



usted Logic

# **EDEN : A formal framework for high-level security CC evaluations**

**Carolina Lavatelli  
Trusted Logic S.A.**

**e-Smart 2004  
Sophia Antipolis, France**

This work is carried on in the framework of the RNTL project EDEN by Trusted Logic S.A., Verimag, CEA-LIST, CEA-LETI and Axalto S. A.

# Overview

- The role of (semi-) formal methods in IT security
  - The CC approach.
- The challenge of the unified framework
  - TL FIT: customized UML for semiformal specification.
  - EDEN: formal specification language and methodology.
- Smart card example
  - From semiformal to formal.
- Work in progress

# (semi) formal methods in IT security the Common Criteria approach

## Functional Requirements

Define desired security behavior of the Target of Evaluation (TOE)

## Assurance Requirements

Basis for gaining confidence that the claimed security measures are effective and implemented correctly.

## Evaluation Assurance Levels (EAL)

Constructed using components of the assurance families.

EAL4 : methodically designed, tested and reviewed

EAL5 : **semiformally** designed and tested

EAL6 : **semiformally** verified, designed and tested

EAL7 : **formally** verified, designed and tested

# ADV: Development assurance requirements

The Security Functions of the TOE from Security Target to the implementation.

## Representation levels

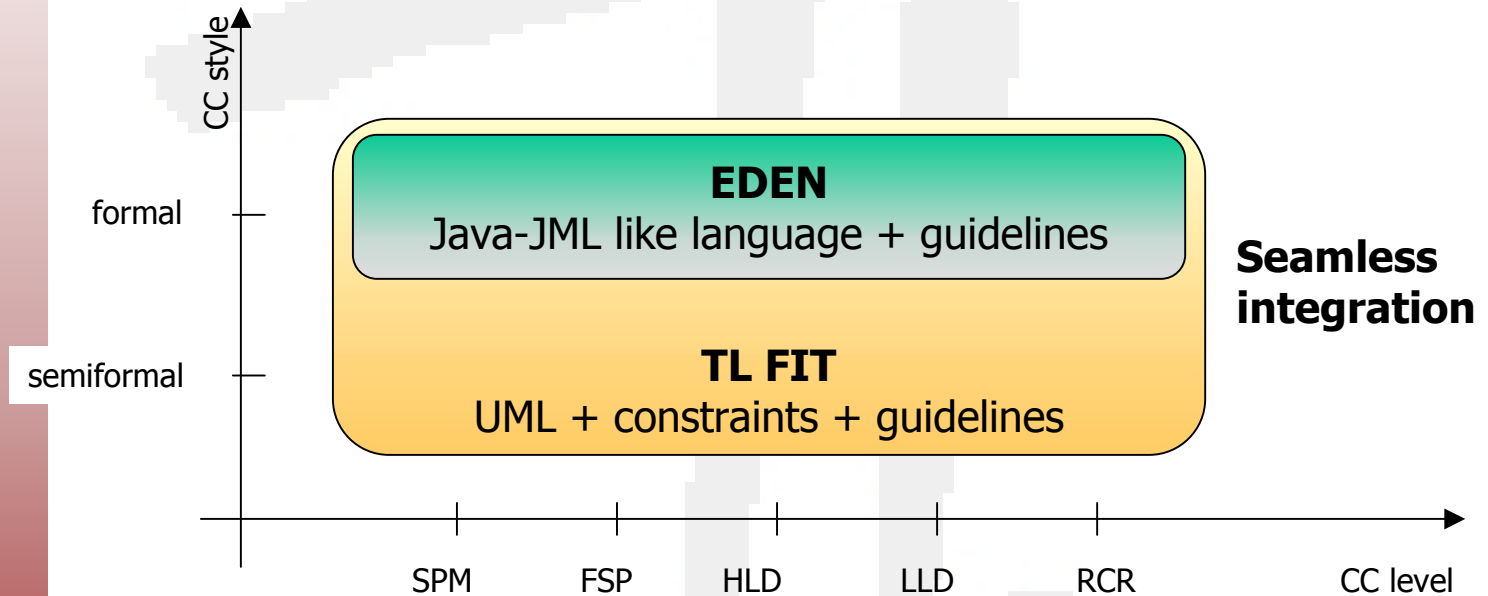
SPM: Security Policy Model  
FSP: Functional Specification  
HLD: High Level Design  
LLD: Low Level Design  
IMP: Implementation  
RCR: Correspondence Relation

## Description styles

Informal  
Semiformal  
Formal

**Challenge: Unified description and verification framework**

# Languages and methodologies of the unified framework



TL FIT runs on top of Rational Rose™ and Objecteering™ UML environments.

