Re-engineering Software Education: OSS-aware Software Education in the Era of Utilizing External Resources

Toshihiko Yamakami
Research and Business Development, ACCESS
Nakase 1–10–2, Mihama-ku, Chiba-shi,
Chiba, Japan 261-0023
Email: Toshihiko.Yamakami@access-company.com

***Abstract*—Open source software (OSS) is becoming increasingly visible throughout the software industry. Not only for the software industry, it is also penetrating numerous aspects of today's world, such as open innovation and social innovation. The power of OSS, as a radical driver of social innovation, presents a wide array of challenges, not just for the real world, but also for education. The author presents a seven-aspect view model of OSS that describes the diversity and heterogeneity of OSS. Then, the author discusses the impacts and implications of these aspects from the viewpoint of education.**

***Index Terms*—OSS-aware education; diversity of OSS; heterogeneity of OSS;**

I. Introduction

Open source software (OSS) has become mainstream in the software industry since it was coined in the late 1990's. This has had a significant impact on the software industry in facilitating a business model changes. It is also having brought an emerging impact on computer software education.

In the early days, OSS was considered to be simply a free candidate for educational software. It was perceived as the alternative to expensive proprietary software. This view was emphasized by academic institutions that had small budgets for educational software.

As OSS extends its coverage, it is continuing to cover an ever-wider range of software engineering, from a philosophy of software, to best practices in software development. This extended coverage has complicated the requirements of OSS-aware software engineering education.

At the same time, OSS is continuing to push forward with radical changes in the commoditization of software. This open and commoditized software trend also brings a new requirement for OSS-aware education.

With the success of OSS, the code provided by OSS is continuing to increase in size. This is bringing with it a shift from writing skills to reading skills.

The author discusses the impact of OSS on computer software education in order to understand the changes that are being brought about with the wide acceptance of OSS.

II. Purpose and Related Works

A. Purpose of Research

The aim of this paper is to identify the impacts and implications of the advances of OSS from the viewpoint of education.

B. Related Works

The penetration of OSS has attracted the attention of researchers mainly in the context of the cost-reduction efforts and customizability of source code [1]. Rooij discussed the survey results of cost-cutting efforts in the higher education from a survey of 772 academic and information officers [2]. One particular use of OSS is courseware authoring tools. Christian presented an open source curriculum development tool [3]. In a related domain, Yue discussed open source courseware use by computer science educators [4]. Yue argued that the impart of open source courseware is likely greater than that of use of OSS. Belloni discussed interactive curricular material created in an OSS project for physics [5].

With new trends in computing, educators have realized the need for a new education. Hawthorne discussed the role of software education in coping with out-sourcing, distributed development, and OSS [6] [7]. Patterson argued that most schools taught "write programs from blank sheet of paper" programming, of which there was very little bearing on the real-world [8]. He suggested a few OSS-oriented computer science education courses for remedying that situation.

Expanding the focus beyond just reducing costs or attracting students, Morelli discussed the importance of motivating community-minded undergraduate students using the humanitarian focus of socially useful projects [9]. Although not specifically in an educational context, Chopra discussed the ethical superiority of a software economy based on OSS [10]. Berry discussed the uses that OSS might have in government for furthering democracy and public participation [11]. These arguments represent philosophy and social movement, which are other aspects of OSS, in literature.

As the accepted use of OSS, Meneely discussed an education-friendly repository to help eliminate some of the burdens of students and educators [12].

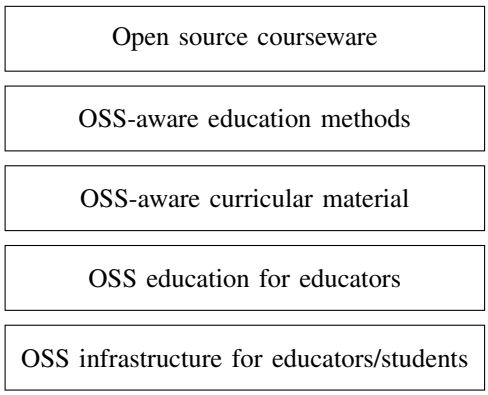

Fig. 1. OSS-aware software education model

Hislop discussed the educational effects and results of student participation in OSS [13].

