EVALUATING AUTOMATED REVIEW PROCESS: JUPITER INCORPORATING WITH HACKYSTAT

A THESIS PROPOSAL SUBMITTED TO MY THESIS COMMITTEE

MASTERS

IN

COMPUTER SCIENCE

By
Takuya Yamashita
Collaborative Software Development Laboratory
Department of Information and Computer Sciences
University of Hawai'i
Honolulu, HI, 96822
takuyay@hawaii.edu

Thesis Committee:

Philip M. Johnson, Chairperson

December 2, 2004
Version 1.0.0

Abstract

Over many years, there is general agreement that software inspection reduces development costs and improves product quality by finding defects in early software development, and these software tools help the inspection process efficiently and effectively. Even though the agreement, developers have not uses one review tool for a long time. Or they even have gave up the functional tool any more and went back to use extremely lightweight text editor tool.

By using Jupiter (review tool) and Hackystat (automated metric collection system), I proposed that the issue is addressed if the automated review framework provides the better understanding of inspection process and as a result, helps developers to continuously use the lightweight functional tool rather than text editor based tool. By providing the transparent inspection process and tool, reviewers and team can be aware of the efficiency use of inspection tool.

To investigate my research questions, I will set a controlled review experiment in software engineering class to give Java based assignments to around 20 students. In first several round, students are encouraged to learn both text editor based and Jupiter based review. Qualitative evaluation will be conducted before and after the round to see the usefulness of review tool. In the next several round, Hackystat metric collection system will be introduced to gather review metrics. It could provide the review analysis such as prepared reviewers, categorized defect types and severity, number of confirmed issues, and so forth. The second qualitative evaluation will be conducted before and after the introduction to see the usefulness of automated review framework.

The expected results would be that automated review framework, including the review tool, provides some useful aspects for inspection process and tool. This result would be one milestone for empirical software inspection community.

Finally, this thesis is going to be done by not later than May, 2005 with first milestone for the implementation of analysis tool by January 2005, second milestone for the evaluation of the review tool by March 2005, and final milestone for the evaluation of the automated review system by May 2005.

Chapter 1

Experiment Evaluation

In order to evaluate if the automated review framework provides the better understanding of inspection process as a result of the fact that developers continuously use the lightweight functional tool rather than text editor based tool, I propose the two hypotheses:

1. Claim 1: Jupiter provides the useful functionalities enough to adopt the lightweight code review tool rather than the text editor tool.

2. Claim 2: Jupiter and Hackystat automated review process provides better understanding of inspection process.

To test the hypotheses, I propose two stages to evaluate claim 1 and 2: 1) evaluation of the usefulness of Jupiter review tool and 2) evaluation of better understanding of the automated review process with Jupiter and Hackystat. The second evaluation is supposed to be conducted after the significant result for the evaluation of Jupiter review tool.

1.0.1 Experiment Design for Jupiter review tool

To evaluate the usefulness of Jupiter plug-in for Eclipse (Claim 1), I propose that the evaluation is considered to be based upon qualitative approach: Questionnaire. The questionnaire will be conducted before and after the use of Jupiter.

Subjects

The experiment will be carried out during the spring semester in 2004 from January to May as a part of both introductory Software Engineering course for approximately 40 undergraduates and graduates in University of Hawaii at Manoa's information and Computer Sciences depart-

ment. It is assumed that the subjects has the background of the required computer science such as programming language, algorithm. However, there will be a room that different students have different ability of skills. To minimize the deficiency of skills, students are supposed to review the basic knowledge which would be necessary to evaluate the claim 1. That is to say, the basic knowledge of Java language, Text editor, Eclipse IDE, and so forth will be reviewed. All students are supposed to have the identical educational tutorial on how to use the text editor in Eclipse IDE before the post questionnaire for Text editor is conducted. They are also supposed to have the identical educational tutorial on how to use the Jupiter after the text editor review process is mastered and before the post questionnaire for the Jupiter editor is conducted.

Materials

The material to be used for review is the code actually students wrote in Java during the course. For each session of review including preparation, individual, and rework phase, the same material is used for all students. The appropriate material to be used in each session will be determined by the course instructor and/or me in such a way that it contains if 1) there are proper lines of code, 2) there exist enough defects, and 3) there are enough parts that educate students. I will also prepare for each session the backup materials to have all of them just in case that there exist no materials as such. It is considered that this approach would be better way than the approach that all materials are prepared before course starts. It is because 1) Instructor has own plan to organize his class and 2) the source codes students wrote are more useful for them to understand and recognize the defects.

Instruments

The main instrument is Eclipse IDE, which is the integrated software development tool, and Jupiter, which is the code review plug-in to the Eclipse. As student experience software engineering course, some software engineering tools such as unit test framework, ant build too, configuration management tool, and web application tool are introduced step by step.

