ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 3, Issue 12, December 2014

Reliability estimation of object oriented design: A systematic review

Anil Kumar [1], Namrata Dhanda [2]

Scholar, Department of Computer Science, GITM, Lucknow, India[1]

Associate Professor, Department of Computer Science, GITM, Lucknow, India [2]

Abstract: Over the past decades, software quality attributes have been extensively studied. In compare, fewer attentions have been paid to the field of software reliability. The size and complexity of computer systems have grown during the past decades in a very impressive manner. Complexity is a major factor of software reliability which degrades the performance of reliability. High complexity makes the system unreliable. There are a number of approaches to estimate reliability, several of which have been discussed in this paper. Software reliability can be increased by controlling Object-Oriented (OO) constructs such as coupling, cohesion, inheritance and polymorphism.
This review paper presents the results of a systematic literature review conducted to gather facts on software reliability estimation of object oriented design. In this review paper, our contribution is to discover the available recognized comprehensive and an absolute model or frameworks for measuring the reliability of object oriented design at an early phase of development life cycle.

Keywords: Software reliability, reliability measurement, Software metrics, Object Oriented Design, Software Quality.

1. INTRODUCTION

Software reliability is the possibility of failure-free software operation for a specified period of time in a specified surroundings.IEEE-Std-729-1991: "Software reliability is defined as the probability of failure-free operation for a specified *period of time* in a specified *background*". ISO9126: "Reliability is the capacity of the software product to continue a particular level of performance when used under specified situation". Easy definition of reliability: Reliability is a measurement of how very well the software provides the services expected by the customer. In new words it is the possibility of the product running "appropriately" larger than a specified period of time. More informally it indicates a product's honesty or dependability. In order to formulate comparison among different software products, software quality is a big parameter. There are a variety of attributes of software quality[9,14]:

- Reliability
- Functionality
- Flexibility
- Performance
- Serviceability
- Modifiability
- Analyzability
- Maintainability
- Testability.

Software reliability is one of the most vital characteristics of software quality, collectively with above attributes. Although software reliability is tough to achieve, for the reason that the complexity of software tends to be high. Despite the fact that the complexity of software is inversely connected to software reliability, it is openly related to other important factors in software quality, particularly functionality and capability.
Software reliability area in addition includes a range of issues such as software reliability modeling, forecast analysis, reliability quantity, fault categorization, inclination analysis, opinion data analysis, software metrics, software testing, error-tolerance, mistake trees, simulation, and a lot more. In this paper I will essentially center of attention on software reliability modeling, reliability quantity, software metrics and testing issues [15]. For the inspiration that reliability measurement is the key to accomplishing high reliability software. Our inspiration is that without measurement, software engineers would not be capable to achieve high reliability software[16,17]. Thus, design phase measurement is important to developing reliable software.

II. RELIABILITY

***Software Reliability* R (t):** The chance of failure-free operation of a computer program for a particular duration under a precise environment.
***Failure*:** The going away of program operation from user requirements.
***Fault*:** A deficiency in a program that causes failure.
Failure Intensity (rate) f (t): The projected quantity of failures experienced in a specified time interval.
***Mean-Time-To-Failure* (MTTF):** projected value of a failure interval.
***Expected total failures* m (t):** The number of failures projected in a time period t.

Reliability Theory

Let "T" be an arbitrary variable representing the failure time or life span of a physical system.
For this system, the chance that it will fail by time "t" is:
The chance of the system in existence until time "t" is:

ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 3, Issue 12, December 2014

Failure rate - the chance that a failure will happen in the interval $[t_1, t_2]$ given that a failure has not occurred earlier than time t_1. This is written as:

Hazard rate - boundary of the failure rate as the span of the interval approaches zero.

A reliability intention articulated in terms of one reliability measure can be effortlessly transformed into a further measure as follows (assuming an "average" failure rate, l, is measured).

Software reliability is a branch of software quality. It associates to a number of areas where software quality is concerned. Consequently quantifying software reliability leftovers a complex problem as we don't have a high understanding of the character of software. Reliability is considered as the possibility that a system will not fail to perform its proposed functions over a specified time period[18]. Consumers are sincerely attentive of the reliability of software; they are probable to be mostly unworried with the point of the reusability of the components making up the source code.

