A Family of Testbenches to Support Testing of Real-Time Avionics Systems

V.V. Balashov, M.V. Chistolinov** and R.L. Smeliansky ****
** Department of Computational Mathematics and Cybernetics, Lomonosov Moscow State University*
Leninskie Gory, MSU, 1, Bldg. 52, Room 764, Moscow, Russia, hbd@cs.msu.su
*** Department of Computational Mathematics and Cybernetics, Lomonosov Moscow State University*
Leninskie Gory, MSU, 1, Bldg. 52, Room 764, Moscow, Russia, mike@cs.msu.su
**** Department of Computational Mathematics and Cybernetics, Lomonosov Moscow State University*
Leninskie Gory, MSU, 1, Bldg. 52, Room 764, Moscow, Russia, smel@cs.msu.su

Abstract
Systematic testing is performed on different phases of real-time avionics (RTA) system development to ensure fulfilment of requirements to the RTA system. In this paper we focus on the problems of testing of an RTA system as a hardware/software system, involving the software running on the target hardware. A testing toolset and a family of testbenches to support solving of these problems are presented, along with methodological aspects of their application on appropriate RTA system development phases.

1. Introduction

Real-time avionics (RTA) systems are subject to strict requirements to functionality, dependability and real-time operation. To ensure fulfillment of these requirements, systematic testing of the RTA system is performed on different phases of RTA system development.
Figure 1 shows a typical V-model of the RTA system's software lifecycle. Testing activities are performed on the phases shown on the right side of the "V" shape. While unit testing of RTA system software can be performed on instrumental computers, integration and acceptance testing, as well as testing of series-produced RTA systems, necessarily involve the real target hardware.

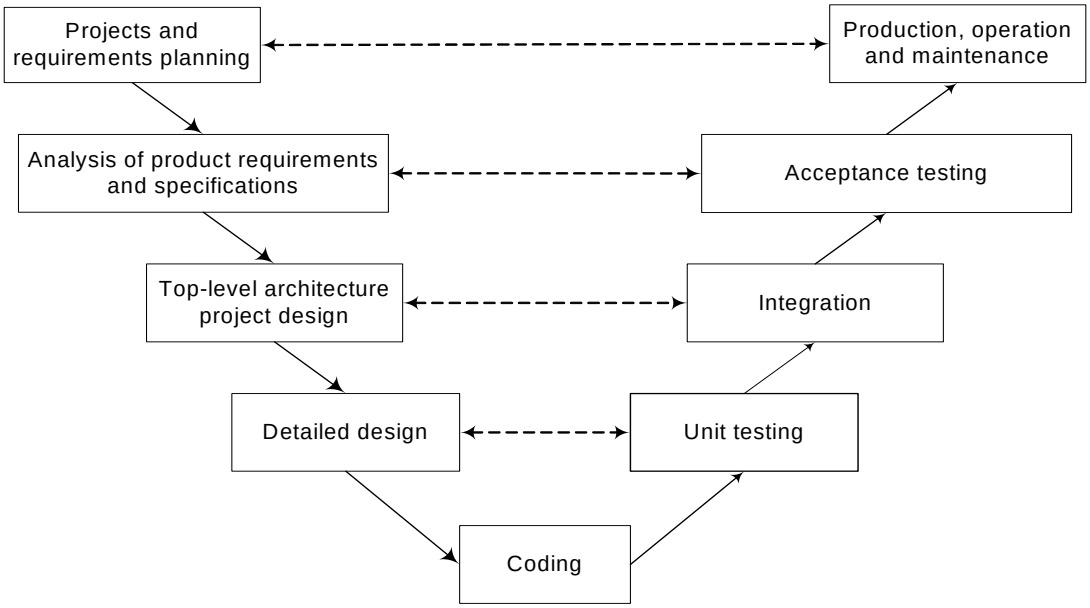


Figure 1: The RTA system's software lifecycle V-model

In particular, the target hardware is required to test the requirements to real-time operation, to data exchange through the onboard channels, and to interaction of different software layers (OS, service software, application software).
General-purpose functional testing tools, such as Rational Test RealTime [1] or VectorCAST [2], require instrumentation of the RTA system's software to support testing the application software on the target hardware. Instrumentation includes loading of auxiliary software modules to the target hardware, which is incompatible with integration testing and acceptance testing. These tools do not support analysis of data exchange through onboard channels by the testing scripts. Consequently, application of Rational Test RealTime, VectorCAST or similar tools to RTA systems testing requires significant customization of the tools, which is hindered by the fact that their source code and internal interfaces are closed.
In this paper we present a toolset for functional testing of RTA systems (farther referred to as the FT toolset). The FT toolset is aimed at testing without target instrumentation. This toolset is developed in the Computer Systems Laboratory (CS Lab) of Computational Mathematics and Cybernetics Department of Lomonosov Moscow State University. Testbenches based on the FT toolset are utilized by Sukhoi Design Bureau for testing of modern aircraft RTA systems. These systems consist of multiple devices (computational nodes, sensors, actuators, indicators) connected by dozens of data transfer channels.
The rest of the paper is structured as follows. Section 2 highlights the main RTA system testing problems occurring on different phases of the RTA system's lifecycle. In Section 3, according to these problems, the requirements to RTA systems functional testing tools are stated. Section 4 analyzes applicability of two general purpose functional testing toolsets to the task of target hardware-based testing of RTA systems and emphasizes significant issues undermining this applicability. Section 5 presents the FT toolset, describes its architecture and features. Section 6 describes the typical architecture of a testbench based on this toolset. Section 7 describes the family of testbenches based on the presented toolset, each testbench aimed at a particular activity (or a group of activities) of the RTA system lifecycle. Section 8 presents a scheme for joint operation of these testbenches during RTA system development and maintenance. In the last section, future directions for development of the described technology are proposed.