III. Method

OSS has gained maturity and exposed its diversity and heterogeneity in the last decade. The discussion of OSS-aware education has the following requirements:

- Position of OSS in an education that utilizes OSS,
- Position of OSS in an OSS-aware curricular of computer software education,
- Identify a framework to provide measures of OSS-awareness in computer software education.

IV. OSS-aware Software Education Model

There is increased use of education tools in academic courses, e.g. Moodle. It is important to extend OSS-aware software education beyond the use of software tools.

The author proposes the OSS-aware Software Education Model depicted in Fig. 1. This layered view provides a multi-faceted view of OSS-aware software education, from infrastructure, to courseware.

V. Impacts of OSS on software education

A. Real Programming

Legacy computer software education teaches programming techniques. This means that courses do not teach the joy of programming, but only methodologies for programming. It is analogous to a cooking course that does not teach the joy of cooking, but only how to use a knife. This misses has the main point of education: increasing motivation.

Without OSS, it was really difficult to teach real-world computing at academic sites. However, the situation was drastically changed by the wide acceptance of OSS. Now, the teaching of real programming can be done on the Internet, without any restrictions on the locations of facilities. This is also a good opportunity to educate how to work with people with diversity.

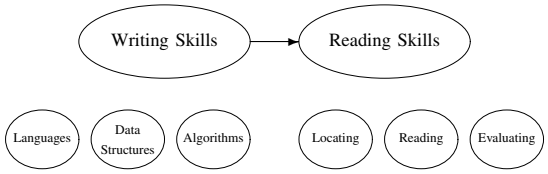

Fig. 2. Transition from writing skills to reading skills

B. Global Distributed Software Development

The core part of today's programming is distributed development. Without this, the programming courses lack the reality of computing. OSS provides an ideal context for teaching global distributed software development.

C. From Writing to Reading

OSS represents a turning point that indicates a shift in emphasis from writing to reading. It is more important to focus on reading skills rather than writing skills. Rather than having a lesson that consists of an exercise of writing 100 lines of code, a computer programming course should teach how to read tens of thousands of lines of code. It is more important to read and evaluate existing source code than to write code from scratch.

In legacy programming education, there is an assumption that people need to understand programming basics and perform simple programming exercises. This assumption is based on a focus toward writing source code. It assumes that most of the students will be engaged in some kind of code writing, even though not all students will become programming gurus. The basic building block of software development competence is writing code from scratch with knowledge of data structures and algorithms. This basic assumption needs to be revisited in an era where externally available web APIs and APIs provided by SDK are integrated with in-house code.

The transition is depicted in Fig. 2.

D. Tools and Process

In the early days, the tools were available at academic sites were not comparable to those found in the enterprises. This hindered the quality of the development environment. Now that OSS has become mainstream in the software industry, and a lot of large-scale software code has been implemented using OSS practices, the tools have improved to a commercial grade. It is well worth teaching these tools for software development in academic facilities. And, since OSS has adopted the widely accepted best practices of software development, it is worth teaching the software development process that is followed in OSS.

E. Quality Assurance

In the old days, computer software education in academics focused on programming, and lacked the viewpoint of quality

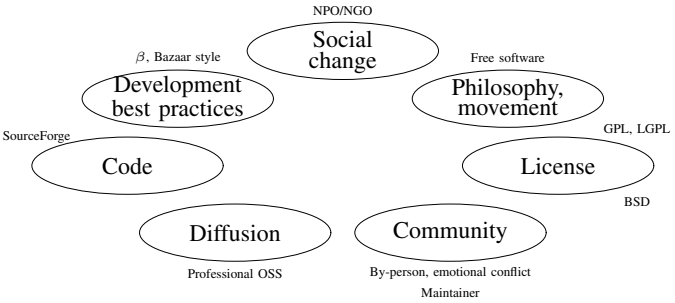

Fig. 3. Seven views of OSS.

assurance. Since OSS is autonomous and globally distributed, quality assurance is an important issue in OSS. OSS case studies offer excellent opportunities to teach testing and quality assurance.

