**Associate Lecturer Mihai GHEORGHE, PhD**
**E-mail: mihai.gheorghe@gdm.ro**
**Department of Economic Informatics and Cybernetics**
**The Bucharest University of Economic Studies**
**Professor Marian Dârdală, PhD**
**E-mail: dardala@ase.ro**
**Department of Economic Informatics and Cybernetics**
**The Bucharest University of Economic Studies**

# AN AUTOMATED RECRUITING MODEL FOR AN OPTIMAL TEAM OF SOFTWARE ENGINEERS FROM GLOBAL FREELANCING PLATFORMS

***Abstract.*** *The COVID-19 pandemic has dramatically changed the shape of conventional recruitment processes over the past few months. Remote working where it is technically possible has become a de facto standard all over the world. For software development, Freelancing platforms, as virtually unlimited resources of globally distributed talents, pose a great opportunity but also plenty of risks. This study proposes an automated model for recruiting remote specialists in a framework meant to reduce the time-to-market for software products. The results are optimal in relation to the project's cost, team's productivity and overall product success. The research methodology along with the system used to collect data and the relevance of the data set are described.*

***Keywords****: Freelance Marketplaces, Follow the Sun, Global Software Development, Project Management, Automated recruiting.*

**JEL Classification: F66, J20, J46, M15, O30.**

## 1. Introduction

Freelance Marketplaces such as Upwork, Freelancer.com or Guru facilitate access to remote services from an unprecedented pool of specialists in various fields. For instance, Upwork, arguably the largest platform of its kind, was reported in 2018 as having 12 million active freelancers, out of which 500.000 were registered as "Web, mobile and software development" specialists. Although advanced filters for criteria including area of expertise, hourly rates, work history and feedback are available, a client might still end up with a selection of thousands of candidates which by all means is impossible to manually evaluate in a



**DOI: 10.24818/18423264/54.4.20.03**

reasonable time interval. This leads to a significant risk of not being able to recruit the most suitable providers, which the current study aims to mitigate.

In a world where tech start-ups frequently emerge with the help of web and mobile Minimum Viable Products, reducing the time-to-market for a software product is crucial for the success of a business. Similar to many other fields, a Software Development Process can be accelerated by vertically or horizontally scaling the resources. Given the fact that software development is performed by human operators, vertical scaling can be translated to hiring better prepared and more productive team members, while horizontal scaling implies hiring more resources to contribute in parallel. The first strategy has an impact on the development cost, while the second on the complexity of the project management process. Also, an important constraint is that most software architectures break the product into logical components and technology layers where contributions cannot be committed in parallel. Although modern web and mobile development technologies take advantage of loosely-coupled architectures, with a precise separation between data, user interface and logic layers, and working on multiple functional features at the same time is possible, a limitation for parallel work is reached within the same feature. To overcome this, programming in shifts has been proposed and experimented starting with the late 90's at IBM (Carmel, 1999), and later on in other companies (Carmel et al., 2010), (Treinen & Miller Frost, 2006). The concept, known as "Follow the Sun" or "Round the Clock Development", aims to reduce the time-to-market by contracting parallel teams of software engineers, with complementary time-zones and with daily handoffs between shifts.

Offshore outsourcing for software development has been a common practice for many years and with the emergence of global Freelancing platforms along with recent large-scale adoption of remote work and coordination techniques improvements is arguably the first option given the COVID-19 Pandemic context when assessing the development framework for a web or mobile product.

This study identifies the cumulated benefits of the "Follow the Sun" development, referred to as *FTS*, using individual freelance software engineers recruited from acknowledged global marketplaces and proposes an automated theoretical model for the optimal team.

## 2. Proposed Follow the Sun development framework

Due to the nature of its outcome, a large pool of globally-distributed specialists and an overwhelming and easy to access documentation, web and mobile software development is one of the most suitable sub-fields to be dealt with remote resources. A continuous development process is theoretically possible with development cells consisting of 3 full-time software engineers that are 8 hours apart, based on their time-zones. It is a common practice that a full-time employee to allocate 8.5 or 9 hours / day for the job with breaks included, allowing the model


DOI: 10.24818/18423264/54.4.20.03

to accommodate handoffs. Figure 1 depicts such a scenario, the configuration for which the current study builds the optimal selection of candidates.

Other evaluated configurations of development in shifts models, with 4 shifts, having full or part-time allocation, are listed in Table 1 along with impact on various metrics or risks compared to a conventional development process. A significant positive impact is emphasized with a light green background, while a negative impact has a red background.

