Building Dependable COTS Microkernel-based Systems using MAFALDA

Jean-Charles Fabre, Manuel Rodríguez, Jean Arlat, Frédéric Salles and Jean-Michel Sizun

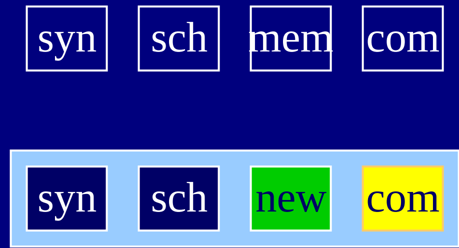
LAAS-CNRS
Toulouse, France

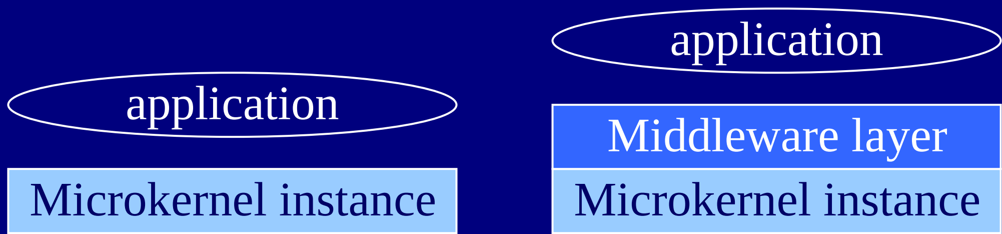
PRDC-2000, UCLA, Los Angeles, CA, USA — 18-20 December 2000

Problem statement

- **Building executive supports for dependable systems, two options:**
  - Development from scratch is complex & expensive
  - Use of commercial components is questionable

- **Main tendency for embedded systems**
  - Use of COTS componentized microkernels
  - Define a specific instance for the application
  - System development : two options

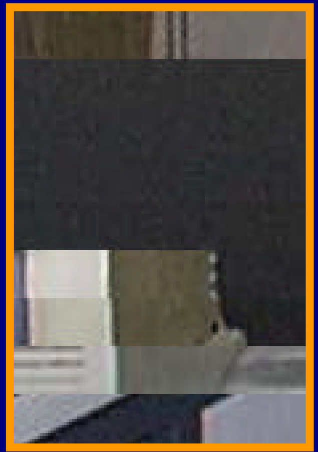




Outline

MAFALDA
Microkernel
Assessment by
Fault injection
AnaLysis and
Design
Aid

- **The objective of MAFALDA**
  - Failure mode analysis
  - Development of fault containment wrappers
- **Description of the tool**
  - Organization
  - Type of measurements
- **MAFALDA in action**
  - Campaign definition
  - Conducting experiments
- **Lessons learnt**



Rack of target machines

Host Machine controlling the experiments

Objectives of MAFALDA

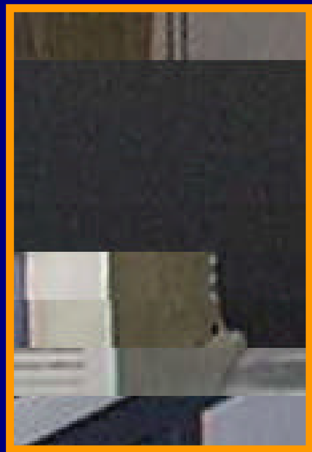
MAFALDA
Microkernel
Assessment by
Fault injection
AnaLysis and
Design
Aid

- **Characterization by SWIFI**

  (S/W Implemented Fault Injection)

  – Identification of failure modes
  – Evaluation of error detection coverage
  – Identification of propagation channels
  – Assessment of interface robustness

- **Wrapping framework**
  – Definition of formal wrappers
  – Definition of a reflective implementation framework
  – Application to both white-box & black-box candidates



