

**Theses of the PhD thesis**

**An exploration of modern domain specific  
software architectures**

**Ulvi Shakikhanli**

Supervisor

**Vilmos Bilicki, PhD, Associate Professor**

**Doctoral School of Computer Science**

**University of Szeged**

**Department of Software Engineering**

**2024**



# **1 Introduction**

This PhD thesis examines two main repository structure types and the relationship between them and parameters of project like development productivity, software team collaboration, development period and developer team size. To get general results a special database containing more than 50 000 mono and multi repository projects was created. The main goal of this study accretion the connection between repository structure and project development process and see how it affects the overall development procedure. Since it is a narrow area of software development, we created our own unique algorithms for the identification and collection of Mono and Multi repository projects from the GitHub platform. Besides this, a new machine learning approach for the estimation of development productivity and a new mathematical way for calculating software team collaboration were created.

## **2 Background**

The focus of Chapter 2 is to present several key terms that this dissertation frequently utilized. These include repository structures, version control systems, branching strategies, GitHub platform, and Multi repository management tools. The upcoming chapters of the dissertation will discuss all these technologies in much more detail.

## **3 Mono and Multi repository structures**

Chapter 3 of the thesis focuses on the first stage of our research on mono and multi repository structures. Until now there have been only a few research and academic studies that analyse properties of Mono/Multi repository structures [6 - 9]. However, most of these analyses were done either on a local scale hence lacked objectivity or from a narrow perspective which didn't have any general ideas or understanding. In this study I solved both issues by choosing my research project from GitHub, which is the biggest project repository of all time. According to the current data, GitHub hosts more than 420 million repositories by today's estimates. Due to this huge number of repositories, we chose it as a source for our projects. The GitHub platform

provides GitHub API which can be helpful for collecting information about repositories but unfortunately this API doesn't provide any information about the structure type of repository. Because of this, we decided to develop our own algorithm and approach for the identification and collection of Mono/Multi repository projects got from GitHub platform.

### **3.1 Identification of frontend and backend repositories.**

The procedure for the identification of multi repository projects is much more complicated than Mono repository projects. The main reason for this is the structure of the multi repository itself. Since these types of projects contain several repositories, it means that first we must locate this project and find which repositories belong to its frontend and backend sections. We developed a Machine learning model for this purpose [5]. This model was trained on the file structures of previously identified frontend and backend repositories. After several tests and trials, it became clear that using the *Decision Tree* algorithm was the best approach for this purpose. Creating a model takes time because of the long process of retrieving the file structure from several repositories but in the end, we were able to identify frontend and backend repositories with a 90% plus accuracy which is a great achievement.

### **3.2 Identification of multi repository projects.**

After implementing a new approach for the identification of frontend and backend repositories, the main issue was to find potential users who had multi repository projects in their GitHub account. This issue was solved by using a search feature of the GitHub API. An example of a search query was used here. This query was designed to find frontend repositories and it increases our chances to find projects with a multi repository structure.

<https://api.GitHub.com/search/repositories?q=frontend+language:typescript+created:2023-01-01..2023-06-30>

Above we said that there were more than 420 million projects, which made search operations impossible, but this query gave us a total of 1286 repositories. Obviously analysing fewer than 1300 repositories is a much easier and better

approach than employing same random search strategy. After finding the repositories which could potentially have a Multi repository structure, the next step was to identify which frontend and backend repositories belonged to the same project. We used a heuristic approach that considers the name, definition, readme files and other explanatory parameters of repositories. This method was applied with the help of the K nearest neighbour method, and we got an accuracy of 89% in our tests [1]. With the help of this algorithm, it was possible to collect multi repository projects from the GitHub platform automatically. However, there are also some drawbacks of this approach that others should be aware of before using it. As we mentioned before, the final stage for matching the frontend and backend repositories of Multi repository projects is based on the information by repository owners themselves. In some cases, repository owners may write bad definitions for their repositories, and this can lead on algorithm to draw the wrong conclusions. Also working with GitHub Api itself can have its own difficulties [10]. In spite of this, our approach provides a unique way to identify and collect multi repository projects, and this may be extremely useful for researchers in this area.

