Identifying and Analyzing Engineer Roles from Github Repositories

Evangelos Tsimpouris, Charalampos Papadiakos
Electrical Engineering and Computer Science
Aristotle University of Thessaloniki
Thessaloniki, Greece
tsimpour@ece.auth.gr, charaldp@ece.auth.gr

***Abstract*—The aim of this research is to mine a collection of GitHub repositories for useful information regarding engineer roles. At first a general model is introduced. The goal of this model is to find similarities between engineer recordings activity. Then try to analyze and identify if these recordings represent a major software engineer role. In the end some specialized models are deployed in order to analyze further the groups that emerged based on the quality of their features.**

Keywords—mining software repositories; Devops; GitHub;

I. Introduction

The way IT industry functions is going though a wave of change. The traditional software development process has given its way to a more agile model, suitable to deal with modern age requirements. Part of this change was the creation of the term DevOps. DevOps is the practice of operations and development engineers participating together in the entire service lifecycle. This role was introduced in order to increase time-cost efficiency and deliver high quality software products.

Another major change in the field of IT was the introduction of web-based hosting services for version control regarding computer code. Most notable one the GitHub service. GitHub allows to view/write/edit code not only for professionals working on a project but also users that can access the project. It also allows comments and opening issues threads when for example a bug is found. When a bug is fixed the issue thread usually closes. Apart from the user created the issue Project owners and admins also have the right to close issues.

In our case traditional engineer models predict that the developer team will write the code and then commit it to the GitHub repository. Also, they predict that the operations team will monitor performance through the issues created. The modern approach requires a Devop team that will combine all the skills mentioned above.

II. Research Overview

Section III will provide a brief overview of the models' design, the hypothesis made and the algorithms used. Section IV shows the general model results and Section V provides an analysis of these results. Section VI presents the specialized models results and Section VII gives an analysis based on these results.

III. System Design

A. System Overview

The main design for every model introduced in this research is presented in the following flowchart:

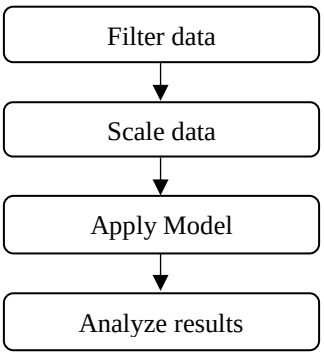

Figure 1:System Overview

The flowchart explains that before each model is applied there is some filtering and data processing. It is important for the model to work that it is not provided with "bad" instances. After the model processing is complete an analysis is made to interpret its results.

B. Data Preprocessing

The Data Preprocessing procedure is conducted at many levels. First the basic engineer activity features such as Commits_Authored, Issues_Opened, Issues_Closed and Issues_Participated are scaled within each project so that they are independent from the project size.

Next the dataset is filtered to remove outliers or bad recordings. To achieve this the vanilla dataset is filtered according to the following criteria:

- The commit rate (Active Commits) in a project exceeds a certain threshold defined by the "knee" in the feature's distribution function
- The participance in project issues exceeds a threshold defined similarly as above.

Dataset instances that do not meet any of these criteria are omitted since a conclusion regarding the role of inactive users cannot be made.

This basic preprocessing procedure is presented in the following flowchart:

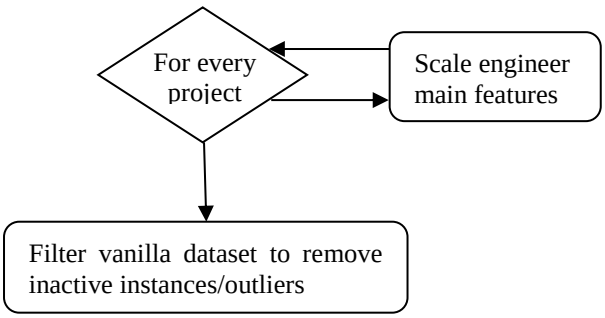

Figure 2:Preprocessing Method

C. General Model Construction

Hypothesis: the engineer roles in each project vary, and the roles diversity is heavily depended on each project's size in workforce. For example, in smaller groups it is common to find engineers who are active in many areas and have a much more general role. In contrast it is assumed that big groups have a bigger distribution of roles, some with specific responsibilities, some with more general ones.

