American Journal of Software Engineering and Applications
2013; 2(2) : 25-31
Published online April 2, 2013 (http://www.sciencepublishinggroup.com/j/ajsea)
doi: 10.11648/j. ajsea.20130202.11

Intelligent assessment and prediction of software characteristics at the design stage

Oksana Pomorova, Tetyana Hovorushchenko

Department of System Programming, Khmelnitskiy National University, Khmelnitskiy, Ukraine

Email address:
o.pomorova@gmail.com (O. Pomorova), tat_yana@ukr.net (T. Hovorushchenko)

To cite this article:
Oksana Pomorova, Tetyana Hovorushchenko. Intelligent Assessment and Prediction of Software Characteristics at the Design Stage, *American Journal of Software Engineering and Applications.* Vol. 2, No. 2, 2013, pp. 25-31. doi: 10.11648/j.ajsea.20130202.11

Abstract: This article is dedicated to intelligent method and system of design results evaluation and software characteristics prediction on the basis of processing of software metrics sets.

Keywords: Software Complexity, Software Quality, Software Metrics, Artificial Neural Network (Ann), Neural Method For Design Results Evaluating And Software Characteristics Prediction, Intelligent System Of Assessment And Prediction Of Software Characteristics

1. Introduction

The development of software is the knowledge-based activity that requires a detailed study of the subject area and a full understanding of the developed product goals. The characteristics of software include: software cost, software protection, completeness of requirements realization, the size of software files, requirements to system software and hardware, the size of required RAM and disk storage. But the most important characteristics of the software are its complexity from the developer's perspective and its quality from a user perspective.

The crisis in the software quality providing was noticeable more than 50 years ago. Since then, many methods, techniques and tools have been developed, the best specialists were involved in the development of technologies and standards to ensure the software systems quality. But the quality of software is still dependent on the knowledge and experience of developers.

According to approximate estimates the cost of software development is about 275 billion dollars, but only 72% of projects reach the implementation stage and only 26% of projects are completed successfully [1]. According to The Wall Street Journal, more than 50% of corporate software projects fall short of expectations and over 42% of projects are terminated long before their logical completion [2].

One of the criteria for the success of software projects is their complexity, and hence cost. Research of Standish Group showed that projects costing less than 750 thousand dollars are successful in 55% cases, projects costing from 1 to 2 million dollars are successful in 18% cases, projects costing from 5 to 10 million doolars are successful only in 7% cases [2]. The permanent growing of software functions complexity inevitably leads to increasing of their size and creation complexity. Modern software with millions of lines of code in principle can not be infallible, faultless and accurate, so the problem of achieving the required level of quality is actual.

1.1. Results of Manifestation of Software Quality Level in Complicated Hardware-Software Complexes

Thus incidents caused by software failures continue to appear. The most known incidents for the severity of its consequences are:

1) Six patients received overdoses of radiation during sessions of radiation therapy with radiation therapy machine Therac-25 in 1985-1987. Catastrophic consequences of programming bugs and defects in the project development and formulation manifestated repeatedly and for a long time [3];

2) The subject of European Community pride, rocket Ariane 5 self-destructing 37 seconds after launch because of a malfunction in the control software (1996). This explosion led to huge losses - only scientific equipment on it was worth half a billion dollars plus astronomical "profits" from not occurred commercial launches [4];

3) The Space Shuttle Columbia disaster occurred in 2003, resulting in the death of all seven crew members. The fire was happened because the report on plating damage was incorrectly prepared in the program MS PowerPoint [5];

4) While attempting its first overseas deployment to the Kadena Air Base in Okinawa, Japan, in 2007, twelve fighter stealth aircrafts F-22s flying from Hickam AFB, Hawaii experienced multiple computer failures while crossing the International Date Line (or 180th meridian of longitude dependent on software programming) [6];

5) The triple satellites, critical for the Russian navigation system Glonass (rival American GPS), fell into the Pacific Ocean in 2010. Loss of satellites caused by the programming bug is estimated 1387 million dollars [7];

6) On 1 February 2013, during the launch of Intelsat-27, a Russian rocket Zenit-3SL launch vehicle suffered a premature engine shutdown, as the rocket strayed from its lift-off trajectory, plunging into the Pacifc Ocean shortly after launch. Falling of rocket caused by failures in the control system [8].

