# Models and Methods of Evaluation of Information Sufficiency for Determining the Software Complexity and Quality Based on the Metric Analysis Results

Tetiana O. Hovorushchenko

***Abstract*—** The aim of this study is the development of the models and methods for evaluating the information sufficiency for determining the software complexity and quality based on the metric analysis results. In this paper, the models and methods based on the comparative analysis of base ontology of subject area and ontology of concrete software are developed. The developed models and methods provide the sorting of all indicators, that absent in the software requirements specification (SRS), in descending the weights values, i.e. to prioritize additions in SRS.

***Keywords*—**Information sufficiency, metric analysis, ontologies, software complexity, software quality

## I. INTRODUCTION

Analysis of [1-5] revealed the fact that the causes of many software incidents are rooted in the SRS. The software quality is the degree of satisfaction of users or the degree of compliance to customers' needs [6-8]. Then, if the project objectives at the early lifecycle stages don't meet the needs of users, the software will not have high quality. Therefore, the quality and success of software project implementation significantly depend on the SRS and on the sufficiency of information in it. The *sufficiency of information* is the rational information saturation that eliminates information incompleteness (lack of necessary information).

Today the evaluation of indicators for the software quality and complexity metrics is conducted only at the stage of the quality evaluation for the ready source code [5]. But the SRS have all indicators, which are needed to the metrics calculation [5]. So the information sufficiency (as presence in the SRS all necessary indicators for metrics calculation) for future definition of the software complexity and quality can be evaluated on the basis of the SRS. And if some indicators are absent, then the SRS has insufficient information for metrics calculation and the developers have to make the necessary adjustments in the SRS.

The evaluation of sufficiency of the SRS information (presence in the SRS all necessary indicators for metrics calculation) provides the choice of software project in terms of its predicted quality and complexity at the early lifecycle stages, increases efficiency of project management due the validity of decisions, reduces the time of decision-making, reduces the costs for collection and processing of information at the later lifecycle stages (for example, during the software quality audit stage). The insufficiency of SRS information reduces the effectiveness and veracity of evaluating the software quality and complexity.

The *actual task* is the evaluating the sufficiency of the SRS information - for example, the possibility of calculating the values of the metrics of the software complexity and quality based on available indicators in SRS. So *the aim of this study* is the development of the models and methods for evaluating the information sufficiency for determining the software complexity and quality based on the metric analysis results.

T. O. Hovorushchenko, Khmelnytskyi National University, Khmenlytskyi, Ukraine (e-mail: tat_yana@ukr.net).

## II. Formalized and Ontological Models of the Software Complexity and Quality Based on the Metric Analysis

During the analysis of software metrics as sources of information on its characteristics, the presence of cross-correlation of metrics was revealed because they have some joint indicators. The models of the software quality and complexity based on the metric analysis are necessary to develop for evaluating the correlation and the mutual influences of metrics and their indicators. In [9] it was proved that the software quality at the design stage ( $QDS$ ) depends on 14 metrics, and software complexity at the design stage( $CXDS$ ) depends on the 10 metrics with exact or predicted values: $QDS = \psi(sqm_1,...,sqm_{14})$ , $CXDS = \phi(scxm_1,...,scxm_{10})$ .

The set of software quality metrics at the design stage is: $SQM = \{Chp, Cpp, Rup, Mmt, Mbq, Sct, Sdt, Scc, Sqc, Cpt, Ccc, Fp, Lc, Dp\}$ , where $Chp$ – cohesion metric, $Cpp$ – coupling metric, $Rup$ – metric of the global variables calling, $Mmt$ – time of models modification, $Mbq$ – quantity of found bugs during the models inspection, $Sct$ – software design total time, $Sdt$ – design stage time, $Scc$ – software design expected cost, $Sqc$ – software quality audit expected cost, $Cpt$ – software realization productivity, $Ccc$ – code realization expected cost, $Fp$ – functional points, $Lc$ – effort applied by Boehm's model, $Dp$ – expected development time by Boehm's model.

The set of software quality metrics at the design stage can be presented in the form of $SQM = \{SQM_{exv}, SQM_{prv}\} = \begin{Bmatrix} \{Chp, Cpp, Rup, Mmt, Mbq\}, \\ \{Sct, Sdt, Scc, Sqc, Cpt, Ccc, Fp, Lc, Dp\} \end{Bmatrix}$, where $SQM_{exv}$ – subset of software quality metrics with the exact values at the design stage, $SQM_{prv}$ – subset of software quality metrics with the predicted values at the design stage.

