Quality of Student Contributed Questions Using PeerWise

Paul Denny
Computer Science
University of Auckland
Private Bag 92019, Auckland,
New Zealand
paul@cs.auckland.ac.nz

Andrew Luxton-Reilly
Computer Science
University of Auckland
Private Bag 92019, Auckland,
New Zealand
andrew@cs.auckland.ac.nz

Beth Simon
Computer Science and Engineering
University of California, San Diego
La Jolla, CA
USA
bsimon@cs.ucsd.edu

Abstract

PeerWise is an online tool that involves students in the process of creating, sharing, answering and discussing multiple choice questions. Previous work has shown that students voluntarily use the large repository of questions developed by their peers as a source of revision for formal examinations – and activity level correlates with improved exam performance.

In this paper, we investigate the quality of the questions created by students in a large introductory programming course. The ability of students to assess question quality is also examined. We find that students do, very commonly, ask clear questions that are free from error and give the correct answers. Of the few questions we examined that contained errors, in all cases those errors were detected, and corrected by other students. We also report that students are effective judges of question quality, and are willing to use the judgements of their peers to decide which questions to answer. We include several case studies of questions that are representative of the kinds of questions in the repository and provide insight for instructors considering use of PeerWise in their classrooms.

Keywords: PeerWise, MCQ, contributing student, question test bank, peer assessment, self assessment.

1 Introduction

PeerWise is an online tool that enables students to create multiple choice questions with appropriate distracters and an accompanying explanation [1]. When these questions are submitted, they become part of a pool of questions that are shared by the entire learning community. Other students in the same course can use the questions for self-assessment, learn from the explanations, evaluate the quality of the questions they answer (using a 6 point, 0-5 scale), and leave comments on questions.

Earlier studies of PeerWise show that students answer many more questions than they are required to, and voluntarily use the PeerWise system during the examination study period as a revision resource [2]. Additionally, students who engaged more actively with PeerWise performed better than their less active peers on a final exam [3].

Since students use and seem to benefit from the content generated by other students, we wanted to investigate the quality of the questions that were being produced. Specifically, we wanted to know "Were the questions created by students of high quality?"

The most important definition of quality for us, as instructors, is whether the questions are an effective and efficient tool for helping students learn what they need to know for a course – and specifically for a portion of a written exam featuring multiple choice questions. The 0-5 rating scale in PeerWise was designed to allow students to express something similar. Assuming they give higher ratings to questions they find most valuable for revision and/or learning, these ratings will be a somewhat subjective measure of overall quality.

In a review of the literature on peer review, Topping [6] reported that almost three-quarters of the studies that compared grades assigned by students with the grades assigned by teachers showed a high degree of consistency between student and teacher assigned grades. Sitthiworachart and Joy [5] reported that grades assigned by Computer Science students engaged in peer review activities had a significant and substantial correlation with the grades assigned by tutors. Although it is clear that students are capable of evaluating the quality of work produced by their peers, we wanted to investigate whether the ratings assigned by students using PeerWise were consistent with the quality assessment of staff. To compare our evaluation of quality with the overall ratings assigned by students using the rating system supported by PeerWise, we rated a sample of questions on the same 0-5 scale.

We found that our subjective ratings did not answer questions we had about more objective characteristics of the student created questions. To be certain that students were not wasting their time or being frustrated by "poor" questions, we wanted to determine whether the great majority of questions were "good" in that they were clearly worded, were free of errors, had a reasonable set of distracters, and provided good explanations. To investigate this, we used a simple rubric for classifying questions according to these characteristics.

Finally, we were interested in studying the quality of the questions as it relates to their representativeness for multiple choice style examinations. Here we categorise the structure of the contributed questions in five ways: (a) is there code in the stem, (b) is there code in the answer choices, (c) is there code in both the stem and answer choices, (d) did the question require computing or matching output, and (e) did the question require identification of an error. In previous work, the Leeds group [4] reported on a large study of questions involving code in the stem only (fixed code problems) or code both in the stem and the answer choices (skeleton code questions). In that study, students scored lower on skeleton code questions than fixed code questions. We were interested to see if students generated questions with similar characteristics. Specifically, we were interested if student generated questions would be skewed towards the types of fixed code questions students answered more correctly in the Leeds study.

