International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878, Volume-1, Issue-3, August 2012

An Efficient Software Quality Models for Safety and Resilience

Namita Malhotra, Shefali Pruthi

Abstract-Software Quality Engineering is an emerging discipline that is concerned with improving the approach to software quality. The goal is to propose a quality model suitable for such a purpose through the comparative evaluation of existing quality models and their respective support for Software Quality Engineering. Cost, schedule and quality are highly correlated factors in software development, and difficult to increase the quality without increasing either cost or schedule or both for the software under development.

I. INTRODUCTION

What is Quality?

1. The degree to which a system, component, or process meets customer or user needs or expectations.
2. It is conformance to requirements.
3. The degree to which assets and corresponding artifacts meet the needs and expectations of the user.

Quality has been defined "fitness for use". Other definitions focus on the utility of the product or service.

A. Conformance to specification:

Quality that is defined as a matter of products and services whose measurable characteristics satisfy a fixed specification that is, conformance to in beforehand defined specification.

B. Meeting customer needs:

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.

II. QUALITY MODELS

A. McCall's Quality Model (1977)

The McCall quality model has three major perspectives for defining and identifying the quality of a software product 5]: product revision (ability to undergo changes), product transition (adaptability to new environments) and product operations (its operation characteristics).

Revised Manuscript Received on 30 August 2012.
* Correspondence Author
Scholar Namita Malhotra, Department of CSE, DIET, Karnal, India.
Scholar Shefali Pruthi, Department of CSE, JMIT, Radaur, India.

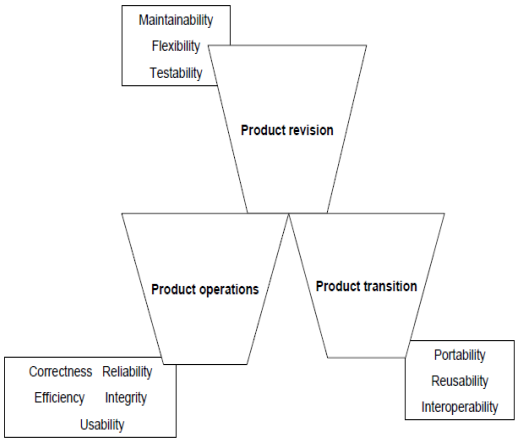

Fig.1 The McCall quality model

B. Boehm's Quality Model (1978)

Boehm's model is similar to the McCall Quality Model in that it also presents a hierarchical quality model structured around high-level characteristics, intermediate level characteristics, primitive characteristics - each of which contributes to the overall quality level.[11,12]

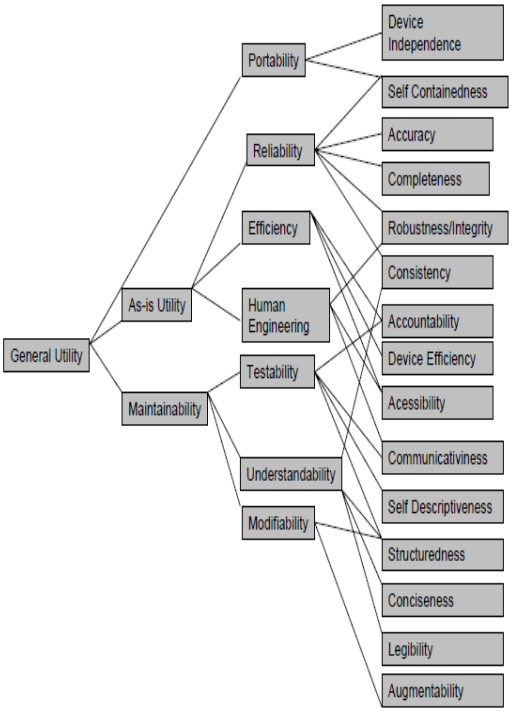

Fig.2 Boehm's Software Quality Characteristics Tree

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)

C. ISO 9126

The ISO 9126 standard was based on the McCall and Boehm models[7].Besides being structured in basically the same manner as these models ISO 9126 also includes functionality as a parameter, as well as identifying both internal and external quality characteristics of software products.

