Computer Science and Information Systems 23(3):1053-1076 https://doi.org/10.2298/CSIS251109030P

Role of Software Metrics in Practical Quality Management

Filip Prentović[1], Zoran Budimac[1], Marjan Heričko[2], and Gordana Rakić[1]

[1] University of Novi Sad, Faculty of Sciences, Department of Mathematics and Informatics
Trg Dositeja Obradovica 4, Novi Sad 21000, Serbia
filipprentovic@yahoo.com (corresponding author),
zjb@dmi.uns.ac.rs, gordaa.rakic@dmi.us.ac.rs
[2] University of Maribor, Faculty of Electrical Engineering and Computer Science,
Smetanova 17, 2000 Maribor, Slovenia
marjan.hericko@um.si

Abstract. The goal of software metrics is to provide continuous insight into products and processes. Software product and software development process are tightly coupled since high-quality software products represent the result of a high-quality process. To understand, predict, and evaluate software development and maintenance process, it is widely recognized that software metrics, as an important technique in the static and dynamic analysis, should be integral part of these processes. Software metrics are a critical tool for building reliable software and assessing software quality, especially in complex domains and/or mission-critical environments. The aim of the research conducted in this paper is to determine the existence of different perspectives regarding the impact of software metrics on software quality between participants in software development. The contribution of this paper consists of analyzing the impact that the application of software metrics has on the quality of software and on the overall project success by statistically analyzing data obtained from the survey. Overall, there were 108 respondents from ten countries in the survey. The paper contains a comprehensive survey of code analysis techniques, software metrics, and the analysis of the application of software metrics in practice. Also, importance, influence, and the level of usage of software metrics in IT companies have been considered, as well as the estimation of results significance obtained in the process of measuring software.

Keywords: Code Analysis, Software Metrics, Software Quality.

1. Introduction

Software quality is a term that has different meanings to different people involved in creating, maintaining, and using software. Its presence can be difficult to define, but its absence can be easy to see instantly. There are many different definitions of quality. According to some definitions, it is the "capability of a software product to conform to requirements" [1], while for others it can be synonymous with "customer value" [2] or even defect level.

The main quality attribute of each software product is correctness. Without correctness, it is pointless to talk about other quality attributes such as functionality, usability, reliability, efficiency, portability, compatibility, security, and maintainability [3].

These quality attributes can be analyzed, controlled, and monitored at the early stages of software development by examining source code and other artifacts (software architecture, system specification, etc.), or later during the execution of the system. These activities should ideally, be performed by all the parties involved in the process of software development: developers, testers, QA (quality assurance) analysts, and managers. Those activities are performed statically and dynamically, depending on the phase and state of the project. Evaluation of software quality attributes that is performed on a source code or any of its internal representations (like syntax trees, graphs and various meta-models which represent abstract syntactic structure of source code written in a programming language) without executing the program belongs to a domain of *static analysis*, while analysis of a software product during its execution represents *dynamic analysis*. Both kinds of code analysis rely on various metrics, with growing importance of static analysis and software metrics since a lot of emphasis is being put on monitoring and quality control in the early stages of the development [3].

The goal of software metrics is to provide continuous insight into products and processes. Software products and their development process are tightly coupled since high-quality software products represent the result of a high-quality process. To understand, predict, and evaluate software development and maintenance process, it is widely recognized that software metrics, as an important tool in static and dynamic analysis, should be an integral part of these processes. Software metrics are a critical tool for building reliable software and assessing software quality, especially in complex domains and/or mission-critical environments. A useful metric typically performs a calculation to assess the effectiveness of the underlying software or process [4]. An increasing need for measurement data used in decision-making on all levels of the organization has been observed when implementing the software process. Although the benefits of measurement in the software process are indisputable, the popularity of the use of measurement in practice is rather limited [5]. It is expected that the measurement provides reliable and precise information, but at the same time to be cost-effective and unassuming during the software development process.

Measuring software quality is motivated by at least two reasons [6]:

- Risk management – software errors are not just a nuisance – they can cause serious damage, especially in the embedded systems used in medical devices, airplanes etc.
- Cost management – recent report revealed that in 2022, software failures cost the economy US$2.41 trillion in financial losses [7]. If the cost of fixing a requirements error, discovered during the requirements phase, is defined to be one unit, the cost to fix that error if found during the design phase increases to three to eight units; at the manufacturing/build phase, the cost to fix the error is 7–16 units; at the integration and test phase, the cost to fix the error becomes 21–78 units; and at the operations phase, the cost to fix the requirements error ranged from 29 units to more than 1500 units [8].

This paper investigates whether different stakeholders involved in software development hold differing perspectives on the impact of software metrics on software quality. The study focuses on examining how the use of software metrics influences both software quality and overall project success. To achieve this, a statistical analysis was conducted on data collected through a survey involving 108 respondents from ten different countries. In addition, the paper provides a comprehensive overview of code analysis techniques and

commonly used software metrics, along with an examination of how these metrics are applied in real-world practice. Particular attention is given to the perceived importance, influence, and level of adoption of software metrics within IT companies. The research also includes an assessment of the statistical significance of the results obtained through software measurement. Finally, the findings of this study are compared with results from similar surveys conducted in previous years in order to identify trends and changes over time.

This paper is organized as follows. Section Related Work contains references to the similar surveys done before, as well software metrics reviews. Definitions and standards of software quality, overview of code analysis techniques, and relation between software metrics and software quality is presented in the next section. Section Survey and Results describes survey, and its results and analysis. Survey results are compared to results of the previously conducted surveys and discussed and analyzed in section Discussion and analysis. Section Conclusion contains concluding remarks.

2. Related Work

Since this paper contains comprehensive review of software metrics, the role of software metrics in measuring software quality, and analysis of survey of software metrics usage, this section contains an overview of the papers reviewing existing software metrics, and papers in which surveys on software metrics usage were conducted. Finally, the section highlights the novelty of this study in comparison to all referred ones.

Study conducted by Setiadi [9] investigates the measurement of software quality in the Academic Information System at Marshal Suryadarma Aerospace University. The research considers both direct metrics, such as cost, code size, execution speed, and documentation, and indirect metrics related to functionality, efficiency, dependability, and maintainability, transforming these metrics into indicators for evaluating software quality. The study aims to assist university administration in monitoring and improving their information systems, while providing a comprehensive understanding of how metrics and indicators can be applied to assess and enhance the quality of academic software.

Dynamic metrics, which are usually obtained from the execution traces of the code or from the executable models are analyzed by Chabbra and Gupta [10]. In this paper advantages of dynamic metrics over static metrics are discussed and then a survey of existing dynamic metrics is carried out. These metrics are grouped to different categories, such as dynamic coupling metrics and dynamic cohesion metrics. Towards end of the paper, potential research challenges and opportunities in the field of dynamic metrics are identified.

Srinivisan et al. [11] analyze and review the most referred object-oriented metrics in software measurement. The paper presents formal definitions of most important categories of object-oriented metrics and their characteristics, while proposing suite for measuring object-oriented design attributes.

In his research, Lee [12] builds an appropriate method of software quality metrics application in quality life cycle with software quality assurance. Successful software quality assurance is highly dependent on software metrics, and it needs a link between the software quality model and software metrics. This link is obtained by using quality factors to offer measure method for software quality assurance. The paper defines some software

metrics and discusses several software quality assurance models and some quality factors measuring methods.

Pavlič et al. [13] conducted the survey to determine how and to what extent the quality of software is measured in Slovenia. Results of the survey were compared to the similar surveys previously taken and the conclusion was that they do not differ significantly. Larger deviations were only found by combining obtained quality estimate into overall quality estimate, which is a more common practice elsewhere than in Slovenia.

