Defect Prediction

Dear software professional:

Our research group at Zhejiang University is conducting a research project. The project aims at understanding practitioners' perception of defect prediction. Defect prediction has been an active area of software engineering research for the past four decades, however, it is not clear if practitioners value it.

Given your solid experience, we kindly ask you to fill in the following questionnaire. Your responses will help us better understand the perception of software practitioners, and guide us in developing useful tools for you and other software practitioners.

The survey is anonymous, and will take approximately 15 minutes. No personal information is collected.

As an appreciation of your time and valuable inputs, we will give out 50 USD Amazon gift cards to two randomly selected participants. If you want to enter the raffle, kindly enter your email address at the end of the survey. Email address will only be used for raffle purpose. We will not keep the addresses after the raffle. Participation in the raffle is completely voluntary.

Thank you!

Zhiyuan Wan
Zhejiang University

Please contact wanzhiyuan@zju.edu.cn if you have any questions.

Part I. Demographics

1. Do you work as a professional software practitioner? [单选题] *

○ Yes

○ No

2. Are you involved in open source software development? [单选题] *

○ Yes

○ No

3. Which of the following roles best describe your software engineering experience? (multiple choice OK) With how many years of experience (decimals OK)? [多选题] *

☐ Software Development _____ *

☐ Software Testing _____ *

☐ Project Management _____ *

☐ Other (Please enter an 'other' value for this selection) _____ *

4. Please describe your English proficiency level: [单选题] *

○ Very Good ○ Good ○ Mediocre ○ Poor ○ Very Poor

5. What is your current country of residence? [填空题] *

_____

6. Does your organization/cooperation use any defect prediction tool in software development process? [单选题] *

○ Yes (Please specify the tool name) _____

○ No

Part II. Please answer the below questions according to your experience in your most recent project.

1. How many people are involved in the project? [单选题] *

○ 0-5

○ 6-10

○ 11-20

○ 20-40

○ Other (Please enter an 'other' value for this selection) _____ *

2. What is the primary programming language of the project? [填空题] *

_____

3. Do you use static analysis tools (e.g., Coverity) for the project? [单选题] *

○ Yes (Please specify the tool name) _____

○ No

4. Do you feel these static analysis tools help you find bugs? [单选题] *

○ Yes

○ No

Part III. Defect prediction

Defect prediction tends to prevent defects and identify defect-prone software entities in advance, e.g., defect-prone source code file, binary, module and change. Defect prediction tools could help software practitioners prioritize which code to inspect or test, and improve future quality by learning from prior mistakes.

A defect prediction tool can tell whether a new instance is bug-prone or not. The instance could be a package, a file, or a commit. Before using a defect prediction tool, we need to feed data to the tool. The data include source code repositories, code review history, etc.

1. How do you decide which file needs more code inspection / testing effort (i.e., the file has high chance to have bugs)? Please rate the following options from very often to very rare: [矩阵量表题] *

	Very Often	Often	Neutral	Rare	Very Rare	I don't know
a. most recently buggy file	○	○	○	○	○	○
b. most recently changed file	○	○	○	○	○	○
c. file changed by more developers	○	○	○	○	○	○
d. file changed by fewer developers (e.g., only one developer)	○	○	○	○	○	○
e. file with more LOC	○	○	○	○	○	○
f. file with fewer LOC	○	○	○	○	○	○
g. file created/changed by junior developers or newcomers	○	○	○	○	○	○
h. file created/changed by senior developers	○	○	○	○	○	○

	Very Often	Often	Neutral	Rare	Very Rare	I don't know
i. file called by many other files	○	○	○	○	○	○
j. file that calls many other files	○	○	○	○	○	○

2. What factors played a role in your previous answer?
Please choose the relevant factors from the list below, and rank them from most to least important: [排序题，请在中括号内依次填入数字] *
[ ]a. Personal experience
[ ]b. What I hear from my peers
[ ]c. What I hear from my mentors/managers
[ ]d. Articles in industry magazines
[ ]e. Research papers
[ ]f. Other (Please enter an 'other' value for this selection)

3. Based on which granularity do you prefer to prioritize your code inspection / testing effort? [单选题] *

○ a. Feature (a requirement/feature proposed by the customer)

○ b. Component

○ c. File

○ d. Method

○ e. Commit

○ f. Session (every time you save a file)

○ g. Other (Please enter an "other" value for this selection) _____ *

4. Please select the top 3 granularities based on their "importance" in defect prediction: [多选题] *

☐ a. Feature (a requirement/feature proposed by the customer)

☐ b. Component

☐ c. File

☐ d. Method

☐ e. Commit

☐ f. Session (every time you save a file)

☐ g. Other (Please enter an 'other' value for this selection) _____ *

5. Please describe your selection reasons: (Optional) [填空题]

_____

6. Please select top 3 granularities based on their "difficulty" of implementation: [多选题] *

☐ a. Feature

☐ b. Component

☐ c. File

☐ d. Method

☐ e. Commit

☐ f. Session

☐ g. Other (Please enter an 'other' value for this selection) _____ *

7. What do you think the curve of probability density function of bugs in a project looks like? [单选题] *

○ a. Normal distribution
○ b. Uniform distribution
○ c. Long tail distribution

○ d. Other (Please enter an 'other' value for this selection) _____ *

○ e. I don't know.

