An Approach to Software Quality Model Based (SQuaM): (QMGenerator a Framework Supporting this Approach)

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

[1,2, 3] Ben M'Sik Faculty of science, University Hassan II, PB 7955 Casablanca, Morocco

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

Abstract: ***The computing technologies and IT platforms are in exponential continuous growth. This growth generated several problems; hence, the necessity of following an approach of software quality for needs and requirements, avoiding the problems engendered in the conception, and the development and the use of the software. In the last years, several approaches of software quality were proposed (IEEE, PDCA, CMMI,…), these approach differ from their vision to quality, the level of intervention in the life cycle of software and the nature of activities are required to be ensured and control software quality. So, the things that they are common between them are: its processes are completely repeated for each software without exploiting the cumulative experiments during the cycles of previous projects. For this one, which is based on the architecture driven model (MDA).We propose in this article, a process based on software quality models, named SQUAM (software quality model based). Moreover, we propose the framework QMGenerator that a model generator of software quality dedicates to our process SQUAM.***

Keywords: MDA, Quality Model, Meta-model, Metric, Software Quality Approach, Model generation

1. Introduction

The computing technologies and IT platforms are in exponential continuous growth. This growth generated several problems; hence, the necessity of following an approach of software quality for needs and requirements, avoiding the problems engendered in the conception, and the development and the use of the software. In the last years, several approaches of software quality were proposed (IEEE, PDCA, CMMI,), these approach differ from their vision to quality, the level of intervention in the life cycle of software and the nature of activities are required to be ensured and control software quality. So, the things that they are common between them are: its processes are completely repeated for each software without exploiting the cumulative experiments during the cycles of previous projects. For this one, which is based on the architecture driven model (MDA).We propose in this article, a process based on software quality models, named SQUAM (software quality model).Moreover, we propose the framework QMGenerator that a model generator of software quality dedicate to our process SQUAM. This paper is organized as follows: section 2: the concepts of software quality and Model-Driven Architecture (MDA). Section3: our approach to software quality SQauM. Section4: Our QMGenerator Framework. Finally, Section 5: conclusion and description of the future work.

2. Software and model Quality

2.1. Software Quality

Software quality is the most important element in development of the software because the quality could reduce the cost of maintenance, software testing, etc. Quality has got very different meanings for customers: users, managers, developers, testers…etc. Many institutes and organizations have their own definitions of software quality, and also modals quality. Below some definition 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 (IEEE Std 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 could be defined as: An effective process for software development, which applied in a manner that creates a useful product and deliver measurable value for those who produce and t use it.

2.2. Software Quality process

An approach quality is the process to establish a system of quality and to ensure a continuous improvement of software quality. It includes activities, tasks and actions connected between them and aiming at the improvement and the management of software quality.

2.3. Model and Metamodel

2.3.1. Our Software quality Metamodel

The model-driven architecture (MDA) [15] is an approach to software development which emphasizes the use of models. It places the models in the center of the processes of engineering software. It is a shape of generative engineering, in which, all part of a computer application is generated from models (Figure 1).

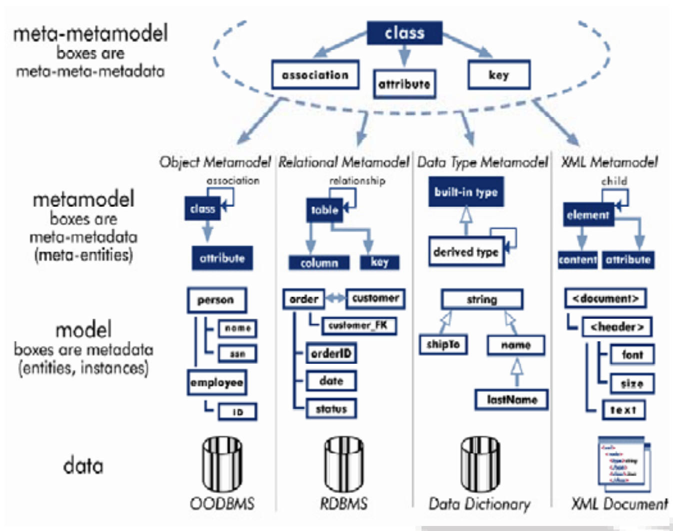


Figure 1: Process model developed by OMG

According to the MDA approach, a model is written in the language of its metamodel. A meta-model describes the concepts of language, the relationship between them, the mapping rules, and transformation of model elements to comply with the rules of the domain.

