

Title of Paper

Automated IMA Module Configuration Testing

Presenter

Dr. Oliver Meyer / Verified Systems International (D)

Julien Pfefferkorn / Airbus (D)

Instructional Level

Introductory Intermediate Advanced

Target Group

Test Practitioner and Engineers, Software and Test Managers, QA Managers and Development Managers as well as other professionals interested in building and delivering better software.

Keywords

- Automated Configuration Testing
 - Test Aspects for Operating System Testing
 - Test Validation by Mutation Testing
 - Integrated Modular Avionics (IMA)
-

Abstract

This abstract describes a joint project of Airbus and Verified Systems International GmbH about the testing of configured Integrated Modular Avionics (IMA) modules. The IMA approach aims at providing a unified architecture for various avionics controllers which used to be implemented as specialized hardware/software solutions designed for specific application needs. Moreover, the IMA modules, also called Core Processing I/O Modules (CPIOM), are able to host multiple controller applications on a single controller module by means of partitioning, multitasking and shared I/O. IMA modules will be employed as the main controller platform for many systems of the A380 and future Airbus aircraft.

In order to be able to support the needs of very different controller applications as well as possible future extensions, IMA modules are highly configurable. Among other aspects, an IMA Module Configuration Definition (ICD) includes

- the spatial segregation of partitions, i.e. memory sizes and mappings as well as accessibility restrictions,
- the temporal segregation of partitions, i.e. the hard-real time scheduling tables, defined as multiple layers in order to assign different scheduling times and cycles to single partitions,
- I/O port access assignments and message format configurations for a unified access to data on different I/O types including AFDX, CAN, Arinc 429, discrettes, analogue and more specialized I/O variants,
- I/O monitoring configurations for internal checking of various aspects of output I/Os, e.g. currents, voltages and switch states.

Some aspects of these configuration parameters do only have local influence within a single partition whereas others have to be set consistently for a set of partitions or even the complete module.

Therefore, the ICD has been split up into several layers. The corresponding data of these layers is defined and administered by different stakeholders with different roles – module supplier, system supplier and Airbus as module integrator. During early integration phases, all layers are repeatedly modified, but changes on a specific layer cannot be performed without taking into account the restrictions of interdependent ICD layers, where consistency needs to be strictly controlled.

Owing to the complexity of IMA module configurations, it is necessary to gain confidence both in their consistency and in the way the IMA module operating system interprets these configurations. Therefore, a test approach for the *configured module testing* has been set up with the following main objectives:

- Detect and fix configuration inconsistencies as early as possible.
- Ease the localisation of anomalies on application and system test level by means of a multilevel testing approach. For example, if a specific port is not working correctly on application test level, but has been demonstrated to work as expected on configuration test level, the configuration test results can be used to exclude a faulty port configuration. This allows restricting the debugging activities to the application itself, i.e. the bug must be owing to an incorrect *usage* of the port.
- Check the spatial and temporal partitioning robustness of the IMA module. Application and system level tests are usually not able to explicitly trigger memory and timing faults; they do only occur in case of major application errors and therefore, the module behaviour in case of such faults has to be checked separately on configuration testing level.

The system under test on configured module testing level is the CPIOM equipped with its operating system and the configuration to be tested. Since the operating system is a medium layer of the complete system, both a test application executed within the CPIOM on top of the configured operating system and an external test component are necessary for stimulating and checking the CPIOM behaviour. The automated test environment set up for these tests is based on Verified Systems' RT-Tester and an ICD Parser, which allows to instantiate tests and the test environment exactly according to the given ICD. A test application executed in parallel on all configured partitions on the CPIOM is compiled using include files based on ICD Parser outputs and thus, allows to access operating system components with parameters matching ICD contents. The test application itself is a command interpreter reacting to stimuli from the outside via I/O ports selected from the ICD and it returns results of operating system calls back to the calling external component. Thus, it can be completely controlled from an external test entity. This external test component sets up specific test situations by triggering sequences of test application activities and/or combinations of I/O activities. At the same time, it performs checks in real-time both on the results of the test application and on related and possibly unrelated I/O value changes of the output interfaces of the CPIOM.

