UDC 004.89:004.9
DOI: 10.15587/1729-4061.2019.154074

DEVELOPMENT OF AN INTELLIGENT AGENT FOR ANALYSIS OF NONFUNCTIONAL CHARACTERISTICS IN SPECIFICATIONS OF SOFTWARE REQUIREMENTS

T. Hovorushchenko
Doctor of Technical Sciences,
Senior Researcher, Associate Professor,
Head of Department*
E-mail: tat_yana@ukr.net
O. Pavlova
Postgraduate Student*
E-mail: olya1607pavlova@gmail.com
M. Bodnar
Postgraduate Student*
E-mail: nikas.bodnar@gmail.com
*Department of Computer Engineering & System Programming
Khmelnytskyi National University
Instytutska str., 11,
Khmelnytskyi, Ukraine, 29016

Сьогодні актуальною задачею є автоматизований аналіз специфікацій вимог до програмного забезпечення (ПЗ) на предмет достатності інформації щодо нефункційних характеристик-складових якості ПЗ. Проведений аналіз відомих інтелектуальних агентів на основі онтологічного підходу показав, що відомі агенти не розв'язують задачу кількісного оцінювання достатності інформації у специфікації вимог до ПЗ для визначення нефункційних характеристик ПЗ.

Задачею даного дослідження є реалізація інтелектуального агента на основі онтологічного підходу для аналізу інформації щодо нефункційних характеристик у специфікаціях вимог до ПЗ.

Розроблено модель діяльності інтелектуального агента на основі онтологічного підходу для оцінювання специфікацій вимог до ПЗ. Вона відображає особливості оцінювання достатності інформації для визначення нефункційних характеристик-складових якості ПЗ. Розроблена модель є теоретичним підґрунтям для реалізації інтелектуального агента на основі онтологічного підходу для оцінювання специфікацій вимог до ПЗ.

Реалізовано інтелектуальний агент на основі онтологічного підходу для оцінювання інформації щодо нефункційних характеристик у специфікаціях вимог до ПЗ. Реалізований агент формує висновок про достатність або недостатність інформації щодо нефункційних характеристик-складових якості ПЗ у специфікації вимог до реального ПЗ. Крім цього, він кількісно оцінює рівень достатності інформації у специфікації вимог до реального ПЗ для визначення кожної нефункційної характеристики ПЗ та для визначення всіх нефункційних характеристик-складових якості ПЗ разом. Агентом надається список атрибутів, якими варто доповнити специфікацію вимог для підвищення рівня достатності її інформації, а також візуалізація прогалин у знаннях про всі нефункційні характеристики-складові якості ПЗ.

Результати функціонування реалізованого агента в комплексі забезпечують підвищення рівня достатності інформації у специфікації вимог до ПЗ. Розроблений інтелектуальний агент дозволяє частково усунути людину з процесів опрацювання інформації, уникнути втрат істотної інформації і мінімізувати виникнення помилок на ранніх етапах життєвого циклу ПЗ

Ключові слова: специфікація вимог до програмного забезпечення (ПЗ), нефункційні характеристики ПЗ, інтелектуальний агент на основі онтологічного підходу

1. Introduction

Today's mankind increasingly relies on software when solving complex problems while the number of high-cost software projects is growing rapidly. However, as statistics [1] shows, the share of problem software projects (with over expenditure of time or funds or with insufficient functional [1]) is about half of all software projects. The share of unrealizable software projects (which are canceled and never used [1]) is about 1/5 of all software projects. Consequently, only one third of all software projects are successful, reliable and worth spending.

Multitudes of errors creep into software at the initial stages of its lifecycle. Vast majority of software-related accidents occur because of errors in specifications of requirements [2]. Software projects with requirement specifications containing insufficiently accurate, incomplete and contradictory information cannot be successful in realization [3]. In addition, the earlier the defect (error, violation, drawback, malfunction) is revealed the cheaper its correction. There-

fore, in order to provide required functionality and quality of software, requirement specification study is obligatory. This study objective consists in identifying and eliminating drawbacks occurring at initial stages of software life cycle and the facts of insufficient information relevant to theses drawbacks. Although completeness of software requirements is desirable, determination of completeness of the set of requirements is not realistic as was proved in [4]. Consequently, the process of this study should include assessment of the extent to which the specification reflects information about the target software. Particular attention is required to the requirements that characterize nonfunctional characteristics of software [5].

Hence, the issue of the day consists in an automated analysis of the specifications of requirements to software information sufficiency concerning nonfunctional component characteristics of software quality. According to ISO 25010 [6], nonfunctional component characteristics of software quality include Reliability, Functional Suitability, Performance Efficiency, Compatibility, Maintainability, Portability, Security, Usability.

2. Literature review and problem statement

Currently, when the era of semantic web evolves, ontology is a key technology. The notion of ontology in the field of information technology was used for the first time by Tom Gruber [7]. Ontologies are used for representation of existing knowledge as well as acquisition, structuring of this knowledge and formation of a new knowledge in the subject field. Advantages of using ontologies include system approach to studying the subject field, possibility of holistic presentation of known information in the subject field, identification of knowledge duplications and gaps based on visualization of missing logical links. In addition, it is ontologies that provide the opportunity for automatic understanding and analysis of information, elimination of human participation and minimization of information losses in the process of its processing and gaining knowledge [7, 8].

The use of ontologies for software engineering makes it possible to avoid repeated conceptualization of the subject domain, reduce the use of resources at the early stages of software life cycle and the number of conceptualization errors [7–9].

Knowledge of the professionals already possessing such experience is of significant value in improving reliability of evaluation of nonfunctional component characteristics of software quality. For example, knowledge of interaction and correlation of nonfunctional attribute characteristics is valuable since some of the attributes according to which correlation takes place can be omitted at all. As a result, accuracy and reliability of the estimates obtained, etc. can be worsened. The effect of mutual correlation is mitigated by identifying common attributes, ensuring their availability in the specification of software requirements and improving accuracy of their values. Consequently, information about evaluation of nonfunctional component characteristics of software quality (e. g., interrelation of nonfunctional characteristics according to attributes) can conveniently be presented in a form of ontologies that make it possible to represent cause-and-effect relations between concepts.

Many studies are devoted to the idea of using ontologies in software engineering. For example, methods and tools for constructing software systems based on ontological models of problems are proposed in [8, 9]. The authors of [8, 9] suggest to use ontological models at all stages of software life cycle through conceptualization of the subject field in a context of the problems being solved. That is, necessary essentials are distinguished in the subject field, their attributes and constraints and dependences between them are determined and ontology of the subject field which can further be used by program writers is constructed. The authors of [8, 9] have proved that the use of such an ontology of the subject field simplifies the process of initial conceptualization when developing software prevents conceptualization errors at early stages of the software life cycle. However, the use of such an approach is impossible if software is developed for a subject field for which no ontology has yet been developed. Besides, software development should take into consideration requirements and standards not only in the subject field but also software development standards which in no way is taken into consideration in the approach described in [8, 9]. Also, methods and tools for developing software systems based on the ontological model of problems do not allow to automate analysis of software requirement specifications, in particular, sufficiency of their information although automated processing of knowledge itself provides minimization of information losses.

Approaches to tracing the software requirements based on weighted ontologies were developed in [10, 11]. The authors of [10] propose an ontological structure of traceability of MUPRET multipurpose requirements for processing heterogeneity of software requirements based on automatic generation of traceability ratios. Accuracy of traceability ratios generated by the MUPRET structure is verified by comparing it with the set of ratios manually identified by users. Weighted ontologies are used in [11] to process natural language in the transition from the specification of requirements written in a natural language to software development. As the authors of [11] prove, namely ontologies are capable of revealing non-compliance of requirements presented in a natural language. The approaches presented are aimed at elimination of heterogeneity or non-compliance of existing requirements in specifications but do not verify in any way whether all user's needs and requirements have been reflected in the specifications. Consequently, they are not suitable for assessing sufficiency of information about nonfunctional characteristics in the specifications of software requirements.