Therefore, it was vital to apply the general model in regard of a project's size in workforce.

The main approach to achieve this was to split the filtered dataset to instances belonging to small, medium and big projects. Then apply the general model to each dataset individually.

The general model is presented in this flowchart:

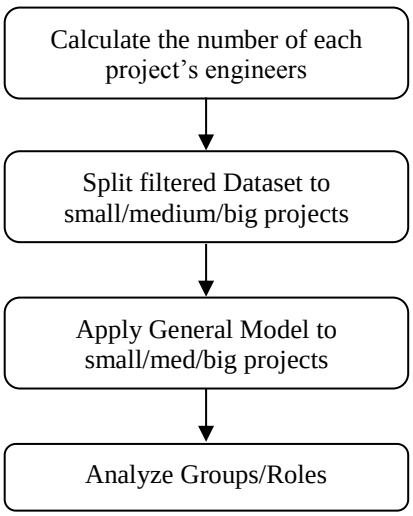

Figure 3:General Model Diagram

Projects Workforce distribution:

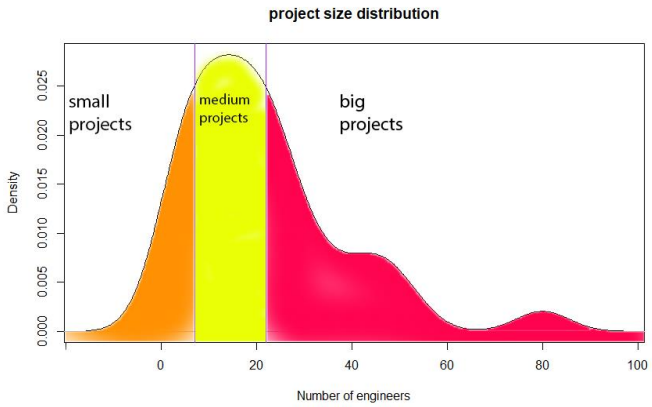

Figure 4:Project Separation by Workforce

Features Selected: The following is a list of features considered to summarize a developer's role in a project.

1. Commits Authored: the number of commits a developer has authored.
2. Issues Participated: the amount of issues an engineer has participated. An engineer might participate to an issue either to fix a bug, add a feature etc.
3. Issues Opened: the amount of issues an engineer has opened. An issue might be opened to address a bug or perhaps add an idea for future development.
4. Issues Closed: the amount of issues closed by this developer, either because a feature was completed or because a bug was removed.

Algorithm selected:

K-Means was selected to group the developer roles in this dataset because of the following advantages:

1. Easy to implement.
2. With a large number of variables, K-Means may be computationally faster than hierarchical clustering.
3. K-Means may produce higher clusters than hierarchical clustering

An example of K-Means clustering

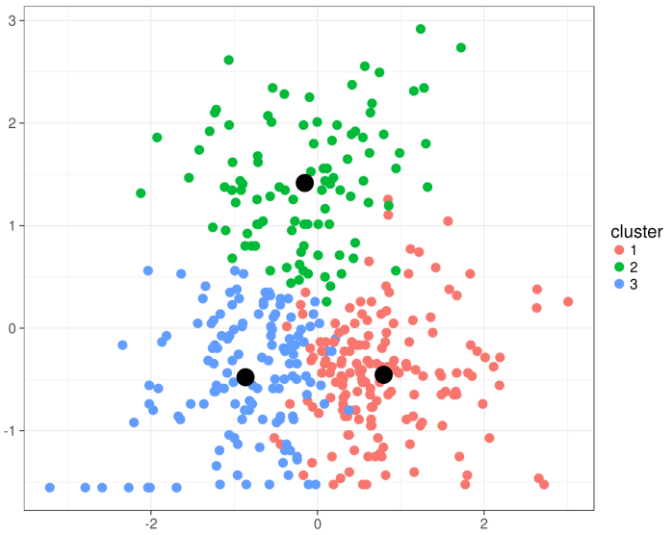

Figure 5:K-Means Clustering

Construction of general model:

K-Means algorithm was executed in 20 iterations, in each iteration different number of centers was chosen ranging within 2 to 13 centers.

