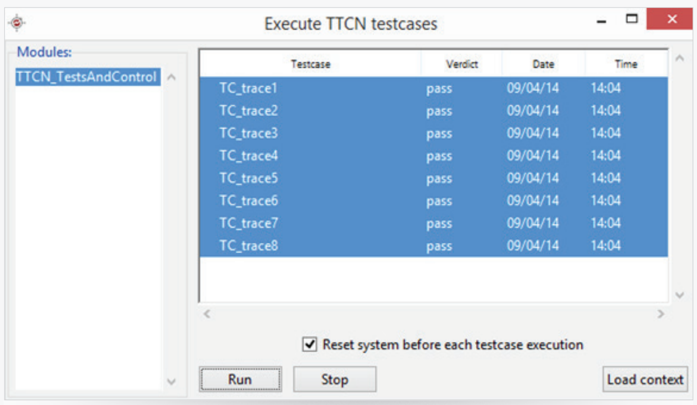
# PRAGMAList

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

## Integrated solution

PragmaDev and CEA LIST join forces to offer **an integrated solution to address two challenges: modeling and validation.**



| Testcase | Verdict | Date | Time |
|---|---|---|---|
| TC_trace1 | pass | 09/04/14 | 14:04 |
| TC_trace2 | pass | 09/04/14 | 14:04 |
| TC_trace3 | pass | 09/04/14 | 14:04 |
| TC_trace4 | pass | 09/04/14 | 14:04 |
| TC_trace5 | pass | 09/04/14 | 14:04 |
| TC_trace6 | pass | 09/04/14 | 14:04 |
| TC_trace7 | pass | 09/04/14 | 14:04 |
| TC_trace8 | pass | 09/04/14 | 14:04 |

[✓] Reset system before each testcase execution

Generated test cases

❶ The integration of both technologies provides a **user friendly** way to describe the system architecture and functions.

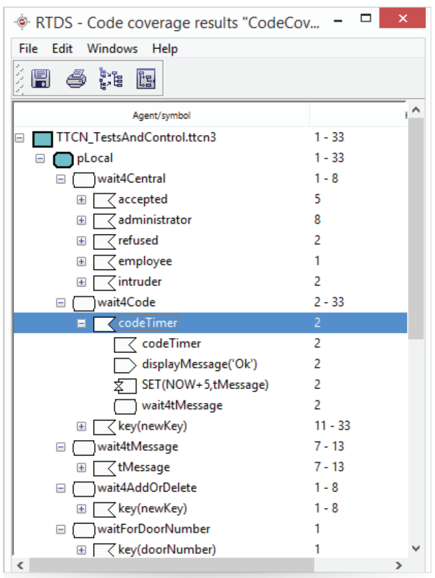
❷ Once the system is fully described, **validation options** are configured to define a set of objectives.

❸ These objectives may be **structural covers**, as **transition cover**, or more sophisticated ones, as condition/decision or to target one or more elements of the model defined by the user.

❹ Then the **minimum number of test cases** corresponding to these objectives are generated and can be played on a real implementation of the system. Moreover the generated test cases may be filtered by properties defined by the user.

## Pragmatic approach

Dedicated to modeling and testing technologies, PragmaDev has spent 13 years developing its Real Time Developer Studio (RTDS) tool with industrial users. **Built on a bottom up approach, the solution has moved up the V cycle from code to executable models and test cases.** As a result of this pragmatic approach, today when taking a plane, driving a car, or giving a phone call, there is quite a chance PragmaDev has been involved in the design.



| Agent/symbol | |
|---|---|
| TTCN_TestsAndControl.ttcn3 | 1 - 33 |
| pLocal | 1 - 33 |
| wait4Central | 1 - 8 |
| accepted | 5 |
| administrator | 8 |
| refused | 2 |
| employee | 1 |
| intruder | 2 |
| wait4Code | 2 - 33 |
| codeTimer | 2 |
| codeTimer | 2 |
| displayMessage('Ok') | 2 |
| SET(NOW+5,tMessage) | 2 |
| wait4tMessage | 2 |
| key(newKey) | 11 - 33 |
| wait4tMessage | 7 - 13 |
| tMessage | 7 - 13 |
| wait4AddOrDelete | 1 - 8 |
| key(newKey) | 1 - 8 |
| waitForDoorNumber | 1 |
| key(doorNumber) | 1 |

Graphical view of the coverage

### ■ Executable models

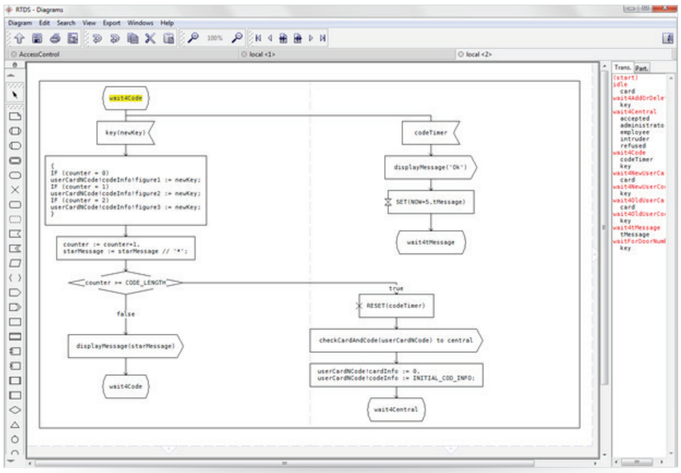
PragmaDev tools are exclusively based on recognized international standards. Easy to take in hand, the system is organized with graphical elements and an action language is integrated for a precise description of the system behavior.

