Ontological Approach to the Assessment of Information Sufficiency for Software Quality Determination

Tetiana Hovorushchenko, Oksana Pomorova

Khmelnitsky National University, 11, Institutskaya St.,
29016, Khmelnitsky, Ukraine

tat_yana@ukr.net, o.pomorova@gmail.com

Abstract. The aim of this study is the development of approach to the assessment of information sufficiency for software quality determination (according to ISO 25010: 2011). The proposed approach to assessment of information sufficiency is based on the comparative analysis of fragments of ontology of the subject domain "Software Engineering" and ontologies, that are developed on the basis of software requirements and system specification of the developed software. The approach provides the improvement the specifications for the presence of measures, that are necessary to the determination of software quality sub characteristics and characteristics. The work of developed approach is illustrated by the assessment of information sufficiency for software quality determination of automated system for large-format photo print.

Keywords. Software, software project, software requirements specification (SRS), software quality, ontology, ISO 25010:2011

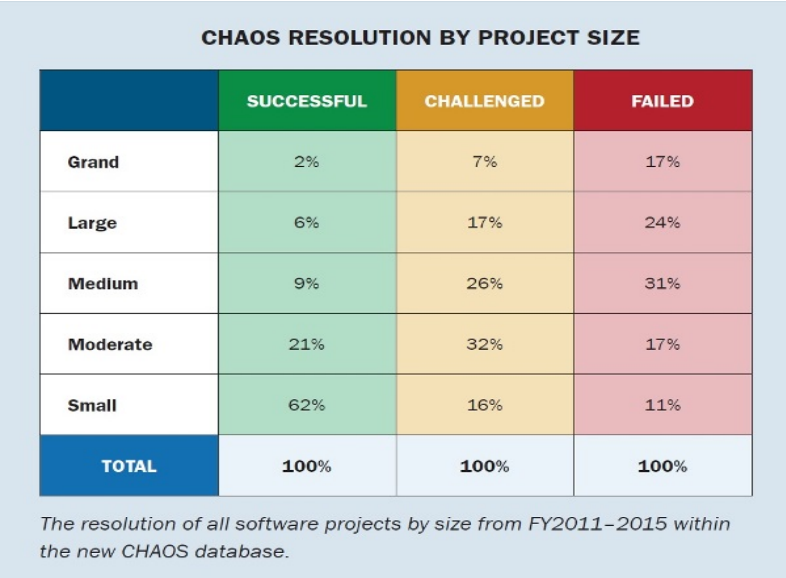
Key Terms. Model-Based Software System Development, Software Component, Software System, Method

1 Introduction

The software quality is basic factor for its successful implementation and exploitation. According to the standards of ISO 25010 [1], ISO 25030 [2] the software quality is the ability of the software to meet the stated and predicted needs when using under certain conditions. This definition differs from the definition of software quality of ISO 9000 [3] mainly because this definition of software quality provides for needs satisfaction, while the definition of ISO 9000 [3] provides for requirements satisfaction.The development of modern software system is user-oriented [4], namely the software quality is the important characteristic in terms of the stakeholders (especially customers). Obviously, even attracting the best experts for the development of technologies and standards of the software systems quality assurance doesn't guarantee sufficient software quality.

ICTERI 2016, Kyiv, Ukraine, June 21-24, 2016

The essential and integral feature of modern software systems is their complexity, so the attempts to describe the software objects with abstraction from their complexity lead to the abstraction from their essence. The constant growth of the software functions complexity inevitably leads to the increasing their volume and laboriousness (effort applied) [5].One of the most important causes of poor quality of large software projects are the increasing the number of components (subsystems) and the interfaces between them, and uncontrolled complexity of software systems, in opinion of the researchers The Standish Group International [6].Research [6] shows that statistics of success of small, moderate, medium, large and grand software projects is significantly different - Fig. 1.



CHAOS RESOLUTION BY PROJECT SIZE

	SUCCESSFUL	CHALLENGED	FAILED
Grand	2%	7%	17%
Large	6%	17%	24%
Medium	9%	26%	31%
Moderate	21%	32%	17%
Small	62%	16%	11%
TOTAL	100%	100%	100%

The resolution of all software projects by size from FY2011–2015 within the new CHAOS database.

Fig. 1. Statistics of success of small, moderate, medium, large and grand software projects in 2011-2015

Analysis of Fig. 1 provides the conclusion that 62% of small projects are successful while both only 6% of large projects and only 2% of grand projects are successful, i.e. small projects are ten times more successful than large and thirty times more successful than grand projects. This conclusion is confirmed by the statistics of software projects, that is presented in [5], on the basis of the function points as the main modern units of software size - Fig. 2.

During the software project, we often can not estimate the share of the informational indeterminacy of the project. Identification of the information, that appears in the process of interaction of "subsystems-interfaces-data-external influences", is especially difficult. Identification of future properties of developed software, that will display this information, is even more difficult task.

The cause of appearance of informational indeterminancy of the project is the low level of knowledge documentation, especially at the system level (Fig. 3 [7]).

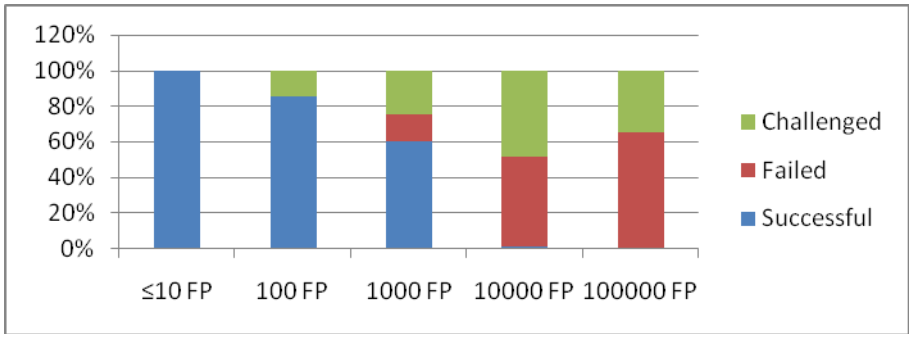


Fig. 2. Statistics of success of software projects with size 10, 100, 1000, 10000, 100000 functional points in 2010-2013

