International Journal of Innovation and Scientific Research
ISSN 2351-8014 Vol. 19 No. 1 Nov. 2015, pp. 179-189

http://www.ijisr.issr-journals.org/

Le Framework QMGenerator pour la qualité logicielle et la capitalisation des expériences

Youness Boukouchi[1], Adil Khamal[1], Mohammed Amine HANINE[2], Abdelaziz Marzak[1], and Hicham Motachaouik[3]

[1]Laboratoire de Traitement de l'Information et Modélisation (LTIM),
Faculté des sciences Ben M'Sik, Université Hassan II,
Casablanca, Maroc

[2]Faculté des sciences et Techniques,
Université Hassan I,
Settat, Maroc

[3]L'Ecole National des arts et Métiers,
Université Hassan II,
Casablanca, Maroc

ABSTRACT: Software quality plays a very important role in the success of software projects, it is a point of interest of all stakeholders in the software development cycle, using a variety of approaches and models to describe and measure software quality. However, these approaches and models are little far from presenting a description of the quality reliable and faithful to the needs for the users. As well as the absence of an approach allow the capitalization of the experiences accumulated during the cycle of software development. For this, the Framework QMGenerator is done to overcome these obstacles by facilitating quality modeling, and making profile reuse experiments using an extensible knowledge base that provides direction and guidance to stakeholders the software development cycle.

KEYWORDS: Model and Metamodel, knowledge base, Ontology, Quality Software

RESUME: La qualité du logiciel joue un rôle très important dans la réussite des projets logiciels, c'est un point d'intérêt de tous les intervenants dans le cycle de développement logiciel, ils utilisent une variété de démarches et modèle pour décrire et mesurer la qualité logicielle. Cependant, ces démarches et modèles sont un peu loin de présenter une description de la qualité fiable et fidèle aux besoins des utilisateurs, ainsi que l'absence d'une démarche permettant la capitalisation des expériences cumulées au cours du cycle de développement logiciel. Pour cela, le Framework QMGenerator est fait pour dépasser ces obstacles en facilitant la modélisation de la qualité, et en faisant profil de la réutilisation des expériences à l'aide d'une base de connaissance extensible permettant d'orienter et guider les intervenants dans le cycle de développement logiciel.

MOTS-CLEFS: Modèle et Métamodèle, Base de connaissance, Ontologie, Qualité logicielle.

1 INTRODUCTION

La qualité logicielle joue un rôle très important dans la réussite de développement des logiciels, c'est un point d'intersection de tous les intervenants dans le cycle de développement logiciel. Au cours de ces dernières années, plusieurs modèles et démarches ont été utilisés pour déterminer la qualité logicielle et produire des milliers des logiciels, mais

malheureusement qu'aujourd'hui, ces modèles sont un peu loin de refléter une image réelle de la qualité logicielle et de présenter une description fiable et fidèle aux besoins des utilisateurs et développeurs. En plus, cette démarche souffre de l'absence d'un processus ou outils permettant la capitalisation des expériences cumulées au cours du cycle de développement logiciel pour être partagé par la suite entre tous les membres de la communauté de l'ingénierie logicielle et d'exploiter le retour d'expériences cumulées dans les cycles de développement précédents.

Pour dépasser ces obstacles, nous avons proposé le Framework QMGenerator qui permet de gérer la qualité dans le cycle développement logiciel en se basant sur un métamodèle de qualité qui permet de modéliser et instancier les modèles sous forme d'un fichier XML pour présenter un support d'interopérabilité des concepts de qualité entre les artéfacts logiciels. Le Framework QMGenerator permet aussi la capitalisation des expériences cumulées au cours du cycle de développement des logiciels ainsi il facilitera aux utilisateurs la gestion de la qualité en tirant profils des points forts des modèles connus et des modèles éprouvés par l'expérience.

Cet article sera réparti comme suit : la section 2 introduit les modèles connus de la qualité logicielle et propose un métamodèle de qualité. La section 3 propose une ontologie de qualité logicielle développée en langage OWL. La section 4 introduit les processus et l'architecture de Framework QMGenerator. Finalement la section 5 présentes une conclusion générale.

2 LE MÉTAMODÈLE DE QUALITÉ LOGICIELLE

2.1 LES MODÈLES DE QUALITÉ LOGICIELLE

Un modèle de qualité est défini comme étant un cadre qui explique la relation entre les différentes approches de la qualité, il permet de définir et d'évaluer la qualité sous différents aspects (Produit, Processus, Ressources, etc.), dans la littérature nous avons trouvé de nombreux modèles de la qualité logicielle, la plupart d'entre eux sont de nature hiérarchique organisée sous la structure suivante: Facteurs, Critères et Métriques (FCM). L'évaluation d'un logiciel commence par la mesure des métriques de chaque critère de qualité. Ci-dessous quelques modèles de la qualité logicielle.

2.1.1 LE MODÈLE DE MCCALL

Le modèle de McCall (1) combine onze critères autour de trois visions : les opérations, les révisions, et les transitions de produits. Ce modèle est reparti comme suit: Facteurs, critères et métriques.

2.1.2 LE MODÈLE DE BOEHM

Le modèle de Boehm (2) est similaire au modèle de McCall, il présente également un modèle de qualité hiérarchique structuré autour de caractéristiques de haut niveau, de niveau intermédiaire, de niveau primitif et les métriques.

