QMGenerator Framework and Knowledge Base for Software Quality Model-based

Youness BOUKOUCHI[1], Adil KHAMAL[2], Abdalaziz MARZAK[3], Hicham MOUTACHAOUIK[4]

[123]Ben M'Sik Faculty of Science, University Hassan II, PB 7955, Casablanca, Morocco

[4]National School of Arts and Trades (ENSAM), PB 7955, University Hassan II, Casablanca, Morocco

Abstract: ***The software quality plays a very important role in the success of the development of the software. To determine the software quality, several standards have been proposed and several quality models have been developed. During these years, thousands of software have been produced, and for each software, an approach and a model of software quality was used in the development cycle to ensure software quality, but unfortunately, today we do not have any trace of this cumulated experiments, which can be stored in a knowledge base to be shared later by all the members of the engineering software community. To solve this problem, we present in this article the QMGenerator Framework for software quality supporting the SQuaM approach, this Framework is endowed with a knowledge base exploiting the semantic technologies of the Web in particular RDF(S) which allows managing the models of software quality and sharing experiences and accumulated knowledge during the development cycle of the software.***

Keywords: SQuaM, QMGenerator, Knowledge Base, RDF(S), Ontology, software quality

1. Introduction

The software quality plays a very important role in the success of the development of the software; it is an intersection point of all stakeholders in the development cycle of the software. To determine the software quality, several standards have been proposed and several quality models have been developed. During these years, thousands of software have been produced, and for each software, an approach and a model of software quality was used in the development cycle to ensure software quality, but unfortunately, today we do not have any trace of this cumulated experiments, which can be stored in a knowledge base to be shared later by all the members of the engineering software community.

To solve this problem, we have already developed in [3] a new approach called SQuaM (**S**oftware **Qua**lity **M**odel based) that is based on the quality models software as support of modeling of the quality, whereas in this article, we present the Framework QMGenerator for software quality supporting the SQuaM approach, this Framework is endowed with a knowledge base exploiting the semantic technologies of the Web in particular RDF(S) which allows managing the models of software quality and sharing experiences and accumulated knowledge during the development cycle of the software.

This paper is organized as follows: section 2 presents the concepts of software quality and SQuaM approach, Section 3 presents the concepts of Knowledge Base and RDF(S), Section 4 presents our Framework QM Generator and its process, section 5 presents conclusions and describes future work.

2. SQuaM & Software Quality Models

2.1 Software Quality

Software quality is the most important element in the development of software, because the quality could reduce the cost of maintenance, software testing, etc... Quality has very different meanings for customers, users, managers, developers, testers, etc… Many institutes and organizations have their own definitions of software quality and also quality models. Below some definitions of software quality [2]:

- ISO 9126: Is a set of attributes of a software product which describes and evaluates the quality.
- ANSI: Quality is the totality of features and characteristics of a product or service that relies on its ability to meet the specific needs.
- IEEE (IEEEStd 729-1983): The totality of features and characteristics of a software product that influence on its ability to meet specific needs.

In the most general sense, software quality can be defined as: An effective process for software development, applied in a manner that creates a useful product and delivers measurable value for those who produce it and those who use it.

2.2 Software Quality Models

ISO/IEC 9126-1 defines a quality model as a "framework which explains the relationship between different approaches to quality". In general, a quality model has a hierarchical structure, it divides the quality into three main levels: factors (and Sub-factors), criteria and metrics. The evaluation of a software begins with the measure of the value of each metric, then, to evaluate the criteria by their metrics, and finally, to evaluate factors by their criteria, and so we have a general evaluation of the quality with a quality model. Below are some models and methods of software quality [1]:

2.2.1 MCCALL MODEL

McCall model presents eleven criteria grouped into three visions: operations, revisions and product transitions. This model is allocated as follows: factors, criteria and metrics.

2.2.2 BOEHM MODEL

Boehm's model is similar to the model of McCall. It also presents a hierarchical quality model structured around four levels: high-level characteristics, intermediate-level characteristics, primitive-level characteristics and metrics.

