Developers' importance from the leader perspective

Guilherme Costantin Tângari
Faculdade de Ciência da Computação
Universidade Federal de Uberlândia
Uberlândia, Brazil
guilhermecostantin@mestrado.ufu.br

Marcelo de Almeida Maia
Faculdade de Ciência da Computação
Universidade Federal de Uberlândia
Uberlândia, Brazil
marcelo.maia@ufu.br

***Abstract*—Several companies use the amount of deliveries as a metric of performance evaluation of the developer. However, the productivity of a developer and his importance for the company is not only related to the amount of lines of code produced. There are a variety of factors that can contribute to the relevance of a developer for a team. This paper aims at mapping some of these factors, measuring those that are more important for companies and propose an evaluation model of developer importance that considers more than just deliveries. We have found that some factors are more important than others and that there are minor differences for different companies. We have also developed a high accuracy classifier that can indicate the importance of the developer based on a set of attributes.**

Keywords—productivity, developers' importance, pattern recognition, human factors

I. INTRODUCTION

All kinds of companies have been investing in techniques to increase productivity in order to increase competitiveness, and this is no different in the software industry, which still continues investing in new methods, tools and best practices that could lead organizations to productivity improvement [1].

However, unlike hardware, which improves their price-performance ratios by orders of magnitude per decade, software productivity seems to have trouble to evolve in a similar pace [2]. The current productivity rates are similar to the rates of decades ago (one to two lines of code per man-hour) [2]. Brooks et al. [3] states that there is no technical or management technique that by itself promises one order-of-magnitude improvement in software productivity, simplicity or reliability.

Traditional productivity metrics for software development are based either on lines of code (LOC) or function points (FP) [4], for example, the amount of LOC or FP developers deliver per hour. A slightly more abstract definition for productivity is the ratio of delivered outputs to consumed inputs, where outputs may be LOC, FP, or other relevant delivery, and inputs are the resources used to produce that output, e.g., time, people [2], [5], [6].

Nonetheless, the use of only these traditional metrics can mislead the management of software teams. LOC does not take into account the effort and knowledge required to write them. Complex problems often require experienced developers to solve them, and often, they do not require lots of LOC. In that case, experienced developers would be penalized.

There are other notions of productivity that are not also taken into consideration when evaluating just lines of code, for example, developers with greater experience or with knowledge in specific tools may be frequently consulted by other developers to streamline and improve the development process of the team as a whole, so the formers have an indirect notion of productivity.

Talent retention and team motivation, for example, are two fundamental issues for any software company [2], [7]–[11]. Software is made by people, and people, when have their work recognized and well evaluated tend to produce more and better. A performance evaluation that considers only one aspect, such as the number of deliveries, and does not take into account tlhe different levels of difficulty and the purpose of the code, so the every day relationship with colleagues and the company would be compromised by unfair assessment, demotivating individuals and teams. Employee turnover its a common problem in software companies [12] , and a high turnover rate would lead to productivity losses, in addition to the increased cost of hiring and training, and most importantly, the loss of talents that search for recognition in other companies.

Several studies are devoted to discover the factors that have influence in productivity of software development and maintenance activities [1], [4], [7], [8]. Understanding those factors and having some mechanism to evaluate productivity in a fair way could provide to software team leaders a better tool to evaluate and compare their developers. Several companies are beginning to gain awareness of these issues and are committed to improve the way they evaluate developer performance. This work aims to investigate how team leaders understand the notion of importance, indicating which factors are most relevant in their developer overall assessment. We are interested on the investigation of these questions:

1. What are the most important criteria used by leaders while assessing developers?
2. It is possible to build a developer's classifier with high accuracy, using the proposed criteria? This question can be refined in other two:
    a. Is it possible to have a generic classifier, i.e., company-independent? or
    b. Is it more appropriate to build customized classifiers for each company?

(DOI Reference Number: 10.18293/SEKE2015-056)

As mentioned before, the performance of developers is highly related with their productivity, within a classical concept of the amount of deliveries. Nevertheless, managers and leaders on their daily work with them have a different perception of each one.

In this paper, we show an elicitation of factors, based on previous studies, which can have an influence in the leader's evaluation. We conducted a survey with team leaders representing software companies, where they evaluated their developers based on those factors. The result was analyzed in attempt to recognize a pattern in their evaluation. That way, we identified factors that mostly influence the leaders evaluation about their developers, and also built a high accuracy classifier for developers' importance.