### **3.3 Identification of mono repository projects.**

The identification process for mono repository projects is like the Multi repository. Here once again we use the file structure of the project to define Mono repository projects. As is known, due to their structure mono repository projects contain most of the essential parts of a project in one repository or directory. Again, we construct a special query to narrow our search field. After obtaining a certain number of potential Mono repository projects, we start to analyse their file structure las we did in the previous case. But here we are looking for some special folder names like *Frontend*, *Backend*, *UI*, *API*, *Client*, *Server*, *UI*, *Front*, and *Back*. The list of folder names can be modified according to the type of research requested. For the sake of clarity, the whole procedure can be described in the following steps:

1. Compile a list of potential Mono repository projects and their respective users.
2. Analyse all the repositories belonging to the identified GitHub users.
3. If the analysed projects meet the stipulated criteria for validity, append their names to the temporary database.
4. Retrieve all the requisite project data in JSON format and archive it in the database.

### **3.4 Identification of multi repository management tools.**

There are several notable differences between mono and multi repository projects and one of them is the management process of projects. Due to their complex structure, multi repository projects demand additional tools to control the workflow. Collectively these tools are called multi repository management tools (MRMT) and there have been several studies on them. There are several popular MRMTs like [11-13]. Some of them have a huge share like the GitHub platform itself and hence these tools have been thoroughly analysed by researchers. We developed a heuristic approach to identify other MRMTs. For this purpose, we collected a list of configuration files where each represents a certain tool and this way other researchers can easily identify which tool has been used for the management of any given multi repository project.

## **4 Branching strategies in Mono and Multi repository projects**

Three main branching strategies called *GitFlow*, *GitHub Flow* and *Trunk-based* were analysed in this study. The main reason for choosing these three is their popularity among the developers, which has also been proven via our own analyses of thousands of projects. The method that we developed for identification work is based on the structure of these branching strategies themselves [3]. In other words, we use the count of branches and their characteristics. Once again for the sake of clarity, it is presented in the following steps:

1. Remove all the branches created automatically by bots or MRMTs.
2. Record the total count of the branches and their names for identification.
3. If a project has only one branch which shares names like *main*, *master* and *product*. then this project uses a Trunk-based approach.
4. If a project has more than one branch and some of them are called *dev*, *development*, etc, then this project most probably uses the GitFlow approach.
5. If a project has more than one branch but it doesn't have, any development branches and it has several bug fix or error fix branches, then this project most probably uses the GitHub Flow branching strategy.

It should be added that this approach assumes that all the branches have been named correctly and developers apply their branches according to the main rules of the branching strategy.

