UDC 378.147:004.8

Oleksandr Sharyhin
PhD of Technical Sciences,
Senior lecturer of the Department of Information Technologies and Programming
Dragomanov Ukrainian State University, Kyiv, Ukraine,
Project Lead,
Miratech, Kyiv, Ukraine
ORCID ID 0009-0006-9405-6997
exhaustic@gmail.com

Vasyl Fedorets
PhD of Medical Sciences, Associate Professor,
Associate Professor of the Department of Pedagogy, Administration and Special Education,
State Institution of Higher Education ‘‘University of Educational Management”
National Academy of Educational Sciences of Ukraine, Kyiv, Ukraine,
Associate Professor of the Department of Pedagogical Sciences, Primary and Correctional Education,
Public Higher Educational Establishment ‘‘Vinnytsia Academy of Continuing Education”, Vinnytsia, Ukraine
ORCID ID 0000-0001-9936-3458
bruney333@yahoo.com

Oksana Klochko
Doctor of Pedagogical Sciences, Professor,
Professor of the Department of Mathematics and Informatics,
Vinnytsia Mykhailo Kotsiubynskyi State Pedagogical University, Vinnytsia, Ukraine
ORCID ID 0000-0002-6505-9455
klochkoob@gmail.com

MONITORING AND ANALYSIS OF STUDENTS' PERFORMANCE DURING SOFTWARE DEVELOPMENT

Abstract. This study presents a comprehensive approach to monitoring and analyzing students’ performance in the context of software development projects. The primary focus is on the development of novel performance metrics that effectively capture the level of control required from the Team Leader and the quality of work assessed by the QA Engineer. The introduced metrics, namely Relative Control Degree and Relative Quality Assurance Degree, offer a robust framework for evaluating and understanding students’ performance across project, company, and industry levels.
This study presents an innovative application that calculates the defined performance metrics, allowing for real-time monitoring of students’ performance as a Developer. This practical tool offers means to track and evaluate student progress throughout the project lifecycle, enabling timely feedback and intervention as needed.
The authors provide recommendations for interpreting the results and utilizing the proposed performance metrics.
The utilization of the developed performance metrics holds significant implications for project management and student development. By implementing the recommended practices, educational institutions and industry professionals can enhance the effectiveness of software development projects, improve student learning outcomes, and promote a culture of continuous improvement and innovation.
The following methods, models, and approaches were used to solve the task of improving monitoring and analyzing the progress of students in the process of participating in software development projects: System and Problem Approaches, Cluster Analytical and Monitoring Models of Software Development, Expert Survey, Consistency Index, Consistency Ratio, the Fundamental Scale of Absolute Numbers by T. L. Saaty, and others.
The proposed performance metrics, supported by recommendations and a dedicated application, empower project stakeholders to optimize control, quality assurance, and student development.

Keywords: project-based learning; indicators of students' performance; software development project; numerical and mathematical modeling; monitoring model of software development; cluster model.

1. INTRODUCTION

The problem statement. Nowadays, software development is traditionally presented as a technologized process dominated by the production and innovation component. An important feature of this activity is the high level of its innovation, reflexivity, dynamism and evolutionary nature, which is primarily driven by market demands. Accordingly, software development is a structured, specialized, intellectualized and technologized production process.

This process is formed on the basis of target, temporal and organizational structuring of activities in which repetitive cycles can be identified. Each such cycle can also be considered as a quantum of professional activity that can potentially be implemented within a certain period of time. The specified production cycle consists of the following: Team Leader sets a technical task for Developer and the subsequent quality control of the code created by the Developer; then the Developer creates the code; then Team Leader checks it; after that, software build (which is ready for testing) is transferred to QA Engineer.

It is important that this cycle of activities is aimed and adapted not only at solving typical tasks, but also at implementing non-standard, atypical and new tasks. This aspect of multidimensionality and variability, as well as the ability to work in a state of cognitive uncertainty, determines the importance of the pedagogical component that is contextually present in the software development process. In addition, constant adaptation to the requirements and demands of the market determines the need for continuous training of specialists in this area, thereby defining the pedagogical component, which is considered as an actual and formalized dimension of professional activity. An important aspect that requires updating the pedagogical component is also the work on mistakes.

The technology-oriented understanding of the presented importance of the pedagogical component as a systemic and cross-cutting condition necessary for effective software development determines the central and system-organizing role of monitoring and analysis of students' performance. Monitoring ensures the implementation of corrective, informational, analytical, evaluative, stimulating, motivational regulatory and prognostic functions that give the possibility to carry out training and software development integrally – as a systematic, cyclic and multidimensional process aimed at achieving production goals and the development and improvement of a specialist. Accordingly, the analysis based on the data obtained through monitoring both underlies the formation of a specialist and is a prerequisite for his/her professionalization and professional and personal growth, and is also a significant factor in further improvement of production.

Analysis of recent studies and publications. Let us analyze the current research that is relevant to our topic.

B. Fernandez-Gauna, N. Rojo, M. Graña studied the process of evaluating team coding tasks based on DevOps technologies [1]. They developed a system for automating the evaluation of team tasks and a methodology for its use. According to the developed methodology, the progress of a group of students is tracked based on the performance metrics of the respective group, individual indicators of each student's contribution to the overall team performance are measured, reports are generated using these indicators and feedback on the work of the team and each student as a team member [1].

Another approach to improving the efficiency of teamwork was applied by M. Wijga, M. D. Endedijk, and B. P. Veldkamp [2]. The focus of their research was to increase productivity based on team reflexivity. In contrast to the traditional approach to team reflection, which was implemented upon completion of individual stages of work, they proposed an approach based on reflective moments during each stage of work. The researchers propose to study team reflection through social regulation using a model consisting of 4 components - "joint construction of knowledge and regulation; regulation activities; regulation focus and type of interaction" [2].

An important area of software engineering, which is the prediction of software defects, was investigated by J. Hernández-Molinos, A. J. Sánchez-García, R. E. Barrientos-Martínez, J. C. Pérez-Arriaga, and J. O. Ocharán-Hernández [3]. The study evaluated three algorithms for assessing a project's susceptibility to defects using Bayesian approaches, since algorithms for building Bayesian networks are less variable [3].

N. Salleh, B.I. Ya'u, and A. Nordin conducted a study on the effectiveness of requirements engineering teams and the impact of team members' personal qualities on their performance [4]. This study was conducted on the basis of scientists' research. The researchers have found that the effectiveness of requirements engineering (RE) depends on the personal qualities of RE engineers and their team behavior, in particular, extroversion and conscientiousness are important personal qualities for them [4]. Also, the dominant personal qualities in RE teamwork are flexibility, cooperation, creativity, innovation, and compliance with established norms [4]. The researchers point out that this issue remains open today.

A person-oriented approach based on the empowerment approach is important in the project activities of future specialists, as it is based on increasing confidence, predicting the results of one's own activities, responsibility for decision-making, which is important for team project work and increasing the effectiveness of professional training [5]. The authors propose to use such basic approaches as competence, system, activity and others in the training of students of engineering specialties, which we also used in this study [6].

Y. Jin, Y. Bai, Y. Zhu, Y. Sun, and W. Wang addressed code recommendation for predicting behavior of future developers [7]. They proposed the graphical CODER framework for the interaction of developers in the process of developing Open Source Software (OSS) [7]. This framework provides opportunities for joint modeling of interactions.

A. B. Bondi and L. Xiao studied the issue of improving software development courses in order to improve students' knowledge of software performance during the development life cycle, since these issues are often not fully considered in the learning process [8].

T. Fatma, N. A. Khan, and S. Sarwar reviewed software defect evaluation techniques [9]. They emphasize that the field of software development requires communication between all parties involved in the process. Evaluation of software defects and their reduction is important for the customer. The methods of support vector machines, C4.5, batching, naive Bayes, K-nearest neighbors, neural networks, random forest, decision trees and radial displacement were used to improve the efficiency of defect estimation [9]. According to the results of the research, Support Vector Machine is the most effective tool for solving these tasks [9].

G. Mahajan and N. Chaudhary dealt with issues of localization of software errors to improve the quality of software and the efficiency and productivity of teams due to these actions [10]. They developed a convolutional neural network based on deep learning, which includes eaps of file preprocessing, feature vector extraction, and optimized error localization classification [10].

S. Shafiq, A. Mashkoor, C. Mayr-Dorn, and A. Egyed studied the issue of an Agile approach to the distribution of tasks in a team based on roles [11]. They suggested defining team roles rather than individual roles for project managers to effectively allocate tasks. The evaluation was done by integrating a plugin that provides recommendations for relevant roles with JIRA [12].

T. N. E. Amarasekara, H. G. P. Isurindi, E. H. D. T. D. Navanjana, O. M. Gamage, Uthpala Samarakoo, and Archchana Kugathasan proposed an advanced project management system for developing software to manage student work [12]. Their goal was to develop a system for monitoring students' work, creating project groups and interacting with clients [12].

In [13] some software development metrics (performance, quality, maintainability, true test coverage, team velocity, escaped defects, release burndown) are proposed.

Unsolved aspects of the problem. In the scientific pedagogical literature, the problem of monitoring and analysing students' progress in the process of participating in a software development project is insufficiently covered, which, as well as its importance for the effective training of information technology specialists, determines its relevance. Such important components of this issue as the development of digital and mathematical models for monitoring and analysing students' progress in the process of participating in a software development project are not sufficiently developed either.

The research goal. The purpose of the research is to improve the monitoring and analysis of students' progress in the process of participating in a software development project based on the use of digital and mathematical modelling.

2. RESEARCH METHODS

To achieve the goal of improving the monitoring and analysis of students' progress in the process of participating in a software development project, we use several models and approaches. Initially, we use the systemic and problem-based approaches for monitoring and analysis, which is in line with established traditions. These approaches determine the effectiveness of professional activities and training, and monitor the time spent on certain tasks.

We are also developing the Cluster Analytical and Monitoring Model of Software Development, which is aimed at intellectualizing, optimizing and technologizing the analysis and monitoring processes. The idea behind this model is to analyze software development activities based on structuring them in accordance with the logic of building certain clusters, followed by their research and interpretation. This approach allows us to reveal seemingly understandable, familiar and trivial phenomena in new purposefully defined formats, in which the meanings are initially set. In essence, a cluster matrix of software development is formed, which is variable and can be changed in accordance with requests. The use of the cluster approach also makes it possible to consider data that are generally considered irrelevant and that may at the same time play the role of weak factors, the effects of which manifest themselves in the long term or are implemented “disguised” as other conditions and factors. As one of the possible options, we propose a model, where clustering is carried out on the basis of several criteria that define clustering. Accordingly, we distinguish cluster structuring according to the following criteria, which integrate several (2 or more) aspects: goal-result, process-result, time-result, innovation (novelty)-result, typical tasks-result, errors-time, communication (teamwork)-result, compliance-result, result-efficiency.