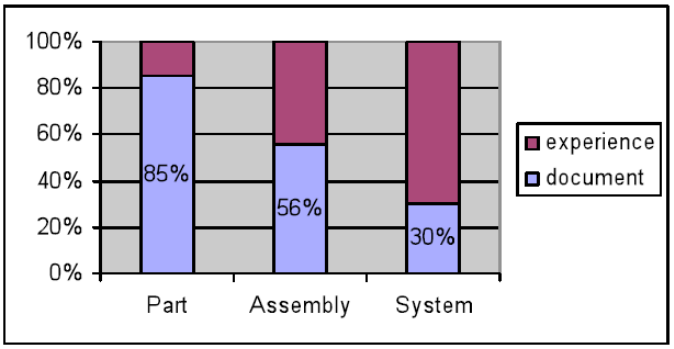


Fig. 3. How well system level knowledge is documented

Fig. 4 depicts the situation, characterized by premature design decisions and their documentation, prior to understanding the design. Fig. 4 shows an area, referred to as the “knowledge gap,” that is the result of the low level of knowledge documentation and the root cause of many engineering failures [8].The size of knowledge gap is not constant for software project – during the lifecycle it can increase and decrease, since new information appears and it should be taken into account.The presented on Fig. 4 viewpoint on the knowledge gap does not quite correspond to reality. We assume that partial consideration of the subject domain information in the software quality models and impact of this information on only finished product lead to increase of the knowledge gap size during the life cycle (new boundaries of knowledge gap are delineated dotted line on Fig. 5), that can be the cause of the software accidents and disasters [9]. For safe software functioning the knowledge gap size is desirable to reduce. This can be done by the consideration of as much subject domain information in the software quality models and standards.Reducing the knowledge gap size will provide the improvement of the software quality.

Given the above, all the available knowledge and information about the software system can be represented as the diagram, which has the sector that reflects the volume of unsufficient (unknown) information (knowledge gap) - Fig. 6. This sector consists of unconsiderated subject domain information.The size of this sector is not

determined, because it is unclear what information and how much information is unknown. Sector of the unknown information should be narrow by fully consideration of subject domain information.The smaller size of sector of the unknown information indicates to the higher quality and safer work of software system, i.e. the main task is the reducing the share of unknown information about software system.

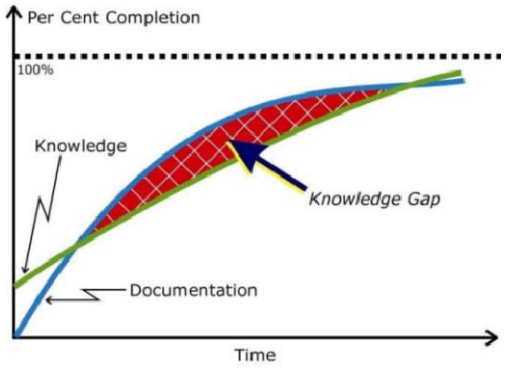


Fig. 4. Knowledge gap

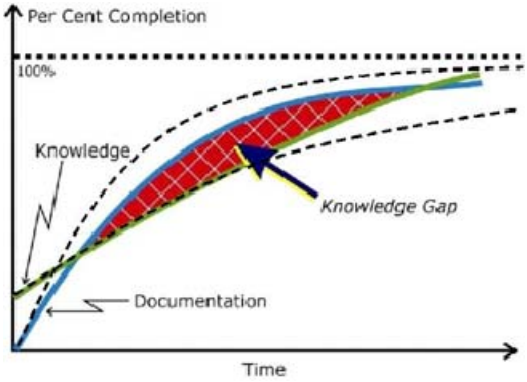


Fig. 5. Real size of Knowledge gap

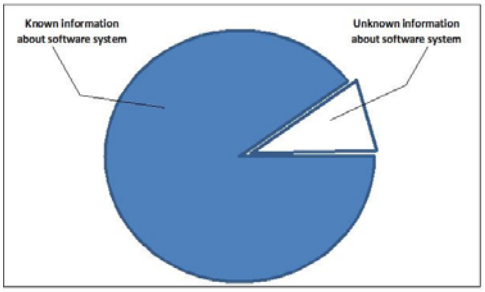


Fig. 6. Field of knowledge about software system with sector of the unknown information

Then the *actual task* is the assessment of information sufficiency as to software (for example, the possibility of obtaining of trustworthy information on the measures for calculation of the values of the software quality characteristics and subcharacteristics), on the basis of which software quality (by ISO 25010 [1]) is determinated.Incompleteness and inaccuracy of such information lead to fall of veracity of software quality assessments.So the *purpose of this study* is the development the approach to the assessment of information sufficiency for software quality determination.

2 Ontological Approach to the Assessment of Information Sufficiency for Software Quality Determination

The most used model for software quality assessment is the model ISO 25010 [1].The idea of this standard is that each of the characteristics is something we can analyse directly at the software product.Model ISO 25010 proposes to assess the software quality as a function of the eight characteristics, each of which is a function of several subcharacteristics (total 31 subcharacterics).But subcharacteristcs, in turn, are the functions of several measures.Analysis of [10-13], ISO 9126-2 [14], ISO 9126-3 [15] and revised on their basis ISO 25023 [16] provide the determination the dependence of quality subcharacteristics from the measures (total 203 measures).The basic idea is that the assessment of quality, its characteristics and subcharacteristics should be comprehensively performed, considering all these characteristics, subcharacteristics and measures accordingly.

Some of the measures are part of several quality subcharacteristics.So, if such measures are inaccurate or missing, then simultaneously use of these subcharacteristics in determining the several quality characteristics will significantly affect to the veracity of the software quality estimates.In such situation the condition of the mitigation of influence of these subcharacteristics cross-correlation is the important when using them in the quality models.Such mitigation is performed by identifying the joint measures, improving the accuracy of their values, or, if possible, limitating the simultaneous using of subcharacteristics that containing the same measures.

