Metric-based method of software requirements correctness improvement

Svitlana Yaremchuk[1,*], *Nikolaos Bardis*[2] and *Kharchenko Vyacheslav*[3]

[1]Danube Institute, National University "Odessa Maritime Academy", 68600, Ukraine
[2] Department of Mathematics & Engineering Science Hellenic Military Academy, 16673, Greece
[3] Department of Computer Systems and Networks, National Space University named after N. E. Zhukovsky "KhAI", 61070, Ukraine

Abstract. The work highlights the most important principles of software reliability management (SRM). The SRM concept construes a basis for developing a method of requirements correctness improvement. The method assumes that complicated requirements contain more actual and potential design faults/defects. The method applies a newer metric to evaluate the requirements complexity and double sorting technique evaluating the priority and complexity of a particular requirement. The method enables to improve requirements correctness due to identification of a higher number of defects with restricted resources. Practical application of the proposed method in the course of demands review assured a sensible technical and economic effect.

1 Introduction and Background

Nowadays humanity utilizes a tremendous amount of software tools and software systems (SWS) for different domains and applications. Customers impose more complex and diverse requirements. Software produces more and more considerable influence on our life and our safety. In its turn, complexity of SWS increases [1]. Consequently, number of defects imposed by developers increases, the SWS reliability decreases, design costs rise. Huge losses and the decrease in the efficiency of human activity caused by the imperfection of SWS and design faults is one of the key challenges in modern IT engineering encouraging increased software reliability [2].

Motivation of research is, firstly, the need to analyze problems of reliability assessment and management. Secondly, in view of importance of software requirements correctness from the point of view of the SWS reliability we propose technique for assessment of complexity and improving of requirements' correctness.

2 Principles of Software Reliability Management. State of the Art

Researches worldwide developed a number of models, methods and software application to evaluate and manage the SWS reliability [3-11]. However, these methods don't take into due account variety and specific features of SWS development business processes and require great expenditures for adaptation and implementation. The above factors prevent to implement widely available model methods and reliability aids in software engineering routine practice, making such implementation impossible in a number of cases. Therefore, the first principle of the (SRM) provides adaptability, simplicity and low costs contributing to easy implementation of the assessment technique into business processes of software corporations.

The SWS lifecycle (LC) SWS consists of a sequence mutually dependent processes. These are stages of requirements' construction, design, codification, testing and the SWS maintenance. At these stages experts of various profile and qualifications create unique artifacts. They are requirements' specification, architecture and detailed SWS design, initial codes, testing procedures and versions, ready software product. The developed SWS reliability is pre-determined by correctness of each individual artifact, depends on its complexity and forms at each stage of its LC. However, existing reliability models don't take into sufficient account complexity and evaluate reliability at each LC stage fragmentarily. Therefore, the second SRM principle provides thorough complexity evaluation and artifacts reliability management throughout the entire SWS LC.

Different production data accumulates throughout the SWS LC stages. It includes user demands in textual, tabulated, graphical and verbal formats, architecture and design descriptions, algorithms, initial software code, testing scenarios, incoming data massive, video files of the SWS behavior under various conditions, technical

* Corresponding author: svetlana397@yandex.ru

documentation, etc. This multi-type structured and non-structured data of various SWS versions and assemblies are stored and actively used by developers during the long term of operation. This data, also called as Big Data may be measured in terabytes. Methods and applications for Big Data processing and analysis to improve SWS reliability are not sufficiently developed. Therefore, the third SRM principle provides for development Big Data analysis methods and application for SWS LC.

Competitive market compels software corporations to reduce SWS development costs with sharp need to provide the demanded reliability level. The developers often face the problem how to spend limited resources in the most efficient manner? What artifacts subsets should be verified in the available resources don't allow to afford complete verification? What particular pages of e.g. 200 available should be checked to identify maximum of errors, controversies, inaccuracies or incompleteness? What of thousands modules should be tested to detect maximum number of defects? So, the fourth SRM principle consists in maximum possible reliability improvement under restricted resources condition.

