Software Testing for Embedded Avionics Systems

Institute for System Programming
of the Russian Academy of Sciences (ISPRAS) *

1. Goals of airborne software testing.

Testing of software for embedded avionics systems has two complementary objectives. One objective is to demonstrate that the software satisfies its requirements. The second objective is to demonstrate with a high degree of confidence that errors, which could lead to unacceptable failure conditions, as determined by the system safety assessment process, have been removed. Requirements-based testing is emphasized because this strategy has been found to be the most effective at revealing errors. Therefore, to satisfy the airborne software testing objectives in accordance with the DO-178B [3] standard:

- test cases should be based primarily on the software requirements;
- test cases should be developed to verify correct functionality and to establish conditions that reveal potential errors;
- software requirements coverage analysis should determine what software requirements were not tested;
- structural coverage analysis should determine what software structures were not exercised during testing.

2. Conformance testing for ARINC-653 standard.

The ARINC-653 standard – "Avionics Application Software Standard Interface" – was developed by ARINC (Aeronautical Radio, Inc.) in 1997. This standard defines an application program interface APEX (Application/Executive) between an operating system of the avionics computer and application software. At present, it has been chosen as the main standard to be implemented in real-time operating systems (RTOS) for embedded avionics systems.

To support the requirements-based conformance testing and to simplify the certification process of commercial of-the-shelf software products, the ARINC-653 standard includes Part 3 – Conformity Test Specification – containing description of test cases for checking functionality of an RTOS application program interface (API) on conformance with the requirements from ARINC-653 Part 1 – Required Services. This test suite includes about 250 test cases covering the requirements to all 56 functions required to be implemented in any ARINC-653 compliant RTOS. The operating system obtains the status of full compliance with ARINC-653 Part 1 when all test cases from this test suite pass. The development of conformity test suite specification for ARINC-653 Part 2 – Extended Services – is now in progress by the ARINC-653 standardization committee.

The Institute for System Programming of the Russian Academy of Sciences (ISP RAS) has developed the test suite for checking an RTOS on compliance with the ARINC-653 Part 1. This test suite includes the test cases specified in the ARINC-653 Part 3. However, many errors were revealed in the ARINC-653 Part 3 test specifications during test suite development. The main problem with these test cases specifications is that many checks do not correspond to the ARINC-652 Part 1 requirements and sometimes violate them. Apparently, the test specification was not adapted to modifications in API functional requirements specification. As a result, the test specification became outdated and detached from the RTOS API specification. To overcome this problem, the test suite was improved and significantly extended with:

- test cases for combinations of some requirements left uncovered in ARINC-653 Part 3 (these additional test cases detected errors in the target RTOS unable to be detected by the initial tests specified in ARINC-653 Part 3);
- test cases for functionality of Interpartiton Communication interfaces in the system partitions supporting POSIX API;
- system test cases for Interpartition Communication, Intrapartition Communication and Health Monitoring subsystems.

The resulting test suite includes over 370 test cases covering all aspects of functionality of individual interfaces as well as the functionality of subsystems as a whole. The test suite is implemented in C programming language. The structure of the test suite is as defined in the ARINC-653 Part 3 and includes the following layers:

- service macros;
- individual tests;
- test sequence.

For each function (service) of the API under testing, one procedure is implemented to check the correctness of this API function execution on one set of input parameters by comparing the results of the target function execution with the expected ones. This procedure is called "service macro". The layer of individual tests is implemented as a set of files with the names corresponding to the test identifiers in ARINC-653 Part 3. The test sequence layer is implemented as a test script and supports test execution in any order using system configuration individually developed for each test. This approach avoids possible interference (mostly temporal) of tests during test execution.

The developed test suite was used and approved in the OS2000 [4] RTOS testing project. It detected several errors in the target system which could not be detected with the initial test suite specified in ARINC-653 Part 3.

3. Conformance testing for POSIX standard.

Another standard widely used in RTOS for embedded avionics systems is POSIX (Portable Operating System interface for unIX) [2]. It defines portable API sources of operating systems. The main specification was developed as IEEE 1003.1 specification and became the international standard ISO/IEC 9945-1:1990. Three POSIX standards are of the most interest for RTOS for embedded avionics systems: 1003.1a (OS Definition), 1003.1b (Realtime Extensions) и 1003.1c (Threads).

ISP RAS has developed the test suite for conformance testing of RTOS against POSIX requirements. Comparing to existing test suites for POSIX (for example, Open Group certification test suite and the Open POSIX Test Suite project results), this test suite has the following features and advantages:

- **requirements formalization**: functional requirements to the target system behaviour extracted from the standard are represented as contract specifications;
- **automatic test generation**: tests or sequences of test stimuli are generated during test scenario execution that checks functional requirements and ensures the test coverage according to the given coverage criterion;
- using of a **"test agent" feature** to facilitate test suite execution on target platforms with limited resources (ex., embedded platforms).