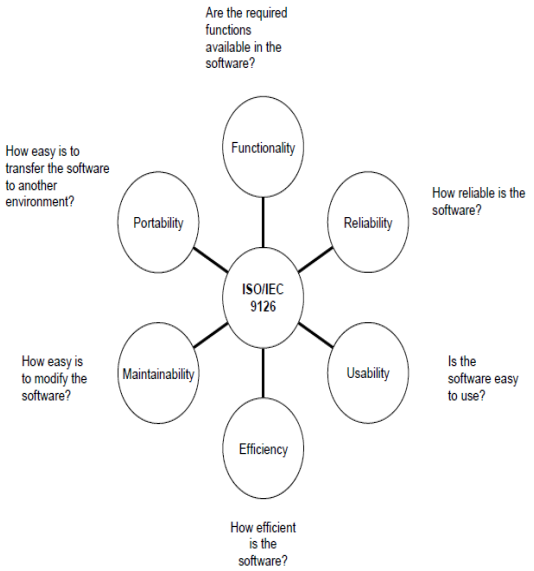

Fig.3 The ISO 9126 quality model

D. Dromey's Quality Model

Dromey's is focusing on the relationship between the quality attributes and the sub-attributes, as well as attempting to connect software product properties with software quality attributes.

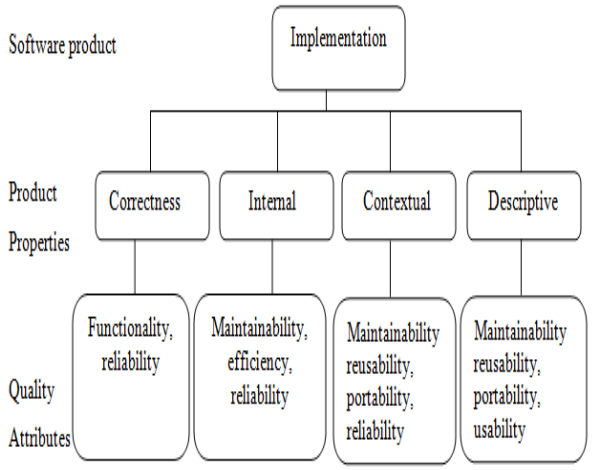

Fig.4 Principles of Dromey's Quality Model

E. FURPS

FURPS model originally presented by Robert Grady[10]. FURPS stands for: Functionality, Usability, Reliability, Performance and Supportability.

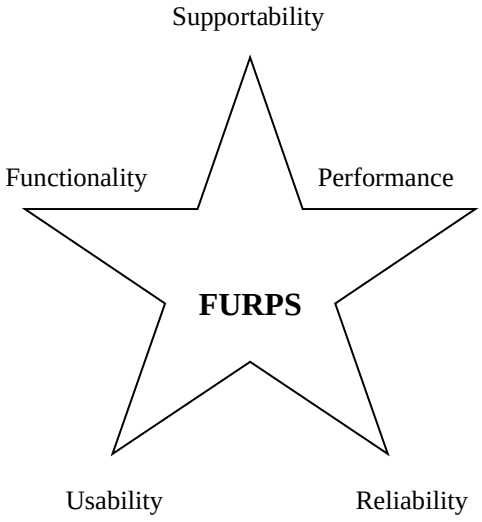

Fig.5 The contents of FURPS quality model

F. Comparison between five quality models

Factors/Attributes/Characteristics	McCall	Boehm	Dromey	FURPS	ISO 9126
Maintainability	*		*		*
Flexibility	*				
Testability	*	*			
Correctness	*				
Efficiency	*	*	*		*
Reliability	*	*	*	*	*
Integrity	*				
Usability	*		*	*	*
Portability	*	*	*		*
Reusability	*		*		
Interoperability	*				
Human Engineering		*			
Understandability		*			
Modifiability		*			
Functionality			*	*	*
Performance				*	
Supportabilty				*	
	17	11	7	7	5	6

I. PROPOSED WORK

The proposed approach uses the concept of safety contracts to define safety requirements between objects in the system. An analysis process is described which can be used to generate these contracts. It is shown how the safety contracts can be used in constructing a safety argument which demonstrates that the system is acceptably safe. Safety is freedom from accidents. Object-oriented approach successfully in safety critical applications they must be able to demonstrate that the resulting software system is sufficiently safe to operate. Software safety has to deal with the hazards identified by safety analysis in order to make tire n-system safe, risk free and fail-safe. Software safety is a composite of marry factors.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

At present there does not exists any standard framework that comprehensively addresses the Factors, Criteria and Metrics (FCM) approach of the quality models in respect of software safety.

Definition:

"Software safety" is the notion that software will execute within a system context without contributing to hazards. Software for safety-critical systems must deal with the hazards Identified by safety analysis in order to make the system safe. Software safety is a composite of many criteria.

