A Survey on Fault Injection and Bebugging

Divya Khurana
Krishna Institute of Engineering and Technology, Ghaziabad, India.

Anand Prakash Shukla
Krishna Institute of Engineering and Technology, Ghaziabad, India

Abstract: In Development phase, a numerous faults may arise at any time in a system due to which many system failures report require attention because of unpredictability. So, we need to improve the quality of system and made system more reliable and enhance effectiveness of the system. In order to have highly reliable system, this paper will explore the software faults for a complex fault tolerance system through Software fault injection and its techniques. Nowadays, various fault injection tools and techniques are proposed and their behaviour respectively. The process of introducing faults in the system in order to assess the impact of these faults in other components or the whole system and measure the behaviour of the complex system. This paper shows a survey on several fault injection techniques that are categorised as hardware implemented based fault injection, software implemented based fault injection, simulation implemented based fault injection, and emulation implemented based fault injection and hybrid fault injection. Moreover, Fault seeding which is also a type of fault injection used for introducing faults in the system in order to predict the other remaining faults in the system. Furthermore, this paper presents the comparison between Fault injection Techniques and Fault seeding Technique.

Keywords: Faults, Software Fault tolerance, fault injection, fault seeding.

I. INTRODUCTION

There are numerous faults occurs in a system and it is practically impossible to guarantee that the system is perfect. A fault is condition that causes the software to fail to perform its vital task suspiciously. Fault can be occurs either in Hardware or Software. Faults may also be injected in the System, when a fault occurs in a system it causes some incorrect changes through which system cannot fulfil it complete specification [2]. There are various faults that cannot be detected during testing that remains in a system and causes issue in the system.

Some of the terms that are co-related with the term faults [15] which are as follows:

Error: The difference between the actual output and expected output.

Failure: System that is not capable to perform its specification.

Bug: Error, Failure or fault in computer program that causes it to produce an incorrect or unexpected result.

System deals with large Number of files and Function that make a system more complex to perform their task through which system behaviour will suffer. During Software Development Process, Faults that occurs in the system are tolerated or detected by Fault Tolerance Mechanism [14] that the Software developer can adopt. Furthermore, we accomplished, Fault injection which technique for injecting fault in a system in order to detect other faults in a system and recover them and made available the system to perform its particular tasks properly. This paper also discussed about some of the fault injection techniques. In recent few years, during design phase of the system, developers use fault injection technique to check the fault-tolerant system.

1.1 Categories Of Faults

Faults which are not removed or detected during testing that remains in the system and show their visibility in the performance and behaviour of system that causes inefficient performance and behaviour of system. Faults can be occurs in both Hardware and Software. According to specification of system, faults [1] can be categorised as Hardware or Physical Faults and Software Faults [2].

Hardware or Physical Faults:

A Hardware faults is a faults that arises in the actual component of the system. It is also known as Physical Fault. It can be predominantly improved by replacement and redesigning of that faulty component. Hardware faults can be classified as:

- **Permanent faults:** It is a Hardware break down that occurs when a particular set of conditions exists. It can be take place due to improper manufacture or misuse of component. It can be recovered by simply replacement of a component and its issues can be resolve by the manufacturing department.
- **Transient faults:** It can be arises due to some ecological conditions such as power fluctuation, momentary tree contact, lighting strike, conductor clashing, bird or other animal contact etc. These faults can generate network damage. It is difficult to become aware of but some can be resolved. Some of the system that stated as underground that typically observes these faults and that can be harder to resolve.

- **Intermittent faults:** It is a temporal fault that can be done for some time period. It is caused due to unstable hardware or varying state of hardware. It can be easily repair by alternative of component or redesign of that faulty component.

TABLE 1. Types of Faults

TYPES OF FAULTS	
HARDWARE FAULTS	**SOFTWARE FAULTS**
Permanent faults	Function faults
Transient faults	Algorithm faults
Intermittent faults	Checking faults
	Assignment faults
	Timing or serialisation faults

Software Faults:

A software faults is an incorrect stage or process in a computer program that exhibits the program to perform its tasks in an insufficient manner. In other words we can say that a software fault is a condition that occurs in software that causes a system fail to perform their specific tasks in an appropriate manner. Software faults are also known as Design faults.