2.1.3 LE MODÈLE DE DROMEY

Le modèle de Dromey (3) (4), est structuré autour d'un processus concentré sur les relations entre les attributs de qualité et les sous-attributs, ainsi que la tentative de connexion des propriétés de produits avec des attributs. L'idée principale pour créer ce nouveau modèle était d'obtenir un modèle suffisamment large pour satisfaire des différents systèmes. Les couches de ce modèle sont définies comme suit: les propriétés du produit, les attributs de la qualité, les sous-attributs et les métriques.

2.1.4 LE MODÈLE ISO 9126

Le modèle ISO9126 (5), décrit une série de caractéristiques de qualités d'un produit logiciel (caractéristiques internes et externes, caractéristiques à l'utilisation) qui peuvent être utilisées pour spécifier les exigences fonctionnelles et non fonctionnelles des clients et des utilisateurs. Chaque caractéristique est décomposée en sous-caractéristiques, et pour chacune d'elle, la norme propose une série de métriques à mettre en place pour évaluer la conformité du produit développé par rapport aux exigences formulées.

2.1.5 LE MODÈLE GQM

GQM (Goal, Question, Metric) est une approche de la mesure des systèmes logiciels qui a été promue par Victor BASILI (6), GQM définit un modèle de mesure à trois niveaux :

- Niveau conceptuel (Goal) : ce niveau définit les buts ou les objectifs à atteindre. Un but relève de différents points de vue selon lesquels le logiciel est analysé et peut concerner les différents aspects de logiciels : le produit, le processus ou les ressources mises en œuvre.
- Niveau opérationnel (Question) : ce niveau est formé de l'ensemble des questions associées à un objectif particulier, et qui essayent de caractériser l'objet de mesure pour le qualifiée d'un point de vue particulier.
- Niveau quantitatif (Metric): ce niveau est formé de l'ensemble de données associées à une question, elles peuvent être objectives ou subjectives, elles fournissent des réponses quantitatives.

2.1.6 LA MODÈLE IEEE 1061-1998

La norme IEEE1061 (7) fournit une méthodologie pour établir des exigences de qualité ainsi que: la mise en œuvre, l'analyse et la validation des processus de mesure de la qualité produit. Cette méthode s'applique à tous les logiciels pendant les phases de cycle de vie du logiciel. Le système de qualité logiciel est conçu pour être flexible, il permet des ajouts, des suppressions et des modifications des facteurs de qualité, des sous-facteurs, et des métriques. Chaque niveau peut être étendu à plusieurs sous-niveaux. Cette méthodologie peut donc être appliquée à tous les systèmes.

2.2 LE MÉTAMODÈLE DE QUALITÉ LOGICIELLE

Malgré cette diversité des modèles de qualité et leur popularité dans le domaine du génie logiciel, ils ont montré quelques limites et faiblesse au niveau de définition, d'implémentation, ou d'évaluation de la qualité, à savoir :

- certains modèles présentent une difficulté à mettre en œuvre dans un environnement, vu le nombre des critères et les métriques définies, par exemple McCall présente plus de 23 critères avec une proposition de 300 métriques.
- certains modèles ne présentent pas une vision claire qui explique la correspondance entre les métriques et les critères, ainsi qu'ils ne séparent pas clairement les différents points de vue, par exemple lorsqu'un critère obtient une faible note, il est difficile de relier cette note directement au problème qu'elle pointe, surtout lorsque le critère est composé de plusieurs métriques.
- la plupart de ces modèles souffrent de l'absence de directives et de critères de décomposition des concepts de qualité complexe, ce qui rend difficile leur raffinement ainsi que leur localisation dans certains modèles de qualité de grande taille.
- ces modèles se limitent généralement à un nombre fixe de niveaux (4 niveaux maximum), ce qui limite la définition et la structuration des attributs qualité complexe en trois ou quatre niveaux, par exemple l'utilisabilité ne peut pas être décomposée vers des propriétés mesurables en seulement deux niveaux.

Alors, pour pallier ces limitations, nous avons utilisé les concepts de base de l'ingénierie dirigée par modèle (MDA, Model Driven Architecture) pour proposer un métamodèle de qualité logicielle qui permet de créer des modèles selon les exigences spécifiques des utilisateurs, ce métamodèle permet aussi de dépasser la contrainte de nombre limite des niveaux en décomposant les caractéristiques complexes en plusieurs niveaux jusqu'aux métriques. Ce métamodèle se décompose en éléments hiérarchiques, il structure la qualité en trois niveaux : vue, caractéristique, et métrique (8):

a) vue (Point de vue) : la qualité peut être perçue avec différents points de vue, les divergences des vues sont principalement dues au fait que le projet a de nombreuses parties prenantes, chaque intervenant perçoit la qualité de sa manière.
b) caractéristique : après les vues, on trouve les caractéristiques qui décrivent en détaille la perspective de la qualité pour chaque vue, (appelé Facteurs, Buts, Propriétés, etc.), ces caractéristiques sont décomposées en plusieurs sous-caractéristiques jusqu'à l'arrivée à des caractéristiques granulaires indécomposables et sont directement mesurables par des métriques.
c) métrique : c'est l'élément de base qui permet de mesurer et évaluer une caractéristique par une valeur significative.

