

PRAGMAlist

An integrated tool for modeling and optimized test generation
driven by ✓ coverage and ✓ properties

Automatic test generation based on functional coverage

Emmanuel Gaudin
PragmaDev

PragmaDev

- French SME,
- Created in 2001 by 2 two experts in modelling tools and languages
- Since creation dedicated to the development of a modelling tool for the development of **Event driven software**.

Aero/Defence



Automotive



RENAULT

Telecoms



Korea Electronics Technology Institute

Semi-conductor



TOSHIBA

MITSUMI

Several Collaborative Projects with big accounts

Alcatel·Lucent 




Focus on Model Checking

Started in 2005
finished in 2009

THALES



Focus on property verification

Started in 2012
finished in 2014

list

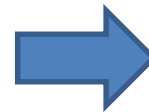


PRAGMAlist
Focus on Model Based Testing

Started in 2013

Requirements for a good modelling language

- The abstract model must be platform independent, as its name states.
- The abstract model must be translatable to an execution platform.
- For that purpose, the abstract model is based on a virtual machine offering:
 - Some **basic services**.
 - An execution **semantic**.



Key features for Model Based Testing capabilities

Verify the model

Model Simulator is similar to a graphical *debugger* for fully formal models:

- Breakpoints
- Step
- View internals
- Graphical traces

The screenshot displays the SDL simulator interface with several windows:

- Process information:** A table showing the state of various processes.

| Name | Pid | Sig | SDL state |
|--------------|-----|-----|--------------|
| Layer3 | 8 | 1 | sending |
| Layer3 | 6 | 0 | disconnected |
| Layer3 | 7 | 0 | connected |
| L3Dispatcher | 4 | 0 | idle |
| Layer2 | 5 | 1 | connected |
| RTDS_Env | 2 | 2 | done |
| L3API | 3 | 0 | idle |
| Application | 1 | 1 | Displaying |
- Timer info:** A table showing system time and time left for various processes.

| Pid | Name | Time left |
|-----|----------|-----------|
| 5 | tAck | 10 |
| 8 | tAck | 24 |
| 1 | tDisplay | 250 |
- Watch window:** Shows local variables for the current process, including SELF, senderPid, and i.
- RTDS - Diagram "CoDec" (modified):** A state transition diagram showing states like idle, disReq, conReq, and Connecting. A breakpoint is set on the transition to the state where stringTable(0) = '36'.
- Debugger console:** Shows the execution of testcases and the current state of the simulator (SDL).
- MSC Tracer:** A sequence diagram showing the interaction between local, central, and tt_registerUser processes over time.

Model coverage

- Graphical model coverage analysis
- Merge feature



The model is :

- Correct
- Fully covered

| Agent/symbol | Hits |
|--------------------|-------|
| Phone | 0 - 6 |
| pCentral | 0 - 6 |
| pLocal | 0 - 5 |
| - | 5 |
| Idle | 5 |
| Connected | 0 - 2 |
| sCnxReq | 0 |
| sBusy TO SENDER | 0 |
| - | 0 |
| sDisReq | 2 |
| sDisConf TO SEND | 2 |
| Idle | 2 |
| sHangUp | 2 |
| sDisReq TO remot. | 2 |
| Disconnecting | 2 |
| Connecting | 0 - 2 |
| sBusy | 0 |
| sBusy VIA cEnvLo.. | 0 |
| Idle | 0 |