In December 2011, NASA specialists successfully fix a bug in the software of onboard computer complex of spacecraft, which moved to Mars with rover "Curiosity" on board. But this example of software troubleshoot at the operation stage is very rare. Typically, these incidents lead to significant human and economic losses.

Therefore, the software quality requirements are necessary to formulate and test at the the early stages of software life cycle, preferably already at the design stage. The dependence of the cost of bugs correcting from the stage of the life cycle is shown on Fig.1 [9].

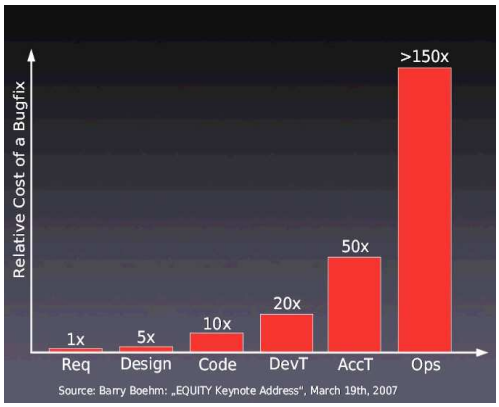

Figure 1. *Cost of a Bug Within a Software Lifecycle.*

Generally, according to known scientist in the software engineering Barry Boehm, software industry is nearing to technogenic catastrophe, which is caused by defective software that penetrates into all spheres of human activity.

While the complexity of some software development today already exceeds the complexity of many engineering and infrastructure projects, and the consequences of errors are catastrophic and disastrous, software engineering is not ensured with the fundamental theory and methodology.

If the software development industry will not radically changes, the world will inevitably not be able to avoid disasters caused by errors in the code or failures in the management of complex software systems.

1.2. Problems of Using of Software Characteristics Assessment Tools and Means

The series of software testing methodologies are developed to improve the software quality. They are based on the code reveiw. Review of the entire code is not possible due to the high laboriousness, so different methods of reducing the tested code size are used. For this purpose static code analyzers (PVS Studio [10], CAST Application Intelligence Platform [11], IBM Rational AppScan Source Edition [12], SofCheck Inspector [13], Visual Studio Team System [14]) are used. They on the basis of specialized rules and metrics allocate the code with low quality.

Software acquires high quality not so much a result of comprehensive testing of the final product, but in the during of its development. If bugs trapping at all project stages is the basis of methodology of creating software, the project will be almost infallible. IBM Corporation provides the methodology to creation of complex software systems - Cleanroom Software Engineering [15]. It provides teams of developers to plan, to measure, to specify, to design, to code, to test and to certify the software products. The tool for automated testing and software reliability evaluation in this methodology is environment Cleanroom Certification Assistant [15], which uses statistical test results to calculation of software reliability metrics by mathematical methods. The global market proposes many products for automation of metrics calculation: IBM Rational Logiscope [16], TestCenter [17], Rational Purify [18], IBM Rational Software Group [19].

The common lacks of these quality assessment tools are: 1) subjective dependence of choice of metrics that tool calculate; 2) subjective interpretation of metrics values, because exact (etalons) values of metrics are available; 3) tools of automation of metric information calculation are oriented on testing and metrology of finished source code and not oriented on prediction and calculation of software metrics at the design stage. At the design stage the finished product is missing, only informational, functional and behavioral model of requirements analysis for the software are. Consequently, the existing tools of software quality evaluation are ineffective at the design stage.

Software metrics can be useful for the evaluating of the project quality and complexity and the prediction of quality and complexity of the developed software by the project. The modern software industry has accumulated a large number of metrics that assess individual software features. However, the desire of their versatility, ignoring the type and scope of the developed software, ignoring the stages of the software life cycle and ungrounded their using significantly undermined the confidence of developers and users to software metrics. These circumstances require: the careful selection of metrics for a particular type and scope of the developed software, the considering their limitations at various stages of the life cycle, the establishing of possibilities and order of metrics sharing, the accumulation and integration of metrics to make timely production decisions.

