МАТЕРІАЛИ МІЖНАРОДНОЇ НАУКОВО-ПРАКТИЧНОЇ КОНФЕРЕНЦІЇ СТУДЕНТІВ І АСПІРАНТІВ «ІНЖЕНЕРІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 2012»

УДК 004.415.2.045 (076.5)

O.P. Dyshlevyy, K.V. Shaposhnikova
National Aviation University

TOOL OF DEFINING MINIMAL SCOPE OF MEASUREMENTS FOR MAINTAINABILITY ANALYSIS

The article is dedicated to defining of minimal scope of measurement necessary to perform objective maintainability analysis. Basic statistical characteristics of initial statistical analysis have been calculated for maintainability metric, such as: mathematical expectation, median, kurtosis and skewness; distribution law has been determined. Technique and tool of defining minimal scope of metrics measurements have been developed for maintainability analysis. Investigation results of several open source projects have been presented.

Стаття присвячена визначенню мінімального обсягу вимірювань, необхідного для проведення об'єктивного аналізу супроводження. Розраховувалися базові статистичні характеристики первинного статистичного аналізу для метрики супроводження:математичне сподівання,медіана, коефіцієнти ексцесу та симетрії; визначався закон розподілу метрики. Була розроблена методика та засіб визначення мінімального обсягу вимірювань метрик для аналізу супроводження. Представлені результати досліджень декількох відкритих проектів.

Статья посвящена определению минимального объема измерений, необходимого для проведения объективного анализа сопровождаемости. Рассчитывались базовые статистические характеристики первичного статистического анализа для метрики сопровождаемости: математическое ожидание, медиана, коэффициенты эксцеса и ассиметрии; определялся закон распределения метрики. Была разработана методика и средство определения минимального объема измерений метрик для анализа сопровождаемости. Представлены результаты исследований нескольких открытых проектов.

Key words: Metrics, Maintainability, Measurement, Statistical analysis, Distribution, Expected value

Introduction

Software metric is a quantitative measure of the degree to which a system, component, or process possesses a given attribute [1]. Importance of metrics can't be overestimated. Different aspects of software are measured using appropriate set of metrics [2].

One of the aspects is software maintainability. IEEE Standard Glossary of Software Engineering Terminology defines maintainability as "the ease with which a software system or component can be modified to correct faults, improve performance or other attributes, or adapt to a changed environment." [1]. It is a very critical property of many developed systems. Maintainability is measured as a dependence of other metrics [3]. It can be calculated after changes are made to software system. Then conclusions are made, if these changes improved maintainability or not [4]. Relevant question appears, whether such conclusions can be considered objective and truly reflect the result of changes. May be there is not enough scope of measurements.

The basis on which such scope can be defined is formulated in the law of large numbers. It states that the average of the results obtained from a large number of trials should be close to the expected value, and will tend to become closer as more trials are performed [5].

Last researches overview

Maintainability as one the most important software attributes is constantly studied. Software engineers try to design better models to asses it, conduct experiments to reveal some trends of its values [3, 4, 6-8].

Done Coleman and Dan Ash from "Hewlett-Packard", Bruce Lowther from "Micron Semiconductor" and Paul Oman from University of Idaho conducted valuable research in the area of maintainability analysis. Their intent was to demonstrate how automated software maintainability analysis can be used to guide software-related decision making. [4]. Rikard Land from Malardalen University investigated how maintainability of a piece of software changes as time passes and it is being maintained by performing measurements on industrial systems. [3]. Pfleeger describes maintainability as the probability that a maintenance activity can be carried out within a stated time interval, it ranges from 0 to 1. [9]. Y. Kataoka, T. Imai .H. Andou T. Fukaya discussed program refactoring as a technique to enhance the maintainability of a program. They proposed a quantitative evaluation method to measure the maintainability enhancement effect of program refactoring. [6]

The described above research proposed different models for maintainability assessment, thresholds to make conclusions about obtained results, investigated how maintainability changes in time and how it varies depending on different changes make in software. All this researches deal with maintainability explanation and interpretation, and don’t pay attention to analyzed data scope. It is important to know how many input data software researchers must use for making conclusion on maintainability analysis.

