Technology Oriented Assessment of Software Reliability: Big Data Based Search of Similar Programs

Vyacheslav Kharchenko[1], Svitlana Yaremchuk [2]

[1] Computer networks and systems department of, National Aerospace University "KhAI"
17, Chkalov St. Kharkiv, 61070, Ukraine,
v.kharchenko@csn.khai.edu

[2] Department of General Scientific Disciplines, Danube Institute of National University
"Odessa Maritime Academy", 9, Fanagoriyskaya St. Izmail, 68600, Ukraine,
svetlana397@yandex.ru

Abstract. The work contains a review of concept, tasks and some solutions for the Methodology of Technology Oriented Assessment of Software Reliability (TOAStRa) methodology. Implementation of the methodology, in particular, prediction of software system (SWS) reliability can be based on processing information about software with similar attributes and metrics, which is extracted from big data storages. The technique to search of similar programs has been suggested. The similarity principle is based on complexity and structure SWS metrics and metrics of program language similarity. The work represents formulas for calculation of group and average deviation rates describing the SWS similarity. Software Agent for Search of Similar programs and data processing (ASS) is developed. Case study related to search programs with the same complexity metrics in data storage is described.

Keywords. Reliability, software system, technology oriented assessment, big data storages, search of similar programs.

Key Terms. KnowledgeEngineeringMethodology, Metric, Agent, Data.

1 Introduction

1.1 Motivation

Growing customer's demands to SWS is a cause of their continuous sophistication [1]. Consequently, cost of verification, number of undetected design faults and charges to cover consequences of failures increase [2]. Available methods are insufficient to encompass variety and specific features of business processes in the SWS developing and require substantial expenditures for their adaptation and implementation. Quality analysis (QA) managers of the IT companies check results of testing and encourage correction of a set of test units to ensure correctness of specified functions "at

the moment" and don't take into account actual level of software reliability during and after finalization of the project. In fact, they don't apply software reliability growth model (SRGM) and assessment techniques based on the SRGM, since they require additional time and resources. In critical domains the required level of software reliability and safety is ensured due to multi-step and resource-intensive procedures without measurement and analysis of software reliability indicators to optimize them. The enlisted factors prevent to implement available models and techniques for reliability assessment into working practice of software and software-based systems development companies.

Thus, one of the key challenges for model-based software reliability assessment is a limited use of mathematical methods, SRGMs and SRGM-based techniques to analyze and control reliability indicators. To decrease this gap between theory and practice such methods and procedures of reliability assessment must be embedded into software life cycle by means of applying special tools and processes. Such tools should be "envisioned" for developers and applied by QA service for analysis and feedback for current and prospective software projects assessment.

Besides, there are a lot of software projects having similar functional and complexity characteristics. Reliability related information (testing, operation data) about such projects located in data storages may be used to predict corresponding indicators for "new" SWS to take into account reliability assurance processes as well, as business processes in general. Hence, technology of big data should be developed, adapted, and applied to find and to use data for such goals.

1.2 Work Related Analysis

In the course of the SWS development necessity arises to estimate achieved reliability and make a predicting reliability indicators dynamics in future by means of SRGMs described in works [3-9]. Great number of various SRGMs, SWS types, specifics of development processes creates difficulties in choice of an optimum model for individual system. SRGM selection method based on matrix of their allowances has been proposed in the work [10]. However, the matrix in question should be supplemented with newer models. Further development of the method is possible on the basis of priority setting for allowances with numerical evaluation of their level of implementation in the course of the SWS development. Such an approach may enable to obtain more precise numerical evaluations of SRGM adequateness for particular SWS.

Practical SRGM application requires substantial resources and experts' qualification. Therefore models should be simplified and form a basis for software tools development. Their flexibility allows to adapt tools to specific systems and development processes without substantial expenditures and resources consuming. Embedding tools into software corporations business processes allows developers to evaluate reliability of their products in restricted resources conditions. Nowadays multiple software reliability assessment tools [11] are available for different stages of the SWS life cycle.