Some of the software faults are as follows:

- **Function faults**: Function fault occurs when on implementing an incorrect or missing function statement or code in a program.
- **Algorithm faults**: This type of faults occurs when algorithm is missing or incorrect that causes erroneous program code.
- **Checking faults**: Faults that includes the missing checks or incorrect condition checks in software.
- **Assignment faults:** When an assignment is missing and incorrect either not given in software i.e. assignment faults.
- **Timing or serialisation faults:** Faults occurs when serialisation is missing or incorrect or repeatedly change in a program code.

II. SOFTWARE FAULT TOLERANCE

Software fault tolerance deals with the software to detect and recover the software from various types of hardware or software faults in order to perform its tasks in the running state consistently according to its requirement [8]. Software fault tolerance is essential component for a system that made available for future building systems in order to make software more reliable and highly available. According to Laprie in 1996, "Fault tolerance is defined as how to provide, by redundancy, service complying with the specification in spite of faults having occurred or occurring". During Software Development Process, Faults that occurs in the system are tolerated or detected by Fault Tolerance Mechanism that the Software developer can adopt.

1.2 Software Fault Tolerance Techniques

Software fault Tolerance is an ability of software to perform its particular tasks in the presence of hardware or software faults in a system. Following are the different mechanisms of software fault tolerance are as given below:

1. **Recovery blocks:** The recovery block is developed by Randell, in which system is broken down into a Fault Recoverable Blocks. It is used to operates and confirm the produced results. It can be done by using Rollback mechanism (In rollback mechanism, once the changes caused by an aborted condition that have been undone) for each modules in the system.
2. **N-version software:** Firstly, the main objective is to increase the range of the system to avoid the commonly formed failures. By using N-version software, for each different versions that are implemented in a particular range which are possible to be held is an optimistic method. The N-version Method [14] represents the possible faults that are being generated but ignored within the system due to successfully masked. The distinction between Recovery block method and The N-version method are not much abundant, but it is important. It is important because to detect and correct the faults before it becomes errors.
3. **Self-checking software:** Self-checking Software is already deployed in safety-critical system. It is used for additional checking of the system which further includes check-pointing and rollback mechanism also that must be added in fault-tolerant mechanism as a precautions base for a system. It is a most effective method for fault-tolerance system.

Software fault tolerance [11] is not completely sufficient for recover software from hardware or software faults. The main root of occurring faults needs to be identifying at each levels of program code in order to detect faults root and recover it. Most of the software faults can be handled by software fault tolerance method but many of them are not because it can be done by human error or mistake. Current software fault tolerance methods are based on traditional hardware fault tolerance. In the traditional hardware fault tolerance method that was

designed to conquer manufacturing faults primarily and environmental and other faults secondarily. N-version programming is closely parallels to N-way redundancy in the hardware fault tolerance paradigm Hardware faults are not easy to detect or recover; it can only recover by replacement or redesigning the particular component of the system.

III. FAULT INJECTION

Fault injection is the process of introducing faults in a system in order to access its behavior and to measure the efficiency of fault tolerance mechanism of the system. Fault injection is a technique for injecting fault in a system in order to detect other faults in a system and recover them and made available the system to perform its particular tasks properly. It consists of introducing a fault in a system and made few changes in a program code of a system.

According to Arlat [2],"The validation technique of the dependability of fault tolerant systems which consists in the accomplishment of controlled experiments where the observation of the system's behaviour in presence of faults is induced explicitly by the writing introduction (injection) of faults in the system.

In software testing, Fault injection is a technique for improving the test coverage by introducing the faults in a code in order to test the code paths of software [6].

There are two main issues regarding fault injection is given below:

i. Recreation is opposed to an execution is to be done in this technique. Fault injection technique works slowly but made changes easily in the system. Later, the system is deployed itself.

ii. Fault injection Techniques can be classified into two sets that are Invasive and Non-Invasive techniques .The main difficulty with complex systems is to eliminate the traces of testing mechanism from the performance of the system. Invasive techniques can be described as the techniques which departs its footprints whenever being tested. Non-Invasive Techniques are those which are being masked their presence through which it does not effect on the system except on injected faults.

The Process of Fault injection techniques [3] can be done by using the faulty version system in order to introduce faults in a specific location. This process can be initiated by the source files and then implemented over any Pre-processor through which files are to be analysed and build an abstract tree representation of the code. Moreover, that abstract tree representation of code helps in identifying the location where any fault type can be injected through which the faulty code files can be generated and the system becomes faulty or faulty version of the target system.

