First assignment

Motivation: Software engineers make different types of design decisions, such as to create components or select technologies of a system. However, we know little about the impact of these decisions on the source code design, size and quality. This makes it hard to identify decisions from source code.

Goal: Explore differences and correlations in code quality and design changes for the different types of design decisions.

Each group will focus on a specific Java (sub)-project with a specific list of issues (attached). Each issue is classified based on three types of design decisions.

Group	Project	Code
Group 1	Hadoop common	https://github.com/apache/hadoop
Group 2	Hadoop common	
Group 3	Hadoop common	
Group 4	HDFS	
Group 5	HDFS	
Group 6	Yarn	
Group 7	Yarn	
Group 8	Map-Reduce	
Group 9	Cassandra	https://github.com/apache/cassandra
Group 10	Cassandra	
Group 11	Cassandra	
Group 12	Tajo	https://github.com/apache/tajo

To achieve this goal, I propose these guiding steps in the following weeks:

1st week: **Explore software repository**.
In this week, you will explore the software repository of the projects you will analyze. Specifically, you will make the following steps:

1. *Determine commits that implement design issues*: For each project, you are provided with a list of issue IDs (see attachment). Each issue ID is associated with one or more types of design decisions. In this step, we would like to determine the commits that implement these issues, and their parent commits. Usually, the issue IDs are added in the commit messages. Thus, you will need to loop on the commits to find the commit(s) that implement each issue using PyDriller. For issue IDs without a commit, we can exclude them for now. After this, you need to determine the parent commits.

2. *Compile the software automatically through script*: You need to be able to compile your project in an automatic way to be able to run the clustering algorithms in the next week. Thus, create a script that can loop on the given commits, and compile the project, create a jar file or .class files. This can be challenging, because sometimes the setting of the project change by time. Thus, try to rank the commits chronologically to know, which commits can be compiled in the same way, and adjust your scripts to be able to compile all commits.

3. *Determine size and quality information of commit changes*: Using PyDriller, you can determine the following information for each commit:

- No. of added, modified and deleted files. (per commit)
- No. of added and deleted lines of code (per file)
- No. of added, deleted, and changed methods (per file)
- DMM metrics (per commit)
- Complexity (per file)

4. *Generate charts*: Given the collected size and quality information per commit, you can create charts such as boxplots that presents the size and quality information per decision type, remember that each commit is associated to an issue, which is associated to a decision type.

2nd week: **Run clustering algorithms.**
In this week, you will run four clustering algorithms (WCA, Limbo, ACDC, and PKG) on the source code of each commit (including the parents).
To execute clustering algorithms, you need perform the following steps:

1. *Extract the dependencies of each commit*. We need the dependencies as the features to execute the clustering algorithms. To do this, you need to create a script that call the "JavaSourceToDepsBuilder" jar file (attached). It takes the following parameters:
    - classesDirPath or jar file
    - path of depsRsfFilename
    - path of ffVecsFilename
    - packagePrefix to filter dependencies.

The "JavaSourceToDepsBuilder" jar file produces files with dependencies in an rsf format.

2. *Run PKG method on the dependencies file of each commit.* You need to write a script that call the "Pkg" jar file on each of the dependency's files. Check documentation to execute it.
3. *Run ACDC method on the dependencies file of each commit.* You need to write a script that call the "ACDC" jar file on each of the dependency's files. Check documentation to execute it.
4. *Run WCA and Limbo algorithm on the dependencies file of each commit.* You need to write a script that call the "Clusterer" jar file on each of the dependencies files. The "Clusterer" jar file takes many parameters. Check documentation to execute it. For both algorithms, experiment with different numbers of clusters, a small number (12), and a number like the number produced by PKG and ACDC. For WCA, execute it both using *UEM* and *UEMNM similarity measures.*

5. Given the result of each clustering algorithm, write a program that can help answering the following questions:

3rd week, **Analyze the results of clustering algorithms.**
In this week, you will analyze the results from the clustering algorithms to determine the following:
- How big are the clusters for each clustering algorithm for each commit?
- How do clusters from different algorithms intersect?
- How big are the architectural changes for each type of design decisions, and for each clustering algorithm?

To answer the above questions, you need to write a program that analyze the results of clustering algorithms. To calculate the a2a metric and cvg metrics, you can call their jar files and see their documentation. Calculate a2a and cvg.

4th week

- Apply statistical analysis.
- Create charts.