Agent-Oriented Information Technology for Assessing the Initial Stages of the Software Life Cycle

Tetiana Hovorushchenko [1[0000-0002-7942-1857]], Artem Boyarchuk [2[0000-0001-7349-1371]], Olga Pavlova [3[0000-0003-2905-0215]] and Kira Bobrovnikova [4[0000-0002-1046-893X]]

[1, 3, 4] Khmelnytskyi National University, Instytutska str., 11, Khmelnytskyi, Ukraine
[2] National Aerospace University "Kharkiv Aviation Institute"
[1] tat_yana@ukr.net
[2] a.boyarchuk@csn.khai.edu
[3] olya1607pavlova@gmail.com
[4] bobrovnikova.kira@gmail.com

Abstract. The paper presents the development of agent-oriented information technology (AOIT) for assessing the sufficiency of information at the initial stages of the software life cycle. This AOIT performs automatic assessments and provides improving the level of sufficiency information of requirements for determination of each non-functional characteristic separately and all non-functional characteristics together, with the result that the gap in knowledge about non-functional characteristics for software is reduced. In addition, the developed AOIT minimizes the impact of the human factor and simplifies the performance of this assessment both by the developer and the customer. The developed agent-oriented information technology also provide: automation of the tedious, time-consuming, fatiguing and error-prone task of parsing the SRS; instantly show where re-work of requirements is needed; speed training for new systems engineers and project managers; the authoring of high-quality requirements; the correction and elimination of the requirements errors where they originate – during the early stages of the project; the tool for choosing the more qualitative software requirements specification; free online access, at any time, without any registration.

Keywords: Software Requirements, Software Requirements Specification (SRS), Sufficiency of Requirements Information, Non-Functional Software Characteristics, Ontology-Based Intelligent Agent (OBIA), Agent-Oriented Information Technology (AOIT).

1 Introduction

Humanity is now increasingly relying on software when it comes to solving complex problems and the number of software projects with a high cost is growing rapidly. Today in the world more than 250 billion USD is spent annually on the development of approximately 175 thousand software projects. The average cost of a software project for the large company is 2.322 million USD, for the average company – 1.313 million USD, and for the small company – 434 thousand USD [1, 2]. At the same

time, a significant number of software projects are unsuccessful (with overtime, over costs, lack of functional or canceled to completion and never used). On average, only 16-29% of software projects are executed within the scheduled time and budget (for large companies - 9-16% of projects); software projects of the largest US companies have only about 42% of the required capabilities and functions [1, 2] – Fig.1 [3].

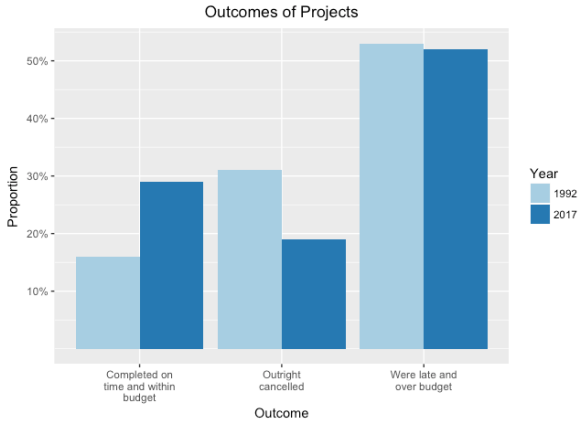

Fig. 1. Comparative statistics on the success of software projects in 1992 and 2017, according to The Standish Group International [3].

A significant quantity of bugs is introduced into the software at the early stages of the life cycle due to information losses due to the incomplete and different understanding of the needs and information context (10-23% of all bugs) [4, 5]. The authors of [6] give an even bigger percentage – 56% of all defects of software projects are introduced during the requirements definition stage. About 50% of requirements defects are the result of poorly written, unclear, ambiguous or incorrect requirements; *the other 50% is due to the incompleteness of specification* (incomplete, insufficient and omitted requirements) [6]. So, deficient requirements are the single biggest cause of software project failure.

The study on the relative cost of fixing engineering errors during the various phases of a project development cycle provided one finding common conclusion to all the software studies they examined and all the systems development projects they studied: the cost to fix software defects rose exponentially with each successive stage of the project life cycle [6]. This is one more compelling argument in favor of finding and correcting requirements errors where they occur – at the very beginning of the software project.

Consequently, the issues of the analysis and evaluation of the initial stages of the life cycle have a critical impact on software projects and on the success of their implementation. Then today, when the number of high-budget software projects is rapidly growing, the analysis of the software requirements specifications (SRS) and the possibility of the automated evaluation of the level of elaboration of the initial stages of the software life cycle are *actual problems* – in particular, identifying and eliminating the disadvantages of the initial stages of the software life cycle and the facts of the insufficiency of information of requirements to software (moreover special attention needs information about the non-functional characteristics of software).

2 Related Works

Today the following approaches for assessing the sufficiency of information of requirements to the software are known – Table 1. The review of the known information technologies, tools and intelligent agents for analyzing the software requirements is given in Table 2.