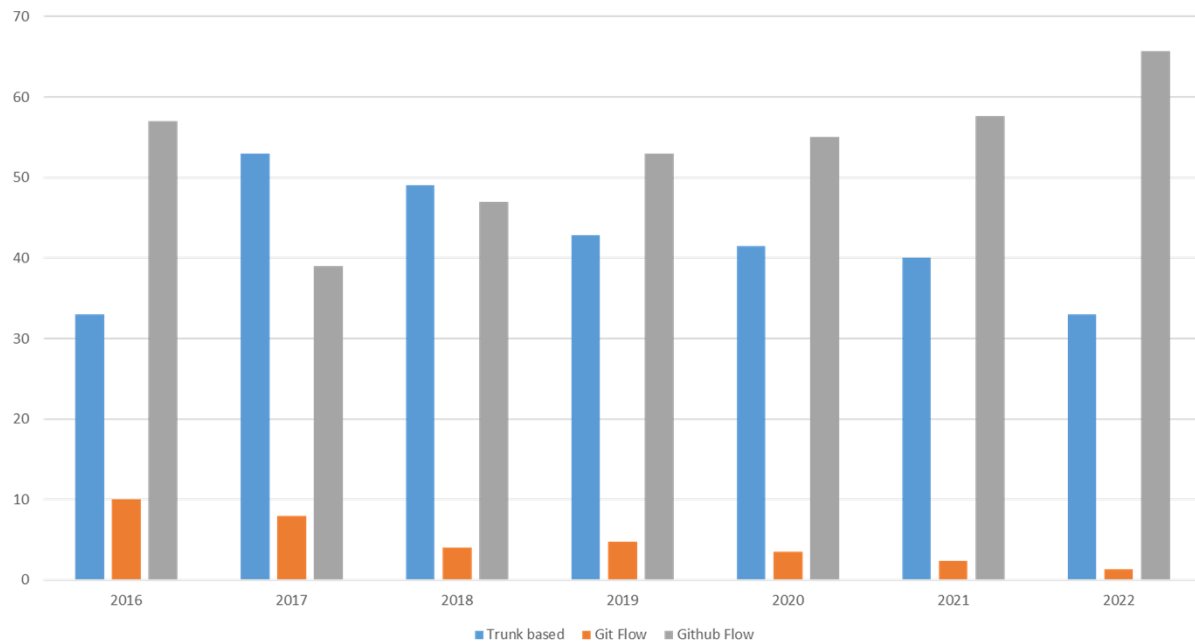


Figure 4.1 Popularity of the three main branching strategies over the years.

## 5 Productivity of software development

This chapter presents our findings on the productivity of software development. The main issue was the calculation of productivity itself and we analysed several different approaches for this purpose [14 - 17]. In the end the method which described in [18] was chosen as it seemed the most suitable. The main reason for choosing this method is:

- It calculates the productivity based on the activity of developers, which is more complex and general approach than that employed by other researchers.
- The database used by this method is like our own database.

We calculated the productivity for all the projects in our database and divided it into three groups; namely High, Low and None. We noticed that there was a correlation between this productivity value and other parameters of projects like repository structure, branching strategy, and development period [4]. Because of this, we

devised a new Machine Learning approach to calculate productivity without doing any mathematical calculations used by above approach.

The model was trained based on the following features of repositories: *Commit count, developer team size, project size, issue count, event count* and *pull request count*. etc. More features from our database could be used, but it is known that in certain cases using too many features can create noisy data [19]. The results of our tests can be seen in the table below.

Model	Accuracy	Precision	Recall	F1 Score
Logistic Regression	0.9003	0.8125	0.9019	0.8533
Decision Tree	0.6802	0.4952	0.6705	0.5765
Random Forest	0.9344	0.8410	0.9274	0.8948
Support Vector Machine	0.6397	0.4732	0.6221	0.5255

Table 5. The results of ML testing.

### 5.1 Prediction of Software Development Period.

The other Machine learning approach we proposed is for the calculation of the software development process. There are some approaches available that can be used to calculate the development period [20 - 24].

As mentioned earlier, statistical connection was found between the development period and productivity of software development process, so we used this and other several connections to create a model for estimating the development time for a project. This method calculates the development period in months and uses the following parameters for it: *Productivity, Branching Strategy (Trunk-Based, GitFlow, GitHub Flow), Number of Contributors, Branches, Pull Requests, and Issues*. The model's performance was evaluated using three key metrics; namely Mean Absolute Error (MAE), Mean Squared Error (MSE), and the coefficient of determination (R-squared). The results were as follows: Mean Absolute Error (MAE): 3.40 months, Mean Squared Error (MSE): 52.42 months<sup>2</sup>, R-squared (R<sup>2</sup>): 0.441.

These results have a low level of error, with the model predicting the development time, with an error of approximately 3.40 months. The R<sup>2</sup> value was



medium, indicating that the model significantly explains the variability in the development periods of the projects.