A. ***New Approach***

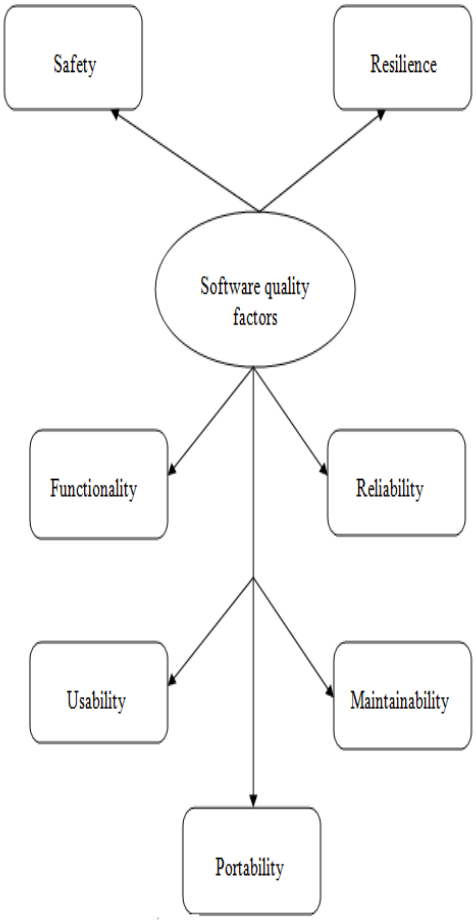

Fig.6 Software quality factors with new proposal

Proposed model for software safety

The proposed model for software safety based on the factor, criteria and metric approach. The quality factor software safety may be decomposed into six qualities. Criteria as shown below

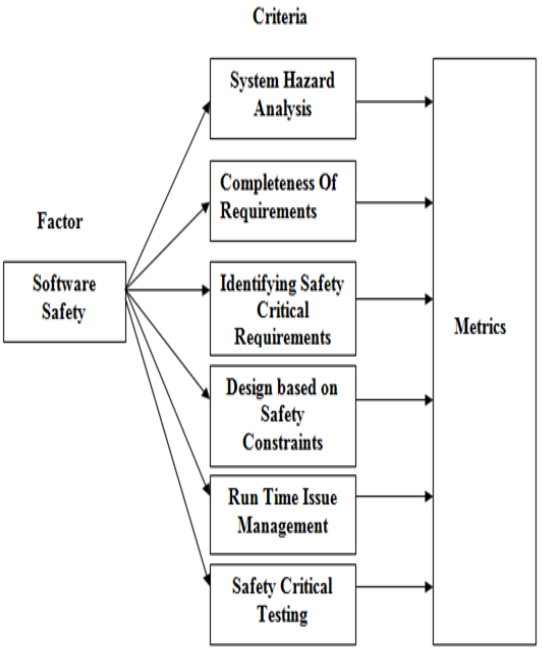

Fig.7 Software safety model

1) System Hazard Analysis

While modeling software safety is the focus of this approach it is important to note that no Software works in isolation. The entire system must be designed to be safe. The system contains the software, hardware, the users and the environment.
In essence there are four safety-relevant parts a system development process:

- Identifying hazards and associated safety requirements.
- Designing the system to meet its safety requirements.
- Analyzing the system to show that it meets its safety requirements.
- Demonstrating the safety of the system by producing a safety case.

2) Completeness of Requirements

Completeness can he defined as the property that requirements are sufficient to distinguish the desired behavior of the program from that of any other undesired program. Completeness is a quality often associated with requirements. Therefore the following different kinds of requirements may be: Functional Requirements, Data Requirements, Interface Requirements, Quality Requirements and Constraints.

3) Identification of safety-critical requirements

A safety critical software requirement may be understood as a software requirement identified:

- Controls or directly influences the function of safety critical hardware.
- Controls or directly influences hazardous systems.
- Monitors the state of the system for purposes of ensuring its safety.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

- Senses hazards and /or displays information concerning the protection of the system.
- Handles or responds to fault detection priorities.
- Disables or enables interrupt processing software.
- Generates output that displays the status of safety critical hardware.
- Computes safety critical data.

After identifying the individual safety-critical requirements, criticality analysis is performed, a method for determining the safety integrity level (SIL) from the interaction among safety-critical, safety-related, and non-safety-critical components method each module is classified into one of four criticality levels:

C3- Safety Critical: a module where a single deviation from the specification may cause the system to fail dangerously.

C2- Safety Related: a module where a single deviation from the specification cannot cause the system to fail dangerously, but in combination with the failure of a second module could caw a dangerous fault.

C1- Interference Free: a module that is not safety critical or safety related, but has interfaces with such modules.

C0 - Not Safety Related: a module that has no interfaces to safety-related or safety-critical Modules.