The set of software complexity metrics at the design stage is: $SCXM = \{Is, N_{ZV}, MP, I, LOC_{оч}, HDiff, V(G), cl, N_{прогн}, Cmp\}$ , where $Is$ – Chepin's metric, $N_{ZV}$ – Jilb's metric (absolute), $MP$ – McClure's metric, $I$ – Kafur's metric, $LOC_{ep}$ – expected Lines Of Code, $HDiff$ – Halstead's metric, $V(G)$ – McCabe's metric, $cl$ – Jilb's metric (logical), $N_{ep}$ – expected quantity of program statements, $Cmp$ – expected estimate of interfaces complexity.

The set of software complexity metrics at the design stage can be presented in the form of $SCXM = \{SCXM_{exv}, SCXM_{prv}\} = \left\{\{Is, N_{ZV}, MP, I\}, \{LOC_{ep}, HDiff, V(G), cl, N_{ep}, Cmp\}\right\}$, where $SCXM_{exv}$ – subset of software complexity metrics with the exact values, $SCXM_{prv}$ – subset of software complexity metrics with the predicted values at the design stage.

Then the models of software quality and complexity on the basis of metric analysis:

$$QDS = \psi(Chp, Cpp, Rup, Mmt, Mbq, Sct, Sdt, Scc, Sqc, Cpt, Ccc, Fp, Lc, Dp), \quad (1)$$

$$CXDS = \phi(Is, N_{ZV}, MP, I, LOC_{ep}, HDiff, V(G), cl, N_{ep}, Cmp). \quad (2)$$

Each of these metrics is a function of several indicators, moreover, quality and complexity metrics depend on 72 indicators, but only on 42 different indicators, then set of indicators of the software quality and complexity for further metric analysis has the form $SQCXI = \{sqcxi_1,...,sqcxi_{42}\}$ .

The set of indicators of the software quality for further metric analysis has the form $SQI = \{sqi_1,...,sqi_{24}\}$ ($SQI \in SQCXI$), because the software quality metrics depend on 39 indicators, but only on 24 different indicators. The set of indicators of the software complexity for further metric analysis has the form $SCXI = \{scxi_1,...,scxi21\}$ ($SCXI \in SQCXI$),because the software complexity metrics depend on 33 indicators, but only on 21 different indicators (there are indicators that affect both quality metrics and complexity metrics, therefore they are both in the set $SQI$ and in the set $SCXI$).

The models of software quality metrics have the form:

$$Chp = \psi_1(Cam, Iaom), \tag{3}$$

where $Cam$ – cohesion of actions in module, $Iaom$ – importance of actions order in module;

$$Cpp = \psi_2(Tmid, Tmopd, Pcd, fp, gp), \tag{4}$$

where $Tmid$ – type of module input data, $Tmopd$ – type of module output data, $Pcd$ – presence of common data, $fp$ – quantity of preceding modules, $gp$ – quantity of following modules;

$$Rup = \psi_3(Aup, Pup) = \frac{Aup}{Pup}, \tag{5}$$

where $Aup$ – quantity of real access to global variables, $Pup$ – quantity of potential access to global variables;

$$Mmt = \psi_4(Qcl, Pd, Sdslc), \tag{6}$$

where $Qcl$ – quantity of code lines, $Pd$ – project duration, $Sdslc$ – share of design stage;

$$Mbq = \psi_5(Qbm, Q_m), \tag{7}$$

where $Qbm$ – quantity of bugs of module, $Q_m$ – quantity of modules;

$$Sct = \psi_6(Qcl, Pd); \tag{8}$$

$$Sdt = \psi_7(Qcl, Pd, Sdslc); \tag{9}$$

$$Scc = \psi_8(Qcl, Col) = Qcl \cdot Col, \tag{10}$$

where $Col$ – cost of one line;

$$Sqc = \psi_9(Svvtqlc, Sqavvtq, Qcl, Col), \tag{11}$$

where $Svvtqlc$ – share of VVTQ stage, $Sqavvtq$ – share of quality audit in VVTQ;

$$Cpt = \psi_{10}(Qcl, Pd); \tag{12}$$

$$Ccc = \psi_{11}(Qcl, Col, Srslc), \tag{13}$$

where $Srslc$ – share of realization stage in lifecycle;

$$Fp = \psi_{12}(EI, EO, EIN, ILF, ELF), \tag{14}$$

where $EI$ – quantity of external inputs, $EO$ – quantity of external outputs, $EIN$ – quantity of external requests, $ILF$ – quantity of internal logic files; $ELF$ – quantity of external logic files;

$$Lc = \psi_{13}(Qcl, Pt) = a \cdot Qcl^{b}, \tag{15}$$

where $Pt$ – project type, which determines the COCOMO coefficients $a$, $b$;

$$Dp = \psi_{14}(Qcl, Pt) = c \cdot a \cdot Qcl^{b \cdot d}, \tag{16}$$

where $Pt$ – project type, which determines the COCOMO coefficients $a$, $b$, $c$, $d$.

