Generalised simulation environment for software testing

Famantanantsoa Randimbivololona, Philippe Le Meur, Sébastien Lansiaux

Session 2A: Testing

Title: Generalised simulation environment for software testing

Authors:

Famantanantsoa RANDIMBIVOLOLONA (speaker)
AIRBUS France
EYYW
e-mail: famanta.randim@airbus.com
phone: +33 561 930 825

Philippe LE MEUR
AIRBUS France
EYYW
e-mail:philippe.lemeur@airbus.com
phone: +33 561 183 049

Sébastien LANSIAUX
SOPRA Group

e-mail: slansiaux@sopragroup.com
phone: +33 561 008 500

Topics of the conference:
- Verification and validation

Abstract:
The cost of the verification process is certainly one of the major engineering issue in the domain of embedded safety-sensitive systems such as avionis. Test environments, made up of dedicated software and hardware means, are among the most significantly contributing factors. This paper reports about an alternative approach where test environments are totally simulated for all test phases. Simulation is not per se a new approach, the expected positive gap is expected from its generalisation. test phases coverage. Our works on this generalised simulation approach are part the on-going RNTL/ATLAS research project.

1 - Software testing in avionics domain

In the present verification engineering state-of-the-art, testing is the privileged verification technique. It is based on the execution of the software on a set of data selected on a requirements-driven basis. Manual techniques of reviews and inspection complete the tests. According to the DO-178B standards, the tests of an on-board software have two complementary objectives:

(a) The one is to demonstrate that the software satisfies its specifications (conformance with low level and high level requirements).

(b) The second objective is to demonstrate with a high degree of confidence that the errors, which could lead to unacceptable conditions of failure (as determined by the system safety analysis), have been eliminated.

Testing activities are structured in successive phases that are: unit test, integration test of sub-assemblies, integration test with the hardware, final integration test.

- Unit test address the verification of the properties of one decompositon unit of the software in the sense of verification (one verification unit corresponds to at least one realisation unit).
- Integration test of sub-assemblies address the correctness of units composition, the invariance of unit properties by composition, the correctness of a complex property by unit properties composition.
- Integration test with hardware address the correctness of the control of hardware resources by the software adn also some unit tests closely related with hardware characteristics of the target equipment.
- Final integration test address the conformity of the complete software running on the real target in a configuration that may require the presence of one, two or three target computers depending on the requirements points that have to be tested.

The ending of the whole testing activities is determined by the achievement of the test coverage which is defined by both functional and structural criteria.

This process supposes that in each of the test phases it is possible to control the execution of the program and to observe the changes of the relevant variables values. But neither embedded software, nor avionics have “natural” interfaces which enable/support these testability conditions. This can be achieved only with the use of test-oriented software and hardware means which are dedicated, intrusive, specific of each computer and its active environment. They are emulators, logic analysers, integration benches, test boards, test runtime/drivers, … Most of these means come from independent vendors so their mutual integration within a coherent test framewrok can rise difficult problems, sometimes unsolvable.

2 – The generalised simulation solution

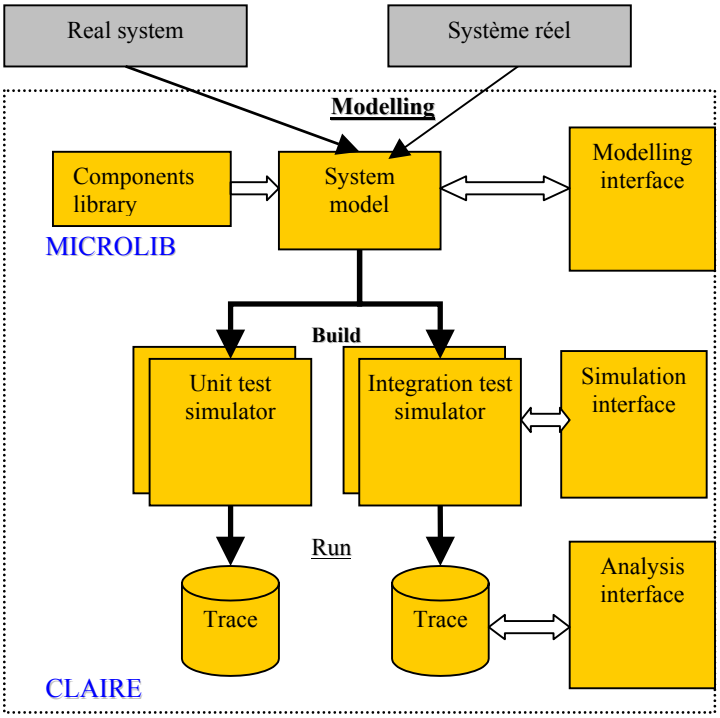
In the generalised simulation approach, all the test environment are modelled and simulated by software. The main expected benefits are:

- Significant reduction of the test environments cost
- Homogeneous, easy to use, dedicated-hardware-free test envronments (“all tests on a workstation”),
- Parallelisation of hardware cycle and software cycle,
- Non intrusive test,
- Debug capabilities for multiprocessors or distributed architectures

However, there are some points that require extensive experiments and validation :

- Effective performances of the simulated computer/board,
- Accuracy of the simulated computer (notably for floating point),
- Address evenly all the test phases (unit, subassemblies integration, final integration)
- Qualification issues in a certified process

In this approach, only the relevant elements are taken into account in the model so that the simulation does not incur penalties due to test phase irrelevant extra-features (e.g. there is no need for a detailed micro-architecture model of a processor to deal with unit functional testing, an instruction set architecture model is sufficient). Typically for unit test, only a basic simulator -processor with memory- is needed. This basic simulator can be extended smoothly to include peripherals required for integration testing up to full equipment interacting with its active environment. This extension from simple to more complete model is done by component composition and reuse. The component are either basic components that are part of the tool (based on the CEA/CLAIRE tool) or user-defined models that have been stored in a components library.



The generalised simulation tool has three main functions:

- Graphical modelling. The real system of interest is asbtracted into a model that retains only the relevant characteristics. The model is expressed in terms of "typed" components interconnection.
- Then the simulator is built: description of the dynamics behavior of new user-components (in C code) and the executable simulator is generated automatically
- The execution of the simulator with the target software can be done in an interactive mode with the use of gdb debugger, or in a "blind" batch mode for a totally automatic test process
- The results of the execution are finally exploited using the trace analysis facilities that enable the verification of temporal properties over a finite execution trace.

3 - The current point

An industrial prototype of this tool is now available which handles a set of microprocessors among which the PPC 755 processor from Motorola and the SHARC 21060 DSP from Analog Devices. These processors are representative of current hardware that are used in avionics computers. Experiments with realistic avionics test workloads show that unit test as well integration test (including communication test) can be supported in a very simple way. More complex test environments are now under investigation.