VI. Implications of OSS-oriented software education

A. Multi-faceted Nature of OSS

The skills and experiences gained from OSS development are proliferating and are leveraging the creation of new OSS projects for different roles, goals, and communities.

The globalization and advances of computer and communication technologies facilitate of even more complicated OSS projects. Attitudes towards OSS and the social acceptance of OSS have improved through the evolution of OSS.

This increased acceptance further exposes different aspects of OSS.

The seven views of OSS are illustrated in Fig. 3. These seven aspects represent the different functions that OSS performs. As OSS has matured, each aspect has evolved independently, leveraging complicated relationships among all aspects. The autonomous distributed nature of OSS has harnessed the diversity and heterogeneity of OSS.

B. Community Computing

Interactions within a community are a fundamental aspect of software development. Using global distributed development, communication, and coordination skills is crucial in software development. OSS provides an opportunity to teach such practices through real programming.

Development in a loosely connected group with shared social norms and values provides a new perspective on software development for students of software engineering. It is a departure from the days of one-man computing. Collaboration is the fundamental factor in real world computing.

Project management in a heterogeneous group also provides a good educational material for learning software project management skills.

C. Legal Issues

Copyright and patent protection is an indispensable part of the software business these days. OSS represents opportunity to teach legal aspects of software development. Good courseware for copyright, handling of derivative works, licensing including multiple licenses is needed.

D. Business Models and Professional OSS

In some cases, OSS provides an opportunity to teach business models in the software industry. The software industry has depended heavily on business models. Early examples include the leasing of main-frame computers by IBM. Utility computing in the cloud computing is a recent example.

OSS provides publicly available software code, which improves visibility and leverages the sharing of technical information. Business model engineering with these assumptions is a real challenge in real world business, and good educational material for students of software engineering. Basic techniques like dual licensing are good topics for today's software engineering.

E. Social Changes

OSS provides a good educational opportunity to teach public participation in software inspection, where many lifelines are supported by the software.

Publicly available source code can leverage new standards in developing countries. This methodology brings with a new method of diffusion for new social standards. It also facilitates increased awareness of public participation in technical and social changes.

This is another good opportunity for students of the information era. And it can be applied to students in areas other than software engineering, too.

OSS is a departure from free software, to some extent. OSS-based business model development is very active In the current context. Commercialization was a difficult topic for academic institutions, however, OSS provides an opportunity to teach commercialization with industrial best practices such as dual licensing.

F. Organizational OSS

OSS is a distributed autonomous best practice in software development. In order to coordinate this, it is important to provide proper governance. OSS provides an educational opportunity to teach non-profit organizations how to manage software development.

To build a non-profit organization, it is important to build shared software in a non-core business, for cutting-edge strategic importance, or through shared maintenance among business competitors.

Communication is becoming increasingly important in software engineering, especially in large-scale projects.

Building and managing organizations to promote OSS is an important educational subject in the software engineering. It is an important topic for engineering in general.

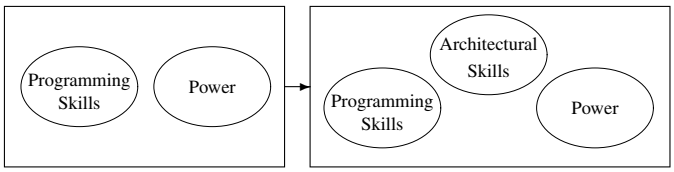

Fig. 4. Transition to Architectural Skills

VII. Paradigm Transitions

A. Software Engineering

Software engineering has been significantly impacted by the emergence of OSS. More generally, this is the era of utilizing external resources.

In legacy computing, two things are required in programming. One is skills and the other is power. The skills required are programming language and algorithm skills. The power required is the physical and mental energy to pursue programming and debugging with a deadline approaching. The combination of these two skills facilitated the software development.