Thus, *the model of software quality based on the metric analysis (at the design stage)*:

$$QDS = \psi \begin{pmatrix} \psi_1(Cam, Iaom), \psi_2(Tmid, Tmopd, Pcd, fp, gp), \dfrac{Aup}{Pup}, \\ \psi_4(Qcl, Pd, Sdslc), \psi_5(Qbm, Q_m), \psi_6(Qcl, Pd), \\ \psi_7(Qcl, Pd, Sdslc), Qcl \cdot Col, \psi_9(Svvtqlc, Sqavvtq, Qcl, Col), \\ \psi_{10}(Qcl, Pd), \psi_{11}(Qcl, Col, Srslc), \\ \psi_{12}(EI, EO, EIN, ILF, ELF), a \cdot Qcl^{b}, c \cdot a \cdot Qcl^{b \cdot d} \end{pmatrix}. \tag{17}$$

As seen from formulas (3)-(16), some functions for calculating the software quality metrics are known (functions $\psi_3, \psi_8, \psi_{13}, \psi_{14}$), the remaining functions are uncertain.

The models of software complexity metrics have the form:

$$Is = \phi_1(P, M, C, T, Q_m) = Q_m \cdot (P + 2M + 3C + 0{,}5T), \tag{18}$$

where $P$ – quantity of variables for calculations and output, $M$ – quantity of modified or created variables, $C$ – quantity of control variables, $T$ – quantity of not used variables;

$$N_{ZV} = \phi_2(Q_m, Q_{lem}) = Q_m \cdot Q_{lem}, \tag{19}$$

where $Q_{lem}$ – quantity of links of each module;

$$MP = \phi_3(Q_m, fp, gp, X(Pm), Y(Pm)) = \sum_{i=1}^{Qm} (fp \cdot X(Pm) + gp \cdot Y(Pm)), \tag{20}$$

where $X(Pm)$ – quantity of calls to module $Pm$, $Y(Pm)$ – quantity of calls from module $Pm$;

$$I = \phi_4(Q_m, W, R, WrRd) = Q_m \cdot (W \cdot R + W \cdot WrRd + WrRd \cdot R + WrRd \cdot (WrRd - 1)), \quad (21)$$

where $W$ – quantity of procedures to update data structure, $R$ – quantity of procedures to read from data structure, $WrRd$ – quantity of procedures to read and update data structure;

$$LOC_{ep} = \phi_5(Qcl) = Qcl; \quad (22)$$

$$HDiff = \phi_6(Qcl, NUOprtr, NUOprnd, NOprtr, NOprnd) = \frac{NUOprtr}{2} \cdot \frac{NOprnd}{NUOprnd}, \quad (23)$$

where $NUOprtr$ – quantity of unique operators, $NUOprnd$ – quantity of unique operands, $NOprtr$ – total quantity of operators, $NOprnd$ – total quantity of operands (depend on $Qcl$);

$$V(G) = \phi_7(E, N, NOprtr) = E - N + 2, \quad (24)$$

where $E$ – quantity of control transfers, $N$ – quantity of computing operators and expressions (depend on total quantity of operators $NOprtr$);

$$cl = \phi_8(NOprtr, L_{IF}, L_{LOOP}) = \frac{L_{IF} + L_{LOOP}}{NOprtr}, \quad (25)$$

where $L_{IF}$ – quantity of logic operators, $L_{LOOP}$ – quantity of cycle operators;

$$N_{ep} = \phi_9(NOprtr, Qcl); \quad (26)$$

$$Cmp = \phi_{10}(NOprnd, NUOprnd, Q_m). \quad (27)$$

As seen from formulas (18)-(27), some functions for calculating the software complexity metrics are known (functions $\phi_1 - \phi_8$), the remaining functions are uncertain.

Thus, *the model of software complexity based on the metric analysis (at the design stage)*:

$$CXDS = \phi \begin{pmatrix} Q_m \cdot (P + 2M + 3C + 0{,}5T), Q_m \cdot Q_{lem}, \sum_{i=1}^{Qm} (fp \cdot X(Pm) + gp \cdot Y(Pm)), \\ Q_m \cdot (W \cdot R + W \cdot WrRd + WrRd \cdot R + WrRd \cdot (WrRd - 1)), Qcl, \\ (NUOprtr / 2) \cdot (NOprnd / NUOprnd), (E - N + 2), \frac{L_{IF} + L_{LOOP}}{NOprtr}, \\ \phi_9(NOprtr, Qcl), \phi_{10}(NOprnd, NUOprnd, Q_m) \end{pmatrix}. \quad (28)$$

The models of the software quality and complexity based on the metric analysis show that there are indicators, which affect more than one metric. Thus, there is the metrics correlation

by some indicators. The existence of relationships between metrics affect their significance and weight [10], therefore should identify joint indicators for the metrics and should determine the significance (probability) of the indicators with the purpose of improving the veracity of the evaluations of the software quality and complexity. The knowledge of experienced professionals about the mutual influences and correlation of metrics are valuable in identifying the joint indicators, so they should be stored and used. The ontologies were selected for this knowledge reflection and accumulation.