2. Problems of RTA system testing on different lifecycle phases

Let us consider the main RTA system testing problems, specific for different phases of RTA system's lifecycle. Table 1 shows these problems for each phase and activity of the RTA system lifecycle on which the target hardware-based testing is performed.

Table 1: RTA system testing problems on different lifecycle phases

Activity	Phase	RTA system testing problems
1. Software integration with the target hardware, debugging of the software on the target platform (typically, an individual device of the RTA system)	Integration	Selective execution of tests according to the software functionality being debugged. Manual testing control to allow choosing the tests execution order and repetitive execution of tests.
2. Integration of subsystems of the RTA system. Integration of the whole RTA system.	Integration	Ensuring the correctness of integrated RTA devices' communication through the onboard interfaces.
3. Acceptance testing of the RTA system's software.	Acceptance testing	Automated execution of the full set of RTA system's functional tests. Checking the correctness of inter-device communication for all devices of the RTA system.

4. Acceptance testing of series-produced RTA devices and complete RTA system instances.	Acceptance testing, Production	Automated execution of the full set of RTA system's functional tests. Easy replacement of RTA devices during the change of the tested RTA system instance.
5. Diagnostics of RTA devices subject to reclamations.	Maintenance	Construction of a "reference" environment for the faulty device. Automated execution of test sets specific for a particular device.
6. Diagnostics of RTA devices onboard the aircraft.	Maintenance	Need for a mobile and compact hardware solution for testing.

Activities 1-6 represent a virtually complete set of testing activities on the integration, acceptance testing and production/operation/maintenance phases. For each of these activities, testing is performed in dedicated hardware/software environments known as testbenches. The testing technology, including instrumental software and testbench architecture, should be unified to reduce the complexity of testing and to provide reusability of test scripts through different phases of the RTA system lifecycle.
In this paper we do not consider unit testing of the RTA system's application software, as it is typical to perform unit testing on software developers' instrumental PCs without involving the target hardware.

3. Requirements to the tools for functional testing of RTA systems

The main type of software testing performed during the activities 1-6 (see Section 2) is functional testing of software running on the target RTA devices. Each target device is considered a "black box", inputs of which are the input onboard interfaces and outputs are the both output[1] onboard interfaces and visual indication facilities (if present). For more details on functional testing and black-box testing, see [3].
Sending of the test data to the RTA devices and obtaining the resulting output data is performed through onboard channels. During the workout of inter-device communication through the channels, it is necessary to verify the correctness of data exchange. So the testing scripts must have access to the data exchange monitoring results.
As the correctness of data indication on the onboard display devices is visually estimated, the testing tools must support interactive mode of operation in which the user is asked for confirmation of testing success.
It is reasonable to use both testing tools and data exchange monitoring tools on the testbench. The monitoring tools can be used to analyze "boundary" issues in which it is unclear whether a test has failed due to an error in the RTA device operation or in the test itself. An overview of several monitoring tools is provided in [4].
According to the present testing experience of Sukhoi design bureau, the following requirements to the tools for functional testing of RTA systems can be stated.
1. Support for functional testing of software on the target device without instrumentation of the device. The testing tools must support testing of software on the target device without loading of any auxiliary software modules to the device.
2. Support for onboard interface standards used in the RTA system. To perform full-scale testing of the RTA system as a hardware/software system, the testing tools must support all the onboard interface standards used in the system. Following main features are necessary:

- sending of the test data through the onboard interfaces and obtaining the responses for subsequent analysis;
- monitoring of data exchange through the onboard interface channels and providing access to monitoring results to the testing scripts;
- composition and analysis of low-level contents of the messages transferred through the channels, in order to test the devices which operate "raw" binary data without decomposition into individual parameters.

3. Support for testing of real-time constraints on the RTA system operation. To test these critical constraints on the RTA system operation (including the data exchange), the testing tools must support precise timing of the RTA system's responses.

[1] For certain types of interfaces, e.g. MIL STD-1553B, the same channel may operate as both input and output for the RTA device.

4. Support for testing of dependability features of the RTA system. Dependability testing requires simulation of faults of the devices connected to the RTA device or subsystem under test, including data exchange faults.
5. Support for multiple-computer testbench configurations. The number of onboard channels in an RTA system can reach several dozens or even hundreds. Multiple instrumental computers with onboard interface adapters must be involved to provide full-scale testing of the RTA system. The testing tools must support such multiple-computer configurations, in particular provide time-synchronized sending of data into channels from different instrumental computers, as well as synchronized checking of the RTA system's responses.
6. Support for sharing the onboard interface adapters with monitoring tools. This feature allows to avoid installation of extra adapters to simultaneously perform testing of the RTA system and monitoring of data exchange through channels in the system under test. It is important both for large testbenches with dozens of adapters and for mobile testing and monitoring workstations, with a limited number of adapters (no more than two in an industrial laptop).
7. Support for different testing modes:

- automatic testing (without user's intervention) – for checking the responses received from the RTA system through onboard channels;
- interactive testing (the user replies to requests from the tests) – for checking the data indicated on the display devices of the RTA system.

