GROUP 7

Hadoop-Common(Group2)

Data Science for Software Engineering

Report

Professor:

Dr.Mohamed Soliman

Submitted By

Name	**Matr. No.**
Niharika Aggarwal	6988971
Jayesh Chavan	4010138
Muhammad Ammar	4012525
Salah	6956862
Vaibhav	6880720
Sayak Mitra	6987519

.

Introduction

Knowing how design choices affect software codebases is essential to maintaining the quality, maintainability, and scalability of the code.

Nevertheless, little is known about how these choices impact the code, which makes it challenging to identify the root causes of problems with code quality and make wise choices during the development stage.

The purpose of this study report is to investigate how various design choices relate to the quantity, quality, and design of source code. To shed light on the differences in code quality resulting from different design choices, we attempted to provide a set of guidelines for this investigation. These processes included repository analysis, commit data inspection and visualization generation.

Study Design

The proposed steps include:

WEEK 1: Exploring the software repository.

1). Determining the commits that execute the design issues: The initial stage involves the examination of the software repository. We used PyDriller to extract commit information and parent commits to trace how these choices are made throughout the codebase and we hoped to find the commits associated with the particular design flaws.

2). Automating Software Compilation: We created a script to build the project automatically at various commit points to make the analysis easier. This script ensures compatibility with the compilation process by taking into account possible changes in project parameters over time. After the compilation we need to run the clustering algorithms so for that we need to create jar files or .class files.

3). Analyzing Commit changes: Using PyDriller, we extract relevant information from each commit, including the file changes, lines of code, methods, DMM metrics, and complexity metrics. This data provides insights into the size and quality of commit changes, enabling assessment of design decision impacts on the codebase.

4). Generating Visualizations: Using size and quality data gathered every commit, we produce boxplot-style visualizations that show how code metrics vary among various design choices. By associating each commit with its corresponding issue and decision type, we analyze the relationship between design choices and code quality.

WEEK 2: Run clustering algorithms.

For Week 2, We first fixed the issue of compiling jars which was not working before Previously we had included the Parent commits in our previous work.

So, we were left with the task of improving the analysis based on the feedback we got. The number of lines of code was normalized. Also, we differentiated between lines of code and configuration lines of code. We can also differentiate between the number of lines of code on the whole commit and per code file or configuration file and proceed with the tasks of week 2.
We ran four clustering algorithms (WCA, Limbo, ACDC, and PKG) on the source code of each commit (including the parents).
1). Extract the dependencies of each commit: Based on the previous steps we got jars files after compiling the jars of the commits and their parent commits too. After that, we wrote a script to run the “JavaParser” to analyze jar files and generate the dependencies in our cases. This is a requirement for running the clustering algorithms. By passing the required parameters to run this jar file we got files with dependencies in an rsf format.

2). Run PKG method on the dependencies file of each commit: After that, we ran the pkg method on the dependencies files (.rsf) files for each commit. We wrote a script that calls the ‘Pkg’ jar file on each of the dependency’s files.

3). Run ACDC method on the dependencies file of each commit: Furthermore, we ran the ACDC method on the dependencies files (.rsf) files for each commit. We wrote a script that calls the ‘ACDC’ jar file on each of the dependency’s files.

4). Run WCA and Limbo algorithm on the dependencies file of each commit: Next, we ran the WCA and Limbo algorithm on the dependencies file of each commit. We wrote a script that calls the “Clusterer” jar file on each of the dependencies files by passing the necessary parameters. To experiment with different clusters, we tried to give a small number =12 and a number like the number produced by PKG and ACDC. For WCA, we executed it both using UEM and UEMNM similarity measures. For LIMBO, we executed IL parameter.

WEEK 3: Analyze the results of clustering algorithms

For week 3, we have done the following steps to analyze the clustering algorithms:

1) Number of clusters per clustering algorithm per commit: We have determined the number of clusters per clustering algorithm per commit. Here, we iterated through each commit. For each commit, we applied the clustering algorithm. Then, we counted the number of clusters generated by the algorithm. This is then stored as the number of clusters for each commit and each clustering algorithm.

2) Number of Entities in a Cluster per Clustering Algorithm per Commit: We wrote a code snippet to read the files generated by the clustering algorithm. Then we parsed the files to identify clusters and count the number of entities (e.g., classes) in each cluster. This is then stored as the count of entities in each cluster for each commit and each clustering algorithm.

3). A2a and CVG Metrics for Architectural Changes: We have used Option 1, where we called the jar files directly. Then, we executed the jar files that calculate A2a and CVG metrics, directing their output to a file. This then parsed the output file to read the values of A2a and CVG.

