Process Improvement of Peer Code Review and Behavior Analysis of its Participants

WANG Yan-qing
School of Software
Harbin Inst. of Technology
Harbin 150001, P.R.China
86-451-86417732-200
yanqing@hit.edu.cn

LI Yi-jun
School of Management
Harbin Inst. of Technology
Harbin 150001, P.R.China
86-451-86414007
liyijun@hit.edu.cn

Michael Collins
School of Computing
Dublin Inst. of Technology
Dublin 8, Ireland
353-1-402-2866
michael.collins@comp.dit.ie

LIU Pei-jie
Printing House
Harbin Inst. of Technology
Harbin 150001, P.R.China
86-451-86281378
lpj1378@yahoo.com.cn

ABSTRACT

The software industry is urging on universities and colleges to cultivate software engineers who can write high quality programs. Peer code review (PCR) is accepted as an ideal way to maximize the learning outcome of students in writing quality code. Using this learning process, students improve their skills while scientific and efficient management removes the extra burden from instructors such as checking programs written by every student. In this paper, the previous PCR process was improved and the definitions of the relevant roles and documents were refined as well. After implementing this process in two academic years, some problems were found. By means of summarizing the email submissions by the students and also interviewing a few students, the behavior of all participants was preliminarily analyzed. With regards to the further quality assurance and high efficiency, a web-based management information system with a built-in blind review mechanism was discussed for solving the problems with process control, and also a game theory model was proposed for addressing the ethical issues in the whole PCR process.

Categories and Subject Descriptors

D.2.5 [**Software Engineering**]: Testing and Debugging – Code inspections and walk-throughs; D.2.9 [**Management**]: *Software quality assurance (SQA)*

General Terms

Management, Performance, Design, Human Factors, Verification.

Keywords

peer code review (PCR), behavior analysis, software quality assurance (SQA), computer science education, learning outcome

1. INTRODUCTION

The Internet and information technology revolution are accelerating the increase in size and complexity of software making the software crisis problem, yet to be well solved, more rigorous [13]. As a result, the software quality assurance is receiving more attention by the global software industry. Since cooperative and active learning attracts emphasis in the education circle, many computer science educators have become more interested in introducing code review into their courses [5,6,11]. Even though code reviews are time consuming, they are much more efficient than testing [19]. A typical engineer, for example, will find approximately 2 to 4 defects in an hour of unit testing but will find 6 to 10 defects in each hour of review code [19].

Generally, code review research is divided into three categories [9,16,17]: self code review [19], peer code review and tutor code review, among which PCR is recognized as the most practical [17]. PCR is a technique which is generally considered to be effective on promoting students' higher cognitive skills [9], since students use their own knowledge and skill to interpret, analyze and evaluate others' work to clarify and correct it [2].

The research on PCR can be classified into three types, (i) specific PCR approaches [2,5-7,9,11,16,17], (ii) performance improving or quality assurance [1,10] in PCR, and (iii) supporting tools to enhance the learning outcome of PCR [3,4,8,12,14,18]. Different from other PCR approaches just mentioned, the process proposed in this paper is a well-defined process; it is based on email submission and has a high practicability.

A PCR process was refined based on our previous publication [17] and the refined process will be presented in section 2. After conducting on Year 2 students using this approach, the students enhanced their ability to assess other students' work and this improved the quality of their own future work. It was found that this PCR process has a high operability and thus a better learning outcome was achieved. Nevertheless, after implementing this process during two academic years, i.e. 2005-2007 in an introductory programming class, some problems were found when checking the documents submitted by the students and also interviewing some students. The analysis of these problems will be put forward in section 3 where possible reasons that gave rose to these problems are discussed. The behavior of all three roles is analyzed in section 4. Also, the actions to solve these problems such as grouping strategies are offered in section 5.

2. IMPROVEMENT OF PCR PROCESS

2.1 Improvement of Each Role's Definition

In [17], the acceptor, inspector and instructor were taken as the names used for the roles. These names sometimes differ in other literatures so it was determined to reference a popular naming convention used for these roles in publication industry. The following modifications were made:

SIGCSE'08, March 12–15, 2008, Portland, Oregon, USA.