2.2.3 DROMEY MODEL

Dromey model is structured around a process focused on the relationship between quality attributes and sub-attributes, and the attempt to connect the properties of products with attributes. To create this new model, the main idea was to obtain a large model to satisfy various systems. The levels of this model are defined as follows: the properties of the product, quality attributes, sub-attributes and metrics.

2.2.4 ISO 9126 ET 25000 MODEL

The model ISO9126 and ISO 25000, defines and describes a series of characteristic qualities of a software product (internal and external characteristics, characteristics of use) that can be used to specify the functional and non-functional requirements of customers and users. Each characteristic is divided into sub-characteristics, and for each of them, the standard provides a set of metrics to put in place to assess the conformity of the product that is developed from the guidelines requirements.

2.2.5 GQM APPROACH

GQM (Goal, Question, and Metric) [15] is an approach to software metrics that has been promoted by Victor BASILI, GQM defines a measurement model on three levels:

- Conceptual level (goal): A goal is defined for an object, for a variety of reasons, with respect to various models of quality, from various points of view and that is relative to a particular environment.
- Operational level (question): A set of questions is used to define models of the object of study and then focuses on that object to characterize the assessment or achievement of a specific goal.
- Quantitative level (metric): A set of metrics, based on the models, is associated with every question in order to give answers in a measurable way.

2.3 Quality Metamodel

After a detailed study [1] on quality models we have determined the specifics of each model and we have proposed [2] a general metamodel for software quality (Figure 1). This quality metamodel is divided into hierarchical elements. It structures quality into three levels: view, characteristic and metric, whose characteristics can be divided into several sub-characteristics and so on.

a) Overview (Point of view): Quality can be perceived with various points of view, differences of views are mainly due to the fact that the project has many stakeholders, each stakeholder perceives the quality of its manner, what implies a prospect focused on the specific requirements of stakeholder towards the system.
b) Characteristic: After the view, we find the characteristics, (called Factors, Goals, Properties, etc), these characteristics are broken up into several under-characteristics until arrived in granular indecomposable characteristics and which are directly measurable by metrics.
c) Metric: It is used to measure and evaluate a characteristic by values.

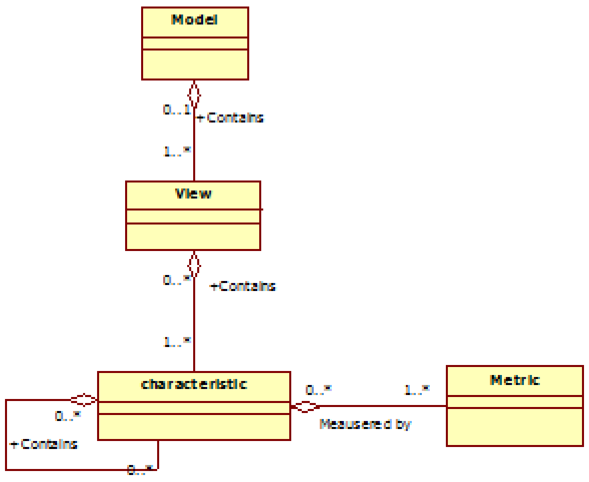


Figure 1: A Metamodel for Software Quality Models

2.4 SQuaM : Software Quality Model based

SQuaM (for **S**oftawre **Qua**lity **M**odel based) is a process of definition, modeling, implementation, evaluation and continuous improvement of software quality. It's an approach based on a model of software quality (McCall, ISO 9126,...) to evaluate the quality, it can transform the needs and requirements of users in a model of software quality. SQuaM is applicable for all types of software, and during all phases of the life cycle of software, it is only enough to have specific metrics for each phase of the life cycle, and involve them in the proposed model. The approach SQuaM provides at any time the status of software quality using measured metrics.

SQuaM consists of five steps: DMIEI (Figure 2), each step is related to the other step. Its implementation allows measuring to what extent a software product of the model quality must respect the model that is defined according to the requirements of the users. Every step is assured by experts in every domain quality, and in every step there is a set of the interacted activities by flows of object: Model, Report, code.