The ontological model of software quality based on the metric analysis has the form: $O_{Q_{metr}} =< X_{Q_{metr}}, RX_{Q_{metr}}, F_{Q_{metr}} >$, where $X_{Q_{metr}}$ – finite set of metrics and indicators of the software quality, $RX_{Q_{metr}}$ – finite set of relationships between concepts, $F_{Q_{metr}}$ – finite set of interpretation functions for the software quality metrics and indicators.

Considering the model of software quality based on the metric analysis, the set of metrics and indicators of the software quality is:

$$X_{Q_{metr}} = \{SQM, SQI\} = \{x_{Q_{met1}}, ..., x_{Q_{met38}}\}, \quad (29)$$

where $\{x_{Q_{met1}}, ..., x_{Q_{met14}}\} \in SQM$, i.e. $\{x_{Q_{met1}}, ..., x_{Q_{met14}}\} = \{sqm_1, ..., sqm_{14}\}$, $\{x_{Q_{met15}}, ..., x_{Q_{met38}}\} \in SQI$, then $\{x_{Q_{met15}}, ..., x_{Q_{met38}}\} = \{sqi_1, ..., sqi_{24}\}$.

The set of relationships between concepts $RX_{Q_{metr}}$ consists from relationship «depends on», i.e. $RX_{Q_{metr}} = \{"depends\ \ on"\}$. The set $F_{Q_{metr}}$ of interpretation functions for metrics and indicators of the software quality consists from function for quality depending on the metrics and functions for quality metrics depending on the indicators, i.e. $F_{Q_{metr}} = \{f_{Q_{met1}}, ...f_{Q_{met15}}\} = \{\psi(), \psi_1(), ..., \psi_{14}()\}$.

Thus *the base ontological model of the software quality based on the metric analysis*:

$$O_{Q_{metr}} = \{sqm_1, ..., sqm_{14}, sqi_1, ...sqi_{24}, "depends\ \ on", \psi(), \psi_1(), ..., \psi_{14}()\}. \quad (30)$$

*The ontological model of the concrete software quality based on the metric analysis*:

$$O_{Q_{metr\_real}} = \{sqm_1, ..., sqm_{nqm}, sqi_1, ...sqi_{nqi}, "depends\ \ on", \psi(), \psi_1(), ..., \psi_{14}()\}, \quad (31)$$

where $nqm$ ($nqm \le 14$) – quantity of software quality metrics, which can be calculated on the basis of the available indicators in the SRS of concrete software, $nqi$ ($nqi \le 24$) – quantity of quality indicators, which are available in the SRS of concrete software.

The ontological model of software complexity based on the metric analysis has the form: $O_{CX_{metr}} =< X_{CX_{metr}}, RX_{CX_{metr}}, F_{CX_{metr}} >$, where $X_{CX_{metr}}$ – finite set of metrics and indicators of the software complexity, $RX_{CX_{metr}}$ – set of relationships between concepts, $F_{CX_{metr}}$ – set of interpretation functions for the software complexity metrics and indicators.

Considering the model of software quality based on the metric analysis, the set of metrics and indicators of the software complexity is:

$$X_{CX_{metr}} = \{SCXM, SCXI\} = \{x_{CX_{metr1}}, ..., x_{CX_{metr31}}\}, \quad (32)$$

where $\{x_{CX_{metr1}}, ..., x_{CX_{metr10}}\} \in SCXM$, i.e. $\{x_{CX_{metr1}}, ..., x_{CX_{metr10}}\} = \{scxm_1, ..., scxm_{10}\}$, $\{x_{CX_{metr11}}, ..., x_{CX_{metr31}}\} \in SCXI$, then $\{x_{CX_{metr11}}, ..., x_{Q_{metr31}}\} = \{scxi_1, ..., scxi_{21}\}$.

The set of relationships between concepts $RX_{CX_{metr}}$ consists from relationship «depends on», i.e. $RX_{CX_{metr}} = \{"depends\ \ on"\}$. The set $F_{CX_{metr}}$ of interpretation functions for metrics and indicators of the software complexity consists from function for complexity depending on the metrics and functions for complexity metrics depending on the indicators, i.e. $F_{CX_{metr}} = \{f_{CX_{metr1}}, ...f_{CX_{metr11}}\} = \{\phi(), \phi_1(), ..., \phi_{10}()\}$.

