**Assignment of Software Engineering**

1. *Research Paper*

*Title* **:** Understanding What Software Engineers Are Working on

*Auther Name* : Ralf Lämmel, Alvin Kerber, and Liane Praza

*Conference* **:** This paper appears in Proceedings of 28th

International Conference on Program Comprehension, ICPC

2020.

Summary

This paper is about the work-item prediction challenge, predicting what a software engineers (a developer, an incident responder, a production engineer) working around ,he might would not know what diff that is what system in engineers working on. as it applies to software engineering workflows in software intensive organizations.

It is challenging because we may not easily understand each action on the timeline of a developer, so when developer publish a diff or may be review a ‘diff’ that is obvious a person working on this and that diff but person said a query DB or read some documentations then it is not clear that what diff it could be about and this is what we could **dark matter** it also called in between situation, some alert form of ‘automatic documentation’ is needed such as suboptimal performance of an important system component. challenges faces by developer : i) engineers rely on a multitude (perhaps hundreds) of loosely integrated tools; ii) engineers engage in concurrent and relatively long running workflows; iii) infrastructure (such as logging) is not fully aware of work items; iv) engineering processes (e.g., for incident response) are not explicitly modeled; v) too many tools used by engineers on a given day for many of facebook employees . So these tools come and go they are not integrated they have not perfect tool so we don’t easily know what are these tools; vi) Too many work items, at a time. An extended related work discussion which connects the work-item prediction challenge to research in the areas of program comprehension, mining software repositories, process mining, and machine learning. This discussion documents the need for and the potential of leveraging, advancing, and combining existing techniques to tackle the work-item prediction challenge more efficiently in practice.

1. *Second Research Paper*

*Title* **:** How Graduate Computing Students Search When Using an Unfamiliar Programming Language

*Authers Name* : Gina R. Bai North Carolina State University Raleigh, Joshua Kayani North Carolina State University Raleigh, NC, USA , Kathryn T. Stolee North Carolina State University Raleigh, NC, USA

*Conference* **:** This paper appears in Proceedings of

28th International Conference on Program

Comprehension, ICPC 2020.

Summary

This research paper is about how graduate computing students search when solving programming tasks in a new or unfamiliar language. Software is often reading using multiple programming languages it espect the developer is master in the multiple languages well learning a new or working in unfamiliar language, developer research online information example recent studies have shown that the knowledge of one language can interfere with learning with new language for example

If I way to learn how to print “ hello world “ using java and c++ and I know this is called print statement so when I would like to learn HELLO WORLD in javascript i can just search with the same query print statement in javascript. Similarly, if we were given a task to create a button that pops a message box with ‘hello world’. We can just search hello world in VBA. If this not provide enough information to us that how we can create a button we can add a query how to create a button: mission complete but is this search always easy!

*EXAMPLE SEARCH QUERIES AND REFORMULATION* :

Each query was followed by three result clicks on average.

S 1 Search Query + 0 Result

SC 1 Search Query + 1 Result Click

SCC+ 1 Search Query + 2 or more Result Clicks

Click An average participant spent about five minutes before reformulating a query. Learners generated more verbose queries than professionals did (5.6 words vs. 1.9 words) and learners required longer time to scan the search results than professional developers. This terminologies were use in search queries by participant in this study. The participant conceder their programming skills as expert in java and c++ and research online when solving programming problem. We can see their participant followed terminologies form java when composing and editing queries. In the fact the function called VLOOKUP in java.

CHALLENGES IN CODE RESEARCH :

Those developer or computing students follow terminologies from other languages there are familiarities. It is common that , they experience the challenges such as I don’t know how to search for; as the task may be hard to describe or what i searched for is not what I actually want

**KNOWLEDGE OF ONE LANGAUGE CAN INTERFERE WITH LEARNING A NEW LANGUAGE OFTEN DIFFERS DRASTICALLY**

Prior studies on code search

Professional developer and their daily work

Studies on code search on software engineering seek understand how and why developers search when performing their daily work. Researchers introduce new code search tools for purpose potential improvement of existing code search. In this study we are interested the code search behaviors of experienced programmers when working with an unfamiliar programming language and we called them subsequent language learner. We find that subsequent programming language learners search with the purposes of exploring for example code, designing new features and understanding why code performs as it does. Learners composed more verbose queries, and required longer time to scan through the search result than professional developers did.