Paper objectives

Main objectives of paper are:

- analysis existing problems in maintainability estimation, especially minimal scope of measurements definition;
- analysis of statistical methods for minimal data scope defining;
- development of a software tool that solves the minimal measurements scope problem;
- case studies of software tool.

Minimal data scope defining technique

To solve the problem of minimal scope determining special technique was designed. It implies the processing of different projects and definition of minimal scope for each concrete one. The average value obtained from the experiments results can be assumed as minimal scope of measurements for projects that a like those under research.

The general algorithm consists of several steps (Fig. 1).

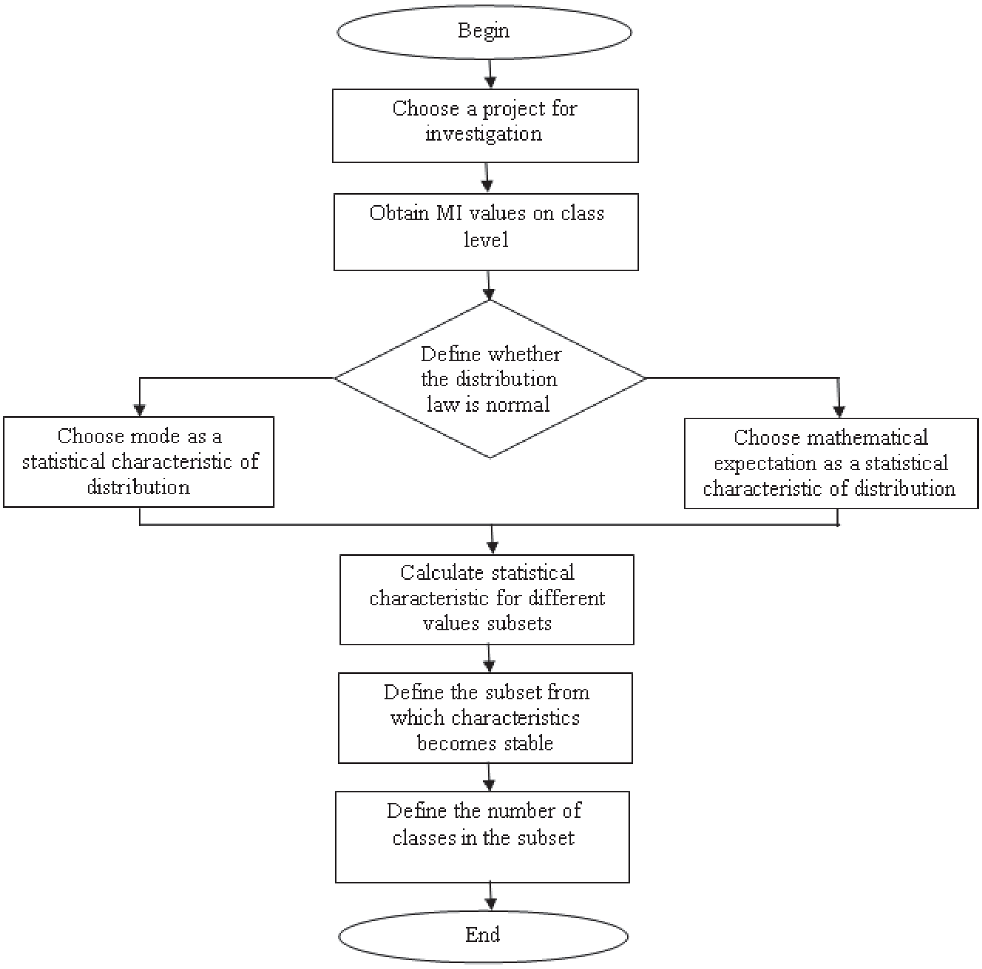
Begin
Choose a project for investigation
Obtain MI values on class level
Define whether the distribution law is normal
Choose mode as a statistical characteristic of distribution
Choose mathematical expectation as a statistical characteristic of distribution
Calculate statistical characteristic for different values subsets
Define the subset from which characteristics becomes stable
Define the number of classes in the subset
End