Then an average cohesion for all executions of all different number of centers was calculated.

To cluster the main roles an optimal number of clusters was selected around the "knee" of the cohesion plot. Meanwhile separation was checked to be big enough for the chosen number of groups.

D. Special Models Construction

The results of the general experiment provide us with some idea about the developer's roles that exist in a project. However further analysis was needed though to find more specific details of each role that emerged from the original experiment.

For this purpose special models were created in an attempt to first find an then analyze each group that was found.

All the different special models share a similar logic with the general model. Kmeans was also used here. An average cohesion is measured for every experiment and an optimal number of groups is selected.

The main difference between the general model and the special ones are the features selected. For every special model different features are selected. The main reason for that is that these models try to cluster specific developer's roles and not to identify them in a project.

Project owners:

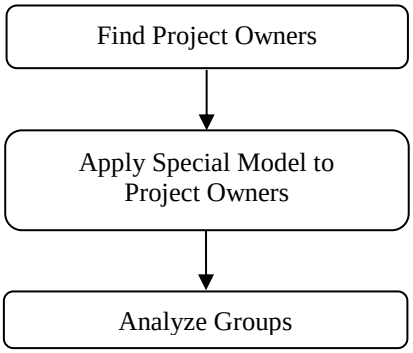

Figure 6:Project Owners Analysis

It is common project owners to be highly active members in a project. Their role includes monitoring performance and write code in high frequency especially in smaller projects.

Hence this category's activity can be easily mistaken with the activity of devops.

In order to successfully identify owners, a different approach from clustering was made. A filter was implemented to cross – check all the repository names and contributor logins. Then create a dataset with all the owners found.

The basic activity features were selected:

1. Commits Authored.
2. Issues Participated.
3. Issues Opened.
4. Issues Closed.

Dev/Devops/Ops:

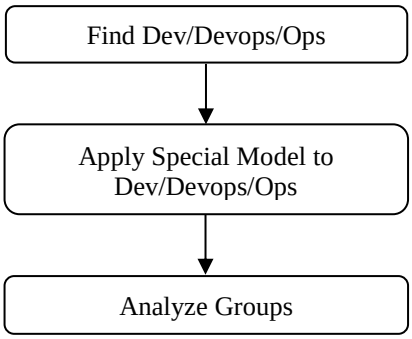

Figure 7:Common Engineer Groups Analysis

Modern projects usually consist of these categories. One,two or perhaps all of these roles can be identified in a project. A model for each of these roles was implemented in order to analyze them further.

First the dataset was filtered and smaller datasets were created each one containing one of the above groups.

At first a filter is implemented to split the dataset between instances with medium or high commit rate and instances with relatively low commit rate. The first group can be roughly considered to constist from pure developers and devops. The second one contains instances that by definition are characterized as pure Ops.

A brief overview of the first filter:

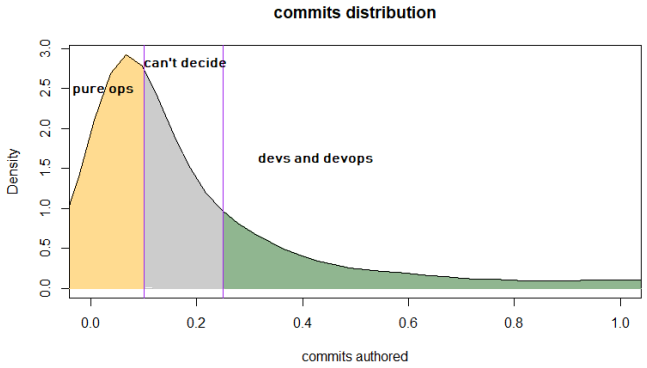

Figure 8:Pure Ops - Devs/Devops Separation

A second filter must be introduced to distinguish pure devs from devops. From the definition of these roles a pure developer has very low participation in issues and comments, whereas a devop has a higher activity in these areas.

