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

2nd October 2023

1 RESEARCH OBJECTIVE

The survey aims to evaluate the accuracy of with heuristics predicted readability of Java code snippets. We define readability as a subjective impression of the difficulty of code while trying to understand it [2, 1]. The first part of the code snippets are selected from GitHub repositories with known high code quality. The second part is generated by Readability Decreasing Heuristics. Therefore, code snippets of the first part are manipulated to become less readable using heuristics. The survey goal is to verify the following two assumptions:

1. **(well-readable-assumption)** The selected repositories contain only well readable code.
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. The survey will be conducted using Google Forms[2] and Prolific[3].

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 survey will consist of three pages. The first page will contain Prolific specific questions and the questionnaire attributes (see section 3.5). The other pages will have ten code snippets each to be rated by the participant (20 in total). We will

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

create multiple survey sheets with different snippets to evaluate more than 20 in total.

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 citeauthor, cite scalabrio. According to an online calculator[4] 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 1.3596 from the real value, which is more than one rating point. We need to consider this when evaluating the accuracy of our assumptions (assumption 1 and 2).

3.2 TIME REQUIREMENT

The first page will take the participant about one minute. As one does not need to fully understand the code snippet in order to rate its readability, the average time for doing so is estimated with 30 seconds per snippet. Rating two pages with ten snippets each will take about 10 minutes. The total time for the survey is estimated with 11 minutes.

3.3 SAMPLING DESIGN

The survey participants will be selected using Prolific. The only restriction for the participants will be 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 will use stratified sampling. 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. Scalabrio cite autor cite developed a tool to generate various metrics for Java source code. We will 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 will be used to create the strata.

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

3.4 INTERNAL QUESTIONS

The internal research questions are as follows:

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

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

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 automatically 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 properly.

4 SOFTWARE

We will use Prolific to pay participants for completing our survey. Our participants will fill out the survey using TODO: ADD TOOL.

5 SURVEY EVALUATION GROUPS

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

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

6 THREADS TO VALIDITY AND RELIABILITY

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	November 2023
Presentation to survey evaluation groups	January 2024
Survey conduction	February 2024
Results evaluation	March 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 X. We make those measurements with respect to stratified sampling. Our null hypothesis is, that both assumptions (1 and 2) hold, as the expected readability does not differ too much from the survey participant's mean. We will visualize our results by generating a bar chart for each evaluated snippet. We will also provide a visualization that summarizes the mean deviations of the participant's rating from the expected readability valued, summarized 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] Daryl Posnett, Abram Hindle and Premkumar Devanbu. 'A simpler model of software readability'. In: *Proceedings of the 8th working conference on mining software repositories*. 2011, pp. 73–82.