Assignment 1: Part A (Towards Better Bugs Fixing: Combining Static Analysis Tool Reports and Community Knowledge)

Sahar Hassan Alzahrani
Faculty of Sciences, Engineering and Technology
Adelaide University
Adelaide, SA, Australia
a1938372@student.adelaide.edu.au

Task 1: Problem Description

Background

Bugs have a crucial influence on the life cycle of software development, their introduction, discovery, and fixing account for a good portion of development costs (Jin et al. 2023). Therefore, in modern software development practices, increasing numbers of developers rely on automated tools for maintaining code quality and identifying potential issues at the initial stages of the development process. Specifically, automatic static analysis tools (ASATs) such as Infer and Fxcop have gained much popularity for their ability to find common bugs such as null dereferencing, resource leaks, and thread safety violations. Despite the effectiveness of such tools, Vassallo et al. (2020) argue that several developers do not consider the output to be easily comprehensible or believable and thus ignore all warnings since false positives tend to be high or since there is no explanation in context. This discordance is a fundamental problem in the practical application of ASATs: developers are not always aware of how to interpret or act upon the tool-generated bug reports. This demonstrates that we need better support to allow developers to understand these notifications properly.

In contrast, developer forums like Stack Overflow offer highly contextualised, detailed explanations of real coding problems and serve as a large archive for several issues of which are may the same as those ASATs aim to target (Galappaththi, Nadi & Treude 2022). Yet these are not coupled with static analysis tools, and there is some doubt as to whether the types of errors caught by those tools are well represented on community websites or how developers perceive and repair them. This project aims to bridge that gap by inquiring whether Microsoft's Infer finds the same types of bugs that one might find on Stack Overflow, how developers identify and repair them, and how this could enhance the static analysis tools usability and reliability.

Motivation

I noticed that although static analysis tools are usually extremely good at detecting bugs early, their warnings confuse developers or are lack of details or may be difficult to understand. For example, Infer can report a THREAD_SAFETY_ VIOLATION like "Unprotected write to field _variableName outside synchronization," which is accurate but lacks any context about why these matters or how to fix it. At the same time, community forums like Stack Overflow rich of valuable insights and real-world solutions; however, it might not always exactly correspond to what static analysis detects. There are three key reasons why this disjunction matters: (1) static analysis warnings may be too technical or context-free, (2) these warnings tend to include false positives, therefore reducing developer trust, and (3) they seldom provide real-world examples or solutions. This disconnects motivated me to

examine the degree to which these tools reflect developers' real needs and how forums reveal gaps or misunderstandings and may pave the way to refine tools.

Questions

RQ2: To what extent are the bug types reported by Infer also discussed on Stack Overflow, and how frequently do these overlaps occur?

RQ1: What kind of solutions or explanations are provided on Stack Overflow for bugs that match those reported by Infer, and how effective or complete are these answers?

Task 2: Dataset Description

This project utilizes two empirical datasets that can be processed, cleaned, and combined to examine the relation between automatic static analysis warnings and developer conversations. The first dataset provided by **Microsoft and called InferredBugs**, which keeps systematic records of Java bugs from Infer and C# bugs from InferSharp. Another one is offered by **StackExchange,** which is a known question-and-answer website for developers and provides a large archive of their post data. Collectively, these datasets facilitate a large-scale comparison between tool-detectable problems and those addressed by human. The aim is to gather, clean, and merge both datasets with the purpose of examining how the types of bugs that Infer detects are reflected in actual real-world discussion, in order to enhance the detection tools. A comprehensive explanation of the structure and fields of both datasets is provided below:

1. Microsoft Inferred Bugs Dataset (InferredBugs)

- **Source:** Microsoft Research open data (Microsoft Research 2025).
- **Description:** It is a comprehensive metadata dataset of bugs reports generated by Infer for Java and InferSharp for C#. Open-source repositories had been automatically analysing to construct this dataset. (Microsoft 2023).
- **Fields Included**:
    - `File`: File path where the bug is located.
    - `Bug_type`: Type of detected error such as `THREAD_SAFETY_VIOLATION` and `RESOURCE_LEAK`.
    - `Severity`: Reported severity level contain values such as `HIGH` or `LOW`.
    - `Line`: Line number where the issue occurs.
    - `Qualifier`: A human-readable explanation of the bug.
- **Usage:** The information will be parsed and filtered by bug type. Common bug types will be identified and associated with developer keywords used on Stack Overflow platform.

2. Stack Overflow Dataset (Stack Exchange Data Dump)

- **Source**: Stack Exchange Data Dump (Stack Exchange, Inc 2025).
- **Description**: Contains real developer questions, answers, and discussions tagged with programming languages, error types, and bug terms.
- **Fields Included**:
    - `Title`: Subject line summarizing the developer's question.
    - `Body`: Explanation of the issue in details.
    - `Tags`: Classification of issue type such as `java`, `nullpointerexception`.