Study conducted by Paulinus [14] looks at the role of transparency as a non-functional requirement in the software engineering process, addressing the lack of appropriate measurement methods and empirically validating metrics for evaluating and improving transparency throughout the software development lifecycle. The study investigates correlations between transparency factors and metrics, and proposes a Goal Question Metric (GQM)-based transparency evaluation and improvement model. Results from a controlled experiment demonstrate that enhanced transparency improves maintainability of software requirements specifications, supports better communication, and increases stakeholder productivity, offering practical guidance for early phases of requirements engineering and design.

In the study taken by Tahir et al. [15], a model of 18 success factors for implementing measurement processes is proposed, while a survey is conducted to evaluate the state of measurement practices and to validate the proposed model based on the feedback from 200 software professionals working in Pakistani software industry. The survey showed that, for example, only 10% software organizations in this survey follow any measurement model, that 75% organizations do not follow any measurement standard, and that there are 80% software organizations in Pakistan which do not use any measurement tool. The paper presented mitigation strategies for key issues identified in software organizations.

In his research, Kasunic [16] conducted a comprehensive survey consisting of 17 questions with the help of random sampling, with individuals from 84 countries responded. The aim of the research was to find out: a) what measurement definition and implementation approaches are being adopted and used by the community, b) the most prevalent types of measures being used by organizations that develop or acquire software, and c) what behaviors are preventing the effective use of measurement. Among other things, it was observed that administration and staff have different opinions regarding measurement process, and that the larger organizations had better measurement infrastructure.

The goal of the research taken by Chen et al. [17] was to investigate how the use of large language models (LLMs) can distort traditional size-based software metrics, particularly lines-of-code (LOC), and to assess the practical applicability of simple versus complex efficiency metrics across multiple software repositories. Additionally, the research aimed to evaluate the effectiveness and cost-efficiency of static analysis tools such as SonarQube and PMD in detecting commits that introduce significant changes to software quality. Finally, the goal was to validate these findings in an industrial context through a case study in a large financial-sector organization, which included a rapid review of existing practices and the development of a software metrics migration plan.

Eisty and colleagues [4] conducted a survey to gather information about research software developers' knowledge and use of code metrics and software process metrics. The survey results, from 129 respondents, indicated that respondents have a general knowledge of metrics. However, their knowledge of specific metrics was lacking and their use even

more limited. Software developers appear to be interested and see some value in software metrics but may be encountering roadblocks when trying to use them.

The research of Atanasova and Temole [18] is addressing the persistent challenge of implementing software quality models in complex organizational environments, such as large companies undergoing agile transformation. It proposes a pragmatic, value chain-oriented Quality Gate Framework that operationalizes software quality by integrating measurable quality indicators across both product and process dimensions. Empirically grounded through 58 expert interviews, the framework improves transparency, stakeholder alignment, and quality assurance while remaining compatible with agile and DevOps practices, offering a transferable approach for organizations seeking sustainable improvements in software development outcomes.

Survey conducted by Padmini et al. [19] is characteristic because it is only concerned with the use of metrics in the agile development process. Aim of the research was to explore metrics suitable for the agile development process, use of those metrics in practice, perceived benefits, and related tools. The survey and interview-based analysis of 24 development companies in Sri Lanka identified ten metrics that can be beneficial to the agile development process, where their benefits outweigh the overheads involved.

Schaffernak et al. [20] tried to track and measure significant software quality attributes to quantify the current state of a system and support strategic and technical decisions. They analysed existing measures related to the ISO/IEC 25022:2016 quality in use (QiU) model to evaluate sustainability aspects and conducted a systematic literature review of 961 articles, followed by internal evaluations of the identified measures. Their results identified 100 measures covering economic, social, technical, and environmental sustainability, showing a strong focus on economic sustainability and underrepresentation of other dimensions, providing guidance for enhancing the QiU model in a more systematic and sustainable way.

Colakoglu et al. [21] review the present studies in the area of software product quality metrics (SPQM) to allow for the analysis of the situation at hand, as well as to predicting future research areas. Paper presents research aims to analyze the active research areas and trends on this topic appearing in the literature between 2009 and 2019. A Systematic Mapping (SM) study was carried out on 70 articles and conference papers published between 2009 and 2019 on SPQM as indicated in their titles and abstract. The result is presented through graphics, explanations, and the mind mapping method, as well as trend map, knowledge about this area and measurement tools, issues determined to be open to development in this area, and conformity between conference papers, articles and internationally valid quality models.

Research conducted by Molnar and colleagues [22] is proposing a comprehensive study of software metric values in the context of three complex, open-source applications. Their methodology and tooling is aligned with that of existing research, and presented in detail in order to facilitate comparative evaluation. Metric values obtained during the entire 18-year development history of the target applications were studied, in order to capture the longitudinal view that was found lacking in the existing literature. Also, metric dependencies were identified and their consistency checked across applications and their versions, along with comparative evaluation with existing research.

The aim of the paper by Mishra et al. [23] is to systematically investigate research on software metric threshold calculation techniques, which were identified as important in

evaluating different aspects of quality. In the study, electronic databases were systematically searched for relevant papers; 45 publications were selected based on inclusion/exclusion criteria, and research questions were answered. Results showed that most of the methods proposed to calculate the thresholds are based on statistical analysis techniques, and that the field of identifying the object-oriented metrics threshold values has received more attention recently.

Research conducted by Vogel at al. [24] studies metrics used in the automotive sector and the quality attributes they address. The HIS, ISO/IEC 25010:2011, and ISO/IEC 26262:2018 are utilized to draw a big picture illustrating (i) which metrics and boundary values are reported in literature, (ii) how the metrics match the standards, (iii) which quality attributes are addressed, and (iv) how the metrics are supported by tools. Most of the identified metrics are concerned with source code, are generic, and not specifically designed for automotive software development. Research concludes that although many metrics exist, a clear definition of the metrics' context, notably regarding the construction of flexible and efficient measurement suites, is missing.

Finally, Rashid and colleagues [25] describe how the software metrics affect the quality of the software and in which stages of its development software metrics have applied. Also, different software metrics are described, along with the impact these metrics have on software quality and reliability, specifically on the aspects of improving the quality of software and increasing the revenue.

The research conducted in this paper differs from previously mentioned papers by aiming to establish a statistical correlation on the impact of using software metrics to software quality and success of software product. Specifically, aim of the survey is to show potentially different perspectives on software metrics usage and understanding between managers and software development team members. Also, it contains a detailed review of software quality, code analysis in general, and software metrics, giving the comprehensive assessment of these topics.

3. Software Quality, Code Analysis, and Software Metrics

The concept of software quality is very broad, spanning through various aspects of software product and its development process. Therefore, it is useful to consider different perspectives on this concept. Measuring software quality is important because it enables creating a baseline for quality, associating specific costs with the improvement of quality, as well as presenting the level of quality on which further improvements can be made [26]. During time, several software metrics were introduced in order to capture various aspects of software, including software quality characteristics.

3.1. Software Quality – Definitions and Standards

Various software quality models have been proposed to define quality. A global initiative for describing the concept of software quality taken by experts around the world led to the standardization of the quality concept by the ISO (the International Organization for Standardization) in the form of documents ISO 9126 [27] and ISO 9000:2000 [28], whereas ISO 9126 is about the quality characteristics of a software product, and the ISO 9000:2000 document is a quality assurance standard mostly directed to processes [26].