Unreliability has a quantity of regrettable consequences and as a result for many products and services is a severe warning[19]. For example low reliability can have inference for:

- Protection
- Competitiveness
- Profit boundaries
- Charge of repair and maintenance
- Delays further up supply chain
- Reputation
- Good will

III. SOFTWARE FAILURE MECHANISMS

There are a few key characteristics of software breakdown. These are:

- Failure grounds
- Wear-out
- Repairable system thought
- Time dependency and life cycle
- ecological factors
- Reliability forecast
- Redundancy
- Interfaces
- Failure rate motivators
- Built with standard mechanism

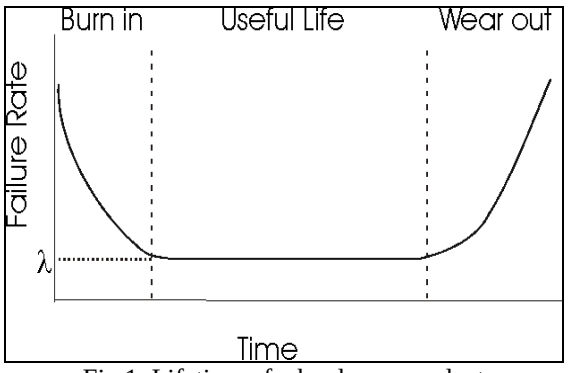


Fig 1. Lifetime of a hardware product

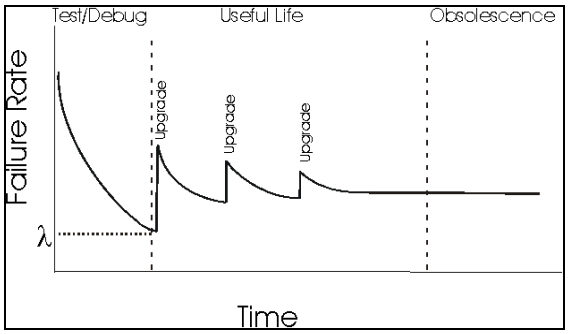


Fig 2. Lifetime of a software product

IV. SOFTWARE RELIABILITY MODELS

Software reliability models have emerged as community try to recognize the individuality of how and why software fails, and try to measure software reliability. In excess of 200 models have been developed since the early 1970s, but how to measure software reliability still leftovers mostly unsolved. There is no particular model that can be used in all situations. No model is comprehensive or even representative[20]. The majority software models have few assumptions, a few factors, and a statistical function which relates the reliability through the factors and is habitually higher order exponential or logarithmic.

Software modelling methods can be separated into two subcategories:

- Prediction modelling.
- Estimation modelling.

Both kinds of modelling methods are based on observing and collecting failure data and analyzing with statistical conclusion. An evaluation among these categories is specified in below table:

Issues	**Prediction Models**	**Estimation Models**
Data Reference	Uses past data	Uses data from the current software development effort
When Used In Development Cycle	Generally made prior to enhancement or test phases; can be used as near the commencement as perception phase	frequently made later on in life cycle(later than some data have been collected); not usually used in thought or development phases
Time organization	estimation reliability at some prospect time	Estimate reliability at either current or some future time

Table 1. Comparison between modeling techniques

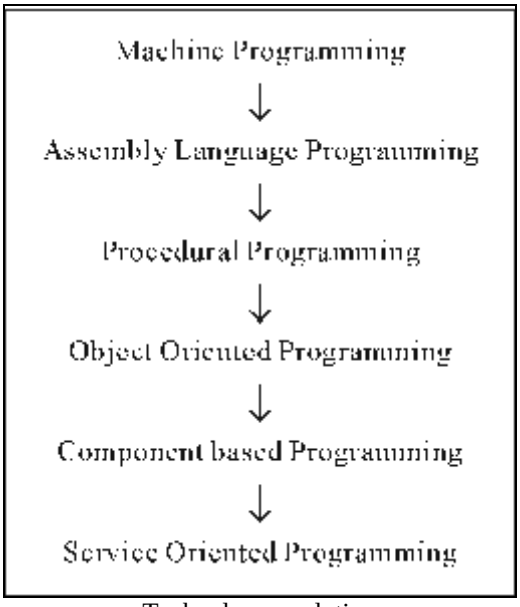
ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 3, Issue 12, December 2014

V. RELATED WORK

Goseva and Trivedi [1] offered a skeleton for software reliability modelling based on Markov renewal process which logically introduced dependence between successive software runs. The offered approach enabled the phenomenon of failure clustering to be particularly characterized and also its impacts on software reliability to be analyzed. In addition, it furthermore provided support for a more reliable and flexible model formulation and explanation.

Markov renewal model formulation had a number of advantages, in cooperation hypothetical and practical. It makes available flexible and extra steady modelling of software reliability. Taking into consideration the freedom in the midst of software runs is a unique case of the planned modelling framework. The model was appropriate to different phases of the software life cycle in view of the fact that the planned modelling approach was valid for testing (repairing) stage, as fine as for confirmation time and operational segment.