Fig.1 Algorithm scheme

First, it is necessary to choose a project that is going to be investigated. MI values for each class of chosen project will be input data for the algorithm. These values represent the distribution. The next step is to define whether the distribution is normal or not. Depending on the result statistical characteristic is defined for further research [5]. In case of normal distribution ME (Mathematical expectation) is chosen, otherwise – mode. Subsets of values are formed as shown on Fig.2.

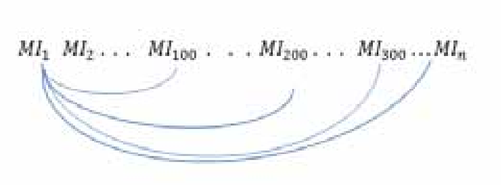
MI_1 MI_2 . . . MI_{100} . . . MI_{200} . . . MI_{300} ...MI_n

Fig.2 Division into subsets

For each subset of values, the statistical characteristic is calculated. As a result for each subset of values (each value corresponds to definite class) ME or mode is obtained. The subset starting from which this characteristic remains relatively stable can be considered as minimal scope for maintainability analysis.

The input data for the designed algorithm is array of MI values for concrete project on the class level. Important requirement to project is that it must be rather big (more than several thousands of classes). Otherwise the calculations based on the LLN won't have any sense. All the values in input array have constraints on possible values as maintainability index can have values in range from 0 to 100.

Distribution law determining

As it was mentioned above one of the algorithm steps is to define is the distribution normal or not.

The normal distribution is the most widely known and used of all distributions [5, 10]. Fig.3 shows the example of values distributed normally.

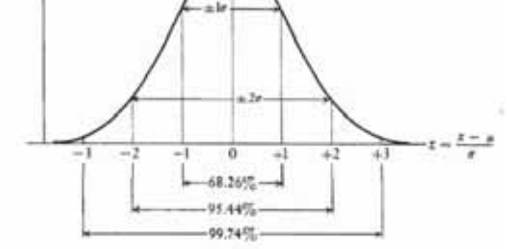

Fig.3 Normal distribution

To define whether the distribution is normal or not a histogram of the sample was built.

Except visual estimation mathematical calculations must be applied to make conclusion about the distribution law. The basic criteria by which distribution law can be obtained are kurtosis and skewness coefficients [10]. In current investigation the distribution is considered normal if both coefficients are less than 0.3.

In the subject domain of maintainability almost all the projects will not have normal distribution of MI values. One of the reasons is that possible values have described earlier definite range and most of values lie from approximately 80 to 100. Conducted investigations on variety of projects with several thousands of classes confirmed this assumption.

Statistical characteristics

The next step after distribution law defined is to make decision about statistical characteristic that is going to be used for further calculations. Two of such characteristics that can represent the average value of distribution: ME and mode [5, 10]. But for the same set of numbers all these parameters can have absolutely different values. To design the algorithm for minimal measurements scope definition it was important to decide what characteristics to choose depending on the character of distribution. Below the description of ME and mode, their comparison can be found. Except these two ones other statistical values exists, for example median. For this algorithm it was decided not to consider it because as shown by investigation for such distribution as MI values it doesn't give objective results. If the distribution law is normal it is better to use ME, otherwise – mode.

Software tool development

To implement the algorithm before special software tool was developed. Then aim of the tool is to assist in research of minimal measurement scope analysis. The logic of the application implements all necessary statistical calculations on array of input metric values. This tool is not bounded to estimation of MI.