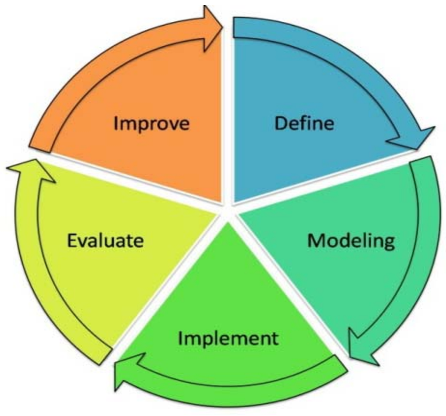


Figure 2: Five steps of SQuaM process

3. The representation of knowledge base

3.1 Semantic Web

Web evolved in a exponential way during these last decades, causing a proliferation of heterogeneous documents that are difficult to consult because their structure is very limited and which did not allow a real sharing of the knowledge. The arrival of XML (Extensible Markup Language) in 1998, gave a framework to the knowledge by the structuring and creation of new Web languages intended for a perfect sharing of the knowledge, and a new idea of the Web was presented in May, 2001 by Tim Berners-Lee et al, it is the Semantic Web. The Semantic Web provides a common framework that allows data to be shared and reused.

The W3C (for World Wide Web Consortium) has worked for the implementation of the new languages and the tools supporting the semantic Web (Figure 3) , as: XML, RDF/RDFS, OWL, etc., whose common objective is to participate in a representation of the knowledge according to the new architecture, and thus to allow easier sharing and better organization of the knowledge.

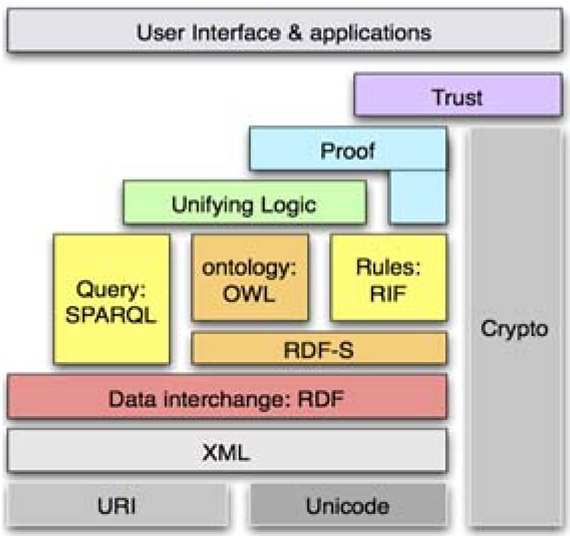


Figure 3: Semantic Web Layer Cake

3.2 Quality model & Ontologies

3.2.1 Ontologie

The term "ontology" comes from the field of philosophy that is related to the study of being or existence. By analogy, this term is used in computer sciences at the beginning of the 90s to indicate among other definitions [8]: " *explicit specification of a conceptualization* ". Ontology is one of the basic concepts of the semantic Web, they are used for the modeling of resources of the Web starting from conceptual representations of the concerned fields, thus, to allow programs to make inferences. It defines the concepts, relationships, and other distinctions that are relevant for modeling a domain. The specification takes the form of the definitions of representational vocabulary (classes, relations, and so forth), which provide meanings for the vocabulary and formal constraints on its coherent use. There are several types of formalisms and languages to represent ontology [7] as: RDF (S), OWL, UML (OCL).

3.2.2 Ontology for Software Quality Models

Based on the metamodel for software quality, we can determine the basic concepts in the field of software quality (model, view, characteristic metric) as well as the properties of every concept and relations that connect these concepts between them. The following figure (Figure 4) shows the ontology model of software quality that we realized after modeling concept in this area based on the study presented in [1], a model of software quality is evaluated using several point of view, each point of view is defining a set of characteristics, these characteristics can consist of several sub characteristics, and each characteristic is measured by one or more metrics, finally, these metrics are evaluated by a value.

