In Support of Peer Code Review and Inspection in an Undergraduate Software Engineering Course: A Case Study

Sai Krishna\(^1\)  
Raghu Reddy\(^1\)  
Ashish Sureka\(^2\)

\(^1\)Software Engineering Research Center (SERC)  
IIIT Hyderabad (India)

\(^2\)Software Analytics Research Lab (SARL)  
IIIT-Delhi (India)

18/May/2015
Table of Contents

1 Introduction
   ■ Motivation
   ■ Benefits of code review process

2 Research Questions

3 Related Work

4 Study Design

5 Static code analysis tools and reports

6 Threats to validity and Conclusion
   ■ Threats to validity
   ■ Conclusion
Introduction

- Peer code review and inspection is a quality improvement software engineering activity consisting of systematic examination of source code.

- We conduct a study on the use of peer code review in a sophomore level introductory Software Engineering course consisting of more than 200 students and present our experiences, findings and challenges.
Introduction

- Peer code review and inspection is a quality improvement software engineering activity consisting of systematic examination of source code.

- We conduct a study on the use of peer code review in a sophomore level introductory Software Engineering course consisting of more than 200 students and present our experiences, findings and challenges.
Benefits of code review process

In academic setup

- Peer reviews teach students about constructive criticism and consensus building.
- Peer reviews provide opportunities for deep learning and critical thinking.

In industry

The main motivation within industry is to find bugs in early stages. Also as a by product this process also helps in knowledge transfer, increased team awareness, and creation of alternative solutions to problems.
Benefits of code review process

In academic setup

- Peer reviews teach students about constructive criticism and consensus building.
- Peer reviews provide opportunities for deep learning and critical thinking.

In Industry

- The main motivation within industry is to find bugs in early stages.
- Also as a by product this process also helps in knowledge transfer, increased team awareness, and creation of alternative solutions to problems.
Benefits of code review process

In academic setup

- Peer reviews teach students about constructive criticism and consensus building.
- Peer reviews provide opportunities for deep learning and critical thinking.

In Industry

- The main motivation within industry is to find bugs in early stages.
- Also as a by product this process also helps in knowledge transfer, increased team awareness, and creation of alternative solutions to problems.
Contributions:

- Addresses the question of peer code review process improving compliance to coding standards and bringing consistency among the code written by students.
- Relates the outcome of peer code review process to early identification of bugs/defects.
Contributions:

- Addresses the question of peer code review process improving compliance to coding standards and bringing consistency among the code written by students.
- Relates the outcome of peer code review process to early identification of bugs/defects.
- Investigates the importance of peer code review process in knowledge sharing, project cognition, increased team cohesion and team communication.
Contributions:

- Addresses the question of peer code review process improving compliance to coding standards and bringing consistency among the code written by students.
- Relates the outcome of peer code review process to early identification of bugs/defects.
- Investigates the importance of peer code review process in knowledge sharing, project cognition, increased team cohesion and team communication.
Related Work:

- A study within Microsoft revealed that while finding defects remains the main motivation for review, it provided additional benefits such as knowledge transfer, increased team awareness, and creation of alternative solutions to problems.

- Empirical studies on integrating formal code reviews within the course curriculum of software engineering course offered at the University of Mary Washington in 2005 have shown two broad benefits:
  - Peer reviews teach students about constructive criticism and consensus building.
  - Peer reviews provide opportunities for deep learning and critical thinking.

However, previous studies were based on classroom projects and may or may not be applicable to live projects.
Related Work:

- A study within Microsoft revealed that while finding defects remains the main motivation for review, it provided additional benefits such as knowledge transfer, increased team awareness, and creation of alternative solutions to problems.

- Empirical studies on integrating formal code reviews with in the course curriculum of software engineering course offered at the University of Mary Washington in 2005 have shown two broad benefits:
  - Peer reviews teach students about constructive criticism and consensus building.
  - Peer reviews provide opportunities for deep learning and critical thinking.
Related Work:

- A study within Microsoft revealed that while finding defects remains the main motivation for review, it provided additional benefits such as knowledge transfer, increased team awareness, and creation of alternative solutions to problems.