1. At the stage of requirements development: Matrix Requirements Medical, Modern Requirements Suite, Orcanos Requirements Management, Polarion Requirements, RequirementsHub, Test Requirements Agile Metric, Visure Requirements, etc.
2. At the stage of software architecture design: GenieBelt, Bluebeam PDF Revu, progeCAD 2010 Professional, ArchiCAD, AutoCAD, Praesto AE, SketchUp Pro, MicroStation, SmartDraw, Chief Architect, Clearview InFocus, Newforma PIM solution, Arcon Evo, ConceptDraw PRO, SoftPlan, CorelCAD, Envisioneer, Easy Blue Print, etc.
3. At the testing stage: Issue Tracking, Software - Asset Bug, BUGtrack, BookingBug, Bug-Track.com, BugApp360, BugAware, BugHerd, BugHost, etc.

However, these tools don't take into account complexity of developed artefacts (requirements, architectural elements, and modules), preventing developers to concentrate their efforts on the most complicated and defective, or exposed to defects, artefacts. Development and generation of newer tools taking into note the artefacts' complexity may enable to overcome existing gap between theory and practice of the SWS reliability.

Experts of software corporations need experimental data to review and forecast SWS reliability indicators. Processes of data accumulation run intensely in modern digital world. From 2005 to 2020, the digital universe will grow by a factor of 300, from 130 to 40,000 Exabytes, or 40 trillion gigabytes (more than 5,200 gigabytes per each man, woman, and child in 2020). From now until 2020, the digital universe will be increasing almost twice every two years [12]. This data includes data necessary for reliability evaluation from SWS demands management systems, configuration control systems, defects control systems, and prototypes, initial software code, metrical data, test-cases, temporary rows of defects' identification, code analyzers reports, system logs, users' reports on SWS failures, etc. They may include audio, visual, graphical, text and numerical data either structured, or partially structured, or non-structured. However, such data is not normalized and convenient for direct application. Big data should be used for its processing.

Big data [13] is an umbrella term that often refers to a process of applying computer analytics to massive quantities of data in order to discover new insights and improve decision-making. It often describes data sets that are so large in volume, so diverse in variety, and moving with such a velocity that it is difficult to process it using traditional data processing tools. Big data construes a basis for Business intelligence. Business intelligence refers to the set of technologies and applications that transform crude data into operational insights that may improve business performance and decision-making.

The SWS development is a manufacturing sector. It stores more data than any other sector. As a result, manufacturers have a lot to gain from better use of data to boost efficiency, drive quality, and improve the way products are designed, composed, and distributed. According to estimation made in [14], better use of data in manufacturing may yield up to a 50 percent decrease in product development time and assembly costs. In fact, International Data Corporation estimates that manufacturing companies that take full advantage of their data are poised to achieve a $371 billion data dividend

within four years. Companies that use data-based decision-making report about 5% to 6% boost in productivity [15]. Using big data, companies may also better track and manage global supply chains, and reduce product defects.

The improvement of software reliability is investigated in [16]. This research proposes data mining techniques and studies two bug detection methods, including CP-Miner that detects copy pasted code and related bugs, and PR-Miner that extracts application-specific programming rules and detects violations that indicate potential bugs. Although this study has shown that data mining techniques is efficient for static analysis, it still requires further efficiency improvement by means of the code complexity accounting. In works [17, 18] the achieved results and numerous problems of research directions for engineering big data analytics software are reviewed.

