COMPONENT TEST STRATEGY FOR COMS ON-BOARD SOFTWARE USING ATTOL

Su-Hyun PARK, Soo-Yeon KANG, Koon-Ho YANG, Seong-Bong CHOI

Korea Aerospace Research Institute, psh@kari.re.kr

ABSTRACT ... COMS (Communication Ocean Meteorological Satellite) is the geostationary satellite being developed by Korea Aerospace Research Institute for multi-mission: experimental communication, ocean monitoring and meteorological observations. The COMS operation is controlled by the on-board software running on the spacecraft central computer. The software is written in ADA language and developed under the software life cycle: Requirement analysis, Design, Implementation, Component test and Integration test. Most functional requirements are tested at component level on a software component testing tool, ATTOL. ATTOL provides a simple way to define the test cases and automates the test program generation, test execution and test analysis. When two or more verified components are put together, the integration test starts to check the non-functional requirements: real-time aspect, performance, the HW/SW compatibility and etc. This paper introduces the COMS on-board software and explains what to test and how to test the on-board software at component level using ATTOL.

KEY WORDS: COMS, On-Board Software, Component Test, ATTOL

1. INTRODUCTION

COMS (Communication, Ocean and Meteorological Satellite) is a geostationary satellite being developed by KARI (Korea Aerospace Research Institute). It is planned to launch in 2009 and designed for 7 years mission life. COMS carries three payloads: one for meteorological observation, another for ocean monitoring and the other for experimental communications. One of topmost geostationary satellite manufacturer, EADS-Astrium is the main contractor of KARI for the joint-development of COMS. COMS is based on Eurostar3000 (E3000) satellite of EADS-Astrium.

The COMS operation is controlled by the On-Board Software (OBS) running on the spacecraft central computer. Based on the E3000 heritage software, the COMS OBS is developed under the software life cycle: Requirement Analysis, Design, Implementation, Component test, Integration test. Both component test and integration test are essential for the software development of the satellite system. This paper focuses on the component level test of the COMS OBS.

This chapter introduces the software testing concept and a software component testing tool, ATTOL. Then, this paper describes the functions and design of the COMS OBS and explains the test strategy on the COMS OBS at component level using ATTOL.

1.1 Software Testing Concept

Software testing is the act of exercising software with test cases to find failures and to demonstrate correct execution. A test case has a set of inputs and a list of expected outputs.

There are two fundamental approaches to identifying test cases: functional testing and structural testing. Functional testing is based on the view that any program can be considered to be a function that maps values from its input to the output. This leads to the term Black Box Testing, in which the content of a black box is not known, and the function of the black box is understood completely in terms of its inputs and outputs. Structural testing is called White Box Testing, because the implementation is known and used to identify test cases.

One of the fundamental limitations of functional testing is that it is impossible to know either the extent of redundancy or the possibility of gaps corresponding to the way a set of test cases exercises a program. On the other hand, there are several widely accepted test coverage metrics for structural testing. Test coverage metrics are a device to measure the extent to which a set of test cases covers a program. (Paul, 1995) For example, each evaluation point (such as a true/false decision) must be executed to meet the branches coverage.

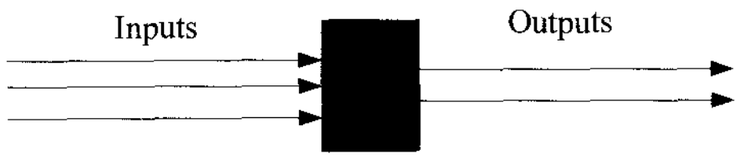

Figure 1 Black Box Testing

There are two levels of testing in the software development life cycle: test the individual components (component test) and then integrate these into subsystems until the entire system is tested (integration test). The component test aims at verifying most functional requirement, while the integration test aims at verifying the non-functional requirement such as real-time aspects, performance, HW/SW compatibility and etc.

