Revisiting Multi-language Code Smells: Evaluating, Refining, and Validating Detection Techniques– Replication Package

This repository contains the utilities, datasets, and documentation required to replicate the study *"Revisiting Multi-language Code Smells: Evaluating, Refining, and Validating Detection Techniques"*

Our study evaluates the accuracy of an existing detection approach, identifies limitations in its implementation, and proposes improvements. We also analyze the prevalence of multi-language code smells in open-source JNI projects.

Contents

- **`/Dataset/`**: Scripts and metadata to fetch and prepare the 60 open-source multi-language projects.
- **`/GroundTruth/`**: Manually curated ground truth for each project and each smell, created using detailed labeling guidelines and the detailed labeling guidelines.
- **`/Detection Approach Revised/`** – Refined implementation of the detector.
- **`/Results/`** – Detection results and precision/recall comparison.

Getting Started

1. Requirements

- Java 11+
- srcML (used to parse Java/C/C++ into XML).

Make sure `srcml` is installed and accessible in your PATH.

---

1. Dataset/

Contains all data and scripts necessary to retrieve and prepare the set of 60 open-source multi-language projects used for evaluating the detector.

- **fetch_projects.sh** – A shell script that automatically clones all required projects from GitHub.
- **projects_list.csv** – A CSV file listing each project's name, repository URL, and commit hash used in the analysis.

2. Ground Truth/

Provides manually validated data used for evaluating the precision and recall of both the baseline and revised detectors.

- **labeling_guidelines.txt** – Documentation of manual labeling criteria used during ground truth creation.
- **[15 subfolders]** – One per design smell, each containing the manually identified smelly files across all 60 projects.

This data was used to benchmark and validate both detection implementations.

3. `/Detection Approach Revised/`

–Contains refined implementation of the detector.

Running the Detector

Each smell must be run separately for clarity. To do this:

1. Navigate to: `Detection Approach Revised/mlssdd/`
2. Open `DetectCodeSmellsAndAntiPatterns.java`.
3. Uncomment the line corresponding to the smell you want to detect and assign dir=”smellName” (e.g., `TooMuchClusteringDetectionModified and dir=”TooMuchClustering”`).
4. Recompile
5. Run `DetectCodeSmellsAndAntiPatterns.java`

Each run will produce a CSV file under:

`/Results/<SmellName>/<ProjectName>.csv`

Example (detecting **Too Much Clustering**):

- Uncomment `TooMuchClusteringDetectionModified`.
- Put dir=”TMC”
- Run:

  `java “Detection approach/mlssdd.DetectCodeSmellsAndAntiPatterns”`

- Output: `Results/TMC/<ProjectName>.csv`

Why One Smell at a Time?

We intentionally run one smell per execution to:

- Keep results for each smell separate and easier to validate.
- Simplify comparison with manually curated ground truth.
- Improve reproducibility and clarity for future researchers.

---

4. Results/

Contains all outcome data produced from the replication study.
It is divided into two main components: **Performance Evaluation** and **Prevalence**.

4.1 Performance Evaluation/

Includes results comparing the baseline and refined detectors when applied to the new dataset.

- **results_of_baseline_detector_on_new_data/** – Detection results from the original (unrefined) detector.
- **results_of_revised_detector_on_new_data/** – Detection results from the refined detector applied to the same dataset.
- **precision_and_recall_of_baseline_detector.csv** – Summary of precision and recall for the baseline detector across all 15 smells.
- **precision_and_recall_of_revised_detector.csv** – Summary of precision and recall for the refined detector.

These results demonstrate the improvement in detection accuracy achieved through rule-level refinement.

4.2 Prevalence/

Contains updated results on the **distribution of multi-language code smells**, both on the dataset from the earlier study and on the newly analyzed dataset.

- **general_smelly/** – Lists of files affected by at least one design smell for each project and release.
- **prevalence_of_code_smells_on_old_data/** – Smell prevalence results using the dataset from the earlier study by Abidi et al.
- **prevalence_of_code_smells_on_new_data/** – Smell prevalence results using the newly analyzed dataset of 60 projects.
- **percentage_of_general_smelly_files_releaseWise.csv** – Percentage of smelly files per release across projects.
- **percentage_of_smelly_files_affected_by_each_smell_old_data.csv** – Distribution of individual smells (old dataset).
- **percentage_of_smelly_files_affected_by_each_smell_new_data.csv** – Distribution of individual smells (new dataset).