In this research, we base on our reference meta-model presented in [2]. It is a general meta-model allows to generate all the models of software qualities as well as to make personal models quality; according to the requirements expressed by the engineers of the software quality. Our Meta-quality model (Figure 2) is divided into hierarchical elements. It structures the quality into three levels: view, characteristic, and metrics, knowing that the characteristics can be decomposed into several sub-characteristics and so on.

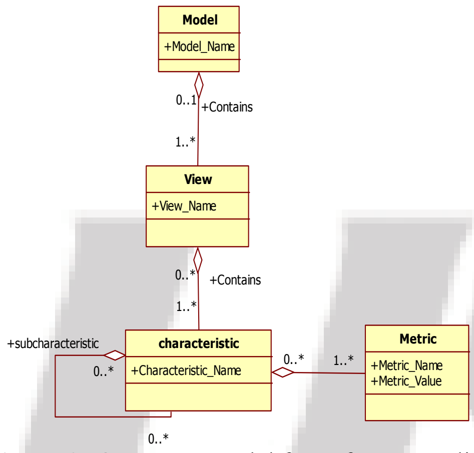


Figure 2: Our Meta-model for software quality

2.3.2. Softawre Quality Model

ISO/IEC 9126-1 defines a quality model as a "framework which explains the relationship between different approaches to quality". Quality models decompose in hierarchical elements. An approach to quality is to decompose quality in Factors, Sub-factors, and criteria. Evaluation of a program begins with measuring each quality criteria with numerical value from metrics. Then, each quality sub-factors is assessed using their criteria. Finally, numerical values are assigned to quality characteristics from their quality sub-factors [2]. Ci-dessous quelques modèles et méthodes de la qualité logicielle [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.
- *BOEHM Model:* Boehm's model is similar to the model of McCall.It also presents a hierarchical model quality structured around four levels: high-level characteristics, intermediate-level characteristics, primitive-level characteristics and metrics.
- *ISO 9126 and 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 decomposed 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 developed from the requirements contained.

3. Software Quality Models Based

3.1. Our approach of Software Quality

We propose in this section our quality approach called "process-based software quality models" (Squam for Software Quality Model based), it 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.
Our approach 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 provide at any time the status of software quality using metrics measured.
Among the strengths of our approach:

- SQuaM transforms users needs in to a quality model, these needs are structured according to the structure of meta-model (Views, characteristic and metric), and which will be thereafter used easily by users and quality engineers in a new project.
- SQuaM presents a model of quality independent of all technical specifications related to a platform, Malthus, and SQuaM guaranteed the portability of the models between several platforms.
- SQuaM is applicable throughout life cycle of software; it is simply enough to develop the adequate metrics for every phase of cycle (Conception, development, test, etc.) .And to implement them according to platforms.
- SQuaM allows a continuous improvement of the modals quality to satisfy the variable needs for the users, and guaranteed a maximal quality.
- SQauM allows the accumulation of the experiences in every life cycle of software. In every cycle, a model is improved then referenced in a database of models; this database supply: on one hand, to the users the possibility of Defining Easily Their Needs and requirements. And at

the other hand, the experts to allow for reduce the cost of modeling, development, and implementation of the metrics.

3.2. Cycle of SQuaM approach

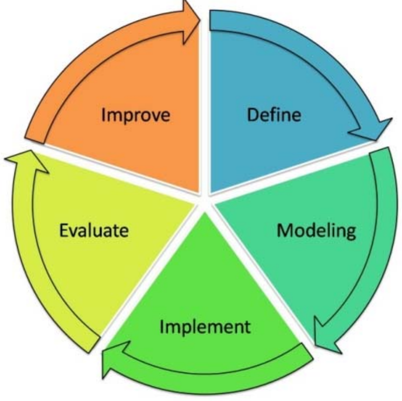


Figure 4: Five steps of SQuaM process

Our approach, based on software models quality. It consists of five steps (Figure 4).Each step, is related to the other step. Its implementation Allows to measure 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 (Figure 5), and in every step there is a set of the activities interacted between them by flows of object: Model, Report, code, ... (Figure 6).

