Method for Determining the Informativeness of the Software Requirements Specifications

Ivan Lopatto[a], Mykyta Lebiga[a], Yurii Forkun[a] and Artem Boyarchuk[b]

[a] *Khmelnytskyi National University, Institutska str., 11, Khmelnytskyi, 29016, Ukraine*
[b] *National Aerospace University "Kharkiv Aviation Institute", Chkalova str., 17, Kharkiv, 61000, Ukraine*

Abstract

A critical influence on software projects and on the success of their realization is exerted by issues related to the analysis and evaluation of the software requirements specifications (SRS). Today, when the number of high-budget software projects is growing rapidly, it is important to analyze the SRS, in particular, to determine their informativeness to assess the quality of software according to ISO 25010: 2011. The conducted analysis of the known tools for SRS' analysis showed that none of the known tools provides the possibility of determining the informativeness of the requirements specification. Therefore, it is necessary to design and implement the criteria and method for determining the informativeness of the SRS (which would be the theoretical basis for developing a future tool for determining the informativeness of requirements specifications), which is the purpose of this study. The paper proposes criteria and a method for determining the informativeness of the software requirements specifications, which determine the informativeness of the specification in terms of determining on its basis the characteristics of the software quality. The proposed criteria and method for determining the informativeness of the specifications of the software requirements allow determining the informativeness of the specification in terms of determining on its basis each of the 8 characteristics of the software quality.

Keywords

Software requirements specification (SRS), the informativeness of the requirements specification, criteria for determining the informativeness of the requirements specifications, method for determining the informativeness of the requirements specifications.

1. Introduction

The software development life cycle begins with the formation of a set of requirements and specification of software requirements based on them.

The largest number of software bugs are made at the stage of requirements formation [1, 2] – by the statistics, 56% of all software bugs are introduced at the stage of formation of requirements; about 50% of the bugs in requirements are the result of poorly written, unclear, ambiguous or incorrect requirements; the other 50% are due to incomplete specifications (incomplete and omitted requirements) [3]. The distribution of software bugs by development phase is represented in Figure 1. Clear requirements have 13% of the weight in project success factors [4]. The source of 56 percent of all software bugs is the requirements formation phase (Figure 2) [3]. Statistics show that inaccurate requirements are one of the primary causes of software failures [5, 6]. Software projects, the specification of the requirements for which contains insufficient, inaccurate, incomplete, and contradictory information, cannot be successfully realized [2]. The vast majority of software-related

IntelITSIS'2021: 2nd International Workshop on Intelligent Information Technologies and Systems of Information Security, March 24–26, 2021, Khmelnytskyi, Ukraine
EMAIL: ivan.lopatto@gmail.com (I. Lopatto); lebiganikita1996@gmail.com (M. Lebiga); forkun@ridne.net (Y. Forkun), a.boyarchuk@csn.khai.edu (A. Boyarchuk)
ORCID: 0000-0001-6886-2238 (I. Lopatto); 0000-0002-5849-1514 (M. Lebiga); 0000-0002-7906-4191 (Y. Forkun), 0000-0001-7349-1371 (A. Boyarchuk)

accidents are caused by erroneous requirements, not coding errors. 60% time and cost are paid on projects with requirements of poor quality [3].

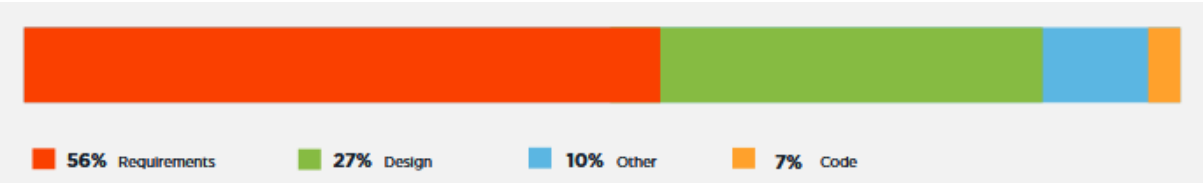

Figure 1: Distribution of defects in software projects by development phase [3]

