MLCQ: Industry-Relevant Code Smell Data Set

Lech Madeyski*
Faculty of Computer Science and Management, Wroclaw University of Science and Technology
Wyb.Wyspianskiego 27, 50370 Wroclaw, Poland
lech.madeyski@pwr.edu.pl

Tomasz Lewowski
Faculty of Computer Science and Management, Wroclaw University of Science and Technology
Wyb.Wyspianskiego 27, 50370 Wroclaw, Poland
tomasz.lewowski@pwr.edu.pl

ABSTRACT

Context Research on code smells accelerates and there are many studies that discuss them in the machine learning context. However, while data sets used by researchers vary in quality, all which we encountered share visible shortcomings—data sets are gathered from a rather small number of often outdated projects by single individuals whose professional experience is unknown.

Aim This study aims to provide a new data set that addresses the aforementioned issues and, additionally, opens new research opportunities.

Method We collaborate with professional software developers (including the *code quest* company behind the `codebeat` automated code review platform integrated with GitHub) to review code samples with respect to bad smells. We do not provide additional hints as to what do we mean by a given smell, because our goal is to extract professional developers' contemporary understanding of code smells instead of imposing thresholds from the legacy literature. We gather samples from active open source projects manually verified for industry-relevance and provide repository links and revisions. Records in our MLCQ data set contain the type of smell, its severity and the exact location in source code, but do not contain any source code metrics which can be calculated using various tools. To open new research opportunities, we provide results of an extensive survey of developers involved in the study including a wide range of details concerning their professional experience in software development and many other characteristics. This allows us to track each code review to the developer's background. To the best of our knowledge, this is a unique trait of the presented data set.

Conclusions The MLCQ data set with nearly 15000 code samples was created by software developers with professional experience who reviewed industry-relevant, contemporary Java open source projects. We expect that this data set should stay relevant for a longer time than data sets that base on code released years ago and, additionally, will enable researchers to investigate the relationship between developers' background and code smells' perception.

*Corresponding author.

EASE 2020, April 15–17, 2020, Trondheim, Norway

ACM ISBN 978-1-4503-7731-7/20/04.
https://doi.org/10.1145/3383219.3383264

CCS CONCEPTS

• **Software and its engineering** → *Software organization and properties*; *Software creation and management*;

KEYWORDS

data set, code smells, bad code smells, software development, software quality

ACM Reference Format:
Lech Madeyski and Tomasz Lewowski. 2020. MLCQ: Industry-Relevant Code Smell Data Set. In *Evaluation and Assessment in Software Engineering (EASE 2020), April 15–17, 2020, Trondheim, Norway.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3383219.3383264

1 INTRODUCTION

Code smells are an important concept for software maintainability (e.g., [15, 17]) and even defect prediction (e.g., [14]). However, their traditional definitions are rather vague and cannot be easily applied in automated analysis. Moreover, the evolution of programming languages (e.g., combining object-oriented and functional programming paradigms in a single programming language) and the fact that the concept of code smells was introduced over 20 years ago may require that we revise some of the original ideas and perceptions.

Our overall research goal is to understand the meaning of specific code smells (what is not and what is a smell, as well as how severe it is) in the way that experienced software developers do in real, contemporary software projects—projects that, according to our earlier research [7], either are or can be used in commercial setting. This research backs the development of the `codebeat`[1] tool by the `code quest` company. The tool is freely available to open source projects and also used by many commercial projects. The goal of this tool is to provide sort of automated reviews—project-specific guidelines as to which parts of source code should be modified, possibly also how to alter them. Our part of research, funded by NCBR POIR 01.01.01-00-0792/16 research grant, was focused on detecting code smells. To do that, we needed a body of source code snippets reviewed by professional developers. We decided to collect data also from junior developers to discover if understanding of the code smell concept changes with experience.

We gathered reviews of four code smells–two on the class level (*Blob* and *Data Class*) and two on the function level (*Feature Envy* and *Long Method*)–from a total of 26 developers with professional experience (8 hired as senior developers, 4 hired as regular developers[2], 8 hired as junior developers and 6 with unknown background).