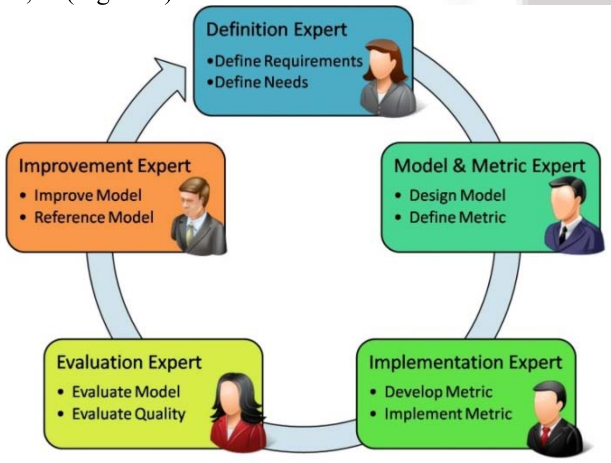


Figure 5: The five experts involved in the process SQuaM

- The first step is "Define" :
  In this step, we define the requirement and the needs of the users' software product. It is a critical stage in this process. It requires a professional definition to requirement and the needs of these users (customers, responsible of SI…etc).This step, is possibly done by some experts of the definition of needs, and requirement. These experts must be written with their experience in a professional way.
- The second step is "Modeling"
  In this step, we do the model quality of requirements, which is defined in the previous step. These requirements are transformed into a model of software quality under the following structure: Views, characteristics, metrics, this step is done by an expert of quality models and metric.
  According to the needs and requirements which is defined, the expert chooses:
  - A model of existing quality: McCall, ISO9126, etc..
  - Improved model quality from the database of models
  - Or to design a new quality model for the specific requirements of the users.

  The generating model quality is in accordance with our meta-model, which is , presented above.

  Also In this section, the proper metric measure will evaluate the model quality by values that are chosen. Each characteristic (or sub-characteristic) is associated with one or more metrics.
- The third step is "Implement": this step, consists of implementing the functions of the metric calculation in the platform. These functions are developed by expert of development and implementation according to the specificity of platform (Java, PHP, Dot.net…) .And which is integrated in the platform. Metrics receive values by their function of calculate which are already implemented in the platform. In this way; we measure the software quality throughout software life cycle.
- The fourth step is "Evaluate'', in this step, the measurements made by the metric, is utilized to evaluate, and analyzed the software quality in accordance with the model quality that is defined above. This assessment provides useful and relevant information to software quality that will help users and engineers of quality make clear decisions on the quality throughout the life cycle of software.
- The fifth step is "Improve"
  In this step, is:
  - Firstly, improving our model quality is to avoid problems encountered in modeling, or bad combination of some characteristics to suitable metrics, and also to generalize the model to prevent, and meet the future needs of users.
  - Secondly, referencing our model in a database to be usable in the software future projects. Instead of starting the cycle from zero, we exploit these improved models to be referenced in the database, and avoid repetitive work (requirements definition, design, metric and calculation functions).

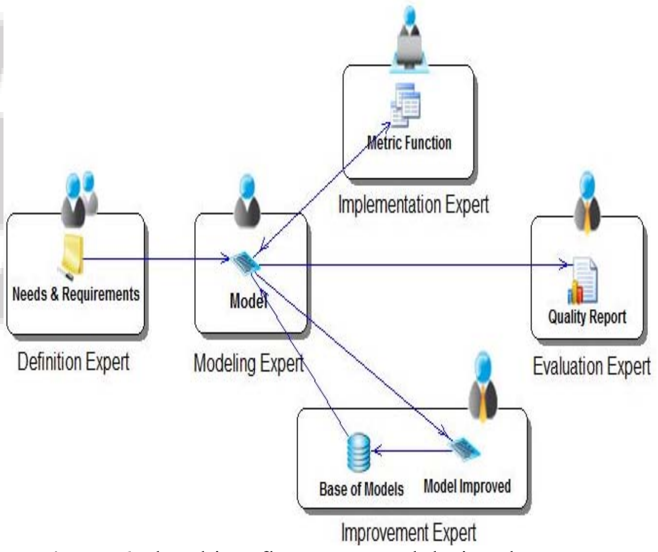


Figure 6: the object flow generated during the process SQuaM

4. QMGenerator: Quality Model Generator

4.1. Architecture and Features

The QMGenerator is a design tool for software models quality, based on SQuaM process. It allows from our meta-model to instantiate models quality in XML file (Figure). QMGenerator defines: plans, characteristics, sub-characteristics, and metrics. The generated file, contains only the metric syntax and the basic information (name, description, type of value, etc..).Without, their values which will be added later.