## 6 Collaboration of Software Developer Team

This last chapter focuses on the collaboration of the software developer team during the development process and the connection between this collaboration and above-mentioned parameters of the project like repository structure, branching strategy and productivity.

One of the novelties of this chapter is its new approach for calculating software team collaboration with a mathematical formulation. We analysed several approaches used for the calculation [25 - 28]. It became evident that most of the approaches focus on the calculation of the overall workload of each developer. Our procedure in contrast calculates the number of commits, pull requests, etc. Taking this into account, we devised a similar but more objective approach with a good mathematical basis.

Each contributor of the GitHub repository has a value which is called *contribution*, and this value is represented by an integer number. This number represents the overall contribution of a developer, and it may be needed for the calculation of his workload during the development process. We did a mathematical calculation to find the percentage share of each developer's work and this way we placed projects into the following categories: *Very high collaboration*, *High collaboration*, *Medium collaboration*, and *Low collaboration*.

### 6.1 Predictive Modeling for Developer Team Sizing.

One of the novelties here is the creation of an advice system to the most suitable number of developers for a given project. We created these systems based on the results which we got from a previous study. Our methodology employed a robust quantitative analysis, exploiting an extensive dataset of GitHub projects. In our analysis, we first pre-processed the data to find the projects that met our criteria for 'high performing'. This categorisation was multi-faceted; projects must not only have a high count of stars and forks (top 25th percentile) but must also be tagged with 'High' and 'Very High' productivity. This subset of projects was used in our advice

system. Later the projects were placed into groups defined using several parameters. For the testing of our approach, we used three parameters called *Programming language*, *Branching Strategy* and *Development period*.

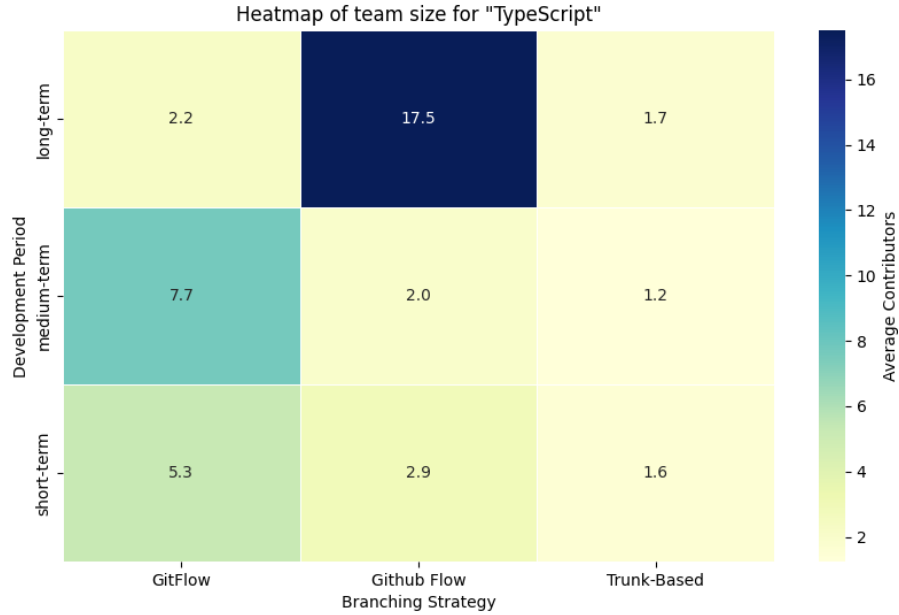


Figure 6.1 Heatmap of the average team size for TypeScript projects.

## 7 Contributions of the thesis

In the first thesis group, the contributions are related to the publications “Comparison between mono and multi repository structures”, “Machine learning model for identification of frontend and backend repositories in GitHub” and “Multi Repository Management tools”. A detailed discussion can be found in Chapter 3.

