An Ontological Approach to Analyze the Data Required by a System Quality Scheme

María Julia Blas, Silvio Gonnet

INGAR, Instituto de Desarrollo y Diseño UTN-CONICET, Santa Fe, Argentina
{mariajuliablas,sgonnet}@santafe-conicet.gov.ar

Abstract. According to IEEE, software quality is the degree to which software possesses a desired combination of attributes. Quality attributes have been of interest to the software community since 1970 and in the past few years this interest has increase. However, still is not clear how product quality should apply in the development process. The need to know what quality characteristics influence a specific entity and which is the information required to cover its measurement is the objective of this work. This paper presents an ontology to document a quality scheme with a specification based on ISO/IEC 25010. This contribution includes rules and queries that help to determine the data required to carry out the measurement process.

Keywords: Ontology, quality scheme, software metrics, software product.

1 Introduction

The delimitation of what defines an adequate level of quality in a software system is a highly context dependent question [1]. This problem changes with the product and perspective of the stakeholders and, therefore, software product quality can easily become an area full of problems and conflicts as long each stakeholder group has its own perspective on what is important.

Everyone involved in the software engineering process is responsible for quality [2]. Achieving quality attributes must be considered throughout design, implementation, and deployment [3]. However, maintain the traceability of this attributes in the development process is very difficult. Frequently, the attributes get lost between different stages because there is no mechanism that supports the quality decisions made in the previous phases. Or worse, the development team does not know which attributes are related and how they impact in the overall quality of the software product (SP). In this context, software community needs a mechanism that not only provides the basic information related to quality but also leads to the correct definition of a quality scheme (QS). A QS can be defined as a set of triplets where each element is composed by one software attribute (that is, a part of a software entity that requires some specific quality property), one software metric (that should be used to measure the quality of the attribute) and one quality subcharacteristic (that should be evaluated over the attribute). The application of a QS over a SP can be done by executing the measurement process over the specified artifact. However, before this application is necessary to collect all the required information. Given that this information may not be available is useful to analyze how a subset of data covers the required metrics. To this purpose, this paper presents an ontological approach that allows developers to document a QS and helps to analyze its coverage (taking into account the availability of the information). This proposal defines three basic elements: i) a quality scheme ontology (QSO) that establishes a way to define an adequate QS; ii) a set of SWRL (Semantic Web Rule Language) rules [4] and SPARQL (SPARQL Protocol and RDF Query Language) queries [5] that allows developers to know the degree in which the available information covers the scheme; and iii) an interactive activity which indicates how to use the previous elements. The QSO proposed is based on three semantic models: software product quality semantic model, metric semantic model and software semantic model. Although a lot of authors have proposed ontologies for quality models and software metrics [6-8]. These ontologies are very detailed representation of a specific domain and their combination is a complex task. Furthermore, the existing quality model ontologies are outdated because the current normative [9] is relatively new. Therefore, none of these ontologies is valid in the actual context.

The remainder of this paper is organized as follows. Section 2 describes the QSO that should be used to define a QS for a specific SP. Section 3 explains the set of rules and queries developed in order to derivate the coverage degree. Section 4 presents the activity defined to apply the ontological approach. Finally, Section 5 is devoted to the conclusions of the work.

2 Quality Scheme Ontology

The QSO developed is a domain ontology, since is applicable to a domain with a specific view point [10]. This ontology is based on the combination of three semantic models that represent specifics domains:

1. A *software product quality semantic model* that represents a product quality model.
2. A *metric semantic model* that represents concepts related to software metrics.
3. A *software semantic model* that represents the content of a software product.

A quality model is an essential component of a QS since is a model with the objective to describe, assess and/or predict quality [11]. Software quality models are a well-accepted means to support quality management of software systems. Furthermore, the use of metrics to develop strategies for improving the quality of the end product is a good practice [2]. A software metric is a measure of some property of a piece of software code or its specifications [12]. Software quality metrics are useful to register the current quality state of an end-product or process. To define a correct QS is necessary to specify which artifacts of the software product have to be evaluated.