An ontological model for describing and defining the subject and operational knowledge on software quality was developed in [12]. The authors of this study have proposed an ontology that combines knowledge on the terminology and semantic relations of SWEBOK, IEEE and ISO standards aimed at quality assurance. The developed ontology simulates the process of the software life cycle and quality assurance. However, quality assurance takes place only at the end of the software life cycle for the finished program code. Today, software quality is interpreted as its ability to meet customer's needs when used under certain conditions [6]. Therefore, all necessary information about the customer's needs must already be laid down in the specification of requirements to software, that is information sufficiency can be already assessed based on the specification for further achievement of software quality. The approach proposed in [12] does not assure software quality at early stages of its life cycle, in particular, does not assess information sufficiency

concerning nonfunctional component characteristics of software quality in the specifications of software requirements.

Ontologies and weighted ontologies of the Software Engineering subject field (Software Quality, Software Quality: Metric Analysis, Specification of Software Requirements sections) were developed in [13]. They are readable and adaptable since they represent Standard ISO 25010:2011 widely used for various problems. These are ontological reusable models that are relevant for solving the problem of automated analysis of the specifications of software requirements as regards information sufficiency in order to improve quality of software developed according to the specifications. Therefore, these ontologies will be used for implementation and operation of the intelligent agent being developed.

Ontologies provide possibility of access, understanding and analysis of information by intelligent agents. In its operation, the intelligent agent uses information obtained from environment, analyzes it by comparison with the already known facts and makes a decision on further actions based on the results of this analysis [14]. The agent intelligence consists in formation of new knowledge, in particular, certain conclusions and recommendations.

A number of studies are devoted to development of intelligent agents based on the ontological approach for the software engineering field. For example, authors of [15] investigated the use of ontologies for agent-oriented software engineering and experimentally confirmed benefits of using ontological agents for software engineering. As a result of this study, an agent proposed in [15] uses existing ontological structures to write a program code based on specifications. Disadvantage of such an agent is that it does not verify and validate correspondence of the requirements available in the specifications to the needs of customers, does not assess information sufficiency in the specification of requirements, does not identify the needs that were not reflected in the requirements (in particular, nonfunctional ones). Application of the proposed agent may result in development of correct software which, however, will be lacking quality because of noncompliance with all customer needs.

Study [16] is dedicated to elimination of uncertainty in software requirements and improvement of communication between the parties concerned through introduction of intelligent agents based on the ontological approach. Authors of [17] propose development of an ontology-based intelligent agent to minimize existing semantic uncertainty when developing a specification of software requirements in a natural (Spanish) language. The proposed agent is also used for automatic generation of main specification elements and automatic plotting of objectives. The agents presented in [16] and [17] are designed for elimination of uncertainty in the existing requirement specifications but they do not check whether all user requirements were represented or whether these requirements are sufficient.

Ontological agent-oriented models are used in [18] to formalize initial requirements to software in order to reduce costs on the example of development of applications for Ambient Assisted Living system for patients with Parkinson's disease. The authors of [19] have proposed the basis of formal presentation of requirements of resource-limited contextual systems of critical application in a form of intelligent agents based on the ontological approach. That is, the authors of works [18, 19] provide just mechanisms based on the ontological approach to formalize software requirements. However, these approaches do not analyze requirements nor assess sufficiency of requirements in the specification and do not establish information losses in formation of requirements. To say more, software projects with requirement specifications containing insufficient or incomplete information cannot be successful in realization.

Authors of [20] offer a task-oriented architecture based on an agent-oriented paradigm and ontological design for decision support systems. The proposed architecture makes it possible to iteratively transfer functional requirements to architectural components. But this architecture in no way works with nonfunctional requirements: it does not check them for completeness and sufficiency nor does transfer them to the architectural components of the software being developed. Consequently, the approach proposed in [20] is also unsuitable for automated analysis of software requirement specifications for sufficiency of information on nonfunctional component characteristics of software quality.

Analysis of known intelligent agents based on the ontological approach has shown that well-known agents are aimed either at automated development of the program code according to the specification, formalization of the existing requirements or, as a maximum, eliminate uncertainty in requirements. They do not solve the problem of assessment of information sufficiency in requirements to software based on automated analysis of software requirement specifications.

In terms of assessing sufficiency of information in requirements to software, assessment of sufficiency of software security requirements is known for today [4, 21]. Authors of [4, 21] provide validation metrics that use hazard analysis as well as derivative requirements to software to mitigate identified hazards in order to assess sufficiency of software security requirements at the start of the software development process. In particular, studies [4, 21] propose a new model for validating security requirements by focusing on sufficiency of hazard identification, analysis of hazards and traceability of security requirements. Besides, authors of [4, 21] prove that any metrics that measure depth of the causal-factor analysis of software should be adapted to the standards of sufficiency measurement. Several metrics have been introduced in [4, 21], in particular, a security requirement indicator as a ratio of the number of security requirements to the total number of requirements. The authors suggest to make comparison of this indicator with a similar indicator for another, already implemented software on the basis of which a conclusion on security requirement sufficiency can be drawn. Obviously, only sufficiency of the number of security requirements can be estimated based on such a metric but not the sufficiency of their information since unscrupulous developers can artificially increase the number of requirements, for example, by their duplication. Another disadvantage of this approach is impossibility of interpretation by comparing the proposed indicator for fundamentally new software. In addition, this approach is not automated as the authors did not propose a tool for automatic identification and calculation of security requirements in the specification of software requirements.

Another approach to assessing sufficiency of information in software development consists in assessment of testing sufficiency as achievement of levels of test coverage recommended or approved by safety standards and industry recommendations [22]. This study proposes an empirical assessment of sufficiency of testing for onboard software systems taking into consideration software verification according to the requirements set forth. However, this approach is aimed

only at verification of software and requirements and not at validation of the developed software and customer needs. In addition, this approach uses specifications of requirements solely as an input for the developed tool but does not check for sufficiency of requirements in the specification.

One of solutions in assessing sufficiency of information in specifications of software requirements is presented in studies [5, 13] in which theoretical and applied principles of information technology for assessing sufficiency of quality information contained in the specifications of software requirements are proposed. The principles developed in [5, 13] are aimed at determining sufficiency of information on quality in the software requirements but automation of such evaluation is not realized in these works. Therefore, it is the approach proposed in [5, 13] that will be further developed by solution of the problem of automated analysis of the software requirements set forth in specifications as regards sufficiency of information on nonfunctional component characteristics of software quality.

3. The aim and objectives of the study

The study objective was to automate analysis of the software requirement specification for sufficiency of information and improve the level of sufficiency of information on the nonfunctional component characteristics of software quality in the specification of requirements. Achievement of this objective should make it possible to partially eliminate human participation in information processing and knowledge gain, avoid loss of essential information and minimize occurrence of errors at early stages of the lifecycle.

To achieve this objective, the following tasks were addressed:

– develop a model of activity and implement an intelligent agent based on the ontological approach for assessing information about nonfunctional characteristics in the specification of software requirements (the agent will be intelligent as it will provide new knowledge, in particular, conclusions about sufficiency of information and recommendations for its improvement);

– analyze information about nonfunctional characteristics in the specification of software requirements using an intelligent agent based on the ontological approach.

