Eliph: Effective Visualization of Code History for Peer Assessment in Programming Education

Jungkook Park (School of Computing, KAIST)
Yeong Hoon Park (School of Computing, KAIST)
Suin Kim (School of Computing, KAIST)
Alice Oh (School of Computing, KAIST)
Rapid Growth of Online CS Courses

Numbers of students and TAs in online CS courses

- Red line: # Students
- Blue line: # TAs

Year
Rapid Growth of Online CS Courses

Numbers of students and TAs in online CS courses

Class Central • 전체 공개

[Coursera] Human-Computer Interaction class starts May 28. Will be the first class to use the new Peer Assessment system

OPEN edX

Peer Grading Overview | June 13, 2014

# Students # TAs
Difficulties of Peer Assessment in CS

Peer assessment involves both

“understanding other’s work” and “giving a proper mark”

Grading Rubric

<table>
<thead>
<tr>
<th>Program Design</th>
<th>Efficiency</th>
<th>Readability</th>
<th>Assignment Specifications</th>
</tr>
</thead>
</table>

Submitted Programming Code
Difficulties of Peer Assessment in CS

Peer assessment involves both

“understanding other’s work” and “giving a proper mark”

Submitted Programming Code

I’ve never seen this syntax.

Why this code works?

This code is too hard to read!

Why did he/she implement in this way?

Grading Rubric

Program Design

Efficiency

Readability

Design/Implementation Specifications
Difficulties of Understanding Other’s Code

Even for skilled programmers, it is difficult to infer the intentions of the code author by merely reading the code.

What is the most difficult question if you are supposed to answer by reading other’s code?
Common Practice in Open Source Community

Split a submission into several small pieces (a.k.a. commit) and provide line-by-line differences between commits.
Proposed Approach

Skeleton Code -> Final Submitted Code
Proposed Approach

Skeleton Code

Implement A()

Implement B()

Fix error in C()

Refactor D()

Final Submitted Code

1. insert " def A(): ">
2. insert " for i in range(2) ">
3. insert " sum += k "

1. See the compiler message
   " C : undefined variable k "
2. insert " k = 0 "

1. insert " def A(): ">
2. insert " for i in range(2) ">
3. insert " sum += k "
Eliph
A web-based peer assessment system for CS education with code history visualization
Eliph - Peer Assessment

Hello, Guest Park! StudentID: 20990114, Problem: Homework 2.1 - Merge Blocks

Code Review  Assessment

cs206c/Main.java

```java
for (Block input : sortedBlocks) {
    if (mergedBlocks.size() == 0) {
        mergedBlocks.add(input);
        continue;
    }
    boolean isModified = false;
    for (int i = 0; i < mergedBlocks.size(); i++) {
        Block temp = mergedBlocks.get(i);
        if (input.start <= temp.start && input.end >= temp.end) {
            temp.start = input.start;
            temp.end = input.end;
            isModified = true;
        } else if (input.start <= temp.start && input.end <= temp.end) {
            temp.start = input.start;
            isModified = true;
        }
    }
    for (Block input : blocks) {
        if (input.start >= temp.start && input.end <= temp.end) {
            temp.start = input.start;
            isModified = true;
        }
    }
    if (isModified) {
        mergedBlocks.add(temp);
    }
}
```