(1) *Author* is a student who accepts the code review activity by someone else;

(2) *Reviewer* is a performer who does the code review activity;

(3) *Reviser*, similar to the naming system in current publication industry, is the author himself/herself. When an author has accepted the form containing the comments and begins his/her revision work, he/she now changes role and becomes a reviser;

(4) *Instructor* is the teacher or qualified teaching assistant responsible for the programming lab class [17]. They check the written code, review results, deliver grades to students, and tutor several students who are slowing in pace. All the activities depend on the relevant documents including: manuscript code, comments form and revision code.

2.2 Improvement of Documents' Definition

To facilitate future research and simplify communication with other scholars, the relevant documents in [17] are redefined as follows:

(1) *Manuscript code* is source code the author has just completed. It is encouraged that the author reviews their own code before submitting, thereby making the following steps more efficient and productive.

(2) *Comments form* carries the comments that the reviewers make when they review the manuscript code written by the authors. As described in [17], the comments form may be filled with design defects, coding defects, code not complying with coding standards, and possible improvements, etc.

(3) *Revision code* is the final revised program after making changes to the manuscript code based on the received comments form. The quality of the revision code depends on the quality of the manuscript code, the reviewer's thoroughness, and the care of the reviser.

(4) *Reference solution* is the answer to the assignment and held by the instructor. It should be the complete program and will be used to assess the learning outcome of the author, reviewer, and even the reviser when required.

2.3 Improvement of Process

Besides the standardization and expansion to the roles' definition and documents' definition, the whole process was improved. Some phases were redefined more precisely and some were adjusted in time sequencing (Figure 1). In Figure 1, the two activities beginning with the letter *S* belong to the *submit* phase and all the activities beginning with letter *Q* fall into the quality assurance phase.

The refined PCR process consists of 6 phases which are clearer than the definitions in [17]. All phases are described as follows:

*Phase 1 – **W**rite*. A student (author) completes his/her own assignment programs (manuscript code) that may pass the compilation and running test (letter *W* in Figure 1).

*Phase 2 – **S**ubmit*. The author emails the manuscript code to the instructor as soon as possible (letter *Q.1* in Figure 1) to enable the instructor to perform the quality management. Following this, the blank comments form (letter *S.1* in Figure 1) is sent to the specific reviewer together with the associated manuscript code (letter *S.2* in Figure 1).

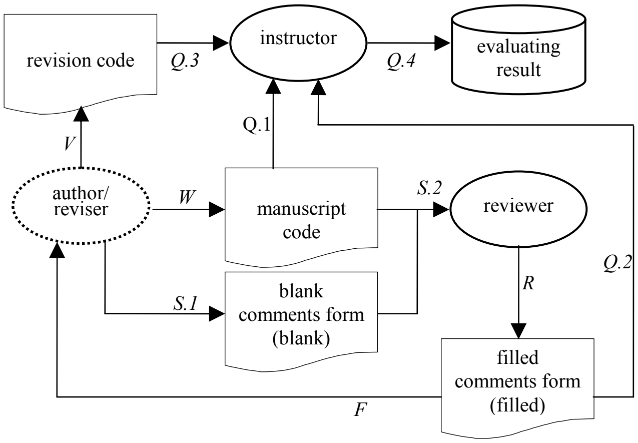

Figure 1. Flowchart of the improved PCR process.

*Phase 3 – **R**eview*. When the reviewer receives the manuscript code, they should perform the review as soon as possible. If there are defects or advices, suggestions or meaningful comments are entered carefully into the comments form (letter *R* in Figure 1) sent by the author.

*Phase 4 – **F**eedback*. The reviewer sends the completed comments form to the author (letter *F* in Figure 1), and send its carbon copy to the instructor for quality management (letter *Q.2* in Figure 1).

*Phase 5 – Re**v**ise*. After receiving the comments form, the author makes revisions to their manuscript code (letter *V* in Figure 1) referencing the reviewer's comments and suggestions. The completed revision code will be emailed to the instructor for quality management (letter *Q.3* in Figure 1).

*Phase 6 – **Q**uality assurance*. Having the manuscript code in *Q.1*, comments form in *Q.2*, and revision code in *Q.3*, given sufficient time, it is not difficult for instructors to check that the author and reviewer perform their work responsibly. Through instructor's checks (letter *Q.4* in Figure 1), it is easy for students' marks to be saved in the evaluating result database.