Several unresolved issues is in the assessment and pre-

diction of software characteristics at the design stage: 1) quality measurement technology has not yet reached maturity - only 1,5% software companies are trying to evaluate the quality of processes and ready product using metrics, and only 0,5% software companies are trying to improve their work on the basis of quantitative criteria of software quality; 2) the lack of unified standards for metrics - over a thousand metrics were created, each developer of "measurement" system offers own measures of software quality evaluation and proper metrics; 3) difficult interpretation of metrics values - for most users, customers and programmers the metrics and their values are not informative; 4) metrics are calculated only for the finished software (for code); 5) low level of automation of the software metrics analysis and processing - only the processes of gathering, recording and computing of metric information are automated today.

The unresolved of these issues is one of the factors that interferes to develop the defect-free and high-quality software. Difficulty of grounding of software metrics selection and interpretation does not provide to use metrics for assessing and predicting of the software characteristics of at the design stage and for improving of the software quality.

The theory of software engineering is still missing. Theory is needed as a guide for researchers and developers. That theory would provide the selection of metrics for evaluation of the results at each stage of the life cycle and would provide the prediction of developed software characteristics. Certainly, there are some fundamental research (works of Boehm, Dijkstra, Meyer), but finished, tested and approbated theory is missing.

From the results of the analysis of software evaluation methods the conclusion follows, that the perspective research direction to improve the software quality is development of intelligent methods (IM) and systems (IS). IM and IS will analyze and process the design stage metrics, will evaluate the project and will provide the prediction of the characteristics of designed software.

2. Software Metrics at the Design Stage

At the design stage the number of requirements on software complexity and quality is important to provide: requirements on software structure, on navigation by software, on design of user interfaces, on multimedia components of software, on usability; technical requirements.

The answer to the question "How software system will be implemented its requirements?" is formed at this stage.

The information flows of the design stage: software requirements (information, functional and behavioral analysis models). Information model describes the information that must be processed by software according to customer. Functional model defines the list of functions and the list of modules of software system. Behavioral model captures the software modes. The results of the design stage are: developed data, developed architecture and procedural software development.

2.1. Software Complexity Metrics

From the analysis of software complexity metrics and information flows of the design stage the conclusion follows, that the following software complexity metrics with exact or predicted values appropriate to use at the design stage (table 1) [1].

Table 1. *Software complexity metrics at the design stage.*

№	Software Complexity Metrics With Exact Values at the Design Stage	Software Complexity Metrics With Predicted Values at the Design Stage
1	Chepin's metric	Expected Lines of Code (LOC)
2	Jilb's metric (absolute modular complexity)	Halstead's metric
3	McClure's metric	McCabe's metric
4	Kafur's metric	Jilb's metric (relative modular complexity)
5		Expected quantity of program statements
6		Expected estimate of interfaces complexity

So, at the design stage 10 basic software complexity metrics we will use - 4 of them have the exact values and the other 6 metrics have the predicted values.

2.2. Software Quality Metrics

The analysis of software quality metrics and information flows of the design stage provides the conclude that next software quality metrics with exact or predicted values are applicable at the design stage (table 2) [1].

Table 2. *Software quality metrics at the design stage.*

№	Software Quality Metrics With Exact Values at the Design Stage	Software Quality Metrics With Predicted Values at the Design Stage
1	2	3
1	Cohesion metric	Software design total time (in working days)
2	Coupling metric	Design stage time (in working days)
3	Metric of the global variables calling	Software realization productivity (in minutes for code line)
4	Time of models modification (in working days)	Software quality audit cost (in USD)
5	Quantity of found bugs in models	Software design cost (in USD)
1	2	3
6		Program code realization cost (in USD)
7		Functional points (FP)
8		Effort applied by Boehm (in man-months)
9		Development time by Boehm (in working days)

Therefore the 10 basic software complexity metrics and

14 basic software quality metrics were used at the design stage. Other metrics are derived from selected basic metrics.

