Modeling of NonFunctional Characteristics of the Software for Selection of Accurate Scope of Information for Their Evaluation

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

[1, 2] Khmelnytskyi National University, Institutska str., 11, Khmelnytskyi, Ukraine
[3] National Aerospace University "Kharkiv Aviation Institute", Chkalova str., 17, Kharkiv, Ukraine
[1] tat_yana@ukr.net
[2] olya1607pavlova@gmail.com
[3] a.boyarchuk@csn.khai.edu

Abstract. The article proposed set-theoretic models of nonfunctional characteristics of software used to systematize information on nonfunctional characteristics and bring it to the common (unified) form in accordance with ISO 25010:2011, ISO 25023:2016. In addition, basic and real ontological models of nonfunctional characteristics of software are developed based on ISO 25010:2011 and ISO 25023:2016 standards. These models provide the basis for choosing the sufficient information to evaluate nonfunctional characteristics of software.

Keywords: Software, Software Project, Software Requirements Specification (SRS), NonFunctional Software Characteristics, Ontology.

1 Introduction

Now mankind is often requiring the software for solving the difficult problems and the high-value software projects are rapidly developed. Therefore, software errors threaten with disasters that lead to human and environmental casualties, disasters, losses of time and finances. Most software failures were due to errors in the requirements specification [1]. So, the success of the software project's implementation (both the timely implementation of the software project within the allocated budget and the implementation of all necessary features and functions [2]) essentially depends on the early life cycle stages, therefore, the *urgent task* is automation of evaluating the level of early life cycle stages based on the specifications' analysis (assessing the sufficiency of specification's information as a key factor of the software projects' success [3]), and special attention should be focused on the nonfunctional characteristics–components of quality of software (according to ISO 25010 [4] these nonfunctional characteristics are: Reliability, Performance Efficiency, Functional Suitability, Compatibility, Maintainability, Portability, Usability, Security).

The evaluation of nonfunctional characteristics-components of quality of software according to ISO 25010 standard [4] is carried out as follows: based on the attributes

specified in ISO 25023 [5], subcharacteristics are evaluated, which in turn assess the nonfunctional characteristics. Then, with sufficient information for the nonfunctional characteristics in the specification, we will understand the presence in the requirements of all elements (attributes) necessary to determine the nonfunctional characteristics. The analysis of ISO 25010 [4], ISO 25023 [5] standards made it possible to conclude that there are attributes on which more than one nonfunctional characteristic depends, that is, there is a correlation of nonfunctional characteristics according to certain attributes (so according to ISO 25023 [5] subcharacteristics of nonfunctional characteristics depends on 203 attributes, but only from 138 various attributes). The relationships (correlations) between characteristics and subcharacteristics on attributes influences on the significance of such attributes [6]. If the attributes that are part of several nonfunctional characteristics are absent (there is a lack of information), the simultaneous use of these attributes will significantly affect the reliability of the estimates of nonfunctional characteristics-components of quality of the software. In such case, it is important to mitigate the influence of the mutual correlation of nonfunctional characteristics on attributes. Such mitigation is done by identifying common attributes and ensuring their availability. Thus, it was found that information about nonfunctional characteristics is conveniently presented as ontologies [7], which allow to reveal the causal relationships between concepts and to identify missing attributes, their influence on one or more nonfunctional characteristics, and also the correlation of nonfunctional characteristics for attributes. For this, we use the ontologies, which systemate the subject industry, holistic represent the subject industry, identificate gaps in knowledge, visualize missing logical connections, provide basis for intellectual agents [8-10].

In order to solve the problem of assessing the adequacy of the volume of information for determining the nonfunctional characteristics in the specifications of the requirements, it's necessary the basic (universal) ontology for software quality domain based on ISO 25010 [4], ISO 25023 [5], the components of which will be ontologies for each of the nonfunctional characteristics-components of quality of software. Basic ontologies will display the required information (attributes) that should be available in the specification of software requirements to ensure that its information about nonfunctional characteristics is sufficient. During developing the specific software, in addition to the basic ontologies, it is necessary to have the ontology of this software, which will reflect the requirements in the specifications for the specific software attributes necessary for determining the nonfunctional characteristics. Comparison of developed ontology of the specific software with basic ontologies allows to identify the deficiency (Fig. 1 [3]) or sufficiency (Fig. 2 [3]) of information about nonfunctional characteristics in the specification of requirements for the specific software.