Execution events

```java
public static List<Block> mergeBlocks(List<Block> blocks) {
    List<Block> mergedBlocks = new ArrayList<Block>();
    List<Block> sortedBlocks = new ArrayList<Block>();

    int inputLength = blocks.size();
    for (int i = 0; i < inputLength; i++) {
        Block block = blocks.get(i);
        int minIndex = 0;
        for (Block b1 : blocks) {
            if (b1.start > block.start) {
                min = b1;
                minIndex = i;
            }
        }
        blocks.remove(minIndex);
        sortedBlocks.add(min);
    }
    for (Block input : blocks) {
        for (Block temp : mergedBlocks) {
            if (input.start <= temp.start && input.end >= temp.end) {
                temp.start = input.start;
                temp.end = input.end;
                isModified = true;
            } else if (input.start >= temp.start && input.end <= temp.end) {
                temp.start = input.start;
                isModified = true;
            }
        }
    }
    return sortedBlocks;
}
```
Character-Level Code History

```java
1 // DO NOT MODIFY THE FUNCTION DECLARATION
2 public static List<Block> mergeBlocks(List<Block> blocks) {
3    // Implement here
4
5    return mergedBlocks;
6 }
```
Selection-Based History Tracking

```java
// Problem 1
// DO NOT MODIFY THE FUNCTION DECLARATION
public static List<Block> readBlocks() {
    // Implement here
    List<Block> blocks = new ArrayList<Block>();
    Scanner input = new Scanner(System.in);
    while(input.hasNext()){
        int start = input.nextInt();
        int end = input.nextInt();
        Block temp = new Block(start, end);
        blocks.add(temp);
    }
    return blocks;
}

// Problem 2
// DO NOT MODIFY THE FUNCTION DECLARATION
public static List<Block> mergeBlocks(List<Block> blocks) {
    // Implement here
    List<Block> mergedBlocks = blocks;
    int size = blocks.size();
    for (int i=0; i<size; i++){
        for(int j = 0; j<size; j++){
            if (i!=j&&i<size&&i<size){
                if (blockIntersect(mergedBlocks.get(j),mergedBlocks.get(i)))
                    Block temp = new Block(Math.min(mergedBlocks.get(i).getStart(),mergedBlocks.get(j).getStart()),
                                            Math.max(mergedBlocks.get(i).getEnd(),mergedBlocks.get(j).getEnd()));
                mergedBlocks.set(i,temp);
                mergedBlocks.remove(j);
            }
        }
    }
    return mergedBlocks;
}
```
Execution Events

executions

Location of code changes  Execution with no error  Execution with error
Execution Events

- Testcase #1: correct
- Testcase #2: correct
- Testcase #3: wrong

Score: 80

Exception in thread "main"
java.lang.ArrayIndexOutOfBoundsException

- Testcase #1: correct
- Testcase #2: correct
- Testcase #3: correct

Score: 100

Location of code changes
Implement
Fix bugs
Refactor

Execution with no error
Execution with error
Evaluation
in a real classroom environment
Hypotheses

Visualization of code history

**H1** - promotes higher quality of peer feedback

**H2** - helps *student to get positive learning outcomes

**H3** - improves the **reliability** of peer assessment

*student : assessor + code author

**reliability : the variance of scores given by peers
Step 1. Feedback Generation

58 students

assess with code history →
assess without code history
Step 2. Feedback Evaluation

- Feedback Evaluation 1
- Feedback Evaluation 2
- Feedback Evaluation 3
- Feedback Evaluation 4

Feedback

code author
Analysis
hybrid method of quantitative and qualitative
H1: Eliph Promotes Higher Quality of Peer Feedback

Post-feedback Survey from Step 1

Strongly Disagree  Strongly Agree

- Understand how the code works
  - 18.97% 39.66%

- Understand the code quickly
  - 24.14% 36.21%

- Understand author’s intention
  - 13.79% 68.97%

- Assess the code
  - 22.41% 39.66%

- Provide feedback for the code
  - 18.97% 39.66%

n=58, 5-point Likert scale
H1: Eliph Promotes Higher Quality of Peer Feedback

Feedback Evaluation Result from Step 2

- Peer’s understanding
- Improving readability
- Improving efficiency
- Fairness and unbiasedness
- Satisfaction on overall quality

