Quality Category Matrix to Ensure the Quality of Software Product

Article *in* International Journal of Computer and Communication Engineering · July 2012

DOI: 10.7763/IJCCE.2012.V1.46

CITATION
1

READS
1,476

3 authors, including:

Nadeem Majeed
University of the Punjab
40 PUBLICATIONS **580** CITATIONS
SEE PROFILE

Ali Javed
University of Engineering and Technology Taxila
186 PUBLICATIONS **4,595** CITATIONS
SEE PROFILE

Quality Category Matrix to Ensure the Quality of Software Product

Samina Jan, Ali Javed, and Muhammad Nadeem Majeed

***Abstract*—This research work is to describe some issues and their solutions with respect to quality management of software products. Software is the only type of product which is famous for the internal complexity and providing a quality in software products is more difficult. So Quality should be managed, at the beginning stage of the Software Development Life Cycle (SDLC). In general, No one can give guarantee that a product is a quality product, until all the modules, components of product are tested by the teams and until they approved that the product is high-quality product. While doing integration of components, problems can comes, so iterative approach of Software Quality implementation and testing required during Integration, which should ensure the quality throughout the development process of product. In this paper I proposed an iterative solution for performing testing for ensure the quality with in software products. 'Quality Category Matrix' is proposed for monitoring the quality internally during each artifact development. Regardless the fact that in world many standards (ISO 9126), process improvement practices (CMM) and Models (MC call's FURPS) exist for the quality assurance of software products. But existence and implementation of quality in the software products is still an issue in software development. Professional software Developers are still looking for better solution for putting into practice new approaches to ensure the quality.**

***Keywords*—Quality, artifacts, category .**

I. Introduction

Software development is combination of artifacts, and composite, invisible, changeable nature are the characteristics of each artifact, so ensuring the quality of whole product is a critical activity. Quality has different attributes; any product cannot have all attributes so while developing a product, the quality attributes demanded by a customer should be at top Priority level. Higher quality products in software can be attained if one maintained the three main constrains of quality which are the scope of the product, product cost and product time.

Fig. 1.

Fig. 1 shows that quality is defined under the 3 main constraints. Unluckily, the majority of the developers have to compromise on these three main constraints. With the product quality constraints, the quality factors consist of efficiency, expandability, flexibility, reliability, integrity, maintainability, traceability, portability, interoperability, correctness, usability, reusability and supportability. Issues occur when while achieving the quality attributes, many developers by-pass the constraints. Therefore quality management becomes a tricky task to be undertaken. Some other issues concerning software quality begin when development teams (including testers) try to join together quality for testing, they done these activities at the later stages of development so they increase the cost of development. The proposed research paper encompasses the assurance of different quality factors in the software product with minimum trade-off among the scope of product, cost or time of product determined to attain the product.

Software quality management consist of two main aspects, first one is to ensuring quality of software product and second one is make sure the processes used for the development of software are followed or not. To improve software development processes, CMMI (Capability Maturity Modal Integration) was introduced; now-a-days CMMI is an international standard for measuring the maturity level also capability level of software development organizations. Likewise, Crosby proposed prevention approach. His concepts were that any things should be right at first time so no defect will exist in the product. How one can develop things right? First establish the requirements then measure that requirement (feasibility study and elicitation of requirements), then perform development with the assessment of the requirements. M. Juran proposed fit for use approach, for accomplishment of fitness approach, Juran proposed 'Juran trilogy'. He wrote that quality management included three activities which are interconnected. These activities are quality planning (QP), quality control (QC) and quality improvement. Fitness is totally depending on the customer requirement, if the customer is satisfied with the product, so the product is fit.

One main thing should be consider that attributes or factors of the quality to be acquired by the product on demand and the processes follow during development also conforms the quality. Ian Somerville [1] categorized the activities of Software quality management software in three major conditions:

The first one to be quality assurance (QA) procedures and standards established at the organizational level to ensure quality. The second is the quality planning performed at the project level. In this category the standards, Process and procedures are chosen for a specific project should be defined under the QC activity and customized when required. A third part is the quality control (QC) now Quality is at the organization team level so throughout the QC assurance required for the evaluation of implementation

Manuscript received March 8, 2012; revised May 11, 2012.

The authos are with Department of Software Engineering, Faculty of Telecommunication and Information Engineering, University of Engineering & Technology Taxila (e-mail:1ifsami@gmail.com).

of standards whether they are followed or not.

