JSST Journal of Space Science and Technology

Vol. 15 / Special Issue / 2022 / (No. 51)
Print ISSN: 2008-4560 / Online ISSN: 2423-4516
https://doi.org/10.30699/jsst.2021.1283

OPEN ACCESS

Pages: 45-53 / Research Paper / Received: 02 June 2020 / Revised: 20 February 2021 / Accepted: 09 March 2021

Journal Homepage: https://jsst.ias.ir

Evaluation of Diversity Effects on Increasing the Reliability of Space Systems

Ghasem Kahe[1]* and Mehdi Alemi Rostami[2]

Aerospace Research Institute, Ministry of Science, Research and Technology, Tehran, Iran

*Corresponding Author's E-mail: kahe@ari.ac.ir

Abstract

Diversity in both hardware and software plays an essential and unmatched role in increasing the reliability of redundant systems, especially in safety and mission critical applications. The onboard computer of satellites and the flight computer of spacecrafts, which are ultra-reliable systems, utilize various hardware platforms for their redundant architecture to resolve a common cause failure (CCF) problem. Furthermore, the software is also developed by separate teams based on different software platforms to mitigate the specification and design flaws, and implementation mistakes. This paper focuses on modelling the diversity of redundant architectures in space systems using CCF modelling and Markov reliability analyzing. The proposed scheme is explored in two types of applications: mission critical applications (with long mission time) and safety critical applications (with short mission time). Analytical and simulation results show the effectiveness of diversity in increasing the reliability of these systems. Since a significant percentage of all failures appear as common cause failures, which restrict reliability improvement through similar redundant modules, achieving ultra-reliability necessitates considering diversity in these systems.

Keywords: Reliability, Redundancy, Diversity

Nomenclature

λ	Failure Rate
λ_s	H.W. Failure Rate
λ_h	S.W. Failure Rate
λ_c	Common cause failure rate
MTTF	Meat Time To Failure
MTTR	Meat Time To Repair
$R(t)$	System Reliability
$F(t)$	System Failure Probability
$P_i(t)$	The probability that the process will be in state i at time t.

Introduction

Redundancy means the duplication or multiplication (three and more) of a system for doing the same task to resolve the problem of single system failure with active or passive replication scenarios. If the system reliability is of concern, then redundancy is the straightforward solution, and therefore, multiple modules or subsystems should be used. According to the reliability requirements, redundancy can be employed in hardware, software, information, and time domain. Ideally, the failure probability of two or more redundant systems at the same time is close to zero. Nevertheless, this does not account the common cause failures (CCF), which affect multiple and similar modules at the same time. Hereon, redundancy technique with similar components is ineffective against common cause failures. Design mistakes, implementation bugs, human errors, and out of specification operating conditions are some of common cause failures in which it can affect all the similar redundant modules of a system at the same time, and therefore, deactivate the system. Actually, failure mechanisms which are dependent may affect all similar redundant modules simultaneously and lead to system failures.

Failures can be classified according to the failure causes [1] (see Fig. 1) or failure effects [2]. Some of the failure causes may appear as CCF. It is recommended to split CCF causes into root causes and coupling factors [8, 9]. A root cause is a basic cause of a component failure (see Fig.

1. Assistant Professor

2. Assistant Professor

How to cite this article:
Gh. Kahe and M. Alemi Rostami, "Evaluation of Diversity Effects on Increasing the Reliability of Space Systems," *Journal of Space Science and Technology*, Vol. 15, Special Issue, pp. 45-53, 2022, https://doi.org/10.30699/jsst.2021.1283.

2). The root causes of CCF events are classified according to design phase and proceeds through the manufacturing, construction, installation, commissioning phases, plant operation, maintenance and environment. The source of common causes of failure may be external or internal. The tsunami which disabled all the power sources of dissimilar cooling pump related to Fukushima Daiichi reactors (the electrical transmission grid, diesel electric generator, and batteries used for backup), has been identified as an external cause [3]. Interfaces, the environment, and catastrophic events are identified as the external sources of systematic failures. The interfaces include power, cooling, material inputs, and external controls. The environment can produce harsh environmental conditions such as excessive pressure, very high temperature, hazardous vibration and impact, noise, and contamination. Catastrophic events include hurricane, tornado, flood, earthquake, tsunami, and blizzard. Some of the other common cause failures occur in different and wondrous ways. Failure of one subsystem in a redundant system may damage its redundant modules, as when an internal failure in Apollo 13 caused one oxygen tank to explode and its explosion destroyed the second redundant oxygen tank [3]. This is a cascade type of common cause failure.