A brief overview of the second filter:

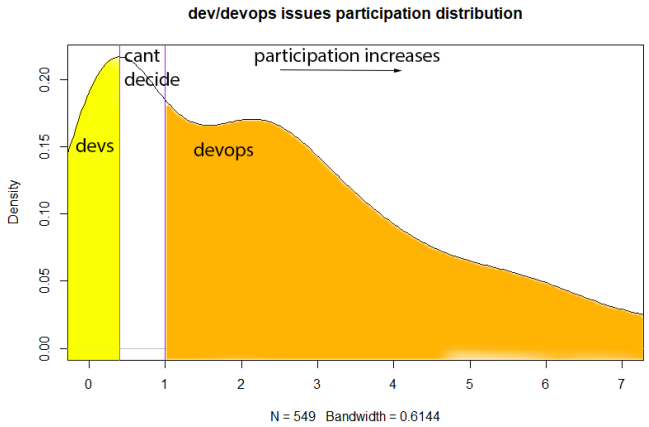

Figure 9:Devs - Devops Separation

After the filters are implemented the different groups are clustered into sub-groups based on their features.

Dev features:

1. issues_closed_per_day: this feature measures the efficiency of a developer and it can be viewed as the amount of work he completes in a day.
2. violations_added_per_loc: this feature measures the amount of coding violations added per line of code. This is an unwanted feature for a developer.
3. violations_eliminated_per_loc: this feature measures the amount of code violation a developer has eliminated per line of code he wrote. This feature is usually high when violations are created in the first place and need to be removed.

IV. General Model Evaluation

Firstly an experiment was conducted in an attempt to find a general pattern of roles in the filtered dataset. Average cohesion was measured after 20 iterations and an optimum choice of groups ranges roughly at around 7-9 groups.

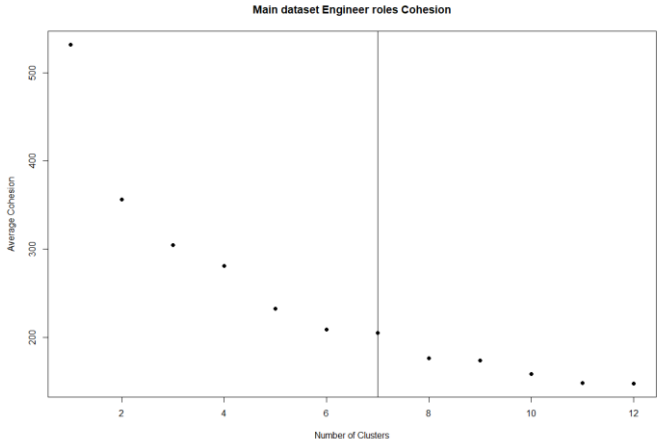

Figure 10:Vanilla Dataset Groups Cohesion

In Fig. 10, it is shown that the minimum average cohesion is relatively high. This is an unwanted behavior providing misleading results. It is assumed the high cohesion was a result of a too general dataset, thus a more elegant approach was made.

