Jaca Tool Improvements for Speeding Up Fault Injection Campaigns

Naaliel Mendes[1], Regina Moraes[2], Eliane Martins[3], Henrique Madeira[1]

[1]CISUC, University of Coimbra
3030-290 – Coimbra – Portugal

[2]CESET, University of Campinas (UNICAMP)
Caixa Postal 456 – 13.484-370 – Limeira, SP – Brazil

[3]IC, University of Campinas (UNICAMP)
Caixa Postal 6.176 – 13.084-971 – Campinas, SP – Brazil

{naaliel,henrique}@dei.uc.pt, {regina@ceset, eliane@ic}.unicamp.br

Abstract. *This paper presents the improvements that were implemented in the current version of Jaca, a fault injection tools that is used to validate Java applications. Due to these improvements Jaca reports based on .CVS file format allow statistical analysis and require less effort to handle the experiments results. Jaca monitoring abilities provide detailed information to follow exceptional behavior during the execution of fault injection campaign. These improvements also provide more robustness, minimize the user interaction in the fault injection campaign and improve the performance of an injection execution.*

1. Introduction

Dependable systems are currently required in any software development project and have special emphasis on mission critical and complex systems. In this sense, a system is dependable if it survives in the presence of failures. In critical and complex systems, an error means a disaster because human lives or economic resources are in danger.

A real trend in development environment is to construct complex systems by assembling several components. Custom made components for specific tasks can be used, however it is common the use of off-the-shelf (OTS) components for general tasks. Despite the increase in productivity, the use of components still presents some difficulties, especially concerning validation and maintenance. Unspecified dependencies and the complexity of the interaction among components can cause unexpected errors to emerge from component interfaces [Voas, 1997]. Furthermore, the coupling between components to achieve the system's goals makes them highly interdependent. In consequence, a failure in one component can affect the status of other components [Voas, 1997].

Many researches have been pointed to software engineering validation techniques to increase the system dependability. Validation is thus, a necessary step to establish whether a solution achieves the required system's qualities. Moreover, it is important to assess the robustness of the interfaces with respect to component failures as well as problems that enter the system from external sources [Voas, 1998].

An important software engineering validation technique is the use of fault injection technique that helps the evaluation of system dependability. To make this technical approach feasible, the use of an injection tool is often necessary. Jaca [Leme, 2001], the tool presented in this work consists in a software injection tool that inject errors through the interface by corrupting attribute values, methods parameters and return values. Since its first version, Jaca has received relevant improvements and has been used in practical experiences reports in several scientific publications that use fault injection techniques and software robustness evaluation [Moraes, 2005][Moraes2, 2005][Moraes3, 2005] [Jacques-Silva, 2004][Moraes, 2003].

This paper describes the new features available to Jaca last version. The aim is to give a picture of the relevant benefits that Jaca improvements bring to the users as these improvements speed up the time performance of an injection execution and, among other positive points, generate reports that allow statistical analysis.

The remainder of this paper is organized as follows. Section 2 presents some aspects of Fault Injection. Section 3 describes the Jaca Fault Injection Tool. Section 4 emphasizes the improvements done into Jaca. Finally, Section 5 presents our conclusions and future works.

2. Fault Injection

Fault injection is a technique that corresponds to the artificial insertion of faults into a computer system aiming at the acceleration of the occurrence of errors and failures in order to observe the system behavior in the presence of faults in its components or in its environment [Voas, 1998]. Fault Injection techniques have been widely used to evaluate a system's dependability and to validate its error-handling mechanisms. Fault injection enables accelerated system testing under stressful conditions and forces a fault tolerant system to deal with faults, enabling the solutions projected for exceptional situations to be validated and can help the localization of uncover design and implementation faults in the systems [Arlat, 1990].

Fault Injection approaches may vary according to the system life cycle in which they are applied and to the type of faults that are injected. Among the various existent approaches (see [Hsueh, 1997] for an overview), software-implemented fault injection has been widely used. It has become more popular due to its lower costs (it does not require special developed circuits, as does hardware fault injection), better versatility (it is easier to adapt codes to make fault injection in another system than to adapt of circuits) and better control, which together facilitate the observation of the system during tests. Software fault injection consists of altering a system's code or state in order to emulate software faults as well as faults that occur in external components that somehow affect the software [Voas, 1998]. The injection of errors through the interfaces can be useful to evaluate the system robustness and to understand how these errors propagate among the system components. To inject interface faults and to evaluate the system robustness Jaca is an important resource.

