A Peer-Assessment Experiment in Object-Oriented Design

Edward F. Gehringer[1]
Ferry Pramudianto[2]
{[1]efg, [2]fferry}@ncsu.edu
North Carolina State University

***Abstract*—Computer science students seem to have great difficulty learning how to create effective designs. We have observed successive classes of students making the same errors, despite emphasis on good design principles in class, accompanied with active-learning exercises. This presentation reports on an experiment to have students assess designs with known weaknesses in class, and then come up with their own designs, on which they receive peer feedback. The hypothesis is that this exercise would increase students' learning gains in performing CRC-card design.**

I. Introduction

The ability to review teammates' work and provide useful feedback without creating negative tensions is an important skill for software engineers when they work in teams. In formal method, peer review can also help to verify formal specifications by several reviewers, which can cancel out individual errors. Related to software development, agile methodology such as pair programming encourages developers to be critical when reviewing their peers' code. Unfortunately, this skill does not come naturally, but research shows that it can be obtained through training.

One method that can be used to train computer science students to become critical reviewers is involvement in assessing the assignments of their classmates. Peer assessment has been used in an educational setting for nearly fifty years. It has demonstrated its ability to promote learning gains [1] in fields of computer science [2]. In the past 20 years or so, it has become more prevalent. A survey of 118 K-12 teachers in a school district in Western Canada reported that 74% had used some form of peer or self-assessment [3]. Many of our students report participation in peer review in some other class, supporting the perception that usage is widespread. The following table summarizes the benefits of peer assessment that have been identified.

TABLE I
PEER ASSESSMENT BENEFITS

Benefit	References
Improves the ability to generalize situations, self-reflections which lead to greater metacognitive self-awareness and critical thinking skills	Wang & Wu, 2002; Topping 2009
Allows learners and teachers to experience multiple perspectives	Brutus & Donia, 2010
Improves transparency of the assessment process, thus student trust	Fry, 1990
Ensures continuous feedback, supports self-reflection & trains self-organization among students	Topping, 2009
Provide students with skills to form judgments about what constitutes high quality work	Topping, 1998
Improves writing skill	Olson 1990;O'Donnell & Topping, 1998
Fosters teamwork skills, promote active learning, communication skills, negotiation skills, and diplomacy	Riley, 1995

II. Our Experiment

We employed peer assessment in a novel way in NCSU's Object-Oriented Design and Development class. It has been observed [7,8] that graduating students have great difficulty creating acceptable program designs. In fact, the second author has observed them making the same mistakes semester after semester, even after being told to watch out for certain deficiencies.

In the class, students are taught UML and the Class-responsibility-collaboration (CRC)-card method of o-o design. CRC cards are a brainstorming tool used in the design of object-oriented software that aims at minimizing the design complexity. For the last several years, the instructor has presented a requirements statement in class and then asked the students to identify the key classes, their responsibilities, and their collaborators. Students have submitted their CRC cards on a Google form, and subsequently the instructor has critiqued the design. Unfortunately, students have continued to make the same mistakes that they were warned against during the critique.

In Spring 2016, we used a different approach to teaching CRC cards. We went through one sample design in class, and then presented the students with the list of common errors in Table II. Students were then asked to run through a simple design of a portfolio-manager system with about a half-dozen classes. After doing this, they worked in teams of two and three, and submitted their designs to the online peer-review system Expertiza [9, 10].

Each student was then asked to identify instances of the eight errors in three versions of the o-o design: two versions with known defects created by the instructor, and one version created by another student team. After submitting their reviews of the instructor-created designs, they were shown a list of defects that the instructor had injected into the design. They could check their answers against the instructor's, and then go back and revise their review of the student-authored design. We hypothesized that this active-learning exercise would allow students to be more engaged with actual o-o designs, and thereby produce more durable learning.

The experiment seemed to be successful. Over half the time, a plurality of students identified the correct number of defects of each type. In only one case were the plurality of students off by more than one in the number of defects of a particular type (in one of the designs, most students missed both responsibilities that had too many collaborators (Error #8)).

TABLE II

COMMON ERRORS IN CRC-CARD DESIGN

1. *Using a class name that is not a singular noun.* "Customers", "Segments", "Buy"
2. *Naming a system class as a key abstraction of the program.* "String", "Date"
3. *Defining a new class where an existing (usually primitive) object would suffice.* "Cost", "Time"
4. *Thinking that something can't be a key abstraction because it is part of a larger abstraction.* "Seat" can't be a key abstraction, because it's part of the plane. "Wheel" can't be a key abstraction, because it's an attribute of the plane.
5. *Confusing inheritance with aggregation.* "Seat" inherits from "Plane"
6. *Confusing an object with an aggregation of such objects.* Responsibilities of Seat include knowing the available number of window, aisle, and exit-row seats
7. *Confusing ambiguity with synonyms.* "Segment" and "leg" are ambiguous with regard to flights, because they mean the same thing.
8. *Treating collaboration as a transitive relationship.* Class: Customer Responsibility: Buy ticket Collaborators: Passenger, Flight, Segment, Airport

III. RESULTS

On the next in-class exam, the instructor included a similar question on o-o design. This question was worth 27 points, or more than one-quarter of the exam. The average score was 22.25, or 82.4% of full credit for that question. Compare that with the average score on the similar question from the fall semester, where the new teaching technique was not used: 21.45 points out of a total of 26 points, or 82.5% of full credit for that question. Unfortunately, there was no significant difference in the two average scores ($p = 0.95$).

Despite this, we are hopeful that our approach really can produce learning gains. We suspect that a single exercise in a one class period is not enough for these principles to become ingrained in students' minds. Future work will involve creating practice exercises that the students can attempt before the next exam.

IV. REFERENCES

1 Topping, K., and Ehly, S.: 'Peer-assisted learning' (Routledge, 1998. 1998)

2 Piech, C., Huang, J., Chen, Z., Do, C., Ng, A., and Koller, D.: 'Tuned models of peer assessment in MOOCs', arXiv preprint arXiv:1307.2579, 2013

3 Noonan, B., and Duncan, C.R.: 'Peer and self-assessment in high schools', Practical assessment, research and evaluation, 2005, 10, (17), pp. 1-8

4 Gehringer, E.F.: 'Electronic peer review and peer grading in computer-science courses', ACM SIGCSE Bulletin, 2001, 33, (1), pp. 139-143

5 Gehringer, E.F.: 'Building an Ethics in Computing Website using peer review', in Editor (Ed.)^(Eds.): 'Book Building an Ethics in Computing Website using peer review' (Citeseer, 2001, edn.), pp.

6 Gehringer, E.F., Ehresman, L.M., and Skrien, D.J.: 'Expertiza: students helping to write an OOD text', in Editor (Ed.)^(Eds.): 'Book Expertiza: students helping to write an OOD text' (ACM, 2006, edn.), pp. 901-906

7 Eckerdal, Anna, et al. "Can graduating students design software systems?." 37th *SIGCSE Technical Symposium on Computer Science Education, SIGCSE Bulletin* 38.1 (2006): 403-407.

8 Loftus, Chris, Lynda Thomas, and Carol Zander. "Can graduating students design: revisited." *Proceedings of the 42nd SIGCSE Technical Symposium on Computer science education*. ACM, 2011.

9 Gehringer, Edward, et al. "Reusable learning objects through peer review: The Expertiza approach." *Innovate: Journal of Online Education* 3.5 (2007): 4.

10 Gehringer, Edward F. "Expertiza: Managing feedback in collaborative learning." *Monitoring and assessment in online collaborative environments: Emergent computational technologies for e-learning support* (2010): 75-96.