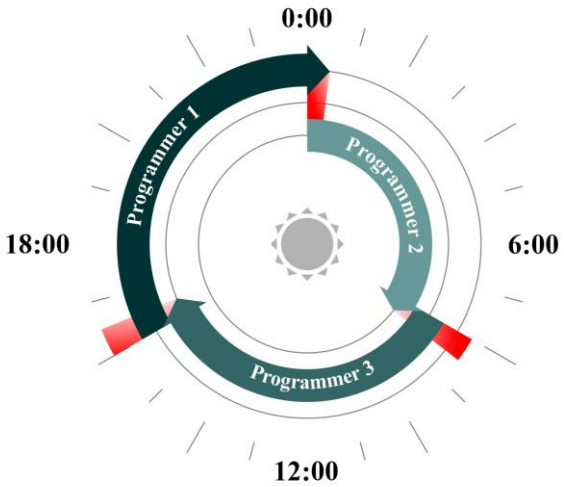


**Figure 1. Follow the Sun development with 3 full-time members ($FTS_1$) overlapping schedule**

Assuming there is an equal hourly cost for all developers in a configuration, all the *FTS* scenarios generate additional costs due to the schedule overlap. In addition, the management of a software development process with globally distributed human resources is more difficult.

However, the time-to-market is theoretically reduced to one third of the one consumed in a conventional process. Furthermore, a positive impact on software quality and innovation is possible due to taking advantage of continuous knowledge sharing between team members. Higher overlapping contexts such as the proposed $FTS_2$, has redundant developers for 45% of the daily time schedule. This facilitates pair-programming and consistent code-reviewing which as shown in (Beck & Fowler, 2001), (Layman et al., 2006) can even simplify the Quality Assurance processes.



DOI: 10.24818/18423264/54.4.20.03

**Table 1. Various Follow the Sun configurations compared to conventional development**

| | Conv. 1x8.5h / day | $FTS_1$ 3x8.5h/ day | $FTS_2$ 4x8.5h / day | $FTS_3$ 4x6.5h / day |
|---|---|---|---|---|
| Time-to-market reduced with | 0 | 64.44% | 63.64% | 63.64% |
| Hourly rate cost increase | 0 | 6.66% | 45.45% | 9.09% |
| Impact on software quality and innovation | - | + | +++ | + |
| Impact on efficiency | 0 | 0 | 0 | + |
| Reducing QA effort | - | - | + | - |
| Impact on project when one member becomes unavailable | High | Low | Low | Low |
| Synchronization and handoff risk | *n/a* | High | Low | High |
| Synchronization between all members | *n/a* | + | - | - |
| Impact on Project Management effort | + | - | - | - |

### 3. The overall architecture of the automated selection flow

Considering the *$FTS_1$* framework, and the aforementioned shortcoming that a list of compatible candidates may be too large for a human operator to evaluate, the problem to which the current study proposes a solution is to create an automated flow that would return the optimal set(s) of 3 developers, by analyzing the same information to which a human decision maker might have access. The optimal selection has to be in relation with the following criteria:

- **The duration** of the development processes based on the estimated project effort divided by the predicted productivity of each team member.
- **The cost** of development, as the total amount paid to the developers, which computes to hourly rates multiplied by the duration.
- **The Success Rate** of the development process, as a probability predicted from the technical expertise and work history of each team member.

From an architectural perspective, the proposed solution has 4 stages as also depicted in Figure 2:


DOI: 10.24818/18423264/54.4.20.03

1. **Defining the requirements of the software project,** representing the flow's input with the objective of narrowing the stack of specialists to a sub-set of required technologies. Additionally, it has to provide a quantitative estimation of the development effort. Acknowledged Agile software methodologies determine the duration based on a total of estimated story-points and the amount of story points a developer from a certain expertise / seniority class typically delivers / day or week.