The following use case diagram on the Fig. 4 reflects the basic functionality of system, describes what opportunities user has while using a system.

Class diagrams are the mainstay of object-oriented analysis and design. UML 2 class diagrams show the classes of the system, their interrelationships and the operations and attributes of the classes (Fig.5).

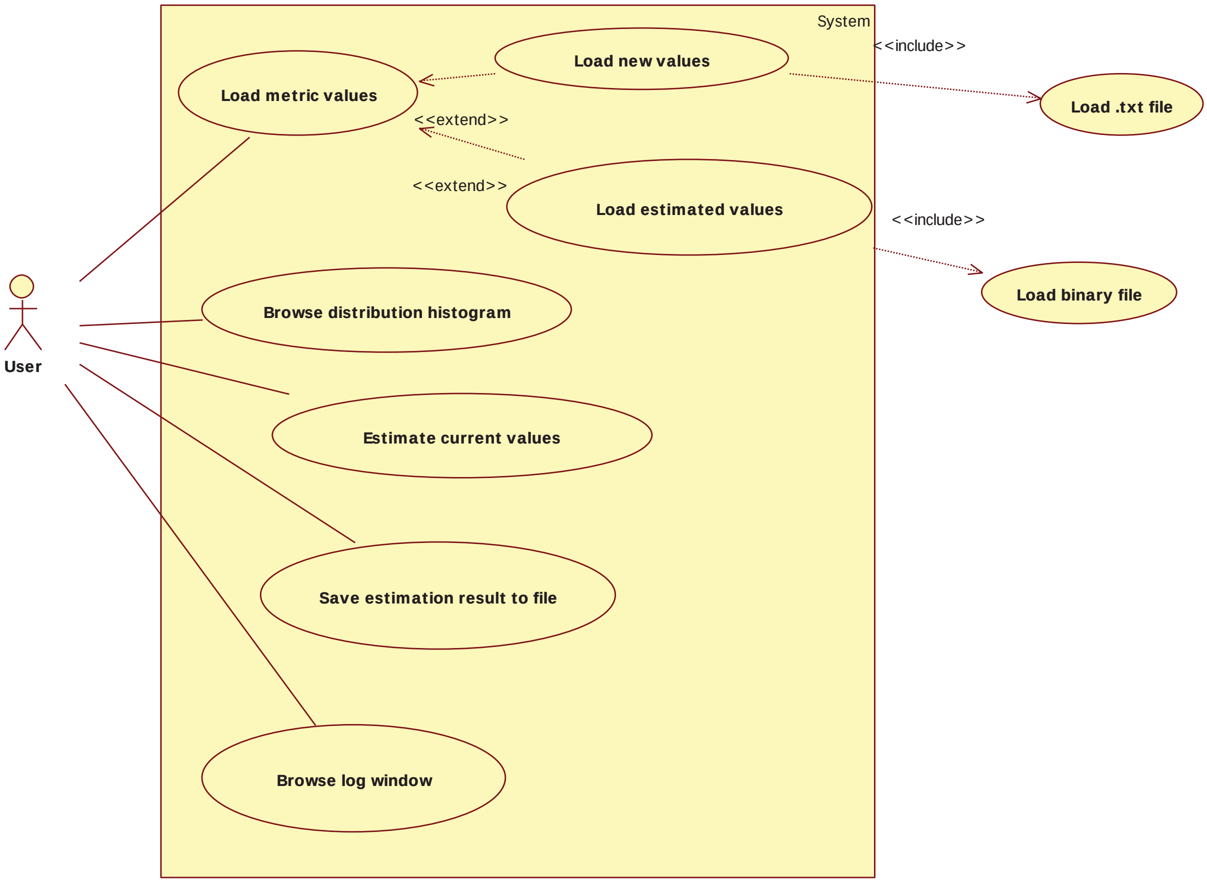

Fig.4 Use case diagram

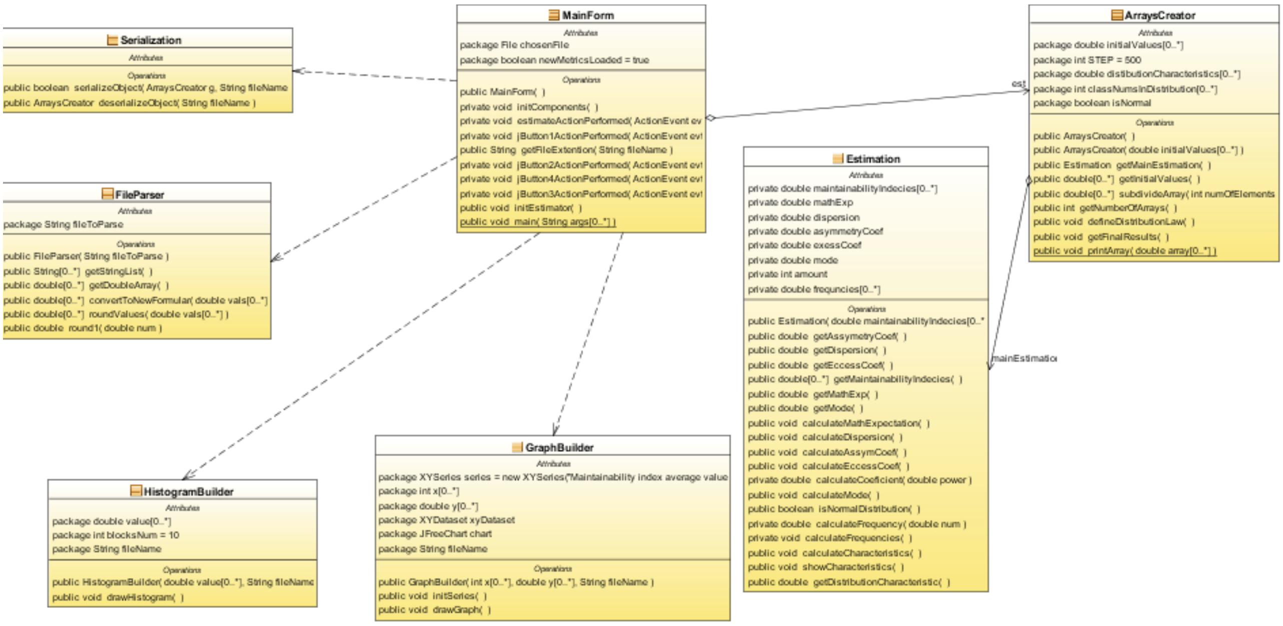
Serialization
Attributes
Operations
public boolean serializeObject(ArraysCreator g, String fileName
public ArraysCreator deserializeObject(String fileName)
MainForm
Attributes
package File chosenFile
package boolean newMetricsLoaded = true
Operations
public MainForm( )
private void initComponents( )
private void estimateActionPerformed(ActionEvent ev
private void jButton1ActionPerformed(ActionEvent evt
public String getFileExtention(String fileName)
private void jButton2ActionPerformed(ActionEvent evt
private void jButton4ActionPerformed(ActionEvent evt
private void jButton3ActionPerformed(ActionEvent evt
public void initEstimator( )
public void main(String args[0..*])
ArraysCreator
Attributes
package double initialValues[0..*]
package int STEP = 500
package double distibutionCharacteristics[0..*]
package int classNumsInDistribution[0..*]
package boolean isNormal
Operations
public ArraysCreator( )
public ArraysCreator(double initialValues[0..*])
public Estimation getMainEstimation( )
public double[0..*] getInitialValues( )
public double[0..*] subdivideArray(int numOfElements
public int getNumberOfArrays( )
public void defineDistributionLaw( )
public void getFinalResults( )
public void printArray(double array[0..*])
est
FileParser
Attributes
package String fileToParse
Operations
