Active and Passive Testing in an Industrial Setting

ANA CAVALLI- KHALIFA TOUMI- WISSAM MALLOULI

INSTITUT MINES-TELECOM/TELECOM SUDPARIS-MONTIMAGE

TAROT 2015

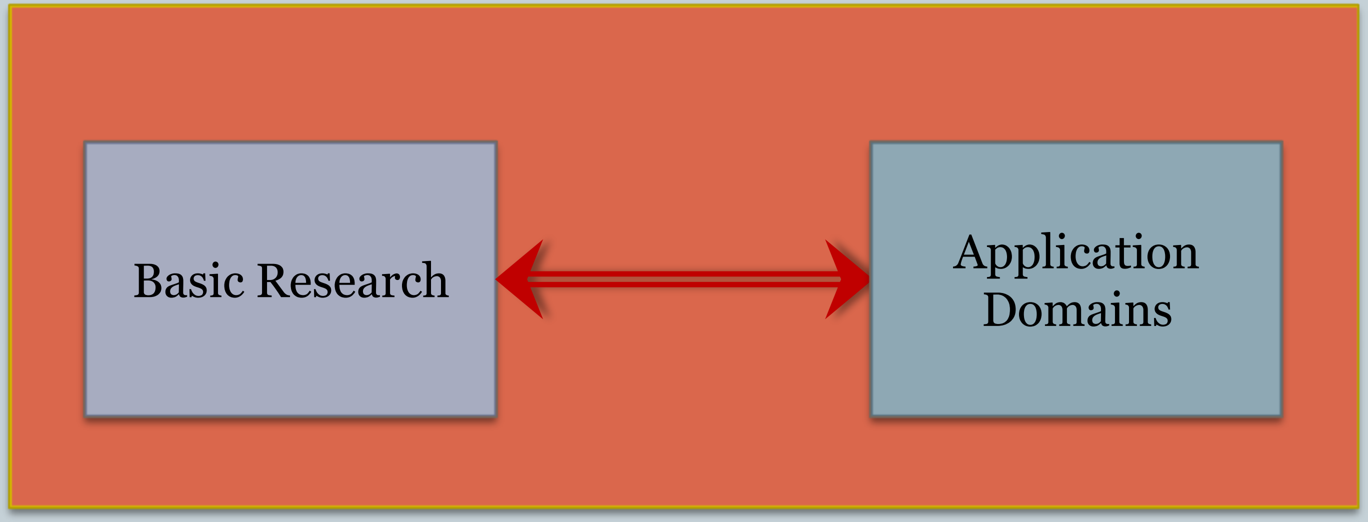
CADIZ- JUNE 29-JULY 2,2015

Motivation

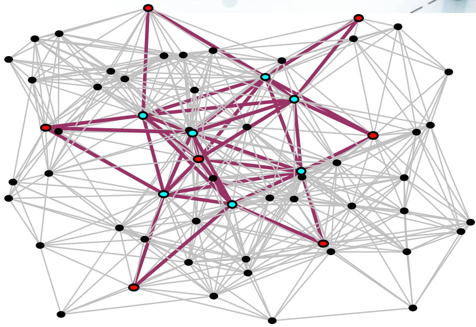
- Show the evolution of active testing to monitoring (passive testing) techniques
- Explain the differences and complementarity of these techniques
- Application to an industrial case study

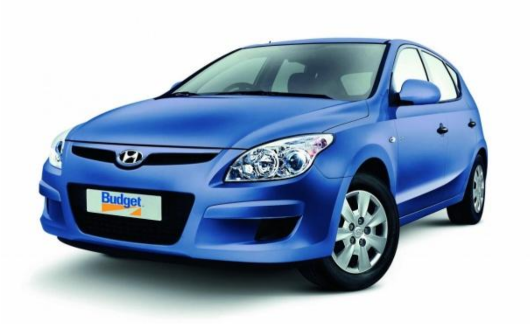
A collaborative model of research

- Our research model is based in:
  - Basic and applied research
  - Evaluation of results in real environments
  - Strong collaboration with industrial partners



Different Application Domains





DETAILED WEB SERVICES PROCESS

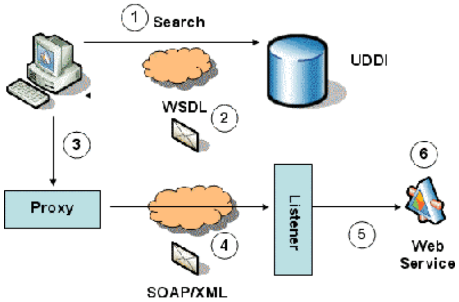
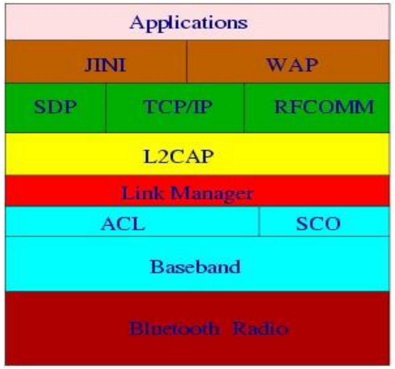


Figure 1: The process flow of a Web service



Applications

JINI | WAP

SDP | TCP/IP | RFCOMM

L2CAP

Link Manager

ACL | SCO

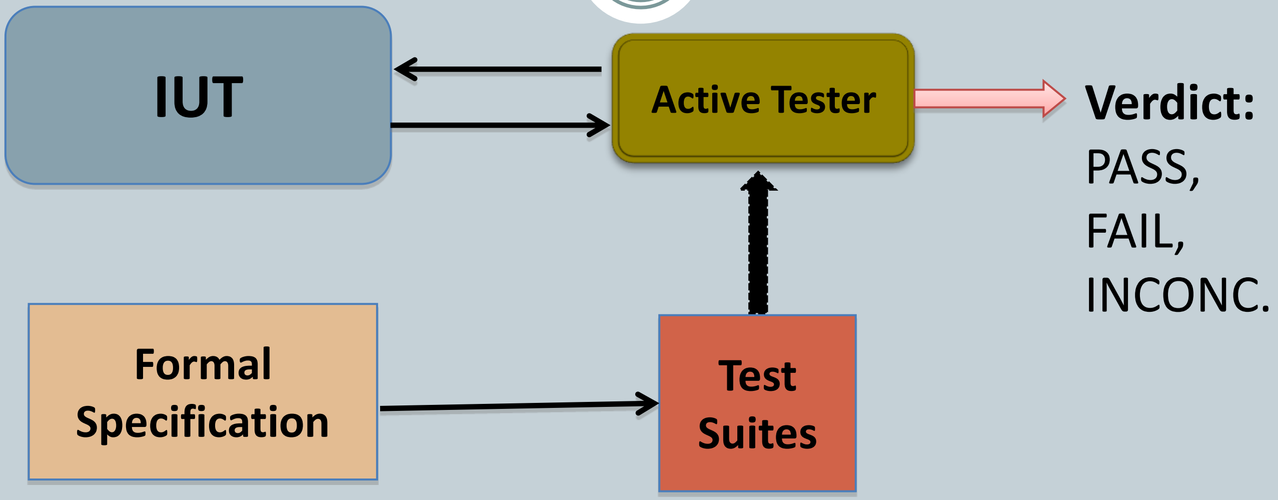
Baseband

Bluetooth Radio

Testing

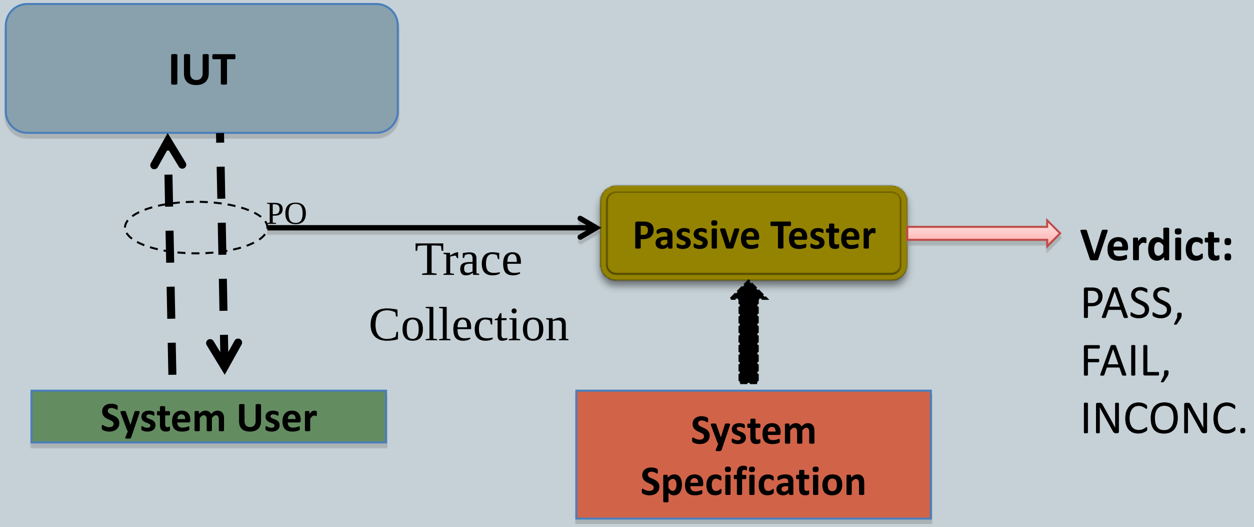
- Testing: The process of executing software with the intent of finding and correcting faults
- Conformance testing: The process of checking if the implementation under test conforms the specification
  - Two techniques: active and passive testing (monitoring)
  - This presentation will focus on both of them, to show that there are many common objectives and challenges

What is active testing ?



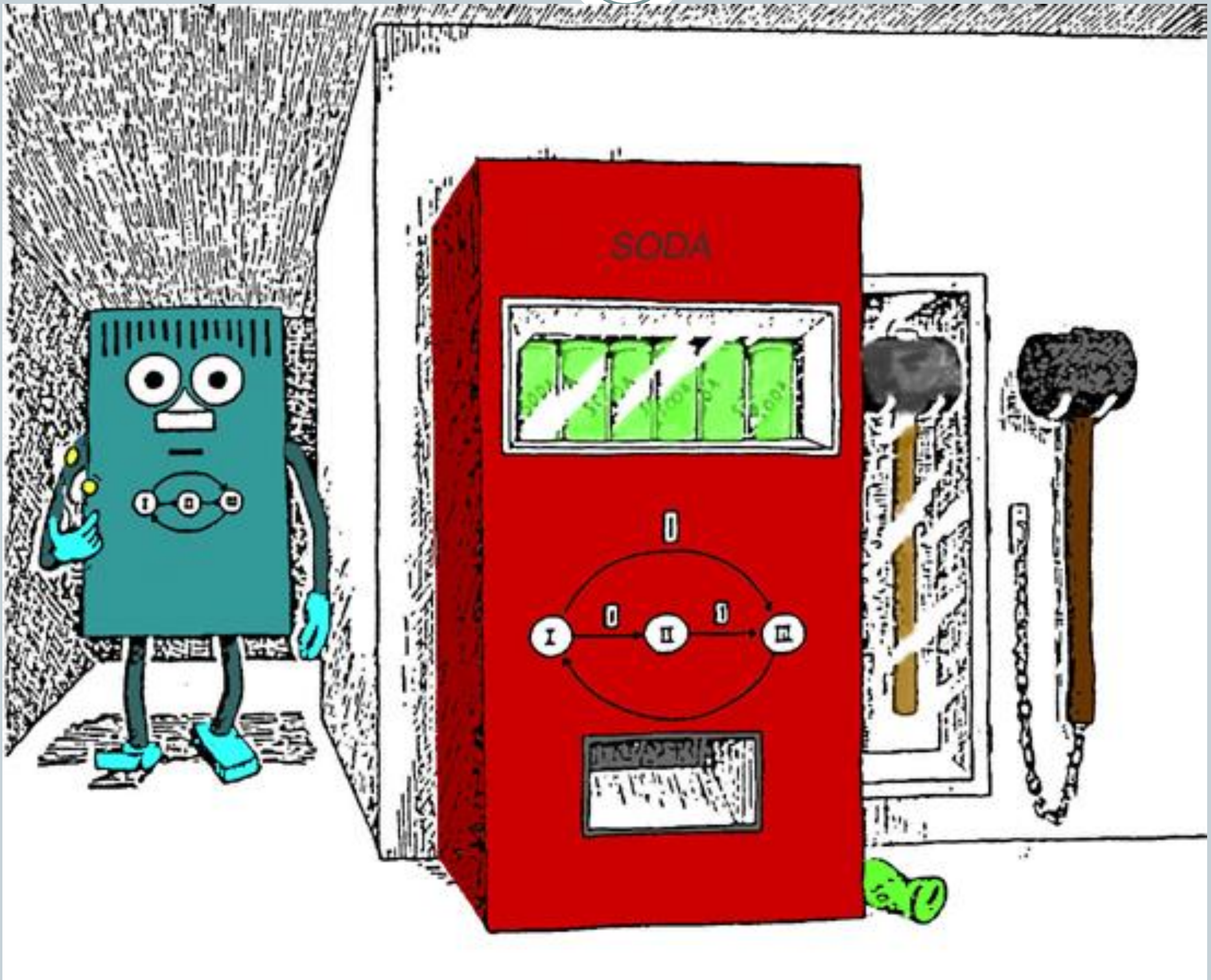
- Usually called Model Based Testing (MBT)
- It is assumed that the tester controls the implementation. Control means: after sending an input and after receiving an output, the tester knows what is the next input to be send
- The tester can guide the implementation towards specific states
- Automatic test generation methods can be defined
- Usually a test case is a set of input sequences

What is monitoring (passive testing) ?



