Evolution of testing techniques: from active to passive testing

Ana Cavalli

TELECOM SudParis

August 28-2012

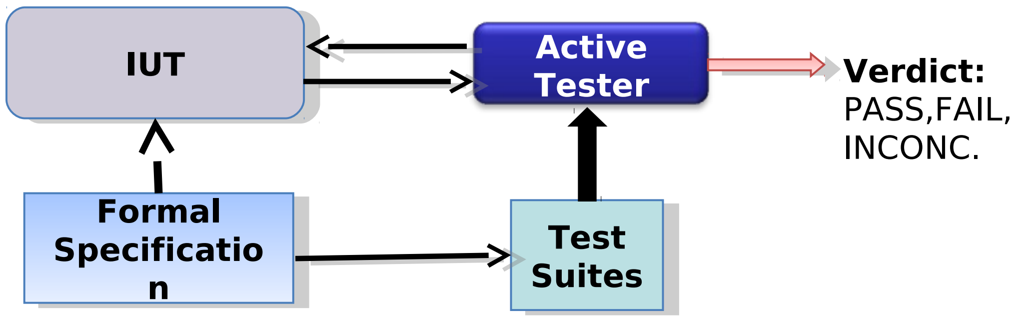
Planning

- Conformance testing
- Active testing techniques
- Fault models
- Control and observation
- Passive testing (Monitoring)
- Tool and case study (DIAMONDS project)

Conformance testing

- Conformance testing:
  - to check that an implementation conforms a specification
- Faults detected :
  - **output** faults, if the implementation transition produce a wrong output
  - **transfer** faults, if the implementation transition go in a wrong state of the machine
  - **mixtes** faults, both output and transfer faults

What is active testing ?



- It is assumed that the tester controls the implementation
- Control means: after sending an input and after receiving an output, the tester knows what is the next input to send
- The tester can guide the implementation towards specific states
- Automatic test generation methods can be defined

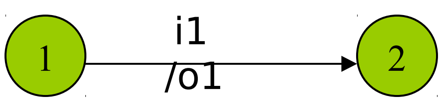
Formal methods for test generation

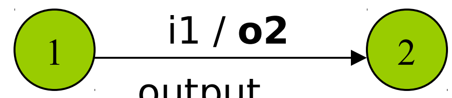
- **Objectives**
  - To optimize tests production by reduction of time and cost
    - An engineer produces three handcrafted tests per day
    - A test suite of a real protocol is composed of an average of 800 tests
  - To improve faults coverage