4. The model of activity of the intelligent agent for analysis of nonfunctional characteristics in the specification of software requirements

The proposed intelligent agent based on the ontological approach uses in its operation base ontologies of nonfunctional component characteristics of software quality (which were developed in [5, 13]) as the facts known to it. It is the ontologies that reflect cause-and-effect relations between notions that allow to identify the attributes missing in the specification and determine which nonfunctional component characteristics of the software quality cannot be determined without such attributes. For example, a fragment of the base ontology for Compatibility implemented in Protégé 4.2 is presented in Fig. 1. The agent compares these base ontologies with the information obtained from the specification of requirements to actual software presented as real ontologies to enable comparison of basic and real ontologies. Based on this comparison of ontologies performed in Protégé 4.2, the intelligent agent obtains a list of attributes missing in the specification since the actual ontologies differ from the base ones by the attributes that are missing in the specification. The intelligent agent analyzes the resulting list of missing attributes and dependences of nonfunctional attributes on the attributes (by the base ontologies) and determines which nonfunctional component characteristics of the software quality cannot be determined without the missing attributes. In addition, the intelligent agent counts the number of missing attributes and nonfunctional characteristics that cannot be calculated without certain attributes to form a numerical estimate of sufficiency of information in the specifications of software requirements. The intelligent agent then evaluates information in the specification of software requirements and decides on further actions, in particular, provides conclusions about sufficiency of the information and recommendations for its improvement.

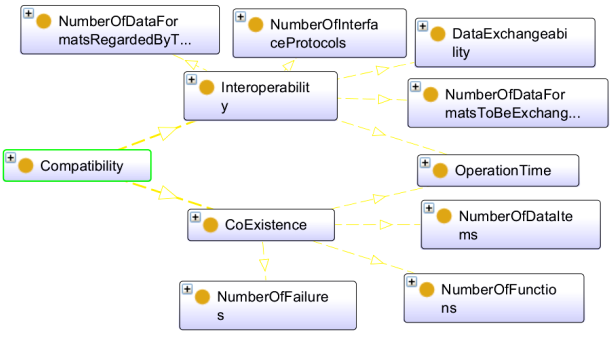

Fig. 1. Base ontology for Compatibility

Thus, the *process of evaluating the requirements specification by the intelligent agent* consists of:

1) comparison of ontologies of nonfunctional component characteristics of quality of actual software with base ontologies of nonfunctional component characteristics of software quality in order to identify the attributes absent in the specification of requirements to actual software according to which real ontologies were built;

2) identification of sub-characteristics and nonfunctional component characteristics of software quality which cannot be calculated on the basis of the attributes in the specification of requirements to actual software;

3) drawing up a conclusion on sufficiency or insufficiency of information in the specification of requirements to determine each nonfunctional characteristic of the software separately and to determine all nonfunctional characteristics of the software in general;

4) calculation of numerical estimates of the level of sufficiency of the information available in the specification of requirements for determining each nonfunctional characteristic of the software by formula (1):

$$D_j = \frac{\left(k_j - \sum_{i=1}^{k_j} \frac{qm_i}{qn_i}\right)}{k_j}, \quad (1)$$

where k_j is the number of sub-characteristics of the j-th nonfunctional characteristics of the software (j=1...8 since the Standard ISO 25010 [6] specifies exactly 8 nonfunctional component characteristics of software quality), qm_i is the

number of requirements missing in the specification for real software of attributes for the i-th sub-characteristic of the j-th nonfunctional characteristic of the software, qn_i is the number of required attributes for the i-th sub-characteristic of the j-th nonfunctional characteristic of the software (determined by the base ontologies for each nonfunctional component characteristic of software quality);

5) calculation of a numerical estimate of the level of sufficiency of the information available in the specification of requirements for determination of all nonfunctional component characteristics of software quality by formula (2):

$$D=\frac{\left(k-\sum_{j=1}^{k}\frac{qmc_j}{qnc_j}\right)}{k}, \quad (2)$$

where k is the number of nonfunctional component characteristics of software quality (k=8 according to ISO 25010 [6]), qmc_j is the number of requirements to the actual attribute software missing in the specification for the j-th nonfunctional characteristic of software, qnc_j is the number of attributes required for the j-th nonfunctional characteristic of the software (determined by the base ontologies for each nonfunctional component characteristic of software quality);

6) visualization of gaps in knowledge of nonfunctional component characteristic of software quality.

The proposed agent is intelligent as it automatically processes available knowledge (requirements to nonfunctional characteristics presented in a form of ontologies) and forms new knowledge (conclusions on the level of information sufficiency and recommendations for improving the level of information sufficiency in the specification of requirements).

This intelligent agent does not work with fuzzy data since the task of assessing information sufficiency does not imply irregularity. The required attribute is either present in the specification or absent in it and there is a drop in the level of information sufficiency to a value that depends on how many nonfunctional characteristics depend on this attribute (correlate according to it).

Let:

$$SMM_{Fs}=O_{Fs}\setminus(O_{Fs}\cap O_{Fs\_real}),$$

where SMM_{Fs}={fsa_1,..., $fsa_{(19\text{-}m)}$} is the set of attributes of Functional Suitability characteristics absent in the specification of requirements to actual software as well as the number of Functional Suitability characteristics that cannot be calculated on the basis of available attributes (required number of attributes is defined by ISO 25023:2016 [23]); O_{Fs} is the base ontology of Functional Suitability; O_{Fs_real} is the ontology of Functional Suitability for actual software;

$$SMM_{Pe}=O_{Pe}\setminus(O_{Pe}\cap O_{Pe\_real}),$$

where SMM_{Pe}={pea_1,..., $pea_{(30\text{-}n)}$} is the set of attributes of Performance Efficiency sub-characteristics absent in the specification of requirements to actual software as well as the number of Performance Efficiency sub-characteristics that cannot be calculated based on the available attributes; O_{Pe} is the base ontology of Performance Efficiency; O_{Pe_real} is the ontology of Performance Efficiency for actual software;

$$SMM_{Ub}=O_{Ub}\setminus(O_{Ub}\cap O_{Ub\_real}),$$

where SMM_{Ub}={uba_1,..., $uba_{(56\text{-}k)}$} is a set of attributes of sub-characteristics. Usability of the requirements absent in the specification of requirements to actual software as well as the number of sub-characteristics of Usability that cannot be calculated based on available attributes; O_{Ub} is the base ontology of Usability; O_{Ub_real} is the ontology of Usability for actual software;

$$SMM_{Rb}=O_{Rb}\setminus(O_{Rb}\cap O_{Rb\_real}),$$

where SMM_{Rb}={rba_1,..., $rba_{(35\text{-}o)}$} is the set of attributes of the Reliability sub-characteristics absent in the specification of requirements to the actual software as well as the number of sub-characteristics of Reliability that cannot be calculated based on the available attributes; O_{Rb} is the base ontology of Reliability; O_{Rb_real} is the ontology of Reliability for actual software;

$$SMM_{Cb}=O_{Cb}\setminus(O_{Cb}\cap O_{Cb\_real}),$$

where SMM_{Cb}={cba_1,..., $cba_{(12\text{-}p)}$} is the set of attributes of Compatibility sub-characteristics absent in the specification of requirements to the actual software as well as the number of Compatibility sub-characteristics that cannot be calculated based on available attributes; O_{Cb} is the base ontology of Compatibility; O_{Cb_real} is the ontology of Compatibility for actual software;

$$SMM_{Scr}=O_{Scr}\setminus(O_{Scr}\cap O_{Scr\_real}),$$

where SMM_{Scr}={$scra_1$,..., $scra_{(29\text{-}q)}$} is the set of attributes of the Security sub-characteristics absent in the specification of requirements to actual software as well as the number of Security sub-characteristics that cannot be calculated based on available attributes; O_{Scr} is the base ontology of Security; O_{Scr_real} is the ontology of Security for actual software;

$$SMM_{Mb}=O_{Mb}\setminus(O_{Mb}\cap O_{Mb\_real}),$$

where SMM_{Mb}={mba_1,..., $mba_{(39\text{-}r)}$} is the set of attributes of the Maintainability sub-characteristics absent in the specification of requirements to actual software as well as the number of Maintainability sub-characteristics that cannot be calculated based on the available attributes; O_{Mb} is the base ontology of Maintainability; O_{Mb_real} is the ontology of Maintainability for the actual software;

$$SMM_{Pb}=O_{Pb}\setminus(O_{Pb}\cap O_{Pb\_real}),$$

where SMM_{Pb}={pba_1,..., $pba_{(22\text{-}l)}$} is the set of attributes of Portability sub-characteristics absent in the specification of requirements to the actual software as well as the number of Portability sub-characteristics that cannot be calculated based on available attributes; O_{Pb} is the base ontology of Portability; O_{Pb_real} is the ontology of Portability for the actual software.