<table>
<thead>
<tr>
<th></th>
<th>w/ code history</th>
<th>w/o code history</th>
<th>n_w=36, n_w/o=42, 5-point Likert scale</th>
</tr>
</thead>
<tbody>
<tr>
<td>Peer’s understanding</td>
<td>4.0</td>
<td>3.6</td>
<td></td>
</tr>
<tr>
<td>Improving readability</td>
<td>3.8</td>
<td>3.2</td>
<td></td>
</tr>
<tr>
<td>Improving efficiency</td>
<td>3.8</td>
<td>3.4</td>
<td></td>
</tr>
<tr>
<td>Fairness and unbiasedness</td>
<td>3.8</td>
<td>3.4</td>
<td></td>
</tr>
<tr>
<td>Satisfaction on overall quality</td>
<td>3.8</td>
<td>3.4</td>
<td></td>
</tr>
</tbody>
</table>
H1: Eliph Promotes Higher Quality of Peer Feedback

"How did browsing the code history help you assess?"
H1: Eliph Promotes Higher Quality of Peer Feedback

"How did browsing the code history help you assess?"

By inferring the intention of the code author

"It allowed me to understand ... why they implemented some of the functions." (Student 13)
H1: Eliph Promotes Higher Quality of Peer Feedback

"How did browsing the code history help you assess?"

By following the thought process of the code author

"... was helpful in understanding the author’s flow of thought" (Student 23)
H1: Eliph Promotes Higher Quality of Peer Feedback

"How did browsing the code history help you assess?"

By seeing the trial-and-error of the code author

"... I was able to understand where the author had been mistaken." (Student 4)
H1: Eliph Promotes Higher Quality of Peer Feedback

"How did browsing the code history help you assess?"

By understanding the code more easily

"In cases of code with poor readability, I had to browse its code history…" (Student 58)

"… I didn’t have to understand the entire code at once." (Student 57)
H2: Eliph Helps Students Get Learning Outcome

Post-feedback Survey from Step 1

Strongly Disagree

Learn how to write correct code
- 22.41%
- 41.38%

Learn how to write readable code
- 22.41%
- 36.21%

Learn how to write efficient code
- 24.14%
- 29.31%

Strongly Agree

Number of participants: n=58, 5-point Likert scale
"How did browsing the code history help you learn?"
"How did browsing the code history help you learn?"

By seeing how to write a readable code

"I learned some techniques such as naming variables, ..., splitting code into small pieces, which could prevent potential problems as the code gets bigger" (Student 14)
"How did browsing the code history help you learn?"

By seeing similar ways of coding

"I realized that people write code using steps in different order. I learned more from code written by someone who codes more like myself." (Student 48)
H2: Eliph Helps Students Get Learning Outcome

"How did browsing the code history help you learn?"

By seeing how to overcome errors in specific situations

"... watching the trials and errors gave me insights into particular cases where some approaches simply don’t work." (Student 33)
H2: Eliph Helps Students Get Learning Outcome

"If browsing the code history did not help you learn, why?"

"If a well-written code is given, I could see the process of writing good code by looking only at the final version of the code" (Student 51)

"... it contains wrong or inefficient code." (Student 44)
H3: Eliph Does Not Improve Reliability of Assessment

Code Assessment Result from Step 1

Program Design

Efficiency

Readability

Assignment Specifications

0.0  2.0  4.0  6.0  8.0  10.0  12.0  14.0  16.0  18.0  20.0

w/ code history  w/o code history

n_{w}=43, n_{w/o}=47
Conclusion

• We have introduced Eliph, a web-based peer assessment system with code history visualization.

• We have showed that Eliph has multiple benefits,

  • Looking at the code history helps student assessor understand the code structure as well as the author's intention more clearly.

  • Overall quality of feedback is higher when evaluated with the code history.