In the most frequent case, an internal cause such as design flaw or programming bug can cause all identical redundant modules to fail in the same way at the same time. Many sources of systematic failures have been identified as internal sources within systems. Specification and requirement weakness, design mistakes, and manufacturing issues are major sources of systematic failures which result from flawed redesign and modification procedures. Other failures occur due to insufficient requirements and design weaknesses for reliability and incomplete coverage of tests. Poor monitoring and maintenance are other sources of failures which are major contributors to systematic failures. Human errors in different phases are another source of systematic failures. Software bugs and errors, which are classified as internal causes, are a major source of common cause failures [3]. Recent studies have found that software failures are the cause of most system outages [4].

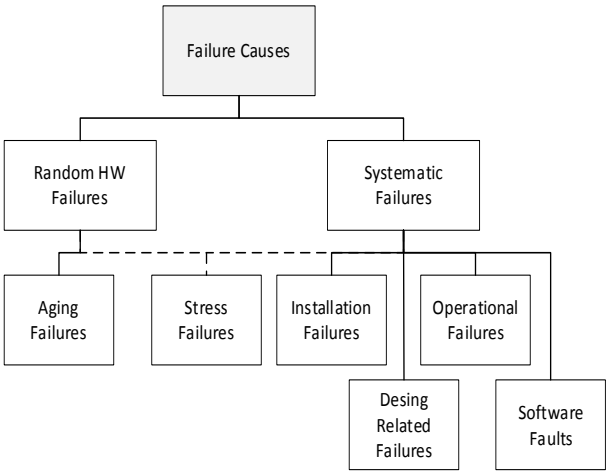

Fig. 1: Failure classification IEC-61508 [1]

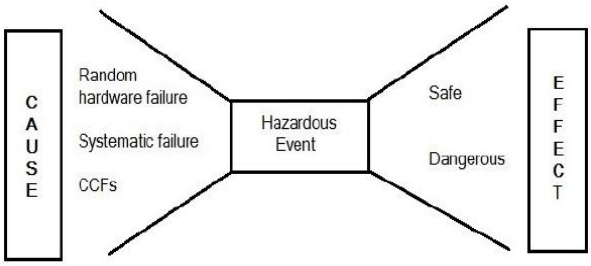

Fig. 2: Failure classification scheme [11]

A coupling factor explains why several components are affected by the same root cause (see Fig. 3). Coupling factor is a property that makes multiple components susceptible to failure from a single shared cause such as same design, same hardware, same software, same installation staff, same maintenance or operation staff, same procedures, same environment, and same location.

Diversity in redundant systems means that the redundant modules are functionally the same but they vary in different aspects: different implementation, different hardware, different software platforms, different materials, and different manufacturers [6]. Principally, diversity removes common features and similarities in redundant modules, and therefore, greatly protects the system against dependent failure mechanisms, and particularly provides protection against inherent dependencies and human error related dependencies. Taking diversity into account at all stages of design and implementation using parts and components made by different manufacturers, many of common causes of failures due to design mistakes and implementation bugs can be avoided. Redundancy along with diversity is more effective when failures occur randomly, "failures that can occur unpredictably during the lifetime of a hardware element" [7] [8]. They are less effective when failures are due to wear out or when failures are systematic, "failure related in a deterministic way to a certain cause." [7] [8]. For example, if failure is due to corrosion, two identical systems with the same materials will corrode at the same time. While two diverse systems with different materials will corrode differently, and therefore, provide more reliability. Due to the incomparable role of diversity in increasing reliability, International Electrotechnical Commission (IEC) necessitates diversity in redundant systems. Safety Integrity Level (SIL) means the relative level of risk-reduction provided by a safety function, or to specify a target level of risk reduction. According to the SIL 3, the IEC standards expect to employ not only redundancy but also diversity to improve system reliability. At SIL 4 it should resolve the common cause failures by employing redundancy along with diversity in all parts of the system including sensors, actuators, and processing units.

Diversity can be applied in different ways for different systems. If for example, multiple redundant modules are used, it is preferable to connect them to different power resources through different mechanisms.

A software bug, which is initiated in a certain situation or condition, may appear in all redundant and synchronized modules with the same hardware and software platform simultaneously.

Most of passenger aircrafts benefit from triple-triple redundancy along with diversity in different parts of flight computer and controller [21]. The Primary Flight Computer, PFC, of a passenger aircraft, Boeing 777, employs both triple modular redundancy (TMR) and diversity together to cover the safety critical application requirement. In this system, three different processors (from different manufacturers AMD, Intel and Motorola) are used to implement the PFC of the Boeing 777 and their software has also been developed by three independent teams based on different operating systems [22]. The control surfaces are actuated using redundant and different servo systems and the flight conditions are acquired using different sensors with different technologies. The interface between the flight computer and the input/output devices are handled using redundant buses with different technologies and protocols [23]. This professional configuration provides a high level of reliability for passenger aircraft which enables them to be operational about 20 years without any catastrophic accidents [24]. Similarly, in deep space missions, the reliability of life support system must be high and its failure probability should be less than one in a thousand in a multi-year mission. The most feasible way to achieve the high reliability for these systems is to use diverse redundant designs to reduce or to eliminate internal system common cause failures [25]. Three, four, or five diverse redundant systems are considered in [25] for a reliable space life support to provide the sufficient reliability. In [26] a design diversity fault tolerance technique is applied to a mixed-signal system: three different and diverse implementations of a second order low-pass filter (diverse TMR). Their simulation results show that the design diversity is a feasible technique that can increase the system reliability. Several TMR based systems on FPGA were implemented using different topologies from different design techniques, and their reliability was studied accordingly in [27]. Results indicate that diverse-design-based TMR systems show higher reliability to common mode failures exposure, and using different design techniques offers improved reliability for randomly injected faults at minimal additional cost and effort. A novel multi-objective problem for reliability allocation of a redundant system is formulated in paper [10] with the objectives of minimizing the system cost and maximizing the system reliability in the presence of common cause failures. In this article, two types of common cause failures are explored, namely fatal and non-fatal CCFs, in addition to random failures. These types of failures lead to simultaneous failure of redundant modules because of a common cause.

