Derivation of Reliability Estimation Model for Metrically Reliable Software Product Development

Maruf O. Alimi, Ph.D.[1*]; F.E. Usman-Hamza, Ph.D.[1]; I.O. Mustapha, M.Sc.[1]; and B.S. Ogidan, M.Sc.[2]

[1]Computer Science Unit, Department of Physical Sciences, Al-Hikmah University, Ilorin, Nigeria.
[4]ICT Centre, Al-Hikmah University, Ilorin, Nigeria.

E-mail: moalimi@alhikmah.edu.ng [*]

ABSTRACT

Flawless software with 100% functionality, reliability, compatibility, and flexibility is not yet available. Many researchers, software developers, and software engineers have tried their best, but yet are to produce a perfect software system. This research attempts to derive a model that can be used to make software more reliable at the design and development stage. Reliability formula was derived using existing serial and parallel reliability formula as applied to modular systems of software development. For better error prevention than rework, functionality profile of modules was suggested and was used to calculate reliability with minimum acceptance level for critical and non-critical systems. With these reliability formulas, if fully implemented will enhance the productivity and reliability of software products.

(Keywords: software, reliability, serial-module, parallel-module, critical systems)

INTRODUCTION

Software metrics in software engineering serve as a core standard of measuring or evaluating software and quantitative way of assessing parameters with estimators which should originally be valid formulas. To have software reliability metrics, there is a need to derive formulas for estimation. Software reliability metrics compared to hardware reliability have shown that software component failures are often transient rather than permanent since these failures are due to some inputs and if the data or code is undamaged, the system can often continue in operation after failure has occurred.

The reliability of a system uses probability, in a given period of time, that the system will correctly deliver services as expected by the user (Sommerville, 2004). Metrics are unit of measure for the software reliability, which are used to measure the reliability of software product. The measurement of software can be used to plan the time needed for development and testing before product release (Kaur et al., 2013).

A reliable, trouble-free (fault tolerant) product continues to satisfy customers for a long time after development. In statistics, reliability refers to the consistency of a measure. A measure is said to have a high reliability if it produces consistent results under consistent conditions (Carlson, 2009). This shows that quality of a software product also rely on the reliability of such software.

Related Works

Gokhale et al., (2004) used an analytical approach to architecture-based software performance and reliability prediction. They examined software application treated as a whole in conventional system while components making a whole may have effect on the software performance and reliability. The model used relates the failure intensity of a component to the expected number of faults initially present in the component and it is time–dependent.

Jung-Hua et al., (2004) also did their own on reliability assessment and sensitivity analysis of software reliability growth modelling based on software module structure where they claim that most of the parameters in software reliability model are obtained from the failure data just like usage profile system.

Wen-Li et al., (2005) developed architecture-based software reliability model, while Marko et al., (2011) based their own research on component-based for reliability estimation, prediction and measuring. The probability for a failure state is calculated using the probability vector.

Most of researchers didn't measure reliability metrics using module functionality profile or a combination of serial and parallel modules and were unable to put critical system software into thoughtfulness. In this research, the derived formulas used considered functions expected of a software module as very important to its reliability while backward module considerations in serial modules are also very important. The derived formula can now be used for software reliability during design stage and used to prove the dependability especially in critical systems.

OBJECTIVE

Derivation of reliability formula for software development, since no software is fully perfected, is the major reason behind this research. Many software developers are used to usage profiles for the finding of software reliability while this research shows that functionality profile at software design level will be a better option for critical systems or life-based and non-critical systems. This is another way to prove how reliable a software product is using software module functionality profile to calculate the product reliability metrics for its acceptance or rejection.

METHODOLOGY

The existing reliability formula which have been in existence for decades were used with simple probability to calculate probability of module functions, average value of a function applied to where there are combinations of modules functions, and reliability function which is being substituted with results of the module functions. A flowchart was developed to show the flow of steps to follow in getting successful reliability value and determine if it meets the acceptable standard or not and determine either to reject or accept such software product. A sample of real life software was used to show metrically and analytically, how it can be used to determine any software reliability.

