Volume 4, Issue 1, January 2014 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

# Examining the Character of Software Characteristics to Improve Software Quality

**Nitin Deepak,**
Ph.D. Scholar (Bhagwant University)
India

**Dr. Shishir Kumar**
Professor, Department of CSE
Jaypee Univ. of Eng. Tech., Guna, M.P., India

***Abstract :Software quality has a great impact on individuals as well as on society. Through this paper issues related to software quality and its characteristics have been analyzed. Further look-up action on the role of software characteristics for improving the quality has been performed. In majority of cases people understand quality, appreciate quality but not able to express quality. Many author(s) define the quality as a different meaning like: Conformance of requirements, Fitness for the purpose, Levels of satisfaction etc. Bug deficiency in the program is also the part of our study to improve quality. Through this paper study on critical parameters like Usability, Scalability and planning perspective on which quality stands in the market has been performed. This paper will be finally concluded by emphasizing some critical aspects in improving the software quality.***

***Keywords: Software Quality, Software Characteristics, Usability, Scalability, Software design.***

## I. SOFTWARE QUALITY

To understand the landscape of software quality it is necessary to understand well about *Quality*. Once the quality is understood it is easier to understand the differentiation the say among different author(s). Many authors explain many different definitions of quality. By learning from all we are not here to give any specific new definition for quality but need to discuss two basic aspects, which surely fitted to define quality as follows [1]:

**1. Conformance to satisfaction**:
Quality that is defined as a matter of products and services whose measurable characteristics satisfy a fixed specification – that is, conformance to an in beforehand defined specification.
Quality of conformance is the degree to which the design specifications are followed during manufacturing. Again, the greater degree of conformance, the higher the level of quality of conformance [2].
In the software development, quality of design encompasses requirements, specifications, and the design of the system. Quality of conformance is an issue primarily focused on implementation. If the implementation follows the design and the resulting system meets its requirements and the performance goals, conformance quality is high [2].

**2. Meeting Customer requirements**:
Quality that is identified independent of any measurable characteristics. That is, quality is defined as the products or services capability to meet customer expectations – explicit or not.
Definitions given in dictionaries on quality are generally focused on excellence and some technical author(s) like to describe software quality in terms of "*Fitness for the purpose*".
Quality is defined by International Organizations as follows[3]:
*"Quality comprises all characteristics and significant features of a product or an activity which relate to the satisfying of given requirements."* – German Industry Standard DIN 55350 Part 11
*"Quality is the totality of features product that bears on their ability to satisfy given needs."* –

A. *"The totality of features and characteristics of a software product that bear on its ability to satisfy given needs: for example, conform to specification."*
B. *"The degree to which software processes a desired combination of attributes."*
C. *"The degree to which a customer or user perceives that software meets his/her composite expectations."*
D. *"The composite characteristics of software that determine the degree to which the software in use will meet the expectations of the customer."*

- IEEE Standard (IEEE Std 729-1983)

## II. MEASURE SOFTWARE QUALITY

Quality can determine a software product's success or failure in today's competitive market.
[5] Rapid IT growth and the rise of PCs have resulted in numerous software products. Although the market is rapidly growing, users are often dissatisfied with software quality.

To address the issues of software product quality, the Joint technical Committee 1 of the International Organization of Standardization and International Electro technical Commission published a set of software product quality standards known as ISO/IEC 9126.

**ISO/IEC 9126**

Quality, according to ISO 8402, is "the totality of characteristics of an entity that bear on its ability to satisfy stated and implied needs." ISO/IEC 9126-1 defines a quality model that comprises six characteristics and 27 sub-characteristics of software product quality (See table 1).

**TABLE 1**
**Characteristics and sub-characteristics in ISO/IEC 9126**

| **Characteristic** | **Sub characteristics** |
|---|---|
| Functionality | Suitability, accuracy, interoperability, security, functionality, compliance |
| Reliability | Maturity, fault tolerance, recoverability, reliable compliance |
| Usability | Understandability, learn ability, operability, attractiveness, usability compliance |
| Efficiency | Time behavior, resource utilization, efficiency compliance |
| Maintainability | Analyzable, changeability, stability, testability, maintainability compliance |
| Portability | Adaptability, install ability, replace ability, coexistence, portability compliance |

## III. SOFTWARE QUALITY ATTRIBUTES/ CHARACTERISTICS AND THE ISO-9126 STANDARD

ISO-9126 is a Software Product Evaluation Standard published by the International Standard Organization (ISO) in 1991, claims the software quality can be measured or defined by using the following attributes:

**Functionality**

A set of attributes that bear on the existence of a set of functions and their specified properties. The functions are those that satisfy the stated or implied needs of the client.