- `Score`: Net votes (upvotes minus downvotes) reflecting the question's community value.
- `CreationDate`: Date and time when the question was posted.
- `AcceptedAnswerId`: ID of the selected answer that resolved the issue.

- **Usage**: Posts are scanned for applicability to Infer bug types. Posts without code examples or appropriate tags will be disregarded.

These datasets represent a rich source of Big Data, which align with the defined 4Vs model in our course: Volume, Variety, Velocity, and Veracity. In Volume, the Stack Overflow data set contains millions of records representing diverse programming issues, and the InferredBugs dataset contains thousands of detailed static analysis notices produced by executing Infer on open-source Java and C# projects. The diversity manifests in the data structure: Stack Overflow encompasses semi-structured data characterized by community answers, tags, and natural language descriptions, while InferredBugs generates structured metadata regarding bugs, including bug type, severity, file path, and explanations. The velocity dimension is exemplified by the steady stream of developer questions and answers on Stack Overflow, along with Infer's capability to analyse changing codebases. Veracity is provided by the credibility of the sources as InferredBugs is a mature static analysis tool written by Microsoft and released openly on GitHub, and Stack Overflow standout as real developer experience, curated through community votes and accepted answers. Together, these datasets form a solid foundation for studying how automated bug warnings relate to real developer problems.

Task 3: Initial Data Processing

The early phase of data processing is the preparation and cleaning of the two large datasets for effective analysis in purpose of understanding the reflection of static analysis-detected bug types to real-world developer discussions.

1. Data Extraction and Processing

InferredBugs: This dataset was available in GitHub published by Microsoft. It produces bug reports in the JSON format. Every report indicates an issue discovered, such as NULL_ DEREFERENCE or RESOURCE_LEAK. To transform this cluttered output into an analysable form, I wrote a Python script to extract relevant information and store it in a clean CSV file. The Python script recursively looks for .json files in the directory inferredbugs/. For each file, it opens and reads the JSON data using the json module. All the inferred fields are written to a CSV file named inferredbugs_summary.csv.

Stack Exchange Data Dump: The Stack Overflow dataset downloaded as 7z. file from SatckExchange website then extract the Posts.xml file. In order to work effectively with such as large file, I used the xml.etree.ElementTree.iterparse() function in Python to parse the file line by line. Every post is filtered to extract relevant fields such as title, body, tags, and score. I searched for keywords related to the Infer bug types in Posts to find matches mentioned any Infer bug_type names or keywords from the bug descriptions. The results were stored in matched_posts.csv, for further analysis.

2. Data Integration and Mapping

The processed data are significant for facilitating depth analysis and interpretation. The conversion of raw JSON bug reports into CSV form file enhances the analysability of the data, which lead to support statistical analyses and enable comparisons throughout various aspects. Furthermore, the dataset provides a thorough understanding of bugs that reported by tool which

are recognised, discussed, or resolved by professionals by connecting static analysis findings with developer conversations on Stack Overflow. The integration in question is instrumental in determining the type of problems that are more relevant in the developer community. In addition, the resultant merged dataset constitutes the ground for conducting frequency analysis, relevance monitoring, and bug classification by severity, all aimed at enhancing the knowledge of the interaction between static analysis tools and the problems encountered in real-world development environments. Moreover, Stack Overflow offers useful information, helpful examples, and possible solutions that are applicable to make static analysis tool more precise. By integrating developers' experience with community wisdom, the tools can be providing more explanations, lower false positives, and offer more effective fixes, which will improve usability and developers' trustworthy.

3. Possible Deficiencies, Pitfalls and Mitigation Strategies

Despite the usefulness of these datasets, they have various limitations that should be taken into the consideration. One concern is the lack of consistency in the use of language, whereby developers use varying terminology to describe issues compared to the bug_type labels that Infer uses. To mitigate this challenge, I can incorporate synonym detection or embedding-based similarity techniques to enhance keyword matching. Another limitation is the existence of noisy data in Stack Overflow as short or irrelevant posts, which may distort results; the issue can be solved by applying filters to filter out low-score or low-content posts. Furthermore, ambiguity is created when similar classes of bugs have varying meanings based on context, so either manual verification or clustering algorithms must be utilized to interpret the data. Outputs generated by Infer also tend to vary across different versions or projects and therefore need standardization of field extraction along with scrutiny of the schema to ensure consistency. Finally, the large size of the XML dataset obtained from Stack Overflow creates challenges on memory handling; this was resolved by taking advantage of Python's iterparse() module that provides low-memory and efficient parsing of large files. Collectively, these approaches guarantee the integrity and usability of the dataset for the targeted research analysis.