Each SWS development starts from collection and analysis of requirements. This process includes clarifying and documenting what should be done by an application and what features it should possess. This process involves all the concerned parties – investors, purchasers, users, from the one side, and managers, analysts, designers, codifiers, testers, i.e. SWS developers, at the other side. In the course of extended joint work errors may occur with any party category at both sides. As a result, incorrect requirements contain defects of various types, such as erroneous data, unclearness, ambiguities, incompleteness, doubling, and contradictions. Imperfect requirements may cost very much. As source [12] marks, requirements defects are 20-50 times more expensive to correct once the defects got into development process. Inspection of one full-scale software project showed that problems with requirements led to US$ 600.000.000 development budget exceeding, 8 years time delay and reduced functionality. Another research [13] showed that it takes about 30 minutes to correct error detected at the stage of requirement processing, whereas it takes 5 – 17 hours to correct an error detected in the course of system testing. As identified in the [14] eliminating requirements' defects are 100 times expensive than those detected in the course of requirements development, if indicated by the customers. Evidently, methods directed to identify as much defects, as possible, in requirements saves plenty time and costs. Therefore, method of improvement of SWS requirements correction is suggested. Below standards and terms applicable at the stage of requirements construction are described.

3 Standards and Terms Applicable at the Stage of Requirements Construction

Standards provided in [15, 16] lost their actuality and are inapplicable. Actual standard is IEEE/ISO/IEC 29148-2011 "Systems and software engineering – Life cycle processes – Requirements engineering" [17]. Terms definitions are enlisted in compliance with the said standard.

SWS Project (software system project) – a full set of developed agreed artifacts including requirements specification, architecture and detailed design, initial code, test procedures and versions, technical documentation, etc.

Requirements are a document specifying users' needs and aims, condition, features and capabilities of the software product.

Concerned parties are individuals, groups and/or organizations taking an active part in the project.

Requirements Analyst is a person in requirements developers team in charge of the requirements extraction, analysis, definition, approval and their management.

Requirements Development is a process aimed to determine project scope, users class and representatives, requirements' extraction, analysis, specification and approval.

Requirements Analysis is Process of categorizing data concerning requirements, requirements evaluation to determine desirable quality, requirements representing in various formats, detailed requirements extraction from requirements of higher levels, requirements priority discussion, etc.

Software requirements specification, SRS – result of a system requirements documenting in structured, accessible and controllable format. According to source [18], requirements specification is a set of technical documentation containing detailed SWS under development and its functionality, requirements to external interfaces, non-functional requirements, glossary, processes and subject zone models, other SWS data, as may be required by parties in concerned.

Requirements Attributes – description data of requirements with unique requirements numbers, data sources, logical groundings, priorities, persons in charge, SWS versions numbers, etc.

Requirement's Elements – any elements composing the requirement – constant values, variables, arithmetic and logical operations, etc.

Requirement complexity – difficulties met in understanding and realization of a requirement related with great quantity of components and links with other requirements and business rules.

Business rule – Policy, guiding principles or provisions which determine or restrict certain aspects of business activities.

Requirement Priority – numerical and/or quality evaluation of requirement attribute, taking into account importance and term of its realization. According to source [18], priorities construe a way to solve conflict between the requirements competitive struggle for restricted resources/

Requirement metric – mathematical representation of numerical evaluation of a particular requirement's feature.

4 Method of SWS Requirements Correctness Improvement

Modern SWS include plenty multi-type requirements. They include functional requirements, requirements to user and software interfaces, hardware interfaces, communications interfaces, requirements to capacity, data safety storage, system safety, etc. Therefore, the SWS specification is a bulky document of hundreds pages. Theoretical approach supposes total review of all these requirements. In practice, "even in medium-size specification experts review thoroughly only the first part, some of the most stubborn may look through half, but it is unlikely that anyone reaches the end" [18]. Entire SWS requirement review is impossible due to lack of time, financial and human resources. In this view, the offered method or requirements correctness improvement consists in extracting the most priority and complicated requirements from their entire set to be thoroughly reviewed to detect the highest possible quantity of defects in prevailing restricted resources situation. Priorities are assigned to requirements after their collection and documenting jointly by the manager, analyst and customers' representatives. Requirements complexity is determined by means of metrics.