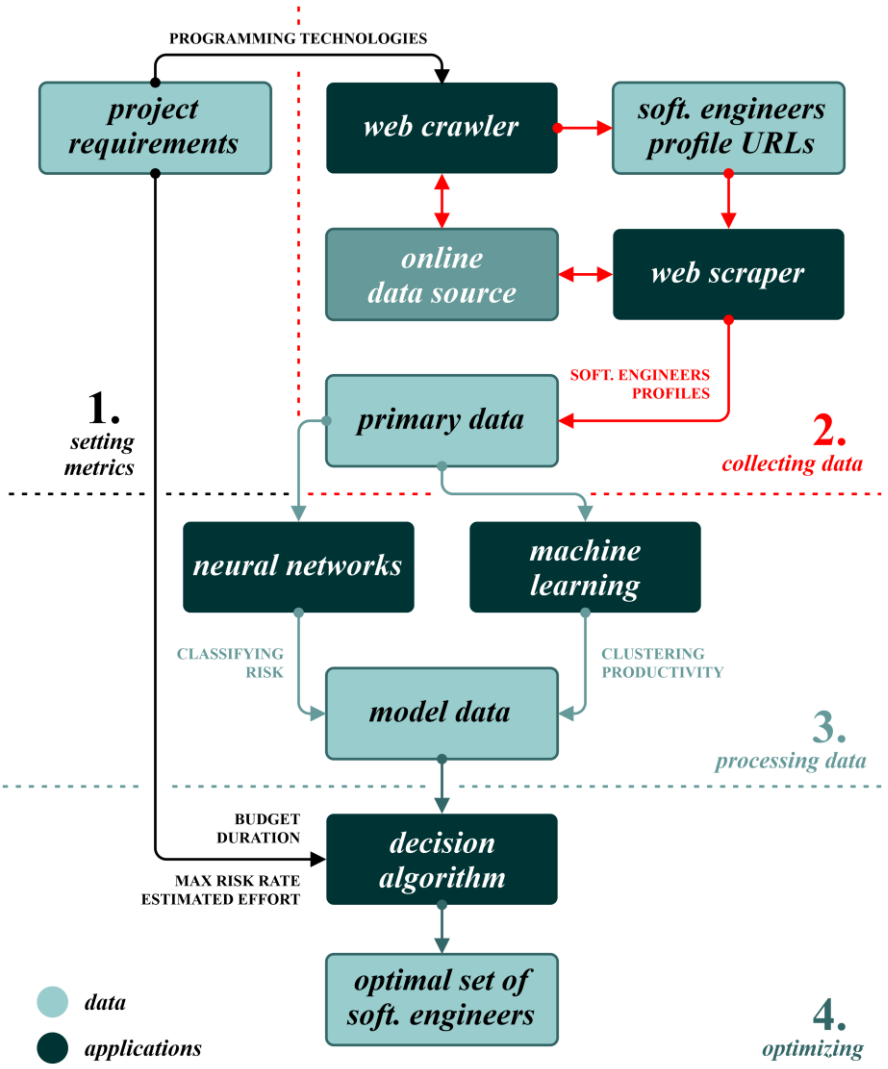


**Figure 2. The Architecture of the automated selection solution**



DOI: 10.24818/18423264/54.4.20.03

2. **Collecting Data** is the phase in which the profiles of compatible candidates are retrieved using a custom-built crawling and scraping application from the Freelancing platform and converted to a structured data set.
3. **Predicting individual Job Success Scores and Seniority Classification** involves applying and validating Machine Learning algorithms over the data set in order to be able to compute a risk factor and the productivity class for each candidate.
4. **Finding the optimal team** consists of theoretically solving and implementing an optimization problem to select the most suitable set of 3 developers that have complementary time-zones in relation to the project requirements.

## 4. The data set

In order to validate the selection model, real data was considered. Upwork.com was chosen as the data source given its market share but also the authors familiarity with the platform.

Freelancer profiles on Upwork resemble a personal Curriculum Vitae, showcasing expertise areas, qualifications for various technical assessments, hourly rates and the history of jobs delivered through the platform.

An individual metric of high importance for this study, available on each freelancer's profile, is the **Job Success Score** which is a subjective boolean input from customers in regard to the outcome of each project delivered through the platform.

At the moment of the study, the website indicated a modern architecture built around a Single-Page Application and multiple back-end APIs. As shown in Gheorghe et al. (2018), the web page for this type of applications is dynamically rendered when data is retrieved from the server, making automated data collecting difficult through conventional methods such as interpreting HTTP responses. To overcome this, a mechanism to automate a headless browser and extract the necessary data was created. The solution, built with Python and *Selenium WebDriver*, consists of a web-crawler to retrieve the software engineers profile URLs based on various search criteria, and a scraper to extract information from an individual profile to a structured data set. In order to minimize the server stress, a crawl politeness strategy was set in place, mainly through issuing page requests at intervals of 150 seconds or higher.

To emphasize the relevance of the data set, a summary of the collected data is presented in Table 2.

Furthermore, analyzing the histograms for each variable showed anticipated distributions with no abnormal frequency drops, so we are entitled to state that the dataset is statistically significant.


