

# Jaca - A Software Fault Injection Tool

Regina Lúcia de Oliveira Moraes  
[regina@ceset.unicamp.br](mailto:regina@ceset.unicamp.br)

Eliane Martins  
[eliane@ic.unicamp.br](mailto:eliane@ic.unicamp.br)

State University of Campinas (UNICAMP)  
Superior Center of Technology Education (CESET)      Institute of Computing (IC)

## Abstract:

This work presents Jaca, a fault injection tool that aims to validate Java applications. Jaca injects high-level software faults by affecting attributes and methods of an object's public interface. Jaca is source-code independent: instrumentation needed for injection and monitoring purposes are introduced at bytecode level during load-time. Jaca's usefulness is shown in a case study used to validate a third-parties component. The case study consists of an OODBMS named Ozone as a component and a performance benchmark, Wisconsin OO7, which uses the OODBMS' stored objects.

## I.1 Introduction

The Fault Injection technique is very useful in that it allows to evaluate the behavior of computing systems in the presence of faults. Among the various methods to perform fault injection, Software Implemented Fault Injection is getting more popular, where faults are injected at the software level by corrupting code or data.

Many tools for that purpose have been developed. Most of these tools consider the injection at low level software (assembly or machine code) and can hardly be adapted to inject high-level software faults.

Jaca is a fault injection tool that offers mechanisms for the injection of high-level faults in object-oriented systems and has the ability to be adapted to any Java application with a minimum of rewriting. It does not need the application's source code, but as in any test it needs minimum information about the application, such as classes, methods and attributes names. Jaca uses reflective programming, to inject faults into applications, which allows the instrumentation to be introduced at byte code level when the system is loading. Computational reflection allows the target system's instrumentation to carry out their functions through introspection (useful for the system's monitoring) or to alter the system during runtime (useful for the injection) without changing the system's structure.

Jaca was developed in Java, and allows for a high portability using Javassist reflection toolkit[1], which is also fully compliant with standard JVM. Jaca monitors the system under test, presenting a logfile that allows us to observe whether or not the system under test behaves in an expected way in the presence of faults. A detailed architecture of the Jaca tool can be found in [2].

## I.2 Using Jaca

Jaca's current version has a graphical interface to help the user to indicate the parameters of the application under test to execute the fault injection. The user can inform the directory where the target application resides. Using Java Reflective API's introspection capabilities, the names of the attributes and methods of each class are obtained. The steps needed for using Jaca are:

1. Select the directory in which the application and Jaca's classes reside (storage in the same directory).
2. Select the logfile's name and it's resident directory.
3. Specify how to invoke the system under test.
4. Select the type of fault to inject (Attribute Fault, Method Return Fault or Parameter Fault).
5. Select the application's class to be injected. Jaca's interface presents a combo box with a list of all the classes' names existing in the directory chose in step 1.
6. Select the elements to be injected, according to the fault type (step 4), attributes, methods' return values or method's parameters of the class selected in step 5.
7. Select the type and value of the mask, which Jaca uses to alter current values of the elements to be injected.
8. Select the operation to be performed with the mask and the current value of the chosen element (step 6). Now only integer and real type values are being considered.
9. Choose the repetition pattern: permanent, transient or intermittent.
10. Select the start time for injection.
11. Select classes to be monitored. The selected classes are recorded in the monitored file.
12. Start execution by pressing the execution button.

A fault specification file (Fault Spec) is created containing the information collected in steps 5 to 10. Jaca uses this file and the monitored file created in step 11, as input. As output, a logfile is written containing monitored information about the target system behavior. Afterwards, the user can see the logfile at Jaca's interface. For the demonstration we plan to show, the use of Jaca and some results obtained in the validation tests done.

## References

- [1] Chiba, S. "Javassist – A Reflection-based Programming Wizard for Java". *Proc. of the ACM OOPSLA'98* Oct/98.
- [2] Leme, N. G. M.; Martins, E; Rubira, C. M. F. "A Software Fault Injection Pattern System" *Proc. of the IX Brazilian Symposium on Fault-Tolerant Computing*. Florianópolis, SC, Brasil, Mar/01, pag 99-113.