This is no longer applicable in today's software development landscape. Mastering software development skills requires full coverage the three competences, skills, power, and architectural capability. Architectural capability is the competence to make use of externally available resources and to design the interaction between the externally available resources and the in-house code under development. Without this competence, the combination of programming skills and power do not make much sense.

The transition is depicted in Fig. 4.

For example, the author learned a lesson from a student programming exercise. It was an exercise for student teams to perform some programming code for the Android platform, and required several weeks. One team implemented Tetris, a popular game. It required 6,000 lines of programming with highly integrated collaboration. The other team implemented an augmented reality tool. With this code, a user can take a photo and input some destination address or landmark, which will display a virtual arrow signal on the photo to indicate the direction in which the designated destination lies. When the second team was asked about the size of their code, they could not provide a specific answer. Their guess was around 300 or 1,000 lines of code. This demonstrates an important reality in the today's software development. The capabilities to manage Google Map and a camera are provided in the Google Web APIs and the Google Android SDK. They did not need to code these functions. They said that the hardest part of this programming was to obtain the author key for the Google Web API for which they consumed two days to parse the explanation in the programming guide. The programming consists of a sequence of Web API calls and framework API calls, and they did not even bother counting the number of lines in their code. This highlights the importance of architectural competence in the current landscape of software development.

B. Information Retrieval

Code-reading skills heavily depend on code-searching skills. While reading code, most of the time is spent searching for functions, types, and definitions. Code-searching skills are essential when reading code.

It should be noted that this Internet era provides a large-scale repository in terms of the Internet. The fundamental skill, not only in software engineering, but also in business model engineering or any other information engineering, is to search external resources and to integrate them into the problem solving. From this aspect, the shift from writing code to reading code is a fundamental shift in this era of Internet knowledge. There is a large volume of knowledge and resources available externally to the general public, and it is crucial to locate, parse, and utilize them.

C. Framework Computing

In order to facilitate the improvement of reading skills, it is important to improve skills that involve using reading tools. It is a good question, what is the most appropriate reading tool for reading software code? The most feasible answer is that it depends on the coding environment.

Framework computing is another computing trend visible in today's industrial landscape. In legacy computer software education, students learn how to write the *main() routine for C, for example. Writing a main() routine or an event dispatcher was important in the era of programming from scratch. Nowadays, approximately 80 % of functions are provided by the framework. The framework implements a main routine, or an event dispatcher. Because 80 % of functions are already built into the framework, it is crucial to use framework-conforming reading tools to understand the interaction between the code and the framework. This is the reason why programmers use the development environment-specific reading tools bundled into each SDK with the knowledge of each specific framework. For example, for iPhone programming, the Xcode IDE environment is used to read the code. For Android programming, the Android SDK is used to read the code.*

In terms of the author's personal experience, he was reminded that one of his bosses told him that he needed software skills as well as electric circuit engineering skills in early 1970's. The boss told him that in the late 1980's. It was the era of software, therefore, he took it as granted. However, now we are witnessing the era of the Internet, and, externally available resources. Considering the rapid advances in technology over the last two decades, it is difficult to believe that the transition from programming to something else has happened in the last two decades, in the same way as that the transition from hardware engineering to software engineering occurred from the early 1970's to the late 1980's. Framework computing is

Fig. 5. Transition of Engineering

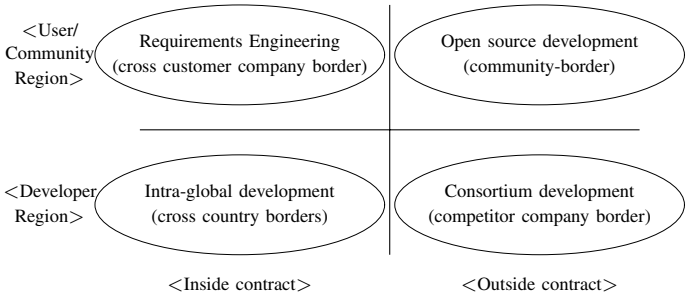

Fig. 6. Cross-boundary Software Engineering

one of the candidates for the target from this transition from programming. This is depicted in Fig. 5.

