Validation of the Satellite Speed Measurement System with the Use of Finite State Machine for Test Case Generation

Antonio Cassiano Júlio Filho[1], Auro Tikami[2] and Ana Maria Ambrosio[3]
National Institute for Space Research (INPE)
[1]cassiano.filho@inpe.br, [2]auro.tikami@inpe.br, and [3]ana.ambrosio@inpe.br

Abstract

The area of Model Based Software Testing is of interest to researchers, professional experts in testing and development as well as to the industrial sector. Several test methods are proposed in the literature for the validation of a computational system. This paper presents the experience of using finite-state machine (FSM) to model a system for automatically test generation applied to a software system of the space area. Starting from the functional requirement of the Satellite Speed Measurement (SSM) developed by National Institute for Space Research (INPE), the system behavior is modeled and the JPlavisFSM platform is used to automatically generate test-case sets. This platform provides different FSM based methods for system validation purposes. The test-case sets (each set is resulted from one method) are compared and evaluated according to the cost of generation and their size. The paper also discusses the efficiency of practical application.

1. Introduction

Telemetry, Tracking and Command (TT&C) Ground Station (GS) receives, demodulates, and records spacecraft data telemetry, transmits commands in S-band frequency and performs measurements of distance (ranging data) and speed of satellites (range rate), angle data and weather data.

Figure 1 shows the INPE’s Ground System including the Satellite Control Center (SCC), the Mission Center (MC) and the main systems of the its ground station: telecommand, telemetry, satellite speed measurement, and telemetry payload. The telemetry data and commands generated at the SCC, located in São José dos Campos, Brazil, are routed to/from the GS, located in Cuiabá and Alcântara, via dedicated communication lines. The SCC is responsible for controlling the orbit of the satellites. Besides the telemetry service, the GS receives payload telemetry mainly from Data Collection Platforms (DCP). The Mission Center in the city of Natal is responsible for the data storage and its dissemination to DCP users.

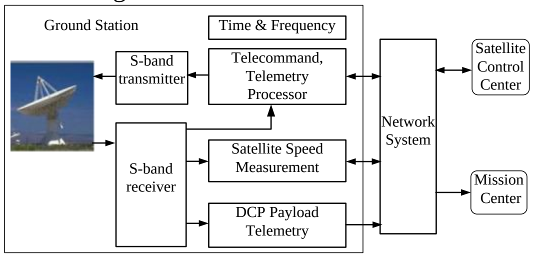

Figure 1. Ground system

One of the main systems of the ground station is the Satellite Speed Measurement which is presented in details in section 2.

In order to complete the validation [6] of the Satellite Speed Measurement, we created models to represent its behavior [3] and took advantages of the JPlavisFSM platform facilities to automatically generate test cases at the system validation level. The JPlavisFSM platform allows to apply different methods for test generation based on the FSM models. In addition to it, it evaluates sets of test cases based on the concept of FSM mutant.

The results obtained with the different sets of test cases were compared to the score of mutation and efficiency of the test cases. Additionally, we performed a manual review of test cases, as presented in section 6.

2. Description of the satellite speed measurement

The SSM is a system which aims to estimate the radial velocity of a satellite in orbit by measuring the Doppler shift in the frequency of an RF carrier transmitted to the ground and provides a mass of data for orbit prediction processing.

The SSM measures the Doppler shift under request of an application in local or remote mode. This request is interpreted by a software module that generates a request that is sent to SSM through a communication network. The SSM checks and sends a reply message to the requester whether the commands are valid or not. If so, the SSM performs the required measures, collects the status information of the equipment and sends a reply message to the requester.

The measurements obtained are formatted reports such as "XML" according to the communication standard of the ground segment and sent to SCC for further processing.

SSM is composed of the following modules:

• Frequency Measurement - hardware module responsible for the measurement of frequency deviation.

• Dating - hardware module that provides the timing of the messages.

• Control Software Module (CSM) - responsible for process control, user interface, configuration, monitoring and treatment of the SSM measures.

3. Control software module requirements

The requirements of CSM are listed below. Each requirement is uniquely identified by a number.

REQ.01 - The system shall provide the user with options: Request, Logoff and Report Measures.