Task 4: Refined Problem and Plan

This project aims to investigate of how static analysis warnings, from Microsoft's Infer-based InferredBugs dataset, correspond to real developer issues manifested on Stack Overflow. Preliminary data processing has allowed structured access to numerous bug reports and corresponding forum discussions, thus providing the foundation for additional analysis.

Primary Question 1: To what extent are bug types reported by Infer discussed in Stack Overflow posts, and how frequently do these overlaps occur across different bug categories and severity levels?

Primary Question 2: What kind of responses or solutions do developers receive for posts discussing issues that align with Infer bug types, and how effective or complete are these answers?

These refined questions improve upon earlier ones by foregrounding frequency, representation by category, and quality of answer, which will support both quantitative and qualitative analysis.

Next steps: Data Analysis Plan

To address the two research questions, this project will establish a mixed methodology. That includes quantitative analysis to measure both overlap and popularity. While qualitative analysis will apply to validate the quality of developer responses.

1. **Quantitative Matching and Frequency Analysis**

    - Count the occurrences of each Infer bug_type in Stack Overflow posts (title, body, and tags).
    - Group matched posts by bug type and severity to observe which bugs are discussed the most.
    - Quantify the extent of participation by employing scoring measures such as upvotes and accepted solutions on different bugs.
    - Recognise bug categories with limited developers support which receiving less engagement or no high-quality answers.

2. **Qualitative Answer Quality and Insight Evaluation**

- Choose a representative sample of matching posts for each significant bug class.
- Check for completeness and accuracy of accepted or highly upvoted answers.
- Identify recurring motifs of confusion, typical solutions, or commonly referenced tools or methodologies.
- Use thematic coding to extract explanation quality and educative value.

The integration of these methods will enable the project to evaluate both relevance and importance of community responses to the results found in static analysis tool.

Backup Question and Dataset

If the initial matching strategy fails to provide adequate overlap or coverage, the following project questions and dataset may guarantee the progress of this project:

Backup Question 1: What programming errors are most queried on developer Q&A platforms, and how are these related to tag usage and community engagement?

Backup Question 2: What is the quality of responses, in terms of upvotes and accepted answers, for static analysis tool questions on the Software Engineering Stack Exchange?

Backup Dataset: Software Engineering Stack Exchange is a specialized technical question-and-answer website within the Stack Exchange universe that specializes in development processes and bug examination (Stack Exchange, Inc 2025). The site has advanced discussions on development tools and can offer additional context for bugs found in Infer reports.

Conclusion

The first stage of the project involved selecting appropriate datasets and formulating research questions to investigate the relationship between static analysis warnings and developer conversations. Through initial data processing, a foundation was built for diving into further analysing and evaluating relevance of community response to bugs reports. The next step of the project seeks to deep analysing of both datasets. By examining both quantitative examples and qualitative opinions of Infer-assigned bug categories on Stack Overflow, the goal is to determine whether such tools align with actual developer requirements, what are the limitations that currently exist, and how developers can cultivate this knowledge so that static analysis descriptions are enhanced. A contingency route guarantees that, even if matching is infrequent or imprecise, the project can continue to provide valuable information regarding the linguistic and perceptual gaps between developers and tools.

REFERENCES

Galappaththi, A, Nadi, S & Treude, C 2022, 'Does this apply to me? An empirical study of technical context in Stack Overflow', *Proceedings of the International Conference on Mining Software Repositories*, vol. 19, no. 1, viewed 11 June 2025, pp. 23–34, <https://doi.org/10.1145/3524842.3528435>.

Jin, M, Shahriar, S, Tufano, M, Shi, X, Lu, S, Sundaresan, N & Svyatkovskiy, A 2023, 'InferFix: End-to-End Program Repair with LLMs', *Proceedings of the ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, vol. 31, no. 1, viewed 8 June 2025, pp. 1646–1656, <https://doi.org/10.1145/3611643.3613892>.

Microsoft Research 2025, *InferredBugs – Tools*, Microsoft Research, viewed 9 June 2025, <https://www.microsoft.com/en-us/research/tools/?facet%5Bdate%5D%5Bfixed%5D=any&facet%5Btax%5D%5Bmsr-product-type%5D[]=243083&filter_queries%5B%5D=InferredBugs&pg=1&sort_by=most-relevant>.

Microsoft 2023, *InferredBugs Dataset*, GitHub, <https://github.com/microsoft/InferredBugs/>.

Stack Exchange, Inc 2025, *Stack Exchange Data Dump*, Internet Archive, viewed 12 June 2025, <https://archive.org/download/stackexchange.>.

Vassallo, C, Panichella, S, Palomba, F, Gall, H & Zaidman, A 2020, 'How developers engage with static analysis tools in different contexts', *Empirical Software Engineering*, vol. 25, no. 2, viewed 8 June 2025, pp. 1419–1457, <https://doi.org/10.1007/s10664-019-09750-5>.