Serial and Parallel Modules

Development of software requires modules with transition from one stage to the other. Often-times, one module relies on another module before its own execution. As stated by Kitchenham et al. (1996), it is generally accepted that more complex modules are more difficult to understand and have a higher probability of defects than less complex modules. Due to these, one may likely have serial form of modules whereby one module must finish its execution before the other with parallel executions whereby all the modules at a stage must have all their results for the next input of another module.

Figure 1 is an example of combined serial and parallel modules. The modules in the figure were used to calculate the overall reliability of the software using reliability formula that combined both the serial and parallel reliability equations after each module probability has been calculated. The modules A, B, and H are serially connected while modules D, C, and E formed first parallel modules and modules F and G formed the second parallel modules.

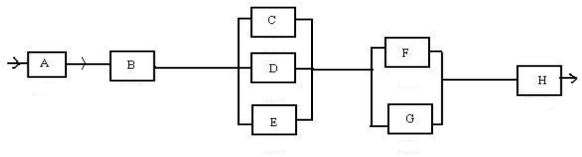

Figure 1: Model of Serial – Parallel Module Combination.

Symbolic Model for the Serial and Parallel Modules

Methods to develop the equation from analogue models along with serial and parallel combination will depend on the complexity of the software. This is applicable where there are more modules probabilities working together to give single result. This is related to dependability of one module on the other in which result of a module is required in another before final output of the software.

(i) **Average value of a function:** The concept of the average value of a function is often useful. This process suggests that we ought to get an approximation to the average value of a function f(x) (Mary, 1966). The average of f(x) is approximately equal to:

$$\frac{f(x_1)+f(x_2)+\ldots+f(x_n)}{n} \ldots\ldots (1)$$

This indicates that after finding the probability of every module or boxes, one can find the final average of all functionality probability values.

(ii) **Simple Probability:** Sometimes reliability calculation is not possible without probability as mentioned by some authors or researchers (Irwin and John, 1965), (Wen-Li et al., 2005). Mathematically or statistically, probability calculation in case of this research goes thus;

$$\Pr = \frac{No\ of\ successful\ funtions}{Total\ no\ of\ funtions} = \frac{n(s)}{t(f)} \ldots\ldots (2)$$

This formula was used to calculate each module or box probability.

(iii) **Reliability function:** Reliability and life testing applicability cut across all forms of manufacturing of products especially software that drive most of the current and modern machines world-wide today. The task of designing and supervising the manufacturing of a product has been made increasingly difficult by rapid strides in the sophistication of modern products and the severity of the environmental conditions under which they must perform. Therefore, this research adopts and adapts the existing reliability models. Since software development consist of serial and parallel modules, both approaches are put into consideration for this research.

(a) **Serial System:** A series system is one in which all components are so interrelated that the entire system will fail if any one of its components fails. A system of n components connected in series is said to have probability to function by the special rule of multiplication for probabilities. The general formula is:

$$R_s = \prod_{i=1}^{n} R_i \ldots\ldots (3)$$

where R_i is the reliability of the ith component and R_s, is the reliability of the series system. This simple *product law of reliabilities,* applicable to series systems vividly demonstrates the effect of increased complexity on reliability.

(b) **Parallel System:**A parallel system is one that will fail if all of its components failed. The parallel model derivation goes thus:

If $F_i = 1 - R_i$ ……………………..………… (4)

is the “unreliability” of the ith component, then the special rule of multiplication for probabilities was applied to obtain

$$F_p = \prod_{i=1}^{n} F_i \ldots\ldots (5)$$

where F_p is the unreliability of the parallel system, and

$R_p = 1 - F_p$ ………..……………………………(6)

is the reliability of the parallel system.

For parallel systems we have a product law of un-reliabilities analogous to the product law of reliabilities for series systems. This law in another way for reliability of a parallel system is:

$$R_p = 1 - \prod_{i=1}^{n}(1 - R_i) \ldots\ldots (7)$$

General Reliability Model (Combined both Serial and Parallel)

Equations for all necessary metrics calculations for the software reliability can now graduate to higher level after combination of all boxes modules considering those in series and parallel using Equation 3 and Equation 7.

$$R_{sp} = \left(\prod_{i=1}^{n} R_i\right)\left(1 - \prod_{j=1}^{m}(1 - R_j)\right) \ldots\ldots (8)$$

where R_{sp} =General and final reliability
R_i=Reliability for serial modules
R_j=Reliability for parallel modules

This is possible because of commutative law. This can be combined as many as possible depending on the number of series and parallel modules of the software.