3.3 RDF/RDFS et SPARQL

3.3.1 RDF(S)

RDF (Resource Description Framework) was introduced and recommended by W3C, it is the model of data that organizes within triplets the factual knowledge of the domain. This triplet consists of a subject, a predicate and an object. A triplet is interpreted as follows: " *the subject has for predicate the object* ". RDF is a higher level of description than XML that is based on a hierarchical structure of documents. An RDF expression is also called an RDF graph. The language RDF plan (RDFS) has as an objective the definition of the vocabulary of the structural knowledge of the domain. It allows describing the classes and the properties for resources RDF; it was introduced and recommended by W3C.

The use of RDF and RDFS is often noted RDF(S), they are widely used in computer systems to share knowledge. In our study we focus on quality models, we have created an RDF file which will be a knowledge base of software quality models, and to give a vocabulary and semantic to the data containing in the file RDF, we have defined a file RDFS (figure 5b) which presents the ontology of the quality models.

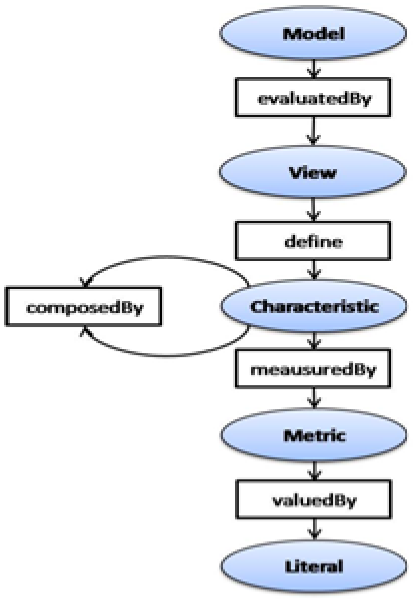


Figure 4: A graphic design for Ontology of quality meta model

```
<?xml version="1.0"?>
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-
rdf-syntax-ns#">
<Model rdf:ID="ISO9126">
<evaluatedBy>
<rdf:Seq>
<rdf:li>
<View rdf:ID="External_Quality">
<define>
<rdf:Seq>
<rdf:li>
<Characteristic rdf:ID="Functionality">
<composedBy>
<rdf:Seq>
<rdf:li>
<Characteristic rdf:ID="Suitability">
<meausuredBy>
<Metric rdf:ID="Functional_adequacy">
<valuedBy>0.85</valuedBy>
</Metric>
```

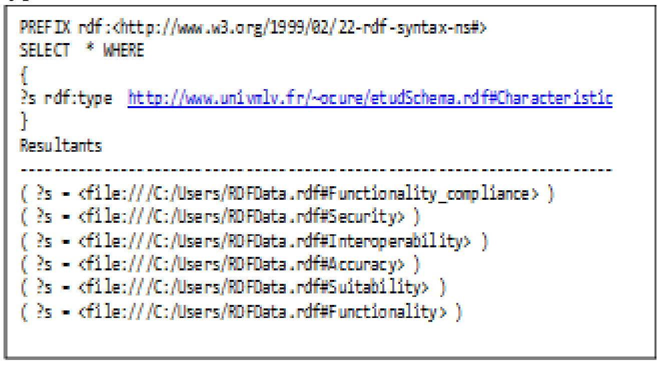
Figure 5a : RDF file

```
<?xml version="1.0"?>
 <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-
syntax-ns#"
 xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">
 <rdfs:Class rdf:ID="Model"/>
 <rdfs:Class rdf:ID="View"/>
 <rdfs:Class rdf:ID="Characteristic"/>
 <rdfs:Class rdf:ID="Metric"/>
 <rdfs:Property rdf:ID="evaluatedBy">
              <rdfs:domain rdf:resource="#Model"/>
              <rdfs:range rdf:resource="#View"/>
 </rdfs:Property>
 <rdfs:Property rdf:ID="define">
              <rdfs:domain rdf:resource="#View"/>
              <rdfs:range rdf:resource="#Characteristic"/>
 </rdfs:Property>
 <rdfs:Property rdf:ID="composedBy">
              <rdfs:domain rdf:resource="#Characteristic"/>
              <rdfs:range rdf:resource="#Characteristic"/>
 </rdfs:Property>
 <rdfs:Property rdf:ID="meausuredBy">
             <rdfs:domain rdf:resource="#Characteristic"/>
```