The Satellites need to function properly without interruptions for several years in space without any maintenance. While the commonly used techniques employed for ground systems cannot be directly used for satellites, fault-tolerance needs to be achieved without much penalty in weight and computational requirements. Diversity has been employed in space systems specially satellites. The attitude determination and control system (ADCS) employs diversity in different subsystems. In sensing a subsystem, different types of sensors are used to measure and estimate the attitude [28]. It also uses various actuators to control the satellite to the desired attitude.

Researchers recognize both the essential and accidental difficulties and problems of producing software [28]. Essential difficulties originate from the inherent challenges of understanding a complex application and operating environment, and from having to construct a structure comprising an extremely large number of states, with very complex state-transition rules. Accidental difficulties in producing a good software program arise from the fact that people make mistakes in even relatively simple tasks [28]. In general, fault tolerance in the software domain is not as well understood and matures as fault tolerance in the hardware domain. Software does not degrade with time. Its failures are mostly due to the activation of specification or design faults in a specific and deterministic condition by the input sequences. Therefore, if a fault exists in software, it will appear the first time that the relevant conditions of software bug occur. This makes the reliability of a software module dependent on the environment that generates the input to the module over time. Different environments might result in different reliability values.

However, software is inherently different from hardware. Software is a major source of common cause failures [3]. The authors of the study [29] investigated the behavior of flight software of NASA missions to determine their failure conditions. This includes multiple software applications, consisting of millions of lines of code in over 8000 files. The results show that the most common sources of failures were requirements and coding faults, each contributing to about 33 percent of the failures [30]. The traditional hardware fault tolerance techniques, such as hardware/software redundancy, were developed to mitigate primarily permanent component faults, and secondarily transient faults caused by environmental factors [5]. Although these techniques are efficient in hardware domain, they do not offer sufficient protection against design faults, which are dominant in software domain. Obviously, we cannot tolerate a fault in a software module by simply triplicating similar modules and voting their results, because all copies have identical design faults. Instead, each of the redundant modules has to be re-implemented in a different way and preferably using components from different developer teams. Multi-version techniques use multiple versions of the same software component, which are developed following design diversity rules consisting different teams, different coding languages, or different algorithms to minimize the

probability of common mode failures. The design diversity techniques applied to software design aims to build program versions that fail independently and with low probability of coincidental failures [31].

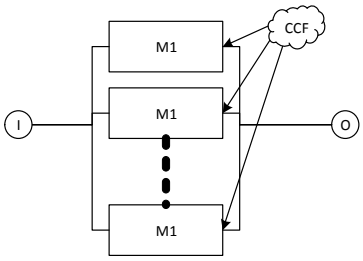

Fig. 3: A redundant system with N similar spares affecting CCF

If this goal is achieved, the probability of not being able to select a good output at a particular point during the program execution is greatly reduced or eliminated.

CCF and The Upper Bound of Reliability

Assume a redundant system with *N* ideal spares and the reliability of each redundant module be $R = 1 - F$ (see Fig. 4). Without CCF, the reliability of *N* redundant system becomes:

$$R_{Red} = 1 - F^N \quad (1)$$

And with a large number of spare modules and without CCF we have:

$$\lim_{N\to\infty} R_{Red} = \lim_{N\to\infty} (1 - F^N) \approx 1 \quad (2)$$

According to [3] we assume *10* percent of failures are CCF, and the remaining is the individual failures of each module. The redundant modules are similar. Therefore, the reliability of redundant system with N modules becomes:

$$R_{CCF} = 1 - 0.1F - (0.9F)^N \quad (3)$$

And for large *N*, the reliability is computed as follows:

$$\lim_{N\to\infty} R_{CCF} = \lim_{N\to\infty} (1 - 0.1F - (0.9F)^N) \approx 1 - 0.1F$$

Which is smaller than the case without CCF and does not allow the system reliability to exceed $1 - 0.1F$. In other words, $1 - 0.1F$ is the upper bound of reliability of a redundant system without diversity. As a result, diversity is essential for mission and safety of critical systems such as passenger aircrafts, satellites, and manned spacecraft's to become as an ultra-reliable system.