Table 1. The review of approaches to assessing the sufficiency of information of requirements to software

The approach	*Limitations of the approach*
Model for validation of sufficiency of safety requirements, focusing on sufficiency of hazard identification, hazard analysis, and software safety requirements traceability [7, 8]: several different metrics have been introduced, in particular, Percent Software Safety Requirements – as the ratio of the number of software safety requirements to the total number of software requirements; the authors suggest comparing this metric with a similar metric for implemented software, on the basis of this comparing the conclusion about the sufficiency of software safety requirements is made	On the basis of the proposed metric, only the sufficiency of the number of safety requirements can be assessed, but not the sufficiency of their information; the impossibility of interpretation by comparing the proposed metric for fundamentally new software; there is no tool for automatically identifying and calculating the safety requirements in the SRS
Evaluation of testing sufficiency as the achievement of the test coverage levels recommended or mandated by safety standards and industry guidelines [9]	The approach is aimed only at verification of software and requirements, but not at the validation of the developed software and customer needs; the approach uses the SRS solely as an input for the developed tool, but doesn't check the requirements of the specification for their sufficiency
Theoretical and applied principles of evaluating the sufficiency of the information on quality in the SRS [10]	Automation of such evaluation is not realized. This approach will be evolved during further solving the task of automated analysis of SRS

Table 2. The review of information technologies, tools and intelligent agents for software requirements analysis

Information technology (IT) or tool or intelligent agent (IA)	*Limitations of the IT or tool or IA*
1	*2*
CORE: enables the user to extract the requirements from the source documentation and then analyzes them for completeness, consistency and testability [11]	Checks the completeness of the SRS with respect to business requirements, but doesn't provide to check business requirements for their completeness

1	*2*
Visure – checker of requirements quality using natural language processing and semantic analysis [12]	Doesn't provide validation of compliance with the requirements of the SRS to the needs of the customer
Accompa: provides automatic requirements gathering; automatically detects & track dependencies between requirements [12]	Commercial tool – costs at just $199/month with no installation and maintenance [13]; doesn't check if all needs and requirements of the user have been reflected in the SRS
Innoslate: analyzes requirements using natural language processing technology [12]	Commercial tool – Innoslate Cloud is priced at $49/user/month and Innoslate Enterprise – $199/user/month [13]; doesn't provide quantitative assessments of the properties of SRS information
ReqView: organizes requirements into a tree hierarchy, uses rich text format for requirements description [12]	Doesn't verify and validate the requirements of the specifications for the needs of the customers
Modern Requirements4TFS: runs traceability analyses to ensure quality and find gaps or dependencies in requirements [12]	Doesn't identify the needs of the user that were not reflected in the requirements; doesn't provide visualization of the found gaps in the requirements
Natural language processing (NLP) Requirements Analysis Tools (for example, QVscribe): automate and significantly speed the searching the possible errors in natural language documents with requirements; provide visual scoring of each assessed requirement [6]	Doesn't reveal information losses in the formation of requirements
Requirements Analysis Tool: uses of user-defined glossaries to extract structured content; Semantic Web technologies are leveraged for deeper semantic analysis of the extracted structured content to find various kinds of problems in requirements documents [14]	Limitation of the glossary on which the tool is based
QARCC (Quality Attribute Risk and Conflict Consultant): is the tool for supporting conflict identification and requirements negotiation; it is a knowledge-based tool used to identify and analyze the conflicts in early development cycle [15]	Doesn't check the sufficiency of SRS requirements (in particular, non-functional requirements)
Requirements Assistant: identifies the missing requirements and inconsistency in requirements; detect the lack of some type of non-functional requirement such as reliability, security, safety [16]	Commercial tool; doesn't provide quantitative assessments (metrics) about missing non-functional requirements

1	*2*
QuARS Requirements Analysis Tool: provides screening of the requirements on consistency, completeness; identifies 37% of requirements defects [15, 17]	Commercial tool
DESIRe: ensures that the rules of completeness, non-ambiguity and comprehensibility are respected [18]	Commercial tool
RQV Tool (Requirement Quality Verification Tool) [18]: a semi-automatic verification tool for SRS based on a comprehensive quality model; can help in the verification of the quality of the SRS on the basis of ISO 29148:2011	Provides assessments of the quality of SRS information, but not its completeness or sufficiency
The ontology-based intelligent agent (OBIA) for eliminating the ambiguity in gathered software requirements and for facilitating the communication with the stakeholders [19]	Doesn't check whether all user needs have been reflected in such requirements, whether the requirements of the SRS are sufficient
The OBIA-based tool that uses instantiated ontological designs to generate programming code on the basis of the SRS [20]	Doesn't verify and validate the STS requirements for the needs of the customers, doesn't assess the sufficiency of the available information in the SRS requirements, doesn't identify the needs that were not reflected in the requirements (in particular, non-functional)
Information technology for assurance of veracity of quality information in the SRS: evaluation of the sufficiency of the volume of the quality information in the SRS [21]	The required SRS, which is structured according to ISO 29148:2011 [22]. This IT will be evolved during the further automated analysis of the SRS

The review of known information technologies, tools and intelligent agents for the SRS analysis has shown, that there are a number of effective solutions, but they all belong to different methodological approaches and are designed to different tasks. But, there is currently no information technology for assessing the sufficiency of information at the initial stages of the software life cycle. The actuality of the task of automated evaluation of the level of elaboration of the initial stages of the software life cycle on the basis of the analysis of specifications (in particular, the automated assessment of the sufficiency of information in the SRS), and the lack of information technology for assessing the sufficiency of information at the initial stages of the software life cycle causes *the need of development of agent-oriented information technology for assessing the sufficiency of information at the initial stages of the software life cycle*. This IT will be based on natural language processing (NLP) and will significantly reduce the cost of fixing requirements errors by finding them earlier and faster, and to free experts of the concrete subject domain from tedious, time-consuming tasks that waste their expertise.

3 Agent-Oriented Information Technology (AOIT) for Assessing the Sufficiency of Information at the Initial Stages of the Software Life Cycle

Agent-oriented information technology (AOIT) for assessing the sufficiency of information at the initial stages of the software life cycle solves the task of assessing the sufficiency of requirements information for determining only non-functional characteristics of software, which are software quality characteristics. According to ISO 25010 [23], such non-functional characteristics are reliability, functional suitability, performance efficiency, compatibility, maintainability, portability, security, usability. Subcharacteristics of such non-functional characteristics are also determined by ISO 25010 [23]. Attributes, on the basis of which such non-functional characteristics and their subcharacteristics are calculated, are determined by ISO 25023 [24]. Then, the sufficiency of requirements information for determining above non-functional characteristics is determined by the presence in the SRS all the attributes, which are necessary for determining the non-functional characteristics (203 attributes, including 138 different attributes).