Rack of target machines

Host Machine controlling the experiments

- **Evaluation of the wrapped microkernel instance**

Characterization of the failure modes

Evaluation

- interface robustness
- error detection mechanisms

Injection targets

- kernel-call parameters
- microkernel components

Fault model

- corruption of input data
- corruption of microkernel code & data segments

Fault types

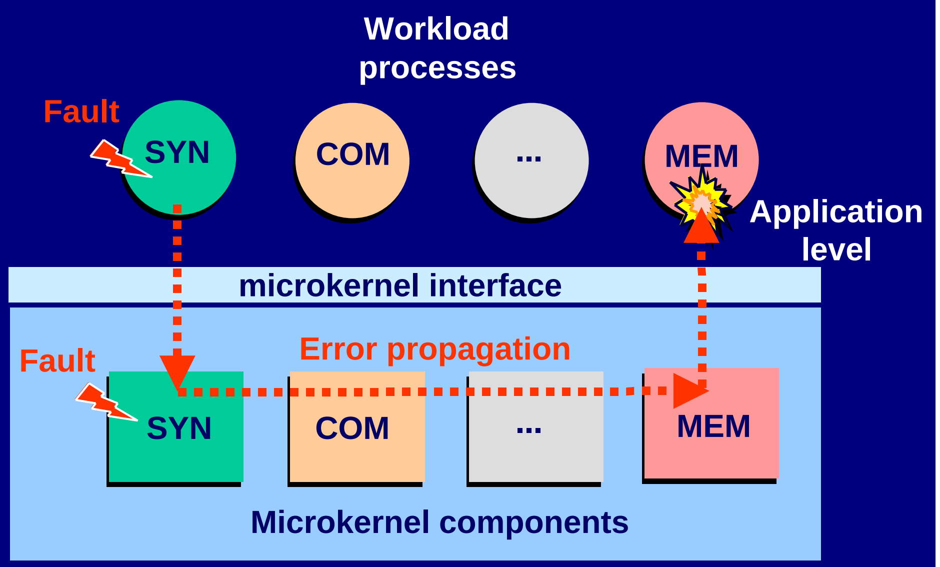
- bit-flip
- random selection

Observation

- behavior of a functional component
- error propagation between components

Results

- statistics
- raw data (*a posteriori* in-deep analysis)

