# Tutorial

## Title of the tutorial

**Software Fault Injection for the Assessment of Critical Systems**

## Presenters

Domenico Cotroneo and Roberto Natella
Università degli Studi di Napoli Federico II, Naples, Italy

## Contact info

Email: cotroneo@unina.it, roberto.natella@unina.it

## Tutorial type

Half-day tutorial (expected duration: 3 hours)

## Description of the topic and objectives of the tutorial

Unfortunately, it is impossible to assure that software is fault-free, and we must assume that a complex software system has software faults ("bugs") and that it will eventually fail. As software is becoming more and more pervasive in safety-critical systems, it is increasingly important to assure that the system will be safe in such circumstances. Software Fault Injection consists in the deliberate introduction of software faults into a system, in order to assess the impact of faults hidden in the software, and to improve fault tolerance algorithms and mechanisms (e.g., exception and error handlers). This approach has been included as a recommended practice in the recent ISO/DIS 26262 safety standard, and it has therefore gained interest among practitioners. Software Fault Injection is also a valuable approach for researchers that allows investigating dependability issues in complex software systems and to evaluate and compare novel solutions.

The aim of this tutorial is to provide a comprehensive overview of the state-of-the-art on Software Fault Injection, in order to illustrate and critically compare the many techniques and tools that have been proposed in the last decades, each one with different applications and requirements in mind (e.g., stochastic modeling, robustness testing, error propagation analysis, dependability benchmarking). This tutorial will:

- provide a conceptual framework on Software Fault Injection for analyzing the state-of-the-art;
- highlight the potential scenarios in which Software Fault Injection can be adopted in the design and verification of fault-tolerant systems;
- enable both researchers and practitioners to select the best Software Fault Injection technique, among the several existing ones, that fits their specific needs.

## Structure of the tutorial

- **Basic concepts and definitions [15 min]**: this part provides fundamental concepts of fault injection (e.g., the main elements of fault injection experiments and related issues such as workload and faultload representativeness, non-intrusiveness, portability, repeatability, practicability).
- **Characterization of software faults [30 min]**: field failure data studies and taxonomies about software faults are reviewed, in order to support the artificial emulation of software faults by Software Fault Injection.
- **Software Fault Injection techniques and tools [45 min]**: existing techniques and tools are discussed and compared, by distinguishing between:
    - ”data error injection” approaches that emulate the effects of software faults by corrupting the program state, in a similar way to hardware fault injection techniques;
    - ”interface error injection” approaches, which are specifically aimed at testing the robustness of software components with respect to interactions with other components;
    - "code changes" approaches that inject actual software faults by introducing small faulty changes in the program code.
- **Applications of Software Fault Injection [45 min]**: this part reviews a selection of relevant past studies that adopted Software Fault Injection, in order to highlight its applications and usage scenarios (such as the assessment and improvement of fault tolerance mechanisms and dependability benchmarking). Emerging forms and applications of Software Fault Injection are also discussed (e.g., integration with formal methods, applications to security assessment).
- **Hands-on session with a Software Fault Injection tool [45 min]**: a Software Fault Injection experiment is actually performed and discussed. Participants will be provided with a ready-to-run Software Fault Injection environment (e.g., in a virtual machine) that will allow them to replicate the experiment.

## Expected background of the audience

The target audience includes students, academic/industrial researchers, and practitioners in the area of safety-critical software systems. The assumed background is basic knowledge of computer architecture and dependable computing.

## Supporting materials

Supporting materials will include a survey paper about the topics of the tutorial. A preliminary version of material that will be discussed in the tutorial and in the survey is available in the PhD thesis of one of the presenters (http://www.mobilab.unina.it/tesi/thesis_natella.pdf, Chapter 2). The tutorial will include a hands-on session using a mature Software Fault Injection tool available for research and educational purposes.

## About the presenters

**Roberto Natella** received a PhD degree from the Federico II University of Naples in Computer Engineering in 2011. He is currently a post-doc researcher at the Consorzio Interuniversitario Nazionale per l'Informatica (CINI). His research interests are in software dependability assessment and certification, and in particular in software fault injection techniques. He authored several publications in international journals and conferences on these topics, and he is involved in industrial research projects with companies in the Finmeccanica group. More information can be found at the following website: http://wpage.unina.it/roberto.natella/.

**Domenico Cotroneo** received his Ph.D. in 2001 from the Department of Computer Science and System Engineering at the Federico II University of Naples, Italy. He is currently Associate Professor at the same university. His main interests include software fault injection, dependability assessment techniques, and field-based measurements techniques. Domenico Cotroneo has served as Program Committee member in a number of scientific conferences on dependability topics, including DSN, EDCC, ISSRE, SRDS, and LADC, and he is involved in several national/european projects in the context of dependable systems.