Taking into account the theoretical principles of the information technology for evaluating the sufficiency of information on quality in the SRS [10] and the movement of information flows in the process of assessing the sufficiency of requirements information for determining the non-functional characteristics, the AOIT for assessing the sufficiency of information at the initial stages of the software life cycle is developed (as a set of processes using tools and methods of accumulation, processing and transmission of primary information for obtaining information of new quality on the status of the object, subject or phenomena) – Fig. 2.

The purpose of the developed AOIT is the automation of a quantitative assessment of the level of sufficiency of requirements information for determining the non-functional characteristics, minimization of the impact of the human factor, facilitation of the implementation of this assessment both by the developer and the customer. AOIT provides the identification of the need for forming the query to add attributes, which are necessary for determining the non-functional characteristics, and, if necessary, forms and visualizes its contents. The developed AOIT automatically processes available knowledge (non-functional requirements of the SRS) and generates new knowledge (conclusions about the sufficiency of requirements information, about the level of information sufficiency, recommendations for improving the level of information sufficiency in the SRS). As can be seen from Fig. 2, the developed AOIT is based on the ontology-based intelligent agent (OBIA) for semantic parsing of the SRS [25] and OBIA for assessing the early stages of the software life cycle [26].

The OBIA for semantic parsing the SRS accepts the SRS as the input data and automatically parsing the SRS on the subject of finding the attributes, which are necessary for determining the non-functional characteristics of the software. The SRS template, which demonstrates all the necessary attributes for determining the non-functional characteristics, and their location in the SRS, is proposed to the user in the form of the base ontology of domain "Software Engineering" (part "Software Requirements Specification (attributes)"), developed in [21] on the basis of ISO 29148 [22].

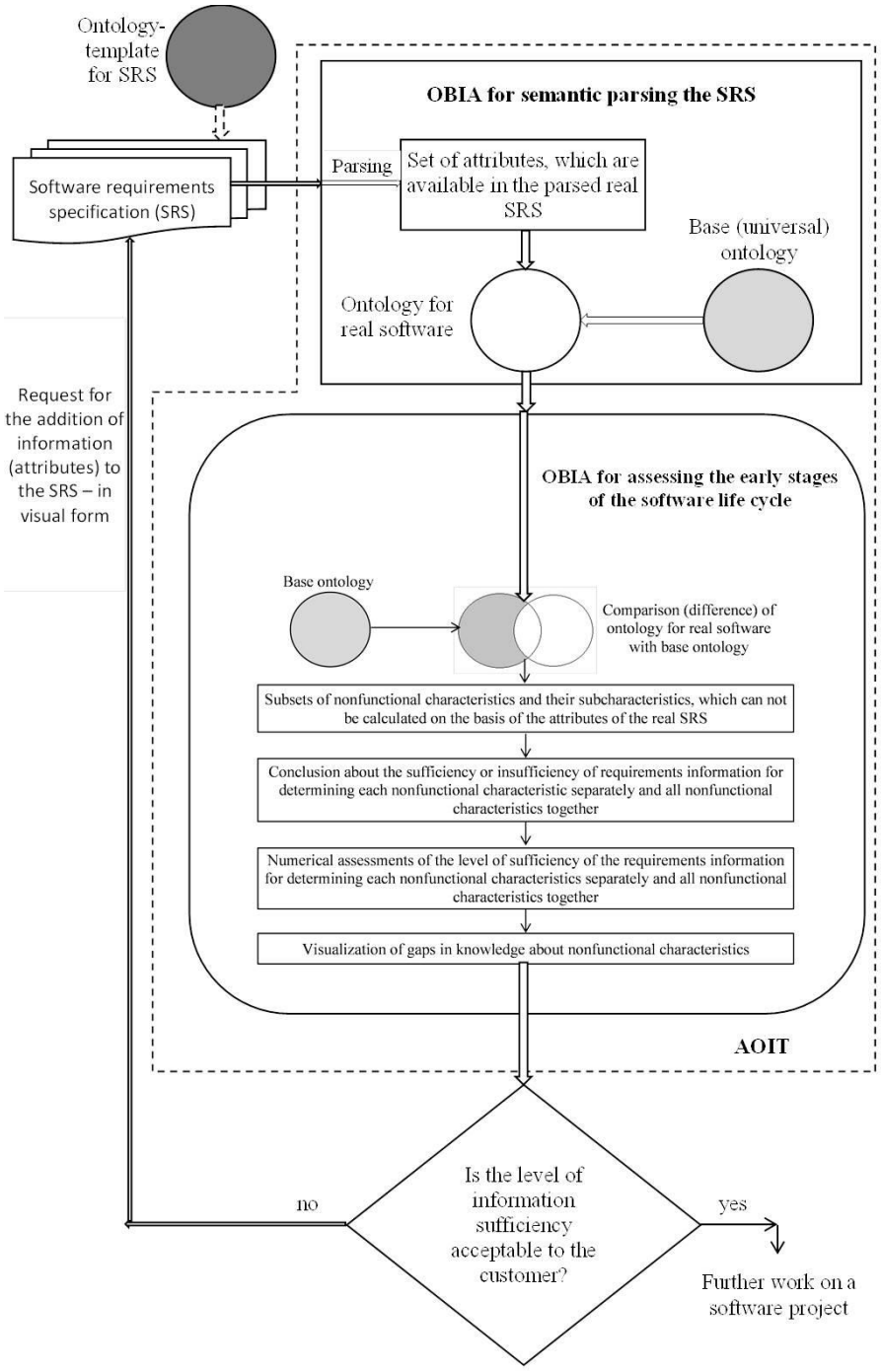

Fig. 2. AOIT for assessing the sufficiency of information at the initial stages of the software life cycle.

