LEIRIOS TECHNOLOGIES

Mastering Test Generation from Smart Card Software Formal Models

Bruno LEGEARD
University of Franche-Comté & Leirios Technologies

CASSIS'04 - 12 March, 2004

Outline

- **Model-Based Testing**
  - ✓State of the practice in functional validation
  - ✓ Main challenges in Model-Based Testing
  - ✓ Leirios Test Generator: from Research to Industry
- **Test Generation Process for Smart Card Software**
  - ✓ Modeling for Testing
  - ✓ Test Generation Strategies
  - ✓ Executable Test Script Generation

The State of the Practice in Functional Validation

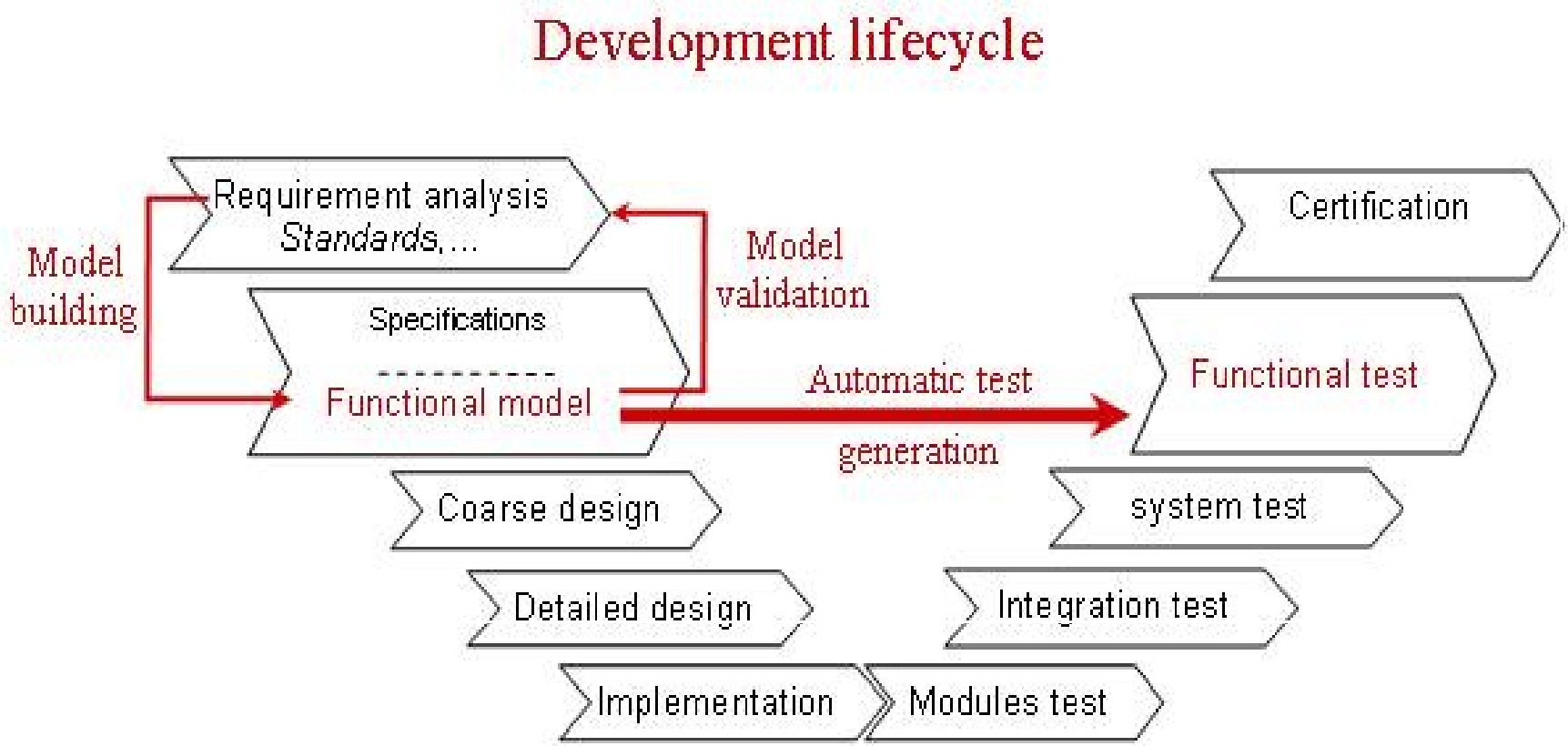
Verifying the compliance of the application to its specifications

How to :

- **Specify** tests?
- Master **number** of tests?
- Find the more **relevant**?
- Optimise **functional test coverage**?
- Manage **specification change**?
- Minimize **empiricism**?