The initial models of quality such as McCall, Boehm, FURPS and ISO 9126 are limited to measure of external software characteristics such as reliability, maintainability, portability and functionality which do not consider other necessities such as conformance of user requirements and expectation. Software quality was more on customer satisfaction and software correctness was not sufficient to be declaring as good quality without satisfaction by the users [2].Nigel Bevan [3] analyzed the quality of the product to distinguish internal and external quality. Nigel Bevan wrote in their documents that software product requirements must be display in the form of figures that can be calculated at what time system is used in context provided, like, measurement of satisfaction, efficiency and effectiveness. He says that the reason behind the designing of an interactive system is that the system should realize the requirements of customer for the purpose of providing the 'Quality in use'. Software product consists of internal attributes, which find out the quality in specific context. While working on the quality of software product, the achievement of 'Quality in use' should be the main purpose. External quality can only evaluated by whole team / software which the software product is a part. It is essential that software internal quality attributes are linked to external software quality requirements. Thus the under development software products quality nature can be evaluated with respect to ultimate system 'in-use quality' requirements. Internal metrics have less value until the proof exist that internal quality attribute metrics are related to external quality. According to Nigel Bevan wrote that ISO/IEC 9126 model supports a range of assessment requirements, like:

- A user or a unit of a business user can assess fitness of a software product with quality metrics in use;
- A buyer can assess a software product beside the standard of the external actions of the value, functionality, efficiency, reliability, and;
- A maintainer to evaluate a software product maintenance using metrics;
- A person responsible for implementing the software in different environments could evaluate a software product using the metric for portability;
- A developer can evaluate a software product against the criterion values based on measurements of either internal quality characteristics.

Many Models are proposed for measuring software quality and that are being in used now-a-days. Dromey[4] built a bottom-up quality model that links the relationship between internal software characteristics and external software quality attributes using the appropriate metrics. Dromey's model fixes some problems of earlier models such as the dependency between quality attributes, and the effect of each attribute on the number of quality attributes of software. In general, Dromey's model identifies the quality in three stages: identify high-level quality attributes, identify the product's components with its quality-carrying properties, and link the quality attributes to the product properties. QMOOD model extends the Dromey's model. QMOOD was proposed by Bansiya and Davis [5] to assess software evolution. QMOOD provides a direct and indirect measurement of the software quality. Design quality in QMOOD is evaluated using ISO 9126 six quality attributes: effectiveness, understandability, extendibility, reusability, and flexibility. The QMOOD model is used to determine the internal and external properties of software components and their relationships. An OO system has many components such as classes, objects, and the relationships between them. Bansiya and Davis selected the OO internal properties, OO quality attributes, and OO metrics to be used in the verification model. The selected properties are: inheritance, encapsulation, polymorphism, abstraction, coupling, cohesion, messaging, composition, design size, and complexity.

Nihal Kececi et al [6] evaluate Goal Tree Success Tree and Master Logic Diagram and Master Logic Diagram (GTST-MLD) as a new software development process model. GTST-MLD is a model for DLC to ensure software quality which is based on the based on the accomplishment of the condition which was set for the security systems of high integrity.

The GTST-MLD based SDLC framework performs following things:

1) Explain how a local change (within one stage) may affect other stages of software development.

2) GTST-MLD represents hierarchically SDLC to incomplete requirements and also discovers missing requirements.

3) is easy to automate equipment, expand and upgrade.

Aziz Derman et al [7] proposed the SCM-prod model provides procedures and guidelines for certifying software product operational in certain environment. Second advantage of this model is the involvement of users in the certification process that justify the confirmation of user expectation and requirement for software quality. With the involvement, the certification process is moving towards user-centered approach.

II. Proposed Approach

Approach presented in this research included two steps: The first step is getting the quality factors and the specified level that show whether they are implementing or not, Levels show with signs □ (for yes) or× (for no) . Next step for ensure quality, requires a report which should be submit. According to the solution, 'Quality Category' report will make.

An intangible thing is difficult to measure, but it is more complex when of software products as the software itself is invisible. Therefore, the project manager has an obligation, for them it is very difficult to ensure quality. QA activity itself required a complete team for performing QA, a dedicated quality manager who leads the team and a proper QA process. Software products do not physically exist so their internal understating is also required especially when checking the, maintaining factors such as accuracy, efficiency, reliability, completeness, usability, maintenance, flexibility, testability, portability, interoperability and reuse approach within the constraints of the intended scope, cost and time.

