ISSN:0975-9646
Anandam et al, / (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 2 (4) , 2011, 1651-1654

SOFTWARE REQUIREMENTS & QUALITY APPROACHES THROUGH MDRE AND BESPOKE PROCESS MODELS

[1]Anandam. V, [2]Kailasa Rao. K , [3]Sasidhar Kothuru ,[4]Pradeep V
1. *Computer Science & Engineering, VBIT, Ghatkesar, Hyderabad.*
2. *Computer Science & Engineering, MRCET, Secunderabad.*
3. *Information Technology department, SNIST, Hyderabad.*
4. *Department of MCA, MRCET, Secunderabad*

***Abstract*-Software Quality assurance is an important aspect of product development. Unlike the other engineering disciplines, software quality assurance does not have control [1]. The main objectives of SQA are acceptable levels of confidence, conformance to functional and technical requirements. It assures the acceptable levels of scheduling and budget requirements. This paper describes quality aspects for different types of system in different environments. Quality differs according to the subject as well as environment.**

Keywords: **Product development, Aspect of quality, Market Driven Requirements Engineering, bespoke**

1 INTRODUCTION

Software Quality Assurance is a collection of activities designed to evaluate the process through which the products are developed or manufactured [1].

Kurt F. Fischer identified seven Quality Assurance functions to assure well planning, reporting, and control to affect the development of software products which meet the requirements.

1. Initial quality planning
2.Development of software standardsand procedures.
3. Development of quality assurance tools
4. Conduct of audits and reviews
5. Inspection and surveillance of formal Tests.
6. Configuration and verifications
7. Management of the discrepancy

This paper discusses about aspects of quality for different situations such as Market Driven Requirement Engineering (MDRE) and Customer Driven Requirement Engineering (Bespoke) using different quality management tools. We also discuss about the problems associated with the quality of service of the process management and software quality activities.

2 QUALITY IN MDRE AND BESPOKE

Quality is perceived differently for different system in different environment. Even for same product, different person can have different quality measures. So, it is incomprehensible in nature. We identify the quality for MDRE and Bespoke. MDRE focus on mass market where as BESPOKE focus on the single customer or single organization. Customers judge the quality of the products by accepting or denying the products. Every organization has several customers and they are considered as the stakeholders of the organizations.

Customer acceptance of end product also shows that the product meets the quality. For Bespoke, stakeholders for the product can be easily identified since requirements solely come from the defined customers. Quality is measured by the acceptance of the product by specific customer where as for MDRE stakeholders for the product cannot be easily identified since there are no primary customers and requirements are drawn from the market, competitors, focus groups, key customers, internal developers. In MDRE, there are no defined users so quality of the product is measured on the basis of market share, users' demands, product sales and features adaptation by competitors.

In Bespoke, quality can be measured and attend easily, since it has defined sets of requirements that usually drawn from the specified customers. Where as in MDRE there are no defined sets of requirements and requirements are changing throughout the software development. So, it is very hard to attain and measure quality because there are no specified customers against whom it has to be measured.

Quality of bespoke totally depends on the users' value for the product, user satisfaction and user acceptance. In the Bespoke, quality is considered to be met if products confirms to the specified requirements in given deadline and with allocated budget. In MDRE there are no specific requirements and requirements changes throughout the development lifecycle resulting in hard to conformance of requirements.

Quality is measured through the success criteria of the product within budget and schedule. In Bespoke, budget and schedule can be fixed and it can be measure easily at the end of the project assuring all the requirements have met and quality can be improved during maintenance phase. But in MDRE, there are frequent changes in requirements throughout the development lifecycle thus quality is hard to measure in terms of budget and schedule. The quality is measure in terms of budget and schedule. The quality is measured in terms of sets of release planning having Alpha and Beta testing.

In Bespoke quality can be increased with the direct involvement of customers in design phase whereas due to lack of key customers in MDRE.

Bespoke has defined period of schedule, defined numbers of requirements and resources, so cost of quality can be easily measured in Bespoke but it is very hard to measure in MDRE because it has not defined numbers of requirements and resources and hard to finalize the deadline for the project.

Quality is important for the both situation, but quality differs due to different stakeholders, target groups, processes, situations.

3. QUALITY IN DIFFERENT SYSTEMS

- **Safely critical systems**: The most common quality attributes of safely critical systems are reliability, availability, maintainability, fault tolerance and safely software for controlling the nuclear fusion chain reaction in an atomic reactor is an example of a safety critical software system. The quality of such a software system dramatically varies from the quality of audio playback software.
- **Banking systems:** Reliability and security are the most important quality attributes than other attributes as per the customers' point of view. Availability is also another important attribute.
- **Database Software:** The database software quality attribute is reliability, availability, scalability and recoverability. In addition to this, the quality may expected differently by different stakeholders. As an example internal customers' view of quality can be greatly varied from the external customers' view of quality. While the software product needs continuous addition of functionalities the design expectation should be more flexible and modifiable and the developers should expect the code more reliable.

