Mining Software Repositories

Assignment 1 – “Lucene study”

Kyriakos Fragkeskos 4516923
Mengmeng Ye 4407326
Anelia Dimitrova 4501667
Delft University of Technology, the Netherlands

Abstract

We want to replicate the study [1] done by Bird et al. and published in FSE'11: to examine the relationship between different ownership measures and software failures, but not in the software projects from that study. Instead, we will use an OSS system. We will be working on Lucene [2], a search engine written in Java.

Phase I – Data Collection

The first phase for a data mining project is always collection of the data to be worked on. We will consider the relation between ownership information and post-release bugs. The investigation is on file level, since the project is java based and every class has a separate **.java** file. At the end of the research, we will have a CSV file, containing number of major, minor contributors, ownership, number of bugs and file name.

Using Python and terminal, the following steps are performed:

1. To collect the github data, we need to make a local copy (clone) of the repository [3] for faster computations.
2. We decided to compare the data as per release 4.9.0., so we first checkout to the release:

```
import sh

git = sh.git.bake(_cwd='lucene-solr')
git.status()


shaV490 = (git("rev-list","-n 1","--before=\"2014-09-21\"","trunk")).stdout[:-1]
shaV490
// output (Hash): 6ba94d19538518afb6f56b097e0f3c6c6275cb06

git.checkout(shaV490)
git.status()
```

3. Next, the command below is executed in terminal in order to get the lines for all .java files:
   git ls-files | grep "\.java$" | xargs wc -l > ~/Desktop/fileLines.txt
4. Then the following command is executed in the terminal again to get the before release file:

git log --numstat --pretty=format:"%cn,%H,%f" > ~/Desktop/beforeRelease.txt
Placeholders: %cn – committer name, %H – commit Hash value, %f – subject line

5. We need the hash of the commit from 1st of Aug 2015, so we checkout to the current release and get the commit hash id as follows:

```
import sh

git = sh.git.bake(_cwd='lucene-solr')
git.status()

shaAug15 = (git("rev-list","-n 1","--before=\"2015-08-01 00:01\"","tru
nk")).stdout[:-1]
shaAug15
// output (Hash): 8260dfd75163d40e40744362bee5f1693f9415b9

git.checkout(shaAug15)
git.status()
```

6. To get all the files (bugs) up until Aug 2015, so finally, we execute the following command in the terminal:
git log --numstat --pretty=format:"%f" 6ba94d19538518afb6f56b097e0f3c6c6275cb06 8260dfd75163d40e40744362bee5f1693f9415b9 > ~/Desktop/AfterRealease.txt

- Our exported raw git data is located at:
https://drive.google.com/open?id=0B-lxy7DLSUV-M3Q5OFNyNTMxSFE

Phase II – Realization & Metrics

Metrics:

At the end of the assignment we will need to have data for the following metrics:

- **Minor**: number of minor contributors (contributors with less than 5% ownership)
- **Major**: number of major contributors (contributors with more than or equal to 5% ownership)
- **Total**: total number of contributors
- **Ownership**: proportion of ownership for the contributor with the highest number of commits

In order to gather the bugs, we need to go through the .json issue archive. The range of files that is in the time range we need, is approximately from file 12723002 to file 12743006. We will be looking for “issuetype” = “Bug” and get the key of the file, which equals to “LUCENE-XXXX” where XXXX refers to a 4-digit number. The key

helps us to relate the bugs to the files we have already exported and make other calculations later on.

Approach 1:

Java program was developed for checking through the files and getting the required information for the computations we need.

Arrays were created to hold the values for the file names, contributors' names and the total commits of every developer. A two-dimensional array holds values of the commits and the name of the user and the file name, i.e. developer X has made 20 commits to file Y.java.

The code workflow is as follows:

1. Scanning the log file, we find the name of the contributor and the file name for each line on the log.
2. Checking the names array for the last found name. If the name is not yet in the in the array, we store it in there.
3. Checking the files array for the last found file name. If the file name is not yet in the in the array, we store it in there.
4. Adding 1 to the commits two-dimensional array
5. Get the total number of commits for each file after scanning it all.
6. Calculating the minor and major and total contributions as well as the ownership as per the given paper.

Approach 2:

Java program was developed for checking through the files and getting the required information for the computations we need.

A package was created to perform the following operations:

- Transform raw data into flexible format (e.g line per commit)
- Look for the files names and save them in a file
- Get the indexes of the contributor and the file name.
- Calculate the ownership
- Find and Save the major and minor values
- Find the maximum percent of ownership
- Find the bugs
- Find the number of bugs for each file

Another Java application was developed, using the package above, to calculate the contribution and execute the operation from the package. This code is reusable and adjustable for any other version of Lucene with the appropriate raw input data.

Java source code, CSV and Report:
https://github.com/KyriakosFrang/MiningReposAssignment1

References

[1] http://dl.acm.org/citation.cfm?doid=2025113.2025119
[2] https://lucene.apache.org/core/
[3] https://github.com/apache/lucene-solr