CCF Modelling

The β-factor model [31] is still the most widely used CCF model which requires only one extra parameter, β. Due to its simplicity, the beta-factor model is often recommended in practical applications of reliable systems. IEC 61508 [8] recommends using the β-factor model to evaluate the reliability of safety instrumented systems. In this method two kinds of failure rate are introduced for the component of a redundant system: independent failure rate (λ_i) and common cause failure rate (λ_c). Therefore, the total failure rate for each component is computed as:

$$\lambda = \lambda_i + \lambda_c \quad (4)$$

And the β factor can be expressed by:

$$\beta = \frac{\lambda_c}{\lambda} \rightarrow \lambda_i = (1 - \beta)\lambda \quad (5)$$

Determining β-Factor for CubeSat OBC

Satellites are categorized as mission critical systems with long mission duration time. According to the tasks assigned to their OBC, they must have a fault tolerant architecture. Various types of redundant architectures are used for satellite OBCs that make them vulnerable to CCFs. Since the satellite OBC can tolerate short interruptions, the cold sparing mechanism (standby) is usually used to provide the required reliability for longer mission time. Accordingly, the failure probability of the operating component would be calculated from equation (6) while the failure probability of the standby component would be calculated from equation (7) [32]:

$$P(t) = \frac{\lambda}{\lambda}(1 - \mathrm{e}^{-(\lambda)t}) \quad (6)$$

$$P_{mean} = \frac{\lambda\tau}{2} + \lambda\, MTTR \quad (7)$$

Where λ is the component failure rate and τ is the test interval for the standby component. We use beta-factor to model CCFs in which its parameter can be estimated through expert judgments, checklists, estimation models, and historical data. There are two IEC checklists, IEC 61508-6 and IEC 62061 checklists, for determining the Beta-factor. To estimate β in this approach, about 40 specific questions have to be evaluated and answered. Because of the lack of useful historical data for satellite OBC, in this paper we use an estimation model to determine *β*: Unified Partial Beta-Factor Method (UPM) [33]. The UPM identifies eight factors that are important for the parameter *β*. These factors, shown in Table 1, are grouped in design, operation, and environment which are weighted based on our experts' judgment and some knowledge and statistical data from the history of small-satellites [34].

Table 1: Determining β-factor for satellite

Factor	Sub Factor	Weights [=]				
		a	b	c	d	e
Design	Separation	2400	580	140	35	8
	Similarity	1750	425	100	25	6
	Complexity	1750	425	100	25	6
	Analysis	1750	425	100	25	6
Operation	Procedures	3000	720	175	40	10
	Training	1500	360	90	20	5
Environment	Control	1750	425	100	25	6
	Tests	1200	290	70	15	4

We applied this approach for a satellite OBC with two layer redundant architecture (see Fig. 5). In the first layer, it has a TMR architecture with diversity in hardware and software. Therefore, it has the least similarity while its complexity has increased. At the second layer, it has dual redundant architecture with cold sparing mechanism. The cold sparing mechanism leads to a very good isolation between modules. According to these explanations, the β-factor is determined in Table 1 and the result is β=0.057. It means about 5.7 percent of failures are considered to be CCF.

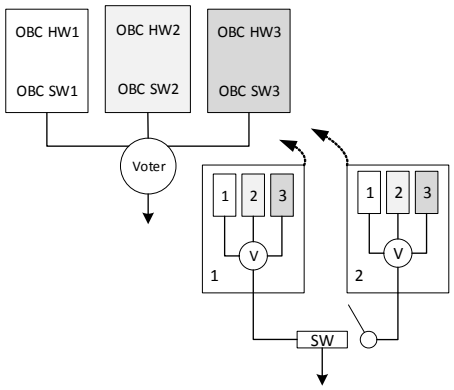

Fig. 4: Satellite OBC redundant architecture

Determining β-Factor for Spacecraft FC

Manned spacecraft has a highly reliable flight computer with multi layered redundant architecture. The spacecraft flight computers are constituted of multi-lane processing units and each lane has a redundant architecture. Usually the primary flight computer (PFC) has a TMR architecture taking advantages of diversity in hardware and software.