In the “Cluster Analytical and Monitoring Model of Software Development”, monitoring and analysis are integrated and represent analytical and synthetic tools for correcting and improving professional and educational activities. Accordingly, in this model, we distinguish the following levels: monitoring, analytical and reflective, synthetic and interpretive, metacognitive and strategic. The first level of monitoring is aimed at obtaining broader or targeted amounts of relatively reliable information and data systems about activities and learning. The second level is analytical and reflective, which involves the primary processing of the data and information obtained, for which traditional systemic, targeted, problematic, reflective and other approaches are used in an integrative manner, as well as cluster and other approaches that operate with large data sets. The third level is synthetic and interpretive, which reveals the information obtained by correlating it with various criteria and semantic contexts. At this level, the analytical component is actualized through interpretation. The fourth metacognitive-strategic level is aimed at a practically oriented synthesis with the subsequent development of practical strategies for improving production and educational processes.

This hierarchization of analytical and monitoring activities makes it possible to purposefully update and form at each level its own specific architecture of functions, which makes it possible to maximize the technological, algorithmic and illustrative nature of the studied issues. In essence, the "Cluster Analytical and Monitoring Model of Software Development" is an integrative anthropo-techno-digital cognitive and epistemological system with a corresponding target orientation.

Let's consider typical development process in Agile technology, which uses Jira (Figure 1). Epics in Agile methodology are large bodies of work that can be broken down into a number of smaller tasks (called stories) [14]. Epic provides a higher level of abstraction, it helps to organize and group different stories/tasks into one logical block. Besides, epics may contain bugs, which QA Engineers find and register during testing of functionality related to the given epic.

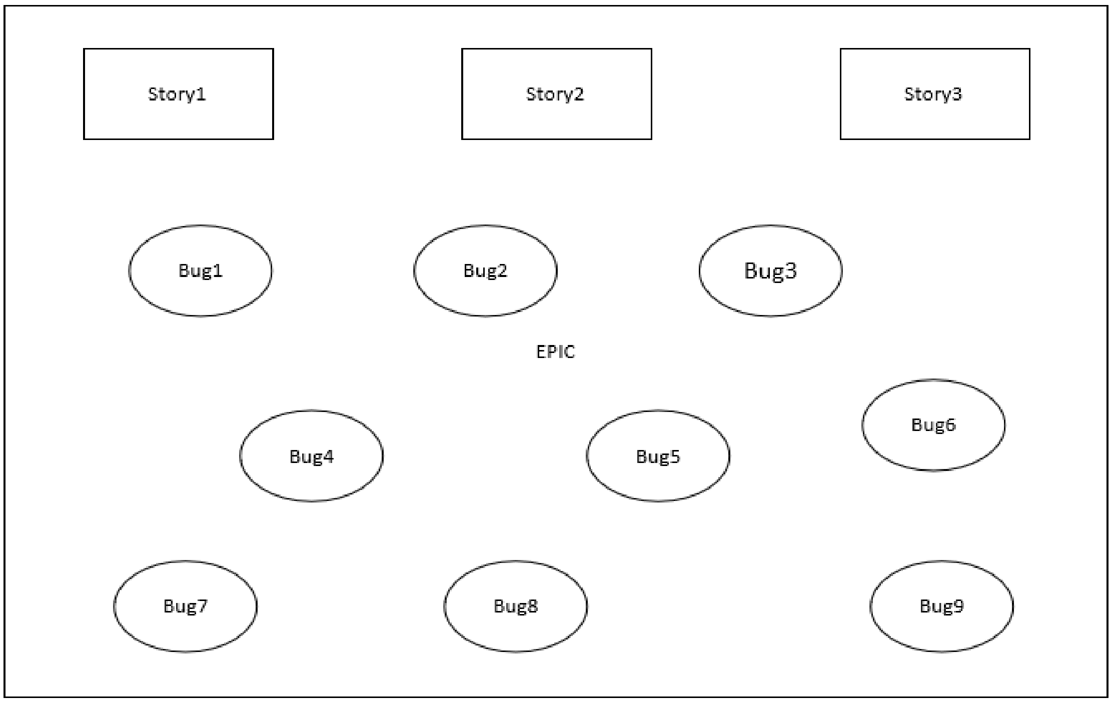


Figure 1. Epic with stories and bugs inside

Before the implementation, a person responsible for epic or Team Leader provides estimation of the epic and all stories inside the epic. Epic estimation contains time needed for testing and bug-fixing. Estimation can be done using different approaches: story points or functional points. The way of estimation is out of scope of this article, what really important is the fact that the final grade will be measured in hours.

Logging of time is required from all team members, it would not be possible to make any performance calculations without it (Table 1, Figure 2).

We propose to define metrics which give understanding of the Developer performance. The simplest approach to evaluate performance is to compare the story estimated time and the actual time the Developer spent on the story implementation. It can be called time ratio and is calculated using formula (1):

$$TR_i = \frac{tact_i}{test_i}, \quad (1)$$

where $tact_i$ – actual time which was spent on the implementation of story St_i, $test_i$ – estimated time of story St_i.

Table 1

Activities which require logging time for different JIRA roles / items

JIRA item \ Role	Team Leader	Developer	QA Engineer
Story	Pull request review	Development Fix according to pull request comments	Testing
Bug	Analysis and linking to story Pull request review	Fix Fix pull request comments	Registering Testing

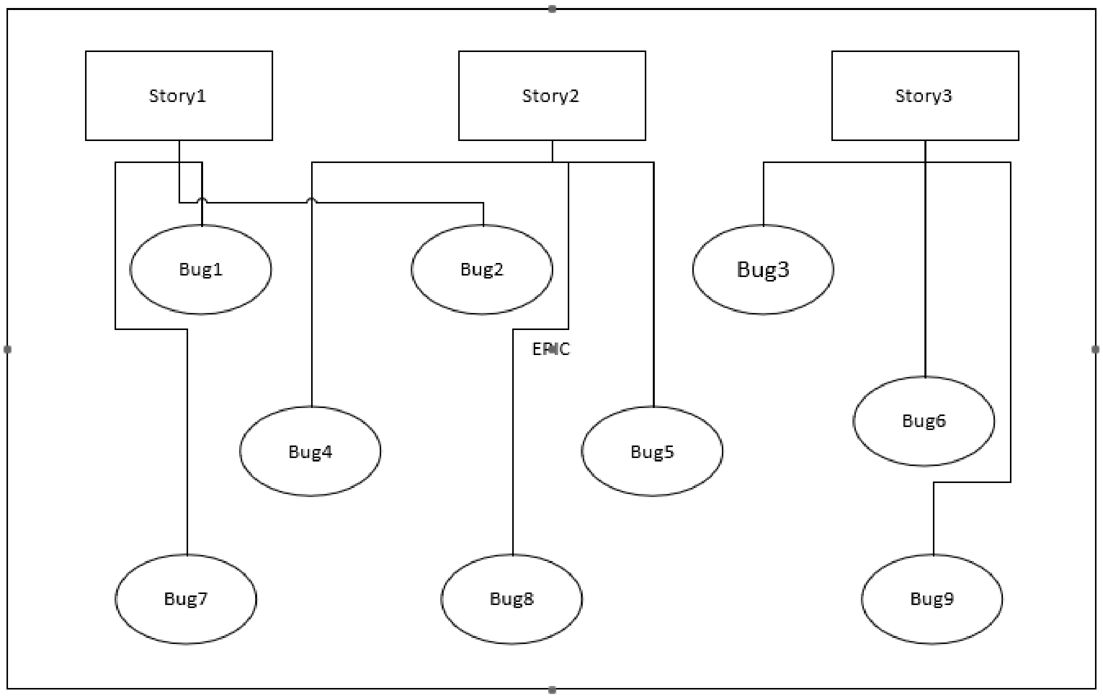


Figure 2. Epic with stories, bugs and relations between them

We defined our own metrics for a student (acting as a Developer) performance, so it is not possible to use standard features of JIRA to monitor values of the defined metrics. However, it is feasible to achieve it in an automated way using Atlassian SDK [15] Nuget package.

Formula 1 does not take into consideration time spent on bugs found by QA Engineer while testing functionality of this story, but its value definitely depends on the developer performance. So, for more correct evaluation it is required to have a link between a story and a bug (inside the same epic). Analysis of bugs and determining which story they belong to are responsibilities of the Team Leader. Jira can link any items, including stories and bugs.

Taking into consideration relations between stories and bugs, we can specify:

$$TR_i = \frac{tact_i + \sum_{j=1}^{nb_i} tb_{ij}}{test_i}, \tag{2}$$

where nb_i – number of bugs related to story St_i, tb_{ij} – time spent on finding and resolving bug b_j related to story St_i.

Ideally, the value of this metric should tend towards 1. It means that Developer performance is as expected (because performance of Team Leader and QA Engineer are as expected by default). Obviously, this metric is characteristic of the team, not a team member.

By applying the methodological ideas laid down in the Cluster Analytical and Monitoring Model of Software Development we have developed to analyze and structure this metric, we get a somewhat expanded understanding of this issue. According to this model, the activity in the system of this metric can be structured into the following blocks (clusters), which are simultaneously considered as indicators: “mistakes-time”, “process-result”, “time-result”, “communication-result”, “goal-result”, “compliance-result”. “Process-result” reveals the procedural nature of the student's activity in relation to its result. This indicator also raises the issue of the need for such proceduralism, because the result, including the correction of errors, does not come from nowhere, but is formed through activities that may be larger or smaller in scope and time spent. Since this indicator focuses on performance rather than time spent, it allows effective error correction to be viewed in terms of the cognitive potential of the team and a particular specialist. The “time-result” system as an integrative temporal indicator of activity reveals not only the speed of work on errors, but also represents the time spent on the activity in general, including errors in relation to the final result, which includes work on the whole epic. “Communication-result” represents the ability to actively and effectively communicate professionally in the process of working on bugs as well as in general. This indicator indirectly reflects the cognitive potential of both the development team and the individual specialist. Thanks to professional communication, internal knowledge is transformed into external knowledge and vice versa, which ultimately leads to an increase in knowledge, as well as to its practical orientation and specification (in the sense of specification). “Goal-result” as an indicator diagnoses and accordingly updates metacognitive strategies of goal setting and reflection of one's resources (including, first of all, professional and intellectual ones), which is aimed at improving the efficiency of professional activity, including work on mistakes. The “compliance-result” indicator clarifies and specifies goal setting by directing activities to achieve the planned result first and foremost. This indicator also contains a reflexive component, which is seen as a potential ability to find and correct errors. The introduction of these clusters (blocks) of activities, which are considered as relevant indicators in the monitoring system, is aimed at revealing and shaping the professional potential of a specialist by actualizing his or her resources through reflection, communication, goal setting, motivation, intellectualization, responsibility, and subjective temporality. This approach, also in the context of teamwork, is personality-oriented, as well as humanizing, intellectualizing, creative, and psychologizing both learning and professional activity. It represents a departure from the established scheme, which is rather linear and similar to a “temporal conveyor belt”. We use these blocks (clusters) in the study in different metrics.