3. PROBLEM FINDING

In the above PCR process, it is assumed that all the participants fulfill every phase carefully and responsibly. However, it was not very satisfactory. Through the Object Oriented Programming (OOP) laboratory class in the HIT-DIT joint programme [1] in spring semester of 2007, it was found that the design of this PCR process was not perfect even though most of the students received satisfactory marks in this course. This may have been a result of careless authors, irresponsible reviewers and busy instructors in the review process, which might discount the learning outcome of the whole PCR process.

[1] A joint programme developed by the School of Software at Harbin Institute of Technology and the School of Computing at Dublin Institute of Technology starting from September 2003, its objective is to cultivate industry oriented undergraduates.

By summarizing students' email submission and interviewing several students after they finished their academic year, some problems were discovered and discussed as follows:

(1) Some students' lack of qualification. A few students write very poor programs which are difficult and boring for the reviewers to review carefully. Also, some students lack the programming ability to be qualified reviewers so they either give few comments or write lines of confusing comments. After the preliminary analysis, it was believed that these two problems are associated with the students' qualification and can be solved by specific training so they are not the focus of this paper.

(2) The process flow is difficult to control. With the OOP course in the spring semester of 2007, nine students[2] were taught six chapters from chapter 9 to 14 of a textbook. The current PCR process is based on e-mail and we received 178 mails in total (see Appendix 1 for partial data in MS Excel format). Since it is not easy to obtain information from APPENDIX 1, the data was retrieved from the 178 emails and the submission problems by students were summarized in Table 1.

Table 1. Summary of submission problems by nine students.

Student	Late submission	Missed submission	Invalid submission	Repeated submission	Cluster submission	Total
S1	0	2	0	0	1	3
S2	0	3	1	2	1	7
S3	2	2	3	1	1	9
S4	2	2	0	0	0	4
S5	1	1	0	0	0	2
S6	1	1	0	0	0	2
S7	1	0	0	2	0	3
S8	6	3	1	0	1	11
S9	6	0	0	15	1	22
Total	19	14	5	20	5	63

Although students were asked to submit their work complying with the sequence described in Figure 1, a few of them did not submit on time. From Table 1, it can be found that *repeated submission*, *late submission* and *missed submission* are the top 3 problems. The *repeated submission* results from that a minority of students sometimes worry whether their emails have been received by the instructor. The problems of *late submission* and *missed submission* arise when students forget to send their work (carbon copy) to the instructor. In essence, the email-based strategy and the somewhat complicated process are the cruxes of all the five main problems listed in Table 1 and they make it impossible for the instructor to control the quality of this process.

(3) Conspiracy exists. The fixed grouping strategy[3] provides the opportunity for authors and reviewers, having a good relationship, to invent a method to blunder through. In order to justify our claims, we made four separate interviews (three face-to-face interviews and one telephone interview) to four students[4] in the autumn semester of 2007 (see Table 2 for recorded problems).

In our interviews (see Table 2), the problems *P4*, *P5* and *P6* are still associating with process control just mentioned in (2) of this section. The problems *P1*, *P2*, *P3* and *P8* belong to attitudinal issues which will surely affect the learning outcome of students. By the way, although some of the learning objectives can be met with *P7*, it is not encouraged because the quality of the whole PCR cannot be guaranteed with it.

The last problem *P9* is a means of data falsification. One of students in the same review group finishes the assignment program; their group mates might copy the written program and make some changes, such as adding some program comments or even make some negligible defects. After that, the author's manuscript code enters the whole PCR process. This behavior causes a severe negative influence on the process assessment and its quality assurance.

Table 2. Recorded problems in four separate interviews.

Student	P1	P2	P3	P4	P5	P6	P7	P8	P9
1			X	X			X	X	
2								X	X
3								X	X
4		X						X	

Note:

If a problem once happened to a student, the item was checked with an X.

P1 - Write code instead of the author;

P2 - Review code instead of the reviewer;

P3 - Revise code instead of the reviser (author);

P4 - Did not cc manuscript code to instructor when submitting to reviewer;

P5 - Did not cc the comments to instructor when emailing to author;

P6 - Did not email the revision code to instructor when finishing it;

P7 – Perform a private review in advance of the formal PCR process;

P8 - Rush through the review;

P9 - Copy code from a good programmer and do a minor modification.