It was hypothesized in model construction that the difference between projects may interfere with the roles that exist in them. For this reason, the filtered dataset was split in 3 smaller ones. The split criterion was the size in workforce of each project. The same experiment as above was then run for each smaller dataset individually. The measured average cohesion is presented in Fig 11 to 13:

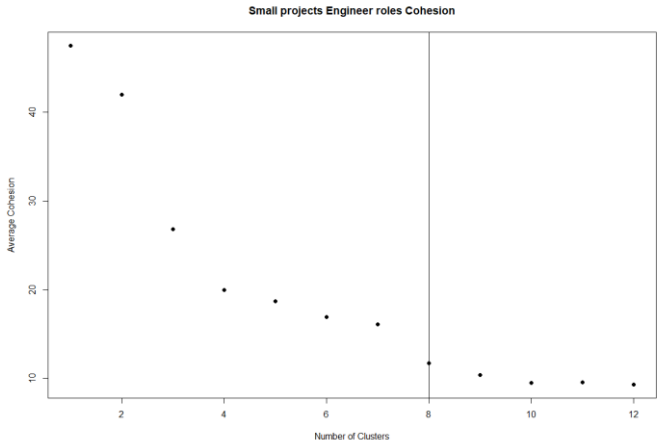

Figure 11:Small Projects Groups Cohesion

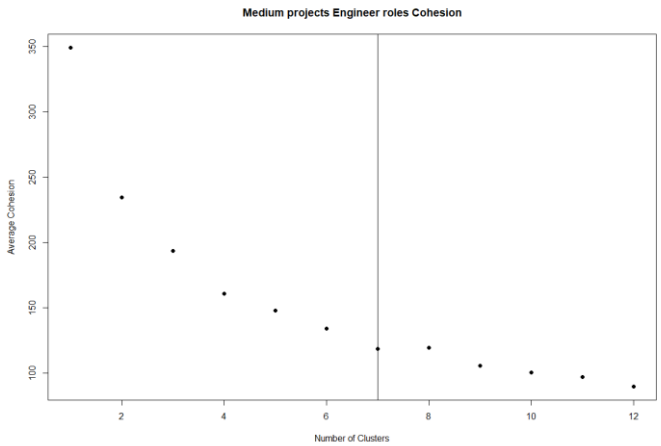

Figure 12:Medium Projects Groups Cohesion

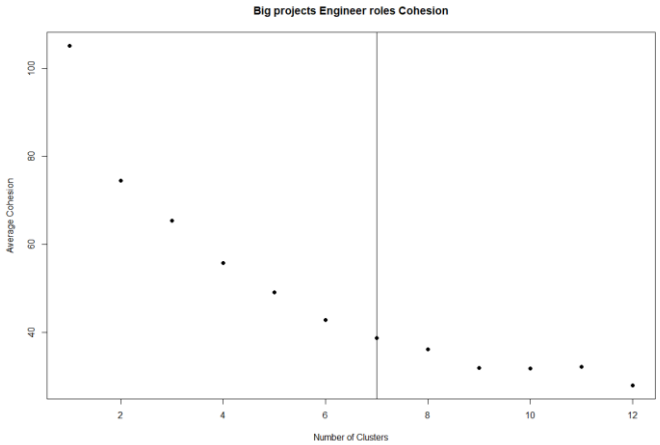

Figure 13:Big Projects Groups Cohesion

The range of the optimal number of groups is similar between both experiments. However there is significant improvement in the calculated cohesion between the first experiment and the medium projects experiment. More importantly there is a massive improvement between small/big projects experiments and the first one.

V. General Model Evaluation Results

First experiment results were interpreted to real life roles and combined to the following table:

Commits Authored	Issues Participated	Issues Opened	Issues Closed	Role
Low	Low	Mid	Low	OP
Low	Low	Low	Low	Undecided
High	High	High	High	Devops

Table 1:Vanilla Dataset Roles

It seems that the first experiment provided obscure results. This was expected as the average cohesion was relatively high as stated previously.

After the second model experiments came through an attempt to translate the results to real-life roles was made. The following tables show a breakdown of the main roles that emerged.

Commits Authored	Issues Participated	Issues Opened	Issues Closed	Role
Mid	Low	Mid-Low	Low	Devops
Low	Low	Low	Low	Undecided
Mid-High	High	Mid-High	High	Devops
Low	Low	Mid-High	Low	OP/Spam?