**Reliability**

A set of attributes that bear on the capability of software to maintain its level of performance under stated conditions for a stated period of time.

**Usability**

A set of attributes that bear on the effort needed for use; and on the individual assessment of such use by a stated or implied set of users.

**Efficiency**

A set of attributes that bear on the relationship between the level of performance of the software and the amount of resources used under stated conditions.

**Maintainability**

A set of attributes that bear on the effort needed to make modifications to the finished system.

**Portability**

A set of attributes that bear on the ability of software to be transferred from one environment to another.

## IV. EVALUATION AND MESUREMENT OF SOFTWARE PROCESS IMPROVEMENT

To increase the efficiency and effectiveness of a software development needs the systematic approach called *software process improvement* to enhance the software products.

With the increasing importance of software products in industry as well as in our everyday life [6], the process of developing software has gained major attention by software engineering researchers and practitioners in the last three decades [7], [8], [9], [10]. Software Process Improvement (SPI) encompasses the assessment and improvement of the process and practices involved in software development [11]. *SPI* initiatives henceforth refer to activities aimed at improving the software development process.

*SPI Initiative:*

Author categorized the studies according to the presented SPI initiative as follows:

- **Framework.** This group contains frameworks/models like CMM, international standards like ISO/IEC 15504 (SPICE) and business management strategies like Six Sigma. For the analysis, we further broke down this category into:
    - Established Frameworks – CMM, CMMI, ISO/IEC 15504 (SPICE), Six-Sigma, PSP, TSP, QIP, TQM, IDEAL, PDCA.
    - Combined Frameworks – two or more established frameworks combined together to implement SPI initiative.
    - Derived Frameworks – an established framework is extended or refined to fulfill the specific needs.
    - Own Framework – the study proposes a new framework without referencing the established frameworks.
- **Practices.** Software engineering practices which can be applied in one or more phases of the software development life-cycle (e.g. inspection, test driven development etc.)
- **Tools.** Software applications that support software engineering practices.

*Measurement Perspective:*
We use the concept of "measurement perspective" to define and categorize how the improvement is being assessed. Concretely, a measurement perspective describes the view on the improvement i.e., which entities are measured in order to make the change visible in either a quantitative or qualitative manner. We defined the following measurement perspective based on the five software entity types proposed by Buglione and Abran [12]:

- **Project perspective.** The measurement is conducted during the project where the *SPI* initiative takes place. Examples of metrics that are used to measure from this perspective are productivity during the development phase, defect rates per development phase, etc. These measures assess the entity type's process, project and resources.
- **Product perspective.** The evaluation of *SPI* initiative' impact is conducted by measuring the effect on the delivered products. An example of a metric that is used to measure from this perspective is the number of customer complaints after product release.
- **Organization perspective.** The measurement and evaluation of the *SPI* initiatives' impact is conducted organization-wide. An example of a metric that is used to measure from this perspective is return on investment. Other qualitative measurements such as employee satisfaction and improved business opportunities are also measured from this perspective.

## V. INSEPECTION OF BUGS FOR THEIR DEFICIENCY FOR TO MAINTAIN QUALITY

Deficiency of bugs in a program specifies quality in the software product. If too many functional defects exist in software, then it means basic functionalities does not meet with specific requirements. Let's express in two ways:

- *Defect rate*: number of defects per million lines of source code, per function point or any other unit
- *Reliability*: Time to failure or probability of failure free environment in a specified time under the specified environment.