public FileParser(String fileToParse)
public String[0..*] getStringList( )
public double[0..*] getDoubleArray( )
public double[0..*] convertToNewFormular(double vals[0..*]
public double[0..*] roundValues(double vals[0..*])
public double round1(double num)
Estimation
Attributes
private double maintainabilityIndecies[0..*]
private double mathExp
private double dispersion
private double asymmetryCoef
private double exessCoef
private double mode
private int amount
private double frequncies[0..*]
Operations
public Estimation(double maintainabilityIndecies[0..*
public double getAssymetryCoef( )
public double getDispersion( )
public double getEccessCoef( )
public double[0..*] getMaintainabilityIndecies( )
public double getMathExp( )
public double getMode( )
public void calculateMathExpectation( )
public void calculateDispersion( )
public void calculateAssymCoef( )
public void calculateEccessCoef( )
private double calculateCoeficient(double power)
public void calculateMode( )
public boolean isNormalDistribution( )
private double calculateFrequency(double num)
private void calculateFrequencies( )
public void calculateCharacteristics( )
public void showCharacteristics( )
public double getDistributionCharacteristic( )
mainEstimatio
HistogramBuilder
Attributes
package double value[0..*]
package int blocksNum = 10
package String fileName
Operations
public HistogramBuilder(double value[0..*], String fileName
public void drawHistogram( )
GraphBuilder
Attributes
package XYSeries series = new XYSeries("Maintainability index average value
package int x[0..*]
package double y[0..*]
package XYDataset xyDataset
package JFreeChart chart
package String fileName
Operations
public GraphBuilder(int x[0..*], double y[0..*], String fileName)
public void initSeries( )
public void drawGraph( )