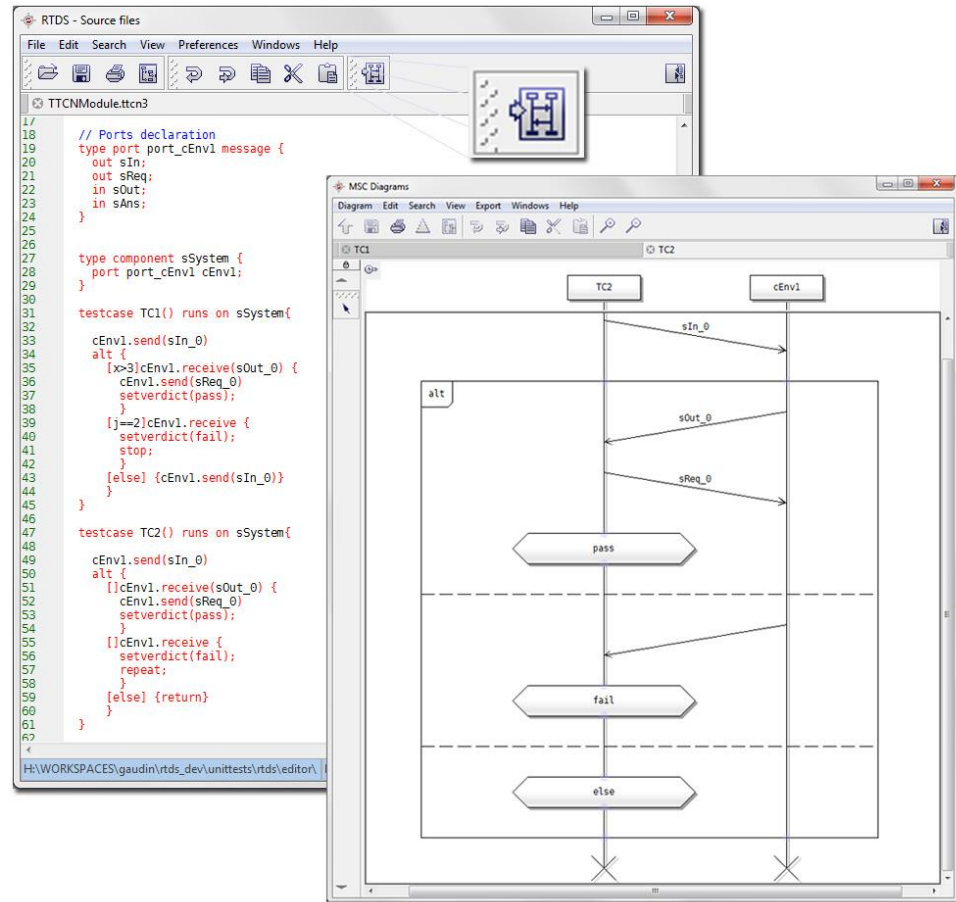
Requirements for a good testing language

- Relies on the same basic services:
 - Messages
 - Procedures
 - Timers
 - Parallel execution
- TTCN-3 international standard:
 - Data types definitions or ASN.1,
 - Templates definitions,
 - Test cases,
 - Verdict,
 - Execution control.