After the parsing of the SRS, this OBIA forms the set of attributes, which are available in the parsed real SRS. Using the developed in [21] base ontology for the non-functional characteristics (as the known fact) and the obtained set of the available attributes, this OBIA generates the ontology for real software, which transmits as input data to other OBIA (to OBIA for assessing the early stages of the software life cycle).

OBIA for assessing the early stages of the software life cycle carries the comparison of the base ontology for non-functional characteristics (known fact) with the obtained ontology for real software. As a result of this comparison, this OBIA, according to the developed in [26] method, forms the subsets of non-functional characteristics and their subcharacteristics, which can not be calculated on the basis of the attributes of the real SRS; concludes about the sufficiency or insufficiency of requirements information for determining non-functional characteristics; calculates numerical assessments of the level of sufficiency of the requirements information for determining each non-functional characteristics and all non-functional characteristics together; provides visualization of gaps in knowledge about non-functional characteristics.

If the level of information sufficiency is 100% (for example, for critical software) or is acceptable for the customer (but we recommend the level of sufficiency no less 85% with the purpose of minimization of information losses during the initial stages of the software life cycle), then the further work on the software project is performed; otherwise, the SRS developers are requested to add the required attributes to the SRS (with visualized hints, which attributes should be added), after which the SRS may be re-analyzed by the developed AOIT.

The developed AOIT makes it possible to compare different SRS for software projects with the same cost and duration; to guarantee the inclusion in the requirements of the information, which are necessary for the further determination of non-functional characteristics, thereby reducing the gap in knowledge about non-functional characteristics for software projects. The main advantage of the developed AOIT is the automation of the processes of parsing the SRS and of assessing the sufficiency of the requirements information for determining the non-functional characteristics, due to this eliminating the subjective influence of the person, and saving the information in the software company in the event of dismissal of a specialist.

Another important advantage of the developed AOIT is low cost of parsing (translating) the human-spoken (natural language) requirements, because it provides the analysis of the requirements on the subject of identification of the availability or absence (miss) of attributes, which are necessary for determining the non-functional characteristics. Since for determining the sufficiency, it is only necessary to know whether there is the attribute in the requirements or it is missing in the requirements, so the rules for parsing the SRS, on the basis of which the developed agent works, are so simple (if <attribute> is found in the SRS, then <attribute> is the element of the set of available attributes, else <attribute> is the element of the set of missing attributes). The simplicity of these rules provides a high speed and low cost of parsing the natural language requirements.

4 The Results of Functioning the AOIT for Assessing the Sufficiency of Information at the Initial Stages of the Software Life Cycle

AOIT for assessing the sufficiency of information at the initial stages of the software life cycle is implemented in the PHP language in the form of free software and is available by the link – https://olp-project.herokuapp.com.

Before uploading the SRS for its processing, the user of AOIT can acquaint with the template of the SRS, which demonstrates all the necessary attributes for determining the non-functional characteristics, and their location in the specification. This template of SRS is presented in the form of ontology. The SRS can have any structure, any form, but for further evaluating the non-functional characteristics it should have the values of the attributes from ISO 25023:2016. Exactly these attributes are in the visual ontology-template as the hints to the user. The fragment of such ontology-template for SRS is represented on Fig. 3.

For analysis, the user of developed AOIT must upload the SRS in pdf-format. After this, the AOIT parses the uploaded specification and on the basis of it generates the ontology for real software in owl-format, which the user can download for further work.

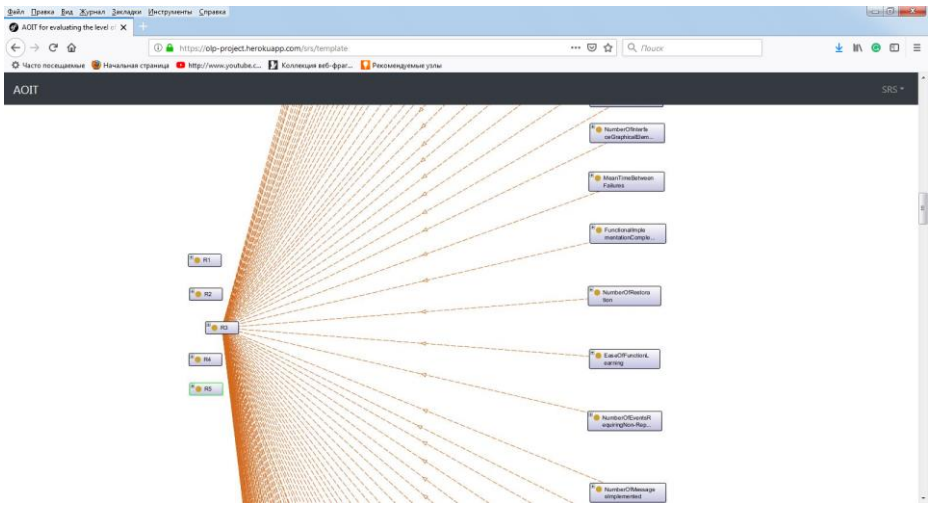

Fig. 3. Fragment of ontology-template for SRS of AOIT for assessing the sufficiency of information at the initial stages of the software life cycle.