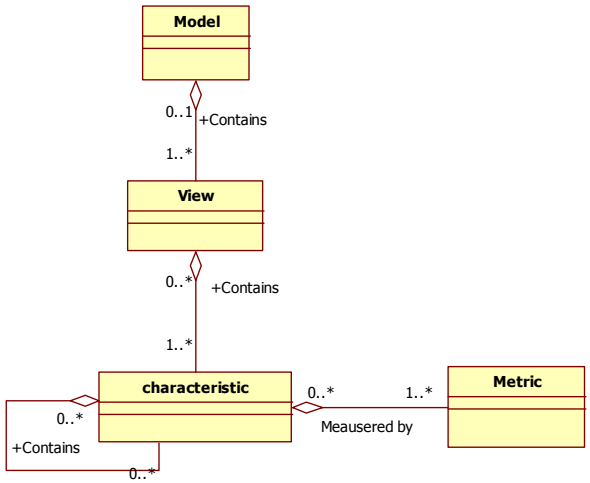

Figure 1 : Un métamodèle de qualité logicielles

Ce métamodèle de qualité (8)permet de générer des modèles de qualité tout en respectant la spécificité de chaque modèle de qualité ou de générer d'autre modèle de qualité personnels (Tableau 1). Il permet d'offrir des apports pertinents au domaine de l'ingénierie logiciel, à savoir: la définition des modèles de qualité d'une façon simple grâce à la structuration prédéfinit imposé par le métamodèle, un cadre de validation et de compréhension des modèles de qualité qui sont générés par d'autres outils de modélisations, un pont de communication entre les plateformes de développement logiciel, la simplicité de la mise-en œuvre des modèles proposés dans des environnements différents grâce à la possibilité d'instanciation des modèles en langage XMI/XML, la possibilité de décomposition à plusieurs niveaux des concepts complexe de qualité et les raffiner dans des projets de qualité de grande taille, la description fine de ce domaine de qualité facilite la description d'un domaine de connaissance à l'aide des ontologies qui complètent cette description, la possibilité d'automatisation de processus de modélisation et d'abstraction des modèles de qualité par des outils informatiques, et finalement, la possibilité de la réutilisation des modèles ou leurs éléments dans d'autre projets de qualité logicielle.

Tableau 1 : Comparaison entre la structure des modèles de qualité logiciel

Niveau	Métamodèle	McCall	Boehm	ISO9126	GQM	IEEE1061	Dromey	Modèle personnel
1	Vue	Vue	Vue	Vue	Vue	Vue	Vue	Vue
2	caractéristique	Facteur	Niveau supérieur de caractéristique	caractéristique	Objectif	Facteur	Propriétés de produit	caractéristique
3	Sous-caractéristique	Critère	Niveau Intermédiaire primitive	Sous-caractéristique	Questions	Sous-facteur	Attributs de qualité	Sous-caractéristique
4	Sous-Sous-caractéristique	-	Caractéristique primitive	Attributs de qualité	-	-	Sous-attributs	Sous-Sous-caractéristique
N	N Sous-caractéristique	-	-	-	-	-	-	N Sous-caractéristique
Base	Métrique	Métrique	Métrique	Métrique	Métrique	Métrique	Métrique	Métrique

3 L'ONTOLOGIE DES MODELES DE QUALITE

3.1 LA REPRÉSENTATION DES CONNAISSANCES

La structuration et la représentation des connaissances dans le domaine de qualité logicielle, sont les solutions idéales pour encourager la collaboration et le partage des expériences entre les artefacts logiciels et les intervenants dans le cycle de développement logiciel. Ceci exige une structuration et conceptualisation commune afin de permettre une meilleure

description des modèles de qualité, l'absence de toute notion commune de structure, de syntaxe et de sens entraîne obligatoirement l'un des problèmes d'intégration et d'interopérabilité (9).

Nous avons adopté des approches sémantiques basées sur des ontologies en particulier OWL pour permettre une description sémantique des modèles de qualité (10) (11). En se basant sur le métamodèle de qualité logicielle, nous avons déterminé les concepts de base dans le domaine de qualité logicielle (model, vue, caractéristique, métrique) ainsi que les propriétés de chaque concept et les relations liant ces concepts entre eux (12). L'ontologie développée dans ce domaine nous a permis de partager la compréhension commune de la structure des modèles entre les utilisateurs et les développeurs logiciels, de permettre la réutilisation des expériences et l'évolution des spécifications ce domaine. (13).

La figure 2 présente l'ontologie de modèles de qualité logicielle et ses concepts. Un modèle de qualité logicielle est évalué selon plusieurs points de vue, chaque point de vue définissent un ensemble de caractéristiques, ces caractéristiques peuvent être composé en plusieurs sous caractéristiques, et chaque caractéristique est mesurée par une ou plusieurs métriques, finalement, une métrique est évaluée par une valeur.