Table 2:Small Projects Roles

Commits Authored	Issues Participated	Issues Opened	Issues Closed	Role
Mid	Low	Low	Low	Dev
High	High	High	High	Devops
Mid	Mid-High	Low	Mid-High	Devops
Low	Mid-Low	Mid-High	Low	OP
Low	Low	Mid	Low	OP/Spam?
Low	Low	Low	Low	Undecided

Table 3:Medium Projects Roles

Commits Authored	Issues Participated	Issues Opened	Issues Closed	Role
High	Mid	Low	Low	Dev
High	High	High	High	Devops
Mid-High	High	Low	High	Devops
Low	Mid-High	Mid-High	Mid-High	OP
Low	Low	Mid-High	Low	OP/Spam?
Low	Low	Low	Low	Undecided

Table 4:Big Projects Roles

Main Roles Analysis:

Instances with high activity in authoring commits and low activity in every other sector are characterized as pure devs. Furthermore, some groups have high activity only in features regarding issues, those groups are considered to be pure OP. Their general role is to add new features for development or review features in development for bugs etc.

It should be noted that some groups (group 5 medium projects, group 4 small projects) have neither participation in issues nor write any code, but they are opening issues in high frequency. It is safe to note that it is unclear wether these engineers have some role in the project perhaps monitoring performance or they spam a project in development while they are not working on it.

A group with very low activity in every sector was flagged as undecided. It is unclear if this group had some role in a project's development or not.

Lastly instances with high activity in every feature were characterized as Devops . As we saw earlier this is a new model of engineer having a more general role in the development of a project. It is interesting that there are more than one kind of Devops. It seems that there are Devops with higher activity on OP features but have also committed a considerable amount of code. On the other side there are Devops with specialization in the development field but also there is some calculable participation on a project's issues. Lastly there are Devops with very high activity in both OP and Dev matters.

Analysis regarding project workforce size:

Between different size of projects there are some similarities and some interesting differences that will be addressed. It seems that in smaller projects the role of the pure dev is absent, most instances are characterized as devops, ops or perhaps spam. This does make sense since in small groups (less than 7 engineers) developers have to be more flexible and adjust their roles to the project's needs.

Medium and Big projects share more similarities than differences. Certainly, there is much bigger role distribution in contrast to smaller projects. Pure dev groups exist in both datasets, different kind of Devops are also present in both datasets.

It seems in big projects OP roles are present and they have a distinctly higher activity in issue-regarding features in contrast to the ones in mid-size and smaller projects.

VI. Special Models Evaluation

For every group an average cohesion was calculated after 20 iterations. Then an optimal number of clusters was selected.

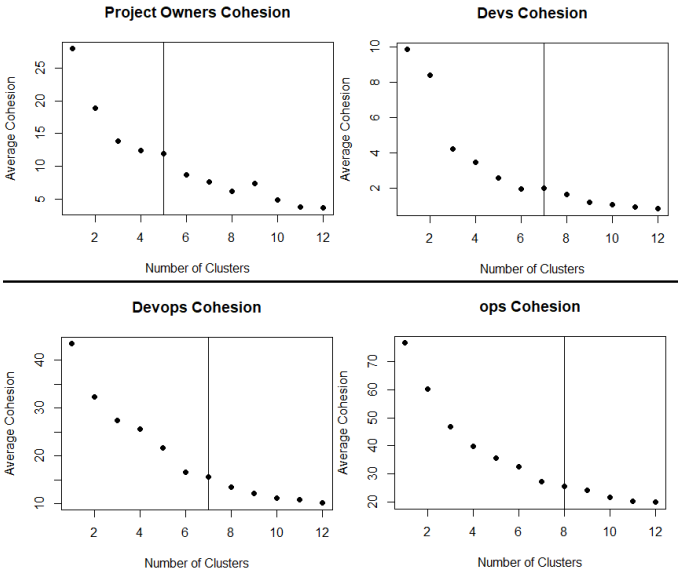

Figure 14:Cohesion of Special Groups
UL)Project Owners UR)Devs DL)Devops DR)Ops

VII. Special Models Evaluation Results

Commits Authored	Issues Participated	Issues Opened	Issues Closed	Role
High	High	High	High	Opening Issues Project Owners
High	High	Low	High	Opening Issues Project Owners

Table 5:Project Owners Roles

Project owner analysis:

Only two types of owners were found, the ones who open issues and the ones who don't. It is vital for a project owner to communicate with both the client and the developer team. As a result, we consider only the first type of owner suitable.