We propose the following metrics for evaluation of student (acting as a Developer) performance:

1. Control Degree
2. Testing (Quality Assurance) Degree

Let’s review the above-mentioned performance metrics in detail.

2.1. Control Degree

This metric evaluates involvement of Team Leader in the development of a story/epic. Team leader takes his/her time to:

1. Pull requests review.
2. Analyze a bug and link it to a story.

It is a common practice that a Team Leader finds some issues while reviewing code in pull request and asks the Developer to fix them. Normally this process is linear:

1. Developer creates pull request for a story/bug.
2. Team Leader reviews it and comments.
3. Developer fixes code issues mentioned in comments.
4. Developer updates pull request for the story/bug.
5. Team Leader merges it and the story/bug is available for testing.

But in practice it is an iterative process, as shown in Figure 3.

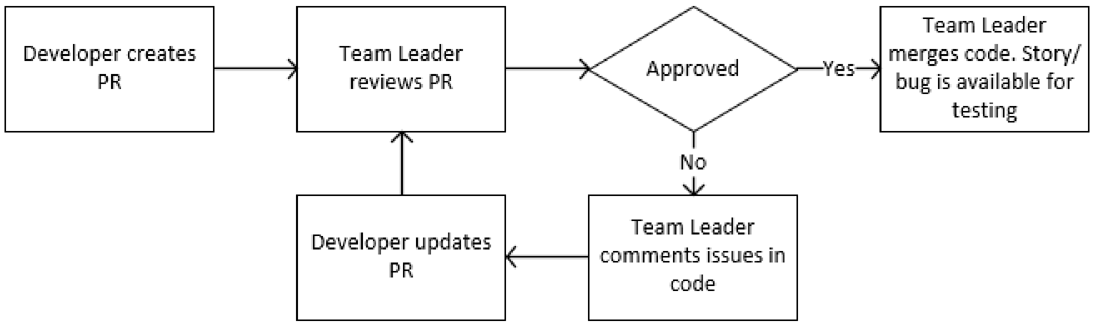

Developer creates PR
Team Leader reviews PR
Approved
Yes
Team Leader merges code. Story/ bug is available for testing
No
Team Leader comments issues in code
Developer updates PR

Figure 3. Epic with stories, bugs and relations between them

Absolute value of *Control Degree* can be calculated by using formula (3):

$$CDabs_i = \frac{tact_{iTL}+\sum_{j=1}^{nb_i} tb_{ijTL}}{tact_i+\sum_{j=1}^{nb_i} tb_{ij}}, \tag{3}$$

where $tact_{iTL}$ – actual time spent by Team Leader on the story St_i, tb_{ijTL} – time spent by Team Leader on the bug (bug analysis, linking it to a story and review of pull requests created by Developer) b_j related to the story St_i.

The value of this metric reflects the degree of the Team Leader's involvement in the implementation of the story, and therefore the degree of the Team Leader's control. The range of possible values is (0, 1). 0 means that the Team Leader did not spend any time, 1 means that only the Team Leader was working at the story. Both of these boundary cases are impossible, so the range of values does not include the bounds. The closer the value is to 0, the less participation of the Team Leader, and vice versa, the closer the value is to 1, the greater the participation of the Team Leader.

The experience of projects at the Miratech IT company [16] shows that for stable teams (without students or interns), the value of this metric should be in the range [0.06; 0.12]. That is, if the value is in this range, the student works at the level of a specialist.

If the value is 0.2 or more, the Team Leader spends a lot of time monitoring the implementation of tasks, and therefore the student's work is not successful (very unautonomous). In the process of student participation in this project, it is important that this value decrease and get as close as possible to the interval [0.06, 0.12].

It is also of interest to compare the Absolute Control Degree with the average value of absolute control degrees. Depending on situation, different options can be used as this parameter value:

1. Average Control Degree by industry.
2. Average Control Degree by company projects.

3. Average Control Degree by current project (but only stories where students did not participate, because they are not indicative).

So, in case of using the third option, the calculation Relative Control Degree can be evaluated using formula (4):

$$CDrel_i = \frac{CDabs_i \times ns}{\sum_{j=1}^{ns} CDabs_j} \tag{4}$$

where ns – number of stories in the project where students did not participate.

Student study is successful if the Relative Control Degree value decreases and tends to 1.

2.2. Quality Assurance Degree

This metric gives the possibility to evaluate involvement of QA team in the implementation of a story/epic. QA Engineer spends time on:

1. Story testing.
2. Bug registering.
3. Bug testing.

Taking into consideration activities of QA Engineers, Figure 3 transforms into more detailed Figure 4.

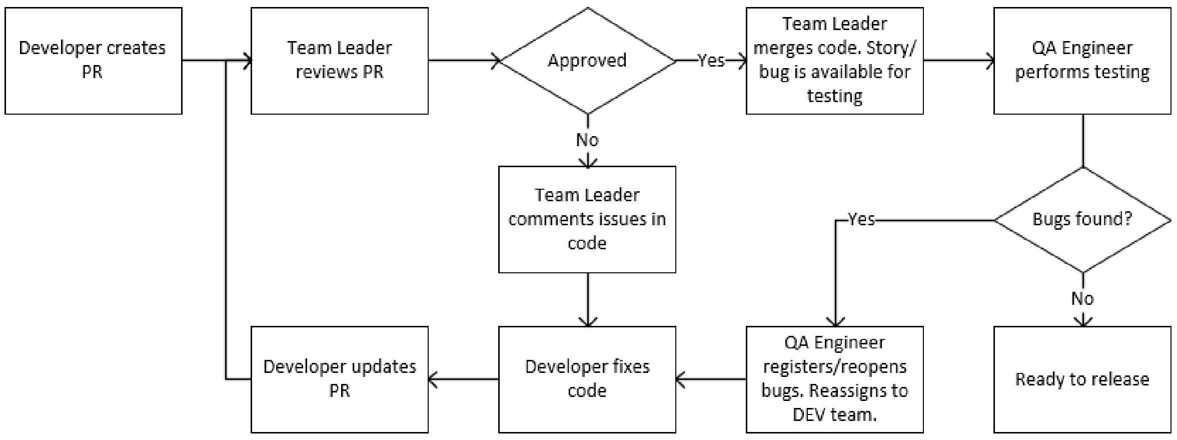

Developer creates PR
Team Leader reviews PR
Approved
Yes
Team Leader merges code. Story/ bug is available for testing
QA Engineer performs testing
No
Team Leader comments issues in code
Yes
Bugs found?
No
Developer updates PR
Developer fixes code
QA Engineer registers/reopens bugs. Reassigns to DEV team.
Ready to release

Figure 4. Full workflow of an epic's story

Absolute value of *Quality Assurance Degree* can be calculated by using formula (5):

$$QADabs_i = \frac{tact_{iQA} + \sum_{j=1}^{nb_i} tb_{ijQA}}{tact_i + \sum_{j=1}^{nb_i} tb_{ij}}, \tag{5}$$

where $tact_{iQA}$ – actual time spent by QA Engineer on a story St_i, tb_{ijQA} – time spent by QA Engineer on a bug (registering/testing) b_j related to story St_i.

The experience of projects at Miratech [16] shows that for stable teams (without students/interns), the value of this metric should be in the range of [0.2-0.3] on average. That is, if the value is in this range, the student works at the level of an established specialist.

If the value is 0.4 or more, it means that the student acting as a Developer makes a lot of bugs in the business logic and QA Engineers spend a lot of time checking the task, and therefore the student's work is not successful. In the course of the student's participation in this project, it is important that this value decrease and get as close as possible to the interval [0.2, 0.3].

The value of this metric reflects the degree of QA Engineer involvement in the implementation of the story. The range of possible values is (0, 1). The closer the value is to 0, the less participation of QA Engineer, and vice versa, the closer the value is to 1, the greater the participation of QA Engineer.

As with Control Degree, it is also interesting to calculate not only the absolute value, but the relative value too. Relative Quality Assurance Degree can be calculated by formula (6):

$$QADrel_i = \frac{QADabs_i \times ns}{\sum_{j=1}^{ns} QADabs_j}. \quad (6)$$

Student study is successful if the Relative Quality Assurance Degree value decreases and tends to 1.

2.3. Evaluation of metrics

In order to assess the effectiveness of the implemented methodology of teaching students using the metrics of Absolute/Relative Control Degree and Absolute/Relative Quality Assurance Degree, we conducted a survey. The experts in determining the feasibility of using this methodology were the following:

– an experienced specialist, who performs the role of Team Leader on the project;
– experienced specialists, who perform the role of QA Engineer;
– student teachers, who monitor the learning process;
– students, who performed the role of Developer while participating in the project.

There were 3 categories of questions proposed:

1. Questions related to Absolute/Relative Control Degree metrics (1, 3, 5);
2. Questions related to Absolute/Relative Quality Assurance Degree metrics (2, 4, 5);
3. General questions (6).

The questions of the questionnaire, which were answered by the experts, are as follows:

1. Does taking into account the average *Control Degree (Absolute/Relative)* metrics for a particular student acting as a Developer help to assess the involvement of *Team Leader* when planning Epic of *standard/typical functionality*?
2. Does taking into account the average *Control Degree (Absolute/Relative)* metrics for a particular student acting as a Developer help to assess the involvement of *Team Leader* when planning Epic of *non-standard/custom functionality*?
3. Does taking into account the average *Quality Assurance Degree (Absolute/Relative)* metric for a particular student acting as a Developer help to assess the involvement of QA *Engineers* when planning Epic of *standard/typical functionality*?
4. Does taking into account the average *Quality Assurance Degree (Absolute/Relative)* metric for a particular student acting as a Developer help to assess the involvement of QA *Engineers* when planning Epic of *non-standard/custom functionality*?
5. How long does it take to interpret the calculated metrics values?
6. What can be improved in the calculation of *Control Degree* and *Quality Assurance Degree* metrics?