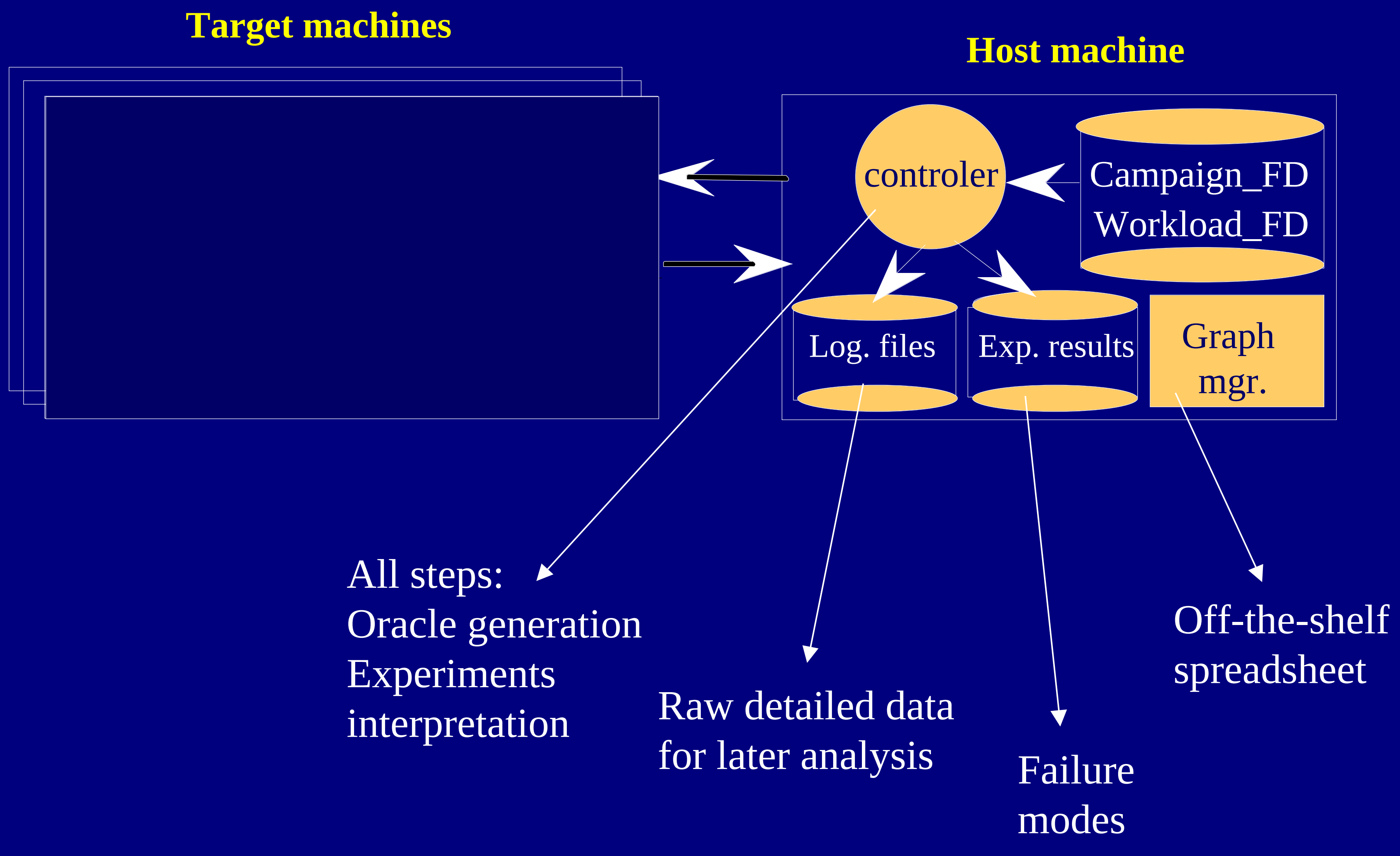


Fault containment wrappers

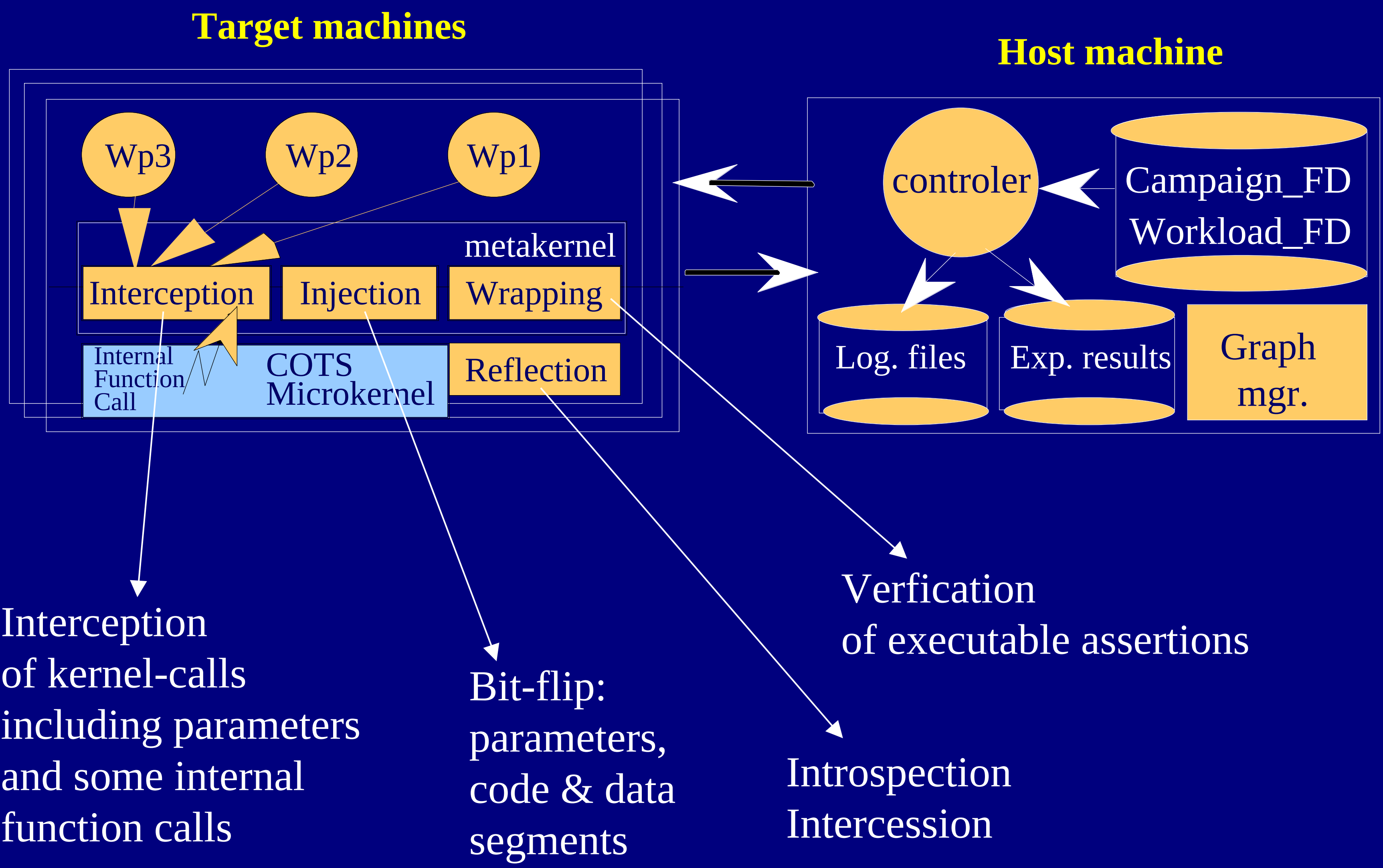
- **Principles and basic techniques**
  - Encapsulation of weak components
  - Modeling microkernel functional classes
  - On-line verification of expected properties

- **Implementation of generic wrappers**
  - Principle
    - Verification of executable assertions
    - Verification of formal expressions (model-checking)
  - Implementation based on the notion of reflective component
    - Interception of system calls and internal events
    - Some internal information is made observable from outside
    - Microkernel + observation/control = reflective microkernel