As outlined, the test specifications and the test environment are instantiated according to the ICD under test. They are generic with regard to the following parameters:

- Type of the CPIOM
- The number of tests to be executed sequentially or in parallel, depending on the number of partitions configured in the given ICD
- Large range of ICD entries; tests are written on an abstract level, e.g. specific checks are performed "for all configured ports" or "for spatial boundaries as defined in the ICD"

The tests performed on configured modules cover a wide range of aspects, which can be roughly separated into the three following test classes:

- Normal behaviour tests including read/write port access, AFDX bandwidth limitation, debouncing timing of discretes, I/O monitoring (so-called readback) access, non-volatile memory access, non-volatile memory write timing, scheduling in case of maximum load on user level (infinite loops), memory mapping, memory access to allowed areas (read/write), operating system objects memory allocation, stack allocation and power consumption in various situations
- Robustness behaviour tests including port access with illegal and/or wrong parameters, logbook access with illegal and/or wrong parameters, operating system objects memory allocation over-

flow, stack overflow, trying to send illegal data, illegal port data reception detection (out-of-range situations on input ports)

- Partitioning robustness tests including port creation of other partitions, port access of other partitions, I/O independencies (single I/O change, all I/O check of same I/O type), logbook creation of other partitions, logbook access of other partitions, memory access violations (data area, code area, system memory, read and write) and health monitoring reactions, scheduling robustness in case of maximum I/O load (uninterruptible system calls at scheduling slice boundaries)

The test instantiation, compilation and execution are batch executable. A complete automated test suite compilation and execution currently takes approximately one day, depending on the ICD complexity. This does not include the time for the test evaluation, since the evaluation effort depends on the number of discrepancies found during test execution.

A cluster of three multi-processor standard PCs running Linux with a hard real-time CPU reservation kernel extension is used dynamically both for parallel test compilation and for real-time test execution on multiple PCs in parallel. Whenever an exclusive CPU reservation for a hard real-time execution is necessary on a specific cluster node, the compilation process is delayed and moved to a non-reserved CPU.

For checking the correctness of the test specifications and the test environment, two independent validation processes have been set-up and performed. In addition to the classical validation approach of manually checking the written test procedures against the test design documents, the error detection capability has been demonstrated by running the tests with corrupted configurations. This run-time validation approach is based on slightly modified ICDs which are derived from original ICDs and modified according to the pass/fail criteria given in the test design document such that the behaviour of the CPIOM with the modified ICD must show a deviation which has to be detected during test execution. While deriving the modified ICD, the expected test results have been written down and have been compared later on with the test results after the validation test was executed.

The run-time validation was performed after the manual validation process was completed and turned out to be useful since it detected three bugs in the test environment. One problem was simply overlooked during the manual validation approach whereas a second problem was based on a performance issue and the third problem was owing to incorrect default value assumptions in case of certain empty ICD fields. The latter two bugs could only be found during test execution in the original environment with the original test timing, but both problems were relevant and actually decreased the error detection capabilities of the test approach.

The high degree of automation of this configuration testing approach allows the test of modified ICDs with a minimum effort. As long as the selected I/O port configurations for the test application commanding are unchanged, it is sufficient to insert the ICD files at the correct location in the test environment, select the new ICD as test target and start the test environment generation and test execution. Selecting new command ports requires scanning the ICD for appropriate I/O definitions, which can also be done in a few hours for a complete module configuration. Usually, similar ICDs are defined for a set of four redundant IMA modules. Thus, the effort for finding command ports has to be invested only once for the first ICD of such a set, as long as identical ports exist for all four module positions, which is usually the case.

The configured module testing has revealed anomalies of different criticality levels while testing five different ICD sets for redundant module positions. The detection of these anomalies was helpful during the development phase. A certain amount of inconsistencies detected in the configurations under test were already found during test setup and compilation since the derivation of the test environment and the test procedure compilation itself relies on a number of underlying assumptions. If these assumptions are not fulfilled, the test environment generation or the test specification compilation already fails and the deviations are detected before the test is actually executed.

Currently, automated regression tests of extended IMA configurations are performed. Furthermore, the test environment is ported to the new version of Verified's RT-Tester which uses an advanced test specification language. The results of these ongoing activities will also be outlined during the presentation of this abstract at the ICSTest.



Biography

Dr. Oliver Meyer has studied computer science at the University of Bremen and received his diploma about the design, risk analysis and formal verification of safe autonomous systems in 1997. Afterwards, he worked for four years at the University of Bremen as a Research Assistant in the area of formal language based real-time testing approaches. As a member of the Bremen Institute of Safe Systems, Oliver Meyer participated in a number of real-time testing and simulation projects for avionics and space controllers. He received his Ph.D. in 2001 with his work about temporal aspects of formal specification based real-time testing and their practical implications. Since 2001 he is project manager at Verified Systems International GmbH, mainly working in test projects of embedded controllers in the avionics domain.

Contact information of Presenter

Dr. Oliver Meyer
Verified Systems International GmbH
Parkstrasse 123
D-28209 Bremen
Germany
E-mail: om@verified.de
Phone: +49 (0)421 57204-15
Fax: +49 (0)421 57204-22

Julien Pfefferkorn
Department BCRVM
Airbus
Kreetslag 10
D-21129 Hamburg
Germany
E-mail: julien.pfefferkorn@airbus.com
Phone: +49 (0)40 743 83709
Fax: +49 (0)40 743 81870
