An online appendix for the paper "MLCQ: Industry-relevant code smell data set" by Madeyski and Lewowski (MLCQ: Datasheet)

Tomasz Lewowski*, Lech Madeyski†

February 13, 2020

1 Data set descriptors

1.1 Data set identity

Data set of code smells and reviewers' experience survey.

1.2 Data set identification code

The data set on Zenodo is available using `https://doi.org/10.5281/zenodo.3590101` upon acceptance of the related data paper. This DOI will always resolve to the newest version of the data set. The data set will be available in the subsequent version of the `reproducer` R package on CRAN as well [2].

1.3 Data set description

1.3.1 Originators

The research was initiated as part of NCBR POIR 01.01.01-00-0792/16 research grant.

1.3.2 Abstract

This data set contains samples from software projects together with review done by developer that states whether the sample represents a given code smell severity, ranging from none to critical. There are nearly 15 000 samples of 4

*Faculty of Computer Science and Management, Wroclaw University of Science and Technology, ORCiD: 0000-0003-4897-1263, email: tomasz.lewowski@pwr.edu.pl

†Faculty of Computer Science and Management, Wroclaw University of Science and Technology, ORCiD: 0000-0003-3907-3357, email: lech.madeyski@pwr.edu.pl

code smells in total, acquired by 26 reviewers. The data set contains also surveys filled in by the reviewers that describe their experience in detail. There are 20 surveys, 6 reviewers did not complete the survey.

1.4 Key words

Code smells, Bad smells, Java, Software Engineering, Data set

2 Code smell data description

2.1 Site description

Participants were reviewing code smells at their own pace and in their own habitats, without any external supervision. Decision if and when to review code samples was entirely up to the reviewer.

2.2 Sampling design

Code samples were generated from Java projects selected from GitHub. Used project data set is described in [1]. We did not apply any manual filters and all samples from all 792 projects were used.

In total we gathered 14739 reviews of 4770 code samples, 8040 reviews of classes (2340 distinct classes from 437 projects) and 6699 reviews of functions (2430 functions from 426 projects). Reviews were done by 26 developers, 20 of which have completed the survey described in Section 5.2.

Sample selection was not uniform during data acquisition, and there were four phases. In the first two phases, we sampled from 678278 classes and 5101141 functions from 792 projects (standard deviation for number of samples was 15738 samples), while in the last two - from 552750 classes and 2297722 functions from 785 projects:

1. The first 2175 samples (date range: 27.03.2019-02.04.2019) were selected when there was a defect in selection query, and only 1 sample from each hundred could have been selected. Nevertheless, they were still selected randomly, but from a smaller population, and inserted randomly into database—therefore we decided to leave those as potentially valuable samples.

2. The next 2648 samples (date range: 03.04.2019-12.04.2019) were selected randomly from all available.

3. The next 4801 samples (date range: 13.04.2019-25.07.2019) were selected randomly from all samples longer than 4 lines[1]. This was the only filtering rule, and it was only removing most trivial functions and classes. This filter was suggested by code reviewers and the aim was not to waste the

[1] including opening and closing braces

precious time of developers and to better focus their effort on reviewing potentially smelly code samples.

4. The last 5115 samples (date range: 26.07.2019-13.09.2019) were selected to perform a crosscheck—only from samples already graded to a severity higher than 'none'. Samples selected from crosscheck were those that had only 1 review or 2 reviews with different severities. A developer could not crosscheck his own grade.

2.3 Research methods

2.3.1 Instrumentation

A new tool was developed specifically to address the need of creating this data set. This tool was running on AWS cloud and, as of today, is not published. It was used to generate all samples from selected projects and later on to select samples for review.

2.3.2 Legal requirements

This research might have been concerned with two types of laws:

- privacy—to address any possible issues here we anonymize the data before publishing,
- copyright—we received agreement from the funding company and use only open source projects with permissive licenses

3 Data set status

3.1 Status

3.1.1 Latest update

Last survey completed 05.07.2019,

Last code sample reviewed 13.09.2019.

3.1.2 Latest archive date

The data set was archived on 13.02.2020.

3.1.3 Data verification

CSV and XLSX file SHA-256 checksums:

MLCQCodeSmellDevelopersSurvey.csv
a19c80b405cbe77499ca074e4a0048ab9ffeb9af731a15809e38d1a95871238c

MLCQCodeSmellDevelopersSurvey.xlsx
5bc9c15109f08501538d896536bd93ca252d9cdbf8f81b1f3913e03cf3d3e7d6

MLCQCodeSmellSamples.csv
8e05db55150a984f0b08ab211f5f612374790ea918ba08afc290cf122b63551b

MLCQCodeSmellSamples.xlsx
18cee0ce663094ba4fb818dc7a3df492002247e1cacb47e68cbe5883a8bdfdea

3.2 Accessibility

3.2.1 Storage information

The data set is primarily stored on Zenodo for the purpose of review. It will be available from the `r-reproducer` R package [2] on CRAN upon acceptance of the data paper [3].