  • Evaluators feel that looking at the code history is helpful for their own learning.
Eliph: Effective Visualization of Code History for Peer Assessment in Programming Education

Jungkook Park (School of Computing, KAIST)
Yeong Hoon Park (School of Computing, KAIST)
Suin Kim (School of Computing, KAIST)
Alice Oh (School of Computing, KAIST)

pjknkda@kaist.ac.kr
## Undergraduate CS course (Data Structure)

### Problem 2

<table>
<thead>
<tr>
<th>Code 1</th>
<th>Code 2</th>
<th>Code 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code 4</td>
<td>Code 5</td>
<td>Code 6</td>
</tr>
<tr>
<td>Code 7</td>
<td>Code 8</td>
<td>Code 9</td>
</tr>
</tbody>
</table>

### Problem 3

<table>
<thead>
<tr>
<th>Code 1</th>
<th>Code 2</th>
<th>Code 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code 4</td>
<td>Code 5</td>
<td>Code 6</td>
</tr>
<tr>
<td>Code 7</td>
<td>Code 8</td>
<td>Code 9</td>
</tr>
</tbody>
</table>

### Problem 1

<table>
<thead>
<tr>
<th>Code 1</th>
<th>Code 2</th>
<th>Code 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code 4</td>
<td>Code 5</td>
<td>Code 6</td>
</tr>
<tr>
<td>Code 7</td>
<td>Code 8</td>
<td>Code 9</td>
</tr>
</tbody>
</table>

### Step 1. Feedback Generation

#### Code Assessment (Feedback) Criteria

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Points</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>A. Program Specification / Correctness (30pt)</td>
<td></td>
<td>Auto-graded</td>
</tr>
<tr>
<td>B. Program Design (20pt) + Comments</td>
<td></td>
<td>Excellent (100%), Adequate (80%), Poor (60%), Not Met (0%)</td>
</tr>
<tr>
<td>C. Code Efficiency (20pt) + Comments</td>
<td></td>
<td>Excellent (100%), Adequate (80%), Poor (60%), Not Met (0%)</td>
</tr>
<tr>
<td>D. Readability (15pt) + Comments</td>
<td></td>
<td>Excellent (100%), Adequate (80%), Poor (60%), Not Met (0%)</td>
</tr>
<tr>
<td>E. Assignment Specification (15pt) + Comments</td>
<td></td>
<td>Excellent (100%), Adequate (80%), Poor (60%), Not Met (0%)</td>
</tr>
</tbody>
</table>
Step 1. Feedback Generation

### Section A. Peer Assessment

**Q1 ~ Q5 (5-point Likert scale):**

{ To understand how code works, To understand the code quickly, …} + *browsing the code history was helpful than viewing the last version of the code.*

How did *browsing the code history* help you assess the code? If it did not, why?

### Section B. Learning with Assessment

**Q6 ~ Q8 (5-point Likert scale):**

{ To learn how to write correct code, To learn how to write readable code, …} + *browsing the code history was helpful than viewing the last version of the code.*

How did *browsing the code history* help you learn to write a good the code? If it did not, why?
Step 2. Feedback Evaluation

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Evaluation 1</th>
<th>Evaluation 2</th>
<th>Evaluation 16</th>
<th>Evaluation 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code 1</td>
<td>17</td>
<td>18</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>Code 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Problem 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(5-point Likert scale)

**R1.** The peer clearly understood my code.

**R2.** The feedback will help me to improve the style or readability of my future code.

**R3.** The feedback will help me to improve the efficiency or to use a better algorithm for my future code.

**R4.** I feel the feedback is fair and unbiased.

**R5.** I am satisfied with the overall quality of the feedback.
**Quantitative Findings : Post-Evaluation Survey**

Browsing the code history was **more helpful than** viewing the last version of the code to

<table>
<thead>
<tr>
<th>Question</th>
<th>Pos.(%)</th>
<th>Neg.(%)</th>
<th>Mean (SD)</th>
</tr>
</thead>
<tbody>
<tr>
<td>understand author's intention of the code</td>
<td>68.97</td>
<td>13.79</td>
<td>3.86 (1.06)</td>
</tr>
<tr>
<td>learn how to write correct code</td>
<td>41.38</td>
<td>22.41</td>
<td>3.22 (0.89)</td>
</tr>
<tr>
<td>understand how the code works</td>
<td>39.66</td>
<td>18.97</td>
<td>3.19 (0.96)</td>
</tr>
<tr>
<td>provide feedback for the code</td>
<td>39.66</td>
<td>18.97</td>
<td>3.19 (0.97)</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>learn how to write efficient code</td>
<td>29.31</td>
<td>24.14</td>
<td>3.09 (0.92)</td>
</tr>
</tbody>
</table>

n=58, 5-point Likert scale
Quantitative Findings : Post-Evaluation Survey

Browsing the code history was more helpful than viewing the last version of the code to understand the code.

