Procedure of Study on Practitioners’ Expectation of Defect Prediction Tool

Table of Contents

[Introduction 2](#_Toc512941452)

[Further Definition of Defect Prediction 3](#_Toc512941453)

[How to define “defect” 3](#_Toc512941454)

[Motivation of “defect prediction” 3](#_Toc512941455)

[Before ready to use “defect prediction” tools 3](#_Toc512941456)

[Typical use scenarios 3](#_Toc512941457)

[Background Check 4](#_Toc512941458)

[Questions just for managers 4](#_Toc512941459)

[Questions for managers/QA/developers 4](#_Toc512941460)

[Non-adopter Question (Only ask none-adopters) 5](#_Toc512941461)

[Awareness Question (Ask both adopters and non-adopters) 5](#_Toc512941462)

[Time-to-market vs. quality 5](#_Toc512941463)

[Defect prediction experience 5](#_Toc512941464)

[Organizational culture & standards 6](#_Toc512941465)

[Reward & punishment system for software quality 6](#_Toc512941466)

[Organizational structure 6](#_Toc512941467)

[Perceived responsibility 6](#_Toc512941468)

[Tool usage observability 6](#_Toc512941469)

[Practitioner inquisitiveness 6](#_Toc512941470)

[Tool advertisement (awareness knowledge) 6](#_Toc512941471)

[Peer influence 6](#_Toc512941472)

[Education 6](#_Toc512941473)

[Adopter Question (Only ask adopters) 6](#_Toc512941474)

[Open questions: 6](#_Toc512941475)

[Factors-related questions: 7](#_Toc512941476)

[Desired functionality 7](#_Toc512941477)

[Performance 8](#_Toc512941478)

[Cost & potential gain 8](#_Toc512941479)

[Status aspects 8](#_Toc512941480)

[Incentives 8](#_Toc512941481)

[Tool advertisement (how-to & principles knowledge) 8](#_Toc512941482)

[Peer influence **(ask only when peer recommendation is mentioned by the developer)** 8](#_Toc512941483)

[Perceived complexity 8](#_Toc512941484)

[Perceived compatibility 8](#_Toc512941485)

[Perceived Trialability 9](#_Toc512941486)

[Reinvention 9](#_Toc512941487)

[Workflow suitability 9](#_Toc512941488)

[Activities 9](#_Toc512941489)

[1. Role Play 9](#_Toc512941490)

[Awareness 🡪 Need 9](#_Toc512941491)

[Case 1: advertisement 9](#_Toc512941492)

[Case 2: (peer without detail) 9](#_Toc512941493)

[Case 3: (peer with detail) 9](#_Toc512941494)

[Case 4: manager 9](#_Toc512941495)

[Need 🡪 Awareness 10](#_Toc512941496)

[2. Rank the factors/attributes of the defect prediction tool 10](#_Toc512941497)

[Provide better environment of adopting the tool -- company managers 11](#_Toc512941498)

[3. Brainstorm of desired functionalities (Opinion from novice vs experienced) 11](#_Toc512941499)

# Introduction

This study is designed to better understand the gap between current research and practitioners’ expectation of defect prediction tools. We would like to investigate the factors that impact practitioners’ willingness to adopt a defect prediction tool.

(The definition of defect prediction tools)

We define defect prediction tools as those tools that aim to classify software artifacts into defect-prone and non-defect-prone, into different defective severity, or predicted the number of potential defects in each software artifact.

The software artifacts can be at different levels of granularity, e.g., module level, file level, method level, or commit level.

During the interview, you will be asked several questions about your past experience with software engineering, and your opinions about defect prediction.

(In case of people have trouble understanding me)

If you have difficulty understanding what I am asking, feel free to ask me to repeat myself.

(Clarify the privacy issue)

During this interview, your voice will be recorded. In any data collected, or in reports or papers that are published, you will not be identified by name. Please be careful not to discuss any sensitive information about the company you work for. If you do mention any, we will do our best to remove it from our transcripts, but better if you don't mention such sensitive information at all.