4.1. Overview of Requirement Metrics

Publications [12, 18, 20, 21] and standards [17, 19] have been studied for analysis of existing metrics. These standards contain numerical evaluations for about 50 metrics. They include metrics for completeness; precise detailed requirements; non-elementary requirements; requirements not coordinated with other requirements; undetectable and untestable requirements; missed or skipped requirements; non-obvious requirements; defective requirements detected for 1 hour of checking; detailed requirement value; metrics for deleting, adding or modification of the requirement. Their analysis shows that such metrics are designated for quality control, requirements review and analysis efficiency. However, there is no metric evaluating requirement complexity in these standards.

Authors of work [12] apply requirement defect metric as a number of defects per documentation page, provide its interpretation and permissible value as 0.2 defects per page of the requirements' testing description, i.e. one defect per 5 pages. Unclear definition since a number of requirements per page is unknown.

Authors of work [18] stress upon correctness and traceability as the most important feature of requirements without outlining evaluation formulae. The authors define correctness as a precise description of desirable functionality. Traceability is defined as a possibility of analysis both in backward direction to initial sources of requirements and forward to design elements, initial code, tests, scope of application and other SWS elements. The elements include requirements of different types, business rules, architecture and design components, initial code modules, test versions, reference files, traceability is of great importance for requirements management. It enables to determine all the workable artifacts which should be modified once the requirement changes. As per complexity, the authors mark complexity of requirements detection process, complexity of their statement in foreign language, complexity of understanding and technical implementation of requirements with complicated Boolean logics (combination of operators AND, OR, NOT) having, however, refrained from providing numerical evaluation aids for these features.

Authors of [20, 21] propose more advance set of metrics. In work [20] they are numerical metrics of unique, understandable, misinterpreted, modified requirements and requirements under testing. The authors also propose two newer metrics, closer in sense, but different in evaluation manner:

1) correctness $MC = \frac{R_c}{R_t} * 100\%$, with R_c - number of requirements received the same true interpretation by more than 1 referee; R_t - total quantity of requirements;

2) quality $MQ = \frac{R_t - (R_f + R_d + R_m)}{R_t} * 100\%$, with R_f - number of identified and corrected requirements with defects, R_d - number of deleted requirements with defects, R_m - number of modified requirements.

Authors of work [21] added metrics of ambiguity, logical linkage and capability to check the requirements starting from the initial source. Yet, the work does not contain complexity metrics.

Thus, the review of publications and standards showed the lack of metrics enabling to evaluate requirement complexity. The referred enlisted metrics describe construction process and various requirements features, saving the complexity. With such metric missing the most complicated requirements cannot be identified and selected for inspection and review to detect higher defects quantity with prevailing restricted resources situation. The [18] notes that the requirement complexity is evaluated by expert in a subjective manner. Analysis results and author's personal experience in SWS development enable to state that requirements complexity is not numerically evaluated in the software engineering. Evaluations are not applied to improve requirements correctness and the SWS reliability. Therefore, it is necessary to develop a metric for precise numerical evaluation of requirements' complexity.

4.2. Proposed Metric for Requirement Complexity Evaluation

The new metric should take into account internal and external complexity inherent with the requirement. Internal complexity is determined by quantity of components. External complexity is represented by links connecting a particular requirement with other requirements, business rules, etc. These links are illustrated at Figure 1 representing an entity relationships

(ER) diagram of logical contacts of the requirement with other elements.

The ER – diagram shows that the requirement may include one element, or more; may affect, or does not affect several dependent requirements; may depend, or not, on some affecting requirements; may depend (or not) on some affecting business rules; may depend, or not, on some affecting entities, strictly specified for each SWS being developed. The higher is quantity of elements in requirement and links of the requirement with other requirements, business rules and other entities, the higher is the requirement and its complexity rate by metric.