Thus *the base ontological model of the software complexity based on the metric analysis*:

$$O_{CX_{metr}} = \{scxm_1, ..., scxm_{10}, scxi_1, ...scxi_{21}, "depends\ \ on", \phi(), \phi_1(), ..., \phi_{10}()\}. \quad (33)$$

*The ontological model of the concrete software quality based on the metric analysis*:

$$O_{CX_{metr_{real}}} = \{scxm_1, ..., scxm_{ncxm}, scxi_1, ...scxi_{ncxi}, "depends\ \ on", \phi(), \phi_1(), ..., \phi_{10}()\}, \quad (34)$$

where $ncxm$ ($ncxm \le 10$) – quantity of software complexity metrics, which can be calculated on the basis of the available indicators in the SRS of concrete software, $ncxi$ ($ncxi \le 21$) – quantity of complexity indicators, which are available in the SRS of concrete software.

## III. Formalized and Ontological Models of Software Requirements Specification (in Terms of the Availability of Indicators for Software Metrics Calculation)

Considering the SRS structure according to ISO 29148 [11], the SRS can be represented in the following formalized form (in terms of the availability in it of indicators for software quality and complexity metrics calculation):

$$SRS_{metr} = < R1_{metr}, R2_{metr}, R3_{metr}, R4_{metr}, R5_{metr} >, \quad (35)$$

where $R1_{metr}$ – set of complexity and quality indiactors of section 1 of SRS, $R2_{metr}$ – set of indiactors of section 2 of SRS, $R3_{metr}$ – set of indiactors of section 3 of SRS, $R4_{metr}$ – set of indiactors of section 4 of SRS, $R5_{metr}$ – set of indiactors of section 5 of SRS.

Some indicators may be contained in section 1 "Introduction" of the SRS, some indicators may be contained in section 3 "Specific requirements", some indicators may be contained in section 5 "Supporting information" of the SRS.

The ontological model of the SRS (in terms of the availability of indicators for software complexity and quality metrics calculation) has the form: $O_{SRS_{metr}} = < X_{SRS_{metr}}, RX_{SRS_{metr}} >$, where $X_{SRS_{metr}}$ – finite set of the software complexity and quality indicators in the SRS, $RX_{SRS_{metr}}$ – finite set of relationships between concepts.

Thus *the model of the SRS (in terms of the availability of indicators for software complexity and quality metrics calculation)* has the form:

$$SRS_{metr_i} = \left\langle \begin{array}{l} \{Pup, Sdslc, Svvtqlc, Sqavvtq, Srslc, Pt\}, \\ \varnothing, \\ \left\{ \begin{array}{l} Cam, Iaom, Tmid, Tmopd, Pcd, fp, gp, Aup, Q_m, EI, EO, EIN, \\ ILF, ELF, P, M, C, T, Q_{lem}, X(Pm), Y(Pm), W, R, WrRd \end{array} \right\}, \\ \varnothing, \\ \left\{ \begin{array}{l} Qcl, Pd, Qbm, Col, NUOprtr, NUOprnd, NOprtr, NOprnd, N, \\ L_{IF}, L_{LOOP} \end{array} \right\} \end{array} \right\rangle. \quad (36)$$

Considering the model of the SRS, the set of the indicators:

$$X_{SRS_{metr}} = \{SRS_{metr}, SQCXI\} = \{x_{SRS_{metr_1}}, ..., x_{SRS_{metr_{47}}}\}, \quad (37)$$

where $\{x_{SRS_{metr_1}}, ..., x_{SRS_{metr_5}}\} = \{R1_{metr}, ...R5_{metr}\}$, $\{x_{SRS_{metr_6}}, ..., x_{SRS_{metr_{47}}}\} = \{sqcxi_1, ..., sqcxi_{42}\}$. The set of relationships between concepts $RX_{SRS_{metr}}$ consists from relationship «contained in», i.e. $RX_{SRS_{metr}} = \{"contained\ \ in"\}$.

Thus *the base ontological model of the SRS (in terms of the availability of indicators for software complexity and quality metrics calculation)* has the form:

$$O_{SRS_{metr}} = \{R1_{metr}, ..., R5_{metr}, sqcxi_1, ...sqcxi_{42}, "contained\ \ in"\}. \quad (38)$$

*The ontological model of the SRS of concrete software (in terms of the availability of indicators for software complexity and quality metrics calculation)* has the form:

$$O_{SRS_{metr_{real}}} = \left\{R1_{metr}, ..., R5_{metr}, sqcxi_1, ...sqcxi_{ni}, "contained\ \ in"\right\}, \quad (39)$$

where $ni$ ($ni \le 42$) – quantity of complexity and quality indicators, which are available in the SRS of concrete software.

## IV. Ontological Methods of Evaluation of Information Sufficiency for Determining the Software Complexity and Quality Based on the Metric Analysis Results

Foremost, *the ontological method of evaluation of information sufficiency for determining the software complexity and quality based on the metric analysis results* was developed [12]. The base ontology for the subject domain "Software Engineering" (part "The software quality and complexity. Metric Analysis") was developed in [12]. For this ontology 4 software characteristics were selected: software project complexity, software complexity, software project quality, software quality. These characteristics are calculated on the basis of metrics, which in turn are based on indicators, according to the above models. The concept of the base ontology for the subject domain "Software Engineering" is shown on Figure 1.