DOI: 10.24818/18423264/54.4.20.03

**Table 2. Summary of the collected profiles**

| Profiles | Projects | Projects value | Hours worked | Technical tests | Countries | Hourly rates |
|---|---|---|---|---|---|---|
| **2,670** | **55,594** | **$71,065,349** | **3,107,467** | **8,168** | **103** | **$3 - 250** |

After being collected, the data set was pre-processed, reducing the number of metrics from 120 to 21 by grouping similar technical assessment scores into categories relevant for web and mobile development, and taking the maximum score, if any, for each of these. Also for each candidate, the country, a categorical variable, was assigned to one of 7 distinct geographical regions. A high number of metrics, later on serving as independent variables, would have raised *multicollinearity* issues with the Machine Learning algorithms used in the following stage of the model.

## 5. Predicting the Job Success Score

Although the Job Success Score, *JSS*, was computed for each freelancer, the platform doesn't display this metric if its value is less than 60%. Equally, we considered that the model should accommodate new entries as well, for which a *JSS* cannot be determined until the first project is delivered through the platform. Therefore, it was of great interest to predict the chances a software engineer successfully delivers a new project based on his country, hourly cost, expertise and work history.

Using Python and the *scikit-learn* library (Pedregosa et al., 2011), the first attempt was to implement a *Multiple Linear Regression* with a *Backward Elimination* algorithm. Although the program removed 7 independent variables until the predictor reached a theoretical statistical significance by having an acceptable *Adjusted* $R^2$ and *p-value*, validating the model through multiple training and test sets, *k-fold cross-validation* (Kohavi, 1995), returned a poor accuracy of only 30.58%. This leads to the following preliminary conclusion: a linear dependency between the profile and the *JSS* cannot be determined.

The following step was to try to predict the *JSS* with *Random Forrest Regression* which is an *Ensemble Learning* algorithm that computes the average prediction of multiple *Decision Trees*. Using a *Grid-Search* strategy to find the most performing set of parameters and validate the regressor through *k-fold cross-validation*, the returned accuracy over our data set was 63.03%, a noticeable improvement from the previous predictor. However, running a *feature-importances* analysis revealed a strongly dominant feature identified as the *number of delivered projects* variable from the data set, with a score of 62.44%. Therefore, a second preliminary conclusion is that in estimating the *JSS*, the work experience has the



**DOI: 10.24818/18423264/54.4.20.03**

most significant weight. Other factors, such as technical assessments scored less than 1%. Furthermore, the less significant variable was the country of origin. We interpret this as highly relevant for a software development methodology that pursues employment of globally-distributed specialists.

Given the fact the goal of this study is to select the optimal set of 3 software engineers, the focus was changed towards estimating the project's overall success chances, as per the following formula:

$$S = \prod_{i=1}^{3} JSS_i \quad (1)$$

In this new context, imagining all 3 team members with individual *JSS* of 80% or lower would render the project's success to less than 50% as showcased in Figure 3, which many might find unacceptable.

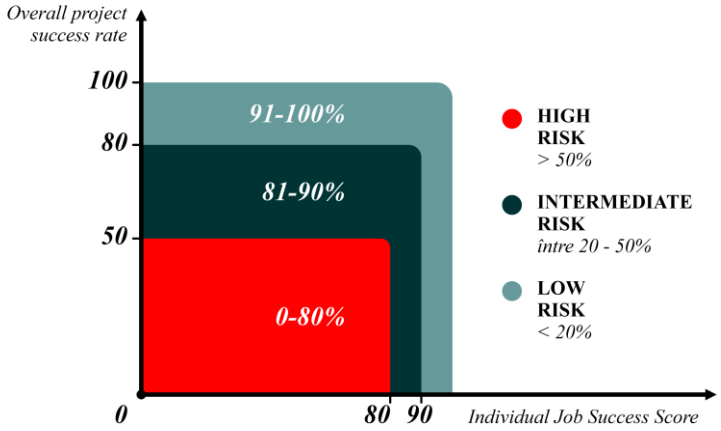


**Figure 3. Classes of risk**

Therefore, instead of the aiming to precisely estimate individual *JSS*, the problem was transformed to a classification one, still relevant to the selection model. To enforce this reasoning, at the time of this research, it was notorious among users of popular Freelancing platforms that technical test scores were arguably easy to bias, due to their quick disclosure over the Internet.