The information on determining the software quality characteristics and subcharacteristicsis conveniently presented as semantic networks or other structures, which provide the displaying of the causal relationships between concepts. One of these structures is ontology.The advantages of ontology are the systematic approach to the study of the subject domain, the possibility of the holistic filing of known subject domain information, the identification of the overlaps and gaps in knowledge on the basis of the visualization of missing logical relationships.

Researhers *have already used the ontologies in software design*.E.Burov proposed methods and tools for deve-lopment of software systems based on the ontological models [17, 18]. I.Shostak & I.Butenko developed the ontological models and methods of forming the profile during the software certification [19]. L.Babenko proposed the ontological approach to specifying the features of software systems and their components [20].We use ontologies for assessment of information sufficiency for software quality determination.

For development and visualization of ontologies the large number of software tools, including universal, that provide the work with different subject domains, are today developed: Ontolingua Server, SMART, Protégé, OntoEdit, WebOnto, ODE (Ontological Design Environment), DOE (Differential Ontology Editor), CONE, OntoEditor +. The authors use a free software Protégé 4.2, which provides the work (creation, edition, vizualization and comparison) with ontologies of the various subject domains (http://protege.stanford.edu/).

First and foremost, the base ontology of the subject domain "Software Engineering" was developed. In it there is information about the software quality characteristics, subcharacteristics and measures.For development of this ontology the 8 software quality characteristics by ISO 25010 [1] (Functional Suitability, Reliability, Usability, Security, Performance Efficiency, Maintainability, Compatibility, Portability) were used.For determination of the Functional Suitability ISO 25010 proposed 3 subcharacteristics, which in turn are based on 15 measures.For determination of the Reliability ISO 25010 proposed 4 subcharacteristics, which are based on 30 measures.For determination of the Usability ISO 25010 proposed 6 subcharacteristics, which are based on 49 measures.For determination of the Security ISO 25010 proposed 5 subcharacteristics, which are based on 23 measures.For determination of the Performance Efficiency ISO 25010 proposed 3 subcharacteristics, which are based on 26 measures.For determination of the Maintainability ISO 25010 proposed 5 subcharacteristics, which are based on 33 measures.For determination of the Compatibility ISO 25010 proposed 2 subcharacteristics, which are based on 9 measures.For determination of the Portability ISO 25010 proposed 3 subcharacteristics, which are based on 18 measures.The idea of developed base ontology is shown on Fig. 7.

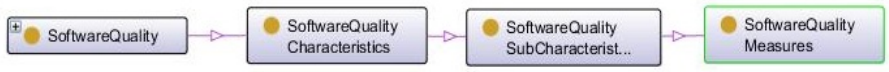


Fig. 7. Base ontology for subject domain “Software engineering” (part “Software quality”)

The components of the base ontology are: base ontology for Functional Suitability (Fig. 8), the base ontology for Reliability (Fig. 10), the base ontology for Usability (Fig. 12), the base ontology for Security (Fig. 14) , base ontology for Performance Efficiency (Fig. 16), the base ontology for Maintainability (Fig. 18), the base ontology for Compatibility (Fig. 20), the base ontology for Portability (Fig. 22).

The developed base ontology provides the following conclusions: 1) Functional Suitability: subcharatceristcs Functional Completeness, Functional Appropriateness have 4 joint measures; Functional Appropriateness, Functional Correctness have 2 joint measures; 2) Reliability: subcharacteristics Maturity, Availability, Recoverability have 1 joint measure; Maturity, Fault Tolerance have 2 joint measures; Fault Tolerance, Recoverability have 1 joint measure; 3) Usability: subcharacteristics Learnability, Operability have 2 joint measures; Appropriateness Recognisability has 1 joint measure with the Learnability, Operability; Operability, User Error Protection have 1 joint measure; Operability, User Interface Aesthethics have 1 joint measure;4) Security: subcharacteristics Confidentiality, Integrity have 8 joint measures;5) Performance Efficiency: subcharacteristics Time Behaviour, Resource Utilization

have 2 joint measures; Time Behaviour, Capacity have 1 joint measure; 6) Maintainability: subcharacteristics Modularity, Modifiability have 3 joint measures; Testability has 2 joint measures with Modularity, Modifability; Modularity, Analysability have 1 joint measure; Analysability, Modifability have 1 joint measure; 7) Compatibility:subcharacteristics Co-existence, Interoperability have 1 joint measure; 8) Portability: subcharacteristics Adaptability, Replaceability have 2 joint measures; Adaptability, Installability have 1 joint measure.

In addition, there are measures, which are included in the formulas of several subcharacteristics of different characteristics (for example, measure Operation Time is included in subcharacteristics of all 8 quality characteristics).One of the basic properties of the base ontology is precisely the possibility of manifestation of cross-correlation of characteristics and subcharacteristics when using them in quality models.Because the important condition is the mitigation of the cross-correlation of such subcharacteristics when using them in quality models, therefore during assessment of the software quality it is necessary to pay special attention to those measures, which are part of simultaneously several subcharacteristics.

Ontological approach to the assessment of information sufficiency for software quality determination (by ISO 25010:2011 [1]) consists of the next steps: 1) analysis of the software requirements specification for the concrete software project for the presence of measures, that necessary for determining the quality characteristics and subharacteristics of software project and software; 2) the development of ontology for determining the quality of the concrete software; 3) comparison of the developed ontology with base ontology for software quality determination, components of which are shown on Fig. 8, 10, 12, 14, 16, 18, 20, 22; 4) identification of measures, which are absent in the ontology for determination of the quality of the concrete software; 5) identification of quality characteristics and subcharacteristics, that cannot be calculated on the basis of the existing measures (at the same time should remember about the basic idea of ISO 25010 [1], which says that the quality assessment should be performed comprehensively, considering all quality characteristics;the assessment of quality characteristics also should be performed comprehensively, considering all subcharacteristics; the assessment of quality subcharacteristics, in turn, should be performed comprehensively, considering all measures); 6) the presence of subcharacteristics and characteristics, values of which cannot be determined on the basis of measures, that available in the software requirements specification, indicates the need to complement of this specification by the neccessary measures (at this stage adding the necessary information and deleting other relevant information are possible); 7) repeating the steps 2-6 until all quality characteristics and subcharacteristics will be possible to identify or until the conclusion will be formed, that data for software quality determination are insufficient.