4) Design based on safety-constraints

The first step in the safety-constraint centered design approach is the Design based on safety-constraints. The first step in the safety-constraint centered design approach is the specification of safety constraints.

- Design for minimum risk
- Incorporate safety devices
- Provide warning devices
- Develop and implement procedures and training.

5) Run time issues management

There is always the risk that an a priori verified program behaves slightly differently-and faultily-at runtime. This may simply be the result of compiler bugs or it may be due to mismatches between the expected and actual behavior of the execution environment.

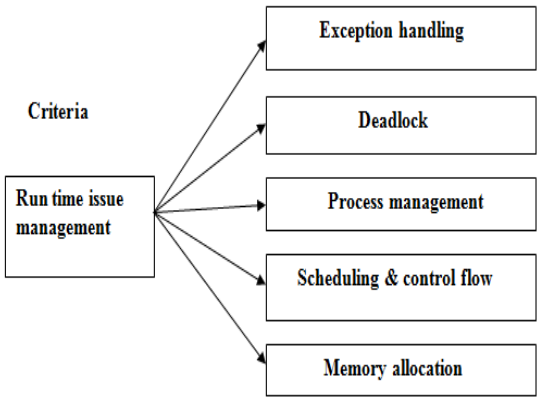

Fig.8 Criteria

6) Safety Critical Testing

Safety critical software functions provide the source of requirements to be tested. Testing shall be performed to verify correct incorporation of software safety requirements. Testing must show that hazards have been eliminated or controlled to an acceptable level of risk.

B. Resilience Factor

Resilience is defined in this context as the ability of the Data Center Manageability Interface (DCMI) manageability controller to recover from or adjust to changes in the external factors of the data centers and continue to provide availability of DCMI capabilities.

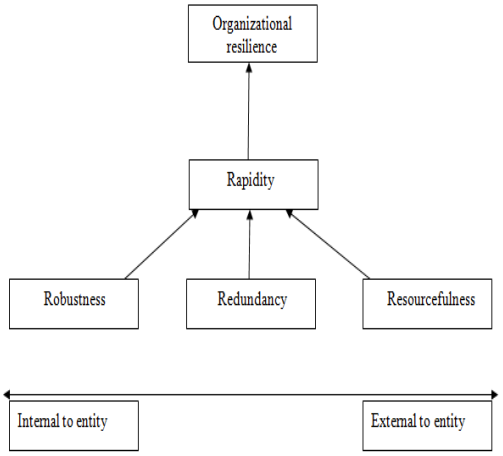

Fig.9 Resilience factor

Rapidity: The ability to quickly restore systems or processes.
Robustness: The ability of an economic entity to resist or forestall damaging or catastrophic events robustness. The degree to which a system or component can function correctly in the presence of invalid inputs or stressful environmental conditions.
Redundancy: An organizational unit's ability to provide alternative processes for inline or critical systems.
Resourcefulness: A characteristic of an entity's tenacious response to and creative solutions for disaster related instance.

1) Resilience Conceptual model

H1: Higher levels of robustness will be positively associated with greater gains or increased levels of organizational rapidity.
H2: Higher levels of redundancy will be positively associated with greater gains or increased levels of organizational rapidity.
H3: Higher levels of resourcefulness with be positively associated with greater gains or increased levels of organizational rapidity.
H4: Higher levels of rapidity will be positively associated with greater gains or increased levels of overall organizational or entity resilience.

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)

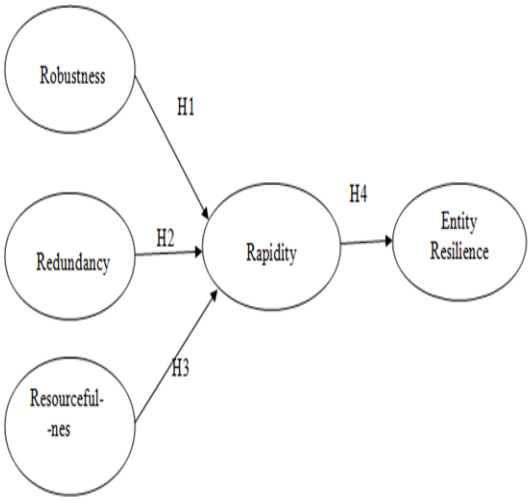

Fig.10 Conceptual model

2) Resilience

"This software quality factor named resilience increased the both satisfaction i.e. customer and user. It provide all the climate to the user. With the help of resilience factor increased software quality and reliability."

This document presents the reliability and resilience guidelines for Internet Portal servers which are conformant to the Data Center Manageability Interface (DCMI) specification Reliability is defined in this context as the predictable and consistent behavior of the DCMI manageability controller within the server and across the Data Centers.