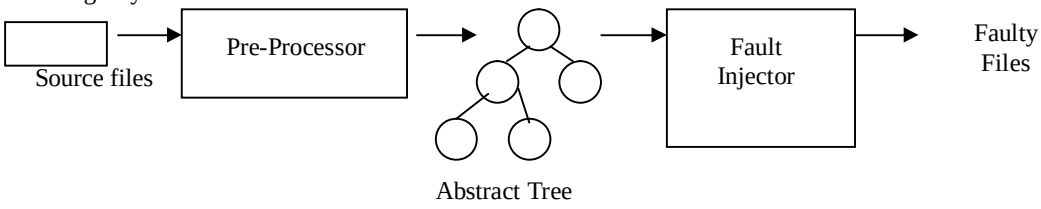

Figure 1: Process of Injecting Faults in a Target system

1.3 Objective Of Fault Injection

Injecting faults in a system to assess the impact of faults in the whole system in order to measure the efficiency and tolerating behaviour of the system. The main objective of fault injection is to inject faults in the system in order to determine that in the presence of these injected faults how system behaves and what changes can occurs in the system after injecting faults. Moreover, to check the system that it reaches to its goal or not (i.e. the system meets its specification or not). Tester should know the system design phase from depth so that it made some test cases in order to check the system behaviour.

Mainly, Fault injection provides the evaluation of system behaviour through the fault detection (Fault detection is a process of distinguish the faults that may occur in the system and generate hindrance in the path of system specifications.) and Faults Removal Process (Fault removal can be defined by correcting the system faults that occurs in the system during testing.) and fault forecasting (Fault forecasting is the process used to evaluate the system latency and coverage distribution through which the faults can be handled, fault forecasting information can be given by the tester.) with the help of injecting faults. Furthermore, Fault forecasting can be done by the analysis of pasts statistical records of testing a Hardware or Software.

Following are the some benefits of fault injection Techniques through which fault injection become useful [4][2], are as given below:

1. Understanding the behaviour of real fault and injecting fault in the system.
2. Reporting the approximate system failure point of fault tolerance mechanism.
3. Recognise the fragile part of the system.
4. Measure the capacity of the system to produce the desired effect.

5. Evaluate the system behaviour and efficiency of the system in the presence of faults.
6. Observe the workloads of the system
7. Estimate the coverage location of real faults in the system.

Through all these benefits, the fault injection technique for introducing faults in the system becomes system more effective and must be significantly able to meet all the system specification.

3.2 Fault Injection Techniques

Fault can be injected in hardware or software in order to measure system capability. Fault Injection can be implemented [2] as follows:

1. Software implemented based fault injection:

The main objective of software Implemented based fault injection (SIFI) is to inject fault at software level of the system in order to measure the efficiency and behaviour of the software system. In this, fault needs to introduce in the specific location in the software that are accessible easily. Some of the specific software faults can be implemented for injecting the fault in the software at a particular location. Software fault injection is being quite attractive these days because it does not require any extensive Hardware. Moreover, it can be also used for target applications and operating system that are difficult to implement by Hardware Fault Injection. Software Implemented based fault injection (SIFI) can be done either Compile time fault injection or Run- time fault injection. Some of the Software implemented based fault injection Tools are GOOFI, FERRARI, FTAPE, FIAT, DOCTOR, XCEPTION, NFTAPE etc.

2. Hardware implemented based fault injection:

Hardware implemented based fault injection can be used to inject fault in the system at physical level. It considered the hardware with few parameters such as Electromagnetic interference, heavy ion radiation, power fluctuation, momentary tree contact, lighting strike etc. Hardware fault injection technique can access locations that is hard to be accessed by other means. It injects those faults which have low act of causing disorder. Software fault injection is used for hardware components also but hardware fault injection is not needed for testing a software components. Some of the Hardware implemented based fault injection Tools are RIFLE, FOCUS, FIST, MARS etc.

3. Simulation implemented based fault injection:

Simulation implemented based fault injection can be used to inject faults at high-levels models (for example VHDL model). It is used to evaluate the system dependability earlier when the system is available. Simulation implemented based fault injection accomplished an act of giving a false appearance on representing something. This type of fault injection provides maximum flexibility. Some of the Simulation implemented based fault injection Tools are VERIFY, HEARTLESS, FTI, GSTF ETC.