In this paper we seek to answer these questions. Essentially we want to know how capable students are of creating questions to be used as a learning resource, what the characteristics of those questions are, and how capable students are of recognizing high quality questions.

2 Methodology

This study is based on a repository of 617 questions collected from a standard Java based CS1 course taught in the first semester of 2008 at the University of Auckland. There were 407 students who attempted the final exam for this course, and 366 of these students had contributed at least one question to PeerWise during the semester. These questions were answered a total of 11,189 times.

Each time a question is answered, the student is given an opportunity to rate the question on a 6 point scale, from 0 to 5. This allows the student to apply their own judgements and make a subjective measure of the overall quality of the question. Figure 1 plots the number of responses each question received against the average rating assigned to that question by the students who answered it.

2.1 Overall quality ratings

Our methodology was driven by an observation we made from Figure 1. It is clear from the figure that poorly rated questions do not receive a large number of responses. Given a large number of questions from which to select, students prefer to choose those that are more highly rated. This is supported in PeerWise as students are able to list all of their unanswered questions in decreasing order of rating. This provides evidence that students do value the ratings that are assigned to questions by their peers, as they use those ratings to help determine how to spend their time revising.

This observation motivated the first aspect of our study. Given that students clearly value the ratings of their peers, we wanted to measure the accuracy of these ratings. We examined a sample of 61 questions, approximately 10% of the repository, by selecting every 10th question in chronological order. For each question in the sample, each author of this paper assigned a subjective quality score to the question using the same 6 point scale (0 to 5) that the students used. In this paper, we will refer to these ratings as "instructor ratings". Our only agreement was that we would define

- 0 “complete nonsense”
- 1 “pretty poor”
- 2 “some merit, but not that useful”
- 3 “average question, learned something”
- 4 “good question, not quite exam quality”
- 5 “good enough for an exam (or almost)”

We expected differences in our application of this scheme, likely based on content areas stressed in the questions. We report on each instructor’s ratings in comparison with each other and with student ratings.

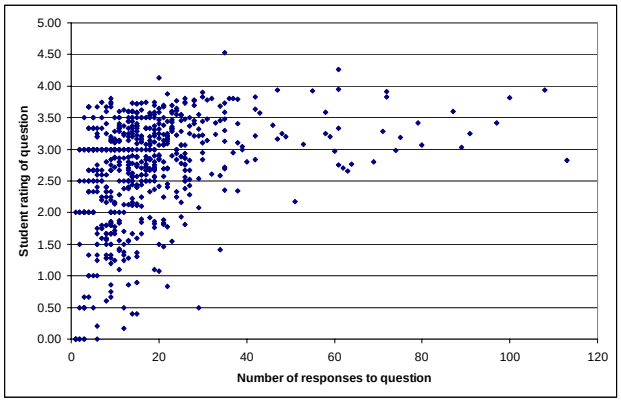

Figure 1: The number of ratings and the number of responses for each of the 617 student questions

2.2 Objective quality ratings

Our overall quality ratings measure the quality of a question as a whole. There are also certain individual characteristics of the questions that we would like to evaluate. The goal here is to improve our understanding of these different elements in order to identify common problems. Ultimately, such an understanding may allow us to help guide students in designing better questions, which would in turn lead to questions receiving higher overall quality ratings in the future.

The rubric we defined and applied to the sample of 61 questions (defined in Section 2.1) is described in Table 1.

One of the elements of this rubric assesses the "correctness" of a question. The correctness of student created questions is likely to be of great importance to instructors. A repository of questions in which a high proportion are not correct would be of little value and it could be argued may even be misleading to students.

2.3 Quality in relation to exams

The nature of PeerWise specifically targets students in preparing and revising for formal examinations. We therefore wanted to evaluate the repository of student produced questions in relation to various “real” exam questions. Previously, the Leeds study [4] had identified two types of problems typically found in CS1 examinations: fixed code problems (which had code only in the question stem) and skeleton code questions (which had code in both the stem and the answer choices).

