Ministry of Education and Science of Ukraine
National University of "Kyiv-Mohyla Academy"
Network Technologies Department of the Faculty of Informatics

Investigation of the relationship between software metrics measurements and its maintainability degree

Text part of course work
in specialty “Computer Science and Information Technologies” 112

Coursework supervisor
associate professor, doctor of technical science
Hlybovets A.M.

_____
(signature)
“____” _____ 2020 yr.

Made by student
Shapoval O.O
“____” _____ 2020 yr.

Kyiv, 2020

Ministry of Education and Science of Ukraine
National University of "Kyiv-Mohyla Academy"
Network Technologies Department of the Faculty of Informatics

APPROVED
Head of the Department of Informatics
PhD, associate professor
_____ Gorokhovsky S.S.
(signature)
"____" _____ 2020 yr.

INDIVIDUAL TASK
for course work

For the student of the Faculty of Informatics of 1 course master's degree studying
THEME Investigation of the relationship between software metrics measurements and maintainability degree

Output data:

Text part content of coursework:

- An individual task
- Calendar plan
- Annotation
- Introduction
- Tools description
- Primary statistical analysis
- Correlations analysis
- Regression analysis
- Summary
- References

Issue date "____" _____ 2020 yr. Supervisor _____
(signature)
Task received _____
(signature)

Theme: Investigation of the relationship between software metrics measurements and maintainability degree

Calendar plan of coursework execution:

№	Stage name	Deadline	Note
1.	Getting of coursework topic	01.09.2020	
2.	Searching of appropriate literature	10.09.2020	
3.	Researching in software measurement and empirical engineering area	10.12.2020	
4.	Exploring of technologies and tools	01.01.2020	
5.	Specification definition	05.02.2020	
6.	Documentation forming	10.02.2020	
7.	Writing introduction and tools part	23.04.2020	
8.	Coursework analysis with the supervisor	24.04.2020	
9.	Writing second part of the coursework	30.04.2020	
10.	Coursework analysis with the supervisor	01.05.2020	
11.	Writing coursework summary	01.05.2020	
12.	Coursework analysis with the supervisor	02.05.2020	
13.	Coursework changing according to the supervisor's remarks	03.05.2020	
14.	Creating of the presentation	05.05.2020	
15.	Defending of the coursework	14.05.2020	

Student Shapoval O.O.
Supervisor Hlybovets A.M.
"____" _____

Table of Contents

Annotation

The goal of this course work was to practically learn methods of empirical engineering software, algorithms for data collection and data analysis. Results include software measurement, analysis and selection of direct and indirect metrics for research and identification of dependencies between direct and indirect metrics. On the basis of received result were built dependencies between software metrics and software expertise properties. Metrics and properties selected by individual variation.

Relationship between metric and expertise includes building direct relationships between the metric and expertise, indirect metrics and expertise. Additionally, was determined whether they have common trends of the relationship between those direct metrics and expert estimates, indirect metrics and expert estimates.

Key words: Software quality attributes, Understandability, Functionality Completeness, Direct Metrics, CYC, NOM, NOC, CALL, FOUT, Indirect Metrics, AMW, ATFD, BOvR.

1. Introduction

1.1 Understandability property

Software developers and accompanying persons should read and understand the source programs and other types of program documents in their work. The clarity of software documents is therefore important. To understand the level of comprehensibility of the software, we need to answer the following questions [8].

Are variable names describing a physical or functional property provided? Do uniquely recognizable functions contain adequate comments so that their purpose is clear? Are deviations from the direct logical stream sufficiently commented? Are all array elements functionally related?

When programmers try to reuse a software developed by other programmers, the difficulty in understanding the system limits reuse. It is not easy to measure the comprehensibility of software, because understanding is an internal process of people. There is the notion of the “intelligibility integral” as a model for measuring software intelligibility, which can first extract the best software intelligibility value from higher weight factors. In other words, we have given an integrated dimension to measuring software comprehensibility through the literature on this subject [4].

1.2 Functionality completeness property

Completeness refers to the absence of omission errors in the program and database. It is evaluated against a specification of software requirements that define the desired degree of generalization and abstraction (selective omission). “Data completeness” is a measurable error between the database and the specification [8]. Even highly generalized databases can be “complete data” if they contain all the objects described in the specification. A database is a “complete model" if its specification is appropriate for the application.

Validity is a measure of the attribute of the accuracy of software functionality following the specification of requirements. Each attribute must have a specific domain and range. Program validation is the process of determining whether the values of program processes are sufficiently accurate, complete, and logically compatible with the intended use of the data. Verification often consists of several steps, including logical checks, accuracy estimates, and error analysis.

1.3 Lines of Code (SLOC)

Lines of code is a software metric used to measure the size of a program by counting the number of lines in the text of the program source code. SLOC is typically used to predict the amount of effort that will be required to develop a program, as well as to evaluate the performance or programming effort after a software release [10].

Many useful comparisons include only the order of magnitude of the lines of code in a project. Software projects can vary from 10 to 100,000,000 or more lines of code. Using lines of code to compare a project with 10,000 lines and a project with 100,000 lines is much more useful than comparing a project with 20,000 lines and a project with 21,000 lines. Although it is debatable exactly how to measure lines of code, discrepancies of the order can be clear indicators of software complexity or man-hours [10].

There are two main types of SLOC measures: physical SLOC and logical SLOC. The specific definitions of these two indicators differ, but the most common definition of physical SLOC is the number of lines in the source code of the program, including comment lines. Blank lines are also included unless the lines of code in the section contain more than 25% blank lines. In this case, empty lines exceeding 25% are not taken into account in the lines of code [10].

1.4 Number of Methods (NOM)

The metric “Number of methods” is used to calculate the average number of operations of all classes in a class. A class must have several, but not excessive, operations.

This information is useful in identifying a lack of primitiveness in class operations (prohibition of reuse) and in classes that are slightly larger than data types.
For a class, this is a simple count of the number of operations.
For a package, this is the average number of operations per package class.
This value should remain between 3 and 7. This will mean that the class has operations, but not too many. A value greater than 7 may indicate the need for further object-oriented decomposition, or that the class does not have a consistent goal. A value of 2 or less indicates that this is not a class, but just a data construct [1].

Table 1. NOM statistical thresholds

	Java			C++		
Metric	Low	Average	High	Low	Average	High
LOC/Operation	7	10	13	5	10	16
NOM/Class	4	7	10	4	9	15

Table 2. NOM characteristic table

NOM - Number of Methods								
Definition	The number of methods of a class							
Used for	Refused Parent Bequest(145), Tradition Breaker(152)							
Measured Entity	Class	Definition	Abstract					
		user-defined	+					
Involved Relations								
HAS (contains)	Method	Definition	Visibility	Get/Set	Constr.	Static	Abstract	
		measured class	all	+	+	+	+	

1.5 Number of Direct Descendants (NDD)

This metric is the number of direct descendants (subclasses) for each class. It is believed that classes with a large number of children are difficult to modify, and, as a rule, they require additional testing due to the effect of changes on all children. They are also considered more complex and error-prone, because a class with many children may need to provide services in more contexts and therefore should be more flexible [2].

1.6 Height of Inheritance Tree (HIT)

The height of the inheritance tree for a class or structure is the number of base classes (including the System.Object class, so DIT> = 1). Types for which HeigthOfInheritance is above 6 may be difficult to maintain. However, this is not the rule, because sometimes your classes can inherit from level classes that are of high importance for the depth of inheritance. For example, the average depth of inheritance for wireframe classes derived from System.Windows.Forms.Control [2].