Data located in the big data storages is a great and insufficiently used resource for SWS reliability evaluation, forecasting and management. At the same time big data based software reliability assessment may be applied using methods of data filtering and processing, as described in [19], such as: Data cleaning, Classification, Clustering, Frequent Pattern Mining, Probabilistic & Statistical Methods, Anomaly & Outlier Detection, Feature Extraction, Selection and Dimension Reduction, Mining with Constraints, Mining Unstructured and Semi Structured Data, Mining Complex Datasets. Review of processed data enables to evaluate and predict reliability indexes and adopt efficient solutions aimed to improve SWS reliability. The solutions enable to answer such questions as which artefacts should be verified, applicable sequence and duration of verification, what modules should be exposed to re-factoring, what test-cases should be applied, where the required reliability should be admitted as obtained, number of technical support experts to be assigned, etc.

Necessity arises to find similar data with already known reliability indexes in big data storages. They include research data [20], open data of major public bodies [21], corporative data [22], data of internet communities of developers [23]. Accumulation and systematizing of required data enable an individual software corporation to form a data field for reliability assessment and forecasting. The essential problem is to find data for reliability evaluation of particular SWS being under development in huge storages. This work proposes a common approach and techniques to solve this problem.

1.3 Aim and Tasks

Strategic aim of the research consists is the SWS reliability improvement in restricted resources conditions by means of embedding methods and tools of reliability evaluation and control into lifecycle processes. The work is aimed to develop concepts, principles, tasks and elements of methodology for technologically oriented evaluation of the SWS reliability.

The tasks of the research are, as follows:

1. Review of available models and methods evaluation of the SWS reliability;
2. Development of concepts, principles, tasks and elements of methodology for technologically oriented evaluation of the SWS reliability;

3. Development of search method in big data storage to find experimental data of similar SWS to evaluate reliability indexes.

2 The TOAStRa Methodology

The proposed approach is titled as Methodology of Technology Oriented Assessment of Software Reliability (TOAStRa). This methodology is based on general concept and a few principles (Fig. 1).