4). CVG Metric between Results of Different Clustering Algorithms for the Same Commit: We applied different clustering algorithms to the same commit and calculated the CVG metric between each pair of clustering algorithms. We stored the CVG metric values for each pair of clustering algorithms and each commit in CSV files.

Results

We had to switch the repository from "**YARN"** to the "**Hadoop common**"project due to issues in yarn. So we did all the tasks of Week 1 and Week 2 again. From where we got quite several commits and parent commits which we stored in a file and then checked if these commits were checkout and if yes then generated jar files for both commits and parent commits.

From these jar files again we had to run the parser jar file to extract dependencies for each commit. Resulting in rsf and fv files.Out of which the rsf files were passed to the clustering jar files PKG, WCA, LIMBO, and ACDC to generate a different number of clusters.
Then we switched to week 3 tasks where we analyzed the number of clusters per clustering algorithm per commit. And also calculated the number of entities in a cluster per clustering algorithm per commit using a small code that reads the files and counts the number of classes per cluster which called the jar files directly and then directed their output to a file and then tried to analyze the results to read the value of a2a and cvg.

After that, we used a2a and cvg metrics for calculating the architectural changes between a commit and its parent commit, per the clustering algorithm. The last part was where we

calculated the cvg metric between the results of the different clustering algorithms for the same commit. To conclude, we stored the results in CSV files so that we could use them for the next step of statistical analysis.

Discussion

WEEK1:

1. Overall, we can see that Existence and Property decisions are taken more frequently.
2. Executive decisions exhibit the lowest complexity among the analyzed types.
3. Existence decisions tend to involve significant structural changes, while Property decisions focus more on refining system attributes.
4. Executive decisions, although less complex, still result in a considerable number of modified methods, reflecting changes in a business environment or development processes.

WEEK2:

1. The number of modified methods is notably higher than added or deleted methods, indicating more refinement and optimization rather than wholesale changes.
2. The existence category stands out with the highest total count of updates.
3. Executive category has moderate values for DMM metrics, indicating moderate levels of interfacing, complexity, and size within the executive files.
4. Property files exhibit moderate to high values for DMM metrics, particularly in terms of complexity and size.
5. Text files generally have lower values for DMM metrics compared to other file types, indicating lower interfacing, complexity, and size.
6. Config files show relatively lower values compared to other categories across all DMM metrics.

WEEK3:

1. Will discuss the output with the faculty to cross-verify.
2. WCA LIMBO taking too much time.
3. Clusters by PKG > Clusters by ACDC.

Threats to Validity

Week 1

1. Possible issues in 3rd party library used
2. Unidentified issues in our code
3. Imbalanced distribution of issues/decisions

Week 2

1. Possible issues in 3rd party library used (Arcades)

WEEK3:

1. Possible issues in 3rd party library used (Arcades)

Conclusion and Future Work

Weekly Updates

Time Spent (Week 1)

Name	Time Spent(Hrs)	Tasks Done
Niharika	10hrs	Step 2(Script for compiling) and Step 4(Charts and Report)
Jayesh	9.5hrs	Step 2(Java Compilation), Step 4(Charts and Report)
Muhammad Ammar	12hrs	Step1, Step3
Salah	12 hrs	Step1, Step2(some debugging), Step3(help in code implement)
Vaibhav	10 hrs	Step1, Step3
Sayak	10hrs	Step 2 (Java Sciprt for Compilation, Report)

Time Spent (Week 2)

Name	Time Spent(Hrs)	Tasks Done
Niharika	12 hrs	Step 4(Script and output for WCA and LIMBO Algo) and (Charts and Report)
Jayesh	14 hrs	Jar generation, Charts, Report
Muhammad Ammar	0hrs	Out of city
Salah	10 hrs	Step 1 (parser script and RSF and FV files output) Step 2 (PKG script and output)
Vaibhav	0hrs	Health issues
Sayak	12 hrs	Step 3(Script and output for ACDC Algo),Step 4(Script WCA Algo and output testing), Report

Time Spent (Week 3)

Name	Time Spent(Hrs)	Tasks Done
Niharika	8hrs	Step 4 Again (Script and output for WCA and LIMBO Algo) and (Report)
Jayesh	14hrs	Generating jars again; troubleshooting; organising; report; Week 3 step1&2
Muhammad Ammar	8hrs	Step 1 Again of Week 1 and Step 4 of Week 3
Salah	12 hrs	Week 2:Step 1 Again (parser script and RSF and FV files output) and Step 2 (PKG script and output) Week 3: 3 tasks
Sayak	8hrs	Step 3 again (output for ACDC Algo), Step 4 again(

		WCA,Limbo execution), Report
Vaibhav	**0hrs**	**Dropped course**