Automated Test Generation from a Model of the Specifications

Model-Based Testing



Some Challenges in Automated Model-Based Testing

And how we address it?

- Mastering model complexity

➔ *"Partial formal models focused on a specific feature or component of the software under test, for the sole purpose of test generation, is feasible and provide good results in a realistic industrial setting"* Farchi-Hartman-Pinter - IBM Systems Journal 41:1 - 2002

- Symbolic animation of the model

➔ Using a customized set-oriented constraint solver able to compute the next system state

- Providing functional coverage of the system under test on the basis of the model

➔ Test each behavior (effect) of the system with boundary input values

- Controlling the test case explosion

➔The validation engineer controls the test generation using model coverage criteria

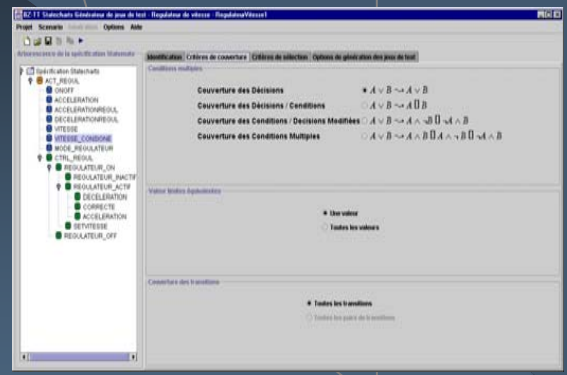
LEIRIOS Test Generator: From Research to Industry

- Research in Automated Model-Based Testing since 1996 at the University of Franche-Comté (LIFC – CNRS – INRIA):
  - Based on a symbolic animation of the formal model
  - Use cause-effect and boundary-testing strategies
- Industrial applications since 1999:
  - Smart Card area (GSM 11-11, Java Card transaction mechanism, Key Management 2G/3G) with Axalto,
  - Urban Systems (EMV Payment, Transport Ticket) with Parkeon and G. Carte Bancaire
  - Embedded Automotive Software (wiper controller, lightings) with PSA Peugeot Citroën
- LEIRIOS Company founded in 2003 as a Spin-off of the LIFC:
  - Currently 13 R&D Engineers develop the LEIRIOS Test Generator – LTG – tool-set.

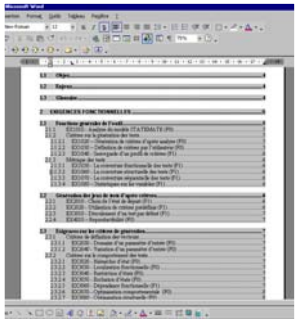
LEIRIOS Test Generator: Test Generation Principles

- Automated Boundary-Testing
  - Test each expected behavior of the system
  - Test boundary states and input values
- Generates Nominal and Robustness Test Cases
- Generates both Test Cases **(with expected results)** and Test Drivers ➔ Automated diagnosis assignment

- Test Generation Driving Criteria:
  - Model coverage criteria
  - Selection criteria
  - Specification evolution



Test Generation Process



Functional Requirements

Functional Modeling

Functional Model (B, Statecharts, UML/OCL)