In the next section, we will present the factors that we will use to achieve the developers importance classification, and the studies from where those factors were retrieved. Section III will present the methodology used to create and conduct a survey with human subjects. Section IV presents the results, and Section V discusses those results. Finally, Section VI provides the conclusion.

II. Importance Factors

In this section, factors used to represent concrete evaluation items for leaders about their developers are presented. Those factors will be used to define the metrics under the Goal-Question-Metric (GQM) approach used to design the survey elaboration.

Those factors were extracted from several studies available in the literature and were grouped into categories that present semantic affinity. Those categories will guide the formulation of the questions in the GQM model.

TABLE I. Elicitation of the importance factors

Groups	Importance factors	References
Technical characteristics	Past experiences	[5], [7], [8], [10], [13]–[20]
	Specialization (expert in some technology or tool)	
	Generalization (diversity of skills)	
	Solve complex problems	
	Productivity (quantity of deliveries per month)	
Behavioral characteristics	The main behavior of the developer when faces a problem	[5], [7], [8], [10], [17], [18], [19], [21],[22]
	Communication with the team members	
	Willingness to help a colleague	
Individual characteristics	Creativity	[23],[24]
	Entrepreneurship	
	Pro-activity	
	Leadership	
Commitment to the team / company	Planning and organization	[22], [25],[26]
	Focus on the costumers	
	Focus on the results	
	Time of work in the organization	

III. Methodology

To investigate the current practice of developers' evaluation, we decided to perform a survey with human subjects in real software companies to extract the desired information and analyze it. In this survey, we ask the respondent firstly to classify a subject developer and then fill the rest of the survey with the respective developer's characteristics. To analyze the obtained data, we decided to use automatic classification methods to get a clear view of how those characteristics affect leaders' classifications, and as a product we still may have a classifier that can be used to help leaders gain more insight about their teams' productivity.

This section is aimed at explaining how the survey was designed, and show how we conducted our data analysis obtained from that survey, including the criteria analysis and the classifier construction.

A. Survey

1) Goal-Question-Metric

We used an approach called Goal Question Metric (GQM) [27] that helped us define our survey. GQM is a top-down approach, that is based on the assumption that first, to measure something, you need to specify goals, from what is possible to derive questions that define those goals, and then specify the metrics that need to be collected to answer those questions.

To fulfill the purpose of a goal, we have to determine three coordinates:

a) Issue: The subject/matter you are dealing with.

b) Object (process): What is the central object of the analysis.

c) ViewPoint: Under whom perspective the analysis is being made.

TABLE II. shows our GQM model, with our purpose, the questions derived from it and the metrics defined to answer those questions.

For all those factors, the leader used a Likert scale with 5 options, ranging from "Very low" to "Very high", except from two factors: "The time of work in the organization", that receive a numeric value representing the months that the developers work in the organization, and "The main behavior of the developer when faces a problem", where the leader have to choose between one of the following options:

- Try to solve on your own (Introspective)
- Search in documentation or books (Introspective)
- Search or ask in Question and Answer sites and forums (Comunicative)
- Ask helps for the team or leaders (Comunicative)

TABLE II. Goal Question Metric

Goal	Purpose	Measure
	Issue	the importance
	Object	of a developer
	Viewpoint	under the leader perspective

Question	What is the technical-skills level of that developer?
Metrics	Productivity
	Past experiences
	Specialization (expert in some technology or tool)
	Generalization (diversity of skills)
	Solve complex problems
Question	What is the social-skills level of that developer?
Metrics	The main behavior of the developer when facing a problem[a]
	Communication with the team members
	Willingness to help a colleague
Question	What is the level of these behavior characteristics in the developer's profile?
Metrics	Leadership
	Creativity
	Entrepreneurship
	Pro-activity
Question	How is the commitment of the developer with the company?
Metrics	Planning and organization
	Focus on the costumers
	Focus on the results
	Time of work in the organization[a]

[a.] Factors that had different types of evaluation

2) Survey application

We applied the survey remotely, to give the freedom that our respondent needs to answer the question. For that, we used the Google Form tool.

We also, to preserve the companies' privacy, we did not get any kind of identification, both for the respondent and the developer being analyzed. The only asked identification was the company name from where those evaluations are. This was necessary for a deeper investigation specific for cases where companies reach the minimum of 10 developers evaluated.

3) Participant characterization

The survey was applied to software companies that has a software development environment with a minimum hierarchical structure where exists the role of leaders, or managers, or chief engineers, etc. (for future references, we call that person, the leader). All participant companies work in their own products (they are not only software factories), but they vary in size, considering amount of employees (developers), sector of operation (ERP, Telecom, etc.) and may vary in used technologies.