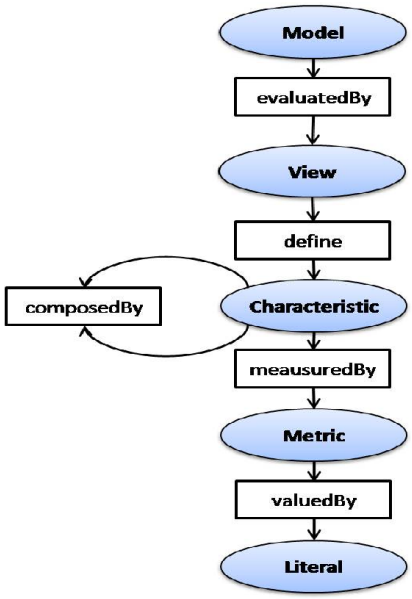

Figure 2 : Conception graphique d'ontologie de métamodèle de qualité

3.2 LA DESCRIPTION EN LANGAGE OWL

Pour décrire notre ontologie, nous avons utilisé le langage OWL qu'est un langage de représentation des connaissances construit sur le modèle de données de RDF. Il fournit les moyens pour définir des ontologies web structurées (14). Sa deuxième version est devenue une recommandation du W3C fin 2012. Ci-dessous la présentation de notre ontologie de modèle de qualité dans le langage OWL :

```
 // Object Properties
<!-- http://www.uh2m.ac.ma/QualityModel#composedBy -->
<ObjectProperty rdf:about="&QualityModel;composedBy">
        <rdf:type rdf:resource="&owl;ReflexiveProperty"/>
        <rdf:type rdf:resource="&owl;SymmetricProperty"/>
         <rdfs:range rdf:resource="&QualityModel;Characteristic"/>
        <rdfs:domain rdf:resource="&QualityModel;Characteristic"/>
</ObjectProperty>
<!-- http://www.uh2m.ac.ma/QualityModel#definedBy -->
<ObjectProperty rdf:about="&QualityModel;definedBy">
        <rdfs:range rdf:resource="&QualityModel;Characteristic"/>
        <rdfs:domain rdf:resource="&QualityModel;View"/>
</ObjectProperty>
<!-- http://www.uh2m.ac.ma/QualityModel#measuredBy -->
<ObjectProperty rdf:about="&QualityModel;measuredBy">
         <rdfs:domain rdf:resource="&QualityModel;Characteristic"/>
```

```
        <rdfs:range rdf:resource="&QualityModel;Metric"/>
</ObjectProperty>
<!-- http://www.uh2m.ac.ma/QualityModel#viewedBy -->
<ObjectProperty rdf:about="&QualityModel;viewedBy">
        <rdfs:domain rdf:resource="&QualityModel;Model"/>
        <rdfs:range rdf:resource="&QualityModel;View"/>
</ObjectProperty>
// Data properties
<!-- http://www.uh2m.ac.ma/QualityModel#Value -->
<DatatypeProperty rdf:about="&QualityModel;Value">
        <rdfs:domain rdf:resource="&QualityModel;Metric"/>
        <rdfs:range rdf:resource="&xsd;string"/>
</DatatypeProperty>
// Classes
<!-- http://www.uh2m.ac.ma/QualityModel#Characteristic -->
<Class rdf:about="&QualityModel;Characteristic"/>
<!-- http://www.uh2m.ac.ma/QualityModel#Metric -->
<Class rdf:about="&QualityModel;Metric"/>
<!-- http://www.uh2m.ac.ma/QualityModel#Model -->
<Class rdf:about="&QualityModel;Model"/>
<!-- http://www.uh2m.ac.ma/QualityModel#View -->
<Class rdf:about="&QualityModel;View"/>
```

L'ontologie proposée pour la conceptualisation des connaissances dans le domaine de qualité logicielle, présente plusieurs avantages à savoir : elle peut être utilisée pour rechercher et repérer l'information pertinente, elle permet de gérer les connaissances et le retour d'expériences cumulées dans le processus de définition, modélisation et l'évaluation de la qualité, elle permet de favoriser le partage des expériences entre tous les intervenants dans le cycle de développement logiciel, elle permet l'interopérabilité des modèles vue que la base de description de cette ontologie sont des langage standardisés par l'OMG (comme XML, OWL), et finalement, elle permet aux systèmes multi-agents d'utiliser le retour d'expériences pour les rendre plus intelligent et performant dans l'assurance de la qualité logicielle.

Cependant, cette ontologie reste pour le moment la première version introduite dans le domaine de la qualité logicielle, nous la présentons aux chercheurs et les experts de domaine pour la discuter, la réviser, l'évaluer et la mettre en œuvre dans des applications et solutions informatiques.

4 LE FRAMEWORK QMGENERATOR

4.1 PRINCIPES DE FONCTIONNEMENT

Le QMGenerator est un Framework à base d'un métamodèle de qualité logicielle qui permet de gérer les modèles de qualité ainsi que le retour d'expérience cumulée dans le processus de définition, d'évaluation et l'amélioration de la qualité logicielle (15). QMGenerator offre un environnement permettant de faire la modélisation, l'évaluation, l'amélioration des modèles de qualité et le partage des modèles comme des expériences dans une base de connaissances. Il permet d'assurer les fonctions principales suivantes: la modélisation des modèles de qualité, le contrôle et l'évaluation de la qualité, l'amélioration de modèle de qualité, et le partage des expériences dans la base de connaissances.

Le Framework QMGenerator permet à partir de métamodèle proposé de définir des modèles de qualité logicielle tel que ISO9126 ou d'autres modèles personnels, ensuite le Framework permet d'instancier les modèles en format XML pour permettre leur utilisation par les autres intervenants dans le cycle de vie de logiciel, finalement, et finalement, Le Framework exploite le retour d'expérience pour construire une base de connaissances (Figure 3).