Model Validation

LTG Model Animator

LTG Test Case Generator

Test Generation Criteria

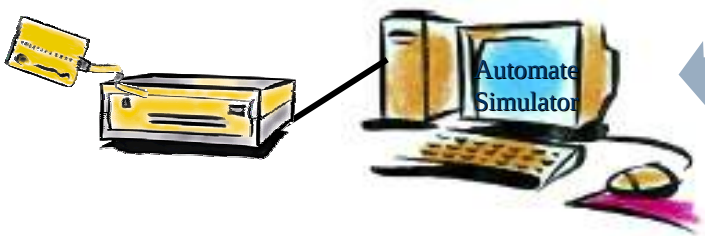
Test Case Generation

Test Cases

LTG Test Driver Generator

Test Driver Schemas

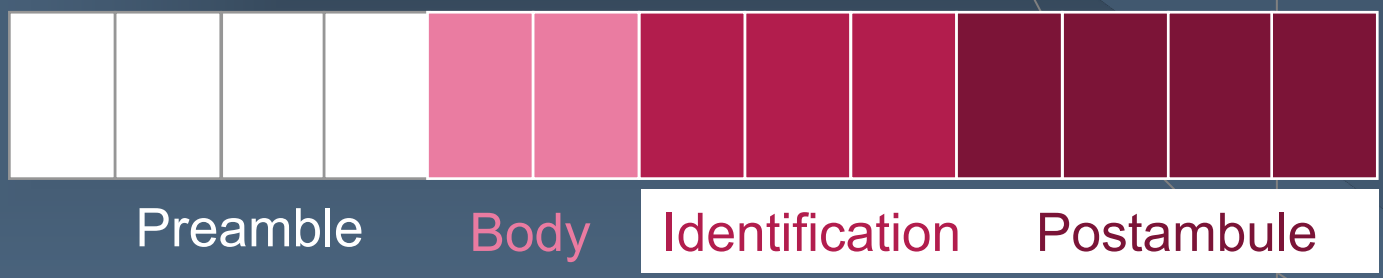
Test Driver Generation



Test Execution Environment

Definition of a test case

- A test case is a sequence of invocations on the system under test. It is divided in four parts:
  - Preamble: sequence of operations to reach the state to test
  - Body: invocation of tested effect
  - Identification: invocation of read-only operation to improve the expected results
  - Postamble: Return path to the initial state



Modeling for Testing

- LTG takes as input:
  - B abstract machines
  - UML (class and state diagrams with OCL expressions)
  - Statecharts Statemate
- For Smart Card OS and applications:
  - The model is developed using B or UML/OCL
  - The abstraction level and scope of the model depend of the test objectives
  - The model describes the functional behavior of the system under test

B Modeling for Test Generation

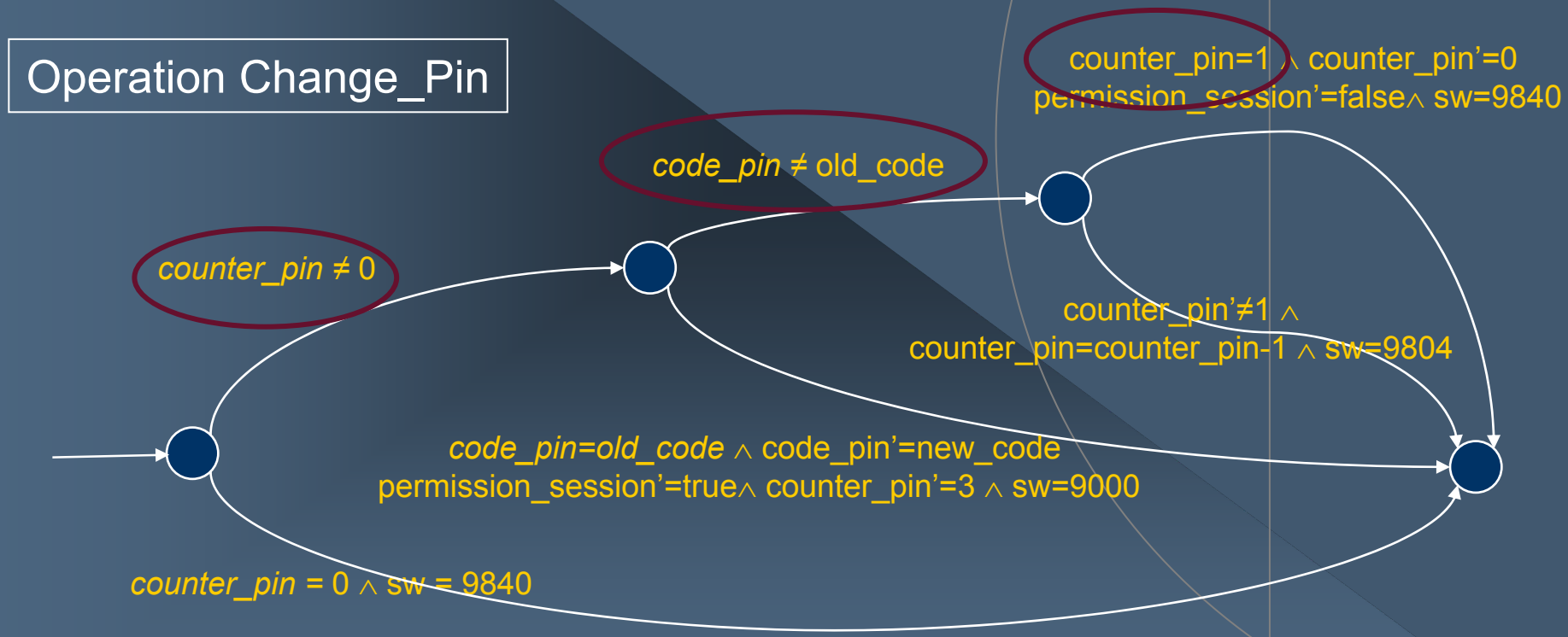
- The B abstract machine is used as a Pseudo-Code
- Control-oriented and Data-Oriented
- Good abstraction level

