International Journal of Advances in Electronics and Computer Science, ISSN(p): 2394-2835 Volume-6, Issue-8, Aug.-2019
http://iraj.in

RELIABILITY ASSESSMENT OF COMBINED HARDWARE-SOFTWARE TIME CRITICAL SOFTWARE SYSTEMS: ISSUES AND CHALLENGES

ANJUSHI VERMA

Ph.D Research Scholar, Jawaharlal Nehru University, New Delhi
E-mail: anjuverma0313@gmail.com

Abstract - Although there exists several models for estimating the reliability value of software and hardware yet very few of them consider in compatibility issues and failure dependencies between software and hardware systems**.** Most of the models consider either pure hardware failure or pure software failure leading to system failure. Neither do they consider hardware related software failures nor software related hardware failures. Also they do not consider the incompatibility issues resulting in system failures. This paper discusses drawbacks, deficiencies and limitations of various existing approaches and finally suggests suitable strategies for the effective reliability assessment of time critical software systems comprising of hardware failure, software failure, hardware related software failure etc.

Keywords - Reliability, Time Critical Software Systems, Components, Subcomponents.

I. INTRODUCTION

A time-critical software system is a system in which the accuracy of the system behaviour depends on both logical results of computations and on physical instant at which these results are obtained [1]. These systems may fail because of functional errors as well as timing bugs. For the time critical system, timing bugs (temporal incorrectness) means the defect where the system is not able to perform its service within specified time deadline. In these systems, failure might cause catastrophic consequences (such as someone dying, damage to property, severe financial losses, etc.). To avoid these losses, it is necessary to provide temporal correctness of programs in addition to provide functional correctness.

There exist several model for estimating the reliability of hardware and software systems. However many models either consider the assessment of hardware reliability or consider the assessment of software reliability. As it is known that software and hardware both are highly co-related and both of them are equally affecting the system but very few researchers [2,3] considered assessment of reliability of both hardware and software subsystem. Failures are divided into two types: pure hardware failure and pure software failure. In pure hardware failure, system fails because of change or degradation in hardware only. There is no interaction with software system. In pure software failure, software fails because of error in design of software or attack of virus etc. In some of the approaches, it is assumed that both hardware and software failures are independent. They do not consider the interactions between them. But in reality, there exists some interaction between software and hardware failures. Because of change or failure in hardware, software may not be able to work properly or it may fail.

This paper discusses the drawbacks, deficiencies and limitations of various existing approaches finally suggests suitable strategies for the effective reliability measurement of time critical software systems comprising of hardware failure, software failure, hardware related software failure etc.

II. PAST INCIDENCES OF SOFTWARE FAILURES

Most of the software failures are only inconvenient but some of them have very serious consequences either financially or threat to human well-being**.**List of past incidences with consequences are as:

- A booster went off course during launch, resulting in the destruction of NASAMariner 1. This was the result of the failure of a transcriber to notice an overbar in a written specification for the guidance program, resulting in the coding of an incorrect formula in its FORTRAN software. (July 22, 1962). Note that the initial reporting of the cause of this bug was incorrect [4].
- The European Space Agency's Ariane 5 Flight 501 was destroyed 40 seconds after takeoff (June 4, 1996). The US$1 billion prototype rocket self-destructed due to a bug in the on-board guidance software.
- The European Space Agency's CryoSat-1 satellite was lost in a launch failure in 2005 due to a missing shutdown command in the flight control system of its Rokot carrier rocket.
- NASA Mars Polar Lander was destroyed because its flight software mistook vibrations due to atmospheric turbulence for evidence that the vehicle had landed and shut off the engines 40 meters from the Martian surface (December 3, 1999).
- A bug in the code controlling the Therac-25radiation therapy machine was directly responsible for at least five patient deaths in the

1980s when it administered excessive quantities of beta radiation.

- A Medtronic heart device was found vulnerable to remote attacks in March 2008.
- An error in the payment terminal code for Bank of Queensland rendered many devices inoperable for up to a week. The problem was determined to be an incorrect hexadecimal number conversion routine.