The proposed approach is that must deal with the scope of the project, cost factor in the project and Duration limit of the project in all artifacts during artifacts delivery. Therefore,

it should be adopt by us the iterative approach to software development and take into account the intangible nature of software. Like all begins with requirement elicitation then planning the event it is QA.

Preparation before testing is supposed to be well-built in the case of performing QA. First the all Development Team should create in development team Project Manager, QC, developers and Designers comes. QA representatives should be concerned with planning of the project. Because from the starting point of the SDLC, if one participate maximum in the QA activity, on other stages of Development, less effort is required.

Software quality management team should participate in the requirements engineering stage of software engineering process as well, so that QA initiate when project start. This action will advantage for the QA of specific product.

Because the software is always a new application and not works on the project itself again, some new features come always with new software. Software has to do with customer agreement. Often we define quality in terms of customer needs, like if we satisfied the customer by making his/her product, so implement quality.

Make a plan for achieving the quality attributes. In general, after the evaluation of project constraints, make a report of review. If estimation shows unbalance conditions like budget is increasing or delivery taking time etc then negotiate with the client. Request the customer that either he increase the budget, reduces the scope of the project or to extend the time limit, after that proceed with the agreement on the quality attributes and limitations.

In proposed solution, category is made on the base of functionality. Separate functionalities (features) are categorized according to their specifications, and again each category the Quality factor will check. Either the required quality level is implemented in the category or not. For ensure the quality, check should perform after completion each stage. Make a report in which matrix will present the checklist for each test. Keep the record of quality attain after each stage testing. Obtain feedback from customers and maintain the report for guidance for other projects. Layout of the Quality Category report should be in the form of matrix that comprises of Categories in horizontal order and Quality attributes in vertical order, and. First classify all the Categories (which contain the requirements) that require the similar quality attributes, and a table for each Artifact. As the following:

TABLE 1.1

Quality Category				
First Development Stage				
	Cat1	Cat2	Cat 3	Cat 4
Q1	✓	✓	×	✓
Q2	✓	✓	✓	✓
Q3	×	✓	✓	✓
Q4	✓	✓	×	✓
Result	Repeat	Done	Repeat	Done

Results:

Mark '□' if quality exists in specified category and × if quality is not existed.

Confirm from each column that whether all quality attributes are implemented (□ sign will show it) or not, after that set the concluding results at last row, another time check that □ is present on the last row. If the result after applying confirming on the last row is □, now approve the 'Quality Category matrix' otherwise do again QA activity unless answer for that matrix is □. At this moment combine conclusion from matrices of each specify stage and again confirm whether quality is in on that results or not, when outcome is □ only then approve the quality, and make quality report. Once approved, team will move to next artifact development. When working at the development of Artifact 2, do again the quality checking method, create maintenance matrix of quality category for artifact 1 to make sure the results concluded from that artifact are still maintained:

TABLE 1.2

Quality Category Maintenance Artifact 2				
	Cat1	Cat2	Cat 3	Cat 4
Q1	✓	✓	✓	✓
Q2	✓	✓	✓	✓
Q3	✓	✓	✓	✓
Result	Done	Done	Done	Done
First Development Artifact				
	Cat1	Cat2	Cat 3	Cat 4
Q1	✓	✓	✓	✓
Q2	✓	✓	✓	✓
Q3	✓	✓	✓	✓
Q4	✓	✓	✓	✓
Result	Done	Done	Done	Done

TABLE 1.3

QA Summary					
First Development stage					
Artifacts	Cat1	Cat2	Cat3	Cat 4	Cat5
Art 1	✓	✓	✓	✓	TF
Art 2	✓	✓	✓	✓	✓
Art. n	✓	✓	✓	✓	✓
Second Development stage					
Artifacts	Cat1	Cat 2	Cat 3	Cat 4	Cat5
Art. 1	✓	✓	✓	✓	✓
Art. 2	✓	✓	✓	✓	✓
Art. 3	✓	✓	✓	✓	✓
Art. n	✓	✓	✓	✓	✓
Third Development stage					
Artifacts	Cat1	Cat2	Cat 3	Cat 4	Cat5
Art. 1	✓	✓	✓	✓	✓
Art. n	✓	✓	✓	✓	✓
N Development stage					
Artifacts	Cat1	Cat2	Cat 3	Cat 4	Cat5
Art. 1	✓	✓	✓	✓	✓
Art. 2	✓	✓	✓	✓	✓
Art. n	✓	✓	✓	TF	✓
Final Results: Approved					

In this approach, quality is not only achieved, but also quality will maintain throughout the development life cycle. It is repetitive approach, so at the end of the each iteration, quality manager will approve the quality at development level (alpha testing) and customer will approve the quality at his/her environment (beta testing will perform). All the test cases, quality assurance documents will finalized at the end of the project.