Figure 5b: RDFS file

3.3.2 SPARQL

SPARQL (SPARQL Protocol for RDF and Query Language) is one of the key technologies of the Semantic Web, it is a query language and protocol -such as SQL for relational databases, which allows searching, adding, modifying or deleting RDF data. It was created by W3C and became an official recommendation 15 January 2008. Above is an example of SPARQL query that returns a list of characteristics from the knowledge base RDF (Figure 6) type:



```
PREFIX rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT  * WHERE
{
?s rdf:type  http://www.univmlv.fr/~ocure/etudSchema.rdf#Characteristic
}
Resultants
-----------------------------------------------------------------------
( ?s = <file:///C:/Users/RDFData.rdf#Functionality_compliance> )
( ?s = <file:///C:/Users/RDFData.rdf#Security> )
( ?s = <file:///C:/Users/RDFData.rdf#Interoperability> )
( ?s = <file:///C:/Users/RDFData.rdf#Accuracy> )
( ?s = <file:///C:/Users/RDFData.rdf#Suitability> )
( ?s = <file:///C:/Users/RDFData.rdf#Functionality> )
```

Figure 6: The result of executing a SPARQL query on the RDF knowledge base

4. QMGenerator Framework

4.1 Principles and Functionality

The QMGenerator is Framework for the software quality supporting the SQuaM approach for the software quality, it offers an environment allowing to make the modeling, the evaluation, the improvement of the quality models and as well as the saving and the sharing of the models used and improved in a knowledge base (Figure 7). The framework QMGenerator provide the following functions:

- Quality Modeling
- Quality evaluation.
- Quality Improvement
- Quality storing and sharing in the knowledge base

4.2 Process and Architecture

4.2.1 Modeling Process

The modeling of the models of quality is done by respecting a metamodel of software quality which we have already defines in [2], the generated models respect the following the hierarchy: Views, characteristics (and subcharacteristics) and the metric (Figure 8). Considering the importance of XML technologies in the computer sciences and in particular the Web, QMGenerator instantiate quality models in a file RDF/XML. In the phase of definition of metric, QMGenerator presents to the users a glossary of metric to simplify their tasks, this glossary contains the syntax of each metric (ID, Name, Description, Reference, Formula, Audience, scaleType, Value, Interpretation, ...).

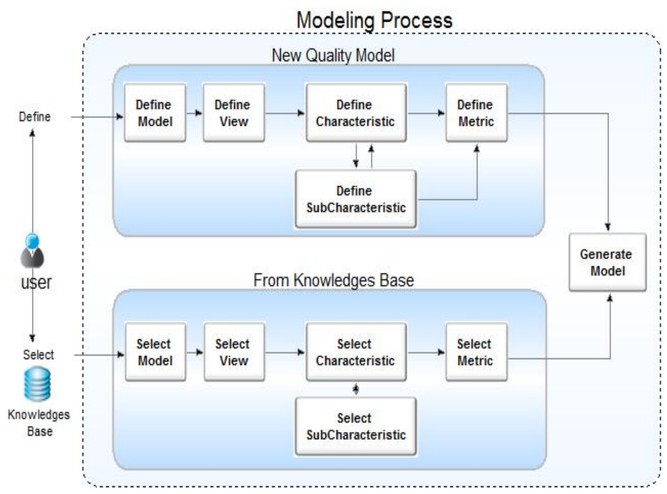


Figure 8: Modeling Process & models generation

4.3 Evaluation process

The evaluation of the quality is made through the generated quality model, developers (or other participants responsible for the quality) associate values to the metrics after measuring them. And then, the QMGenerator, starting from these measured values during the life cycle of software, creates a digital or graphic representation of these characteristics composing the model of quality and creating a software quality report (Figure 9).