1.2 ATTOL

ATTOL is a software component testing tool for ADA programs. It offers the user a language to describe the execution of the component tests. A test plan written in the ATTOL language is converted into a test program in ADA through the test generator. The test program is compiled and linked to all/part of the ADA program under test. The ATTOL runtime must also be linked to

the test program to take into account of the nature of the target environment. After the test program is executed, the test report is generated automatically. Figure 3 illustrates the ATTOL test flow.

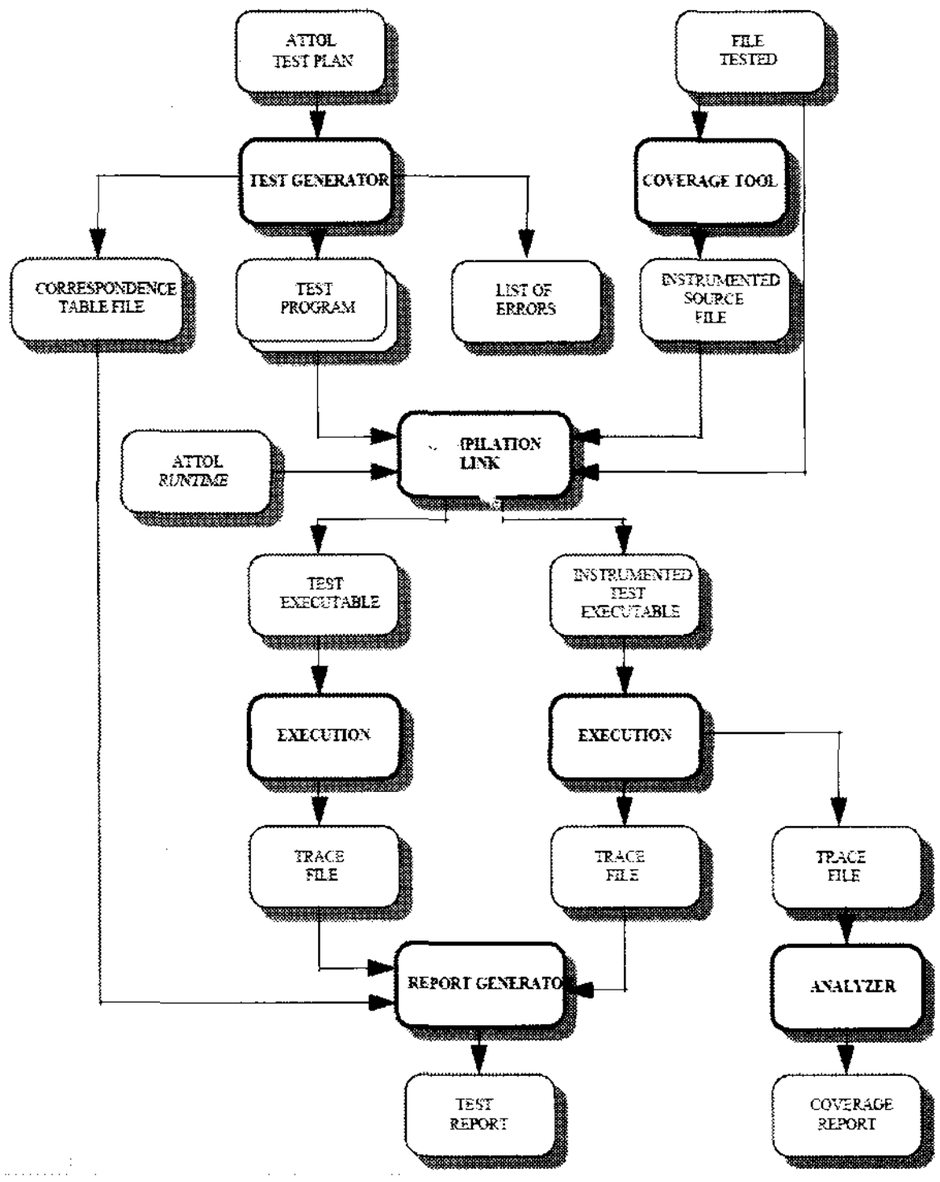

Figure 2 ATTOL test flow

The component tests described by ATTOL are black box tests. By the very nature of black-box testing, it is not possible to test variables that are internal to a procedure. The test program generated by ATTOL only controls the initialization of global variables and then hands over control to the function under test. It only regains control on return from the function.