The number of points that experts can give to questions 1-4 (where 1 is the lowest value and 10 is the highest value):

– 1..2 – does not help;

- 3..5 – mostly does not help;
- 6..8 – mostly helps;
- 9..10 – helps to the fullest extent.

The number of points that experts can give to question 5 (where 1 is the lowest value and 10 is the highest value):

- 1..2 – too much time, which makes the use of metrics impossible;
- 3..5 – a significant amount of time, which makes the use of metrics inconvenient;
- 6..8 – an insignificant amount of time that does not significantly affect the time performance;
- 9..10 – the interpretation of the values is obvious, there is no negative impact on the time indicators.

Answers to questions 1 and 3 characterize the extent to which the proposed approach can be used to assess the reproductive quality of a student, and answers to questions 2 and 4 characterize the cognitive component.

To optimally determine the number of points that assess the possibility of using the proposed approach, we introduced weights according to the competence of each expert in industrial project development (Table 2).

Table 2

Weights (coefficients) of answers to each question, depending on roles

Role	Question 1	Question 2	Question 3	Question 4	Question 5
Team Leader	4	2	4	2	4
QA Engineer	1	3	1	3	1
Professor	2	2	2	2	2
Student Developer	1	1	1	1	1

The Consistency Index (CI), (formula (7)), Consistency Ratio (CR) formula (8)), and the Fundamental Scale of Absolute Numbers by T. L. Saaty [17] were used to determine the consistency of expert opinions:

$$CI = \frac{\lambda_{max} - n}{n-1}, \tag{7}$$

where λ_{max} – largest eigenvalue of the matrix, n – the number of elements compared.

$$CR = \frac{CI}{RI}, \tag{8}$$

where RI – random index, determined according to the table [17].

3. THE RESULTS AND DISCUSSION

3.1. Developing application

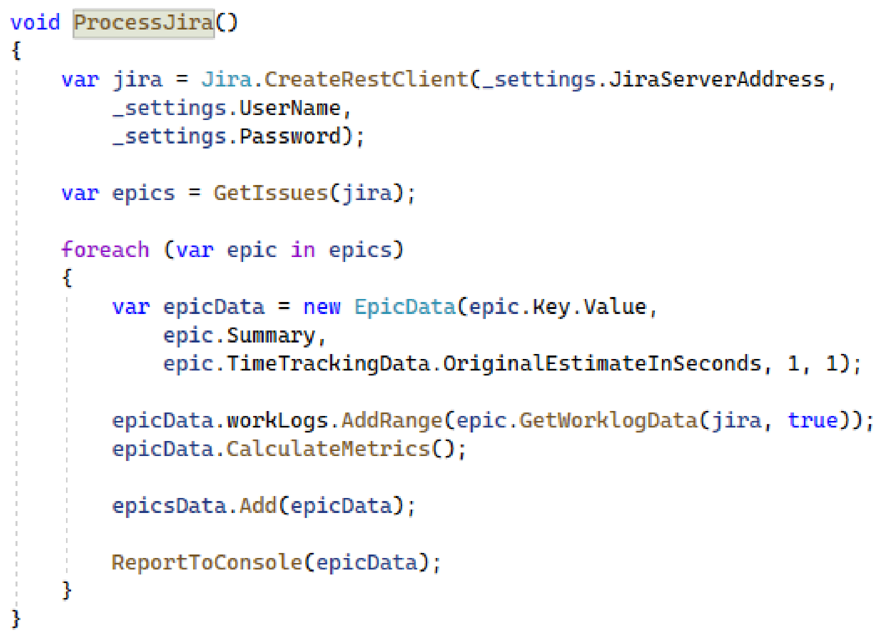
In this study, we developed a console application utilizing C#, .NET Framework and above-mentioned SDK to establish access to the Jira server. The application retrieves comprehensive data, including information about an epic, stories, bugs, work logs and estimations. It is possible to use it in different environments, as the application supports settings/configuration files.

Using data retrieved from JIRA server, the application calculates:

- average Control Degree for the current project (stories where experienced Developers only participated);

– average Quality Assurance Degree for the current project (stories where experienced Developers only participated);
– Absolute Control Degree for every story;
– Relative Control Degree for every story;
– Absolute Quality Assurance Degree for every story;
– Relative Quality Assurance Degree for every story.

Figures 5 and 6 present source code for initialization of connection with JIRA server and calculation of the above-mentioned metrics.



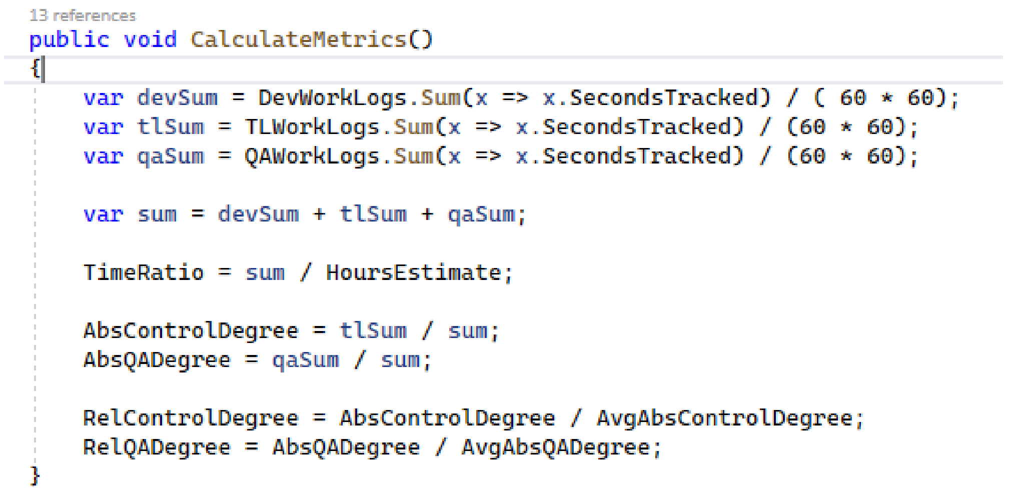
```
void ProcessJira()
{
    var jira = Jira.CreateRestClient(_settings.JiraServerAddress,
        _settings.UserName,
        _settings.Password);

    var epics = GetIssues(jira);

    foreach (var epic in epics)
    {
        var epicData = new EpicData(epic.Key.Value,
            epic.Summary,
            epic.TimeTrackingData.OriginalEstimateInSeconds, 1, 1);

        epicData.workLogs.AddRange(epic.GetWorklogData(jira, true));
        epicData.CalculateMetrics();

        epicsData.Add(epicData);

        ReportToConsole(epicData);
    }
}
```

Figure 5. Source code for JIRA server initialization and main loop of epics processing

We also configured pipeline which automatically updates statistics to reflect the inclusion of new stories. It triggers after completion of each epic. This ensures that the pipeline remains up-to-date and provides accurate insights into the progress and performance of students (acting as Developers).

3.2. Verification of the developed model and experiment

Gemicle [18], a company specializing in software development, built an internal portal, which is used by company employees. The project was led by an experienced Team Leader and supported by a skilled QA Engineer. The core functionality of the portal was developed by a dedicated developer, while three talented students were also part of the team, actively contributing as developers.



```
13 references
public void CalculateMetrics()
{
    var devSum = DevWorkLogs.Sum(x => x.SecondsTracked) / ( 60 * 60);
    var tlSum = TLWorkLogs.Sum(x => x.SecondsTracked) / (60 * 60);
    var qaSum = QAWorkLogs.Sum(x => x.SecondsTracked) / (60 * 60);

    var sum = devSum + tlSum + qaSum;

    TimeRatio = sum / HoursEstimate;

    AbsControlDegree = tlSum / sum;
    AbsQADegree = qaSum / sum;

    RelControlDegree = AbsControlDegree / AvgAbsControlDegree;
    RelQADegree = AbsQADegree / AvgAbsQADegree;
}
```

Figure 6. Source code for calculation of metrics

In addition to the core functionality, the experienced Developer took charge of creating CRUD (Create, Read, Update, Delete) operations for a couple of basic entities within the internal portal.

Based on these developments, the average Control Degree and the average Quality Assurance Degree for the current project were calculated.

Students of Vinnytsia State Technical University were integrated into the software development process. The student Developers were assigned the responsibility of developing CRUD operations for the other entities within the internal portal.

The students worked on the project activities for 3 months and participated in the development of 18 epics.

The average Control Degree and the average Quality Assurance Degree (both for the current project) were calculated first, their values are in Table 3.

Table 3

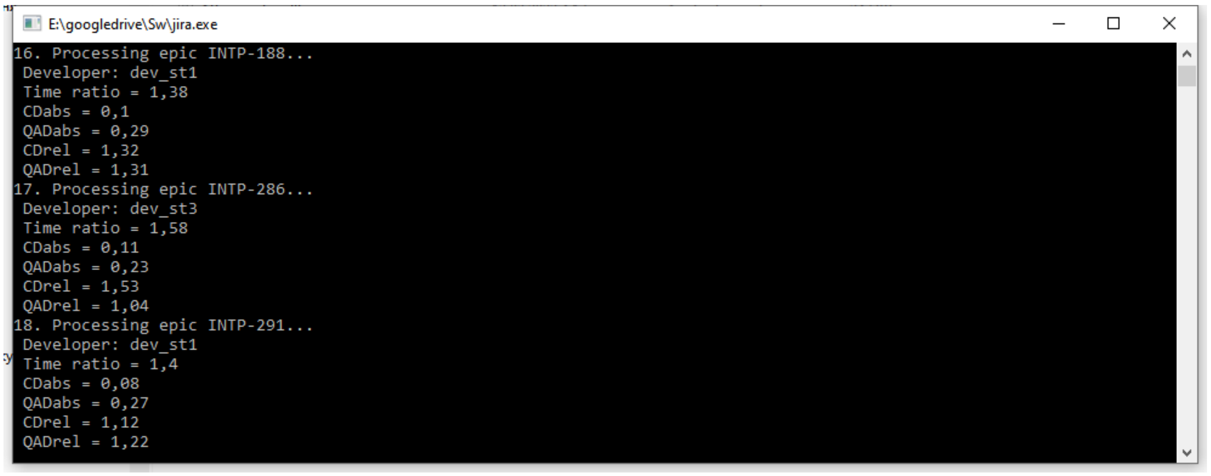
Average control degree and average quality assurance degree values

Average Control Degree	**Average Quality Assurance Degree**
0.073	0.221

Let’s review performance metrics calculated by application for dev_st1 (all names of team members are masked) and interpret them (Figures 7-12).