To eliminate the external factor of usability of editor, Text editor embedded in Eclipse is used for the text based review. So students are supposed to experience both text editor review and Jupiter based review within Eclipse IDE.

To see the estimated review time in individual phase, Jupiter sensor for Hackystat is used. This sensor records the review time without any special action such as writing start time and end

time or even pushing start and end button in the tool. Text editor used is a special editor for review, whose most function is the same as regular Eclipse editor, but it can be opened by Jupiter so that the review time for text editor and Jupiter is recorded.

Experiment Execution

Before the experiment starts, subjects are given lecture on how to conduct review process. The lecture includes the explanation of the research purpose, review process, review iteration period, review tool, and so forth.

To iterate though review sessions, the first round is set as three review sessions. The three sessions include two general review process and one group review process. A general review process is determined as the process that one review material would be reviewed in individual phase and team phase by all students. On the contrary, an individual group review process is determined as the process that each small groups has own individual materials to be reviewed in the individual, team and rework phases. The reason why there are two different type of review process is that only individual review phase can conduct the rework phase since the author of the materials would be the rework person. A session is determined as the process that includes individual phase, team phase, and rework phase. The small groups conduct the first round with the text editor in Eclipse.

In individual phase, students are given a source code and supposed to review for just an hour. They are supposed to finish the review regardless of any reason. Jupiter review sensor helps seeing the review time for students. If there is a significant time difference from the time period, students are asked for the reason, and the problem would be solved in the next iteration. After the review done, the review files for students are emailed to Instructor or me.

In team phase, students conduct the team review with the small team members. Instructor or I measure one hour with physical timer in the class. They review all raised issues and validate issues as much as time is allowed. They can the validate issues by setting the resolution status such as ”Vaid-NeedsFixing”, ”Valid-FixedLater”, ”Valid-WontFix”, and so forth. They are supposed to finish the team review in an hour regardless of any reason and send the review file to Instructor or me.

In rework phase, students conduct the rework review only with the individual group review process. The time spent for the rework phase is not calculated because it would be hard to measure the time for rework phase. However, students are asked to fix the all confirmed review issues in a specific period of time. This phase is especially evaluated by the questionnaire.

The review process including as individual, team, and rework phase is the same for the text editor based review and Jupiter based review.

To evaluate the distinct usefulness of Jupiter compared to text editor in Eclipse, I will ask students to have questionnaire after both tool are used. The first questionnaire will be conducted after the first round with the Text editor is done. This would be the evaluation for the usefulness of the Text editor and its review. The second questionnaire will be conducted after the second round with Jupiter is done. This would be the evaluation for the usefulness of the Jupiter and its review.

1.0.2 Experiment Design for Automated Review System

Appendix A

Jupiter plug-in for Eclipse Evaluation Questionnaire

INFORMED CONSENT FORM

Collaborative Software Development Laboratory

University of Hawaii at Manoa

Takuya Yamashita

Primary Investigator

(808) 956-6920

This research is being conducted to examine the usability of the Jupiter plug-in for Eclipse. The purpose of the study is to investigate the usability of the Jupiter plug-in in Software Engineering class in University of Hawaii at Manoa, Information and Computer Sciences department, for the purpose of determining how Jupiter plug-in are easily used by the users. You are being asked to participate because you are computer science students, who are taking ICS 413 software engineering class. Your participation will help see the usefulness of Jupiter plug-in.

Participation in the questionnaire will take approximately fifteen minutes. The questionnaire will involve completing a simple task to evaluate the code review and Jupiter functions.

Confidentiality will be applied to the questionnaire. The names of the participants will not be published or recorded at the questionnaire. The materials gathered during the questionnaire will be destroyed after the data analysis and report are complete.

Participation is voluntary, and refusal to participate will not result in any loss of benefits to which the participant is otherwise entitled. Participants may at any time withdraw from the test with no penalty or loss of benefits to which the subject is otherwise entitled. At the point of withdrawal, all information gathered from the participant will be destroyed.

The participant will be required to use the Eclipse IDE and Jupiter plug-in, and will be held responsible for any injuries that are a result of using the Eclipse IDE and Jupiter plug-in. By participating in this questionnaire, the participant accepts all responsibilities that may occur as a result of completing the questionnaire.

The participant is entitled to see the final research and report upon request. For additional information about the questionnaire, procedures, and/or results, please contact the primary investigator, Takuya Yamashita, at (808) 956-6920.

If you have any questions regarding your rights as a research participant, please contact the UH Committee on Human Studies, at (808) 956-5007.

Participant:

I, hereby, understood the above information, and agree to participate in this research project.

_____

Name (printed)

_____ _____

Signature Date

INSTRUCTIONS

The following is a questionnaire on your experiences using Jupiter plug-in, code review system. There are 22 questions, and it will probably take you between 15 minutes to fill out. If you have any questions, please don't hesitate to ask.

You can put an answering number by select one proper answer and circle it.