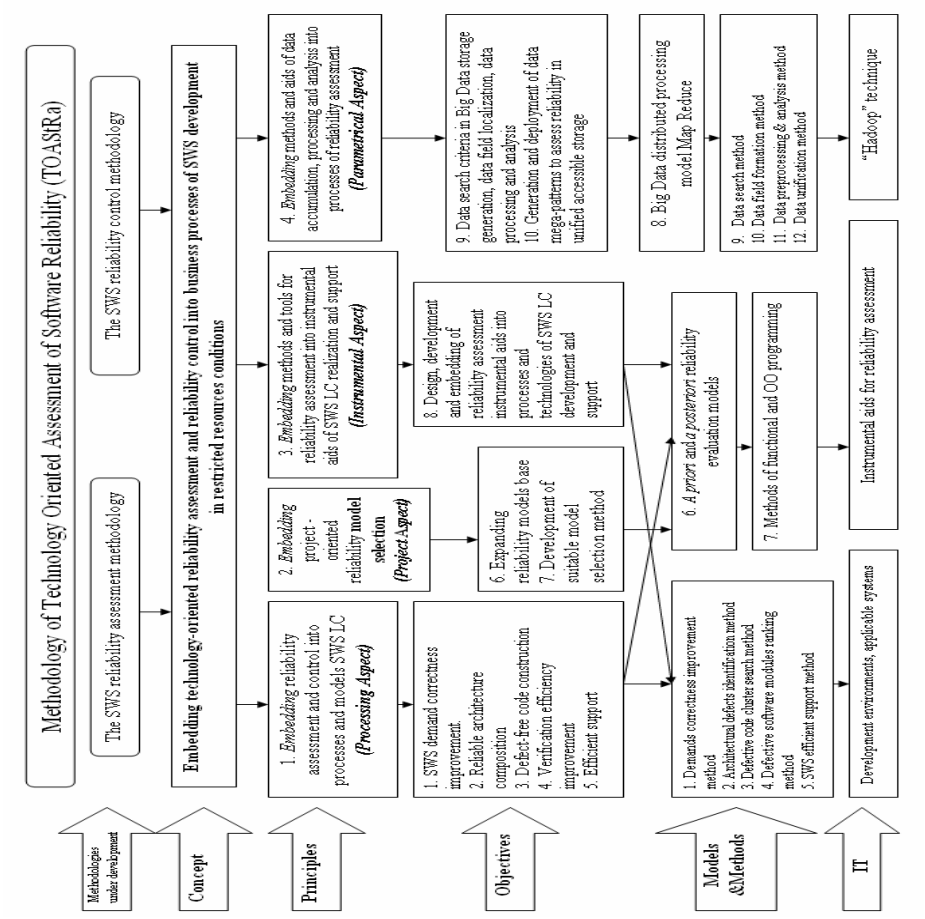

Fig. 1. The structure of methodology TOAStRa.

2.1 The Principles

Principle 1. *Embedding evaluation procedures and reliability management procedures into SWS life cycle processes and models.*

The principle means embedding into processes of requirements analysis, architectural design, realization, verification, and maintenance, as specified in the ISO/IEC 15288:2015, Systems engineering — System life cycle processes.

The principle further supposes embedding models, as enlisted: V-model, XP, SCRUM, Rapid Application Development, Dynamic Systems Development Method, Rational Unified Process, Microsoft Solutions Framework, Kanban Development, Cleanroom Software Engineering, Waterfall model into SWS life cycle processes.

This principle implements *Processing Aspect* of solving problems of SWS reliability evaluation and control. It enables to evaluate and control reliability of generated artefacts at various stages of SWS life cycle.

Principle 2. *Embedding project-oriented selection, complexion and parameterization of models to evaluate achieved reliability indexes at the final stage of the SWS development.* Implementation of this principle supposes applied models' base expansion and their allowances; development of models selection, complexion and parameterization methodology. This principle reflects the *Project Aspect*.

Principle 3. *Embedding reliability evaluation methods and instrumented aids into instrumental tools of implementation and supporting processes of SWS life cycle*. This principle supposes design and development of software reliability evaluation aids for their flexible embedding in design environment and control systems in artefacts operating within SWS life cycle. The principle also supposes usage of existing standard software applications for functional reliability evaluation. This principle reflects *Instrumental Aspect*.

Principle 4. *Embedding methods and aids of data search, accumulation, storage, processing and analysis applied in similar projects and systems into SWS under development reliability evaluation processes using big data storages, principles and technologies.* Realization of this principle supposes search of necessary data in *big data* storages, generation of broad corporative data field for reliability evaluation, data accumulation and storage in cloud storage, *big data* processing and review applying specialized techniques, using resulting data for reliability assessment *(software reliability big data*), and, finally, making data usage convenient *(software reliability big data usability).* This principle implements *Parametrical Aspect*.

A sequence of tasks is proposed below for realization of above principles.

2.2 The Tasks

Tasks implementing principle 1:

Task 1 – Requirements correctness improvement (method taking into account priorities and complicatedness of requirements embedded into SWS LC);

Task 2 – Building reliable architecture (UML-models, mathematical models, method of "bottleneck" search in architecture, embedding into SWS LC processes);

Task 3 – Defect-free code generation (simulating models, defective clusters search method within code, embedding into SWS LC processes);

Task 4 – Achievement of efficient verification (method of software modules ranking taking into account their complicatedness, embedding into SWS life cycle processes);

Task 5 – Efficient support («rear bench technique» - bringing changes only into essential functions, putting support off obsolete SWS functions).

Tasks implementing principle 2:

Task 6 – Expanding applicable models' base and their allowances;

Task 7 – Development model selection method based upon setting priorities for allowances and numeric evaluation of their practical implementation extent. The proposed evaluation technique enables to obtain more precise numeric evaluation for various models meeting the needs of individual project.

Task implementing principle 3:

Task 8 – Development and embedding instrumental aids of reliability assessment into technologies of SWS life cycle development and support.