- I/1. A new definition for Mono and Multi repository projects was proposed that incorporates both of their characteristics and structures.
- I/2. Creating a new ML method for the identification of frontend and backend repositories on the GitHub platform. This method is especially useful for quick identification of both types of repositories.

- I/3. A new algorithm for the identification of Mono and multi repository projects was proposed. This unique approach for the identification and collection of projects belonging to both repository structures can be used for all types of projects, and it is possible to adopt it to other projects as well.
- I/4. A heuristic approach for the identification of different multi repository management tools was applied. Developers and researchers can use this approach to assist their work and research.

In the second thesis group, the contributions are related to the publication “Optimizing Branching Strategies in Mono and Multi repository Environments: A Comprehensive Analysis”. Detailed discussion can be found in Chapter 4.

- II/1. We proposed a heuristic approach for the identification of branching strategies used by the project during the development phase. Branching strategies are essential parts of the project management process and this way they can be identified much faster than any other approach.
- II/2. Conducting research and an analysis into the connection between branching strategies, repository structure and productivity.

In the third thesis group, the contributions are related to the publication “Analyzing Branching Strategies for Project Productivity: Identifying the Preferred Approach”. Detailed discussion can be found in Chapter 5.

- III/1. A new machine learning method was proposed for the assessment of productivity of the software development process. This approach is based on the correlation between productivity and several parameters of the project and development process.
- III/2. A machine learning method was proposed for the estimation of the development period.

In the fourth thesis group, the contributions are related to the publication “Repository Structures: Impact on Collaboration and Productivity”. Detailed discussion can be found in Chapter 6.

IV/1. Use of a mathematical method for the calculation of the software team collaboration rate.

IV/2. A special advice system was created for the estimating the number of developers required for a project based on the given parameters.

Table 2 summarizes the relation between the thesis points and the corresponding publications.

Publications	Thesis point					
	I/1, I/3	I/2	I/4	II	III	IV
[1]	*					
[2]			*			
[3]				*		
[4]					*	
[5]						*
[6]		*				

Table 2. Correspondence between the thesis points and my publications.

## The author's publications on the subject of the thesis

### Journal Publications

- [1] U. Shakikhanli, V. Bilicki, Comparison between mono and multi repository structures. *Pollack Periodica*, vol. 17, no. 3, pp. 7-12, 2022.
- [2] U. Shakikhanli, V. Bilicki, Multi Repository Management tools. *The journal of CIEES*, vol. 2, no. 2, pp. 13-18, 2022.
- [3] U. Shakikhanli, V. Bilicki, Optimizing Branching Strategies in Mono-and Multi-Repository Environments: A Comprehensive Analysis. *Computer Assisted Methods in Engineering and Science*, vol. 31, no. 1, pp. 81-111, 2024.
- [4] U. Shakikhanli, V. Bilicki, Analyzing Branching Strategies for Project Productivity: Identifying the Preferred Approach. *Journal of Electrical Systems*, 2024.
- [5] U. Shakikhanli, V. Bilicki, Repository Structures: Impact on Collaboration and Productivity. *Pollack Periodica*, 2024.

### Full papers in conference proceedings

- [6] U. Shakikhanli, V. Bilicki, Machine learning model for identification of frontend and backend repositories in GitHub. *Multidisciplinary Science Journal*, vol. 5, 2023.

### Other references

- [7] Monorepo, Manyrepo, Metarepo. Burke Libbey 2019. [Online] Available: <https://notes.burke.libbey.me/metarepo/> (Last accessed 6 April 2024).
- [8] Ciera Jaspán, Matthew Jorde, Andrea Knight, Caitlin Sadowski, Edward K. Smith, Collin Winter, "Advantages and Disadvantages of a Monolithic Repository: A Case Study at Google", 2018 ACM/IEEE 40th International