1.7 Fan-out (FOUT)

The number of classes to call, this is calculated as the sum of the FANOUT metric (that is, the classes from which the operations call the methods) for all user operations. This metric provides raw information about distributed activity calls in classes.

Branching is a measure of the ability of an electronic logic gate output to control multiple inputs of other logic gates of the same type. In most designs, the logic gates are connected to form more complex circuits, and usually, one output of the logic element is connected to several inputs of the logic element. The technology used to implement logic gates typically allows direct connection of gate inputs without the need for additional interface circuits [1].

Table 3. FOUT statistical thresholds

	Java			C++		
Metric	Low	Average	High	Low	Average	High
FANOUT /Call	0.56	0.62	0.68	0.20	0.34	0.48

1.8 Average Method Weight (AMW)

Table 4. AMW characteristic table

AMW - Average Method Weight								
Definition	The average static complexity of all methods in a class. McCabe's cyclomatic number is used to quantify the method's complexity (Mar02a, McC76)							
Used for	Refused Parent Bequest(145), Tradition Breaker(152)							
Measured Entity	Class	Definition	Abstract					
		user-defined	–					
Involved Relations								
HAS	Method	Definition	Visibility	Get/Set	Constr.	Static	Abstract	
		measured class	all	+	+	+	+	

Table 5. AMW statistical thresholds

	Java				C++			
Metric	Low	Ave-rage	High	Very High	Low	Ave-rage	High	Very High
WMC	5	14	31	47	4	23	72	108
AMW	1.1	2.0	3.1	4.7	1.0	2.5	4.8	7.0

1.9 Access to Foreign Data (ATFD)

Access to external data (ATFD) is the number of external classes from which this class accesses attributes directly or through access methods. Because ATFD measures how many external attributes are used by a class, it is clear that the higher the ATFD for a class, the higher the likelihood that the class is (or is about to become) a class of God. It quantifies one of the key disharmonies of identity distortion, that is, the rude use of attributes from other classes. As you can see, this is again one of the reasons why the method is considered disharmonious [1].

Table 6. ATFD characteristic table

ATFD - Access To Foreign Data							
Definition	The number of attributes from unrelated classes that are accessed directly or by invoking accessor methods (Mar02a)						
Used for	God Class(80), Feature Envy(84)						
Measured Entity	Class	Definition	Abstract				
		user-defined	+				
Measured Entity	Method	Definition	Visibility	Get/Set	Constr.	Static	Abstract
		user-defined	all	+	+	+	–
Involved Relations							
USES (accesses)	Method	Definition	Visibility	Get/Set	Constr.	Static	Abstract
		measured class or method	all	+	–	+	–
	Attribute	Definition	Visibility	Static	Const.		
		neither measured class nor a class from the same hierarchy	public	+	–		
USES (calls)	Method	Definition	Visibility	Get/Set	Constr.	Static	Abstract
		measured class or method	all	+	–	+	–
	Method	Definition	Visibility	Get/Set	Constr.	Static	Abstract
		neither measured class nor a class from the same hierarchy	public	only	–	–	–

1.10 Base Class Overriding Ratio (BOvR)

Quantifies the degree of overriding and specialization of base class methods [1].

Table 7. BOvR characteristic table

BOvR - Base Class Overriding Ratio								
Definition	The number of methods of the measured class that override methods from the base class, divided by the total number of methods in the class							
Used for	Refused Parent Bequest(145)							
Measured Entity	Class	Definition	Abstract					
		user-defined	–					
Involved Relations								
USES (overrides)	Method	Definition	Visibility	Get/Set	Constr.	Static	Abstract	
		measured class	all	+	–	–	–	
	Method	Definition	Visibility	Get/Set	Constr.	Static	Abstract	
		in base class of measured class	all	+	–	–	+	
HAS (contains)	Method	Definition	Visibility	Get/Set	Constr.	Static	Abstract	
		in measured class	all	+	–	–	–	

1.11 Tight Class Cohesion (TCC)

Is the relative number of methods directly connected via accesses of attributes [1].

Table 8. TCC characteristic table

TCC - Tight Class Cohesion							
Definition	The relative number of method pairs of a class that access in common at least one attribute of the measured class (BK95)						
Used for	God Class(80), Brain Class(97)						
Measured Entity	Class	Definition	Abstract				
		user-defined	–				
Involved Relations							
USES (accesses)	Method	Definition	Visibility	Get/Set	Constr.	Static	Abstract
		measured class	all	+	–	+	–
	Attribute	Definition	Visibility	Static	Const.		
		measured class	all	+	+		

1.12 Weight of Class (WOC)

Table 9. WOC characteristic table

WOC - Weight Of a Class							
Definition	The number of "functional" public methods divided by the total number of public members (Mar02a)						
Used for	Data Class(88)						
Measured Entity	Class	Definition	Abstract				
		user-defined	–				
Involved Relations							
HAS (contains)	Method	Definition	Visibility	Get/Set	Constr.	Static	Abstract
		measured class	public	–	–	+	–
	Method	Definition	Visibility	Get/Set	Constr.	Static	Abstract
		measured class	public	only	–	–	–
	Attribute	Definition	Visibility	Static	Const.		
		measured class	public	+	–		

2. Tools Description

2.1 IPlasma

iPlasma is a software development instrcument for the source code measurement and analytics. Initially it was developed as a university project in Romania to measure and evaluate the quality of system source code and design desicoins. As a result of analysis in creates a metrics pyramid with project measurement results and provides comparison with the standard values across the industry.
Each line has a coloured percentage; the percentage is determined by the ratio of the number in this row to the number below it. Table 10 shows what the numbers indicate starting from the top:

Table 10. Metrics in iPlasma

Code	Description
NDD	Number of direct descendants
HIT	Height of inheritance tree
NOP	Number of packages
NOC	Number of classes
NOM	Number of methods
LOC	Lines of code
CYCLO	Cyclomatic complexity
CALL	Calls per method
FOUT	Fan out (number of other methods called by a given method)

The numbers indicate the ratios; the colors indicate where the ratios fit into the industry-standard ranges (derived from numerous open source projects). Each ratio is either green (within the range), blue (below the range), or red (outside the range). For the Struts code base, NDD and CYCLO are outside the industry standards for those values, and LOC and NOM are below. [9]

Table 11. iPlasma industry ranging for metrics

	Low	Medium	High
CYCLO / Line	0.16	0.20	0.24
LOC / method	7	10	13
NOM / class	4	7	10
NOC / package	6	17	26
CALLS / method	2.01	2.62	3.20
FANOUT / call	0.56	0.62	0.68

2.2 STATISTICA

STATISTICA is a statistics and analytics software developed by StatSoft that provides data analysis, management, mining and visualization procedures. It's categories include Enterprise, Web-Based, Concurrent Network Desktop, and Single-User Desktop. Statistica provides different packages of analytical techniques:

- Basic statistics options
- Multivariate techniques and advanced linear/nonlinear regression models.
- Quality control techniques, process
- Data mining, predictive analytics, and neural networks

3. Primary Statistical Analysis

Primary data analysis is the process of comprehending the data collected to answer research questions or to support or reject research hypotheses that the study was originally designed to evaluate. The choice of data analysis methods depends on the type of data being collected, quantitative or qualitative.
After the primary statistical analysis is completed a conclusion can be made about which metrics and expert estimations belongs to normal distribution and which not. Analyzing coefficient of excess and skewness, I can assume that direct metric HIT, indirect metrics TCC WOC and all expert estimations (reliability and understandability) belong to normal distribution [5].