3. The Jaca Tool

This work presents the last version of Jaca [Leme, 2001], called JacaC3.0, a software injection tool to inject high level faults in object-oriented systems written in Java programming language. Jaca uses reflective programming to inject interface faults. The

reflection mechanism introduces a new architectural model by the definition of two levels: the meta-level (implements fault injection and monitoring features) and the base level (implements the system's functionalities) [Maes, 1987]. Computational reflection allows the target system's instrumentation to carry out its functions through introspection (useful for the system's monitoring) or by altering the system during runtime (useful for the injection) without changing the system's structure.

Jaca does not need the application source code to perform fault injection. This occurs because Jaca was implemented using the Javassist reflection toolkit [Chiba, 1998], which allows the instrumentation to be introduced at byte code level during load time. Jaca may seem similar to traditional mutation techniques used in mutation testing. However, the fact that Jaca can inject faults and can alter parameters that pass through the interfaces directly at the executable code without requiring the target source code makes the difference between both technique. Jaca can affect the public interface of an application by altering the values of attributes, the parameters of methods and return values.

Jaca was developed based on a fault injection tool architectural pattern. In one simplified version, the base concepts of this pattern are shown in Figure 1. Each rectangle represents packages.

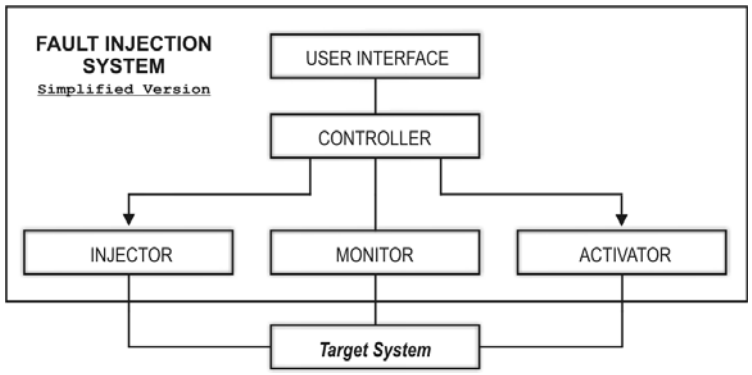

Figure 1. Fault Injector architectural pattern structure.

The main elements of the architectural pattern are: i) Activator - activates the target system, allowing it to be tested in its normal conditions; ii) Injector – injects faults into the target system; iii) Monitor – monitors the target system in order to verify if it is operating as expected; iv) Controller - controls the previous subsystems, so they do their activities coordinately; v) User Interface – receives the specifications from the user for the execution of the experiment and it gives back the results. A complete Jaca structure is described in more detail in [Leme, 2001] and [Martins, 2002]. Relevant improvements were implemented in Jaca new version.

4. Jaca Improvements

Jaca has been developed in an incremental way. Considering several version of the tool, it is evident the relevant features that have been add throughout the time. Jaca 1.0 had been developed in 2001 and the second version, called Jaca 2.0, had been released in 2004. JacaC3.0 is the last version and has relevant new features implemented. The last version of the tool may be downloaded in [Jaca, 2006].

Figure 2 presents the new Jaca graphic user interface with a special emphasis on the new features. The capital letters are defined in a counter-clockwise direction.

