Abstract—Researchers apply machine-learning techniques for code smell detection to counter the subjectivity of many code smells. Such approaches need a large, manually annotated dataset for training and benchmarking. Existing literature offers a few datasets; however, they are small in size and, more importantly, do not focus on the subjective code snippets. In this paper, we present DACOS, a manually annotated dataset containing 10,267 annotations for 5,192 code snippets. The dataset targets three kinds of code smells at different granularity—multifaceted abstraction, complex method, and long parameter list. The dataset is created in two phases. The first phase helps us identify the code snippets that are potentially subjective by determining the thresholds of metrics used to detect a smell. The second phase collects annotations for potentially subjective snippets. We also offer an extended dataset DACOSX that includes definitely benign and definitely smelly snippets by using the thresholds identified in the first phase. We have developed Tagman, a web application to help annotators view and mark the snippets one-by-one and record the provided annotations. We make the datasets and the web application accessible publicly. This dataset will help researchers working on smell detection techniques to build relevant and context-aware machine-learning models.

I. INTRODUCTION

Code smells are symptoms of poor design and implementation. Existing literature shows that code smells have a negative impact on maintainability, development effort, and reliability among other quality attributes. Given its importance, the software engineering community needs to complement existing smell detection techniques by a dataset. The dataset offers annotated code snippets that are definitely benign or smelly. Furthermore, we developed a web-application Tagman to make it easy for annotators to see one snippet at a time, and indicate whether a smell is present in the snippet.

Currently, the majority of commonly used tools use metrics and heuristics to identify code smells. It is often argued that due to the subjective nature of smells, one cannot come up with universally accepted metric thresholds to classify a snippet in a smelly or benign code in all contexts. To overcome the challenge introduced by the subjective nature of smells, researchers propose smell detection using machine-learning techniques. Such approaches rely on a code smells dataset, ideally manually annotated, to train a machine-learning model. However, existing datasets offer little on multiple fronts. First, the literature offers only a handful of datasets such as LANDFILL. Second, existing datasets contain a small number of annotated samples; for example, LANDFILL offers annotations for only 243 snippets. A dataset with a small number of annotated samples would help a little to train state-of-the-art deep-learning models with reasonable accuracy. Next, existing code smells datasets do not filter out code snippets that are definitely benign or smelly. For example, a snippet with a very few (say, three) lines of code cannot have a long method; similarly, a snippet with a very large (e.g., 200) number of lines of code definitely suffers from a long method smell. Given that the value, in terms of effectiveness, of a smell dataset lies in the captured subjectivity, such definite snippets, either definite benign or smelly, reduce the efficacy offered by a dataset. Lastly, the available support for different types of smells is limited; for example, LANDFILL offers annotated snippets for five types of smells. Given the huge amount of effort involved in annotating code snippets, the software engineering community needs to complement existing smell datasets for other kinds of actively researched smells.

In this paper, we offer a manually annotated dataset of code smells viz. DACOS—A Manually Annotated Dataset of Code Smells (DACOS). To create an effective dataset, we filtered the code snippets that are likely to be subjective by removing the snippets that are either definitely benign or smelly. This approach helps us better utilize the annotators’ effort by considering their inputs where we actually need them. The dataset offers annotated code snippets for three code smells—multifaceted abstraction, complex method, and long parameter list. In addition to a manually annotated dataset on potentially subjective snippets, we offer DACOSX containing a large number of snippets that are either definitely benign or smelly. Furthermore, we developed a web-application Tagman to see one snippet at a time, and indicate whether a smell is present in the snippet. We have made source code of Tagman available publicly.
We make the following contributions to the state of the art.

- We offer DACOS, a manually annotated code smell dataset, containing 10,267 annotations for 5,192 code snippets for the considered code smells.
- We also provide DACOSX, an extended dataset containing 207,605 snippets that are either definitely benign or smelly. These datasets will help researchers in the field to train and validate their machine-learning models.
- A configurable web application TAGMAN for easy smell annotations. The community may use the application for similar code annotation purposes.

II. DATASET CONSTRUCTION

Figure 1 provides an overview of the dataset construction process. We elaborate on the steps in detail below.

A. Downloading repositories

In step 1 from Figure 1 we perform the following tasks to identify and download repositories.

- We use searchgithubrepo package, which in turn uses the GitHub GraphQL API to filter GitHub repositories.
- To identify high quality Java repositories, we select repositories with more than or equal to 13 thousand stars and more than ten thousand lines of code.
- Also, we discard the repositories that are not modified in the last one year.
- In addition, we use QScored to filter out repositories based on their code quality score. QScored assigns a weighted quality score based on the detected smells at various granularities. We select repositories that have a quality score less than ten (the higher the score, the poorer the quality).
- Finally, we obtained ten repositories after applying the filtering criteria. We download the selected repositories.

B. Dividing the repositories into classes and methods

We need to split a repository into individual methods and classes so that TAGMAN can show individual snippets one by one to an annotator. We use CODESPLITJAVA in step 2 to split each repository into individual methods and classes.