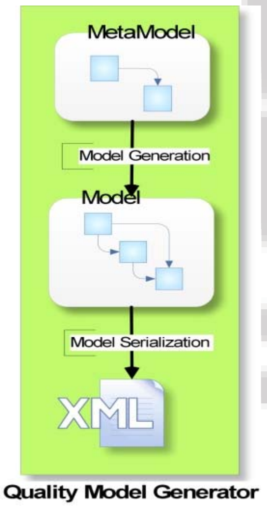


Figure 7 : processus de génération des modèles de qualité

4.2. QMGenerator is a Framework for SQuaM

In fact, QMGenerator is more than one tool of generation of models quality .It is a tool designed for the SQuaM process. It is a Framework that provides: definition, modeling, measurement and evaluation of the software quality. It calls upon the experts of SQuaM throughout his process (Figure 8). It uses XML technology, which could be easily ,used to generate ,and communicate models and metrics quality between all the stakeholders that is responsible for the quality without considering the diversity of platforms. In this business process (Figure 8), we present the activity of each expert in our Framework QMGenerator:

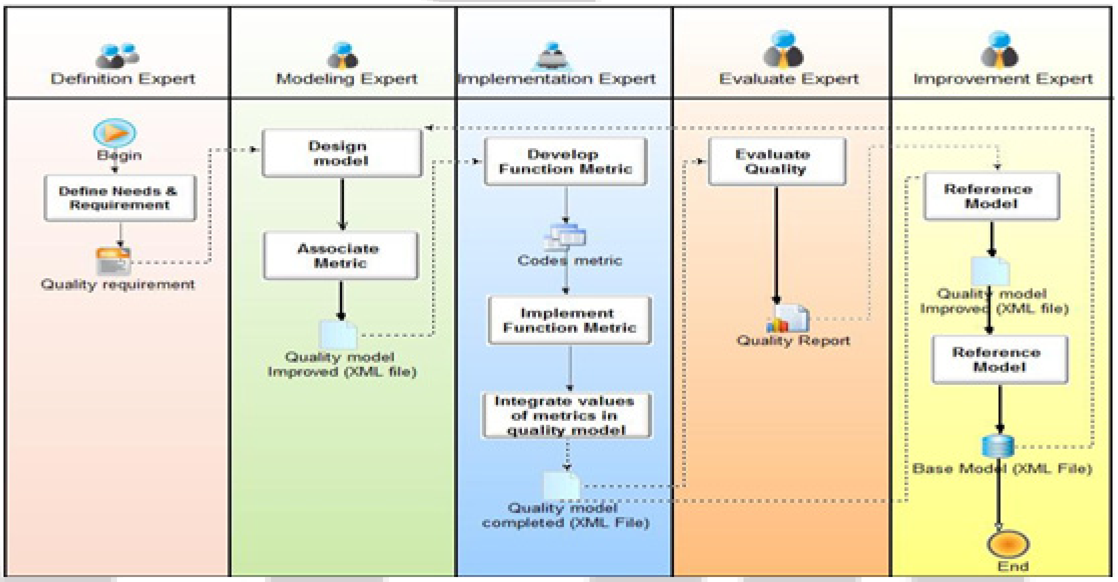
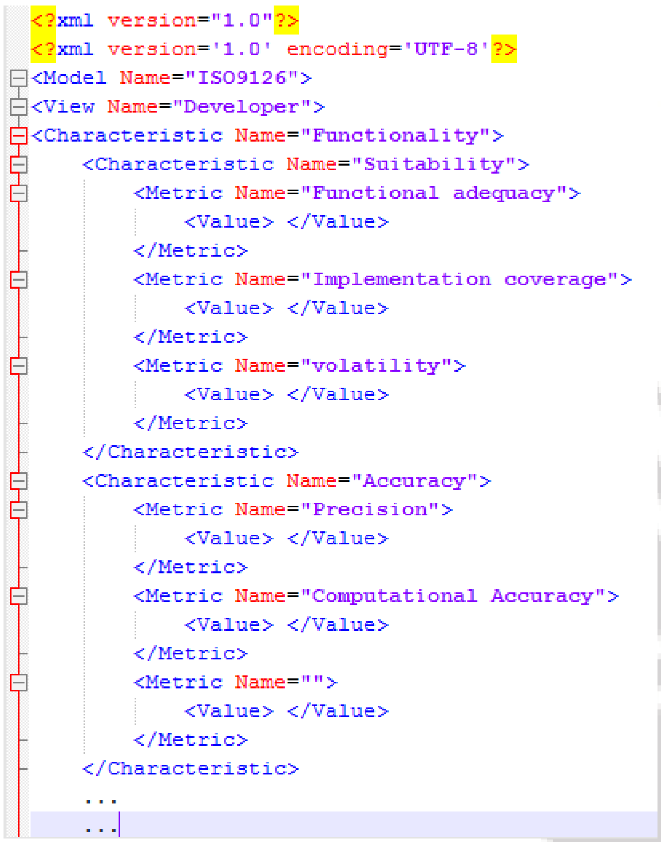


