A View on the Past and Future of Fault Injection

Nuno Silva, Ricardo Barbosa
Critical Software SA
Coimbra, Portugal
{nsilva, rbarbosa}@criticalsoftware.com

João Carlos Cunha
Polytechnic Institute of Coimbra/CISUC
Coimbra, Portugal
jcunha@isec.pt

Marco Vieira
CISUC-University of Coimbra
Coimbra, Portugal
mvieira@dei.uc.pt

Abstract**—**Fault injection is a well-known technology that enables assessing dependability attributes of computer systems. Many works on fault injection have been developed in the past, and fault injection has been used in different application domains. This fast abstract briefly revises previous applications of fault injection, especially for embedded systems, and puts forward ideas on its future use, both in terms of application areas and business markets.

Keywords-Fault Injection, Dependability, Fault Models

I. Introduction

In the past decades, research on fault injection (FI) has specially targeted the emulation of hardware faults, where a large number of works has shown that it is possible to emulate these faults in a quite realist way. More recently the interest on the injection of software faults has increased, giving raise to several works. In terms of application areas and business markets, fault injection has been mainly used in the context of validation of safety critical embedded systems. The aerospace market is an unavoidable example, where fault injection for embedded systems has been largely applied. The problem is that injection tools are quite dependent on the computer technology being used, thus they have to evolve according to the evolution of the application domains, systems complexity, criticality and new technology trends.

This paper briefly discusses the past of fault injection, namely in what concerns basic concepts, typical fault models, and well-known tools. Based on this analysis we then put forward ideas on new application areas, fault models, needs and markets. With this we want to contribute towards starting the discussion on what should be the future of fault injection research and technology development.

II. Fault Injection Basics

Critical systems are designed to include fault and error handling mechanisms, able to tolerate development, physical or interaction faults [1]. A classical application of fault injection is to study the effectiveness of such fault tolerant mechanisms during system development. Fault injection tools provide means for measuring fault coverage, error detection latency, or the impact of fault tolerance in the system.

Other successful application of fault injection is on the robustness testing of embedded systems. By deliberately corrupting parameters provided to operating system calls, the systems under test are evaluated by their resilience in terms of avoiding crashes. In distributed environments, the injection of faults in messages has been useful for designers, system integrators and users to test protocol implementations or even system security.

A. Typical Fault Models

A fault model describes the scope of the faults considered for the injection experiments. These models are a representation of real faults, and are usually limited by the capacity of the tool to reproduce them, or to emulate their closest effects.

When considering hardware faults, the most common models consider the corruption of bits, in the form of *bit-flip* or *stuck-at*, representing the effects of radiation or power disturbances at memory or connection elements. Other models may consider *bridging*, emulating the effects of short circuits, or *open*, representing broken lines. These models are complemented by defining the location of the faults, persistence, activation time, dimension, and duration. On the other hand, software fault models describe real mistakes by software developers. These models may describe common defects or the manifestation of such defects at the program state.

B. Fault Injection Tools

Several fault injection tools have been developed in the past, for both hardware faults and software faults. The first include hardware-implemented fault injection, software-implemented fault injection, and radiation-based fault injection. The later include the mutation of source code and of machine-code. An overview of tools can be found at [2].

Among the many hardware fault injection tools developed, csXception is the unique commercial fault injector available today (www.xception.org) for embedded systems. It uses the debugging and monitoring capabilities of the modern processors. This tool provides a set of spatial, temporal, and data manipulation fault triggers like FERRARI or FTAPE, but with a minimal intrusion on the target system, besides being able to target also system space.

For the injection of software faults FINE and DEFINE were among the first tools implementing mutations. An advanced technique, called Generic Software Fault Injection Technique (G-SWFIT), for emulation of software faults by mutations at the machine-code level is presented in [3]. However, existing tools are limited to prototypes and no commercial tool has been developed so far (although csXception implements some operators).

Other tools do exist but are more oriented towards specific utilization and not really applicable for the safety critical embedded systems. For example the Holodeck tool uses fault injection to simulate real-world application and system errors for Windows applications and services. Moreover, several of the commercial automated testing tools (e.g. LDRA and VectorCast) are starting to consider and name some of the tests they allow as fault injection tests, providing facilities to exercise boundary values and unit tests, for example.

III. Fault Injection Towards the Future

The fault injection technologies have evolved according to the evolution of the application domains, systems complexity, criticality and new technology trends. Nowadays, different types of fault injection are required to keep up with the technology and domains evolution, especially since the solutions and the certification requirements have changed. On one side, we experience more and more complex, ubiquitous and critical systems, with more powerful architectures (e.g. multicore architectures, FPGAs, virtualization, etc.), on the other side we get more stringent requirements for systems that must go under certification (e.g. some standards already suggest the use of fault injection, such as ISO26262).