It is important to remind you that it doesn't matter what your answers are. So please be honest; that will make it most useful for this research. If you have questions, please ask them before you begin. If you get stuck, then ask questions on a need-to basis.

In order to better understand the strengths and weaknesses of the current version of Jupiter code review system, and to help guide future improvements, please take a few minutes to answer the following questions. It is important to remind you that your identity will be removed before performing any analysis on this data. There are no right or wrong answers: We want to know what your personal experience was.

Some questions ask you to respond with a number from 1 to 5, where the 1

Indicates the "best" and the 5 indicate the "worst". The last question of this questionnaire requests your comments as the response.

I. DEMOGRAPHIC QUESTIONS

1. Have you ever reviewed a source code with an editor such as MS Note pad, Textpad, or the equivalent text editor?

1 Yes

2 No

2. Have you ever used Jupiter code review plug-in for Eclipse?

1 Yes

2 No

II. INSTALLATION/CONFIGURATION

Please provide us with your opinions regarding the installation and configuration of the Jupiter plug-in.

3. What is the current version of Jupiter plug-in? (e.g. 2.1.917.3x)

** Try to see "Help | About Eclipse Platform", click "Plug-in Details" button to scroll down to the Jupiter Code Review Plug-in.*

(Version:)

4. Did you use the Eclipse Update manager specifying the URL which looks like the flowing: http://csdl.ics.hawaii.edu/Tools/Jupiter/Update2 .

1 Yes

2 No - I downloaded the Jupiter zip package from the Jupiter web site and install it manually.

5. Installing the Jupiter code review plug-in was:

(Very Easy) 1 2 3 4 5 (Very Difficult)

6. Configuring Jupiter to add the new review ID

(Very Easy) 1 2 3 4 5 (Very Difficult)

7. Configuring Jupiter to manager the Review ID (i.e. modification, and deletion of a Review ID)

(Very Easy) 1 2 3 4 5 (Very Difficult)

III. OVERHEAD OF USE

In this section, we are interested in learning about the "overhead" of code review you experienced with the Jupiter plug-in, in other words, how much work was required after installation and configuration to gather data and perform analyses:

INDIVIDUAL REVIEW PHASE

8. The amount of overhead required to add an issue with Jupiter was:

(Very Low) 1 2 3 4 5 (Very High)

9. The amount of overhead required to fill an issue information with Jupiter was:

(Very Low) 1 2 3 4 5 (Very High)

TEAM REVIEW PHASE

10. The amount of overhead required to review the list of issues with Jupiter in a team was:

(Very Low) 1 2 3 4 5 (Very High)

11. The amount of overhead required to review each issue information such as the summary, description, and so forth with Jupiter in a team was:

(Very Low) 1 2 3 4 5 (Very High)

12. The amount of overhead required to jump to the target point of a source code from the list of issues (i.e. Jupiter Issue View, double-clicking on an issue in the list) was:

(Very Low) 1 2 3 4 5 (Very High)

13. The amount of overhead required to jump to the target point of a source code from the reviewing issue (i.e. Jupiter Editor, and clicking Jump button) was:

(Very Low) 1 2 3 4 5 (Very High)

REWORK PHASE

14. The amount of overhead required to review the assigned issues after team review done was:

(Very Low) 1 2 3 4 5 (Very High)

15. The amount of overhead required to fix the problem which was assigned to you was:

(i.e. find the point of problem, and change (fix) the source code)

(Very Low) 1 2 3 4 5 (Very High)

FILTER SETTING

16. The amount of overhead required to filter the issues in the list was:

(Very Low) 1 2 3 4 5 (Very High)

17. The amount of overhead required to set up filtering the issues in the list was:

(Very Low) 1 2 3 4 5 (Very High)

IV. FUTURE USE

In this section, we are interested in learning whether you would consider the Jupiter plug-in to be feasible (i.e. appropriate, useful, beneficial) for use in a professional software development context.

18. If you were a professional software developer, using Jupiter at your job would be:

(Very Feasible) 1 2 3 4 5 (Not Feasible at All)

19. If you were a professional team (or project) leader, using Jupiter at your job would be:

(Very Feasible) 1 2 3 4 5 (Not Feasible at All)

20. Please provide any other feedback you could have experience on the use of Jupiter, as well as any suggestions you have on how we could improve its use in future.

Bibliography

[1] J. Barnard. Managing Code Inspection Information. *IEEE Software*, 11(2):59–69, March 1994.

[2] S. lawrence Pfleeger, editor. *Software Engineering: Theory and Practice*. Prentice Hall, 2001.

[3] D. L. Parnas. The Role of Inspection in Software Quality Assurance. *IEEE Transactions on Software Engineering*, 29(8):674–676, August 2003.

[4] K. E. Wiegers. Seven Deadly Sins of Software Reviews. *Software Development magazine*, March 1998.