Parameterized Unit Tests for Opel and Vauxhall Brands

Opel Automobile GmbH is a car maker of the Opel and British Vauxhall brands.

The software and code for many embedded controllers of these vehicles with conventional and alternative engines are developed here.

Reactis® - a product developed by Reactive Systems, Inc - was integrated in the global software development process already 10 years ago. This was an important initial effort and the tool is used now to validate software components and to perform parameterized unit tests.

Application Example

A good example for an application is the software actively controlling the particle filter of gasoline engines. This software will be integrated in a high number of engines to meet the newest emission standards.

The software is safety relevant due to the very high operating temperatures of the filter during active regeneration.

Therefore all software-components for the gasoline particle filter were tested each using Reactis. A component consists of an algorithm part, diagnostics and communication. To increase the Reactis performance the error management was slightly simplified by replacing some library blocks. This lead to a very high performance of Reactis compared to an integrated system of all these components (or even an integrated controller software). No initialisation step in Reactis takes more than a few seconds.

Golden Situation

We have reached a golden situation:

- The tedious and unproductive work for manually programming code coverage inputs is eliminated. This is automated now.
- The repeating of this manual work needed for all the production code variants is also eliminated of course and multiplies the benefit.
- The methods for automating this (using formal methods etc.) is put into the hands of experts and can be shared with a wide community.
- We can increase the software quality while also saving money.

Why we selected Reactis

We want to highlight here only a few features of Reactis which make this possible and are important reasons why we selected Reactis:

- Guided execution (relation to symbolic or concolic execution)
- Supporting also our own Simulink®-blockset for code generation consisting of custom level-2-S-functions (Look-Up-Tables etc.) - the tool integration was considerably simplified.
- The extremely good debugging to understand the software and to find the root cause of errors. (You only have to take care about non-executed code like disabled conditions or conditionally executed subsystems. Here the scopes show the last valid values)
- Tracking of requirements down to the test report.
- Very smart handling of calibration variants in Reactis and Simulink.

Quality of the test cases

The quality of the test-case input of a single automatically generated test case might be poorer than a manual created test case on the first glance. A simple example for this can be an accidental symmetry on two input signals of a function in the automatically generated test. This symmetry could hide for example that the two signals are erroneously twisted. But looking at a generated test suite will give a different result: The tool can easily surpass branch coverage (one short test case per condition). Increasing now the path coverage will produce more than one test per branch and the likelihood of accidental symmetries at the same position in all these longer tests will almost always disappear to zero due to the fact that the execution uses random values for the model inputs.

Use Cases

Here is a short incomplete list of different used cases how Reactis is used in the process:

- After implementation the software must be executed without time-delay (caused e.g. by controller builds, simulation environment updates) to test changes.
  Assertions are added.
- Compare software execution behaviour with requirements (Reactis Assert-Statements)
- Generate inputs needed from environment: (automatic generation).
- Execute all paths of the software behaviour automatically (coverage)
- Compare behaviour of new and old implementation (change).
- Compare behaviour of two different implementations (supplier/In-House) in a bisimulation.

Conclusion

The initial effort to implement Reactis is widely outperformed by the gain in speed and quality. The technology is mature. We can fully recommend Reactis and for us it is the benchmark for products of it's kind having ever higher advantages for the demanding future needs.