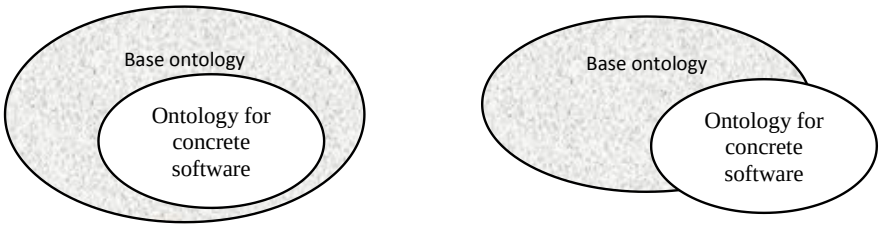

Fig. 1. Two cases of deficiency of requirements' information

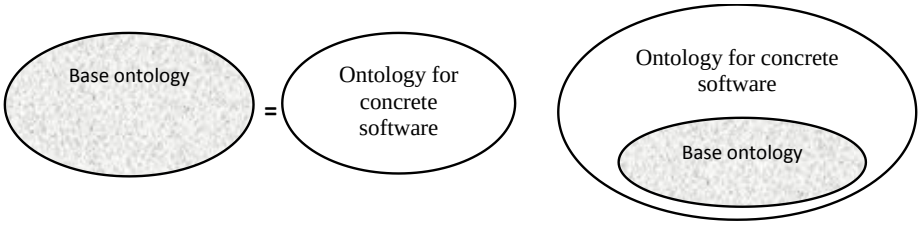

Fig. 2. Two cases of sufficiency of requirements' information

Criterion of the sufficiency of requirements' information was developed in [3]. Let SAMI is the set of missing measures:

$$SAMI = Base\ ontology \setminus (Base\ ontology \cap Ontology\ for\ concrete\ software). \quad (1)$$

Then:

- if SAMI $=\varnothing$, then requirements' information is sufficient;
- if SAMI $\neq\varnothing$, then information in the SRS is insufficient, and the SRS requires the complement by attributes of nonfunctional characteristics.

In order to determine the structure and filling of the basic (universal) ontologies of software quality domain, set-theoretical and ontological models for nonfunctional characteristics are needed, which is the *objective of this study*.

2 Models of NonFunctional Characteristics-Software Quality Components

Models of nonfunctional characteristics-components of quality of software contain information about nonfunctional characteristics, their subcharacteristics and attributes, taken from the standards [4, 5]. Since nonfunctional characteristics models are used to systematize information about nonfunctional characteristics and bring it to the common (unified) view in accordance with standards, then it is expedient to develop such models in the set-theoretical form.

According to the ISO 25010: 2011 standard, such nonfunctional characteristics as Functional Suitability (Fs), Performance Efficiency (Pe), Usability (Ub), Reliability (Rb), Compatibility (Cb), Security (Scr), Maintainability (Mb), Portability (Pb) are functions of several subcharacteristics, then *we represent each of the above-mentioned nonfunctional characteristics as a function of the subcharacteristic* [7]:

$$F\_S = f_1(F\_Com, F\_Cor, F\_Appr), \quad (2)$$

where F_Com – Functional Completeness, F_Cor – Functional Correctness, F_Appr – Functional Appropriateness;

$$P\_E = f_2(T\_B, R\_U, Cc), \quad (3)$$

where T_B – Time Behaviour, R_U – Resource Utilization, Cc – Capacity;

$$Ub = f_3(A\_R, Lb, Ob, U\_E\_P, U\_I\_A, Ab), \quad (4)$$

where A_R – Appropriateness Recognisability, Lb – Learnability, Ob – Operability, U_E_P – User Error Protection, U_I_A – User Interface Aesthethics, Ab – Accessibility;

$$Rb = f_4(Mat, Avb, F\_T, Rcv), \quad (5)$$

where Mat – Maturity, Avb – Availability, F_T – Fault Tolerance, Rcv – Recoverability;

$$Cb = f_5(C\_E, Ib), \quad (6)$$

where C_E – CoExistence, Ib – Interoperability;

$$Scr = f_6(Conf, Int, N\_R, Acb, Auth), \quad (7)$$

where Conf – Confidentiality, Int – Integrity, N_R – Non Repudiation, Acb – Accountability, Auth – Authenticity;

$$Mb = f_7(Mod, Rub, Anb, Mdfb, Tsb), \quad (8)$$

where Mod – Modularity, Rub – Reusability, Anb – Analysability, Mdfb – Modifability, Tsb – Testability;

$$Pb = f_8(Adb, Inb, Rpb), \quad (9)$$

where Adb – Adaptability, Inb – Installability, Rpb – Replaceability.