Fault Models

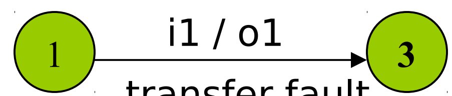
IMPLEMENTATION

SPECIFICATION

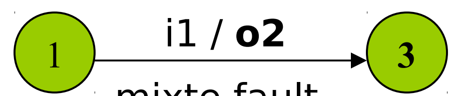




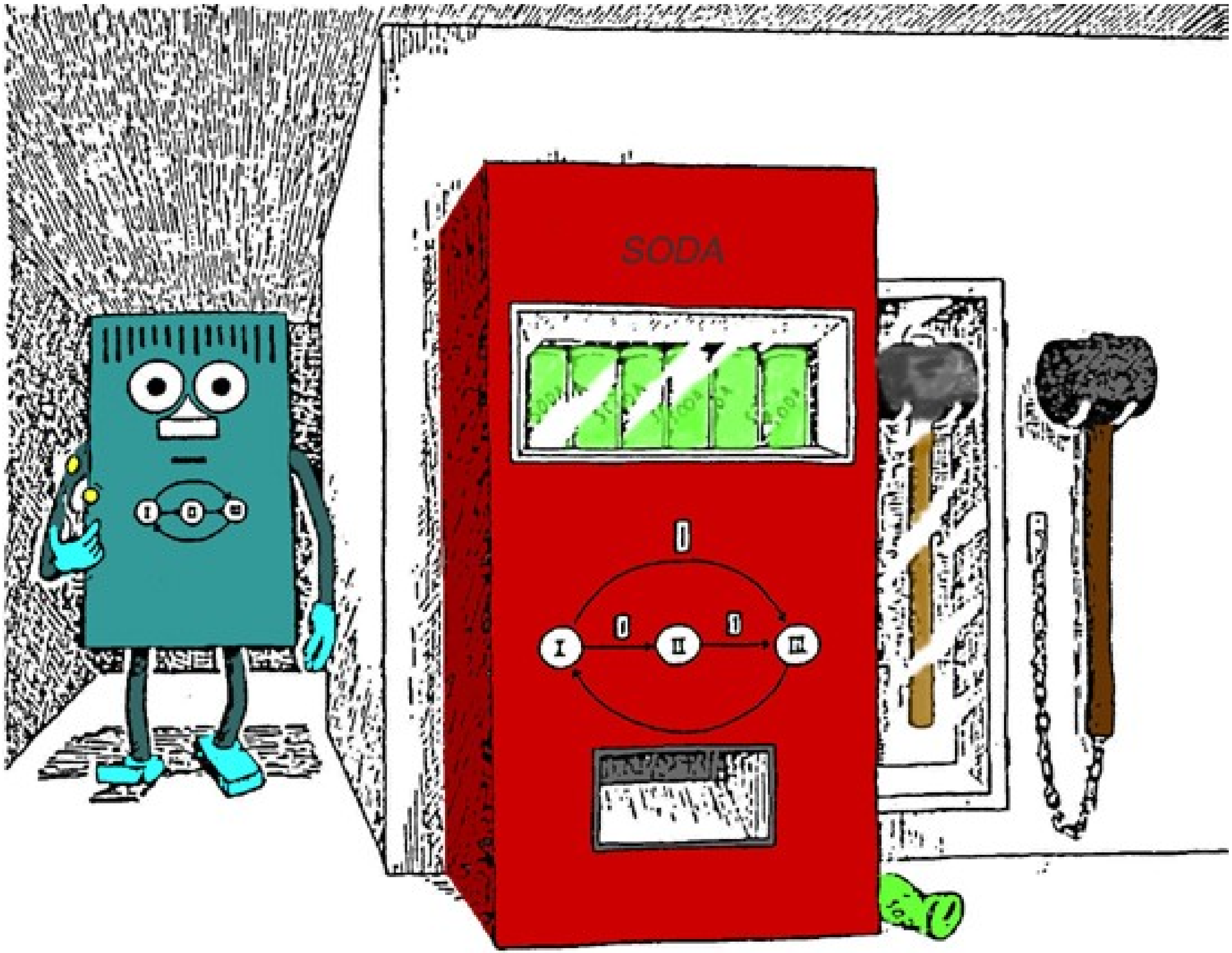
output
fault



transfer fault



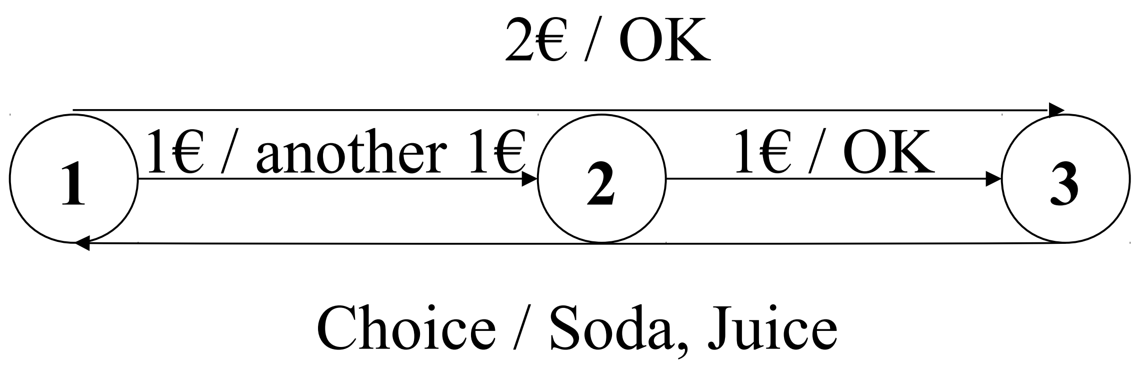
mixte fault



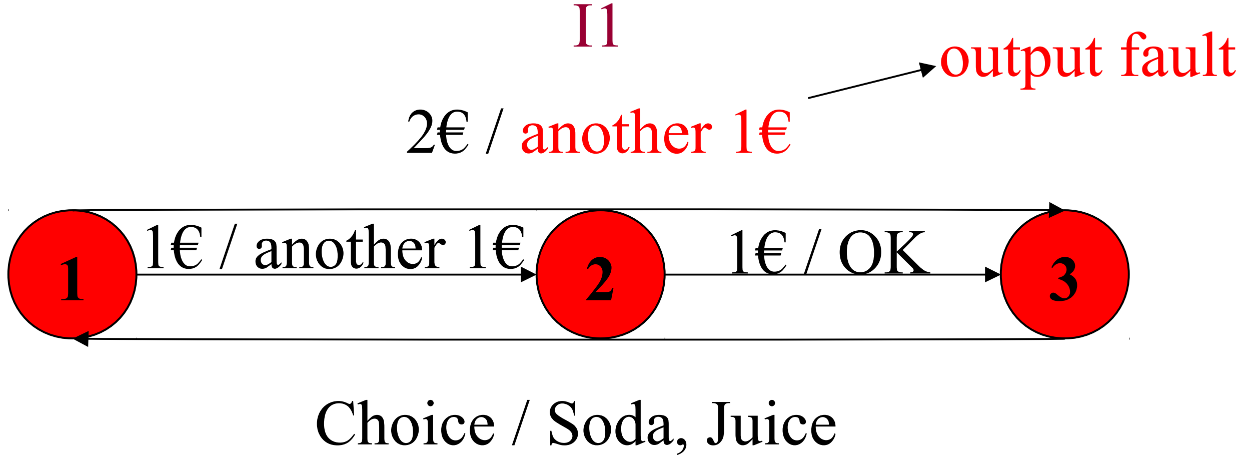
Example: soda vending machine (Lamport's example)