<table>
<thead>
<tr>
<th>Question</th>
<th>Pos.(%)</th>
<th>Neg.(%)</th>
<th>Mean (SD)</th>
</tr>
</thead>
<tbody>
<tr>
<td>understand author's intention of the code</td>
<td>68.97</td>
<td>13.79</td>
<td>3.86 (1.06)</td>
</tr>
<tr>
<td>learn how to write correct code</td>
<td>41.38</td>
<td>22.41</td>
<td>3.22 (0.89)</td>
</tr>
<tr>
<td>understand how the code works</td>
<td>39.66</td>
<td>18.97</td>
<td>3.19 (0.96)</td>
</tr>
<tr>
<td>provide feedback for the code</td>
<td>39.66</td>
<td>18.97</td>
<td>3.19 (0.97)</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>learn how to write efficient code</td>
<td>29.31</td>
<td>24.14</td>
<td>3.09 (0.92)</td>
</tr>
</tbody>
</table>

n=58, 5-point Likert scale

More positive response than negative for all questions

support H1(Quality), H2(Learning)
### Quantitative Findings: Feedback Evaluation

<table>
<thead>
<tr>
<th>Criterion</th>
<th>Exp. Group</th>
<th>Control Group</th>
<th>P-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Peer's understanding</td>
<td>3.97</td>
<td>3.79</td>
<td>0.33</td>
</tr>
<tr>
<td>Help to improving readability</td>
<td>3.72</td>
<td>3.24</td>
<td>0.04</td>
</tr>
<tr>
<td>Help to improving efficiency</td>
<td>3.72</td>
<td>3.21</td>
<td>0.05</td>
</tr>
<tr>
<td>Fairness and unbiasedness</td>
<td>3.81</td>
<td>3.55</td>
<td>0.31</td>
</tr>
<tr>
<td>Satisfaction on overall quality</td>
<td>3.89</td>
<td>3.38</td>
<td>0.04</td>
</tr>
</tbody>
</table>

$n_{exp}=36$, $n_{control}=42$, 5-point Likert scale
Quantitative Findings : Feedback Evaluation

<table>
<thead>
<tr>
<th>Criterion</th>
<th>Exp. Group</th>
<th>Control Group</th>
<th>P-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Peer's understanding</td>
<td>3.97</td>
<td>3.79</td>
<td>0.33</td>
</tr>
<tr>
<td>Help to improving readability</td>
<td>3.72</td>
<td>3.24</td>
<td>0.04</td>
</tr>
<tr>
<td>Help to improving efficiency</td>
<td>3.72</td>
<td>3.21</td>
<td>0.05</td>
</tr>
<tr>
<td>Fairness and unbiasness</td>
<td>3.81</td>
<td>3.55</td>
<td>0.31</td>
</tr>
<tr>
<td>Satisfaction on overall quality</td>
<td>3.89</td>
<td>3.38</td>
<td>0.04</td>
</tr>
</tbody>
</table>

**Significant effect toward improving "style" and "efficiency" of the code**

$n_{exp}=36$, $n_{control}=42$, 5-point Likert scale
# Quantitative Findings: Feedback Evaluation