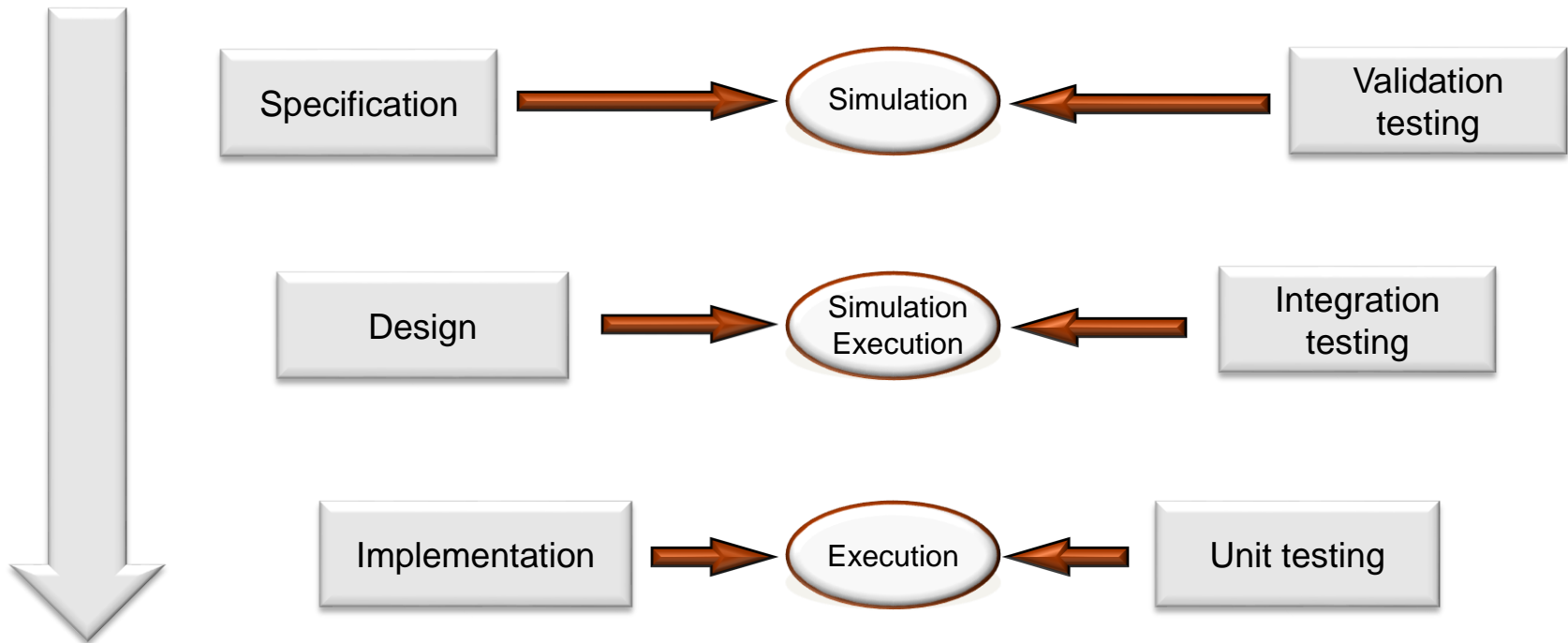


TTCN-3 support

- Textual editor
- TTCN-3 to MSC generation
- Simulator including a Test manager
- C++ code generator
- MSC to TTCN-3 generation
- TTCN-3 generation from a property on the model (Verimag)
- TTCN-3 generation based on model coverage (CEA List)