Obviously quality cannot attain fully, in a project, there are factors that require compromises, it is fact that the compromise is not avoidable, so client should have trust on you, and then take an agreement that trade-off comes while Quality attribute, so it can't perform perfectly as required. Trade-off comes between the time and quality attributes, between scope and quality or it can exist between cost and quality factor.

Below it is mentioned the Trade-off Summery table is mention in which compromises 'TF' summary shows compromises detail which were mention early matrices and include 'Quality Category Trade-off Summary Report' in the final QA document as follow:

TABLE 1.4

Quality Category Trade-off Summary Report				
S. No	Stage	Artifact	Cat	Reason
1	1	1	5	**Scope**
2	N	N	4	**Budget**

Trade-off between quality factors of Software Product should not exceed from 5 % of total quality attributes.

Practical approach towards the outcome and data concerning quality in quality document required at the lower to higher level. Otherwise this solution will remain a hypothetical and may no longer stay.

Application Areas oF Proposed Solution:

Application areas of this proposed approach contain the software development organizations, also suitable for ensure the quality of real time systems also for safety critical systems. The proposed solution ensures the highest possible quality.

This technique will be appropriate to the small level projects also for high level projects which required high investment. Solution can be use while implementing advanced software development approaches like Agile (XP, SCRUM), Incremental, and Lean. At the beginning of the project, more expenses are required, it may required a dedicated complete team for performing testing and Quality assurance activity, at the closing stage, quality will ensure.

With some enhancements, this approach will at some level helpful at global level, version control system may use for monitoring very large level projects quality.

III. Future Work

Possible future work of research contains the classification of the field areas of the project and their relation with each other. The percentages of committed quality factors are different for each specified area.

In future, given solution can enhance at higher levels, and can use in other domain rather than Software, the results come after the implementation of the approach, use as effort evaluation for the project.

So with respect to this article, possible areas of research include the analysis of the Product quality with respect to Domain area, which describes the categories and Quality Factors. Future studies may conclude the quality factors for each specific domain area, also ensure that project is within Scope, time and budget with quality is implementing in it.

IV. Conclusion

From the above discussion, it should realize that the process of attaining quality is not so simple, it takes time. Consider the quality as a goal and for achieving that goal, many questions/ queries comes, and after resolving these issues, quality implementation process starts, so it can be accomplished after a long-long cycle (continue process). The process of quality assurance of product should start from requirement elicitation stage of the project at what time gathering requirement phase starts, till when the product is in maintenance phase after deployment. The main problem which happens in software quality assurance that normally implements quality at the end of the development stage so the quality is automatically offered too costly. For that reason, more money spent for quality achievement at the inauguration of project is better than spending after development of the project. The proposed approach can use also at the start phase of SDLC, and can use till the end process. It's a check list, which works as a monitor on development phases. If development teams monitor the DLC from beginning, money and time will save and performing this act will remain the quality product delivery on time and within budget.

References

[1] Sommerville, *Software Engineering 7/e* , vol. 50, February 2009, pp. 950-957,

[2] P. J. Denning, *What is software quality?* A Commentary from Communications of ACM, January, 1992.

[3] N. Bevan, *Quality in use: incorporating human factors into the software engineering lifecycle*, 1-6 Jun 1997, pp. 169 - 179

[4] G. R. Dromey, "A Model for Software Product Quality," *IEEE Transactions of Software Engineering*, vol. 21, no. 2, pp. 146-162, 1995.

[5] J. Bansiya and C. Davis, "A Hierarchical Model for Quality Assessment of Object-Oriented Designs," *IEEE Transactions of Software Engineering*, vol. 28, no. 1, 2002, pp. 4-17.

[6] N. Kececi and M. Modarres, "Software development life cycle Model to ensure software quality," University of Maryland, USA.

[7] A. Deraman, J. Yahaya, F. Baharom, and A. R. Hamdan, "User-Centred Software Product Certification: Theory and Practices," University of Northern Malaysia, National University of Terengganu, Malaysia.