As a result of cooperation between ISO and IEC (the International Electrotechnical Commission) in the field of standardization, ISO/IEC 25000 standard, based on ISO 9126, was proposed [28]. ISO/IEC 25000 is a series of standards under the general title Systems and Software Quality Requirements and Evaluation (SQuaRE). New standardization uses new definition of software quality and related terms, defining them as an expression of a degree to which a software product satisfies stated and implied needs when used under specified conditions.

As per ISO 9126 standard, quality is defined as "the totality of features and characteristics of a product or service that bears on its ability to satisfy given needs" [1]. This definition refers to the level of compliance of requirements.

Software quality has also been defined in terms of two types of product characteristics [29]:

- external quality (how the product works in its environment), such as, usability and reliability.
- internal quality (how the product was developed), such as, software structure and complexity.

In other words, software quality refers to two related but distinct notions [30]:

- software functional quality reflects how well it complies with or conforms to a given design, based on functional requirements or specifications. That attribute can also be described as the fitness for purpose of a piece of software or how it compares to competitors in the marketplace as a worthwhile product. It is the degree to which the correct software was produced.
- software structural quality refers to how it meets non-functional requirements that support the delivery of the functional requirements, such as robustness or maintainability. It has a lot more to do with the degree to which the software works as needed.

In software, the narrowest sense of product quality is commonly recognized as the lack of "bugs" in the product. It is also the most basic meaning of conformance to requirements, because if the software contains too many functional defects, the basic requirement of providing the desired function is not met. This definition is usually expressed in two ways: defect rate (e.g., number of defects per million lines of source code, per function point, or other unit) and reliability (e.g., number of failures per n hours of operation, mean time to failure, or the probability of failure-free operation in a specified time) [31].

3.2. Software Quality Analysis

Monitoring and controlling software quality are key processes in software development lifecycle, so it is important to assess software quality from the early stages of software development. It is achieved by analyzing the source code and other artifacts throughout the entire life cycle of the observed software. Assessment of quality attributes of software that is made on the source code or any of its internal representations without executing the program is called static analysis, while analysis of the program during execution time is called dynamic analysis. Numerical values reflecting a software quality characteristic or some of its aspects, and that are obtained by software analysis, are known as software metrics [3].

Static analysis can be defined as a set of techniques for program analysis that do not require software to execute to obtain needed information. Various intermediate representations of a source code, like syntax trees and graphs are being used for the purpose of analyzing code in this manner. Since it is crucial to obtain information on software quality in the early phases of software development, static analysis becomes more important in modern software development methodologies.

Static analysis plays an important role in software evolution and during maintenance phase of software development process. The term *evolution* generally refers to progressive change in software properties or characteristics. This process of change in one or more of their attributes leads to the emergence of new properties or, in some sense, to improvements. These changes affect both functionality and quality of the software product. The related concepts also include software reengineering and software maintenance. *Software reengineering* implies that new user-required features are added to existing software. *Software maintenance* is defined as the modification of a software product after delivery to correct faults, to improve performance or other attributes, or to adapt the product to a changed environment [32]. These changes usually do not affect functionality of a program, but they can have an influence on its quality. Since software quality can be compromised during software evolution and maintenance, all these changes need to be closely controlled and monitored.

Most of the static analysis techniques produce quantitative data along the way or are focused on expressing some program properties. Some of these techniques rely on other methods and corresponding metrics.

Dynamic program analysis represents analysis of properties of a running program [33]. In general, dynamic analysis involves capturing of dynamic state of the program. It usually comprises the analysis of a system's execution through interpretation (e.g., using the Virtual Machine in Java) or instrumentation. After these activities, the resulting data are used for such purposes as reverse engineering and debugging [34].

Dynamic program analysis provides a means to overcome the shortcomings of static analysis. It is effective in examining the actual, exact runtime behavior of a program, as it leaves no uncertainties about the control flow path taken, the values stored in variables, the status of the memory, the time of the execution of the program, etc. Dynamic analysis is as fast as the program being analyzed, which is usually not possible with static analysis techniques. On the other hand, dynamic analysis incurs large runtime overheads, while results obtained during dynamic analysis cannot be generalized for future executions [35].

3.3. Software Metrics

There are several techniques and methods that can be used in a program analysis. Since the survey taken in this research considers using software metrics in the software development process, and the role of software metrics in assessing software quality, software metrics will be described in detail in this section.

Measuring and providing means for quantitative analysis of software is crucial for the overall success of software development process, and consequently, the product. Software metrics provide a quantitative basis for planning and predicting software development processes [36]. By using software metrics during software development process, the quality of software can be controlled and improved easily. This means better productivity

and maintainability of software product, improving overall process and resulting in more reliable and better software.

Software metrics can be defined as numerical values that reflect the properties of a software development processes and software products [37]. Software metrics have been used for a long time: there were attempts of applying software metrics in the 1960s [38]. In the beginning, software execution time was measured and compared to gain insight in the performance of the running programs. As the complexity of software systems increased, software metrics was introduced to measure complexity and the level of cohesion between different parts of software. Measuring characteristics of software design became more important when it was observed that quality of software design corresponds to the number of errors discovered in the latter stages of software development, hence influencing the maintenance costs.

At the highest level, software metrics can be classified into three categories: product metrics, process metrics, and project metrics [31]. Product metrics describe the characteristics of the product. Some of the product metrics refer to characteristics visible to the users (performance, design features etc.), and they are called external metrics. Product metrics related to the characteristics visible only to the development team (size, complexity, structure etc.) are called internal metrics. Process metrics are related to the characteristics of process and can be used to improve software development and maintenance. Examples include the effectiveness of defect removal during development, the pattern of testing defect arrival, and the response time of the fix process. Project metrics describe the project characteristics and execution. Examples include the number of software developers, the staffing pattern over the life cycle of the software, cost, schedule, and productivity. Some metrics belong to multiple categories. For example, the in-process quality metrics of a project are both process metrics and project metrics.

Software quality metrics are a subset of software metrics that focus on the quality aspects of a product, process, and a project. In general, software quality metrics are more closely associated with process and product metrics than with project metrics. However, the project parameters such as the number of developers and their skill levels, the schedule, the size, and the organization structure certainly affect the quality of the product. Software quality metrics can be divided further into end-product quality metrics and in-process quality metrics. The essence of software quality engineering is to investigate the relationships among in-process metrics, project characteristics, and end-product quality, and, based on the findings, to engineer improvements in both process and product quality [31].

Size of the software product can, in most fundamental form, be represented by the number of lines of code. During the years, a class of Lines of Code (LOC) metrics was proposed, containing several variations of this basic measure of length of code [39] [40]:

- SLOC: Source Line of Code reflecting number of lines containing source code without empty lines and lines containing comments.
- CLOC: Comment Line of Code reflecting number of lines containing comments.
- BLOC: Blank Lines of Code reflecting number of lines containing neither source nor comment.
- PLOC: Physical Line of Code reflecting size as it is without considering programming style or code formatting.
- LLOC: Logical Line of Code reflecting size after applying codding style rules or formatting and logical restructuring of statements over the lines.

- EPLOC: Executable Physical Lines of Code reflecting number of lines of source code minus blank lines and comment lines.
- ELLOC: Executable Logical Lines of Code reflecting number of statements which are executed.

Some of the other metrics that have been proposed are Halstead metrics (H) [41], Cyclomatic complexity (CC) [42], Chidamber and Kemerer [43], Lorenz and Kidd [44], Morris [45], Li and Henry [46] and others.

Practical influence of using software metrics cannot be analyzed without considering programming languages, technologies, and paradigms which are used in software creation. Some of the metrics described previously (such are LOC metrics family) are programming language dependent, i.e., these metrics do not consider programming language specificities, which makes comparing results obtained from calculating metrics in heterogenous systems challenging. Also, most of the frameworks and tools that were created to calculate software metrics are language-specific, although there were some attempts in introducing language-independent frameworks [47] [48].