Table 1. Objective Quality Rubric

Clarity of question	0: No, had something that made it hard to understand 1: Yes, could understand what was being asked
Error free question	0: No, it contained minor errors, including grammatical errors in the stem or minor syntax errors in code presented in the question 1: Yes
Distracters feasible	0: Less than 2 distracters feasible (of correct type, and could be justified) 1: At least 2 (but not all) distracters feasible 2: All distracters feasible (note: not necessarily "a perfect set" of distracters, such as one might devise for an exam)
Explanation	0: Poor or missing 1: Good, explained why the correct answer was correct 2: Explained not only the correct answer, but included some discussion of common mistakes/misconceptions or explicit analysis of distracters
Correctness	0: Answer indicated by the author of the question was not correct 1: Answer indicated by the author of the question was correct

Analysis of the student produced questions led us to define a more detailed set of structures common to questions including: code appearing in the question stem, code appearing in the alternatives, code appearing in both the stem and alternatives, questions requiring computing or matching output of code, and questions addressing errors, such as syntax errors. We coded all n = 617 questions from PeerWise, coded the 12 Leeds study questions, and coded all multiple choice questions from all 10 formal exams and tests since 2004 from the University of Auckland's CS1 course.

3 Results

Here we present the results of the analyses described in the corresponding sections of the methodology.

3.1 Do students' ideas of quality match those of staff?

As evidenced in Section 2.1, students do value the quality ratings assigned to questions by their peers – those questions with the highest response rates were rated highly by students. Given that this is the case, the accuracy of these ratings is of some importance. To measure the "accuracy" of the student ratings, we compared them with the staff ratings of the same questions. For a meaningful comparison, we considered only those questions (42 of the 61 in our sample) that had at least 5 student ratings.

Two of the authors teach this course at the institution on a regular basis (and one taught the term these student questions were gathered). The other author regularly teaches a quite similar course, but in another educational and cultural context.

Table 2 shows the averages and standard deviations for the three authors' codings. Table 3 reports the correlation between each author's coding and the student ranks and also the pairwise correlation between the three author's codes.

Table 2. Differences in Overall Staff Quality Assessment

Staff Coder	Average Rating	Standard Deviation
Coder A	2.8	1.2
Coder B	2.1	0.9
Coder C	3.4	1.2

Table 3. Correlation between Staff Coders and Students

Students	0.58	0.22	0.52	1
C	0.52	0.47	1	
B	0.42	1		
A	1			
	A	B	C	Students

Figure 2 shows a plot of the average instructor rating and the average student rating for each of these 42 questions using the same 0-5 scale. The correlation between the average instructor and student ratings is 0.54, and a positive trend is visible in the chart.

It appears as though students not only value the ratings that have been assigned by their peers, but these ratings are also reasonably effective at assessing the overall quality of a question. When assigning our overall quality ratings, we took into account the clarity and correctness of the question, the plausibility of the distracters and the quality of the explanation. It is likely that students have taken some of these elements into account as well, and it would be interesting to know which of them the students regard as most important.

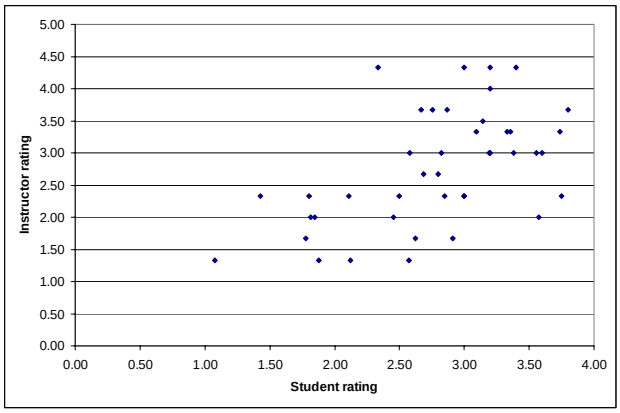

Figure 2: Comparison of the instructor and student ratings, on a 0-5 scale, for 42 questions from our examined sample with at least 5 student ratings

3.2 Objective Quality

Table 4 summarises our classifications of the 61 questions in our sample according to the objective quality rubric presented in Table 1.