- Empirical studies on integrating formal code reviews with in the course curriculum of software engineering course offered at the University of Mary Washington in 2005 have shown two broad benefits:
  - Peer reviews teach students about constructive criticism and consensus building.
  - Peer reviews provide opportunities for deep learning and critical thinking.

- However, previous studies were based on class room projects and may or may not be applicable to live projects.
Related Work:

- A study within Microsoft revealed that while finding defects remains the main motivation for review, it provided additional benefits such as knowledge transfer, increased team awareness, and creation of alternative solutions to problems.

- Empirical studies on integrating formal code reviews with the course curriculum of software engineering course offered at the University of Mary Washington in 2005 have shown two broad benefits:
  - Peer reviews teach students about constructive criticism and consensus building.
  - Peer reviews provide opportunities for deep learning and critical thinking.

- However, previous studies were based on classroom projects and may or may not be applicable to live projects.
This Study is done in 3 phases: Phase I

- All the students were given a two week assignment wherein they had to develop a pacman game in Python.
- Students are to inherently use simple object oriented programming principles like modularity, abstraction and polymorphism.
This Study is done in 3 phases: Phase I

- All the students were given a two week assignment wherein they had to develop a pacman game in Python.
- Students are to inherently use simple object oriented programming principles like modularity, abstraction and polymorphism.
- At this juncture students were still not familiar with any coding standards.
This Study is done in 3 phases: Phase I

- All the students were given a two week assignment wherein they had to develop a pacman game in Python.
- Students are to inherently use simple object oriented programming principles like modularity, abstraction and polymorphism.
- At this juncture students were still not familiar with any coding standards.
- Students focus was on primarily getting the functionality of the program to work.
This Study is done in 3 phases: Phase I

- All the students were given a two week assignment wherein they had to develop a pacman game in Python.
- Students are to inherently use simple object oriented programming principles like modularity, abstraction and polymorphism.
- At this juncture students were still not familiar with any coding standards.
- Students focus was on primarily getting the functionality of the program to work.
Another two week assignment was given to get the students familiarized with coding standards.

The students were then asked to re-submit the Pacman code written in the first phase by conforming to the PEP-8 standards.
This Study is done in 3 phases: Phase II

- Another two week assignment was given to get the students familiarized with coding standards.
- The students were then asked to re-submit the Pacman code written in the first phase by conforming to the PEP-8 standards.
- Additionally the students were taught about code inspection process and were given some activities in class consisting of Java code instrumented with bugs.
This Study is done in 3 phases: Phase II

- Another two week assignment was given to get the students familiarized with coding standards.
- The students were then asked to re-submit the Pacman code written in the first phase by conforming to the PEP-8 standards.
- Additionally the students were taught about code inspection process and were given some activities in class consisting of Java code instrumented with bugs.
- Students identified the most common types of bugs among the code written by peers.
This Study is done in 3 phases: Phase II

- Another two week assignment was given to get the students familiarized with coding standards.
- The students were then asked to re-submit the Pacman code written in the first phase by conforming to the PEP-8 standards.
- Additionally the students were taught about code inspection process and were given some activities in class consisting of Java code instrumented with bugs.
- Students identified the most common types of bugs among the code written by peers.
This Study is done in 3 phases: Phase III

- First phase of projects from clients commence. Students implement features that are assigned for the first iteration of the project.
- Students performed peer code reviews using format code inspections against the code written by their team members both manually and using static analysis tools.
This Study is done in 3 phases: Phase III

- First phase of projects from clients commence. Students implement features that are assigned for the first iteration of the project.
- Students performed peer code reviews using format code inspections against the code written by their team members both manually and using static analysis tools.
- Students submitted code inspection reports based on template provided by the Instructors.
This Study is done in 3 phases: Phase III

- First phase of projects from clients commence. Students implement features that are assigned for the first iteration of the project.
- Students performed peer code reviews using format code inspections against the code written by their team members both manually and using static analysis tools.
- Students submitted code inspection reports based on template provided by the Instructors.
- Survey on advantages of code reviews is performed.
This Study is done in 3 phases: Phase III

- First phase of projects from clients commence. Students implement features that are assigned for the first iteration of the project.
- Students performed peer code reviews using format code inspections against the code written by their team members both manually and using static analysis tools.
- Students submitted code inspection reports based on template provided by the Instructors.
- Survey on advantages of code reviews is performed.
Projects distribution across languages.