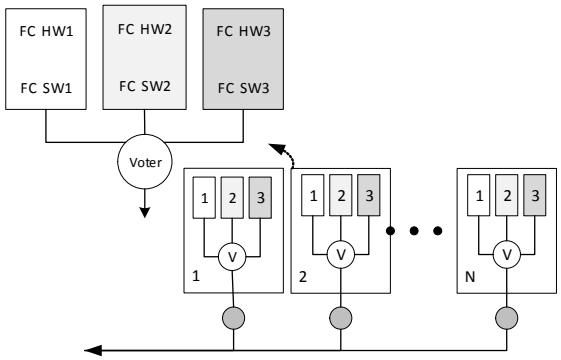

Fig. 5: Spacecraft FC redundant architecture

The manned spacecraft design is mainly based on the aircraft and compared to the satellites, the aircraft is subject to much stricter checks and inspections, and the relevant standards are the result of years of experience and careful investigation of air accidents. Therefore, the maturity of processes, training, control, and testing in this area is more than satellites and fewer failures are expected to lead to disaster. Unlike satellites, where it is impossible to access the relevant space accidents, air accidents are usually studied very carefully and the results are provided to the relevant industries in the form of process and troubleshooting standards. According to these considerations, the β-factor is determined as shown in Table 2 and the result is β=0.029. It means about 2.9 percent of failures are considered to be CCF.

Table 2: Determining β-factor for spacecraft

Factor	Sub factor	Weights [=]				
		a	b	c	d	e
Design	Separation	2400	580	140	35	8
	Similarity	1750	425	100	25	6
	Complexity	1750	425	100	25	6
	Analysis	1750	425	100	25	6
Operation	Procedures	3000	720	175	40	10
	Training	1500	360	90	20	5
Environment	Control	1750	425	100	25	6
	Tests	1200	290	70	15	4

CCF in TMR Systems

In this section, the reliability of a triple modular redundant (TMR) system, which is the basic structure for satellite OBCs (Fig. 4) and spacecraft FCs (Fig. 5) with and without hardware/software diversity has been investigated using Markov chain according to values obtained for β in the previous sections. In [38] a Markov based component wise sensitivity analysis method is applied to evaluate the importance measures of components and subsystems. The Markov chain provides the possibility of modelling reliability in redundant systems and allows considering redundancy and diversity in different levels. The Markov property implies that in order to predict the future state of a system with Markov property, it is sufficient to know its present state. This freedom from the need to store the entire history of the process is of great practical importance: it makes the problem of analyzing Markovian stochastic processes tractable in many cases. The probabilistic behavior of a Markov chain can be described as follows. Once it moves into some state i, it stays there for a length of time that has an exponential distribution with parameter λ_i. This implies a constant rate λ_i of leaving state i. The probability that, when leaving state i, the chain will move to state j (with $j = i$) is denoted by P_{ij} ($j=i$ $P_{ij} = 1$). The rate of transition from state i to state j is thus $\lambda_{ij} = P_{ij}\lambda_i$ ($j=i$ $\lambda_{ij} = \lambda_i$). We denote by $P_i(t)$ the probability that the process will be in state i at time t, given it started at some initial state i_0 at time 0. Based on the above notations, we can derive a set of differential equations for $P_i(t)$ ($i = 0, 1, 2, \ldots$), and then, the system reliability can be computed.

TMR System without diversity

Consider a TMR system without hardware/software diversity that has triple redundant and active processors (Fig. 4 and Fig. 5). Let λ_h be the fixed failure rate of each of the processors hardware, λ_s be the fixed failure rate of each of the processors software, and λ_c $(= \beta\lambda)$ be the fixed common cause failure rate of the processors. The corresponding Markov chain is shown in Fig. 6. The state ID represents the number of good processors. Since the redundant modules run in synchronous manner, their software experiences the same states and condition. Software failures are due to design mistakes and bugs which appear in certain conditions. Therefore, if the failure situation appears in one module, the other consistent and synchronized modules will also be in the same situation leading to the simultaneous failure of software in all modules. Therefore, if a software failure occurs, it leads to the failure of all redundant modules and obviously, the system fails.

The hardware part of redundant modules has a distinguished and particular failure rate and a common cause failure rate. The distinguished failure rate affects each redundant module individually (separately); while the common cause failure affects all redundant modules simultaneously (in β-factor model). Therefore, the common cause failure leads to the system failure. According to these descriptions, the TMR system without hardware/software redundancy has four distinguishable states:

- S3: All three modules are healthy. System is operational.
- S2: Two modules are healthy. System is operational.
- S1: Only one module is healthy. System is non operational.
- S0: No module is healthy. System is non operational.

As we described, the software failure or the common cause failure leads to the system failure, while the hardware failure causes one module failure in each state. Therefore, neglecting some details, the Markov chain of a TMR system without diversity is obtained as shown in Fig. 6.