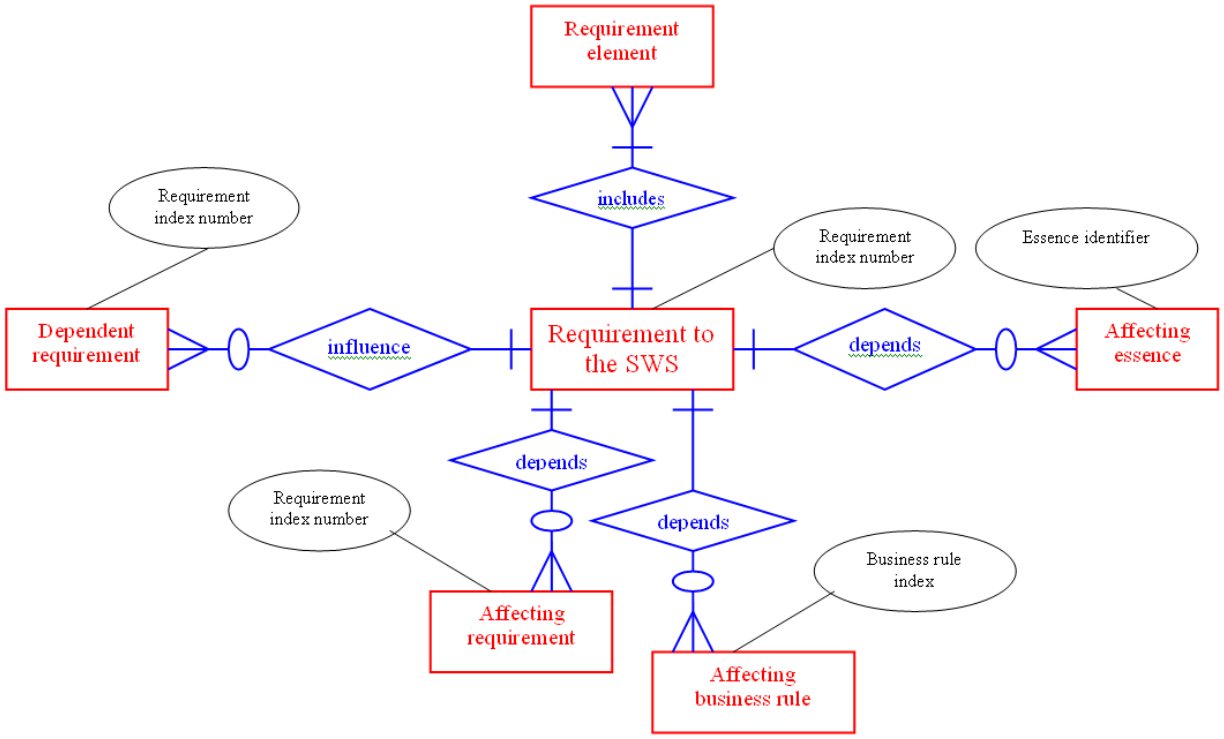

Fig. 1. ER – diagram of logical interconnections of the requirement with other components.

Table 1 contains proposed requirement complexity metric structure and examples of its evaluation. Data required to evaluate requirement complexity by metric is, as follows:

1. Individual requirement index within the system being under development;
2. Number of elements composing the requirement (e.g, to calculate tax rate (variable 1st element) multiplied (operation 2nd element) by 18% rate (constant value, 3rd element) from income value (variable, 4th element) deducting (variable, 5th element) maternity leave (variable, 6th element) and amount payable under sick leave (variable, 7th element);
3. Number of affecting business rules (e.g. global – currency exchange rate, regional – applicable tax rate, corporate – terms to earn quarter bonus in the corporation);
4. Number of affecting and dependent requirements;
5. Number of other affecting entities (e.g. hardware capacity, network traffic speed).

Table 1. Requirement Complexity Metric Structure.

Requirement Complexity Characteristics					Requirement Complexity Metric 6=2+3+4+5
Unique Requirement Index	Internal Requirement Complexity	External Requirement Complexity			
	Number of Components composing the Requirement	Number of Affecting Business Rules	Number of Affecting and Dependent Requirements	Number of Affecting and Dependent Entities	
-1-	-2-	-3-	-4-	-5-	-6-
F63	2	1	5	1	9

F64	3	0	2	2	7
F65	2	0	2	0	4
F66	5	0	1	2	8
F67	2	0	1	0	2
F68	4	3	0	1	8
Mean Requirements' Complexity					6,3

Mathematical representation of the requirement complexity evaluation by metric (requirement complexity metric, RCM) looks, as below:

$$RCM = EQ + IBRQ + IDRQ + IDEQ \quad (1)$$

EQ – the elements quantity (column 2), *IBRQ* - the influencing business rules quantity (column 3), *IDRQ* – quantity of the affecting and dependent requirements (column 4), *IDEQ* - quantity of the influencing and dependent entities (column 5).