Then *the sets of the subcharacteristics of nonfunctional characteristics* have a form:

A = {F_Com, F_Cor, F_Appr} – the set of the subcharacteristics of Functional Suitability;

B = {T_B, R_U, Cc} – the set of the subcharacteristics of Performance Efficiency;

C = {A_R, Lb, Ob, U_E_P, U_I_A, Ab} – the set of the subcharacteristics of Usability,

D = {Mat, Avb, F_T, Rcv} – the set of the subcharacteristics of Reliability;

E = {C_E, Ib} – the set of the subcharacteristics of Compatibility;

F = {Conf, Int, N_R, Acb, Auth} – the set of the subcharacteristics of Security;

G = {Mod, Rub, Anb, Mdfb, Tsb} – the set of the subcharacteristics of Maintainability;

H = {Adb, Inb, Rpb} – the set of the subcharacteristics of Portability.

At the same time, each subcharacteristic of nonfunctional characteristics is a function of certain attributes described in ISO 25023. *Depending on the subcharacteristics from the attributes* is represented as follows:

$$F\_Com = \varphi_1 (N\_o\_F, F\_I\_Cn, F\_Aq, F\_I\_C), \quad (10)$$

where N_o_F – Number of Functions, F_I_Cn – Functional Implementation Completeness, F_Aq – Functional Adequacy, F_I_C – Functional Implementation Coverage;

$$F\_Cor = \varphi_2 (O\_T, N\_I\_C, N\_D\_I, C\_A, Pc), \quad (11)$$

where O_T – Operation Time, N_I_C – Number of Inaccurate Computations Encountered by Users, N_D_I – Number of Data Items, C_A – Computational Accuracy, Pc – Precision;

$$F\_Appr = \varphi_3 (O\_T, N\_o\_F, F\_I\_Cn, F\_Aq, F\_I\_C, Pc), \quad (12)$$

$$T\_B = \varphi_4 (O\_T, Nm\_o\_T, R\_T, N\_o\_E, Tn\_T, Tsk\_T, M\_A\_Thr), \quad (13)$$

where Nm_o_T – Number of Tasks, R_T – Response Time, N_o_E – Number of Evaluations, Tn_T – Turnaround Time, Tsk_T – Task Time, M_A_Thr – Mean Amount of Throughput;

$$R\_U = \varphi_5 (O\_T, N\_o\_Fl, N\_o\_E, N\_IO\_R\_E, U\_W\_T, N\_M\_R\_E, N\_T\_R\_E,$$

$$T\_Cc, IO\_U, N\_o\_L\_C\_D, IO\_L\_L, M\_M\_U, M\_T\_U, M\_O\_T\_E), \quad (14)$$

where N_o_Fl – Number of Failures, N_IO_R_E – Number of IO Related Errors, U_W_T – User Waiting Time of IO Device Utilization, N_M_R_E – Number of Memory Related Errors, N_T_R_E – Number of Transmission Related Error, T_Cc – Transmission Capacity, IO_U – IO Utilization (Number Of Buffers), N_o_L_C_D – Number of Line of Code Directly, IO_L_L – IO Loading Limits, M_M_U – Maximum Memory Utilization, M_T_U – Maximum Transmission Utilizaton, M_O_T_E – Mean Occurrence of Transmission Error;

$$Cc = \varphi_6 (N\_D\_I, N\_C\_U, C\_Bw, M\_A\_Thr, S\_Db), \quad (15)$$

where N_C_U – Number of Concurrent Users, C_Bw – Communication Bandwidth, S_Db – Size of Database;

$$A\_R = \varphi_7 (N\_o\_F, N\_o\_Tt, N\_IO\_D\_I, Cn\_D, F\_Ua, Ua\_IO), \quad (16)$$

where N_o_Tt – Number of Tutorials, N_IO_D_I – Number of IO Data Items, Cn_D – Completeness of Description, F_Ua – Function Understandability, Ua_IO – Understandable Input and Output;

$$Lb = \varphi_8 (N\_o\_F, O\_T, E\_F\_L, Nm\_o\_T, H\_Fq, E\_U\_D\_H\_S, H\_Aa,$$