For the classification problem, due to the complexity of factors that might determine a customer to set a project as successful or not, doubled by the poor statistical performance of other regression models, we opted for a *Feed Forward Neural Network* with *adam* as the *Stochastic Gradient Descent* improved technique to optimize the cost function (Kingma & Ba, 2014). The implementation of the classifier was done in Python with the *Tensorflow* ANN framework and *Keras*. *Grid-Search* was also used to find the best performing model parameters.

Validating the model through a confusion matrix returned a remarkable accuracy of 81.04% compared to the real observations.


DOI: 10.24818/18423264/54.4.20.03

An interesting finding was revealed when the classifier was run on the dataset after job history related metrics were removed. Although inferior to the previous one, the resulted 65.40% accuracy leads us to a third preliminary conclusion which is: new entries can be risk classified only with technical assessments, country of origin, hourly rate and English score. It is reasonable to consider a performance improvement of this classification model in relation with a more rigorous technical testing framework.

## 6. Segmenting candidates into productivity classes

Rigorously establishing a hierarchy or classification among software engineers based on their expertise level and productivity represents an endeavor across industry. Although seniority classes have similar titles, for instance variations for junior / mid / senior developer, technical leader and solution architect, different companies may have different standards in assigning these roles as quantifying experience, efficiency and productivity hasn't been yet standardized for all software technologies or project complexity levels (Duarte, 2014), (Raza & Faria, 2014).

Therefore, especially for evaluating a high number of candidates, it is of high importance to segment the population into sub-sets based on empirical measured metrics such as working history, productivity and technical abilities. Given that there is no observable dependent variable to predict or to classify, the solution is an unsupervised analysis.

As one of the most established methods in the field of unsupervised learning, the *K-means* algorithm was implemented to group the candidates into a specified number of clusters. The number of relevant clusters, 4, has been determined with the *Elbow Method*, which is a graphical analysis of the *Within Cluster Sum of Squares* parameter variation function to the number of clusters. Therefore, running the *K-means* implementation over the collected data returned 4 distinct populations whose characteristics were then evaluated. The frequencies were as follows: 1%, 2%, 14% and 83%. It was an interesting finding to realize that the cluster with the smallest population had the highest average and median values for hourly costs, earnings, number of delivered projects, number of worked hours and technical assessments scores. As opposite to this, the cluster with 83% share had the poorest scores. The middle clusters followed the same reasoning.

Although the analysis is far from a scientific formalization, it leads the path to estimate the productivity of a target candidate based on the cluster he qualifies for. In conventional agile development methodologies this is already a common empirical practice to the extent of stating that a developer from a certain class, can deliver a rather precise number of story points per time unit.



DOI: 10.24818/18423264/54.4.20.03

### 7. Selecting the optimal set of Software Engineers

Taking into account the previously processed set of data, the optimization solution we considered takes as inputs software engineers objects, referred as $w_i$ with the following relevant variables:

- $c_i$ representing the hourly cost
- $p_i$ representing the productivity associated to the corresponding cluster
- $c_i$ representing the success rate, $s_i = \overline{0,1}$
- $t_i$ representing the time-zone integer variable, $t_i = \overline{0,23}$

In a *FTS1* configuration, the optimal team consists in a set of 3 candidates, $w_i, w_j, w_k$ who need to have complementary time-zones, which translates to: $|t_i - t_j| = 8$,, $|t_j - t_k| = 8$ and $t_i \neq t_k$.

The criteria to calculate the optimal set on top of is as follows:

- $D = \frac{U}{p_i + p_j + p_k}$, as the duration of the development process where $U$ is the estimated project effort, using the same units as estimating productivity.
- $C = D * 8 * (c_i + c_j + c_k)$, as the cost of the project
- $S = s_i * s_j * s_k$, as the probability that the development process is successful.

In relation to this criteria, we formulated 3 optimization problems: for the minimum project cost, the minimum development duration and the maximum process success rate.

**The minimum cost problem** requires finding the pair(s) of 3 developers which generate the minimum project cost, restricted by a maximum duration and a minimum success rate which are set as inputs.

**The minimum duration problem** requires finding the pair(s) of 3 developers which deliver the project with the minimum duration, restricted by a maximum duration and a minimum success rate which are set as inputs.

**The maximum success rate** requires finding the pair(s) of 3 developers which have the maximum multiplied success probabilities, restricted by a maximum duration and a maximum project cost which are set as inputs.