Production rules for drawing conclusions on sufficiency or insufficiency of information in the specification of requirements for determination of each nonfunctional component characteristic of software quality:

– for Functional Suitability:

if $SMM_{Fs} = \varnothing$ *then* "SRS information is sufficient for Functional Suitability"
else "SRS information is insufficient for Functional Suitability";

– for Performance Efficiency:

if $SMM_{Pe} = \varnothing$ *then* "SRS information is sufficient for Performance Efficiency"
else "SRS information is insufficient for Performance Efficiency";

– for Usability:

if $SMM_{Ub} = \varnothing$ *then* "SRS information is sufficient for Usability"
else "SRS information is insufficient for Usability";

– for Reliability:

if $SMM_{Rb} = \varnothing$ *then* "SRS information is sufficient for Reliability"
else "SRS information is insufficient for Reliability";

– for Compatibility:

if $SMM_{Cb} = \varnothing$ *then* "SRS information is sufficient for Compatibility"
else "SRS information is insufficient for Compatibility";

– for Security:

if $SMM_{Scr} = \varnothing$ *then* "SRS information is sufficient for Security"
else "SRS information is insufficient for Security";

– for Maintainability:

if $SMM_{Mb} = \varnothing$ *then* "SRS information is sufficient for Maintainability"
else "SRS information is insufficient for Maintainability";

– for Portability:

if $SMM_{Pb} = \varnothing$ *then* "SRS information is sufficient for Portability"
else "SRS information is insufficient for Portability".

The production rule for drawing up a conclusion on sufficiency or insufficiency of information in the specification of requirements for determination of all nonfunctional component characteristics of software quality:

if $(SMM_{Fs} \cup SMM_{Pe} \cup SMM_{Ub} \cup SMM_{Rb} \cup \cup SMM_{Cb} \cup SMM_{Scr} \cup SMM_{Mb} \cup SMM_{Pb}) = \varnothing$
then "SRS information is sufficient"
else "SRS information is insufficient".

The developed model of activity of the intelligent agent based on the ontological approach for evaluation of information about nonfunctional requirements in specifications reflects peculiarities of estimation of information sufficiency to determine nonfunctional component characteristics of software quality. This model serves as a theoretical basis for implementation of an intelligent agent based on the ontological approach for evaluating specifications of software requirements.

The method of activity of an intelligent agent based on the ontological approach for assessment of information about nonfunctional requirements in specifications of software requirements was developed in [24].

5. Analysis of nonfunctional characteristics in specifications of software requirements with the help of the intelligent agent

Based on the proposed model and the developed method, an intelligent agent based on the ontological approach was implemented to assess the specification of software requirements in PHP language using the Protégé 4.2 system (for work with ontologies).

Analysis of the information in the Specific Requirements section (in accordance with Standard ISO 29148:2011 [25]) (this particular section of the specification has Attributes of the Software System subsection which may contain values of all attributes necessary to determine the nonfunctional component characteristics of software quality of the specification of requirements to the software of transport and logistics system of Gilea Ltd., Ukraine) has provided the opportunity of building ontologies of nonfunctional characteristics for actual software. In accordance with the developed model and the method of activity, the intelligent agent based on the ontological approach performs comparison of ontologies of nonfunctional characteristics for actual software with the base ontologies of nonfunctional component characteristics of software quality.

As a result of this comparison, the intelligent agent based on the ontological approach has provided a set of attributes missing in the specification of software requirements for the Gilea transportation and logistics system:

– for Usability: Number of IO Data Items, Function Understandability, Number of Tasks, Help Accessibility, Operation Time, Number of Screens or Forms, Number of Interface Elements, Number of Unsuccessfully Recovered Situations, Number of Functions Implemented with the User Error Tolerance, Number of Interface Graphical Elements, Degree of Ergonomic Attractiveness, Freedom from Risk for Users with Specified Disabilities;

– for Functional Suitability: Functional Adequacy, Functional Implementation Completeness, Operation Time, Number of Data Items;

– for Performance Efficiency: Operation Time, Number of Tasks, Mean Amount of Throughput, Number of Failures, IO Loading Limits, Maximum Memory Utilization, Size of Database;

– for Reliability: Operation Time, Number of Failures, Number of Resolved Failures, Number of Breakdowns;

– for Compatibility: Operation Time, Number of Failures, Number of Data Items, Data Exchangeability;

– for Security: Operation Time, Number of Data Items, Access Controllability, Number of Access Types, Number of Provided Authentication Methods, Number of Events Processed using Digital Signature;

– for Maintainability: Operation Time, Number of Failures, Number of Resolved Failures, Number of Variables, Number of Data Items;

– for Portability: Operation Time, Number of Data Items, Number of Data Structures.

The analysis of influence of each element of the set of missing attributes on nonfunctional characteristics and their sub-characteristics carried out by an implemented intelligent agent based on the ontological approach has made it possible to calculate the number of missing attributes for sub-characteristics of each nonfunctional characteristic. In addition, this analysis has enabled drawing of a conclusion that based on the requirements to real software of attributes available in the specification, it is impossible to calculate sub-characteristics of nonfunctional characteristics, such as:

– sub-characteristics of Usability: Appropriateness Recognizability, Learnability, Operability, User Error Protection, User Interface Aesthetics, Accessibility, that is, information in this specification is insufficient to determine all 6 Usability sub-characteristics;

– sub-characteristics of Functional Suitability: Functional Completeness, Functional Correctness, Functional Appropriateness, that is, information in this specification is insufficient to determine all 3 Functional Suitability sub-characteristics;

– sub-characteristics of Performance Efficiency: Time Behavior, Resource Utilization, Capacity, that is, the information in this specification is insufficient to determine all 3 Performance Efficiency sub-characteristics;

– sub-characteristics of Reliability: Maturity, Availability, Fault Tolerance, Recoverability, that is, information in this specification is insufficient to determine all 4 Reliability sub-characteristics;

– sub-characteristics of Compatibility: CoExistence, Interoperability, that is, information in this specification is insufficient to determine both sub-characteristics of Compatibility;

– sub-characteristics of Security: Confidentiality, Integrity, NonRepudiation, Authenticity, that is, information in this specification is insufficient to determine 4 (of 5) sub-characteristics of Security;

– sub-characteristics of Maintainability: Modularity, Analyzability, Modifiability, Testability, that is, information in this specification is insufficient to determine 4 (of 5) sub-characteristics of Maintainability;

– sub-characteristics of Portability: Adaptability, Replaceability, that is, information in this specification is insufficient to determine 2 (of 3) sub-characteristics of Portability.