3.2.2 Contact persons

Tomasz Lewowski ORCiD: 0000-0003-4897-1263, email: tomasz.lewowski@pwr.edu.pl

Lech Madeyski ORCiD: 0000-0003-3907-3357, email: lech.madeyski@pwr.edu.pl

3.2.3 Copyright restrictions

4 Code smells data set structural description

4.1 Data set file

4.1.1 Identity

Code samples together with reviews for specific code smells linked with reviewers.

4.1.2 Files

CSV file name `MLCQCodeSmellSamples.csv`

XLSX file name `MLCQCodeSmellSamples.xlsx`

4.1.3 Size

Number of reviews 14739

Number of reviewed smells 4

Number of reviewers 26

Number of reviews with 'none' severity 11448

Number of reviews with 'minor' severity 1787

Number of reviews with 'major' severity 1129

Number of reviews with 'critical' severity 375

Table 1: Number of reviews per severity per smell

Code smell	#reviews	#critical	#major	#minor	#none
Blob	4019	127	312	535	3045
Data Class	4021	146	401	510	2964
Long Method	3362	78	274	454	2556
Feature Envy	3337	24	142	288	2883

4.1.4 Format and storage mode

CSV file in UTF-8 encoding. Fields are separated by semicolons. We also include an XSLX (Office Open XML) file with the same data.

4.2 Variable information

Each of the records (reviewed samples) contains the following information:

id a numeric identifier of the review,

reviewer_id a numeric identifier of the reviewer,

smell a name of the code smell (*Blob*, *Data Class*, *Feature Envy*, *Long Method*),

severity severity of the code smell (*critical*, *major*, *minor*, and *none*),

review_timestamp date and time (millisecond precision) when the sample was acquired,

type whether the reviewed code sample is a class or a function,

code_name a fully qualified name of the code sample – format: $Package.ClassName[\#FunctionName \quad arg1$ (e.g., $org.eclipse.swt.widgets.Menu\#setLocation \quad int|int$), in case of constructors and static methods a dot is used instead of a hash.

repository a git url of the repository,

commit_hash SHA checksum of repository revision that the sample was acquired at,

path path in the repository that can be used to retrieve the sample,

start_line line in the file in which the sample starts,

end_line line in the file in which the sample ends,

is_from_industry_relevant_project denotes whether source project was classified as industry-relevant in [1],

link a link that can be used to view the sample in a browser.

5 Survey data set structural descriptors

5.1 Data set file

5.1.1 Identity

Survey results from code smell reviewers

5.1.2 Files

CSV file name `MLCQCodeSmellDevelopersSurvey.csv`

XLSX file name `MLCQCodeSmellDevelopersSurvey.xlsx`

5.1.3 Size

20 records, 55 questions each. The survey contained 59 questions. Four of them were related to acquisition of required agreements and contact information, thus were removed from the final data set. In section 5.2 we present questions and possible answers (for multi-choice questions). All questions were optional and participant could decide to skip any of them. Every question with a possible "other" answer accepted also a text input describing what exactly does "other" mean.

5.1.4 Format

CSV file in UTF-8 encoding. Fields are separated by commas. We also include an XSLX (Office Open XML) file with the same data.

5.1.5 Acquisition

The data was acquired using Typeform[2] system for online surveys. Then it was exported using Typeform built-in export facilities.

[2] www.typeform.com

5.2 Questions

1. Which is the highest level of formal education that you graduated?
    - Secondary school (or lower)
    - Bachelor of Science (BSc) / Bachelor of Engineering (BEng)
    - Other Bachelor-level
    - Master of Science (MSc) / Master of Engineering (MEng)
    - Other Master-level
    - Doctoral (Ph.D.) or higher
2. How long professional experience (in any profession) do you have? (¡A¿Y¡B¿M)
3. How long do you code? (Please include also coding for fun)
4. How long professional experience in software development do you have? (Please include programming, testing, administration, management etc.)
5. How long professional experience in programming do you have?
6. How long professional experience in programming in object-oriented paradigm do you have?
7. How long professional experience in programming in functional paradigm do you have?
8. How long professional experience in programming in JVM-based languages do you have?
9. How long professional experience in programming in Java do you have?
10. In which country did you spend most of your programming career?
11. Which company are you currently working for?
12. Which programming languages did you use throughout your career?
    - Assembly
    - C
    - C++
    - C#
    - Clojure
    - COBOL
    - Delphi
    - Elixir
    - Erlang