[1] https://codebeat.co
[2] "Regular developer" is a position in industry with experience expected to be somewhere between a junior developer and a senior developer.

All the reviewers were volunteers. In case of senior and regular developers they were mostly coming from `code quest` software development company, while the rest of reviewers were volunteers with industrial experience recruited among MSc students from the software engineering track involved in the R&D project in software engineering course provided by the first author. The second author, since employed as a lead developer, was involved in code smells assessment as well. Our MLCQ[3] data set consists of 14739 reviews in total.

To allow further research based on developer experience profile, we collected data from these developers using comprehensive surveys on Typeform[4]. Data obtained via survey is published together with the primary code smell data set. Six out of 26 reviewers did not complete the survey. The total number of samples reviewed by users that did not complete the survey equals 454 (≈3% of the data set). Only one of those users has reviewed more than 60 samples. All those samples still have a reviewer identifier, so they can be used in research that requires only to be able to assign each sample to a unique developer (without relying on the developer's experience etc.). These 454 samples can be easily removed if appear not needed. Hence, we decided not to remove them upfront.

The contributions of this paper are:

(1) Contemporary and large data set including code samples from actively developed software projects available from GitHub that were assessed for industry-relevance, where four code smells (*Blob*, *Data Class*, *Feature Envy* and *Long Method*) were manually assessed on four-level severity scale (*critical*, *major*, *minor*, and *none*) on both the class level, as well as the method level, by developers with industrial experience. More details about how the software projects were selected can be found in [7], while the R script to filter possibly relevant GitHub projects, built by the authors using the GitHub GraphQL API, can be found in the `reproducer` R package available from CRAN [10]. Please note that our previous study [7] included only the selection of software projects and did not concern the selection or review of any code samples, which is the core part of this paper.
(2) Auxiliary context data, collected via survey, describing the background of software developers assessing the severity of the code smells. Both, the primary code smell data set and the auxiliary data set is available on Zenodo at https://doi.org/10.5281/zenodo.3666840, as well as in the subsequent version of the `reproducer` R package on CRAN, as we did in our previous papers (e.g., [4, 5, 7–9]) to streamline the usage of the research results.

The rest of the paper is structured as follows. Section 2 outlines briefly some other code smell data sets published to date. Section 3 describes the procedures we used for data acquisition. In Section 4 we present some numeric characteristics of the gathered data. Section 5 contains detailed metadata, while in Section 6 we list several threats to validity and misunderstandings that are likely. We conclude the paper describing possible applications in Section 7.

[3]MLCQ is an abbreviation formed from the initial letters of the words: Madeyski Lewowski Code Quest
[4]https://www.typeform.com/

2 RELATED DATA SETS

Some researchers [1, 12] decide to create new data sets for the task(s) at hand and publish them later on as part of their research. This has some benefits—for example, the data set exactly matches their research needs—but it also has several drawbacks. First and foremost, access to senior software developers is fairly hard and expensive, and it is not at all obvious whether smells reviewed by students or junior developers align with those that would be assigned by senior developers. Second, since junior developers may not yet know about code smells, they are hastily trained to spot them—however, these developers are often trained using rules. As a result, it is possible that the decision about the meaning of code smell is implicitly taken by the researcher when training junior developers to spot them, instead of by each of the developers himself. Third, those data sets are often not published in a form that can be used for reliable reproduction—i.e., including class paths, source code revisions, URLs, etc.

An important contribution to the field and one of the first public data sets was published by Palomba et al. [13]. This data set contains 243 instances of five types of code smells—*Divergent Change*, *Shotgun Surgery*, *Parallel Inheritance*, *Blob* and *Feature Envy*—across 20 open source projects. The paper introduced also a tool for code smell acquisition—however, hyperlinks present in the paper are no longer valid (as of Dec 11, 2019), and the source code for the tool itself does not seem to be open-sourced. The tool itself is accessible at another URL.