- Conference on Software Engineering: Software Engineering in Practice, May 27-June 3, 2018, Gothenburg, Sweden.
- [9] Liu, Yi, Taghi M. Khoshgoftaar, and Naeem Seliya. "Evolutionary optimization of software quality modeling with multiple repositories." *IEEE Transactions on Software Engineering* 36.6 (2010): 852-864.
  - [10] Weber, Jens H., Anita Katahoire, and Morgan Price. "Uncovering variability models for software ecosystems from multi-repository structures." *Proceedings of the 9th International Workshop on Variability Modeling of Software-Intensive Systems*. 2015.
  - [11] Eirini Kalliamvakou, Georgios Gousios, Kelly Blincoe, Leif Singer, Daniel M. German, Daniela Damian, "The promises and perils of mining GitHub", *MSR 2014: Proceedings of the 11th Working Conference on Mining Software Repositories*, May 2014 Pages 92–101.
  - [12] GitHub. <http://www.GitHub.com/> . Accessed March 16, 2024.
  - [13] Mercurial. <https://www.mercurial-scm.org> . Accessed March 16, 2024.
  - [14] Slintel.  
<https://www.slintel.com/tech/source-code-management/GitHub-market-share>. Accessed March 16, 2024.
  - [15] A. MacCormack, C. Kemerer, M. Cusumano, and B. Crandall, "Trade-Offs between Productivity and Quality in Selecting Software Development Practices," *IEEE Software*, pp. 78-79, Sept./Oct. 2003.
  - [16] Kitchenham, Barbara, and Emilia Mendes. "Software productivity measurement using multiple size measures." *IEEE Transactions on Software Engineering* 30.12 (2004): 1023-1035.
  - [17] Helie, Jean, Ian Wright, and Albert Ziegler. "Measuring software development productivity: A machine learning approach." *Proceedings of the Conference on Machine Learning for Programming Workshop, Affiliated with FLoC, Oxford, UK*. 2018.
  - [18] Zou, Weiqin, et al. "Branch use in practice: A large-scale empirical study of 2,923 projects on GitHub." *2019 IEEE 19th International Conference on Software Quality, Reliability and Security (QRS)*. IEEE, 2019.
  - [19] Choudhary, Samridhi Shree, et al. "Modeling coordination and productivity in open-source GitHub projects." *School of Computer Science, Carnegie Mellon University* (2018).

- [20] García, Salvador, et al. "Dealing with noisy data." *Data preprocessing in data mining* (2015): 107-145.
- [21] Idri, Ali, Fatima azzahra Amazal, and Alain Abran. "Analogy-based software development effort estimation: A systematic mapping and review." *Information and Software Technology* 58 (2015): 206-230.
- [22] Shepperd, Martin, and Chris Schofield. "Estimating software project effort using analogies." *IEEE Transactions on software engineering* 23.11 (1997): 736-743.
- [23] Khoshgoftaar, Taghi M., and Naeem Seliya. "Analogy-based practical classification rules for software quality estimation." *Empirical Software Engineering* 8 (2003): 325-350.
- [24] Amazal, Fatima Azzahra, Ali Idri, and Alain Abran. "Software development effort estimation using classical and fuzzy analogy: a cross-validation comparative study." *International Journal of Computational Intelligence and Applications* 13.03 (2014): 1450013.
- [25] Usman, Muhammad, et al. "Effort estimation in agile software development: a systematic literature review." *Proceedings of the 10th international conference on predictive models in software engineering*. 2014.
- [26] Saadat, Samaneh, et al. "Analyzing the productivity of GitHub teams based on formation phase activity." *2020 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)*. IEEE, 2020.
- [27] Constantino, Kattiana, et al. "Perceptions of open-source software developers on collaborations: An interview and survey study." *Journal of Software: Evolution and Process* 35.5 (2023): e2393.
- [28] Constantino, Kattiana, et al. "Understanding collaborative software development: An interview study." *Proceedings of the 15th international conference on global software engineering*. 2020.
- [29] Scholtes, Ingo, Pavlin Mavrodiev, and Frank Schweitzer. "From Aristotle to Ringelmann: a large-scale analysis of team productivity and coordination in Open-Source Software projects." *Empirical Software Engineering* 21.2 (2016): 642-683.