The processing of above 24 software complexity and quality metrics with exact and predicted values is the basis for obtaining the evaluation of design results and prediction of software complexity and quality characteristics of the developed for the project software.

3. Neural Method for Design Results Evaluating and Software Characteristics Prediction (NMEP)

NMEP provides the evaluation of the project and prediction of designed software charateristics on the basis of complexity and quality metrics with exact and predicted values at the design stage, listed in the previous section.

NMEP is based on processing of the following sets:

1) the set of complexity metrics with the exact values at the design stage

$$CMEV = \{cmev_i \mid i = 1..4\};$$

2) the set of quality metrics with the exact values at the design stage

$$QMEV = \{qmev_j \mid j = 1..5\};$$

3) the set of complexity metrics with the predicted values at the design stage

$$CMPV = \{cmpv_k \mid k = 1..6\};$$

4) the set of complexity metrics with the predicted values at the design stage

$$QMPV = \{qmpv_n \mid n = 1..9\}.$$

The results of these sets processing are:

1) project complexity estimate PCE;
2) project quality evaluation PQE;
3) designed software complexity prediction SCP;
4) designed software quality prediction SQP.

The basis of project complexity estimate are elements of set $CMEV$. The basis of project quality evaluation are elements of sets $CMEV$ and $QMEV$. The basis of designed software complexity prediction are elements of set $CMPV$, but elements of sets $CMEV$ and $QMEV$ are taken into account. The basis of designed software quality prediction are elements of sets $CMPV$ and $QMPV$, but elements of sets $CMEV$ and $QMEV$ are taken into account.

The problem of identifying the correlation between metrics values and quality and complexity of project and software should be solved for the software quality and complexity evaluation and prediction on the basis of metric analysis results. One of the means, which makes it possible to summarize the information and identify dependencies between input data and resulting data, are artificial neural networks.

Metric analysis results we will to process using artificial neural network (ANN), that performs the approximation of software metrics and provides an estimate of complexity and quality of the project and predict of complexity and quality characteristics of developed for the project software.

NMEP consists of next stages:

1) the preparation of metrics with exact and predicted values at the design stage to the inputs of ANN;
2) the checking of obtained metrics values on the subject of exceeding of ANN inputs ranges limits;
3) the processing of metrics values by artificial neural network;
4) the analysis of ANN output values;
5) the forming of conclusion about the complexity and quality of project and designed software on the basis of ANN output values.

Input data for ANN are: the sets of complexity and quality metrics with the exact values at the design stage and the sets of complexity and quality metrics with the predicted values at the design stage.

The results of ANN functioning are 4 characteristics: project complexity estimate; project quality evaluation; designed software complexity prediction; designed software quality prediction.

NMEP concept is represented on Fig.2.

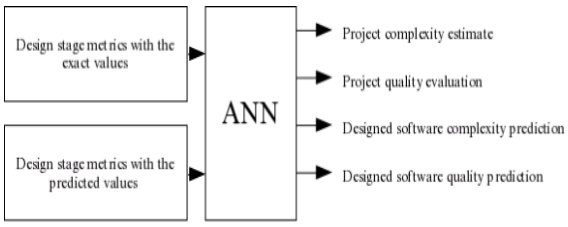

Figure 2. *NMEP conception.*

The multilayer perceptron was chosen as a result of analysis of the artificial neural networks architectures to analyze the software metrics at the design stage and to predict of software quality characteristics.

ANN has 9 one type inputs for the quantitative values of exact metrics at the design stage and 15 another type inputs for the quantitative values of predicted metrics at the design stage. If a certain metric is not determined, then -1 is given on the proper ANN input.

The conclusion about the project quality and complexity and the expected quality and complexity of designed software is based on an analysis of 4-th obtained results. Project complexity estimate, project quality evaluation, designed software complexity prediction, designed software quality prediction are values in the range [0, 1], where 0 - proper metric was not determined, approximately 0 - the project or designed software has a high complexity or low quality and 1 - the project or software is simple (non-complexity) or high quality.

The ANN has 24 neurons of the input layer, 14 neurons of approximating layer and 8 neurons of the adjusting layer and 4 neurons of the output layer.