Palomba et al. [12] use a data set containing 17350 instances of 13 different code smell types—*Class Data Should Be Private*, *Complex Class*, *Feature Envy*, *God Class*, *Inappropriate Intimacy*, *Lazy Class*, *Long Method*, *Long Parameter List*, *Message Chain*, *Middle Man*, *Refused Bequest*, *Spaghetti Code* and *Speculative Generality*—across 395 releases of 30 projects. However, in another study [11] the authors claim that the number of instances reaches 40888 (number of code smell types, projects and releases match).

Fontana et al. introduced a data set of four code smells using both—binary classification [1] and a severity scale [2]. Those data sets contain several hundred samples per smell. Software projects used there are taken from Qualitas Corpus [16]. While Qualitas Corpus is often used in software engineering research, it contains Java projects even as old as from 2002—much before the introduction of many recent techniques and features [3]. Thus might be much less relevant for code smell detection now. The hyperlinks listed in the publication are no longer valid (as of Dec 22, 2019), but the data is still present on the site.

Another approach to creating code smell data set is presented in [6]—authors used existing tools to analyze selected repositories. The end result contains code smell annotations, but ones obtained by a tool, not by real developers with professional experience.

The main limitation of all those data sets is the procedure used for their creation—they were generally created either automatically or by students or single researchers. Our data set also contains some data from students and researchers, but there are three important differences:

- all of the reviewers involved in the code smell assessment are actively employed in the software development industry,

- the majority of samples is gathered by developers that are neither students nor researchers,
- the data set provides unique and detailed insights related to professional and academic background of the reviewers.

In [15] authors conducted a survey about code smell perception. While [15] does not focus on developers' experience, results can be compared to the ones from our survey to identify differences between analysed populations. It is worth noting that survey results is one of main contributions of [15], while in our case it only brings in the auxiliary information.

3 DATA ACQUISITION PROCEDURE

The whole data set consists of two parts—code smell reviews and surveys. Surveys are manually anonymized, so while they still can be linked to reviews by using the identifier of the reviewer, they cannot be linked to any physical person.

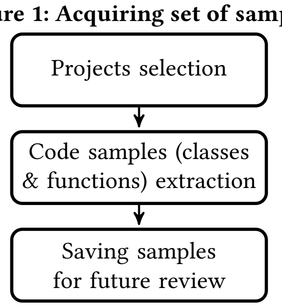
3.1 Acquisition tool

To simplify the process of collecting data, a supporting tool for displaying code samples and gathering the results of code smell reviews was developed at *code quest*. The tool is not publicly available, thus we do not include hyperlink to the UI or to the code there. The tool may be published by *code quest* at some point in the future, but the URLs would then change.

3.2 Sample creation procedure

Code samples were generated from Java projects selected from GitHub. We used the project data set described in [7]. We did not apply any manual filters and all samples from all 792 projects were used. Detailed sample acquisition flow is shown in Figures 1 and 2.

Figure 1: Acquiring set of samples



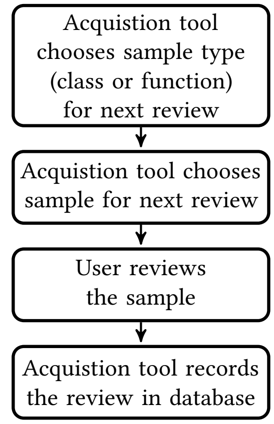
In the rest of this paper, a *sample* will mean either a class or interface (for the class-level smells) or a method (for the method-level smells). Developer is presented with a piece of source code (if method is presented, the class is not immediately visible) together with fully qualified name and link to the source code on GitHub if there is a need for further, more contextual investigation.

We did not record whether developers used any external information to review the samples.

3.3 Smell selection and gathering reviews of code samples

We decided to focus on four code smells: *Feature Envy* and *Long Method* on the method level and *Data Class* and *Blob* on the class level. They were selected as they appeared to be the most popular code smells analysed in literature according to our internal report (yet unpublished) prepared in a form of systematic review for the *code quest* company in 2017 and 2018.

Figure 2: Sample review flow



To gather review samples the following procedure is executed:

(1) Code sample is selected for review using process described below.
(2) Developer assigns the severity of possible code smells on the four-level scale (*critical*, *major*, *minor*, and *none*). Developer is free to skip any sample that he or she is uncertain of.
(3) Developer approves the selection by clicking "Next" button.
(4) Each pair (sample, smell) is saved in the database as a separate review.