The proposed QSO unifies the three individual domains in a single model. An attribute of a SP needs to be described by a quality subcharacteristic and has to be measured by a metric. In this sense, the three elements define one specification of the set of triples included at QS. Therefore, for each quality subcharacteristic, the capability of a SP is determined by a set of internal attributes that can be measured.

Figure 1 shows the QSO. The gray nodes represent the concepts while the arrows refer to the relationships. The arrow head indicates the direction of the relationship and its label indicates the name. The empty arrows model "is-a" relationship, disjoint and complete. The relationships highlighted in yellow represent links between the different models. The boxes model the data properties of the concepts.

The upper part of Figure 1 shows the *Software Product Quality Semantic Model*. Many quality models have been proposed to support stakeholders in dealing with software quality. By novelty and completeness, the SP quality model presented in ISO/IEC 25010 [9] is the most rigorous and complete of all. For these reasons, this model has been taken as basis for this work.

The product quality model identifies the main characteristics of a SP in different levels of hierarchy. It is composed of eight characteristics which are further subdivided into subcharacteristics. A characteristic represents an external quality view while a subcharacteristic refers to properties that can be evaluated when the software is used as a part of a system. Each subcharacteristic is decomposed into a set of attributes. An attribute is an entity which can be verified or measured in the SP. A detail description of these concepts and relationships can be checked in [9].

In order to build this ontology, each characteristic and subcharacteristic identified in the quality model was transformed in a concept. The relationships between these concepts were modeled defining links between them. This definition took the form *is-decomposed-in*. For example, the characteristic *Functional Suitability* is related with the subcharacteristic *Functional Appropriateness* by the relationship called *is-decomposed-in-functional-appropriateness*. To group these concepts in different categories, the concepts *Characteristic* and *Subcharacteristic* were included. This classification was made linking the concepts by an *is-a* relationship. This relationship allows to modeling the taxonomy proposed in ISO/IEC 25010 [9]. The *Quality Model* concept was also included in the semantic model. The relationship of this concept with each characteristic was made by defining a new relationship that took the name *contains* (e.g. *contains-security*). The *includes* relationship helps to determine which characteristic are included in a *Quality Model* (since none of the characteristic is mandatory). Once the main concepts were defined, a set of properties was included in the model with the aim to refine the represented semantic. These properties include: *characteristic description*, *subcharacteristic description* and *source*.

To complete the definition of the ontology, a set of SWRL rules related with the *contains* relationship were specified. Equation 1 describes an example restriction that shows that all security characteristic decomposed in an integrity subcharacteristic must contain this subcharacteristic. Similar restrictions were added to the model to guarantee that all characteristics contain the appropriate set of subcharacteristics.

$$\text{Security}(?x) \wedge \text{isDecomposedInIntegrity}(?x,?y) \rightarrow \text{contains}(?x,?y) \qquad (1)$$

The bottom part of Figure 1 shows the *Metric Semantic Model*. The metric ontology focuses in the traditional metric definition [2, 13, 14] and incorporates some concepts of the current normative [15, 16]. According to ISO 9126, a metric is basically defined by the specification of its name, purpose, application method, measurement formula, interpretation, scale type, measure type, input to measurement and target audience. The identification of the metric is given by its name. For this reason, the metric name must be related with the information obtained when the metric is applied. The purpose of the metric is expressed as the question to be answered by the application of the metric. The metric application method provides an outline of application while the measurement formula stipulates the mathematical expression used for the calculation and explains the meanings of the used data elements. The interpretation of the measured value supplies the range and preferred values. The scale type defines the dimension of the metric. Scale types used are: nominal scale, ordinal scale, interval scale, ratio scale and absolute scale. The measure type involves the specification of the way in which the metric is obtained. Types used are: size, time and count type. The input to the measurement process refers to the source of data used in the measurement. Finally, the target audience identifies the users of the measurement results.