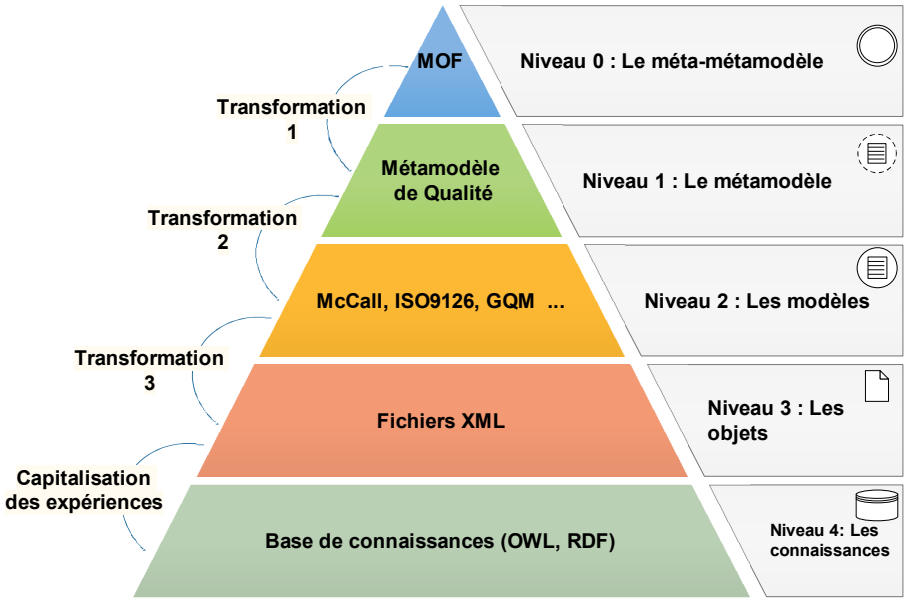

Figure 3 : Les niveaux de modélisation et de capitalisation des expériences

4.2 QMGENERATOR ET LES DÉMARCHES DE QUALITÉ

La démarche qualité logicielle est l'ensemble des actions visant l'amélioration et la gestion de la qualité logicielle, elle a pour but de faire évoluer l'organisation pour réaliser les meilleures prestations en matière de produits logiciels et services, tout en faisant intervenir l'ensemble du personnel (16). Dans la littérature nous avons trouvé un ensemble de des démarches de la qualité logicielle que nous citons ici les plus populaires et les plus utilisées dans le domaine du génie logiciel tel que la démarche PDCA et DMAIC :

- La démarche PDCA (Plan, Do, Check et Act) modélisée par Deming consiste à reproduire continuellement quatre processus, à savoir: Planifier (Plan) qui identifie les besoins et planifier la mise en œuvre des actions correctives. Réaliser (Do) qui exécute le plan d'action et mettre en œuvre toutes les opérations correctives mentionnées dans le plan. Contrôler (Check) qui vérifie les indicateurs de performance et les ressources mises en œuvre dans l'étape précédente. Agir **(Act)** qui vérifie que les solutions mises en place sont efficaces et rechercher les points d'améliorations.
- La démarche DMAIC (Define, Measure, Analyse, Improve, Control), est composée de cinq processus pour l'amélioration et l'optimisation continue de la qualité, à savoir : Définir (Define) qui définit les objectifs et les limites du projet. Mesurer (Measure) qui rassemble les informations de base sur la performance actuelle. Analyser (Analyse) qui identifie les causes originelles des problèmes de qualité. Améliorer (Improve) qui met en place des solutions s'adressant aux problèmes identifiés précédemment. Contrôler (Contrôle) qui évalue et visualiser les résultats de la phase précédente.

Le Framework QMGenerator présente un support pour ces démarches de qualité logicielle, il est composée de trois processus importants dans le cycle de l'amélioration contenue de la qualité (ACQ) logicielle, le Framework permet d'assurer la modélisation, l'évaluation, et l'amélioration contenue de la qualité, ces processus sont la base de plusieurs démarches de la qualité logicielle telle que PDCA et DMAIC (Tableau 2).

Tableau 2 : Les processus assuré par le Framework QMGenerator

	Cycle PDCA				Cycle DMAIC				
Processus de qualité	**Plan**	**Do**	**Chek**	**Act**	**Define**	**Measure**	**Analyse**	**Improve**	**Controle**
Framework QMGenerator	✓	-	✓	✓	✓	-	✓	✓	✓

4.3 LES TROIS PROCESSUS DE QMGENERATOR

4.3.1 LE PROCESSUS DE MODÉLISATION

La modélisation des modèles de qualité se fait en respectant notre métamodèle de qualité logicielle que nous l'avons déjà présenté, les modèles générés respectent la hiérarchie suivante : les vus, les caractéristiques (et les sous-caractéristiques) et les métriques (Figure 4), et vu l'importance des technologies XML dans le monde informatique et notamment le Web, le QMGenerator permet d'instancier les modèles de qualité dans un fichier RDF/XML.

Au cours de processus de modélisation, les utilisateurs ont le choix entre de créer leur propre modèle ou de faire l'aide de la base de connaissances pour concevoir un modèle de qualité.