# Further Definition of Defect Prediction

## How to define “defect”

According to IEEE standard 610, the term ‘defect’ (also used as ‘bug’) is commonly applied as a general expression of ‘fault’. For the sake of convenience, ‘defect’ is adopted in our study.

## Motivation of “defect prediction”

Previous studies have shown that most of the defects are detected in a small portion of software modules. More specifically, around 20% of the software modules result in 80% of the software defects. Therefore, if these 20 percent of defective modules can be identified before code review or testing, the limited code review or testing resources can be allocated reasonably. This is the basic motivation driving defect prediction.

## Before ready to use “defect prediction” tools

Users need to feed data to the tool (e.g., source code repository, version control history, and code review history), and help to the tools to recognize previous “buggy” software artifacts.

## Typical use scenarios

1. Before a release, users leverage a defect prediction tool to evaluate the quality of the software project. A module-level defect prediction tool will tell us whether each module of the project has a bug or not.
2. Before code review, users leverage a defect prediction tool to predict the defective severity of each source code file. A file-level defect prediction tool will rank the source code files according to the predicted defective severity.
3. Before testing, users leverage a defect prediction tool to predict the number of potential defects in each module. A module-level defect prediction tool will rank the modules according to the predicted defect numbers.

Do you have any questions about this definition before we continue?

# Background Check

Are you working as a QA/developer in your company, or manager or both?

(Our participants are QA/developers or managers. Managers are treated as opinion leader in their company, but they will also be asked questions as QA/developers because they usually either worked as QA/developer before or working as both manager and QA/developer now)

# Questions just for managers

1. How many people do you supervise?
2. Can you tell me a little bit about your job's duty?
3. Can you tell me a little bit about the projects that you have participated in? programming languages, and properties, e.g., start-up, legacy, long-term, refactor?

(historic data availability for defect prediction)

1. Do you know how decisions are made at your company about tool purchasing? Who makes the decision?
2. Where do you fit into the purchasing chain?
3. What is the most important factor when you consider tool purchasing?
4. Does your company have budget just for tool purchasing? how about defect prediction tools?
5. If budget is a big concern, why don’t you consider open source defect prediction tools?
6. Typically, how do you know about a defect prediction tool? What is the information channel that you rely on? How do you trust that channel?
7. After purchased the tool, what is the company’s strategy to let people actually use that tool? Did this strategy succeed?

(Talk about specific cases, if any)

Have you ever adopted any defect prediction tool for your group?

(adoption case)

What is the name of the tool you adopted? Can you tell me the situation when you adopted that tool? What are the concerns?

What is the result of adoption?

(main reason for non-adoption)

What is the main reason you haven’t adopted any defect prediction tool for your group?

# Questions for managers/QA/developers

1. Ask “awareness question”

2. Have you ever adopted any defect prediction tool? {

Yes. 🡪 Go to the Adopter Question part

No. 🡪 Go to the Non-adopter Question part

}

## Non-adopter Question (Only ask none-adopters)

What is the main reason you think that you haven’t used any of defect prediction tools? (An open-ended question before all the specific questions. Elicit more if possible)

## Awareness Question (Ask both adopters and non-adopters)

(Do “Activity 1: role play” before asking awareness question)

### Time-to-market vs. quality

What are the domains of the application you have developed? Was quality a big concern of the software you have developed?

Which programming language you have used?

Is there a demand for rapid release of software to the market?

How long does it usually take for code review/testing? (percentage of a release)

### Defect prediction experience

**Before**

What will you / your team do before unit testing / code review / testing?

How do you decide which file to test / for code review? What factors do you look into? e.g., lots of changes, number of developers, size of file, complexity etc.

How long does the preparation for code review / testing take?

**Ongoing**

What makes you decide to stop code review or testing?

Do you look into corner cases?

At which granularity level do you do testing? File level, method level, module level etc. Why do you select this granularity level?