The respondents of the survey are team leaders. We understand that they are the right people to do it because, unlike the owner or higher level managers, they are close enough to the daily work, and can judge who are the most important developers and why, even if they do not use a formal method to assess it. They should answer one assessment per developer, i.e., if they evaluated 10 developers to reach the minimum to have their company individually analyzed, they answered 10 questionnaires.

B. Feature Selection

In order to conduct the analysis to determine which factors are the most relevant and have major influence in the leader evaluation, we use the WEKA[28] tool, an open-source software for data mining and machine learning.

Many real world problems, like ours, have a lot of features involved and only some of them are relevant to the target concept [29], in our case, the importance of a developer. To solve this issue, we will use a strategy called feature selection, where we select a subset of features to focus our attention, and ignore the rest to speed up learning, improve our classifier quality and achieve the best accuracy of the learning algorithm [29], [30].

The algorithm that we will use is called *GainRatioAttributeEval*, which is a single-attribute evaluator, that evaluates the attributes one by one independently and then rank them. Our feature selection will make a choice based on that ranking. That method does not eliminate the redundant attributes (only the irrelevant ones), but that is not a problem because we know all the attributes, and this kind of evaluator does not need a search method, what makes it very fast.

C. Classification

As a result of our survey, one dataset with several leaders' evaluations about the developers is generated. In this dataset machine-learning algorithms are applied to generate a classifier. The machine learning algorithms need two sets of data: one for training and one for testing, to verify the accuracy of the classifier. Fig. 1 shows the schema that best represent this scenario.

To evaluate the performance of a classifier, we used 10-fold cross-validation that divided the dataset in 10 equal parts (called folds), take 9 pieces to use for training and use the last piece for testing, and then do it 9 more times, always alternating the piece used for testing, that way, a single fold will be used 9 times for training and 1 for testing. The result will be the average of the 10 runs.

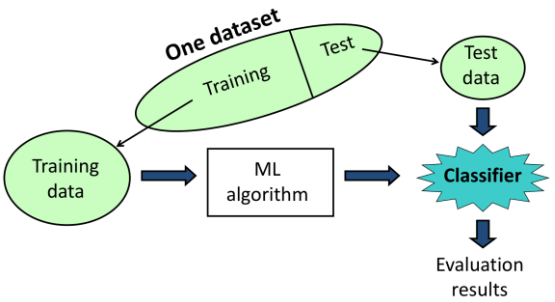

Fig. 1. Machine Learning algorithms schema

The used machine-learning algorithms are J48, a tree classifier, and Naïve Bayes, a bayesian classifier. There is no strong reason to choose them, but they tend to produce high quality classifiers in general, whenever possible.

J48 is a variation of a famous system called C4.5 which is described by Quinlan [28] that uses decision trees to build a classifier (WEKA actually let us have a look in the tree generated with all the weights).

Naïve Bayes is a probability method that has two assumptions: that the attributes are equally important and that

they are statistically independent (this independence assumption is never correct but the methods based on it often works well in practice).

We will also use a third algorithm called *AttributeSelectClassifier*, which actually use a method of feature selection (in our case, *Gain Ratio*) and an algorithm to perform the classification (in our case, J48 or Naïve Bayes). This way to apply feature selection only selects features in the training set, assuring we get more reliable results.

Finally, to conduct all those analysis, we will use a feature from WEKA called *EXPERIMENTER*, that allow us to run the same experiment more than one time and determine the mean and standard deviation, to avoid a misleadingly high or low accuracy based on the attribute selection to the training and testing sets. It also let us to compare the results of different algorithms.

IV. Results

Following the steps presented in the previous section, we show the results of the application of the survey, the feature selection performed in the dataset generated by the survey and the application of the classifiers and their accuracy in the developers' classification.

A. Survey

Firstly, we present the data achieved with the survey application. Eleven respondents (leaders) provided 61 answers (unique developers evaluated). In a few cases, some leaders work at the same company, but they run different teams. There were eight companies involved in the collected data.

We asked for the leaders to classify the developer in five degrees of importance. Those degrees and the distribution of the 61 developers among them are shown in Fig. 2.