4. Emulation implemented based fault injection:

Emulation implemented based fault injection solve the problem of simulation based fault injection technique by reducing time by the use of Field Programmable Gate Arrays (FGPAs). FGPAs is used for Speeding-up the fault simulation and reduce fault injection experimental time. In this, it requires high bandwidth interference between host computer and emulation board.

5. Hybrid implemented based fault injection:

Hybrid implemented based fault injection consists of the combination of any two or more of the other fault injection techniques for more analysis. This type of approach is a best for measuring extremely short latencies. One of the Hybrid implemented based fault injection tool is LIVE.

3.3 Applications Of Fault Injection

In recent few years, Fault injection techniques are used by most of the target applications and by Faulty versions system. During the testing of operating system, many forms are taken by fault injection. Fault injection can be done in both High level software and low level software types. It is common for using Instrumentation for a managed code. Fault injection is basically designed for minimising the possible numbers of false positives from the system. Fault injection can reproduce multiple and distributed errors during the code easily. It is simply a Safety verification process and security assessment for the system.

IV. FAULT SEEDING

Fault seeding is the process developed in 1970s by Gerald M. Weinberg [20] for modifying the program code that is being tested to measure the coverage of test. Basically, fault seeding is used to explore the performance of testing process of a system. Fault seeding is used to introduce one or more faults in a program code without taking the permission of tester. This process is only used to enhance the effectiveness of test cases. Fault seeding can be capable for making software more reliable. Fault seeding is also known as Bebugging or error seeding.

According to Gerald M. Weinberg," Overconfidence by the programmer could be attacked by a system that introduced random errors into the program under test. The Location and Nature of these errors would be recorded inside the system but concealed from the programmer".

According to Pfleeger [16] , the basic theory related to fault seeding that is established by Harlan Mills in 1970s in an unpublished paper, " Assume the seeded faults (S) are of same rigorousness and supposed to be equal to be exposed than an equivalent latent fault (N) ", that relation is given below :

$$\frac{s}{S} = \frac{n}{N} \rightarrow N = \frac{nS}{s}$$

where, S is the number of seeded faults in the program code, N is the number of original faults in the program code(which we want to calculate), s is the number of seeded faults which are found, n is the number of original faults which are found. This relation is said to be a relation between detected seeded faults (s) and number to real errors (N) to be found.

With the help of fault seeding [12], it is possible to predict the number of remaining faults in a software system that are still require testing process. Bebugging is also a type of Fault injection. Fault seeding is the part of Development testing phase. This method of introducing fault in a system is for enhancing the effectiveness of a system and to make a system more reliable to user.

V. COMPARISON BETWEEN FAULT INJECTION AND FAULT SEEDING

The Comparison of Fault injection and fault seeding is shown in Table 2. Both are correlated terms with each other. The comparison is based on some of the following parameters that are given below:

1. System Survivability: System survivability is the process in which system continue its works during the natural disturbance and man-made disturbance.
2. System Reliability: It is a ability of the system or a component in which system can works within conditions for some time period.
3. Fault Tolerance: describes the property in which system continue its execution in the event of failure of some of its components.
4. Controllability (Space and time): It is based on two factors time and space. It specifies the system control process in which it has a ability to move the system around its entire configuration.
5. Physical Reachability: It basically describes the system can successfully meets its goal specification in the presence of faults.

Table 2: Comparison between Fault Injection and Fault Seeding

PARAMETERS	**FAULT INJECTION**	**FAULT SEEDING**
1. Definition	Fault injection is the process of introducing one or more faults to evaluate the fault tolerance mechanism in order to measure the efficiency and behaviour of system in the presence of faults.	Fault seeding is the process of adding one or more faults in the system in order to estimate or detect or remove the number of remaining fault in a program code.
2. Its purpose	To assure that the injected faults are representative of actual faults.	To assess other remaining faults in the system.
3. System Survivability	High Capacity	Low Capacity
4. System Reliability	More Reliable for complex system	More Reliable for web services.
5. Fault Tolerance	Makes system highly fault tolerant.	Can be able to tolerate real faults.
6. Controllability (for time and space)	Medium, low	Medium, high
7. Physical Reachability	High	Medium

VI. CONCLUSION AND FUTURE WORK

Faults make the system performance unreliable and occur delay in order to meet their specification. Nowadays, as complex systems are to be designed, the possibility of finding errors in the system decreases and the inbuilt Fault tolerance mechanism will not sufficient to discover numerous faults in the system. So, fault injection technique can be able to predict the occurrence of faults in a system and remove it from the system.