After comparing the ontology for real software with base ontology for non-functional characteristics, the developed AOIT gives the conclusion on the sufficiency of requirements information, which consists of: the number and percentage of missing attributes (during this counting, AOIT gives two these numbers – without and with taking into account how many times the missing attribute is used when the determination of subcharacteristics of non-functional characteristics (without and with repetitions)); quantitative assessments of the sufficiency of requirements information for

determining each non-functional characteristic and all non-functional characteristics together. In addition, the developed AOIT proposes the list of missing attributes in the form of a list, which is divided by subcharacteristics of non-functional characteristics, which provides visualization of missing attributes for determining one or another subcharacteristics of non-functional characteristic.

For the experiment, three requirements specifications to software for accounting for the provision of Internet access services, which are developed for ITT Ltd (Khmelnitsky, Ukraine) by various software companies of Khmelnitsky. These SRS have approximately the same cost and duration, so the choice of the SRS according to these criteria is difficult.

As a result of the analysis of SRS No. 1, the developed AOIT has provided the following conclusions – Fig. 4, Fig. 5. As a result of the analysis of SRS No. 2, the developed AOIT has provided the conclusions, which are presented in Fig. 6. After analysis of the SRS No. 3, the developed AOIT has provided the following conclusions – Fig. 7.

The analysis of the results of the developed AOIT confirmed the regularity, which is identified in [21], that more important and priority are attributes, which impact on more than one subcharacteristic of non-functional characteristics. So, in SRS No. 1 there are no 21 attributes without considering the number of uses of each attribute when determining the subcharacteristics of non-functional characteristics (but there are no 78 attributes, considering the number of uses of each attribute). In SRS No. 2 there are no 37 attributes without considering the number of uses of each attribute (but there are no 39 attributes, considering the number of uses of each attribute). In SRS No. 3, there are no 37 attributes without considering the number of uses of each attribute (but there are no 74 attributes, considering the number of uses of each attribute). At the same time, the level of sufficiency of requirements information of SRS No. 1 for determining all non-functional characteristics is 58,23%, the level of sufficiency of requirements information of SRS No. 2 – 81,26%, the level of sufficiency of requirements information of SRS No. 3 – 60,85%.

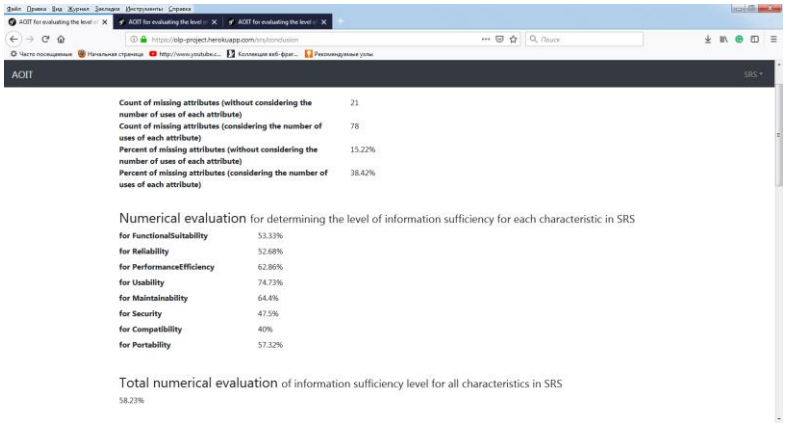

Fig. 4. Quantitative assessments of the sufficiency of requirements information (SRS No. 1) for determining the non-functional characteristics (which are provided by the developed AOIT).

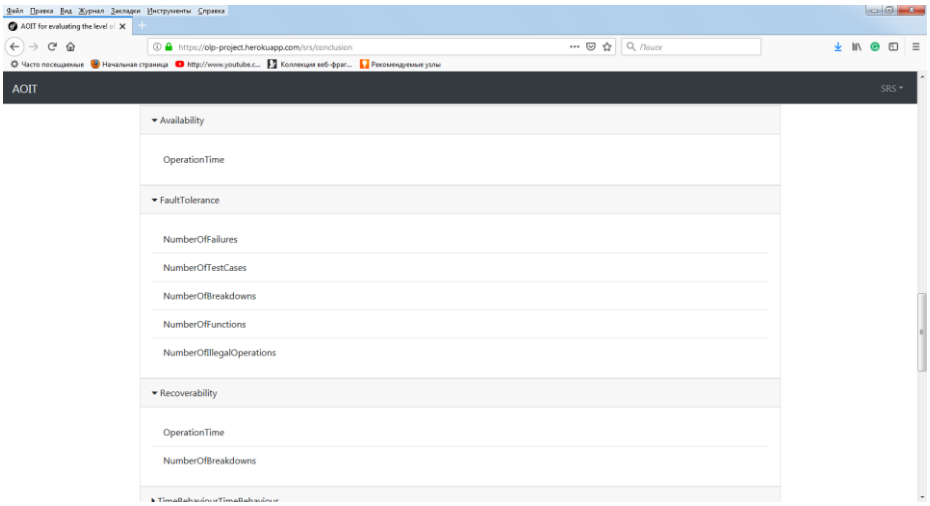

Fig. 5. Visualization of missing attributes (in SRS No. 1) for determining the three subcharacteristics of the non-functional characteristic "Reliability" (which is provided by the developed AOIT).