Fig.5 Class diagram

Software tool is implemented using Java programming language in NetBeans 7.0 development environment. Several libraries were used for development. GUI was created via Javax.Swing. JFreeChart library was used to build result chart and distribution histogram.

Case studies

The developed software tool can take as input different projects of large size and shows the project scope which is enough for maintainability analysis. Making conclusion about obtained minimal scope values can help to define the average one that can be further used by developers.

Numbers of projects were studied to find the average values of minimal measurements scope for maintainability analysis. The main criterion of choosing the projects for investigations was large number of classes. Otherwise there is no sense to investigate the project because the LLN won't be applicable to it. It was chosen several projects: AgroUML 3.0.4, Apache Tomcat 7.0.25, JBoss (partly), NetBeans 7.0 (partly). Last two projects were investigated partly because of problems in measurement.

The first step is to obtain MI values on class level for project. To do this Semantic Designs JavaMetrics tool was used. Then it is necessary to load measurement results in developed tool, and do investigations. As it can be seen from Fig.6 starting from 2500 classes the value of mode remains relatively stable for AgroUML 3.0.4.

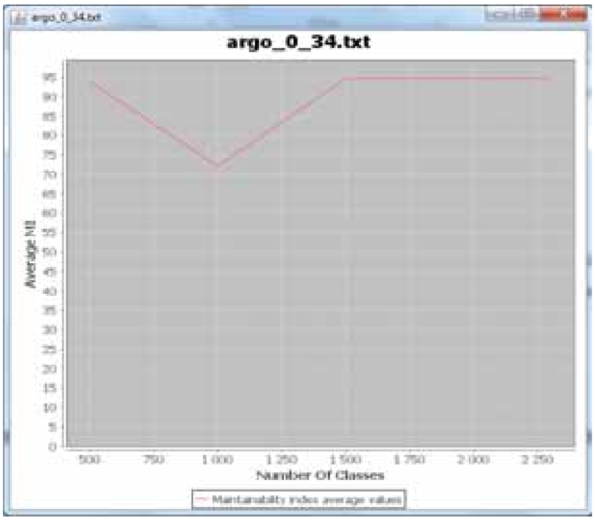

Fig.6 Investigation of AgroUML 3.0.4

It means that maintainability analysis can be performed with this project and increasing or decreasing MI values will correspond to reality. This result can be also fixed as the minimal scope for measurements for some project. Then it is going to be compared with results obtain from other projects. All results are represented in table 1.