<table>
<thead>
<tr>
<th>Criterion</th>
<th>Exp. Group</th>
<th>Control Group</th>
<th>P-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Peer's understanding</td>
<td>3.97</td>
<td>3.79</td>
<td>0.33</td>
</tr>
<tr>
<td>Help to improving readability</td>
<td>3.72</td>
<td>3.24</td>
<td>0.04</td>
</tr>
<tr>
<td>Help to improving efficiency</td>
<td>3.72</td>
<td>3.21</td>
<td>0.05</td>
</tr>
<tr>
<td>Satisfaction on overall quality</td>
<td>3.89</td>
<td>3.38</td>
<td>0.04</td>
</tr>
</tbody>
</table>

Significant effect toward the satisfaction on the quality of feedback

$n_{exp}=36$, $n_{control}=42$, 5-point Likert scale
## Quantitative Findings: Assessment Statistics

<table>
<thead>
<tr>
<th>Assessment Criterion</th>
<th>Avg. Score</th>
<th>P-value</th>
<th>T-Test</th>
<th>Levene-Test</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Exp. Group</td>
<td>Control Group</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Program Design</td>
<td>18.42</td>
<td>17.87</td>
<td>0.282</td>
<td>0.286</td>
</tr>
<tr>
<td>Efficiency</td>
<td>16.37</td>
<td>16.68</td>
<td>0.667</td>
<td>0.539</td>
</tr>
<tr>
<td>Readability</td>
<td>13.33</td>
<td>12.70</td>
<td>0.234</td>
<td>0.494</td>
</tr>
<tr>
<td>Assignment Specifications</td>
<td>13.81</td>
<td>14.36</td>
<td>0.309</td>
<td>0.298</td>
</tr>
<tr>
<td><strong>Σ</strong></td>
<td><strong>61.93</strong></td>
<td><strong>61.62</strong></td>
<td><strong>0.846</strong></td>
<td><strong>0.710</strong></td>
</tr>
</tbody>
</table>

\[ n_{\text{exp}}=43, \quad n_{\text{control}}=47 \]
## Quantitative Findings: Assessment Statistics

<table>
<thead>
<tr>
<th>Assessment Criterion</th>
<th>Avg. Score</th>
<th></th>
<th></th>
<th>T-Test</th>
<th>Levene-Test</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Exp. Group</td>
<td>Control Group</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Program Design</td>
<td>18.42</td>
<td>17.87</td>
<td></td>
<td>0.282</td>
<td>0.286</td>
</tr>
<tr>
<td>Efficiency</td>
<td>16.37</td>
<td>16.68</td>
<td></td>
<td>0.667</td>
<td>0.539</td>
</tr>
<tr>
<td>Readability</td>
<td>13.33</td>
<td>12.70</td>
<td></td>
<td>0.234</td>
<td>0.494</td>
</tr>
<tr>
<td>Assignment Specifications</td>
<td>13.81</td>
<td>14.36</td>
<td></td>
<td>0.309</td>
<td>0.298</td>
</tr>
<tr>
<td>Σ</td>
<td>61.93</td>
<td>61.62</td>
<td></td>
<td>0.846</td>
<td>0.710</td>
</tr>
</tbody>
</table>

No significant difference in both mean and variance

reject H3 (Reliability)

\( n_{\text{exp}} = 43, \ n_{\text{control}} = 47 \)
"How did browsing the code history help you assess?"

**Intention**

*It allowed me to understand ... why he implemented some of the functions.*

**Though process**

*it was helpful in understanding the author's flow of thought.*

**Trial-and-error**

*... I was able to understand where the author was mistaken.*

**Code readability**

*... I didn't have to understand the entire code at once, ...*
"If it did not (help you assess), why?"

Since it wasn't a big project, I couldn't get much extra information out of the code history.

It did not help too much because the code was easy to understand.

... I think code history is something that should be hidden. ...
"How did browsing the code history help you learn?"

Writing readable code
I learned some techniques such as naming variables, splitting code into small pieces, ...

Different code styles
... I feel like I came to realize the right way how one should write code.

Trial-and-error
... watching the trials and errors gave me insight into particular cases some approach doesn't work.
Qualitative Findings : Post-Evaluation Survey

"If it did not (help you learn), why?"

