

# Software Development, Maintenance and Operations

## 811372A

Fall 2024  
Course Project - Option B

September 16, 2024

INSTRUCTIONS: You may use any books, references, notes and programs. The course already provides some supporting material to complete the course projects. Please provide your codes (preferably *Python*), output, and brief comments in a file and submit it to Moodle. Likewise, write your *name* and *student number* in your file. (*Recommendation*: For a structured submission procedure, consider using a *version control* platform (e.g. GitHub) to keep track of your work and provide it within the submission file).

PRESENTATION: Source code **refactoring** is a well-established approach to improving source code quality without compromising its external behaviour. Providing refactoring **recommendations** closer to what developers perceive as relevant may support the broader application of refactoring in practice. Therefore, you will **mine** the refactoring activity of the involved software projects, and further, you will capture the developers **effort** performing such activity. Similarly, you will put hands on the **issue tracking system** as an advanced data mining research experience.

Consider the dataset provided in the following link <sup>1</sup>.

- (a) Download the zipped file. Take a look on the *project* column. Get the unique instances. Find the pattern to obtain the GitHub links of the listed projects. (**Hint**: All the included projects are inside the *Apache* source foundation.)
- (b) Clone the GitHub projects into your machine from the built GitHub links. (**Hint**: *subprocess* library might be *one* option.)
- (c) Mine the refactoring activity applied in the history of the cloned projects. For that, you might use *RefactoringMiner* library <sup>2</sup>. The provided link contains detailed steps for using this tool. **Hints**:
  - First, *build* the tool into your machine, it can be anywhere but make sure to remember the path.
  - Looking at the data dimension to be mined, you may consider using the CLI options for *RefactoringMiner* (You should mine all the commits from each repository.)
  - If you are using Python, take a look, for instance, at the *subprocess* library to run commands in the CL.
  - Provide for each project a table presenting the total number of refactorings made of each refactoring type and the average time of the inter-refactoring periods.
- (d) Calculate and collect the *diff* change between the detected commit and the previous commit. (**Hint**: You might use *pydriller* library if you are using Python for this project.)
- (e) Collect developers *effort*. Collect the total count of touched lines of code (TLOCs) for each refactoring and each developer. (**Hint**: You might need *subprocess* library and the *scc* tool seen in Weekly exercise lectures.)

---

<sup>1</sup><https://filesender.funet.fi/?s=download&token=b46c8a2d-7f03-4184-9cdf-baad0c210d0e>

<sup>2</sup><https://github.com/tsantal/RefactoringMiner?tab=readme-ov-file#general-info>

(f) Mine *bug-fixing* commits.

- Check in GitHub whether the projects are using GitHub as Issue Tracking System (ITS) or not (**Hint(s)**: Check out GitHub's REST API documentation. You might need *Requests* library to work with APIs.)
- For projects using GitHub as ITS, mine all the issue data. (**Hint**: Beware of the API request rate limit. More info in the GitHub docs.)
- For projects not using GitHub as ITS, a special lab lecture will be announced for JIRA API data collection.

(g) **NOTES:**

- Think about the logic to perform the data mining process. (**Hint**: Cloning all the projects may overload your machine.)
- The submission of this part must consider all the collected data, the scripts utilized and a brief report on each with the reasoning of each of the steps performed.

(h) **SUBMISSION FORMAT:**

- The output from the RMiner should create a **json** file, please name is with the repository name, or within a folder with the repository name (e.g. "../accumulo/rminer-output.json" or "../rminer-outputs/accumulo.json")
- The output for the commit *diff* could be a json file with attributes "commit\_hash", "previous\_commit\_hash", "diff".
- The output for the developers' effort could be a csv with headers such as ("refactoring\_hash", "previous\_hash", "TLOC").
- Consider for instancer compression the submission through the attached file compression engine <sup>3</sup>. Choose the link generation option.

**BONUSES: Anyone in the course completing the following points should be confident on passing the course with high grades.**

- **UNIQUE BONUS:** Reporting the data collection process and the structure of the mined data in the following LATEX template <sup>4</sup> (You can open it directly in Overleaf for instance).

---

<sup>3</sup><https://filesender.funet.fi>

<sup>4</sup><https://www.overleaf.com/latex/templates/acm-conference-proceedings-primary-article-template/wbvngbjbwpc>