of Lomonosovs Moscow State University run for 12 weeks and 2 days a week, including two days at the end of practice devoted to preparation of a written report.

Each student gets a mentor from university and a mentor from the organization where the training occurs. Responsibilities between mentors are defined in each case particularly. Number of students per a mentor may vary from 3 to 40.

Special seminars are designed for detail study of the topic of students scientific expertise. Teaching methods used at special seminars are determined by its scientific adviser and may

include both theoretical and practical components. Special seminars usually take about two hours a week. As a rule number of students per scientific adviser is limited by 3-7.

Term and diploma work is a self-teaching and research work of a student under the guidance of a scientific adviser. The purpose of it is to study methods of research work and to get new knowledge and skills.

Term work in software engineering combines research and practical components. The research component includes analysis of a problem and search of an appropriate solution. The practical component is implementation of the solution proposed.

Diploma is a qualification work in which students have to demonstrate professional knowledge of the theory and practice of the subject area, ability to solve specific tasks in this area. Usually a scientific adviser for term and diploma work is the same as for a special seminar.

3 Open Source Certification: Goals and Tasks

The purpose of the open source certification community is to establish the quality assessment of open source software projects especially involving rigorous, mathematically based, methods in software analysis and engineering. We see the following directions that can be used for achieving this goal:

- code review;
- testing;
- light weight verification;
- heavy weight verification.

Code review is frequently used in open source projects. But in our context, this direction does not look too promising. First of all, code review is not very attractive to students. And secondly, benefits, such as skills of someone else's code analysis, identification of common issues and successful design patterns, can be obtained within more interesting activities. In addition to code review itself there are other important tasks. For example, it would be useful to develop code review guidelines helping to detect different kinds of errors.

Testing is a huge area that provides great number of tasks. It is necessary to have tests of different kinds:

- functional testing;
- conformance testing;
- regression testing;
- stress testing;
- load testing;
- etc.

and tests for components of different levels:

- unit testing;
- integration testing;
- system testing.

Testing activities include:

- new test development;
- test execution and result analysis.

Testing activities allow to obtain various skills: test development itself, documentation analysis, source code analysis, reverse engineering, etc.

Lightweight verification includes various methods of static analysis and model checking. In this regard there are the following tasks:

- identification of domain specific restrictions and typical bugs for automatic detection;
- formal representation of the restrictions in terms of the tools used;
- development of simplified models of target system to be used for automatic analysis;
- automatic analysis of target source code with verification tools and investigation and classification of results.

In addition, if verification tools are implemented using extensible architecture, it is possible to develop new verification techniques and to improve existing ones on base of experience of its application to open source software analysis.

If lightweight verification allows to find only certain kinds of problems in the source code, heavyweight verification provides a more complete analysis of the properties of the target program. There are different approaches to heavyweight verification. For example, let us consider classical Floyd-Hoare methods of verification of sequential programs.

First of all the method requires to formally describe requirements to the target function in the form of precondition and postcondition. Moreover, all auxiliary functions used by the target ones should be specified in terms of precondition and postcondition as well. Then, invariants and variants should be defined for all cycles of the target program. Invariants describe restrictions on values of the program variables and memory when the control flow is in the given point of the program. Variants define a function, which decreases throughout every pass of a cycle. After that verification tools automatically generate conditions in high order logic, a proof of which leads to a proof of full correctness of the target function against the given specification. Proof of the conditions is usually conducted within interactive theorem provers such as PVS [PVS], Coq [COQ], Isabella HOL [HOL].

Accordingly, in this approach to verification the following tasks occur:

- specification of auxiliary functions (library functions, operating system functions, etc.);
- specification of target functions and invariant and variant development;
- proof of lemmas generated in interactive provers.

4 How to Mutually Benefit Each Other

Let us discuss how to mutually benefit open source software certification community and educational process.

Seminars and practical workshops of courses can use open source certification tasks if a subject of a course has explicit intersection with testing or verification methods and tools, which are used in the open source certification community. However, workshops require small learning tasks and work with source code of real projects can be too difficult. That is why we believe the work in this direction seems not very promising. Nevertheless testing tasks can be successfully used here under some conditions.

Practical training is rather long and allows to solve real and significant enough tasks. But its success requires a set of clearly defined goals, accompanied by a methodical material, as well as a mentor for each student, who will answer student questions regarding the methods and tools involved and the target software as well. If a representative of the educational institution can be a such mentor it is a very good situation. Otherwise some responsibilities should be taken by a representative of the open source certification community. But in any case some participation of the community is required to consult students and mentors on complex issues. Of course, the ideal situation is when a mentor of the university is an active member of the open source certification community as well.

Tasks for practical training, for example, can be of the following kinds:

- to develop tests of certain types with certain quality using a certain technology for a particular component;
- to apply a particular method of static analysis or model checking to a particular component and to analyse results;
- to specify requirements for a particular function formally and to prove its correctness with respect to the specification.