Figure 8: Le processus métier de QMGenerator

- Definition Expert: its role is to define in a professional way the needs and the requirements of the users and the engineers of the quality.
- Modeling Expert: its role is to describe the model of quality; from the needs and the requirements that we have already mentioned. It consists to be determined the designs, characteristics, sub- characteristics, and then assigns each sub-characteristic the appropriate metric. He consults the database of models to simplify its mission.
- Expert of implementation: its role is to develop the functions of calculations for each metric. It must be developed these functions in several language of programing (Java, C, php,…).This for ,being easily implemented them with any platform. It must make them ensure that the functions associate the values of metric in XML file (Figure 9).
- Expert evaluation: its role is to analyze and evaluate software quality at any time depending on the model quality previously generated, and ultimately, make decisions using a dashboard presented by QMGenrator.
- Improvement Expert: its role is to improve the models which is generated to satisfy the future needs , to exceed

the obstacles, and at the same time to make the referencing of this model ,which is improved, in the database of the models.



```
<?xml version="1.0"?>
<?xml version='1.0' encoding='UTF-8'?>
<Model Name="ISO9126">
<View Name="Developer">
<Characteristic Name="Functionality">
    <Characteristic Name="Suitability">
        <Metric Name="Functional adequacy">
            <Value> </Value>
        </Metric>
        <Metric Name="Implementation coverage">
            <Value> </Value>
        </Metric>
        <Metric Name="volatility">
            <Value> </Value>
        </Metric>
    </Characteristic>
    <Characteristic Name="Accuracy">
        <Metric Name="Precision">
            <Value> </Value>
        </Metric>
        <Metric Name="Computational Accuracy">
            <Value> </Value>
        </Metric>
        <Metric Name="">
            <Value> </Value>
        </Metric>
    </Characteristic>
    ...
    ...
```

Figure 9: The XML file generated by framework QMGenerator

5. Conclusion

Our process is based on model quality (SQuaM) and its Framework QMGenerator is a new vision to software quality, it is a process based on model to see the software quality, and which presents a perfect process for the development of the software, and which guaranteed software quality which is requested by the engineers of the software engineering. This work doesn't have only one explanation to our process, but we have to explain in detail the process for each step,and to provide practical case in the utility of our process in the field of business of software development as well.

References

[1] Comparative Study of Software Quality Models, Youness Boukouchi, Abdelaziz Marzak, Habib Benlahmer et Moutachaouik Hicham, the International Journal of Computer Science Issues, April 2013.

[2] A MetaModel for Quality Software Based on The MDA Approach, Youness BOUKOUCHI, Adil Khammal, Abdelaziz Marzak and Hicham Moutachaouik, International Journal of Computer Science and Information Technologies(IJCSIT) ISSN: 0975–9646 , May 2014.

[3] A Quality Model for Design Patterns. Khashayar Khosravi, Summer 2004

[4] IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications

[5] IEEE Std 1061™-1998 (R2009) IEEE Standard for a Software Quality Metrics Methodology

[6] IEEE Std 982.1™-2005 (Revision of IEEE Std 982.1-1988) IEEE Standard Dictionary of Measures of the Software Aspects of Dependability

[7] ISO/IEC TR 9126, Software engineering –Product quality – Part 1,2,3. 2002-03-15.

[8] "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

[9] "A Metamodel for Specifying Quality Models in Model-Driven Engineering", ParastooMohagheghi and VegardDehlen, Engineering Research Institute, University of Iceland , 2008

[10] 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

[11] 12 Steps to Useful Software Metrics, Linda Westfall,The Westfall Team, westfall@idt.net

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

[13] 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.

[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

Author Profile

Youness BOUKOUCHI was born in Casablanca City, in 1980. He received the bachelor's degree in Computer Science from Hassan II-Mohamadia University, Ben M'Sik Faculty, Casablanca, Morocco, in 2005 and the Master's degree in Internet and Computer Engineering from Hassan II-Ain Chok, AinChok Faculty, Casablanca, in 2010. He is currently pursuing the Ph.D. degree in Software Quality models, HASSAN II-Mohamadia University, Ben M'Sik Faculty Casablanca, Morocco. He teaches design and software programming in several institutes and he is a consultant and software developer. His research interest includes the software quality, Model Driven Engineering, and Software Engineering.

Adil KHAMMAL received his Degree in Informatics Engineering from the Saint-Petersburg Electro technical University "LETI" in 2002. He later prepares his PhD degree in the university Hassan 2 MOHAMMEDIA faculty of science Ben M'SIK (FSB) since 2011. He managed many projects in software development and Information system. Currently, he is a head of the division of the continuing training at the national airports authority. He is teaching Software Engineering and Urbanization of Information Systems at the International Academy Mohammed VI of Civil Aviation (AIAC). His research interest focuses Model Driven Engineering, agility systems and modernization and Business process management (BPM).