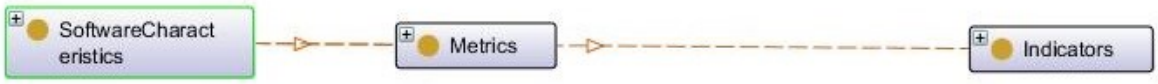


Fig. 1 Concept of the base ontology for the subject domain "Software Engineering" (part "The software quality and complexity. Metric Analysis")

In evaluating the software complexity and quality should focus on those indicators that are part of multiple metrics simultaneously. By analogy with the method of evaluation of weights of software quality measures [13] let's evaluate the weights of SRS indicators, which are necessary for metrics calculation.

For evaluation of the weight of $g$-th indicator let's use the next formula:

$$\omega_{m_g} = \frac{k_{metr_{ind_g}}}{k_{ind}}, \tag{40}$$

where $k_{metr_{ind_g}}$ – quantity of metrics, which depend on $g$-th indicator; $k_{ind}$ – total quantity of indicators (analysis of the above models showed that nowadays metrics of complexity and quality depend on 72 indicators, but on 42 different indicators, i. e. today $k_{ind} = 42$).

The developed models of the software complexity and quality, the base ontology for the subject domain "Software Engineering" (part "The software quality and complexity. Metric Analysis") provide the conclusions about the indicators, which are used for calculation of more than one metric, and about the quantities of metrics, which depend on each indicator (numerator of weights) [12]. In evaluating the quality and complexity metrics it's important to satisfy the availability in the SRS of those indicators, which have larger weights, with the purpose of providing the appropriate level of evaluations veracity.

*The weighted ontology* of the subject domain "Software Engineering" (part "The software quality and complexity. Metric Analysis") will be called the ontology, in which the complexity and quality indicators have the weights with the purpose of recommendations about the further satisfaction of these indicators in the SRS.

*The method of evaluation of information sufficiency for determining the software complexity and quality based on the metric analysis results using the weighted ontology* consists from next stages:

1) development of the weighted base ontology for the subject domain "Software Engineering" (part "The software quality and complexity. Metric Analysis");

2) analysis of the sections of SRS of concrete software for the availability of the indicators, which are necessary for metrics calculation, i.e. for the availability of the elements of set $X_{SRS_{metr}} = \{SRS_{metr}, SQCXI\} = \{x_{SRS_{metr_1}}, ..., x_{SRS_{metr_{47}}}\}$; generation and filling the template of ontology for concrete software, i.e. generation and filling the template of ontology $O_{SRS_{metr_{real}}} = \left\{R1_{metr}, ..., R5_{metr}, sqcxi_1, ...sqcxi_{ni}, "contained \quad in"\right\}$;

3) comparing the developed weighed ontology for concrete software with the weighted ontology of the subject domain "Software Engineering" (part "The software quality and complexity. Metric Analysis"), i.e. comparing the set of indicators $\{sqcxi_1, ..., sqcxi_{ni}\}$ from ontological model of the SRS of concrete software $O_{SRS_{metr_{real}}} = \left\{R1_{metr}, ..., R5_{metr}, sqcxi_1, ...sqcxi_{ni}, "contained \quad in"\right\}$ with the appropriate sets

$\{sqi_1,...,sqi_{24}\}$, $\{scxi_1,...,scxi_{21}\}$ of the base ontological models of the software quality and complexity based on the metric analysis $O_{Q_{metr}} = \{sqm_1,...,sqm_{14}, sqi_1,...sqi_{24}, "depends\ \ on", \psi(), \psi_1(),..., \psi_{14}()\}$ and $O_{CX_{metr}} = \{scxm_1,...,scxm_{10}, scxi_1,...scxi_{21}, "depends\ \ on", \phi(), \phi_1(),..., \phi_{10}()\}$;

4) identifying the indicators, which are absent in the weighed ontology for concrete software, i.e. forming set $\{sqcxi_1,...,sqcxi_{(42-ni)}\} = \{sqcxi_1,...,sqcxi_{42}\} \setminus \{swcxi_1,...,sqcxi_{ni}\}$, where $\{sqcxi_1,...,sqcxi_{42}\} \in (O_{Q_{metr}} \cup O_{CX_{metr}})$, $\{sqcxi_1,...,sqcxi_{ni}\} \in O_{SRS_{metr_{real}}}$ (if these sets are not empty, then SRS information is not sufficient for calculating the metrics of software complexity and quality; the more elements are in these sets, the smaller sufficiency of SRS information is); sorting of the missing indicators in descending the values of weights; herewith the numerator of the weight of each missing indicator indicates the number of software metrics that cannot be calculated without this indicator;

5) identifying the metrics, which cannot be calculated on the basis of available indicators;

6) identifying the software characteristics, which cannot be calculated on the basis of the metrics, which can be calculated on the basis of available indicators;

7) making the decision on the need to supplement of the SRS by the indicators, if there are metrics and characteristics whose values can not be determined based on available indicators; herewith the indicators with larger weights (the first in the sorted list of missing indicators) should be added in the SRS first of all;

8) repeating the stages 2-7 until it will be possible to identify all the metrics and software characteristics, or until forming the conclusion about insufficient data for determining the software complexity and quality with high veracity degree.