**After**

Did code review / testing usually cost more resources but achieve less effects than you expected?

### Organizational culture & standards

Is developing software of high quality a big concern in your company?

Does your company have any standards to follow in terms of quality development?

### Reward & punishment system for software quality

Does your company have any reward and punishment system for software quality?

### Organizational structure

Does your company have a dedicated testing team?

### Perceived responsibility

Do you think you as a developer/QA is responsible for software quality? Or the testing team should responsible for software quality? Or other dedicated teams?

### Tool usage observability

Can you describe the environment that you usually work in? (share a cubicle with some peers;

sitting in a private cubicle but have peers sitting nearby; a private office)

### Practitioner inquisitiveness

Are you interested in exploring new tools and techniques relate to your work? What is your patient level for looking for new tools?

### Tool advertisement (awareness knowledge)

Have you ever seen any defect prediction tool advertisement?

Where did you see it? When? What type of advertisement did you see?

### Peer influence

Has any of your colleague recommended a defect prediction tool to you before?

Does anybody else in your company use the defect prediction tool? (Nobody use it? “working environment” and “are people around you using it” do make difference. Or management issue?

Why doesn’t the company introduce defect prediction tool to their developers?)

Has your manager ever required or encouraged you use any defect prediction tools?

### Education

Have you learned about any defect prediction tools through university courses or company trainings?

Does your company provide this kind of training? Is it mandatory or optional?

# Adopter Question (Only ask adopters)

## Open questions:

What is the name of the tool you adopted?

Could you please describe the tool to me?

Which part of this tool you like most?

Which part of this tool you like least?

When did you adopt the tool?

How did you know about this tool?

What type of application you were developing when you adopted this tool? Did this tool help?

What made you decide to adopt this tool?

What made you decide to try out this tool?

Have you ever recommended this tool to people you know?

How did you recommend this tool to others? How does this topic bring up between you and your friend? Did he or she see you using that tool?

What is the result of the recommendation? Have them tried it out or adopted the tool?

Are you still using this tool? {

Yes. {

Have you ever tried other tools that have similar functionality? If so, what makes you continue to use this tool?

If I recommend XXX to you, which has more advantages than the one you are using, will you consider discontinue using the current tool?

}

No. {

how long did you use that tool?

Why discontinue using that tool?

What kind of effort the development team make can change your mind?

}

}

What do you think is the main reason for the defect prediction tool underused problem?

## Factors-related questions:

### Desired functionality

What output do you expect from a defect prediction tool? (added) Classify software artifacts into defect-prone and non-defect-prone, into different defective severity, or predicted the number of potential defects in each software artifact?

Does that tool have all the desired functionalities you want?

### Performance

Does that tool give a good performance? In what form?

### Cost & potential gain

How much does this tool cost? (financial cost)

Was it hard to learn to use that tool? (learning cost)

How long it took for you to get familiar with all the operations of that tool? (learning cost)

### Status aspects

Will use this tool help you gain status aspects? i.e., treated as more experienced in your company?

### Incentives

Did your company provide any incentives for asking you to adopt this tool? Or any punishment if you refuse to adopt it?

### Tool advertisement (how-to & principles knowledge)

How did you learn how to use this tool?

### Peer influence **(ask only when peer recommendation is mentioned by the developer)**

Did you trust the colleague who recommend this tool to you?

Is the situation of that colleague similar to yours?

How did the colleague recommend this tool to you? highly recommended? or just mentioned

about it?

### Perceived complexity

Was the use interface of the tool complex to you?

### Perceived compatibility

Was this tool compatible with the operating system (OS) you are using?

Was this tool compatible with the Integrated Development Environment (IDE) you are using?

Was the operation of that tool similar to dominating tools?

Did this tool come with a bunch of other tools as a cluster?

Did this tool have some functionalities or strengths that other tools do not have? **(Did this tool fit the niche of customers’ requirement?)**

### Perceived Trialability

Did this tool have detailed tutorial?

Did this tool have complete documentation?