C_U_D_H), (17)

where E_F_L – Ease of Function Learning, H_Fq – Help Frequency, E_U_D_H_S – Effectiveness of the User Documentation and / or Help System, H_Aa – Help Accessibility, C_U_D_H – Completeness of User Documentation and / or Help Facility;

$$Ob = \varphi_9(N\_o\_F, O\_T, E\_Cr, N\_S\_F, N\_U\_E\_C, N\_A\_C, N\_IO\_R\_E, N\_Op,$$

$$N\_I\_W\_C\_C\_V\_D, N\_M\_I, N\_I\_E, Ph\_A, N\_E\_U\_M), \quad (18)$$

where E_Cr – Error Correction, N_S_F – Number of Screens or Forms, N_U_E_C – Number of User Errors or Changes, N_A_C – Number of Attempts to Customize, N_Op – Number of Operations, N_I_W_C_C_V_D – Number of Items which Could Check for Valid Data, N_M_I – Number of Messages Implemented, N_I_E – Number of Interface Elements, Ph_A – Physical Accessibility, N_E_U_M – Number of Easily Understood Messages;

$$U\_E\_P = \varphi_{10}(N\_U\_R\_S, N\_U\_E\_C, O\_T\_P\_D\_O, N\_O\_U\_H\_E\_O,$$

$$N\_I\_E\_W\_U\_S\_C, N\_A\_C\_I\_E, N\_E\_C\_W\_U\_S\_C, T\_N\_E\_C\_T, N\_F\_I\_U\_E\_T,$$

$$T\_N\_F\_R\_T\_C, T\_N\_I\_O\_P), \quad (19)$$

where N_U_R_S – Number of Unsuccessfully Recovered Situation, O_T_P_D_O – Operation Time Period During Observation, N_O_U_H_E_O – Number of Occurrences of User's Human Error Operation, N_I_E_W_U_S_C – Number of Input Errors which the User Successfully Corrects, N_A_C_I_E – Number of Attempts to Correct Input Errors, N_E_C_W_U_S_C – Number of Error Conditions which the User Successfully Corrects, T_N_E_C_T – Total Number of Error Conditions Tested, N_F_I_U_E_T – Number of Functions Implemented with User Error Tolerance, T_N_F_R_T_C – Total Number of Functions Requiring the Tolerance Capability, T_N_I_O_P – Total Number of Incorrect Operation Patterns;

$$U\_I\_A = \varphi_{11}(N\_I\_E, N\_I\_G\_E, D\_I\_P\_U, D\_I\_S\_U, D\_E\_A, D\_R\_W\_M\_U), \quad (20)$$

where N_I_G_E – Number of Interface Graphical Elements, D_I_P_U – Degree of Increase the Pleasure of User, D_I_S_U – Degree of Increase the Satisfaction of User, D_E_A – Degree of Ergonomic Attractiveness, D_R_W_M_U – Degree of Real World Metaphors Use;

$$Ab = \varphi_{12}(E\_W\_S\_C\_B\_U\_U\_S\_D, E\_W\_U\_S\_D, F\_F\_R\_U\_S\_D, S\_U\_S\_D,$$

$$P\_P\_S\_A), \quad (21)$$

where E_W_S_C_B_U_U_S_D – Extent to which Software Can Be Used by Users with Specified Disabilities, E_W_U_S_D – Effectiveness of Work of Users with Specified Disabilities, F_F_R_U_S_D – Freedom from Risk for Users With Specified Disabilities, S_U_S_D – Satisfaction of Users With Specified Disabilities, P_P_S_A – Presence of Properties That Support Accessibility;

$$Mat = \varphi_{13}(O\_T, N\_o\_Ft, N\_o\_Fl, P\_S, N\_T\_C, N\_R\_F, N\_C\_F, F\_D\_A\_T\_C, F\_Rn, F\_Rl, M\_T\_B\_F, T\_My, E\_L\_F\_D, F\_Dy), \quad (22)$$