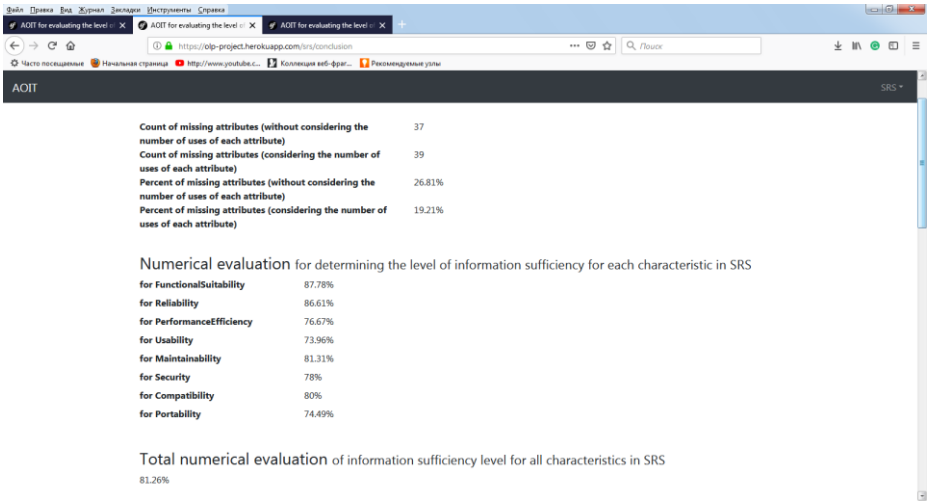

Fig. 6. Quantitative assessments of the sufficiency of requirements information (SRS No. 2) for determining the non-functional characteristics (which are provided by the developed AOIT).

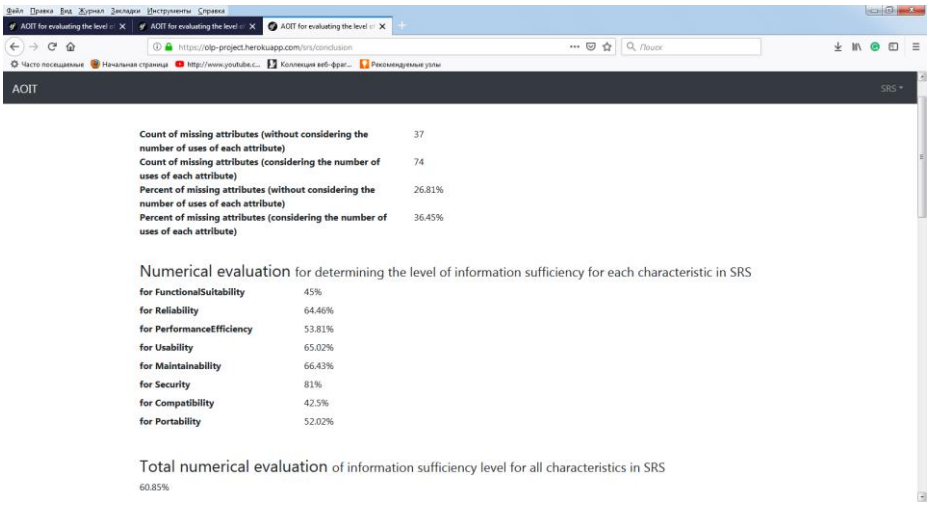

Fig. 7. Quantitative assessments of the sufficiency of requirements information (SRS No. 3) for determining the non-functional characteristics (which are provided by the developed AOIT).

Thus, with less number of missing attributes (without considering the number of uses of each attribute), SRS No. 1 has a lower level of information sufficiency than SRS No. 2, in which more attributes are absent (without considering the number of uses of each attribute). This situation is explained by the fact that in SRS No. 1 there is no greater number of attributes (in comparison with the SRS No. 2), which impact on more than one subcharacteristic of non-functional characteristics. This fact is proved by the number of missing attributes, which are provided by the AOIT (considering the number of uses of each attribute).

In the result of the analysis of the conclusions of the developed AOIT, the customer of the software for accounting for the provision of Internet access services (ITT Ltd) decided that the level of sufficiency of the requirements information in all three SRS isn't acceptable for the transition to further work on the software project. Therefore, all three SRS were sent back to developers for revision (in part of supplementing the attributes for determining the non-functional characteristics). The revised SRS were also analyzed by the developed AOIT. The conclusions of the AOIT after the analysis of the revised SRS are presented in Fig. 8-10.

In the SRS No. 1, 5 attributes were added without considering the number of uses of each attribute in determining the subcharacteristics of non-functional characteristics (47 attributes, considering the number of uses of each attribute). In SRS No. 2, 10 attributes were added without considering the number of uses of each attribute (and 10 attributes, considering the number of uses of each attribute). In SRS No. 3, 10 attributes were also added without considering the number of uses of each attribute (25 attributes, considering the number of uses of each attribute). At the same time, the level of sufficiency of requirements information of the SRS No. 1 for determining all non-functional characteristics is already 86,02%, the level of sufficiency of requirements information of the SRS No. 2 – 85,97%, the level of

sufficiency of requirements information of the SRS No. 3 – 73,7%. Consequently, the conclusions of the developed AOIT for assessing the sufficiency of information at the initial stages of the software life cycle provide increasing the level of sufficiency of requirements information for determining the non-functional characteristics, respectively, by 27,79%, 4,71% and 12,85% for SRS No. 1-3.

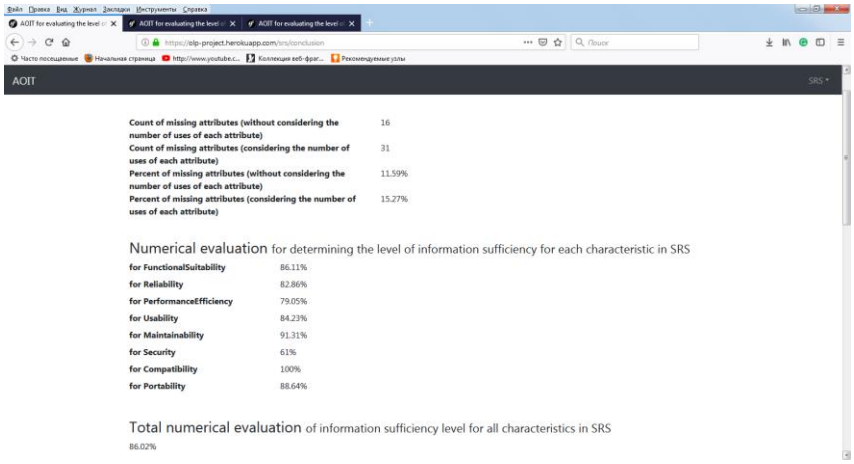

Fig. 8. Quantitative assessments of the sufficiency of requirements information (SRS No. 1, after revision), which are provided by the developed AOIT.