3.1 Direct metrics measurements results

Figure 1 depicts results of project measurements for Lines of Code metric per classes. As we can see, most of the classes (about 4250) have for about 1000 kines of code.

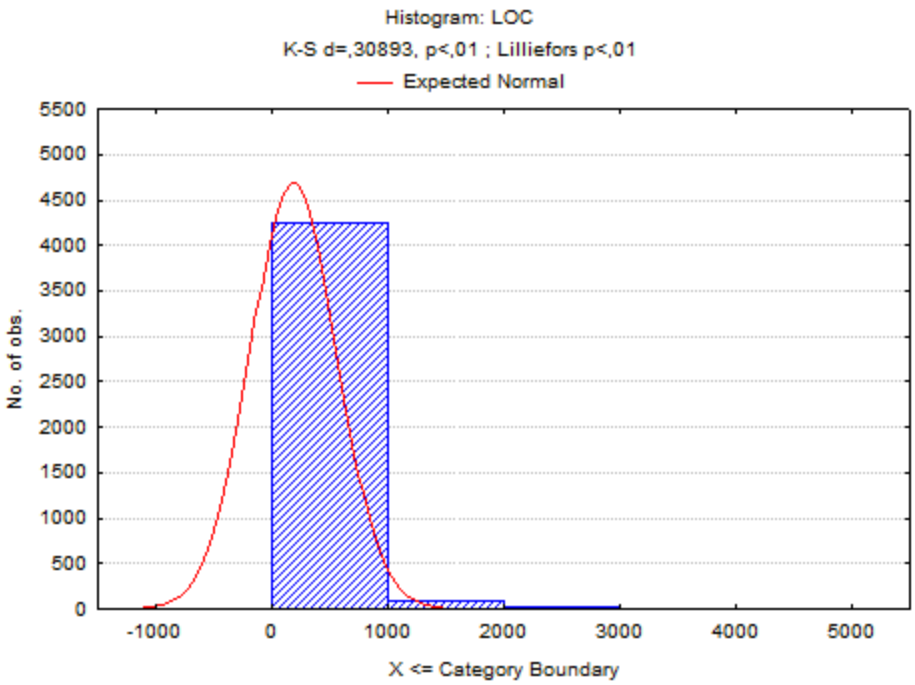

Fig. 1. LOC Histogram

It the same time, most of that have an amount of methods for about 1 to 50 (Figure 2). Which does not correlate nicely with may postulates of good software engineering.

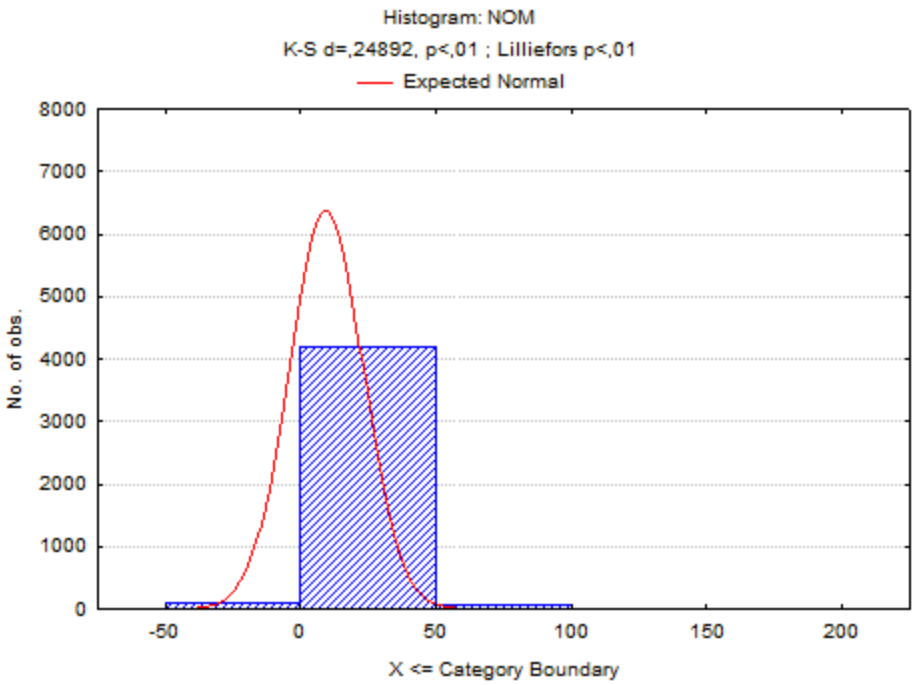

Fig. 2. NOM Histogram

The Average number of direct descendants is some cases may be in the range up to 100 (Figure 3).

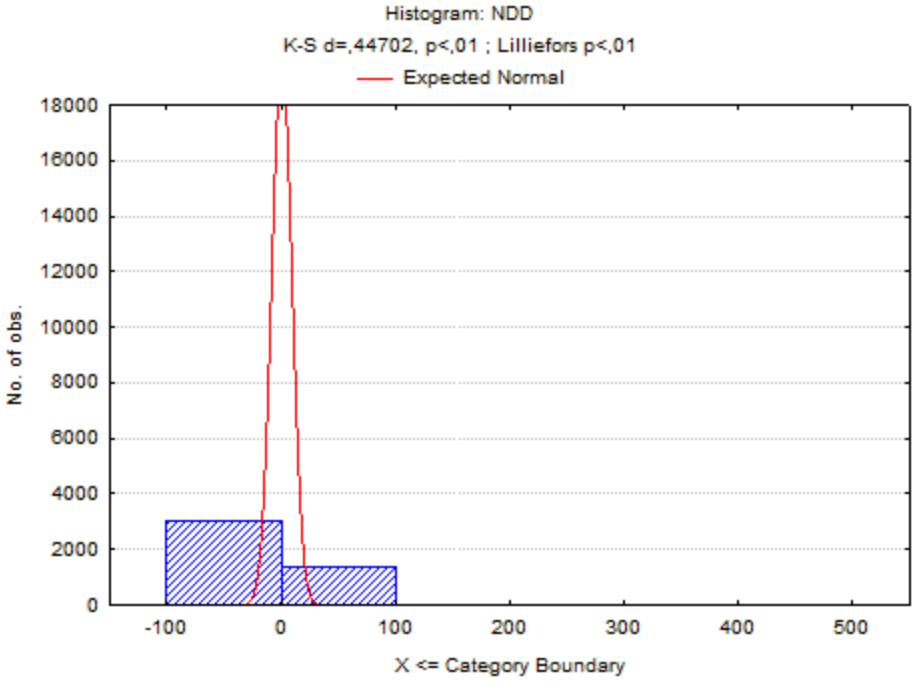

Fig. 3. NDD Histogram

HIT and FOUT metrics values (Figure 4 and 5) also are deviating from normal distribution or expected values.