where N_o_Ft – Number of Faults, P_S – Product Size, N_T_C – Number of Test Cases, N_R_F – Number of Resolved Failures, N_C_F – Number of Corrected Faults, F_D_A_T_C – Failure Density Against Test Cases, F_Rn – Failure Resolution, F_Rl – Fault Removal, M_T_B_F – Mean Time Between Failures, T_My – Test Maturity, E_L_F_D – Estimated Latent Fault Density, F_Dy – Fault Density;

$$Avb = \varphi_{14}(O\_T, T\_T\_D\_W\_S\_I\_S, N\_O\_B, T\_D\_T), \quad (23)$$

where T_T_D_W_S_I_S – Total Time During which the Software Is in an UpState, N_O_B – Number of Observed Breakdowns, T_D_T – Total Down Time;

$$F\_T = \varphi_{15}(N\_o\_Fl, N\_T\_C, N\_Bd, N\_o\_F, N\_I\_O), \quad (24)$$

where N_Bd – Number of Breakdowns, N_I_O – Number of Illegal Operations;

$$Rcv = \varphi_{16}(O\_T, N\_Bd, T\_t\_R, D\_T, N\_Rs, N\_Rn, Ray), \quad (25)$$

where T_t_R – Time to Repair, D_T – Down Time, N_Rs – Number of Restarts, N_Rn – Number of Restoration, Ray – Restartability;

$$C\_E = \varphi_{17}(O\_T, N\_o\_Fl, N\_o\_F, N\_D\_I), \quad (26)$$

$$Ib = \varphi_{18}(O\_T, N\_D\_F\_R\_T, N\_D\_F\_B\_E, N\_I\_P, D\_Eay), \quad (27)$$

where N_D_F_R_T – Number of Data Formats Regarded by Tool, N_D_F_B_E – Number of Data Formats to Be Exchanged, N_I_P – Number of Interface Protocols, D_Eay – Data Exchangeability;

$$Conf = \varphi_{19}(O\_T, N\_I\_O, N\_T\_C, N\_I\_D\_C, N\_D\_I, N\_A\_T, N\_C\_R, A\_Ca, N\_D\_I\_C\_E\_D, N\_D\_I\_B\_R\_E\_D), \quad (28)$$

where N_I_D_C – Number of Instances of Data Corruption, N_A_T – Number of Access Types, N_C_R – Number of Controllability Requirements, A_Ca – Access Controllability, N_D_I_C_E_D – Number of Data Items Correctly Encrypted Decrypted, N_D_I_B_R_E_D – Number of Data Items to be Required Encryption Decryption;

$$Int = \varphi_{20}(O\_T, N\_I\_O, N\_T\_C, N\_I\_D\_C, N\_D\_I, N\_A\_T, N\_C\_R, A\_Ca), \quad (29)$$

$$N\_R = \varphi_{21}(N\_E\_P\_U\_D\_S, N\_E\_R\_N\_R\_P), \quad (30)$$

where N_E_P_U_D_S – Number of Events Processed Using Digital Signature, N_E_R_N_R_P – Number of Events Requiring Non-Repudiation Property;

$$Acb = \varphi_{22}(N\_A\_S\_D\_R\_S\_L, N\_A\_A\_O), \quad (31)$$

where N_A_S_D_R_S_L – Number of Accesses to System and Data Recorded in the System Log, N_A_A_O – Number of Accesses Actually Occured;

$$\text{Auth} = \varphi_{23}(\text{N\_P\_A\_M}), \tag{32}$$

where N_P_A_M – Number of Provided Authentification Methods;

$$\text{Mod} = \varphi_{24}(\text{O\_T, N\_o\_Fl, N\_R\_F, N\_M\_M, N\_V, N\_o\_F, N\_M}), \tag{33}$$

where N_M_M – Number of Modifications Made, N_V – Number of Variables, N_M – Number of Modules;

$$\text{Rub} = \varphi_{25}(\text{F\_Cy, N\_F\_Cy, V\_Rn, Aay, Tay, C\_Ra}), \tag{34}$$