Although all this concepts define a metric, only a set of them was used as part of the metric ontology. The selection was made taking into account the final objective: document the metrics related with different quality attributes that should be used to evaluate a specific SP. In order to do this, the input to measurement and the target audience are not represented in the model. To define a QS, the specific source of information for a metric is not important since the measurement process is not executed. The target audience is also a concept that shows a dependency with the use of the metric and, therefore, is excluded of the model. The rest of the concepts were taken to compose the semantic model. Furthermore, in order to improve and complete the ontology some properties were refined. The first refinement was the incorporation of the concepts *Direct Metric* and *Indirect Metric*. A *Direct Metric* is an atomic metrics and an *Indirect Metric* is defined in term of another metrics. To define a metric three complementary changes were made: the addition of the hierarchies *Unit* and *Scale* and the incorporation of the *Equation* concept. The *Unit* hierarchy was designed using as reference the proposal of Rijgersberg [17] while the *Scale* hierarchy was modeled using the approach described by Olsina [8]. In order to include an approach that allow to describing how a metric should be calculated, the *Equation* concept was added to the model. This concept represents a mathematical formula composed of mathematical terms, operators and variables.

The derivation of knowledge that refers to specific types of operations, units and range delimitations is made by mean of SWRL rules. Equation 2 shows as example that a unit must be assigned into a metric only if its scale is numerical, then the unit will be the one specified in the scale.

$$\text{Metric}(?m) \land \text{NumericalScale}(?s) \land \text{isDimensionedIn}(?m,?s) \land \text{isMeasuredIn}(?s,?u) \land \text{Unit}(?u) \rightarrow \text{hasAsUnit}(?m,?u) \quad (2)$$

The middle part of Figure 1 shows the *Software Semantic Model*. Its purpose is to model the domain of SPs. A SP is a set of computer programs, procedures, and possibly associated documentation and data, designed for delivery to a specific user. It always includes the development of one or more computer programs.