III. RELATED WORK

For the time-critical software used in time-critical systems, reliability plays an important role in software quality. These systems requires high reliability beacuse the severity of consequence resulting from the failure of thesetime-critical systems are usually very high. In accordance with Biswas et al. [5], for software used in real-time systems, there exist various execution dependencies. These software can fail not only because of functional error, but also because of timing bugs. Hence, it is necessary to provide temporal correctness in addition to provide functional correctness.Various work have been done for estimating reliability of software and hardware. Yacoub et al. [6] proposed a path based approach which considered various execution scenarios. It obtained the reliability of component based system using reliability values of the components, their interface and link reliabilities. But, no guidelines were provided for finding the reliability values of used components. In 2008, Kapur et al.[7] proposed Non-Homogeneous Poisson Process (NHPP) based SRGM which was flexible enough to describe various software failure. In it, testing effort and time dependent fault detection rate both were considered for software reliability prediction and also included the effects of imperfect debugging/error generation. Gayen[8] analyzed the shortcomings in the conventional failure rate based models and proposed an error based model for predicting the minimum reliability of software. In 2011, Gupta et al.[9] used a Yamada Delayed S Shaped model which considered fault dependency, debugging time lag and imperfect debugging. In it, imperfect debugging was used, probability of failure free execution of software is less than using perfect debugging. Shelbi et al. [10] estimated reliability of complete system by considering both elements: software and hardware. This model also did not consider incompatibility between hardware and software. There was no interactions among hardware and software components. Park et al.[11] suggested two new reliability models which were Random based model and Weibull distribution model. These models also did not consider incompatibility between hardware and software. There may be some pure hardware failures which are not considered. In 2012,Nagar et al.[12] selected a Goel-Okumoto model to find out the reliability of any software. This model provided a reasonable estimates of predicted errors with the assumption that whenever any failure occurs because of any fault it will be corrected immediately and perfectly. But this may not be very realistic as there may be a possibility that while correcting that fault some new faults may be introduced. This aspect was not considered here. Tyagi et al.[13] considered four factors- component dependency, application complexity, reusability, and the operational profile, whose impact on Component based software system is very high. Based on these factors, a model was proposed for the assessment of reliability of component based system which was based on fuzzy logic. Here failure dependencies among the components were not considered. Teng et al. [14] proposed a unified model for measuring reliability. It considered software failure due to hardware changes but did not consider hardware failure due to software changes and incompatibility issues between hardware and software had not been taken into consideration. They considered the entire system as a black box and used $R_{combined}(t)=e^{-\lambda t^{\beta}} e^{-(m(t+T)-m(T))}$ where $m(t)$ is a mean function, λ is failure rate, β is Weibull parameter, t is duration, T is software startup time, to obtain the reliability values of the composite system including software and hardware.

IV. ISSUES AND CHALLENGES

From the survey, it has been found that many approaches consider software system as a black box and ignore internal structure of the application, and hence, the reliability of the various modules of the application are not individually obtained. Many models do not consider the timing bugs explicitly for time-critical software systems.

It is also observed that many models do not consider the hardware related software failure or software related hardware failure resulting in system failure. There may also exist incompatibility issues between hardware and software because of which the system may not be able to perform its function properly. But these incompatibility issues between hardware and software has not explicitly been taken into consideration in most of the approaches. Many approaches again do not consider interactions among hardware and software components. It may be possible that there may occur some dependency among software and hardware components which is also not considered in various models. Teng et al. [14] proposed a unified model for measuring reliability. It considered software failure due to hardware change but do not consider hardware failure due to software change and other incompatibility issues between hardware and software. Several approaches consider the entire system as a black box hence the internal structure of the components are not taken into consideration. There may also be a possibility that because of hardware or software

failures, system maynot work properly (leading to failed state) or it may work in degraded mode. These types of failure leading to system failure has not been taken into consideration in existing approaches. There may be another possibility that hardware failure can lead to software failure which results into either system failure or sometimes it may not lead into system failure (but may lead to working in degraded mode). Similarly software failure can lead to hardware failure which results into either system failure or sometimes it may not lead into system failure. These issues are also not considered in approaches proposed earlier. It is also to be noted that failure of some sub systems (especially Fault Tolerant systems), components,subcomponents(comprising of either software or hardware or both)may not result in the overall system failure. But many approaches have not considered this aspect for the reliability assessment of time critical software systems. Also there are some fault tolerant intelligent software systems which can restore back to its normal working condition after the occurrence of a failure. But again this feature has not been taken into consideration by any of the existing approaches for the reliability assessment of time critical software systems.