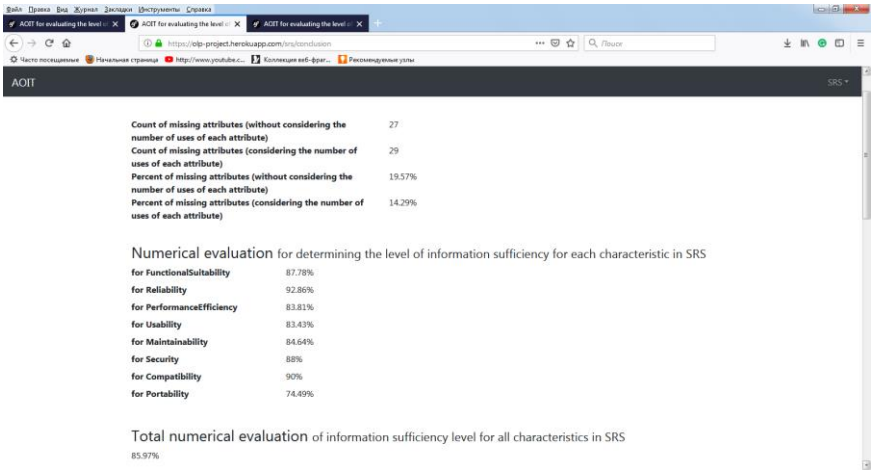

Fig. 9. Quantitative assessments of the sufficiency of requirements information (SRS No. 2, after revision), which are provided by the developed AOIT.

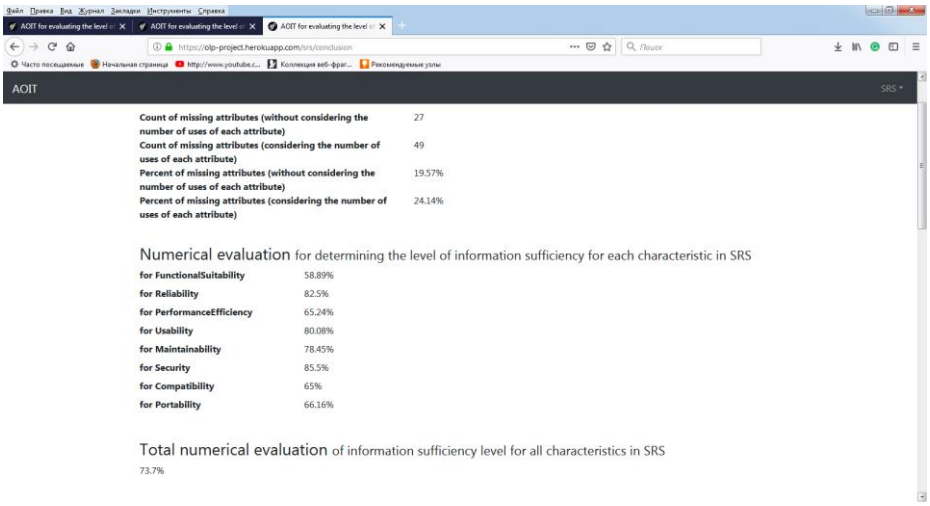

Fig. 10. Quantitative assessments of the sufficiency of requirements information (SRS No. 3, after revision), which are provided by the developed AOIT.

We (the developers of AOIT) recommend the level of sufficiency equal 85% is acceptable, but the final decision regarding the required level of sufficiency is taken by the customer of the software. The customer can set a lower threshold of this level, for example, 90%, 95% or 100% (even for non-critical software). So, in the result of the analysis of the conclusions of the developed AOIT, ITT Ltd decided that the level of sufficiency of the requirements information of SRS No.1 and SRS No. 2 is already accepted for the further work on the software project. ITT Ltd selected SRS No. 1 for further work. So, the customer was able to make a choice the SRS from the view of its information's sufficiency, but not only from the view of its cost and duration.

5 Conclusions

At present, the task of automated assessment of the level of elaboration of the initial stages of the software life cycle based on the analysis of specifications is actual (in particular, the automated assessment of the sufficiency of requirements information).

In this paper, the agent-oriented information technology for assessing the sufficiency of information at the initial stages of the software life cycle was developed. This AOIT assesses and provides the increase (for example, from 58,23% till 86,02% for SRS No. 1, from 81,26% till 85,97% for SRS No. 1, from 60,85% till 73,7% for SRS No. 3) of the level of sufficiency of requirements information for determining software non-functional characteristics – the gain of the level of suffiiciency is from 4,71% to 27,79%.

In addition to the above, *the advantages of the developed AOIT* also are: 1) automation of the time-consuming, routine and error-prone task of parsing the SRS, and almost instantly accomplishment of it; 2) indication where re-work on SRS is needed (the user can browse missing attributes and see SRS areas which the extra attention is needed, and which requirements need re-work); 3) provision of training for new SRS developers,

systems engineers and project managers (using this AOIT helps them see mistakes they might be making, and helps them recognize those mistakes in others' work); 4) help of developing the high-quality requirements; 5) help of correct and eliminate of requirements errors where they originate – during the early stages of the software project life cycle – before they become more expensive to correct; 6) provision of the tool for choosing the more qualitative software requirements specification; 7) free online access, at any time, without any registration.