Such is the proposed requirement complexity evaluation metric. Calculation of metric consists in summing up the numerical characteristics in internal and external complexity of a particular requirement. The metric enables to evaluate the requirement complexity in complexity units. Obtained values enable to sort the requirements by their complexity level.

4.3. RCM Metric Practical Application

The proposed *RCM* metric has been applied for functional requirements complexity evaluation within a particular corporate system of accounting and analysis. This medium-size system has been developed in Ukraine for the Danube Institute of National University "Odessa Maritime Academy" in 2016. The realized system consisted of 65 components and a database of 26 interlinked tables. Total size of the system amounted to 40000 lines of initial code. System specification encompassed 120 functional requirements. Requirements analysts calculated complexity rates applying the RCM metric and basing on Table 1. The rates in question have been obtained in the course of collection, review and documenting the requirements. The evaluation process was rather easily performed in MS Excel and did not demand additional time or additional software and depended only on organization aspect and personal qualities of experts. Requirements mean complexity RC_{avg} has been calculated being 6,3 complexity units. Total system complexity amounted to $120 \cdot 6{,}3 = 756$ complexity units. The obtained value has been applied to compare complexity of existing and previously developed systems enabling to update development terms and budget. It was also revealed that 15 % requirements had complexity characteristics exceeding 10 units (*RC>10).* The analysis showed that those requirements had been hard enough to be realized. Detailed review of the requirements in question enabled the experts to form a complete idea of the complexity of the system being developed and to involve necessary resources. Certain complicated requirements have been modified or decomposed into easier. Some requirements, after discussion with the customers have been deleted from specification. After these activities total complexity of the system reduced to 83 units, i.e. by 11%.

4.4. Algorithm of SWS Requirement Correctness Improving

The offered method is based on logical admission that the more is requirement complexity, the more actual and potential defects it contains, such as contr5adictions, discrepancies, incompleteness, inaccuracies which at further stages of system design may likely become defects. Really, the more elements are assembled into a requirement, the higher is chance of errors in elements. The more entities are interconnected with the requirement, the harder it is to describe the requirement clearly, correctly, to codify it properly and to modify it in the course of the SWS maintenance.

The method may be applied once requirement construction process is completed and priorities of their realizations are assigned. Method utilized the proposed RCM metric to evaluate the requirements' complexity. *The method's incoming data* contains characteristics of each requirement.

1. Unique requirement index;
2. Detailed text description of the requirement;
3. Priority of realization of the requirement *Pr* (high *Pr=1*, medium *Pr=2*, low *Pr=3*).

This data is accumulated in the course of requirements construction and does not demand any additional activities and costs. Method procedure includes seven (7) steps:

Step 1. Full set of requirements should be sorted in the order from higher to lower.

Step 2. Subset requirements with $Pr = 1$ R_{pr1} should be derived from R_{full} set. Complexity values should be calculated using metric RCM according to expression (1). The requirements should be additionally sorted according to complexity decreasing criterion. Obtained *List 1* should contain rising rank to each requirement starting from 1 and further to n.

Step 3. Requirements R_{pr1} should be verified. Real and potential requirements' defects should be detected and eliminated (contradiction, mismatching, incompleteness, inaccuracies, in ambiguity).

Step 4. Resources permitting, derive requirements subset R_{pr2} with priority $Pr = 2$. Complexity values should be calculated using metric RCM according to expression (1). The requirements should be additionally sorted according to complexity decreasing criterion. Obtained *List 2* should contain rising rank to each requirement starting from $n+1$ and further to k. Further proceed to steps 3, 5, 6,7. With lack of resources proceed to Steps 6 and 7.

Step 5. Resources permitting, derive requirements subset R_{pr2} with priority $Pr = 3$. Complexity values should be calculated using metric RCM according to expression (1). The requirements should be additionally sorted according to complexity decreasing criterion. Obtained *List 3* should contain rising rank to each requirement starting from *k+1*. Further proceed to steps 3, 6, 7. With lack of resources proceed to Steps 6 and 7.