Therefore, it is necessary to analyze the behavior of all the participants, to address the crux of the problems and to build up a control mechanism to maximize the risk of every irresponsible game player so as to achieve a better learning outcome.

4. BEHAVIOR ANALYSIS

4.1 Behavior Analysis of Author

The behavior of author depends. Author students with strong self-discipline always write their programs carefully and positively, so the comments form to them is often very short or becomes shorter even when the program size increases. When they find the comments are constructive and valuable, they will revise their code sincerely and carefully. However, there are a few author students who might perform in the following two ways: (i) do not pay attention to their written manuscript code, neither compile nor test the code, or rush to send the email to the reviewer and wait for their comments; (ii) after receiving the comments form, the author will briefly look through it and make only casual revisions.

The two ways just mentioned are the particular cases. It is more common that an author with poor self-discipline will often guess whether the reviewer will review their manuscript code carefully and whether the instructor can find time to inspect them. Weighing the careful work against the penalty for careless work, the author can decides to finish an assignment carefully or not.

4.2 Behavior Analysis of Reviewer

The behavior of reviewer is diverse too. Some positive and responsible reviewers might finish review work consistently while other reviewers might not always be devoted in their roles. Most

[2] There are nine students in this class in total.

[3] The groups are fixed in the whole course.

[4] The other students went to Dublin to continue their study in September, 2007.

of the reviewers may not be willing to do their best to review the written code by their peer students, especially when they are assigned poor program writers. Even though students can enhance their own programming ability when they are reviewing other student's programs [17], it is no doubt that reading very poorly written programs frequently is not desirable to the reviewers. The original power for code reviewing is not enough. Mere encouragement by the instructor cannot educate excellent reviewers and the data in columns *P8* and *P2* of Table 2 helps us make this conclusion.

Since the reviewer is the key role in this PCR process and determines the stand or fall of this game, a strategy has to be explored to make the reviewer review code carefully and thoroughly. For example, instructor can give bonus marks to responsible reviewers and minus marks to irresponsible ones.

4.3 Behavior Analysis of Instructor

The behavior of instructor cannot be neglected due to the quality assurance concerns.

(1) At the very beginning of introducing PCR process, every instructor might try to inspect all submission (manuscript code, comments form and revision code) timely and carefully. The inspection must take time and effort. Although the cost of inspection is relatively lower than the case when PCR has not been introduced, the time cost of inspection cannot be ignored, especially in such countries as China where the number of students is far greater than that of instructors. So it is almost impossible for instructor to inspect all submissions for every assignment by every student.

(2) After finding the impossibility of inspecting all submission of students, instructors might change strategy to inspect comments only. There raises a new problem unfortunately because the comment forms submitted by reviewers cannot tell the instructor the whole story. As just mentioned in the behavior analysis of author and reviewer, many personality factors are involved in the whole PCR process, so it is difficult for the instructor to acquire all information. The similar cases in software companies can help us make sense of it. On one hand, a long testing result sheet often tells the managers that programmers did a bad job while testers did a good one except that the testers catch at shadows. On the other hand, a short testing result sheet results from either a careful programmer or a lazy tester. The similarity between code review and software testing sets the instructor a challenge of efficient management.

(3) Since the limitation of comments by reviewers, the instructor often performs inspection with a certain probability. The random check sounds like a good idea since the instructor could likely spend minimal time and reach maximal learning outcome. However, without the scientific policy such as reward and punishment mentioned above, the desire to save time will be a cause of a lazy instructor.

The behavior analysis of instructor can tell us that the whole PCR process has to be supported by a certain mechanism in order to minimize the instructor's working load while also maximizing the overall learning outcome of students, with which an instructor may see exciting learning curves of his/her students.

From the overall analysis to three roles in PCR, it is easy to find that the stand or fall of this game depends on many personality factors, and the behaviors of these three parties have obvious motivations and characteristics of game theory [15]. So in order to obtain learning objective, the application of game theory may help get to the destination.

