

Fault Injection Representativeness

Eliane Martins

Institute of Computing - UNICAMP

Fault Injection

- What?
 - Observe system behavior in the presence of faults

Fault Injection

- What?
- Aim?
 - hw system validation
 - hw design validation
 - sw validation
 - sw testing effectiveness evaluation
 - sw testability
 - sw component failure impact
 - ...

Fault Injection

- What?
- Aim?
- Approaches?
 - simulation
 - hardware-implemented fault injection
 - software-implemented fault injection

Software-Implemented Fault Injection

- a kind of testing



- how to introduce faults?
- what faults to introduce?
- what target?
- how to evaluate results?

Software Implemented Fault Injection

- A kind of testing



- how to introduce faults?

source code
trap/exception
time-out
trace mode
middleware
computational reflection

Software-Implemented Fault Injection

- A kind of testing



- what faults to introduce?

Categories of faults that can compose the faultload:

internal faults

external faults

Software-Implemented Fault Injection

- A kind of testing



– what target?

component:
 module or object,
 subsystem,
 system

Software-Implemented Fault Injection

- A kind of testing



– how to evaluate results?

comparison with a “golden run”
use of an oracle

☞ evaluation of fault injection testing quality?

Selecting the Faultload

- Dependent on:
 - classes of faults: internal, external
 - target system
 - validation objectives
 - workload
 - desired measures
 - implementation language

Internal faults

- What?
 - faults introduced by developers
- How?
 - pre-runtime injections: mutation operators
 - runtime injections: what are the consequences of software faults?

External faults

- What?
 - Faults caused by human interaction, operating system, hardware, other sw system
- How?
 - determination of failure modes
 - known for some classes of system
 - ex.: an operating system call returns NULL when failing in opening a file

Meaning of “representative”?

- Faults representing:
 - all faults that could appear during software lifetime: infinite and unknown
 - all existing internal faults: finite but unknown
 - all external faults that could occur: infinite and partially known
 - all external faults that will occur: finite but unknown

Fault representativeness: why?

- Is representativeness really relevant for software testing?

sw seems to present homogeneous propagation behavior



inject faults at random is still meaningful to predict software behavior

Final comments

- Fault injection \Rightarrow dependability benchmarking:
 - improve tests (or set of tests) used to quantify dependability
- issues:
 - criteria for faultload selection
 - fault injection approach
 - results evaluation
 - fault injection evaluation

Creating representative classes of faults

- Information required
 - Where do software inputs come from?
 - Where are outputs handled?
 - What are the events to be observed?
 - What are the input distributions?
 - What are the test suites used to activate the target software?

Reference

J.M.Voas; G.McGraw. “Software Fault Injection. Inoculating Programs Against Errors”.

John Wiley & Sons, 1998.