3.1 Quality Differences

Unlike functional requirements, quality is usually seen as implied expectations of the external customers. The requirement engineers should be smart enough to explore and clarify the actual customer quality expectations. For example, a customer might want the software system to be faster. Since, the term faster is not absolute hence, it need to be defined what faster actually mean to the customer. May be the customer is interested in such a software that is able to handle one million users at a time and process some example functionality within the time limit of 2 minutes.

4 SOFTWARE PRODUCTS AS GOODS OR SERVICES

Software is unique in its nature. In these days software is considered to be equivalent to product at the same time it is indefinable like a service. For instance some corporate companies do not provide service to the users but they sell the product. These show that software is a good rather than a service.

4.1 Quality Dimension of software

The quality dimension of software can be stated as follows:

Reliability: Shown by the performance and correctness of functionality of the software in terms of its output.

Performance: It is measured by how fast and easily user can use the product features.

Usability: ease of use for client

Intangible: The virtual existence of software products in the world.

Fault tolerance: The product should not cause damage in case of failure

Credibility: it is a measure in which the trust of a customer on the service or a good

Accuracy: It is also a measure in which how closely we suggest actual use of a software product or service.

Similarities:

The similarities between software as a service and as a good are:

1 Solver: Both give solution of any problem

2 Seller: Both can be traded to the outside customers.
Example: selling software products and also providing services of calling on the phone lines through this product.

3 Indefinable: Since we do not have any physical existence both can be considered as intangible

4 Goal: the goal of service or good should be to satisfy customers' requirement

5 Common feature: Reliability is the common feature in the goods or service

Differences:

1 When the software is considered as a product then the buyer holds the property where as when it is considered as a service supplier holds the property.

2 Patent right are obtained when the software is considered as a good but as service it cannot hold any patent right

3 The services do not hold copyrights property where as the product can hold a copyrights property

4 Products are single entity where as service is a combination of many products to fulfill the required functionality.

5 SEVEN QUALITY TOOLS

These seven tools are very useful to keep a track on the control of different process in software development. These tools are also important to manage the work in a company or an organization in a systematic manner and also figure out the problems in simple way.

5.1 Data Collection

This tool is very important for a software project or program for the improvement of its quality. There are some predefined tables and sheets for the collection of data.

5.2 Pareto Charts

There may be many problems connected with a program. But only one problem can be solved at a time. The Pareto charts are helpful in prioritizing the problem. The top most priority is solved first and then it is moved to the next. The data collected are represented graphically where we get a clear picture of various errors. These charts show that very few problems account for a large number of errors. These are called as 80-20 rule that 80% of problem is produced by 20% causes.

5.3 Stratification

It is one way to figure out caused of variations. With the data collected from different sources we need to classify it into subgroups and display each group separately for example by histograms. The cause of the problem is identified where the histograms vary significantly.

While sub grouping we must follow the basic rule that the data from different origins must not be mixed together.
Some ways for classification for stratification are [1]:

1. Material
   - Supplier
   - Store
   - Time of purchase
2. Machine
   - Type
   - Age
   - Factory
3. Time
   - Time of day
   - Season
4. Environment
   - Temperature
   - Humidity

5.4 Histogram

It is the best way to display the characteristic variation of a product or a process. It is done keeping the frequency table as a basis. The measurement axis is divided into different parts, classes and number of values in each class is represented as a rectangle. The sum of these areas of rectangles is unity.

For instance, the project manager or the team lead can see the rate of bugs appearing in the product as they develop and they map the total bugs found in every week. These histograms will be very useful to team lead in order to develop the product without any bugs.

5.5 Cause effect diagram

These are also mostly used in the software development processes. This tool is used to find the root causes of a quality problem. The problems in the software quality may cause due to the management, material, measurement and so on. Actually this tool is used during brainstorming sessions or meetings. For a project in order to analyze the quality problems and its root causes. These diagrams help the quality teams to guess the main reasons for the quality problems and check out the root cause for the problems. For example some module is not working properly in a project then the analyst will make a sketch of the problems by using this tool to find the main root cause of the problem in that module. The analyst will check which input causes what effect and why it happens? These diagrams are used where the stratification is not possible.

5.6 Control Charts

These charts are used to represent the individuality of a process performed in a correct way or not. In software development most of the project managers uses this tool to measure the output of a team and performance of the team. With the help of this charts the team lead or project managers are able to find how the process, productivity and maintenance of the project is going on. Managers or team leads can also detect where the team lost its productivity in the project and can do some changes in the project team in order.