Example: Soda Vending Machine

Specification

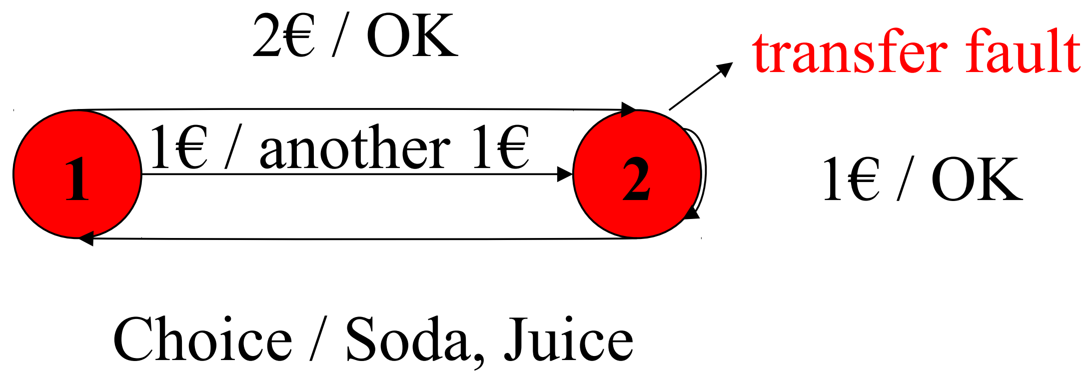


I1

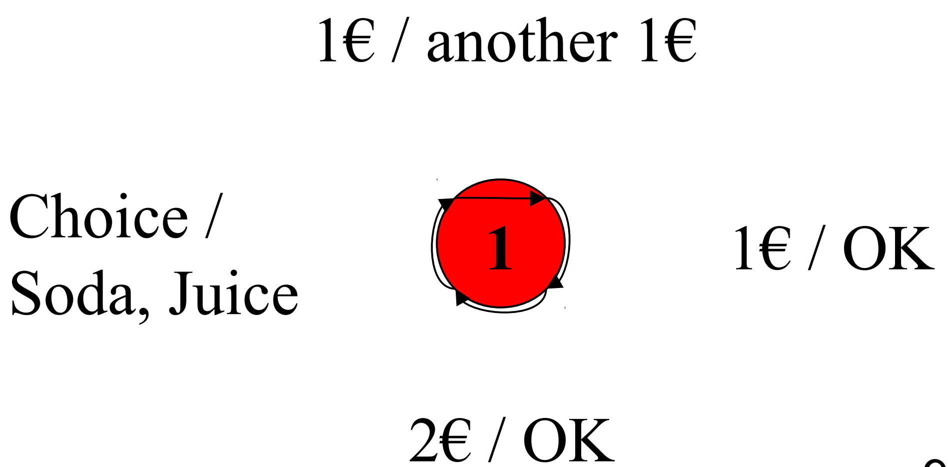


Example: Soda Vending Machine

I2



I3



Definition of a test

- **Step 1 : Put the finite state machine implementation into S_i**

(Drive the protocol implementation into the head state of the transition to be tested)

- **Step 2 : Apply input, observe output**

(Apply the input corresponding to the transition to be tested and check that the output is as expected)

- **Step 3 : Verify S_j**

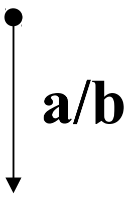
(Verify that the new state of the finite state machine is the same as the tail state of the transition being tested)

Controllability issue in active testing

- How to bring the finite state machine implementation into any given state at any given time during testing ?(Step 1)
  - Non trivial problem because of limited controllability of the finite state machine implementation.
  - It may not be possible to put the finite state machine into the head state of the transition being tested without realizing several transitions.

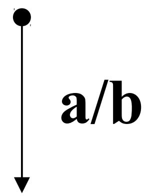
Controllability: examples

Specification



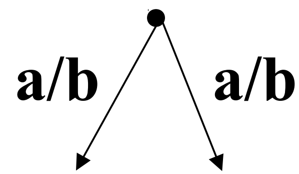
Controllable

Imp1

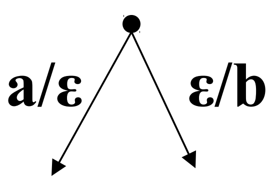


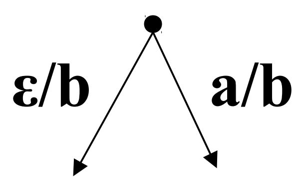
Non controllable

Imp2

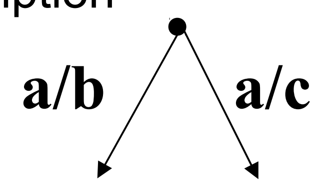


Non controllables





Controllable under fairness assumption



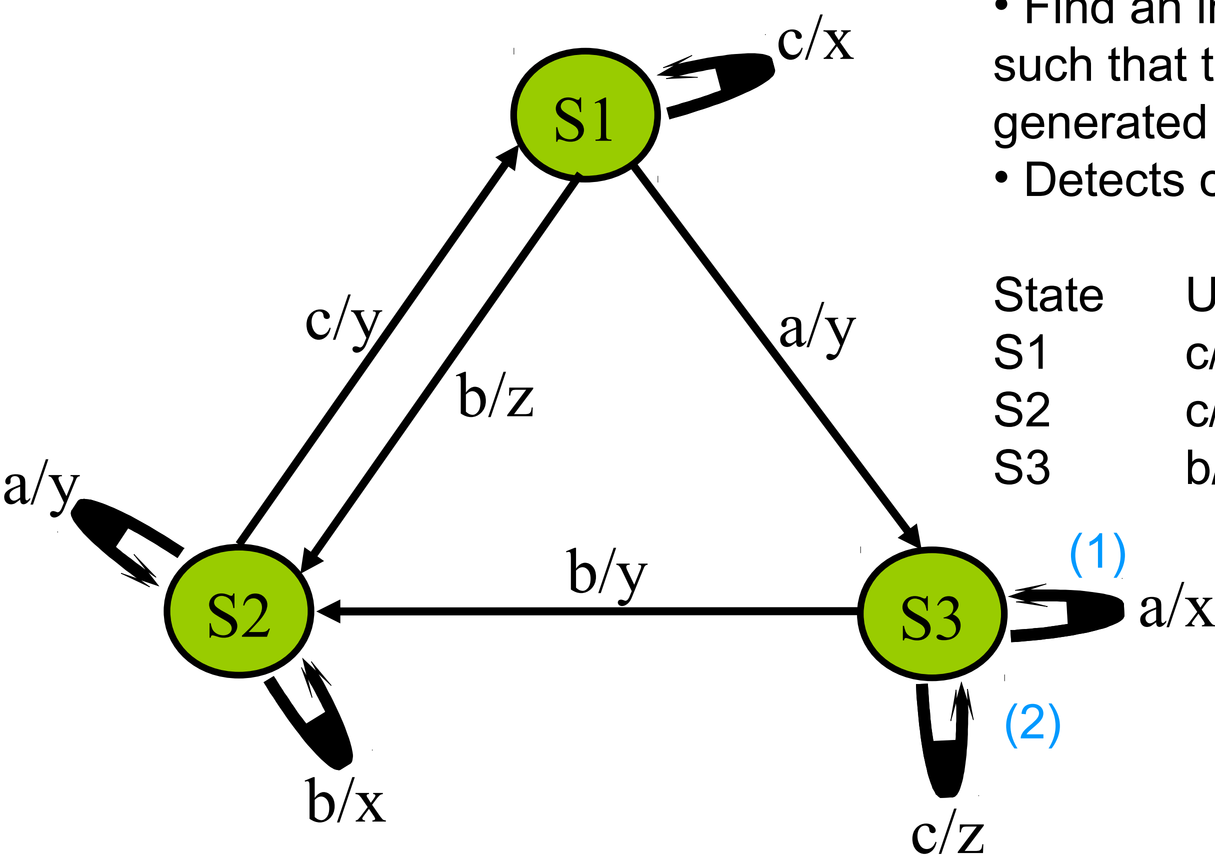
Observability issue in testing

- How to verify that the finite state machine implementation is in a correct state after input/output exchange? (Step 3)
  - *State identification* problem. Difficult because of limited observability of the finite state machine implementation, it may not be possible to directly verify that the finite state machine is in the desired tail state after the transition has been fired.