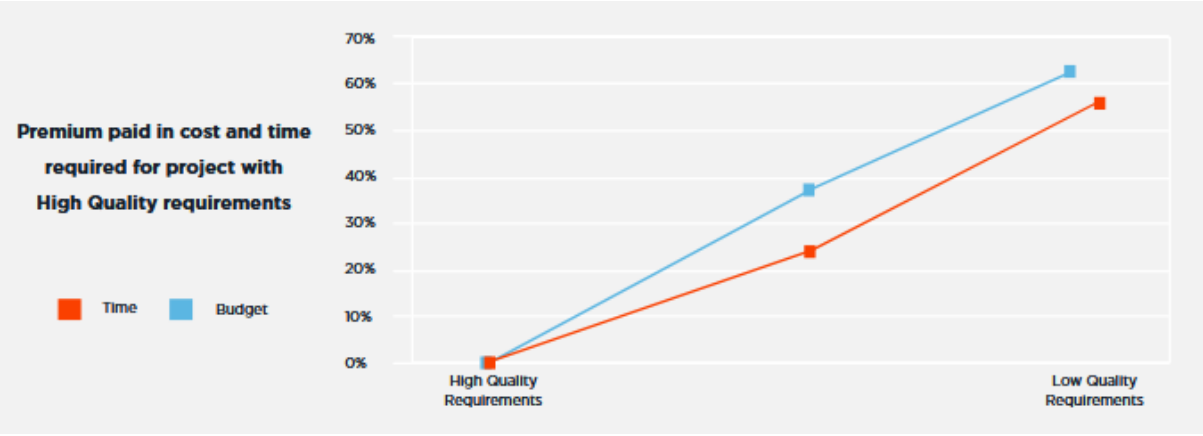

Figure 2: Time and cost premiums on requirements of low quality [3]

82% of application rework is related to requirements bugs [3]. The earlier the bug (error, violation, defect, and malfunction) is detected, the cheaper it will be to correct it. As the interval between the time of introduction and detection of the defect increases, the cost of its correction increases sharply. The longer the bug persists in the software development chain, the more it penetrates other parts of the software, the more damage it causes in the following stages, and the more money will have to be spent on its elimination. By the statistics, the cost of correcting the software bugs, which are made in the early stages of the life cycle, increases exponentially with each following stage of the life cycle (Figure 3) [3]. Figure 3 shows a comparison of the system cost-to-fix results NASA obtained from various methodologies, while Figure 4 compares system results with the software cost models from the earlier examined studies [3]. The cost of correcting incorrect requirements in the specification, which are identified after the release of the product, is almost 100 times higher than the cost of correcting the shortcomings of the specification, which were identified at earlier stages, in particular, at the same stage of requirements' formation [7].

The importance and impact of business requirements on the success of the enterprise through software projects found that 68% of companies suffer from the poor practice of technical requirements and that these companies [3]:

- Spent 49% more money on application delivery
- 79% of projects have overtime and overbudget
- More than 39% of the time was spent on application delivery
- More than 41.5% of new resources were used to develop projects for poorly defined requirements

In fact, the study [3] showed that 79% of companies use “common” (unstructured) natural language requirements' documents, 16% use “structured” natural language, templates and forms. Only 5% of the surveyed companies said they were using formal approaches like Model-based system engineering (MBSE) (Figure 5).