Same level of abstraction

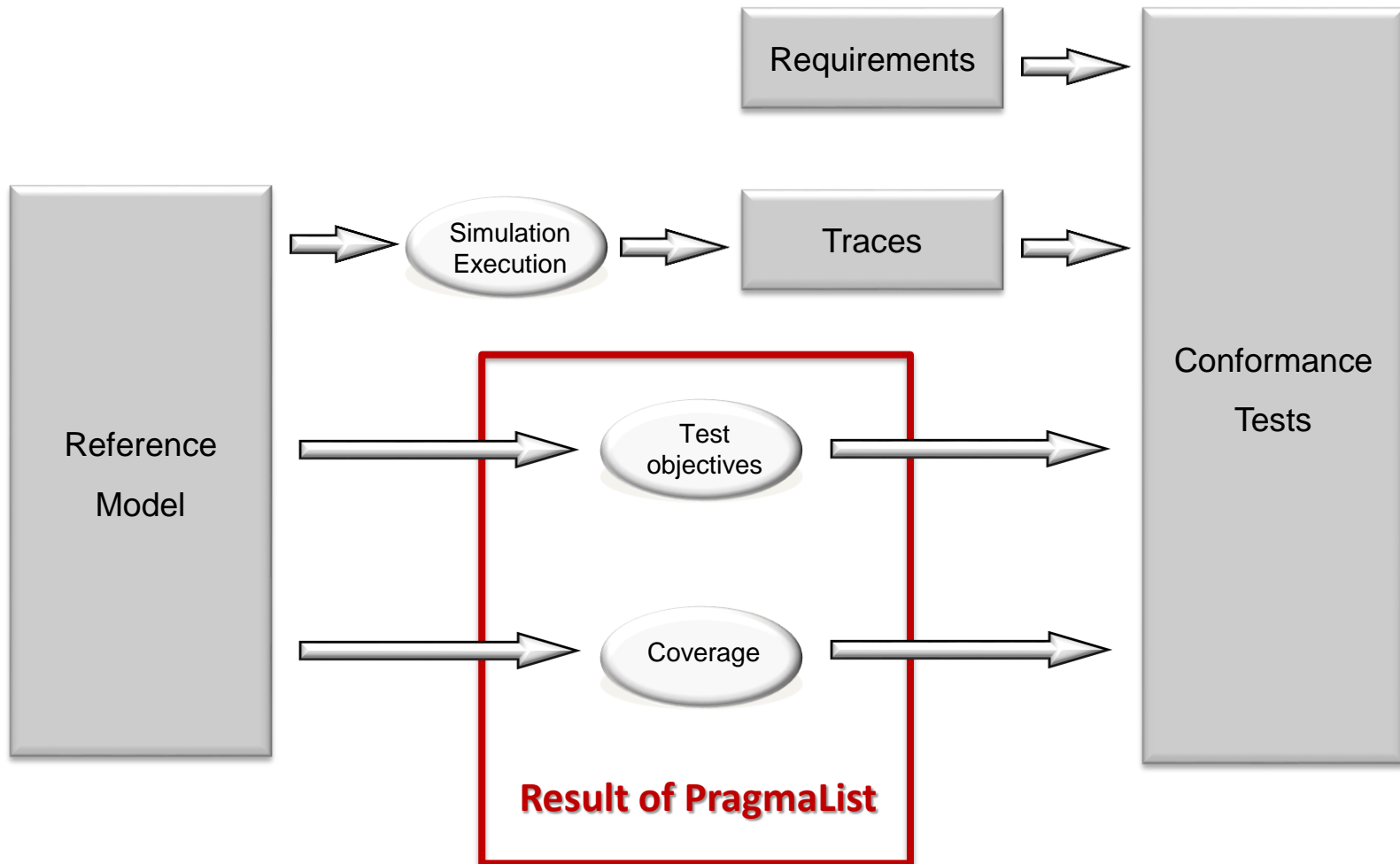


Model analysis technologies

- Partnership with specialized labs:
 - Exhaustive simulation,
 - **Symbolic resolution.**
- Properties:
 - **Model coverage,**
 - Static or dynamic property:
 - Property verification,
 - Test objectives.