On the basis of the analysis, the implemented intelligence agent has formed a conclusion about insufficiency of information in the specification of requirements to software of Gilea Ltd. transport and logistics system for determining all nonfunctional characteristics. Numerical estimates of the level of information sufficiency in the specification of software requirements of the Gilea Ltd transport and logistics system for determination of all nonfunctional component characteristics of software quality are as follows:

– to determine Usability:

$$D_{Ub}=\frac{\left(6-\left(\frac{2}{6}+\frac{3}{8}+\frac{3}{13}+\frac{2}{11}+\frac{3}{6}+\frac{1}{5}\right)\right)}{6}=0,70;$$

– to determine Functional Suitability:

$$D_{Fs}=\frac{\left(3-\left(\frac{2}{4}+\frac{3}{5}+\frac{3}{6}\right)\right)}{3}=0,47;$$

– to determine Performance Efficiency:

$$D_{Pe}=\frac{\left(3-\left(\frac{3}{7}+\frac{4}{14}+\frac{3}{5}\right)\right)}{3}=0,56;$$

– to determine Reliability:

$$D_{Rb}=\frac{\left(4-\left(\frac{3}{14}+\frac{1}{4}+\frac{2}{5}+\frac{2}{7}\right)\right)}{4}=0,71;$$

– to determine Compatibility:

$$D_{Cb}=\frac{\left(2-\left(\frac{3}{4}+\frac{2}{5}\right)\right)}{2}=0,43;$$

– to determine Security:

$$D_{Scr}=\frac{\left(5-\left(\frac{4}{10}+\frac{4}{8}+\frac{1}{2}+\frac{0}{2}+\frac{1}{1}\right)\right)}{5}=0,52;$$

– to determine Maintainability:

$$D_{Mb}=\frac{\left(5-\left(\frac{4}{7}+\frac{0}{6}+\frac{2}{6}+\frac{2}{8}+\frac{2}{6}\right)\right)}{5}=0,70;$$

– to determine Portability:

$$D_{Pb}=\frac{\left(3-\left(\frac{3}{11}+\frac{0}{4}+\frac{1}{3}\right)\right)}{3}=0,80.$$

Numerical estimate of the level of sufficiency of information for determination of all nonfunctional component characteristics of software quality available in the specification of software requirements of Gilea Ltd. transport and logistics system is as follows:

$$D=\frac{\left(8-\left(\frac{14}{49}+\frac{8}{15}+\frac{10}{26}+\frac{8}{30}+\frac{5}{9}+\frac{10}{23}+\frac{10}{33}+\frac{4}{18}\right)\right)}{8}=0,63.$$

For example, *the implemented intelligent agent has provided the following conclusion*: "The available attributes in the analyzed specification are insufficient to determine all nonfunctional characteristics. Levels of information sufficiency in the analyzed specification: 70 % for Usability; 47 % for Functional Suitability; 56 % for Performance Efficiency; 71 % for Reliability; 43 % for Compatibility; 52 % for Security; 70 % for Maintainability; 80 % for Portability; 63 % for all nonfunctional characteristics in general. The specification needs to be supplemented with attributes for all nonfunctional characteristics".

The intelligent agent has also provided visualization of gaps in knowledge of all nonfunctional characteristics. This visualization is presented in Fig. 2–9 where the missing attributes are labeled as crossed out from the base ontology implemented in Protégé 4.2 and the sub-characteristics needing more attributes for their determination are encircled in the corresponding base ontology.