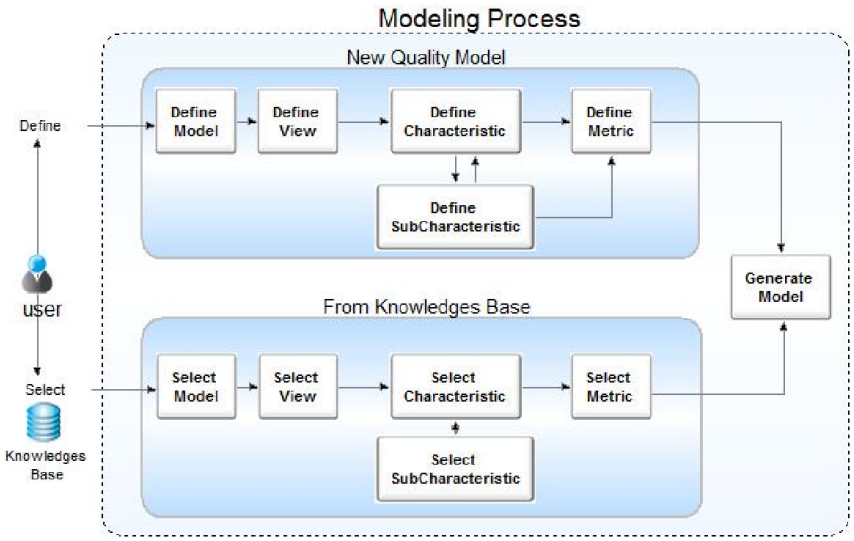

Figure 4 : Le processus de modélisation et de génération des modèles

4.3.2 LE PROCESSUS D'ÉVALUATION

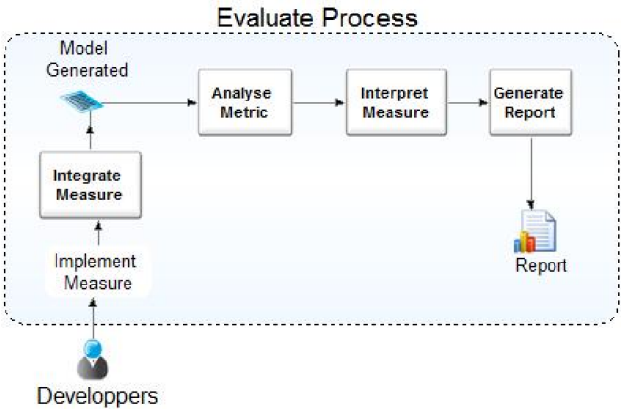

Figure 5 : le processus d'évaluation de la qualité logicielle

L'évaluation de la qualité se fait à travers le modèle de qualité généré, les développeurs ou les autres intervenants responsables de la qualité font la mesure des métriques, QMGenerator permet à partir de ces valeurs mesurées au cours du cycle de vie de logiciel de faire une représentation de ces caractéristiques et de générer un rapport de la qualité logiciel (Figure 5).

4.3.3 PROCESSUS D'AMÉLIORATION DES MODÈLES

Le QMGenerator permet aux responsables de la qualité de faire des améliorations pour faire des ajouts, des modifications ou des suppressions au niveau des vues, des caractéristiques ou des métriques. Une fois que ces améliorations sont faites, le modèle (ou certains éléments de ce modèle) sera une connaissance valide partageable à partir de la base de connaissances entre tous les membres de la communauté de l'ingénierie logicielle qui vont les exploiter dans des futurs projets (Figure 6).

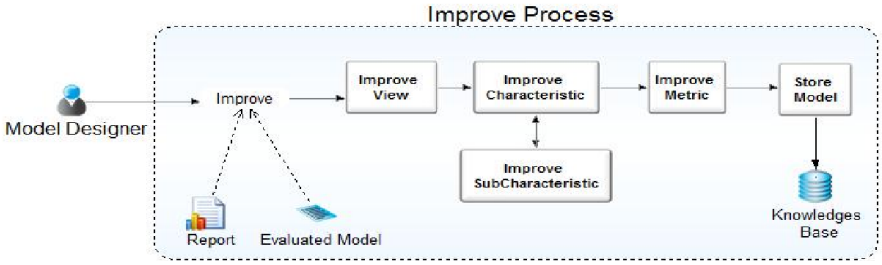

Figure 6 : Le processus d'amélioration des modèles de qualité

4.4 ARCHITECTURE TECHNIQUE

Le Framework QMGenerator est développé par le langage java, nous avons utilisé le langage OWL pour la modélisation et la représentation des connaissances, nous avons utilisé les fichiers RDF et OWL pour la sauvegarde des connaissances sous formes des triplets, et nous avons utilisé l'API comme JENA et OWL pour la gestion de ces connaissances (Figure 7).