C. Analyzing repositories

In step 3, we employ a metrics-based filtering process in the phase-2 of manual annotation. We use DESIGNITEJAVA to compute code quality metrics. DESIGNITEJAVA computes a variety of code quality metrics and detects smells; it has been used in various studies. We elaborate the process to filter out non-subjective samples in the manual annotation step.

D. Tagman

TAGMAN, shown as 4 in Figure 1, is a web-based tool that we created to facilitate the annotation process. Figure 2 shows a screenshot of the application showing a code snippet and an option to annotate the snippet with a smell. The front-end of TAGMAN is written in Thymeleaf and HTML/CSS. The back-end of the application is developed in SpringBootJava and the data is stored in a MySQL database. Figure 3 shows the schema of the database.

At the beginning of the code smell annotation cycle, we upload a CSV file containing the repository names and URL of selected GitHub repositories. TAGMAN back-end uses a set of Python scripts to download GitHub repositories, split the code into class and method files, and run DESIGNITEJAVA.

Once the data import is completed, the tool is ready to start accepting annotations. To start an annotation session, a user login (or sign up) to the application. Then, the user is presented with instructions including the definitions of code smells. The user can then start annotating the presented code snippets.

E. Manual annotation

We employ a snippet selection mechanism to identify potentially subjective snippets w.r.t. a code smell. We do so to improve the effectiveness of the resultant dataset by only including manual annotations for potentially subjective code.
snippets. Also, such a strategy helps us better utilize the available annotators’ time. A potentially subjective snippet is a code snippet that may get classified as benign or smelly based on context. The rest of the snippets that are not identified as potentially subjective snippets are either definitely benign or smelly snippet. For example, cyclomatic complexity (CC) is commonly used to detect complex method smell. A code snippet is definitely benign if \( CC = 1 \); similarly, a snippet is definitely smelly if \( CC = 30 \). We divide our annotation process into two phases. In the first phase, we show all snippets, i.e., without any filtering, to annotators to identify metrics thresholds to determine whether a snippet is potentially subjective or not. The second phase uses the identified metrics thresholds and show the filtered code snippets to annotators.

1) Phase-1: In the first phase, we show code snippets without any filtering to annotators. TAGMAN presents one snippet at a time to the annotators and collects their response on whether the shown snippet has a code smell or not. We show a code snippet to two randomly chosen annotators and record their responses. The annotators recruited, on a volunteer basis, for this phase were graduate students of Computer Science enrolled in a software engineering course (during summer 2022) that cover code smells extensively. A total of 110 annotators participated in this phase.

We compute the minimum and maximum threshold for metrics that are used to decide the presence of a code smell based on the collected responses in Phase-1. We received a total 17,869 responses in this phase. We compute the lowest metric value \( t_l \) where the smell is identified, for each smell individually, to obtain the threshold at the lower side. Similarly, we extract the highest metric value \( t_h \) where the smell is not identified. Then, we compute the standard deviation \( s_d \) of the metric value for the samples where the smell is identified. Finally, we obtain the low threshold using \( \min(m_l, t_l - s_d) \) and high threshold using \( \min(m_h, t_h + s_d) \) for subjective snippet identification. Here, \( m_l \) and \( m_h \) represent the lowest and highest possible values of a metric. Table I summarizes the quality metrics corresponding to each code smell and their thresholds for identifying subjective snippets. For instance, for cyclomatic complexity metric, we obtain 4 and 7 from the above calculation after rounding the values to the nearest integer.


<table>
<thead>
<tr>
<th>Code smell</th>
<th>Code quality metric</th>
<th>Metric threshold</th>
</tr>
</thead>
<tbody>
<tr>
<td>Complex method</td>
<td>Cyclomatic complexity</td>
<td>4–7</td>
</tr>
<tr>
<td>Long parameter list</td>
<td>Parameter count</td>
<td>3–6</td>
</tr>
<tr>
<td>Multifaceted abstraction</td>
<td>Lack of cohesion among methods (LCOM)</td>
<td>0.4–0.8</td>
</tr>
</tbody>
</table>

2) Phase-2: We configure our filtering mechanism based on the thresholds obtained from Phase-1 and invite annotators by advertising the link of TAGMAN installation on social media platforms such as Twitter and LinkedIn. The invitation was open to all software developers, software engineering students, and researchers who understand Java programming language and at least basic object-oriented concepts. We kept the invitation open for six weeks during Dec-Jan 2022-23. A total of 82 annotators participated in this phase. TAGMAN showed snippets that have metric values falling between the low and high thresholds (inclusive). We configured TAGMAN to get two annotations for each sample to improve the reliability of the annotations.