On the basis of the base ontology of the subject domain "Software Engineering" (part "The software quality and complexity. Metric Analysis"), which is represented in [12], let's develop the weighted base ontology for the subject domain "Software Engineering" (part "The software quality and complexity. Metric Analysis"). In this weighted ontology, there is information about the weights of the SRS indicators, which are necessary for the metrics calculation. The parts of this weighted base ontology are: the weighted base ontology for the software project complexity (Figure 2), the weighted base ontology for the software complexity, the weighted base ontology for the software project quality, the weighted base ontology for the software quality (all these ontologies are similar to ontology on Figure 2 and are developed according to above models).

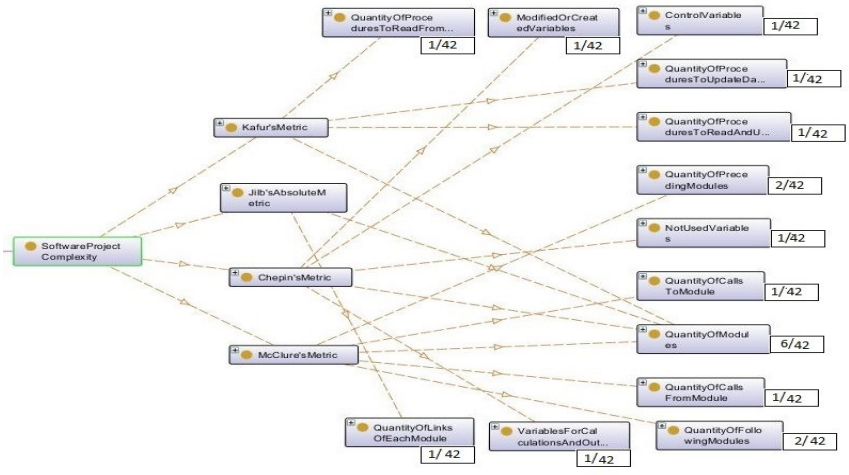


Fig. 2 The weighted base ontology for the software project complexity

Marking the weights of complexity and quality indicators in the weighted base ontology provides the sorting all the missing in the SRS indicators in descending values of weights, i.e. prioritizes their additions in the SRS.

## V. Experiments

The SRS of the automated system for large-format photo print was analyzed, on the basis of this SRS the ontology for the concrete software was developed.

On Figure 3 the ontology for the complexity of concrete software project (the part of the ontology for the concrete software project) is presented. The metrics that cannot be calculated on the basis of the available in the SRS indicators are circled in Figure 3.

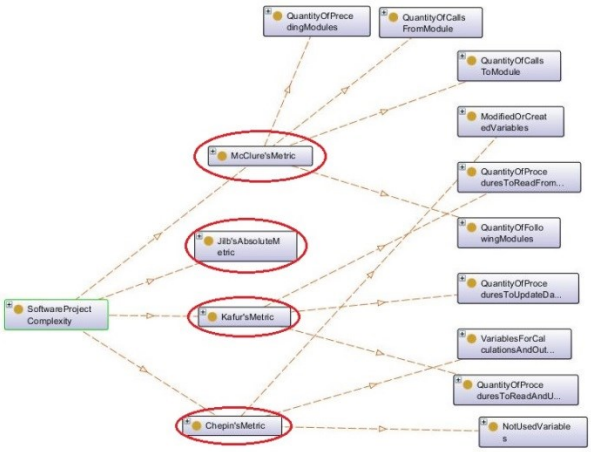


Fig. 3 The ontology for the complexity of concrete software project

Comparative analysis of the developed ontology for the automated system for large-format photo print with the base ontology of the subject domain "Software Engineering" (part "The software quality and complexity. Metric Analysis") provides the conclusion that 9 (from 42) indicators are absent in the developed ontology for the concrete software project, i.e. *the SRS information is insufficient for software metrics calculation* (20 metrics from 24 cannot be calculated). For example, the compare Fig. 2 and Fig. 3 provides the conclusion that: the SRS information is insufficient for calculating the Chepin's metric, McClure's metric, Kafur's metric, and in the SRS information for calculating the Jilb's metric (absolute) is at all absent.