Reference testing

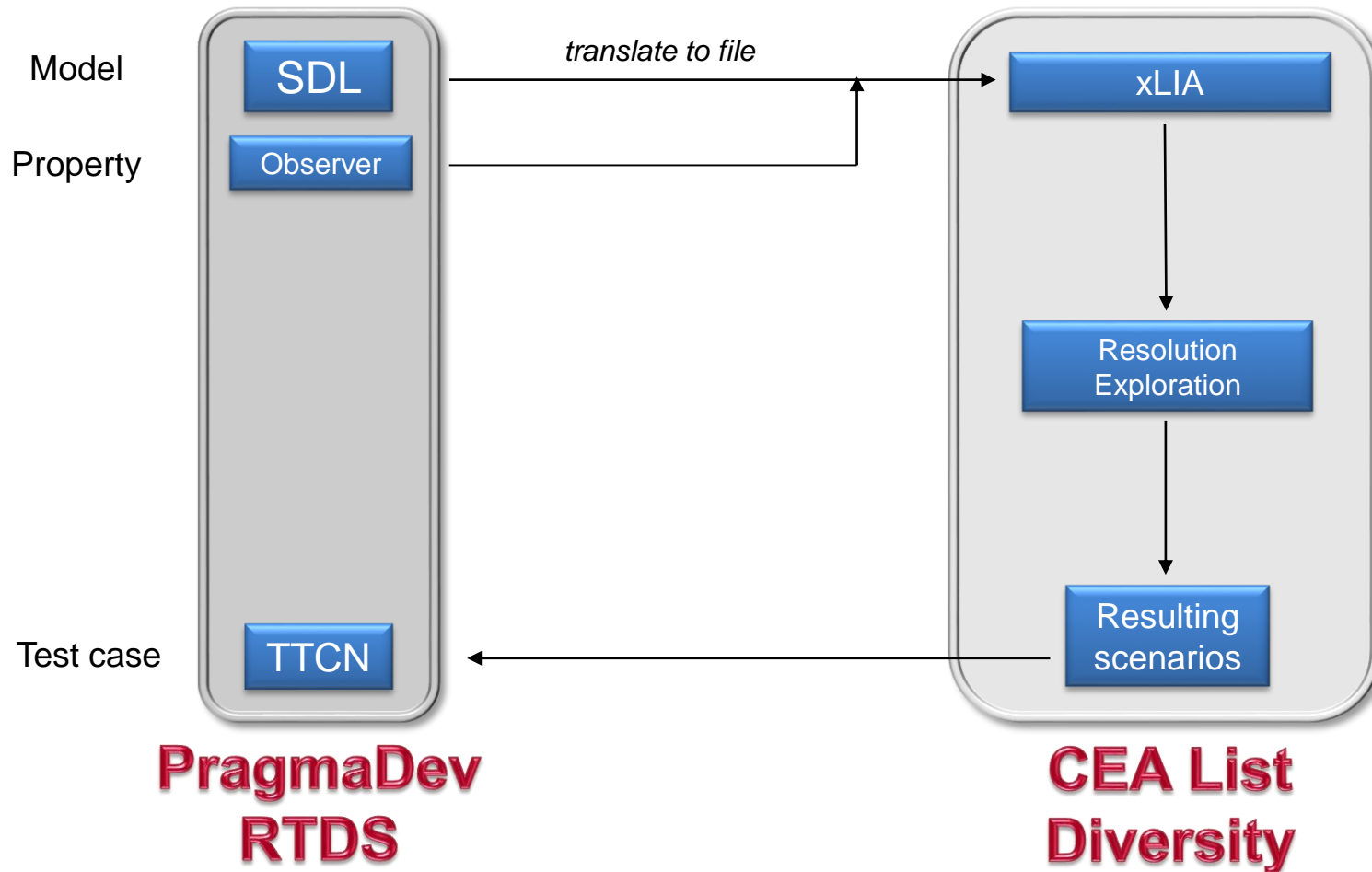


The project in four steps.

- **Step 1 : SDL to xLIA translation rules :**
 - Write the translation rules to convert SDL to xLIA.
- **Step 2 : SDL to xLIA translator :**
 - Write the xLIA generator from an SDL model.
- **Step 3 : Diversity adaptation to support SDL semantic :**
 - Work on SDL communication semantic,
 - Work on SDL timer semantic.
- **Step 4 : TTCN-3 formats output generation :**
 - TTCN-3 test cases formatting to be supported by RTDS.