Developer can choose whether she or he wants to assess the severity of both smells on a given, class or method, level or only one of them. Developer is able to change his (or hers) assessment later. We did not conduct any training related to code smell identification, because we believe that our goal is to extract professional developers' understanding of code smells, instead of imposing thresholds from the legacy literature or our own expectations what constitutes particular code smells.

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 performed by 26 developers, 20 of which have completed the survey described in Section 3.4.

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 785 projects (7 projects could not be analysed), the third one - from 552750 classes and 2297722 functions (due to filtering out less than 4 line samples) from 785 projects while in the last phase we only performed crosscheck on already reviewed samples:

(1) The first 2175 samples (date range: March 27, 2019 - April 2, 2019) were selected when there was a defect in randomisation part of 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: April 3, 2019 - April 12, 2019) were selected randomly from all available.
(3) The next 4801 samples (date range: April 13, 2019 - July 25, 2019) were selected randomly from all samples longer than 4 lines, including opening and closing braces. 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 precious time of developers and to better focus their effort on reviewing potentially smelly code samples.
(4) The last 5115 samples (date range: July 26, 2019 - September 13, 2019) were selected to perform a crosscheck—only from samples already tagged with severity higher than 'none'. Samples selected from crosscheck were those that had only 1 review or 2 reviews with different severities. These samples were gathered by the group of developers involved in earlier phases, with a restriction that developer could not crosscheck his or her own review.

The split into phases was not initially intended, but we encountered the defect from the first phase only after starting the acquisition. Then it turned out that the amount of "none" samples is higher that we have estimated, thus the need to focus the effort of professional developers on valuable samples that can reasonably constitute code smells to balance the data set. The last phase, crosscheck, was needed to reach more reliable conclusions.

Since there were many more method samples than class samples, and we wanted to gather similar number of both, we decided that the first selection step in the first three phases will be selecting whether we are looking for a class sample or method sample (and both were selected with equal probability). The last step (crosscheck) is the reason why the number of reviews for classes is higher than for functions—there was less agreement on the former.

While the whole data set includes reviews of code samples gathered from projects that are industry relevant, semi-industry-relevant, and industry irrelevant, it is easy to select the subset of samples from industry-relevant projects, which constitute 80% of the total number of projects. Also the number of reviews of code taken from industry-relevant projects constitutes 92.5% of total code samples and reviews.

3.4 Survey

The survey was prepared using Typeform[5] and all reviewers were asked to complete it. However, six reviewers did not complete the survey. The survey was an internal one and we did not publish it in any external services. Its sole purpose was to provide us with information about reviewers and it was not meant to be a tool for general software development research.

The survey contained 59 questions, including detailed questions about professional experience, programming languages used throughout the career, used tools, occupied positions, sizes of projects that developer was involved in, known paradigms, open source involvement, review habits, knowledge about code smells and opinions about state-of-the-art tools. Completing the survey took 48 minutes on average.

One of the questions in the survey was GitHub login of reviewer—by utilising it, we were able to map survey answers to reviews (our review tool used GitHub-based authentication).

Of course, published version of the survey is stripped of all personal data (emails, logins). We also decided to remove company names, so that there will be no misunderstanding—this research was only supported by the *code quest* company and all other participants completed the survey and reviews in their own free time.

4 DATA CHARACTERISTICS

We provide basic characteristics of the collected code smell and survey data.

4.1 Smell data

The basic characteristics of the code smell data (presented in Table Table 1) illustrate the size of the collected data set (e.g., in terms of the number of sample reviews conducted by developers, the number of unique samples, and the number of projects from which the samples were gathered). It has taken about half a year to collect the data set from over 500 software projects, and the number of software developers with industrial experience involved in data collection exceeded two dozens. Hence, the effort behind the data collection can be considered large.