Flowchart for Reliability Determination

Pictorial steps to follow for development of software using an Object Oriented Programming Language is necessary, thus the flowchart to show its flow and development along with reliability value calculation, and rejection of non-compliance with confidence level is as shown in Figure 2.

The flowchart depicts options for user to continue or quit immediately after starting in case the operator decides not to continue. The users are expected to list the system requirements and experts are to give detail specifications or functions for the software, module by module, if not, return to where to specify the functions as expected. Individual module reliability is expected to be calculated followed by both serial and parallel modules reliabilities. The combination of both calculates the final and general reliability of the software using reliability formulas. The result is either accepted or rejected based on life-based or non-life-based significant level of 99% and 95%, respectively.

Applicability

Model presentation and evaluation is necessary to have clear understanding of the model using test data. It can be used for system software, application software or any other software that makes quality its priority. The general reliability model that combined all the modules can be applied to real life software system development. Some software are now used to experiment the model. They are real life software developed in some reliable and well known organisations and federal institutions in Nigeria.

Software developed for Staff Multipurpose Cooperative Society, Usmanu Danfodiyo University Teaching Hospital, Sokoto was studied to determine its reliability using the new model. The modules obtained are home page, login, menu with three parallel modules such as update contribution and balances module, View and Report module; View module has personal information module in parallel with viewing balances module.

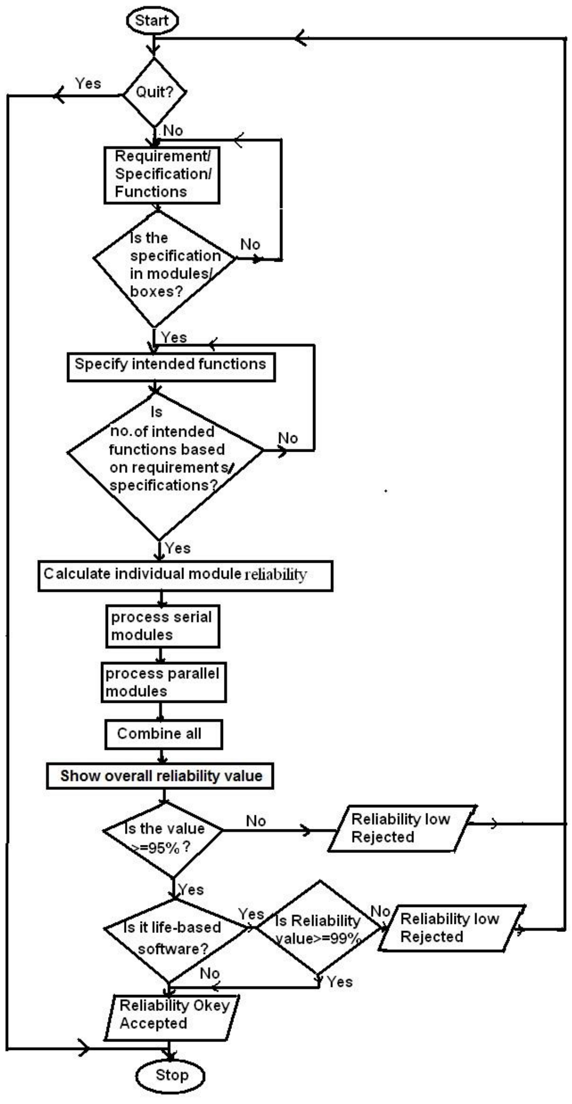
Start
Quit?
Yes
No
Requirement/ Specification/ Functions
Is the specification in modules/ boxes?
No
Yes
Specify intended functions
Is no. of intended functions based on requirements/ specifications?
No
Yes
Calculate individual module reliability
process serial modules
process parallel modules
Combine all
Show overall reliability value
Is the value >=95% ?
No
Reliability low Rejected
Yes
Is it life-based software?
Yes
Is Reliability value>=99%
No
Reliability low Rejected
Yes
No
Reliability Okey Accepted
Stop

Fig. 2: Reliability Determination Flowchart

Figure 2: Reliability Determination Flowchart.

Sample Software

Before exit message module we have warning module (Figure 3) for each module reliability value as calculated using new reliability formulas.

