Java Code Readability Study

Lukas Krodinger

Master Thesis Proposal in M.Sc. Computer Science
Faculty of Computer Science and Mathematics
Chair of Software Engineering II

Matriculation number	89801
Supervisor	Prof. Dr. Gordon Fraser
Advisor	Lisa Griebl

21st October 2023

1 RESEARCH OBJECTIVE

The survey aims to evaluate the accuracy of Java code snippet readability predicted with heuristics. The first part of the code snippets are selected from GitHub repositories with assumed high code quality. The second part is generated by Readability Decreasing Heuristics. That is, heuristics are used to manipulate the code snippets of the first part to make them less readable. The main objective of the survey is to validate the following two assumptions:

1. **Assumption 1 (well-readable-assumption)** The selected repositories contain only well readable code.
2. **Assumption 2 (poorly-readable-assumption)** After the manipulations, the code is poorly readable.

The survey is conducted by the Chair of Software Engineering II[1] of the University of Passau under the supervision of Lukas Krodinger. Survey participants are selected using Prolific[2], and conducted online using Tien Duc Nguyen's Code Annotation Tool. This survey summary was prepared following the guidelines of Linåker et al. [4].

2 TARGET POPULATION

The target population is split into two groups:

- **Students**: Computer science students with Java experience
- **Programmers**: Java programmers in industry

To be more precise, the target population of the survey is as follows:

Target Audience	Java programmers
Unit of Observation	Java programmers
Unit of Analysis	Java programmers
Search Unit	Selected by Prolific (Programming Languages: Java)
Source of Sampling	Prolific

3 SURVEY

The first part of the survey is devoted to Prolific-specific questions and questionnaire attributes, as described in Section 3.5. The second part of the survey consists of 20 code snippets to be evaluated by the participants. In total, we

[1]https://www.fim.uni-passau.de/en/chair-for-software-engineering-ii/
[2]https://www.prolific.co/

create 20 unique questionnaires, with each questionnaire being rated by 10 individual participants. Consequently, we generate a dataset of 400 code snippets. The expected expenditure for this survey, involving 200 participants via the Prolific platform, is estimated to be around €500.

3.1 SAMPLE SIZE

To get accurate study results, multiple participants have to rate the same code snippets. We specify a size of ten participants per code snippet, similar to Scalabrino et al. [5]. According to an online calculator[3] we end up with a margin of error of 33.99% at a confidence of 95%. This means that we are 95% confident that the actual readability of a code snippet is within 33.99% of the survey result score. With a probability of 5%, the readability of a code snippet differs by more than 33.99% from the score of the survey result. On a Likert scale ranging from 1 to 5, the mean of the survey results is between 1.0 and 5.0. In this context, an offset of 33.99% is an offset of more than one rating point [3]. We need to consider this when evaluating our the gathered data.

3.2 TIME REQUIREMENT

The first part (Prolific-specific questions and questionnaire attributes) takes the participant about one minute. The average time for rating a snippet is estimated with 30 seconds. Thus, rating 20 snippets takes about 10 minutes. The total time for completing one questionnaire is estimated with 11 minutes.

3.3 SAMPLING DESIGN

The survey participants are selected using Prolific. The only restriction for the participants is that they must be familiar with Java.

A big part of our code snippets might be getters and setters. However, we do not want to mostly evaluate getters and setters in our study. Therefore, we use stratified sampling [6]. Thus, code snippets are split into groups with high similarity, so-called strata. When we then randomly sample within the strata, we make sure to not only sample getters and setters.

To create the required stratas we need to measure the similarity of code snippets. Scalabrino et al. developed a tool to generate various metrics for Java source code [5]. We use this tool on each code snippet and create a code vector for each snippet. The Euclidean distance between these vectors provides an estimation of source code similarity, which is used to create the strata.

[3]https://www.calculator.net/sample-size-calculator.html

After sampling downloaded code snippets this way, we also want to include the same snippets after certain manipulations with a certain frequency are applied. Multiple snippets from the same source snippet (including the source snippet itself) are not in the same questionaire, so that a user can not compare the snippets to each other.

After sampling the downloaded code snippets in this way, we want to include the same snippets after performing manipulations. There are several variants with differences in type and frequency for each downloaded snippet. Multiple snippets from the same source snippet (including the source snippet itself) are not in the same questionnaire. Thus, a user cannot compare the snippets with each other. This does not diminish the inferences that can be made about all the data based on stratified sampling. However, conclusions can be drawn about the effectiveness of different heuristics.

3.4 INTERNAL QUESTIONS

The internal research questions are as follows:

- Does the well-readable-assumption (Assumption 1) hold?
- Does the poorly-readable-assumption (Assumption 2) hold?

To answer these questions, the assumptions are considered as hypotheses along with the following associated null hypotheses:

1. **For Assumption 1**: The mined code exhibits a normal distribution of readability scores.
2. **For Assumption 2**: The readability of code does not significantly deteriorate compared to the original code snippet.