The *economic effect* of the use of the developed AOIT is the ability to save software projects' budget for processing and correcting (during the life cycle) defects and bugs, which are made at the early stages of the life cycle - due to the demonstration of weaknesses in the SRS, that need to be finalized or re-worked, at a time when they arise.

The limitations of the developed AOIT are: 1) the assessment of the only sufficiency of requirements information for determining the non-functional characteristics as the sufficiency of the attributes in the SRS; 2) consideration of only non-functional characteristics, which are regimented by ISO 25010 standard as the software quality characteritics; 3) non-consideration the other SQUARE standards of 25000-series (ISO 25011, ISO 25012); 4) during parsing the SRS the search of only attributes, which are defined by ISO 25023as necessary for the non-functional characteristics-components of software quality. For elimination of these limitations, further efforts of the authors will be directed.

References

1. Hastie, S., Wojewoda, S.: Standish Group 2015 Chaos Report – Q&A with Jennifer Lynch, http://www.infoq.com/articles/standish-chaos-2015, last accessed 2019/02/12.
2. The Standish Group Report CHAOS, https://www.pro jectsmart.co.uk/white-papers/chaos-report.pdf, last accessed 2019/02/12.
3. A Look at 25 Years of Software Projects. What Can We Learn?, https://speedandfunction.com/look-25-years-software-projects-can-learn/, last accessed 2019/02/12.
4. McConnell, S.: Code complete. Microsoft Press, Redmond (2013).
5. Levenson, N. G.: Engineering a safer world: systems thinking applied to safety. MIT Press, Massachusetts (2012).
6. Leveraging Natural Language Processing in Requirements Analysis: How to eliminate over half of all design errors before they occur, http://qracorp.com/wp-content/uploads/2017/03/Leveraging-NLP-in-Requirements-Analysis.pdf, last accessed 2019/02/12.
7. Cruickshank, K. J.: A validation metrics framework for safety-critical software-intensive systems. Naval Postraduate School, Monterey (2009).
8. Michael, J. B., Shing, M. T., Cruickshank, K. J., Redmond, P. J.: Hazard analysis and validation metrics framework for system of systems software safety. IEEE Systems Journal. 4 (2), 186-197 (2010).
9. Baker, R., Habli, I.: An empirical evaluation of mutation testing for improving the test quality of safety-critical software. IEEE Transactions on Software Engineering. 39 (6), 787-805 (2013).
10. Hovorushchenko, T., Pomorova, O.: Information technology of evaluating the sufficiency of information on quality in the software requirements specifications. CEUR-WS. 2104, 555-570 (2018).

11. Software Requirements Engineering Tools, http://ecomputernotes.com/software-engineering/softwarerequirementsengineeringtools, last accessed 2019/02/12.
12. 30 Best Requirements Management Tools in 2019, https://www.guru99.com/requirement-management-tools.html, last accessed 2019/02/12.
13. Top 20+ Best Requirements Management Tools (The Complete List), https://www.softwaretestinghelp.com/requirements-management-tools/, last accessed 2019/02/12.
14. Verma, K., Kass, A.: Requirements Analysis Tool: A Tool for Automatically Analyzing Software Requirements Documents. Lecture Notes in Computer Science. 5318, 751-763 (2008).
15. Mahmood, H., Rehman, M. S.: Tools for software engineers. International Journal of Research in Engineering & Technology. 3 (10), 75-86 (2015).
16. Jones., V., Murray, J.: Evaluation of current requirements analysis tools capabilities for IV&V in the requirements analysis phase, https://www.slideserve.com/shlomo/evaluation-of-current-requirements-analysis-tools-capabilities-for-ivv-in-the-requirements-analysis-phase, last accessed 2019/02/12.
17. Raffo, D. M., Ferguson, R.: Evaluating the Impact of The QuARS Requirements Analysis Tool Using Simulation, https://pdfs.semanticscholar.org/7e7d/4e6f5ab13d00ca57c87711e30cd080730f34.pdf, last accessed 2019/02/12.
18. Konig, F., Ballejos, L., Ale, M.: A semi-automatic verification tool for software requirements specification documents. In: Simposio Argentino de Ingeniería de Software Proceedings. Sociedad Argentina de Informática e Investigación Operativa (2017).
19. Ossowska, K., Szewc, L., Weichbroth, P., Garnik, I., Sikorski, M.: Exploring an ontological approach for user requirements elicitation in the design of online virtual agents. Information Systems: Development, Research, Applications, Education. 264, 40-55 (2017).
20. Freitas, A., Bordini, R. H., Vieira, R.: Model-driven engineering of multi-agent systems based on ontologies. Applied Ontology. 12 (2), 157-188 (2017).
21. Hovorushchenko, T.: Information Technology for Assurance of Veracity of Quality Information in the Software Requirements Specification. Advances in Intelligent Systems and Computing. 689, 166–185 (2018).
22. ISO/IEC/IEEE 29148:2011. Systems and software engineering. Life cycle processes. Requirements engineering (2011).
23. ISO/IEC 25010:2011. Systems and software engineering. Systems and software Quality Requirements and Evaluation (SQuaRE). System and software quality models (2011).
24. ISO 25023:2016. Systems and software engineering. Systems and software Quality Requirements and Evaluation (SQuaRE). Measurement of system and software product quality (2016).
25. Hovorushchenko, T., Pavlova, O., Bodnar, M.: Modern Problems of Semantic Analysis of Software Requirements Specifications. Scientific notes of Taurida National University after named V. I. Vernadsky: series "Technical Sciences". 30 (269) (2019). [in Ukrainian]
26. Hovorushchenko, T., Pavlova, O.: Method of Activity of Ontology-Based Intelligent Agent for Evaluating the Initial Stages of the Software Lifecycle. Advances in Intelligent Systems and Computing. 836, 169-178 (2019).