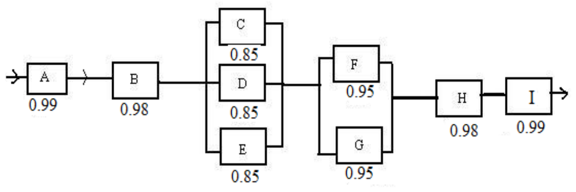

Figure 3: Module Reliability for Sample Software.

Each module probability has been calculated. The reliability for the software, having the reliabilities in Figure 3, has four modules in series and two parallel with 3 and 2 modules, respectively. Since it is commutative, the reliability can be calculated using the following equation.

$$R_{sp} = \left(\prod_{i=1}^{4} R_i\right)\left(1-\prod_{i=1}^{3}(1-R_i)\right)\left(1-\prod_{i=1}^{2}(1-R_i)\right) \quad (9)$$

The final reliability calculation goes thus:

Let Parallel C, D, E be C'

Parallel F, and G be F'

The we have A,B, C', F' H and I

Then$C'=1-(1-0.85)^3=1-0.15^3=1-0.003375=0.997$

$F'=1 - (1 - 0.95)^2 = 1 – 0.05^2 = 1 - 0.0025=0.998$

The original system, has the reliability:
R_{sp}=(0.99) (0.98) (0.997) (0.998) (0.98) (0.99) =0.9366

Equivalent to 93.66% reliability

This is rejected because it is not up to 95% as minimum reliability for non-life-based requirement.

RESULTS AND DISCUSSION

It is a known fact that the possibility to develop flawless software and have 100% functionality, reliability, compatibility, and flexibility is not yet feasible. This can be reduced with the use of combinations of different software development techniques. The software functionality approach used in this research has advantages such as, satisfactory development of all functions expected of the software derived from users' requirements, programmers, and all other stakeholders.

Functions that were not included at the development stage can sometimes be discovered through instinct and this may have important consequences on such development. This situation requires continuous software maintenance for its correction, adaptation, perfection, enhancement, and prevention of total failure of the software.

Failure of software can be attributed to so many parameters (e.g., hardware failure, virus attack, fragmentations, Operating System (OS) platform, compatibility, memory capacity, processor speed, system upgrade (hardware/OS), and the like). All of these may not be directly related to or have adverse effect on the design stage of the software but at the implementation stage.

Comparison

Many theoretical reliability model, lack practical applicability and use of statistical values to proof their implementation. This statement is supported by Mark C. van Paul (1994) which says, "The preoccupation with model building has resulted in a large number of theoretical models which seem to lack immediate appeal from a statistical point of view and failed to reduce the gap between statistical theory and applications in the area of software reliability".

In addition, Ritika Wasen (2012) added that traditional software reliability estimation methods depend on assumptions like statistical distributions that are dubious and unrealistic. The new model compared with other models developed by other researchers, shows that most of them used to calculate for error prevention, fault detection and removal, measurements to maximize reliability, specifically measures that support the first two activities. Most of them are based on time, usage profile and a lot of assumptions.

The new models make use of expected functions that are realistic of every module of software during design and development and make sure they are working perfectly before deployment. The requirements and specifications given by the users and those added by the developer or development team determine the reliability of the software product. The developed software that can perform all those functions correctly and optimally has already encapsulates the error prevention, fault detection, and removal.

Carlo Copp (1996) added that software reliability has relationship with the following failure; Numerical Failure - bad result calculated, Propagated Numerical Failure - bad result used in other calculations, Control Flow Failure - control flow of thread is diverted, Propagated Control Flow Failure - bad control flow propagates through code, Addressing Failure - bad pointer or array index, and Synchronization Failure - two pieces of code misunderstand each other's state. Looking at these failures, serial modules reliability calculation will take care of one module relying on another along with parallel modules for better flow of intra-communication among modules with metric values to support and proof their reliability.

CONCLUSION

Factors or indicators such as minimum processor speed, memory capacity, hard disk space, and OS are sometimes put into consideration during design and development stage. In addition, software evaluations require priori and posteriori analyses to determine the actual problem of the software and guide against unforeseen circumstances. But these may not be able to depict actual functions expected of the software, therefore, software reliability determination during design stage will be of immense benefit to all forms of software product development as any function expected of the software that cannot be solved shows where rework or recoding will be necessary and improve its reliability.