V. THE PROPOSED VIEW FOR A SOFTWARE SYSTEM

A software system [15] comprises of Application Software components, Operating System components and hardware components having interdependencies among themselves. To obtain the reliability value of each component individually and not the system as a whole, each component needs to be tested individually (unit testing). This may be done by testing the component in its ideal working environment. It is because the intention is to detect the failures which are caused by the faults present only in this component and not contributed by the faults present in other components of the system. In general, a Software System may comprise of the Application Software with the underlying Operating System (which may even include hardware).

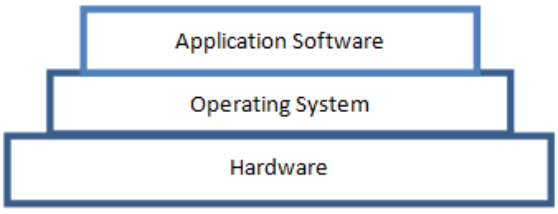

Figure 1: A diagrammatic representation of the 3-tier architecture of a Software System[16].

Reliability values of the each component used may be obtained (and not the system as a whole), by testing each component separately. This may be done by testing the componentseparately in its ideal working environment in accordance with a suitable functional and structural analysis for reliability assessment.

Reliability of pure software excluding hardware can be estimated by variousapproaches [17]. Similarly reliability of pure hardware system can be estimated using Exponential distribution, Teng et al. model [14] etc. But it is not proper to estimate reliability either considering only software or hardware because there may exist some dependency between hardware and software so it is necessary to estimate reliability of both. Only few approaches consider dependency between software and hardware components and estimate reliability of these components.

The system (comprising of software, hardware) may fail in various ways:

- Pure hardware failureleading to system failure($p_{phf\text{-}sysf}$),
- Pure software failure leading to system failure($p_{psf\text{-}sysf}$),
- Incompatibility issues between hardware and software leading to system failure($p_{incomp\text{-}sysf}$).

Due to pure hardware failure (irrespective of whether software has failed or not failed), the system may be in the working condition or in the failed condition ($p_{phf\text{-}sf\text{-}sysf}$). Similarly due to pure software failure (irrespective of whether hardware has failed or not failed), the system may be in the working condition or in the failed condition (($p_{psf\text{-}hf\text{-}sysf}$). Hence probability of pure software failure which may lead to system failure has to be taken into consideration. Probability of pure hardware failure (with respect to time) leading to system failure has also to be taken into consideration. Incompatibility issue between software and hardware which can result into system failure should also be considered.

Hence all above issues should be taken into consideration for the reliability assessment of time critical software system. Thus reliability of total system can be estimated as:

$$\mathbf{Rel_{Total}= 1-(p_{phf\text{-}sysf}+ p_{psf\text{-}sysf}+ p_{hf\text{-}sf\text{-}sysf}+ p_{sf\text{-}hf\text{-}sysf}+p_{incomp\text{-}sysf})} \quad \ldots\ldots(1)$$

Reliability of pure hardware can be estimated by using Exponential distribution hence,

$$\mathbf{Rel_{Total}} = 1-(e^{-\lambda t} + p_{psf\text{-}sysf} + p_{hf\text{-}sf\text{-}sysf} + p_{sf\text{-}hf\text{-}sysf} + p_{incomp\text{-}sysf})$$

where,

$p_{phf\text{-}sysf}$denotes system failure due to pure hardware failure,

$p_{psf\text{-}sysf}$denotes system failure due to pure software failure,

$p_{sf\text{-}hf\text{-}sysf}$ denotes system failure because of hardware failure causing software failure,

$p_{incomp\text{-}sysf}$ denotes system failure because of software failure causing hardware failure.

Here hardware can be like ICs, Resistor etc. Suppose ICs are considered as hardware part so its reliability can be estimated by $e^{-\lambda t}$.
Failure rate λ of IC is 0.0038 failure/10^6 hrs.[18]
Assuming time t = $10*10^6$hrs,
So failure probability ($p_{phf\text{-}sysf}$) of IC i.e. hardware can be calculated as :1- $e^{-\lambda t}$
$p_{phf\text{-}sysf}$= 1-0.9627
= 0.0372
Failure probability of software can be estimated by the approached proposed by Verma et al.[17] which is calculated as 0.724 considering timing bugs, dependency present in the software components.