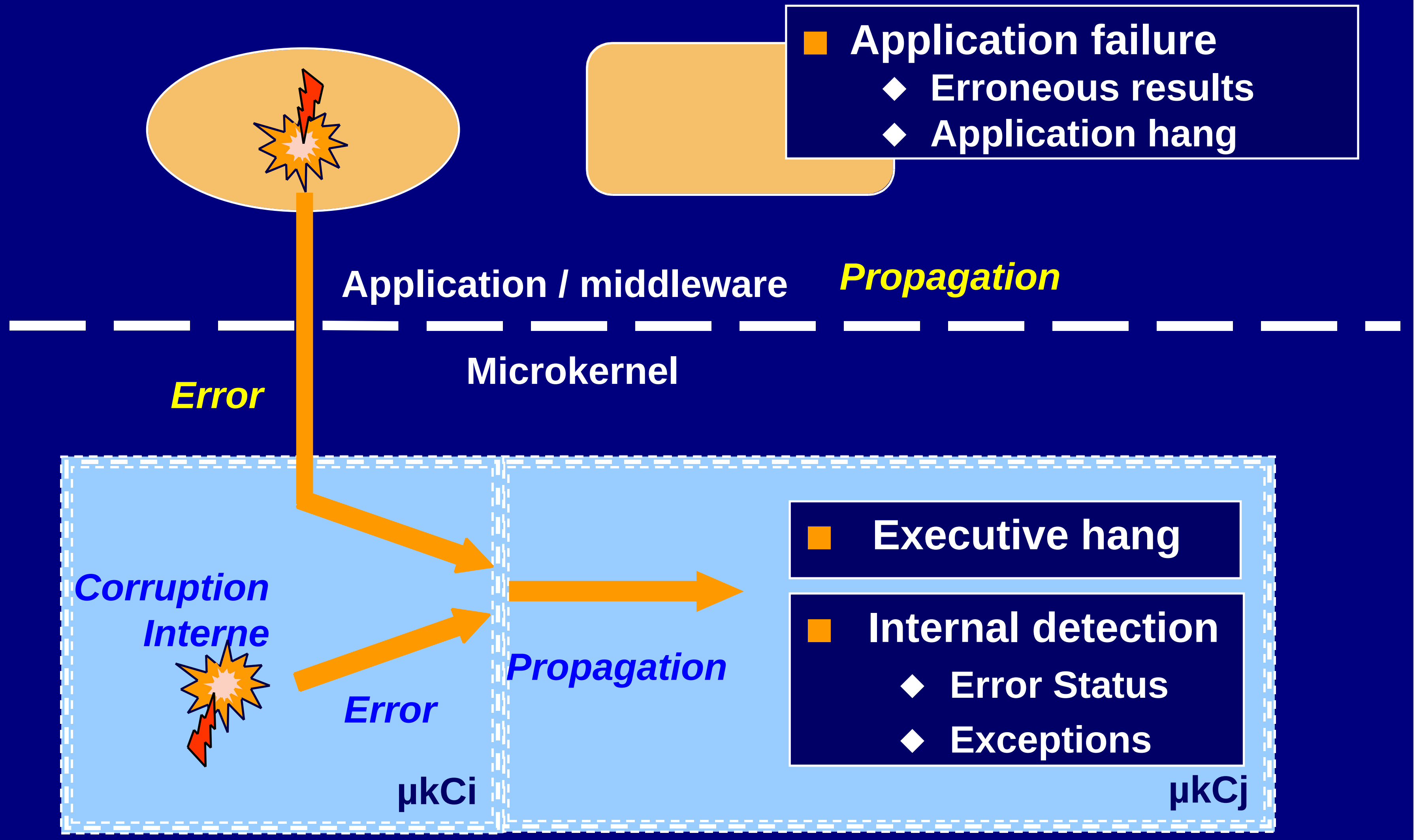
Description of the tool



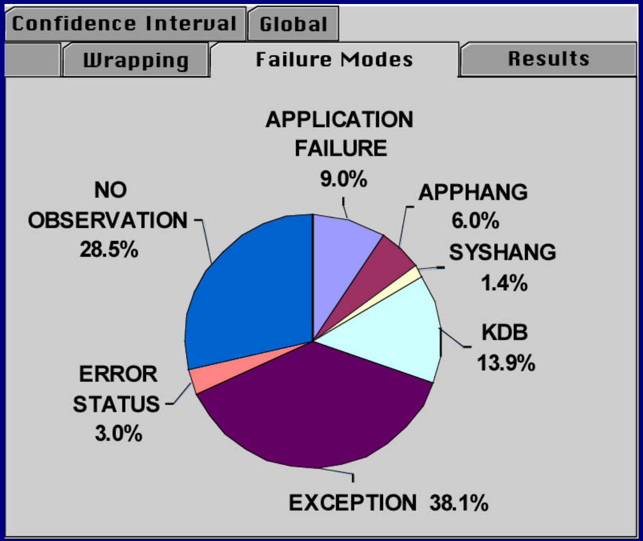
Description of the tool

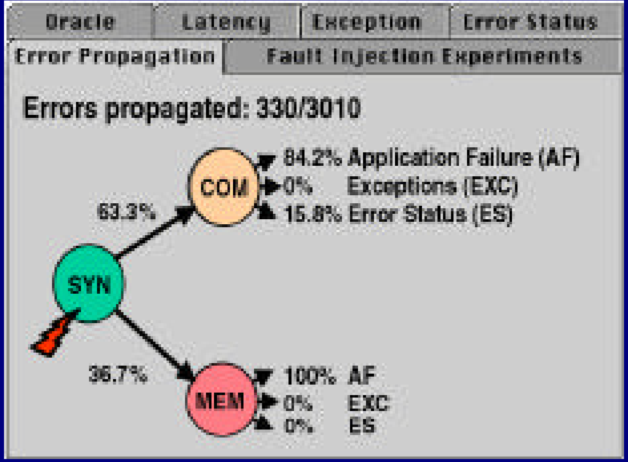


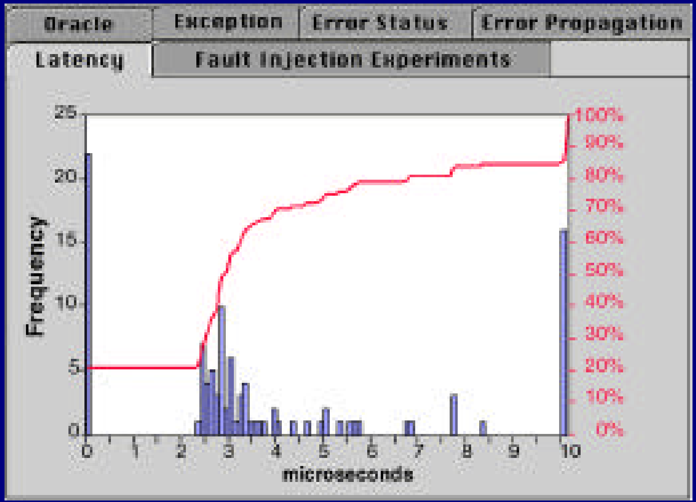
Campaign outputs

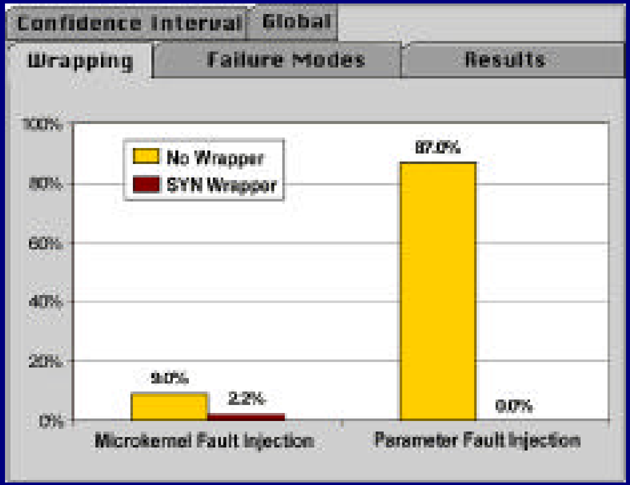


Sample of measures

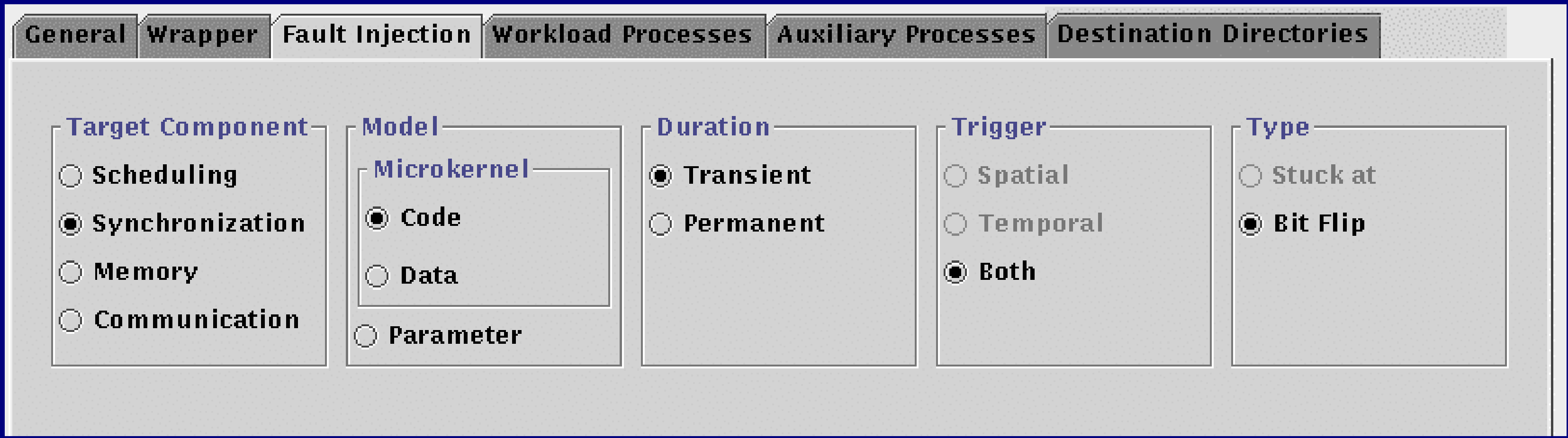




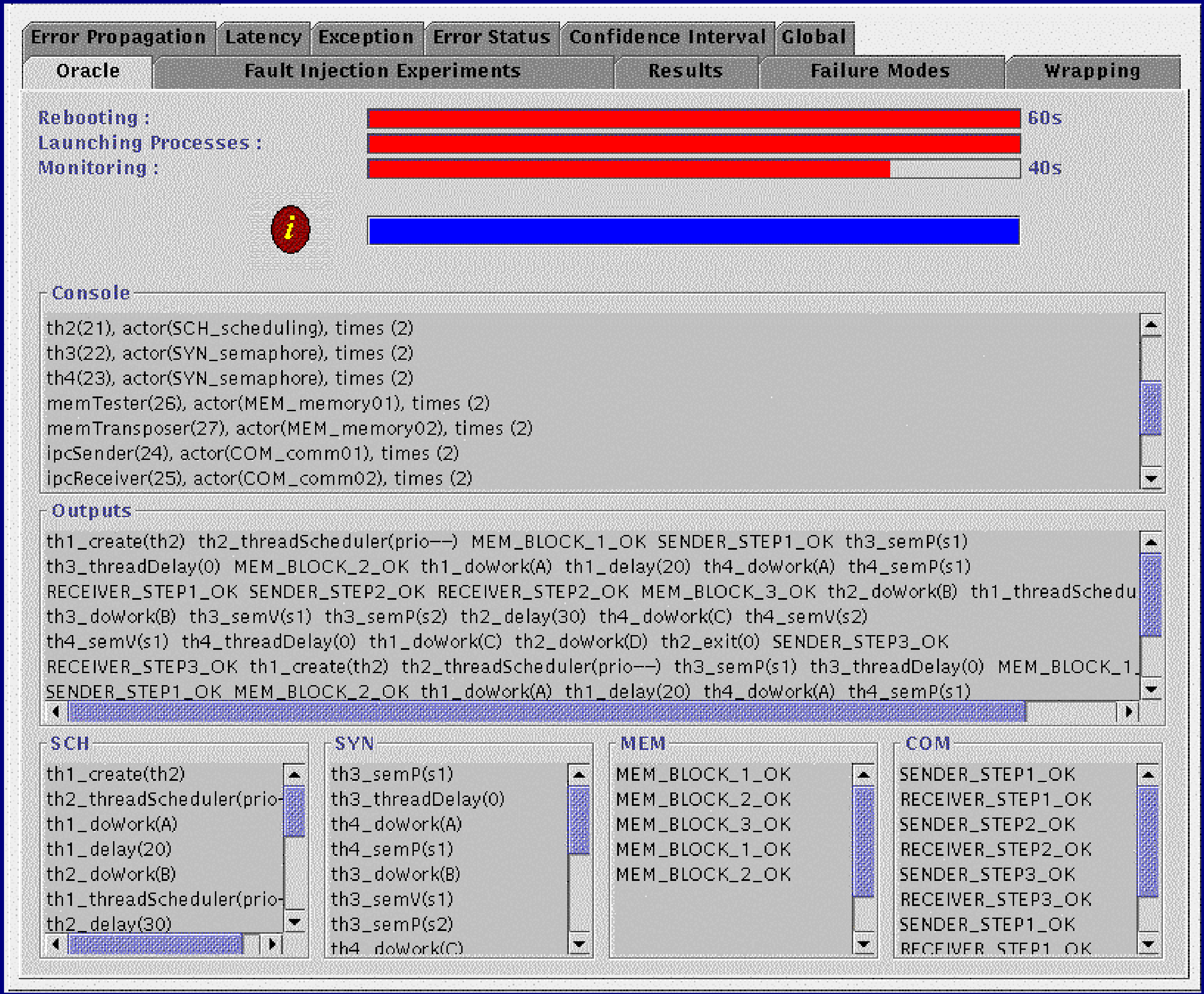




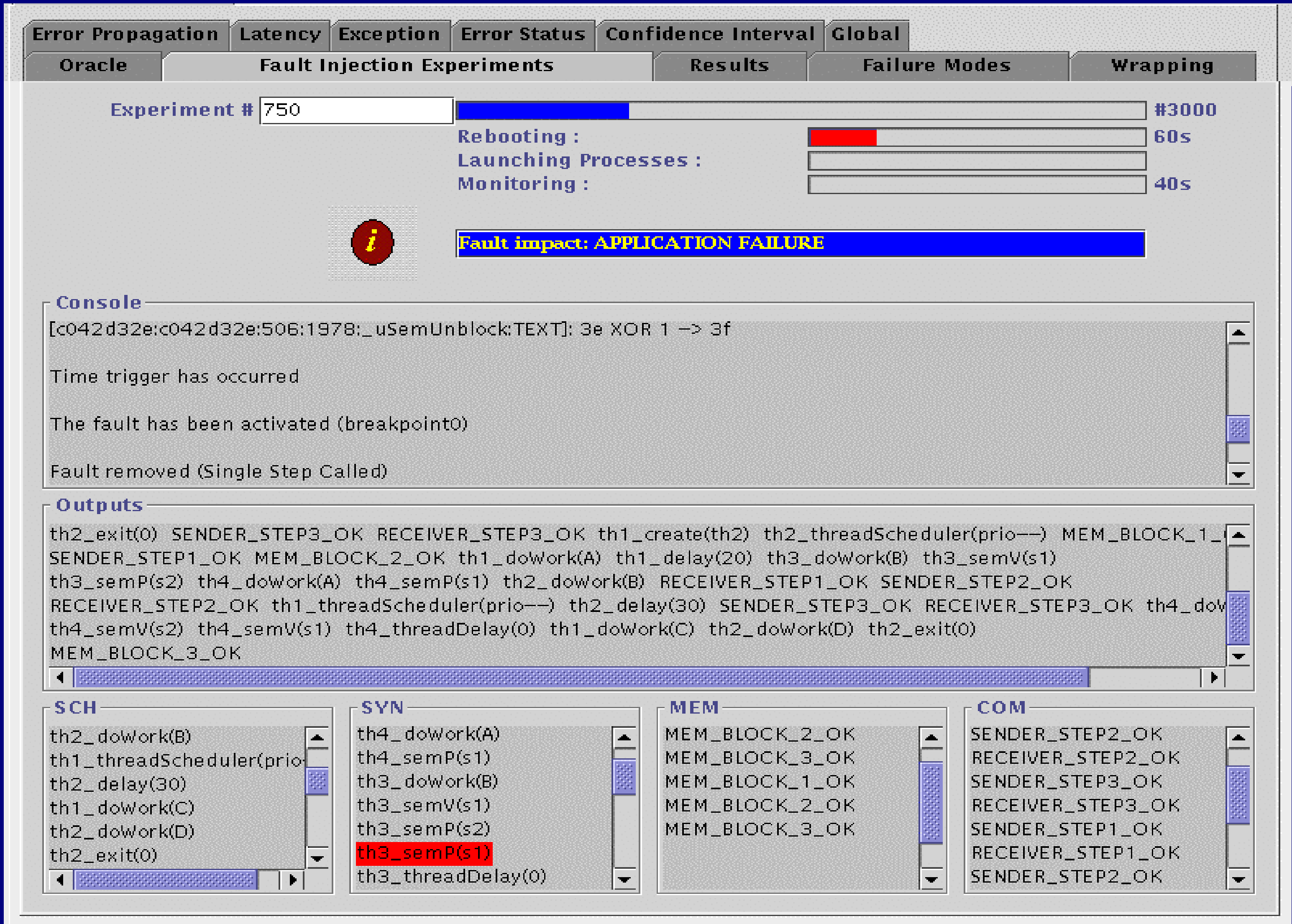
MAFALDA in action



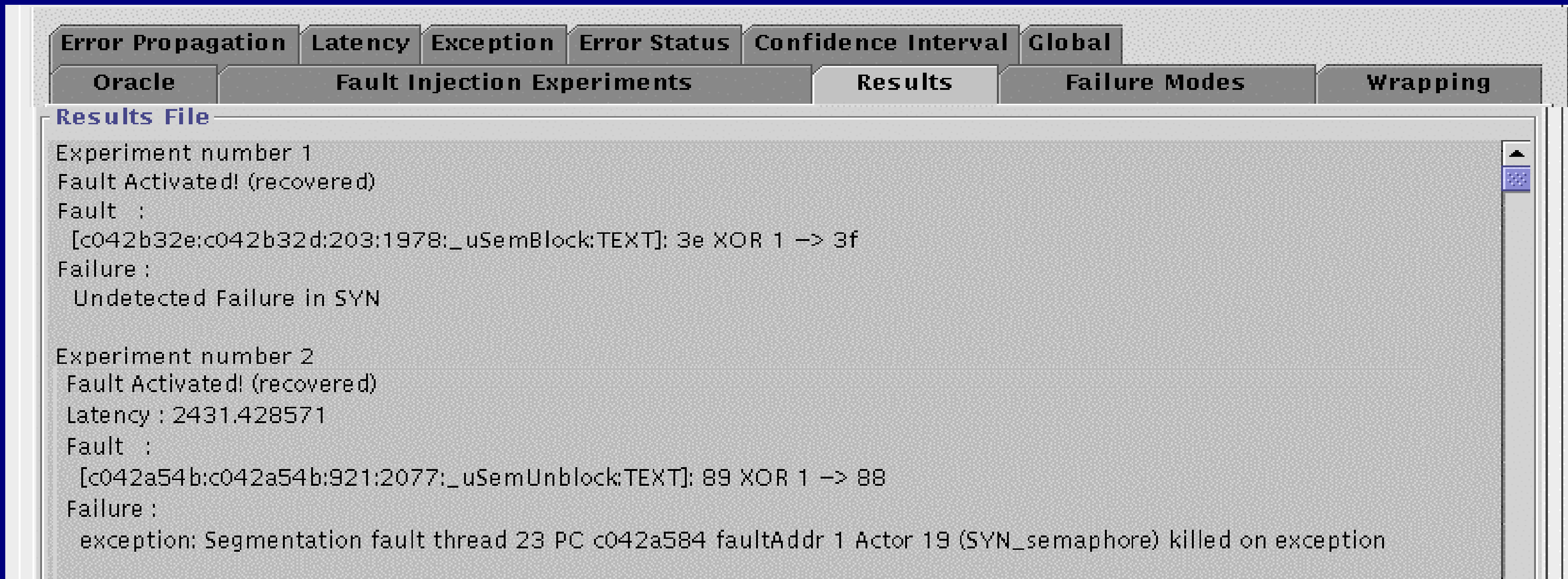
Running experiments: the oracle



Running experiments: fault injection



Running experiments: results