8. Support for online visualization of the testing process. This feature allows the user to track the current state of testing and to react to possible deviations. Following information is subject to online visualization:

- values of test data and the data received from the system under test;
- testing logs.

9. Support for requirements traceability and generation of reports on the testing results. Functional testing is an integral part of the RTA system lifecycle and is aimed at verification of requirements to the RTA system. Consequently, the testing tools must support:

- binding the requirements to the test cases;
- generation of the requirements traceability matrix;
- generation of reports on the testing results and on fulfilment of requirements.

10. Integration with RTA system software development tools, such as:

- version control tools (for keeping the testing scripts for a particular RTA system's software version together with the software source code for this version);
- requirements management tools.

11. Integration with onboard interfaces database. This feature allows the testing tools to generate the interface part of the testing scripts, in particular the specification of the set of sent/received messages and structure of their data fields, based on the information from the database. The onboard interfaces database contains the description of data exchange protocols for the RTA system.
12. Support for operation on a mobile platform. To create a mobile testing workstation based on the testing tools it is required that both the user interface and the test execution environment run on a common hardware/software platform.

4. Applicability of general-purpose functional testing tools to testing of RTA systems

In this section we consider two existing functional testing toolsets and estimate their applicability to the task of testing the RTA system software on the target hardware without target instrumentation. The toolsets, namely Rational Test RealTime and VectorCAST, are not "purely" general purpose. They are designed to support testing of distributed real-time systems, however there are several common issues that substantially complicate their application to the above mentioned task.
Rational Test RealTime (RTRT) [1] from IBM is a cross-platform solution for testing real-time applications, including networked and embedded software. RTRT provides facilities for automated target-based software testing, memory leaks detection and code coverage analysis. RTRT is integrated with revision control and configuration management tools from IBM (ClearCase, ClearQuest).
RTRT was originally developed for unit testing. System testing support for distributed applications was implemented later. Target-based testing requires loading auxiliary modules (system testing agents) to every device (computer) of the target system which runs software to be tested. Agents are responsible for supplying test data to the software under test, reading the resulting data and communicating with the testing control computer.
RTRT supports two test description languages: Component Testing Script Language for unit tests and System Testing Script Language for system tests.
Conventional (recommended by vendor) technology of RTRT application to target-based testing of avionics software requires loading auxiliary software modules to the target devices. This approach is not suitable for acceptance and

field testing. It is also questionable whether the auxiliary modules significantly affect real-time characteristics of target software operation.
RTRT was applied in Sukhoi company for testing of Sukhoi Superjet civilian aircraft avionics. This application involved custom “adaptation layer” software running on dedicated computers and translating test data from RTRT to onboard channel messages sent to the (non-instrumented) target system. Development of such intermediate software is a complex task which requires access to internal details of RTRT implementation.
Testing solution based on RTRT with adaptation layer provides communication with the target system on “parameter” level, where parameter is a variable taken by the target application software as an input or generated as an output. This level is not suitable for testing devices that take or generate “binary” data (e.g. video frames, digital maps etc). It is also hardly suitable for dependability testing which requires low-level stimulation of the target system (e.g. injection of communication faults into channels or simulation of power failures).
VectorCAST [2] from Vector Software is an integrated toolset for unit and functional testing. It supports test generation, execution, code coverage analysis and regression testing. VectorCAST can be integrated with third-party requirement management tools through VectorCAST / Requirement Gateway module.
VectorCAST provides special features for avionics software testing. In particular, it implements recommendations of DO-178B standard which describes avionics software development processes. VectorCAST was applied to development of avionics software for several aircraft including JSF, A380, Boeing 777, A400.
VectorCAST supports unit and integration testing of avionics software on target platform. Similarly to RTRT, VectorCAST requires loading auxiliary software modules (VectorCAST/RSP, Runtime Support Package) to target devices. This raises the same issues as mentioned above for RTRT.
There is no open information available on existence of “adaptation layer”-like software for VectorCAST which could allow application of VectorCAST to testing of non-instrumented RTA systems through onboard channels. Development of such software layer is problematic as VectorCAST is a closed commercial product, and even if such layer existed, the resulting solution would have same limitations as noted above for the RTRT-based solution.
Both testing toolsets reviewed in this section provide a rich feature set, but have similarly limited applicability to the task of testing of non-instrumented RTA systems.
Table 2 provides a comparison of RTRT, VectorCast and the FT toolset according to the criteria (requirements) introduced in Section 3.