Since most of the metrics are used predominantly in certain phases of software development, software metrics usage cannot be analyzed without considering underlying software development process model and methodology. Iterative and incremental nature of some of these models (e.g., agile software development) can make use of software metrics that are regularly used in other software development processes challenging. In some cases, these metrics are not even applicable [19]. Also, some of the software metrics are meant to be used only within the certain software development methods [49].

Not all the software metrics are of the equal importance to all the participants in the process of the software creation, which is closely related. Also, software metrics serve different purpose to the interested parties. Managers can use software metrics to identify, prioritize, track, and communicate any issues to promote better team productivity. This enables effective management and allows assessment and prioritization of problems within software development projects. The sooner managers can detect software problems, the troubleshooting process is easier and less expensive. On the other hand, software development teams, consisting of developers, testers, QA analysts etc. can use software metrics to communicate the status of software development projects, pinpoint and address issues, and monitor, improve on, and better manage their workflow.

The aim of the survey conducted during the research described in this paper was to reveal the degree of usage of software metrics in the IT industry. Also, importance and relevance of software metrics in the software development process were described. These features were analyzed considering diversity in programming languages, technologies, methodologies, and roles of the participants in the process of software creation. Finally, the survey tried to find out if there are different perspectives on software metrics usage and understanding between managers and software development team members.

4. Survey and Results

Data were collected using an anonymous online survey administered via Google Forms, containing 26 questions divided into three sections, which can be found in the Appendix.[3]

[3] Google Forms link

The survey link was distributed electronically to various companies, mostly in Serbia, and other countries, through members of SQAMIA initiative (http://sqamia.org/). A non-probability convenience sampling strategy was employed, whereby companies that were accessible to the researchers were invited to participate. Participation was voluntary and respondents were informed that no personally identifiable information would be collected. All responses were recorded anonymously and treated confidentially. In total, 108 respondents from ten countries responded to the survey. The characteristics of the respondents and the companies for which they work are shown in Table I.

Most of the respondents (43%) responded that they have average knowledge of software metrics in general (Fig 1.), and 36% of the respondents had a low and very low level of knowledge. This was further corroborated by the fact that most of the respondents did not specify any software metrics they are familiar with, or they specified only a couple of metrics (Lines of Code metric being the most prevalent among those), while the minority of respondents had a thorough list of metrics in their response.

In most of the projects on which respondents are working, measuring is being performed either continuously or in a certain point in time during coding and testing phases of software development.

The survey employed several Likert-type scales to measure respondents' opinions and experiences. For example, respondents rated frequency (e.g., Never to Always), intensity (e.g., Very Low to Very High), impact (e.g., No impact to Crucial), and agreement (e.g., Strongly disagree to Strongly agree) on these ordinal scales. Additionally, a limited number of open-ended questions allowed for free-text responses to capture qualitative insights.

Most of the time IDE (Integrated Development Environment) or separate metrics or testing tools are being used for capturing metrics, with visualization of results and support for a particular programming language or a technology as a main advantage of a preferred tool.

Table 1. General Characteristics of the Respondents

Characteristic	Freq.	%	Characteristic	Freq.	%
Experience in IT industry			*Technology project is written in*		
0–2 years	11	10.2	Java	82	75.9
2–5 years	17	15.7	JavaScript (or frameworks)	56	51.9
5–10 years	27	25.0	.NET / C#	32	29.6
10–15 years	20	18.5	Python	25	23.1
15+ years	33	30.6	C / C++	21	19.4
			PHP	10	9.3
			Ruby	8	7.4
			Kotlin	8	7.4
			Swift	8	7.4
			Smalltalk	4	3.7
			Haskell	3	2.8
Country of respondent's company			*Software methodology used*		
Serbia	62	57.4	Scrum	72	66.7
Slovenia	25	23.1	Waterfall	10	9.3
Germany	6	5.6	Iterative–incremental	9	8.3
Croatia	3	2.8	Extreme programming	4	3.7
Austria	3	2.8	Lean	3	2.8
Bosnia and Herzegovina	2	1.9	Spiral model	2	1.9
Hungary	2	1.9	Scrum-ban	2	1.9
France	2	1.9	Other	6	5.6
Other	3	2.8			
Company size			*Role of the respondent in project*		
Small (<50 employees)	28	25.9	Developer	60	55.6
Medium (51–1000 employees)	61	56.5	Team leader	33	30.6
Large (1000+ employees)	19	17.6	Solution architect	28	25.9
			Project manager	24	22.2
			Test engineer	27	25.0
			QA analyst	20	18.5
			DevOps	18	16.7
Company type			*Project phase*		
Service-oriented	56	51.9	Active development / released	75	69.4
Product-oriented	31	28.7	Active development / unreleased	13	12.0
Global In-House Center	9	8.3	Planning / requirements gathering	9	8.3
Startup	5	4.6	Initial development / prototyping	4	3.7
Other	7	6.5	Maintenance / no new development	3	2.8
			Other	4	3.7

Almost half of the respondents (47%) stated that measurement is of crucial or great importance to the success of a project/product they are working on (Fig. 2).

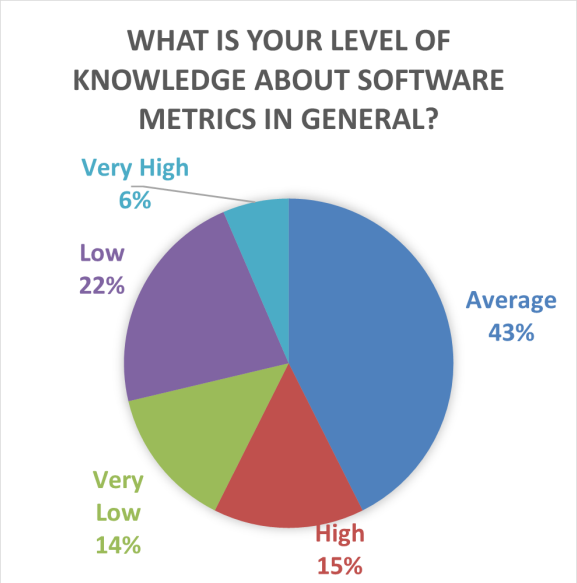

Fig. 1. Most of the respondents have average knowledge about software metrics

Almost the same distribution of answers was obtained from the next question, where respondents were asked about the influence of using software metrics on the overall quality of the software product. Based on the respondents' answers, software metrics are sparingly used to evaluate overall team productivity, with 33% of respondents saying they are sometimes being used for this purpose.

Almost two thirds of the respondents agree, to a greater or lesser extent, that measurement-based data helps their team to perform better than without using it.

Next three questions were questions regarding activities monitored by the software metrics and using and accessing results of the measurements. Most of the respondents think that activities of developers and test engineers are being measured and controlled by managers in various roles, who also have access to measurement results. There were several different answers to two questions about resources needed to establish metrics, as well as resources needed to use metrics, ranging from half an hour to a month. Finally, most of the respondents (33%) were not sure that definitions of metrics used in their projects are consistent and clearly defined (Fig. 3).

5. Discussion and Analysis

The main goal of the survey was to show whether there is a statistically significant correlations between different variables obtained from the research sample. In order to identify independent variables, answers from the following questions were used:

- How many years of experience in IT industry do you have?
- In which country is your company located?