<table>
<thead>
<tr>
<th>Language</th>
<th>Projects</th>
</tr>
</thead>
<tbody>
<tr>
<td>Python</td>
<td>20</td>
</tr>
<tr>
<td>Java (Android Development)</td>
<td>13</td>
</tr>
<tr>
<td>Ruby</td>
<td>2</td>
</tr>
<tr>
<td>C++</td>
<td>1</td>
</tr>
<tr>
<td>Rollbase</td>
<td>4</td>
</tr>
<tr>
<td>JavaScript/HTML</td>
<td>6</td>
</tr>
<tr>
<td>Scala</td>
<td>1</td>
</tr>
</tbody>
</table>

**Table:** Project Distribution across Languages
Static Code Analysis Tools

Pylint

- Pylint is a tool that checks for errors in Python code, tries to enforce a coding standard and looks for bad code smells.
- Other tools like pychecker explicitly does not bother with coding style.
- The default coding style used by Pylint is close to PEP 008.
- Pylint gives a rating of -ve infinity to +10 based on the formulae: $10.0 - (\text{float}(5 \times \text{error + warning + refactor + convention}) / \text{statement}) \times 10$
- Pylint classifies flaws in code into error, warning, refactor and convention. Details of the same can be found here [http://pylint-messages.wikidot.com/all-codes](http://pylint-messages.wikidot.com/all-codes)

Findbugs

- FindBug is a program to find bugs in Java programs. It looks for instances of "bug patterns" — code instances that are likely to be errors.
- Findbugs can be used from the command line or within ant, eclipse, maven, netbeans and emacs.
- Findbugs uses static analysis to look for more than 200 bug patterns, such as null pointer dereferences, infinite recursive loops, bad uses of the Java libraries and deadlocks.
Static Code Analysis Tools

Pylint

- Pylint is a tool that checks for errors in Python code, tries to enforce a coding standard and looks for bad code smells.
- Other tools like pychecker explicitly does not bother with coding style.
- The default coding style used by Pylint is close to PEP 008.
- Pylint gives a rating of \(-\infty\) to \(+10\) based on the formulae: 
  \[
  10.0 - \left( \frac{5 \times \text{error} + \text{warning} + \text{refactor} + \text{convention}}{\text{statement}} \right) \times 10
  \]
- Pylint classifies flaws in code into error, warning, refactor and convention. Details of the same can be found here [http://pylint-messages.wikidot.com/all-codes](http://pylint-messages.wikidot.com/all-codes)

Findbugs

- FindBug is a program to find bugs in Java programs. It looks for instances of "bug patterns" — code instances that are likely to be errors.
- Findbugs can be used from the command line or within ant, eclipse, maven, netbeans and emacs.
- Findbugs uses static analysis to look for more than 200 bug patterns, such as null pointer dereferences, infinite recursive loops, bad uses of the Java libraries and deadlocks.
Pylint reports

The figure shows two side-by-side boxplots summarizing the distribution for the two groups of data.

The minimum, maximum, lower quartile, upper quartile, median, mean and interquartile range for the PyLint rating before code review is: $−295, 10, -7.24, 2.44, 0, -8.025$ and $9.68$. The corresponding PyLint rating values after the code review is: $−95, 10, -6.41, 3.875, 0, -1.817$ and $10.285$. 

18/May/2015
Pylint reports

The figure shows two side-by-side boxplots summarizing the distribution for the two groups of data.

The minimum, maximum, lower quartile, upper quartile, median, mean and interquartile range for the PyLint rating before code review is: $-295, 10, -7.24, 2.44, 0, -8.025$ and $9.68$. The corresponding PyLint rating values after the code review is: $-95, 10, -6.41, 3.875, 0, -1.817$ and $10.285$.