Tasks implementing principle 4:

Task 9 – forming search criteria, methods and software aids in *big data* storage of SWS experimental data similar to SWS being developed; processing and review of this data; generation of corporative data field for reliability evaluation based on this data;

Task 10 – Creating data mega-patterns to assess reliability (unified data pattern and necessary context data); placement of this data in unified data storage for common access.

Authors offer big data search technique for similar system to assess reliability of SWS being in development within the framework of proposed concept for 9^{th} task.

3 Software Reliability Assessment Based on Big Data

3.1 General Approach

Let us suppose that a certain software corporation is developing a software system. Initial software code is partly developed. SWS development process is restricted in time and funding. Scope and costs of works associated with forthcoming tests should be evaluated to meet the imposed restrictions. Reliability estimated indexes should be calculated. For example, quantity of defects in particular modules, total quantity of defects in developed code and defects' density should be assessed prior the testing process commencement. Corporation may use experimental data on defects identified in previously developed SWS for such preliminary assessment. They may be further referred to as historical data. As another example, the case may be when limited funds, terms or personnel resources prevent the corporation from accumulation and reasonable usage of such data. Even with available data the system being designed may have nothing common or similar with previously developed systems.

With available historical data found in big data storage the corporation may use them for verification or updating preliminary assessment obtained on the basis of historical data. In any case, it may be feasible to search SWS experimental data from other developers applicable for reliability assessment of own SWS in big data storages with free access. These data storages are well known in global data field. They may be, for instance, software reliability data depositories [20] of international con-

ferences, NASA data portal [21], services of code testing and statistic analysis [22], software sport services [23] and other sources.

The abovementioned big data storages contain gigabytes of codes and experimental reliability data on multiple SWS by different developers. This data may be called "associated". It may consist of artefacts – requirements, initial code, tests, artefacts evaluations and processes of their development under various metrics (e.g. known object-oriented metrics RFC, WMC, LCOM, LOC, NPM, CE, CBO, CA, NOC, DIT). This data includes actual quantity of defects identified in modules, temporal series of defects detection. This data enables to evaluate, forecast or verify reliability indexes of systems at various stages of development.

Thus, there are both necessity and possibility to evaluate, predict and control reliability of SWS being developed using big data storages. To do it, it is necessary to find a system similar by a number of criteria to particular system in development. Therefore a method should be developed to perform big data based search for similar programs.

3.2 Search for Similar Programs

SWS development consists of a number of technological stages (requirements outlining, design, code writing and verification). Artefacts are generated at each of such stages. The artefacts contain defects and affect the SWS reliability. Majority of defects is located in the initial code. Initial codes with various SWS possess different objective features, such as structure, dimensions, complicatedness, and programming languages. In view of such differences the task of search for suitable associated data for reliability indexes assessment and forecasting may be formulated, as follows: SWS should be found in big data storage with initial code of the greatest resemblance with that of the SWS under development in structure, dimension, complicatedness and programming language.

In view of the above the necessity arises to formulate a SWS initial code similarity principle.

Software systems' similarity principle

Structure, dimensions and complicatedness of the SWS may be assessed by means of metrics. The proposed SWS code similarity principle bases upon metrics enlisted below:

1. Initial code dimension in thousands of lines (KLOC);
2. Total quantity of code modules;
3. Complicatedness assessment metrics for code modules;
4. Total, average and maximum evaluation for each metric.

It is worthwhile to mention here, that numerical evaluations of these metrics reflect not only system's dimensions and complicatedness, but also number of system's modules/classes and their links, i.e. system structure. The possibility to assess compli-

catedness of system being developed and associated system by means of uniform set of metrics guarantees resemblance of programming languages of the systems in question.

In general, proximity of ratings under certain metrics for system under development and system taken for comparison provides similarity of dimensions, complicatedness, structure and programming language. System similar to the system under development should be defined as a system having minimum deviations of ratings by nominated metrics. Selection a similar SWS can be made via relative deviations of each of appropriate metrics for system under development and system taken for comparison.