D. Cross-Boundary Engineering

In order to place the above mentioned shifts of focus in software development, the author proposed cross-boundary software engineering, as illustrated in Fig. 6 [14].

The important shifts of key competencies in software engineering are observed where the software development process includes crossing a boundary. In the requirements engineering, the requirements engineering team works closely together with customers. In large-scale systems, there are many requirements from customers. They originate from different customer departments, and sometimes conflict. For example, performance requirements and security requirements are conflicting in some cases. Large systems include different customers from different departments, which need to be coordinated. It is difficult for a customer to capture a large-scale system, therefore, the requirements team needs to address how to map from a set of requirements to a technologically and economically feasible system. System development and deployment requires a long-term relationship with customers in different departments.

In a foundation-based organization, technical teams from different companies need to coordinate large-scale code management, which is not based on a commercial contract. The collective value of collaboration is pursued in this context. Drastic decrease of global communication and coordination costs is one of the key factors of this phenomenon.

In intra-company development, globally distributed software development is common. There are cases where the resources in emerging countries are used to increase the efficiency of development. Also, there are cases where the resources close to the customers are involved in the improvement of customer relations and deployment engineering.

In the case of open source software, OSS communities are developers and bleeding-edge users who facilitate incremental, and short lead-time development. It is important to cross the boundary with non-contracted developers and users.

This indicates the importance of training related to communication skills in software engineering education. This coincides with Patternson's discussion arguing that writing documentation for portions of open source code could be an assignment to facilitate learning of a large system because documentation is important, and yet rare, in the classroom and in the open source movement [8].

VIII. Discussion

A. Advantages of OSS-aware Education Model

The proposed model identifies multiple aspects of OSS and software engineering education. OSS has continued to expand its coverage and impacts on business and ecosystems. The magnitude of the impacts of OSS has been long underestimated. The changes caused by OSS are big and multiple aspects of software education need to be reengineered.

The proposed model clarifies this multi-faceted impact of OSS for software engineering education. OSS impacts the tool aspect of education software, the practical aspect of the engagement of real and distributed software engineering, the basic skills for programming, the communication aspects of software engineering, and the legal aspect of software education.

The open diffusion of OSS influences innovation engineering and alliance-engineering, which are inevitable factors in today's engineering. The use of external resources is essential in today's world. The penetration of OSS is a departure from the private and closed knowledge, which in turn impacts the broad range of education in general.

The author considers that the Internet, the mobile Internet, and OSS to be three major collisions of technology to the everyday reality. These things have collided heavily with the legacy world and fundamentally changed the today's world in many senses. The impacts cover technical, economical, social, and organizational aspects. OSS is the last in the series, however, its impact is not the least. It brings a potentially huge impact on software engineering education, and education in general.

B. Implications for Software Education

OSS has been largely underestimated in the software industry. Similarly, the impacts of OSS have been underestimated in the domain of education.

OSS was conceived as something open, free, and easy to use. This is largely untrue in many aspects. OSS has become one of the most complicated prolegomena in the world. It is a challenge to identify exactly what OSS is. The diversity, heterogeneity, and complexity of OSS has continued to grow.

And, the impacts of OSS on education have also increased its diversity and complexity. This includes the use of OSS, visions of OSS-aware education, and OSS-aware reengineering of engineering skills.

The growth of OSS inevitably increases the need for OSS-aware education, especially in area of software engineering. The increased complexity of OSS also impacts the educational tools, processes and goals of software engineering. This interrelation is just beginning and we are at the very early stage of OSS-aware software engineering education or OSS-aware education in general.

C. Limitations

This is a descriptive study. In-depth examination of education management, and quantitative verification, are beyond the scope of this paper.

OSS has tight relationships with open innovation, OSS-based social activism, and OSS-based business model engineering. These aspects of OSS-aware education are beyond the scope of this paper.

IX. Conclusion

OSS has become mainstream in software development methodology. Social acceptance has improved over the past decade. This has brought a shift in gravity in the computer software education from writing skills to reading skills.