Solutions to observability issue

- To solve this problem different methods have been proposed:
  - DS (Distinguishing Sequence Method);
  - UIO (Unique Input/Output Sequence Method);
  - W (Distinction Set Method).

Unique Input/Output sequence (UIO sequence)

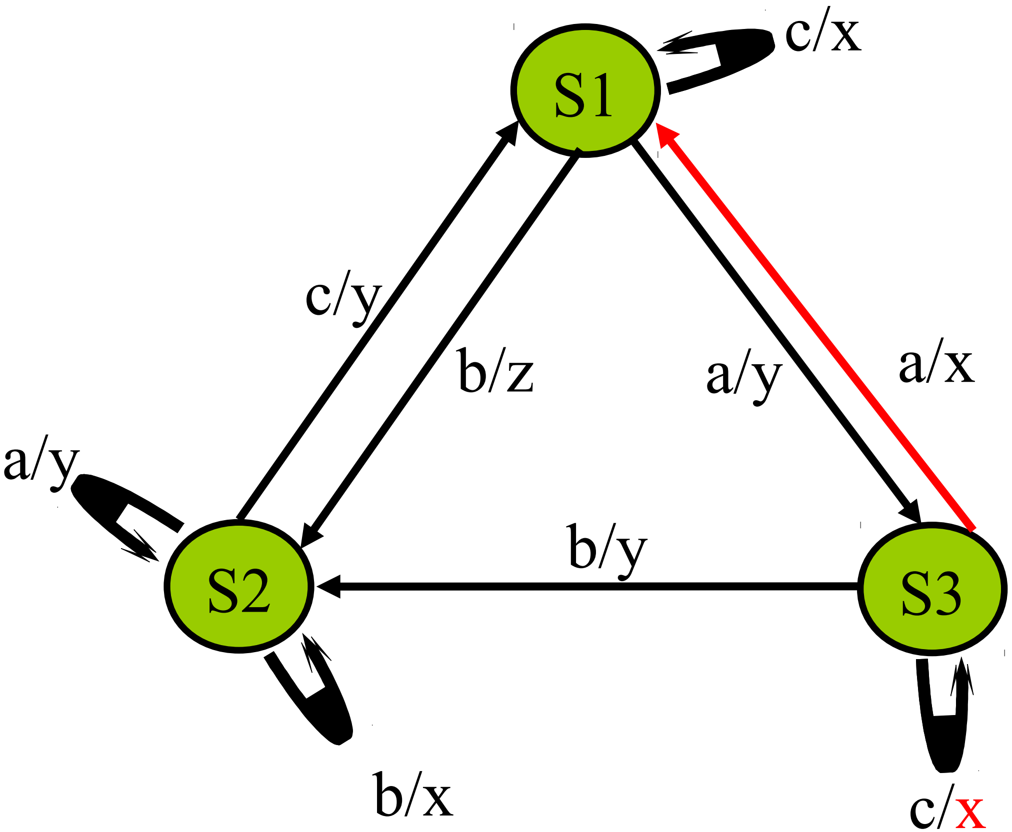


- Find an input sequence for each state such that the output sequence generated is unique to that state
- Detects output and transfer faults.

State	UIO sequences
S1	c/x
S2	c/y
S3	b/y

Test of (1): a/y a/x b/y
Test of (2): a/y c/z b/y

Transfer and output error detection



Implementation

Test of (1): a/y a/x b/y
Test of (2): a/y c/z b/y

Application du test of (1) to the implementation: a/y a/x b/z (transfer error)

Application of test (2) to the implementation: a/y c/x (output error)

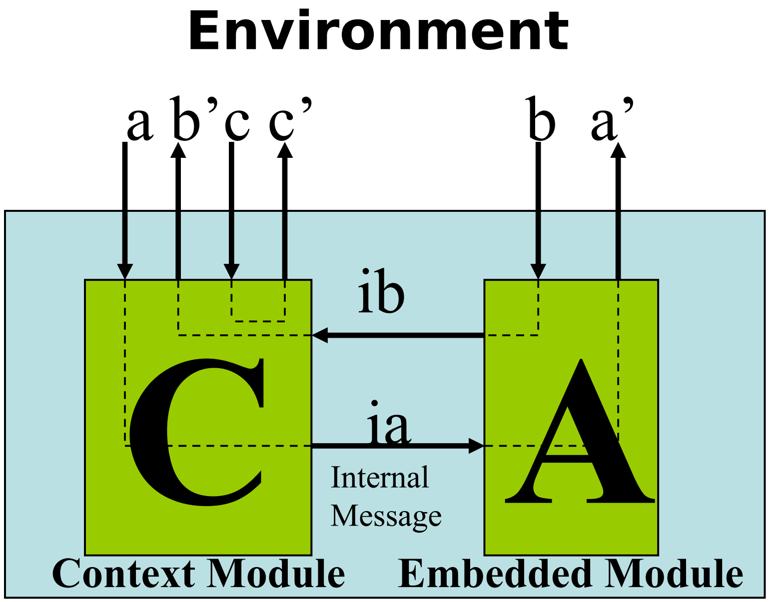
Limitations of active testing

- Non applicable when no direct acces to the implementation under test
- Non controlable interfaces
- Interference on the behaviour of the implementation
- Example: components testing

Components Testing

- Test in context, embedded testing
  - Tests focused on some components of the system, to avoid redondant tests
  - Interfaces semi-controllables
  - In some cases it is not possible to apply

    active testing



Why passive testing?