ANN has 4 input vectors for giving the values of metrics:

1) Input1 vector consists of 4 elements - software complexity metrics with exact values at the design stage;

2) Input2 vector consists of 5 elements - software quality metrics with exact values at the design stage;

3) Input3 vector consists of 6 elements - software complexity metrics with predicted values at the design stage;

4) Input4 vector consists of 9 elements - software quality metrics with predicted values at the design stage.

Output vector Y consists of 4 elements: project complexity estimate, project quality evaluation, software complexity prediction, software quality prediction, therefore ANN has 4 neurons of the fourth (output) layer.

Architecture of neural network component of NMEP is shown on Fig.3.

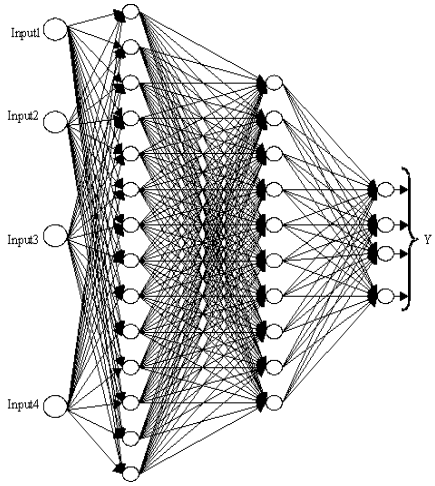

Figure 3. *Architecture of neural network component of NMEP.*

The described ANN has been implemented in Matlab [1].

Structural scheme of the ANN layers in Simulink is shown on Fig.4.

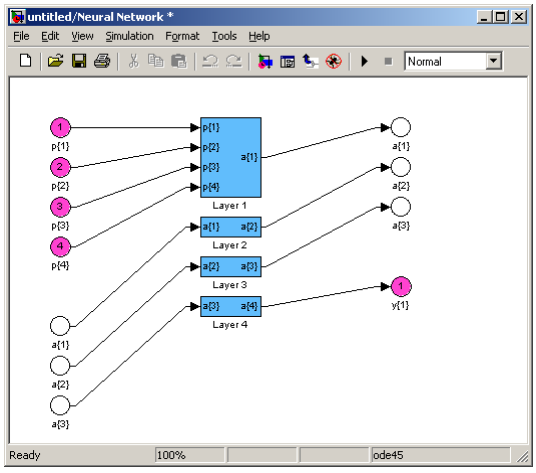

Figure 4. *ANN layers structural scheme in Simulink.*

The above dependences the resulting estimates of the input metrics sets are considered in training of the neural network.

Realized neural network was trained with training sample of 1935 vectors and tested with testing sample of 324 vectors. The research [1] demonstrates that the smallest training performance was obtained with the mean squared error w/reg performance function (msereg); optimal number of neurons in second hidden layer is 14 neurons. The ANN training and testing performance on average is approximately $\xi = 0{,}102197$.

4. Intelligent System of Assessment and Prediction of Software Characteristics (ISAP)

Usually when alternatives of the same software project are available, the choice of a particular version is performed by the criteria of cost and development time. The cost and development time may have similar or equal values, but significant differences in the quality of future software may exist. Therefore, when choosing of software project version, the number of other factors must be considered, including the software complexity and quality. The effective mean of this problem solving is intelligent system of assessment and prediction of software characteristics (ISAP) [1]. The basis for the development of this system is neural method for design results evaluating and software characteristics prediction (NMEP).

Quantitative exact and predicted values of metrics are given to the ISAP inputs, and conclusions about the project and designed software complexity and quality are the results of the system functioning. Structure of ISAP represented on Fig.5.