ACKNOWLEDGEMENTS

We are very grateful to all that contributed to the success of this paper. The developers and users of the software we used for implementation of the new model are greatly acknowledged for their support and valuable information given to us for this research.

REFERENCES

1. Carlson, N.R. 2009. *Psychology: The Science of Behaviour, 4th Canadian Edition*. Pearson: Toronto, Canada. ISBN 978-0-205-64524-4.

2. Gokhale, S.S., E.W. Wong, J.R. Horgan, and K.S. Trivedi. 2004. "An Analytical Approach to Architecture-Based Software Performance and Reliability Prediction". *Performance Evaluation, An International Journal*. Elsevier. 58:391-412.

3. Irwin, M. and J.E. Freund. 1965. *Probability and Statistics for Engineers*. Prentice Hall, Inc.: Englewood Cliffs, NJ. 5-12, 362-370.

4. Jung-Hua, L., H. Chin-Yu, C. Ing-Yi, K. Sy-Yen, and R.L. Michael. 2004. "Reliability Assessment and Sensitivity Analysis of Software Reliability Growth Modelling Based on Software Module Structure". *Journal of Systems and Software*. Elsevier. 76:3-13.

5. Kaur, M., S. Singh and M. Rakshit 2013. "A Review of Various Metrics used in Software Reliability". *International Journal of Computer Science and Engineering Technology*. 4(7) July.

6. Kitchenham, B. and Pfleeger, S.L. 1996. "Software Quality: The Elusive Target". *IEEE Software*. 13:1.

7. Kopp, C. 1996. *System Reliability and Metrics of Reliability*. Peter Harding & Associates, Pty Ltd: Canberra, Australia.

8. Marko, P., E. Antti, and O. Eila. 2011. "The Reliability Estimation, Prediction and Measuring of Component-Based Software". *The Journal of Systems and Software*. Elsevier. 84:1054-1070.

9. Mary, B.L. 1966. *Mathematical Methods in the Physical Sciences*. John Wiley and Sons Inc.: New York, NY. 673-679.

10. Sommerville, I. 2004. *Software Engineering 7, Seventh Edition*, Pearson Addison Wesley: London, UK. 217.

11. Wasen, R., P. Ahemed, and M.Q. Rafiq. 2012. "New Paradigm for Software Reliability Estimation". *IJCA*. 44(14).

12. van Paul, M.C. 1994. "A General Introduction to Software Reliability". *CWI Quarterly*. 7(3).

13. Wen-Li, W., P. Dai, and C. Mei-Hwa. 2005. "Architecture-Based Software Reliability Modelling". *The Journal of Systems and Software*. Elsevier. 79:132-146.

ABOUT THE AUTHORS

Dr. Alimi Olasunkanmi Maruf, is a Senior Lecturer in the Department of Physical Sciences, Al-Hikmah University-Ilorin-Nigeria. He is a member of Nigeria Computer Society; Computer Professional Registration Council of Nigeria, and Teachers Registration Council. He holds B.Sc., M.Sc. and Ph.D. in Computer Science from Federal University of Agriculture Abeokuta and Obafemi Awolowo University Ile-Ife, Nigeria respectively. Currently he serves as the Director, Centre for ICT and Distance Learning. His research interest is in software engineering / software development.

Dr. (Mrs.) Hamza-Usman is a Lecturer in the Department of Computer Science, University of

Ilorin. She holds a Ph.D. in Computer Science. Her research interest is in software engineering.

Mr. Mustapha I.O. is a Lecturer in the Department of Physical Sciences, Al-hikmah University Ilorin. He holds Masters in Computer Scoence. His research interest is in software architecture.

Ogidan B. S. is the Coordinator ICT at ICT Centre, Al-hikmah University Ilorin. He holds a Masters in Computer Science. His research interest is in software development.

SUGGESTED CITATION

Alimi, O.M., F.E. Usman-Hamza, I.O. Mustapha, and B.S. Ogidan. 2018. "Derivation of Reliability Estimation Model for Metrically Reliable Software Product Development". *Pacific Journal of Science and Technology*. 19(1):182-188.

Pacific Journal of Science and Technology