- **Conformance testing is essentially focused on verifying the conformity of a given implementation to its specification**
  - It is based on the ability of a tester that stimulates the implementation under test and checks the correction of the answers provided by the implementation
- **Closely related to the controllability of the IUT**
  - In some cases this activity becomes difficult, in particular:
    - if the tester has not a direct interface with the implementation
    - or when the implementation is built from components that have to run in their environment and cannot be shutdown or interrupted (for long time) in order to test them

Test by invariants: principle

- Definition: an invariant is a property that is always true
- Two step test:
  - extraction of invariants from the specification or proposed by protocol experts
  - application of invariants on execution event traces from implementation
- Solution: I/O invariants

Test by invariants: I/O invariants

- An invariant is composed of two parts :
  - the test (an input or an output)
  - the preamble (I/O sequence)
- 3 kind of invariants :
  - output invariant (simple invariant)
  - input invariant (obligation invariants)
  - succession invariant (loop invariants)

Test by invariants : Simple (Output) invariant

- Definition : invariant in which the test is an output
- Meaning : « immediatly after the sequence *préambule* there is always the expected output »
- Example :

$(i_1 / o_1)\ (i_2 / o_2)$

(preambule in blue, expected output in red)

Test by invariants : Obligation (Input) invariant

- Definition : invariant in which the test is an input
- Meaning : « immediatly before the sequence *preamble* there is always the input *test* »
- Example :

$(i_1 / o_1)(i_2 / o_2)$

(preamble in blue, test in red)

Test by invariants : succession invariant

- Definition : I/O invariant for complex properties (loops …)
- Example :
  - the 3 invariants below build the property : « only the third i_2 we meet is followed by o_3 »

$(i_1 / o_1)\ (i_2 / o_2)$

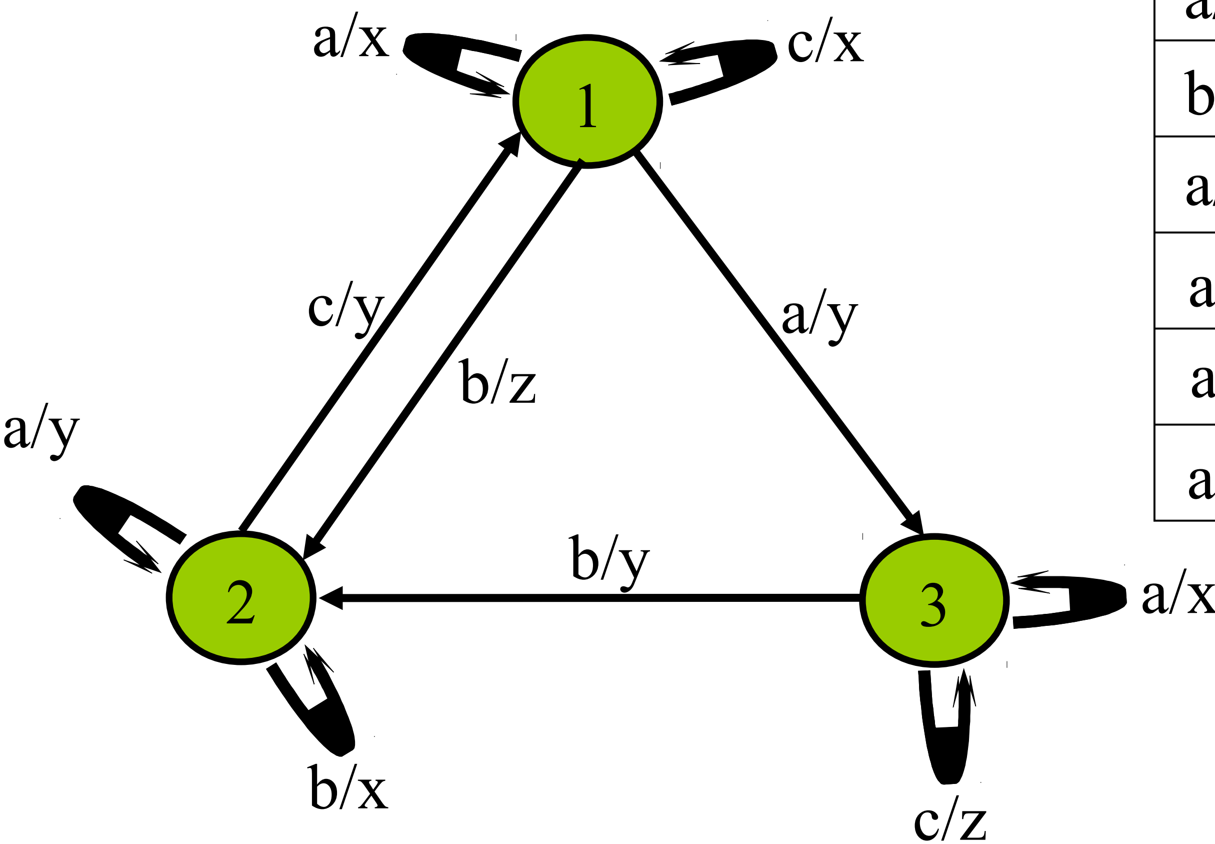
$(i_1 / o_1)\ (i_2 / o_2)\ (i_2 / o_2)$

$(i_1 / o_1)\ (i_2 / o_2)\ (i_2 / o_2)\ (i_2 / o_3)$

SIMPLE INVARIANT

- Simple invariant
  - A trace as $i_1/O_1, \ldots, i_{n-1}/O_{n-1}, i_n/\boldsymbol{O}$ is a simple invariant if each time that the trace $i_1/O_1, \ldots, i_{n-1}/O_{n-1}$ is observed, if we obtain the input i_n then we necessarily get an output belonging to ***O***, where ***O*** is included in the set of expected outputs.
  - *i/o, *, i'/**O*** means that if we detect the transition *i/o* then the first occurrence of the symbol *i'* is followed by an output belonging to the set ***O.***
  - ***** replaces any sequence of symbols not containing the input symbol *i'* and *?* replaces any input or output.