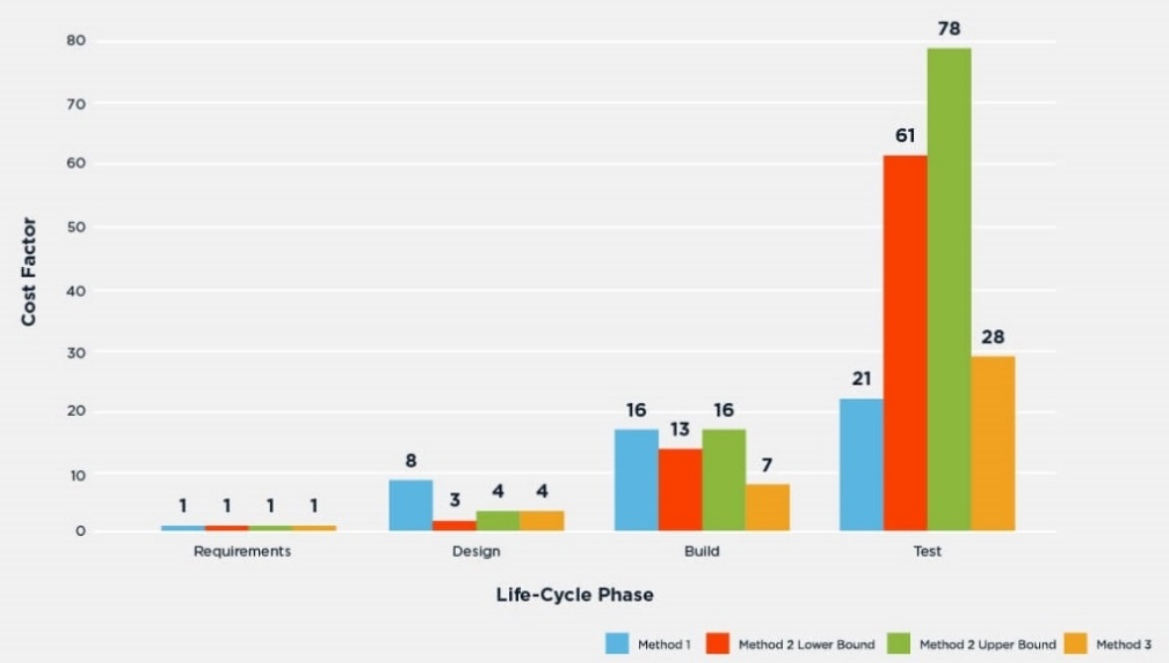

Figure 3: Comparison of System Cost Factors – Excluding Operations [3]

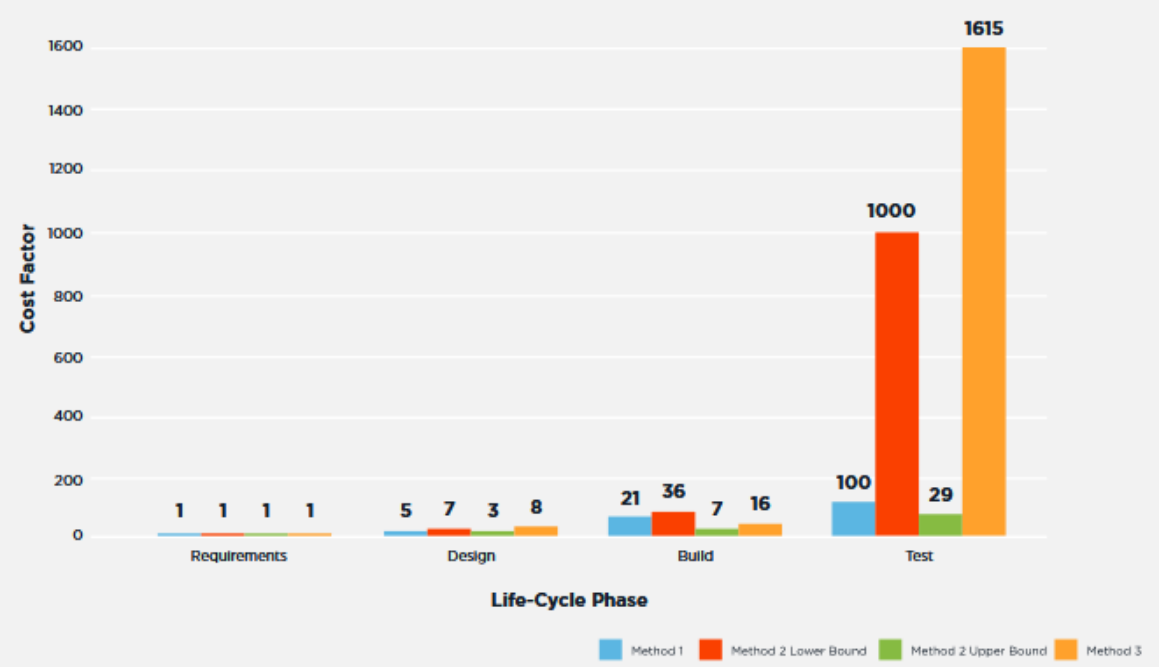

Figure 4: Comparison of Software and System Cost Factors [3]

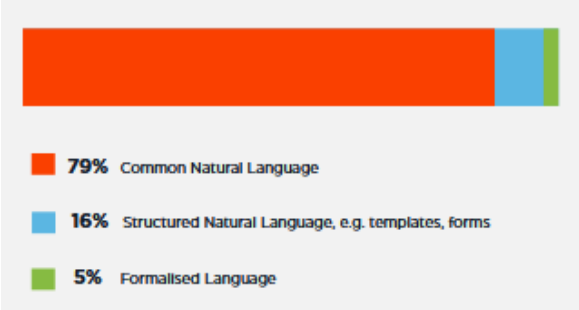

Figure 5: Time and cost premiums on low quality requirements [3]

The risks of an insufficiently worked out stage of requirements formation are non-compliance with project deadlines and financial overspending, which can lead to the closure of the project, and even the collapse of the software company due to its financial instability [8-13].