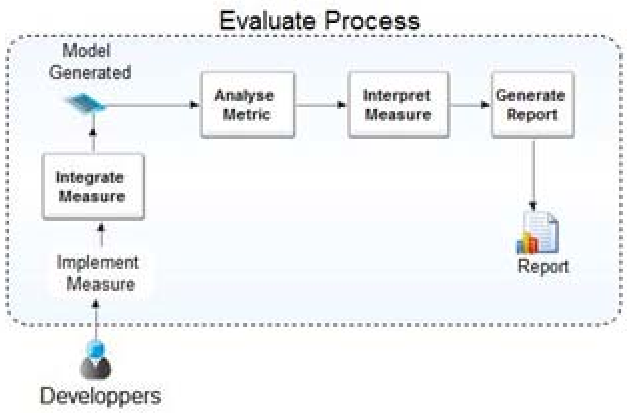


Figure 9: Evaluation Process

4.4 Improvement Process

The QMGenerator allows quality managers to make improvements in the model, these improvements include making additions, updating or deletions of views, features or metrics. Once these improvements are made, the model is a valid shareable knowledge among all members of the community of software engineering.

4.5 Technical Architecture

The QMGenerator Framework is developed by using java language in Eclipse environment , we used RDFS to the modeling and representation of the ontology domain, we used the RDF files to save knowledge in the form of triplets, and we used the JENA API for the management of knowledge (Figure 11). Jena is a Java framework for building Semantic Web applications. It provides an extensive Java libraries for helping developers develop code that handles RDF, RDFS, RDFa, OWL and SPARQL in line with published W3C recommendations. Jena includes a rule-based inference engine to perform reasoning based on OWL and RDFS ontologies, and a variety of storage strategies to store RDF triples in memory or on disk.