1. Relative deviation of initial code dimensions $RD_{size} = \frac{|\ KLOC_d - KLOC_f\ |}{KLOC_d} \cdot 100\%$. The bottom index "d" corresponds to rating of the system under development. The bottom index "f" corresponds to rating of a system taken for comparison.
2. Code modules quantity relative deviation $RD_{mod} = \frac{|\ MC_d - MC_f\ |}{MC_d} \cdot 100\%$, with MC – number of modules;
3. Summarized rate relative deviation $RD_{sum} = \frac{|\ Sum_d - Sum_f\ |}{Sum_d} \cdot 100\%$, average rate relative deviation $RD_{avg} = \frac{|\ Avg_d - Avg_f\ |}{Avg_d} \cdot 100\%$, maximum rate relative deviation $RD_{max} = \frac{|\ Max_d - Max_f\ |}{Max_d} \cdot 100\%$ for each metric of complicatedness.

At the next stage calculated deviations should be grouped into three groups. The first group indicates dimensions deviation rate, the second group indicates structure deviation rates, and the third group indicates code complication deviation rates. Average deviation rate should be calculated for each group. Deviations within a group are feasible to apply with unequal priority indexes for SWS similarity assessment. Under certain circumstances, priority indexes for SWS similarity assessment may be either dimension, or structure, or complicatedness of the system. Common general average deviation rate for all the rates should be also calculated. This value is feasible to apply for indexes with equal significance.

So, search for comparative SWS being similar or the most proximal to the SWS under development requires to know code dimensions, number of modules, estimated complicatedness evaluated applying unified set of metrics, calculated metrical rates' deviation and, finally, system selection with minimum deviations.

The authors state a hypothesis that a similar system may be found in big data storage among available multiplicity. This hypothesis, however, should be checked. Big data storage contains experimental data of multiple various systems. For example,

storage [20] contains data relevant to metrics and defects of sixty one SWS. Manual processing of such data may take too much time and labor. Specialized software Agent for Search of Similar programs could be helpful in the aspect of automation of such a process.

The agent for search of similar programs

ASS performs the following functions:

1. Downloading of metrical rates of system under development as a reference point for comparison with other systems.
2. Downloading of other SWS data (metrical indexes and defects quantity from big data storage into local corporative or cloud storage. This step is necessary to generate a corporative data field for multiple reliability assessment.
3. Data transformation into appropriate format for processing (*.db, *.xml, *.xlsx, etc.);
4. Transformed SWS data import into ASS memory.
5. Data processing – calculation of deviation rates within a group and total average deviation of all the rates.
6. Entering deviations for each system into resulting account.
7. Accounted deviations sorting to ground similar SWS choice.

ASS creates the resulting account with group and average deviations for multiple involved SWS. After the deviation values are sorted the SWS with minimum deviations form metrics of SWS under development are placed into the top of account. The account enables to make a well-grounded choice of SWS with the highest similarity index to the SWS under development. Experimental data on defects of the chosen similar SWS may be used to assess and predict similarity of the SWS under development. The proposed ASS is a program for processing flat (not linked) tables and for calculation of statistic indexes.

If to compare the offered ASS to the known software of the statistical analysis it is necessary to notice that he has two advantages. The first advantage it is lack of excess functionality for this concrete case of application. The second advantage it is support of all technological stages of data processing.

Search of the similar programs

The procedure of similar programs search based on big data consists of seven steps.

Step 1. Calculate metrical rates for SWS structure, dimensions and complicatedness under development.

Step 2. Activate ASS, consistently download identical metrical rates and data of defects of other SWS from big data storage.

Step 3. Transform downloaded data of other SWS into appropriate format for processing.

Step 4. Calculate internal deviation rates and general average deviation rate.