where F_Cy – Functional Commonality, N_F_Cy – Non Functional Commonality, V_Rn – Variability Richness, Aay – Applicability, Tay – Tailorability, C_Ra – Component Replaceability;

$$\text{Anb} = \varphi_{26}(\text{N\_o\_Fl, N\_D\_I, Er\_T, N\_I\_R\_B\_L, N\_D\_F\_R, A\_T\_C}), \tag{35}$$

where Er_T – Error Time, N_I_R_B_L – Number of Items Required to Be Logged, N_D_F_R – Number of Diagnostic Functions Required, A_T_C – Audit Trail Capability;

$$\text{Mdfb} = \varphi_{27}(\text{O\_T, N\_R\_V, N\_R\_F, Er\_T, N\_o\_F, C\_C\_Ca, N\_T\_C\_P\_B\_M, N\_T\_S\_P\_A\_M}), \tag{36}$$

where N_R_V – Number of Revised Versions, C_C_Ca – Change Control Capability, N_T_C_P_B_M – Number of Troubles within Certain Period Before Modification, N_T_S_P_A_M – Number of Troubles in Same Period After Modification;

$$\text{Tsb} = \varphi_{28}(\text{O\_T, N\_T\_C, N\_R\_F, N\_B\_T\_F\_R, N\_T\_D\_O\_S, N\_Cp}), \tag{37}$$

where N_B_T_F_R – Number of Built in Test Functions Required, N_T_D_O_S – Number of Test Dependencies on Other Systems, N_Cp – Number of Checkpoints;

$$\text{Adb} = \varphi_{29}(\text{N\_o\_F, O\_T, N\_o\_Ft, P\_U\_F, N\_D\_I, N\_D\_S, Aa\_D\_S, H\_E\_A, S\_E\_A, N\_O\_F\_T\_W\_N\_C\_A, T\_N\_F\_W\_T\_D\_E}), \tag{38}$$

where P_U_F – Porting User Friendliness, N_D_S – Number of Data Structures, Aa_D_S – Adaptability of Data Structures, H_E_A – Hardware Environmental Adaptability, S_E_A – Software Environmental Adaptability, N_O_F_T_W_N_C_A – Number of Operational Functions of which Tasks Were Not Completed or Adequated, T_N_F_W_T_D_E – Total Number of Functions which Were Tested in Different Environment;

$$\text{Inb} = \varphi_{30}(\text{N\_o\_Ft, N\_S\_O, N\_I\_S, E\_o\_I}), \tag{39}$$

where N_S_O – Number of Setup Operations, N_I_S – Number of Installation Steps, E_o_I – Ease of Installation;

$$Rpb = \varphi_{31}(N\_o\_F, N\_D\_I, N\_Et), \tag{40}$$

where N_Et – Number of Entities.

Then *the sets of attributes for subcharacteristics of nonfunctional characteristics–software quality components* have the form:

I = {N_o_F, F_I_Cn, F_Aq, F_I_C} – the set of attributes for Functional Completeness; J = {O_T, N_I_C, N_D_I, C_A, Pc} – the set of attributes for Functional Correctness; K={O_T, N_o_F, F_I_Cn, F_Aq, F_I_C, Pc} – the set of attributes for Functional Appropriateness;

L = {O_T, Nm_o_T, R_T, N_o_E, Tn_T, Tsk_T, M_A_Thr} – the set of attributes for Time Behaviour; M = {O_T, N_o_Fl, N_o_E, N_IO_R_E, U_W_T, N_M_R_E, N_T_R_E, T_Cc, IO_U, N_o_L_C_D, IO_L_L, M_M_U, M_T_U, M_O_T_E} – the set of attributes for Resource Utilization; N = {N_D_I, N_C_U, C_Bw, M_A_Thr, S_Db} – the set of attributes for Capacity;