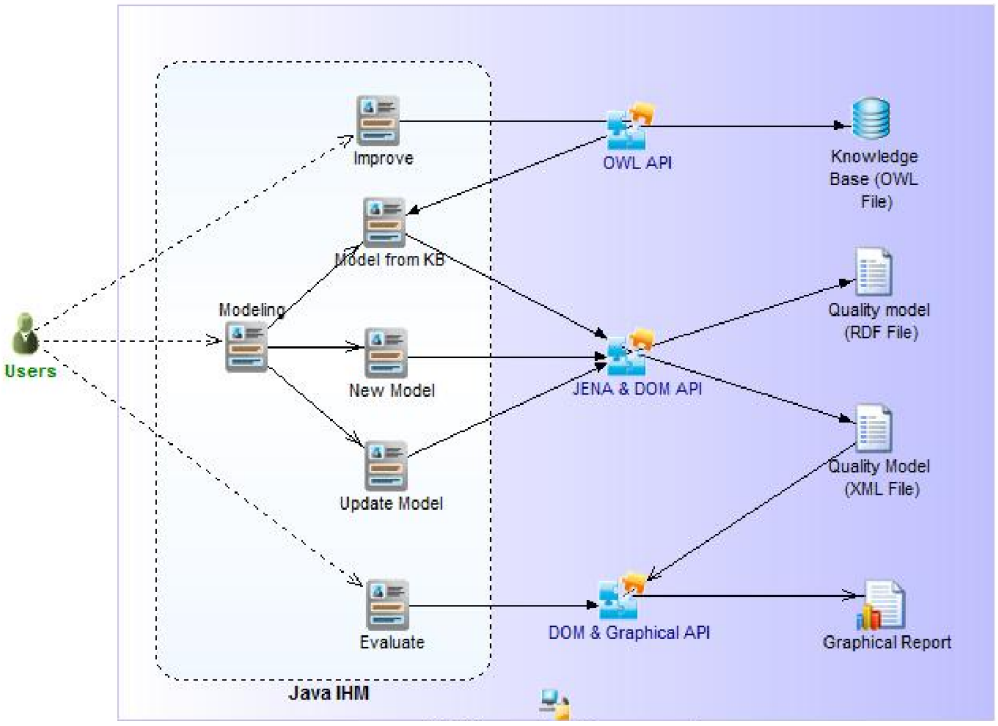

Figure 7 : L'architecture technique de Framework QMGenerator

5 CONCLUSION

Les apports de Framework QMGenerator dans le domaine de l'ingénierie logicielle et l'ingénierie des connaissances sont multiples, nous avons cité quelques applications dans cet article. Cependant il nous reste dans les futurs travaux, d'enrichir ce Framework pour offrir un support fiable aux tous les processus de démarches de qualité (PDCA et DMIAC) et les

automatiser en totalité, d'autre part, c'est exploiter parfaitement la base de connaissances pour tirer des bonnes pratiques de la qualité logicielle, et finalement est de valider le Framework dans le secteur de l'industrie.

REFERENCES

[1] McCall, Jim A, K.Richards, Paul et F.Waiters, Gene. *FACTORS IN SOFTWARE QUALITY, Preliminary Handbook on Software Quality for an Acquisition Manager.* s.l. : General Electric/Command & Information Systems, 1977. RADC-TR-77-369 Vo III.

[2] Boehm, Barry W. *Characteristics of software quality.* s.l. : North-Holland Pub, 1978.

[3] *A Model for Software Product Quality.* Dromey, R. Geoff. 2, s.l. : IEEE Transactions on Software Engineering, 1995, Vol. 21.

[4] *Cornering the Chimera.* Dromey, R. Geoff. s.l. : IEEE Software, 1996.

[5] ISO/IEC-JTC. *Information technology—Software product quality— Part 1 : Qualitymodel.* s.l. : ISO/IEC, 1991. ISO/IECFDIS9126-1:1991.

[6] *THE GOAL QUESTION METRIC APPROACH.* Basili, Victor R., Caldiera, Gianluigi et Rombach, H. Dieter. s.l. : Marciniak, J.J, 1994, Vol. 1.

[7] IEEE, American National Standards Institute. *IEEE Standard for a Software Quality : Metrics Methodology.* s.l. : IEEE Computer Society, 1998.

[8] *A MetaModel for Quality Software Based on the MDA Approach.* BOUKOUCHI (a), Youness, et al., et al. s.l. : International Journal of Computer Science and Information Technologies, 2014.

[9] Grigoris, Antoniou et Harmelen, Frank van van. *A Semantic Web Primer.* s.l. : The MIT Press, 2008.

[10] *Prise en compte des standards de qualité et des préférences utilisateurs pour la modélisation des propriétés non fonctionnelles dans OWL-S.* Stéphane Jean, Francisca Losavio, Alfredo Matteo, Nicole Levy. 1, s.l. : Journal Technique et Science Informatiques (TSI), 2012, Vol. 31.

[11] McGuinness, Deborah L. et Noy, Natalya Fridman. *Développement d'une ontologie 101 : Guide pour la création de votre première ontologie.* s.l. : Université de Stanford, 2001.

[12] *Comparative Study of Software Quality Models .* BOUKOUCHI (b), Youness, et al., et al. 2013, International Journal of Computer Science Issues.

[13] Hillairet, Guillaume. *Génération d'ontologies dirigée par les modèles .* s.l. : Laboratoire L3I, Université de La Rochelle, 2008.

[14] Grigoris Antoniou, Frank van van Harmelen. *A Semantic Web Primer.* s.l. : The MIT Press, 2008.

[15] *An Approach to Software Quality Model Based (SQuaM): (QMGenerator a Framework Supporting this Approach).* BOUKOUCHI (c), Youness, et al., et al. 2014, International Journal of Science and Research .

[16] *Fiche pratique « Les démarches qualités ».* Anact, Réseau. 2007.

[17] Camille Pradel, Ollivier Haemmerlé, Nathalie Hernandez. *Des patrons modulaires de requêtes SPARQL dans le système SWIP.* s.l. : 23es Journées Francophones d'Ingénierie des Connaissances, Paris : France, July 2012.

[18] *Software Engineering Metrics: What Do They Measure and How Do We Know? .* Cem Kaner, Walter P. Bond. 2004, 10TH INTERNATIONAL SOFTWARE METRICS SYMPOSIUM, METRICS.

[19] Monperrus, Martin. *La mesure des modèles par les modèles : une approche generative.* 2008.