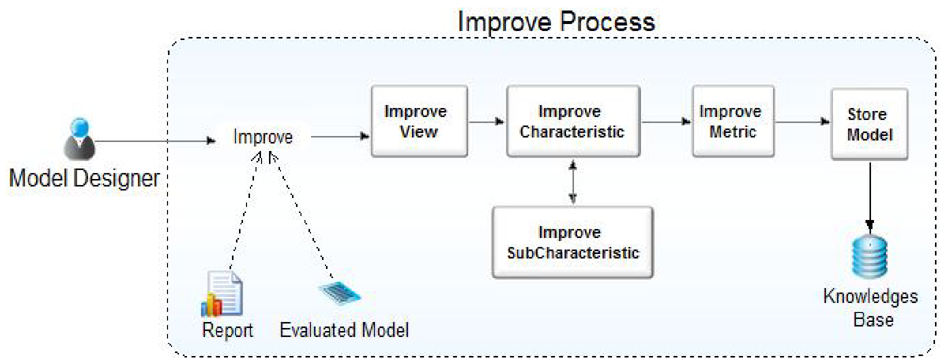


Figure 10: Improvement Process & sharing knowledge

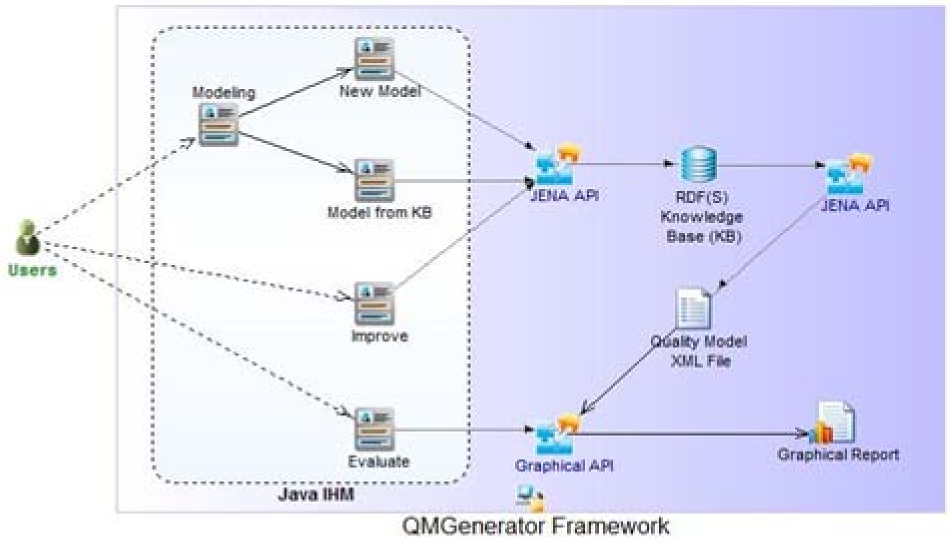


Figure 11: Technical Architecture

5. Conclusion

In this article, we presented our QMGenerator Framework for software quality based model, as well as a knowledge base of quality models to facilitate the sharing of knowledge and expertise in this area. We shall, in future work , implement our Framework in a software development project, and direct our Framework to other sectors such as industry, HR, ...

References

[1] Comparative Study of Software Quality Models, Youness BOUKOUCHI et al, the International Journal of Computer Science Issues, April 2013.

[2] A MetaModel for Quality Software Based on the MDA Approach, Youness BOUKOUCHI et al, / (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 5 (3) , 2014, 4390-4393
[3] An Approach To Software Quality Model Based (SQuaM):QMGenerator a framework supporting this approach, Youness BOUKOUCHI et al. International Journal of Science and Research (IJSR), ISSN: 2319-7064,Volume 3 Issue 6, June 2014
[4] A Quality Model for Design Patterns .Khashayar Khosravi and Yann-Gaael Gueheneuc, Summer 2004
[5] La mesure des modèles par les modèles : une approche generative, Martin Monperrus, octobre 2008.
[6] Software Engineering Metrics: What Do They Measure and How Do We Know? Cem Kaner and Walter P. Bond, 10TH INTERNATIONAL SOFTWARE METRICS SYMPOSIUM, METRICS 2004.
[7] Transition De Modèles De Connaissances Un Système De Connaissance Fondé Sur Owl, Graphes Conceptuels Et Uml, Thomas RAIMBAULT, novembre 2008, UNIVERSITÉ DE NANTES.
[8] Ontology, Tom Gruber, the Encyclopedia of Database Systems, Ling Liu and M. Tamer Özsu (Eds.), Springer-Verlag, 2009
[9] Génération d'ontologies dirigée par les modèles, Guillaume Hillairet, IDM 2008, Laboratoire L3I, Université de La Rochelle
[10] Graphes RDF et leur Manipulation pour la Gestion de Connaissances, Fabien L. Gandon, novembre 2008, INRIA Sophia Antipolis
[11] Langage d'interrogation SPARQL pour R D F, Eric Prud'hommeaux et Andy Seaborne, janvier 2008, W3C.
[12] Resource Description Framework (RDF):Concepts and Abstract Syntax, W3C Recommendation 10 February 2004
[13] RDF Schema 1.1, W3C Recommendation 25 February 2014
[14] Modèles de mesure de la qualité des logiciels, Karine Mordal, Jannik Laval et Stéphane Ducasse, November 7, 2011
[15] Understanding The Model Driven Architecture (MDA), Sinan Si Alhir, Fall 2003, Volume 11 N°3 Page 18, ISSN 1023-4918
[16] THE GOAL QUESTION METRIC APPROACH, Victor R. Basili1 et al, Encyclopedia of Software Engineering, 2 Volume Set, 1994
[17] IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications
[18] ISO/IEC TR 9126, Software engineering –Product quality – Part 1,2,3. 2002-03-15.
[19] "Enterprise Information Integration and the OMG's MDA and MOF", Randall M. Hauch, OMG's Third Workshop on UML for Enterprise Applications: Model Driven Solutions for the Enterprise October 21-24, 2002
[20] "A Metamodel for Specifying Quality Models in Model-Driven Engineering", Parastoo Mohagheghi and VegardDehlen, Engineering Research Institute, University of Iceland , 2008
[21] Definitions and Approaches to Model Quality in Model-Based Software Development – A Review of Literature, PARASTOO MOHAGHEGHI, VEGARD DEHLEN, TOR NEPLE, Engineering Research Institute, University of Iceland , 2009
[22] 12 Steps to Useful Software Metrics, Linda Westfall,The Westfall Team, westfall@idt.net