Table 2: Functional testing toolsets’ features according to the requirements from Section 3

Requirement	**FT toolset**	**Rational Test RealTime**	**VectorCAST**
Support for functional testing of software on the target device without instrumentation of the device	+ (testing is performed through onboard interface channels)	-/+ (development of modules not included in the delivery package of the tool is required)	-/+ (development of modules not included in the delivery package of the tool is required)
Support for onboard interface standards used in the RTA system	+ (support for all onboard interface standards used in RTA systems of the aircrafts to which the toolset is applied: Su-35, T-50)	-/+ (development of modules not included in the delivery package of the tool is required)	-/+ (development of modules not included in the delivery package of the tool is required)
Support for testing of real-time constraints on the RTA system operation	+	-	+/- (loading of auxiliary software modules on the target system is required)
Support for testing of dependability features of the RTA system	+	-/+	-/+

Support for multiple-computer testbench configurations	+	-	-
Support for sharing the onboard interface adapters with monitoring tools	+ (cooperatively with tools from the "Channel analyzer" family [4])	-/+ (adaptation is possible, but is not available out-of-the-box)	-/+ (adaptation is possible, but is not available out-of-the-box)
Support for different testing modes	+	+	+
Support for online visualization of the testing process	+ (online visualization for: testing logs; test data and responses from the tested system, in table and graphical forms)	-/+ (use of external tools is required)	-/+ (use of external tools is required)
Support for requirements traceability and generation of reports on the testing results	+/- (binding of requirements to the test cases; traceability matrix generation; generation of reports from testing results)	-/+ (traceability – via external tools, detailed reports – only for unit testing)	-/+ (traceability – via external tools, detailed reports – only for unit testing)
Integration with RTA system software development tools	+/-	+/-	-/+
Integration with onboard interfaces database	+/- (generation of the interface part of testing scripts from the DB)	-	-
Support for operation on a mobile platform	+	-/+	-/+

5. The toolset for functional testing of RTA systems

In this section we describe the toolset for functional testing of RTA systems (FT toolset) utilized by Sukhoi Design Bureau for testing of modern aircraft RTA systems. This toolset is developed in the Computer Systems Laboratory (CS Lab) of Computational Mathematics and Cybernetics Department of Lomonosov Moscow State University.

5.1 Toolset overview

In contrast to the Rational Test RealTime and VectorCast toolsets considered above, the FT toolset is aimed at testing of RTA system through onboard interface channels without loading any auxiliary software to the RTA devices. The FT toolset supports such types of onboard channels as MIL STD-1553B, ARINC 429, Fibre Channel and some other types used in modern RTA systems. Channel adapters are installed in the instrumental computers which execute the testing scripts.
For every type of onboard channel, the FT toolset supports:

- preparation and sending of test data to the RTA system through onboard interface channels (both packing parameters into messages and composition of "raw" messages are supported);

- receiving data from the RTA system through the channels for subsequent analysis (both unpacking parameters from messages and access to "raw" messages are supported);
- monitoring of data exchange through the channels between RTA devices and processing of monitoring results by the testing scripts.

The FT toolset can be extended to support new types of onboard interfaces. Testbenches based on the FT toolset support testing of RTA systems during all activities mentioned in Table 1.

The FT toolset supports distributed execution of tests on multiple-computer testbench configurations, which is necessary for testing complex RTA systems with large number of interfaces. Instrumental computers of the FT toolset-based testbenches operate in synchronized time and perform coordinated sending of test data and processing of RTA system's responses. Time synchronization is guaranteed by periodical sending of precise time signals from a dedicated instrumental computer (synchronization master) to all other instrumental computers.

For operation with high-speed channels, for instance Fibre Channel-based video channels, it is necessary to use performance-optimized hardware. The FT toolset supports integration with customized signal simulation hardware (SSH) systems which are parts of the testbenches (an example of SSH system is provided in Section 6). SSH systems operate under control from the testing scripts running on the instrumental computers.

The FT toolset supports both fully automatic execution of tests (including batch mode) and interactive testing. Interactive features include:

- generation of requests to the user, e.g. "yes/no" request necessary for visual confirmation of displayed data correctness, or request for text comments to the user's positive or negative response;
- support for manual input of test data by the user during the testing process;
- support for manual selection of tests execution order by the user (needed e.g. for debugging of RTA system's software on the testbench).

During every testing session, a testing log is generated which contains the completion status (success/failure) of every executed test, user's responses to interactive requests, etc. The log is automatically processed to determine which requirements were successfully verified (binding between requirements and test cases is defined within the testing script). During the testing process, the testing logs, values of test data and response data are displayed in the online visualization tool which allows the user to track the testing progress. Custom visualization formats are supported, such as dials, sliders etc; plugin interface is provided for adding new custom views.

The FT toolset also supports recording of user's actions (such as entering of test data, responses to tests' requests, altering the tests execution order) for later replay. The record and replay option is useful for debugging of interactive testing scripts.

To support development of testing scripts in advance, prior to actual availability of RTA system devices to be tested, the FT toolset implements:

- software simulated "virtual" onboard interfaces, including MIL STD-1553B and ARINC 429;
- execution of testing scripts in user input expectation mode, in which the tests request data input from the user instead of receiving the data from the (unavailable) RTA system devices; this mode does not require modification of tests.

The FT toolset supports development and execution of simulation models for the RTA devices, based on hardware-in-the-loop simulation technology described in [5]. Simulation models utilize the resources of instrumental computers, e.g. onboard interface adapters, to imitate the data exchange of the simulated device through the onboard interface channels. Simulation models of the RTA devices can be used for testing the reconfiguration features of the RTA system in case some of the devices participating in the reconfiguration procedures are not available in hardware.