- Go
- Groovy
- Haskell
- Java
- JavaScript
- Kotlin
- Matlab
- Objective-C
- Perl
- PHP
- Python
- R
- Ruby
- Scala
- Smalltalk
- SQL
- Swift
- TypeScript
- Rust
- Visual Basic
- Visual Basic .NET
- Other

13. Which of those tools did you ever use?

    - A compiler
    - A linter
    - Model checker
    - Proof assistant
    - A static analysis tool (e.g., FindBugs)
    - A defect prediction tool
    - Fuzzer
    - Mutation tester

14. Which of those industries did you work in?

    - Banking / Finance
    - Telecommunication

- Retail
- Gaming
- Entertainment
- Medicine
- Manufacturing
- Government
- Academia
- Food
- Automotive
- Consulting
- Other

15. Which of those roles have you held throughout your professional career?

    - Junior Software Developer/Engineer
    - Software Developer/Engineer
    - Senior/Lead Software Developer/Engineer
    - Data Scientist
    - Project Manager
    - Other management role
    - Designer
    - Business Analyst / Requirements Engineer
    - Software Tester
    - Hardware Tester
    - Operations Engineer / System Administrator
    - Architect
    - Other

16. Which of those are your current roles?

    - Student - Bachelor level
    - Student - Master level
    - Student - PhD level
    - Junior Software Developer/Engineer
    - Software Developer/Engineer
    - Senior/Lead Software Developer/Engineer
    - Data Scientist
    - Project Manager

- Other management role
- Designer
- Business Analyst / Requirements Engineer
- Software Tester
- Hardware Tester
- Operations Engineer / System Administrator
- Architect
- Lecturer

17. When were you last involved in programming activities (in any programming language)?

    - I am still involved in them
    - In the last year
    - A year or two ago
    - Somewhere between two and five years ago
    - More than five years ago

18. Did you work in projects of following sizes (counting all people involved, not only developers)

    - >1 MLoC or >50 people
    - >250 kLoC or > 20 people
    - >100 kLoC or > 10 people
    - >50 kLoC or >5 people
    - >1 person
    - 1 person

19. How long does an average project you work on take?

    - Less than 6 months
    - Between 6 months and 1 year
    - Between 1 and 3 years
    - Over 3 years

20. Do you write code as a hobby?

    - Yes
    - No

21. Do you prefer to work in the office or remotely?

    - In the office

- Remotely
- Doesn't matter

22. Do you contribute to open source projects?

    - Yes, I write code or documentation
    - Yes, I create issues
    - Yes, I'm involved in forums/discussions
    - No, I don't
    - Other

23. Which of those programming paradigms do you use during programming?

    - Procedural
    - Object-oriented
    - Functional
    - Logical
    - Other

24. When you choose a language for new project, which reasons do you take into consideration?

    - Availability of libraries
    - Your own skillset
    - Skillsets of your team
    - Toolchain complexity
    - Tool availability (e.g., static analysis, security checks etc.)
    - Current industry trends
    - Community support
    - Learning curve
    - Other

25. Are you familiar with the concept of code reviews?

    - Yes, I know about them and use them
    - Yes, I know about them but do not use them
    - Not really

26. How long have you been conducting regular code reviews for your peers?

    - I don't conduct code reviews
    - Less than half a year
    - Between half a year and two years

- More than two years

27. Code in which of those languages would you feel comfortable reviewing?
    - Assembly
    - C
    - C++
    - C#
    - Clojure
    - COBOL
    - Delphi
    - Elixir
    - Erlang
    - Go
    - Groovy
    - Haskell
    - Java
    - JavaScript
    - Kotlin
    - Matlab
    - Objective-C
    - Perl
    - PHP
    - Python
    - R
    - Ruby
    - Scala
    - Smalltalk
    - SQL
    - Swift
    - TypeScript
    - Rust
    - Visual Basic
    - Visual Basic .NET
    - Other
28. Are you familiar with the concept of code smells?

- Yes, I know what they are
- Somewhat - I heard about them and maybe used some tools, but never dug deeper
- Not really - I heard about them, but never cared much
- Not at all