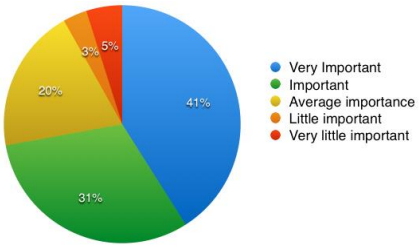

Fig. 2. Distribution of developers per degree of the class

TABLE III. New developer's set of classes

New class	Original class
High importance	Very important
	Important
Low importance	Average importance
	Little important
	Very little important

Analyzing the results of the survey, we came to the conclusion that the leaders were conservative in some degree to classify their developers in the lowest classification of importance.

From this analysis, we decided to group the developers also in only two classes based on the original five classes, as shown in TABLE III. in order to understand a more general picture of the intention of those leaders.

B. Feature Selection

As explained in the Section III.B, we used the algorithm *GainRatio* to rank the proposed attributes, in order to proceed with feature selection. TABLE IV. shows the rank ordered by the Average merit (the rate that the attribute influences in the classification) resulted of that algorithm application, using the original set of classes (five classes) and TABLE V. show the same view, now using the new set of classes (two classes).

TABLE IV. Attribute ranking (original set of 5 classes)

Features	Average merit
Capacity of solving complex problems	0.303
Subjective evaluation of the productivity	0.29
Proactivity	0.226
Past experiences	0.211
Generalization (diversity of skills)	0.202
Specialization (expert in some technology or tool)	0.2
Time of work in the organization	0.184
Creativity	0.18
Focus on the results	0.167
Focus on the customer	0.148
Main behavior of the developer	0.14
Communication with the team members	0.137
Planning and organization	0.122
Leadership	0.097
Entrepreneurship	0.087
Willingness to help a colleague	0.084

TABLE V. Attribute ranking (new set of 2 classes)

Features	Average merit
Proactivity	0.168
Subjective evaluation of the productivity	0.156
Capacity of solving complex problems	0.126
Focus on the results	0.112
Past experiences	0.107
Creativity	0.095
Planning and organization	0.09
Generalization (diversity of skills)	0.08
Specialization (expert in some technology or tool)	0.071
Focus on the customer	0.063
Time of work in the organization	0.063
Willingness to help a colleague	0.057
Leadership	0.052
Communication with the team members	0.039
Entrepreneurship	0.015
Main behavior of the developer	0.016

If we choose the three most relevant attributes, or expand our selection and choose the first ten, we will see that, even in a different order, they are the same, which supports our decision to group the class values.

C. Classification (all companies)

Considering that attributes have been ranked, we applied feature selection technique and use only the most relevant attributes in the classification. To determine how many features need to be selected to get a higher performance, we conducted an exhaustive test (we ran the classifiers with a crescent numbers of features selected, from 2 to 16) and chose the configuration with better performance (8 attributes). As we have two classes, we will show the results for the application of the J48 and Naïve Bayes for both of them, selecting the 8 first attributes more relevant and ignoring the rest.

TABLE VI. shows the results for the application of the algorithms using the first classification schema (5 classes). As we can see, J48 did not present good performance (close of 50% accuracy). Naïve Bayes had a better performance, but the accuracy could be considered still low for our purposes.

Now using the reduced set classes, we achieved better results, as expected, shown in TABLE VII. Again, Naïve Bayes had a better performance than J48, achieving now a relevant accuracy (85% of correctness).

TABLE VI. CLASSIFIERS APPLICATION (ORIGINAL SET OF CLASSES)

Algorithm	Percent correct
J48	51.88%
Naïve Bayes	61.12%

TABLE VII. CLASSIFIERS APPLICATION (REDUCED NEW SET OF CLASSES)

Algorithm	Percent correct
J48	75.88%
Naïve Bayes	85.21%

D. Classification (single company)

In our survey, out of the eight participant companies, 3 of them achieved the minimum numbers of responses that would allow an individual analysis of the company. We show the results of that individual analysis for one company (to preserve the company privacy, we call it *Company A*).

Company A evaluated 20 developers, and they presented reasonable distribution of the developers across the new classes (TABLE VIII.). TABLE IX. shows the attribute ranking for this particular company (we can see that there is a few differences from the relative generic attribute ranking in TABLE V. , which is discussed in Section V).

In this particular case, we had large difference between J48 and Naïve Bayes in what refers to feature selection. The selection did not produce a positive effect, and therefore classification performed better with all the attributes (results are shown in TABLE X.), and coincidently they both presented the same accuracy. For this analysis, we considered only the new classification that proved to improve the accuracy of the classifiers.

