An Approach for Enhance the Software Quality Based on Quality Model

Bharat Bhushan Rohila¹, Aarti Gautam Dinker²

Abstract- The software development process enforces major effects on the quality of software at every development stage; therefore, a mutual goal of each software development phase concerns how to increase software quality. Software quality estimate thus aims to evaluate software quality level periodically and to specify software quality problems early. In order to obtain high-quality software products, the characteristics and costing of quality during the software development process is of crucial importance. The objective of this paper is to propose a quality model for integrating some quality attributes in software development. Model based which is based on ISO/IEC 9126 Quality Model, is used to generate a new approach for software Quality. Analytic Hierarchy Process (AHP) is used to evaluate an improved hierarchical quality model for software development.

Keywords— Crosscutting Concern, Quality Attributes, Software Quality Models

I. INTRODUCTION

Many software organizations have been nervously seeing for actual ways to enhance the quality of software products. The absences of software quality has significant costs both to sellers who face unhappy customers, loss of market share and modify of rejected systems and to shoppers who get faulty systems that miscarried to meet their goals. Software industrial is related to the improvement and evolution of large, composite and critical software-intensive system. These organizations are expected to be more flexible, scalable and recyclable. In order to gain these objectives, development techniques that backing abstraction and modularization in software development system can be useful. Software Modularity perfect traditional abstraction is basic for developing complex modern systems - specifically software and software-intensive systems. The demand for quality has been part of human behaviour for a long time, but the quantification of quality and establishment of formal quality standards are a 20th century phenomena [1, 3].

Practitioners, Researchers and Developers have proposed several metrics and quality models [2] In general, the expert’s definitions of quality fall into two categories: Level one quality applies to products or services whose measurable characteristics satisfy a fixed set of specifications that are usually numerically defined. Level two quality products and services need only satisfy customer expectations [3].

The requirement of a quality model for application development. We planned a quality model as well as delivered many guidelines for the Quality Assurance organization to follow for offering a high quality application to end-user and increase user satisfaction through quality product.

II. QUALITY MODEL

A quality model is a plan to better explain of our observation of quality. Many presented quality models can forecast fault-proneness with realistic exactness in certain contexts. Another quality models attempt at calculating many quality characters- tics but fail at providing realistic correctness, from lack of data mainly. We faith that quality models must estimation high-level quality features with great correctness in terms well-known to software engineers to support maintainers in judge programs and thus in predict maintenance effort. Such quality models can also help developers in building best quality programs by juicy the relationships between internal attributes and external quality characteristics clearly. We earnings a less “quantitative” method than quality models counting, for example, numbers of errors per classes and linking these numbers with internal attributes. We favors a more “qualitative” approach linking quality characteristics related to the maintainers’ observation and work directly.
III. THE PROBLEM

A. All sub-characteristics equal in affect software characteristics

Quality models define the relation between quality faces and sub-characteristics. However, the impacts of quality sub-characteristics on characteristics are not parallel. For example: Insatiability and Adaptableness are two sub-characteristics related to Portability. The question is: If we assess the value of Insatiability as A and the value of Adaptability as B, then is the value of Probability equals to A+B.

B. Main concepts of quality are missing

Quality does not distribute in specific part, when we talk about software quality, we talk about assess entire items which are part of the thought of quality.

IV. OUR APPROACH TO SOFTWARE QUALITY EVALUATION

Some solutions of the above mentioned problems, we deal with the some steps necessary to apply our approach to software quality estimation, which solves some of the open issues.

Step by Step

The following processor highlights the main ideas to appliance software quality assessment while under human necessities.

Step1: Choose Category of People. We must select at least a person from the class of people which our software judgment will be implement for, example: Programmers, End-user

Step2: Recognizing Sample Program. We must choose simple programs (BP) to be estimated as sample evaluation set of our model.

Step3: Structure a Quality Model. The method of structure a quality model break in two main tasks generally:
- Selecting a super-characteristic.
- Selecting and establishing characteristics related to super-characteristic.

Our case study, we consider design patterns especially as link between internal attributes of programs, external quality features, and software engineers.

Step4: Human Evaluation. The small group, or at least one person from the group, must aspect in the program or product and evaluate the quality features we defined in our quality model, the estimation could be in form of numerical value or different levels on a Likert scale.

Step5: Computing Software Metrics over BP. By using software metrics we evaluate BP numerical values related to software internal features.

Step6: Machine Knowledge Tools. As machine learning algorithm generate the relation between human evaluation of software quality (result from Step4) and value of software metrics (result from Step5).

