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
- Airbus
- ESA
- DGA
- CS Communication & Systemes

Automotive
- Renault

Telecoms
- Alcatel-Lucent
- Aeroflex
- Ercom
- Wipro
- NewLogic
- LG Electronics
- Thomson
- KETI

Semi-conductor
- ST Ericsson
- Toshiba
- Mitumi
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

- **céa LIST**
  - 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.
Verify the model

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

- Breakpoints
- Step
- View internals
- Graphical traces
Model coverage

- Graphical model coverage analysis
- Merge feature

The model is:
• Correct
• Fully covered
Requirements for a good testing language

- Relies on the same basic services:
  - Messages
  - Procedures
  - Timers
  - Parallel execution

Ø 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

Reference Model

Simulation Execution

Requirements

Traces

Conformance Tests

Test objectives

Coverage

Result of PragmaList
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

Model
- SDL
- Observer

Property

Test case
- TTCN

PragmaDev RTDS

Implementation

translate to file

xLIA

Resolution Exploration

Resulting scenarios

CEA List Diversity

PragmaDev RTDS
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
Example

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 seconds) whatever are the message parameter ranges.

**Verimag - IF toolbox**

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

<table>
<thead>
<tr>
<th>Card range</th>
<th>Digit range</th>
<th>0..1</th>
<th>0..2</th>
<th>0..3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0..1</td>
<td>13</td>
<td>126</td>
<td>721</td>
<td></td>
</tr>
<tr>
<td>0..2</td>
<td>38</td>
<td>316</td>
<td>2169</td>
<td></td>
</tr>
<tr>
<td>0..3</td>
<td>64</td>
<td>650</td>
<td>28234</td>
<td></td>
</tr>
</tbody>
</table>

*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