➔ Suitable for Smart Card APDU and API modeling

```
sw ← CHANGE_Pin(old_code, new_code) =
    PRE
    old_code ∈ Nat ∧ new_code ∈ Nat
    THEN
        IF counter_pin = 0
            THEN
            sw := 9840
            ELSE
               IF code_pin = old_code
                  THEN
                  code_pin := new_code          ||
                  counter_pin := 3              ||
                  permission_session := true    ||
                  sw := 9000
                  ELSE IF counter_pin =1
                           THEN
                           counter_pin := 0                    ||
                           permission_session := false         ||
                           sw := 9840
                           ELSE
                           counter_pin := counter_pin – 1      ||
                           sw := 9804
    END END END END END ;
```

Test Generation Method

Test all effects: Invoke each execution path for each operation



Effect Predicates == *counter_pin* ≠ 0 ∧ *code_pin* ≠ old_code ∧ counter_pin=1

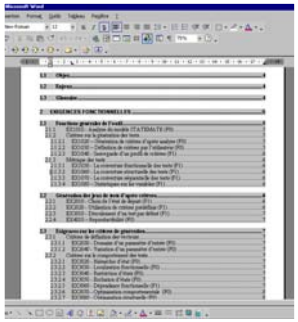
Test Generation Method - Model Coverage Criteria

- Multiple conditions in the decisions criteria
  - All the Decisions (DC)
  - All the Decisions/Conditions (D/CC)
  - All the Modified Conditions / Decision (MC/DC)
  - All Multiple Conditions (MCC)
- Symmetric value coverage
  - One value
  - Several values
  - All values
- Transition coverage
  - All-Transitions
  - All-Transition-Pairs

Test Generation Algorithm

- Step1 – Model partitioning
  - ➔ effect predicates
- Step2 (optional) – Boundary computation
  - ➔ boundary goals
- Step3 – Preamble computation
  - ➔ using symbolic animation and best-first search
- Step4 – Compute body and then identification
  - ➔ invoke tested effect predicates (or pair of tested operations)
- Step5 – Postamble computation

Test Generation Process



Functional Requirements

Functional Modeling

Functional Model (B, Statecharts, UML/OCL)