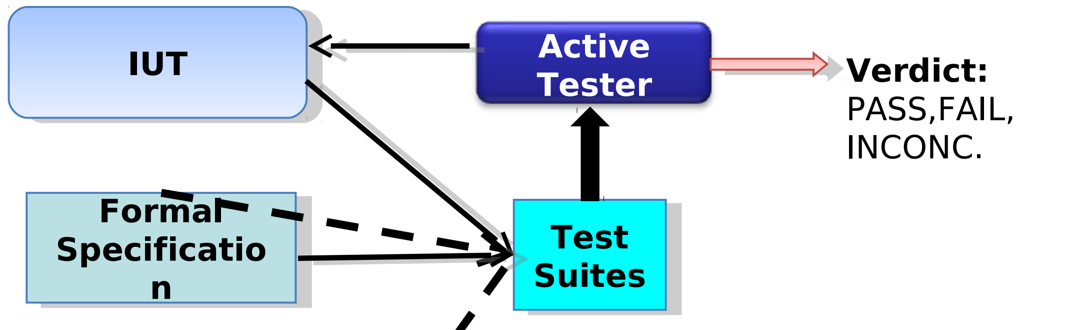
Example



Invariants	Verdicts
a/?, c/z, b/{y}	*True*
b/z, a/{x}	*False*
a/x, *, b/{y, z}	*True*
a/y, ?/$\overline{\{z\}}$	*False*
a/$\overline{\{x\}}$	*False*
a/x, *, ?/$\overline{\{y\}}$	*True*

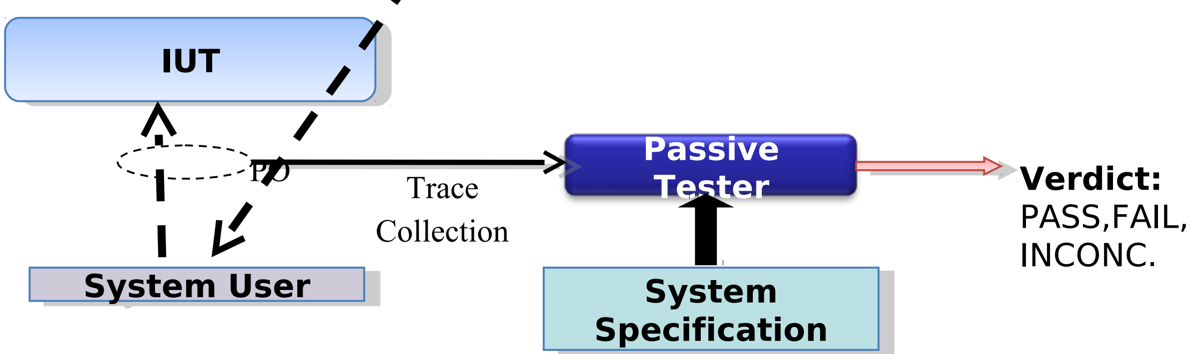
Traces
a/x c/x a/y a/x c/zb/y
c/x a/y a/x c/z b/y
c/y a/x b/z b/x a/y

PASSIVE vs ACTIVE TESTING



pros & cons :

- ☺ Possibility to focus on a specific part of the specification
- ☺ Automatic test generation
- ☹ May modify (crash) the IUT behavior



pros & cons :

- ☺ No interferences with the IUT
- ☺ Test of components
- ☹

DIAMONDS PROJECT
A tool and a case study

Montimage Monitoring Tool

User defined
Reports and views

Add analysis
modules

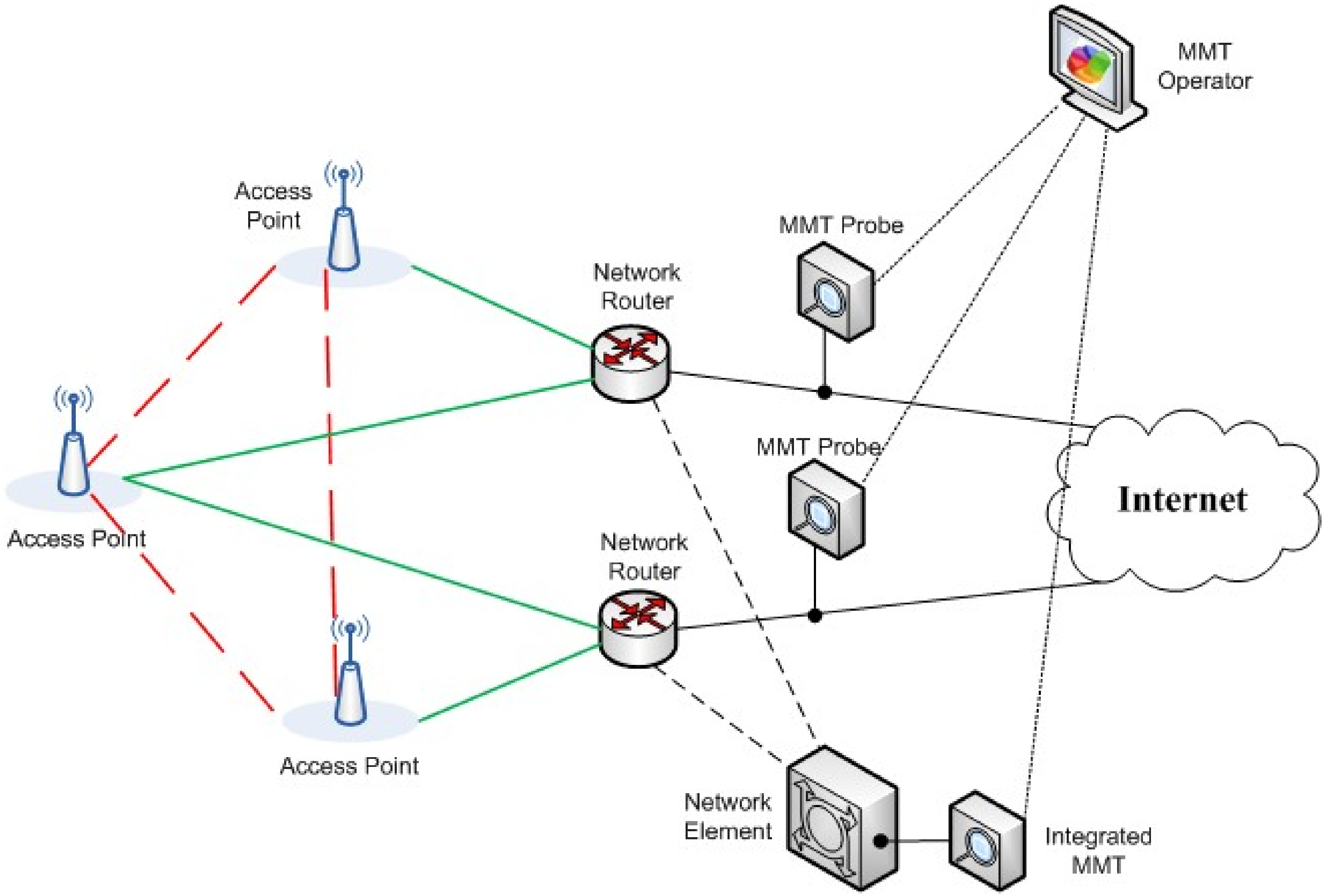
HW/SW Probe
Can be installed
on dedicated HW