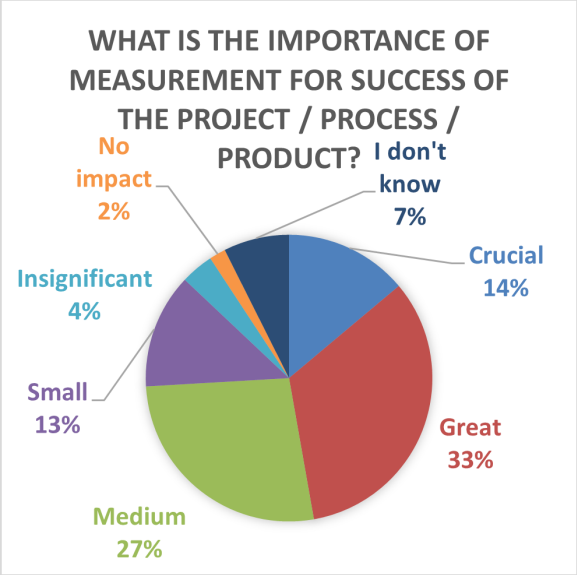

Fig. 2. Measurement has important role in the success of the project

- What is the size of the company you're currently working for?
- What roles do you have in the organizational structure of the project?

Answers to these questions were grouped in the following sample pairs:

- **Experienced and less experienced respondents** – respondents were considered experienced if they had ten or more years of experience in IT industry
- **Company is in Serbia / Company is in other country**
- **Medium size company / Small or large company**
- **Managers and non-managers**

To determine whether someone has managerial role, respondents were designated as the managers if they occupied one of the following roles:

- Senior manager
- Middle manager
- Project manager
- Product owner/Business analyst
- Scrum master
- Team leader

Next, appropriate samples were obtained from the answers given to the following questions:

- What is your level of knowledge about software metrics in general?
- What is the importance of measurement for success of the project / process / product?

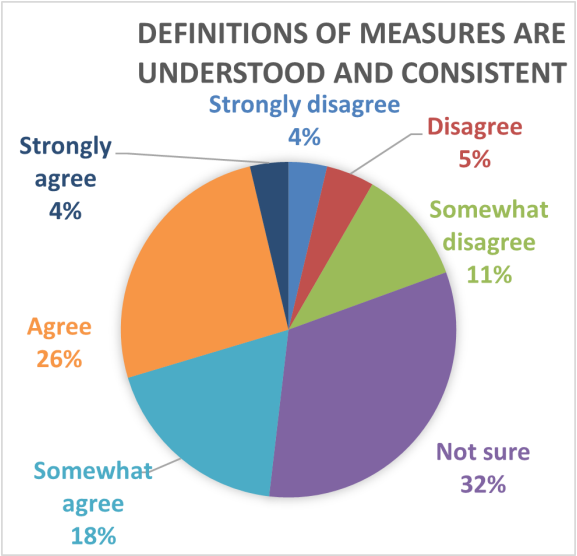

Fig. 3. Definitions of metrics are not always consistent and clearly defined

- How would you describe the influence of using software metrics on the overall quality of the project / process / product?
- How often are software metrics used in the aggregate to evaluate overall team productivity?
- Measurement-based data helps your team to perform better than without using it
- The definitions of measures that are used in my organization/current projects are commonly understood and consistent

To statistically analyze these samples, the weight functions shown in Table 2 were used ('I don't know' and 'Not sure' answers were not taken into account).

Table 2. The Weight functions

Answer	Value	Answer	Value	Answer	Value	Answer	Value
Very Low	1	No impact	1	Never	1	Strongly disagree	1
Low	2	Insignificant	2	Rarely	2	Disagree	2
Average	3	Small	3	Sometimes	3	Somewhat disagree	3
High	4	Medium	4	Most of the Time	4	Somewhat agree	4
Very High	5	Great	5	Always	5	Agree	5
		Crucial	6			Strongly agree	6

Since sample pairs had unequal variances and were different in size, Welch's t-test, a variation of Student's t-test, was used for statistical testing, since it tends to give more accurate results than the latter one when the samples are of unequal size and/or variance [50]. Some of the results of the statistical analysis are shown in Appendix.

By analyzing results obtained from applied test statistic to the sample pairs, the following statistically significant differences appear (with the 99% confidence level, unless stated differently):

- **Managers have higher level of perceived knowledge on software metrics than the regular employees**
- **Respondents in Serbia have lower level of perceived knowledge on software metrics than the respondents from other countries**
- **Software metrics are being used more often to evaluate overall team productivity in Serbia than in other countries** (with the 95 % confidence level)
- **Experienced respondents have higher level of perceived knowledge on software metrics than the less-experienced respondents**
- **Respondents from medium-sized companies think that metrics have stronger influence on team performance than the respondents from small and large companies**
- **Respondents from medium-sized companies think that metrics have stronger influence on project/process/product success than the respondents from small and large companies** (with the 95 % confidence level)

The observed differences in perceived knowledge of software metrics may be attributed to the fact that, in most cases, managers are more experienced, having been promoted to managerial roles after years of working as developers or QA analysts. In addition, the survey indicates that the measurement results are used more frequently by managers, which may contribute to a stronger perception of familiarity with the software metrics.

In general, most of the respondents are seasoned IT professionals with more than five years of experience in the industry, working in medium sized companies, which are, in most cases, service oriented. Also, most of the respondents are developers working in Java, working on projects that are mostly already in production, with Scrum as a software development methodology.

Although most of the respondents reported an average level of knowledge of software metrics, their answers to open-ended questions—particularly the limited or missing examples of concrete metrics—suggest that self-assessed knowledge does not always align with demonstrated familiarity. This discrepancy further supports the need to interpret findings related to expertise as indicators of perceived rather than objectively measured knowledge. In particular, respondents in managerial roles tended to provide more detailed metric lists, which may reflect greater exposure to measurement practices rather than higher technical proficiency.

It is not a surprise (although it is not necessarily the best practice) that the measurements are being performed during coding and testing phases, mostly continuously, by using separate metrics tools, testing tools or IDE. The choice of tools for measurement is usually guided by the platform independence, along with the possibility of the results visualizations, as well as saving and the interpretation of the results.

For most of the respondents, measurements have significant impact on the project success. Similarly, most of the respondents think that software quality is improved if software metrics are used during the project, and that, generally speaking, teams performing better than without using metrics, since productivity of the team is often being evaluated by using some of the metrics.

Unsurprisingly, there is the perception that, in most cases, metrics are being used to assess and evaluate work and the activities of developers and QA and test analysts i.e., persons responsible for results of coding and testing phases, and that the metrics are being used by people in some of the manager roles. Also, most of the respondents were not sure about time needed to establish and run the metrics, with the great variation in time in the available answers. Finally, it seems that there exists uncertainty about the notion of software metrics, since only one third of the respondents think that the definition of software metrics is well understood between all involved in the process of making the software product, which could be influenced by the lack of common knowledge regarding software metrics.

Compared to the results of the similar surveys and research mentioned in section 2, it seems that results of this survey are pretty much alike. The common denominators in the research previously conducted were the lack of knowledge of specific metrics, limited use of software metrics, and the fact that results obtained through the measurement process are mostly used by the upper management. These conclusions are mostly in line with the previously presented results. There are some differences, though – e.g., half of the respondents in the survey conducted by Pavlič et al. [13] responded that metrics are not used to improve the software quality. This is in stark contrast to the answers given in this survey, where vast majority of respondents are of the opinion that metrics have at least medium impact on the quality of the software. In the survey performed by Kasunic [16], most respondents agree with the statement that the definition of measurements is well-understood across the organization, which is not the case with the conclusions drawn in this research.

Some of the differences noted can be attributed to the geographical and cultural differences (surveys were conducted in Slovenia, Sri Lanka, Finland, Pakistan etc.). Also, the samples in the surveys differ greatly when comparing average experience of the respondents, size of the companies respondents are working for, whether the companies are service or project oriented etc. Therefore, it is not always easy to make hard comparisons between different surveys, but it is interesting to observe some templates of thinking and performing, as well as some common practices, producing recurrent benefits and flaws.