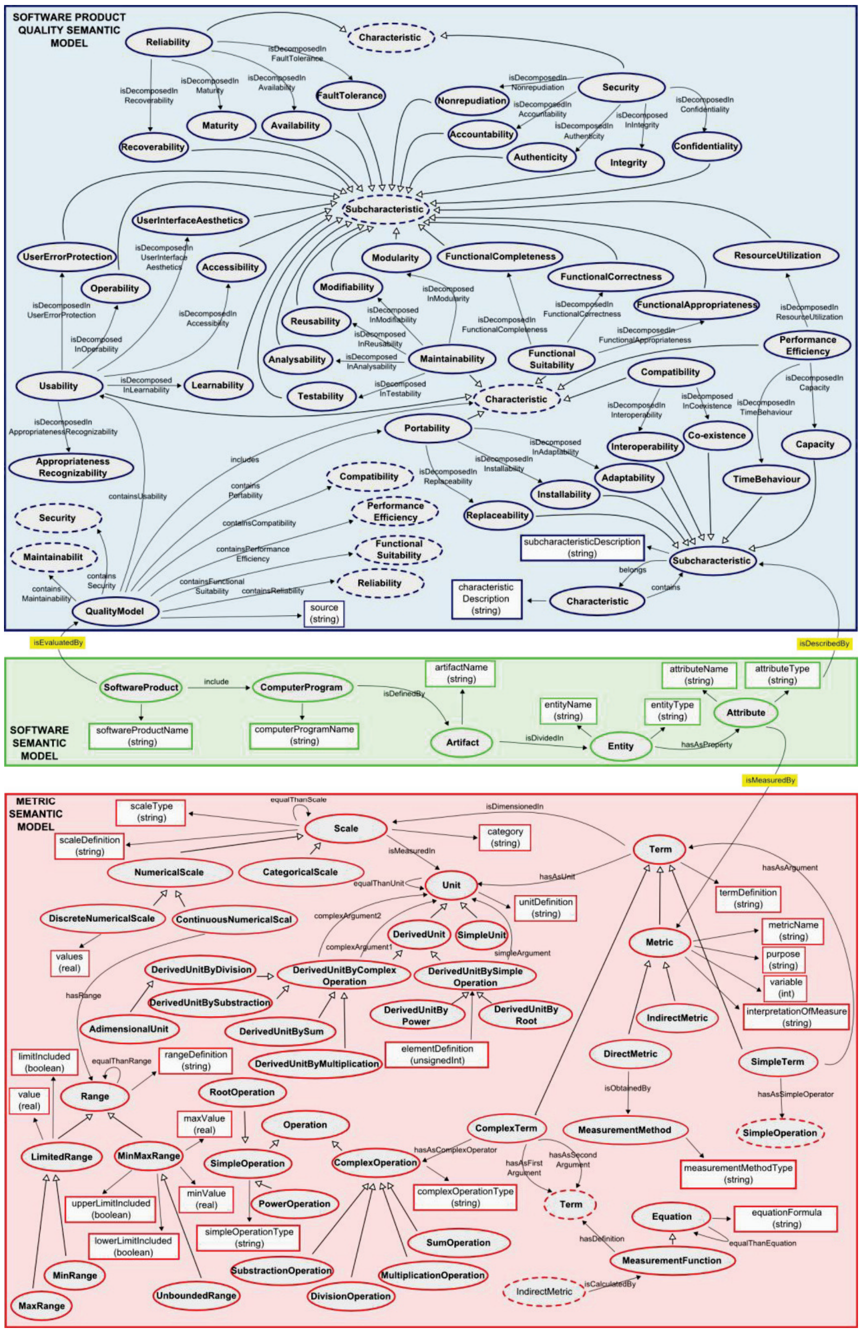

Fig. 1. Quality Scheme Ontology.

The development process carried out for the construction of each computer program usually involves the creation of different artifacts. An artifact is a product produced during the development of software that contains information of some part of it. All artifacts can be divided in a set of entities. An entity is an object that can be characterized by measuring its attributes. Therefore, an attribute is a measurable physical or abstract property of an entity.

The semantic model developed is based on this description which is adapted from [2,15,16,18]. Each of the main concepts identified in the domain was transformed into an ontology concept. The links between the concepts were modeled as relationships. The name of these relationships describes the way in which the concepts are related (e.g. *is-divided-in*). To allow a correct identification of the instances derived from a concept, the model includes some attributes labeled as *name* and *type*.

The designed ontologies and the proposed SWRL rules were implemented using Protégé (http://protege.stanford.edu/). Each ontology was implemented in an individual OWL (Web Ontology Language) file in order to increase the possibility of reuse it in other contexts. The elements were specified in English and Spanish to allow multiple language support. The OWL files were imported in a new document with aim to model the final ontology. This new model incorporates the specification of the relationships that link the three semantic models.

Given that it is difficult to quantify the quality of ontologies due to the absence of formal evaluation methods, an analysis of the structural dimensions was applied over the final ontology. Table 1 describes the set of selected metrics and its results.

Table 1. Metrics applied to the Quality Scheme Ontology.

Abbrev.	*Description*	*Definition*	*Value*
P	Number of non-inheritance relationships.	-	68
H	Number of inheritance relationships.	-	72
NOC	Number of classes.	-	87
NOR	Number of relations.	-	140
NORC	Number of root classes.	-	15
NOLC	Number of leaf classes.	-	72
NAT	Number of attributes for all classes.	-	32
RR	Relationship richness.	P / (H+P)	0.485
IR	Inheritance richness.	H / NOC	0.827
DOSH	Depth of subsumption hierarchy.	-	5
AR	Attribute richness.	NAT/NOC	0.367

The final ontology has an IR of 0.827 which together with the DOSH value (5) indicate that the proposed ontology is of a vertical nature. This means that represents knowledge of a specific domain, allowing to instantiate schemes that fit to the quality specification. Furthermore, the AR value (0.367 attributes per concept) shows that the attributes allow to restricting the domain. The RR is very close to the average (48.5%) which implies that the number of hierarchical relationships is a bit greater than the number of the other kind of associations. In this sense, the ontology maintains an adequate balance between inheritance relations and associations.