In the requirements formation process, information losses emerge because of incompleteness and difference of understanding of the users' needs and context of information – especially such losses are significant for multidisciplinary software projects that are developed at the intersection of subject areas [8-18]. Therefore, for ensuring the quality and safety of software, it is necessary to conduct a study of software requirements – in order to identify and eliminate problems and shortcomings in the early stages of the software life cycle and identify the lack of relevant information.

Thus, the critical impact on software projects and the success of their implementation are issues related to the analysis and evaluation of the initial stages of the life cycle. Today, when the number of high-budget software projects is growing rapidly, it is important to analyze the specifications of software requirements, in particular, to determine their informativeness for assessing the quality of software according to ISO 25010: 2011.

2. State-of-the-art

Today, the key to assessing the quality of the software is the approach based on the SQuaRE model (ISO 25010:2011 standard). Software quality assessment according to ISO 25010:2011 [19] is as follows: based on the quality attributes specified in ISO 25023: 2016 [20], sub-characteristics and quality characteristics are evaluated.

Important criteria for specifying software requirements in terms of future software quality assessment are:

1. Completeness of the requirements specification
2. Sufficiency of information in the requirements
3. Informativeness of the requirements specification

Let's explore the known tools of the analysis of requirements in order to determine such criteria, as well as the possibility of their free use (Table 1).

Table 1

State-of-the-art on known tools for software requirements specifications' analysis

Tools	Completeness of the requirements specification	Sufficiency of information in the requirements	Informativeness of the requirements specification	Free access
CORE [21]	+	-	-	?
Visure [22]	+	-	-	?
Accompa [23]	+	-	-	-
Innoslate [24]	+	-	-	-
ReqView [25]	+	-	-	?
Modern Requirements4TFS [22]	+	-	-	?
QVscribe [3]	+	-	-	?
Requirements Analysis Tool [23]	+	-	-	?
QARCC [24]	+	-	-	?
Requirements Assistant [25]	+	-	-	-
QuARS Requirements Analysis [24, 26]	+	-	-	-
DESIRe [27]	+	-	-	-
RQV Tool [27]	-	-	-	-
IT for assessing the initial stages of the software life cycle [9, 10]	-	+	-	+
Software quality evaluation and prediction system [28, 29]	-	-	-	+

Thus, as the conducted analysis of the known tools for software requirements specifications' analysis showed that none of the known tools provides the possibility of determining the informativeness of the requirements specification. Therefore, it is necessary to develop and implement criteria and method for determining the informativeness of software requirements specifications (which would be the theoretical basis for developing a future tool for determining the informativeness of requirements specifications), which is the *purpose of this study*.

3. Criteria and method for determining the informativeness of the software requirements specifications

Criteria for evaluating information, according to [30, 31], are:

1. Relevance – determining whether this information can help in the future to assess the quality of software according to ISO 25010
2. Veracity – the extent to which the description of the information is true; how the description of the information corresponds to reality
3. Significance – understanding of information, completeness of coverage of the subject of interest, timeliness of the information and its sufficiency for decision-making
4. Novelty – how the information's description is new for the user
5. Actuality – the degree of compliance of information with the current time
6. Objectivity – characterizes the independence of information from someone's opinion or consciousness, as well as from methods of obtaining
7. Completeness – information can be considered complete if it contains a minimum set of quality measures, but sufficient to make a correct decision
8. Compliance – the degree of compliance of information to the needs of consumers
9. Adequacy – compliance with the content of the actually obtained information and its expected content
10. Accessibility – a measure of the ability to obtain information

To assess the informativeness of the specification of software requirements, it is necessary to assess the informativeness of the quality measures of the specification in terms of their impact on the characteristics of the software quality.

Criteria of informativeness of quality measures of the SRS:

1. Relevance Rv = {Rsp, Abci}, where Rsp € [0;1] – the rank of the influence of the quality measure of the specification on a certain characteristic, where 0 – corresponds to the smallest influence of the quality measure of the specification on the characteristic, 1 – corresponds to the greatest influence; Abci € [0;1] – the rank of the ability of the quality measure to clarify the value of the software quality characteristic, where 0 means that the measure least clarifies the value of the characteristic, 1 – means the greatest degree of clarification
2. Veracity Rb € [0;1], where 1 – it's possible to trust to the measure of determining the characteristic of software quality; 0 – cannot be trusted in principle
3. Significance Sf = {Ust, Cnc}, where Ust € [0;1] – the rank of understanding the quality measure, where 0 – the measure is unclear, 1 – the measure is clear; Cnc € [0;1] – the rank of completeness of the coverage of the quality characteristic by the measure, where 0 – the measure least covers the characteristic, 1 – the most illuminates the characteristic
4. Objectivity Obj € [0;1], where 0 – quality measure depends on someone's opinion or consciousness, as well as on the methods of obtaining, 1 – independent measure
5. Compliance Qi € [0;1], where 0 – the lowest degree of conformity of the quality measure to the characteristic, 1 – the highest degree

As for other criteria for evaluating information, they will not be needed to assess the informativeness of the software specification due to their redundancy and repeatability.