Step 6. Requirements Correctness Evaluation. Having completed the verification quantity of requirements should be counted by categories, including total verified R_{total}; defective, fixed R_{fix}; defective deleted R_{del}, potentially defective, modified R_{mod}; skipped R_{gap}. The *requirements defectiveness* index **RDI** may be calculated using the formula (2) below

$$RDI = \frac{R_{fix} + R_{del} + R_{mod} + R_{gap}}{R_{total}} \quad (2)$$

The requirements correctness index, **RCI,** should be calculated using the formula (3) below:

$$RCI=1-RDI \quad (3)$$

Step 7. **Requirements correctness improvement.** Comparing index values of *RCI, RDI* with known under previous projects (if available) and/or permissible corporative values. Referring to the comparative results conclusion should be drawn about achieved requirements' correctness and finalizing or further proceeding with their construction stage. E.g. permissible corporative value for $RCI_{norm} = 0.92$ with $RDI_{norm} = 0.08$, i.e. not more than 8 defective of 100 total requirements are permitted. Should $RCI \geq 0.92$, the requirements are correct enough and fit for further development stages. Otherwise (RCI < 0.92), the requirements are not correct and demand re-proceeding and repeated performance of the procedure under offered methodology.

This step finalizes the method. Table 2 shows an example of ranking the requirements.

Table 2. Requirement Ranking Example**.**

Individual Requirement Index	Requirement Priority	Requirement Complexity Grade by Metric (1, ∞)	Requirement Rank (1, ∞)
F63	1	9	1
F64	1	7	2
F65	2	4	4
F66	2	8	3
F67	3	5	6
F68	3	8	5

The top rank for requirements with priority 1 is caused with the highest evaluated complexity rate (complexity rate = 9, rank = 1). As complexity rate decreases the rank decreases as well (complexity rate = 8, rank = 2). The highest rank for requirements with priority 2 should be equal to next after last assigned rank. The top rank for priority 2 requirements is equal to the rank with index following the last assigned rank. It is assigned to the requirement with the highest complexity index (complexity index = 7, rank = 3).

Figure 2 shows the UML-diagram of activities under the method. The UML diagram represents logical consistence of actions within the method.

After the method is completed the final data is obtained as a through *ranked list of requirements and defectiveness and correctness indexes*. The through ranked list enables to perform individual verification of the top priority and most complex requirements and to increase quantity of detectable potential and actual defects. Defectiveness and correctness indexes indicate either requirements' fitness to further stages of processing, or necessity of their reprocessing with further repeated verification. The method enables to reduce defectiveness and to improve correctness of requirements with prevailing restricted resources situation.

5 Case Study. Conclusions

The offered method has been applied in the course of inspection of functional requirements of corporate system, as referred above. Through ranked script of the requirements of top priority and the highest complexity has been applied. Methodology enabled to increase quantity of detectable actual and potential defects by 9% in average in comparison with total verification of the requirements disregarding their complexity, improve requirement correctness to the same extent and to avoid sensible costs which might be incurred to eliminate such defects at further stages of development.

The most important SRM principles have been outlined in this work:

1. *Compatibility* of models methods and technologies with SWS development business processes;
2. *Complexity evaluation* for SWS development process and artifacts reliability management throughout the SWS LC;
3. Development of methods and applications for analysis of *Big Data* SWS LC;
4. Maximum possible reliability improvement with *restricted resources* situation.

Requirement complexity evaluation metric (RCM) is offered within the SRM framework. The metric calculation consists of summing up quantitative characteristics of internal and external requirement complexity. The metric evaluates requirement complexity in complexity units. The obtained evaluation enables to determine total system complexity and adjust terms and costs of development. Modifications or decomposition of complex requirement enables to reduce total system complexity.

Method of improving requirements correctness has been offered within the SRM framework. The method applies an admission that more complex requirements contain more actual and potential defects. The method applies RCM metric to evaluate requirements complexity. The method applies a double sorting procedure basing on requirements' priority and complexity evaluation. The method is advantageous comparatively to the total verification due to possibility to select the most complicated requirements with top priority by means of generating their through ranked script. The method enables to improve requirements' correctness and defectiveness. The method enables to evaluate achieved indexes of correctness and defectiveness. The application of the method in the course of requirements verification provided substantial technological and economical effect.

