International Journal of Natural and Engineering Sciences 9 (3): 39-44, 2015
ISSN: 1307-1149, E-ISSN: 2146-0086, www.nobel.qen.tr

Issues and Challenges of Automated Software Fault Tolerance Techniques

MEHREEN[1*] Salman QADRI[2] Husnain AHMAD[2] Muhammad FAHAD[2]
[1] Department of Computer Science, Virtual University of Pakistan, Lahore, Pakistan.
[2] Department of Computer Science and Information Technology, The Islamia University of Bahawalpur, Pakistan

*Corresponding author:
Email: ms110400100@vu.edu.pk

Received: September 04, 2015
Accepted: October 09, 2015

Abstract
In software, faults can occur and may disturb the normal behavior or working on a system. Software fault tolerance techniques must implement to remove these faults. Fault tolerance defined as how the system fights when faults occur. A system cannot be truly fault tolerant until software fault tolerance techniques are applied to it. In early researches it is estimated that almost 60-90% of system failures are attributed to software failures. These failures can be controlled by applying software fault tolerance techniques. In this study, a comparative study of different software fault tolerance techniques (like Swift, Trump, Mask and N-version programming) is carried out and current research challenges in these techniques is also discussed.
Keywords: Design Diversity, N-Version Programming (NVP), Swift, Trump, Mask.

INTRODUCTION

Software Engineering provides the mechanism for the development of software by applying the step by step procedure to produce high quality product finally. It guarantees different aspects of software to work as designed by the engineers and allows the architects to improve the quality of the software development efforts. Nowadays, computer can perform human activities quickly more than humans. But computer is a machine which can lead to failure at any time that's why we cannot depend on it completely. Machine failure's means that it gives any kind of error or fault in the result of a program. There are two types of fault: Software and Hardware. This research is about software faults only, different software fault tolerance techniques and issues of these techniques. This research mainly focuses on identification of those issues from different work areas and will provide the solution to some of those identified issues.

Software faults also known as flaw, bug and failures. Types of software faults are Arithmetic faults, Logic faults, Syntax faults, Resource faults, Interfacing faults and Team working faults. The system generates wrong and faulty results due to these faults. Software containing design faults cannot fulfil the requirement of the software and that's why it generate faulty results or will show wrong behavior without any hardware failure. Human made errors are also residing in this category because if programmer made some mistake at the time of designing the system, this mistake is known as a design fault. Any faults which are intentional or unintentional and create a problem at the designing level also resides in the category of design faults. A program goes into error state when a fault occurs during execution instead a program works according to its requirements. Considering this problem, class of faults can be modelled so that it will understand the behavior of the system and correct it [1].

Jim Gray explains two types of faults: Bohrbug and Heisenbug. Bohrbug are the errors occurs during the design phase, which are stable and immovable and identified during testing and debugging phase of the SDLC and Heisenbug are not fixed internal faults, they belong to class of temporary internal faults [2].

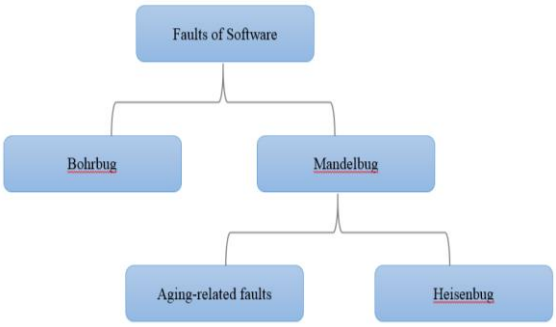

Figure 1. Comprehensive classification of Software Faults.