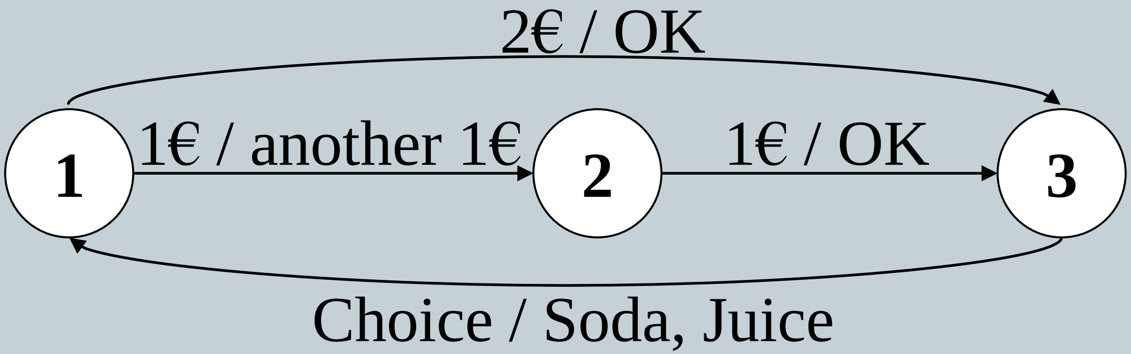
- Passive testing consists in analyzing the traces recorded from the IUT and trying to find a fault by comparing these traces with either the complete specification or by verifying some specifics requirements (or properties) during normal runtime
- No interferences with the IUT
- It is also referred to as monitoring

Fault models for active testing
Example: Soda vending machine

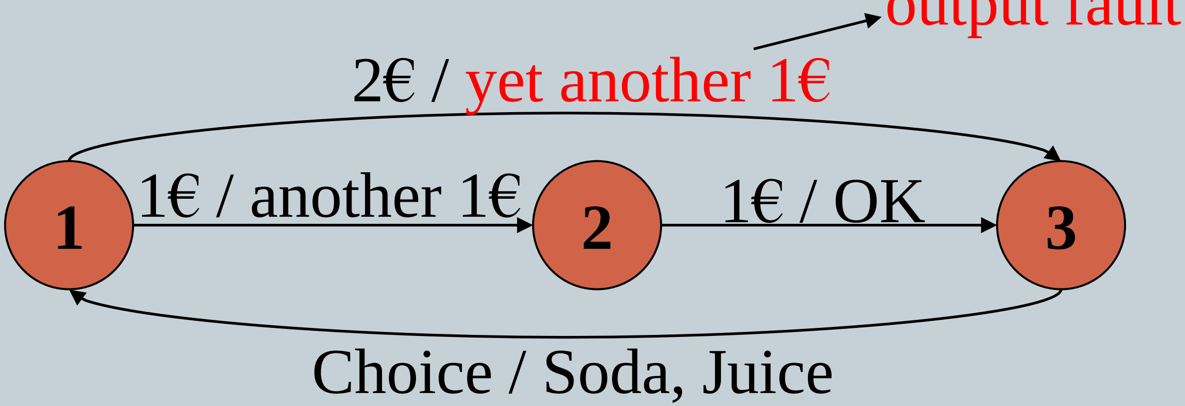


Example: Soda Vending Machine

Specification

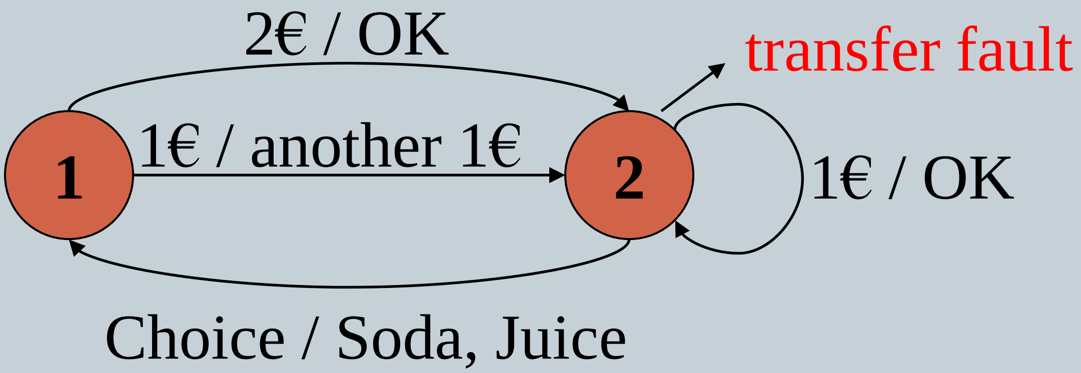


I1

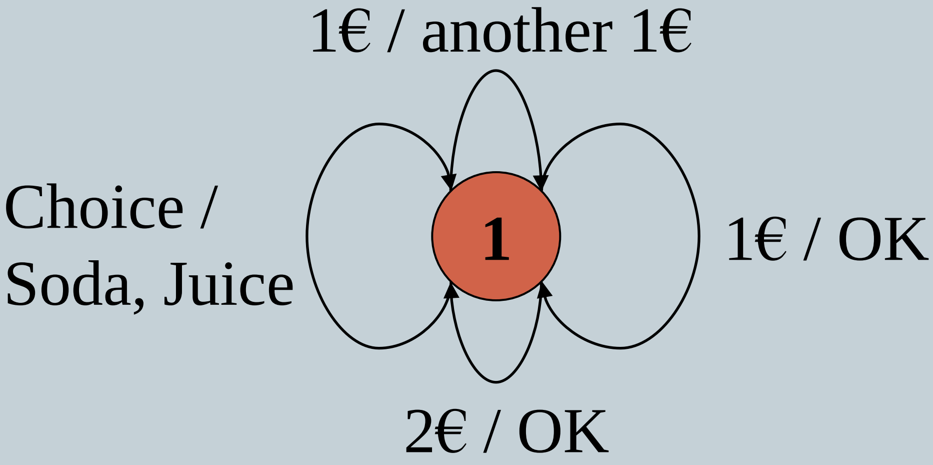


Example: Soda Vending Machine

I2



I3

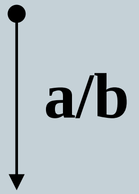


Controllability issue in active testing

- How to bring the finite state machine implementation into any given state at any given time during testing ?
  - 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

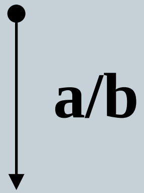
Controllability: examples

Specification



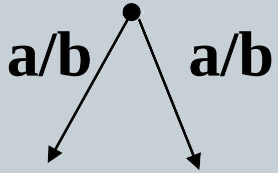
Controllable

Imp1



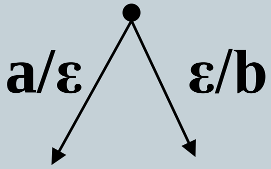
Non controllable

Imp2

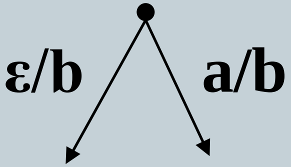


Non controllable

Imp3

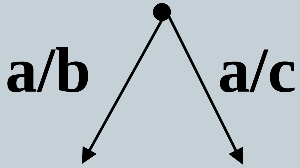


Imp4



Controllable under fairness assumption

Imp5



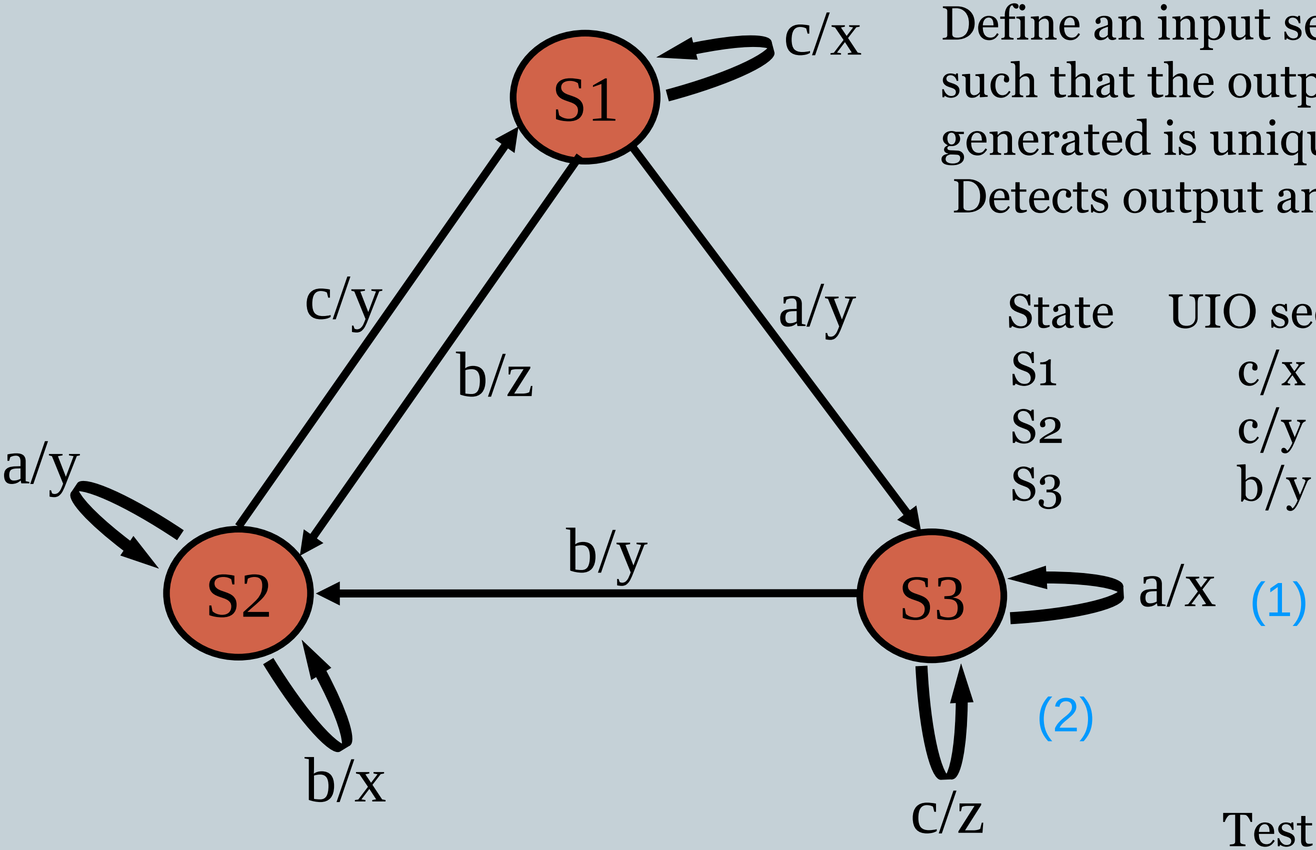
Observability issue in testing

- How to verify that the finite state machine implementation is in a correct state after input/output exchange?
  - *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)
  - UIO (Unique Input/Output Sequence)
  - W (Distinction Set)

Unique Input/Output sequence (UIO sequence)



Define an input sequence for each state such that the output sequence generated is unique to that state.

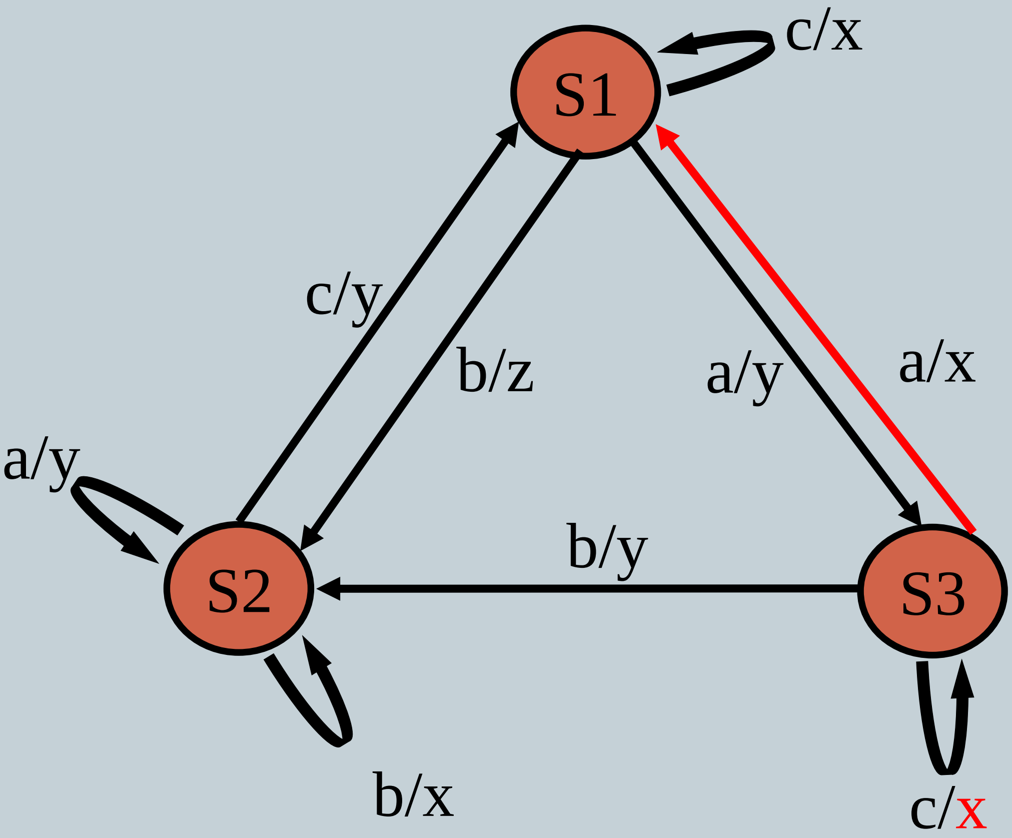
Detects output and transfer faults.