(a) Program under test	(b) ATTOL test plan
function add (a, b: integer) return integer is begin **return (a + b);** end add;	... **# a, b, c : integer;** TEST 1 FAMILY nominal ELEMENT **VAR a, init = 1, va = init** **VAR b, init = 3, va = init** **VAR c, init = 0, va = 4** **#c := add(a,b);** END ELEMENT END TEST

Figure 3. ATTOL test plan

Figure 2 shows an example of ATTOL test plan. The keyword **init** specifies the initialization part and keyword **va** specifies the actual test part (expected value). After executing the procedure *c :=add(a, b)*, the expected value of the variable *c* shall be 4, since the variable *a* and *b* are initialized to 1 and 3, respectively.

ATTOL works with a coverage tool to support a way of structural testing (white-box testing). For example, ATTOL with a coverage tool is able to check if every branch is covered by the test program. Figure 3 shows the ATTOL test flow with a test coverage tool.

2. COMS ON-BOARD SOFTWARE OVERVIEW

This chapter introduces the functions and the design of the *COMS OBS*. The *COMS OBS* performs the major functions as follows:

- TeleCommand (TC): The ground sends a sequence of telecommands to the satellite for the satellite operation. The OBS manages receiving and executing the ground TC.
- Telemetry: The ground receives the telemetry data from the satellite for the satellite observation. The OBS manages formatting and emitting the telemetry data to the ground.
- Attitude and orbit control: The OBS manages the attitude determination and orbit control law algorithms.
- On-board power control: The OBS manages the charge and discharge of the satellite battery.
- Thermal Control: The OBS manages the satellite bus and payload thermal regulation.
- Fault Detection, Isolation and Recovery (FDIR): The OBS monitors the telemetry data to detect a failure at function level (on-board power control, thermal control, etc) and takes the appropriate actions to recover the failure.

The COMS OBS is written in ADA language and designed by the object oriented method. A software component is implemented by an ADA object. An object is an abstraction of a real world entity which gathers both data and operations on those data. The data can be either internal to the object or global to the other objects. For the global data, the object provides the ground the common services (Get/Set interface) so that the ground can get/set the value of the data. There are several layers in the object hierarchy. Each object provides services through its own interface and uses the services provided by the lower level objects.

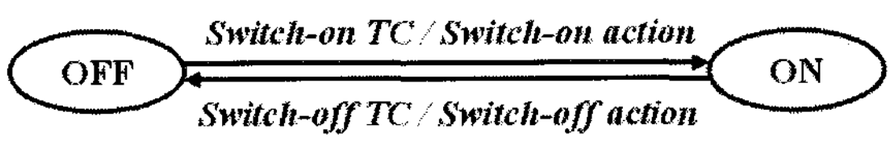

Figure 4. Automaton diagram

An object has several mode states. The mode state of an object can be modified by means of the dedicated TC. For each object, an automaton diagram describes the mode states, event and actions attached to the allowed mode state transitions. Figure 4 illustrates an automaton diagram. On the reception of *Switch-on TC*, the object transits from OFF to ON mode state with performing *Switch-on action*. One of common services, Set/Get interface can be provided if the object is in ON mode.

The COMS OBS is running on the real-time operating system, which is an in-house product of EADS-Astrium. Tasking is the core of the operating system. From a dynamic point of view, the software is a collection of tasks working together. There are several frequency levels of the cyclic tasks in the COMS OBS. The scheduling, monitoring and processing of the object are done cyclically per task frequency.