Study design

Google researchers Sadowski and colleagues conduct a study explore know and why google developers search when they working on their daily tasks.

They also summarize the properties of developer search session.

RQ3: What are the factors that impact the success of search queries?

When looking for APIs or implementations, 71 of 107 (66.4%) searches were successful. In addition, consulting documentation and tutorials led to more successful searches than Q&A sites (e.g., StackOverflow). Learners frequently borrowed terms from languages with which they are familiar when composing queries, these queries were more successful on average than the typical query. Learners borrowed same terms from languages with which they are familiar when searching for examples in an unfamiliar language it is easy to search. Techniques for code searching that can determine the relative APIs/libraries based on the context in the queries, provide query reformulation suggestions according to the search goals, map APIs between programming languages and present sample code are suggested to support more programming languages. Another suggestion for future work is to explore how to detect and correct the term mismatch in queries during the language migration.

1. *Third Research Paper*

*Title*  **:** What is the Vocabulary of Flaky Tests

*Auther’s name* : Gustavo Pinto Federal University of Pará Belém, Brazil

Breno Miranda Federal University of Pernambuco Recife, Brazil

Supun Dissanayake University of Adelaide Adelaide, Australia

*Conference*  **:** This paper appears in Proceedings of

28th International Conference on Program

Comprehension, ICPC 2020.

*Summary*

Flaky tests are software tests that exhibit a seemingly random outcome (pass or fail) despite exercising unchanged code. In this work, we examine the perceptions of software developers about the nature, relevance, and challenges of flaky tests. Our scientific knowledge about flaky tests is still very limited. Flaky tests are test cases that sometimes pass and sometimes fail, without any obvious change in the test code or in its execution environment. Unfortunately, the non-deterministic behavior of flaky tests could severely decrease the value of an automated regression suite. For instance, when dealing with flaky tests, developers may not trust the outcome of these tests and ultimately may start ignoring if a test failure is due to a real bug or its non-deterministic behavior. Ignoring if a test failure is due to a real bug or its non-deterministic  
behavior .Such behavior could be harmful to developers because test failures do not always indicate bugs in the code. Our test suite should act like a bug detector. Without any obvious change in the test code or in its execution environment. Unfortunately, the non-deterministic behavior of flaky tests could severely decrease the value of an automated regression suite. For instance, when dealing with flaky tests, developers may not trust the outcome of these tests and ultimately may start ignoring if a test failure is due to a real bug or its non-deterministic behavior. In the last few years, research on test flakiness has gained significant momentum. Prior work focused on characterizing what is a flaky or identifying the root cause of flaky tests. However, little effort has been placed on how to efficiently recognize a flaky test. This paper focuses on the question of whether there are programming identifiers (e.g., method and variable names) that could be used to automatically recognize flaky tests. More precisely, the paper proposes to answer the question: Is there a programming vocabulary that could distinguish flaky tests from their non-flaky relatives? To answer this question in this paper, we started by extracting test cases from a well-known data set of flaky tests [3]. Since we needed to have flaky and non-flaky tests and the data set only provided flaky data, we decided to rerun the Java projects studied in this data set, but now keeping an eye open for finding flaky tests. We then ran 100 times the 64k test cases of the 24 studied Java projects. We flag a test as flaky if there was disagreement in the test outcomes. After the identification of flaky tests, we extract all identifiers from the test cases using traditional tokenization procedures. Finally, the pre-processed flaky and non-flaky test cases were used as input to five machine learning algorithms.

A good suite of tests should let you decide whether the code is ready to be released. When we have a test suite that we can trust, a successful test run gives me the green light to proceed with a release. It gives me confidence that I can refactor the code safely. In TDD, we should run all our tests after every code change. Sometimes this is not always possible, but at least every now and then we have to run the whole suite of tests. But at least, we have to ensure that all our tests run successfully after committing our changes. If a test constantly fails, this is not flaky test and must not be confused.