Taking into account the above criteria of informativeness of the measures of the specification of requirements, *the method of determining the informativeness of the specification of software requirements* consists of the following stages:

1. Division of the interval into subintervals for each software quality characteristic

2. Evaluation of all measures of each quality characteristic according to each of the criteria of informativeness of quality measures of the specification of requirements
3. Sorting of quality measures of the specification of requirements and assignment to each measure of the corresponding rank
4. Formation of a matrix of informativeness of measures for each characteristic of the software quality
5. Calculation of the indicator of informativeness of the specification on the definition of each of the software quality characteristics

The interval [0; 1] is divided into subintervals depending on how many quality measures of the specification affect the quality characteristic:

$$Quantity\ of\ subintervals = Quantity\ of\ different\ measures - 1. \quad (1)$$

Then for an ideal specification, in which there are absolutely all the measures on which the quality characteristics depend, the number of subintervals, the step and the division of the interval into subintervals for each software quality characteristic are presented in Table 2.

Table 2

The number of subintervals, the step and the division of the interval into subintervals for each software quality characteristic – for an ideal specification, in which there are absolutely all the measures on which the quality characteristics depend

Quality characteristic	Quantity of different measures	Quantity of subintervals	Step	Dividing the interval into subintervals (to determine the ranks)
Functional Suitability	9	8	1/8=0.125	[0; 0.125;…; 0.875; 1]
Performance Efficiency	23	22	1/22=0.0455	[0; 0.0455;…; 0.9545; 1]
Usability	44	43	1/43=0.0233	[0; 0.0233;…; 0.9767; 1]
Reliability	25	24	1/24=0.0417	[0; 0.0417;…; 0.9583; 1]
Compatibility	8	7	1/7=0.1429	[0; 0.1429;…; 0.8571; 1]
Security	15	14	1/14=0.0714	[0; 0.0714;…; 0.9286; 1]
Maintainability	26	25	1/25=0.04	[0; 0.04;…; 0.96; 1]
Portability	16	15	1/15=0.0667	[0; 0.0667;…; 0.9333; 1]

Then there is an expert assessment of the importance of each measure of quality characteristic according to a certain criterion, sorting of measures is performed, and each measure is assigned a corresponding rank.

The result of such assessments is the matrix of informativeness of measures for each j-th characteristic (j=1..8):

$$I_j = \begin{vmatrix} Rsp_{1j} & Abci_{1j} & Rb_{1j} & Ust_{1j} & Cnc_{1j} & Obj_{1j} & Qi_{1j} \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ Rsp_{nj} & Abci_{nj} & Rb_{nj} & Ust_{nj} & Cnc_{nj} & Obj_{nj} & Qi_{nj} \end{vmatrix}$$

where the first line contains estimates for the measure №1 of j-th characteristic, the second line contains estimates for the measure №2 of j-th characteristic, etc., the last line contains estimates for the measure №n of j-th characteristic (number of measures are different for each characteristic and maximum of the number of measures are presented in Table 2 for each characteristic). Thus, we obtain 8 matrices of informativeness of measures for 8 main characteristics of software quality.

Next, it is necessary to assess the informativeness of the specification for the definition of each of the characteristics. To do this, it's necessary to calculate for each matrix the number of elements belonging to a certain range – Qii^j, after this to find the informativeness of the specification for determining each of the characteristics as the ratio of the obtained number Qii^j to the total size of each j-th matrix ($7*n^j$):

$$II\_SRS^j = \left(\frac{Qii^j}{7 * n^j}\right) * 100\%. \quad (2)$$

The obtained indicator will reflect the informativeness of the specification for determining the j-th characteristic of the software quality.

Thus, the criteria and method for determining the informativeness of the specifications of the software requirements are proposed, which determine the informativeness of the specification in terms of determining on its basis the characteristics of the software quality.