Fault Tolerance

Fault tolerance means that if a fault occur, but it's bearable by the system and system do not go to system failure [3]. The computing system should be reliable and reliability can be attained through when system tolerate the fault at maximum level. Two methods ensure the reliability which is fault removal and fault avoidance and in these cases there is no need of fault tolerance approach. These two approaches provide reliability in both software and hardware system [4]. In the design phase, we may use fault avoidance, which stops the entrance of errors while designing the system [1]. The concept of fault tolerance has been introduced in 1950s[3]. At that time, for the hardware reliability, manufacturer use extra hardware to remove the fault from hardware. This technique is called hardware redundancy. Now the concept of hardware redundancy is used for software redundancy. Many researchers work on the concept of redundancy [1].The result of this work is the concurrent systems [5]. Many algorithms have been written for this redundant structure. When a fault occurs, the error state is recorded and some protocol is used to restore the failure state [1].

Software Fault Tolerance

When a software has the ability to find out and remove the faults from the system which are already occurring and it fulfill the requirement of the user. We can say, it tolerates the faults [3]. Mostly hardware fault tolerance techniques are used when a fault occurs in software. The major problem with these techniques that how they find out the problem at the time of execution of a software. The design diversity is not helpful for the faults of hardware. For the removal or tolerating the hardware fault, we have to apply replication of the hardware. It means we have to use the redundancy techniques.

Software Fault Tolerance Techniques

The major fault tolerance techniques are shortly discussed below:

Fault Avoidance

At the time of designing, we use fault avoidance techniques. By using these techniques, we can avoid fault before its occurrence. Fault avoidance gives us different techniques to avoid the entrance of faults while designing and coding of the system.

Fault Removal

At the end time of the development phase, fault removal methodology gives us opportunity to remove the faults and try to stop the initiation of new faults.

Fault Tolerance

In this methodology, faults has not removed. Faults exist in the system, but are bearable. The existing faults do not harm the system and give accurate performance and output.

Fault Prevention

This methodology stop the entrance of faults in the system. Fault prevention takes responsibility of the quality of the software.

Fault Forecasting

By using this technique, we can check the faults which can occur while using the system. Fault Forecasting checks the working of the system and it define the results of these faults before the existence of the faults.

Traditional software fault tolerance techniques

Single version software environment

Single version software fault tolerance (SVSFT) are techniques which checks the application software. It provides a decision algorithm for the verification of code and Exception handling is used for the fault tolerance from the system [6].

Multiple version software environment

Multiple Version Software Environments (MVSE) implement the mechanism by changing the design of software. It provides multiple version of the software and implement different software fault tolerance techniques on them. N-self checking Programming, N-version Programming and recovery blocks are example of MVSE [6].

Multiple data representation environment

Multiple Data Representation Environment (MDRE) implemented by using the data diverse techniques by applying various data representation techniques. MDRE is used to tolerate the faults of software at maximum level. N-self checking programming. N-copy programming and Retry Blocks are examples of MDRE [6].

Design diversity

The main purpose of design diversity is to decrease the software faults. This technique tolerates the faults which occur due to wrong specification and wrong piece of code [6]. There are some design diversity algorithms listed below:

1. Recovery Blocks (RcB)
2. N-Version Programming (NVP)
3. N Self-checking programming (NSCP)

Data diversity

Many faults can tackle through the design diversity techniques, but some remain in the software, due to the restrictions in the design diversity techniques. So, a new technique is introduced by Ammann and knight [6] named data diversity to overcome these remaining faults. Data re-expression algorithms used to input data in data diversity technique. Some data diversity algorithms are listed below:

1. Data Re-expression Algorithm (DRA)
2. Retry Blocks (RtB)
3. N-Copy Programming (NCP)
4. Adaptive N-version programming
5. Fuzzy Voting
6. Abstraction

Some software fault tolerance techniques have been introduced to overcome the problem of extra hardware, such as, design diversity techniques. The main purpose of design diversity is to decrease the software faults. This technique tolerates the faults which occur due to wrong specification and wrong piece of code [6].

Consequently, many techniques have been developed to tolerate the software design faults. The subsequent sections demonstrates a brief introduction to numerous techniques, issues, challenges and a modified version of NVP.