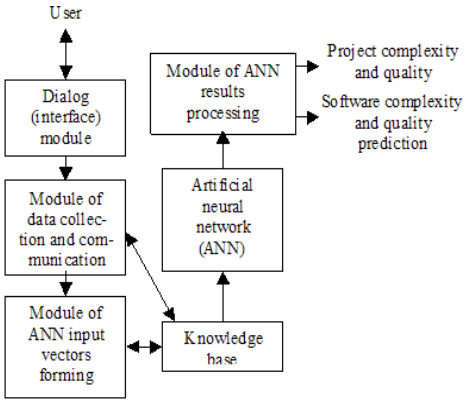

Figure 5. *ISAP structure.*

ISAP consists of modules:

1) dialog (interface) module;

2) module of data collection and communication;

3) knowledge base;

4) module of ANN input vectors forming;

5) artificial neural network;

6) module of ANN results processing.

The dialog (interface) module visualizes the functioning of module of data collection and communication, displays the system functioning and produces the messages to user in an understandable form for him.

The module of data collection and communication reads the user information about the quantitative exact and predicted values of software metrics, saves the obtained information in the knowledge base and transmits its to the module of ANN input vectors forming.

Knowledge base contains the quantitative exact and predicted values of software metrics at the design stage, the ANN input vectors and the rules of ANN results processing.

The artificial neural network (ANN) provides the approximation of software metrics and gives the quantitative evaluation of project complexity and quality and prediction of designed software complexity and quality by NMEP.

The module of ANN input vectors forming prepares the metrics values from the knowledge base for the ANN inputs.

The module of ANN results processing makes the conclusions about the project quality and complexity and the expected quality and complexity of designed software on the basis of ANN results.

For this purpose 12 production rules were developed:

1) if $PCE = 0$, then complexity metrics with the exact values at the design stage were not determined;

2) if $PCE \to 0$, then the project is complicated to realization;

3) if $PCE \to 1$, then the project is simple to realization;

4) if $PQE = 0$, then quality metrics with the exact values at the design stage were not determined;

5) if $PQE \to 0$, then project is a low quality

6) if $PQE \to 1$, then the project satisfies the customer requirements in quality;

7) if $SCP = 0$, then complexity metrics with the predicted values at the design stage were not determined;

8) if $SCP \to 0$, then designed software will have significant complexity;

9) if $SCP \to 1$, then designed software is expected simple;

10) if $SQP = 0$, then quality metrics with the predicted values at the design stage were not determined;

11) if $SQP \to 0$, then designed software is low quality;

12) if $SQP \to 1$, then high quality software is expected.

Using these rules, ISAP gives the evaluation of project complexity and quality and the prediction of the complexity and quality of developed for the project software Nhese evaluation and prediction help the customer to make the right decisions about project selection.

For example, the input data of ISAP are the results of metric analysis of 3 projects developed by the same requirements to solve one problem by a software company "STU-Electronics", Khmelnitskiy, Ukraine (table 3).

Table 3. *The Processing of Results of Metric Analysis at the Design Stage by ANN of ISAP.*

№	Complexity Metrics with Exact Values	Quality Metrics with Exact Values	Complexity Metrics with Predicted Values	Quality Metrics with Predicted Values	ANN of ISAP Results
1	Chepin's metric = 1690 Jilb's metric = 158	Cohesion metric =10 Coupling metric = 1	Expected LOC = 3280 Halstead's mtric = 73500 Jilb's metric = 0,051	Software design total time =26 Software design cost =975 Functional points = 120	Y1=0,94 Y2=1 Y3=0,94 Y4=0,96
2	McClure's metric = 90020 Kafur's metric = 376930 Chepin's metric = 24540	Cohesion metric = 3 Coupling metric = 7 Metric of the global variables calling = 0,72	Expected LOC = 40130 Halstead's metric = 124926 McCabe's metric = 1905	Software design total time = 397 Software design cost = 19000 Functional points = 2222	Y1=0,24 Y2=0,31 Y3=0,21 Y4=0,24
3	Chepin's metric = 14538 Jilb's metric = 1121	Cohesion metric = 7 Coupling metric = 4	Expected LOC = 25533 Halstead's metric = 781231 Jilb's metric = 0,52	Software design total time = 217 Software design cost = 10762 Functional points = 1212 Software quality audit cost = 1100	Y1=0,56 Y2=0,61 Y3=0,5 Y4=0,58

According to the obtained results: the project №1 is simple and has high quality, the future software is expected as a simple and high quality too; the project №2 is quite complicated and has a low quality, the future software is expected as a complicated and low quality also, the project №3 has medium complexity and quality, the future software is expected as medium complexity and quality too.

