Efficient Testing Through Graphical Test Design

Simplifying Test Development via Graphical Notations in vTESTstudio
Schematic Overview: vTESTstudio and CANoe

**vTESTstudio**

**Design & Implementation**
- Test programming (CAPL, C#)
- Table based test design
- Graphical test design
- Parameters and variants
- Classification tree method
- Stimulation curves
- Traceability

**Test Units:**
- Code
- Parameter
- Traceability Information
- ...

**CANoe + VT Modules + Bus Interfaces**

**Execution & Reporting**
- Real-time execution of tests
- Access to SUT via
  - IOs
  - bus systems
  - protocols (diagnostics, XCP, …)
  - debug interfaces
- Analysis of test run
- Automatic detailed reporting

**Symbol databases:**
- DBC
- ARXML
- CDD
- …
Let’s start testing

Live Demo
Efficient Testing Through Graphical Test Design

SUT: Central Locking System

The following requirements are to be tested:

<table>
<thead>
<tr>
<th>ID</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1. HIL test</td>
</tr>
<tr>
<td>3</td>
<td>1.1 static requirements</td>
</tr>
<tr>
<td>7</td>
<td>1.1.1 lock statically</td>
</tr>
<tr>
<td></td>
<td>After a lock request, the car must be locked within 500ms.</td>
</tr>
<tr>
<td></td>
<td>This condition must be fulfilled with engine off and engine running.</td>
</tr>
<tr>
<td>8</td>
<td>1.1.2 unlock statically</td>
</tr>
<tr>
<td></td>
<td>After an unlock request, the car must be unlocked within 500ms.</td>
</tr>
<tr>
<td></td>
<td>This condition must be fulfilled with engine off and engine running.</td>
</tr>
<tr>
<td>13</td>
<td>1.1.3 crash detection statically</td>
</tr>
<tr>
<td></td>
<td>If a crash is detected while the engine is running, the car must be unlocked within 500ms.</td>
</tr>
<tr>
<td>5</td>
<td>1.2 velocity dependent requirements</td>
</tr>
<tr>
<td>12</td>
<td>1.2.1 lock by increasing velocity</td>
</tr>
<tr>
<td></td>
<td>If the car is accelerated to 50 km/h (or higher), the car must be locked automatically within 500ms.</td>
</tr>
<tr>
<td>9</td>
<td>1.2.2 no unlock by slowing down</td>
</tr>
<tr>
<td></td>
<td>If the car is decelerated under 50 km/h, the car must not be unlocked automatically.</td>
</tr>
<tr>
<td>10</td>
<td>1.2.3 crash detection when car is moving</td>
</tr>
<tr>
<td></td>
<td>If a crash is detected while the car is moving, the car must be unlocked within 500ms.</td>
</tr>
</tbody>
</table>
Test Design Editors

Classification Tree Method

- Find test case data by the classification tree method
- Graphical user interface supports the definition of all relevant input values
- Build test vectors by manual and automatic combination of input values
- Specific support for boundary values
- Extend test vectors by expected values
- Use test vectors in implemented test cases, e.g. for the parametrization of test case lists
Test Design Editors

Test Table Editor

- Easy definition of test sequences without programming knowledge
- Comfortable support of test step parameterization by drag & drop
- Direct calls to CAPL and C# test cases and functions possible
- Definition of reusable functions
The expected behavior of the SUT is modeled as a state transition diagram.

The test model is a dedicated system model, e.g., extended by (self-)transitions as unspecified scenarios.

Automatic generation of test cases based on transition coverage.

Different generation algorithm supported, e.g., Chinese Postman and breadth search algorithm.

- **Chinese Postman algorithm**
- **Breadth search algorithm**
Test Design Editors

Test Sequence Diagram Editor

- Test sequence diagram for a clear and concise representation
- Direct calls to CAPL and C# test functions possible
- Test coverage easily to be reviewed
- Easy reuse of test sequence parts
- Support of reusable sub-diagrams
Traceability Matrix

- Traceability Matrix gives an overview over all covered and non-covered requirements / test descriptions in a test project
- Navigation from a requirement / test description to linked test case implementations
- Possible filtering on variants
- Support of requirements change management
- Export to Excel enables documentation and review by customer
- Export to XML enables execution planning in REQM/TDM tools
Test Execution in CANoe

- Automatic test execution and report generation in CANoe
- Select variants just before test start
- Possibility to observe and analyze a test during the test run in the Test Trace Window
- Detailed information about executed test cases and test steps
Test Reporting

- Automatic detailed reporting by test execution in CANoe
- Vector CANoe Test Report Viewer for an easy and comfortable analysis of a test run
- Filtering on failed test cases, test steps, ...
- Grouping by verdict
- Comprehensive user-defined queries on report data, e.g. “all requirements linked to a failed test case”
For more information about Vector and our products please visit

www.vector.com

Author:
Gronau, Simone
Vector Germany