It is quite obvious that values of Absolute Control Degree and Absolute Quality Assurance Degree for dev_st1 are higher than Average Absolute Control Degree and Average Absolute Quality Assurance Degree respectively, but this student shows good progress because values for recent stories are significantly less than in the beginning of the project.

This indicates that the competencies of student Developers are improved, resulting in decreased need for control from the Team Leader, fewer pull request review iterations, and a reduction in new bugs and reopened ones by QA Engineers.



```
16. Processing epic INTP-188...
 Developer: dev_st1
 Time ratio = 1,38
 CDabs = 0,1
 QADabs = 0,29
 CDrel = 1,32
 QADrel = 1,31
17. Processing epic INTP-286...
 Developer: dev_st3
 Time ratio = 1,58
 CDabs = 0,11
 QADabs = 0,23
 CDrel = 1,53
 QADrel = 1,04
18. Processing epic INTP-291...
 Developer: dev_st1
 Time ratio = 1,4
 CDabs = 0,08
 QADabs = 0,27
 CDrel = 1,12
 QADrel = 1,22
```

Figure 7. Screenshot of console application with metrics values

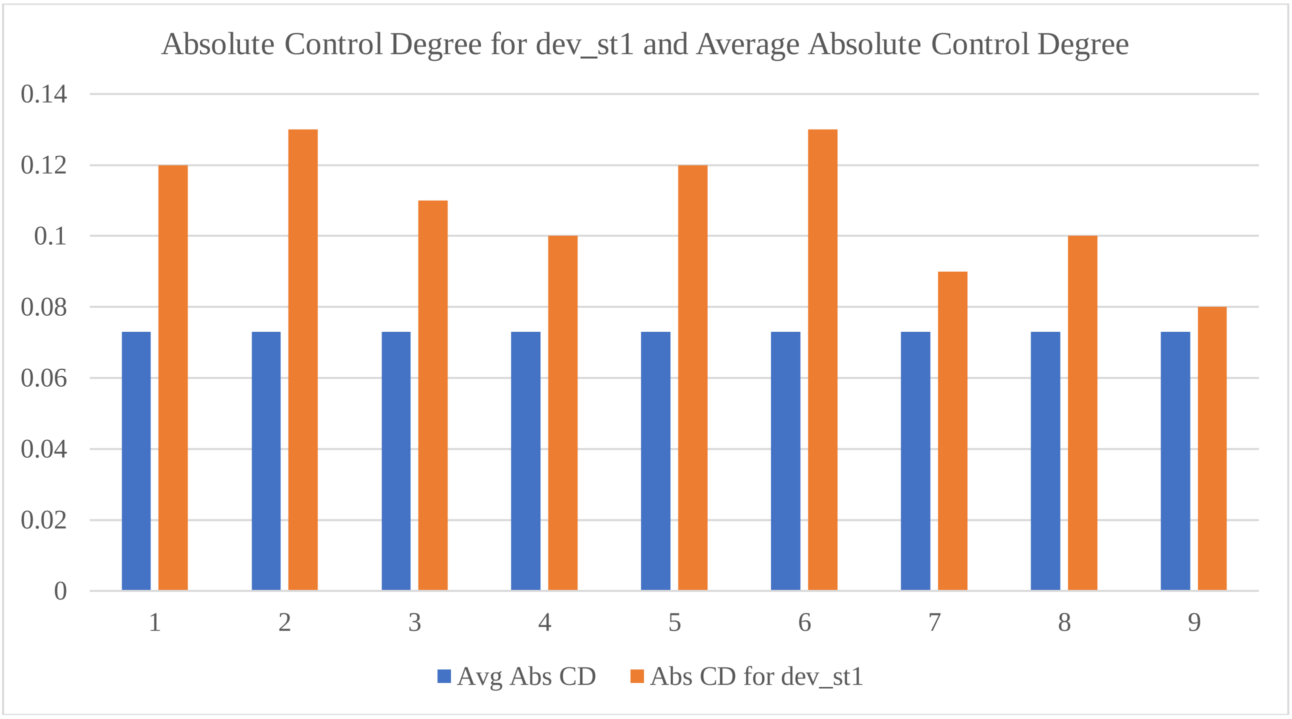


Figure 8. Absolute Control Degree for dev_st1 in comparison with Average Absolute Control Degree

In general, the results achieved by the student dev_st1 can be considered as successful.

We also would like to give an example of a student, whose results do not demonstrate any significant progress.

As we can see from Figures 13-15, Time Ratio, Relative Control Degree and Relative Quality Assurance Degree do not tend to decrease, and definitely do not strive to 1 (ideal value).

Also, we can see that this student (dev_st3) participated in only 4 tasks (in comparison with 9 tasks by dev_st1).

According to the proposed methodology for metrics evaluation in Section 2.3, we conducted a survey. The survey involved 11 respondents, according to their roles: Team Leader - 1; QA Engineer - 2; Professor - 3; Student/Developer - 5. The results of the survey are presented in Table 4. We have introduced the following notations in the table: Team Leader (TL), QA Engineer (QA Engineer), Student Developer (SD).

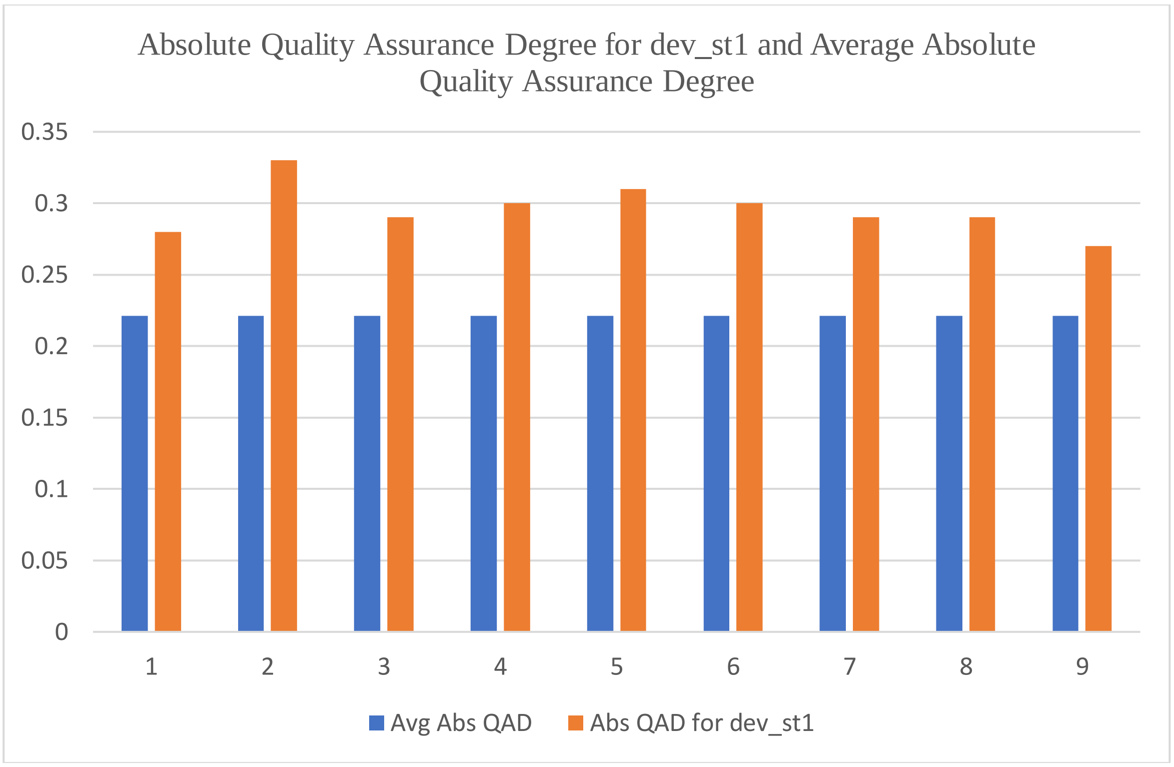


Figure 9. Absolute Quality Assurance Degree for dev_st1 in comparison with Average Absolute Quality Assurance Degree

