Study of Assertions: Understanding Assertion Use in Java Projects on GitHub

Bhavya Chopra (bhavya18333@iiitd.ac.in) | Advisor: Dr. Rahul Purandare
Indraprastha Institute of Information Technology Delhi

Motivation

Assertions are **boolean expressions connected to a program point**, that need to evaluate to true during the execution of the program. If the assertion predicate evaluates to false, JVM throws an **Assertion Error**.

Studies have been conducted to report that developers can **detect up to 80% of the bugs** with the use of assertions [4]. However, assertions are generally missing in practice, or poorly authored. We aim to understand the role of assertions from the programmer's perspective, and extend this work subsequently to automatically generate context-aware assertions to aid software developers in authoring Java programs.

Related Work

- Casalnuovo et al. analyse 69 C and C++ projects to show that **assertions have a small effect on reducing the density of bugs** and developers often add asserts to methods they have larger ownership of [1].
- Pavneet Singh and David Lo partially replicate the above study for Java projects and obtain similar findings [2]. Additionally, they conduct an open card-sort study and **identify 8 categories of assertions**, which we leverage and extend for our study.
- Baudry et al. posit that the **use of assertions eases debugging** and developers value their quality over quantity[3].

Data Collection

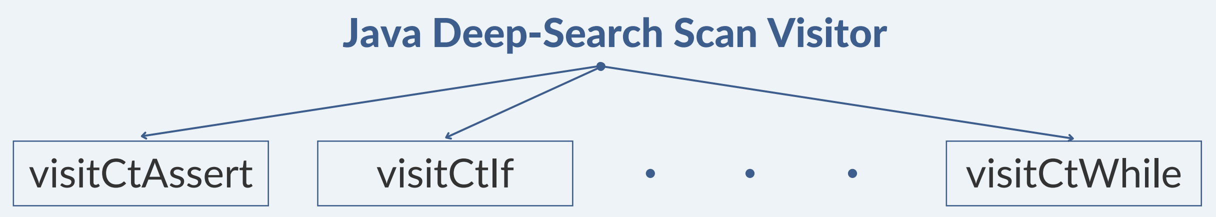
- Cloned **1000 most popular** (most starred) Java repositories on GitHub, extracted metadata using PyGitHub Wrapper.
- Cleaned the dataset by removing any duplicates or forks, and obtained set of 750 repositories, spanning across various software development categories - algorithms, program meta-analysis, development tools, web utility, database management, etc.
- Used **Spoon** (spoon.gforge.inria.fr) to filter out 152 repositories that contained at least one assertion.
- Filtered out 95 repositories that contained at least one assertion not belonging to 'test' files.
- Programmed 8 parsers in Spoon using the **Visitor Design Pattern** to extract information about the subjects.

Research Questions

RQ1: Is there a correlation between code complexity and use of assertions?
RQ2: Does the theme/domain of the project influence the use of assertions?
RQ3: Are assertions added by developers proactively or reactively?
RQ4: At what program points in a method are assertions most frequently seen?
RQ5: Which variables are evaluated by the assertion predicate at these program points?

Methodology

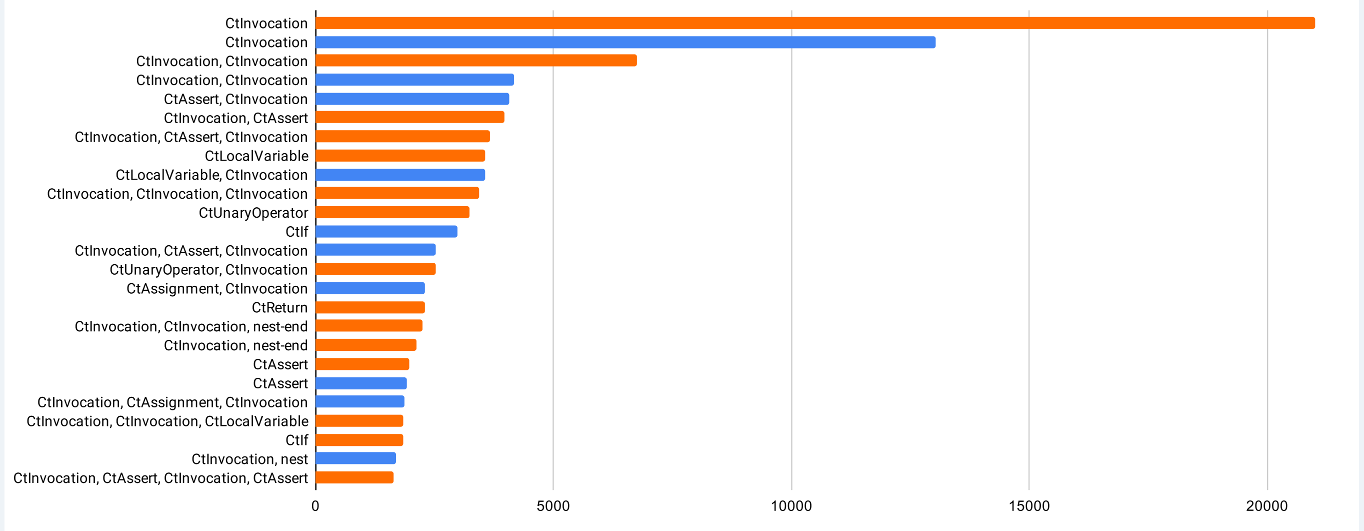
- Developed AST traversers using the **Visitor Design Pattern**, extending the **CtScanner class in Spoon**. Used to extract information about the programs for all RQs - SLOC, Cyclomatic Complexity, assert statement locations, representing methods as sentences, and variable properties.