MATERIALS AND METHODS

Automation

In various ways automation has been used. Oxford English Dictionary describes automation as:

The manufacturer has the automatic control of the product form many succeeding stages.

The use of automatic control to each branch of the science or industry.

Moreover, the mechanical or electronic devices are used for replacing the human labor. The main purpose of using automation in varying control systems, process and technology is used to reduce the necessity of human intervention. In controlling process, the automated devices to implement used by the operator [7].

Automated Software Fault Tolerance Techniques

Automated Software fault tolerance are techniques that find out the faults occurs in a program without any help of the human user. It also provides solution of these faults and repair it without human interaction.

Many automated fault tolerance techniques are discussed below which protects the data at maximum level.Different automated fault tolerance techniques are:

1. Swift
2. Trump
3. Mask
4. Hybrid techniques
5. N-version programming

Swift

A new transient fault detection technique presentedby Reis known as SWIFT [8].Swift is an automated software fault tolerance technique which duplicates whole program's instructions and create a scheduler which schedule the original instruction only. It makes duplication by allocating different register to the original and duplicate version of the program without any interference of both these codes. A validation code is used by the swift to verify that the data generated by the original and duplicated are same to check that both original and duplicated code are generated same results.

Swift-R

Double-modular redundancy approach is used for actual Swift-R transformation to implement in software. Double redundancy finds out the faults, not recover the data. But in Swift-R, it provides the consistency of data with the recovery of data by proving triple modular redundancy. The benefit of three copies is that if one copy is changed or damaged, then other two are used to recover it with correct values. A special voting system is defined in the Swift-R to check which copy is damaged and which is correct. This approach may correct single-bit fault.

Trump

AN-codes is the theoretical mechanism behind the Trump methodology. Trump uses only two replicated registers of the Swift-R for further process. Trump's AN-encoding is not as Swift-R's triplication. Swift-R need more calculation and more space for replication, but Trump uses less space and fewer calculation of error detection. Trump gives alternate solution for error detection to reduce the performance drawbacks of the Swift-R which may be beneficial for protecting the system to damage.

Mask

The mask is a very cheap method to provide reliability. It implements fixed invariants to reduce faults which can occur after the execution. Mask can reduce the number of faults which can damage system, but it does not use the replication technique to remove the faults.

N-version programming

The NVP was introduced by A. Avizienis in 1975. In this approach, N-fold calculations are taken through using autonomously designed software modules which known as "version" in N-version programming and results of these calculations are directed to the decision algorithm that concludes a single decision result. The main different between recovery block and n-version programming is decision algorithm. In the recovery block approach, an acceptance test (AT) is conducted for the selection of the recovery block. But in the n-version programming approach, a selection algorithm is used for the implementation of the selection or for not selection.

"N-version programming is defined as the independent generation of N>=2 operationally same program from the similar initial specification."

Independent generation means that programs generated by different programmer which do not interact with each other as perspective of the process of programming.

The main purpose of N-version programming is to reduce the probability of the same kind of errors.

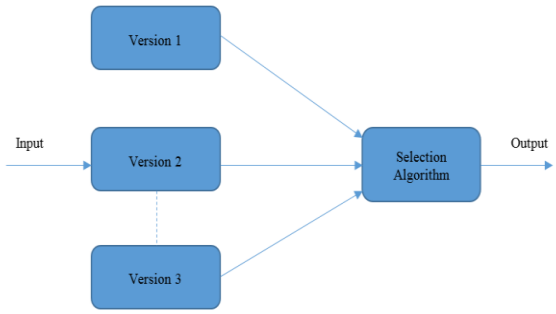

Figure 2. N-version programming model.

Modified N-version programming –A proposed model

In N-version programming, two or more versions are defined for the same functional specification of the software which accepted by applying some validation checks. The main objective of the N-version programming is to cover the effects of the software faults exists in the modules. There can be different problems occur due to writing N-versions of the software, i.e., the cost, time, complexity, quality and space, which will be discussed in detail in chapter No. 4. All of the mentioned problems can be overcome by applying N-version programming only on the critical paths of the software. The main model remained same for modified N-version programming as shown in Figure 3. The DFD of modified N-version programming shown below.