O = {N_o_F, N_o_Tt, N_IO_D_I, Cn_D, F_Ua, Ua_IO} – the set of attributes for Appropriateness Recognisability; P = {N_o_F, O_T, E_F_L, Nm_o_T, H_Fq, E_U_D_H_S, H_Aa, C_U_D_H} – the set of attributes for Learnability; Q = {N_o_F, O_T, E_Cr, N_S_F, N_U_E_C, N_A_C, N_IO_R_E, N_Op, N_I_W_C_C_V_D, N_M_I, N_I_E, Ph_A, N_E_U_M} – the set of attributes for Operability; R = {N_U_R_S, N_U_E_C, O_T_P_D_O, N_O_U_H_E_O, N_I_E_W_U_S_C, N_A_C_I_E, N_E_C_W_U_S_C, T_N_E_C_T, N_F_I_U_E_T, T_N_F_R_T_C, T_N_I_O_P} – the set of attributes for User Error Protection; S = {N_I_E, N_I_G_E, D_I_P_U, D_I_S_U, D_E_A, D_R_W_M_U} – the set of attributes for User Interface Aesthethics; T = {E_W_S_C_B_U_U_S_D, E_W_U_S_D, F_F_R_U_S_D, S_U_S_D, P_P_S_A} – the set of attributes for Accessibility;

U = {O_T, N_o_Ft, N_o_Fl, P_S, N_T_C, N_R_F, N_C_F, F_D_A_T_C, F_Rn, F_Rl, M_T_B_F, T_My, E_L_F_D, F_Dy} – the set of attributes for Maturity; V = {O_T, T_T_D_W_S_I_S, N_O_B, T_D_T} – the set of attributes for Availability; W = {N_o_Fl, N_T_C, N_Bd, N_o_F, N_I_O} – the set of attributes for Fault Tolerance; X = {O_T, N_Bd, T_t_R, D_T, N_Rs, N_Rn, Ray} – the set of attributes for Recoverability;

Y = {O_T, N_o_Fl, N_o_F, N_D_I} – the set of attributes for CoExistence; Z = {O_T, N_D_F_R_T, N_D_F_B_E, N_I_P, D_Eay} – the set of attributes for Interoperability;

AA = {O_T, N_I_O, N_T_C, N_I_D_C, N_D_I, N_A_T, N_C_R, A_Ca, N_D_I_C_E_D, N_D_I_B_R_E_D} – the set of attributes for Confidentiality; AB = {O_T, N_I_O, N_T_C, N_I_D_C, N_D_I, N_A_T, N_C_R, A_Ca} – the set of attributes for Integrity; AC = {N_E_P_U_D_S, N_E_R_N_R_P} – the set of attributes for Non Repudiation; AD = {N_A_S_D_R_S_L, N_A_A_O} – the set of attributes for Accountability; AE = {N_P_A_M} – the set of attributes for Authenticity;

AF = {O_T, N_o_Fl, N_R_F, N_M_M, N_V, N_o_F, N_M} – the set of attributes for Modularity; AG = {F_Cy, N_F_Cy, V_Rn, Aay, Tay, C_Ra} – the set of attributes for Reusability; AH = {N_o_Fl, N_D_I, Er_T, N_I_R_B_L, N_D_F_R, A_T_C}

– the set of attributes for Analysability; AI = {O_T, N_R_V, N_R_F, Er_T, N_o_F, C_C_Ca, N_T_C_P_B_M, N_T_S_P_A_M} – the set of attributes for Modifability; AJ = {O_T, N_T_C, N_R_F, N_B_T_F_R, N_T_D_O_S, N_Cp} – the set of attributes for Testability;

AK = {N_o_F, O_T, N_o_Ft, P_U_F, N_D_I, N_D_S, Aa_D_S, H_E_A, S_E_A, N_O_F_T_W_N_C_A, T_N_F_W_T_D_E} – the set of attributes for Adaptability; AL = {N_o_Ft, N_S_O, N_I_S, E_o_I} – the set of attributes for Installability; AM = {N_o_F, N_D_I, N_Et} – the set of attributes for Replaceability.

Considering the resulting sets of attributes and sub-characteristics of nonfunctional characteristics, obtained functions of dependencies of characteristics on sub-characteristics and characteristics on attributes, as well as the relationship between the concepts of ontology "depends on", we develop *basic ontological models of nonfunctional software characteristics* (principles of developing basic ontological models are presented in [7]).

Then, for example, base ontological model of Functional Suitability is:

$$O_{Fs} = \{\{F\_S, A, I, J, K\}, \text{“depends on”}, \{f_1, \varphi_{1,} \varphi_{2,} \varphi_3\}\} =$$

$$=\{\{fsa_1,\ldots,fsa_{19}\}, \text{“depends on”}, \{f_1, \varphi_{1,} \varphi_{2,} \varphi_3\}\}, \quad (41)$$

where fsa_1 = F_S, $\{fsa_2,\ldots,fsa_4\} \in A$, $\{fsa_5,\ldots,fsa_8\} \in I$, $\{fsa_9,\ldots,fsa_{13}\} \in J$, $\{fsa_{14},\ldots,fsa_{19}\} \in K$.