TABLE VIII. DISTRIBUTION OF THE DEVELOPERS ACROSS THE REDUCED NEW SET OF CLASSES

High importance	10
Low importance	9

TABLE IX. ATTRIBUTE RANKING (INDIVIDUAL COMPANY)

Features	Average merit
Proactivity	0.323
Capacity of solving complex problems	0.298
Communication with the team members	0.254
Focus on the results	0.23
Creativity	0.206
Subjective evaluation of the productivity	0.187
Planning and organization	0.191
Specialization (expert in some technology or tool)	0.189
Past experiences	0.173
Entrepreneurship	0.171
Main behavior of the developer	0.172
Willingness to help a colleague	0.156
Focus on the customer	0.158
Leadership	0.137
Generalization (diversity of skills)	0.136
Time of work in the organization	0.123

TABLE X. CLASSIFIERS APPLICATION (INDIVIDUAL COMPANY)

Algorithm	Percent correct
J48 (with 2 features)	79.50%
Naive Bayes (with all features)	79.50%

V. DISCUSSION

The first point considered in this discussion is the creation of the reduced new set classes. As shown in TABLE III. based on the original set of classes, that had five different classes, we grouped those 5 classes in only 2, creating a new set of classes that proved, as expected, to improve the performance of all the classification algorithms applied. As we could observe in TABLE IV. and TABLE V. , that this new reduced set of classes did not change the importance of attributes. So, this new classification scheme preserves the meaning of the original classification performed by the leaders because of the small variation in the attributes position. Moreover, we could observe that the classifier accuracy is around 80%, which gives a reasonable level of confidence on the coherence of the impact of the respective relevant factors on the importance level of developers.

The top 3 factors, which appear in both rankings, have a positive correlation with the class, which means that the better is the factor evaluation, the better is the position in the developer's importance classification. One of them is the productivity of the developer, under the leader perspective, where productivity represents the amount of work delivered. This was not a surprise because, as we mentioned in the beginning of the paper, because this is the classic metric to evaluate the developer's performance. On the other hand, the other two features bring new information to the discussion.

Capacity to solve complex problems lead to the opposite direction of the classic metric (amount of work delivered), because it often leads to the production of a lower rate of outputs (LOC or FP) over inputs (resources, time) consumed. This is an important qualitative point to consider whenever awarding high productive developers.

Proactivity is actually a required behavior characteristic of teams involved in the solution of complex problems instead of more canonical systems where the tasks are more predictable. The human resources area can conduct better hiring processes knowing that their software team leaders evaluated this as a fundamental requirement for developers.

The classification results evaluating all companies together with Naïve Bayes provided a classifier with 85.2% accuracy that can be considered a successful and useful result. The use of this classifier can help leaders conducting more coherent analysis of the team profile.

When analyzing an individual company, we noticed some major changes in some features' position in the feature ranking (TABLE IX.). Behavioral characteristics (*creativity*) and attributes related to the developer's commitment with the company (*focus on results*) in some cases were more important than the classic metric of productivity. We credit those differences to the culture and values of that particular company. So, different companies may assess the importance factors with some variation.

Finally, it is important to point out some few threats of the validity of this study. The limited number of developers and companies involved in this study may limit the generalization for other contexts. Nonetheless, we have observed several intersections in different companies that mitigate part of this threat. The classification provided by leaders tended to be more positive, maybe because they would not like to say that they maintain developers with low importance in their teams. The reduced classification mitigates part of this threat.

VI. CONCLUSION

In this study, we provided a set of criteria used by the leaders of IT companies to evaluate their developers, and also ranked those criteria, finding that capacity of solving complex problems, quantitative evaluation of productivity and proactivity were generally the most important factors.

Moreover, we created a high accuracy classifier, which can help, for example, the human resource managers to look for candidates that have the necessary needed characteristics and more potential to become an important part of the team.

A qualitative analysis, considering the culture of the company and their values, and the application of that classifier in the collaborators of open-source software repositories, to validate the results or spot the differences, could be suggestions of future work.

REFERENCES

[1] S. C. Sampaio, et al, "A Review of Productivity Factors and Strategies on Software Development," in *5th Proc. of ICSEA*, 2010, pp. 196–204.

[2] B. W. Boehm, "Improving Software Productivity," *Computer*, vol. 20, no. 9, pp. 43–57, 1987.

[3] F. P. Brooks Jr, "No silver bullet-essence and accidents of software engineering," *IEEE Comput.*, vol. 20, no. 4, pp. 10–19, 1987.

[4] S. Wagner and M. Ruhe, "A Structured Review of Productivity Factors in Software Development Technical ," 2008.