It is rather a universal phenomenon in this era of the Internet, and is fostering utilization of external publicly available resources.

Characteristics of OSS such as open innovation, global distributed development, community computing, business model engineering using publicly available resources, and utilization of external resources, are universal skills and techniques used in the today's problem solving.

OSS is not just a method of reducing costs for educational software, but has profound meaning for education for coming generations. The author summarizes the current changing landscape of OSS and presents implications of education reengineering caused by this evolution of OSS. Some of implications are outside the scope of software engineering, and can be applied to general education in the 21st century.

The author hopes this paper serves as a starting point to raise awareness of this profound change in software education, and education in general, in this era of knowledge on the Internet.

References

[1] *H. Coll, D. Bri, M. Garcia, and J. Lloret, "Free software and open source applications in higher education," in* EE'08: Proceedings of the 5th WSEAS/IASME international conference on Engineering education. *Stevens Point, Wisconsin, USA: World Scientific and Engineering Academy and Society (WSEAS), July 2008, pp. 325–330.*

[2] *S. W. Rooij, "Open source software in us higher education: Reality or illusion?"* Education and Information Technologies*, vol. 12, no. 4, pp. 191–209, December 2007.*

[3] *W. Christian, M. Belloni, and D. Brown, "An open-source xml framework for authoring curricular material,"* Computing in Science and Engg.*, vol. 8, no. 5, pp. 51–58, September 2006.*

[4] *K.-B. Yue, T. A. Yang, W. Ding, and P. Chen, "Open courseware and computer science education,"* J. Comput. Small Coll.*, vol. 20, no. 1, pp. 178–186, October 2004.*

[5] *M. Belloni, W. Christian, and D. Brown, "Open source physics curricular material for quantum mechanics,"* Computing in Science and Engg.*, vol. 9, no. 4, pp. 24–31, July 2007.*

[6] *M. J. Hawthorne and D. E. Perry, "Software engineering education in the era of outsourcing, distributed development, and open source software: challenges and opportunities," in* ICSE '05: Proceedings of the 27th international conference on Software engineering. *New York, NY, USA: ACM, May 2005, pp. 643–644.*

[7] *K. J. O'Hara and J. S. Kay, "Open source software and computer science education,"* J. Comput. Small Coll.*, vol. 18, no. 3, pp. 1–7, February 2003.*

[8] *D. A. Patterson, "Computer science education in the 21st century,"* Commun. ACM*, vol. 49, no. 3, pp. 27–30, March 2006.*

[9] *R. Morelli, A. Tucker, N. Danner, T. R. De Lanerolle, H. J. C. Ellis, O. Izmirli, D. Krizanc, and G. Parker, "Revitalizing computing education through free and open source software for humanity,"* Commun. ACM*, vol. 52, no. 8, pp. 67–75, August 2009.*

[10] *S. Chopra and S. Dexter, "Free software, economic 'realities', and information justice,"* SIGCAS Comput. Soc.*, vol. 39, no. 3, pp. 12–26, December 2009.*

[11] *D. M. Berry and G. Moss, "Free and open-source software: Opening and democratising e-government's black box,"* Info. Pol.*, vol. 11, no. 1, pp. 21–34, January 2006.*

[12] *A. Meneely, L. Williams, and E. F. Gehringer, "Rose: a repository of education-friendly open-source projects," in* ITiCSE '08: Proceedings of the 13th annual conference on Innovation and technology in computer science education. *New York, NY, USA: ACM, June 2008, pp. 7–11.*

[13] *G. W. Hislop, H. J. Ellis, A. B. Tucker, and S. Dexter, "Using open source software to engage students in computer science education," in* SIGCSE '09: Proceedings of the 40th ACM technical symposium on Computer science education. *New York, NY, USA: ACM, March 2009, pp. 134–135.*

[14] *T. Yamakami, "Cross-boundary software engineering: Implications of engineering paradigm shift," in* ICIS 2010. *IEEE Computer Society, June 2010, pp. 536–540.*