xLIA is the CEA List Diversity file format to describe the model

Architecture



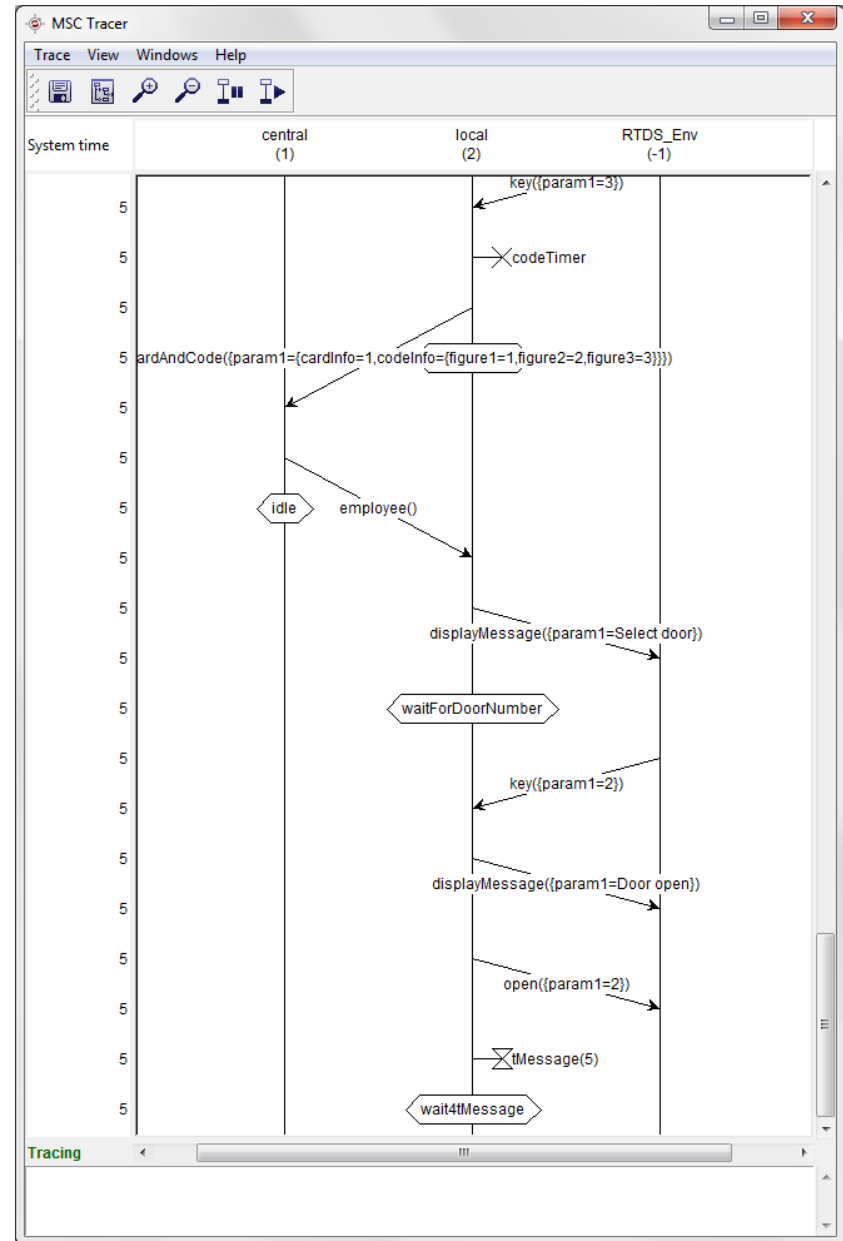
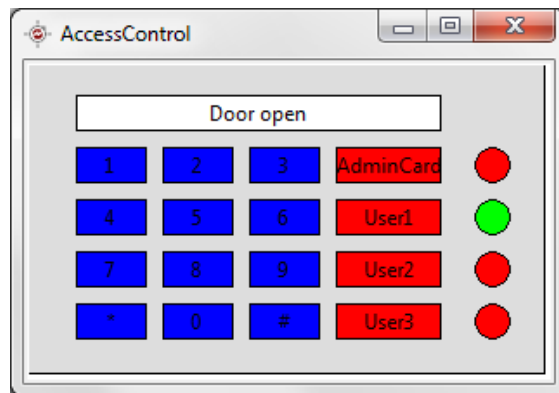
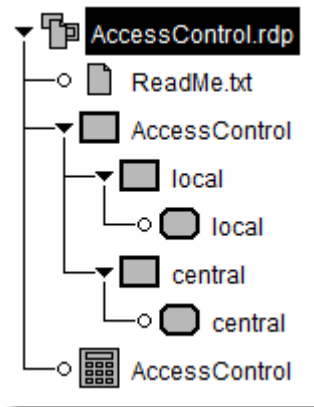
Four types of targets

- **Code coverage :**
 - To generate the minimum number of test cases that cover all transitions.
- **Transition :**
 - To generate a test case that covers a specific transition in the SDL model.
- **Property :**
 - To generate the test cases verifying a static property (process state, variable value, ...).
- **Observer :**
 - To generate the test cases verifying a dynamic property (succession of action or temporal rules). A dynamic property is defined as a state machine called observer.

Demonstration

An Access Control System:

- 2 state machines
- A card input with a 0..65535 integer as a parameter
- A key input with a 0..11 integer as a parameter



The image displays the PragmaList software interface, which is used for validation and model checking. It consists of several windows:

- Validation options:** This window shows configuration for the xLIA options. The 'Profiles' list includes Linux_Coverage, Linux_Observers, Linux_Properties, Linux_Targeting, Windows_Coverage, Windows_Observers, Windows_Properties, and Windows_Targeting. The 'xLIA options' section includes:
 - Path to Diversity: `{RTDS_HOME}\share\3rdparty\Diversity\windows\` (with a 'Browse...' button)
 - Max. calcul steps (-1 for no limit): 500
 - Max. height (-1 for no limit): 500
 - Max. width (-1 for no limit): -1
 - Strategy: BFS
 - Code coverage (selected)
 - Target Transitions
 - Symbol ID to target: [empty field]
 - Properties
 - Path to properties file: [empty field]
 - Observers
 - Path to observers file: [empty field]
- External model checking:** This window displays the results of the model checking process:

```
STOP CRITERIA PROCESSOR
The CONTEXT count : 367
The STEP count : 293

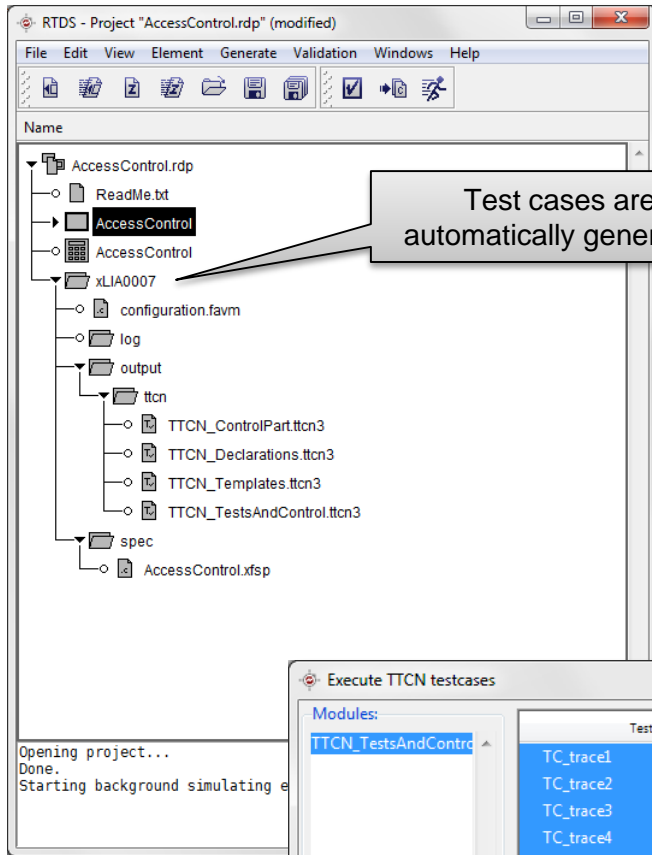
The Max HEIGHT reaching : 26
The Max WIDTH reaching : 76

The DEADLOCK found: 5

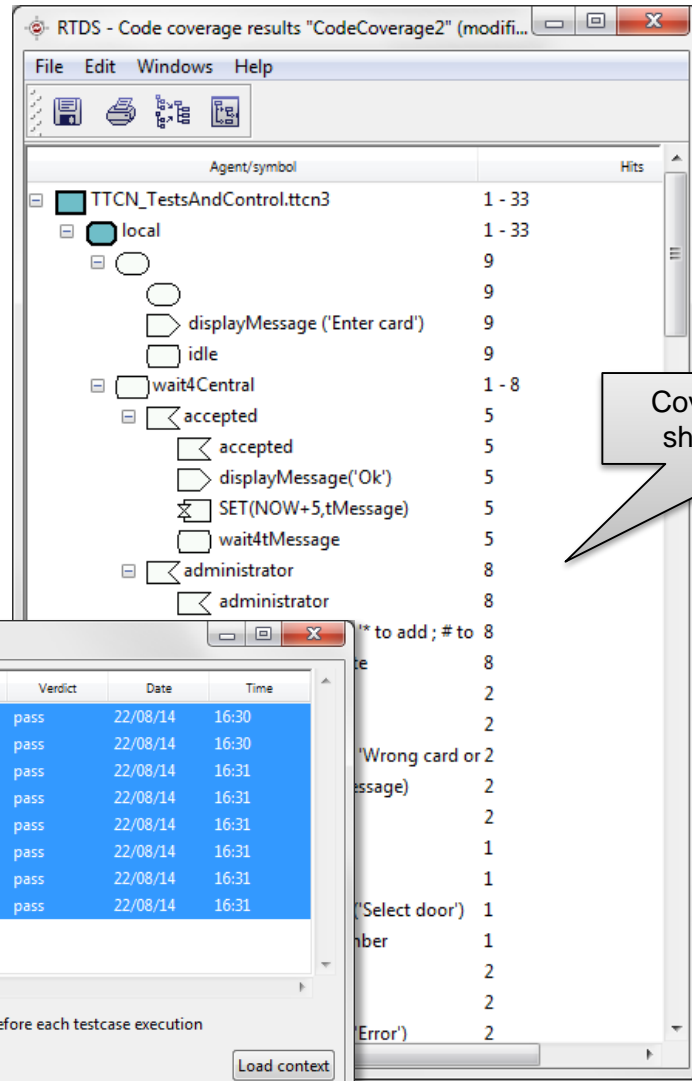
TRANSITION COVERAGE PROCESSOR
All the << 46 >> transitions are covered !
Number of nodes cut back: 322

REDUNDANCY
The positive detection count: 44 for 320 tests !

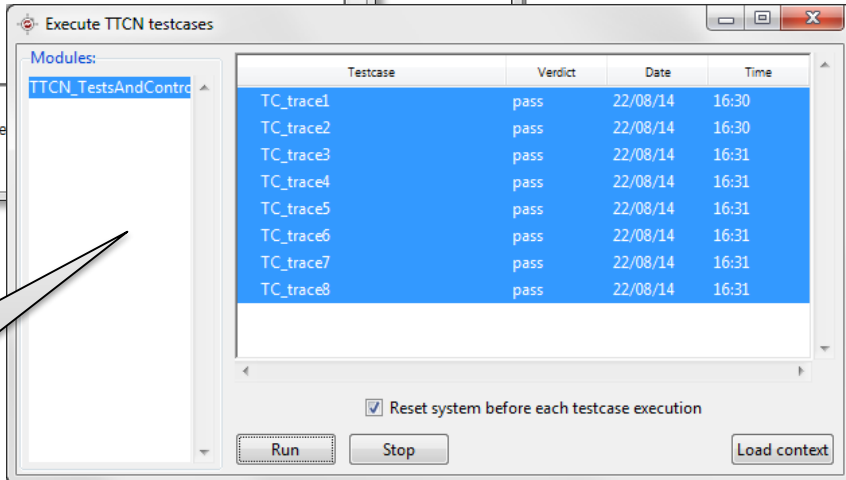
Extension - step: 1 / 500, context: 138, height: 20, width: 34
Extension - step: 2 / 500, context: 365, height: 27, width: 50
Extension - step: 3 / 500, context: 366, height: 27, width: 2
stop: 3 / 500, context: 370, height: 27, width: 50
```