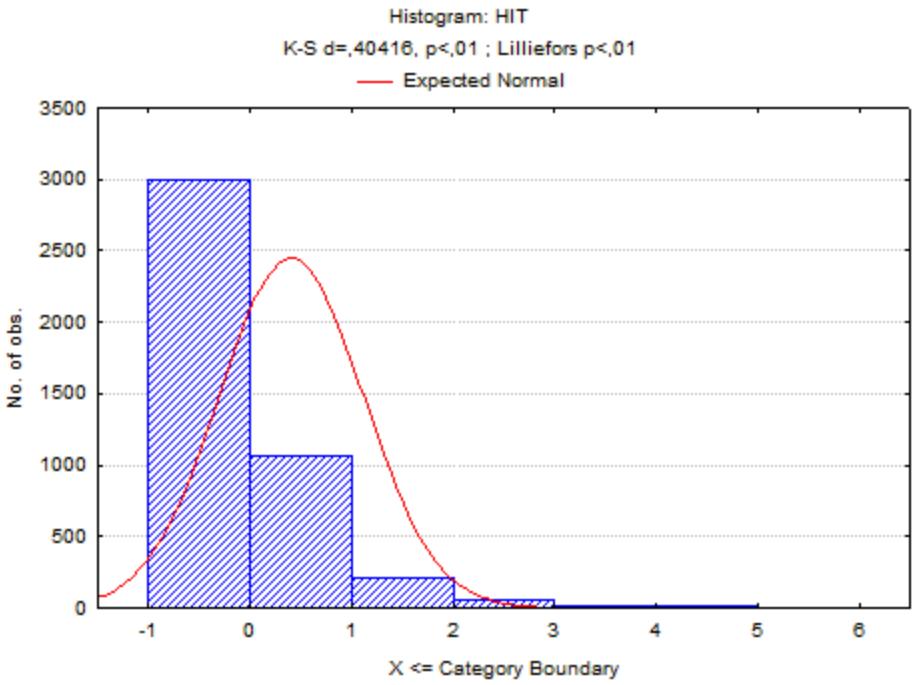

Fig. 4. HIT Histogram

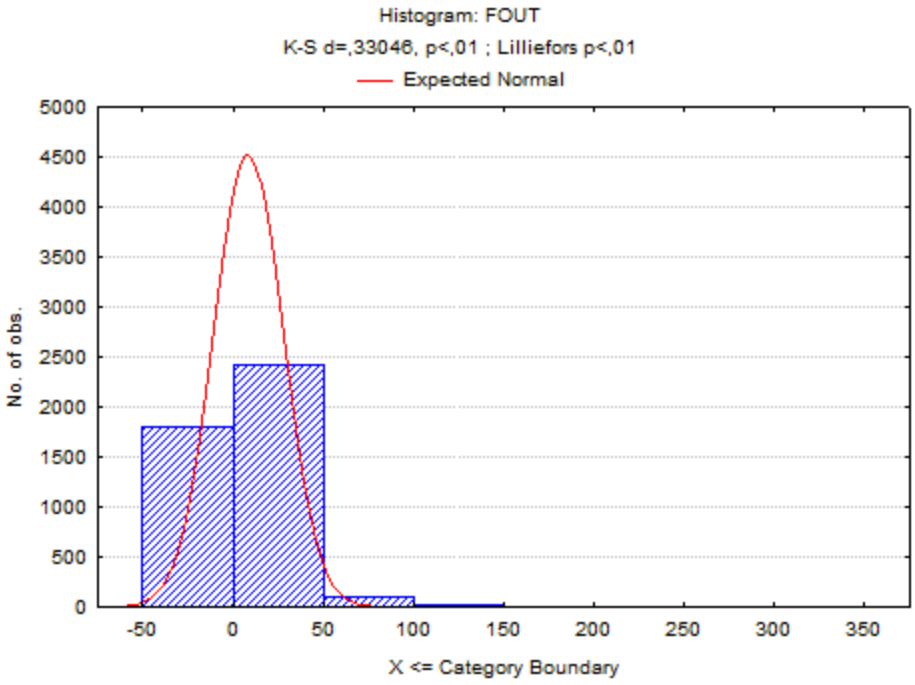

Fig. 5. FOUT Histogram

3.2 Indirect metrics measurements results

To allocate the appropriate resources for the project, we need to predict the efforts, time and costs of developing the project. Forecasting measurement always requires a mathematical model that connects the predicted attributes with some other attribute that we can measure now. Therefore, the forecasting system consists of a mathematical model along with a set of forecasting procedures for determining unknown parameters and interpreting the results [6].

In order to understand mentioned software properties, next indirect measurements were made: AMW (Figure 6), ATFD (Figure 7), BOvR (Figure 8), TCC (Figure 9), WOC (Figure 10).