Step 5. Record deviation rates for each SWS into resulting account. Sort indexes in the account.

Step 6. Select similar system with minimum deviation rates in the account.

Step 7. Use actual data on defects of the selected SWS to assess reliability of the SWS under development.

4 Case Study

The above declared hypothesis stating that a system similar to the SWS under development in structure, dimensions and program language may be found in big data storage requires experimental checking. Such a check was performed in a manner described below. Metrical data and defects data for twenty one SWS have been randomly selected and downloaded from big data storage [20] into local computer disc. One of these systems has been taken as a reference point. Other twenty systems have been explored for similarity of their features (structure, dimensions and complicatedness) to the reference system. Programming language similarity of the systems in question has been supported by unified set of metrics for complicatedness assessment. They are rather common metrics of object-oriented code complicatedness assessment RFC, WMC, LCOM, LOC, NPM, CE, CBO, CA, NOC, DIT.

ASS designed by authors transformed data from *.txt or *.csv format into *.dbf format. Further calculations of relative deviations for metrical rates had been performed by means of SQL instructions for each system. Data processing applying ASS took about two working hours. Group and average deviation rates have been stored in resulting account, as shown in Table 1.

Table 1. Metric rates relative deviations of SWS compared with reference system**.**

№ SWS	Metrical rates deviations, %			
	Structure	Dimensions	Complicatedness	Average rate
1	0,0	0,0	0,0	0,0
2	**5,1**	**9,0**	**5,4**	**6,5**
3	12,2	20,6	41,8	24,9
4	12,7	51,3	35,7	33,2
5	12,8	19,0	28,7	20,2
6	14,4	42,1	46,7	34,4
7	22,4	87,7	34,9	48,3
8	23,6	57,4	33,6	38,2
9	24,5	82,3	29,2	45,3
10	27,4	56,0	43,9	42,4
11	28,6	40,5	40,5	36,5
12	28,6	51,9	44,5	41,7
13	29,9	68,0	20,9	39,6
14	30,3	70,9	26,0	42,4

15	30,9	83,6	45,1	53,2
16	31,4	78,9	38,6	49,6
17	46,0	24,7	44,3	38,3
18	71,8	52,5	24,5	49,6
19	74,6	59,0	25,4	53,0
20	270,1	815,7	52,5	379,4
21	350,6	660,9	66,9	359,5

Indexes in the account have been sorted in increasing order. Reference SWS has number 1. Naturally, deviation rates in corresponding line are zero. SWS No 2 follows directly after it with minimum deviation from reference SWS (highlighted line in Table 1). System with increasing deviation rates are placed downwards. The resulting account enabled to choose SWS with the highest level of similarity to the reference SWS.

5 Conclusions

The paper describes a concept, tasks and some solutions for the TOAStRa methodology. Implementation of the methodology, in particular, prediction of SWS reliability can be based on processing information, which is extracted from big data storages by using the software agent ASS. The technique has been suggested to search and analyse similar programs. The similarity principle is based on complexity and structure SWS metrics and metrics of program language similarity. Calculation formulas to assess group and average deviation rates describing the SWS similarity have been suggested.

Case study allowed to obtain some experimental results. A system has been identified with minimum (5,1 – 9,0 %) and average 6,5% relative deviation of metrical rates among twenty explored systems. Obtained results confirm the allegation that systems with known reliability indexes similar to the SWS under development may be found from great quantity of experimental data kept in big data storage to assess, verify and predict its reliability.

Data processing for twenty SWS by means of ASS took about two hours. The search of similar programs represents practical value for a project manager and personnel of the SWS testing group. The ASS may be adapted by software companies to take into account specifics of developed SWS.

Further research will be focused on formal definition of similar programs-software-SWS including functionality, applied technology, costs, etc. Special interest presents application of big data swapping technique to multi-dimensional matrix of SWS metrical rates. ASS and technique can be integrated with procedures of SWS decomposing, similar software components separated search, results processing, and obtaining integrated reliability assessment. Future researches will be directed to process of software reliability management basing assessment technique and results.