The hardware resources of an FT toolset-based testbench can be shared with monitoring tools from the "Channel analyzer" family [4]. Concurrent operation of FT tools and monitoring tools in a common hardware/software environment allows to create a compact solution for testing and monitoring of an RTA system. Such solution can be used for onboard diagnostics of RTA systems (see Section 7).

Tests development subsystem of the FT toolset supports automatic generation of tests' interface parts from the onboard interfaces database. The database is filled for every version of the RTA system's software and is used for informational coupling of the devices within the RTA system [6].

5.2 Test description language features

Test description language (TDL) implemented in the FT toolset is intended for specification of RTA system testing scripts. The scripts written in TDL are executed on the instrumental computers and interact with RTA devices through onboard interface channels. Testing scripts also control the operation of SSH systems, including the data exchange between these systems and the RTA system.

TDL is an extension of the C language and provides language features for describing the structure of the test set, binding of tests execution to physical time, control of data exchange through the onboard channels, and general control of the testing process.
Basic TDL unit for test coupling is the *test component* (TC). TC source code consists of a *header* and a *body*.
TC header specifies:

- Set, structure and hierarchical naming of test cases;
- Correspondence between test cases and requirements to the RTA system;
- Set and types of interfaces to onboard channels through which communication with the RTA system is performed during testing;
- Structure of data words and messages transferred and received through interfaces;
- Set and types of auxiliary variables (parameters) intended for data exchange between different TCs and between TC and SSH systems;
- Set of testing logs recorded by the TC.

TC body specified testing scripts' activity on:

- Preparing the test data;
- Sending the test data into channels;
- Receiving data from the RTA devices through channels and checking the tested conditions (access to channel monitoring results is transparently provided to the tests);
- Direct control of channel adapters, including turning on/off, setting service flag values, fault injection;
- Interaction with the user during human-assisted (e.g. visual) checking of tested conditions;
- Control for generation of testing logs.

TDL provides following functionality for automatic checking of timing constraints on responses from the RTA devices (i.e. for testing the RTA system's real-time characteristics):

- Waiting for a specified duration and then checking the condition on received data;
- Waiting for the condition on received data to become true, until a specified timeout expires;
- Constantly checking that the condition on received data remains true during a specified duration.

A TDL project may include several TCs, in particular intended for execution on different instrumental computers. Even if only one TC is used, it can access channel adapters located on different instrumental computers. It is useful in case a single TC implements scripts for comprehensive testing of the RTA system or its subsystem through a large set of interfaces attached to different instrumental computers.

5.3 Software structure of the FT toolset

The FT toolset software contains the following subsystems:

1. Tests development subsystem, which supports editing of TDL source code of testing scripts.
2. Testbench configuration tools which support:
   - Binding of TCs to instrumental computers;
   - Binding of TC interfaces to onboard interface adapters within the instrumental computers;
   - Specification of data links between tests and SSH systems;
   - Specification of detail level for testing events recording.
3. Real-time test execution environment which provides:
   - Distributed execution of TCs on several instrumental computers;
   - Data exchange and time synchronization between the instrumental computers;
   - Remote access for the TCs to the onboard interface adapters on different instrumental computers through the testbench Ethernet network;
   - Data exchange between tests and SSH systems;
   - Data exchange between tests and the tested RTA devices;
   - Access of tests to the channel monitoring results;
   - Interaction with the experiment control subsystem;
   - Recording of the testing results in the form of testing logs and testing events traces.
4. Experiment control subsystem which supports user's interaction with tests, in particular for interactive testing.
5. Subsystem for online visualization of the testing process and results, which is closely integrated with the experiment control subsystem.
6. Experiment results processing subsystem which supports:
   - Display of the testing results in the form of testing logs, time diagrams, parameter value graphs, channel monitoring logs;
   - Generation of reports on the testing results.

7. Server subsystem which supports:
   - Version control for the testing scripts' source code, testbench configurations, testing logs;
   - Interaction with the onboard interfaces database;
   - Remote access to the testbench repository from all testbench computers.

Access to all the listed subsystems is provided to the user through the Integrated development environment.

6. Architecture of testbenches based on the FT toolset

The FT toolset described in Section 5 is used in Sukhoi design bureau for over seven years as the main software tool for testing of RTA systems. Architecture of FT toolset-based testbenches, including typical components and scheme of their composition, evolved through years of toolset industrial application. The present section describes this architecture.

An FT toolset-based testbench typically includes the following components:

- Instrumental computers intended for running tests and performing data exchange with the RTA system through onboard interface channels (presently, MIL STD-1553B, ARINC 429 and Fibre Channel);
- Specialized signal simulation hardware (SSH) systems operating under control of instrumental computers;
- Workstations for test engineers and RTA software developers;
- Server responsible for centralized functions such as testbench repository keeping and sharing, user authentication, etc;
- Network of onboard interface channels connecting RTA devices to instrumental machines and SSH systems, and RTA devices with each other;
- Auxiliary equipment, including systems for power supply and cooling, technological networks, racks, etc.

Structural scheme of a typical FT toolset-based testbench is shown in Figure1.