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



Faulty 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)

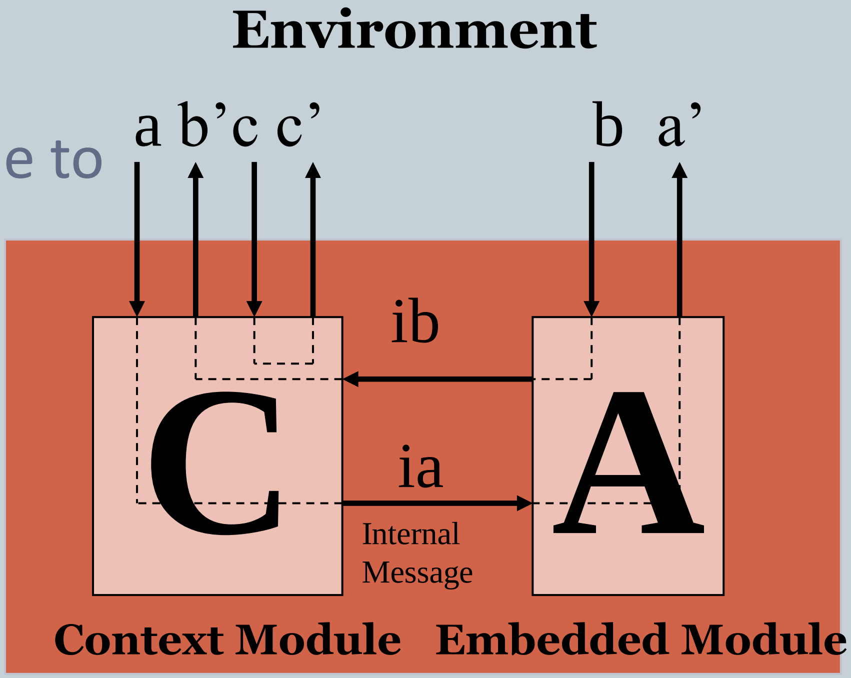
Application of test (2) to the implementation:
a/y c/x (output error)

Limitations of active testing

- Non applicable when no direct access to the implementation under test
- Semi- controllable interfaces (component testing)
- Interferences on the behaviour of the implementation

Components Testing

- Test in context, embedded testing:
  - Tests focused on some components of the system, to avoid redundant 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

Controllability and observability issues in passive testing

- **Controllability**
  - No **controllability** issue because no interaction with the implementation under test
- **Observability**
  - It is assumed that to perform passive testing it is necessary to observe the messages exchanges between modules.
  - Passive testing is a Grey Box testing technique
- **Fault detection using passive testing**
  - It is possible to detect output faults
  - It is possible to detect transfer faults under some hypothesis: to initialise the IUT in order to be sure that the implementation is in the initial state and then perform passive testing

Invariant based passive testing approach

- In this approach a set of properties are extracted from the specification or proposed by the protocol experts, and then the trace resulting from the implementation is analyzed to determine whether it validates this set of properties.

- These extracted set of properties are called invariants because they have to hold true at every moment.

Invariant based passive testing approach

- Definition: an invariant is a property that is always true.
- Two test steps:
  - 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 invariant)
  - Succession invariant (loop invariant)

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 is followed by o_3 »

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

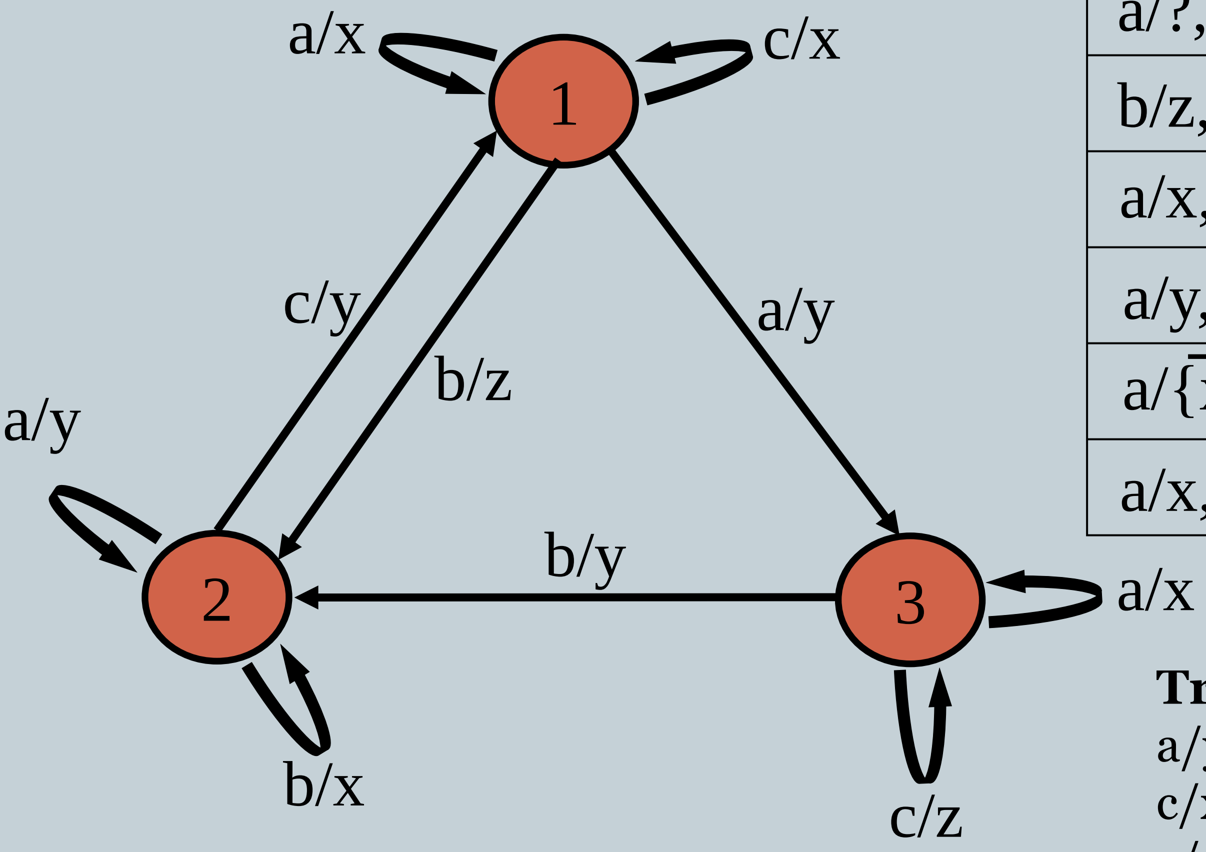
$(i_1 / o_1)\ (i_2 / o_2)\ (i_2 / o_2)$

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

Simple invariant

- 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, *, $i'/\boldsymbol{O}$ 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

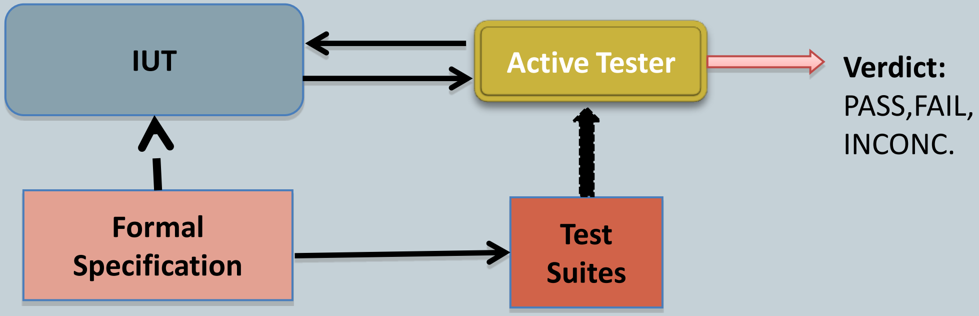


Invariants	Verdict
a/?, c/z, b/{y}	*True*
b/z, a/{x}	*False*
a/x, *, b/{y, z}	*True*
a/y, ?/$\overline{\{z\}}$	*False*
a/$\overline{\{x\}}$	*False*
a/x, *, ?/$\overline{\{y\}}$	*True*

Traces

a/y c/z b/y a/y a/x c/z b/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



IUT

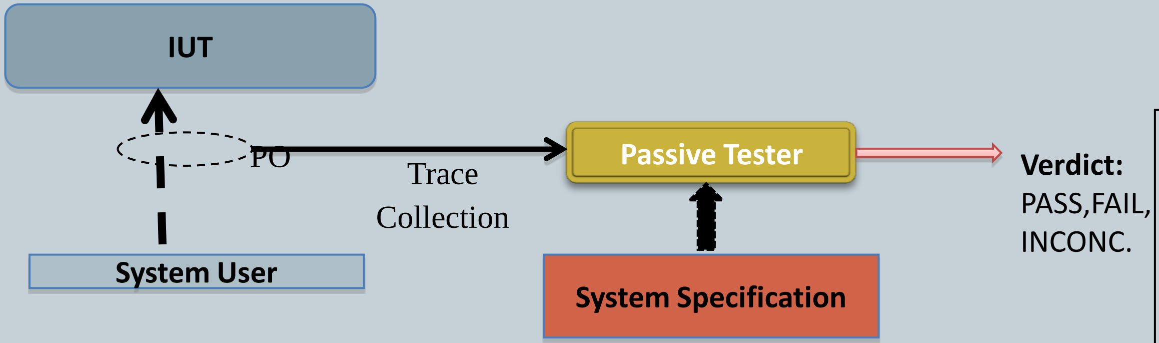
Active Tester

Formal Specification

Test Suites

Verdict:
PASS,FAIL,
INCONC.

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



IUT

PO

Trace Collection

Passive Tester

System User

System Specification

Verdict:
PASS,FAIL,
INCONC.

- ☺ No interferences with the IUT
- ☺ No models needed
- ☺ Full monitoring automation
- ☹ Grey box testing

Related Work

- Monitoring of routing protocols for ad hoc (OLSR protocol) and mesh networks based on a distributed approach (Batman protocol) (TSP)
- Monitoring for secure interoperability – Application to a multi-source information system (TSP)
- Monitoring with time constraints (C. Andrés, M. Nuñez and M. Merayo)
- Other works by (T. Jeron and H. Marchand, A. Ulrich and A. Petrenko)

Run Time Verification

- Approach proposed by researchers of verification (model checking) community
- Passive testing developed by the testing community
- EAGLE and RuleR tools proposed by Barringer and al. in 2004 and 2010 respectively, based on temporal logics and rewriting rules for properties description
- Others tools: Tracematches, [Avgustinov et al. 2007], J-LO [Bodden 2005]and LSC [Maoz and Harel 2006]

Active and Passive Testing in the INTER-TRUST project

Interoperable **Trust** Assurance Infrastructure

softeco sismat information technology

montimage

TELECOM Bretagne TELECOM SudParis

ma UNIVERSIDAD DE MÁLAGA

SEARCH-LAB SECURITY EVALUATION ANALYSIS AND RESEARCH LABORATORY

Scytl Innovating Democracy

indra

UNIVERSITAT ROVIRA I VIRGILI

University of Reading

UNIVERSIDAD DE MURCIA

- Project co-funded by the European Union under the Information and Communication Technologies theme of the 7th Framework Programme for R&D ICT-2011.1.4 Trustworthy ICT contract n. 317731
- November 2012 – April 2015 (30 months)
- www.inter-trust.eu

INTER-TRUST project Overall Objectives

- Develop a dynamic and scalable framework
  - to support **trustworthy** services in **heterogeneous** networks and devices
  - based on the enforcement of **interoperable and changing** security policies
- Addressing the needs of developers, integrators and operators
  - to **develop and operate** systems in a secure trusted manner
  - dictated by **negotiated** security policies through dynamic security SLAs
- Separate the security concerns from the functional requirements => AOP