Software library
(SDK)
Can be integrated
in 3rd party SW

Add plugins

Modular
solution

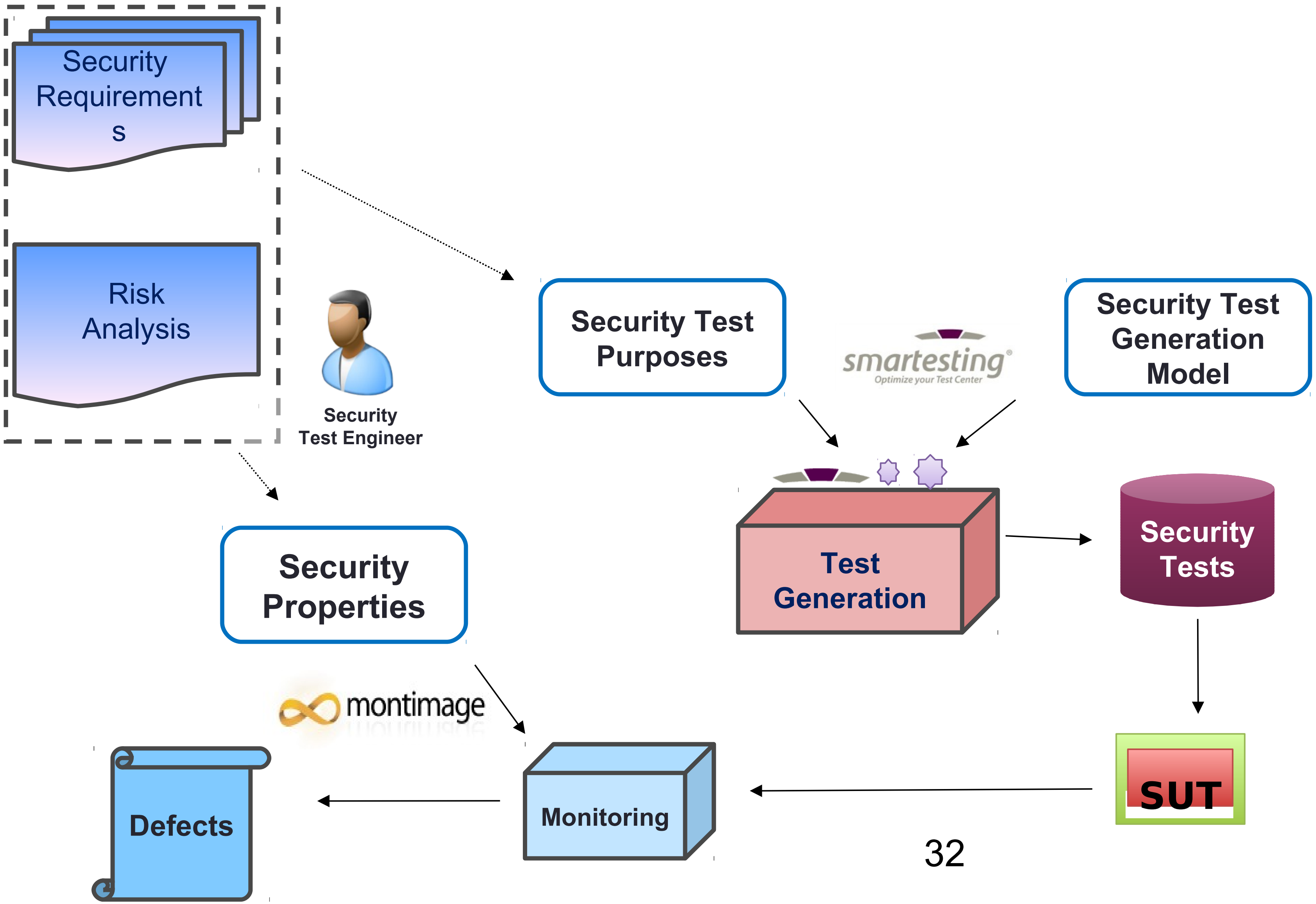
MMT in a transport network scenario



MMT characteristics

- Use of security properties to describe both wanted and unwanted behaviour
  - Not exclusively based on pattern matching like most intrusion detection techniques
  - More abstract description of sequence of events (MMT properties)
  - Can integrate performance indicators, statistics and machine learning techniques; as well as countermeasures
- Allows combining centralised and distributed analysis to detect 0-day attacks (under development)
- Applicable in several domains (at protocol, application and business levels)
- Allows combining active and passive approaches

Composing Active and Passive Testing

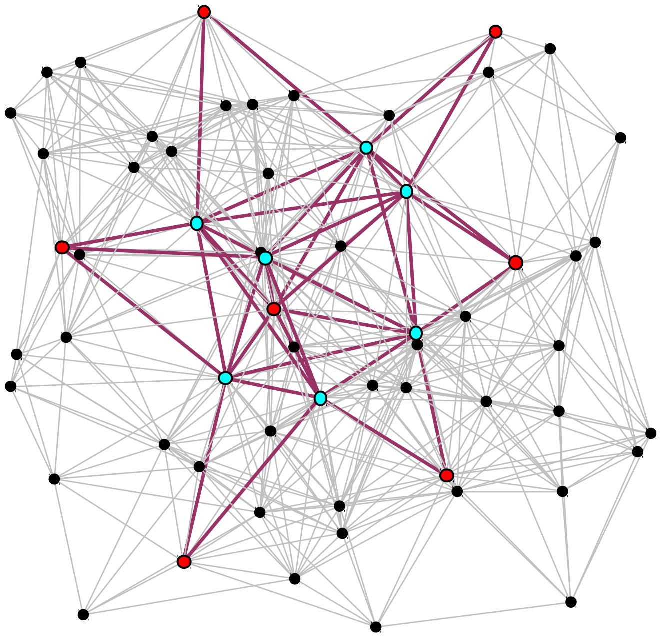


MMT properties

- A security property is composed of 2 parts:
  - A Context
  - A Final condition (trigger)
- The “Context” and “Trigger” are composed of:
  - Events (based on temporal logic)
    - Simple events
    - Complex events linked by logical operators (AFTER/BEFORE/AND/OR)
    - Time constraints, message order, occurrence or non-occurrence, repetition, combination
  - A simple event is composed of:
    - Attributes (values of packet fields, values of sessions attributes, time of reception, length of message, statistics …)
    - Conditions on attributes (IP @ equal to 1.2.3.4)