Assuming
$\mathbf{p_{psf\text{-}sysf}}$ = 0.0010
$\mathbf{p_{hf\text{-}sf\text{-}sysf}}$=0.0002
$\mathbf{p_{sf\text{-}hf\text{-}sysf}}$ =0.0002
$\mathbf{p_{incomp\text{-}sysf}}$=0.00001

After putting these value in the equation (1), whole system reliability can be obtained.

$$\mathbf{Rel_{Total} = 1-(e^{-\lambda t} + p_{psf\text{-}sysf} + p_{hf\text{-}sf\text{-}sysf} + p_{sf\text{-}hf\text{-}sysf} + p_{incomp\text{-}sysf})}$$

$\mathbf{Rel_{Total}}$= 1- (0.0372+0.0010 +0.0002 +0.0002 +0.00001)
= 1-0.038.
= 0.961
Hence from the equation, reliability of total system is obtained which is 0.961.
Reliability of whole system varies based on Time t(as hardware reliability varies with time). It can be shown in Table 1.

S.No.	Time t (hrs)	Reliability R
1.	$5*10^6$	0.979
2.	$10*10^6$	0.961
3.	$15*10^6$	0.943
4.	$20*10^6$	0.925
5.	$25*10^6$	0.907

Table.1 : Reliability value varies based on time t

A plot between Reliability value and Time t based on Table 1 values is displayed in Figure 2:

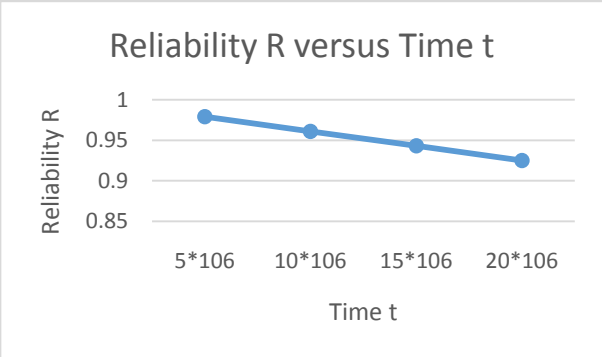

Figure 2: Plot between Reliability value and Time

It can be observed from the plot that with varying time , reliability value also varies. Since here time is increasing hence reliability value of the system decreases.
Now Hardware related software failure, Software related hardware failure and time all these are varying. Based on these variations, reliability of the system is calculated. Table 2 shows varying reliability value based on varying Hardware related software failure, Software related hardware failure and time.

S.No.	Time t (hrs)	H/w related S/w Failure	S/w related H/w Failure	Reliability R
1.	$5*10^6$	0.005	0.003	0.971
2.	$10*10^6$	0.008	0.006	0.947
3.	$15*10^6$	0.010	0.009	0.923
4.	$20*10^6$	0.014	0.012	0.899
5.	$25*10^6$	0.017	0.015	0.875

Table 2: Varying reliability value based on varying Hardware related software failure, Software related hardware failure and time

Based on these values, a plot is obtained and shown in Figure 3.

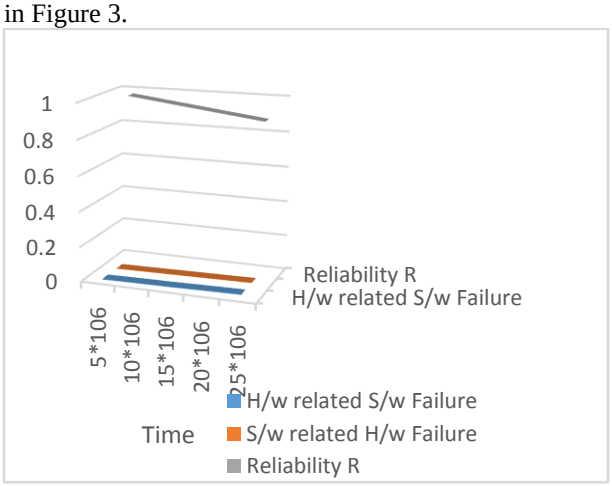

Figure 3: Reliability value based on varying hardware related software failure, software related hardware failure and time.