F. Dataset information

After the phase-2, we received a total of 10,267 annotations for 5,192 samples from 86 annotators. Table II presents the number of annotations and samples per smell type. Dataset availability: The datasets offered in this paper are available online. Also, the repositories containing scripts used to prepare code snippets as well as code annotation application are available online.

https://github.com/SMART-Dal/Tagman
III. POTENTIAL RESEARCH APPLICATIONS

- **Detecting and validating code smells:** Given the subjective nature, traditional smell detection tools that implement rules and heuristics to identify smells, do poorly. The success of a machine-learning approach to detect smells depends on the availability of a large manually annotated dataset. The presented datasets, DACOS and DACOSX, complement existing datasets by offering a large number of subjective code snippets for three code smells that the existing datasets do not cover.

- **Correlating code smells with software engineering aspects:** A variety of exploratory and empirical studies investigating the impact of code smells exists. It includes bug prediction [33], maintainability prediction [34], and maintenance effort [35]. In addition to existing directions, the tools trained or fine-tuned from the offered datasets can be used to effectively establish a relationship between code smells and productivity of a software development team.

- **Extending TAGMAN for code annotation:** The code annotation application that we developed to create this dataset can be extended for similar kinds of code annotation, for example, to spot vulnerable code and to segregate well-written identifiers.

IV. RELATED DATASETS

Software engineering literature offers a small number of manually annotated datasets for code smells. Palomba et al. [36] offered a dataset LANDFILL containing annotations for five types of code smells—divergent change, shotgun surgery, parallel inheritance, blob, and feature envy. They offered annotations for 243 snippets. They also developed an online portal where contributors can annotate code for smells, however as of writing this paper (Jan 2023), the portal is not accessible. Madeyski et al. [37] proposed MLCQ—a code smell manually annotated dataset. The dataset contains 14.7 thousand annotations for 4,770 samples. The dataset considered four smells—blob, data class, long method, and feature envy. Both of the datasets mentioned above do not consider the subjectiveness of a code snippet; hence most of the snippets might not add any new information for a machine-learning classifier when used in training. Also, we chose the code smells that are not covered by any existing code smell dataset and hence complement the existing datasets. There are some code smells datasets such as QScored [23]. Though the QScored dataset is large, the samples are not manually annotated and hence lack the required capturing of context.

V. THREATS TO VALIDITY

*Internal validity* threats concern the ability to draw conclusions from our experimental results. In phase-2 of manual annotation, we invited volunteers with at least a basic understanding of Java programming language and object-orientation concepts. We advertised the invitation on social media professional channels (Twitter and LinkedIn). Given the anonymity of the exercise, we do not have any mechanism to verify the assumption that the participants has sufficient knowledge to attempt the annotations. However, we offered all the major participants (with at least 50 annotations) to include them as contributors to the dataset; we perceive such a measure would have motivated the annotators to perform the annotations to the best of their abilities. Additionally, we configured TAGMAN to obtain two annotations per sample so that we can reduce the likelihood of a random annotation.

*External threats* are concerned with the ability to generalize our results. The proposed dataset is for snippets written in Java. However, our code annotation tool is generic and it can be used to annotate snippets from any programming language. Additionally, scripts used to generate individual snippets can be customized to use any other external tool for splitting the code into methods and classes. Furthermore, the thresholds used in the annotation process to filter snippets based on low and high thresholds of a metric are configurable.

VI. LIMITATIONS, CONCLUSIONS, AND FUTURE WORK

We offer DACOS—a manually annotated code smell dataset containing 10,267 annotations for 5,192 subjective code snippets. We also provide large DACOSX dataset containing definitely benign and definitely smelly snippets in addition to those present in DACOS. The paper offers TAGMAN, a code annotation application, that can be reused in similar contexts.

The proposed dataset covers three code smells. We selected a rather small set of code smells to consider in the dataset because it is better to have more number of annotations for a smell rather than having small number of annotations per smell. Also, we chose a set of smells that are not covered by existing code smells dataset. We configured TAGMAN to obtain two annotations per sample. Though it improves the reliability of the dataset, one may argue that it may introduce a situation where the annotations are contradictory to each other. We can mitigate the limitation by increasing the number of annotations per sample to three; we will incorporate this mechanism in the future version of the datasets. Additionally, we would like to expand the scope of the dataset in terms of programming language, number of samples, and number of supported smells in the future.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Code smell</th>
<th>#Annotations</th>
<th>#Samples</th>
</tr>
</thead>
<tbody>
<tr>
<td>DACOS</td>
<td>Complex method</td>
<td>4,349</td>
<td>2,197</td>
</tr>
<tr>
<td></td>
<td>Long parameter list</td>
<td>3,221</td>
<td>1,634</td>
</tr>
<tr>
<td></td>
<td>Multifaceted abstraction</td>
<td>2,697</td>
<td>1,361</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td>10,267</td>
<td>5,192</td>
</tr>
<tr>
<td>DACOSX</td>
<td>Complex method</td>
<td>–</td>
<td>94,489</td>
</tr>
<tr>
<td></td>
<td>Long parameter list</td>
<td>–</td>
<td>93,442</td>
</tr>
<tr>
<td></td>
<td>Multifaceted abstraction</td>
<td>–</td>
<td>19,674</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td>–</td>
<td>207,605</td>
</tr>
</tbody>
</table>
REFERENCES