Software Engineering denotes a systematic, disciplined, and quantifiable approach to software development, which involves a large part of human effort. Testing alone is not enough for assuring software quality as its effects come late in development also costly affair.
Inspection at every phase can be applied to find the defects early in life cycle to prevent rework later in projects. [13].
Some techniques for software inspection as follows:

- Individual defect detection
- Defect collection &
- defect repair

## VI. QUALITY CAN ALSO STANDS WITH USABILITY OF SOFWTARE

Usability can govern by understanding how efficiently and easily an end user can realize the goal that encourages to an interactive system.
[14]
*Usability factors:* At the SEI, the author isolated 26 usability factors that is required architectural support than separating the user interface.
*Factors mentioned:*

1. *Aggregating data*
2. *Aggregating commands*
3. *Cancellation of commands*
4. *Using applications concurrently*
5. *Checking for correctness*
6. *Maintaining device independence*
7. *Evaluating the system*
8. *Recovering from failure*
9. *Retrieving forgotten password*
10. *Providing excellent help*
11. *Re usability of information*

12. *Support to international use*
13. *Influencing human knowledge*
14. *Modifying interfaces*
15. *Supporting multiple activity*
16. *Steering within a single view*
17. *Examining system state*
18. *Working at user's pace*
19. *Predicting task duration*
20. *Supporting widespread searching*
21. *Including undo*
22. *Working in unfamiliar situation*
23. *Verifying resources*
24. *Operating consistently across views*
25. *Making views accessible*
26. *Supporting visualization*

The author organized these factors into two categories:

- On hierarchical benefits which impacts on usability aspect**.**
- On software engineering hierarchy based on the architectural mechanism used in the pattern.

The *first* one includes 9 common benefits usual to usability specialists such as mounting effectiveness by reducing the impact of slips or user errors, supporting problem solving or learning, and increasing user confidence and comfort and the *second* includes 13 mechanisms familiar to software engineers such as replication of data and commands, indirection, recording, preemptive scheduling, and various forms of separation – encapsulation of function, separating data from views of those data, and so on.

## VII. CONCLUSION

Software Quality practice is a planned and systematic pattern of actions necessary to provide adequate confidence that a software product conforms to requirements during software development. In this paper we are analyzing the role of software characteristics in improving software quality and also suggested the specific pattern of study including measurement of non-functional quality attributes, and inspection of bugs and usability of software.

## References

[1] Hoyer, R. W. and Hoyer, B. B. Y., "What is quality?" Quality Progress, no. 7, pp. 52-62, 2001.

[2] Roger S. Pressman adapted by Darrel Ince, "Software Engineering", Fifth Edition.

[3] Ronan Fitzpatrick, "Software Quality: Definitions and Strategic Issues", Staffordshire University, School of Computing Report, April 1996

[4] Jeff Tian, *Southern Methodist University*, Quality-Evaluation Models and Measurements Published by the IEEE Computer Society, 2004

[5] Ho-Won Jung and Seung-Gweon Kim, Korea Univ., Chang-Shin Chung, *Telecommunications Technology Association*, Measuring Software Product Quality: A Survey of ISO/IEC 9126. Published by the IEEE Computer Society.

[6] B. Kitchenham and S. Charters, "*Guidelines for Performing Systematic Literature Reviews in Software Engineering*," Techni- cal Report EBSE-2007-01, Software Eng. Group, Keele Univ. and Dept. of Computer Science, Univ. of Durham, UK, 2007.

[7] W. Scacchi, "*Process Models in Software Engineering*," Encyclo- pedia of Software Eng., pp. 993-1005, 2001.

[8] M. Shaw, "*Prospects for an Engineering Discipline of Software*," IEEE Software, vol. 7, no. 6, pp. 15-24, Nov. 1990.

[9] D.I.K. Sjoberg, T. Dyba, and M. Jorgensen, "*The Future of Empirical Methods in Software Engineering Research*," Proc. Future of Software Eng., pp. 358-378, 2007.

[10] N. Wirth, "*A Brief History of Software Engineering*," IEEE Annals of the History of Computing vol. 30, no. 3, pp. 32-39, July 2008.

[11] D.N. Card, "*Research Directions in Software Process Improvement*," Proc. 28th Ann. Int'l Computer Software and Applications Conf., p. 238, 2004.

[12] L. Buglione and A. Abran, "*ICEBERG: A Different Look at Software Project Management*," Proc. 12th Int'l Workshop Software Measurement, pp. 153-167, 2002

[13] Stefan Biffl, Member, IEEE, and Michael Halling "*Investigating the Defect Detection Effectiveness and Cost Benefit of Nominal Inspection Teams*" IEEE transactions on software engineering, vol. 29, no. 5, may 2003.

[14] Len Bass and Bonnie E. John, Carnegie Mellon University "*Supporting Usability Through Software Architecture*" Software,Technologies