Automatic test generation from contract specifications implemented in the ISP RAS test suite for POSIX gives more control over the test suite. It allows specifying functional requirements extracted from the standard apart from test data and checking procedures for these requirements. This feature significantly simplifies modification of the test suite when adapting to evolving standard or to the requirements of specific application domain.

Requirements catalogue for POSIX now contains over 10,000 requirements extracted from the text of the standard. These requirements for 916 POSIX functions are formalized in the contact specifications of the total size of about 60,000 lines. The test suite contains 172 test scenarios for testing these POSIX functions and subsystems.

The ISP RAS test suite for POSIX was used and approved in the OLVER (Open Linux VERification) project [5]. In this project, various distributions of OS Linux were tested with the developed test suite for POSIX. The code coverage obtained during this testing is close to results of the debug test suite for Linux glibc library created by the library developers with detailed knowledge of the library implementation, but not aimed at conformance testing against any particular standards. For some groups of functions, the ISP RAS test suite for POSIX provides better code coverage than other existing test suites for functional conformance testing.

4. Testing process support for DO-178B standard.

The DO-178B standard "Software Consideration in Airborne Systems and Equipment Certification" [3] developed by RTCA (Radio Technical Commission for Aeronautics) defines the requirements to software development and certification process for on-board avionics systems. Europe has adopted the ED-12B standard, which is similar to DO-178B and supported by EUROCAE (The European organisation for civil aviation equipment). In 2003 Russia has adopted the GOST R 51904-2002 standard "Software for embedded systems: general requirements to development and documenting", which is also similar to DO-178B. None safety-critical computer system becomes airborne unless all rigid requirements of this standard are satisfied.

Certification methodology defined in DO-178B demands the proven quality of the developed software solutions from software vendors. As a method for providing such a proof, the standard suggests the vendors to develop a requirements-based test suite with the following features:

- test suite contains test cases for each software requirement;
- test cases are traced to requirements and vice versa;
- test cases satisfy the criteria of normal and robustness testing.

To support these works of software testing and test suites development, ISP RAS has developed the UniTESK technology [6] and the FOREST (FOrmal REquirements Specification and Testing) process [7]. The main objectives of this approach are:

- to create formal specification of software requirements;
- to develop on its basis the test suite for testing the software functionality against these requirements.

The process of test suite development includes four stages. The results of each stage are used at the later stages, but also can be used separately.

Stage 1. Requirements are extracted from initial documents (ARINC-653 Part 1 and 2, POSIX etc.) and systematized. This work results in creation of the *requirements catalogue*. In this catalogue, the requirements are unambiguously formulated, categorized, provided with unique identifiers and possibly linked together with appropriate relations. The requirements catalogue is used at the later stages and provides means for futher estimation of test coverage and completeness of software implementation in terms of the initial documents (text of standard).

Stage 2. Requirements analysis is performed. Conceptual models of the requirements are the main means for such analysis. When developing and analysing the conceptual models, the application domain knowledge is restructured and various model properties (adequacy, completeness, etc.) are checked. The model development is based on the requirements catalogue created at the first stage.

Stage 3. Formal representation of requirements. The mathematical formalism of contract specifications is used to formally represent the requirements form the catalogue. The restructuring and analysis of application domain knowledge at the previous two stages significantly facilitates the conversion of the requirements from textual form into formal representation.

Stage 4. Test development using the contract specifications. At this stage, test scenarios are developed for checking all requirements from the catalogue formalized in the contract specifications. Test scenarios are written in the specification language and are executed with the UniTESK tool. The results of test execution are summarized in the test report presenting the exhaustiveness of testing according to the given coverage criteria and possible violations of requirements from the catalogue.

Use of this approach allows creation of test suite with clear traceability of test cases to requirements formulated in the text of the standard. It can significantly simplify the identification and

modification of tests requiring redesign after possible modification of initial requirements. As a result, the UniTESK technology and the FOREST process provide framework for software testing compliant with the rigid requirements of DO-178B standard and hence simplify software certification according to this standard.

5. References.

1. ARINC. ARINC Specification 653-2: Avionics Application Software Standard Interface Part 1 - Required Services. Aeronautical Radio INC, Maryland, USA, 2005.
2. IEEE Std 1003.1, 2004 Edition. The Open Group Technical Standard. Base Specification, Issue 6.
3. RTCA/DO-178B, "Software Considerations in Airborne Systems and Equipment Certification". http://www.rtca.org
4. OS2000. http://en.wikipedia.org/wiki/OS2000
5. http://www.linuxtesting.org
6. http://www.unitesk.com
7. V.V.Kuliamin, N.V.Pakulin, O.L.Petrenko, A.A.Sortov, A.V.Khoroshilov "Requirements formalization in practice". Preprint N.13, ISP RAS, 2006 (in Russian).