We can see that almost all questions were clear and that 80% were error free in the question wording and code (not counting the cases where questions were seeking to have students find errors). Somewhat surprisingly 87% of questions gave distracters that were all feasible. In our definition of feasible, we applied a metric which meant that the answer could be possible from the code – even if it wasn't the "perfect" or even thoughtfully chosen distracter.

Table 4. Object quality classifications

Category	Option	Percentage
Clarity of question	0: No	6.6%
	1: Yes	93.4%
Error free question	0: No	19.7%
	1: Yes	80.3%
Distracters feasible	0: < 2	3.3%
	1: >= 2	9.8%
	2: All	86.9%
Explanation	0: Poor	42.6%
	1: Good	32.8%
	2: Good+	24.6%
Correctness	0: No	11.5%
	1: Yes	88.5%

Explanations were an issue of great interest to the authors – as we hoped they were for the students. We would like students to be able to learn directly from the explanation provided in PeerWise – and certainly we would like explanations not to be incorrect – which would certainly lead to frustration among the students. Unfortunately we ranked 43% of the explanations as poor – they either did not explain the answer clearly or were missing. 33% of the explanations were good in that they explained clearly why the correct answer was correct.

One of the benefits of multiple choice style questions is that students can be directly confronted with misconceptions or alternatives that may be as or more instructive than a simple comparison of the correct answer. We were pleased that 25% of the questions not only featured explanations of the correct answer but specifically discussed either the reasons the various distracters were wrong or in some way addressed a common misconception that may have affected students in answering the question.

3.3 Quality in Relation to Exams

Another objective measure of quality is to describe the structure of the question. Particularly, this analysis draws on the work of the Leeds Working Group which gave a common set of CS1 exam questions to students around the world (n = 556). Those questions fell into two basic categories. First are questions with code in the question stem – called "fixed code" questions in the Leeds study. These problems required reading and tracing of the code – in the Leeds study the answers were either of *int* type or *array* type and asked "what's the value of variable <x> after the code is executed." The Leeds study found that performance on fixed code problems was generally similar and well differentiated student performance.

The second type of question in the Leeds study was a skeleton-code type where code in the multiple choice answers was required to fill in some code in the stem to produce a program meeting certain output requirements. In the Leeds study, fixed code questions were notably easier for students (answered correctly by 68% of students) than skeleton code questions which were correctly answered by 53% of students

We wondered if students' greater ease at answering fixed code questions might translate into a greater propensity to write fixed code questions. The student created questions considered in this study show a greater variety of types than those in the Leeds study, so a strict analysis of fixed code and skeleton code questions would be inappropriate. We broke down question style into 5 more basic components (shown in Table 5), which can be combined to describe the Leeds question styles. We then applied these simple component analyses to both the PeerWise questions and the Leeds questions. It should be noted that none of the "fixed code" Leeds questions actually asked students to identify the output of the code after execution – instead they asked for the value that would be in a particular variable. The 33% (4 questions) of Leeds questions that involved output were in skeleton code questions where skeleton code was filled in to match a desired, described output.

Among student-produced questions, we see that the majority fall in the category of having only code in the stem. Almost all of these are questions that follow the form "what is the output of this code". Approximately 20% of the questions had code in the answer choices, and only 7% had code in both the questions stem and answer choices. Additionally, we note that most of the questions which had code in them involved finding or matching output in some way. In problems where there was code in the answer choices, a common question format might be to ask which of the following pieces of code outputs a certain value. In questions with code in both the stem and the answer choices, a common thing to ask is which of the following "for" loops will produce the same output as this "while" loop.