3 Analysis of the Required and Available Information

3.1 SWRL Rules Applied to the Derivation of Knowledge

In order to analyze a software QS, a set of SWRL rules was specified (Table 2). The SWRL is a language that is used to express rules in the form of an implication between an antecedent (body) and consequent (head) [4] and, therefore, allows to derivate new knowledge from the instances of a model.

Since all direct metric refers to a measure or is used as component in an indirect metric, the first rule specifies that a *Direct Metric* implies a need of information (*Required Data*). However, if a required data is available it establishes an *Available Data* (rule 2). In this sense, if a *Direct Metric* is also an *Available Data* (that is, the information exists when the measurement process take place) then the metric is a mathematical term than can be calculated (*Calculable Term*). This fact is expressed in rule 3. But, the fact that a mathematical term can be calculated it may involve that other mathematical terms can be calculated. To this purpose, rules 4 and 5 are defined. In rule 4, if a *Simple Term* has as argument a mathematical term that is calculable then it is a *Calculable Term*. The same happens with *Complex Terms* (rule 5). Then, if the mathematical term that resumes the *Measurement Function* associated with an *Indirect Metric* is calculable, the metric is itself a *Calculable Term* (rule 6). Finally, rule 7 specifies that if a *Metric* is a *Calculable Term*, then it is a *Calculable Metric*. This rule allows to determinate if a metric (independently of its type) can be calculated with the available data.

Table 2. SWRL rules.

Id.	*SWRL Rule*
1	DirectMetric(?x) → RequiredData(?x)
2	RequiredData(?x) ^ available(?x, true) → AvailableData(?x)
3	DirectMetric(?x) ^ AvailableData(?x) → CalculableTerm(?x)
4	SimpleTerm(?s) ^ CalculableTerm(?a) ^ hasAsArgument(?s, ?a) → CalculableTerm(?s)
5	ComplexTerm(?c) ^ CalculableTerm(?a1) ^ CalculableTerm(?a2) ^ hasAsFirstArgument(?c, ?a1) ^ hasAsSecondArgument(?c, ?a2) → CalculableTerm(?c)
6	IndirectMetric(?m) ^ MeasurementFunction(?f) ^ CalculableTerm(?t) ^ hasDefinition(?f, ?t) ^ isCalculatedBy(?m, ?f) → CalculableTerm(?m)
7	Metric(?m) ^ CalculableTerm(?m) → CalculableMetric(?m)

3.2 SPARQL Queries

A QS is defined for a specific SP. However, the quality is measured using the attributes of the entities of the different artifacts of the software product. Therefore, the coverage analysis of a QS over a set of available information must be done at artifact level and the results must be detailed at entity level. To this purpose, the coverage analysis is done at entity level. These results are combined in order to obtain a value at artifact level.

100% coverage in a quality entity exists if all the metrics related to the entity are calculable. That is, if is available all the information required to calculate the metrics related with the attributes of an entity E in reference to a quality characteristic/subcharacteristic Q. However, a full coverage requires of the availability of all the information and, usually, the data recompilation is not a complete activity. Some data may be difficult to obtain or may not exist in early stages of the development process, but still the available data can lead to an acceptable coverage level of the quality scheme proposed. In these cases, the coverage will be less than 100%. Equation 3 shows how calculate the coverage level of an entity E for a quality characteristic/subcharacteristic Q.

$$(\#\text{ Calculable metrics of E for Q} / \#\text{ Metrics of E for Q}) \times 100 \qquad (3)$$

In order to obtain the coverage level for two variables E (entity) and Q (subcharacteristic), a SPARQL query was specified and implemented in Protégé (Figure 2). SPARQL is a semantic query language for databases, able to retrieve and manipulate data stored in Resource Description Framework format. A SPARQL query contains three main clauses: SELECT, WHERE and FILTER. In the proposed query, the SELECT clause specifies Equation 3 by calculating the relation between the quantity of calculable metrics and the overall metrics. The individuals to be counted in both cases are obtained from the WHERE clause in combination with the FILTER clause. While the WHERE clause searches all the metrics and calculable metrics related to an entity E and a quality property Q, the FILTER clause specifies the values for E and Q. A similar query was designed to link an entity E with a characteristic Q.