ISAP conclusions provide the comparison of different versions of projects, where the cost and development time approximately equal. ISAP help make the right choice and realize the project №1 with the best complexity and quality evaluations.

5. Conclusions

The conclusions by neural method and intelligent system of software characteristics assessment and prediction provide the assessment of project and the prediction of characteristics of the developed software for the project. These conclusions are based on the software complexity and quality metrics with exact or predicted values at the design stage. The NMEP and ISAP conclusions also provide a comparison of different versions of project.

The proposed neural method and intelligent system of assessment and prediction of software characteristics pro-

vide to customer the information to selection of the software project and to comparison the different versions of the project. NMEP and ISAP are the basis for making grounded and motivated decision on the choice of the project and its version with regard the characteristics of complexity and quality of the project and the developed software.

When developing NMEP and ISAP problems were occurred, the main reasons are:

1) absence of a fundamental theory of software engineering (special international organization SEMAT trying to solve this problem);

2) lack of the necessary theoretical and methodological principles of development and implementation of intelligent information technologies for management of the software characteristics;

3) absence of theory and methodology of assessment and prediction of software quality and complexity at the design stage.

Further efforts of authors will be directed to the solving of the aforementioned problems.

References

[1] V.Mishchenko, O.Pomorova, T.Hovorushchenko. CASE-assessment of Critical Software Systems. Volume 1. Quality / Kharkiv: The National Aerospace University "KhAI", 2012. - 201 p. [in Russian].

[2] William J. Brown, Raphael C. Malveau, Hays W. McCormick, Thomas J. Mowbray. AntiPatterns: Refactoring Software, Architectures, and Projects in Crisis - Wiley, 1998 - 336 p.

[3] Nancy Leveson, Clark S. Turner. An Investigation of the Therac-25 Accidents // IEEE Computer, Vol. 26, No. 7, July 1993, pp. 18-41.

[4] When Software Catastrophe Strikes. Ariane 5 explosion (1996) // http://images.businessweek.com/slideshows/2012-08-07/when-software-catastrophe-strikes.html#slide2.

[5] Stephen Turner. Expertise and Political Responsibility: the Columbia Shuttle Catastrophe // http://link.springer.com/chapter/10.1007%2F1-4020-3754-6_6.

[6] F-22 Squadron Shot Down by the International Date Line // http://www.defenseindustrydaily.com/f22-squadron-shot-down-by-the-international-date-line-03087/

[7] GLONASS Triple Satellite Launch Suffers Rare Failure // http://www.insidegnss.com/node/2399

[8] Russian rocket launch fails as engine shuts down // http://in.news.yahoo.com/russian-rocket-launch-fails-engine-shuts-down-105005395.html

[9] Cost of a bug within a software lifecycle // http://www.testically.org/2012/02/09/cost-of-a-bug-within-a-software-lifecycle/

[10] PVS-Studio description [in Russian] // http://www.viva64.com/ru/pvs-studio/

[11] CAST Application Intelligence Platform // http://www.castsoftware.com/products/cast-application-intelligence-platform

[12] IBM Security AppScan Source // http://www-01.ibm.com/software/rational/products/appscan/source/

[13] SofCheck Inspector 2.1268 // http://sofcheck-inspector.findmysoft.com/

[14] Visual Studio Team System Features // http://www.learnvisualstudio.net/series/visual_studio_team_system_features/

[15] Stacy J. Prowell; Carmen J. Trammell; Richard C. Linger; Jesse H. Poore. Cleanroom Software Engineering: Technology and Process - Addison-Wesley Professional, 1999 - 416 p.

[16] IBM Rational Logiscope // http://www.ibm.com/developerworks/rational/products/logiscope/

[17] Locate a Test Center // http://www.pearsonvue.com/vtclocator/

[18] Rational Purify // http://www-01.ibm.com/software/awdtools/purify/

[19] IBM Rational Software: Accelerate product and service innovation // http://www-01.ibm.com/software/rational/