[20] Karine Mordal, Jannik Laval, Stéphane Ducasse. *Modèles de mesure de la qualité des logiciels.* s.l. : Hermès., 2011.

[21] ISO. *Software engineering –Product quality – Part 1,2,3.* s.l. : ISO/IEC TR 9126, 2002.

[22] W3C. *Resource Description Framework (RDF):Concepts and Abstract Syntax.* s.l. : W3C , 2004.

[23] Gandon, Fabien L. *Graphes RDF et leur Manipulation pour la Gestion de Connaissances.* s.l. : INRIA Sophia Antipolis, 2008.

[24] Thomas, RAIMBAULT. *Transition de Modèles de Connaissances d'un Système de Connaissance Fondé dur Owl, Graphes Conceptuels Et Uml.* s.l. : UNIVERSITÉ DE NANTES, 2008.

[25] PARASTOO MOHAGHEGHI, VEGARD DEHLEN, TOR NEPLE. *Definitions and Approaches to Model Quality in Model-Based Software Development – A Review of Literature.* s.l. : Engineering Research Institute, University of Iceland, 2009.

[26] *SOFTWARE QUALITY: THE ELUSIVE TARGET.* KITCHENHAM, BARBARA et PFLEEGER, SHARI LAWRENCE. 1, s.l. : IEEE SOFTWARE , 1996, Vol. 13.

[27] *Ontologies pour le Web sémantique.* Raphaël Troncy, Jean Charlet, Bruno Bachimont. s.l. : Revue I3, numéro Hors Série «Web sémantique», 2004.

[28] *Elements of Software Science.* Halstead, Maurice Howard. s.l. : Elsevier, amsterdam netherlands,, 1977.

[29] Fenton, Norman E et Pfleeger, Shari Lawrence. *Software metrics : a rigorous and practical approach.* s.l. : International Thomson Computer Press, 1996.

[30] Mukerji, Joaquin Miller and Jishnu. *MDA Guide Version 1.0.1.* s.l. : OMG, 2003.

[31] Bézivin, Jean. *Sur les principes de base de l'ingénierie des modèles .* s.l. : Université de Nantes, Equipe ATLAS, (INRIA & LINA) , 2004.
[32] Frankel, David S. *Model Driven Architecture: Applying MDA to Enterprise Computing.* s.l. : Wiley, 2003. ISBN 978-0-471-31920-7.
[33] Gruber, T. R. *Toward principle for the design of ontologies used for knowledge sharing.* s.l. : Standford University, 1993. 93-04.
[34] Psyché, Valéry. *État de l'art sur les ontologies. Application aux systèmes à base de connaissances .* s.l. : Centre de recherche LICEF, 2003. ISBN 2-7624-4513-2 (rapport technique LICEF03NR1).
[35] FÜRST, Frédéric. *Contribution à l'ingénierie des ontologies : une méthode et un outil d'opérationnalisation.* s.l. : Université de Nantes - Ecole des Mines de Nantes, 2004.
[36] Pressman, Roger S. *Software Engineering A PRACTITIONER'S APPROACH.* s.l. : McGraw-Hill, 2010. ISBN 978–0–07–337597–7.
[37] SWEBOK . *Guide to the Software Engineering Body of Knowledge .* s.l. : IEEE, 2004.
[38] Doug Bell, Ian Morrey, John R. Pugh. *Software Engineering: A Programming Approach.* s.l. : Prentice Hall, 1992.
[39] Vaucher, Stéphane. *Modelling Software Quality: A Multidimensional Approach.* s.l. : Université de Montréal, 2010.
[40] Combemale, Benoît. *Ingénierie Dirigée par les Modèles (IDM) - État de l'art -.* s.l. : Institut de Recherche en Informatique de Toulouse, 2008.
[41] Benjamin, Woolley. *The Bride of Science: Romance, Reason, and Byron's Daughter.* s.l. : McGraw-Hill, 1999.
[42] *Software quality models: purposes, usage scenarios and requirements.* Deissenboeck, F. Juergens, E., Lochmann, K. et Wagner, S. s.l. : Proceedings of the Seventh ICSE conference on Software quality, 2009.
[43] *UN META-MODELE DE QUALITE LOGICIEL.* BOUKOUCHI, Youness, et al., et al. s.l. : La deuxième journée sur les Technologies d'Information et de Modélisation TIM'14, 2014.
[44] *A MetaModel for Quality Software Based on the MDA Approach.* BOUKOUCHI, Youness, et al., et al. 3, s.l. : International Journal of Computer Science and Information Technologies, 2014, Vol. 5.
[45] *Etat de l'art sur le développement logiciel dirigé par les modèles.* Diaw, Samba, Lbath, Rédouane et Bernard, Coulette. s.l. : TSI-L'ingénierie dirigée par les modèles, 2009.
[46] *Vers des patrons de méta-modélisation.* MARVIE, Raphaël. s.l. : Laboratoire d'Informatique Fondamentale de Lille, 2003.
[47] Gandon, Fabien et Schreiber, Guus. *RDF 1.1 XML Syntax.* s.l. : W3C Recommendation, 2014.
[48] BAZARGAN, Kaveh. *Le rôle des ontologies de domaine dans la conception des interfaces de navigation pour des collections en ligne de musées.* s.l. : Université de Genève, Suisse, 2004.
[49] McGuinness, Deborah L. et Harmelen, Frank van. *OWL Web Ontology Language.* s.l. : W3C Recommendation, 2004.