3 Experiments: Assessment of Information Sufficiency for Determination of Quality of Software of Automated System for Large-Format Photo Print

During the study the specification of automated system for large-format photo print was analyzed.On the basis of the specification analysis the available measures were

determined, that necessary for determining the quality characteristics and subcharacteristics of software project and software.These measures provides the development of ontology for determination of the quality of this software, consisting of the: onto-logy for Functional Suitability (Fig. 9), ontology for Reliability (Fig. 11), ontology for Usability (Fig. 13), ontology for Security (Fig. 15), ontology for Performance Efficiency (Fig. 17), ontology for Maintainability (Fig. 19), ontology for Compatibility (Fig. 21), ontology for Portability (Fig. 23) for concrete software project.

The comparison of the developed ontology for software project of automated system with fragments of the base ontology for subject domain "Software engineering" provides to find that in the ontology for project the 4 measures (Number of Functions, Operation Time, Number of Data Items, Number of Test Cases) are missing.

In addition, on the basis of the comparison of the ontology for software project of automated system for large-format photo print with base ontology was found that in the concrete ontology the data for determination of some quality characteristics and sub-characteristics are insufficient due to the absence of the above 4 measures.

Analysis of Fig. 8 and Fig. 9 provides the conclusion that the data for determination of all 3 subcharacteristics of Functional Suitability are insufficient. Therefore, none of subcharacteristics cannot be calculated, so Functional Suitability of software project cannot be determined too, and therefore the quality of the software project cannot be determined.

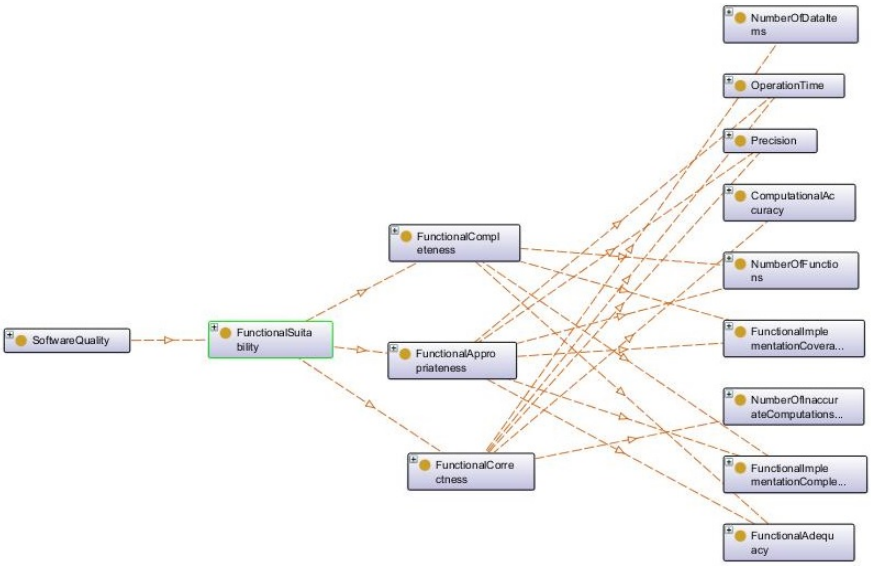


Fig. 8. Base ontology for Functional Suitability

Analysis of Fig. 10 and Fig. 11 provides the conclusion that the data for determination of all 4 subcharacteristics of Reliability are insufficient. Therefore, none of subcharacteristics cannot be calculated, so Reliability of software project cannot be determined, and therefore the quality of the software project cannot be determined.

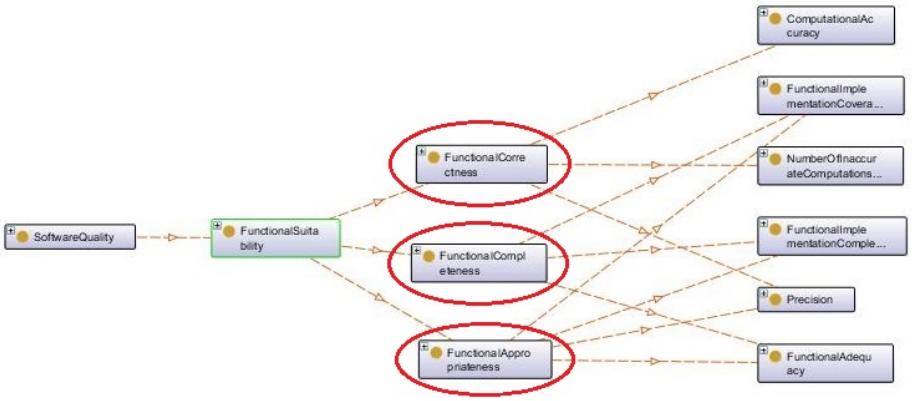


Fig. 9. Ontology for Functional Suitability for automated system for large-format photo print