4. Results & discussion

For example, let's consider the measures on which Functional Suitability depends (according to ISO 25010, ISO 25023):

1. Number of Functions
2. Functional Implementation Completeness
3. Functional Adequacy
4. Functional Implementation Coverage
5. Operation Time
6. Number of Inaccurate Computations Encountered by Users
7. Number of Data Items
8. Computational Accuracy
9. Precision

The number of measures on which the Functional Suitability characteristic depends is 9. All 9 measures of the Functional Suitability are present in the first considered specification. Then the number of subintervals is 8, so the step is 1/8 = 0.125. We then divide the interval as follows: [0; 0.125; …; 0.875; 1]. Next, we evaluate with the help of experts each measure of the characteristic according to a certain criterion, sort the measures and assign each measure the appropriate rank.

For example, let's consider how the experts recommended assessing the influence of each quality measures of the specification on the characteristic Functional Suitability. Thus, the most influential, according to experts, is the measure Number of Functions (rank 1), next in influence is the measure Operation Time (rank 0.875), then the measure Functional Implementation Completeness (rank 0.75), then Functional Implementation Coverage (rank 0.625), then Functional Adequacy (rank 0.5), then Precision (rank 0.375), then Number of Data Items (rank 0.25), then Computational Accuracy (rank 0.125), and the least influential is the Number of Inaccurate Computations Encountered by Users (rank 0).

Similarly, the experts recommended assessing the rank of the ability of each quality measure to clarify the value of Functional Suitability, the veracity of each measure to determine the characteristic Functional Suitability, the rank of understanding of each quality measure of Functional Suitability, the rank of completeness of the coverage of Functional Suitability by each measure, the objectivity of each measure of the Functional Suitability characteristic, compliance of each measure concerning the definition of the Functional Suitability characteristic.

The result of such estimates is a matrix of informativeness of measures for the Functional Suitability characteristic:

I_{FS}^1 =

1	0.625	1	0.875	1	0.875	0.625
0.75	1	0.875	0.625	0.875	0.25	0.875
0.5	0.875	0.75	0.375	0.625	0.125	0.75
0.625	0.75	0.625	0.5	0.75	0	1
0.875	0.5	0.5	1	0.375	0.75	0.375
0	0.125	0	0	0.25	0.375	0.25
0.25	0.375	0.375	0.75	0.5	1	0.5
0.125	0.25	0.25	0.125	0.125	0.5	0
0.375	0	0.125	0.25	0	0.625	0.125

Let's calculate for the obtained matrix the number of elements that belong to a certain range – Qii^j. For example, let's calculate the number of elements that belong to the range [0.5; 1], because experts recommend just such an interval for Functional Suitability. Under such conditions, $Qii^j = 35$ for the obtained matrix.

Then the informativeness of the specification for determining the Functional Suitability characteristic is:

$$II\_SRS^{FS1} = \left(\frac{35}{7 * 9}\right) * 100\% = 55.56\%.$$

Only 5 measures of the Functional Suitability (1.Number of Functions; 2.Functional Implementation Completeness; 3.Functional Adequacy; 4.Functional Implementation Coverage; 5.Operation Time) are present in the second considered specification. Then the number of subintervals is 4, so the step is 1/4 = 0.25. We then divide the interval: [0; 0.25; …; 0.75; 1]. Next, we evaluate with the help of experts each measure of the characteristic according to a certain criterion, sort the measures and assign each measure the appropriate rank. For example, the influence of each quality measures on the Functional Suitability: Number of Functions – rank 1, Operation Time – rank 0.75, Functional Implementation Completeness – rank 0.5, Functional Implementation Coverage – rank 0.25, Functional Adequacy – rank 0.

Similarly, the experts recommended assessing the remaining ranks of each of 5 present quality measures of Functional Suitability.

The result of such estimates is a matrix of informativeness of 5 measures for the Functional Suitability characteristic:

I_{FS}^2 =

1	0. 25	1	0.75	1	1	0.25
0.5	1	0.75	0.5	0.75	0.5	0.75
0	0.75	0.5	0	0.25	0.25	0.5
0.25	0.5	0.25	0.25	0.5	0	1
0.75	0	0	1	0	0.75	0

Let's calculate for the obtained matrix the number of elements that belong to a certain range – Qii^j. For example, let's calculate the number of elements that belong to the range [0.75; 1], because experts recommend just such an interval for Functional Suitability, if Functional Suitability depends on not all 9 measures, but only 5 measures. Under such conditions, $Qii^j = 14$ for the obtained matrix.