In modified N-version programming, there is no need of writing N-versions of the complete software. The programmers have to write versions only for the critical path of the software, which reduce the length of the code and will decline the complexity of the software. It also decreases the cost of the project, which was increased due to writing different versions of the whole software.

When a software starts its execution, it firstly go to the critical path algorithm which find out the critical path. If it finds the non-critical path, then simple execution started. There is no need of writing N-versions of the non-critical paths. If the critical path algorithm finds any critical path after starting the software execution, then it started that critical path execution. All the critical paths of the software have different versions with same functional specification and complexities. All versions executed and their results stored for final decision. After that, all the versions results gathered in a single place. All the N-versions results are compared for acceptance the result that will be done by voting mechanism. So, when all the versions generated the results, then these results are submitted to a voting process.

If that algorithm finds the acceptance of a specific result, then that version executed, complete its working and end the critical path. But there are two aspects that may cause the irregular end of a version:

For watch end: Sometimes the specified time is not enough for a version to complete the process and generate some result.

For error condition: Sometimes there may be an error generated by the operating system while execution of the version, such as, division by zero and an overflow.

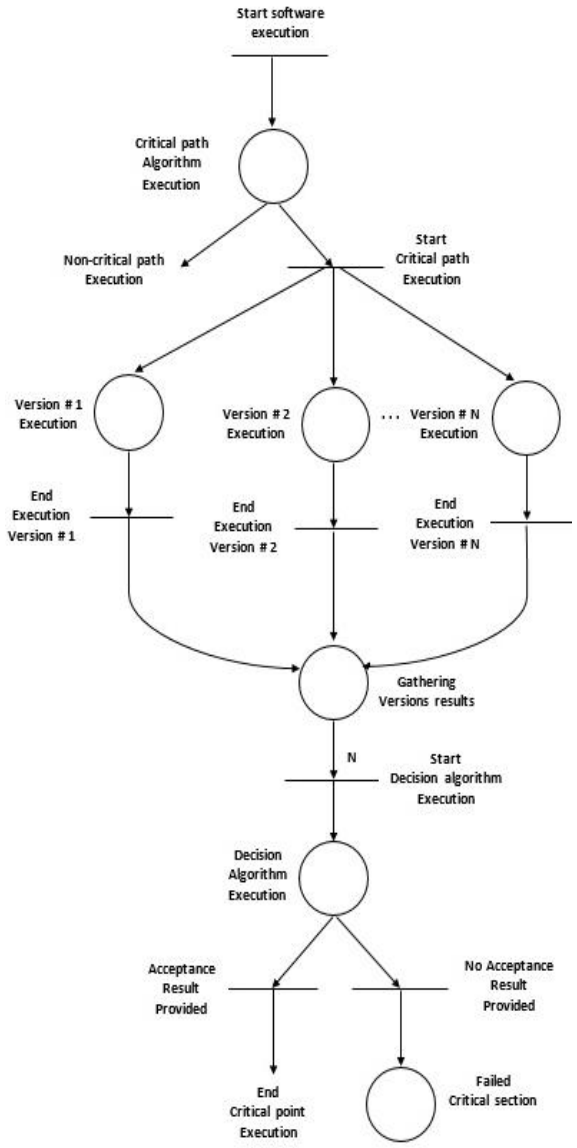

Figure 3. Modified N-version Programming.

RESULTS AND DISCUSSIONS

A survey has been conducted to take view of the issues and challenges face during the implementation of software fault tolerance techniques. Some issues are identified from the survey results which directly affecting the overall performance of a software. They may consume extra memory space to fulfill the requirement of techniques. The detail description of each issue is given below:

Quality Items missing

The team has the responsibility to complete a project in a short time with completing the requirement. Due to applying the Software Fault Tolerance Techniques in a project and completion of the project in time, team ignore the quality of the project or some quality items of the project. Due to this reason, the performance of the overall project will be affected.