INTER-TRUST project Overall Objectives

Validation

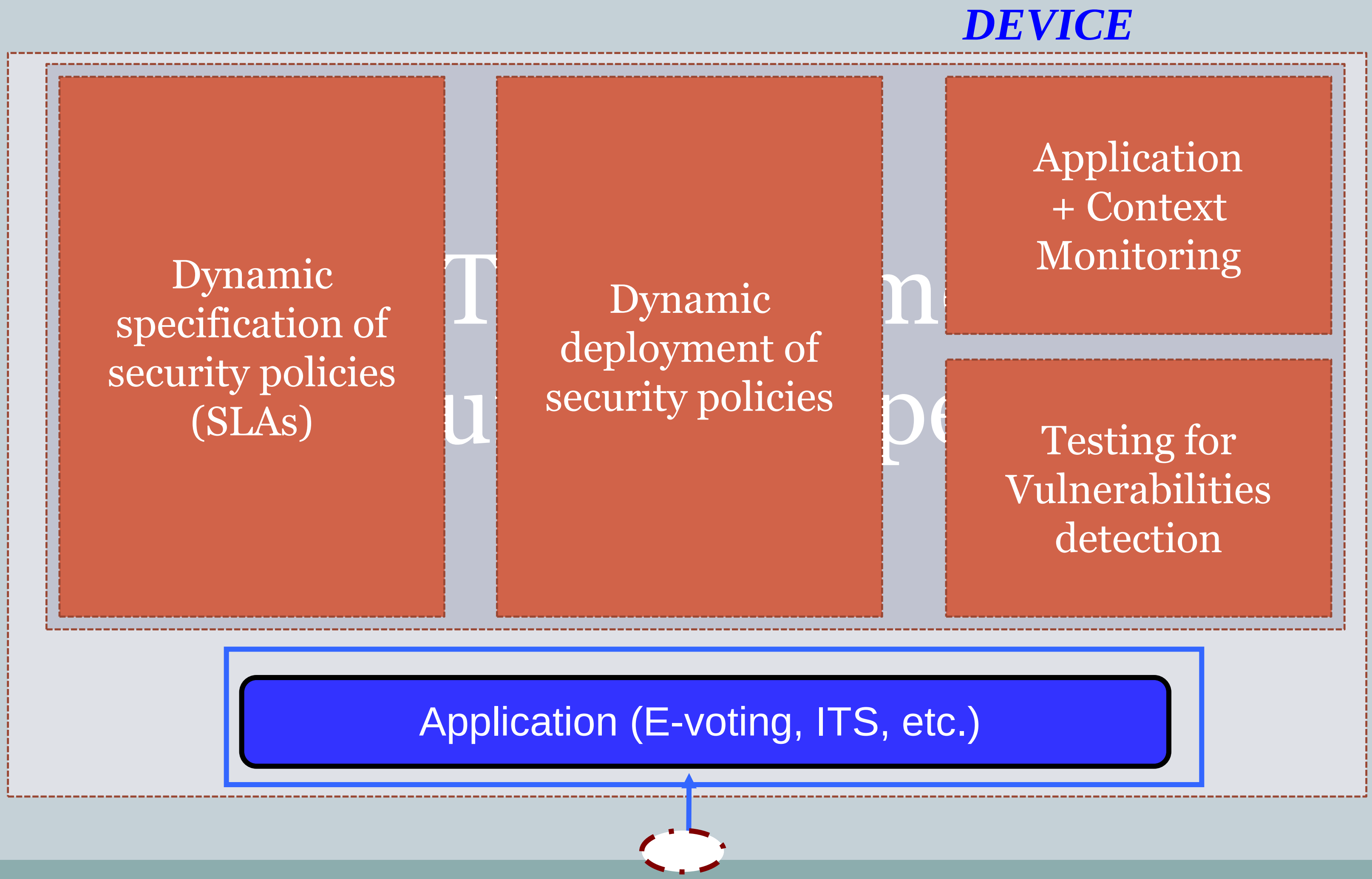
Validate the architecture, techniques and tools developed using **two completely different case studies** with complex, high-demanding critical services

- V2X communications
- E-Voting

INTER-TRUST framework

- The INTER-TRUST framework allows the secure interoperation enforcement and supervision between communicating and heterogeneous systems. It allows:
  - The negotiation and adaptability of security policies according to the available resources and changes in the environment
  - The dynamic deployment of negotiated security policies using a selected AOP framework
  - The automatic verification and validation of security policies by means of testing & monitoring techniques

I-T Framework architecture



DEVICE

Dynamic specification of security policies (SLAs)

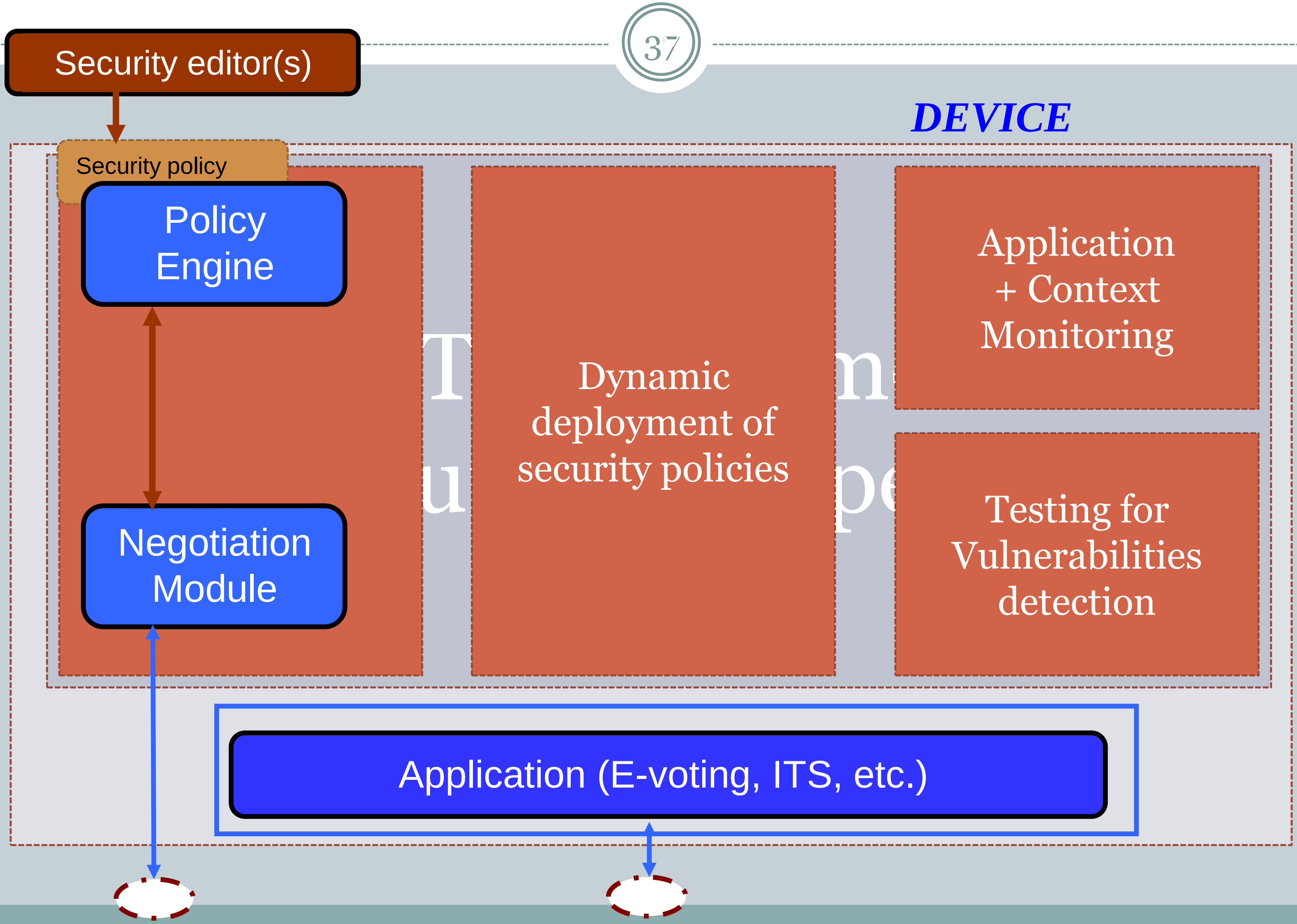
Dynamic deployment of security policies

Application + Context Monitoring

Testing for Vulnerabilities detection

Application (E-voting, ITS, etc.)

I-T Framework architecture



Security editor(s)

DEVICE

Security policy

Policy Engine

Negotiation Module

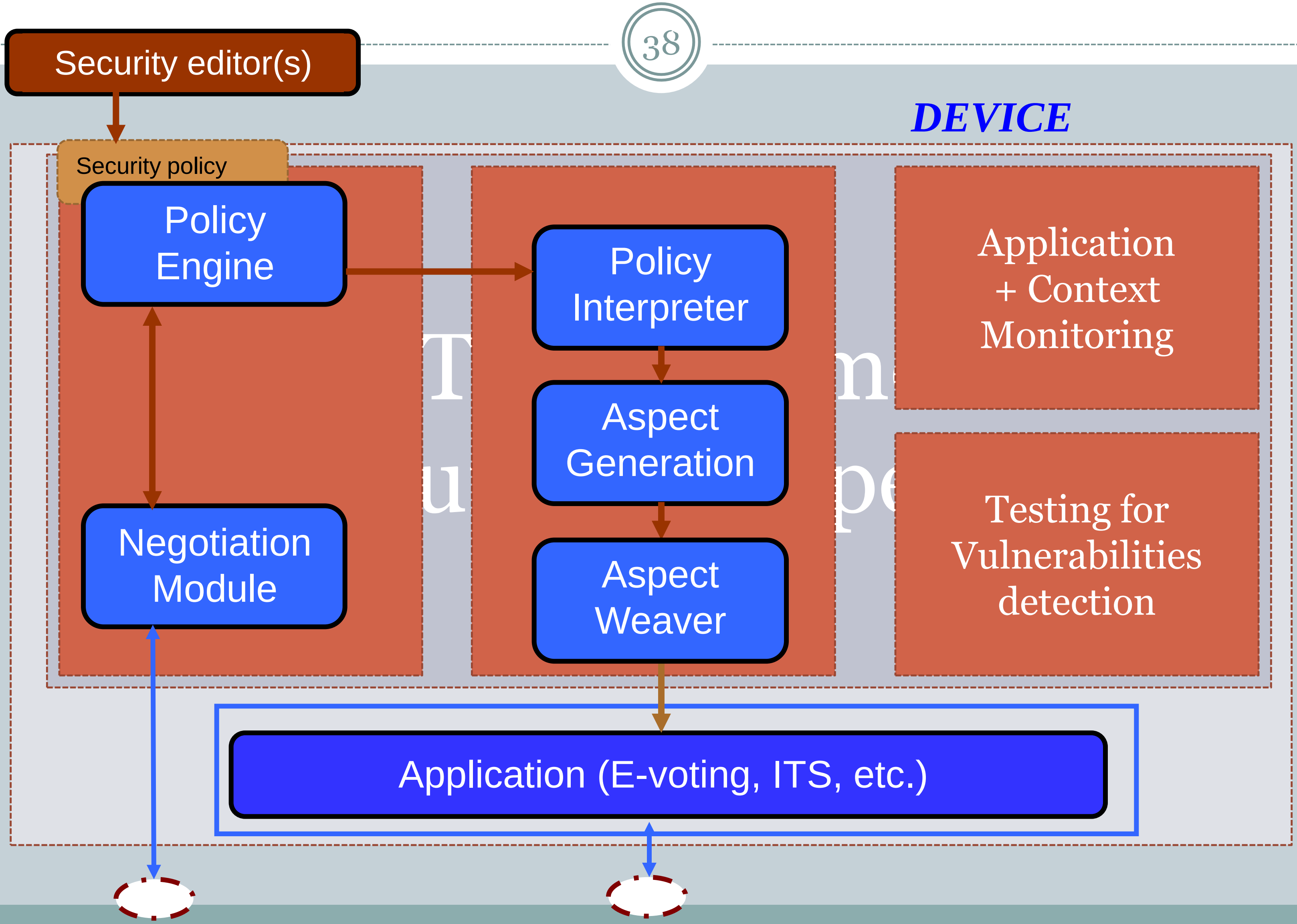
Dynamic deployment of security policies

Application + Context Monitoring

Testing for Vulnerabilities detection

Application (E-voting, ITS, etc.)

I-T Framework architecture



Security editor(s)