Then the informativeness of the specification for determining the Functional Suitability characteristic is:

$$II\_SRS^{FS2} = \left(\frac{14}{7 * 5}\right) * 100\% = 40\%.$$

It is obvious that if the specification of requirements contains a smaller number of quality measures for a certain characteristic, then the informativeness of the specification for determining such a characteristic is lower (in the considered examples – by 15.56% while reducing the number of measures by 4). Similarly, the matrices of the informativeness of the remaining 7 software quality characteristics can be obtained and, accordingly, the informativeness of the specification for determining each of the remaining 7 software quality characteristics can be calculated.

Therefore, the proposed criteria and method for determining the informativeness of the specifications of the software requirements allow determining the informativeness of the specification in terms of determining on its basis each of the 8 characteristics of software quality.

5. Conclusions

A critical influence on software projects and on the success of their realization is exerted by issues related to the analysis and evaluation of the software requirements specifications (SRS). Today, when the number of high-budget software projects is growing rapidly, it is important to analyze the SRS, in particular, to determine their informativeness to assess the quality of software according to ISO 25010.

The conducted analysis of the known tools for SRS' analysis showed that none of the known tools provides the possibility of determining the informativeness of the requirements specification. Therefore, it is necessary to design and implement the criteria and method for determining the informativeness of the SRS (which would be the theoretical basis for developing a future tool for determining the informativeness of requirements specifications), which is the purpose of this study.

The paper proposes criteria and a method for determining the informativeness of the software requirements specifications, which determine the informativeness of the specification in terms of determining on its basis the characteristics of the software quality. The proposed criteria and method allow determining the informativeness of the specification in terms of determining on its basis each of the 8 characteristics of the software quality.

6. References

[1] S. McConnell, Code complete, Microsoft Press, Redmond, 2013.

[2] N. Levenson, Engineering a safer world: systems thinking applied to safety, MIT Press, Massachusetts, 2012.

[3] Leveraging natural language processing in requirements analysis: How to eliminate over half of all design errors before they occur, 2020. URL: https://qracorp.com/wp-content/uploads/2020/10/Leveraging-NLP-in-Requirements-Analysis.pdf.

[4] 4-Step project plan for 2020, 2020. URL: http://projexs.io/project-plan-made-easy/.

[5] PMI's Pulse of the Profession 9-th Global Project Management Survey, 2017. URL: https://www.pmi.org/-/media/pmi/documents/public/pdf/learning/thought-leadership/pulse/pulse-of-the-profession-2017.pdf.

[6] Sh. Sarif, S. Ramly, R. Yusof, N. Fadzillah, N. Sulaiman, Investigation of success and failure factors in IT project management. Advances in Intelligent Systems and Computing 739 (2018) 671-682. doi: 10.1007/978-981-10-8612-0_70.

[7] C. Shamieh, Systems engineering for dummies, Wiley Publishing, Hoboken, 2014.

[8] O. Pomorova, T. Hovorushchenko, The way to detection of software emergent properties, in: Proceedings of the 2015 IEEE 8-th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS'2015, Warsaw, 2015, vol. 2, pp. 779-784. doi: 10.1109/IDAACS.2015.7341409.

[9] T. Hovorushchenko, O. Pavlova, M. Bodnar, Development of an intelligent agent for analysis of nonfunctional characteristics in specifications of software requirements. Eastern-European Journal of Enterprise Technologies 1 2 (2019) 6-17. doi: 10.15587/1729-4061.2019.154074.

[10] T. Hovorushchenko, O. Pavlova, Evaluating the software requirements specifications using ontology-based intelligent agent, in: Proceedings of 2018 IEEE International Scientific and Technical Conference "Computer Science and Information Technologies", CSIT'2018, Lviv, 2018, vol.1, pp.215-218. doi: 10.1109/STC-CSIT.2018.8526730.

[11] T. Hovorushchenko, O. Pomorova, Evaluation of mutual influences of software quality characteristics based ISO 25010:2011, in: Proceedings of 2016 International Scientific and Technical Conference "Computer Science and Information Technologies", CSIT'2016, Lviv, 2016, pp. 80-83. doi: 10.1109/STC-CSIT.2016.7589874.

[12] A. Boyarchuk, O. Pavlova, M. Bodnar, I. Lopatto, Approach to the analysis of software requirements specification on its structure correctness. CEUR-WS 2623 (2020) 85-95.