Technology evolution

Hamlet *et al*. [2] offered a foundational concept for reliability of software system fully based on components. The concept has illustrated how component developers could design and test their components to deliver measurements which can later be used for estimating composite system reliability by system designers without implementation and testing of the designed system. The concept addressed the very important methodological issues inherent in certifying components to be released for later use in a random system**.**

According to Lo *et al*. [3], the constraints in the software reliability models are generally directly gained from the field failure data. In view of the fact that, the systems have active properties and the failure data is not enough, it is actually tough to decide the values of the parameters truthfully.

Gayen and Misra [4] planned a pioneering approach to forecast the higher and lesser bound on the reliability of the COTS Component-Based software application. On the source of the implementation situation analysis for the COTS Component-Based Software System, a unique method was formulated. The planned algorithm for the calculation of higher bound was an enhancement over Dolbec and Shephard [5] replica for reliability evaluation of Component-Based Software. The shortcoming of this model was that it was implementation path self-regulating and component interfacing time was not taken into consideration. As a result, it was not capable to forecast the higher bound on reliability, as the higher bound on reliability gained using Dolbec and Shephard [5] model was much fewer than the value gained in the planned approach.

Fan Zhang *et al*. [6] wished-for a model depends on a CDG. In this model, an operational summary of a system is given, and the model can be used to ensure whether reliability changes while the operational profile changes.

Assuming that control flow transits from component *i* to component *j*, component *j*'s reliability is calculated as

$Tij \times (Rij \times Wij)$, where

Tij = the transition chance from component *i* to component *j*,

Rij = the reliability vector for every sub field of component *j*, and

Wij = the weight vector for each sub field of component *j* for the transition from component *i* to *j*.

Dong *et al*. [7] planned a method for CBSS reliability evaluation in which component associations are analyzed and solved with a Markov model. This method expands the range of Markov models. A drawback of this model is that it assumes that all component reliabilities and transition chances are given, but in follow this is not for all time true.

Huag *et al*. [8] wished-for a method based on algebra which presents a framework that can be put into practice on Maude for describing sentence structure and predicting reliability.

Goswami and Acharya [9] planned an approach to CBSS reliability examination which takes into consideration the system's component usage ratio, considered through mathematical formulas. Due to the suppleness of the component usage ratio, this approach may be used for real-time applications. This approach measured on the whole software system reliability based on the individual component reliabilities which are combining together to shape the system.

Si *et al*. [10] wished-for a framework for measuring reliability through a component composition method.

The approach recommends five essential component composition methods and techniques for their reliability evaluation. Subsequent to calculating the reliability for every composition, a process estimates the overall application reliability based on the component

ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 3, Issue 12, December 2014

composition mechanisms and component consumption frequencies. It is probable to identify added composition mechanisms.

Hsu *et al*. [11] planned an adaptive reliability evaluation method via path testing for complex component- based systems. Three methods are planned for estimation path reliability, specifically, sequence, and branch and loop structures. The planned path reliability can then be used for measuring the reliability of the whole application. This approach can present a capable evaluation of software reliability when testing information is available. A sensitivity examination is also performed to conclude the effect of every node on the system's reliability.

Wang and Huang [12] wished-for an approach for reliability assessment based on rewrite logic (RABRL). This technique considers systems whose condition is given with an operational profile. Maude's rewrite method is used to estimation the reliability. This method statistically analyzes an application's completing progression and uses this to just about guess the transition probabilities among components and the predictable number of visits to components. Though, this approach has a few limits: First, it can only be functional to simple CBSs, and second, it does not think about failure dependencies among components.

Shukla *et al*. [13] offered an organized method which provided a step-by-step process for rising operational profiles for software components. The technique has used together usage data and intended usage assumptions to find out a usage allocation, practice structure and characteristics of consideration.

VI. CONCLUSION

Software reliability is a key part in software superiority but software reliability enhancement is tough for the reason that there are no generic models to estimate reliability at an initial stage of development life cycle. Existing literature introduced basic software reliability concepts such as statistical testing and a few simple reliability models. Software reliability awareness is essential and nonflexible to achieve it. It can be enhanced by appropriate understanding of software reliability, uniqueness of software and sound software design. Guarantee of software reliability is no simple task. Subsequent to the above conversation our conclusion is that reliability is an eminence factor that endeavor to predict that how much attempt will be required for software testing. Following an exhaustive review course of action study found that dropping effort in estimating reliability of object oriented design is necessity in order to deliver superiority software in time and budget.

VII. CRITICAL OBSERVATIONS

Successive successful completion of the systematic literature review a number of important explanations can be enumerated as follows. Reliability is coupled with unpredicted failures of products or services and analyzing, understanding why these failures occur is important to improving software reliability. The main reasons why failures take place include:

- The product is not robust for purpose or more especially the design is inherently incapable due to lack of analyzability.
- Failures can be caused by wear-out
- Failures might be caused by deviation.
- Wrong stipulation may basis failures.
- Misuse of the item may grounds failure.
- Items are designed for a specific operating environment and if they are then used outside this environment then failure can occur.