Quality of Code

In normal condition, when the Software Fault Tolerance Techniques not applied in a project, then due to the short time of deadlines of the projects, the developers need to spend extra working hours to complete the project in time. When a developer works under strict time pressure, then obviously he/ she may not write bug free code. And when the Software Fault Tolerance Techniques applied in a project, then it requires extra skills and time to code a project and if the deadline is near then it is confirmed that the developer will miss the quality of the code. That is why the quality of the code will be affected.

Project Duration

Project duration is the main hurdle for implementation of Software Fault Tolerance techniques in a project. When a project deadline is short, the project manager does not decide to implement any Software Fault Tolerance Techniques and normally a simple mechanism used i.e. fault detection, fault avoidance etc.

Team Problem

In a team, the team member is both i.e. the beginners and the senior team members. In team member, some member has the knowledge of Software Fault Tolerance Techniques and some have not. Which is another issue has been observed from the survey. When whole team does not have the knowledge of these techniques, then the implementation of these techniques is not possible. This is one of the reasons that many software houses don't implement the software fault tolerance techniques in their projects.

Lack of SFT techniques skills and training

According to the survey, it can be noticed that 60% team members have no skills of any of the software fault tolerance techniques. There is no formal training is conducted to train the team members. The team members have only knowledge of these techniques which they learn from their colleagues which is not enough. That is why the adoption of these techniques is minimum in our software houses. Figure 4 shows that the training of SFT techniques is necessary.

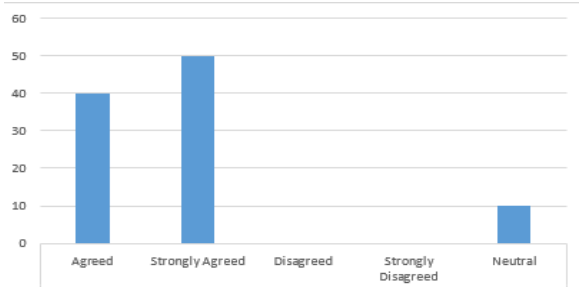

Figure 4. Need Training of SFT

High Cost

When we talk about the N-version programming, we know that in this technique many versions of a program created instead of a single piece of code which will increase the cost of the project because these versions may be of different environments and also affect the development cost.

Bharathi [9] also highlighted the issue of high cost due to implementing n-version programming. She says that due to generating N-versions of a program increases the cost of the software. The cost of development and supporting environment to complete the implementation will be increased.

Duplication

Most of the software fault tolerance techniques use the mechanism of duplication. In Swift, the whole program's instructions are duplicated and this is done by the duplication. In Swift-R, the whole program's instructions duplicated two times. It means double modular redundancy is used for this approach. So, while duplicating a whole program, maybe some problem occurs and it may possibly that some instruction will not duplicate due to this problem. And if we say that all duplication is done perfectly, but it increase the cost because extra registers are used for the redundancy.

More space

Most of the software fault tolerance techniques use the mechanism of duplication. In Swift, the whole program's instructions are duplicated and this is done by the duplication. In Swift-R, the whole program's instructions duplicated two times. It means double modular redundancy is used for this approach. So, while duplicating a whole program, maybe some problem occurs and it may possibly that some instruction will not duplicate due to this problem. And if we say that all duplication is done perfectly, but it increase the cost because extra registers are used for the redundancy.

Code complexity

By implementing the SFT techniques, the complexity of code has been increased. In Swift, the whole program's instructions are duplicated and this is done by the duplication. For duplication, the line of code will increase and the code of the whole program will become complex. Same as, In Swift-R, the whole program's instructions duplicated two times. So, the double modular redundancy approach has been used, which increase the line of code and increase the complexity level of code. Each SFT technique contains different voting algorithms, these algorithms increase the code complexity.