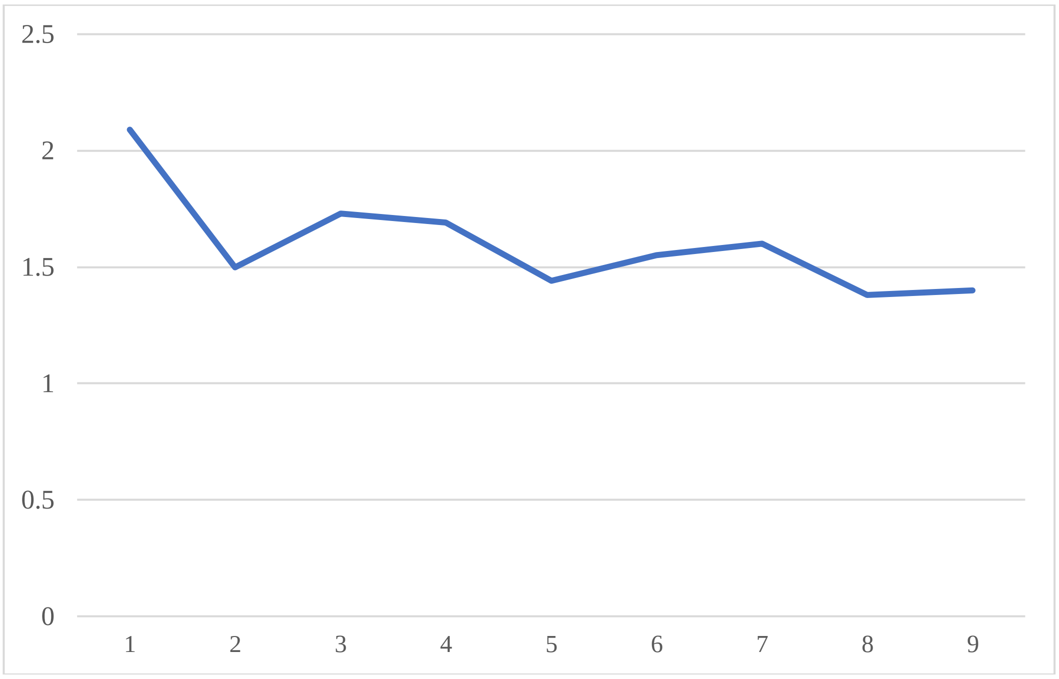


Figure 10. Time Ratio for dev_st1

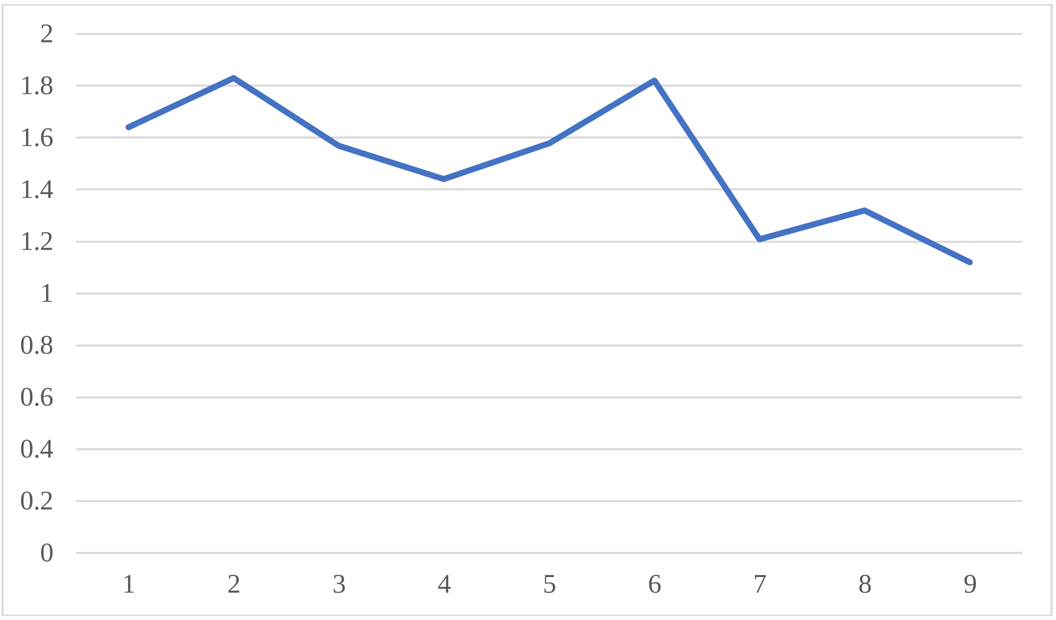


Figure 11. Relative Control Degree for dev_st1

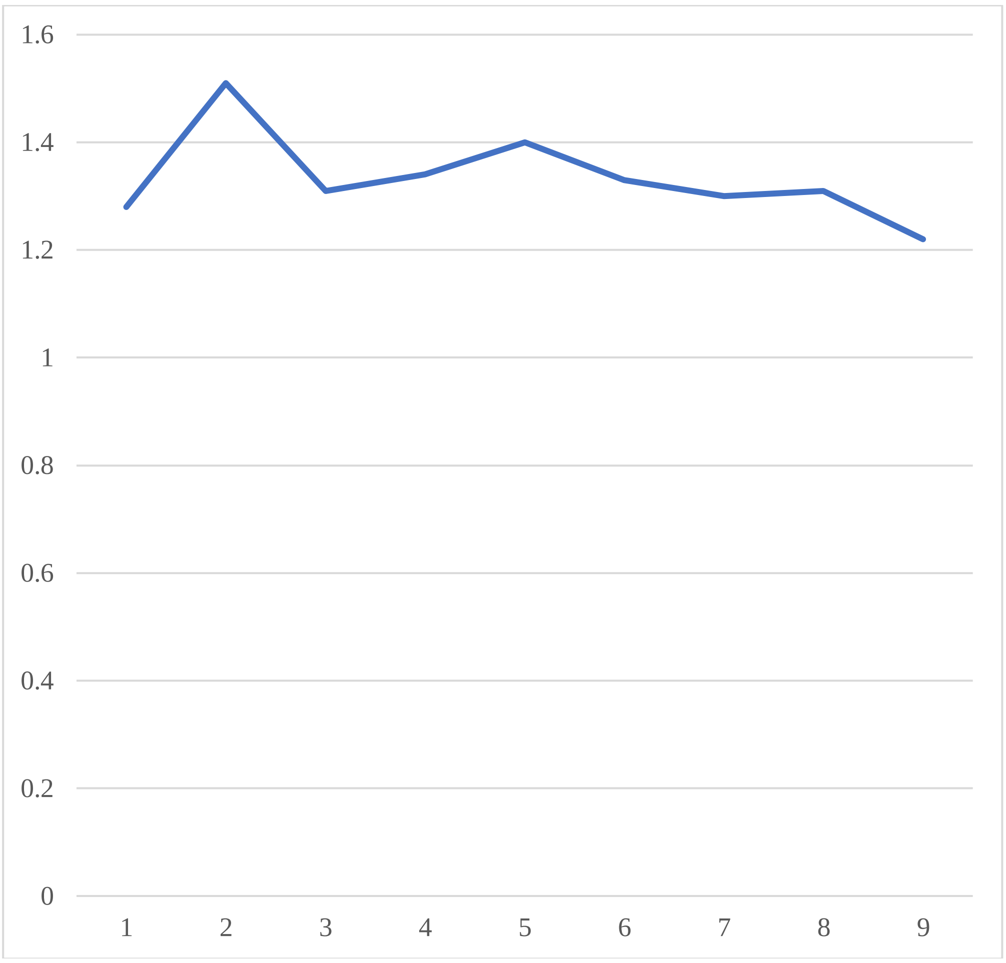


Figure 12. Relative Quality Assurance Degree for dev_st1

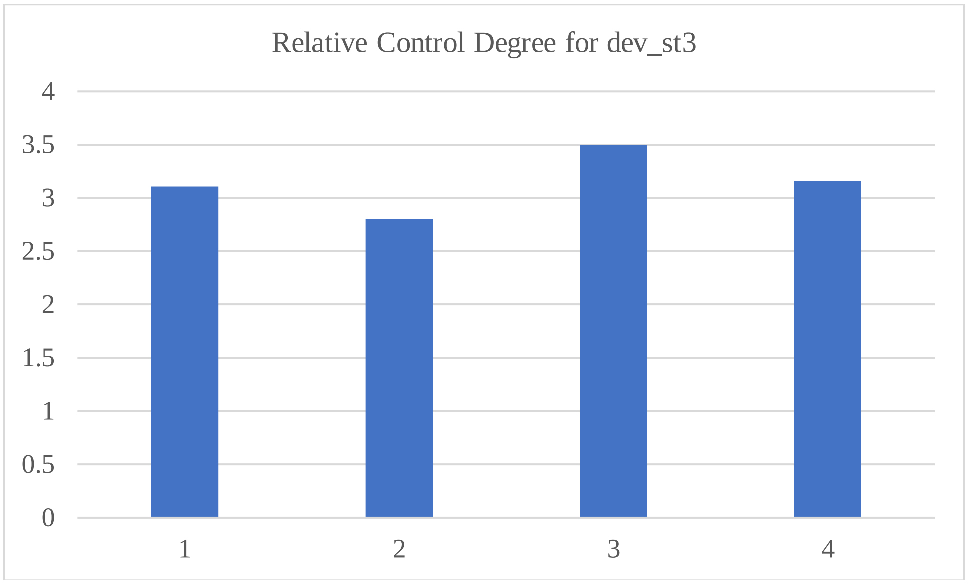


Figure 13. Relative Control Degree for dev_st3

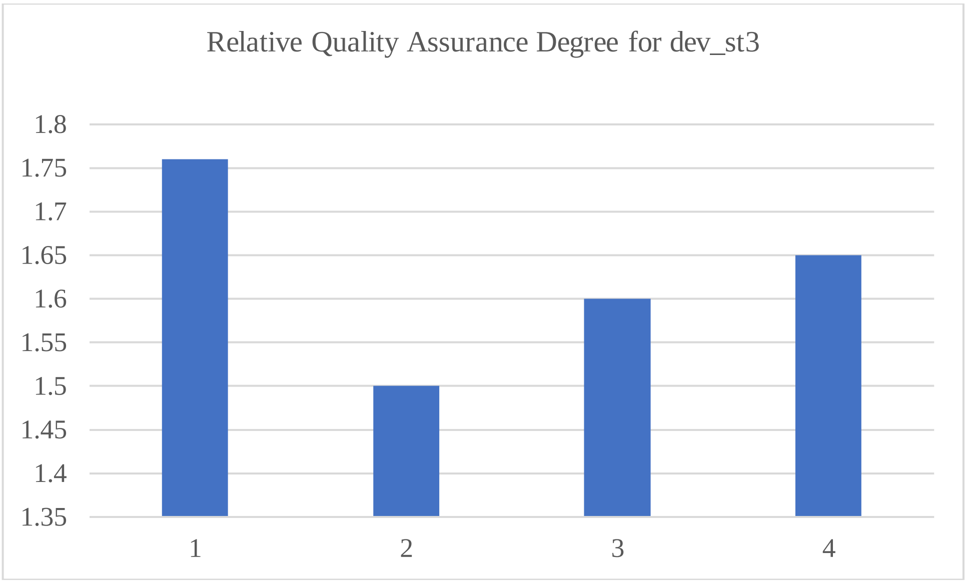


Figure 14. Relative Quality Assurance Degree for dev_st3

In order to determine the consistency of experts' opinions on each question, we constructed tables of pairwise comparisons of experts' responses according to the Fundamental Scale of Absolute Numbers by T. L. Saaty [17]. Table 5 contains pairwise comparisons of experts' answers to Question 1.