Model Validation

LTG Model Animator

LTG Test Case Generator

Test Generation Criteria

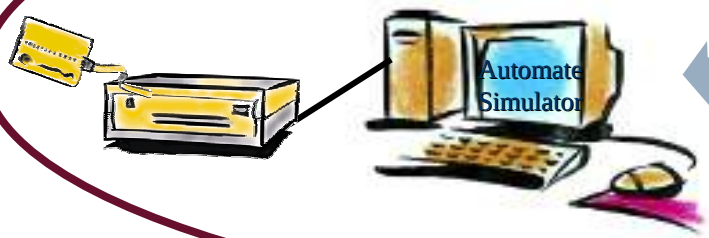
Test Case Generation

Test Cases

LTG Test Driver Generator

Test Driver Schemas

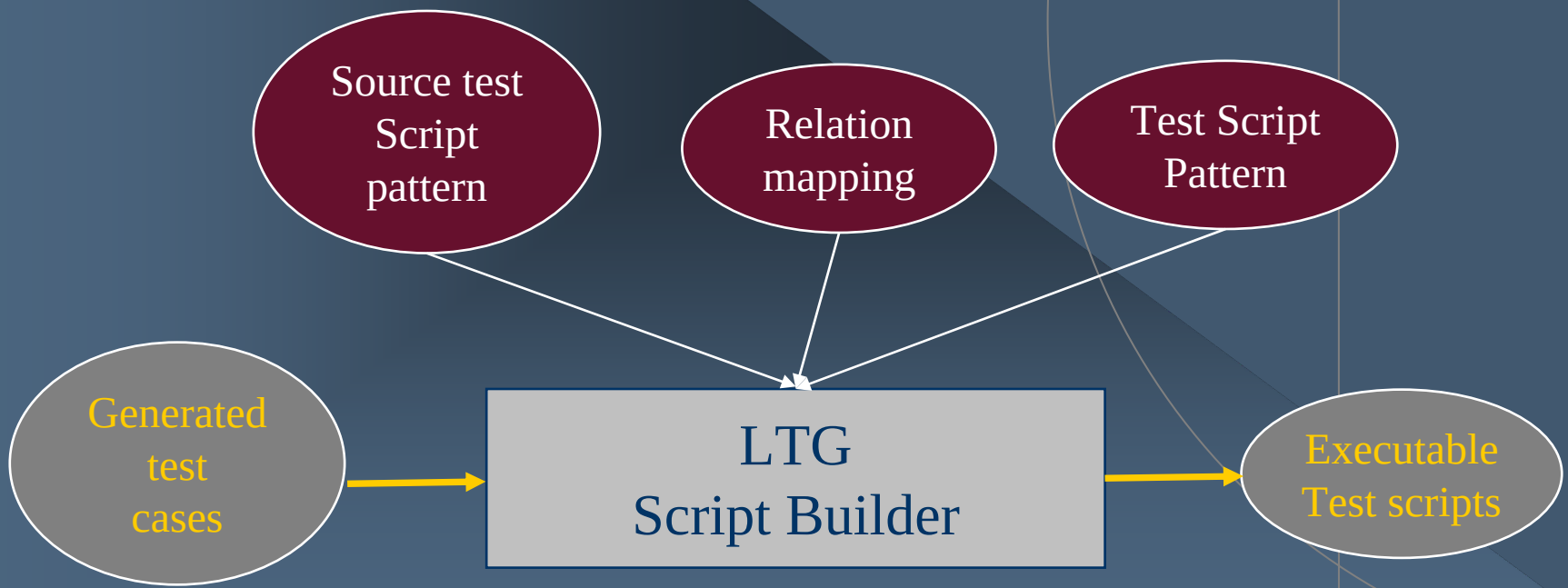
Test Driver Generation



Test Execution Environment

Executable Test Script Computation

- Use a test script pattern and a relation mapping to relate the formal (abstract) model names and the (concrete) implementation names
- Use of an observation table to link observation procedures with state variables
- Automates the verdict assignment – test pass/fail



Summary

- The B abstract machine notation is used as a pseudo-code to model the data and the behavior of Smart Card software
- Test cases are generated on the basis of cause-effect and boundary-testing strategies
- The test engineer controls the test case explosion using model coverage criteria
- Symbolic animation using constraint propagation helps to master scalability