Table 1: Basic characteristics

Characteristic	Value
total # of reviews	14739
# reviews from industry-relevant projects	12710
# reviews from semi-industry-relevant projects	924
# reviews from industry irrelevant projects	1105
total # of samples	4770
# of samples from industry-relevant projects	4129
# of samples from semi-industry-relevant projects	290
# of samples from industry irrelevant projects	351
# projects	523
# reviewers	26
# smell types	4
time span	27.03.2019–13.09.2019

The distribution severities of code smells (presented in Table 2) shows how rare (according to developers with industrial experience) some code smells are, especially *critical* instances of *Feature Envy*.

The distribution of the number of reviews performed by developers that have taken part in the study (presented in Table 3) shows that some developers were more involved in code smell reviews than other. Hence, the data set is imbalanced with this regard. However, the average professional experience in programming of the five developers who preformed more than 1000 reviews was much higher than the average professional experience of all the developers involved in the study (i.e., 10 years vs 4 years).

[5]www.typeform.com

Table 2: 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

Table 3: Number of reviews

# of performed reviews	# of reviewers	professional experience in programming [years]	
		average	median
>1000	5	10.0	7.5
300-1000	5	2.1	1.0
100-299	7	2.1[1]	1.3[1]
<100	9	2.4[2]	2.0[2]

[1] on the basis of 6 of 7 developers who filled in the survey
[2] on the basis of 4 of 9 developers who filled in the survey

4.2 Survey data

20 developers involved in the study worked in at least 8 different companies (four respondents decided not to unveil the company they were working for), while their recent industrial experience in software engineering measured in years (as shown in Table 4) shows that the population of interest to which we aim to generalise the results are developers with industrial software engineering experience. We believe that such generalisation, in case of students or researchers often involved in such studies, would not be possible.

Table 4: Software developers involved in the study

Metric	Value
# respondents	20
# companies	8[1]
average professional experience in programming	4.1 years
median professional experience in programming	1.8 years
min professional experience in programming	0.5 year
max professional experience in programming	19 years
average number of languages used	6.1

[1] on the basis of 16 of 20 developers who filled this field in the survey

Almost all (90%) of the developers involved in the study possessed their university degrees, either on the master or bachelor level, while their positions varied between senior/lead developer and junior developer, as shown in Tables 5 and 6. In our opinion these characteristics reflect the situation in the software industry fairly well. According to a 2019 Developer survey by Stack Overflow[6], questioning over 70,000 employed professional software developers, about 79% of professional developers have some degree in computer sciences or related areas (in our case it was 90%). Furthermore, the percentage of developers with the MSc degree (in the survey by Stack Overflow) was 49.1%, while in our case it was 45%. The most notable difference was the percentage of developers with the BSc degree, which was 25.4% in the survey by Stack Overflow, while in our case it was 45%.

[6]https://insights.stackoverflow.com/survey/2019/#education

Table 5: Education

Metric	Value
# Master (MSc)	9
# Bachelor (BSc)	9
# W/o BSc or MSc	2

Table 6: Positions

Metric	Value
# Senior/Lead Developer	8
# Regular Developer	4
# Junior Developer	8

5 DATA SHEET

The data sheet for this data set is inspired by the ESA[7] datasheet standard, adapted by us to match our needs in software engineering. Due to the paper length limitation we focus here on the primary data sheet (`MLCQCodeSmellSamples`), while the complete description of the data sets, including the auxiliary data sheet (`MLCQCodeSmellDevelopersSurvey`), is presented in detail in the online appendix[8].

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

id a numeric identifier of the (code sample) 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*, *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: $Package.ClassName[\#FunctionName \quad arg1|arg2|...]$ (e.g., $org.eclipse.swt.widgets.Menu\#setLocation \quad int|int$),
link link to view the sample in a browser.

6 THREATS AND LIMITATIONS

Samples selected for review were chosen in four separate phases (described in detail in Section 3.3). This means that the population of source code entities used for selection was not uniform during the whole research. While this is well-documented in this paper,

[7]https://esajournals.onlinelibrary.wiley.com/hub/journal/19399170/resources/data_paper_inst_ecy
[8]http://madeyski.e-informatyka.pl/download/MadeyskiLewowskiMLCQAppendix.pdf

this can still be confusing both for researchers and for machine learning algorithms.