DEVICE

Security policy

Policy Engine

Negotiation Module

Policy Interpreter

Aspect Generation

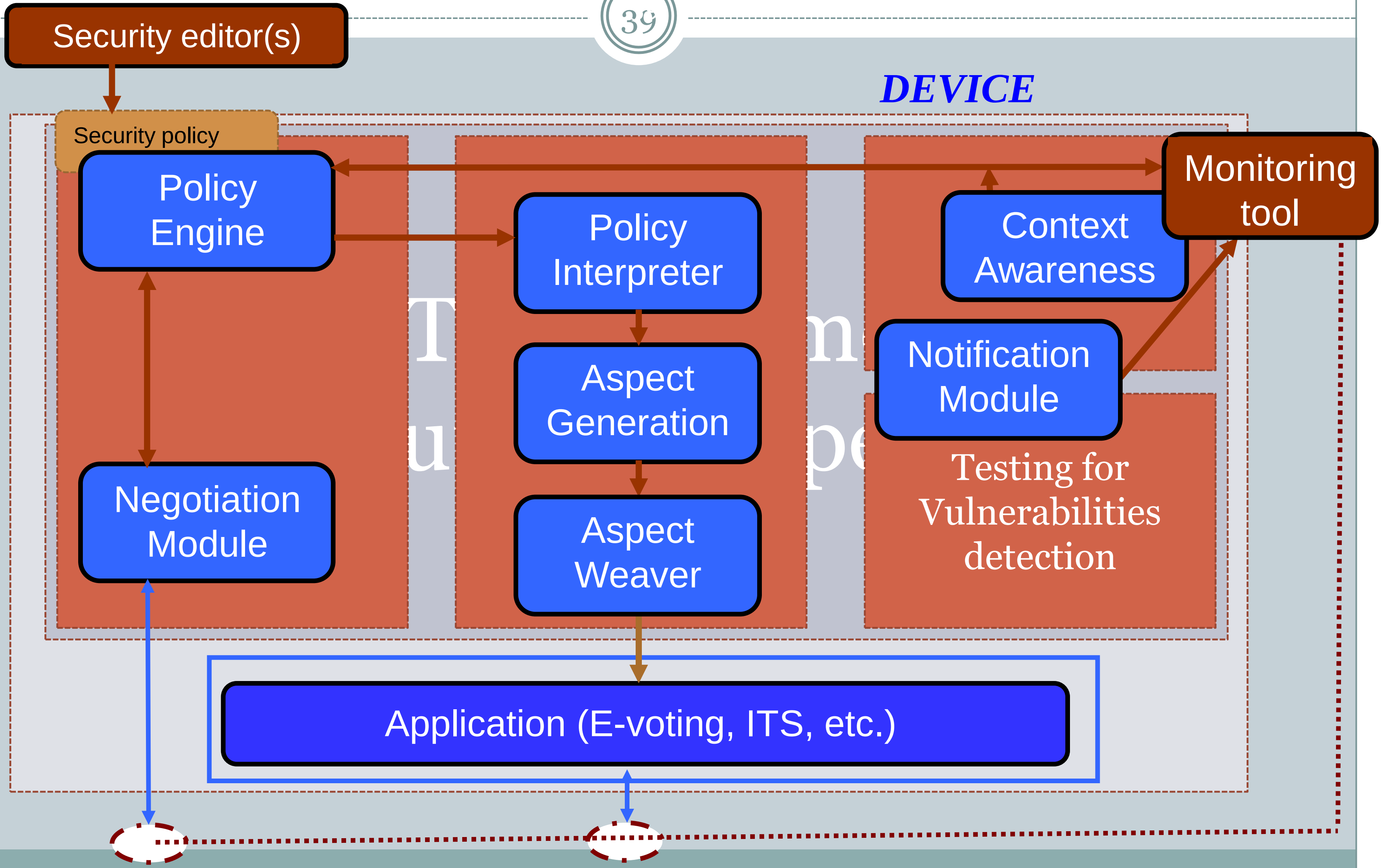
Aspect Weaver

Application + Context Monitoring

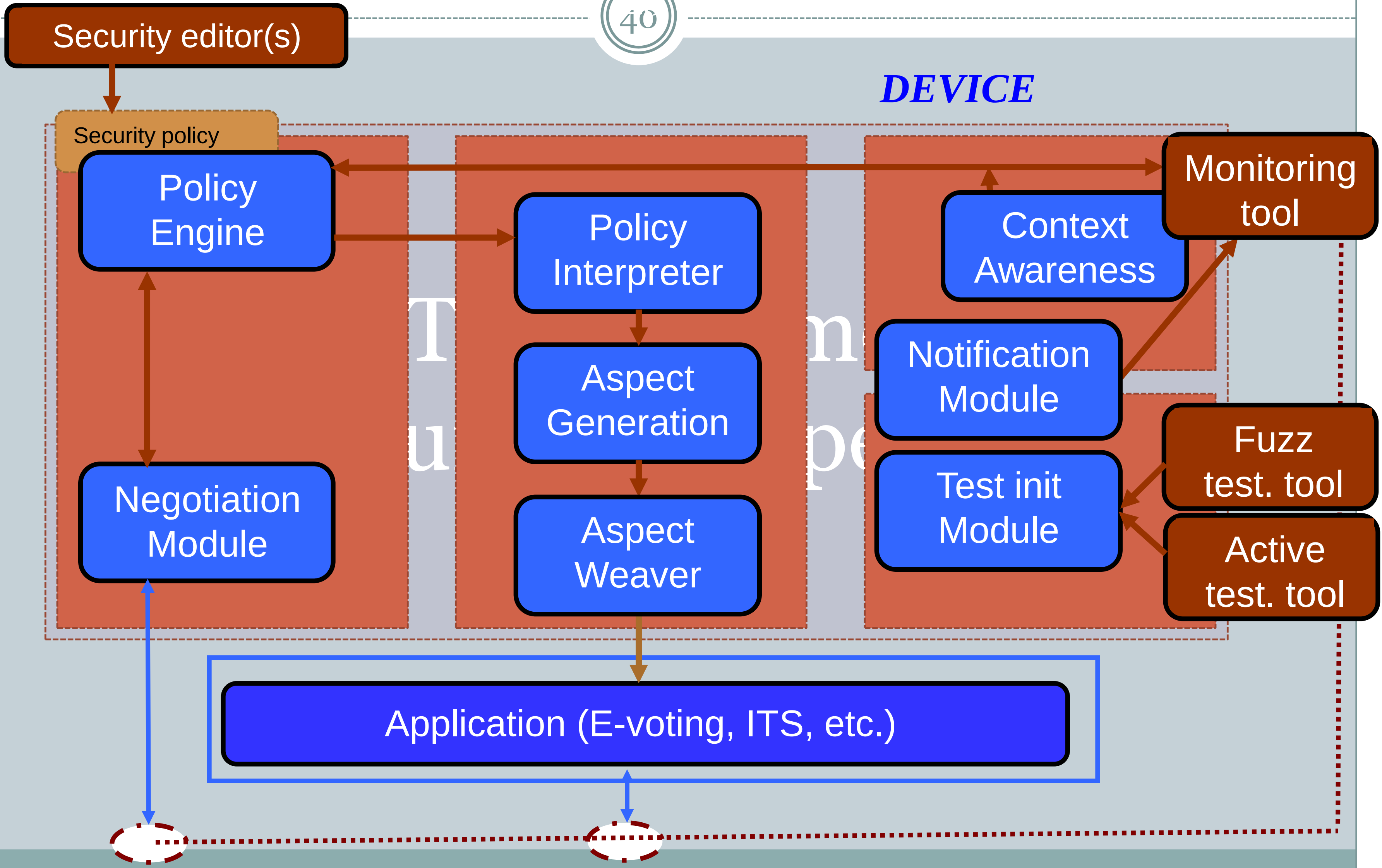
Testing for Vulnerabilities detection

Application (E-voting, ITS, etc.)

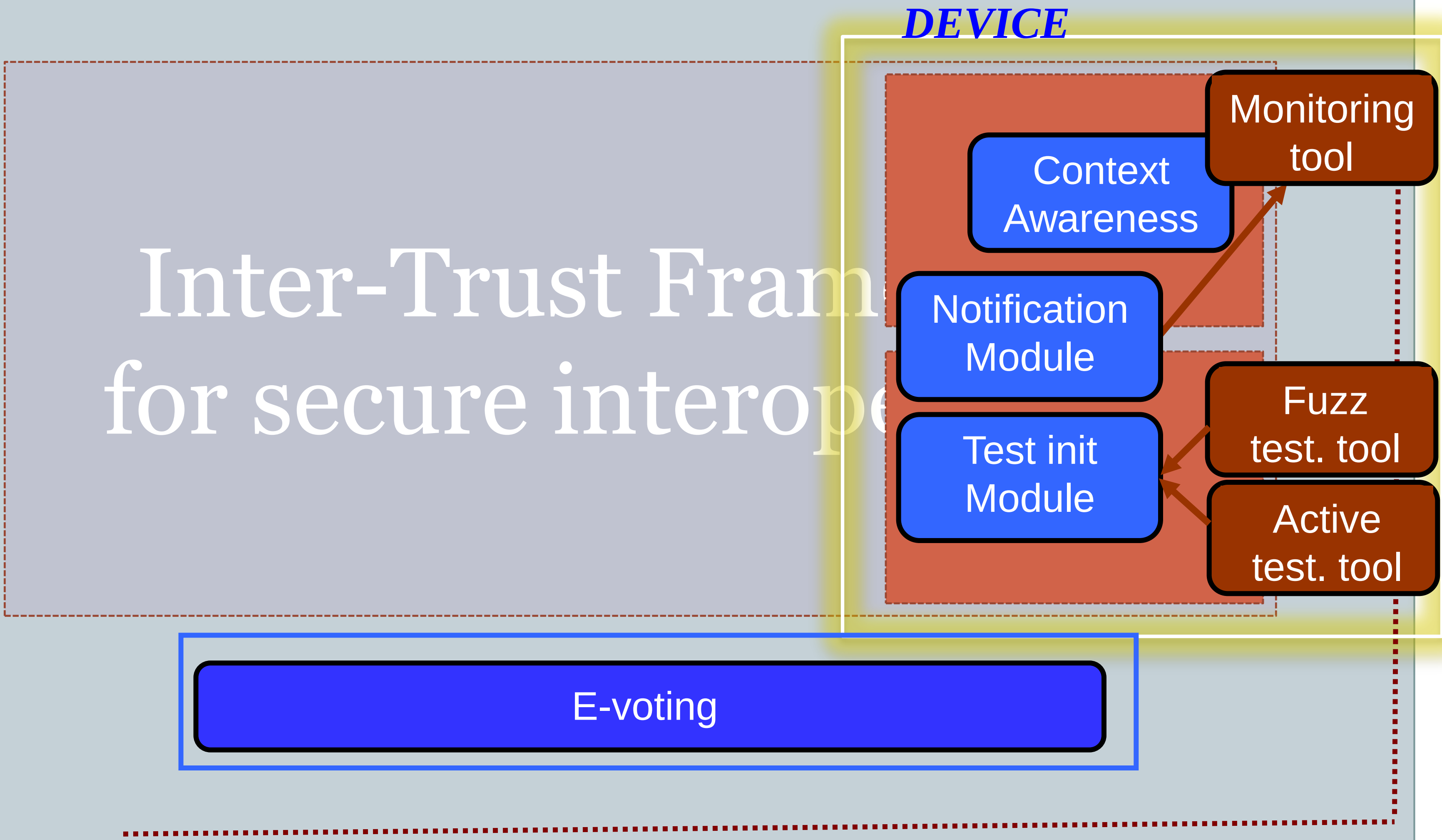
I-T Framework architecture



I-T Framework architecture



I-T Framework architecture

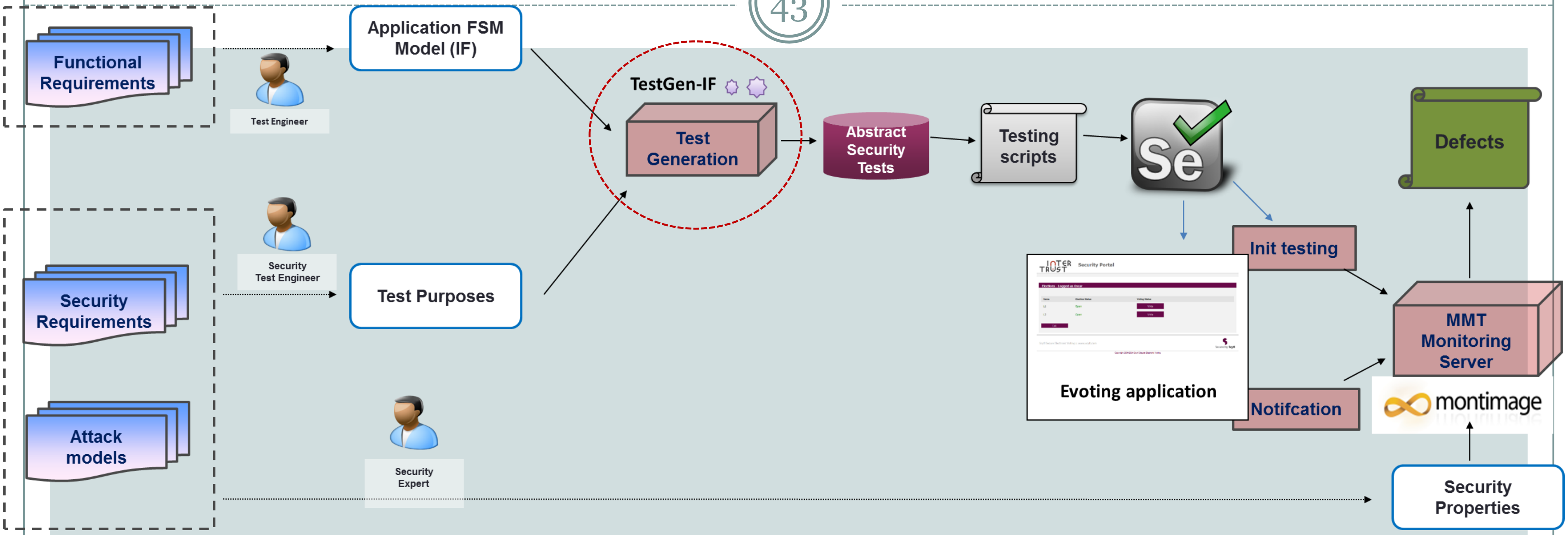


Motivation

- Why testing ? (testing phase)
  - Vulnerabilities can be introduced by AOP used in Inter-trust
    - Functional testing
    - Check the respect of weaved security policies (aspects)
    - Check the robustness of the target application
    - Detect vulnerabilities
    - Simulate attacks