The 5-number summary and mean reveals an improvement in the PyLint rating of the programs after the code inspection process.
Pylint reports

- The figure shows two side-by-side boxplots summarizing the distribution for the two groups of data.
- The minimum, maximum, lower quartile, upper quartile, median, mean and interquartile range for the Pylint rating before code review is: −295, 10, −7.24, 2.44, 0, −8.025 and 9.68. The corresponding Pylint rating values after the code review is: −95, 10, −6.41, 3.875, 0, −1.817 and 10.285.
- The 5-number summary and mean reveals an improvement in the Pylint rating of the programs after the code inspection process.
FindBug reports

- Each finding in FindBugs is reported as a warning, but not all of these warnings are necessarily defects, e.g. warnings related to possible performance issues.
- The figure shows the radar chart summarizing the bi-variate distribution of the data.
FindBug reports

- Each finding in FindBugs is reported as a warning, but not all of these warnings are necessarily defects, e.g. warnings related to possible performance issues.
- The figure shows the radar chart summarizing the bi-variate distribution of the data.
- The decrease in the number of warnings of FindBugs reveals an improvement in the programs after the code inspection process.
FindBug reports

- Each finding in FindBugs is reported as a warning, but not all of these warnings are necessarily defects, e.g. warnings related to possible performance issues.
- The figure shows the radar chart summarizing the bi-variate distribution of the data.
- The decrease in the number of warnings of FindBugs reveals an improvement in the programs after the code inspection process.
## Interpretation of code inspection reports

<table>
<thead>
<tr>
<th>Q1: Common mistakes found out during code inspection?</th>
</tr>
</thead>
<tbody>
<tr>
<td>No doc strings/Doc comments in modules, classes, functions, or methods</td>
</tr>
<tr>
<td>35%</td>
</tr>
<tr>
<td>Importing a module multiple times or using import and from/import</td>
</tr>
<tr>
<td>5.00%</td>
</tr>
<tr>
<td>naming and indentation issues</td>
</tr>
<tr>
<td>32.00%</td>
</tr>
<tr>
<td>Others(part of the bug type description provided to students)</td>
</tr>
<tr>
<td>28.00%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Q2: Classification of bugs identified(Severity 1 being high severity.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Severity1(High)</td>
</tr>
<tr>
<td>13%</td>
</tr>
<tr>
<td>Severity2(Medium)</td>
</tr>
<tr>
<td>23%</td>
</tr>
<tr>
<td>Severity3(Low)</td>
</tr>
<tr>
<td>64%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Q3: Time spent by students on code review?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average time spent by a student on Peer Code Review per 100 LOC</td>
</tr>
<tr>
<td>10 min</td>
</tr>
</tbody>
</table>
# Code inspection review template

<table>
<thead>
<tr>
<th>Team Number</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Student Name</td>
<td>Student1</td>
</tr>
<tr>
<td>Iteration number</td>
<td>Release 1</td>
</tr>
<tr>
<td>Description of feature implemented</td>
<td>action bar + contact list</td>
</tr>
<tr>
<td>Has the functionality being fully implemented w.r.t SRS?</td>
<td>summary screen not there</td>
</tr>
<tr>
<td>Number of lines of code reviewed</td>
<td>Aprox. 400</td>
</tr>
<tr>
<td>Number of classes reviewed</td>
<td>4</td>
</tr>
<tr>
<td>Number of methods reviewed</td>
<td>15</td>
</tr>
<tr>
<td>Time spent on code review</td>
<td>2 hrs</td>
</tr>
<tr>
<td>Static code analysis tool used</td>
<td>IntelliJ</td>
</tr>
<tr>
<td>Defects Found.(description of bug in less than 10 words)</td>
<td>Severity of bug</td>
</tr>
<tr>
<td>Play button not appearing again</td>
<td>Medium</td>
</tr>
<tr>
<td>Number of Defects fixed</td>
<td>3</td>
</tr>
</tbody>
</table>
## Survey on code inspection

### Q1: What is your level of confidence in performing code inspections after conducting code reviews for the first Python based programming assignment?