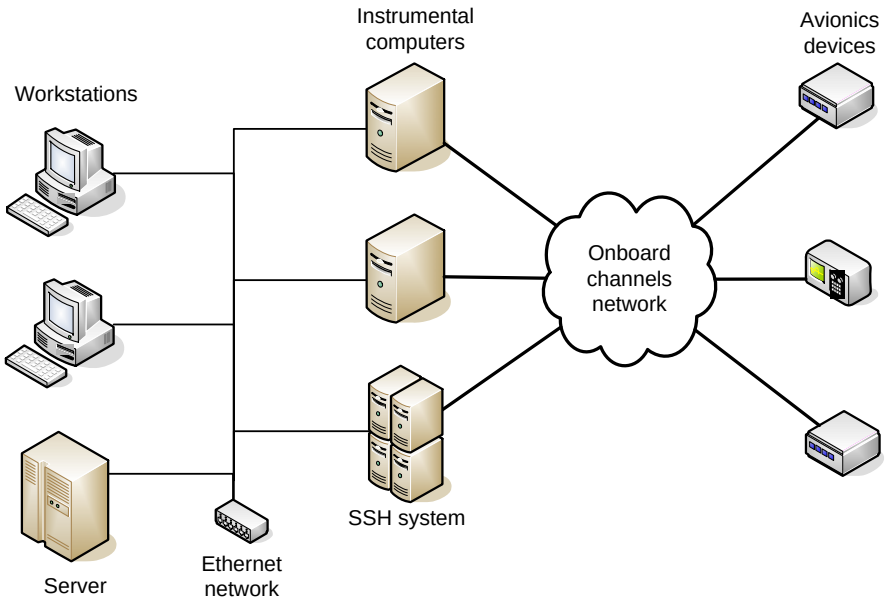


Figure 1: Structural scheme of an FT toolset-based testbench

The number of instrumental computers, workstations, number and types of the onboard interface channels are defined by the specifics of the RTA system under test, as well as by the tasks of the particular testbench.

All computers of the testbench (workstations, server, instrumental computers, SSH systems) run the Linux operating system. In particular, instrumental computers and SSH systems run Linux with real-time extensions.

In addition to the Ethernet network, the instrumental computers are connected by the time synchronization network. It connects LPT ports of the computers and transfers precise time signals from the "master" computer to the other computers. Time synchronization protocol guarantees deviation between computers' clocks of no greater than 10 us.

An example of SSH system is the television signal simulation hardware (SSH TV). It is a multiple-computer system intended for generating, sending, receiving and recording high-speed video streams simultaneously through several optical channels. SSH TV operates as a source of test video data for the RTA system video processing devices, as well as a recorder for output of these devices. SSH TV is also utilized for testing the onboard display devices.

Static (individual frames) and dynamic (video sequences) test streams are prepared in advance, prior to the start of testing. Selection of the video stream to be sent to a particular channel at the given time is performed by the testing script running on an instrumental computer, or by the user through the online experiment control tool.

7. The family of testbenches for RTA systems testing

This section describes the industrial application of the FT toolset presented above. A family of FT toolset-based testbenches was developed for Sukhoi, each of the testbenches aimed at a particular activity (or a group of activities) of the RTA system lifecycle. Complexity of a testbench corresponds to the complexity of the RTA subsystem to be tested on the testbench.

Testbenches used in Sukhoi are integral parts of the technology for data control system (DCS) development. DCS includes the central computer, a set of display devices, and some auxiliary devices. DCS onboard channel network includes:

- Several MIL STD-1553B channels connecting the DCS to other RTA devices;
- Dozens of ARINC 429 channels for data exchange with legacy devices;
- A Fibre channel network for high speed data exchange, including transfer of video data.

Main representatives of the testbench family, as well as their tasks, are described below. All the testbenches share the common architecture described in Section 6. As a total, the family of testbenches covers all the testing activities mentioned in Section 2.

1. Testbench for testing and debugging of an individual RTA device's software. This testbench is intended for working with a single RTA device, for instance with the most complex one – the central computer. The main purpose of the testbench is debugging the RTA device's software on the target hardware, as well as preparation of the device to integration with other RTA devices. For final checks of the device's readiness for integration with other devices, an approved set of tests should be used.

Due to the necessity to support only the interfaces provided by the particular RTA device, the number of instrumental computers in this testbench is usually small, and SSH systems are implemented in single-computer configurations. This enables composition of all testbench hardware, together with the RTA device, into a single rack. Sharing of hardware resources between testing and monitoring tools helps to keep hardware requirements low.

A "unified" testbench for working with different RTA devices from a specified set is organized in a similar way. Instrumental computers of such testbench cover all types of external interfaces provided by devices from the set, in numbers sufficient for working with every single device from this set.

The testbench can be used for testing and debugging of software on a partially equipped RTA device, for instance a central computer with only some of processor and communication modules installed. This approach is reasonable for development of particular subsystems of the RTA device software.

2. Testbench for testing and debugging the software of several connected RTA devices. This testbench is intended for workout of communication between two or several directly connected RTA devices, for instance the central computer and a display device. The testbench can also be used for performing some activities of debugging the software of these devices. This testbench is similar by hardware structure to the one described above.

Monitoring tools are used on this testbench to explore the communication between RTA devices comprising the RTA subsystem under test.

As in the previous case, it is possible to create a testbench intended for workout of different sets of RTA devices. The central computer and one of several display devices from the DCS may serve as examples of such sets.