REQ.02 - The system shall provide a page for configuring a request with the following measurement parameters: Mission Center: Natal, Station: Cuiabá or Alcântara, Satellite: SCD1 or CBERS, Measure Group Number and Measure window.

REQ.03 - The system shall allow the selection of the measurement window of 1 or 10 seconds.

REQ.04 - The system shall form a measure group in every 10 measures.

REQ.05 - The system shall allow programming of the number of measure groups for window of 1 second, the minimum value is 1 and the maximum 255.

REQ.06 - The system shall allow programming of the number of measure groups for window of 10 seconds, the minimum value is 1 and the maximum 25.

REQ.07 - The system shall perform the frequency measurement according to the window size and the number of measure group.

REQ.08 - The system shall make the timing acquisition in each measurement window.

REQ.09 - The system should send a response to the request as being: primary response (message indicating that the transaction has been accepted), transaction response (message with the measures required) and unidentified message response (message indicating that the transaction has not been identified).

REQ.10 - The system should record a history of measures.

REQ.11 - The system shall provide a page with the following table of results: unsolicited requests, responses to requests and measure group.

REQ.12 - The system should allow, after the completion of measures required to view the last 10 requests requested, in descending order, listing the parameters: number of the requisition, command, satellite, station, date/time and number of groups.

REQ.13 - The system should allow the visualization of responses, in descending order, with the requests of the measures, listing the parameters: number of the answer, number of the request, result status, hardware status, and message samples.

REQ.14 - The system should allow viewing a report of the last 10 measure groups listing the parameters: number of the measure, number of the response, date/time, number of the message, totalizing counter, interval counter and the frequency.

REQ.15 - The system shall provide a page with the tables: history of the requisitions and measure group.

REQ.16 - The system should allow the visualization of history with all the requisitions requested, in descending order, listing the parameters: number of the response, number of the request, command, satellite, date/time and number of groups.

REQ.17 - The system should allow the viewing of the last 10 measure groups listing the parameters: number of the measure, number of the response, date/time, number of the message, totalizing counter, interval counter and the frequency.

REQ.18 - The system should allow a XML file generation with the last 10 measure groups, listing the parameters: number of the measure, number of the response, date/time, number of the message, totalizing counter, interval counter and the frequency.

4. Identification of the system under test

The models were created to represent the Control Software Module, which is considered here our System Under Test (SUT), as shown in figure 2. This module manages the processes required to implement the SSM functions: web interface, configuration and monitoring of parameters and treatment of measures.

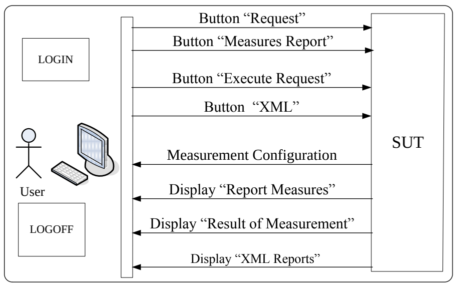

Figure 2. Environment of SUT

5. Modeling

In order to create the FSM models, we first characterized in depth the lists of input events and output actions which are presented in table 1. The output actions and their relationship with the requirements are presented in table 2. The states of the FSM are described in table 3.

Table 1. Lists of events and actions

Input Events		Output Actions	
Code	Description	Code	Description
breqMed	press button "Request"	config Med	Setup parameters of measure
bExeMed	press button "Executes Report"	result Med	Display Result of Measurement
bRelMed	Press button "Report Measures'	relat Med	Display Report Measures
bXML	Press button "XML"	gerRel XML	Display XML Reports
--	--	Idle	no action

Table 2. Action code and requirement numbers

Action Code	Requirement Numbers
ConfigMed	2, 3, 4, 5, 6 and 7
ResultMed	8, 9, 10, 11, 12, 13 and 14
RelatMed	15,16 and 17
gerRelXML	18

Table 3. States of the sut

State	Description
s0	Page "Operation"
s1	Page "Setup parameters of measure"
s2	Page "Result of Measurement"
s3	Page "Measure Reports"
s4	Page "XML Reports"

Figure 3 shows the FSM representing the SUT.

