# Fault Isolation in Discrete Event Systems by Observational Abstraction

Dan Lawesson[1], Ulf Nilsson[1], Inger Klein[2]
(1) TCSLAB / (2) Automatic Control, Linköping University

## Assumptions & motivations

We considered an industrial robot control system developed by ABB Robotics. The system is large, concurrent, has an object oriented architecture and is configurable, supporting different types of robots and cell configurations.

- Object orientation ⇒ encapsulation
- Encapsulation ⇒ propagating error messages

The system is safety-critical and alarms that go off are logged and must be analyzed only when the system comes to a standstill.

- Concurrency ⇒ message order cannot be trusted

## Aim

- Operator support
  Single out the error message or critical event that explains the actual cause of the failure
- Diagnosability analysis
  Determining, at desgin time, if enough error messages are produced to be able to isolate all faults.

## Approach

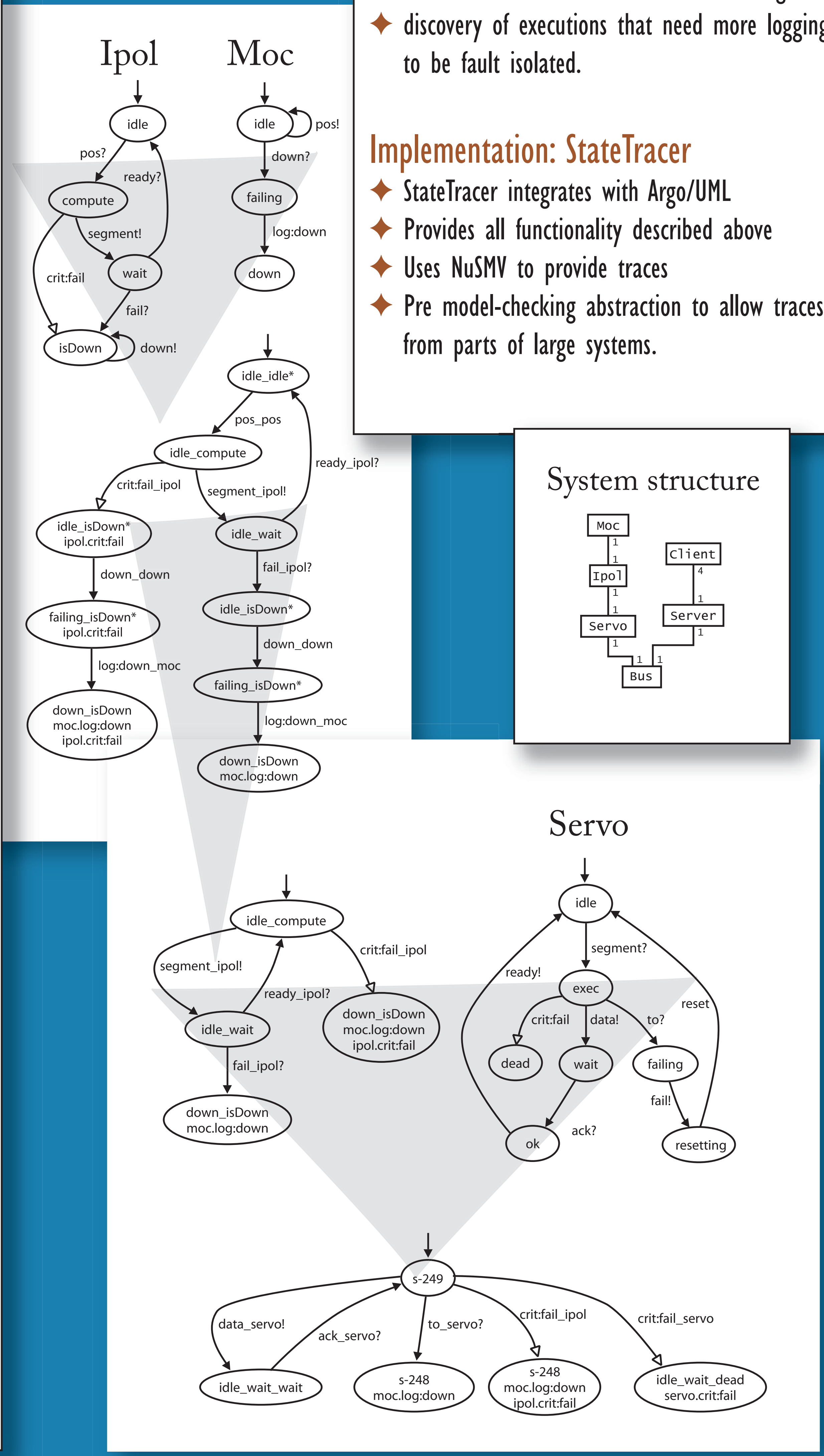
- Model based (UML state machines)
- Searching for critical events in all possible executions that exhibit a given set of logged messages ⇒ state space explosion problems
- Large state space calls for abstraction

## Local abstraction

- Internal component dynamics can be simplified as long as the rest of the system cannot tell the difference.
- Finding less complex but behaviorally equivalent state machines is feasible as long as the computation is local, and does not consider the global state space.

## Composition

- Gradually merge components of the system, thus turning inter-component communication into internal dynamics suitable for abstraction.
- Abstract the merged components and iterate.
- Finally the whole system is abstracted by one state machine.



## Fault isolation table

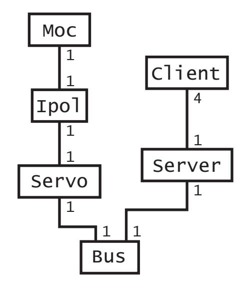
Having the whole system dynamics described in one state machine, the set of all possible error logs and the corresponding failure causes can be computed. This fault isolation table gives

- fault isolation by table look-up,
- diagnosability analysis,
- a least set of non-redundant error messages and
- discovery of executions that need more logging to be fault isolated.

## Implementation: StateTracer

- StateTracer integrates with Argo/UML
- Provides all functionality described above
- Uses NuSMV to provide traces
- Pre model-checking abstraction to allow traces from parts of large systems.

System structure


Moc
1
1
Ipol
1
1
Servo
1
Client
4
1
Server
1
1
1
Bus