Working out the inter-device communication between devices of an RTA subsystem is an intermediate step to integration of the whole RTA system. Use of this testbench relieves the "main" RTA system integration testbench from tasks of detecting and fixing the issues of co-operation between most intensely communicating devices.

3. Testbench for RTA system integration and acceptance testing. This testbench is intended for stepwise integration of the whole RTA system, as well as for acceptance testing of the system. The testbench supplies test data to all external input interfaces of the RTA system and obtains the responses from all external output interfaces. In addition, monitoring of all inter-device channels must be supported.

As it is required to work out the complete inter-device communication protocols, the testbench includes a large number of instrumental computers (5-6 and more, depending on the RTA system characteristics) and full scale multiple-computer SSH systems.

During the acceptance testing of an RTA system software version, an approved test set for this version is used.

4. Testbench for development of testing scripts. This is an auxiliary testbench used solely for test development purposes. The testbench contains several workstations and a server connected by Ethernet network.

Debugging of tests is performed on workstations using such features of the FT toolset as support for virtual onboard interfaces and tests execution in user input expectation mode.

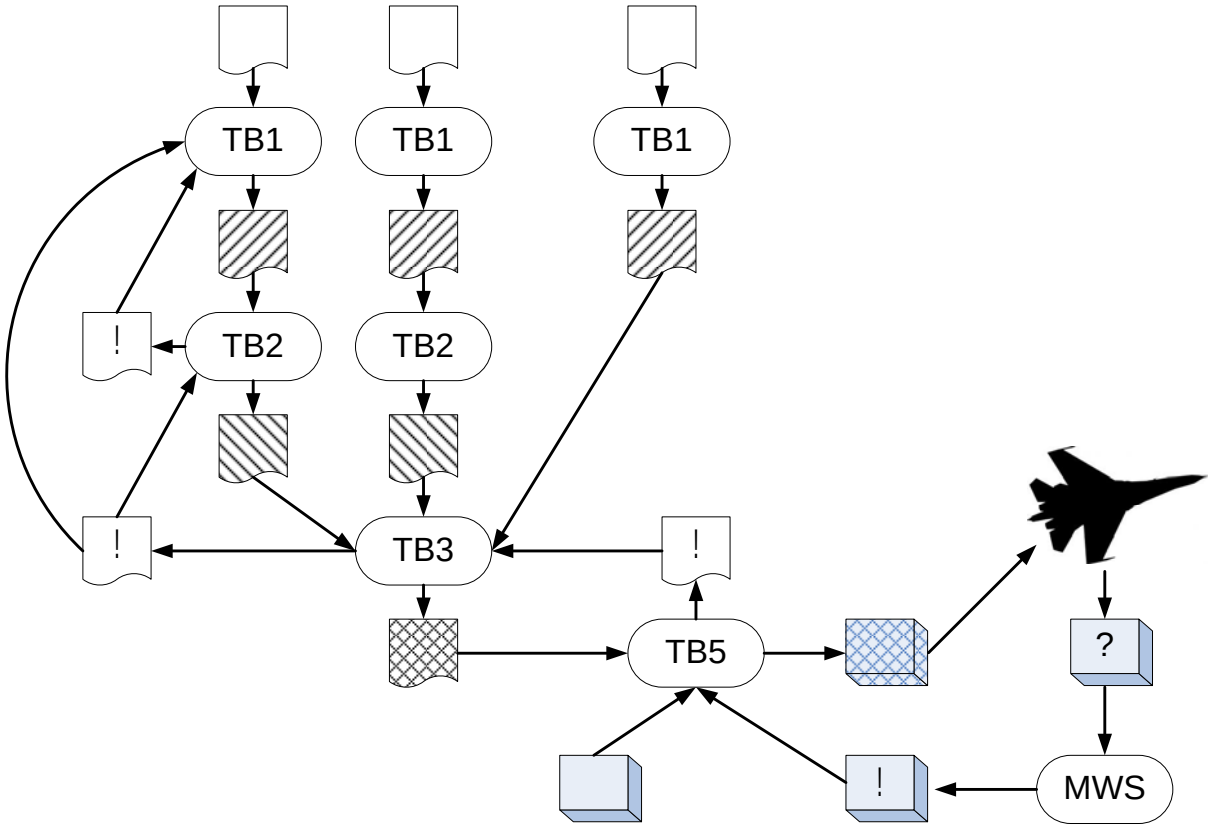
5. Testbench for acceptance testing of series-produced RTA systems. This testbench is intended for performing acceptance tests for RTA system sets before their installation onboard the aircraft. Second purpose of this testbench is diagnostics of RTA devices which are subject to reclamations. The instrumental hardware of this testbench is essentially the same as for the testbench for RTA system integration and acceptance testing.
The testbench includes a completely tested "reference" set of RTA devices. On arrival of a device subject to reclamations, this device replaces the corresponding reference device, and a batch of tests is automatically executed to examine the device under question. Decision on farther operations with the device is made on the base of testing results.
When a new, e.g. recently produced, set of RTA devices arrives for testing, it completely replaces the "reference" set. If some issues are detected in operation of the new RTA system set, some of the new devices can be replaced with the reference ones to help localization of the faulty device.
6. Mobile workstation for monitoring and testing of RTA devices onboard the aircraft. Mobile workstation is the most compact installation of the FT toolset and "Channel analyzer" tools for data exchange monitoring. Workstation hardware includes:

- A rugged industrial notebook with extension chassis providing PCI and/or PCI Express slots;
- Adapters of onboard interface channels (e.g. MIL STD-1553B, ARINC 429, Fibre Channel).