Unfortunately, we are not able to guarantee the good will of all participants. We did conduct an additional crosscheck in the end of research to verify the samples and assigned smell severities. However, we only did this with samples that were initially tagged with severity above none—therefore it is possible that some samples which should have severity above none do not have it.

While we did gather nearly 15 thousand samples, over 77% show no smells (none severity). We believe that this is still a useful result since negative samples are also relevant. However, we also acknowledge the possibility that our code smell data set in large part describes what is not a code smell, instead of what is one.

We do not provide any metrics. This is due to a few reasons:

- the set of possible metrics is ever expanding and for Java they are relatively easy to obtain, so we believe that interested researchers will manage to calculate ones that suit them best,
- any defects in metrics calculated software could then be replicated in future research,
- our initial study did not find out any of popular metrics that would work particularly well.

Some projects or files may no longer be available—if a repository is removed, the project will be no longer accessible. The expected removal rate has to be studied separately.

Of course, since the number of reviewers is relatively small (26), and almost half of them are from the same company, there is always possibility of bias. We acknowledge such a possibility. However, we believe that various backgrounds of the core contributors (as evidenced by the survey) address this problem at least partially.

7 APPLICATIONS

This data set can be applied in a number of research setups. Code smell detection is probably the most obvious one, but this data set can be also used to understand differences in perception between junior developers and senior developers or to find out traits from professional background that correlate with specific code smell perception.

The data set can also be used as an auxiliary data set for defect prediction or other complex scenarios where existence of code smells may serve as a predictor.

ACKNOWLEDGMENTS

This work has been supported by the National Centre for Research and Development (NCBiR) project POIR.01.01.01-00-0792/16 with the aim to improve the `codebeat` platform (http://codebeat.co) by *code quest sp. z o.o.*. We would like to thank all of the developers involved in the study.

A SUPPLEMENTARY MATERIALS

Supplementary materials for this paper include the online appendix[9], as well as data sets. The data set is available on Zenodo[10] and CRAN from the subsequent version of the `reproducer` R package [10]. We also maintain a website to provide the most up-to-date list of code samples with smells (validated by developers)[11].

REFERENCES

[1] Francesca Arcelli Fontana, Mika V. Mäntylä, Marco Zanoni, and Alessandro Marino. 2016. Comparing and experimenting machine learning techniques for code smell detection. *Empirical Software Engineering* 21, 3 (6 2016), 1143–1191.

[2] Francesca Arcelli Fontana and Marco Zanoni. 2017. Code smell severity classification using machine learning techniques. *Knowledge-Based Systems* 128 (2017), 43 – 58. https://doi.org/10.1016/j.knosys.2017.04.014

[3] Hanna Grodzicka, Arkadiusz Ziobrowski, Zofia Łakomiak, Michał Kawa, and Lech Madeyski. 2020. Code Smell Prediction Employing Machine Learning Meets Emerging Java Language Constructs. In *Data-Centric Business and Applications: Towards Software Development (Volume 4)*, Aneta Poniszewska-Marańda, Natalia Kryvinska, Stanisław Jarząbek, and Lech Madeyski (Eds.). Springer International Publishing, Cham, 137–167. https://doi.org/10.1007/978-3-030-34706-2_8

[4] Marian Jureczko and Lech Madeyski. 2015. Cross–Project Defect Prediction With Respect To Code Ownership Model: An Empirical Study. *e-Informatica Software Engineering Journal* 9, 1 (2015), 21–35. https://doi.org/10.5277/e-Inf150102

[5] Barbara Kitchenham, Lech Madeyski, David Budgen, Jacky Keung, Pearl Brereton, Stuart Charters, Shirley Gibbs, and Amnart Pohthong. 2017. Robust Statistical Methods for Empirical Software Engineering. *Empirical Software Engineering* 22, 2 (2017), 579–630. https://doi.org/10.1007/s10664-016-9437-5

[6] Valentina Lenarduzzi, Nyyti Saarimäki, and Davide Taibi. 2019. The Technical Debt Dataset. In *Proceedings of the Fifteenth International Conference on Predictive Models and Data Analytics in Software Engineering (PROMISE'19)*. ACM, New York, NY, USA, 2–11. https://doi.org/10.1145/3345629.3345630