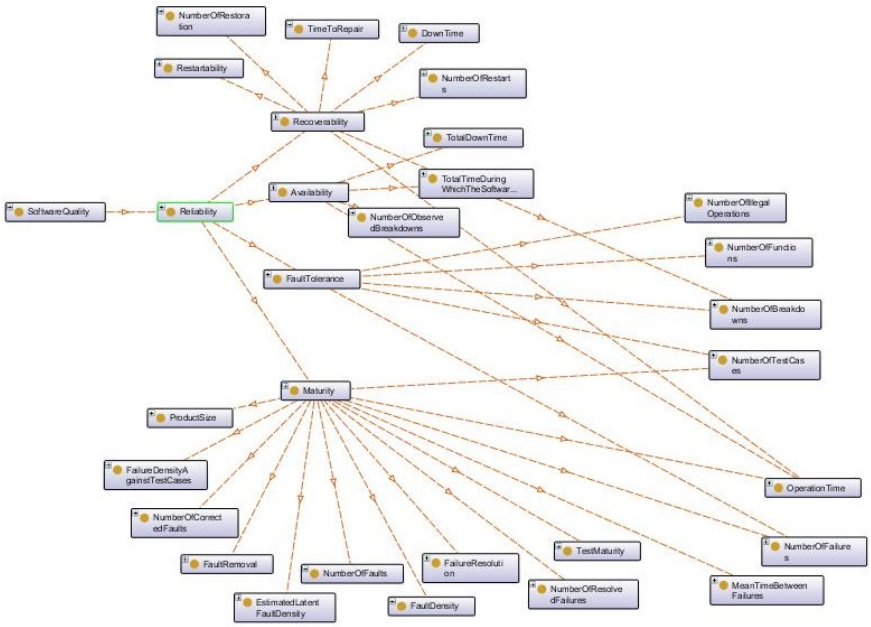


Fig. 10. Base ontology for Reliability

Analysis of Fig. 12 and Fig. 13 provides the conclusion that the data for determination of 3 from 6 subcharacteristics of Usability are insufficient. Therefore, 3 subcharacteristics cannot be calculated, so Usability of software project cannot be determined, and therefore the quality of the software project cannot be determined.

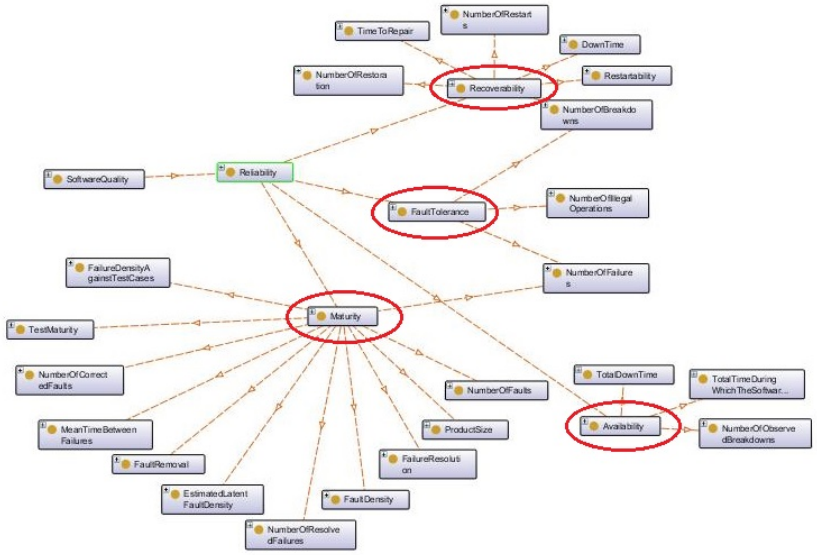


Fig. 11. Ontology for Reliability for automated system for large-format photo print

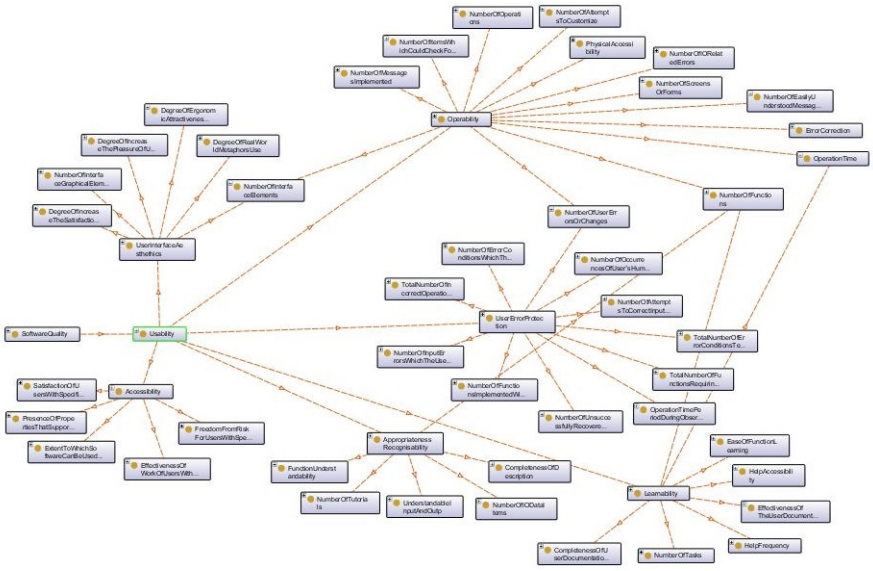


Fig. 12. Base ontology for Usability

Analysis of Fig. 14 and Fig. 15 provides the conclusion that the data for determination of 2 from 5 sub characteristics of Security are insufficient. Therefore, 2 sub characteristics cannot be calculated, so Security of software project cannot be determined, and therefore the quality of the software project cannot be determined.

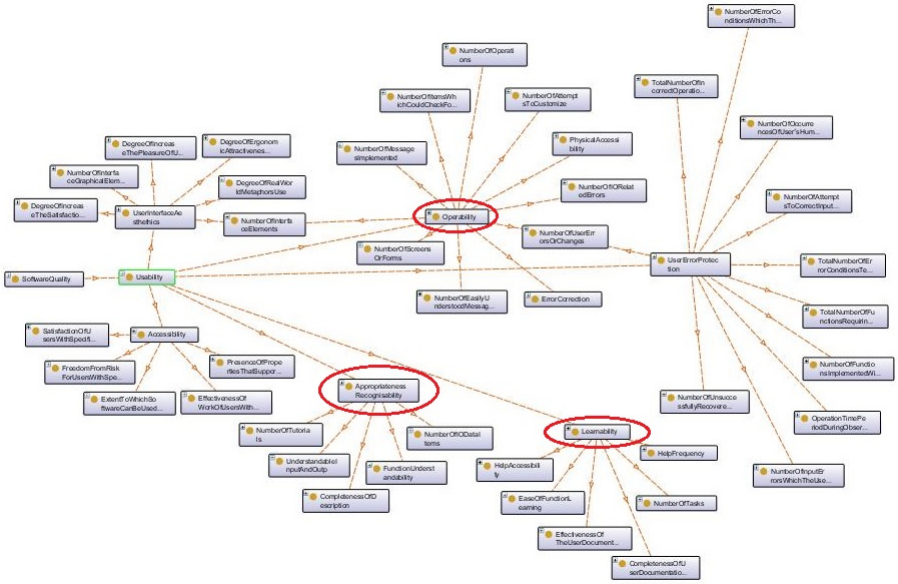


Fig. 13. Ontology for Usability for automated system for large-format photo print