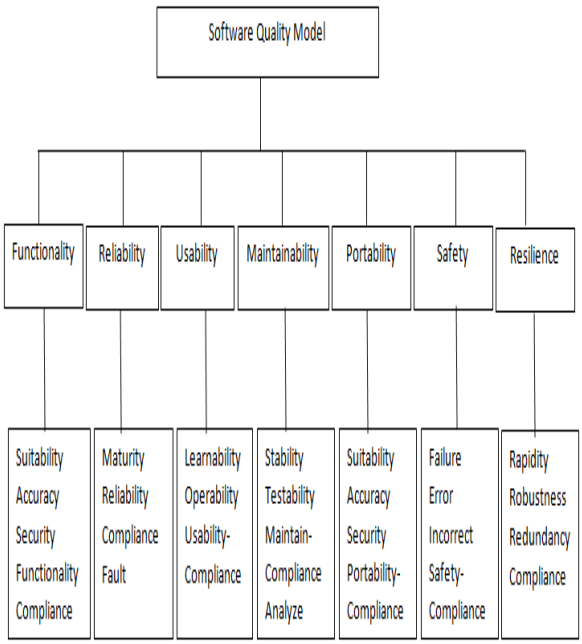

Fig.11 A proposed model with new invented software quality factors

II. CONCLUSION & FUTURE WORK

Software safety aspects that leads to standardization of framework, we proposed a resilience and safety framework for components and implement it as to improve any software quality. Software quality attributes try to describe all aspects of a software system. Software metrics have been providing to reflect software quality. The developers and software engineers use refactoring consciously as a mean to improve the quality of their software.

The proposed model is applied to improve software quality with the help of software quality measurement and metrics. The new approach named Safety and the Resilience describe all aspect of a system which mean that new attribute will be coined as time goes by some of the attributes will strong relation with the software architecture of software engineering. Resilience and reliability a strong guide line for internet portal servers which are conformant to data centre manageability interface (DCMI). It may be easier to describe the quality of separated component individually.

REFERENCES

1. Rafa E. Al-Qutaish, PhD Al Ain University of Science and Technology – Abu Dhabi Campus, PO Box: 112612, Abu Dhabi, UAE Quality Models in Software Engineering Literature: An Analytical and Comparative Study Journal of American Science, 2010.
2. Bhansali. P.V., 2005. Software safety: Current status and future directions. ACM SIGSOFT Software Eng. Notes, 30: 3.
3. Ralph R. Young, The Requirements Engineering Handbook, Artech House, 2004.
4. Dunn, W.,2003. Designing Safety Critical Computer Systems. IEEE-Computer, 36: 40-46. DOI: 10.1109/MC.2003.1244533.
5. John C.Knight, "Safety Critical Systems: Challenges and directions" Proceedings of the 24th International Conference on Software Engineering (ICSE), Orlando, Florida, 2002.
6. John McDermid, "Software Hazard and Safety Analysis". Book chapter in Formal Techniques in Real-Time and Fault Tolerant systems, page 23-24, Springer Link Book Series, 2002.
7. Lutz, R.R., 2000. Software engineering for safety A roadmap. Proceedings of the Conference on The Future of Software Engineering Limerick, June 04-11, Ireland, pp: 213-226.
8. Kitchenham, B. and Pfleeger, S.L., "Software quality: the elusive target [special issues section]", IEEE Software, no. 1, pp. 12-21, 1996.
9. Leveson, N., 1995. Software : System Safety and Computers.1st Edn., Addison-Wesley Publishing Company, Reading, Massachusetts.
10. Grady, R. B., Practical software metrics for project management and process improvement, Prentice Hall, 1992.
11. Boehm, B. W., Brown, J. R., Kaspar, H., Lipow, M., McLeod, G., and Merritt, M., Characteristics of Software Quality, North Holland, 1978.
12. Boehm, Barry W., Brown, J. R, and Lipow, M.Quantitative evaluation of software quality, International Conference on Software Engineering, Proceedings of the 2nd international conference on Software engineering, 1976.

AUTHOR PROFILE

Scholar Namita Malhotra received her B. Tech in computer science from Kurukshetra University, Kurukshetra in 2010. Currently she is pursuing her M. Tech from Kurukshetra University, Kurukshetra. Her research interests include software quality . She is novice in this field and this is her first paper.

Scholar Shefali Pruthi received her B. Tech in computer science from Kurukshetra University, Kurukshetra in 2010. Currently she is pursuing her M. Tech from Kurukshetra University, Kurukshetra. Her research interests include software quality and cloud computing.

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)