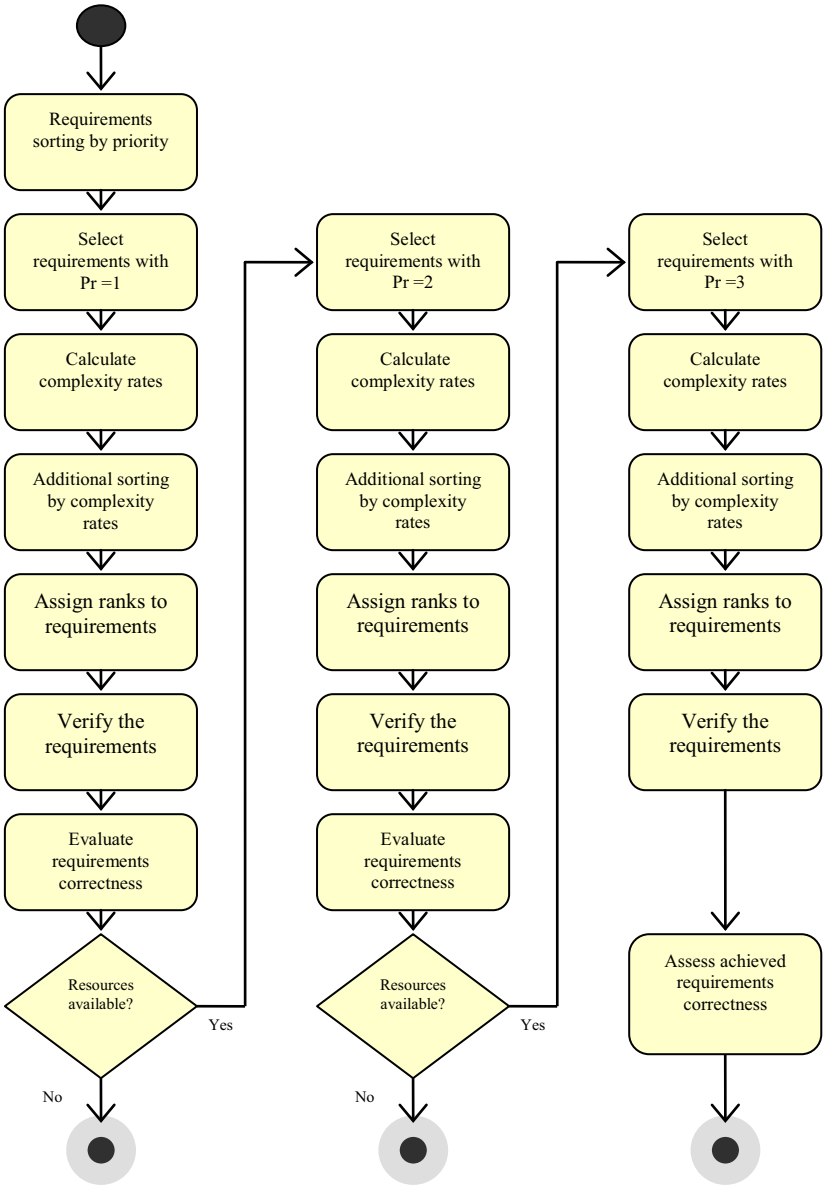

Fig. 2. UML - diagram of activity of SWS requirements correctness improving technique.

The method has been developed taking into account two principles: the principle of artifacts complexity evaluation and the principle of maximum possible reliability improvement in restricted resources situation.

Further researches are planned in two directions. The first lies in achieving and providing organic building-in with different SWS development business processes by means of making method patterns with defining roles and activities of the developers. The second direction

lies in development of methods and applications for Big Data analysis occurring at all stages of the SWS LC.