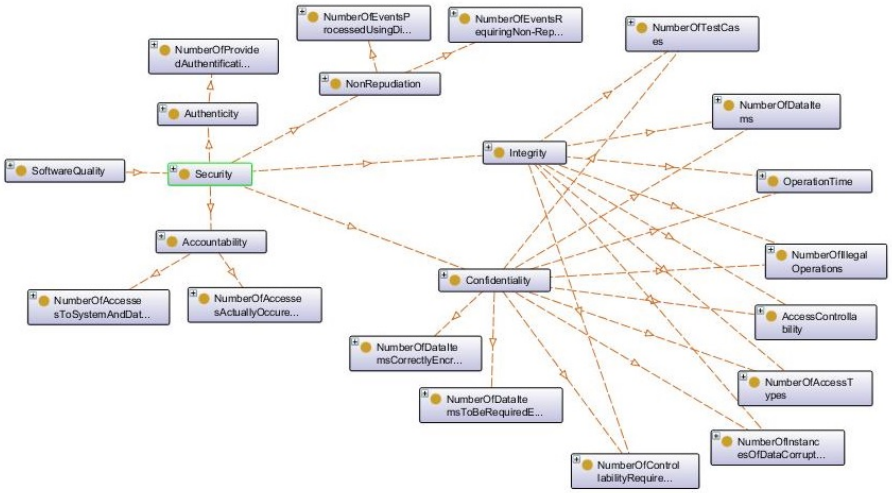


Fig. 14. Base ontology for Security

Analysis of Fig. 16 and Fig. 17 provides the conclusion that the data for determination of all 3 sub characteristics of Performance Efficiency are insufficient. Therefore, none of sub characteristics cannot be calculated, so Performance Efficiency of software project cannot be determined, and therefore the quality of the software project cannot be determined.

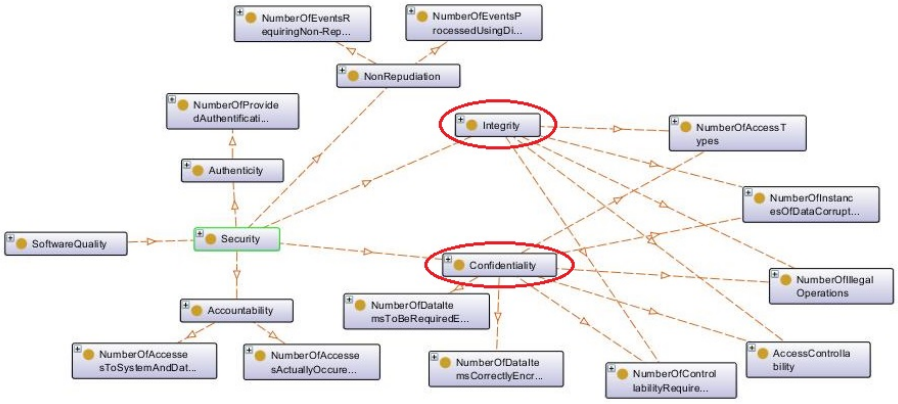


Fig. 15. Ontology for Security for automated system for large-format photo print

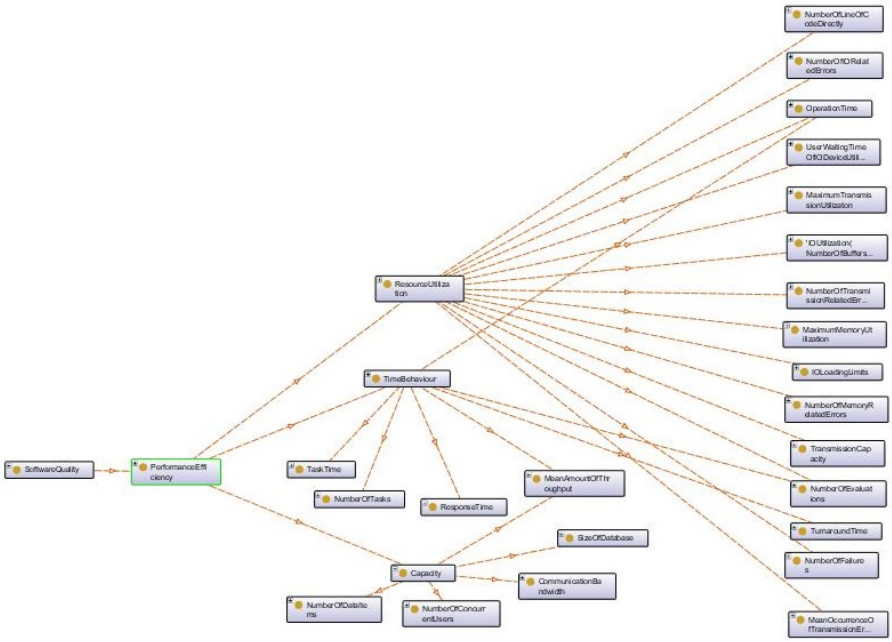


Fig. 16. Base ontology for Performance Efficiency

Analysis of Fig. 18 and Fig. 19 provides the conclusion that the data for determination of 4 from 5 subcharacteristics of Maintainability are insufficient. Therefore, 4 subcharacteristics cannot be calculated, so Maintainability of software project cannot be determined, and therefore the quality of the software project cannot be determined.