## CC semiformal specifications with TL FIT

What should be described ?

SPM : security rules

FSP : external interfaces

HLD : subsystems and interactions

RCR : correspondence relations

### Modular management of CC representation levels

Models.

Refinement units and links.

Access links and access rights.

### Each representation level modeled through a state machine.

Hierarchy of packages and classes.

Subtyping and inheritance.

Encapsulated operations and state variables.

### Behavior description

Text containers for invariants, pre and post-conditions, etc.

Sequence diagrams and call graphs.

Control graphs.

# Checking semiformal specifications with TL FIT

## Internal consistency

### Structural well-formedness of models

- Inter-model access links respect granted rights
- Called operations agree with access links and visibilities
- Control graphs are well-formed

## Refinement consistency

### Concrete models become more and more precise.

- Each refinement unit is refined in a separate model
- Each concrete model refining UR1 may use a component C of its abstract associated model provided the right has been granted.
- Each refinement step preserves the interface of the refinement unit (operations, types, access links)

## What changes in formal style

### The models

What kind of description for each representation level (SPM, FSP, HLD)?

### The languages

What formal languages for describing them ?

## TL FIT-EDEN formal models

### FSP, HLD : state machine approach

Executable prototypes of the TOE at different abstraction levels.

Extend/Redefine semiformal FSP-HLD correspondence relations.

### SPM : complementary approach

Properties (security rules) that the TOE must verify during execution.

# Eden Specification Language (ESL)

## Behavior

Java™-like instructions or logical specification of state changes.

## Observable execution events

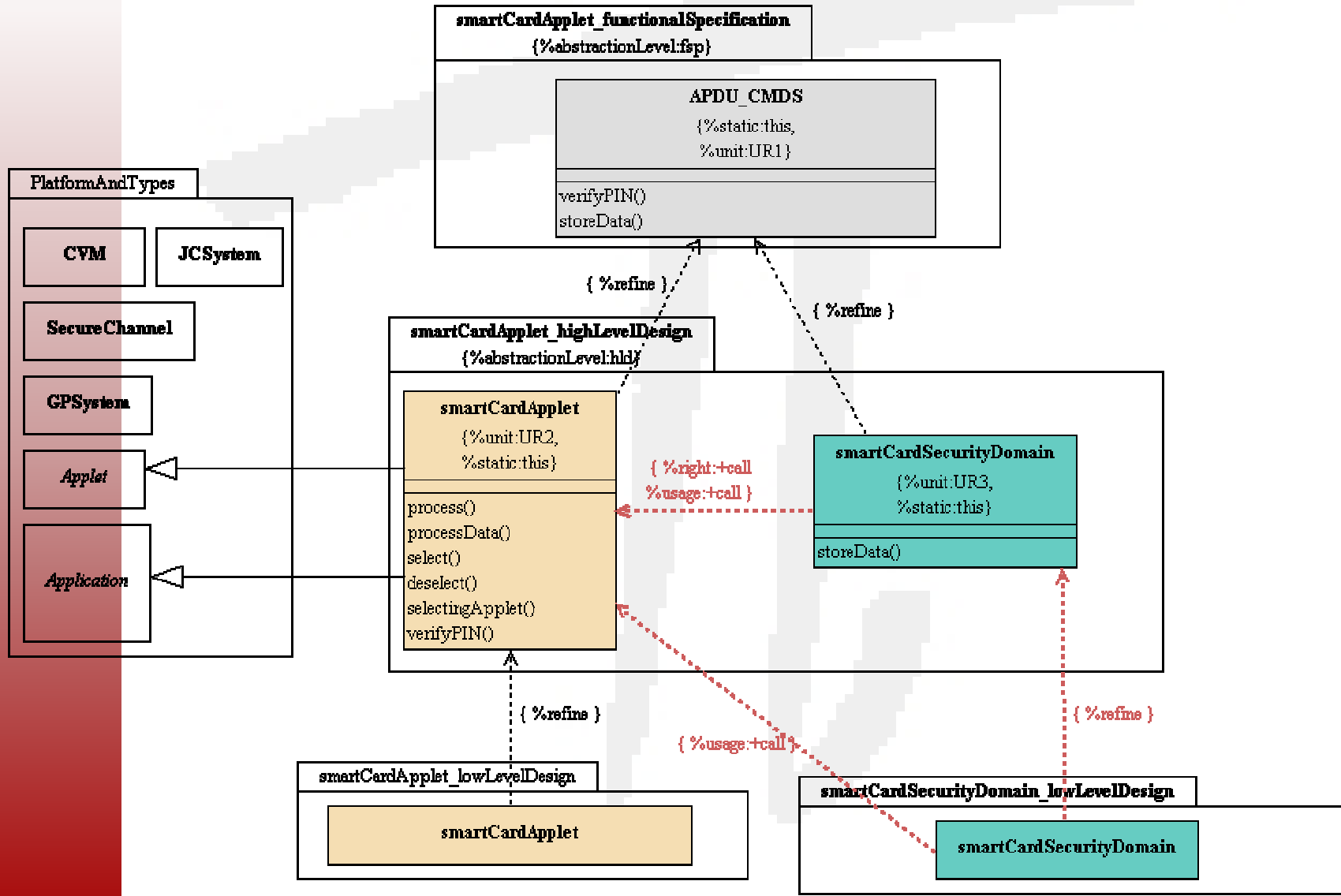
« CALL », « EXIT », « ABRUPT »