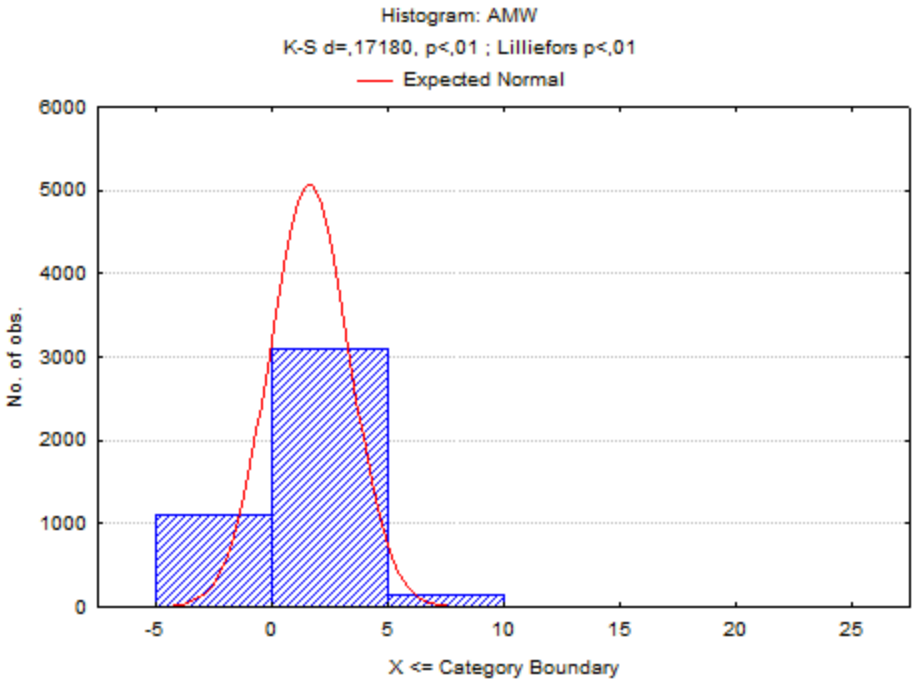

Fig. 6. AMW Histogram

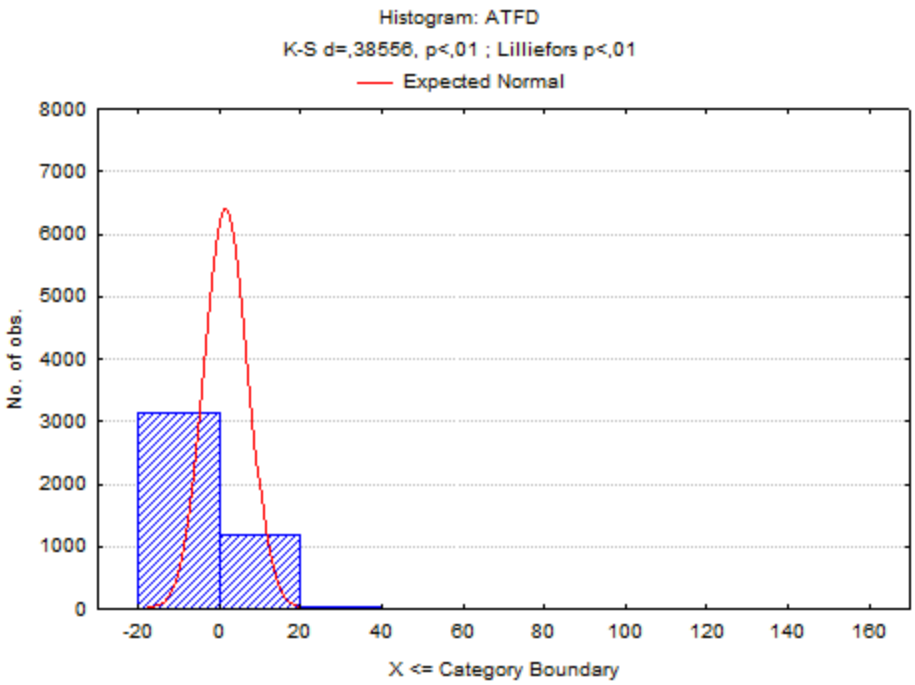

Fig. 7. ATFD Histogram

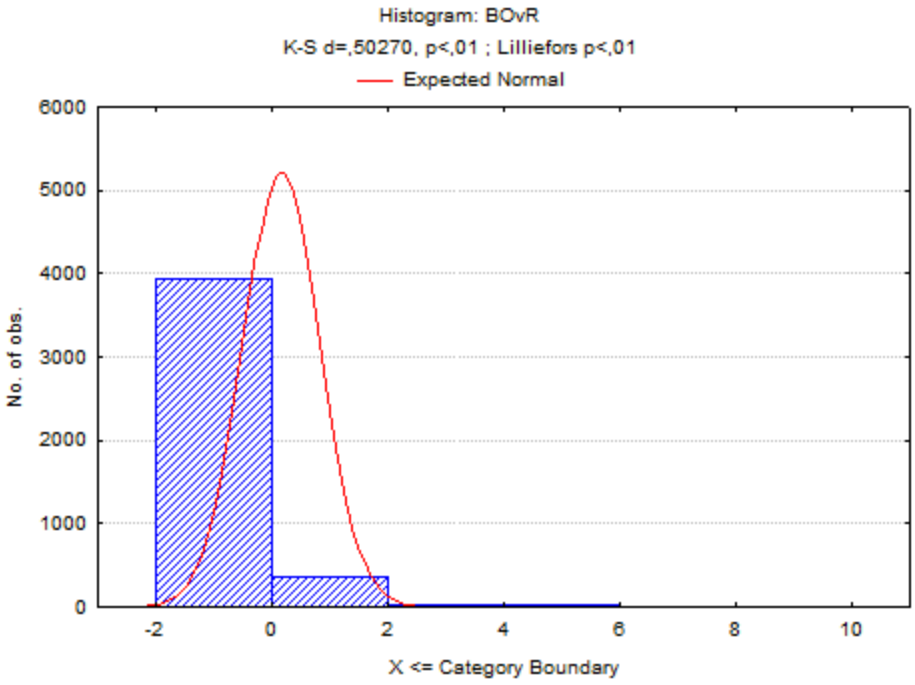

Fig. 8. BOvR Histogram

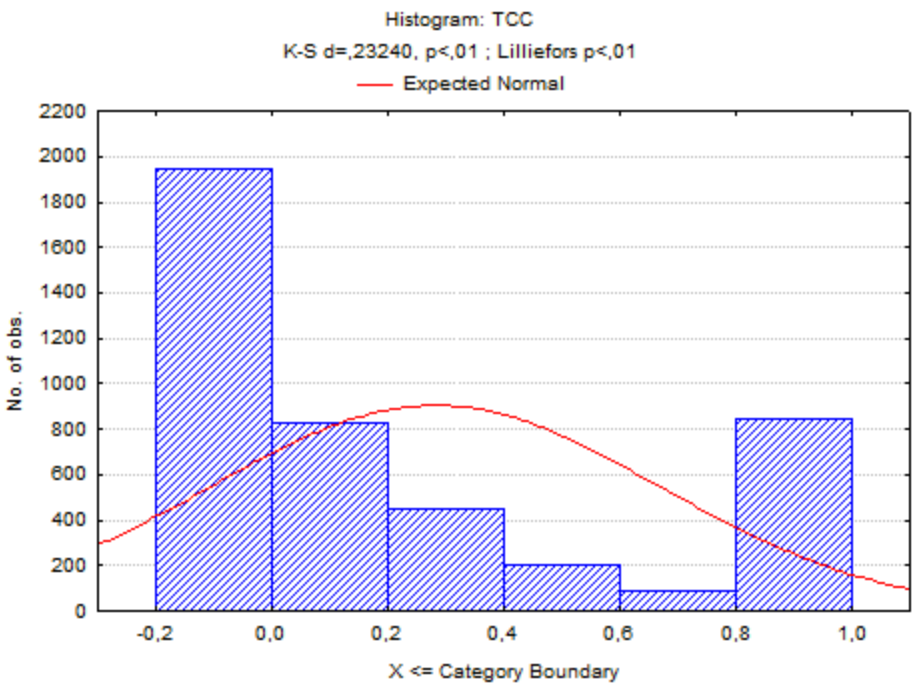

Fig. 9. TCC Histogram

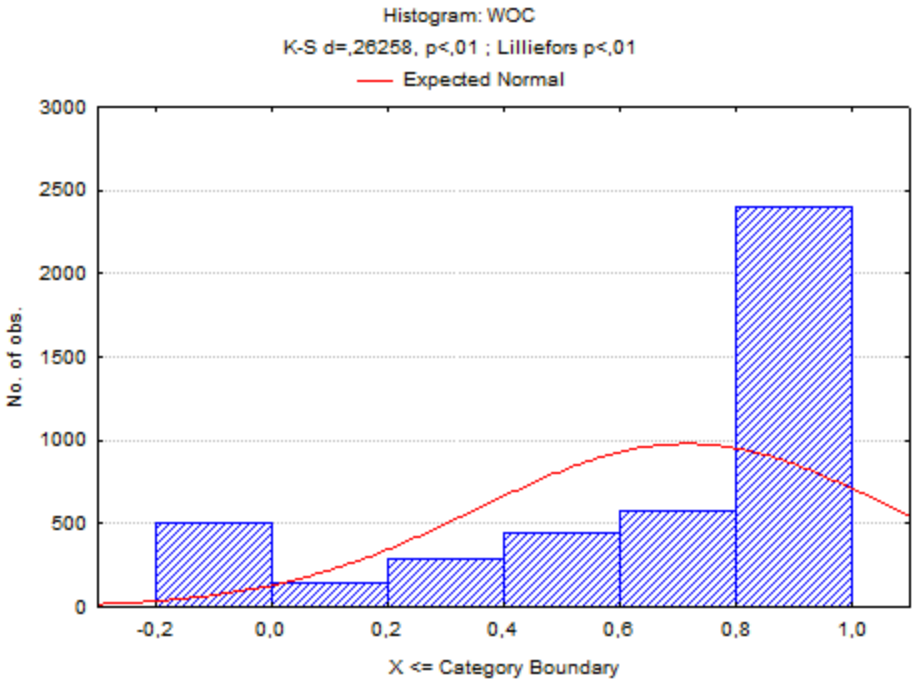

Fig. 10. WOC Histogram

3.3 Expert estimations

Based on measurement results of direct and indirect metrics the next conclusion scan be made regarding system source code understandability (Figure 11) and functional completeness (Figure 12).