- PragmaList spider graph:** Three windows showing spider graphs. Each graph has five axes: coverage (top), width (left), depth (bottom-left), step (bottom-right), and context (right). The coverage axis for all three graphs is labeled with the value 46. The width axis for the first two graphs is labeled with the value 1000. The context axis for the third graph is labeled with the value 1000. A large grey arrow points from the first graph to the second, and another from the second to the third, indicating a sequence of operations or a comparison of results.



Test cases are automatically generated



Coverage information shows full coverage



A Test manager helps to select the test cases

CEA List - Diversity

- Exploration time is always the same (10 secondes) whatever are the message parameter ranges.

Verimag - IF toolbox

- Exhaustive exploration
- Exploration time depends on message parameter range.

| Digit range Card range | 0..1 | 0..2 | 0..3 |
|---------------------------|------|------|-------|
| 0..1 | 13 | 126 | 721 |
| 0..2 | 38 | 316 | 2169 |
| 0..3 | 64 | 650 | 28234 |

Time to explore the model in seconds

On-going use cases

- SNCF: Radio Block Center (RBC)
- Alstom Belgium: Radio Gateway
- Alstom France: Passenger exchange
- Airbus: Air Traffic Control (ATC)
- Other: Secure transactions

Model Based Testing solution

- Integrated tool chain
- Non dedicated model
- Efficient symbolic kernel
 - Test automation
 - Reduce the number of test cases
 - Early in the development process