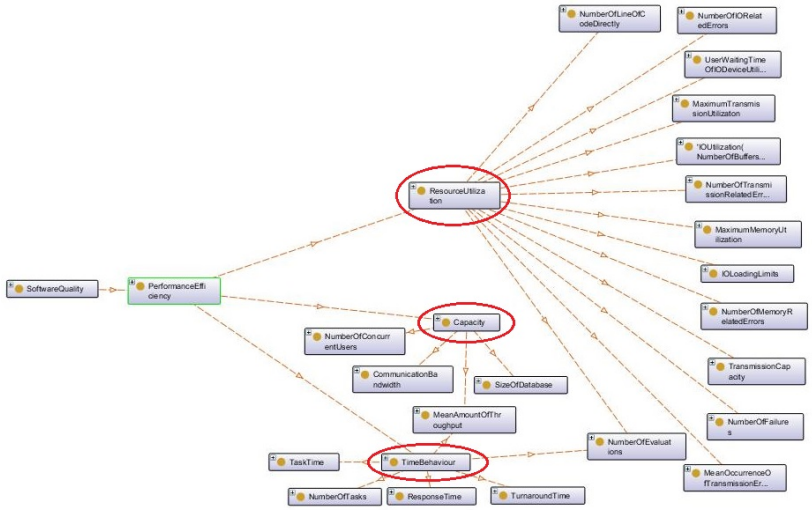


Fig. 17. Ontology for Performance Efficiency for automated system for large-format photo print

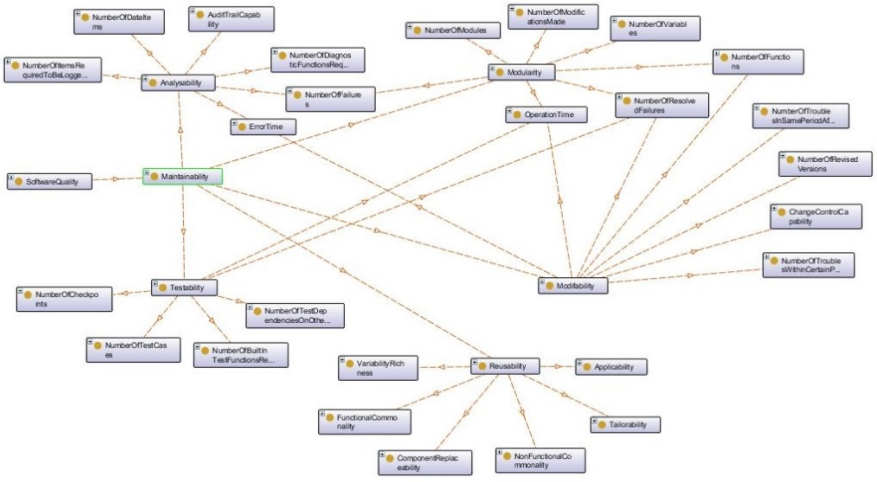


Fig. 18. Base ontology for Maintainability

Analysis of Fig. 20 and Fig. 21 provides the conclusion that the data for determination of all 2 subcharacteristics of Compatibility are insufficient. Therefore, none of subcharacteristics cannot be calculated, so Compatibility of software project cannot be determined, and therefore the quality of the software project cannot be determined.

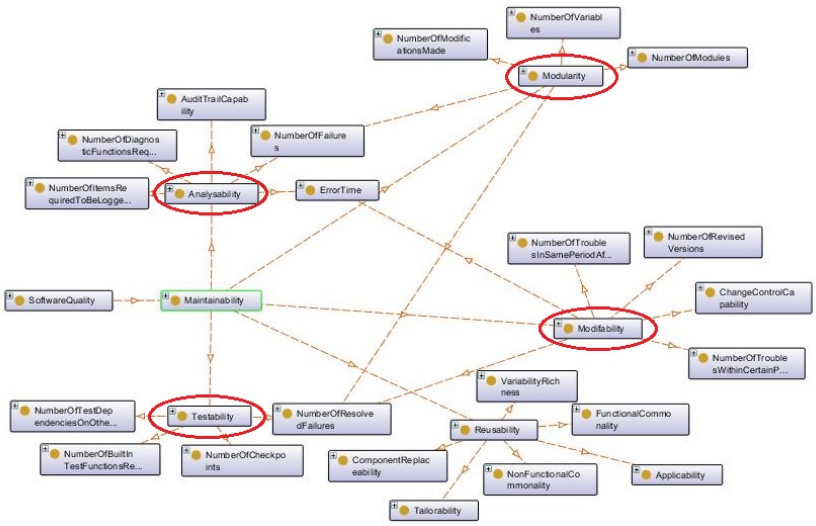


Fig. 19. Ontology for Maintainability for automated system for large-format photo print

Analysis of Fig. 22 and Fig. 23 provides the conclusion that the data for determination of 2 from 3 subcharacteristics of Portability are insufficient. Therefore, 2 subcharacteristics cannot be calculated, so Portability of software project cannot be determined, and therefore the quality of the software project cannot be determined.

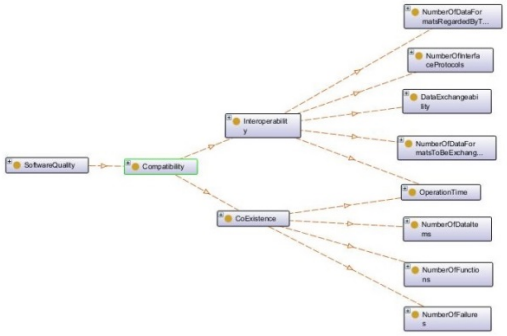


Fig. 20. Base ontology for Compatibility

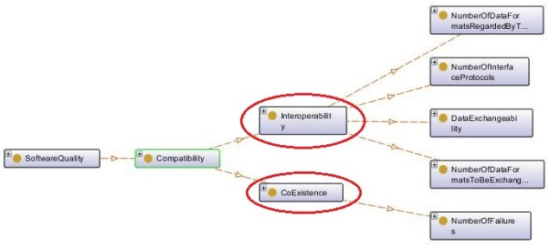


Fig. 21. Ontology for Compatibility for automated system for large-format photo print