Table 5. Objective Question Quality by Style

Question Style	Leeds Study	PeerWise	AU Exam
Code in question stem	100%	82%	82%
Code in answer choices	42%	20%	33%
Code in both stem and choices	42%	7%	17%
Find or match output	33%	78%	55%
Identify error	0%	13%	13%

In addition, we can also compare the student created questions to the structure of the questions traditionally given on exams and tests of the CS1 course in our study. In total, 191 questions from ten exams and tests spanning the last four years were analysed.

The proportion of questions with code in the stem and questions which require the identification of an error appear in identical proportions in the 617 student questions and 191 formal test and exam questions. The students wrote a greater proportion of questions that involved finding or matching output, which is likely to be explained by the fact that this style of question is easier to write.

4 Discussion

The correlation between student judgement of quality and instructor judgement of quality (shown in Table 3) gives us confidence that students can distinguish between good questions and poor questions.

4.1 Question Case Studies

In this section, we present four case studies of questions that are representative of several of the kinds of questions in the repository. These case studies are drawn from the entire repository.

Good questions

To highlight the quality of the student generated content, we present here one example of a good question which received an excellent rating from students. There were 75 questions in the repository that had been rated more than 30 times, and of these, six questions had an average rating greater than 3.9. These six questions were contributed by five different students. The question we present here was answered more times than the others (108 responses), and was contributed by the student with two of the top six questions. The distracters were good, receiving a fairly even spread of responses, and the explanation was very detailed and clear.

Which of the following loops could you use to cycle through all elements of the following array without going out of bounds and causing the program to crash?

```
int[] array = new int[15];
```

Assume that in all cases, array[i] will be used without any other math operating inside the square brackets (dereferencing operator).

Each of the distracters for this question are listed below. The parentheses denote how many students selected each distracter.

A (selected by 5 students):

```
int i = 0;
while (i <= array.length) {
   i++;
   // code
}
```

B (selected by 22 students):

```
for (int i = 0; i <= array.length; i ++) {
   // code
}
```

C (the correct alternative, selected by 46 students):

```
for (int i = array.length - 1; i >= 0; i--) {
   // code
}
```

D (selected by 17 students):

```
for (int i = array.length - 1; i > 0; i--) {
   // code
}
```

E (selected by 18 students):

```
int i = 1;
while ( i < array.length) {
   i++
   // code
}
```

The explanation provided for this question not only explained the correct answer, but considered each distracter and explained why it was incorrect. The complete explanation for this question is given below:

When dealing with arrays, there are a few things to remember. When created, the value used inside the square brackets indicates the length of the array, or how many elements it can contain. The length counts from a starting point of 1. The INDEX however, begins at 0. Meaning that in this case, where we created our integer array with a length of 15, the valid index values are 0-14.

(C) is the correct answer because:
i = array.length - 1, evalutes to 14. The last index of the array. The conditional statement will go down to AND include 0, the first index of the array, but will not pass this point and go out of bounds. i-- means subtract 1 from i every time it goes around, so every number from 14 to 0 will be a value of i during the loops lifespan.

Why are the other's incorrect?

(A) This loop would crash at the end. i = 0, this is fine, it is the first value of the index and is correct. BUT The conditional inside the while loop is: i <= array.length, which means it can be less than OR equal to array.length, which is 15. The last index is 14, thus when it attempted to find index 15 of the array, it would crash with an out of bounds error.

(B) This suffers the exact same problem as A, but has been rendered in 'for' loop format.

(D) The loop shown for D would not crash, but nor would it completely cycle through all values of this array.
int i = array.length - 1 as discussed above will result in 14 which is correct for the last index of our array. However, the conditional: i > 0 will not ever allow this loop to check index 0. It will stop after cycling through 1.

(E) This loop again will not crash, but will notcycle completely through all values of this array. int i = 1 means that 0 will not be evaluated. the conditional inside the while loop will stop the cycle correctly at 14 to prevent the crash. i++ means that it will increment the index until the conditional stops this loop.