The sorted list of the missing indicators in descending the values of weights is: 1) quantity of code lines – 12/42; 2) quantity of modules – 6/42; 3) project duration – 4/42; 4) total quantity of operators – 4/42; 5) cost of one line – 3/42; 6) project type – 2/42; 7) share of design stage in lifecycle – 2/42; 8) quantity of control variables – 1/42; 9) quantity of links of each module – 1/42. This sorted list indicates the priority of indicators and the consistency of their review and addition in the SRS.

## VI. Conclusion

The metric analysis is an effective mean of evaluating the software complexity and quality on condition of the availability of sufficient information for this. One of the factors, which affect to the veracity of such information, is the sufficiency of the information in the SRS regarding the indicators for metrics calculation. So the development of models and methods of evaluation of information sufficiency for determining the software complexity and quality, in general, enhances the veracity of evaluates of the software complexity and quality.

The developed model of the software quality and complexity based on the metric analysis, formalized and ontological model of the SRS (in terms of the availability of indicators for

software complexity and quality metrics calculation) became the basis for the development of the ontological methodology of complex evaluation of the software quality and complexity.

The analysis of the software metrics as sources of information on its characteristics, revealed the cross-correlation of these metrics because they have some joint indicators. The ontologies were selected for the reflection and accumulation of the knowledge of experienced professionals about the mutual influences and correlation of metrics.The ontologies became the basis of the ontological method of evaluation of information sufficiency for determining the software complexity and quality based on the metric analysis results. The correlation of metrics on indicators, that is displayed in the base ontology, taken into account in evaluating the weights of indicators. The lack of indicators, for which there is the correlation, can impair the accuracy and veracity of evaluations of the software complexity and quality. The correlation of metrics on some indicators increases the importance of these indicators in evaluating the software complexity and quality, thus increases the weights of indicators.

The developed method of evaluation of information sufficiency for determining the software complexity and quality based on the metric analysis results using the weighted ontology provides the conclusion about the insufficiency of the SRS information for metrics calculation, the sorting missing in the SRS indicators, prioritization their addition to the SRS.

## REFERENCES


[1] C. Jones, O.Bonsignour, *The Economics of Software Quality*. Boston: Pearson Education, 2012, 588 p.

[2] T. Ishimatsu, N.G. Levenson, J. P. Thomas, C. H. Fleming, M. Katahira, Yu. Miyamoto, R. Ujiie, H. Nakao, N. Hoshino, "Hazard Analysis of Complex Spacecraft Using Systems-Theoretic Process Analysis," *Journal of Spacecraft and Rockets*, vol. 51, no. 2, pp. 509-522, 2014.

[3] E. Yourdon, *Death March: The Complete Software Developer's Guide to Surviving "Mission Impossible" Projects*. Prentice Hall, 2003, 256 p.

[4] C. Shamieh, *Systems Engineering for Dummies*. Wiley Publishing, 2011, 74 p.

[5] D. Maevskiy, Y. Kozina, "Where and When Is Formed of Software Quality?," *Electrical and Computer Systems*, no. 18, pp. 55-59, 2016. *(in Russian)*

[6] ISO/IEC 25010:2011. Systems and software engineering. Systems and software Quality Requirements and Evaluation (SQuaRE). System and software quality models. ISO/IEC, 2011, 34 p.

[7] ISO/IEC 25030:2007. Software engineering. Software product Quality Requirements and Evaluation (SQuaRE). Quality requirements. ISO/IEC, 2007, 36 p.

[8] ISO/IEC TR 19759:2015. Software Engineering. Guide to the software engineering body of knowledge (SWEBOK). ISO/IEC, 2015, 336 p.

[9] O. Pomorova, T. Hovorushchenko, "The Intelligent Decision Support System for Choice of Software Project," *Journal of Information, Control and Management Systems,* vol. 10, no.1, pp.87-96, 2012.

[10] S. Sugiyanto, S. Rochiman, "Integration of DEMATEL and ANP methods for calculate the weight of characteristics software quality based model ISO 9126," in *International Conference on Information Technology and Electrical Engineering,* Yogyakarta (Indonesia), 2013, pp.143-148.

[11] ISO/IEC/IEEE 29148-2011. Systems and software engineering. Life cycle processes. Requirements engineering. ISO/IEC/IEEE, 2011, 28 p.

[12] T. Hovorushchenko, O. Pomorova, "Method of Assessment of Information Sufficiency for Determination of Software Complexity and Quality Based on The Comparative Analysis of Ontologies," *Radioelectronic and Computer Systems*, no.6, pp. 59-68, 2016. *(in Ukrainian)*

[13] T. Hovorushchenko, O. Pomorova, "Evaluation of Mutual Influences of Software Quality Characteristics Based ISO 25010:2011," in *XI International Scientific Conference on Computer Sciences and Information Technologies*, Lviv (Ukraine), 2016, pp.80-83.