### Reinvention

Was this tool configurable?

### Workflow suitability

Did this tool fit into your workflow?

# Activities

## 1. Role Play

(Awareness of the tool drives the need of the tool or the need of the tool makes

people aware of the tool, which one is the case?)

### Awareness 🡪 Need

(Online Advertisement vs interpersonal network ( peers vs opinion leader))

### Case 1: advertisement

One day, you are curious about “how to predict defect-prone software artifacts”, so you google it. This page come out. Will you click the AD inside the red rectangle? (show the picture)

A page contains more detailed information come out. Please read it for 2 minutes.

Will you try this tool out?

### Case 2: (peer without detail)

Suppose I am your colleague. One day, I said this to you when we met in our company:

Hi, I am using a tool called Defect Prediction. This tool is really good for me to prevent defects in my code, you might want to try it out.

Will you try this tool out?

### Case 3: (peer with detail)

Remember I was trying to evaluate the quality of a module last time? I asked you to help me. But we did not find anything. I finally check the defect-prone files predicted by this tool! I think it might help you out later in similar situations.

Will you try this tool out?

### Case 4: manager

Suppose I am your manager. One day, I said this to you: Hi, I know a tool named Defect Prediction. It can save our time and release more reliable products. Why don’t you try it out?

Will you try this tool out?

### Need 🡪 Awareness

(trusted peer vs untrusted peer)

Suppose you are the guy who posted this post in stack overflow. Basically you need a tool to help you prevent defects and allocate code review/testing resources. Please read this post for 1 minute.

Suppose somebody answered you and posted a link to Defect Prediction tool. Will you try this tool out?

## 2. Rank the factors/attributes of the defect prediction tool

Can you rank these factors in terms of how important it is when you make your adoption decision? I will explain the factors one by one.

Please let me know if you have any questions.

First, let’s start with 5 main factors:

**Perceived Complexity:** how complex is the tool?

**Perceived Compatibility:** how compatible this tool with your working environment?

**Perceived Trialability:** how easy can you try this tool out?

**Perceived Relative Advantage:** the advantages this tool gives to you over “not using any tool” or “using other tools”.

**Reinvention:** Can you configure or even customize this tool to more suit your needs?

Can you tell me your opinion about these 5 main factors first?

Next, we are going to look at more detailed factors.

Suppose we make a better tool:

**Desired functionality:** if this tool has whatever functionalities you want?

**Performance**: does this tool give output of good quality? (add)

**Cost & potential gain:** what are the cost and potential gain if you adopt this tool? is it worth

adopting?

**Compatibility with OS:** is this tool compatible with the operating system you are using?

**Compatibility with IDE:** is this tool compatible with the integrated development environment you

are using?

**Operations similar to dominating tools:** is the operations of this tool similar to the dominating

tools? which reduces the difficulty of learning it.

**Framework complexity:** is the framework of this tool hard to understand?

**User interface complexity:** is the UI hard to understand or hard to use?

**Tutorial:** is the tutorial well written and comprehensive?

**Documentation:** is the documentation complete and helpful?

**Technology cluster:** does this tool come out with other tools as a cluster?

**Configurability:** can you configure this tool? In what form?

**Customizability:** can you customize this tool? How? (usually larger changes than configuring).

**Rationale:** Should the tool give rationale or reason for predicting a file or method to be buggy?

Can you think of other factors that are not mentioned here but are important to you when making adoption decision?

### Provide better environment of adopting the tool -- company managers

**Organizational culture & standards:** is this tool compatible with the company’s culture and standards?

**Status aspects:** Will use this tool help you gain status aspects? i.e., treated as more experienced in your company?

**Incentives:** (a factor comes from the company & managers) does anybody provide you incentive to adopt your tool or punishment if you don’t?

## 3. Brainstorm of desired functionalities (Opinion from novice vs experienced)

Can you brainstorm the functionalities you want to have in a defect prediction tool? The functionality can be as fancy as whatever you can think of.