From a practical perspective, the identified differences suggest that the use and understanding of software metrics within organizations are strongly influenced by role, experience, and organizational context. Organizations could improve the effective use of software metrics by increasing transparency around how metrics are selected and used, and by involving both managerial and non-managerial roles in metric-related discussions. Providing targeted training, establishing shared definitions of commonly used metrics, and aligning metrics with improvement goals rather than solely with evaluation or control may help bridge the gap between perceived and demonstrated knowledge. In particular, encouraging teams to use metrics as tools for reflection and continuous improvement—rather than as performance indicators alone—may lead to more meaningful and sustainable measurement practices.

6. Conclusion

Using the software metrics during software development process clearly is not *deux es machina* solution for all the challenges regarding the software quality. Nevertheless, its

usage can greatly help in tackling some of the common pitfalls and shortcomings in the software industry. Survey conducted as part of this research showed that there are no real differences in perspective about the importance of software metrics to overall software quality between managers and employees. It seems that there is a consensus that using of the measurement is welcomed and should be embraced, although the level of knowledge about software metrics appears to be imbalanced between two groups. Results of the survey are pretty much in line with similar surveys conducted in the past, with differences that can be attributed to different sample sizes, cultural diversity, respondent's experience, and role etc.

In the future, similar survey could be conducted with some wider population. Survey should include multiple-choice questions about the knowledge of the concrete software metrics, instead of free-form ones. Also, questions about validation of measurement data could be added as well.

Acknowledgments. Authors thank all the companies and their employees for taking part in the survey, especially Endava d.o.o Belgrade. Also, authors wish to thank SQAMIA initiative (http://sqamia.org/). the following members for their help in conducting this survey: Tihana Galinac Grbac, Goran Mausa, Damir Kalpić, Melinda Toth, Zoltan Horvath, Zoltan Porkolab, Hannu Jaakoola, Jaak Henno, Novica Nosović, Dušanka Bošković, Dražen Brđanin, Anastas Mishev, Bojana Koteska, Klaus Bothe, Marjan Heričko, Stanimir Stoyanov and Asya Stoyanova.

References

1. ISO 9001:2015 Quality management systems — Requirements. Standard, International Organization for Standardization, Geneva, CH, September 2015.
2. James A Highsmith. *Agile software development ecosystems*, volume 13. Addison-Wesley Professional, 2002.
3. Gordana Rakić. SSQSA: Set of Software Quality Static Analysers, 2016.
4. Nasir U Eisty, George K Thiruvathukal, and Jeffrey C Carver. A survey of software metric use in research software development. In *2018 IEEE 14th International Conference on e-Science (e-Science)*. IEEE, October 2018.
5. Rini Solingen and Egon Berghout. The Goal/Question/Metric Method: A Practical Guide for Quality Improvement of Software Development. January 1999.
6. Sahra Sedigh-Ali, Arif Ghafoor, and Raymond A. Paul. *A Metrics-Guided Framework for Cost and Quality Management of Component-Based Software*, page 374–402. Springer Berlin Heidelberg, 2003.
7. Cost of poor software quality in the U.S.: A 2022 Report. https://www.it-cisq.org/wp-content/uploads/sites/6/2022/11/CPSQ-Report-Nov-22-2.pdf. Accessed: 2024-09-01.
8. Bill Haskins, Jonette Stecklein, Brandon Dick, Gregory Moroney, Randy Lovell, and James Dabney. Error Cost Escalation Through the Project Life Cycle. *INCOSE International Symposium*, 14(1):1723–1737, June 2004.
9. Dedi Setiadi, Tata Sumitra, Ahmad Karim, and Ritzkal. Software quality measurement analysis on academic information systems. 08 2024.
10. Jitender Kumar Chhabra and Varun Gupta. A survey of dynamic software metrics. *Journal of Computer Science and Technology*, 25(5):1016–1029, September 2010.
11. Krishnamoorthy Srinivasan and Dr. T. Devi Head. A comprehensive review and analysis on object-oriented software metrics in software measurement. 2014.

12. Ming-Chang Lee. Software quality factors and software quality metrics to enhance software quality assurance. *British Journal of Applied Science & Technology*, 4(21):3069–3095, January 2014.
13. Luka Pavlic, Mojca Okorn, and Marjan Hericko. The use of the software metrics in practice. In *SQAMIA*, volume 2217 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2018.
14. Paulinus Ofem, Bassey Isong, and Francis Lugayizi. Metrics for evaluating and improving transparency in software engineering: An empirical study and improvement model. *SN Computer Science*, 5(8):1097, 2024.
15. Touseef Tahir, Ghulam Rasool, Waqar Mehmood, and Cigdem Gencel. An evaluation of software measurement processes in pakistani software industry. *IEEE Access*, 6:57868–57896, 2018.
16. Mark Kasunic. The state of software measurement practice: Results of 2006 survey. 2006.
17. Wentao Chen, Huiqun Yu, Guisheng Fan, Zijie Huang, and Yuguo Liang. Usage patterns of software product metrics in assessing developers' output: A comprehensive study. *Information and Software Technology*, 189:107935, 2026.
18. Félix Témolé and Desislava Atanasova. An integrated approach to managing software quality in complex systems. *American Journal of Software Engineering and Applications*, 13(1):1–17, 2025.
19. K. V. Jeeva Padmini, H. M. N. Dilum Bandara, and Indika Perera. Use of software metrics in agile software development process. In *2015 Moratuwa Engineering Research Conference (MERCon)*, pages 312–317, 2015.
20. Harald Schaffernak, Birgit Moesl, Philipp Url, Ioana Victoria Koglbauer, and Wolfgang Vorraber. Towards sustainable software quality in use: a review of measures. *Next Research*, 2(3):100680, 2025.
21. Fatima Nur Colakoglu, Ali Yazici, and Alok Mishra. Software product quality metrics: A systematic mapping study. *IEEE Access*, 9:44647–44670, 2021.
22. Arthur-Jozsef Molnar, Alexandra Neamţu, and Simona Motogna. *Evaluation of Software Product Quality Metrics*, page 163–187. Springer International Publishing, 2020.
23. Alok Mishra, Raed Shatnawi, Cagatay Catal, and Akhan Akbulut. Techniques for calculating software product metrics threshold values: A systematic mapping study. *Applied Sciences*, 11(23):11377, December 2021.
24. Martin Vogel, Peter Knapik, Moritz Cohrs, Bernd Szyperrek, Winfried Pueschel, Haiko Etzel, Daniel Fiebig, Andreas Rausch, and Marco Kuhrmann. Metrics in automotive software development: A systematic literature review. *Journal of Software: Evolution and Process*, 33(2), August 2020.
25. Junaid Rashid, Toqeer Mahmood, and Muhamad Wasif Nisar. A study on software metrics and its impact on software quality, 2019.
26. Kshirasagar Naik and Priyadarshi Tripathy. *Software testing and quality assurance*. John Wiley & Sons, Nashville, TN, 2 edition, February 2016.
27. ISO/IEC 9126-1:2001 Software engineering — Product quality. Standard, International Organization for Standardization, Geneva, CH, January 2001.
28. ISO/IEC 25000:2014 Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — Guide to SQuaREs. Standard, International Organization for Standardization, Geneva, CH, March 2014.
29. Jørgen Bøegh. A new standard for quality requirements. *Software, IEEE*, 25:57 – 63, 04 2008.
30. Roger Pressman and Bruce Maxim. *Software Engineering: A Practitioner's Approach*. McGraw-Hill Education, Columbus, OH, 9 edition, November 2019.
31. Stephen H Kan. *Metrics and models in software quality engineering*. Addison Wesley, Boston, MA, 2 edition, September 2002.
32. Tong Li. *An approach to modelling software evolution processes*. Springer, Berlin, Germany, 2009 edition, March 2009.