Poor (but useful) questions

As illustrated again later in Section 4.2, even questions containing errors can end up being useful learning resources thanks to the comments written by other students. We have selected an example of a poor question, but one which has actually benefitted not only other students but the author themselves. Generally, questions with poor ratings do not get a large number of responses. The primary reason for this is that students are able to sort the unanswered questions by rating, and so tend to answer questions with good ratings while avoiding questions with poor ratings. Figure 1, which plots the number of responses a question has received against the rating it has been given by students, shows this trend clearly.

The question below was answered by 14 students and rated by 10 of them, and was given an average rating of 0.4. This was the very worst rated question of all questions with at least 10 ratings.

What is the appropriate boolean variables need to be stored in A and B if the following returns false:

!A || B && !B || A

Although there are some grammatical errors in the question stem, the concept for the question is fine. However, the author had misunderstood the order of precedence of the boolean operators and the suggested alternative was not correct. This was pointed out in several of the comments to the question, two examples of which are shown below:

"Check this page. && is higher than ||.

http://java.sun.com/docs/books/tutorial/java/nutsandbolts/operators.html

so the equation becomes A + !A, which always evaluates to true. So it doesnt matter what values you put into A and B, the expression is never going to be false."

"As explained by the person above me who linked to the sun page, as that expression stands, it cannot be false. In bracket form it would look like: (!A || B) && (A || !B). Of the answers you gave, none of the above is the correct one. :P "

Despite the error in the question, there is evidence that the associated comments and link to Sun's website has transformed it into a useful learning resource. Not only has this question helped other students, but it even corrected the misunderstanding that existed with the author of the question. One student, who answered the question incorrectly, wrote:

Wow that actually helped me alot lol. Totally forgot about the order of && and ||

The question author responded to one of the comments stating:

Sorry everyone..thanx for the reply..i've posted the new version of this question. Feel free to check it out n comment on it (i've 'repaired' my understanding, i hope i got it right this time :D)

Complex questions

From an instructor perspective, good questions are often those which isolate a concept to identify where misconceptions are occurring. Such questions inform the instructor about student performance and thus are useful for summative assessment purposes. From a student perspective, a complex question that involves many concepts can be useful for determining their understanding of a diverse range of topics. This can be valuable for self-assessment, as the student will know where they had problems, and the explanation will help them to understand where they went wrong. We present below one example of such a question:

What is the output of the following code?

```
public class Peerwise {

    public void start() {
        int value = 2;
        int x, y, z;
        String output = "";
        x = 3;
        y = x*2;
        z = value++;
        if (methodOne(x, y)) {
                output += "1";
                z++;
        }
        if (methodTwo(x, y, z)) {
                output += "2";
        } else {
                output += "3";
        }
        while (output.length() > 0) {
                System.out.print(output);
                output = output.substring(1);
        }
    }
```

```
    public boolean methodOne(int y, int x) {
        return y - 3 == x;
    }

    public boolean methodTwo(int x, int y, int z) {
        x++;
        y += y;
        double d = ((y / x) + 1.0) % z;
        return d == 2.0 || d == 0.0;
    }
}
```

The distracters for this question were effective with four out of five of them receiving at least one response. The explanation was also good, and stepped through most of the lines of code. For summative assessment purposes, this question would be of little value to instructors as it would be hard to determine the source of any misunderstanding from an incorrect answer. To a student, answering correctly provides positive feedback about their understanding of a multitude of topics. With the associated explanation, if a student answers incorrectly, they should be able to identify their error. There was one comment written about this question:

That was an awesomely awesome question

It is hard to tell whether this is sarcastic or an exaggeration, but the comment was agreed with by another student and the question received a very good rating (3.75).