Selection algorithm

In n-version programming, a selection algorithm is used for the selection or not select a version. We know that, many versions have been programmed for a single function and all the versions have the same complexity level. The selection algorithm is actually the voter to select the output. The main issue is that the writing of a perfect selection algorithm is difficult which select a perfect match version.

Difficult maintenance

Those projects are difficult to maintain in which SFT techniques have been implemented in them. Because the complexity of code has been increased. Each SFT technique has a different mechanism for selection of versions or module. Different voting algorithms have been used to search that which module is correct.

Need of SFT

Implementation of Software Fault Tolerance Techniques is necessary for the most critical applications, where the loss of human life can happen or costly hardware can be damaged. The quick and accurate decision should be taken when a fault occurs and it should be guaranteed that the decision will be taken without any computational delays and recover the faults. SFT technique must be applied on long life applications, networks and high performance processing.

Work Load

From survey result and from figure 5, it is noticed that most developers think that when they implement software fault tolerance techniques in normal applications, the workload has been increased. Because they think that there is no need of software fault tolerance techniques in normal applications. The time and efforts have been increased when we implement these techniques in normal applications as per as the time duration is short.

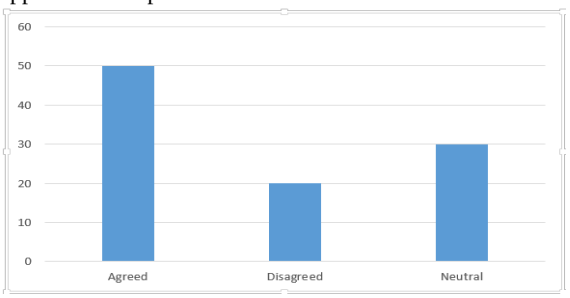

Figure 5. Work Load

Ambiguous specifications of requirements

This issue is not so obvious and not occur each time. N-version programming work successfully when the requirements have been accurate. If the specifications of requirements have been gathered wrongly, it is confirmed that the n-version programming will not give accurate results. Because in n-version programming, different version has been written for single function and these versions written by different programmers. If a programmer does not understand the requirements, then obviously the version generated by him will not be perfect. It will create problems.

Solutions for the Identifies Issues

Solutions to some of the above mentioned issues are proposed. These issues are:

Quality items missing

Need great attention to increase the reliability of the quality items and to attain the required goals i.e. good performance and high maintainability. The things which provide quality should not be disregarded at any cost because they helps to attain the factors such as reliability, maintainability and scalability.

Quality of code

The tasks should be divided into sub tasks to fulfill the deadline requirement and it will also resolve the issue of quality of code. Software fault tolerance technique should be implemented in every complex task which will ensure that the task will be completed without any fault. Due to division of tasks into subtasks, the complexity of code will also be reduced.

Project duration

The software fault tolerance techniques must be implemented in critical applications. So, when the SFT techniques applied in a project, the time duration should be maximized as necessary. And it is the core duty of project manager to divide the tasks accordingly to meet the deadline. So that all the functions coded and tested completely and project delivered on time.

Team problem

The selection of project team plays a very important role to reduce the team problem. When the project manager

assigns a project to a specific team, he must choose the team having the skills of all the techniques which applied in the assigned project. Through this check team problem can be resolved.

Lack of SFT techniques skills and training

To minimize this issue, it is very necessary that the team which develops a project should have the complete skills of these tools and techniques which they implement in this project. In our country, no proper training given to the team members. Software houses should be encourages new developers and trained them accordingly so that, they can do better work individually and with the team.

High cost

In n-version programming, one of the issues is the increase in development cost due to multiple versions have to be written for a single function. It can reduce by applying design diversity to the critical paths only. Due to applying this only on the critical paths, the cost will be decreased because design diversity not applied to the whole project and also the complexity will be decreased.

Selection Algorithm

The special attention will be given to the selection algorithms in the software fault tolerance techniques. While writing a selection algorithm for fault tolerant software, it is necessary to give special attention to the operating system because the cost of the software and its complexity both depend on the development of the software and complexity can be reduced by taking correct system functionality into consideration.