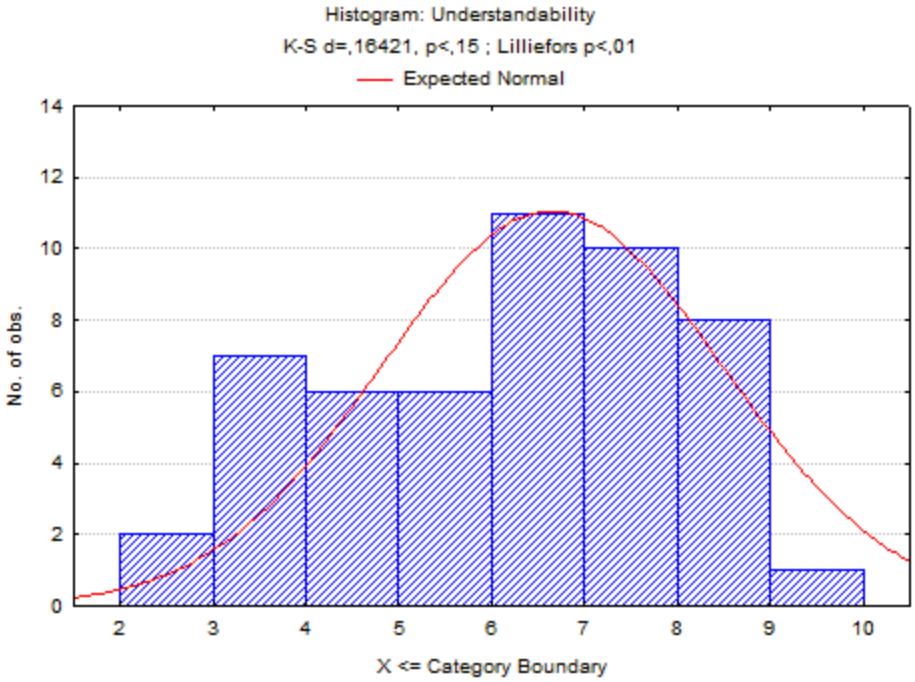

Fig. 11. Understandability Histogram

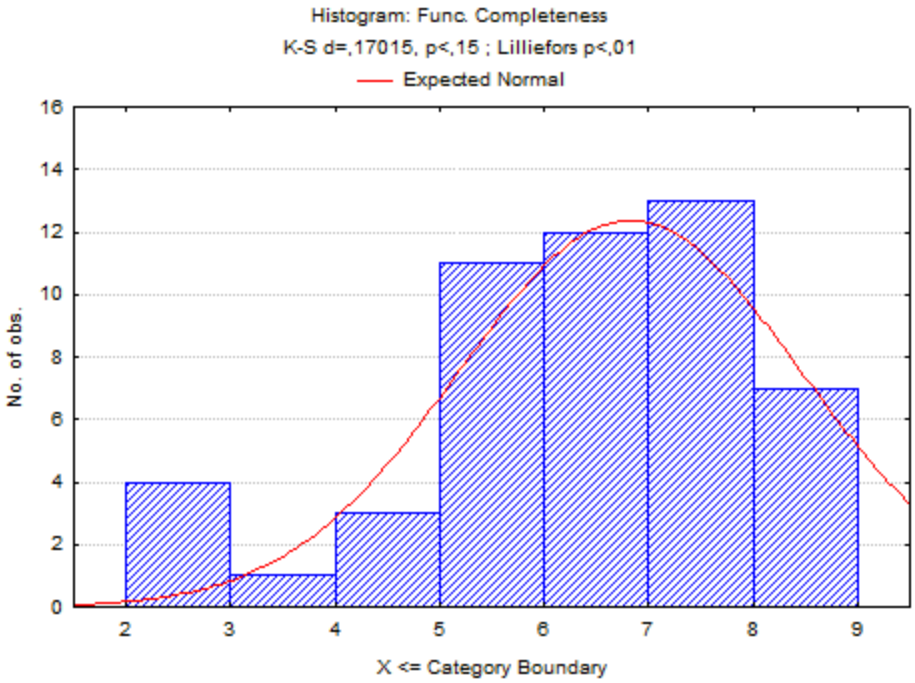

Fig. 12. Functional Completeness Histogram

3.4 Statistical Characteristics for Direct and Indirect Metrics and Expert Estimations

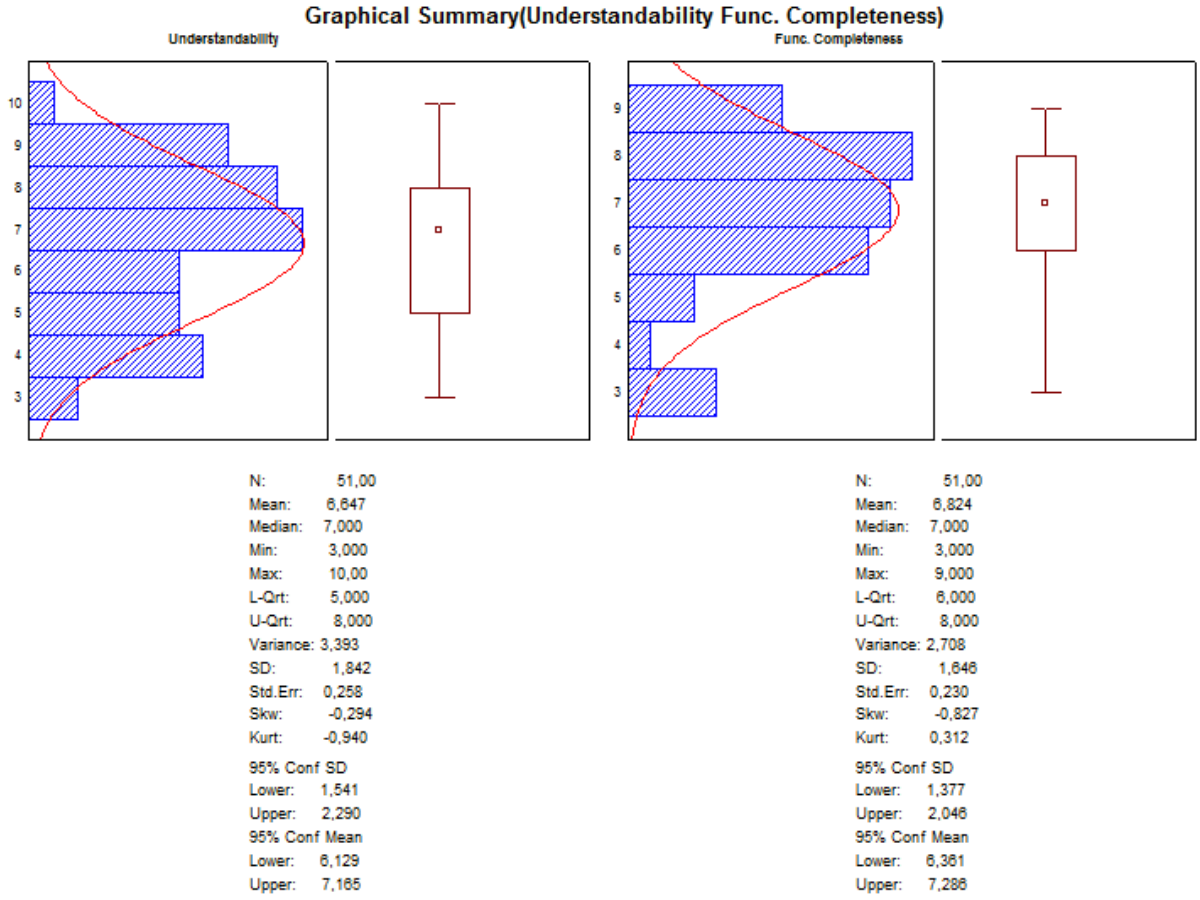

Fig. 15. Statistical Characteristics for Expert Estimations

4. Correlation Analysis

After the correlation analysis is ended we can make some conclusions about which metrics and expert estimations are much coupled and which are not. Analyzing correlation coefficient and Spearman coefficient, I chose several meanings that have most tight relation (in tables as they are marked with red color).