Based on the constructed tables of pairwise comparisons of the experts' answers to the questionnaire, Consistency Index (CI), (formula (7)), Consistency Ratio (CR) formula (8)) were calculated for each question, and the results are presented in Table 5. Here is an example of calculating CI and CR for Question 1 (Q1), formula (9) and formula (10).

$$CI\_Q1 = \frac{\lambda_{max} - n}{n-1} = \frac{11.05015 - 11}{11-1} = 0{,}00501. \tag{9}$$

RI is determined by the table, in our case n=11, so for this value RI=1.52 [17].

$$CR = \frac{CI}{RI} = \frac{0{,}00501}{1.52} = 0{,}00330. \tag{10}$$

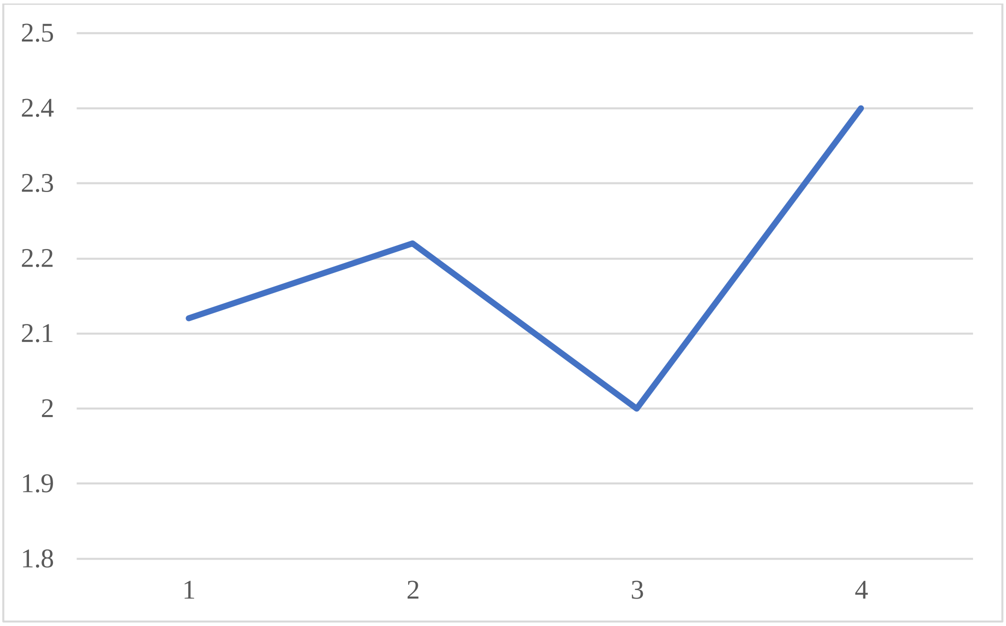


Figure 15. Time Ratio for dev_st3

Table 4

Results of the experts' answers to the questionnaire

Role	Question 1	Question 2	Question 3	Question 4	Question 5
QA 1	9	9	10	9	9
QA 2	10	9	10	9	8
TL 1	9	8	10	9	10
P 1	10	8	10	9	10
P 2	10	9	9	8	9
P 3	9	9	10	9	10
SD 1	7	7	9	8	4
SD 2	9	9	9	9	7
SD 3	8	8	10	9	8
SD 4	9	8	10	9	10
SD 5	10	9	10	9	9

As can be seen in Table 6, the CI values for all questions do not exceed 0.2, and the CR values for all questions do not exceed 0.1. Therefore, we can conclude that the experts' opinions are consistent.

Let's calculate how many points the experts gave to the answers to questions 1-5 of the questionnaire regarding the applicability of the metrics we have developed, taking into account the weights we have introduced in accordance with the knowledge of each expert in industrial project development. The results are presented in Table 7 and Figure 16.

As can be seen from Figure 16, the highest scores were given to Question 4 and Question 3 of the survey, the lowest scores were given to Question 5 and Question 1, and the answer to Question 2 has an average score. The maximum number of points was given to Question 4, and the minimum - to Question 5.

Table 5

Table of pairwise comparisons of the experts' answers to Question 1 of the survey

Role	**Question 1**	**Question 2**	**Question 3**	**Question 4**	**Question 5**
QA 1	1,00	0,50	1,00	0,50	0,50
QA 2	2,00	1,00	2,00	1,00	1,00
TL 1	1,00	0,50	1,00	0,50	0,50
P 1	2,00	1,00	2,00	1,00	1,00
P 2	2,00	1,00	2,00	1,00	1,00
P 3	1,00	0,50	1,00	0,50	0,50
SD 1	0,33	0,25	0,33	0,25	0,25
SD 2	1,00	0,50	1,00	0,50	0,50
SD 3	0,50	0,33	0,50	0,33	0,33
SD 4	1,00	0,50	1,00	0,50	0,50
SD 5	2,00	1,00	2,00	1,00	1,00

Table 6

Consistency Index (CI) and Consistency Ratio (CR) values for each question of the survey

Indexes	**Question 1**	**Question 2**	**Question 3**	**Question 4**	**Question 5**
Consistency Index	0,00501	0,00165	0	0,009771	0,139233
Consistency Ratio	0,00330	0,00109	0	0,006428	0,091601

Table 7

Table of pairwise comparisons of the experts' answers to Question 1 of the survey

Role	**Question 1**	**Question 2**	**Question 3**	**Question 4**	**Question 5**
QA 1	9	27	10	27	9
QA 2	10	27	10	27	8
TL 1	36	16	40	18	40
P 1	20	16	20	18	20
P 2	20	18	18	16	18
P 3	18	18	20	18	20
SD 1	7	7	9	8	4
SD 2	9	9	9	9	7
SD 3	8	8	10	9	8
SD 4	9	8	10	9	10
SD 5	10	9	10	9	9
Total points	**156**	**163**	**166**	**168**	**153**

Such a result for Question 4 and Question 3 of the survey indicates that this metric, in the opinion of experts, takes into account all the components for the average values of the Quality Assurance Degree (Absolute/Relative) for a particular student acting as a Developer, and provides an opportunity to assess the level of involvement of the QA Engineer in planning Epic of standard/typical and non-standard/custom functionality.

The lowest number of points was given by the experts to Question 1, because, in their opinion, the assessment of Team Leader's involvement in planning Epic standard functionality (typical/standard functionality/task) is a multi-criteria task and for its optimal assessment, in addition to taking into account the average values of Control Degree (Absolute/Relative) for a particular student acting as a Developer, additional factors of influence need to be taken into account.

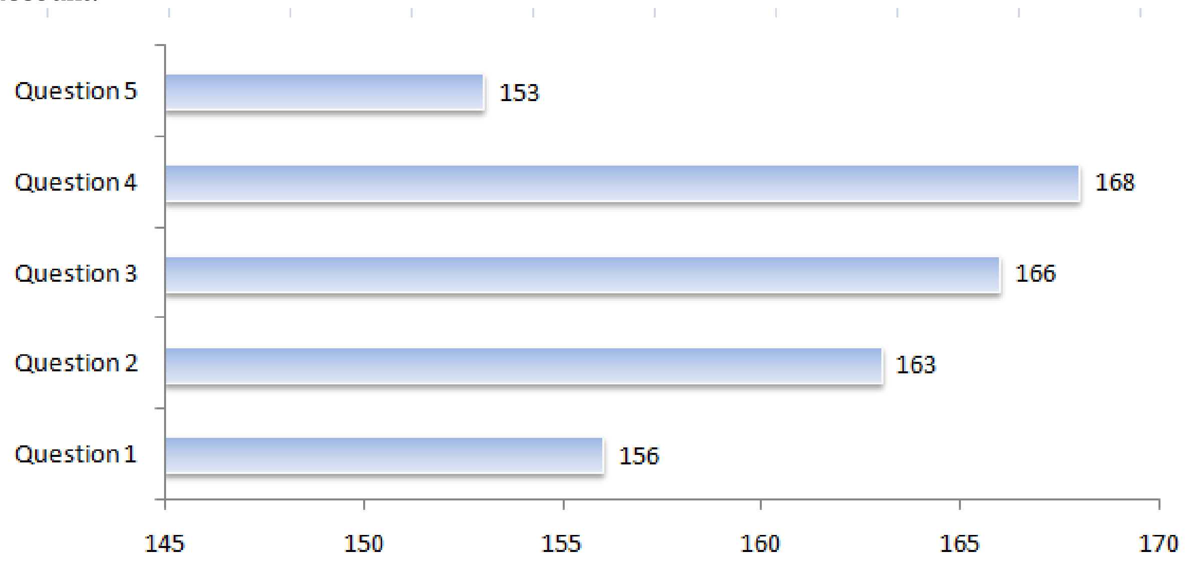


Figure 16. Results of the survey of experts on each question about the use of metrics

Question 5 was given the lowest score by the experts because, in their opinion, many factors need to be taken into account to interpret the results of metrics calculations, for example, to determine the degree of involvement of each participant in the work on project tasks, but each project differs in the complexity of individual tasks and the time spent on their implementation and correction of errors, taking into account the individual approach to each participant and his or her position on the project.

The following reasonable recommendations were given to question 6:

1. The Control Degree metric does not consider the number of pull-request/code review iterations. This can be important because the Team Leader spends time changing the context to switch from another task.
2. The Quality Assurance Degree metric does not consider the number of bugfix/testing iterations. This can be important because the QA Engineer spends time changing the context to switch from another task.
3. There is no consideration of the specifics of a particular task, the values of metrics for which can be very different from the average values.

Let us comment on these remarks:

1 & 2. This can also be taken into account in the proposed approach by including switch context time in each work log for Team Leader and QA Engineer, respectively. An alternative is to modify the formulas for calculating the Control Degree and Quality Assurance Degree of the switch context time indicator.

3. One of the solutions to this problem is task labeling. Each task needs to be assigned a label that characterizes its correspondence to the type of problem. For example, frontend, backend, optimization, configuration, heuristic. In this case, it is appropriate to calculate the values of metrics per label. Note that one task can contain several labels.

4. CONCLUSIONS AND PROSPECTS FOR FURTHER RESEARCH

This study focused on the monitoring and analysis of students' performance throughout their involvement in a software development project. The main contribution of this research is in the development of novel performance metrics that capture both the level of control required from Team Leader and the quality of work assessed by the QA Engineer. These metrics provide a comprehensive framework for evaluating and understanding students' performance within the project context.

The main developed metrics are Relative Control Degree and Relative Quality Assurance Degree and can be defined on project/company/industry level.

The developed "Cluster Analytical and Monitoring Model of Software Development" reveals an expanded understanding of the monitoring issues. In this model, Absolute Control Degree, Relative Control Degree, Absolute Quality Assurance Degree and Relative Quality Assurance Degree metrics that integrally reflect the effectiveness of the activity are used to structure and analyze the results. In addition, this monitoring model uses a system of clusters (blocks), in each of which several aspects are integrated: goal-result, process-result, time-result, innovation (novelty)-result, typical tasks-result, errors-time, communication (teamwork)-result, compliance-result, result-efficiency. These clusters (blocks) of activities in the monitoring system are considered as relevant indicators. The development of these clusters, in addition to monitoring, is aimed at unlocking the professional potential of a specialist. This is done by actualizing his/her resources through reflection, communication, goal setting, motivation, intellectualization, psychologization, responsibility, and disclosure of subjective temporality. This approach is based on the ideas of humanization, intellectualization, and professionalization within the semantic framework of the competence paradigm as the one that actualizes the potential of the individual through professional growth.