Difficult maintenance

When a project is too long and too complex, it is difficult to maintain if a problem occurs or change in requirement. So to resolve this issue, the complex tasks should be divided into sub tasks. Software fault tolerance technique should be implemented in every complex task which will ensure that the task will be completed without any fault. Due to division of tasks into subtasks, the testing can be done properly. Properly tested system may generate less problem and there is no need of maintenance.

Work load

The workload can be reduced as per as applying the fault tolerance techniques in normal applications. Each application have the critical path in it, if the software fault tolerance technique applied only on the critical path of the application then the benefits of fault tolerance techniques will be gained and the work load will be reduced and software will work without any fault.

Ambiguous specification of requirements

To overcome this issue while adopting n-version technique for the fault tolerance, extra meetings should be arranged with the developers and clear the specifications of requirements deeply. This will help the developer to understand the correct requirement and as a result, the version coded by the developer will give correct output.

CONCLUSION

We know that faults are the part of the software, but good software is one which tolerate these faults at maximum level and give the accurate output. Many software fault tolerance techniques have been proposed by many researchers. In this research thesis, different SFT techniques have been discussed in detail. The main purpose of this research is to find out the issues and challenges in software fault tolerance techniques. A survey was conducted to find out the issues of software fault tolerance techniques. The main issues of implementing SFT techniques in a project are: Quality Items missing, Quality of Code, Project Duration, Team Problem, Lack of SFT techniques, skills and training, High Cost, Duplication, More Space, Code Complexity, Critical Application, Selection Algorithm, Difficult Maintenance, Need of SFT, Work Load and Ambiguous Specification of Requirements. A solution of some of the issues have also been suggested in this research work, i.e., Quality Items missing, Quality of Code, Project Duration, Team Problem, Lack of SFT techniques, skills and training, High Cost, Selection Algorithm, Difficult Maintenance, Work Load and Ambiguous Specification of Requirements.

Future Work

In this paper, different issues are discussed in detail which can be arise while implementing the software fault tolerance techniques. A solution of some of the identified issues has been proposed. In future, the solution of remaining issues will be proposed and presented a new SFT technique or combination of existing SFT techniques in which all the identified issues will be removed.

REFERENCES

[1] Liu, Z. 1991. Fault-tolerant Programming by Transformations. Phd Thesis, Warwick Univ. (Coventry, UK).

[2] Vaidyanathan, K., and K. S. Trivedi. 2001. Extended Classification of Software Faults Based on Aging. In Proc. of 12th IEEE Int. Symposium on Software Reliability Engineering. (Hong Kong, China).

[3] Inacio, C. 1998. Software Fault Tolerance. Dependable Embedded Systems, Carnegie Mellon Univ. (Pennsylvania, US).

[4] Avizienis, A. 1976. Fault Tolerant Systems. IEEE Transactions on Software Engineering. C-25(12):1304-1312.

[5] Randell, B. 1975. System Structure for software fault tolerance. IEEE transactions on Software Eng. 1:220-232.

[6] Xie, Z., H. Sun and K. Saluja. 2008. A survey of software fault tolerance techniques. Dept of Electrical and Computer Engineering, Wisconsin Univ. (Wisconsin, USA).

[7] Gegentana. 2011. A systematic review of automated software engineering. M. S. Thesis, Dept. Comp. Sc. Chalmers Univ. (Goteborg, Sweden).

[8] Reis, G. A., J. Chang, N. Vachharajani, R. Rangan and D. I. August. 2005. SWIFT: Software Implemented Fault Tolerance. In Proc. of the 3rd Int. Symposium on Code Generation and Optimization. (Princeton, USA). pp. 243-254. ISBN: 0-7695-2298-X.

[9] Bharathi, V. 2003. N-Version programming method of Software Fault Tolerance: A Critical Review. NCNSD. (Karnataka, India).