[13] A. Melnyk, B. Dunets, FFT processor IP Cores synthesis on the base of configurable pipeline architecture, in: Proceedings of the 7th International Conference on Experience of Designing and Application of CAD Systems in Microelectronics, CADSM'2003, Slavske, 2003, pp. 211–213. doi: 10.1109/CADSM.2003.1255034.
[14] T. Hovorushchenko, A. Herts, Ye. Hnatchuk, Concept of intelligent decision support system in the legal regulation of the surrogate motherhood. CEUR-WS 2488 (2019) 57-68.
[15] O. Drozd, M. Kuznietsov, O. Martynyuk, M. Drozd, A method of the hidden faults elimination in FPGA projects for the critical applications, in: Proceedings of 2018 IEEE 9th International Conference on Dependable Systems, Services and Technologies, DESSERT'2018, Kyiv, 2018, pp. 231 – 234. doi: 10.1109/DESSERT.2018.8409131.
[16] A. Drozd, J. Drozd, S. Antoshchuk, V. Nikul, M. Al-dhabi, Objects and methods of on-line testing: main requirements and perspectives of development, in: Proceedings of 2016 IEEE East-West Design & Test Symposium, EWDTS'2016, Yerevan, 2016, pp. 72 – 76. doi: 10.1109/EWDTS.2016.7807750.
[17] M. Drozd, A. Drozd, Safety-related instrumentation and control systems and a problem of the hidden faults, in: Proceedings of the 10th International Conference on Digital Technologies, Zhilina, 2014, pp. 137–140. doi: 10.1109/DT.2014.6868692.
[18] A. Drozd, M. Drozd, V. Antonyuk, Features of hidden fault detection in pipeline components of safety-related system. CEUR-WS 1356 (2015) 476–485.
[19] ISO/IEC 25010:2011. Systems and software engineering. Systems and software Quality Requirements and Evaluation (SQuaRE). System and software quality models. [Introduced 01.03.2011]. Geneva (Switzerland), 2011. (International standard).
[20] ISO 25023:2016. Systems and software engineering. Systems and software Quality Requirements and Evaluation (SQuaRE). Measurement of system and software product quality. [Introduced 31.03.2016]. Geneva (Switzerland), 2016. (International standard).
[21] Software Requirements Engineering Tools, 2018. URL: http://ecomputernotes.com/software-engineering/softwarerequirementsengineeringtools.
[22] 30 Best Requirements Management Tools in 2019, 2019, URL: https://www.guru99.com/requirement-management-tools.html.
[23] K. Verma, A. Kass, Requirements analysis tool: a tool for automatically analyzing software requirements documents. Lecture Notes in Computer Science 5318 (2008) 751-763. doi: 10.1007/978-3-540-88564-1_48.
[24] H. Mahmood, M. Rehman, Tools for software engineers. IMPACT: International Journal of Research in Engineering & Technology 3 10 (2015) 75-86.
[25] V. Jones, J. Murray, Evaluation of current requirements analysis tools capabilities for IV&V in the requirements analysis phase, 2017, URL: https://www.slideserve.com/shlomo/evaluation-of-current-requirements-analysis-tools-capabilities-for-ivv-in-the-requirements-analysis-phase.
[26] D. Raffo, R. Ferguson, Evaluating the Impact of The QuARS Requirements Analysis Tool Using Simulation, 2018, URL: https://pdfs.semanticscholar.org/7e7d/4e6f5ab13d00ca57c87711e30cd080730f34.pdf.
[27] F. Konig, L. Ballejos, M. Ale, A semi-automatic verification tool for software requirements specification documents, in: Proceedings of Simposio Argentino de Ingeniería de Software. Argentina, 2017.
[28] O. Pomorova, T. Hovorushchenko, Research of artificial neural network's component of software quality evaluation and prediction method, in: Proceedings of the 2011 IEEE 6-th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS'2011, Prague, 2011, vol. 2, pp. 959-962. doi: 10.1109/IDAACS.2011.6072916
[29] O. Pomorova, T. Hovorushchenko, Artificial neural network for software quality evaluation based on the metric analysis, in: Proceedings of IEEE East-West Design & Test Symposium, EWDTS'2013, Kharkiv, 2013, pp.200-203. doi: 10.1109/EWDTS.2013.6673193
[30] I. Nezhdanov, Information assessment criteria (importance, accuracy, significance), 2010. URL: http://www.police-russia.ru/showthread.php?t=44683.
[31] Information bases of computing: Information, 2015. URL: https://intuit.ru/studies/courses/3657/899/info.