Table 1

Experiments results

№	Project name	Class num.	Min. scope
1	AgroUML 3.0.4	2500	1500
2	Apache Tomcat 7.0.25	2000	1000
3	JBoss (partly)	3000	1000
4	NetBeans 7.0 (partly)	5000	1300

Conclusions

The developed software tool assists in conducting research of minimal measurements scope in maintainability analysis. The main idea of the tool is LLN. Using this law and statistical formulas it computes the dependency of project scope and average MI value. A researcher can visually define the point from which the average value becomes stable. This value is minimal measurements scope.

After conducting this research the conclusion can be made that about 1000-1500 classes must be present is a project to objectively estimate its maintainability. Only having such measurements scope, the changes in projects can be reflected truly after MI comparison. Result of conducted investigations can be applied by developers when they want to analyze maintainability. The software tool also can help to understand whether the project has enough classes to perform relevant analysis. This tool is not bounded to estimation of MI. It can be further extended to work with other metrics.

References

1. IEEE Std 610.12-1990, IEEE *Standard Glossary of Software Engineering Terminology*, identifies terms currently in use in the field of Software Engineering. Standard definitions for those terms are established. – The Institute of Electrical and Electronics Engineering, New York, 1990.

2. *Michele Lanza, Radu Marinescu*. Obect-oriented metrics in practice: Using software metrics to characterize, evaluate, and improve the design of object-oriented systems. – Springer- Verlag Berlin Limited 2006.-205p.

3. *Rikard Land*. Measurements of software maintainability. Malardalen University Department of Computer Engineering, Box 883, SE-721 23 Vasteras, Sweden.

4. *Don Coleman*, *Dan Ash*. Using metrics to evaluate software system maintainability. Journal Computer, IEEE Computer Society Press Los

Alamitos, CA, USA. Volume 27 Issue 8, August 1994– 44-49 pp.

5. *Вентцель Е.С.* Теория вероятностей: Учеб. для вузов. – 7-е изд. стер. – М.: Высш. шк., 2001. – 575 с.: ил.

6. *N. Narayanan Prasanth, S.P. Raja, X. Birla, K. Navaz, S. Arif Abdul Rahuman*. Improving Software Maintainability through Rist Analysis. International Journal of Recent Trends in Engineering, Vol 2, No.4, November 2009. – 198-200 pp.

7. *Melis Dagpinar, Jens H.Jahnke*. Predicting maintainability with object-oriented metrics – an empirical comparison. Department of computer science university of Victoria, Canada B.C. – 2003.

8. *P.K. Suri, Bharat Bhushan*. Simulator for Software Maintainability. IJCSNS International Journal of Computer Science and Network Security. VOL7 No.11, November 2007. – 216-220 pp.

9. *Norman E. Fenton, Shari Lawrence Pfleeger* Software Metrics: A Rigorous and Practical Approach.- Cambridge University Press,1996.-638p.

Бабак В.П., Білецький А.Я., Приставка О.П., Приставка П.О. Статистична обробка даних/ Монографія. – Київ: «МІВВЦ», 2001. – 388 с.

Відомості про авторів:

Дишлевий Олексій Петрович

старший викладач кафедри інженерії програмного забезпечення Факультету комп’ютерних наук, Національний авіаційний університет.

E-mail: oleksiy.dyshlevyy@livenau.net

Шапошнікова Ксенія Володимирівна

студентка 5 курсу Факультету комп’ютерних наук (магістр), кафедра інженерії програмного забезпечення, Національний авіаційний університет.

E-mail: shaposhnikovakv@gmail.com