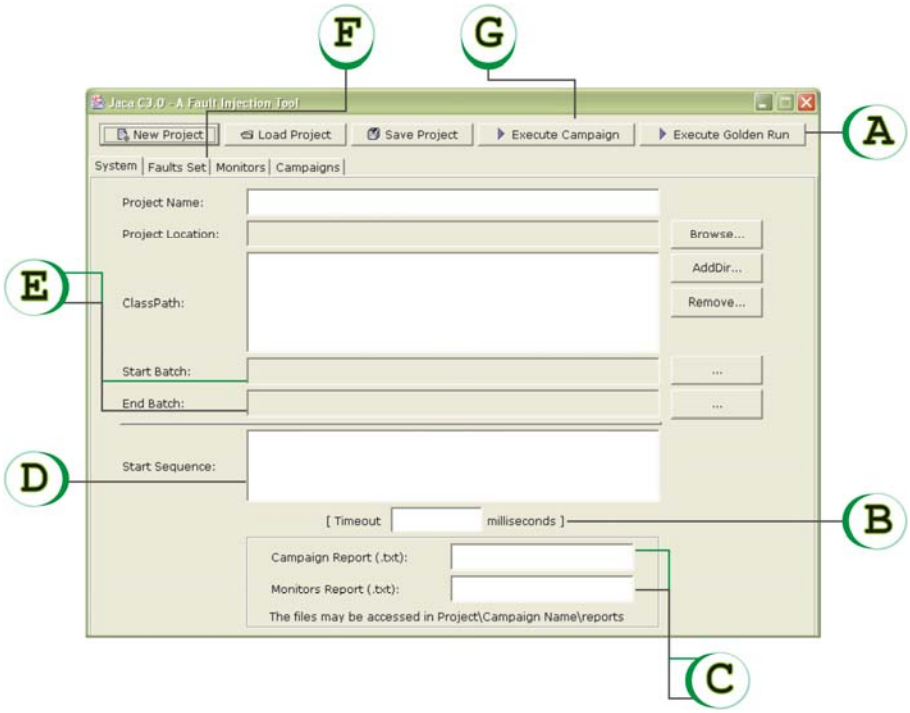

Figure 2. JacaC3.0 Interface in the System Config Tab.

The improvements considered in the JacaC3.0 are:

(A) Golden Run: To allow statistical evaluation Jaca stores the results when no faults are injected. We call this system execution as "Golden Run". The results generated in Golden Run are taken as the system correct behavior and is used to compare the results generated in a run when faults are injected. This comparison allows us to classify the results obtained in accordance with a pre-defined scale. In the previous versions this classification was not automatic and it was not possible to deal with a large number of injections required for statistical results.

(B) Timeout: The user may define the maximum time that an application will be executed in each injection campaign. If, after the timeout time, the system continues running, JacaC3.0 forces the system end.

(C) Report: The reports are divided into three different files. The first one shows the injected errors. The second file monitors what happened at one specific execution. The third file, the unique that does not have the name defined via interface, shows the files exceptions (if it exists). The first and second files are generated to .CVS file format. This means that the results may be imported to different software like ©Microsoft Excel and ©OpenOffice Calc. Indeed, the resulted files are much clearer, given the opportunity to create graphics and summarized results through external software. Furthermore, the first file shows the fault injection results through following *Failure Mode* (the classification of a system failure):

Correct - if, after the fault injection execution, the system has been finished as the system specification and the reported results are corrects, e.g., the results are the same as the Golden Run;

Wrong - if, after the fault injection execution, the system has been finished as the system specification but the reported results are wrong, e.g., the results do not match the ones obtained in the Golden Run;

Crash – if, during the fault injection execution, the system is unexpectedly aborted with reported results unexpected too.

Hang - if, after the fault injection execution, the system does not answer and finished by timeout.

(D) Start Sequence: It is the command line necessary to execute a Java target system. In this new version, when a user aims to define a workload different from the standard workload of the target system, it is necessary to place a special character in this command line.

(E) Batch Files: In the beginning and in the ending of a fault injection, it is possible to run a batch file. This feature can be used to start a target system that should be running before the fault actually is injected.

(F) Injection Run: In the fault injection definitions, the failures values could be: i) defined by user; ii) have automatic increment; iii) received automatically via XML file (jacaCampaign.xml).

(G) Campaign: When Jaca detects an exception, it is not necessary to wait for the target system expiration time (timeout). There is a Jaca internal component that listens continually whether the target system execution process is done. Using this resource, it is possible to improve significantly the performance of a fault injection campaign.

5. Conclusion

This paper presented the last improvements to a software fault injection tool, JacaC3.0. JacaC3.0 is a software injection tool to inject high level faults in object-oriented systems written in Java language and holds important concepts for the fault injection techniques as Golden Run and Failure Mode.