Lessons learnt (1-3)

- **Workload definition and oracle**
  - Generic workload / component Æ design, programming flaws
  - Specific workload / application Æ failure modes (oracle)

- **Fault injection**
  - Selection: random *vs.* predefined location of the bit-flip
  - Kernel injector: debugging features of modern microprocessors
  - Parameter injector: interception of kernel-calls
    (library-based *vs.* trap-based µkernel)

- **Assertions and wrappers**
  - Formalize the expected behavior from the integrator viewpoint
  - Performance: tradeoff between modeling and runtime overhead
  - Temporal logic expressions interpreted on-line by a model checker

Lessons learnt (4-6)

- **Raw data analysis**
  - Analysis of logged data Æ identification of program flaw
  - User-defined semantics of the failure modes
- **Interpretation of results**
  - One campaign targets a microkernel instance & an activation profile
  - Variability of the results:
    - Stand-alone version *vs.* Posix-based version
    - Reactive application *vs.* static application
- **Target system evolution**
  - A slightly new instance Æ new campaign needed
  - Is the new release/version acceptable?
  - Is the new instance compatible with architectural solution?

Lessons learnt (7)

- **Integrator's *vs.* supplier's viewpoint**
  - Integrator
    - Weaknesses revealed
    - Decision: reject or encapsulate
      - » Appropriate wrappers
      - » Tradeoffs measurements
      - » Implementation: reflective framework
  - Supplier
    - identification of bugs not revealed by standard benchmarking activities Æ product improvement
    - Implementation of external error detection mechanisms:
      - » Development of the reflection module
      - » Mechanisms left open to the integrator

Conclusion

- **Experiments**
  - Chorus ClassiX
    - Failure mode analysis and wrapping (SYN & SCHED)
    - Source code Æ implementation of the reflective framework
  - LynxOS
    - Only failure mode analysis on a black-box instance
    - Metainterface delivered to the supplier
- **Current work and perspective**
  - Characterization: extension of MAFALDA to real-time issues
  - Wrapping: formal description in temporal logic + on-line model checking
  - Implementation: reflective framework

Thank you!

Jean-Charles Fabre, Manuel Rodríguez, Jean Arlat, Frédéric Salles and Jean-Michel Sizun

LAAS-CNRS
Toulouse, France