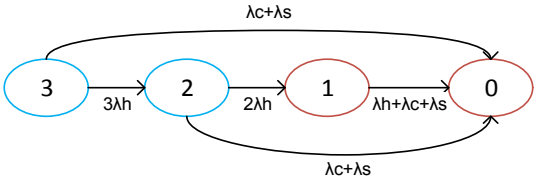

Fig. 6: Markov chain of a TMR system without HW/SW diversity

According to the Markov chain of a TMR system without diversity (Fig. 6) the set of differential equations are obtained as (8).

$$\begin{cases} \frac{d}{dt}P_3(t) = -(3\,\lambda_h + \lambda_s + \lambda_c)P_3(t) \\ \frac{d}{dt}P_2(t) = (3\,\lambda_h)P_3(t) \\ \qquad -(2\lambda_h + \lambda_s + \lambda_c)P_2(t) \\ \frac{d}{dt}P_1(t) = (2\,\lambda_h)P_2(t) \\ \qquad -(\lambda_h + \lambda_s + \lambda_c)P_1(t) \\ \frac{d}{dt}P_0(t) = (\lambda_s + \lambda_c)P_3(t) \\ +(\lambda_s + \lambda_c)P_2(t) + (\lambda_h + \lambda_s + \lambda_c)P_1(t) \end{cases} \quad (8)$$

When the system is in states 3 and 2, the system is operational; otherwise, the system fails and cannot continue its dedicated mission. Therefore, the system reliability without diversity is equal to $R_{TMR-nondiv}(t) = P_3(t) + P_2(t)$

TMR System with Diversity

In a TMR system with diversity, the hardware is implemented using different hardware architectures from different vendors such as Intel, PowerPC, AMD, and Mac. Software diversity can also be applied at different levels, but it is more efficient at higher levels of abstraction [5]: varying the algorithms is more efficient than varying the implementation details such as using different programming. Therefore, it is assumed diversity in the software has been applied at different levels of abstraction to minimize the failure dependency in software domain. Hence, in addition to varying the algorithm and design, the software is developed by different teams based on the same specification and requirements, but on different software platforms and different operating systems. The development environment must also be different and the operating systems must preferably be dissimilar. Therefore, with these considerations, software failures do not lead to the system failure, although, they are all in the same situation. In this case, the failure rate of each redundant module is the sum of hardware failure and software failure (see Fig. 7) and the common cause failure is negligible. According to these descriptions, the Markov chain of the TMR system with diversity is obtained as Fig. 8.

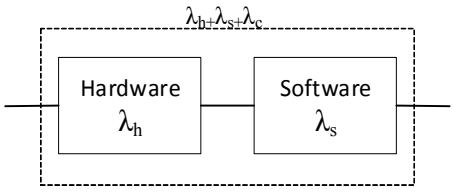

Fig. 7: Serial dependency of hardware and software in a processing unit

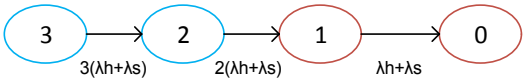

Fig. 8: Markov chain of a TMR system with HW/SW diversity

According to the Markov chain (Fig. 8), the set of differential equations are obtained as follows (9).

$$\begin{cases} \frac{d}{dt}P_3(t) = -3(\lambda_h + \lambda_s + \lambda_{c1})P_3(t) \\ \frac{d}{dt}P_2(t) = 3(\lambda_h + \lambda_s + \lambda_{c1})P_3(t) \\ \qquad\qquad -2(\lambda_h + \lambda_s + \lambda_{c1})P_2(t) \\ \frac{d}{dt}P_1(t) = 2(\lambda_h + \lambda_s + \lambda_{c1})P_2(t) \\ \qquad\qquad -(\lambda_h + \lambda_s + \lambda_{c1})P_1(t) \\ \frac{d}{dt}P_0(t) = (\lambda_h + \lambda_s + \lambda_{c1})P_1(t) \end{cases} \quad (9)$$

In this approach, the system is operational, when it is in states 3 and 2; otherwise, it is nonoperational, or it fails. Therefore, the system reliability with diversity is equal to: $R_{TMR-div}(t) = P_3(t) + P_2(t)$

Results

The investigated TMR systems are analyzed for two application types: safety critical application (manned spacecraft) and mission critical application (telecommunication satellites). For the safety critical application, such as airplane and manned spacecraft, the probability that the system must be operational at the end of a 3-hour mission time is $R_{plane}(3\text{ hour}) = 0.9999999 = 0.9_7$ [5]. Therefore, failure rate and MTTF are computed as follows for these safety critical applications:

$$\lambda_{plane} = \frac{-\ln(R_{plane})}{t_{misson}} \quad (10)$$

$$\lambda_{\text{plane}} = 3.33333 \times 10^{-8}\ \text{failures/hour} \quad (11)$$

$$MTTF_{plane} = 3.\times 10^7\ hour \quad (12)$$

The manned spacecraft failure rate λ_{plane} is obtained and its MTTF is about 3422 years. For a complex system, which constructed from many serial-configuration subsystems or modules, the reliability of each module must be greater than the system reliability to cover the high reliability requirements. For the mission critical application, such as telecommunication or GNSS satellite, the probability that the system must be operational at the end of mission (e.g. 10 years) is $R_{sat}(10\text{ year}) = 0.95$ [5]. Therefore, failure rate and MTTF are computed as follows for these applications:

$$\lambda_{sat} = \frac{-\ln(R_{sat})}{t_{misson}} \quad (13)$$

$$\lambda_{sat} = 0.00512933\ failures/year \quad (14)$$

$$MTTF_{sat} = 194.957\ year \quad (15)$$

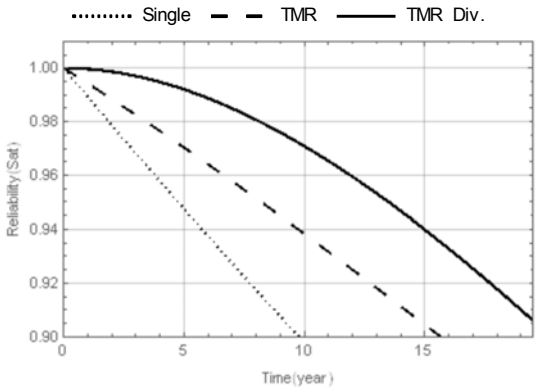

Fig. 9: Reliability of a TMR system (satellite case) with and without HW/SW diversity versus time.

The reliability of a TMR system (satellite case) with and without diversity is shown in Fig 10 and Fig 11. As it can be seen, the reliability of a simple TMR system is severely affected by the common cause failure and leads to diminish rapidly with time. Especially at the beginning of the mission, the reliability of these two systems (with and without diversity) gets apart quickly. However, the reliability of the TMR system with HW/SW diversity has been improved with a large difference with respect to the TMR system without HW/SW diversity.

The reliability improvement due to diversity is much more evident for safety critical applications. Actually, diversity has the greatest improvement of reliability over short times. This finding is also confirmed in paper [3]. 0Fig. 9 and Fig. 10 show the reliability of a TMR system (spacecraft case) with and without HW/SW diversity versus time. As it can be seen in the figure, for short mission time applications, the reliability improvements due to the diversity are considerable and vital, while for longer time (about 0.2 MTTF), the reliability improvements diminish. It can be concluded that, the reliability requirement of a safety critical application cannot be solely covered by redundancy.

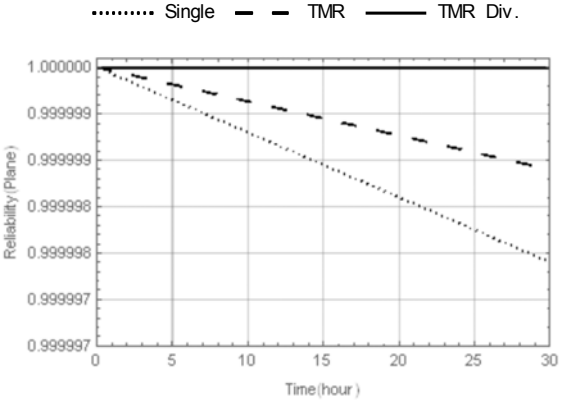

Fig. 10: Reliability of a TMR system (spacecraft case) with and without HW/SW diversity versus time.

Conclusion and Future Works

Reliability improvements due to diversity are investigated in redundant systems through appropriate modelling of diversity feature in HW/SW modules. The proposed scheme is explored in two application types: mission critical applications (with long mission time) and safety critical applications (with short mission time). While redundancy is suggested for reliability improvements, CCFs necessitate removing dependency in redundant modules through diversity. β-factor model is used to model CCF in these two types of applications. Due to the lack of historical data, the β has been determined through UPM method using expert judgments. Accordingly, the calculated dependent and independent failure rates are applied to the Markov chain reliability model. The results show considerable improvement due to diversity in both cases, especially in short mission time applications. The results also show the essential rule of diversity in these applications. Our investigations show that redundancy along with diversity can cover the reliability requirements for safety critical applications. Therefore, diversity is essential and indispensable for these applications.

Conflict of Interest

There is no conflict of interest by the authors.

References

[1] I. E. Commission, "Functional safety of electrical/ electronic/programmable electronic safety-related systems," in *IEC 61508*, 2010, p. Parts 1-7.

[2] M. Rausand, A. Barros and A. Hoyland, System reliability theory: models, statistical methods, and applications, John Wiley & Sons, 2020.

[3] H. W. Jones, "Common cause failures and ultra reliability," in *42nd International Conference on Environmental Systems*, 2012.

[4] K. S. Trivedi, M. Grottke and E. Andrade, "Software fault mitigation and availability assurance techniques," *International Journal of System Assurance Engineering and Management,* vol. 1, no. 4, pp. 340-350, 2011.

[5] E. Dubrova, Fault-Tolerant Design, KTH Royal Institute of Technology, Sweden: Springer, 2013.

[6] O. PIGI, Diversity, redundancy, segregation and layout of mechanical plant, Office for Nuclear Regulation, 2008.

[7] *ISO 26262 Road vehicles – Functional safety,* ISO.

[8]*IEC 61508. Functional safety of electrical/electronic/ programmable electronic safety-related systems,* Geneva: International Electrotechnical (IEC), 1997.