4.1 Metrics with normal distribution

Table 12. Correlation coefficient for pair "Direct – Undirect metric"

Metric	TCC	WOC
HIT	0,57	0,19

Table 13. Correlation coefficient for pair "Direct metric – Expert Estimation"

Metric	Understandability	Functional Completeness
HIT	-0,25	0,02

Table 14. Correlation coefficient for pair "Indirect metric – Expert Estimation"

Metric	Understandability	Functional Completeness
TCC	0,03	-0,05
WOC	-0,03	0,04

4.2 Metrics with not normal distribution

Table 15. Spearman coefficient for pair "Direct – Undirect metric"

Metric	AMW	ATFD	BOvR
LOC	0,005153	-0,008680	0,009386
NOM	-0,005329	-0,012329	0,006565
NDD	0,003027	0,018583	0,048823
FOUT	0,026730	-0,003393	0,103947

Table 16. Spearman coefficient for pair "Direct metric – Expert Estimation"

Metric	Understandability	Functional Completeness
LOC	0,371844	0,112789
NOM	0,085385	-0,059804
NDD	-0,171549	0,064373
FOUT	0,049901	-0,108620

Table 17. Spearman coefficient for pair "Indirect metric – Expert Estimation"

Metric	Understandability	Functional Completeness
AMW	0,254272	0,043496
ATFD	0,060987	-0,058513
BOvR	0,519108	0,577400

5. Regression Analysis

Regression analysis is the last stage in the study on the dependence of metrics and expert estimates. It is carried out only under the condition that the variance of the dependent variable (expert assessment) must remain constant when changing the value of the argument (metric), i.e. first, the variance of the expert assessment is determined for each accepted value of the metric. Next, the regression is identified. It involves both graphical construction and analytical research. Graphical construction begins with the definition of the correlation field. If the correlation field is elliptical, a linear regression relationship is inferred. Next, the construction of linear regression and its evaluation. If the constructed points of the correlation field fall into a circle, it is concluded that there is no dependence. If the correlation field does not fit into a circle or ellipse but has a different form, then a conclusion is made about the nonlinear dependence in the regression line [7].

5.1 Regression Lines for metrics with normal distribution

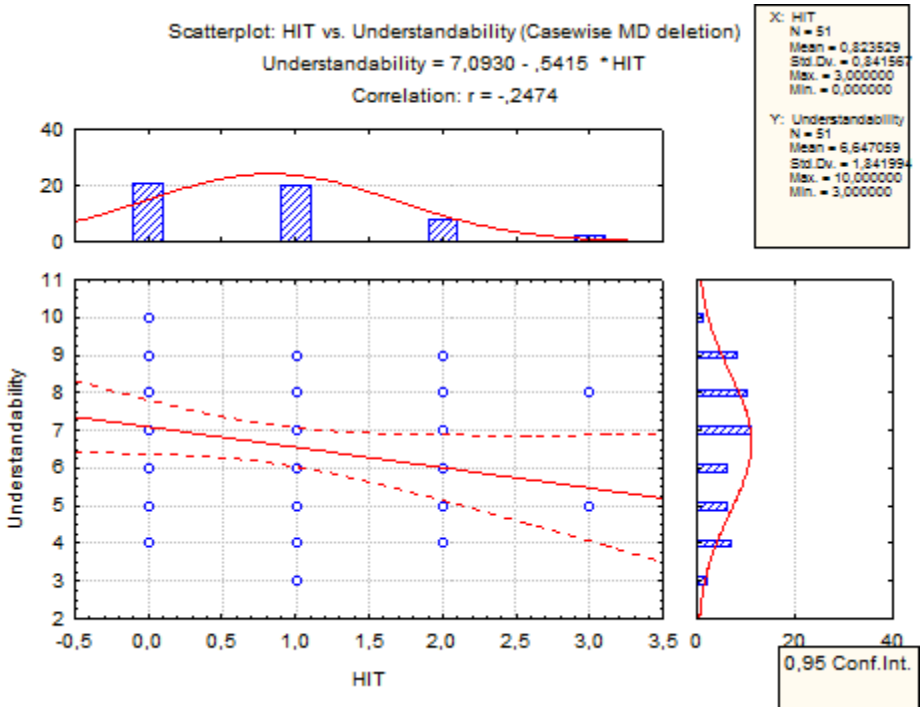

Fig. 16. Regression Line: HIT – Understandability

5.2 Regression Lines for metrics with unnormal distribution

- **LOC – Understandability**

Table 18. Standard Deviation Average Values for LOC – Understandability

Graph Type	Standard Deviation Average Value
Linear	0,0002557373
Polynomial	0,0002583627
Logarithmic	0,0002611963
Exponential	0,0001309233

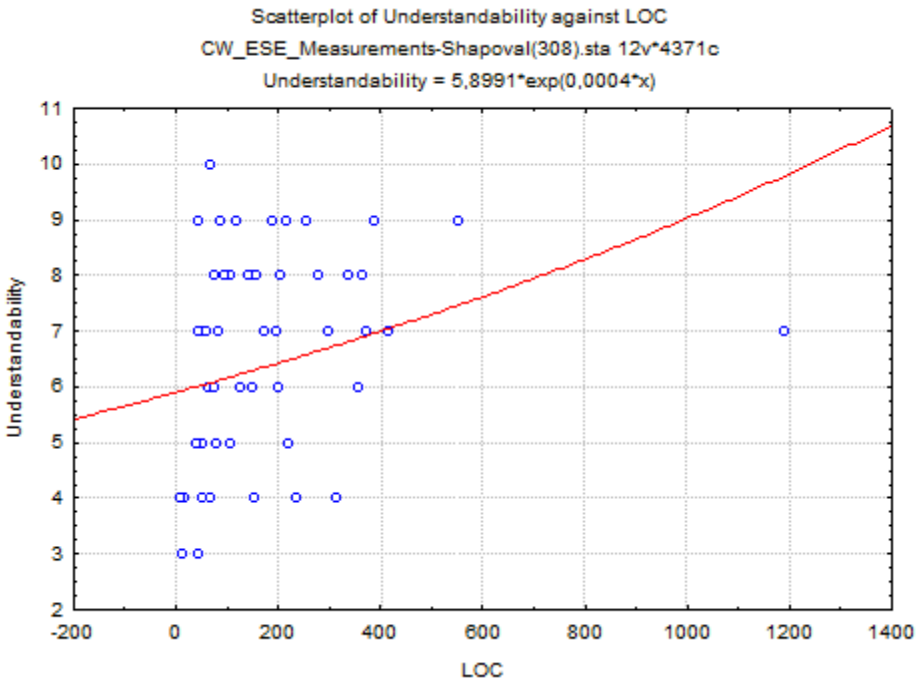

Fig. 17. Regression Line: LOC – Understandability

- **BOvR – Understandability**

Table 19. Standard Deviation Average Values for BOvR – Understandability

Graph Type	Standard Deviation Average Value
Linear	0,0000428552
Polynomial	0,0001159300
Logarithmic	0,0014776881
Exponential	0,0000317471

