Assignment 2
Evaluating the design of an open-source project

Software Design & Modeling

Submit by: Wednesday, 06 November 2024 at 23:00

1 The assignment

Your assignment in a nutshell:

1. Pick a software **project** hosted on GitHub. The project should have:
   - at least 100 stars,
   - at least 100 forks,
   - at least 10 open issues, and
   - at least 50,000 lines of code.[1]
2. Run some **flaw detection** tools on the project.
3. Using the tools' output to guide you, **inspect** the parts of the project whose design is more questionable.
4. Write down your findings in a **report**.

   Maximum length of the report: 7 pages (A4 with readable formatting) including any pictures and code snippets.

The assignment must be done in *individually*.
This assignment contributes to **28%** of your overall grade in the course.

[1] Comments count as lines of code, whereas empty lines do not.

2 Tools and other resources

2.1 How to find projects on GitHub

See the instructions in Assignment 1.

2.2 Tools to detect design flaws

You should use at least one flaw detection tool, but you can also use more than one. If you select a single tool, you should explore different options and rules offered by the tool to get full points. Usually, a tool's analysis checks a fixed set of basic rules by default, but you can select additional rules (if you think they are more useful for the project you are analyzing) or disable some default rules (if you think they are irrelevant or ineffective for the project). If you use multiple tools, you should compare the results of running them on the same project.

We suggest two tools that can detect a variety of design and coding flaws and bad practices:

PMD: https://pmd.github.io/

SonarQube: https://www.sonarqube.org/

How to use PMD: Basic usage (from PMD's `bin` directory):

```
$ ./run.sh pmd -d /path/to/project/root/sources/                ## project path
               -f html                                          ## output format
               -rulesets java-basic,java-design,java-unusedcode ## analysis rules
               > /path/to/output/report.html                    ## output filename
```

For more options (and what each rule does) see the online documentation or run `./run.sh pmd -h`. PMD only requires source files to run.

How to use SonarQube: Download SonarQube community edition, as well as SonarQube Scanner[2] for your computer architecture. Basic usage:

1. From SonarQube's `bin` subdirectory for your computer architecture (such as `bin/macosx-universal-64`) launch the console:[3]

   ```
   $ ./sonar.sh console
   ```

   and wait until the message "`SonarQube is up`" appears on screen.

2. In a different terminal (or after switching the console's process to the background):

   a) Add SonarQube Scanner's `bin` directory to the path

   ```
   $ export PATH="$PATH:$HOME/sonar-scanner/bin"
   ```

   b) Go the the project's root source directory

[2] At https://docs.sonarqube.org/latest/analysis/scan/sonarscanner/.

[3] The script to start the console is called `sonar.bat` in the Windows distribution.

c) Run the analysis by calling:

```
sonar-scanner -DprojectKey=projectName
```

where `projectName` is any identifier that will be used in the report to identify the project.[4]

3. Once the analysis terminates, open a browser to `http://localhost:9000` to inspect the results.

SonarQube requires source files as well as class files for Java projects.

Other languages? PMD and SonarQube support multiple source languages. However, some analyses may not be available for a certain language. If you feel adventurous, or simply prefer to work with languages other than Java, you can apply PMD and SonarQube to projects written in a language other than Java, or even look for similar analysis tools that work for other languages. For example, there exists ReSharper for the .NET languages (https://www.jetbrains.com/resharper/) and Cppcheck for C++ (https://cppcheck.sourceforge.io/). If you choose to use other tools and languages, please check with the instructors that your choice of tools and language is reasonable before working on the assignment.

3 What to write in the report

3.1 Report content

The structure of the report is free; it should include all the significant findings emerged during your work.

Things to include in the report's discussion:

- the project selection process, including mentioning projects you discarded; notice that choosing a suitable project, with a variety of features that can be analyzed with a flaw detection tool, is an integral part of the assignment's work;
- a short description of the selected project with some stats (size, number of contributors, whether it's a library or application, and so on);
- whether you analyzed all project or only some parts of it (e.g., did you analyze testing code as well?);
- a discussion of what predefined rules you selected for your project analysis and why;
- a quantitative summary of the tools' output, such as a table with how many problems have been found, of which kind, by what tool, and so on;

[4] If you need to run the tool multiple times with custom options, you can also create a text file named `sonar-project.properties` with the various command-line options.

- whether you found any false positives in the tools' output, that is reports of rule violations that should not actually be considered violations;[5]
- a qualitative discussion of the kinds of problems the tools are more or less likely to report;
- a comparison of the tools' output at a high level (if you used multiple tools), or of the tool's output with different options (if you used a single tool)
- your assessment of the project's design quality based on the tools' analysis combined with your own judgement; if possible, try to relate your findings to specific characteristics of your project (e.g., it's a library, or it depends on legacy projects, . . .).

3.2 Tips and tricks

Do not trust the tool's output blindly; even when the tool's results are sound, try to understand the origin of the problems to come up with higher-level and more interesting findings.

4 How and what to turn in

Turn in your **report** as a single PDF file using *iCorsi* under *Assignment 2.*

4.1 Plagiarism policy

Students are allowed to generally discuss assignments and solutions among them, but each student must work on and write down their assignment independent of others. In particular, both sharing solutions of assignments among students and reusing solutions of assignments done by students of previous years or by anyone else are not allowed and constitute cheating.

In a similar vein, you are allowed, in fact encouraged, to loop up any examples and material that is available online you find useful; however, you are required to write down the solution completely on your own, and, if there is publicly available material (such as a website, blog, paper, or book chapter) that you based your work on, credit it in the report (explaining what is similar and how your work differs).

ChatGPT & Co.

The plagiarism policy also applies to AI tools such as ChatGPT or CoPilot:

1. You are allowed to get the help of such tools; however, you remain entirely responsible for the solution that you submit.

[5] You can estimate the fraction of false positive by manually inspecting the tool's output (or a smaller sample of the output, if the tool reports very many rule violations).

2. If you use any such tools, you must add a section to the report that summarizes which tool(s) you used and for what tasks, what kinds of prompts you used with the tools, how you checked the correctness and completeness of their suggestions, and what modifications (if any) you introduced on top of the tool's output.

Failure to abide by these rules, including failing to disclose using AI tools, will be considered plagiarism.