Good questions, improved by others

There are several examples in the repository of good questions except for the fact that they lack quality explanations. We note that even questions submitted without a good associated explanation can still be a valuable learning resource thanks to the comments left by other students. The last of our case studies highlights one such example. Consider the following question:

What is the output?

```
Point[] pts = new Point[3];
pts[0] = new Point(100, 300);
pts[1] = new Point(200, 200);
pts[2] = new Point(300, 100);

pts[1] = pts[2];
pts[2] = pts[0];
pts[0] = pts[1];

System.out.println(pts[0].x + " , " + pts[0].y);
System.out.println(pts[1].x + " , " + pts[1].y);
System.out.println(pts[2].x + " , " + pts[2].y);
```

This question was well answered, with 74% of 27 students answering correctly. The explanation given by the author of this question was:

```
pts[1] = pts[2];    pts[1] --> 300 , 100.
pts[2] = pts[0];    pts[2] --> 100 , 300.
pts[0] = pts[1];    pts[0] --> 300 , 100.
```

This does attempt to show the references from the array elements to the Point objects, but it does not discuss the details of reference variable assignment. Another student submitted the following comment as an attempt to clarify this:

The explanation is lacking. It should be made quite explicit that when you make an assignment of one object equalling another object, that the pointer value is copied exactly at that moment in time. It does NOT create a link to whatever the object on the right side of the expression happens to equal after the expression took place. i.e, If we say that:

Pts[0] points to memory address starting at AA0000

Pts[1] points to memory address starting at BBAA00

Pts[2] points to memory address starting at CCBBAA

Then when we perform the expression: Pts[0] = Pts[2]. we are saying copy the memory address in Pts[2] RIGHT NOW. So Pts[0] is now equal to CCBBAA. If we then say Pts[2] = Pts[1], then Pts[2] changes memory address to: BBAA00.

The author of the question posted a reply to this comment, thanking the contributor of the explanation for the improvement:

Thanks for the detailed explaination! Hopefully this helps the other members that are still confused after my pretty vague explaination (sorry! XD)

4.2 Error discovery and correction

As mentioned in Section 4.1, the correctness of the student generated questions is of great importance to instructors. A repository of questions in which a high proportion are not correct would be of little value and it could be argued may even be misleading to students. We randomly selected 61 questions, approximately 10% of the repository, and examined each for correctness. We found 89% (54 out of 61) of these questions to be without error and indicating the correct answer. While this is a positive result, we felt it was important to investigate in more detail the seven questions that were found to be incorrect. In particular, we were interested in the ability of the students using PeerWise to detect, and correct, these errors.

We discovered that in all seven cases the errors were discovered and commented on by students. In addition, where a correct alternative was available, in each case it was the most popular one selected by students.

While our sample of 61 questions represents only 10% of the repository, the results we have seen give us confidence in the students' ability to detect and discuss errors in the small percentage of questions that contain them.

The average rating given to these seven questions by the students was 1.7, considerably lower than the average rating in the repository. The low ratings assigned to these questions reduces their prominence in the repository, as

we have previously noted that questions with low ratings are answered less often.

We include here one of the seven questions that contained an error and the comment from a student that identified the error and offered a correct explanation.

This question was on method tracing, and asked for the output of the following code:

```
public void start () {
   int number = 3;
   calculation(number);
   System.out.println ("1: " + number);
}

private void calculation (int number) {
   number = number + 5;
   System.out.println ("2: " + number);
}
```

A set of plausible distracters was offered:

A	B	C	D
1: 8	*2: 8*	*1: 3*	*2: 8*
2: 8	*1: 8*	*2: 8*	*1: 3*

The question author's error was in selecting C rather than D as the answer. The most popular alternative was in fact the correct response D, which was chosen by 42% of the 12 who answered this question. There was also an excellent comment, which was agreed with by another student, pointing out the error and clarifying with an example:

Sorry, but you are incorrect in the order. While your numbers are correct, 2 will print before 1. The calculate method will run in it's entirety (because there are no early return points) BEFORE the rest of the calling point's code is execute. If you were to write that block of code inline, it would look like this:

```
public void start () {
   int number = 3;
   // replacing the calculate method call:
   System.out.println ("2: " + (number + 5));
   System.out.println ("1: " + number);
}
```

4.3 Question styles

The structure of the questions written by students mirrored the questions used by academic staff in their local context (i.e. in the tests and exams of the institution). Furthermore, the subjective quality ratings assigned by the students correlate well with the overall quality rating of the instructors involved in teaching the course.