These new trends are bringing to the market systems that must be more powerful, necessarily more complex and heavily integrated, and that become naturally critical for safety or the business. Thus, one needs to understand the characteristics of such markets and the implications in terms of the required fault injection technology. However, this needs to be done in a broad manner, with the participation of industry and academia. The goal of this fast abstract is precisely to contribute to sparkling such discussion.

A. Fault Models as a Challenge

Testing systems or components for all possible failures is not feasible, thus a restricted and suitable fault model must be selected. The fault model must be based on deep knowledge of the domain, the systems and the way the system interacts with the environment. Appropriate fault models can be based on known failures, identified hazards and feared events, as well as specific requirements (e.g. non-functional requirements, safety and reliability requirements) and possible physical defects or operational errors.

It is then very important to select and compose the fault models and these must trigger real problems that are recognized as such. Fault models must also be adapted to the available interfaces, and monitoring and control capabilities. The problem is that nowadays, with more critical and more complex systems, as well as with the efficiency of automated testing tools, the fault injection techniques have to overcome new challenges. Having a realistic fault model is one of the biggest challenges, and these fault models need to constantly be updated and adapted to the technology evolutions (both hardware and software). Due to this constant evolution and need to have updated tools one can actually question the advantages of fault injection: *are they effectively and efficiently solving real problems*? Another big question here is *in which areas/domains should research on fault models be focused and what problems is it solving*?

B. Needs and Markets

Most of the industrial domains accept fault injection when it becomes easy and cheap to use, or when it becomes mandatory. Recent standards, such as ISO26262 for the automotive market, strongly suggest the usage of fault injection tests, but there is no real imposition as such in any market. The proof is that there are virtually no public works describing the usage of fault injection as support to the compliance of the standards, even if one can think of FI as an excellent tool to help achieving MC/DC coverage for Do-178B, for example. Some safety or mission critical markets have been traditional markets for fault injection, these include space and aeronautics, since their systems operate in particularly harsh environments and the safety concerns are quite valued. Other markets, where we can include transportation and nuclear power plants, can also benefit from testing their systems with these non-functional techniques, and the standards and the industry itself is also acknowledging this today – mostly because some recent failures cannot occur again (e.g. [4][5]). There is very limited information about the usage of fault injection in these areas, although recently, due to the recommendation of the ISO26262 standard, fault injection has been more broadly applied to the automotive industry. On the academic side, although many fault injection tools exist, none is really a ready to use tool, thus a common framework would be a major breakthrough.

The real needs are sometimes associated with specific requirements of the systems/architecture or with the environment where the system interacts, as well as the maturity of the technologies involved. Fault injection is commonly used to: complement regular/functional testing activities; identify dependability bottlenecks and problematic areas; analyze the system behavior in the presence of faults or abnormal situations; prove the coverage of error detection, isolation and recovery mechanisms; test the fault tolerance mechanisms; study the availability and performance losses. The key question is to *understand new trend***s** *in terms of fault injection applicability* **by learning from the field problems that can be solved by fault injection**.

IV. Conclusion

The goal of this paper is to foster the discussion on the direction of fault injection research, technology development and industrial fit. Although many research groups work and use fault injection, a common view on methodologies and tools is not available. Also, it is not clear what should be the direction to follow and the real problems that can be solved in a very efficient way with these technologies. Markets such as automotive and nuclear, seem quite promising, but it is not clear if fault injection technology would be really used in such scenarios. This is the type of questions we believe should be jointly discussed between academia and industry.

Acknowledgment

This work has been partially supported by the project "CECRIS - CErtification of CRItical Systems", FP7 - Marie Curie (IAPP) number 324334.

References

[1] A. Avizienis et al., "Basic Concepts and Taxonomy of Dependable and Secure Computing," IEEE TDSC, 2004.

[2] R. Barbosa et al., "Fault Injection", Resilience Assessment and Evaluation of Computing Systems, ISBN: 978-3-642-29031-2, 2012.

[3] J. Durães, H. Madeira, "Emulation of Software Faults: A Field Data Study and a Practical Approach", IEEE TSE, 2006.

[4] M. Holt et al., "Nuclear Disaster Summary", Report, Jan. 18, 2012 http://www.fas.org/sgp/crs/nuke/R41694.pdf, visited 12-04-2013.

[5] C. Bubinas, "GM recalls nearly 27,000 vehicles over software problem", April 10, 2013, http://www.klocwork.com/blog/coding-standards/gm-recalls-nearly-27000-vehicles-over-software-problem/, visited 12-04-2013.