29. Which of the following code smells do you believe you can recognize?
    - Blob / God Class
    - Long Method
    - Data Class
    - Feature Envy
    - Blue Box
    - Inappropriate Intimacy
    - Shotgun Surgery
    - Swiss Army Knife
30. Do you use names of code smells (Blob, Shotgun Surgery etc.) during discussions with your peers?
31. Do you feel that code smells in traditional sense (Feature Envy, Blob, Shotgun Surgery) make the code less maintainable?
    - Definitely, that's why the concept was invented
    - I'd assume so, but in my experience I hardly ever find this a problem
    - Not really, there are many worse problems in the code
32. How big problem is Blob (1 - not at all, 5 - significant)? Please don't answer if you're not familiar with this code smell
33. How big problem is Data Class (1 - not at all, 5 - significant)? Please don't answer if you're not familiar with this code smell
34. How big problem is Feature Envy (1 - not at all, 5 - significant)? Please don't answer if you're not familiar with this code smell
35. How big problem is Long Method (1 - not at all, 5 - significant)? Please don't answer if you're not familiar with this code smell
36. How big problem is Shotgun Surgery (1 - not at all, 5 - significant)? Please don't answer if you're not familiar with this code smell
37. How big problem is Inappropriate Intimacy (1 - not at all, 5 - significant)? Please don't answer if you're not familiar with this code smell
38. How big problem is Refused Bequest (1 - not at all, 5 - significant)? Please don't answer if you're not familiar with this code smell

39. Do you feel that a catalogue of code smells may be something useful to you?

    - Yes, I'd gladly use it
    - Not sure, maybe I'd take a glance or two
    - Not really, I don't like this quasi-formal approach

40. Do you feel that concept of code smell can be transferred between programming languages?

    - Not at all
    - Only between very similar languages
    - In a family of languages, with minor adjustments
    - Yes, but it may no longer be meaningful

41. Which factors do you believe are relevant when assessing if a given code fragment is a code smell?

    - Its complexity
    - Its change proneness
    - Its heterogenity
    - Language conventions and restrictions
    - Its size
    - Project conventions and restrictions
    - Its age
    - How well it is tested
    - Its dependencies
    - Other

42. What is your opinion about the concept of code smells?

    - This is a useful concept, that simplifies talking about a problem, but we should not stick to predefined names
    - It's perfect
    - It's just noise
    - The concept itself is useful, but it's an oversimplification (e.g. doesn't include industry and other project-specific needs)

43. What is the biggest problem you encounter in code written by other people?

44. What is the biggest problem you encounter in your code after some time?

45. Which problem is the most frustrating one that you encounter during programming?

46. What are the biggest problems in code that you encounter?

    - Rigid structures (hard to adapt when requirements change)
    - Functional defects
    - Fragile structures (changing very often)
    - Cascading changes
    - Untested code
    - Code not adhering to common style
    - Undocumented code
    - Other

47. Which quality assurance techniques do you perceive as useful?

    - Unit testing
    - Integration/system testing
    - Mutation testing
    - Performance testing
    - Linting
    - Static analysis (e.g. model checking, symbolic execution)
    - Program proving
    - Fuzzing
    - Code review
    - Other

48. Which quality assurance techniques do you use in your daily work?

    - Unit testing
    - Integration/system testing
    - Mutation testing
    - Performance testing
    - Linting
    - Static analysis (e.g. model checking, symbolic execution)
    - Program proving
    - Fuzzing
    - Code review
    - Other

49. Which elements should, in your opinion, be part of code review?
    - Verification of correctness of test cases
    - Manual application execution
    - Code style checking
    - Checking for forbidden patterns (e.g. linters)
    - Manual change correctness assessment
    - Checking for typical code smells
    - Other
50. Do you believe that reviews contribute to code quality?
    - Yes, positively
    - Yes, negatively
    - No
51. Which of the following should be provided by automatic static analysis tool?
    - Code style checking
    - Checking for forbidden patterns (e.g. linters)
    - Checking for common high-level antipatterns
    - Defect prediction
    - Other
52. How important is customization for static analysis?
    - Critical, I always customize rules and often change them
    - Quite important in the beginning, but once best practices for language and organization are settled, they don't change much
    - Not very important - the rules should be given by the toolmaker and only modified rarely
53. How do you feel about current state-of-the-art static analysis tools?
    - They're great
    - They could be better
    - They provide only basic features
    - They're worthless
    - I don't use any
54. What do you believe to be the biggest problem of static analysis tools?
    - Too many false positives

- Too obvious problems
- Most useful errors are raised by compiler anyway
- Too long execution
- Too resource-consuming

55. Please describe your earlier experience with static analysis and linting tools

References

[1] Tomasz Lewowski and Lech Madeyski. Creating Evolving Project Data Sets in Software Engineering. In Stanislaw Jarzabek, Aneta Poniszewska-Marańda, and Lech Madeyski, editors, *Integrating Research and Practice in Software Engineering*, volume 851 of *Studies in Computational Intelligence*, pages 1–14. Springer, Cham, 2020.

[2] Lech Madeyski, Barbara Kitchenham, and Tomasz Lewowski. *reproducer: Reproduce Statistical Analyses and Meta-Analyses*, 2020. R package (http://CRAN.R-project.org/package=reproducer).

[3] Lech Madeyski and Tomasz Lewowski. MLCQ: Industry-relevant code smell data set. (submitted).