5. Discussion on Grouping Strategy

Based on the above problem findings and behavior analysis, it is concluded that one major crux of the problems in PCR is grouping strategy. Up to this point, we have attempted two kinds of grouping strategies: *pair review* with 2 students in a group in autumn semester of 2006 and *circle review* with 3 students in a group in spring semester of 2007. Since the current grouping approach is subject to the threat of conspiracy or falsification of submitted work, it is necessary to discuss some new grouping strategies and analyze the practicability of each.

(1) Switching the *2-student* or *3-student* grouping strategy to *n-student* grouping strategy, in which the *ring-wise review* approach could be adopted. If so, the management complexity will increase and it will become ineffective after a period of time, especially in a small size class.

(2) Switching a *fixed* grouping strategy to a *random* grouping strategy. This will also increase the management cost because the instructor has to generate the grouping result with every assignment.

(3) Switching a *P2P* (one reviewer to one author) strategy to a *T2P* (multiple reviewers to one author) strategy. This idea sounds like effective but it is not practical because of the high cost especially in such nations as China and India where the number of students in one class is very large. So *T2P* can only be taken as the strategy of a case study, for example, could be applied to learn from some excellent students or to help some weaker ones.

(4) Introducing *blind review* mechanism. In view of the above behavior analysis, all conspiracy activities come from the unfavorable usage of good relationship between authors and reviewers. Undoubtedly, blind review mechanism could well solve all the conspiracy problems for the students do not know who their partners are. By the way, in order to make the blind review more practical, the web-based management information system (MIS) has to be relied on. The MIS with a built-in blind review mechanism may appear in our future publication.

(5) Exploring a *ranking policy* when grouping. On one hand, it is a waste of resource if high level students are often assigned to review the code written by low experienced students. The reviewers' feeling of accomplishment will diminish gradually. On the other hand, it is not wise to ask low experienced students to review the programs written by high level students either. The probably complicated programs may frustrate the reviewers while the author's code has not been adequately reviewed. So a ranking policy should be explored in the PCR process to make it a true *peer*, by which a student who has a slightly higher programming ability than the author will be assigned to be the reviewer. This will be a topic of our future work too.

6. CONCLUSION AND FUTURE WORK

The PCR process in our previous publication [17] was improved along with the definition refinement of related roles and documents so that the future research could be standardized and the communication with other scholars might be facilitated.

The improved process was not perfect. Some problems were found after the implementation in two academic years, such as lack of qualification, difficulty of process control, conspiracy activities, and so on. Thereupon, a series of behavior analysis of the three parties was implemented based on email submission problems and interview results. Finally, the following conclusions were made: (1) In computer science angle of view, a web-based MIS with a built-in blind review mechanism should be developed; (2) In ethical angle of view, a game theory model should be constructed. These strategies can enhance the quality of this PCR process, and will be presented in our future publications.

7. REFERENCES

[1] Alessandra Devito Da Cunha and David Greathead. 2007. Does Personality Matter? An Analysis of Code-review Ability. Communications of the ACM. Volume 50, Number 5 (2007), Pages 109-112. http://doi.acm.org/10.1145/1230819.1241672

[2] Ballantyne, R., Hughes, K., and Mylonas, A. 2002. “Developing Procedures for Implementing Peer Assessment in Large Classes Using and Action Research Process”, Assessment & Evaluation in Higher Education, 27(5), 427-441, 2002

[3] Belli, F., Crisan, R. 1996. Towards automation of checklist-based code-reviews. Proceedings of Seventh International Symposium on Software Reliability Engineering. 30 Oct.-2 Nov. 1996 Page(s):24-33.

[4] Bhalerao, A. and Ward, A. 2001. Towards electronically assisted peer assessment: a case study. Association for Learning Technology Journal (ALT-J), 9(1), 26-37, 2001

[5] D.Trytten. 2005. A Design for Team Peer Code Review. Proceedings of the 36th SigCSE Technical Symposium on Computer Science Education. Feb2005. v.37 n.1.

[6] E.F.Gehringer,D.D.Chin,M.A.PerezQuinones, et al. 2005. Panel:Using Peer Review in Teaching Computing. Proceedings of the 36th SigCSE Technical Symposium on Computer Science Education. Feb2005. v.37n.1.