<table>
<thead>
<tr>
<th>Level</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Very Low</td>
<td>3.00%</td>
</tr>
<tr>
<td>Low</td>
<td>5.00%</td>
</tr>
<tr>
<td>Moderate</td>
<td>32.00%</td>
</tr>
<tr>
<td>High</td>
<td>49.00%</td>
</tr>
<tr>
<td>Very High</td>
<td>11.00%</td>
</tr>
</tbody>
</table>

### Q2: How many total team meetings you had since the start of the project?

<table>
<thead>
<tr>
<th>Range</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Very Low (0-5)</td>
<td>3.00%</td>
</tr>
<tr>
<td>Low (6-10)</td>
<td>22.00%</td>
</tr>
<tr>
<td>Moderate (11-15)</td>
<td>36.00%</td>
</tr>
<tr>
<td>High (16-20)</td>
<td>24.00%</td>
</tr>
<tr>
<td>Very High (More than 20)</td>
<td>15.00%</td>
</tr>
</tbody>
</table>

### Q3: Did code inspection and review activity enable you to understand and comprehend your project better?

<table>
<thead>
<tr>
<th>Response</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>42.00%</td>
</tr>
<tr>
<td>No</td>
<td>19.00%</td>
</tr>
<tr>
<td>Partially</td>
<td>39.00%</td>
</tr>
</tbody>
</table>
Survey on code inspection

<table>
<thead>
<tr>
<th>Question</th>
<th>Yes</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q4: Do you expect an increased number of meetings among team members</td>
<td>72.00%</td>
<td>28.00%</td>
</tr>
<tr>
<td>from now (after the first iteration of code review activity) till</td>
<td></td>
<td></td>
</tr>
<tr>
<td>completion of the project?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Q5: Does peer code-review and inspection help in improving collaboration</td>
<td>82.00%</td>
<td>18.00%</td>
</tr>
<tr>
<td>team communication and cohesion?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Q6: What is the likelihood of changing your team communication</td>
<td>12.00%</td>
<td>25.00%</td>
</tr>
<tr>
<td>mechanism due to the introduction of team inspection and peer code</td>
<td></td>
<td></td>
</tr>
<tr>
<td>review activity?</td>
<td>48.00%</td>
<td>12.00%</td>
</tr>
<tr>
<td>activity?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Very Low</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Low</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Moderate</td>
<td></td>
<td></td>
</tr>
<tr>
<td>High</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Very High</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Threats to validity

- In rating the severity of bugs, the Severity levels of the bugs were assigned by the reviewers based on their prior knowledge.
- We could only evaluate OO paradigm, and were restricted to projects implemented using Java and Python.
Best Practises and Recommendations

Based on our experience

- we recommend creation of check-list with **Bug Types** and **code smells** should be provided to students while doing code review.

- Students usually rush while doing code reviews. Hence we recommend students to review 100-200 lines of code at a time so that, the quality of review will not decrease their quality of review.

- BitBucket/GitHub provides good collaborative code review tools – are free and can be effective used in an educational setting. This will also avoid the overhead of ending up with additional code review platforms.

- Both BitBucket and GitHub provides REST API to mine data. The restAPI provided will reduce the effort required in extracting data for textual analysis.
Conclusion

Our study on introducing peer code reviews at sophomore level of undergraduate curriculum has been positive:

- in terms of improving code quality.
- improving maintainability of code.
- code review process enhanced team cohesion and improved communication between the team members.
References

- R. Shukla, A. Sureka, R. Joshi and R. Mall, "A report on software engineering education workshop,"
- P. C. Rigby and C. Bird, "Convergent contemporary software peer review practices,"
- A. Bacchelli and C. Bird, "Expectations, outcomes, and challenges of modern code review,"
- C. Hundhausen, A. Agrawal, D. Fairbrother, and M. Trevisan, "Integrating pedagogical code reviews into a cs1 course: An empirical study."
- C. D. Hundhausen, A. Agrawal and P. Agarwal, "Talking about code: Integrating pedagogical code reviews into early computing courses."
- X. Li, "Using peer review to assess coding standards - A case study."
- K. Anewalt, "Using peer review as a vehicle for communication skill development and active learning."