JacaC3.0 controls the expiration time of a target system (timeout) and generate reports in .CVS file format, allowing their manipulation by external software. JacaC3.0 may define a workload different from the standard workload of the target system and may execute batch files before and after of each injection execution. JacaC3.0 has the ability to inject various types of faults with different values, which could be defined automatically or by the user. JacaC3.0 has an internal component to verify when an injection execution is done.

Considering Jaca current version and our experience in software injection validation approach, it is possible to confirm that JacaC3.0 is a very useful tool for robustness testing. If we compare with its previous versions, JacaC3.0 minimize the time that is necessary to complete a fault injection campaign.

In future work we intend to develop a new version of Jaca to inject errors in web-based systems, due to the relevant role that these system have in modern systems.

Acknowledges

The authors thank CAPES (Brazil), GRICES (Portugal) and FCT (Portugal) to partially support this work.

References

Arlat, J., Aguera, M., Amat, L., Crouzet, Y., Fabre, J., Laprie, J., Martins, E. and Powell, D. (1990) "Fault Injection for Dependability Validation–A Methodology and some Applications", IEEE Transactions on Software Engineering, 16 (2), Feb/1990, pp. 166-182.

Chiba, S. (1998) "Javassist – A Reflection-based Programming Wizard for Java", Proceedings of the ACM OOPSLA'98, October 1998.

Hsueh, M.-C., Tsai, T. and Iyer, R. (1997) "Fault Injection Techniques and Tools", IEEE Computer, Volume 30, Número 4, pp. 75-82, April 1997.

Jaca (2006), Jaca – A Fault Injection Tool, http://www.sed.ic.unicamp.br/~jaca/, Accessed on Apr/06.

Jacques-Silva, G., Moraes, R., Weber, T., Martins, E. (2004) "Validando Sistemas Distribuídos Desenvolvidos em Java Utilizando Injeção de Falhas de Comunicação por Software", Proc. of the V Workshop de Testes e Tolerância a Falhas (WTF), Simpósio Brasileiro de Redes de Computadores, Gramado.

Leme, N., Martins, E. and Rubira, C. (2001) "A Software Fault Injection Pattern System", Proc. of the IX Brazilian Symposium on Fault-Tolerant Computing, Florianópolis, SC, Brasil, Março, 2001, pp. 99-113.

Maes, P. (1987) "Concepts and Experiments in Computational Reflection", Proc. OOPSLA'87, pp. 147-155.

Martins, E., Rubira, C. and Leme, N. (2002) "Jaca: A Reflective Fault Injection Tool Based on Patterns", In Proceedings of DSN'2002, Washington, Estados Unidos, July/ 2002.

Martins, Eliane (1996) "Injection Faults in dependable systems", I Regional Symposium of Fault Tolerance Systems, Campinas, (in Portuguese), pages. 181-196.

Moraes, R., Martins, E. (2003) "A Strategy for Validating an ODBMS Component Using a High-Level Software Fault Injection Tool", In: Proc. of the First Latin-American Symposium, LADC 2003, pages 56-68, São Paulo, Brazil.

Moraes, R., Martins, E., Poletti, E., Mendes, N. (2005) "Using Stratified Sampling for Fault Injection", In Proceedings of LADC'2005, Salvador, Brazil, October 2005.

Moraes, R., Martins, E., Mendes, N. (2005) "Fault Injection Approach based on Dependence Analysis", In Proceedings of TQACBS, Edinburgh, Scotland, July 2005.

Moraes, R., Martins, E. (2005) "Fault Injection Approach based on Architectural Dependencies", In Architecting Dependable Systems III – ADSIII, UK (Book chapter), Rogerio de Lemos, Cristina Gacek, A. Romanovsky Editors, Springer Verlag, Berlin Heidelberg, UK.

Voas, J. and McGraw, G. (1998) "Software Fault Injection: Inoculating Programs against Errors", John Wiley & Sons, New York, EUA.

Voas, J., Charron, F., McGraw, G., Miller, K. and Friedman, M. (1997) "Predicting how Badly Good Software can Behave", IEEE Software, pages 73–83, August 1997.