UNIVERSITÄT PADERBORN
Die Universität der Informationsgesellschaft
Faculty for Computer Science, Electrical Engineering and Mathematics
Department of Computer Science
Research Group Software Engineering

Master's Thesis Proposal

Submitted to the Software Engineering Research Group
in Partial Fullfilment of the Requirements for the Degree of

Master of Science

Responsiveness in Static Analysis Tools

by
Sandeep Varma Ganaraju

Thesis Supervisor:
Prof. Dr. Eric Bodden
M. Sc. Lisa Nguyen

Paderborn, March 17, 2019

Erklärung

Ich versichere, dass ich die Arbeit ohne fremde Hilfe und ohne Benutzung anderer als der angegebenen Quellen angefertigt habe und dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat und von dieser als Teil einer Prüfungsleistung angenommen worden ist. Alle Ausführungen, die wörtlich oder sinngemäß übernommen worden sind, sind als solche gekennzeichnet.

Ort, Datum	Unterschrift

Abstract. The Static Analysis plays a major role in Software Development to find bugs and any vulnerabilities in code. There are different Analysis tools available in market. However, it is found out in different surveys about why the tools are not efficient as expected by Software Developers.

Contents

1 Introduction

The effectiveness of Software Development relies on bug free coding. In our day to day progress in coding leads to complexity of software which brings a broader scope for bugs and vulnerabilities that could be introduced easily. There are many Static Analysis tools available in market to address these primary issues. However in latest surveys by Maria et al. [CB16] and by Johnson et al. [JSMHB13] it is noticed that Software Developers are not quite happy with effectiveness and usability of Static Analysis tools. This brings the scope for improvement of Static Analysis tools and the paper by Nguyen Quang Do et al. [NQDB18] introduces how Gamifying the bug fixing process could enhance the usability of Static Analysis tool.

2 Background

Static analysis tools suffer from well-documented usability issues [CB16, JSMHB13].

3

Motivation

4

Problem Statement

Research Question: How to integrate the results of multiple analysis tools?

RQ1: How to carry traceability?

RQ2: Display results from the same codebase from different analysis tools?

- Scale up? Impact of one fix on different warnings?

RQ3: What feedback works to know that the bug fixing's ongoing?

- How to deal with short/long updates for different tools?

5 Objectives

Static analysis tools suffer from well-documented usability issues [CB16, JSMHB13]. Some analysis tools can report results in milliseconds, while others can take up to hours or days. This discrepancy in waiting times can be confusing to code developers, and in some poorly implemented tools, can lead to them thinking that the tooling froze when it is simply computing results.

In this thesis, it is researched on different user interface designs that allow code developers to navigate around this issue in a non-disruptive way.

1. Research different techniques that tackle the issue of responsiveness in other domains of software engineering.

2. Adapt those techniques and design own techniques for the domain of static analysis.

3. Design prototypes with a mocking tool [B19] of those techniques to improve the usability of analysis tools - with respect to responsiveness- (e.g., FindBugs, Soot, Checkmarx, etc.).

4. Design user studies that evaluate the efficiency of those techniques, with professional code developers.

5. Run the user studies and report on their results.

6 Approaches

The Thesis work follows the Deduction research approach where the theory is established based on own ideas in addition to literature review findings.

The prototypes are created based on the established theory and user study is performed which follows Qualitative and Quantitative approaches in assimilating the results.

Ideas:

1. RQ1: Traceability
    - Every submission to analysis tool is time stamped.
    - Show bugs fixed / new bugs introduced
    - Revert option for every submission (this could help user to ty alternative approach of fixing a certain bug)
2. RQ2: Results
    - Bug List (Column - tool icons)
    - Bug filter
    - Old/ New Bugs - Total Bugs
    - Categorise (ex. Security, Memory consumption etc)
3. RQ3: Feedback
    - Percent bar for each bug in bug list
    - if user other screen, right bottom with latest bug fixed attempt been seen by percent bar
    - popup 'Try Again' and 'Cancel'

- Bug removed from list, - 'decision pending' status beside bug, better animation make believe system running
- Long time tool result check if Short time tool reported bug fixed in after time.

7 Evaluation Plan

User Study protocol:

The User Study is performed following this protocol as guidelines.

Phase 1: Set up study environment

The solution ideas for the Research Questions are assimilated into best 3 prototypes prepared using Mockup tool i.e., Balsamiq. These prototypes as exported as pdf and saved at Online drive which can be given access to the user once he agrees to participate in the study.

Phase 2: Organize appointments with users

An invitation will be sent to users typically software developers and preferably who use Static Analysis tools or at least having minimum experience. This ensures the results to be qualitative. A Consent form is signed to make sure the collected user data can be used for this user study safely without violating user privacy.

Phase 3: Introduction

The User Study is introduced once the user shows up at the appointment. Once again, what this user study about and what data is collected during the process.

Phase 4: General Questions

Video recording is started. The user is asked with questions related to his software development experience and with Static Analysis tools.

Phase 5: Performance of tasks / Observing the usage

The user is given with the task to fix a bug in a way that one could do on an interface. The task would be the following;

- Select the project
- Observe the bugs notification
- Select a bug
- After observing the code editor,
click on an option on screen to indicate the user has fixed the bug

During this process, the user gives feedback on how he/she feels by the responsive nature of the Analysis tool. This could be analysed with predetermined questions to make results quantitative.

This process is repeated for the three prototypes.

Phase 6: Wrap up

The user is thanked for participation and recorded video is saved for data extraction.

Phase 7: Data Extraction

The questionnaire asked during the user study session is filled out in form for analysis in comparison to other users' responses.

This helps to conclude whether the researched Static Analysis Tools User Interfaces are responsive as per the user expectations and usage.

8

Preliminary Structure

1. Introduction
2. Outline
   - 2.1 Problem Description
   - 2.2 Objectives
3. Background
   - 3.1 Static Analysis
   - 3.2
   - 3.3
4. Approache
   - 4.1
   - 4.2
5. Limitations and Future Work
6. Conclusion

9

Time Plan

Bibliography

[And] Paul Anderson. Static analysis results: A format and a protocol: Sarif & sasp.

[B19] Balsamiq. rapid, effective and fun wireframing software. | balsamiq, 18/01/2019.

[CB16] Maria Christakis and Christian Bird. What developers want and need from program analysis: an empirical study. In *Automated Software Engineering (ASE), 2016 31st IEEE/ACM International Conference*, pages 332–343. IEEE, 2016.

[JSMHB13] Brittany Johnson, Yoonki Song, Emerson Murphy-Hill, and Robert Bowdidge. Why don't software developers use static analysis tools to find bugs? In *Proceedings of the 2013 International Conference on Software Engineering*, pages 672–681. IEEE Press, 2013.

[NQDB18] Lisa Nguyen Quang Do and Eric Bodden. Gamifying static analysis. In *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ESEC/FSE 2018, pages 714–718, 2018.