The developed query helps to analyze at entity level and, therefore, can be used as base for the analysis at artifact level. To summarize the artifact level analysis another SPARQL query was specified. This query helps to estimate the coverage of a quality property Q (characteristic or subcharacteristic) over an artifact A. For space reasons the query is not presented in this work.

```
SELECT ((COUNT(DISTINCT ?calculablemetricindividual)/(COUNT(DISTINCT ?metricindividual))*100) AS ?coverage)
WHERE
{       ?entity rdf:type sw:Entity . ?entity sw:entityName ?E . ?entity sw:hasAsProperty ?calculableattribute .
        ?calculableattribute rdf:type sw:Attribute . ?calculableattribute sw:isMeasuredBy ?calculablemetricindividual .
        ?calculablemetricindividual rdf:type sw:CalculableMetric . ?entity sw:hasAsProperty ?attribute .
        ?attribute rdf:type sw:Attribute . ?attribute sw:isMeasuredBy ?metricindividual .
        ?metricindividual rdf:type m:Metric . ?calculableattribute sw:isDescribedBy ?subcharacteristic .
        ?attribute sw:isDescribedBy ?subcharacteristic . ?subcharacteristic rdf:type ?Q .
        FILTER(?E = "E_Value" && regex(str(?Q),"Q_Value"))      }
```

Fig. 2. SPARQL query.

4 Activity: Analysis of the Quality Scheme Coverage

An activity was defined to combine an instantiation of the QSO (that is, a QS) with the coverage queries. After a QS has been created (at any moment of the development process) the coverage can be analyzed. The decision of when execute the coverage analysis depends on the need of analyze the quality of a SP according to the QS specification. The coverage analysis provides a mechanism that helps to know how the available data are useful to estimate several measures.

Given that the coverage analysis is made using the available information, this process can be executed several times using different sets of available data. For example, the data can be obtained (at different times) from an execution process used over the existing components or from outcomes of a simulation run. Whatever be the source of data, the availability of the information is the one that allows estimate the quantity of metrics of the QS that can be derived. To obtain these results, the member of the development team that wants to analyze the coverage (*User*) must follow the activity described in Figure 3. When the process starts, *User* wants to know which the required data is and, in response, the set of SWRL rules (explained in the section 3.1) should be executed over the *Ontology*. After that, *User* must indicate which of the required data (obtained as result of the previous activity) is available. Then, *User* must indicate that wants to find the calculable metrics based on the existing information. In response, the SWRL rules should be executed (again) over the *Ontology*. Once the calculable metrics are obtained, *User* must define which artifact (A) and which quality characteristic (Q) wants to analyze, and then, must indicate that wants to find the coverage level for both elements. Finally, the SPARQL query should be executed over the *Ontology* (with A and Q as arguments). The three final activities can be repeated for multiple pairs (A,Q) once the calculable metrics are obtained.

By following this activity, any user can use the proposed ontology and its complements to analyze how impact the available data in the quality of a SP.

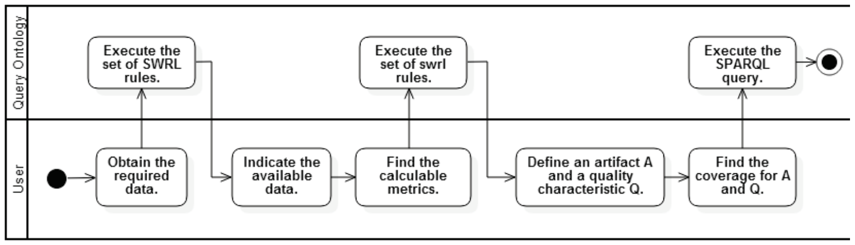
Query Ontology
User
Execute the set of SWRL rules.
Execute the set of swrl rules.
Execute the SPARQL query.
Obtain the required data.
Indicate the available data.
Find the calculable metrics.
Define an artifact A and a quality characteristic Q.
Find the coverage for A and Q.