The authors of the study have provided recommendations on how to interpret the results and obtained values of the proposed performance metrics.

Besides, an application was created, which calculates defined performance metrics and gives the possibility to monitor performance of students acting as Developers in real time environment.

The utilization of these performance metrics holds implications for project management and student development. By striking a balance between control and autonomy, Team Leaders can foster an environment that encourages student growth, self-direction, and accountability. Moreover, the integration of quality assessment promotes continuous learning and provides students with valuable insights into their strengths and areas for development.

In the process of evaluating the effectiveness of the metrics, applying a qualitative modeling technique, we used Consistency Index and Consistency Ratio to assess the consistency of experts' opinions. The results of the study showed that the scores given by the experts are consistent. In general, the applicability of these metrics is positive and above average. The experts were unanimous in their opinion that the best metrics for determining the involvement of the Team Leader and QA Engineer in planning Epic for a particular student acting as a Developer are those that determine the average values of the Quality Assurance Degree (absolute/relative) of the advanced/custom functionality/task and standard functionality/task.

The detailing of task labelling and methodology of calculating the values of metrics per label is the direction of further research.

REFERENCES (TRANSLATED AND TRANSLITERATED)

[1] B. Fernandez-Gauna, N. Rojoand M. Graña. "Automatic feedback and assessment of team-coding assignments in a DevOps context", *International Journal of Educational Technology in Higher Education,* 20(1), 2023. doi: https://doi.org/10.1186/s41239-023-00386-6. (in English)

[2] M. Wijga, M. D. Endedijk, and B. P. Veldkamp, "A social regulation perspective on team reflexivity: The development of an analytical framework," *Vocations and Learning*, vol. 16, no. 2, pp. 251–291, 2023. doi: https://doi.org/10.1007/s12186-023-09315-0. (in English)

[3] M. J. Hernández-Molinos, A. J. Sánchez-García, R. E. Barrientos-Martínez, J. C. Pérez-Arriaga, and J. O. Ocharán-Hernández, “Software defect prediction with Bayesian approaches,” *Mathematics*, vol. 11, no. 11, p. 2524, 2023. doi: https://doi.org/10.3390/math11112524. (in English)
[4] N. Salleh, B. I. Ya’u, and A. Nordin, “Towards understanding the influence of personality and team behaviors on requirements engineering activities,” *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 13, no. 3, p. 3244, 2023. doi: https://doi.org/10.11591/ijece.v13i3.pp3244-3254. (in English)
[5] O. V. Klochko, V. M. Nagayev, V. I. Klochko, M. G. Pradivliannyi, and L. I. Didukh, “Computer oriented systems as a means of empowerment approach implementation to training managers in the economic sphere”, *ITLT*, vol. 68, no. 6, pp. 33–46, Dec. 2018. doi: https://doi.org/10.33407/itlt.v68i6.2484. (in English)
[6] A. Kolomiiets, V. Klochko, O. Stakhova, O. Klochko, V. Petruk, and . M. Kovalchuk, “Improving the Level of Cognitive Component of Mathematical Competence in the Process of Mathematical Training of Students of Technical Specialties”, *RREM*, vol. 15, no. 1, pp. 261-284, Mar. 2023. doi: https://doi.org/10.18662/rrem/15.1/696. (in English)
[7] Y. Jin, Y. Bai, Y. Zhu, Y. Sun, and W. Wang, “Code recommendation for Open Source Software Developers,” *Proceedings of the ACM Web Conference* arXiv:2210.08332, 2023. doi: https://doi.org/10.1145/3543507.3583503. (in English)
[8] A. B. Bondi and L. Xiao, “Early progress on enhancing existing software engineering courses to cultivate performance awareness,” *Companion of the 2023 ACM/SPEC International Conference on Performance Engineering,* 2023. doi: https://doi.org/10.1145/3578245.3584352. (in English)
[9] T. Fatma, N. A. Khan and S. Sarwar, “Software Defect Estimation using Data Mining Techniques: Experimental Study of Algorithms on “PROMISE” Repository,” *2023 10th International Conference on Computing for Sustainable Global Development (INDIACom),* New Delhi, India, 2023, pp. 1580-1585. (in English)
[10] G. Mahajan and N. Chaudhary, “Design and development of novel hybrid optimization-based convolutional neural network for software bug localization,” *Soft Computing*, vol. 26, no. 24, pp. 13651–13672, 2022. doi: https://doi.org/10.1007/s00500-022-07341-z. (in English)
[11] S. Shafiq, A. Mashkoor, C. Mayr-Dorn, and A. Egyed, “Taskallocator: A recommendation approach for role-based tasks allocation in agile software development,” *2021 IEEE/ACM Joint 15th International Conference on Software and System Processes (ICSSP) and 16th ACM/IEEE International Conference on Global Software Engineering (ICGSE),* 2021. doi: https://doi.org/10.1109/icssp-icgse52873.2021.00014. (in English)
[12] T. N. Amarasekara et al., “Project zone : An Advanced Undergraduate Project Management System for Software Development,” *2021 21st International Conference on Advances in ICT for Emerging Regions (ICter),* 2021. doi: https://doi.org/10.1109/icter53630.2021.9774820. (in English)
[13] GlobalLogic, “Which software metrics to choose, and why?,” [Online]. Available: https://www.globallogic.com/insights/blogs/which-software-metrics-to-choose-and-why/ Accessed: Jan. 23, 2024. (in English)
[14] Atlassian, “Epics, stories, themes, and initiatives,” *Atlassian*, [Online]. Available: https://www.atlassian.com/agile/project-management/epics-stories-themes/ Accessed: Jan. 23, 2024. (in English)
[15] Nuget, “Atlassian.SDK 13.0.0,” NuGet Gallery. Atlassian.SDK 13.0.0, [Online]. Available: https://www.nuget.org/packages/Atlassian.SDK (accessed Jan. 23, 2024). (in English)
[16] Miratech, “30 years in software development, it consulting, and CX.,” [Online]. Available: https://miratechgroup.com/ Accessed: Jan. 23, 2024. (in English)
[17] K. Kułakowski, *Understanding the analytic hierarchy process*. Chapman and Hall/CRC 2020. doi: https://doi.org/10.1201/9781315392226. (in English)
[18] Gemicle, “IT-consulting and technologies,” [Online]. Available: https://www.gemicle.com/?lang=en Accessed: Jan. 23, 2024. (in English)

Text of the article was accepted by Editorial Team 26.01.2024

МОНІТОРИНГ ТА АНАЛІЗ ПРОДУКТИВНОСТІ ДІЯЛЬНОСТІ СТУДЕНТІВ ПІД ЧАС РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Олександр Шаригін
кандидат технічних наук,
старший викладач кафедри інформаційних технологій та програмування,

Український державний університет імені Михайла Драгоманова, м.Київ, Україна
керівник проєкту,
Міратех, м.Київ, Україна
ORCID ID 0009-0006-9405-6997
exhaustic@gmail.com

Василь Федорець
кандидат медичних наук, доцент,
доцент кафедри педагогіки, адміністрування та спеціальної освіти,
Державний заклад вищої освіти ‘‘Університет менеджменту освіти”
Національної академії педагогічних наук України, м.Київ, Україна,
доцент кафедри педагогічних наук, початкової та корекційної освіти,
Комунальний заклад вищої освіти ‘‘Вінницька академія безперервної освіти”, м.Вінниця, Україна
ORCID ID 0000-0001-9936-3458
bruney333@yahoo.com

Оксана Клочко
доктор педагогічних наук, професор
професорка кафедри математики та інформатики,
Вінницький державний педагогічний університет імені Михайла Коцюбинського, м.Вінниця, Україна
ORCID ID 0000-0002-6505-9455
klochkoob@gmail.com

Анотація. У дослідженні представлено комплексний підхід до моніторингу та аналізу продуктивності діяльності студентів у контексті проєктів розробки програмного забезпечення. Основна увага зосереджена на розробці нових показників продуктивності діяльності, за допомогою яких визначаються показник рівня контролю, що здійснюється керівником групи, та показник якості роботи, яку оцінює інженер із забезпечення якості продукту. Представлені показники, а саме: відносний ступінь контролю та відносний ступінь забезпечення якості, – є надійною основою для оцінювання та розуміння успішності студентів на рівні проєкту, компанії та галузі.

У межах дослідження розроблено програмне забезпечення, за допомогою якого розраховуються показники продуктивності, що надає можливість у режимі реального часу відстежувати продуктивність діяльності студентів як розробників. Цей практичний інструмент є засобом для моніторингу та оцінки прогресу студента протягом життєвого циклу проєкту, забезпечуючи своєчасний зворотний зв’язок та втручання за потреби.

Автори надають рекомендації щодо інтерпретації результатів і використання запропонованих показників ефективності: ступінь контролю; ступінь забезпечення якості; відносний ступінь контролю; відносний ступінь забезпечення якості. Використання розроблених показників ефективності є важливим для успішного управління проєктами та розвитку фахових компетентностей студентів. Впроваджуючи рекомендовані практики, заклади освіти та професіонали галузі можуть підвищити ефективність проєктів розробки програмного забезпечення, покращити результати навчання студентів і сприяти культурі постійного вдосконалення та інновацій.

Для вирішення завдання вдосконалення моніторингу та аналізу прогресу студентів у процесі участі в проєктах розробки програмного забезпечення використовувались такі методи, моделі та підходи: системний та проблемний підходи, кластерні аналітичні та моніторингові моделі розробки програмного забезпечення, експертне опитування, індекс узгодженості, коефіцієнт узгодженості та фундаментальна шкала абсолютних чисел Т. Л. Сааті та ін.

Запропоновані показники продуктивності діяльності студентів в процесі участі в проєкті розробки програмного забезпечення, їх опис та розроблене програмне забезпечення для їх розрахунку нададуть можливості зацікавленим сторонам проєкту оптимізувати контроль, забезпечення якості роботи над проєктом та розвиток фахових компетентностей студентів.

Ключові слова: проєктне навчання; показники продуктивності діяльності студентів; проєкт розробки програмного забезпечення; цифрове та математичне моделювання; моніторингова модель розробки програмного забезпечення; кластерна модель.