The results for these questions are equally important, and thus none of them is prioritized over the other.

Another goal of the study is to find estimates of which heuristics need to be applied how often to make the code less readable to some degree. This helps us to further adjust the settings of our generation approach for poorly readable code.

Note that the goal of our study is not to label code snippets by readability, but to validate our assumptions. If the assumptions are confirmed, we can use them to make inferences about the readability of all code snippets generated with our approach. This is a crucial difference from comparable other studies [1, 2, 5].

3.5 QUESTIONNAIRE ATTRIBUTES

The survey neither contains demographic questions nor filter questions. Besides the readability questions, each user is asked the following dependent question: "How would you describe your familiarity with Java?". The user can answer within a five point Likert scale: expert (5), advanced (4), intermediate (3), beginner (2), novice (1).

3.6 SURVEY QUESTIONS

For each code snippet, the following closed-ended question is given: "How do you rate the readability of this code?". The answers follow the Likert Scale: very high (5), high (4), medium (3), low (2), very low (1).

3.7 SURVEY RESULT EVALUATION

Once a survey is completed by a participant, the survey result is evaluated to make sure the participants did not choose answers at random. The code snippet rating answers are checked for plausibility. For example, if a code snippet with expected rating 5 (by the heuristics and/or by other participants) is rated with 2 or less, this is an indication that the participant did not fill out the survey carefully. Several such indications lead to the exclusion of the participants and their answers.

4 SOFTWARE

We use Prolific to pay participants for completing our survey. Our participants fill out the survey using Tien Duc Nguyen's Code Annotation Tool.

5 SURVEY EVALUATION

We evaluate the survey construction by presenting it to the following groups:

- Subject-matter experts
- Focus group (discussion with about 7 people)
- Pilot survey group

We are conducting the survey iteratively. We start with three questionnaires and expand the number based on our conclusions from the first ones. This offers several advantages. Firstly, we can still adjust the survey if problems arise. Secondly, we can start building our pipeline for evaluation earlier. Finally, after each iteration, we can decide whether to include more variants of the same source code snippet or to increase the number of source code snippets instead (see Section 3.3). The first option could give us more insights on how to adjust the settings for our heuristics. The second option could increase the number of stratas involved or allow us to increase the evidence for a particular stratum by including more samples of the respective one.

6 THREADS

This section addresses potential threats to the survey's validity and reliability. These threats include:

1. **Ill-defined Target Population:** Ensuring a well-defined target population is critical to the survey's quality. To mitigate this threat, we clearly define our target population within this document (see Section 2). Additionally, we conduct a pre-survey evaluation (see Section 5) to ensure the adequacy of our target population definition. Thereby, we enhance content and construct validity.
2. **Sampling Method (Stratified Sampling):** Our chosen sampling method is well-defined and proven in practice. This approach ensures that our sample represents all parts of the population under investigation. This is improving the survey's external validity.
3. **Insufficient Responses for Drawing Conclusions:** To prevent drawing conclusions from an insufficient number of responses, we scale our survey to an appropriate size. This guarantees that we collect a substantial volume of responses, allowing for robust statistical analysis.

7 SCHEDULE

You can find the survey schedule in Table 1.

Task	**Timeline (finished latest)**
Survey concept proposal	October 2023
Presentation to survey evaluation groups	November 2024
Survey conduction	December 2023
Evaluation of results	January 2024

Table 1: Survey Timeline

8 EXPECTED RESULTS

We expect that the averaged ratings of the selected code snippets matches the predicted code readability to a certain extent. We do not expect an exact match. We measure this extent by applying the mean as a measure of central tendency and standard deviation as a measure of variability, as proposed by Linåker et al. [4]. We make those measurements with respect to stratified sampling.

The primary objective of this study is to validate a novel dataset. It aims to demonstrate its suitability for training readability classifiers, despite its heuristic construction. The study seeks to validate our assumptions (Assumption 1 and 2). After conduction, we visualize our results by generating car charts. We also provide a visualization that summarizes the mean deviations of participants' ratings from expected readability for all snippets.

Bibliography

[1] Raymond PL Buse and Westley R Weimer. ‘Learning a metric for code readability’. In: *IEEE Transactions on software engineering* 36.4 (2009), pp. 546–558.

[2] Jonathan Dorn. ‘A General Software Readability Model’. In: 2012. URL: `https://api.semanticscholar.org/CorpusID:14098740`.

[3] Rensis Likert. ‘A technique for the measurement of attitudes.’ In: *Archives of psychology* (1932).

[4] Johan Linåker et al. ‘Guidelines for Conducting Surveys in Software Engineering’. In: (May 2015).

[5] Simone Scalabrino et al. ‘A comprehensive model for code readability’. In: *Journal of Software: Evolution and Process* 30.6 (2018), e1958.

[6] Steven K Thompson. *Sampling*. Vol. 755. John Wiley & Sons, 2012, pp. 141–156.