References

1. E. Eric. Domain-Driven Design - Tackling Complexity in the Heart of Software. Addison Wesley, 560 p. (2013)
2. Frischknecht Ch. Top 10 Software Fails Of 2014. Access mode: http://www.tricentis.com/blog/2014/12/18/top-10-software-fails-of-2014 (2014)
3. G. Carrozza, R. Pietrantuono, S. Russo. Defect Analysis in Mission-critical Software Systems: a Detailed Investigation. J. Softw. Evol. and Proc. ; 2012/07/12, Vol.**2**, pp. 1-28. DOI: 10.1002/smr (2012)
4. Cotroneo, D., Pietrantuono, R., Russo, S. Testing techniques selection based on ODC fault types and software metrics. The Journal of Systems and Software, Vol.**86**, pp. 1613-1637 (2013)
5. Cotroneo, D., Pietrantuono, R., Russo, S. Combining Operational and Debug Testing for Improving Reliability. The Journal of Systems and Software, Vol. **62**, no. 2, pp. 408- 423 (2013)
6. V.S.Kharchenko, V.V.Sklar, O.M.Tarasyuk. Methods for modeling and evaluation of the quality and reliability of the software. Kharkov: Nat. Aerospace. Univ."KhAI", 159 p. (2004)
7. Yaremchuk, S., Kharchenko, V. Complexity-based Prediction of Faults Number for Software Modules Ranking Before Testing: Technique and Case Study. Proceedings of the 12th International Conference, ICTERI 2016. Kyiv, Ukraine, Vol. **1614**, pp. 461–474 (2016)
8. V. Yakovyna, D. Fedasyuk, O. Nytrebych, Iu. Parfenyuk, V. Matselyukh. Software reliability assessment using high-order Markov chains. International Journal of Engineering Science Invention. — Vol. **3**, Issue 7. – pp. 1–6 (2014)
9. Yakovyna V. S. Software failures prediction using RBF neural network. Researches of the Odessa national polytechnical university. — Vol. **2**(46). – pp. 111–118 (2015)
10. D. A. Maevsky, S. A. Yaremchuk, L. N. Shapa. A method of a priori software reliability evaluation. Reliability: Theory & Applications. — Vol. **9**. – № 1(31). – pp. 64 – 72 Access mode: http://www.gnedenko-forum.org/Journal/2014_1.html (2014)
11. S. A. Yaremchuk, D. A. Maevsky. The Software Reliability Increase Method. Studies in Sociology of Science. Vol. **5**, No. 2 pp. 89 – 95. Access mode: http://www.cscanada.net/index.php/sss/article/view/4845 (2014)
12. 12. Eric J. Braude. Software Engineering An Object-Oriented Perspective. John Wiley and Sons, 575 p. (2011)
13. Kelly, John C., Joseph S. Sherif, Jonathon Hops. An Analysis of Defect Densities Found During Software Inspections. Journal of Systems and Software Vol. **17**(2), pp. 111-117 (1992)
14. Hofmann, Hubert F., Franz Lehner. Requirements Engineering as a Success Factor in Software Projects. IEEE Software Vol. **18**(4):58-66 (2001)
15. IEEE 830-1998: IEEE Recommended Practice for Software Requirements Specifications. Los Alamitos, CA: IEEE Computer Society Press (1998)
16. IEEE 1233-1998: IEEE Guide for Developing System Requirements Specifications. Los Alamitos, CA: IEEE Computer Society Press (1998)
17. IEEE/ISO/IEC 29148-2011: Systems and software engineering – Life cycle processes – Requirements engineering (2011)
18. Wiegers K., Beatty J. Software Requirements, 3rd Edition. Microsoft Press, 673 p. (2013)
19. IEEE 1061-1992: A Software Quality Metrics Methodology. Los Alamitos, CA: IEEE Computer Society Press (1992)
20. Shahid Iqbal, M. Naeem Ahmed Khan. Yet another Set of Requirement Metrics for Software Projects. International Journal of Software Engineering and Its Applications Vol. **6**, No. pp.19-28 (2012)
21. Mohd. Haleem, Mohd.Rizwan Beg, Sheikh Fahad Ahmad. Overview of Impact of Requirement Metrics in Software Development Environment. International Journal of Advanced Research in Computer Engineering & Technology (IJARCET) Vol. **2**, No 5, pp. 1811-1815 (2013)