33. Thoms Ball. The concept of dynamic analysis. In *Proceedings of the 7th European Software Engineering Conference Held Jointly with the 7th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, ESEC/FSE-7, page 216–234, Berlin, Heidelberg, 1999. Springer-Verlag.
34. Bas Cornelissen, Andy Zaidman, Arie Deursen, Leon Moonen, and Rainer Koschke. A systematic survey of program comprehension through dynamic analysis. *Software Engineering, IEEE Transactions on*, 35:684 – 702, November 2009.
35. Michael Ernst. Static and dynamic analysis: Synergy and duality. May 2003.
36. Mrinal Rawat, Arpita Mittal, and Sanjay Dubey. Survey on impact of software metrics on software quality. *International Journal of Advanced Computer Science and Applications*, 3, January 2012.
37. Norman Fenton and James Bieman. *Software metrics*. Chapman & Hall/CRC Innovations in Software Engineering and Software Development Series. CRC Press, London, England, 3 edition, September 2020.
38. Norman E Fenton and Martin Neil. Software metrics: successes, failures and new directions. *Journal of Systems and Software*, 47(2):149–157, 1999.
39. Norman Fenton and Martin Neil. Software metrics: Roadmap. *Proceedings of the Conference on the Future of Software Engineering*, September 2000.
40. Kaushal Bhatt, Vinit Tarey, and Pushpraj Patel. Analysis Of Source Lines Of Code(SLOC) Metric. *IJETAE*, 2, April 2012.
41. Maurice H. Halstead. *Elements of Software Science (Operating and programming systems series)*. Elsevier Science Inc., USA, 1977.
42. T.J. McCabe. A complexity measure. *IEEE Transactions on Software Engineering*, SE-2(4):308–320, 1976.
43. S.R. Chidamber and C.F. Kemerer. A metrics suite for object oriented design. *IEEE Transactions on Software Engineering*, 20(6):476–493, 1994.
44. Mark Lorenz and Jeff Kidd. *Object-oriented software metrics*. Prentice Hall, Philadelphia, PA, May 1994.
45. Kenneth L. Morris. *Metrics for object-oriented software development environments*. PhD thesis, Massachusetts Institute of Technology, 1999.
46. W. Li and S. Henry. Maintenance metrics for the object oriented paradigm. In *[1993] Proceedings First International Software Metrics Symposium*, pages 52–60, 1993.
47. Crt Gerlec, Gordana Rakić, Zoran Budimac, and Marjan Hericko. A programming language independent framework for metrics-based software evolution and analysis. *Comput. Sci. Inf. Syst.*, 9:1155–1186, September 2012.
48. Yania Crespo, Carlos Nozal, M. Manso, and Raúl Sánchez. Language independent metric support towards refactoring inference. January 2005.
49. Martin Kunz, Reiner Dumke, and Niko Zenker. Software metrics for agile software development. pages 673–678, April 2008.
50. Graeme Ruxton. The Unequal Variance T-Test is an Underused Alternative to Student's T-Test and the Mann-Whitney U Test. *Behavioral Ecology*, 17, April 2006.

APPENDIX

What is your level of knowledge about software metrics in general?		
	Serbia	*Other country*
Mean	2.467742	3.195652174
Variance	1.138287	0.871980676
Observations	62	46
Hypothesized Mean Difference	0	
df	103	
t Stat	-3.76819	
P(T<=t) one-tail	0.000137	
t Critical one-tail	1.659782	
P(T<=t) two-tail	0.000274	
t Critical two-tail	1.983264	

What is your level of knowledge about software metrics in general?		
	Experienced	*Less experienced*
Mean	3.094339623	2.472727273
Variance	1.279390421	0.846464646
Observations	53	55
Hypothesized Mean Difference	0	
df	100	
t Stat	3.126495659	
P(T<=t) one-tail	0.001158291	
t Critical one-tail	1.660234326	
P(T<=t) two-tail	0.002316582	
t Critical two-tail	1.983971519	

What is your level of knowledge about software metrics in general?		
	Managers	*Non-managers*
Mean	3.016949	2.489795918
Variance	1.120397	1.046768707
Observations	59	49
Hypothesized Mean Difference	0	
df	104	
t Stat	2.624231	
P(T<=t) one-tail	0.004996	
t Critical one-tail	1.659637	
P(T<=t) two-tail	0.009992	
t Critical two-tail	1.983038	

Measurement-based data helps your team to perform better than without using it		
	Medium	*Small/Large*
Mean	4.697674419	3.948717949
Variance	0.549280177	1.997300945
Observations	43	39
Hypothesized Mean Difference	0	
df	56	
t Stat	2.960815687	
P(T<=t) one-tail	0.002246619	
t Critical one-tail	1.672522303	
P(T<=t) two-tail	0.004493238	
t Critical two-tail	2.003240719	

How often are software metrics used in the aggregate to evaluate overall team productivity?		
	Serbia	*Other country*
Mean	2.912281	2.511111111
Variance	0.974311	1.301010101
Observations	57	45
Hypothesized Mean Difference	0	
df	87	
t Stat	1.870371	
P(T<=t) one-tail	0.032397	
t Critical one-tail	1.662557	
P(T<=t) two-tail	0.064795	
t Critical two-tail	1.987608	

What is the importance of measurement for success of the project / process / product?		
	Medium	*Small/Large*
Mean	4.581818182	4.133333333
Variance	0.988552189	1.618181818
Observations	55	45
Hypothesized Mean Difference	0	
df	82	
t Stat	1.93116504	
P(T<=t) one-tail	0.028458955	
t Critical one-tail	1.663649184	
P(T<=t) two-tail	0.05691791	
t Critical two-tail	1.989318557	

Assessment of software quality tracking

Thank you for taking part in this survey!

This questionnaire is created in order to get insight of techniques and tools used to assess and evaluate quality of software in IT companies.

The answers will be used in a survey document about usage of measurement in practical environments. This survey document will be accessible to all parties taking part in answering the questionnaire.

Warranty: All data will be used and published only in an anonymized survey form.

Author of the questionnaire:
Prof. Zoran Budimac
Department of Mathematics and Informatics
University of Novi Sad
zjb@dmi.uns.ac.rs

* Required

General questions

1. How many years of experience in IT industry do you have? *

Mark only one oval.

- 0-2 years
- 2-5 years
- 5-10 years
- 10-15 years
- 15+ years

2. In which country is your company located? *

_____

3. What is the size of the company you're currently working for? *

Mark only one oval.

- Small (less than 50 employees)
- Medium (51-1000 employees)
- Large (1000+ employees)

4. What is the type of company you're currently working for? *

Mark only one oval.

- Startup
- Service-oriented
- Product-oriented
- Global In House Center
- Other: _____

5. Which software development technologies are used in the project you're currently working on? *

Check all that apply.

- Java
- .NET/C#
- JavaScript (or its frameworks)
- PHP
- Ruby
- C/C++
- Python
- Kotlin
- Swift
- Haskell
- Smalltalk
- Other: _____

6. Which software development methodology is used in the project you're currently working on? *

Mark only one oval.

- Waterfall
- Prototype
- Spiral model
- Iterative-incremental
- Scrum
- Lean
- Extreme programming
- Other: _____

7. What roles do you have in the organizational structure of the project (choose all that apply)? *

Check all that apply.