Hence it can be seen that reliability value varies when hardware related software failure, software related hardware failure and time vary.

VI. CONCLUSION

Although there are several existing models for software system reliability assessment yet many models do not consider the incompatibility issues and dependencies between hardware and software. The proposed strategy considers hardware failure, software failure; hardware related software failure and software related hardware failure and incompatibility issues between hardware and software for time critical software systems. It

evaluates separately the failure probability values of various states likePure software failure, Pure hardware failure, Hardware related software failure considering various dependency and incompatibility issues.

REFERENCES

[1] Kopetz, H. "Real-Time Systems, Design Principles for Distributed Embedded Applications," *Kluwer Academic Publishers*, (1997).

[2] M.A. Friedman and P. Tran, "Reliability techniques for combined hardware/software systems," *Proceedings of Annual Reliability and Maintainability Symposium*,pp.290–293,1992.

[3] S.R. Welke, B.W.Johnson, and J.H.Aylor, "Reliability modeling of hardware/software systems.," *IEEE T Reliab*,pp. 413–418, 1995.

[4] Dalal S. , Chhillar R.S. "Case Studies of Most Common and Severe Types of Software System Failure". *International Journal of Advanced Research in Computer Science and Software Engineering*, 2(8), 2012.

[5] Biswas, S.; Mall, R.; Satpathy, M.: Task dependency analysis for regression test selection of embedded programs. IEEE Embed. Syst. Lett. 3(4), 117–120 (2011).

[6] S.Yacoub."Scenario Based Reliability Analysis of Component Based Software", *IEEETransactions on Reliability*, pp. 22-31,2004.

[7] P.K.Kapur, D.N.Goswami, A.Bardhan and O.Singh, "Flexible software reliability growth model with testing effort dependent learning process", *Appl Math Model* , pp.1298–1307,2008.

[8] T.Gayen, "Analysis and proposition of error based model to predict the minimum reliability of software", *International Conference on Education Technology and Computer*, pp.40-44,2009.

[9] A.Gupta, D.Choudhary D and S.Saxena, "Software Reliability Estimation using Yamada Delayed S Shaped Model under Imperfect Debugging and Time Lag," *Int J Comput Appl T* ,pp. 0975-8887,2011.

[10] J. Shelbi , P.Shouri and V.A. Jagathy, " Model for Reliability Estimation of Software based Systems by Integrating Hardware and Software," *IJCA Special Issue on Computational Science - New Dimensions & Perspectives*,pp.26-29,2011.

[11] J.Park , H.Kim, J.H.Shin and J.Baik, " An embedded software reliability model with consideration of hardware related software failures",*IEEE Sixth International Conference on Software Security and Reliability*,pp.207-214,2012.

[12] P.Nagar and B.Thankachan, " Application of Goel-Okumoto Model in Software Reliability Measurement," Special Issue of International Journal of Computer Applications on Issues and Challenges in Networking, Intelligence and Computing Technologies,pp.0975-8887,2012.

[13] K.Tyagi and A.Sharma , "A rule-based approach for estimating the reliability of component-based systems," *Adv Eng Softw*, pp.24–29,2012.

[14] X.Teng and H.Pham, " Reliability modeling of hardware and software interactions, and its applications," *IEEE T Reliab* ,pp.571-577,2006.

[15] A.Verma. and T.Gayen, "Reviewing and resolving the issues of Black Box Software Reliability Assessment by White Box approach: a case study on time critical software.(Accepted)", *13th International Conference on Development and Application Systems*, Suceava, Romania,2016.

[16] A.Verma and T.Gayen, " Suitability of Black Box Approaches for the Reliability Assessment of Component-Based Software",*18th International Conference on Electrical, Computer, Electronics and Communication Engineering* ,pp. 660-667 ,2016.

[17] Verma, Anjushi, and Tirthankar Gayen. "Reliability Assessment of Time-Critical Software: A Case Study on Stepper Motor Application." Arabian Journal for Science and Engineering (2018): 1-16.

[18] Military handbook MIL-HDBK-217F: Reliability Prediction of Electronic Equipment - Notice F.

★★★