8. Please describe why you choose the above curve: (Optional) [填空题]

_____

9. What do you think the relation of file size (LOC) and probability of defect occurrence looks like? [单选题] *

○ a. Inverted "U" shape
○ b. "U" shape
○ c. Constant
○ d. Linear growth

○ e. Other (Please enter an 'other' value for this selection) _____ *

○ f. I don't know

10. Please describe why you choose the above figure: (Optional) [填空题]

_____

11. Do you agree with the following statements about bug fixing? [矩阵量表题] *

	Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree	I don't know
Location in the software at which I fix a bug is the location at which an error was made.	○	○	○	○	○	○
Sometimes the real error lies too deep. So risk of introducing new errors is too high to solve the real error.	○	○	○	○	○	○
The same bug can be fixed in multiple ways.	○	○	○	○	○	○
Sometimes try-catch block could be anywhere between where an exception was original thrown and the user interface.	○	○	○	○	○	○
To fix a bug, I prefer to change the code I own rather than the code that I have no ownership.	○	○	○	○	○	○
Some defects are not fixed by correcting the "real" error-causing component, but rather by a "work-around" somewhere else.	○	○	○	○	○	○

12. What measures do you use when you evaluate a defect prediction tool? [多选题] *

☐ a. False alarm rate

Proportion of non-defective files among all the inspected files (A higher value indicates that users would encounter more false alarms.)

☐ b. Recall
Proportion of inspected defective files among all the actual defective files (A lower value indicates that less defective files could be detected.)

☐ c. Combination of false alarm rate and recall

☐ d. Top k% LOC precision
Proportion of defective files inspected when k% LOC are inspected (A lower value indicates that, after inspecting the same number of LOC (k%), developers need to inspect more files.)

☐ e. Initial false alarm count
Given a ranking list of bug-prone files, when inspecting files in the list one by one from the top, number of false alarms encountered before finding the first defect.

☐ f. Other (Please enter an 'other' value for this selection) _____ *

13. Do you agree with the following statements? [矩阵量表题] *

	Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree	I don't know
I usually allocate more time to the modules that are most likely to be defective.	○	○	○	○	○	○
I usually learn insights from defective software modules in the past to avoid pitfalls.	○	○	○	○	○	○
a. Code metrics - Semantics						
Semantic information is more capable to distinguish one code region from another than syntax information. (Semantic describes the meaning of code. Syntax describes the structure of code.)	○	○	○	○	○	○
b. Process metrics - # commits						
A file with more commits is more bug-prone.	○	○	○	○	○	○
b. Process metrics - Code churn						
A file with more added lines is more bug-prone.	○	○	○	○	○	○
A file with more removed lines is more bug-prone.	○	○	○	○	○	○
A commit involves more added and removed lines is more bug-prone.	○	○	○	○	○	○
b. Process metrics - Cache-based						
Recently changed files tend to be buggy.	○	○	○	○	○	○
Recently created files tend to be buggy.	○	○	○	○	○	○
Recently bug fixed files tend to be buggy.	○	○	○	○	○	○

	Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree	I don't know
A file recently co-changed with bug-introduced files tends to be buggy.	○	○	○	○	○	○
b. Process metrics - Other						
A file with more bug-fix changes tends to be more bug-prone.	○	○	○	○	○	○
A file with more fixed bugs tends to be more bug-prone.	○	○	○	○	○	○
A file with frequent changes tends to be bug-prone.	○	○	○	○	○	○
A file with a complex pattern of source code modification tends to be buggy.	○	○	○	○	○	○
c. Developer metrics						
A file that is changed by more developers is more bug-prone.	○	○	○	○	○	○
d. Organization metrics						
Files with fewer lines contributed by their owners (who contribute most changes) are more bug-prone.	○	○	○	○	○	○

14. What factors played a role in your previous answer?
Please choose the relevant factors from the list below, and rank them from most to least important: [排序题，请在中括号内依次填入数字] *
[ ]a. Personal experience
[ ]b. What I hear from my peers
[ ]c. What I hear from my mentors/managers
[ ]d. Articles in industry magazines
[ ]e. Research papers
[ ]f. Other (Please enter an 'other' value for this selection)

15. Please rank the aforementioned metrics according to their impact on bug proneness from biggest to smallest: [排序题，请在中括号内依次填入数字] *
[ ]a. Code metrics (e.g., lines of code, complexity)
[ ]b. Process metrics (e.g., recently changed file, frequently changed file)
[ ]c. Developer metrics (e.g., number of developers who changed the file)
[ ]d. Organization metrics (e.g., lines of code changed by the project owner)

16. What will be the barrier for wide adoption of defect prediction? [多选题] *

☐ a. Cost of collecting historic data

☐ b. Lack of IDE integration

☐ c. Lack of code review tool integration

☐ d. Lack of continuous integration adaption

☐ e. Other (Please enter an 'other' value for this selection) _____ *

17. Do you know about research on defect prediction? [单选题] *

○ Yes

○ No

18. Assume you have a defect prediction tool could help you "prioritize" which code to inspect/test and "improve" future quality by learning from prior mistakes.

Will you adopt it? [单选题] *

○ Yes

○ No

Why won't you adopt a defect prediction tool? [填空题]

_____

19. Could you please brainstorm the functionalities you want to have in a defect prediction tool? The functionality can be as fancy as whatever you can think of. [填空题]

_____

Please enter your email address, if you would like to enter into our raffle to win a gift card: [填空题]

_____