Mobile testing and monitoring workstation enables primary diagnostics of RTA devices operation without their transportation to the location of stationary testbenches. This possibility is essential in case of multiple geographically separated locations in which the aircraft containing the RTA devices are operated or field tested. If there are issues in operation of the RTA system "as a whole", use of mobile testing and monitoring workstations allows localization of the problem and its attribution to a specific RTA device. This single device becomes the subject for farther examination on a stationary testbench.

8. Joint operation of the testbenches

This section presents a typical scheme for joint operation of the testbenches described in Section 7. Testbench types are identified by numbers according to items of Section 7. Testbenches of type 4 are not shown on the scheme, as such testbenches are used for development of testing scripts for any of the other testbench types.
It should be noted that "New software" in Figure 2 is actually a new software *version* developed on the basis of the previous version(s) and possibly containing only local modifications.



Legend

Software (SW)

New SW

Integrated & tested SW for a single RTA device

Integrated & tested SW for an RTA subsystem

Integrated & tested SW for the whole RTA system

SW with confirmed defects

Testbenches

TB1 — Testbench for testing and debugging of an individual RTA device's software

TB2 — Testbench for testing and debugging the software of several connected RTA devices

TB3 — Testbench for RTA system integration and acceptance testing

TB5 — Testbench for acceptance testing of series-produced RTA systems

MWS — Mobile workstation for monitoring and testing of RTA devices

Hardware (HW)

New series-produced hardware of RTA devices

HW/SW integrated & tested RTA system

? — RTA system suspected of defects.

! — RTA device subject to reclamations

Figure 2: Scheme of joint operation of testbenches

9. Conclusion

In this paper we addressed the problems of functional testing of RTA systems occurring on different phases of RTA system lifecycle and requiring the involvement of the target system's hardware. A functional testing toolset for solving these problems was presented along with architecture of testbenches based on this toolset.
The presented FT toolset is aimed at testing of RTA systems without their instrumentation, in particular without loading any auxiliary software on the RTA devices. Support for this kind of testing is necessary to perform acceptance and field testing of RTA systems. A brief overview of two general purpose toolsets for functional testing of real-time systems indicated that capabilities of these toolsets for testing of non-instrumented RTA systems are significantly limited.
A family of FT toolset-based testbenches was also described in the paper. Testbenches from this family are utilized by Sukhoi Design Bureau in the process of development and maintenance of aircraft data control systems.
Following directions for future development of the FT toolset and the related testing technology should be noted:

- Support for declarative description of test cases, as an alternative to procedural description in Test description language; an example of declarative description is the table of test cases, each row of which contains the values of test data, delay for their processing in the RTA system, and conditions to be checked on the responses from the RTA system;
- Support for use of a single description of onboard interfaces' messages structure for all test projects created for a particular version of the RTA system software;
- Automatic binding of test components' interfaces to adapters within the instrumental computers, to minimize the adaptation of testing configurations after transfer of test projects between testbenches;
- Automatic analysis of the data exchange sequences recorded on the channels for compliance to the reference schedules from the onboard interfaces database;
- Automatic testing of indication formats of the RTA system's display devices; such devices typically send the "archive" copies of displayed images through onboard channels; to automate testing of indication formats, it is necessary to implement tools for automatic comparison of these archive images with reference images;
- Support for open interfaces for integration of the testbench with external systems for RTA devices workout, for instance the interface specified by the HLA standard [7].

References

[1] IBM. 2013. Embedded software test automation framework – IBM Rational Test RealTime. http://www-01.ibm.com/software/awdtools/test/realtime/. Accessed 26 May 2013.
[2] Vector Software. 2013. How to Improve Embedded Software Unit/Integration Testing with Automation. http://www.vectorcast.com/testing-solutions/unit-integration-embedded-software-testing.php. Accessed 26 May 2013.
[3] Culbertson, R., Brown, C., and Cobb, G. 2002. Rapid testing. Prentice Hall PTR.
[4] Balashov, V.V., Balakhanov, V.A., Bakhmurov, A.G., Chistolinov, M.V., Shestov, P.E., Smeliansky, R.L., and Youshchenko, N.V. 2011. Tools for monitoring of data exchange in real-time avionics systems. Proc. European Conference for Aero-Space Sciences (EUCASS).
[5] Balashov, V.V., Bakhmurov, A.G., Chistolinov, M.V., Smeliansky, R.L., Volkanov, D.Y., and Youshchenko, N.V. 2010. A hardware-in-the-loop simulation environment for real-time systems development and architecture evaluation. Int. J. Crit. Comput.-Based Syst. 1, No. 1/2/3:5–23
[6] Balashov, V.V., Balakhanov, V.A., Kostenko, V.A., Smeliansky, R.L., Kokarev, V.A., and Shestov, P.E. 2010. A technology for scheduling of data exchange over bus with centralized control in onboard avionics systems. Proc. IMechE Part G: J. Aerosp. Eng. 224, No. 9:993–1004.
[7] IEEE. 2010. IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) – Framework and Rules. doi: 10.1109/IEEESTD.2010.5553440.