[5] C.Walston and C. Felix, "A method of programming measurement and estimation," *IBM Syst. J.*, vol. 16. pp. 54–73, 1977.

[6] W. D. Yu, D. P. Smith, and S. T. Huang, "Software productivity measurements," in *Proc. of COMPSAC'91*, 1991, pp. 558–564.

[7] B. W. Boehm, et al, *Software Cost Estimation with Cocomo II*, Upper Saddle River, NJ, USA: Prentice Hall PTR, 2000.

[8] P. D. Chatzoglou and L. A. Macaulay, "The importance of human factors in planning the requirements capture stage of a project," *Intl Journal of Project Management*, vol. 15. pp. 39–53, 1997.

[10] R. A. Scudder and A. R. Kucic, "Productivity Measures for Information Systems," *Inf. Manag.*, v. 20, n. 5, pp. 343–354, 1991.

[11] H. Sharp, et al, "Models of Motivation in Software Engineering," *Inf. Softw. Technol.*, vol. 51, no. 1, pp. 219–233, 2009.

[12] L. Wallace, M. Keil, and A. Rai, "Understanding Software Project Risk: A Cluster Analysis," *Inf. Manag.*, vol. 42, no. 1, pp. 115–125, 2004.

[13] K. D. Maxwell and P. Forselius, "Benchmarking software development productivity," *Software, IEEE*, v. 17, pp. 80–88, 2000.

[14] R. D. Banker, S. M. Datar, and C. F. Kemerer, "A Model to Evaluate Variables Impacting the Productivity of Software Maintenance Projects," *Manag. Science*, vol. 37. pp. 1–18, 1991.

[15] W. D. Brooks, "Software Technology Payoff: Some Statistical Evidence," *J. Syst. Softw.*, vol. 2, no. 1, pp. 3–9, 1981.

[16] G. R. Finnie, G. E. Wittig, and D. I. Petkov, "Prioritizing software development productivity factors using the analytic hierarchy process," *J. Syst. Softw.*, vol. 22. pp. 129–139, 1993.

[17] B. Lakhanpal, "Understanding the factors influencing the performance of software development groups," *Inf. and Softw. Tech.*, v. 35. pp. 468–473, 1993.

[18] J. Vosburgh, et al, "Productivity factors and programming environments," in *ICSE '84 Proceedings of the 7th International Conference on Software Engineering*, 1984, pp. 143–152.

[19] C. Wohlin and M. Ahlgren, "Soft factors and their impact on time to market," *Softw. Qual. J.*, vol. 4, pp. 189–205, 1995.

[20] C. Wohlin and A. Andrews, "Assessing Project Success Using Subjective Evaluation Factors," *Softw. Qual. J.*, v. 9, pp. 43–70, 2001.

[21] R. H. Rasch, "An Investigation of Factors That Impact Behavioral Outcomes of Software Engineers," in *Proceedings of the 1991 Conference on SIGCPR*, 1991, pp. 38–53.

[22] V. Lalsing, S. Kishnah, and S. Pudaruth, "People Factors in Agile Software Development and Project Management," *Int. J. Softw. Eng. Appl.*, vol. 3, pp. 117–138, 2012.

[23] R. E. Boyatzis, "Competencies in the 21st century," *J. Manag. Dev.*, vol. 27, no. 1, pp. 5–12, 2008.

[24] A. Shirazi and S. Mortazavi, "Effective management performance a competency-based perspective," *Int. Rev. Bus. Res. Pap.*, vol. 5, no. 1, pp. 1–10, 2009.

[25] C. Melo, et al, "Agile Team Perceptions of Productivity Factors," in *Agile Conference, 2011*, pp. 57–66.

[26] M. Coram and S. Bohner, "The impact of agile methods on software project management," in *Proc. of IEEE ECBS '05*, 2005, pp. 363–370.

[27] V. R. Basili, G. Caldiera, and H. D. Rombach, "The goal question metric approach," *Encycl. Softw. Eng.*, vol. 2, pp. 528–532, 1994.

[28] Mark Hall, et al (2009); The WEKA Data Mining Software: An Update; *SIGKDD Explorations*, Volume 11, Issue 1

[29] K. Kira and L. A. Rendell, "The Feature Selection Problem: Traditional Methods and a New Algorithm," in *Proc. of the Tenth National Conference on Artificial Intelligence*, 1992, pp. 129–134.

[30] R. Kohavi and G. H. John, "Wrappers for feature subset selection," *Artif. Intell.*, vol. 97, no. 1–2, pp. 273–324, Dec. 1997.