[7] Tomasz Lewowski and Lech Madeyski. 2020. Creating Evolving Project Data Sets in Software Engineering. In *Integrating Research and Practice in Software Engineering*, Stanislaw Jarzabek, Aneta Poniszewska-Marańda, and Lech Madeyski (Eds.). Studies in Computational Intelligence, Vol. 851. Springer, Cham, 1–14. https://doi.org/10.1007/978-3-030-26574-8_1

[8] Lech Madeyski and Marian Jureczko. 2015. Which Process Metrics Can Significantly Improve Defect Prediction Models? An Empirical Study. *Software Quality Journal* 23, 3 (2015), 393–422. https://doi.org/10.1007/s11219-014-9241-7

[9] Lech Madeyski and Barbara Kitchenham. 2018. Effect Sizes and their Variance for AB/BA Crossover Design Studies. *Empirical Software Engineering* 23, 4 (2018), 1982–2017. https://doi.org/10.1007/s10664-017-9574-5

[10] Lech Madeyski, Barbara Kitchenham, and Tomasz Lewowski. 2020. *reproducer: Reproduce Statistical Analyses and Meta-Analyses*. http://madeyski.e-informatyka.pl/reproducible-research/ R package (http://CRAN.R-project.org/package=reproducer).

[11] Fabio Palomba, Gabriele Bavota, Massimiliano Di Penta, Fausto Fasano, Rocco Oliveto, and Andrea De Lucia. 2018. A large-scale empirical study on the lifecycle of code smell co-occurrences. *Information and Software Technology* 99 (2018), 1 – 10. https://doi.org/10.1016/j.infsof.2018.02.004

[12] Fabio Palomba, Gabriele Bavota, Massimiliano Di Penta, Fausto Fasano, Rocco Oliveto, and Andrea De Lucia. 2018. On the diffuseness and the impact on maintainability of code smells: a large scale empirical investigation. *Empirical Software Engineering* 23, 3 (01 Jun 2018), 1188–1221. https://doi.org/10.1007/s10664-017-9535-z

[13] Fabio Palomba, Dario Di Nucci, Michele Tufano, Gabriele Bavota, Rocco Oliveto, Denys Poshyvanyk, and Andrea De Lucia. 2015. Landfill: An Open Dataset of Code Smells with Public Evaluation. In *Proceedings of the 12th Working Conference on Mining Software Repositories (MSR '15)*. IEEE Press, Piscataway, NJ, USA, 482–485. https://doi.org/10.1109/MSR.2015.69

[14] Fabio Palomba, Marco Zanoni, Francesca Arcelli Fontana, Andrea De Lucia, and Rocco Oliveto. 2019. Toward a Smell-Aware Bug Prediction Model. *IEEE Transactions on Software Engineering* 45, 2 (Feb 2019), 194–218. https://doi.org/10.1109/TSE.2017.2770122

[15] Davide Taibi, Andrea Janes, and Valentina Lenarduzzi. 2017. How developers perceive smells in source code: A replicated study. *Information and Software Technology* 92 (2017), 223 – 235. https://doi.org/10.1016/j.infsof.2017.08.008

[16] Ewan Tempero, Craig Anslow, Jens Dietrich, Ted Han, Jing Li, Markus Lumpe, Hayden Melton, and James Noble. 2010. Qualitas Corpus: A Curated Collection of Java Code for Empirical Studies. In *2010 Asia Pacific Software Engineering Conference (APSEC2010)*. 336–345. https://doi.org/10.1109/APSEC.2010.46

[17] A. Yamashita and L. Moonen. 2013. Do developers care about code smells? An exploratory survey. In *2013 20th Working Conference on Reverse Engineering (WCRE)*. 242–251. https://doi.org/10.1109/WCRE.2013.6671299

[9] http://madeyski.e-informatyka.pl/download/MadeyskiLewowskiMLCQAppendix.pdf

[10] https://doi.org/10.5281/zenodo.3666840

[11] http://madeyski.e-informatyka.pl/reproducible-research/#smells