Not much to learn

If a well-written code is given, I could know the process and how to write code only seeing the final version of the code.

Poorly written code

Unless peer's code is perfect, seeing that code history does not seem to have learned something.
Quality of Peer Feedback: Quantitative Analysis

H1. Does Eliph promote higher quality of peer feedback?

YES!

Step 1: Post-feedback Survey

<table>
<thead>
<tr>
<th></th>
<th>Pos.(%)</th>
<th>Neg.(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1</td>
<td>39.66</td>
<td>18.97</td>
</tr>
<tr>
<td>Q2</td>
<td>36.21</td>
<td>24.14</td>
</tr>
<tr>
<td>Q3</td>
<td>68.97</td>
<td>13.79</td>
</tr>
<tr>
<td>Q4</td>
<td>39.66</td>
<td>22.41</td>
</tr>
<tr>
<td>Q5</td>
<td>39.66</td>
<td>18.97</td>
</tr>
</tbody>
</table>

n=58, 5-point Likert scale

Step 2: Feedback Evaluation Result

<table>
<thead>
<tr>
<th>Criterion</th>
<th>w/ code history</th>
<th>w/o code history</th>
</tr>
</thead>
<tbody>
<tr>
<td>Peer's understanding</td>
<td>3.97</td>
<td>3.79</td>
</tr>
<tr>
<td>Help to improving readability</td>
<td>*3.72</td>
<td>3.24</td>
</tr>
<tr>
<td>Help to improving efficiency</td>
<td>†3.72</td>
<td>3.21</td>
</tr>
<tr>
<td>Fairness and unbiasness</td>
<td>3.81</td>
<td>3.55</td>
</tr>
<tr>
<td>Satisfaction on overall quality</td>
<td>*3.89</td>
<td>3.38</td>
</tr>
</tbody>
</table>

n_w=36, n_w/o=42, 5-point Likert scale

*Q1 - Q5 : To do ..., w/ code history was helpful than w/o code history.
Learning Outcome: Quantitative Analysis

**H2.** Does Eliph help student to get positive learning outcomes?

**YES!**

Step 1: Post-feedback Survey

<table>
<thead>
<tr>
<th></th>
<th>Pos. (%)</th>
<th>Neg. (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q6</td>
<td>41.38</td>
<td>22.41</td>
</tr>
<tr>
<td>Q7</td>
<td>36.21</td>
<td>22.41</td>
</tr>
<tr>
<td>Q8</td>
<td>29.31</td>
<td>24.14</td>
</tr>
</tbody>
</table>

n=58, 5-point Likert scale

*Q6 - Q8: To learn how to …, w/ code history was helpful than w/o code history.*
Reliability of Peer Assessment: Quantitative Analysis

**H3.** Does Eliph improve the reliability of peer assessment?

**No.**

### Step 1: Code Assessment Result

<table>
<thead>
<tr>
<th>Assessment Criterion</th>
<th>Avg. Score (SD)</th>
<th>P-value (Levene-Test)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>w/ code history</td>
<td>w/o code history</td>
</tr>
<tr>
<td>Program Design (20pt)</td>
<td>18.42 (2.14)</td>
<td>17.87 (2.59)</td>
</tr>
<tr>
<td>Efficiency (20pt)</td>
<td>16.37 (2.97)</td>
<td>16.68 (3.72)</td>
</tr>
<tr>
<td>Readability (15pt)</td>
<td>13.33 (2.08)</td>
<td>12.70 (2.78)</td>
</tr>
<tr>
<td>Assignment Specifications (15pt)</td>
<td>13.81 (2.97)</td>
<td>14.36 (1.85)</td>
</tr>
<tr>
<td><strong>Σ</strong></td>
<td>61.93 (7.45)</td>
<td>61.62 (7.64)</td>
</tr>
</tbody>
</table>

No significant difference in the variance

\[n_{w/}=43, \; n_{w/o}=47\]