In this paper we have described various different techniques of fault injection for injecting faults in a prototype model for testing inspection. These techniques undergoes into five categories that are: Hardware implemented based fault injection, software implemented based fault injection, simulation implemented based fault injection, emulation implemented based fault

Injection and hybrid fault injection. Most of the studies revealed that a fault injection technique increases the test coverage of a system.

This paper also described the term, Fault seeding technique which is also used for injecting faults the system but for slightly different purpose from fault injection technique. Fault seeding can be able to estimate the location and possibility of remaining faults in the system. The comparison between Fault injection and fault seeding shows how both terms are different from each other.

Future work may be to do an experimental framework regarding comparative study to compare fault injection and fault seeding.

REFERENCES

1. Joao A. Duraes, Henrique S. Maderia, Emulation of software faults: A field data study and a practical approach, IEEE Transaction on Software Engineering, Vol. 32, NO. 11, November2006.
2. Haissam Ziade ; Rafic Ayoubi2 ; Raoul Velazco, A Survey on Fault Injection Techniques, The International Arab Journal of Information Technology, Vol. 1, No. 2, July 2004.
3. Roberto Natella; Joao A. Duraes ; Henrique S. Madeira, On Fault Represenativeness of Software Fault Injection, IEEE Transactions On Software Engineering, Vol. 39, No. 1, January 2013.
4. K. Umadevi ; S. Brintha Rajakumari, A Review on Software fault injection Methods and Tools, International Journal of Innovative Research in Computer and Communication Engineering, Vol. 3, Issue 3, March 2015.
5. Robert Slater, Fault Injection , Carnegie Mellon University 18-849b Dependable Embedded Systems, Spring 1998.
6. Mei-Chen Hsueh; Timothy K. Tsai; Ravishankar K. Iyer, Fault Injection Techniques and Tools, University of Illinois at Urbana-Champaign, April 1997.
7. J. Voas, A Tutorial on Software Fault Injection, Reliable Software Technologies, Sterling VA USA, 2000.
8. Michael R. Lyu, ' Software Reliability Engineering: A Roadmap', Future of Software Engineering (FOSE '07), IEEE Computer Society, Washington, DC, USA, 153-170, 2007.
9. Olah, J.; Majzik, I., 'A Model Based Framework for Specifying and Executing Fault Injection Experiments', Dependability of Computer Systems, 2009. DepCos-RELCOMEX '09. Fourth International Conference on , vol., no., pp.107,114, June 30 2009-July 2 2009.
10. Cotroneo, D.; Lanzaro, A.; Natella, R.; Barbosa, R., 'Experimental Analysis of Binary-Level Software Fault Injection in Complex Software', Dependable Computing Conference (EDCC), 2012 Ninth European , vol., no., pp.162,172, 8-11 May 2012.
11. B. Randell, System Structure For Software Fault Tolerance, IEEE Transactions On Software Engineering, Vol. Se-1, No. 2, June 1975.
12. A.J. Offuts, J.H. Hayes, A semantic model of program faults, SIGSOFT on Software Engineering notes, 1996.
13. W.T. Tsai, D.Z. hang, Y.Chen, H Huang, R. Paul, N. Liao, A Software Reliability For Web Services, Department Of Computer Science And Engineering, Arizona State University, IASTED Conference, U.S.A, 2004.
14. J. Arlat, Y. Crouzet, J.C. Laprie, Fault Injection for dependability validation of fault tolerant computing system on Fault Tolerant Computing, IEEE, 1989.
15. www.softwaretestingtimes.com/2010/04/fault-error-failure.html?m=1.
16. Fabio Grigorjev, Natacha Lascano, Julio L. Staude, A Fault Seeding Experience, Motorola Global Software Group (GSG) Argentina, Avenida Hipolito´ Yrigoyen 146, 9no piso. Cordoba,´ Argentina, 2003