- Senior manager
- Middle manager
- Project manager
- Product owner/Business analyst
- Scrum master
- Team leader
- Solution architect
- Developer
- QA analyst
- Test engineer
- DevOps
- Other: _____

Software metrics in your project

8. Which best describes the current development stage of your project? *

Mark only one oval.

- Planning/Requirements Gathering
- Initial Development/Prototyping
- Active Development/Unreleased Software
- Active Development/Released Software
- Maintenance/No New Development Planned
- Other: _____

9. What is your level of knowledge about software metrics in general? *

Mark only one oval.

- Very Low
- Low
- Average
- High
- Very High

10. List all software metrics which you are familiar with

_____
_____
_____
_____
_____

11. In which stages of the development a measuring is made? *

Check all that apply.

- Planning
- Business modelling
- Design
- Coding
- Testing
- Using
- Never
- I don't know
- Other: _____

12. List all metrics that are used in the measurements?

_____
_____
_____
_____
_____

13. During the project life cycle measuring is done *

Mark only one oval.

- Continuously
- At a certain point (every week, month, some milestones ...)
- If necessary
- At the request of higher instance
- Randomly
- Never
- I don't know
- Other: _____

14. What kind of tools is used for the software measurements (and which)? *

Check all that apply.

- IDE
- Separate metrics tool
- Designing tool
- Modelling tool
- Testing tool
- None
- I don't know
- Other: _____

15. What features of the tools used in the measurements were of crucial importance in order to choose them? (choose all that apply) *

Check all that apply.

- Independency of platform
- Supported certain platforms
- Supporting the particular programming language
- Independency of programming language
- Supported certain metrics
- Supporting the large number of metrics
- Visualization of results
- Saving and interpretation of results
- None
- I don't know
- Other: _____

16. What are the disadvantages of selected tools? (choose all that apply) *

Check all that apply.

- Dependency of platform
- Unsupported certain platforms
- Lack of support for the particular programming language
- Dependency of programming language
- Unsupported certain metrics
- Supporting inadequate set of metrics
- Lack of visualization of results
- Lack of saving and interpretations of results
- None
- I don't know
- Other: _____

17. What is the importance of measurement for success of the project / process / product? *

Mark only one oval.

- Crucial
- Great
- Medium
- Small
- Insignificant
- No impact
- I don't know

18. How would you describe the influence of using software metrics on the overall quality of the project / process / product? *

Mark only one oval.

- Crucial
- Great
- Medium
- Small
- Insignificant
- No impact
- I don't know

19. How often are software metrics used in the aggregate to evaluate overall team productivity? *

Mark only one oval.

- Never
- Rarely
- Sometimes
- Most of the Time
- Always
- I don't know

20. Measurement-based data helps your team to perform better than without using it *

Mark only one oval.

- Strongly disagree
- Disagree
- Somewhat disagree
- Not sure
- Somewhat agree
- Agree
- Strongly agree

Software measurement usage

21. Whose activities are monitored / controlled by the measurements? (choose all that apply) *

Check all that apply.

- Senior manager
- Middle manager
- Project manager
- Product owner/Business analyst
- Scrum master
- Team leader
- Solution architect
- Developer
- QA analyst
- Test engineer
- DevOps
- No one
- I don't know
- Other: _____

22. Who uses measurement results? (choose all that apply) *

Check all that apply.

- [ ] Senior manager
- [ ] Middle manager
- [ ] Project manager
- [ ] Product owner/Business analyst
- [ ] Scrum master
- [ ] Team leader
- [ ] Solution architect
- [ ] Developer
- [ ] QA analyst
- [ ] Test engineer
- [ ] DevOps
- [ ] No one
- [ ] I don't know
- [ ] Other: _____

23. Who has the access to measurement results? (choose all that apply) *

Check all that apply.

- [ ] Senior manager
- [ ] Middle manager
- [ ] Project manager
- [ ] Product owner/Business analyst
- [ ] Scrum master
- [ ] Team leader
- [ ] Solution architect
- [ ] Developer
- [ ] QA analyst
- [ ] Test engineer
- [ ] DevOps
- [ ] No one
- [ ] I don't know
- [ ] Other: _____

24. How much resources are consumed when the metrics are introduced and established (person hours, days ...)? *

_____

25. How much resources are consumed when the metrics are used (person hours, days ...)? *

_____

26. The definitions of measures that are used in my organization/current projects are commonly understood and consistent *

Mark only one oval.

- ◯ Strongly disagree
- ◯ Disagree
- ◯ Somewhat disagree
- ◯ Not sure
- ◯ Somewhat agree
- ◯ Agree
- ◯ Strongly agree

Thanks!

Thank you for taking the time to complete this survey. We truly value the information you have provided. Your responses will contribute to our assessment of software quality tracking in IT companies.

Filip Prentović was born on June 15, 1982, in Odžaci, Serbia. He received a B.S. degree in Computer Science from the Faculty of Sciences, University of Novi Sad, in 2005. He received his M.S. degree from the same faculty in 2018. Currently, he is a Ph.D. student at the Faculty of Sciences. His fields of interest include software architectures, software quality, and software metrics.

Zoran Budimac was a full professor at the Faculty of Sciences, University of Novi Sad, Serbia. He was born on December 24, 1960, in Sombor, Serbia. He obtained all his degrees at the Faculty of Sciences, University of Novi Sad: a B.Sc. degree in Informatics in 1983, an M.Sc. degree in Discrete Mathematics and Programming in 1992, and a Ph.D. degree in Informatics in 1994. His research interests included software engineering, software quality, e-learning, and compiler construction. He was a principal investigator in several domestic and international projects and a participant in many others. He was the author of 16 textbooks and more than 280 research papers, most of which were published in international journals and conference proceedings. He was/is a member of the Organizing and/or Program Committees of more than 70 international conferences and a member of the Managing Board of "Computer Science and Information Systems".

Marjan Heričko is a full professor at the Institute of Informatics. He is the Head of the Information Systems Laboratory and Deputy Head of the Institute of Informatics. He received his Ph.D. in Computer Science from the University of Maribor in 1998. His main research interests include all aspects of information systems development, software and service engineering, agile methods, process frameworks, software metrics, functional size measurement, SOA, component-based development, object orientation, software reuse, and software patterns. Dr. Heričko has been a project or work coordinator in several applied projects, as well as a project or work coordinator in several international research projects, and has served as a committee member and chair of several international conferences.

Gordana Rakić has held the position of Assistant Professor with a Ph.D. at the Faculty of Sciences, University of Novi Sad, Serbia, since 2016. She obtained all her degrees at the Faculty of Sciences, University of Novi Sad: a B.Sc. degree in Informatics in 2006, an M.Sc. degree in Business Informatics in 2010, and a Ph.D. degree in Computer Science in 2015. Her research interests include software engineering, software quality, static analysis, and computer languages. Gordana is one of the founders and a member of the SQLab: Software Quality Laboratory under the Department of Mathematics and Informatics at her home faculty. Furthermore, she is a member of ESUG (European Smalltalk User Group), as well as ACM and IEEE. She has participated in several ongoing and completed bilateral, multilateral, and national research projects. Currently, Gordana is chairing COST Action CA19136 — Connecting Education and Research Communities for an Innovative Resource Aware Society (CERCIRAS). She has published more than 30 scientific papers and articles. She has participated in the organization of several international scientific conferences, symposia, and workshops as a member, secretary, co-chair, and chair of organizing and program committees, and serves as a reviewer for several international journals. In the period from 2011 to 2013, she was one of the managing editors of the "ComSIS: Computer Science and Information Systems" journal.

Received: November 9, 2025; Accepted: May 1, 2026.