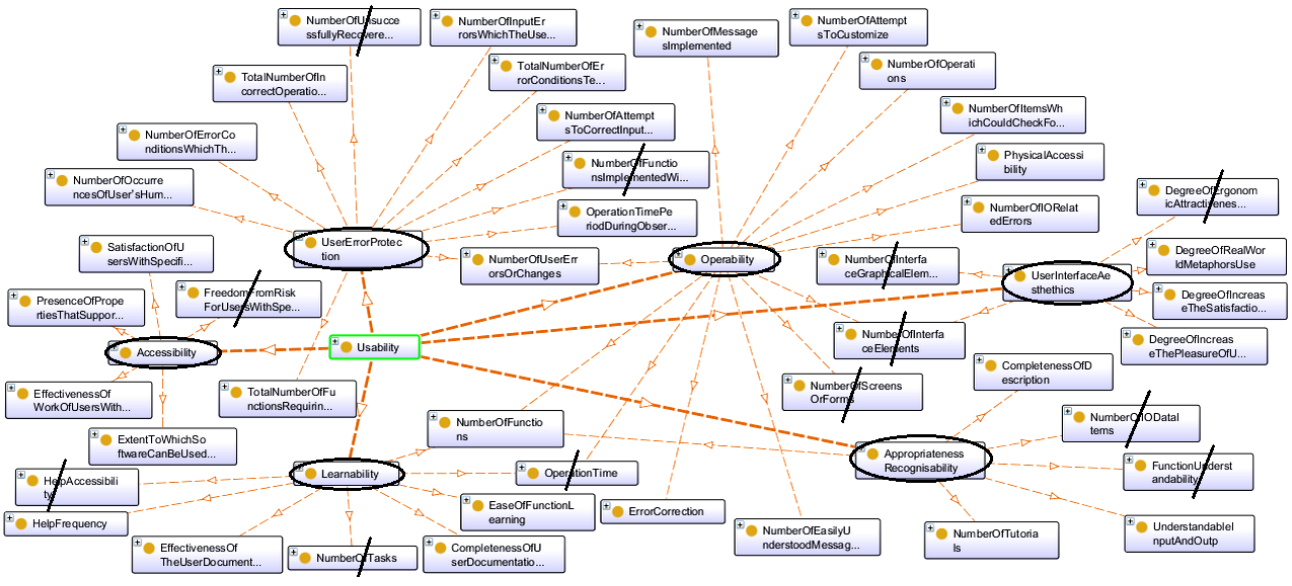

Fig. 2. Visualization of knowledge gaps in Usability

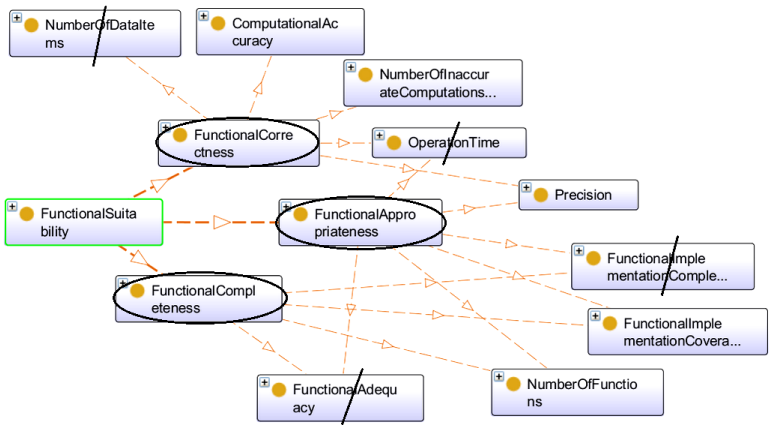

Fig. 3. Visualization of knowledge gaps in Functional Suitability

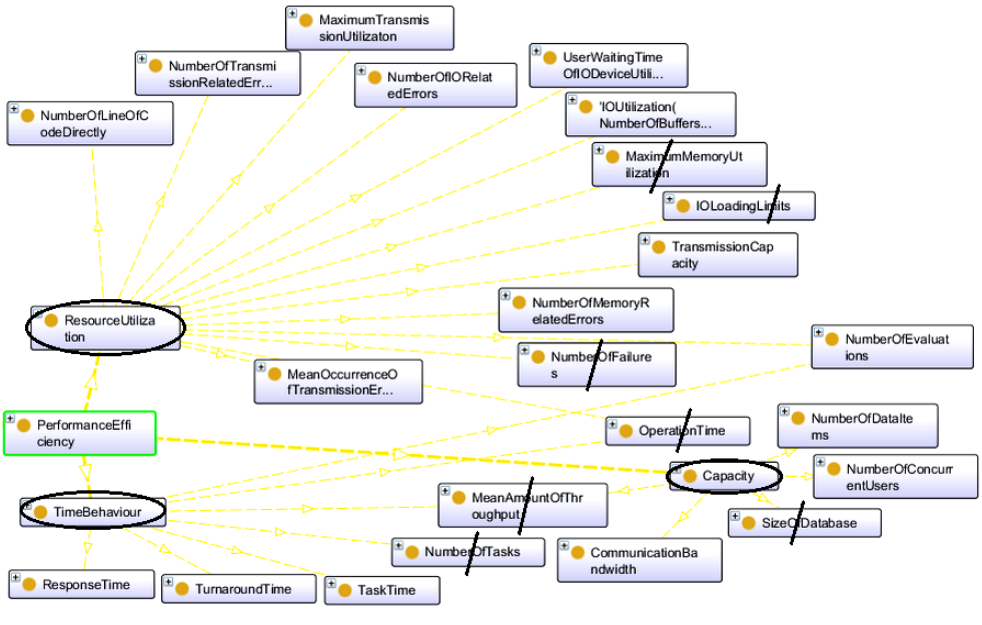

Fig. 4. Visualization of knowledge gaps in Performance Efficiency

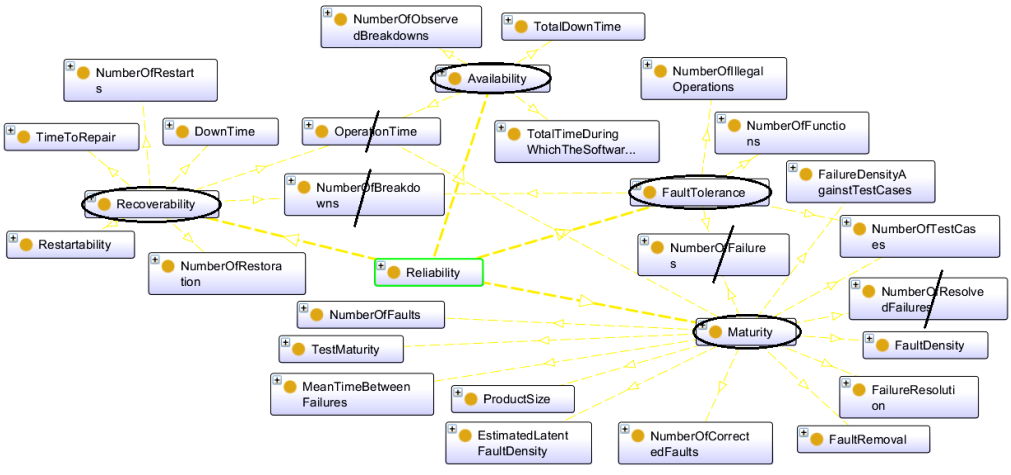

Fig. 5. Visualization of knowledge gaps in Reliability

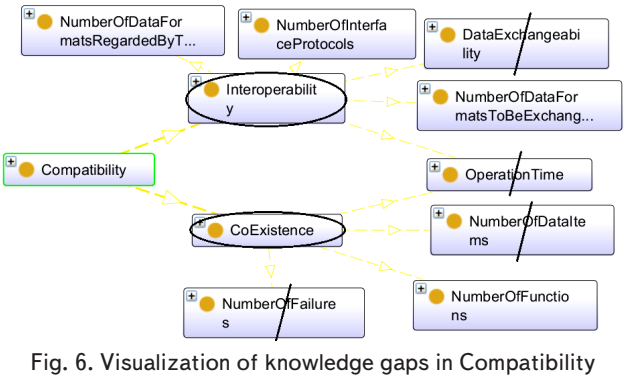

Fig. 6. Visualization of knowledge gaps in Compatibility

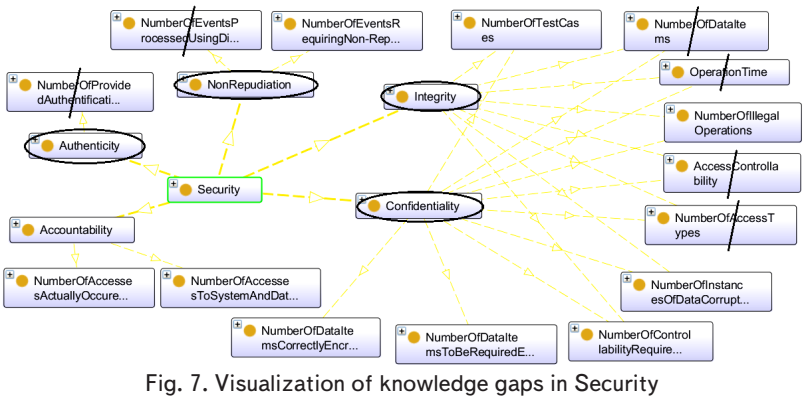

Fig. 7. Visualization of knowledge gaps in Security

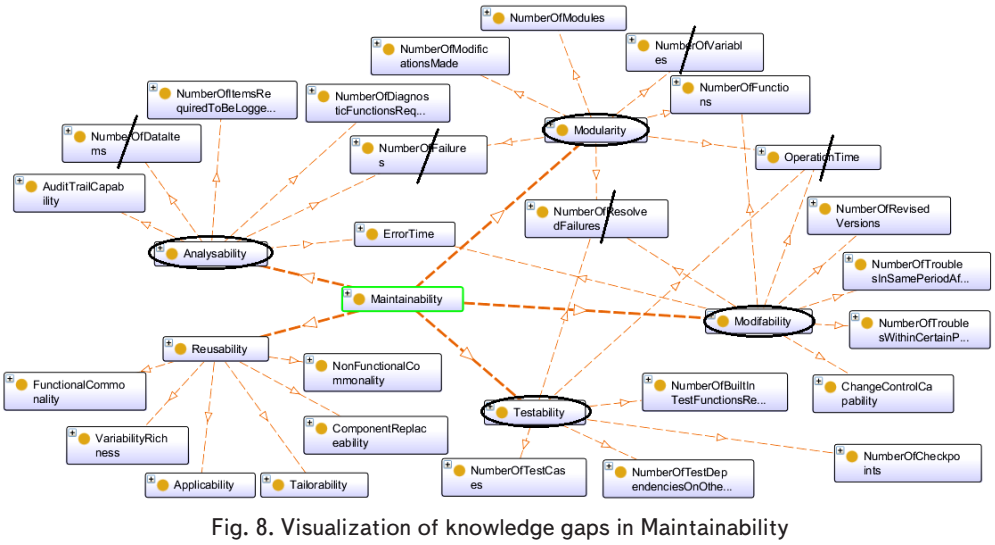

Fig. 8. Visualization of knowledge gaps in Maintainability

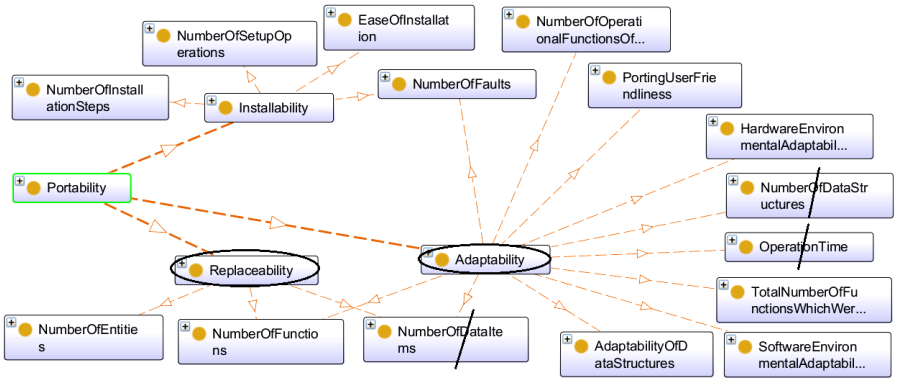

Fig. 9. Visualization of knowledge gaps in Portability

To conduct another experiment, the specification of requirements to the actual software of Deimos Ltd., Ukraine, automatic control system of production processes was used. Analysis of information in the Specific Requirements section of this specification has made it possible to build ontologies of nonfunctional characteristics for actual software. In accordance with the developed model and the method of activity, the realized intelligent agent has performed comparison of real ontologies with the base ontologies of nonfunctional component characteristics of software quality. As a result of this comparison, the intelligent agent based on the ontological approach has provided the following set of missing attributes in the specification of requirements to software of Deimos Ltd. automatic control system of production processes:

– for Usability: Number of Functions, Number of IO Data Items, Number of Tutorials, Number of Tasks, Completeness of User Documentation and/or Help Facility, Number of Screens or Forms, Number of IO Related Errors, Number of Interface Elements, Number of Easily Understood Messages, Total Number of Error Conditions Tested, Total Number of Incorrect Operation Patterns, Number of Interface Graphical Elements, Degree of Ergonomic Attractiveness, Satisfaction of Users with Specified Disabilities;

– for Functional Suitability: Number of Functions, Functional Adequacy, Number of Data Items, Computational Accuracy;

– for Performance Efficiency: Number of Tasks, Number of Evaluations, Number of Failures, Number of IO Related Errors, Number of Data Items, Number of Memory Related Errors, Maximum Memory Utilization, Size of Database;

– for Reliability: Number of Failures, Number of Test Cases, Number of Resolved Failures, Failure Density Against Test Cases, Number of Observed Breakdowns, Number of Functions, Number of Breakdowns, Restartability;

– for Compatibility: Number of Failures, Number of Functions, Number of Data Items, Number of Data Formats To Be Exchanged, Number of Interface Protocols;

– for Security: Number of Test Cases, Number of Data Items, Number of Access Types, Number of Data Items To Be Required Encryption/Decryption, Number of Events Processed Using Digital Signature, Number of Accesses to System and Data Recorded in the System Log, Number of Provided Authentication Methods;

– for Maintainability: Number of Failures, Number of Resolved Failures, Number of Functions, Number of Modules, Variability Richness, Component Replaceability, Number of Data Items, Number of Diagnostic Functions Required, Number of Test Cases, Number of Checkpoints;

– for Portability: Number of Functions, Number of Data Items, Number of Data Structures, Number of Setup Operations, Number of Installation Steps.

The analysis of influence of each element of the set of missing attributes on nonfunctional characteristics and their sub-characteristics which was carried out by the implemented intelligent agent based on the ontological approach has made it possible to calculate the number of missing attributes for sub-characteristics of each nonfunctional characteristic. In addition, this analysis has made it possible to draw a conclusion that it is impossible to calculate all sub-characteristics of nonfunctional characteristics based on the requirements to real software of attributes contained in the specification.

On the basis of the analysis, *the implemented intelligence agent has provided the following conclusion*: "The attributes available in the analyzed specification are insufficient to determine all nonfunctional characteristics. Levels of information sufficiency in the analyzed specification: 64 % for Usability; 59 % for Functional Suitability; 65 % for Performance Efficiency; 60 % for Reliability; 43 % for Compatibility; 45 % for Security; 57 % for Maintainability; 52 % for Portability; 59 % for all nonfunctional characteristics taken together. There is a need to complement this specification with attributes for all nonfunctional characteristics".

The intelligent agent has also provided for visualization of gaps in knowledge of all nonfunctional characteristics. This visualization is similar to visualization in the first experiment presented in Fig. 2–9. This visualization provides the user with a list of attributes missing in the specification of requirements to determine the nonfunctional component characteristics of software quality. In addition, it is this visualization that reflects what sub-characteristics are affected by this or that attribute and how much. After analyzing the visualization of knowledge gaps provided by the agent, developers can determine which attributes should be given priority to be entered to the specification of requirements in order to increase level of information sufficiency.

6. Discussion of results obtained in functioning of the developed intelligent agent based on the ontological approach

The developed model of activity and the implemented intelligent agent based on the ontological approach have provided an opportunity to automate analysis of specifications of requirements to software for sufficiency of their information. Automation of analysis of specifications of requirements to software has become possible due to the use of ontologies in functioning of the developed agent. It is ontologies that have provided detection of duplications and gaps in knowledge based on representation of missing logical links through visualization of cause-and-effect relations between notions and domain conceptualization by fixation of essences and relations. Such visualization of missing logical links reflects which attributes are not enough in the specification, which nonfunctional component characteristics of software quality are influenced by the lack of certain attri-

butes and the level of information sufficiency in a particular specification.

Unlike the agents developed in [15 20], the implemented intelligence agent was specially aimed at automation of evaluation of information sufficiency as regards the nonfunctional component characteristics of software quality in the specifications of requirements. Thus, it represents development of the concept of assessing information sufficiency given in [5, 13].

Intelligence of the implemented intelligent agent lies not only in drawing conclusions about sufficiency or lack of information and providing numerical estimates of the level of information sufficiency in the specification of requirements. It also recommends supplementing this specification with attributes necessary to determine the nonfunctional component characteristics of software quality (with providing a list and visualization of missing attributes) as analysis has shown in Section 5. If the specification writers pay attention to the agent's recommendations and supplement the specification with the necessary attributes, the level of information sufficiency in the specification of requirements necessary to determine the nonfunctional component characteristics of software quality increases.

For example, let the developers add the following attributes for Security in the specification of software requirements of Gilea Ltd. transport and logistics system: Access Controllability, Number of Access Types, Number of Provided Authentification Methods, Number of Events Processed Using Digital Signature. Then, the numerical estimate of the level of information sufficiency in this specification to determine Security will be as follows upon this addition:

$$D_{Scr} = \frac{\left(5-\left(\frac{2}{10}+\frac{2}{8}+\frac{0}{2}+\frac{0}{2}+\frac{0}{1}\right)\right)}{5} = 0{,}91.$$

Thus, completing the requirements specification with four attributes for Security will increase the level of information sufficiency to define Security by 39 % (from 52 % to 91 %). Accordingly, increase in the level of information sufficiency for one of the nonfunctional characteristics will result in an increase in the level of information sufficiency to determine all nonfunctional component characteristics of software quality. In addition, increase in the level of information sufficiency for one of the nonfunctional characteristics can also have a positive effect on the level of information sufficiency for other nonfunctional characteristics. After all, the used ontologies just show a correlation of nonfunctional attributes, that is, they instruct the user: which attributes need to be included in the specification as a matter of priority for a more rapid growth of information sufficiency. The raised level of information sufficiency for several nonfunctional characteristics will result in an even greater increase in information sufficiency to determine all nonfunctional component characteristics of software quality.

Consequently, the developed intelligent agent based on the ontological approach has provided an increase in the level of information sufficiency in the specification of software requirements to determine eight nonfunctional component characteristics of software quality. This is what distinguishes it from the known approaches described in [4, 21] which enable evaluation of information sufficiency just for one nonfunctional characteristic of software (Security) and the approach described in [22] is aimed at determining sufficiency of testing taking into consideration verification of software correspondence to the assigned requirements.

The developed ontology-based intelligent agent can be used for any software. Availability of a specification of software requirements is its only restriction.