Ontologies (ontological knowledge base), which are developed by models (41) etc., are filled up based on the information taken from ISO 25010 [4], ISO 25023 [5] standards.

The ontological models of concrete software nonfunctional characteristics have the form, for example, ontological model of Functional Suitability is:

$$O_{Fs\_real} = \{\{fsa_1,\ldots,fsa_m\}, \text{“depends on”}, \{f_1, \varphi_{1,} \varphi_{2,} \varphi_3\}\}, \quad (42)$$

where $m \leq 19$ – the number of attributes of subcharacteristics of functional suitability, available in the specification requirements for the specific software, as well as the number of functional suitability subcharacteristics that can be calculated based on available attributes.

Ontologies (ontological knowledge bases), which are developed on the models (42) etc., are filled up based on the information taken from the specification requirements for the specific software [3, 7].

Practical case of selection of accurate scope of information for evaluation of the nonfunctional characteristics is represented at [3, 7].

3 Conclusions

The urgency of the task of automation of evaluating the life cycle initial phases based on analysis of specifications necessitates the development of software quality models, in particular, the models of nonfunctional characteristics-software quality components.

This article proposed set-theoretic models of nonfunctional characteristics of software used to systematize information on nonfunctional characteristics and bring it to the common (unified) form according to ISO 25010: 2011, ISO 25023: 2016 standards, as well as basic (universal) ontological models and ontological models of specific software nonfunctional characteristics, based on the requirements of ISO 25010 and ISO 25023standards. These models are basis for selecting the sufficient volume of information to evaluate the software nonfunctional characteristics.

The proposed models will become the theoretical basis for the model and method of activity of the intelligence agent on the basis of the ontological approach for assessing the specifications of the requirements for software, the operation of which will be based on the comparison of ontologies. Such the intelligence agent will compare the ontology of the specific software with basic ontology and determine how many and which attributes are not sufficient to determine the particular nonfunctional characteristic, which will make the decision on the adequacy or lack of information on nonfunctional characteristics in the specification requirements to software.

References

1. McConnell, S.: Code complete. Microsoft Press, Redmond (2013).
2. Hastie, S., Wojewoda, S.: Standish Group 2015 Chaos Report – Q&A with Jennifer Lynch, http://www.infoq.com/articles/standish-chaos-2015, last accessed 2019/10/29.
3. Hovorushchenko, T., Pomorova, O.: Information technology of evaluating the sufficiency of information on quality in the software requirements specifications. CEUR-WS. 2104, 555-570 (2018).
4. ISO/IEC 25010:2011. Systems and software engineering. Systems and software Quality Requirements and Evaluation (SQuaRE). System and software quality models (2011).
5. ISO 25023:2016. Systems and software engineering. Systems and software Quality Requirements and Evaluation (SQuaRE). Measurement of system and software product quality (2016).
6. Sugiyanto, S., Rochiman, S.: Integration of DEMATEL and ANP methods for calculate the weight of characteristics software quality based model ISO 9126. In: Proccedings of the 5-th International Conference on Information Technology and Electrical Engineering. 143-148 (2013)
7. Hovorushchenko, T.: Information technology for assurance of veracity of quality information in the software requirements specification. Advances in Intelligent Systems and Computing II. 689, 166-185 (2018).
8. Burov, E. Complex ontology management using task models. International Journal of Knowledge-Based and Intelligent Engineering Systems. Vol 18, no 2, 111-120 (2014).
9. Shostak, I., Butenko, I. Ontology approach to realization of information technology for normative profile forming at critical software certification. In Herald of the Military Institute of Kiev National University named after Taras Shevchenko. № 38, 250–253 (2012).
10. Bajnaid, N., Benlamri, R., Pakstas, A., Salekzamankhani, Sh.: An Ontological Approach to Model Software Quality Assurance Knowledge Domain. Lecture Notes on Software Engineering. 4 (3), 193-198 (2016).