6. Quality Aspect of Supply Process

Supply process is considered as an important aspect to meet the uniform quality throughout the product lifecycle. Every organization expects its suppliers meeting the international standards of performance in quality, and expects providing reliable products and services within the defined time and budget. The goal of the organizations should be achieving high quality with low cost with increased in customers' satisfactions. This can be achieved by using appropriate process improvement techniques such as supplier certification, Six Sigma, Lean etc.

6.1. Relation of quality aspect of supply management to software development

Third party software or off-the-shelf products are the important portion of the software development lifecycle. It is the fact that most the products are not the standalone products. Therefore, the overall quality of the product also depends on the suppliers' product. If the quality of the suppliers' product is not up to expected, then it may result in dissatisfaction of the customers and may reduce the charm in the market. In order to be competitive in the market, every aspect of the product development lifecycle should be taken into consideration carefully.

Quality of the software development depends on many factors. One of the factors is quality of the suppliers' product. In the planning/elicitation/SRS phase of software development all the service providers should be distinctly identified. The suppliers or service providers plays a major role in the overall quality of the product. Thus it is necessary that a buyer should focus on the overall quality of the supplier's product.

The buyer should build the software quality process (SQP) in order to monitor, measure, control, and improve the quality of the suppliers products. It is important to link the processes of both suppliers and buyers in order to reduce poor quality cost, and it will build trust showing the commitment that quality products will be delivered within the defined deadline [1]. In order to measure quality of the product, the suppliers' process should meet the international standard of quality and with the buyers' standard of quality [1].

SQP focuses on the purchasing process and supplier partnerships. The purchasing should be based on the products benchmark of the suppliers, followed standards, should focus on the definite suppliers rather than maximum suppliers, and should be base on trust[1]. In order to achieve product goal they should establish the notion of supplier partnership, buyer should provide training to suppliers, involve suppliers in the design phase since they consist the domain knowledge of the supplied product, and should share information, risks and rewards to establish long term relationships[1].

6.2 Quality in Software process affecting the quality of end-product.

Due to less quality of software process, it may be possible to spend more cost to maintain the quality of end product in order to be in the competitive market.

The rational way of evaluating suppliers' product lies on

- Selection of international standard of quality for the quality process by suppliers

- Expertise of the suppliers on defined domain.
- Quality process maturity
- Knowledge of the suppliers' quality process actions taken for process improvements, and knowledge of manufacturing process [1].
- Comparison of the suppliers' product and process with the competitors working for the similar domain.

7. RELATION BETWEEN QUALITY MOVEMENT AND QUALITY

The four phases of quality movement discussed by Klefsjo[1] are quality inspection, quality control, quality assurance and quality management. Figure 1 shows how these four phases are aligned with the product development.

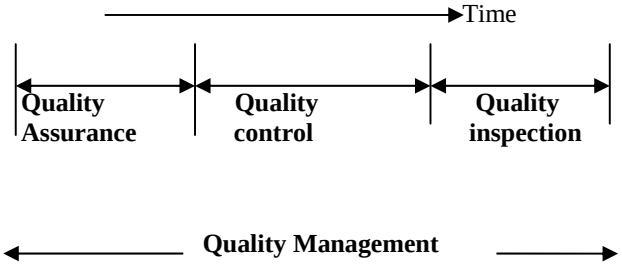

Fig1. Phases of software quality movement

Major software quality activities cover conducting of formal technical reviews, software testing, control of change, measurement and record keeping, reporting etc. Quality inspection is applied on the finished product. That means, this phase is related with the software validation activities like software testing. Quality control takes place during the production. So, quality control is related to on process quality activities like measurement and record keeping, review. Quality assurance is a phase that is related with pre-production quality activities like quality activities like quality assurance planning that focuses on how to measure different metrics, how to divide responsibilities, how to work with change request etc. The three phase's altogether comprises the quality management. The quality management make sure to systematically collect and define customers goal and requirements before designing the system. So, quality management is also related with the activities in requirement engineering that are related to systematic collection of customer requirements.

8. CONCLUSION

In this paper quality issues are discussed with different kind of systems and how quality differs in various situations.

9. REFERENCES

[1]. B. Bergman and B.Klefsjo, "Quality From customer Needs to Customer satisfaction", second ed. Lund Student litteratur, 2003
[2]. Hand Book of Software Quality Assurance by G. Schulmeyer et al(2007)
[3] Quality Software Management Vol 4. Anticipating change by G, Weinberg (1997)
[4] A practical guide to Information sytems Process Improvement by A. Cassidy et al (2000)
[5] CMM in Practice by P. Jalote (1990)
[6] Software Requirement Patterns by S. Withall (2007)
[7] Bespoke software developers Wikipedia
[8] MDRE software requirement developers' Wikipedia