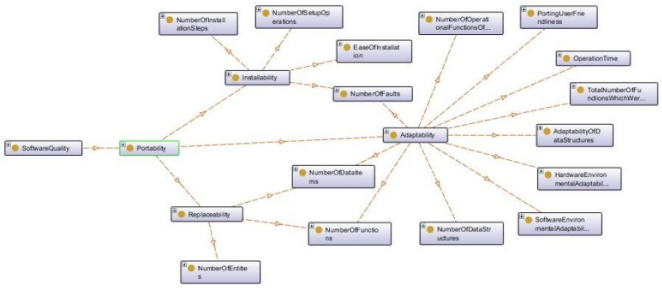


Fig. 22. Base ontology for Portability

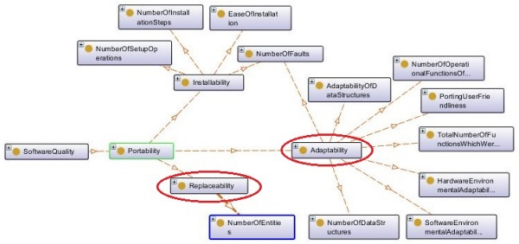


Fig. 23. Ontology for Portability for automated system for large-format photo print

Then the lack of 4 these measures in software requirements specification led to the impossibility of detarmination of all quality characteristics and the quality of the project and developed software.

For the concrete software project there are characteristics and sub characteristics, that are impossible to define or possible to insufficient define according to available information in the specification. Because the proposed approach to the assessment of information sufficiency for software quality determination is iterative, then the complement of the software requirements specification was conducted. The measures Number of Functions, Number of Data Items were added, and then the new version of the ontology for determination of the quality of the concrete software was created.

Comparative analysis of the new version of ontology with the base ontology showed that changes have occurred in the determination of Functional Completeness, Capacity, Appropriateness Recognisability, Analyzability, and Replaceability of the concrete software project. But the lack other 2 measures in specification (Operation Time, Number of Test Cases) leaves impossible the determination of all software quality characteristics and the quality of the project and developed software (still insufficient information).The process of complement the specification is iterative. But customer of developed lautomated system has decided that further complement of the specification is economically inexpedient therefore the conclusion about insufficient data for determination of the software quality was formed.

4 Conclusions

The measures analysis is an effective mean of assessing the software quality upon availability of veracity information for it conduct. One of the factors affecting the veracity of such information is sufficiency of the volumes of information about measures in the SRS. Therefore, solving the task of assessment of sufficiency information about measures in the SRS generally enhances the veracity of software quality assessment.

In the analysis of software quality subcharacteristics (as sources of information) the cross-correlation of these subcharacteristics because they have joint measures. Correlation of subcharacteristics, that displayed by base ontology, should be considered because it can reduce the accuracy and veracity of software quality assessment.

Knowledge of experienced professionals on interference and correlation of software quality subcharacteristics are valuable, so they should be stored and used in assessing the software specifications in terms of information sufficiency for software quality characteristics and subcharacteristics.

For displaying of these knowledge we selected ontologies that became the basis of the approach to the assessment of information sufficiency for software quality determination (according to ISO 25010: 2011).

The proposed ontological approach provides the development of recommendations for improvement of the software specification that illustrated by the example of the assessment of information sufficiency for determination of quality for software of automated system of large-format photo print.

References

1. ISO 25010:2011 Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models (2011)
2. ISO 25030:2007 Software engineering -- Software product Quality Requirements and Evaluation (SQuaRE) -- Quality requirements (2007)
3. ISO 9000:2015 Quality management systems -- Fundamentals and vocabulary (2015)
4. ISO 10002:2014 Quality management – Customer satisfaction – Guidelines for complaints handling in organizations (2014)
5. McConnell, S.: Code complete. Microsoft Press (2013)
6. The Standish Group International: CHAOS Report. Technical report, CHAOS Knowledge Center (2015)
7. Maier, R.: Knowledge Management Systems. Information and Communication Technologies for Knowledge Management (2013)
8. Patterson, Jr F.G.: Life cycles for system acquisition. Encyclopedia of Life Support Systems, Systems Engineering and Management for Sustainable Development, 82--110 (2004)
9. Pomorova, O., Hovorushchenko, T.: The Way to Detection of Software Emergent Properties. In: Proc. 2015 IEEE 8-th Int. Conf. on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications. Vol.2, 779--784 (2015)
10. Abran, A., Al-Quitash, R.E., Desharnais, J.-M., Habra, N.: ISO-Based Models to Measure Software Product Quality. Software Quality Measurement: Concepts and Approaches. Chapter 5, 61--96 (2014)

11. Montagud Gregori, S.; Abrahao Gonzales, SM.; Insfrán Pelozo: A systematic review of quality attributes and measures for software product lines. Software Quality Journal. 20(3-4), 425--486 (2012)
12. Sun Her, J., Hyeok Kim, J., Hun Oh, S., Yul Rhew, S., Dong Kim, S.: A framework for evaluating reusability of core asset in product line engineering. Information and Software Technology. 49, 740--760 (2007)
13. Biscoglio, I., Marchetti, E.: Definition of Software Quality Evaluation and Measurement Plans: A Reported Experience Inside the Audio-Visual Preservation Context.In Software Technologies: 9th International Joint Conference, ICSOFT 2014, Revised Selected Papers, CCIS 555, 63--80 (2015)
14. ISO 9126-2:2003Software engineering -- Product quality -- Part 2: External metrics (2003)
15. ISO 9126-3:2003 Software engineering -- Product quality -- Part 3: Internal metrics (2003)
16. ISO 25023:2015 Systems and software engineering -- Systems and software Quality Requirements and Evaluation (SQuaRE) -- Measurement of system and software product quality (2015)
17. Burov, E.: Complex ontology management using task models. International Journal of Knowledge-Based and Intelligent Engineering Systems, 18(2), 111--120 (2014)
18. Burov, E., Pasitchnyk, V., Gritsyk, V.: Modeling software testing processes with task ontologies. British Journal of Education and Science, 2(6), 256--263 (2014)
19. Shostak, I., Butenko, I.: Ontology approach to realization of information technology for normative profile forming at critical software certification. J. Military Institute of Kiev National University named after Taras Shevchenko, 38, 250–-253 (2012)
20. Babenko, L.: Ontological approach to specification of software systems features and components. Cybernetics and System Analysis, 1, 180--187 (2009) (in Russian)