Fig. 3. Activity to follow to analyze the coverage of a quality scheme.

5 Conclusions and Future Work

The quality of a software system is directly related to the ability of the system to satisfy its functional, nonfunctional, implied, and specified requirements. In this paper, an ontology to document quality schemes is proposed. The ontology is complemented with a set of SWRL rules and SPARQL queries that allow developers to analyze how a set of available data can be used to calculate the required metrics. Also, an activity is defined to show how to use the ontology and the other elements.

The implementation of a tool that automates the elaboration of the quality schemes and the analysis of its coverage is the next step in this direction. The ontology and the set of rules and queries designed can be taken as base of this tool, using an ontology-based approach. The main purpose of this tool should be the creation, storage, modification and query of the quality schemes along with the possibility of analyze the coverage for all quality characteristics. With this technological support, the development team could easily understand the quality aspects related with a specific software artifact and see how impact the available data in the measurement of quality.

The QSO can be adapted to other types of quality models since the software product quality semantic model is an independent ontology. Although in this work the view is centered in the internal quality attributes of a SP, the semantic model can be replaced with a model that refers to quality in use. Also, it can be replaced with quality models developed by other authors. The same changes are applicable to the software ontology. Given that the semantic model is independent, the proposed approach allows refining the model to represent a more detailed description of a SP. The improvement of this model is the main objective of the future work to be made.

References

1. Kitchenham, B., Pfleeger, S.: Software Quality: The Elusive Target. IEEE Softw. 13(1996).
2. Pressman, R.: Software Engineering: A Practitioner´s Approach. McGraw-Hill (2010).
3. Bass, L., Clements, P., Kazman, R.: Software Architecture in Practice. Addison-Wesley Professional (2003).
4. SWRL: A Semantic Web Rule Language, http://www.w3.org/Submission/SWRL/ (2004).
5. SPARQL 1.1 Query Language, http://www.w3.org/TR/sparql11-query/ (2013).
6. Pardo, C., Pino, F., García, F., Piattini, M., Baldassarre, M.: An ontology for the harmonization of multiple standards and models, Computer Standards & Interfaces, 34, 1, pp. 48–59 (2012).
7. Dominguez-Mayo, F., Escalona, M., Mejías, M. Ross, M., Stapes, G.: A quality management based on the Quality Model life cycle, Computer Standards & Interfaces, 34, 4, pp. 396–412 (2012).
8. Olsina, L., Martín, M.: Ontology for Software Metrics and Indicators. J Web Eng. 2, 262–281 (2003).
9. ISO/IEC 25010:2011 - System and software quality models, (2011).
10. Roussey, C., Pinet, F., Kang, M., Corcho, O.: An Introduction to Ontologies and Ontology Engineering, Ontologies in Urban Development Projects, Advanced Information and Knowledge Processing, 1, pp. 9-38 (2011).
11. Deissenboeck, F., Juergens, E., Lochmann, K.: Software quality models: Purposes, usage scenarios and requirements, ICSE Workshop on Software Quality '09. pp. 9–14 (2009).
12. El-Haik, B.S., Shaout, A.: Software Design for Six Sigma: A Roadmap for Excellence. John Wiley & Sons (2010).
13. Fenton, N., Bieman, J.: Software Metrics: A Rigorous and Practical Approach, Third Edition. CRC Press, Boca Raton (2014).
14. Kan, S.H.: Metrics and Models in Software Quality Engineering. Addison-Wesley Professional (2003).
15. ISO/IEC TR 9126-2:2003 - Software Product quality - Part 2: External metrics, (2003).
16. ISO/IEC TR 9126-3:2003 - Software Product quality - Part 3: Internal metrics, (2003).
17. Rijgersberg, H., Wigham, M., Top, J.L.: How semantics can improve engineering processes: A case of units of measure and quantities. Adv. Eng. Inform. 25, 276–287 (2011).
18. ISO/IEC 12207:2008 - Systems & software engineering - Software life cycle processes (2008).