Apparently simple to solve, these problems require an unfeasible amount of time in case of a *brute-force* approach, i.e. generating all the combinations and sorting them by the optimization factors. For example, filtering the Upwork.com candidates by JavaScript as their programming technology, returned in September 2019 more than 100.000 results. Assuming an equal distribution across time-zones, which is the least favorable scenario, this leads to approximately 4.000 candidates


DOI: 10.24818/18423264/54.4.20.03

per time-zone. From a computational perspective, the *brute-force* algorithm has a *polynomial execution time*, $T(n) = 8 * O(n^3)$, where $n$ is the number of candidates per time-zone. A simulation with 50 candidates per time-zone on a Linux virtual machine with 1 processing core and 2 GB or memory required 53 seconds to run. For 4000 candidates, based on the execution time class, we estimate the computation to consume more than 50000 days without taking into consideration that the database should accommodate at least 500 billion records.

A mathematical optimization was therefore considered. The first step was to group the candidates in 8 sub-sets, based on their complementary time-zones and therefore, solving 8 lower complexity problems. The sub-problems were reformulated into *Binary Integer Programming* with linear-fractional constrains which through simple transformations resulted in linear constrains. However, the Objective Function was also linear-fractional. To overcome this, we used Dinkelbach's (1967) *Generalized Fractional Programming* method. After this step, the problem's type was once more transformed to a *Mixed Integer Linear Programming* one which although has a nondeterministic polynomial difficult execution time - *NP-Hard* can be computationally solved in a favorable manner by using acknowledged algorithms such as *Branch and Bound* (Nica, 2011).

The implementation of the algorithm using *R* was computationally viable. The assumption was made after analyzing multiple execution times on generated data sets with more than 5000 candidates per time-zone, which varied between 2 and 15 seconds.

## 8. Conclusions

The current research proposes a model to automatically recruit the optimal development cell of 3 software engineers working in a *Follow the Sun* framework, from an existing global resource of talents, Upwork.com. It's worth mentioning that with minimal technical adjustments to the implementation stages, the model is compatible with other popular Freelancing platforms.

Although the model has been validated per stages, a validation as a whole hasn't been performed yet. This involves developing a relevant number of software products with the teams returned as optimal by the model and evaluating parameters such as budget, duration, software quality and overall success and is part of the authors future research goals.

## REFERENCES

[1] **Beck, K. and Fowler, M. (2001),** ***Planning Extreme Programming.*** *Addison-Wesley Professional*;

[2] **Carmel, E. (1999),** ***Global Software Teams: Collaborating across Borders and Time Zones.*** *Prentice Hall PTR*;



**DOI: 10.24818/18423264/54.4.20.03**

[3] **Carmel, E., Espinosa, J.A. and Dubinsky, Y. (2010),** ***" Follow the Sun" Workflow in Global Software Development.*** *Journal of Management Information Systems*, 27(1), pp.17-38;
[4] **Dinkelbach, W. (1967),** ***On Nonlinear Fractional Programming*****.** *Management science*, 13(7), pp.492-498;
[5] **Duarte, C.H.C. (2014), June,** ***On the Relationship between Quality Assurance and Productivity in Software Companies.*** In Proceedings of the 2nd International Workshop on Conducting Empirical Studies in Industry (pp. 31-38). ACM;
[6] **Gheorghe, M., Mihai, F.C. and Dârdală, M. (2018),** ***Modern Techniques of Web Scraping for Data Scientists*****.** *Romanian Journal of Human-Computer Interaction*, 11(1), pp.63-75;
[7] **Kohavi, R. (1995), August,** ***A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection*****.** In *Ijcai* (Vol. 14, No. 2, pp. 1137-1145);
[8] **Kingma, D.P. and Ba, J. (2014),** ***Adam: A Method for Stochastic Optimization.*** arXiv preprint arXiv:1412.6980;
*[9]* ***Layman, L., Williams, L., Damian, D. and Bures, H. (2006),*** **E*****ssential Communication Practices for Extreme Programming in a Global Software Development Team.*** *Information and Software Technology***,** 48(9), pp.781-794;
[10] **Nica, V.T. (2011),** ***Curs Cercetări Operaţionale I*****.** Academia de Studii Economice din Bucuresti;
[11] **Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V. and Vanderplas, J.( 2011),** ***Scikit-learn: Machine Learning in Python.*** *The Journal of machine Learning research*, 12, pp.2825-2830;
[12] **Raza, M. and Faria, J.P. (2014), May,** ***A Model for Analyzing Estimation, Productivity, and Quality Performance in the Personal Software Process.*** In Proceedings of the 2014 International Conference on Software and System Process (pp. 10-19). ACM.


**DOI: 10.24818/18423264/54.4.20.03**