REFERENCES

[1] Goseva, K. and Trivedi, K.S. (2000) Failure Correlation in Software Reliability Models. *IEEE Transactions on Reliability*, **49**, 37-48. http://dx.doi.org/10.1109/24.855535

[2] Hamlet, D., Mason, D. and Woit, D. (2001) Theory of Software Reliability Based on Components. 23*rd International Conference on Software Engineering* (*ICSE*), Portland State University, 12-19 May 2001, 361-370.

[3] Lo, J.H., Huang, C.Y, Kuo, S.Y. and Lyu, M.R. (2003) Sensitivity Analysis of Software Reliability for Component- Based Software Applications. *Proceedings of the* 27*th International Computer Software and Applications Conference* (*COMPSAC* 2003), Dallas, 3-6 November 2003, 500-505.

[4] Gayen, T. and Misra, R.B. (2008) Reliability Bounds Prediction of COTS Component Based Software Application.*IJCSNS International Journal of Computer Science and Network Security*, **8**, 219-228.

[5] Dolbec, J. and Shepard, T. (1995) A Component Based Software Reliability Model. *Proceedings of the* 1995 *Conference of the Centre for Advanced Studies on Collaborative Research*, Toronto, 7-9 November 1995, 19.

[6] Zhang, F., Zhou, X., Dong, Y. and Chen, J. (2009) Consider of Fault Propagation in Architecture-Based Software Reliability Analysis. *International Conference Computer System and Application*, Rabat, 10-13 May 2009, 783-786.

[7] Wang, D. and Huang, N. (2008) Reliability Analysis of Component Based Software Based on Rewrite Logic. 12th *IEEE International Workshop on Future Trends of Distributed Computing Systems*, 21-23 October 2008, 126-132.

[8] Huang, N., Wang, D. and Jia, X.G. (2008) Fast Abstract: An Algebra-Based Reliability Prediction Approach for Composite Web Services.19*th International Symposium on Software Reliability Engineering*, Mysuru, 16-19 November 2009, 285- 286.

[9] Abdullah, Dr, Reena Srivastava, and M. H. Khan. "Testability Measurement Framework: Design Phase Perspective.",International Journal of Advanced Research in Computer and Communication Engineering Vol. 3, Issue 11, Pages 8573-8576 November 2014

[10] Si, Y.J., Yang, X.H., Wang, X.Y., Huang, C. and Kavs, A.J. (2011) An Architecture-Based Reliability Estimation Framework through Component Composition Mechanisms. 2*nd International Conference on Computer Engineering and Technology*, Chengdu, 16-18 April 2010, 165-170.

[11] Hsu, C.-J. and Huang, C.-Y. (2011) An Adaptive Reliability Analysis Using Path Testing for Complex Component Based Software Systems. *IEEE Transaction on Reliability*, **60**, 158-170. http://dx.doi.org/10.1109/TR.2011.2104490

[12] Wang, D. and Huang, N. (2008) Reliability Analysis of Component Based Software Based on Rewrite Logic. 12th *IEEE International Workshop on Future Trends of Distributed Computing Systems*, 126-132.

[13] Shukla, R., Carrington, D. and Strooper, P. (2004) Systematic Operational Profile Development for Software Components. *Proceedings of the* 11*th Asia-Pacific Software Engineering Conference* (*APSEC*'04), Busan, 30 November-3 December 2004, 528-537.

[14] Abdullah, Dr, Reena Srivastava, and M. H. Khan."Modifiability: A Key Factor To Testability", International Journal of Advanced Information Science and Technology, Vol.26, No.26, Pages 62-71 June 2014.

[15] Abdullah, Dr, Reena Srivastava, and M. H. Khan. "Testability Estimation of Object Oriented Design: A Revisit.", International

ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940

Journal of Advanced Research in Computer and Communication Engineering, Vol. 2, Issue 8, Pages 3086-3090 August 2013.

[16] University of Sunderland, "Software Measurement and Reliability", Quality and Information Systems Strategies, COMM1H, 2003.

[17] Pan, J., "Software Reliability", Carnegie Mellon University, Dependable Embedded Systems, 1999.http://www.ncst.ernet.in/education/apgdst/sefac/slides/Reliability_QA_Standards.ppt

[18] Vilkomir, S., "SQRL's future work in software reliability", International Symposium CTVR, 2004.

[19] Sawada, K., Sandoh, H., "A Summary of Software Reliability Testing Models", International Journal of Reliability, Quality and Safety Engineering, 1999.

[20] Schneidewind, N., F., "Software Reliability Measurement", Reliability Review, The R&M Engineering Journal, 2003.