More concrete examples of the tasks are as follows:

1. Analyze requirements of the given standard (documentation) to a particular group of functions and create a catalogue of elementary requirements to it. Develop formal specification of the requirements to the functions in the form of preconditions and postconditions in the specification extension of the C programming language with traceability to the initial text of the standard. Develop functional tests for these functions using model-based testing technology UniTESK [Uni]. Estimate quality of the tests using the following test coverage metrics: coverage of elementary requirements, coverage of functionality branches as well as source code coverage.
2. Analyze requirements of the given standard (documentation) to a particular group of functions and create a catalogue of elementary requirements to it. Develop tests using the T2C testing technology [T2C] with traceability to the initial text of the standard. Estimate quality of the tests using the following test coverage metrics: coverage of elementary requirements as well as source code coverage.

3. Develop formal specification of requirements to a particular function in the form of preconditions and postconditions in ANSI / ISO C Specification Language [ACS]. Develop cycle invariant and variants in the same language. Generate Floyd-Hoare verification, correctness and termination conditions using Why software verification toolkit [WHY]. Prove the conditions within one of the interactive theorem provers supported by Why (PVS, Coq).

Work at special seminars may have an intersection with the open source certification tasks if a student's academic specialization has relevance to these tasks directly. It is possible if a scientific advisor is interested in software testing and verification techniques. But keeping in mind that students' practical work at special seminars usually is either directly related to their term and diploma works or similar to practical workshops, discussion of this topic can be considered as a part of the discussion below.

Course and diploma work may also include a solution of significant problems as in the case of practical training, but in contrast to it course and diploma work should have a research component. Therefore, either a target system for testing or verification should be interesting from research point of view or a task should be related to investigation and improvement of basic technology of testing or verification. For example, it can be development of automatic strategies for PVS prover, which simplify interactive proof of the conditions generated as a part of Floyd-Hoare verification methods.

Thus, there are two directions for implementing open source certification tasks within educational process: methodical and organizational ones. As a part of the methodical direction it is required to:

- prepare an appropriate number of tasks to be assigned to students;
- prepare methodical materials necessary to perform the work;
- organize support for students and mentors during the educational process.

Within the organizational direction it is required to find representatives of universities who are interested in organization of student work on the proposed directions. They can be interested in it by the following reasons:

- students will be able to get skills of application of advanced technologies to real life software;
- results of the tasks are not thrown out, but benefit society (and it makes positive impact on motivation of students);
- methodical materials and tasks are available.

It may be useful to define a web resource where actual tasks and methodical materials are collected and discussed. Case studies of using the materials within educational process and feedback of students and mentors can be collected at the same place.

5 Conclusions

We have described the possibilities to integrate open source certification tasks into practical work of software engineering and computer sciences students. We have demonstrated that it can be mutually benefit both educational process and the open source certification community. The initial steps, that can be done to prepare a basis for this integration, have been identified.

In the Institute for System Programming of the Russian Academy of Sciences we work a lot with students of Computer Sciences Department of Lomonosovs Moscow State University and Moscow Institute of Physics and Technology and we are interested in discussion of this proposal with the open source certification community to identify the best ways to apply it within our educational process. If a consensus within the open source certification community is achieved, this approach will be able to be applied to more wide range of educational organizations across the world.

Bibliography

[ACS] ACSL:ANSI/ISOCSpecificationLanguage.
http://framac.cea.fr/download/acsl_1.4.pdf

[COQ] The Coq Proof Assistant.
http://coq.inria.fr/

[CS08] A. Cerone, S. Shaikh. Incorporating Formal Methods in the Open Source Software Development Process. In *Proc. of the Second International Workshop on Foundations and Techniques for Open Source Software Certification (OpenCert 2008)*. Milan, Italy, Sept. 2008.

[HOL] The Isabelle/HOL Proof Assistant.
http://isabelle.in.tum.de/

[KM08] A. Khoroshilov, V. Mutilin. Formal Methods for Open Source Components Certification. In *Proc. of the Second International Workshop on Foundations and Techniques for Open Source Software Certification (OpenCert 2008)*. Milan, Italy, Sept. 2008.

[LFT] LSB Testing Program.
http://linuxfoundation.org/en/Testing

[LSB] Linux Standard Base.
http://linuxfoundation.org/en/LSB

[Ope07] *Proc. of the Second International Workshop on Foundations and Techniques for Open Source Software Certification (OpenCert 2008)*. Braga, Portugal, Mar. 2007.
http://opencert.iist.unu.edu/workshop-home-2007.html

[Ope08] *Proc. of the Second International Workshop on Foundations and Techniques for Open Source Software Certification (OpenCert 2008)*. Milan, Italy, Sept. 2008.
http://opencert.iist.unu.edu/workshop-home-2008.html

[PB08a] S. Pickin, P. T. Breuer. Four Compass Points in the Verification of the Linux Kernel. In *Proc. of the International Workshop on Foundations and Techniques bringing together Free/Libre/Open Source Software and Formal Methods (FLOSS-FM 2008)*. Milan, Italy, Sept. 2008.

[PB08b] S. Pickin, P. T. Breuer. Open Source Certification. In *Proc. of the Second International Workshop on Foundations and Techniques for Open Source Software Certification (OpenCert 2008)*. Milan, Italy, Sept. 2008.

[PVS] PVS Specification and Verification System.
http://pvs.csl.sri.com/

[T2C] T2C Test Development Technology.
http://ispras.linuxfoundation.org/index.php/T2C_Framework

[Uni] UniTESK Model Based Testing Technology.
http://www.unitesk.com/

[WHY] Why Software Verification Toolkit.
http://why.lri.fr/