[9] S. Mitra, N. R. Saxena and E. J. McCluskey, "Design Diversity for Redundant Systems," vol. 4, no. 6, 1999.

[10] A. Samanta and K. Basu, "Multi-objective reliability redundancy allocation problem considering two types of common cause failures," *International Journal of System Assurance Engineering and Management,* vol. 10, no. 3, pp. 369-383, 2019.

[11] M. A. Lundteigen, *Hardware safety integrity (HSI) in IEC 61508/IEC 61511, Lecture slides,* 2006.

[12] D. K. Pradhan, Fault-tolerant computer system design, vol. 132, Englewood Cliffs: Prentice-Hall, 1996.

[13] M. Ram, "On system reliability approaches: a brief survey," *International Journal of System Assurance Engineering and Management,* vol. 4, no. 2, pp. 101-117, 2013.

[14] G. Kahe, "Reliable flight computer for sounding rocket with dual redundancy: design and implementation based on COTS parts," *International Journal of System Assurance Engineering and Management,* vol. 8, no. 3, p. 560–571, 2017.

[15] G. Kahe, "Triple-Triple Redundant Reliable Onboard Computer Based on Multicore Microcontrollers," *International Journal of Reliability, Risk and Safety: Theory and Application,* vol. 1, no. 1, pp. 17-24, 2018.

[16] A. Avizienis and J. P. Kelly, "Fault tolerance by design diversity: Concepts and experiments," *Computer,* vol. 8, pp. 67-80, 1984.

[17] J. P. J. Kelly, T. I. McVittie and W. I. Yamamoto, "Implementing design diversity to achieve fault tolerance," *IEEE Software,* vol. 8, no. 4, pp. 61-71, 1991.

[18] M. Hunger and S. Hellebrand, "The impact of manufacturing defects on the fault tolerance of TMR-systems," in *IEEE 25th International Symposium on Defect and Fault Tolerance in VLSI Systems*, 2010.

[19] S. Mitra, N. R. Saxena and E. J. McCluskey, "A design diversity metric and reliability analysis for redundant systems," in *International IEEE Test Conference*, 1999.

[20] J. H. Lala and R. E. Harper, "Architectural principles for safety-critical real-time applications," *Proceedings of the IEEE,* vol. 82, no. 1, pp. 25-40, 1994.

[21] Y. Yeh, "Triple-triple redundant 777 primary flight computer," in *IEEE Aerospace Applications Conference*, Aspen, CO, USA, 1998.

[22] J.D.Aplin, "Primary flight computers for the Boeing 777," *Microprocessors and Microsystems,* vol. 20, no. 8, pp. 473-478, 1997.

[23] Y. Yeh, "Design considerations in Boeing 777 fly-by-wire computers," in *Third IEEE International High-Assurance Systems Engineering Symposium (Cat. No.98EX231)*, Washington, DC, USA, 1998.

[24] J. Fielding, Introduction to aircraft design, Cambridge University Press, 2017.

[25] H. W. Jones, "Diverse Redundant Systems for Reliable Space Life Support," in *45th International Conference on Environmental Systems*, Washington, 2015.

[26] L. F. G. T. R. B. M. L. Gabriel de M. Borges, "Diversity TMR: Proof of Concept Diversity TMR: Proof of Concept," 2010.

[27] R. A. Ashraf, O. Mouri, R. Jadaa and R. F. DeMara, "Design-For-Diversity for Improved Fault-Tolerance of TMR Systems on FPGAs," in *International Conference on Reconfigurable Computing and FPGAs*, 2011.

[28] I. Koren and C. M. Krishna, Fault-tolerant systems, Elsevier, 2010.

[29] M. Hamill and K. Goseva-Popstojanova, "Common trends in software fault and failure data," *IEEE Transactions on Software Engineering,* vol. 35, no. 4, pp. 484-496, 2009.

[30] L. Bergmane, J. Grabis and E. Žeiris, "A Case Study: Software Defect Root Causes," *Information Technology and Management Science,* vol. 20, no. 1, pp. 54-57, 2017.

[31] K. N. Fleming, "Reliability model for common mode failures in redundant safety systems," No. GA-A-13284. General Atomics, San Diego, CA, United States1974.

[32] J. D. Andrews and T. R. Moss, Reliability and Risk Assessment,, New York: The American Sociery of Mechanical Engineers, 2002.

[33] B. D. JOHNSTON, "A structured procedure for dependent failure analysis (DFA)," *Reliability Engineering*, vol. 19, no. 2, p. 125–136, 1987.

[34] M. Langer and J. Bouwmeester, "Reliability of cubesats-statistical data, developers' beliefs and the way forward," in *30th Annual AIAA/USU Conference on Small Satellites*, 2016.

[35] R. A. Humphreys, "Assigning a numerical value to the beta factor common cause evaluation," *Reliability'87*, 1987.