References

1. Schmidt, R., Kaufmann M.: Software Engineering: Architecture-Driven Software Development. Elsevier, pp. 376 (2013)
2. Papows, J.: Glitch: The Hidden Impact of Faulty Software. Prentice Hall, pp. 208 (2010)
3. Musa, D.: Software Reliability Engineering: More Reliable Software Faster and Cheaper (2nd Edition), Author House (2004)
4. Littlewood, B., Strigini, L.: Software Reliability and Dependability: A Roadmap. Proceedings of the 22nd International Conference on Software Engineering (ICSE'2000), Limerick, pp. 177 -- 88 (2000)
5. Shooman, M.L.: Reliability of Computer Systems and Networks: Fault Tolerance, Analysis and Design. Wiley, New York (2002)
6. Chen, M., Lyu, M., Wong E.: Effect of Code Coverage on Software Reliability Measurement. IEEE Transactions on Reliability, vol. 50, no. 2, pp.165--170 (2001)
7. Malaiya, Y., Li, N., Bieman, J., Karcich, R.: Software Reliability Growth with Test Coverage. IEEE Transactions on Reliability, vol. 51, no. 4, pp. 420--426 (2002)
8. Swamydoss, D., Nawaz, K.: Enhanced version of growth model in web based software reliability engineering. Journal of Global Research in Computer Science, vol. 2, no. 12, p. 44--46 (2011)
9. Purnaiah B., Rama Krishna V.: Fault removal efficiency in software reliability growth model. Advances in Computational Research, vol. 4, Issue 1, pp. 74—77 (2012)
10. Kharchenko, V. S., Tarasyuk, O. M., Sklyar, V.V.: The Method of Software Reliability y Growth Models Choice Using Assumptions Matrix. Proceedings of 26-th Annual Int. Computer Software and Applications Conference, COMPSAC, Oxford, England, pp. 541--546 (2002)
11. The Smart Way to Find Business Software, http://www.capterra.com/
12. Gantz, J., Reinsel D.: The digital universe in 2020: Big Data, Bigger Digital Shadow s, and Biggest Grow in the Far East, http://www.emc.com/collateral/analyst-reports/idc-the-digital-universe-in-2020.pdf
13. BSA | The Software Alliance. What's the Big Deal With Data? https://www.bsa.org
14. Manyika, James et al: Big Data: The Next Frontier for Innovation, Competition, and Productivity. McKinsey Global Institute, https://bigdatawg.nist.gov/pdf/MGI_big_data_full_report.pdf (2011)
15. Economist Intelligence Unit. The Deciding Factor: Big Data & Decision Making. Point Of View, http://bigdata.pervasive.com/Solutions/Telecom-Analytics.aspx
16. Zenmin, L.: Using Data Mining Techniques to improve Software Reliability. Dissertation for the degree of Doctor of Philosophy in Computer Science, p. 153 (2006)
17. Carlos, O., Adrian, P.: Research Directions for Engineering Big Data Analytics Software. Florida Institute of Technology. Published by the IEEE Computer Society, pp. 13--19, DOI: 10.1109/MIS.2014.76 (2015)
18. Rademakers, F.: ROOT for Big Data Analysis. Workshop on the future of Big Data management, London, UK (2013)
19. Leskovec, J., Rajaraman, A., Jeffrey D.: Mining of Massive Datasets. Stanford Univ., Milliway Labs., p. 495 (2014)
20. Tera-PROMISE Home, http://openscience.us/repo/defect/ck/
21. NASA'S DATA PORTAL, https://data.nasa.gov/
22. Software Testing and Static Code Analysis, http://www.coverity.com/
23. Topcoder | Deliver Faster through Crowdsourcing, https://www.topcoder.com/