However, other teaching contexts are different. The Leeds study used questions with a different characteristic structure. The correlation between the staff rating of quality and student rating of quality was lowest when the staff member was not involved in teaching the course. It appears that students adapt to the context in which the course is taught and create questions that have similar structure and focus on similar topics to questions that they have seen during class and in tests and exams.

We expect that if PeerWise was used in a different teaching context, then students in that environment would mirror the structure and focus emphasised by the teaching staff in that local context. To verify this prediction, we could replicate this study at a different institution with different teaching staff. If we don't see similar trends, then it might show that students are having difficultly understanding what the instructor wants them to learn, or alternatively, it might be related to the construction of MCQs themselves (i.e. it might be easier to create MCQs about particular topics, and it might be easier to create questions that have a particular structure such as asking "what is the output of the following code?").

Further investigation of the question style and quality in different teaching contexts is required to better understand how students decide to structure their questions.

5 Conclusions

Students are capable of writing questions that faculty judge to be of high quality. The best questions have well written question stems, good distracters and detailed explanations that discuss possible misconceptions.

Although the questions created by students do vary widely in quality, students are quite capable of making accurate judgements of quality and rate the questions appropriately. Student judgements of quality correlate well with faculty when overall subjective judgement is required. Since students use the rating to determine which questions they answer, we are confident that students view the high quality questions more frequently than the low quality questions.

Inspecting a sample of the questions more closely revealed that most of the questions were clear and unambiguous, free from grammatical or other minor errors, had a high number of feasible distracters and had a good explanation.

The structure of the questions written by students is generally similar to that of the questions written by faculty teaching the course (i.e. for examination purposes in mid-semester tests and final exams), suggesting that students are sensitive to the local teaching context and will create questions appropriate for the style of the questions presented in the course.

The majority of questions have correct solutions (i.e. the answer suggested by the author is the correct answer), and in the cases where the solutions are incorrect, other students are able to identify the errors and offer a correct solution or explanation. This suggests that most learning experiences with PeerWise should be error free for students.

6 Future work

We plan to further investigate the issue of question quality by replicating the study at a different institution where the instructor has a different teaching style and focus.

No instruction has been given to students with respect to the creation of MCQs. It would be interesting to investigate whether formal instruction about creating questions, choosing appropriate distracters and the importance of an explanation has an affect on the quality and structure of the questions. Altering the assessment criteria for the contributed questions to explicitly require an explanation might reduce the number of questions that lacked explanations entirely and may generally improve the quality of the explanations produced.

7 References

[1] P. Denny, A. Luxton-Reilly, and J. Hamer. The PeerWise system of student contributed assessment questions. In Simon and M. Hamilton, editors, Tenth Australasian Computing Education Conference (ACE 2008), volume 78 of CRPIT, pages 69-74,Wollongong, NSW, Australia, 2008. ACS.

[2] P. Denny, A. Luxton-Reilly and J. Hamer. Student use of the PeerWise system. In ITiCSE '08: Proceedings of the 13th annual conference on Innovation and technology in computer science education, pages 73-77, Madrid, Spain, July 2008. ACM.

[3] P. Denny, J. Hamer, A. Luxton-Reilly, and H. Purchase. Peerwise: students sharing their multiple choice questions. In ICER'08: Proceedings of the 2008 international workshop on Computing education research, pages 51-58, Sydney, Australia, 2008.

[4] R. Lister, E. S. Adams, S. Fitzgerald, W. Fone, J. Hamer, M. Lindholm, R. McCartney, J. E. Mostrom, K. Sanders, O. Seppala, B. Simon and L. Thomas. A multi-national study of reading and tracing skills in novice programmers. SIGCSE Bulletin, Volume 36, Issue 4 (December 2004), pp. 119 - 150. ACM.

[5] J. Sitthiworachart and M. Joy. Computer support of effective peer assessment in an undergraduate programming class. Journal of Computer Assisted Learning (2008), 24, 217–231

[6] K. Topping. Peer assessment between students in colleges and universities. Review of Education Research (1998), 68 (3), 249-276