The model correctness is verified with the built-in simulator. Interactive or automatic execution of the model with graphical traces and access to all internal information in the system is made to ensure the description is correct.

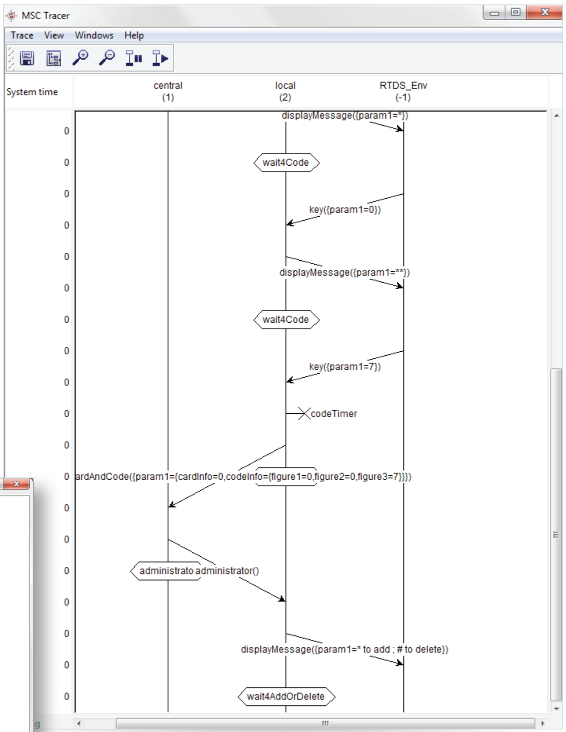
PRAGMADEV real time development tools

cea tech list

# Model Based Testing unique symbolic engine



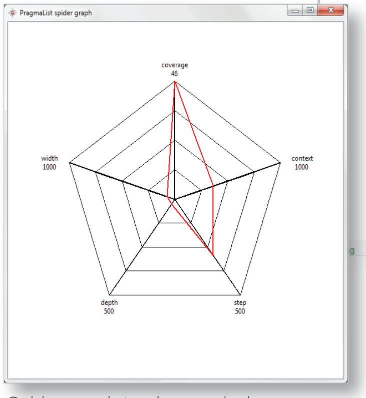
Detailed model behavior



MSC Tracer
Trace View Windows Help
System time
central (1)
local (2)
RTDS_Env (-1)
displayMessage({param1="})
wait4Code
key({param1=0})
displayMessage({param1="*"})
wait4Code
key({param1=7})
codeTimer
ardAndCode({param1={cardInfo=0,codeInfo={figure1=0,figure2=0,figure3=7}}})
administrato administrator()
displayMessage({param1=" to add ; # to delete"})
wait4AddOrDelete

Graphical execution traces

Automatic documentation generation helps communication among the different stakeholders and integration with third party tool gives access to traceability and version management features.

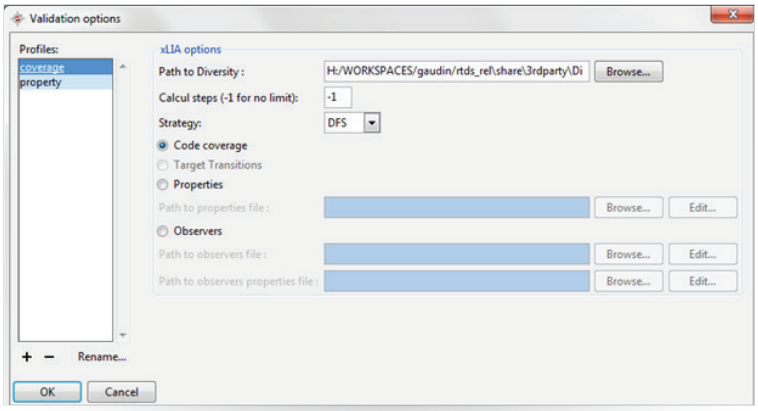


Spider graph to view analysis progress

## Diversity

The Diversity tool developed by the CEA LIST is a validation and verification platform based on model analysis. Models may be described with the help of dataflow languages as well as stateflow-type languages describing potentially concurrent and communicating automata.

On the one hand, Diversity can generate simulation scenarios to validate the input model. These scenarios can optionally be used by a simulator associated with the modeling environment like the one in RTDS. On the other hand, Diversity can generate test cases from the same model to be used to verify the compliance of the implementation (or SUT for System Under Test) with the model.


Validation options

Profiles: coverage, property

xLIA options

Path to Diversity : H:/WORKSPACES/gaudin/rtds_rel\share\3rdparty\Di

Calcul steps (-1 for no limit): -1

Strategy: DFS

[✓ Code coverage] Target Transitions Properties

Path to properties file :

Observers

Path to observers file :

Path to observers properties file :

Verification profiles

For both uses, the Diversity process consists in three steps. First, the input model is analyzed and translated into a Diversity's internal representation, called XLIA. Second, an exhaustive symbolic exploration of nominal behaviors is performed (symbolic, in order to avoid numerical combinatorial explosion). This symbolic execution can highlight application-independent unexpected behaviours such as deadlocks or over-designing (parts of model never activated). Moreover, in order to guarantee termination or to limit the number of generated test cases, several basic structural criteria may be used during the symbolic execution. Third, Diversity associates each selected behavior with one (or more) numerical test case.

**Contact** | info@pragmadev.com | www.pragmadev.com