Radio Protocol case study

- ❑ Provided by Thales: definition of ad-hoc « networking » protocols and algorithms
  - High Data Radio Network Wave Form
- ❑ Technical challenges
  - Automatic network: no initial planning
  - Network continuity whatever are the stations in the network
  - “On the move” automatic network re-organization and operation
  - End-to-end heterogeneous user services transmission: voice, messages
  - Decentralized mesh network. No infrastructure, e.g. base stations



Thales Case study
Security rules specification

Threat: Deny of service by flooding of RLC_CL_UNIT_DATA_ACK messages

Security property: A message RLC_CL_UNIT_DATA_ACK must be preceded with a message RLC_CL_UNIT_DATA_REQ that asked for acknowledgement (R == 00010000) (correlation with the USER_TRANSACTION_ID)

BEFORE

RLC_CL_UNIT_DATA_ACK message

BASE.PROTO == 5152 &&
MSG_RLC_CL_UNIT_DATA_ACK.
USER_TRANSACTION_ID ==
MSG_RLC_CL_UNIT_DATA_REQ.
USER_TRANSACTION_ID.2

Context

RLC_CL_UNIT_DATA_REQ message that asked for acknowledgment

BASE.PROTO == 1056 &&
MSG_RLC_CL_UNIT_DATA_REQ.
QOS_R == 128

Trigger

Results

- A set of 20 security properties have been specified and checked by Montimage
  - Detection of several errors due to a bad generation of traces (using OMNET) and online detection done
  - More properties (~50) are in the design phase

Other works

- Passive testing with time constraints and parameters
- Distributed passive testing
- Differents applications domains (communication and routing protocols, web services)

REFERENCES

1. Pramila Mouttappa, Stephane Maag and Ana Cavalli, "IOSTS based Passive Testing approach for the Validation of data-centric Protocols",12th International Conference on Quality Software (QSIC 2012), X'ian, China, 27th-29th August 2012.

2. Nahid Shahmehri, Amel Mammar, Edgardo Montes de Oca, David Byers, Ana Cavalli, Shanai Ardi and Willy Jimenez, "An Advanced Approach for Modeling and Detecting Software Vulnerabilities", Journal Information and Software Technology, vol 54, issue 9, September 2012.

3. Anderson Morais and Ana Cavalli, "A Distributed Intrusion Detection Scheme for Wireless Ad Hoc Networks", 27th Annual ACM Symposium on Applied Computing (SAC'12), March 25-29, 2012, Riva del Garda (Trento), Italy

4. Fayçal Bessayah, Ana Cavalli, A Formal Passive Testing Approach For Checking Real Time Constraints, 7th International Conference on the Quality of Information and Communications Technology, September 29th 2010, Porto, Portugal.

5. César Andrés, Stephane Maag, Ana Cavalli, Mercedes G. Merayo, Manuel Nunez, "Analysis of the OLSR Protocol by using formal passive testing", APSEC 2009, December 2009, Penang, Malaysia.

6. Felipe Lalanne, Stephane Maag, Edgardo Montes de Oca, Ana Cavalli, Wissam Mallouli and Arnaud Gonguet , An Automated Passive Testing Approach for the IMS PoC Service, 24th ACM/IEEE International Conference on Automated Software Engineering, November 2009, Auckland, New Zealand.
7. Ana Rosa Cavalli, Azzedine Benameur, Wissam Mallouli, Keqin Li, A Passive Testing Approach for Security Checking and its Practical Usage for Web Services Monitoring, invited paper, NOTERE 2009, 29-June 3-July, 2009, Montréal, Canada.

8. Ana Cavalli, Stephane Maag and Edgardo Montes de Oca, A Passive Conformance Testing Approach for a Manet Routing Protocol, The 24th Annual ACM Symposium on Applied Computing SAC'09, March 9-12 2009, Hawaii, USA.

9. Ana R. Cavalli, Edgardo Montes De Oca, Wissam Mallouli, Mounir Lallali, Two Complementary Tools for the Formal Testing of Distributed Systems with Time Constraints, The 12-th IEEE/ACM International Symposium on Distributed Simulation and Real Time Applications (DS-RT 2008), October 27-29, Vancouver, Canada.

10. Wissam Mallouli, Fayçal Bessayah, Ana R. Cavalli, Azzedine Benameur, Security Rules Specification and Analysis Based on Passive Testing, The IEEE Global Communications Conference (GLOBECOM 2008), November 30 - December 04, New Orleans, USA.

11. J.-M. Orset, B. Alcalde and A. Cavalli, An EFSM-Based Intrusion Detection System for Ad Hoc Networks, ATVA 05, Taipei, Taiwan, October 2005.

12. E. Bayse, A. Cavalli, M. Núñez, and F. Zaïdi. A passive testing approach based on invariants: application to the wap. In Computer Networks, volume 48, pages 247-266. Elsevier Science, 2005.

13. César Andrés, María-Emilia Cambronero, Manuel Núñez:
Formal Passive Testing of Service-Oriented Systems. IEEE SCC 2010: 610-613

14. César Andrés, Mercedes G. Merayo, Manuel Núñez: Multi-objective Genetic Algorithms: Construction and Recombination of Passive Testing Properties. SEKE 2010: 405-410

15. César Andrés, Mercedes G. Merayo, Manuel Núñez: Passive Testing of Timed Systems. ATVA 2008: 418-427