Issues Closed per Day	Violations Added per LOC	Violations Eliminated per LOC	Dev Quality
High	Low	Low	Efficient Clean Coder
Low	Mid	Low	Efficient Clean Coder
Low	Mid	High	Efficient Clean Coder
Low	Low	Low	Efficient Clean Coder

Table 6:Developer Work Quality

Developers analysis:

Only the first group holds the required skills needed for an adequate developer engineer. This group closes issues in high frequency and also ensures the code is working flawlessly. The last group may also write clean code, but they complete their work slow.

Average Comment Length	Average Comments per Issue	Issues Participated	Issues Opened	Op Quality
High	High	Mid	Mid	Comments Participative Open Issues
Mid	Mid-High	Mid	Mid-High	Comments Participative Open Issues
Mid	Mid	Mid-Low	Mid	Comments Participative Open Issues
Mid	High	Mid-Low	Mid-High	Comments Participative Open Issues

Table 7:Ops Work Quality

Ops Analysis:

It seems that only groups one and two have adequate participation in issues related topics but not especially high. Sufficient comments also seem to be present in all groups. The last group opens issues in high frequency, writing big comments but they discontinue their participation in the topics they opened. It is deduced that this behavior is highly unwanted. Group 2 has the most skills required by an Ops engineer.

Issues Closed per Day	Violations Added per LOC	Violations Eliminated per LOC	Average Comment Length	Average Comments per Issue	Issues Participate	Devop Quality
High	Low	Low	Mid	Mid	High	Efficient Clean Coder Comments Participative
Mid-Low	Low	Low	Mid-Low	Mid	Mid	Efficient Clean Coder Comments Participative
Low	Mid-High	Mid-Low	Mid-Low	Mid-Low	Mid-Low	Efficient Clean Coder Comments Participative
Low	Low	Low	Mid-Low	Mid-Low	Mid-Low	Efficient Clean Coder Comments Participative

Table 8:Devops Work Quality

Devops Analysis:

This role should combine the skills of developer and an operations engineer. These skills seem to be present only in the first group where all features are positive except the comments. Writing medium size comments isn't necessarily a bad thing though. Group 2 has some participation but their work in coding is slow. Third group clearly doesn't process the skills required. The last group only positive feature is their bug free code while their participation and efficiency is especially low.

VIII. Conclusions

The key results of this paper are summarized below:

- ➔ The main roles identified from the dataset are those of the project owner / developer / devop and operations engineer.
- ➔ The number of a projects workforce in engineers directly affects the distribution of their roles.
- ➔ Each group has been analyzed further based on the quality of their work..

Further analysis may be needed on the repository owners' group. An owner usually has a complicated role and their responsibilities vary. A future analysis can perhaps find more groups based on different set of features.

A case analysis for each project may also be an interesting aspect for future investigation. For instance an assessment can be made whether or not a GitHub project achieved its goal, or if it was a college project or a diploma thesis etc.

References

[1] I. Zafeiriou, "Software engineer profile recognition through application of data mining techniques on GitHub repository source code and comments," *Diploma thesis*, Aristotle University of Thessaloniki, Thessaloniki, Greece, 2017.

[2] M. Virmani "Understanding DevOps & bridging the gap from continuous integration to continuous delivery" Fifth International Conference on the Innovative Computing Technology (INTECH 2015).

[3] Y. Lu "Does the Role Matter? An Investigation of the Code Quality of Casual Contributors in GitHub" 2016 23rd Asia-Pacific Software Engineering Conference (APSEC)