Disadvantage of the proposed solution consists in the fact that information on the nonfunctional characteristics necessary for formation of real ontologies is selected currently from the specification of requirements to actual software in a manual way. To do this, the user opens base ontology of Software Engineering subject field (Software Quality section) and revises the relevant section of the specification for presence of the attributes specified in the base ontology. It is planned to automate this stage in the future: another agent will be developed that will conduct semantic analysis of the specification written in a natural language in a search for the attributes necessary to determine the nonfunctional component characteristics of software quality.

7. Conclusions

A model of activity of the intelligent agent based on the ontological approach for evaluation of the specification of software requirements has been developed. It is based on a comparative analysis of ontologies and is a theoretical basis for implementation of the intelligent agent based on the ontological approach. The intelligent agent has been implemented. It operates on the basis of the developed model and assesses sufficiency of information in the specification of requirements to determine all nonfunctional component characteristics of software quality. The implemented intelligent agent provides a conclusion about sufficiency or lack of information in the specification. In addition, it provides numerical estimates of information sufficiency to determine all nonfunctional characteristics of the software and determine all nonfunctional component characteristics of software quality. The agent also forms a list of attributes which should be added to the specification of requirements to raise level of sufficiency of its information and provides visualization of gaps in the knowledge of all nonfunctional component characteristics of software quality. Consequently, the implemented intelligent agent ensures automation of analysis of specifications of requirements to software for sufficiency of their information concerning the nonfunctional component characteristics of software quality. Thus, the represented agent enables partial elimination of human participation in processing information and gaining knowledge.

Information on nonfunctional characteristics in the specification of software requirements was analyzed with the help of the developed intelligent agent based on the ontological approach. The analysis has shown that all results of operation of the realized intelligent agent based on the ontological approach ensure in aggregate an increase in the level of information sufficiency in the specification of requirements to software to determine nonfunctional component characteristics of software quality. In addition, the results of operation of the implemented intelligent agent are aimed at avoiding loss of essential information and minimizing error occurrence at the early stages of the software life cycle.

References

1. Hastie S., Wojewoda S. Standish Group 2015 Chaos Report – Q&A with Jennifer Lynch. URL: http://www.infoq.com/articles/standish-chaos-2015
2. McConnell S. Code complete. Redmond, 2013. 896 p.
3. Levenson N. G. Engineering a safer world: systems thinking applied to safety. Cambridge, 2012. 560 p.
4. Cruickshank K. J. A validation metrics framework for safety-critical software-intensive systems. Monterey, 2009. 144 p.
5. Hovorushchenko T., Pomorova O. Information technology of evaluating the sufficiency of information on quality in the software requirements specifications // CEUR-WS. 2018. Vol. 2104. P. 555–570. URL: http://ceur-ws.org/Vol-2104/paper_228.pdf
6. ISO/IEC 25010:2011. Systems and Software Engineering. Systems and Software Quality Requirements and Evaluation (SQuaRE). System and Software Quality Models. Geneva, 2011. 34 p.
7. Gruber T. R. A translation approach to portable ontology specifications // Knowledge Acquisition. 1993. Vol. 5, Issue 2. P. 199–220. doi: https://doi.org/10.1006/knac.1993.1008
8. Burov E. Complex ontology management using task models // International Journal of Knowledge-Based and Intelligent Engineering Systems. 2014. Vol. 18, Issue 2. P. 111–120. doi: https://doi.org/10.3233/KES-140291
9. Burov E., Pasitchnyk V., Gritsyk V. Modeling software testing processes with task ontologies // British Journal of Education and Science. 2014. Vol. 2, Issue 6. P. 256–263.
10. Assawamekin N., Sunetnanta T., Pluempitiwiriyawej C. Ontology-based multiperspective requirements traceability framework // Knowledge and Information Systems. 2009. Vol. 25, Issue 3. P. 493–522. doi: https://doi.org/10.1007/s10115-009-0259-2
11. Ontology and Model Alignment as a Means for Requirements Validation / Kof L., Gacitua R., Rouncefield M., Sawyer P. // 2010 IEEE Fourth International Conference on Semantic Computing. 2010. doi: https://doi.org/10.1109/icsc.2010.95
12. An ontological approach to model software quality assurance knowledge domain / Bajnaid N. O., Benlamri R., Pakstas A., Salekzamankhani Sh. // Lecture Notes on Software Engineering. 2016. Vol. 4, Issue 3. P. 193–198.
13. Hovorushchenko T., Pomorova O. Ontological approach to the assessment of information sufficiency for software quality determination // CEUR-WS. 2016. Vol. 1614. P. 332–348.
14. Wooldridge M., Jennings N. R. Intelligent agents: theory and practice // The Knowledge Engineering Review. 1995. Vol. 10, Issue 2. P. 115–152. doi: https://doi.org/10.1017/s0269888900008122
15. Freitas A., Bordini R. H., Vieira R. Model-driven engineering of multi-agent systems based on ontologies // Applied Ontology. 2017. Vol. 12, Issue 2. P. 157–188. doi: https://doi.org/10.3233/ao-170182
16. Exploring an Ontological Approach for User Requirements Elicitation in the Design of Online Virtual Agents / Ossowska K., Szewc L., Weichbroth P., Garnik I., Sikorski M. // Lecture Notes in Business Information Processing. 2016. P. 40–55. doi: https://doi.org/10.1007/978-3-319-46642-2_3
17. Lezcano-Rodriguez L. A., Guzman-Luna J. A. Ontological characterization of basics of KAOS chart from natural language // ITECKNE. 2016. Vol. 13, Issue 2. P. 157–168. doi: https://doi.org/10.15332/iteckne.v13i2.1482
18. García-Magariño I., Gómez-Sanz J. J. An Ontological and Agent-Oriented Modeling Approach for the Specification of Intelligent Ambient Assisted Living Systems for Parkinson Patients // Lecture Notes in Computer Science. 2013. P. 11–20. doi: https://doi.org/10.1007/978-3-642-40846-5_2
19. Rakib A., Faruqui R. U. A Formal Approach to Modelling and Verifying Resource-Bounded Context-Aware Agents // Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering. 2013. P. 86–96. doi: https://doi.org/10.1007/978-3-642-36642-0_9
20. A Task-based Support Architecture for Developing Point-of-care Clinical Decision Support Systems for the Emergency Department / Michalowski W., O'Sullivan D., Farion K., Sayyad-Shirabad J., Kuziemsky C., Kukawka B., Wilk S. // Methods of Information in Medicine. 2013. Vol. 52, Issue 01. P. 18–32. doi: https://doi.org/10.3414/me11-01-0099
21. Hazard Analysis and Validation Metrics Framework for System of Systems Software Safety / Michael J. B., Shing M.-T., Cruickshank K. J., Redmond P. J. // IEEE Systems Journal. 2010. Vol. 4, Issue 2. P. 186–197. doi: https://doi.org/10.1109/jsyst.2010.2050159
22. Baker R., Habli I. An Empirical Evaluation of Mutation Testing for Improving the Test Quality of Safety-Critical Software // IEEE Transactions on Software Engineering. 2013. Vol. 39, Issue 6. P. 787–805. doi: https://doi.org/10.1109/tse.2012.56
23. ISO 25023:2016. Systems and Software Engineering. Systems and Software Quality Requirements and Evaluation (SQuaRE). Measurement of System and Software Product Quality. Geneva, 2016. 45 p.
24. Hovorushchenko T., Pavlova O. Method of Activity of Ontology-Based Intelligent Agent for Evaluating Initial Stages of the Software Lifecycle // Advances in Intelligent Systems and Computing. 2019. P. 169–178. doi: https://doi.org/10.1007/978-3-319-97885-7_17
25. ISO/IEC/IEEE 29148:2011. Systems and Software Engineering. Life Cycle Processes. Requirements Engineering. Geneva, 2011. 28 p.