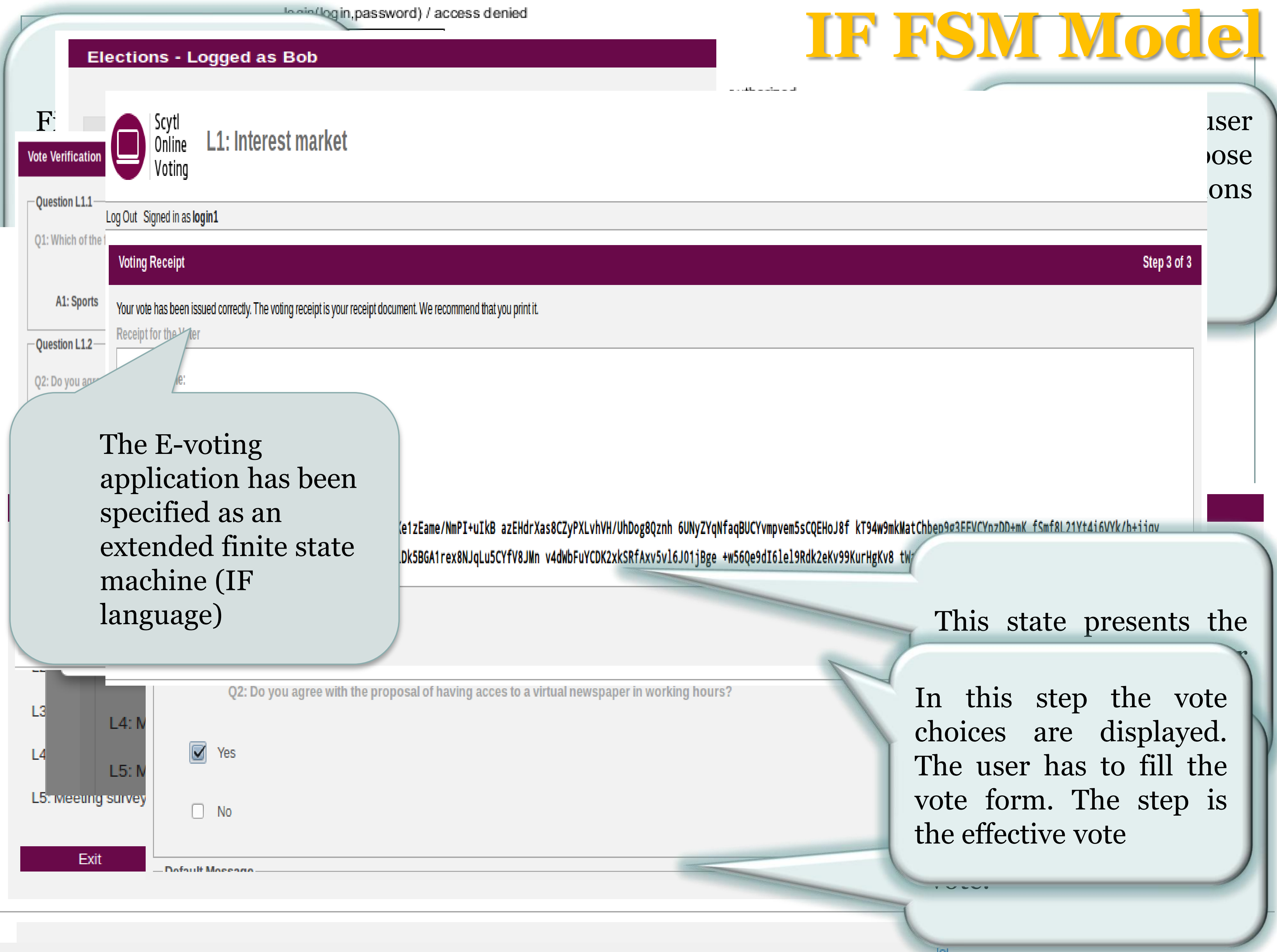
- Why monitoring ? (testing & operation phases)
  - Same as above
  - + detecting context changes (context awareness) at runtime

Model based testing: TestGen-IF



- Generation of tests from IF model and test purposes
  - Target: functional, security properties, attacks*
- Execution relying on Selenium (Web interface)
- Detecting failures using MMT

IF FSM Model



login(login,password) / access denied

Elections - Logged as Bob

Vote Verification

Scytl Online Voting

L1: Interest market

Log Out Signed in as **login1**

Voting Receipt

Step 3 of 3

Your vote has been issued correctly. The voting receipt is your receipt document. We recommend that you print it.

Receipt for the Voter

Question L1.1

Q1: Which of the

A1: Sports

Question L1.2

Q2: Do you agr

{e1zEame/NmPI+uIkB azEHdrXas8CZyPXLvhVH/UhDog8Qznh 6UNyZYqNfaqBUCYvmpvem5sCQEHoJ8f kT94w9mkMatChbep9g3EFVCYnzDD+mK fSmf8L21Yt4i6VYk/b+jiqv

Dk5BGA1rex8NJqLu5CYfV8JMn v4dWbFuYCDK2xkSRfAxv5vl6J01jBge +w56Qe9dI6lel9Rdk2eKv99KurHgKv8 tW

Q2: Do you agree with the proposal of having acces to a virtual newspaper in working hours?

☑ Yes

☐ No

L3

L4: M

L4

L5: M

L5: Meeting survey

Exit

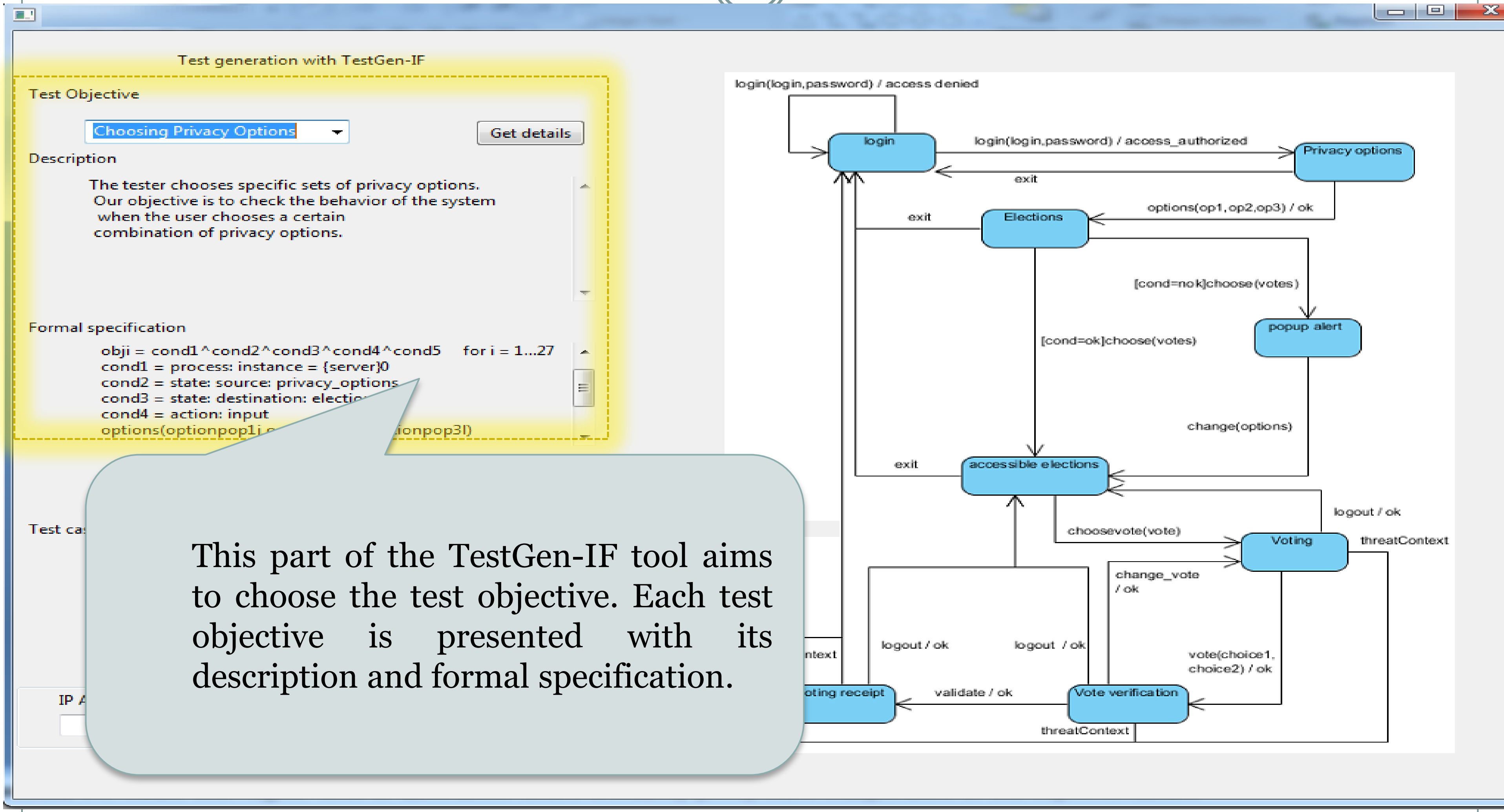
Default Message

The E-voting application has been specified as an extended finite state machine (IF language)

This state presents the

In this step the vote choices are displayed. The user has to fill the vote form. The step is the effective vote

Test Objective specification



Test generation with TestGen-IF

Test Objective

Choosing Privacy Options ▾ Get details

Description

The tester chooses specific sets of privacy options. Our objective is to check the behavior of the system when the user chooses a certain combination of privacy options.

Formal specification

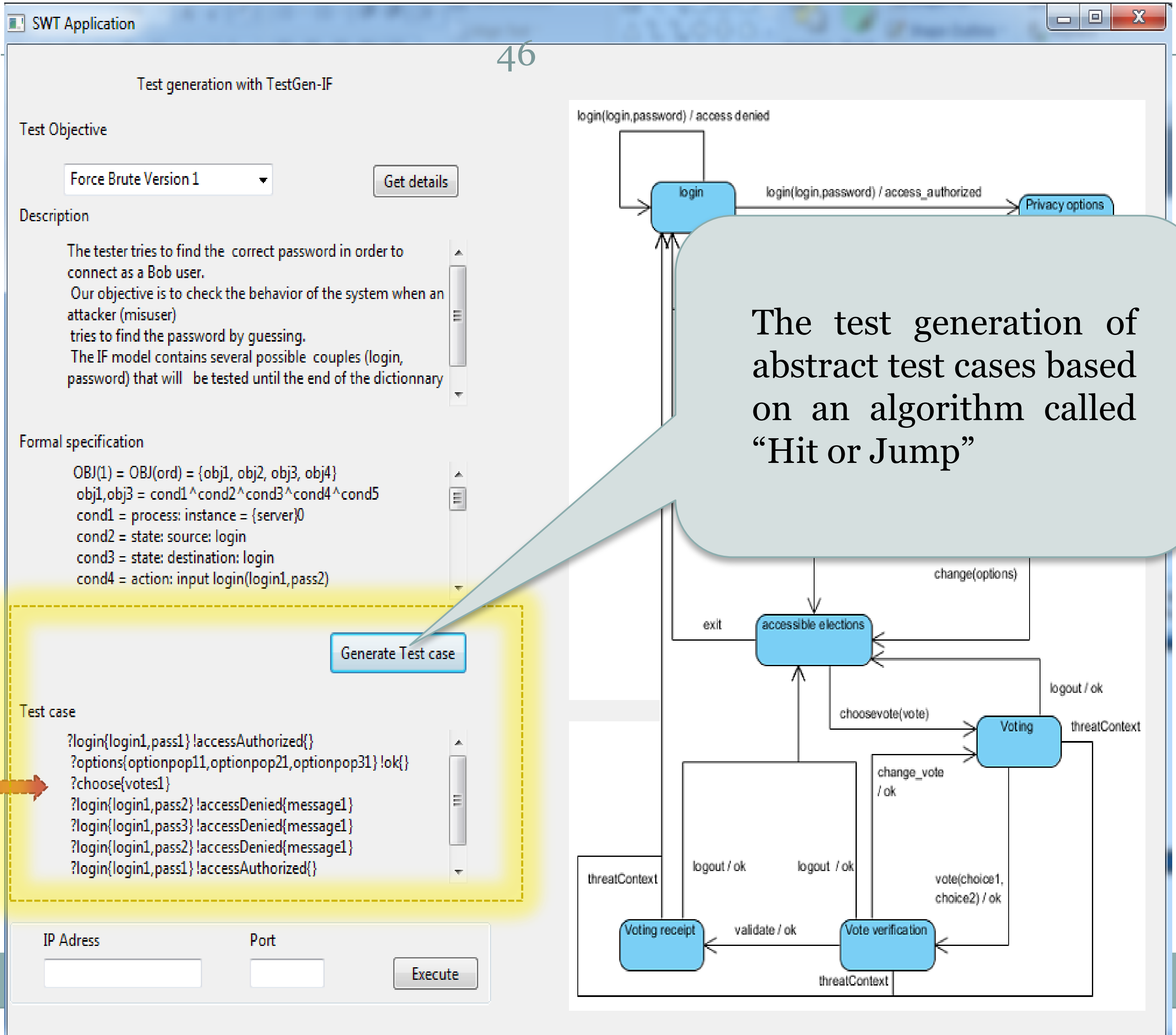
```
obji = cond1^cond2^cond3^cond4^cond5    for i = 1...27
cond1 = process: instance = {server}0
cond2 = state: source: privacy_options
cond3 = state: destination: electio
cond4 = action: input
options(optionpop1i                     ionpop3l)
```

Test ca

IP A

This part of the TestGen-IF tool aims to choose the test objective. Each test objective is presented with its description and formal specification.