Step7: Computing Software Metrics . Software metrics are used to assess the values of internal attributes over the EP in the same way as they were for the evaluation of BP.

Step8: Adapting Metric. By using fraction over the values from Step7 and Step5, we can related the numerical values of Step7 with those of Step5. The resulting method will be used for relation evaluation:
- Phase1. Decision the Max and Min value of each metrics in EP.
- Phase2. Discovery the Max and Min value of same metrics we were calculation Phase1 over the BP.
- Phase3. Seeing the ratio for the values from Phase1 plus values we have in RULE, we shape a new RULE compatible with EP.

Step9: Software Evaluation. Now, we can estimate other programs (EP).

V. PROPERTIES OF QUALITY MODEL

A. Flexibility

A quality model should be flexible because of the context dependency of software quality. There are several quality contexts: industry context, obligation context and process context.

B. Reusability

The need for earning from past experience has led software progress in the way of product lines. In order to assure the development of high-quality products, quality models should also follow this pattern. Instead of “re-inventing the wheel” every next software project, the quality models should allow the recycle of quality experience packaged in the existing quality models through other projects.
C. Clearness

A quality model should provide the rationale of how certain characteristics are connected to others and how to identify their sub-characteristics. Clearness of a quality model does also mean that the meaning of the features and relationships between them are clearly (unambiguously) defined. People involved in model improvement and application should understand it to expand knowledge from it as well as to identify severances or contradictions among quality characteristics.

VI. PROGRESSIVE DEVELOPMENTS OF QUALITY MODELS

There are number of software quality models in software engineering workings. The quality Models are disjointed into two categories: Hierarchical quality Model and Non-Hierarchical quality Model. In this paper, only hierarchy quality models are definite. Each one of these quality models involve of a set of high quality features/factors and sub features/sub-factors.

Two principal models were planned one after another. In 1977, McCall et al. [4] proposed a quality model called McCall’s Software Quality Model and it is also called Classical Quality Model. McCall’s Quality Model was later better and revise as the MQ Model by Watts in 1987. Next year in 1978, Boehm et al. [5] proposed another quality model using McCall’s quality model, called Boehm’s Software Quality Model.

Three quality Models (in 1987, Evans & Marciniak’s Quality Model and FURPS Quality Model and next year 1988, Deutsch & Will’s Quality Model) were proposed. Among these quality models, FURPS Quality Model [6, 7] is more standard because it is first industrial approach based quality model, offered by Hewlett-Packard (HP).

Till 90’s, number of software quality models were planned. This led to lot mistake among practitioners, which model to actually follow. Therefore, International Group for Standardization/International Electro-technical Commission (ISO/IEC) began to develop and standardize a new quality model considering the entire repository of various quality models proposed so far. In 1991, ISO/IEC proposed a quality model, called ISO/IEC Quality Model. Later on, the name was changed to ISO/IEC 9126 Quality Model [8, 9, 10] since ISO 9126 was part of the ISO 9000 standard.


All software quality models were derived based on either legacy software or object-oriented software. An application contain of Class and method. The quality assessment of execution of Class modules is measured by above defined quality models. The quality of class sections cannot be by the above debated models and software quality model for assessing the quality of projects developed using method want to be developed.

So, integrating some new faces/factors and sub-faces/sub-factors of software Quality Model as a base ISO/IEC 9126 Quality

VII. SOFTWARE QUALITY MODELS BACKGROUND

Many software quality models were proposed, in order to estimate different types of software products. This section presents the most popular quality models.

A. McCall’s Quality Model

One of the most eldest and renown precursors of today’s software quality model composed by McCall et al. [4] also identified as the General Electric (GE) Model originates from US Air Force, the Rome Air Development Centre (RADC), to develop the quality of software products. Main purpose of this model is to estimate the relationship between external factors and product quality criteria. The McCall’s Quality Model is separated into top three major viewpoints: Product Operation, Product Revision and Product Transaction. All the three major perspectives are divided into some external factors which define the external view of software system (i.e. User View) and all the external factors are divided into 23 quality’s criteria which describe the internal view of software system (i.e. Developer View). Quality’s criteria related with a set of quality metrics are defined and used to provide a scale and method for measurement [12]. The main contribution of this quality model is the relationship between quality issues and metrics. However, the quality model does not revenue into account the quality aspect of various functionalities of the software product.
B. Boehm’s Quality Model

The software quality model was developed by Boehm et al. (1978), adding emphasis on the maintainability for software product into McCall’s Quality Model is called Boehm’s Quality Model [5]. The useful of this model is to describe the current parallel deficiency of McCall’s Quality Model that automatically and quantitatively estimation the quality of software product. Hence, characteristics of Boehm’s quality model are represented in hierarchical form to manage total quality. The strength of model is mostly invented for common sense reasons, rather than on experiential evidence of their accuracy as a model.