« READ », « WRITE »

## Logical conditions

JML-like first order « requires », « ensures », « invariant », etc.

# TL FIT - EDEN : modelization of a smart card applet





## SPM specification of VerifyPin

### Informal command description

The pin sent by the interface device is compared against the reference user PIN stored in the application/card.

The comparison may fail only a limited number of times. The application keeps track of the last validation result.

### SPM : properties of the authentication security function.

The remaining tries must reflect any attempt to compare the PINs, even when the whole process terminates abruptly.

Only the secure internal pin comparison function may read the value of the reference pin.



## Semiformal FSP specification of VerifyPin

### FSP : external interface behavior.

Normal termination of the command may take one of the three forms:

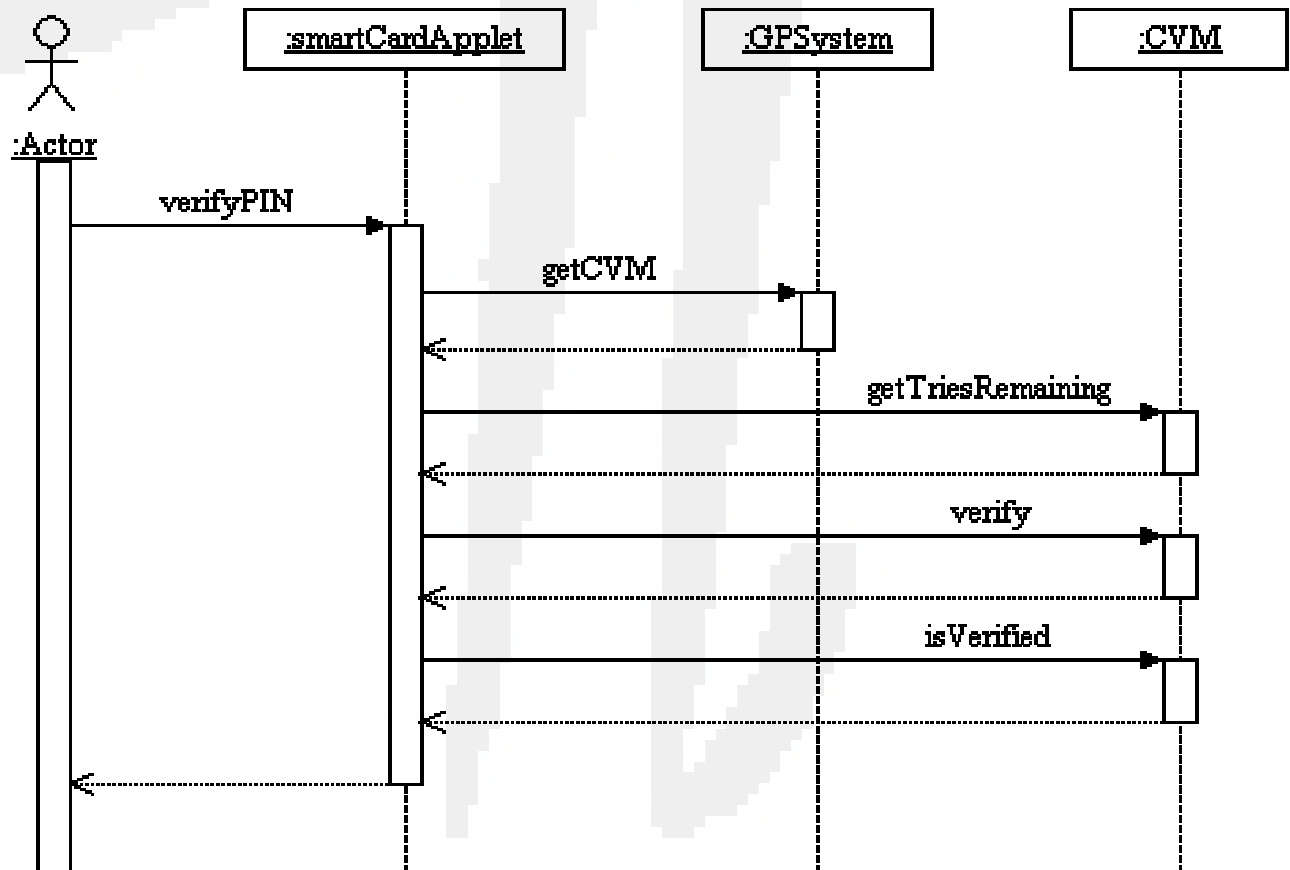
Status Word == 9000 if the input pin and the reference PIN are equal.

Status Word == 6000 if the input pin and the reference PIN are not equal.

Status Word == 6050 in any case, if there is no more remaining tries.

# Semiformal HLD specification of VerifyPin

HLD sequence diagram (interaction between subsystems)





## EDEN formal SPM properties of VerifyPin

### SPM: formal authentication policy

```
Abstract state : REF_PIN, remTries, MAX

INITIALLY(remTries == MAX)

INVARIANT(remTries =< MAX)

HISTORY_CONSTRAINT(REF_PIN == old(REF_PIN) && MAX == old(MAX))