## 8 Összefoglaló

A doktori értekezés témája a két fő repository-struktúra típus, illetve ezek kapcsolatának vizsgálata a szoftverfejlesztési projektek olyan paramétereivel, mint a fejlesztői produktivitás, a szoftverfejlesztői csapaton belüli kollaboráció mértéke, a fejlesztési időszak időtartama, a fejlesztőcsapat mérete stb. Az eredmények kellő generalizációs mértékének elérése érdekében egy speciális projekt adatbázis jött létre, amely több mint 50 000 mono és multi repository projektet tartalmaz. A disszertáció fő célja annak bemutatása, hogy milyen kapcsolat van a repository struktúra és a fejlesztés folyamata között, és ha léteznek ilyen kapcsolatok, hogyan befolyásolhatják a fejlesztési folyamat egészét. Mivel a szoftverfejlesztés szűken kevésbé területéről van szó, saját egyedi algoritmusokat hoztunk létre a Mono és Multi repository projektek azonosítására és begyűjtésére a GitHub platformról. Ezen kívül egy új gépi tanulási megközelítést dolgoztam ki a fejlesztési folyamat produktivitásának becslésére, valamint egy matematikai módszert a fejlesztői csapatok együttműködési mértékének kiszámítására.



## Declaration

In the PhD dissertation of Ulvi Shakikhanli entitled “An exploration of modern domain specific software architectures”, with list of publications:

- [1] U. Shakikhanli, V. Bilicki, Comparison between mono and multi repository structures. *Pollack Periodica*, vol. 17, no. 3, pp. 7-12, 2022.
- [2] U. Shakikhanli, V. Bilicki, Multi Repository Management tools. *The journal of CIEES*, vol. 2, no. 2, pp. 13-18, 2022.
- [3] U. Shakikhanli, V. Bilicki, Optimizing Branching Strategies in Mono-and Multi-Repository Environments: A Comprehensive Analysis. *Computer Assisted Methods in Engineering and Science*, vol. 31, no. 1, pp. 81-111, 2024.
- [4] U. Shakikhanli, V. Bilicki, Analyzing Branching Strategies for Project Productivity: Identifying the Preferred Approach. *Journal of Electrical Systems*, 2024.
- [5] U. Shakikhanli, V. Bilicki, Repository Structures: Impact on Collaboration and Productivity. *Pollack Periodica*, 2024.
- [6] U. Shakikhanli, V. Bilicki, Machine learning model for identification of frontend and backend repositories in Github. *Multidisciplinary Science Journal*, vol. 5, 2023.

Ulvi Shakikhanli contribution was decisive in the following results:

- In **Thesis I**, the focus is analyzing Mono and Multi repository structures. Several new algorithms have been proposed for the identification and analysis of these structures [1], [2], [6].
- In **Thesis II**, involves analyzing implementing new heuristic approach for identification and collection of branching strategies and their analysis [3].

- In **Thesis III**, explores the connection between productivity of software development and repository structures. This part also involves measurement of productivity process and calculation of development period [4].
- In **Thesis IV**, focus on the collaboration of software developer team during the development process. This part involves the introduction of a new mathematical approach for the calculation of collaboration rate and advice system for ideal team size.

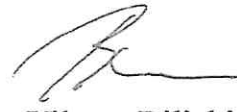
these results cannot be used to obtain an academic research degree, other than the submitted PhD thesis of Ulvi Shakikhanli.

Szeged, 2024.07.02



Ulvi Shakikhanli

PhD candidate



Vilmos Bilicki, PhD.

Supervisor

The head of the Doctoral School of Computer Science declares that the declaration above was sent to all coauthors and none of them raised any objections against it.

Szeged, 2024.07.



Mark Jelasiy, DSc

Head of Doctoral School