3. COMPONENT TEST STRATEGY

This chapter explains the test strategy on the COMS OBS at component level. The relevant tool, ATTOL is used for testing a software component with a test coverage tool.

Basically, the test cases are identified by the black-box testing approach and written in ATTOL language. ATTOL cannot check the internal variable, because the implementation is not known by the black-box testing approach. All ATTOL can specify is the initial value and the expected value of the global variables.

In addition to ATTOL, a coverage tool is used to support the white-box testing approach. As illustrated in figure 3, the coverage tool checks if 100% of ADA branches are covered by the test cases. Finally, the ATTOL test plan shall cover all of the ADA branches.

The ATTOL test plan of the COMS OBS at least includes initialization, request management, processing, monitoring and failure recovery. These are closely related to the major functions of the COMS OBS as described in chapter 2. Each functional requirement shall be tested as much as possible at component level.

3.1 Initialization

An object is in charge of performing the initialization of its data. For each object, the followings shall be verified:

- Each automaton is initialized to a default mode state. Generally, most automata are initialized to the OFF mode state.
- All the global variables are initialized with their default value.

3.2 Request management

An object is in charge of executing its applicable requests (ground TC or internal TC, failure recovery). The hierarchical architecture and object oriented design enables to split a request management in several steps: request dispatching, request validation/actions collection and actions execution.

REQUEST DISPATCHING

When a request comes from the ground, the OBS identifies the object which is in charge of validating and executing it, and then routes the TC to the appropriate level.

The followings shall be verified at this step:

- Each TC and interface request at an object level are well taken into account,
- Each TC is well routed to the relevant low level object.

REQUEST VALIDATION/ACTIONS COLLECTION:

For each object, the followings shall be verified.

- Each TC shall be tested in each mode state and transition context.
- Request rejection causes are tested: mode state transition forbidden, unknown TC, etc.
- If specified in software requirement document, TC rejection entailed by out of range parameter is also tested.
- All the TC and Set Interface Methods action collection are checked.

ACTIONS EXECUTION

For each object, the followings shall be verified.

- All the actions are tested.
- All the algorithms attached to this activity are tested.
- All the interfaces between each object are tested.

3.3 Processing

An object is in charge of scheduling algorithms according to its mode state configuration on several frequency levels.

SCHEDULING

For each object, it is validated that, for each frequency the processings and the monitorings are scheduled as required in the software requirement document according to the mode state.

In order to provide the ground with the Get method, a processing produces the value of the global variable cyclically. The production frequency and the consuming frequency of the Get method data are checked.

PROCESSING

All the processings are exhaustively tested at each object level with data chosen for the branch coverage. A test coverage tool is used to check the branch coverage with ATTOL.

All the processing initializations are tested at object level.

All the Get and Set methods are tested.

3.4 Monitoring

An object monitors its own data. Each monitoring at function level (on-board power control, thermal control, etc) shall be triggered to validate each error report update and each immediate passivation action.

3.5 Failure Recovery

For each object, there are required failure recovery scenarios according to the current mode state. When the failure is detected, all the passivation actions shall be tested.

4. CONCLUSION

The COMS on-board software (OBS) is written in ADA language and developed by the object-oriented approach. A software component is implemented by an ADA object. An object is in charge of initialization, request management, processing, monitoring and failure recovery. Each software function shall be tested as much as possible at component level. This paper introduces the functions and the design of the COMS OBS, and explains the test strategy on the COMS OBS at component level.

4.1 References

Paul C Jorgensen, 1995. *Software Testing – A Craftsman's Approach*. CRC Press

4.2 Acknowledgement

This work is sponsored by Korean Government with the following institutions involved: MOST (Ministry of Science and Technology), KMA (Korea Meteorological Administration), MIC (Ministry of Information and Communication), MOMAF (Ministry of Maritime Affairs and Fisheries).