Test case Generation



SWT Application

Test generation with TestGen-IF

Test Objective

Force Brute Version 1 ▾

Get details

Description

The tester tries to find the correct password in order to connect as a Bob user.
Our objective is to check the behavior of the system when an attacker (misuser)
tries to find the password by guessing.
The IF model contains several possible couples (login, password) that will   be tested until the end of the dictionnary

Formal specification

```
OBJ(1) = OBJ(ord) = {obj1, obj2, obj3, obj4}
 obj1,obj3 = cond1^cond2^cond3^cond4^cond5
 cond1 = process: instance = {server}0
 cond2 = state: source: login
 cond3 = state: destination: login
 cond4 = action: input login(login1,pass2)
```

Generate Test case

Test case

```
?login{login1,pass1} !accessAuthorized{}
 ?options{optionpop11,optionpop21,optionpop31} !ok{}
 ?choose{votes1}
 ?login{login1,pass2} !accessDenied{message1}
 ?login{login1,pass3} !accessDenied{message1}
 ?login{login1,pass2} !accessDenied{message1}
 ?login{login1,pass1} !accessAuthorized{}
```

IP Adress

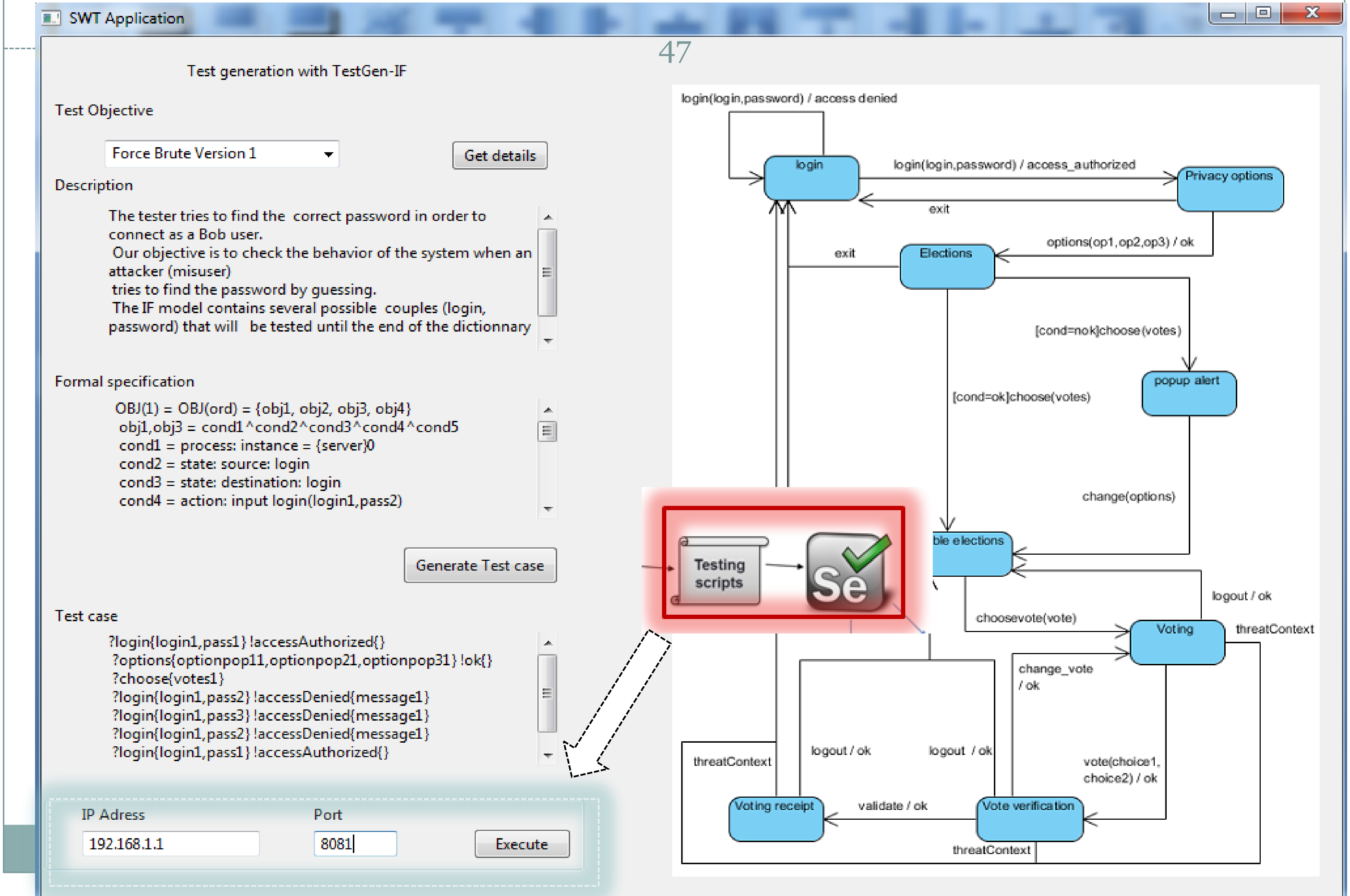
Port

Execute

The test generation of abstract test cases based on an algorithm called “Hit or Jump”

Abstract Security Tests

Automatic test execution



Test generation with TestGen-IF

Test Objective

Force Brute Version 1

Description

The tester tries to find the correct password in order to connect as a Bob user.
Our objective is to check the behavior of the system when an attacker (misuser)
tries to find the password by guessing.
The IF model contains several possible couples (login, password) that will be tested until the end of the dictionnary

Formal specification

```
OBJ(1) = OBJ(ord) = {obj1, obj2, obj3, obj4}
 obj1,obj3 = cond1^cond2^cond3^cond4^cond5
 cond1 = process: instance = {server}0
 cond2 = state: source: login
 cond3 = state: destination: login
 cond4 = action: input login(login1,pass2)
```

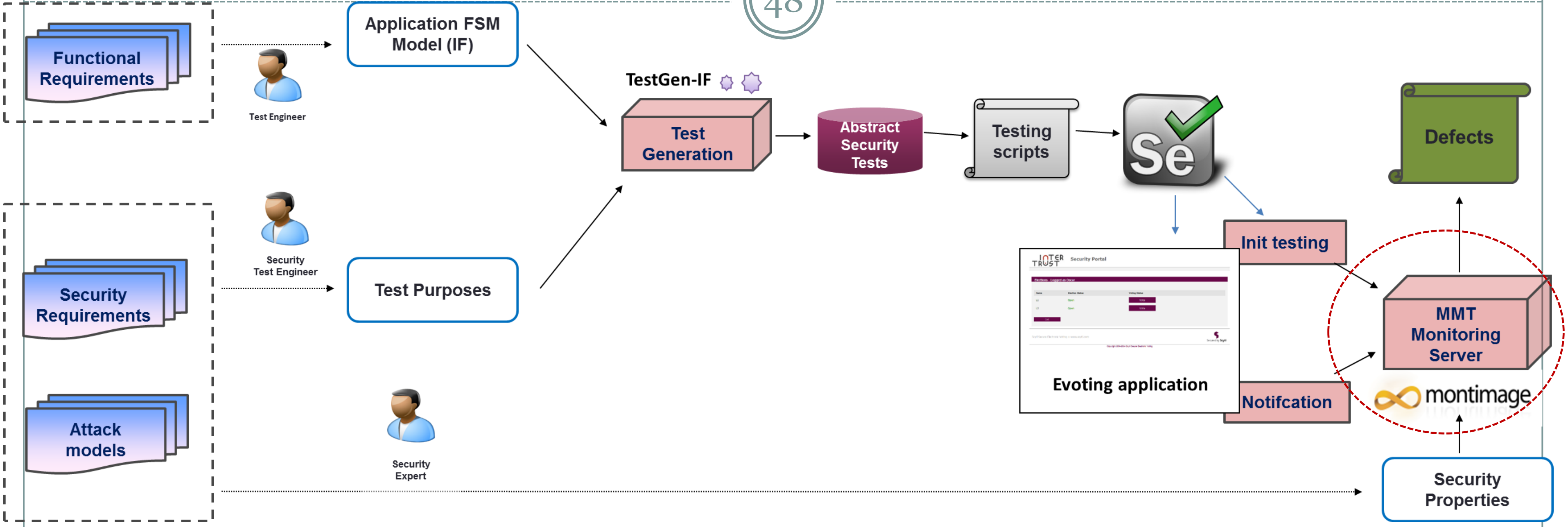
Test case

```
?login{login1,pass1} !accessAuthorized{}
 ?options{optionpop11,optionpop21,optionpop31} !ok{}
 ?choose{votes1}
 ?login{login1,pass2} !accessDenied{message1}
 ?login{login1,pass3} !accessDenied{message1}
 ?login{login1,pass2} !accessDenied{message1}
 ?login{login1,pass1} !accessAuthorized{}
```

IP Adress: 192.168.1.1

Port: 8081

Monitoring tool (MMT)



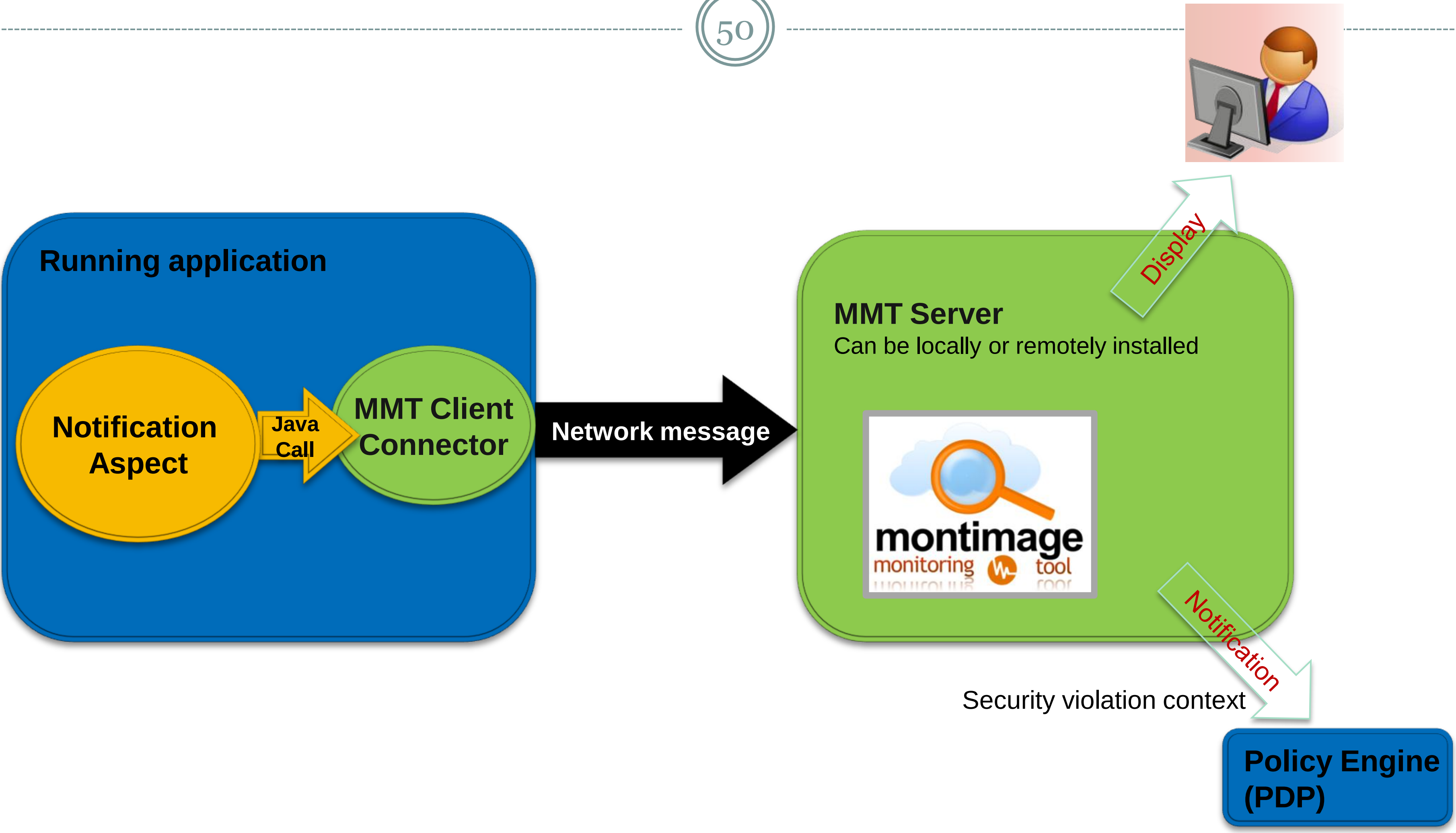
- Detecting failures using MMT
    - Events based detection
    - Properties as FSMs or as LTL properties

Monitoring

- 2 main usages:
  - During the testing phase to complement the testing tools and provide a verdict
  - During the operation phase to monitor security and application context

- Relies on data collected at different levels
  - Network (ex. CAM messages)
  - Application internal events (notification module)*
  - System status (CPU and memory usage)