- Performed **qualitative analysis** of a statistically representative number of methods with assertions for RQs 2, 3, 4 and 5 with three annotators. We calculated **inter-rater agreement** scores and proceeded with annotation after receiving high-almost perfect agreement (Scores between 0.81-0.95)
- Employed **Hurdle Poison Regression** and **Hurdle Negative Binomial Regression** models for analysis due to excessive 0's in quantitative data.

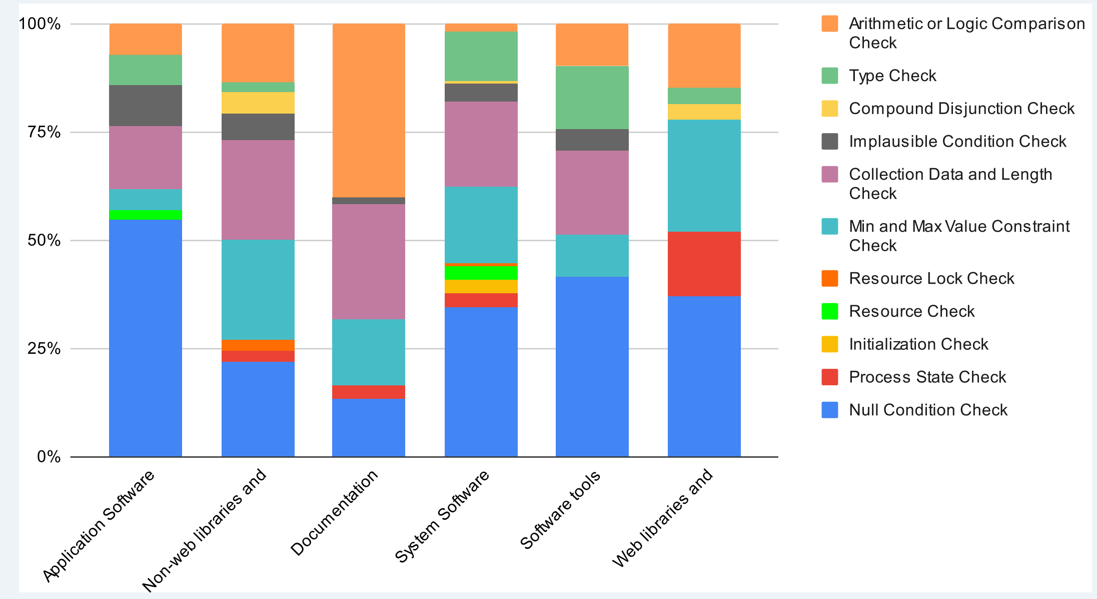
Findings: Locations of Assertions

The following graph represents most frequent statements as n-grams immediately preceeding (blue columns) and immediately succeeding (orange bars) the assert statements. Please visit **bit.ly/rq4-result** for qualitative insights.



Findings: Use of Assertions

- **Hurdle component** models the effect of going from 0 to 1 assertion, and **Count component** models effect of going from non-zero value to another non-zero value.
  - We find that SLOC is **positively correlated** for the hurdle component, & **negatively** for the count component.
  - Cyclomatic Complexity and number of comments are **positively correlated** for the hurdle & count components.
- For RQ 2, we categorized projects into domains and assertions into types. We obtain a **varying distribution of assertions across project domains** in Java:



- For RQ 3, we find that **7.64% assertions are reactive**, whereas **92.35% assertions are proactive**.
  - Reactive assertions are added for making bug-fixes.
  - Proactive assertions are added while rolling out new features, to ensure backward compatibility, for performing code optimization, & for documentation purposes.

Future Work

- Performing backward flow data dependence analysis, considering assertion as source statement to analyse variables present in the assertion predicates (RQ 5).
- Forming a heuristics and learning based approach to predict program points for adding assertions and variables to be evaluated.

References

[1] Casalnuovo, C., Devanbu, P., Oliveira, A., Filkov, V., and Ray, B. Assert use in github projects. In Proceedings of the 37th International Conference on Software Engineering - Volume 1 (2015), ICSE '15, IEEE Press, p. 755–766.
[2] Kochhar, P. S., and Lo, D. Revisiting assert use in github projects. In Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering (New York, NY, USA, 2017), EASE'17, Association for Computing Machinery, p. 298–307.
[3] Baudry, B., Le Traon, Y., and Jezequel, J.-M. Robustness and diagnosability of OO systems designed by contracts. In Proceedings Seventh International Software Metrics Symposium (2001), pp. 272–284.
[4] Briand, L. C., Labiche, Y., and Sun, H. Investigating the use of analysis contracts to support fault isolation in object oriented code. SIGSOFT Softw. Eng. Notes 27, 4 (July 2002), 70–80.