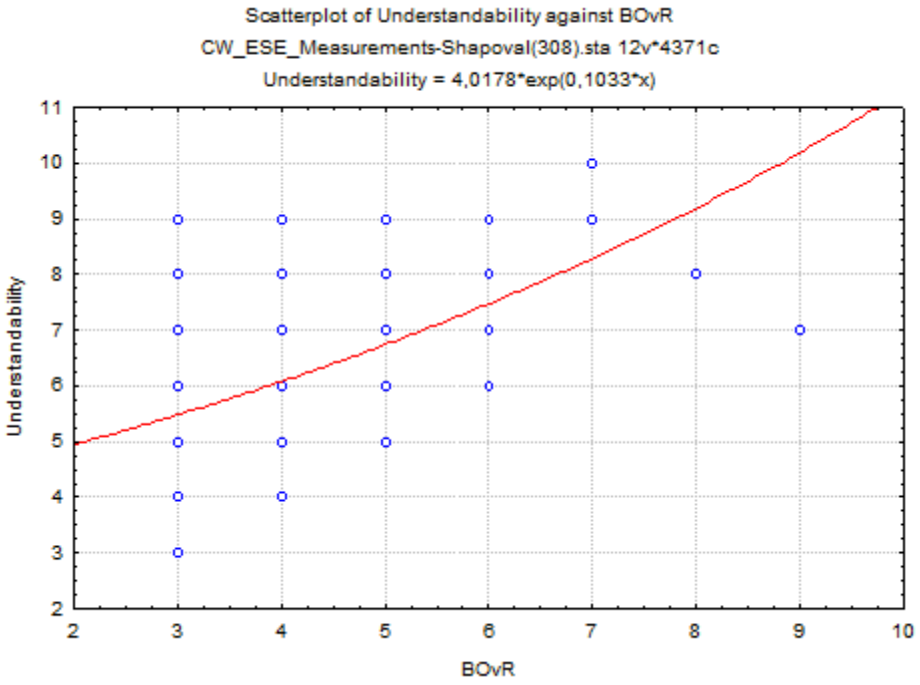

Fig. 18. Regression Line: BOvR – Understandability

- **BOvR – Functional Completeness**

Table 20. Standard Deviation Average Values for BOvR – Functional Completeness

Graph Type	Standard Deviation Average Value
Linear	**0,0000415606**
Polynomial	**0,0001191108**
Logarithmic	**0,0014452261**
Exponential	**0,0000322470**

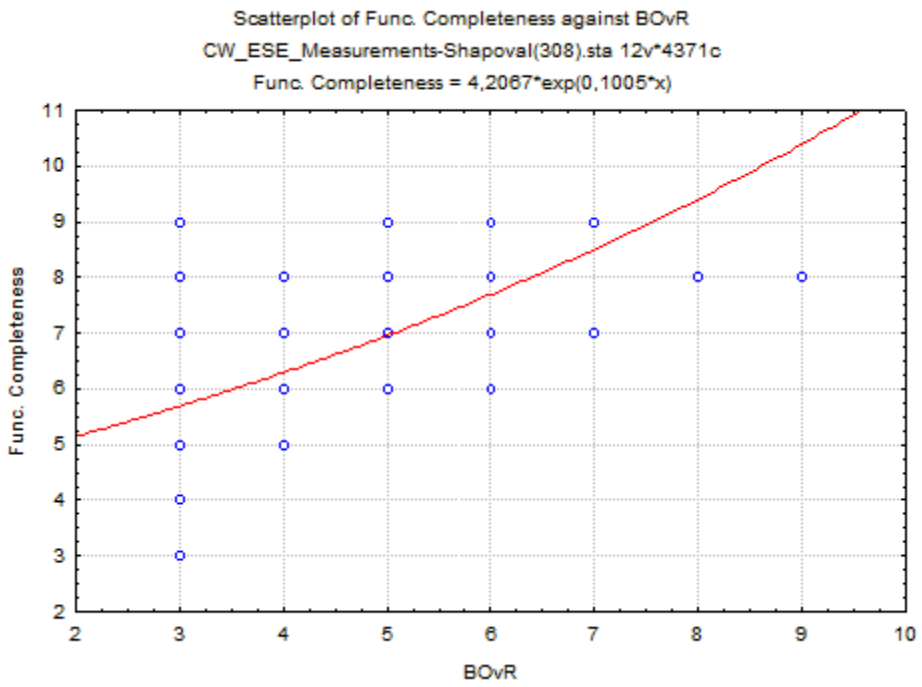

Fig. 19. Regression Line: BOvR – Functional Completeness

After the regression analysis is ended, there can be made some conclusions about which metrics and expert estimations are the most coupled. Analyzing regression lines for metrics of normal and unnormal distributions, I proved the connection between pairs of ‘metric – expert estimation’.

6. Summary

Empirical software engineering is a set of actions to gain knowledge to better understand aspects of software development. The result of the action is several statements about a certain list of problems. These statements are answers to questions and confirm or refute hypotheses.
Identify four main areas of empirical research:

- Research related to the technical part (research of the development environment, research of the used software methods in development, research of the software product itself);
- Research of development processes of each developer individually (everything that the developer does);
- Research of possibilities of integration of software products made by developers;
- Study of interaction, coordination of the development team (for example, whether the team works as a single mechanism or simply as separate developers who interact at some related stages).

As the results of the course work, there can be named tuples of ‘metric – expert estimation’ that are most coupled: HIT – Understandability (for normal distribution – linear dependence), LOC – Understandability, BOvR – Understandability, BOvR – Functional (for unnormal distribution – nonlinear dependence).

The practical results of this work can be applied for software measurements to analyze what changes in the code (affecting given metric) will cause increasing or decreasing of what software property.

7. References

1. Michele Lanza, Radu Marinescu. Object-Oriented Metrics in Practice. - Springer-Verlag Berlin Heidelberg, 2006.
2. Forrest Shull, Janice Singer, Dag I.K. Sjoberg Guide to Advanced Empirical Software Engineering. – Springer-Verlag London Limited 2008.-394p.
3. Norman E. Fenton, Shari Lawrence Pfleeger Software Metrics: A Rigorous and Practical Approach.- Cambridge University Press,1996.-638p.
4. Ian Sommerville. Software Engineering. - Pearson Education Limited, 2016.
5. Christof Ebert, Reiner Dumke, Manfred Bundschuh, Andreas Schmietendorf Best Practices in Software Measurement: how to use metrics to improve project and process performance. Springer-Verlag Berlin Heidelberg, 2005.- 295 p.
6. Software Measurement and Estimation: a practical approach / Linda M. Laird, M. Carol Brennan. John Wiley & Sons, Inc., Hoboken, New Jersey 2006.- 257 p.
7. Stephen H. Kan Metrics and Models in Software Quality Engineering, Second Edition. – Addison Wesley 2002.-560 p.
8. Tom Mens, Serge Demeyer. Software Evolution. – Springer-Verlag Berlin Heidelberg, 2008.-347p.
9. Ford, Neal. “Evolutionary architecture and emergent design: Emergent design through metrics.” *IBM Developer,* IBM, March 04, 2010, https://www.ibm.com/developerworks/library/j-eaed6/
10. “Source Lines of Code.” Academik, https://en.academic.ru/dic.nsf/enwiki/178453