Application internal events notification



Running application

Notification Aspect

Java Call

MMT Client Connector

Network message

MMT Server

Can be locally or remotely installed

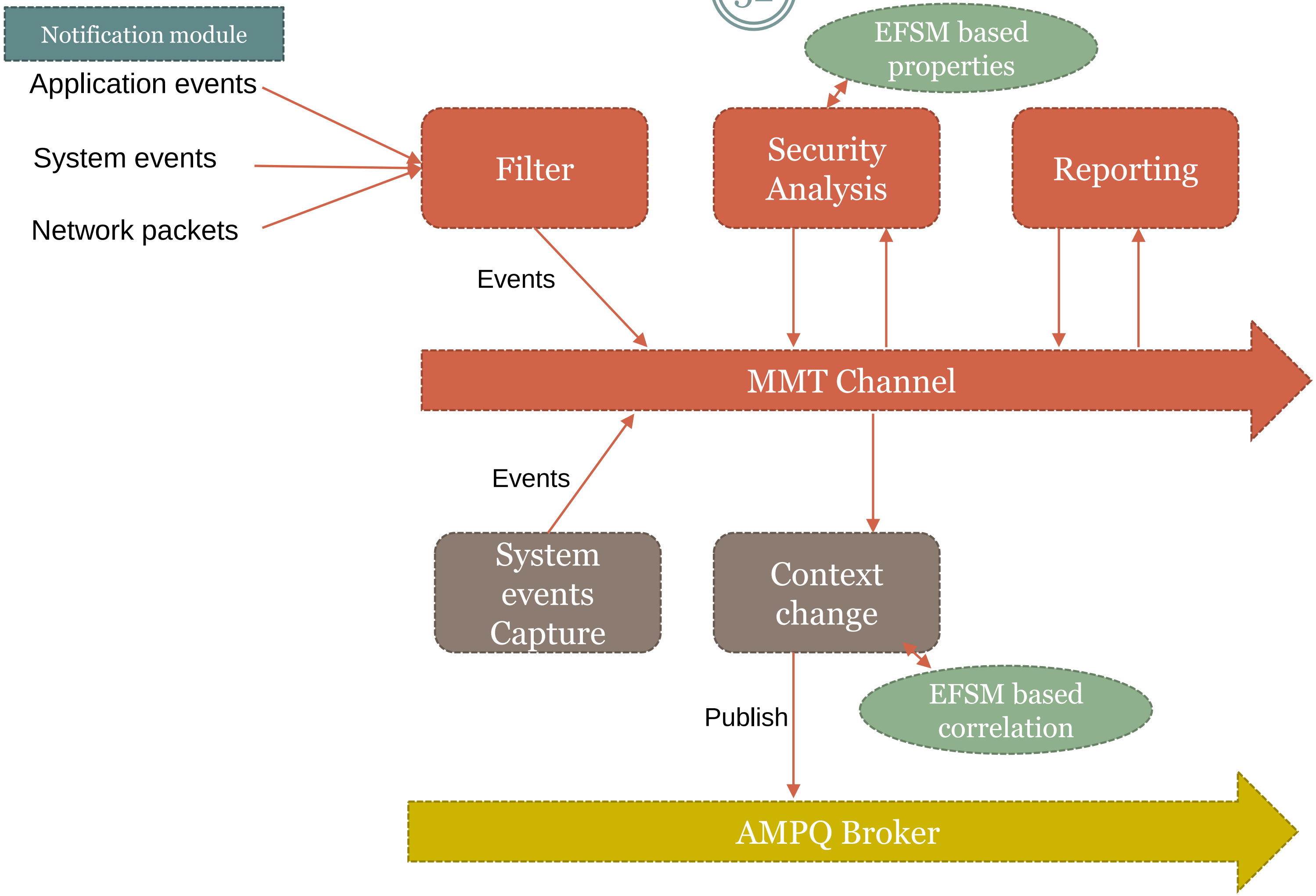
Display

Notification

Security violation context

Policy Engine (PDP)

MMT internal behavior



Notification module

Application events

System events

Network packets

Filter

Security Analysis

EFSM based properties

Reporting

Events

MMT Channel

Events

System events Capture

Context change

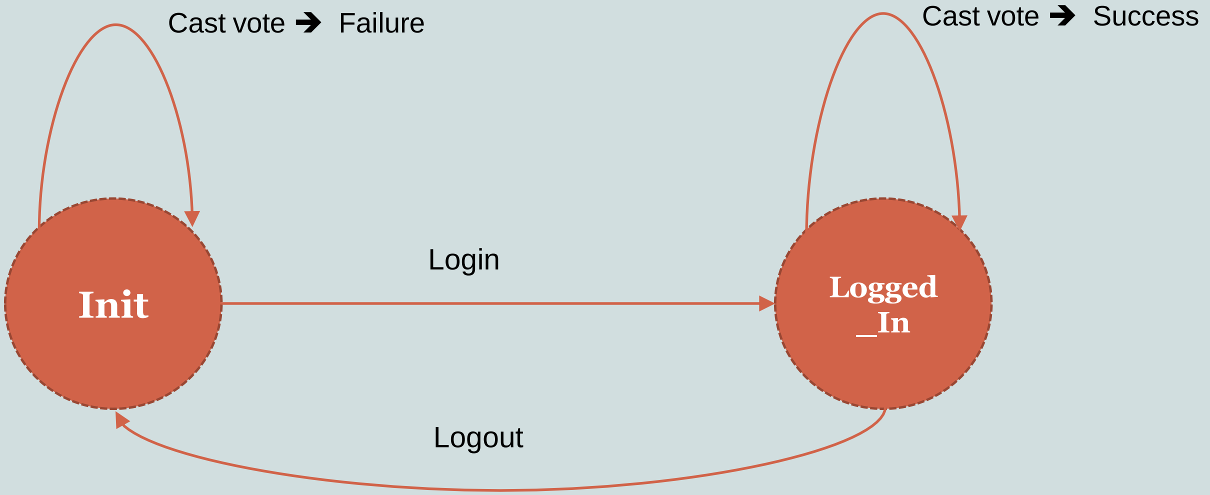
EFSM based correlation

Publish

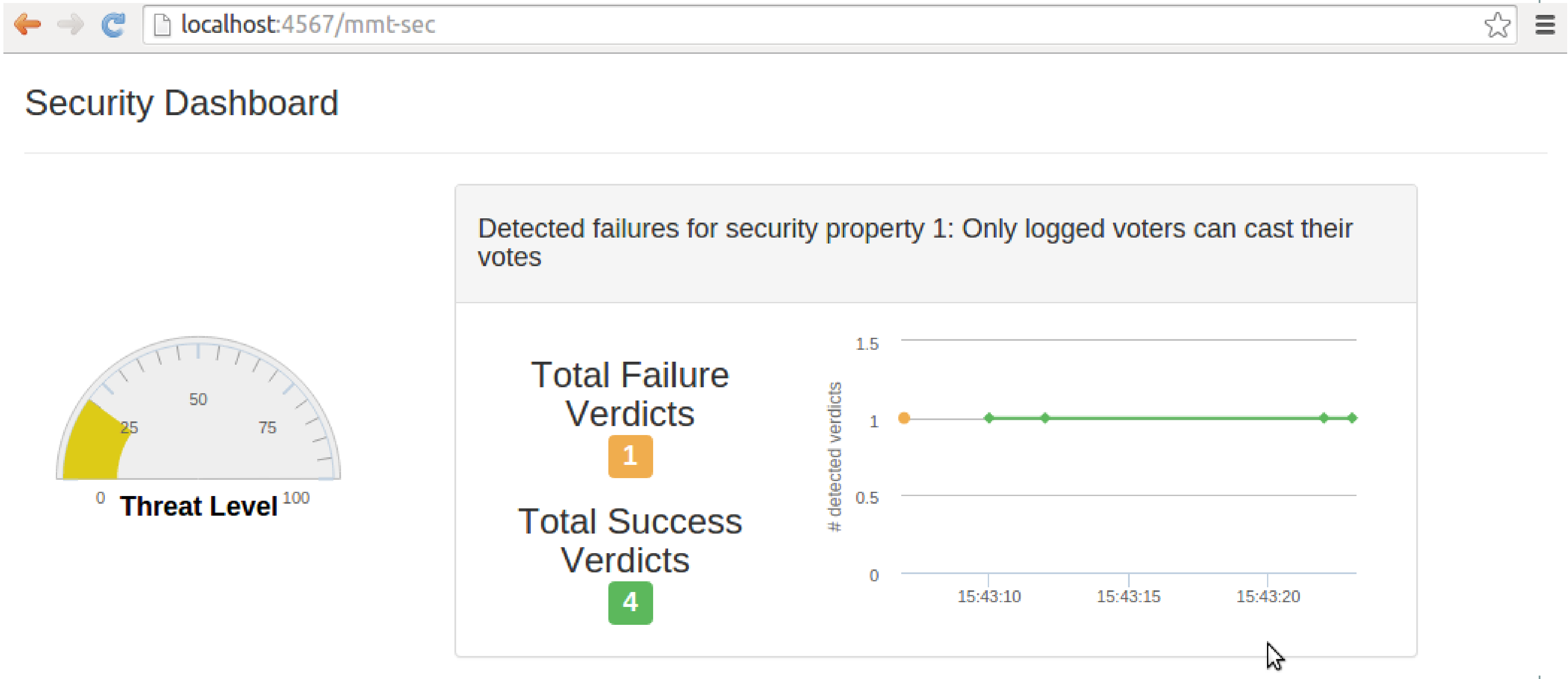
AMPQ Broker

Analysis and failure detection

- Evoting test case – Advanced authentication option
  - Example of property: Only authenticated voters can cast their votes



MMT analysis dashboard – Security property



localhost:4567/mmt-sec

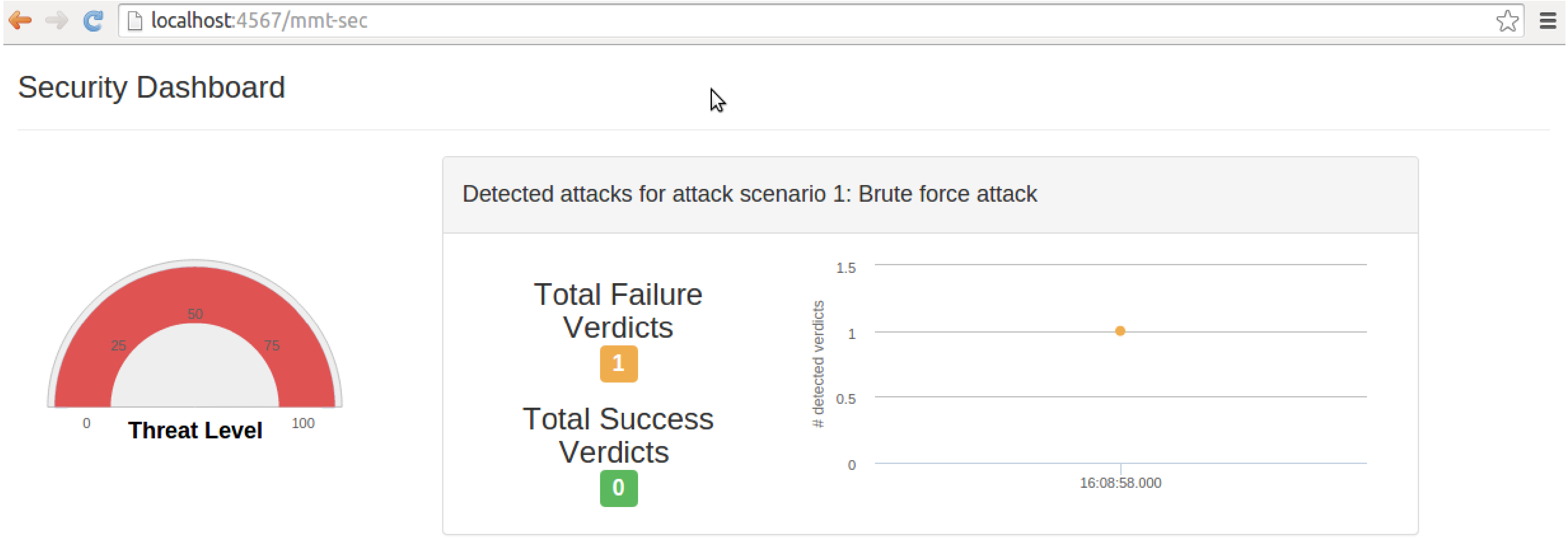
Security Dashboard

Detected failures for security property 1: Only logged voters can cast their votes

Total Failure Verdicts 1

Total Success Verdicts 4

MMT analysis dashboard – Attack detection



localhost:4567/mmt-sec

Security Dashboard

Detected attacks for attack scenario 1: Brute force attack

Total Failure Verdicts 1

Total Success Verdicts 0

Summary

- Model based test generation for security purposes (TestGen-IF)
- Correlation of data from different sources (Network, application, system)
- Detection of attacks and failures at runtime
  → reaction
- Brings dynamicity to system by adapting to different contexts

Selected 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.

Selected References

- 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, 99-113, Maria Emilia Cambronero, Manuel Nuñez María-Emilia Cambronero, Manuel Núñez: Formal Passive Testing of Service-Oriented Systems. IEEE SCC 2010IEEE 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: **Formal passive testing of timed systems: theory and tools.** Softw. Test., Verif. Reliab. 22 (6): 365-405 (2012)
- 16. Robert M. Hierons, Mercedes G. Merayo, Manuel Núñez: **Passive Testing with Asynchronous Communications.** FMOODS/FORTE 2013:

Selected References

- 17 Khalifa Toumi, Fabien Autrel, Ana R. Cavalli, Sammy Haddad: **ISER: A Platform for Security Interoperability of Multi-source Systems**. CRiSIS 2014: 230-238 , 2014
- 18 Mohamed H.E. Aouadi, Khalifa Toumi and Ana Cavalli, "**Testing Security Policies for Distributed Systems: Vehicular Networks as a Case Study**", International Journal of Computer Science Issues (IJCSI) Volume 11, Issue 5, September 2014
- 19 Toumi, K., Mallouli W., Montes de Oca E., Cavalli, A., Andrés, C., " **How to Evaluate Trust Using MMT**", NSS 2014, October 15-17, 2014, Xi'an, China.
- 20 Mohamed H. E. Aouadi, Khalifa Toumi and Ana R. Cavalli, “**On Modeling and Testing Security Properties of Vehicular Networks**”, IEEE SECTEST workshop, March 31 2014, Cleveland, USA