[7] E.Silva, D. Moriera. 2003. WebCoM: a Tool to Use Peer Review Techniques to Improve Student Interaction. Journal on Educational Resources in Computing. March 2003.

[8] Emmanuel Geay, Eran Yahav, and Stephen Fink. Continuous Code-Quality Assurance with SAFE. PEPM ’06 January 9–10, Charleston, South Carolina, USA. 2006, pp145-149

[9] Fallows, S. and Chandramohan, B., Multiple Approaches to Assessment: Reflections on use of tutor, peer and self assessment , Teaching in Higher Education, 6(2), 229-246, 2001 Association for Learning Technology Journal(ALT-J), 9(1), 26-37, 2001

[10] Hidetake Uwano, Masahide Nakamura, Akito Monden and Ken-ichi Matsumoto. Analyzing Individual Performance of Source Code Review Using Reviewers’ Eye Movement. Proceedings of Eye Tracking Research & Applications Symposium 2006(ETRA’2006), San Diego, California, 27-29 March 2006, pp133-140.

[11] J.Sitthiworachart, M. Joy.2004.On-line Feedback and Assessment: Effective Peer Assement for Learning Computer Programming. Proceedings of the 9th Annual SigCSE Conference on Innovation and Technology in Computer Science Education. June 2004.

[12] Jun-Suk Oh, Ho-Jin Choi. A Reflective Practice of Automated and Manual Code Reviews for a Studio Project. Proceedings of the Fourth Annual ACIS International Conference on Computer and Information Science(ICIS’05). 2005, pp37-42.

[13] Murugesan S. & Deshpande Y., 2001, Web Engineering: A new Discipline for Development of web-based systems. In: Web Engineering - Managing Diversity and Complexity of Web Application Development, Springer-Verlag.

[14] Nagoya, F., Shaoying Liu, and Yuting Chen. A tool and case study for specification-based program review. Proceedings of the 29th Annual International Computer Software and Applications Conference, 2005. COMPSAC’2005. Vol 1, 26-28 July 2005 pp375-380

[15] PrajitK.Dutta. Strategy and Game Theory. Printing House of Shanghai Finance and Economics University, 2005: 95-108

[16] Sluijsmans, D., Dochy, F., Moerkerke, G., “Creating a Learning Environment by Using Self-Peer-and Co-Assessment”, Learning Environments Research, 1, 293-319, 1999

[17] WANG Yan-qing, et al. How to Evaluate Students' Learning Outcome: A Peer Code Review Model in Undergraduate Computer Programming Class. Proceedings of the 2nd International Conference on Computer Science and Education. Wuhan, China. July 25-27, 2007, pp1292-1295.

[18] WANG Yan-qing, et al. Quantitative Research on How Much Students Comply with Coding Standard in Their Programming Practices. Proceedings of the 3rd China Europe International Symposium on Software Industry Oriented Education, Dublin, Ireland. Jan 2007, pp116-119.

[19] Watts S. Humphrey. Introduction to the Personal Software Process. Addison-Wesley, Pearson Education, Inc. 2002:159-163

APPENDIX 1: email records in PCR process

Chapter	Author	Reviewer	MCR	Submission Date	Attachment	Comment
14	S6	S8	C	2007-5-11 9:56	Y	4
14	S8		M	2007-5-11 9:55	Y	
13	S6	S8	C	2007-4-20 10:15	Y	3
13	S8		M	2007-4-20 10:14	Y	
12	S8		M	2007-4-20 10:12	Y	
12	S6	S8	C	2007-4-13 12:56	Y	2
11	S8		M	2007-4-6 22:27	Y	
11	S8		R	2007-4-6 22:27	Y	
11	S6	S8	C	2007-4-6 19:51	Y	4
10	S8		R	2007-3-30 15:49	Y	
10	S6	S8	C	2007-3-30 11:30	Y	3
10	S8		M	2007-3-30 11:29	Y	
9	S6	S8	C	2007-3-29 18:38	Y	3
9	S8		R	2007-3-23 16:25	Y	
9	S8		M	2007-3-23 10:13	Y	
…	…	…	…	…	…	…

Note:

MCR denotes this mail is for Manuscript code or *Comments* or *Revision;*

The student names in column Author and column Reviewer are coded as S1-S9 for privacy concern.