C. FURPS Quality Model

All the models proposed so far were developed by academicians as a research activity only. So far industry had not shown any interest in the quality issues of the software development procedures. Robert Grady and Hewlett-Packard are the first one to propose model with the industrial approach. This quality model is known as FURPS Quality Model [6, 7]. The model directed at refining the management of software growth processes by software industry. FURPS Quality Model includes top five level attributes.

D. ISO/IEC 9126 Quality Model

Since, the number of software quality models were proposed, the confusion occurred and new normal quality model was essential. Thus, ISO/IEC Joint Technical Committee (JTC) [8, 9, 10] started to develop model using the necessary consent and motivate standardization worldwide. Further, name was changed to ISO/IEC 9126 Quality Model. The model is an extension of previous work did by McCall (1977), Boehm (1978) and FURPS (1987) etc. ISO/IEC 9126 quality model is divided into two perspectives (i.e. first is External & Internal Quality and second is Quality in Use) for evaluating the quality of software products.

The defined features in external & internal quality perception are applicable to each and every type of software products. Though, ISO/IEC 9126 quality model reasonably covers most of the quality faces, and sub faces, the product perspective are taken as external and internal quality. The model did not take into account the reusability feature.

E. Dromey’s Quality Model

Dromey’s quality model [11] states that every software product has its own process evaluation. So, there are some dynamic ideas required for process modeling. Hence, Dromey proposed a software quality model in 1995 called Dromey’s Quality Model to adding Reusability and Process Maturity as characteristics in ISO/IEC 9126 Quality Model. The main objective of this quality model is to obtain a model in broad area for variety of application. Dromey’s quality model is associated with reliability and maintainability. So, it is typical to judge, that model is feasible before the software system is operational in development area or not.

VIII. A NEW APPROACH FOR SOFTWARE QUALITY

All Software quality model mostly developed for Object Oriented software. This approach for Software Quality is an extension of ISO/IEC 9126 quality model. Four sub-characteristics are included under different characteristics of ISO/IEC 9126 quality model. But this model also lacks some characteristics/factors and sub-characteristics/sub-factors which is important for applications.

We approach a first quality model for application to integrate Reusability into Functionality, complication into Usability, Code-reducibility into efficiency and Modifiability into Maintainability. This approach is useful to enhance the quality of any application in Software Engineering.
Table of Quality attribute

<table>
<thead>
<tr>
<th>Quality Type</th>
<th>Product Perspective</th>
<th>Characteristics</th>
<th>Sub-Characteristics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maintainability</td>
<td></td>
<td>Analyzability</td>
<td>Changeability</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Stability</td>
<td>Testability</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Modifiability</strong></td>
<td></td>
</tr>
<tr>
<td>Portability</td>
<td></td>
<td>Adaptability</td>
<td>Replace-ability</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Recoverability</td>
<td></td>
</tr>
<tr>
<td>Efficiency</td>
<td></td>
<td>Time behavior</td>
<td>Resource behavior</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Reducibility</strong></td>
<td></td>
</tr>
<tr>
<td>Usability</td>
<td></td>
<td>Understandability</td>
<td>Learn-ability</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Operability</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Complication</strong></td>
<td></td>
</tr>
<tr>
<td>Functionality</td>
<td></td>
<td>Suitability</td>
<td>Compliance</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Accuracy</td>
<td>Interoperability</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Security</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Reusability</strong></td>
<td></td>
</tr>
<tr>
<td>Reliability</td>
<td></td>
<td>Maturity</td>
<td>Fault Tolerance</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Recoverability</td>
<td></td>
</tr>
</tbody>
</table>

 IX. CONCLUSIONS

Many investigations have been tackling by scholars to examine the effect of on non- a characteristic for software development an inspiring development of quality models has taken place over the last times. These efforts have resulted in many successes in research and practice. As contexts proposed model are applied successfully in practice. The developments in quality definition models even led to the standardization in ISO 9126 that is well known and serves as the basis for many quality management approaches.

Every proposed model required evaluation. To evaluate the proposed quality model for applications, Analytics Hierarchy Process (AHP) approach could be used which addresses uncertainty and Imprecision in evaluation during pre-negotiation stages. In the comparative judgments of characteristics based on decision maker with the help of different approaches.

REFERENCES