class Authentication{

    void property_interruption{ // velocity checking
        WAIT(EVENT.ABRUPT(critical_compare));
        CHECK(OLD(remTries)==0 || remTries == OLD(remTries)-1)
    }

    void property_read{ // access policy
        WAIT(EVENT.READ(caller,REF_PIN,value));
        CHECK(caller == « critical_compare »)
    }
}
```



# EDEN formal FSP specification of VerifyPin

FSP : formal behavior of the external interface

System state : REF\_PIN, MAX, remTries, sw, isValid

```
void verifyPIN(apdu){ // specification of the apdu command

    REQUIRES(apdu.cla == .. && apdu.ins == ..);

    remTries = remTries-1; // interruption property
    if remTries >= 0
        {isValid = critical_compare(apdu.pin); // access to REF_PIN
            if isValid
                {sw = 9000; // success
                    remTries = MAX
                }
                else sw = 6000 // faillure
            }
        else sw= 6050 // faillure
    }
```

## Formal correspondence between FSP and SPM

<b>SPM state</b>	<b>FSP state</b>
REF_PIN	REF_PIN
remTries	remTries
MAX	MAX
	isValid
	sw

The FSP variables `isValid` and `sw` do not have any SPM counterpart.

More complex mappings may be needed.



## EDEN work in progress

### Verification tools

Check that any execution trace of the FSP specification verifies the SPM properties.

### Test generation tools

Generate tests to cover the FSP specification.

### Validation of the approach

Through industrial case studies  
Through CC evaluation laboratory