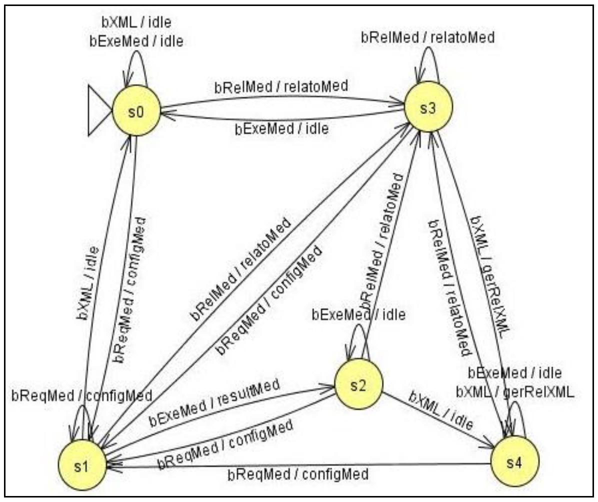

Figure 3. Finite state machine diagram

One important facility provided by the JPlavisFSM is the evaluation of the FSM properties [7]. This machine has the following properties: Deterministic, Fully Specified, Reduced, initially connected and strongly connected. If one property is not satisfied the platform indicates a correction that must be done before running the method. This facilitates the use of formal methods for beginners. The tool allows to properly design a FSM in order to execute successfully the method to generate the test cases. To complete the FSM, transitions and outputs were added in order to get the required properties for the generation of test cases.

6. Generation the test-case sets

JPlavisFSM platform [7] allows one to generate different of test-case sets, each set based in a FSM based method. There are four methods available in the platform: W [4], HSI [2], SPY [1], UIO [8]. It is also possible to manually include a test set.

The W method [4] ensures that the implementation of a corresponding FSM model generates correct output when input sequences produced by this method are applied against the implementation, if this machine is correct. This is because the method is reliable for testing control structures modeled by a FSM. However, the method produces a high number of sequences of inputs, which promotes a high cost to execute the test-case set.

The HSI method [2], which is a modification of the method W, guarantees the coverage of existing defects, as applicable in any reduced specification (minimal), whether partial or complete.

The SPY method [1] reduces the size of the test sets. The completeness of the set is ensured by checking if the various sequences lead to the same state in the implementation, so the method prevents many sequences from being added to the test set, increasing its size.

The UIO method [8] is a particular case of the W method and is expected to generate shorter test suites.

We manually created 2 sets based on the outcomes of the W method, which were the W** and the Manual. The W** set was created by subtracting the list cases containing the “idle” output. The Manual set was created by the system engineer by choosing the most significant test cases according to his assessment.

6.1. Comparison of the sets

Table 4 presents the numbers used to compare the sets of test cases created to validate the SSM system. Six sets were created: W, HSI, SPY, UIO, W** and Manual. They are shown in the columns. In table 4, the TC line indicates the number of test cases generated, the line M shows the number of mutants, the line DM presents the number of dead mutants, AM line indicates the number of alive mutants, while the number of equivalent mutants is given in EM line and finally, the mutation score is shown in MS line.

Table 4. Comparison of the set of test cases

	FSM based Methods					
	W	HSI	SPY	UIO	W**	Manual
TC	32	32	17	7	5	2
M	188	188	188	188	188	188
DM	188	188	188	97	81	39
AM	0	0	0	91	107	149
EM	0	0	0	0	0	0
MS	1.0	1.0	1.0	0.51	0.43	0.20

7. Conclusions

The use of FSM models for automatic generation of tests is useful to reduce the cost of the test generation. Although the methods have a common goal to check whether an implementation is correct with their corresponding specification, the methods differ each other with respect to the cost of generation of test sets, in terms of the size of the test set and its effectiveness. Effectiveness is the capability of the maximum detection of defects in the implementation by the generated test set which must be small enough to allow their application in practice.

For the selection of a method by the tester, the model must comply with the properties required by the method; the size of the sequence must be convenient and the cost of execution affordable.

In the case of the SSM, according with table 4, the most appropriate method for a practical application, is the SPY, considering the score of 1.0 and the lowest number of test cases generated being 17.

The results of the methods W and HSI shall be considered as a theoretical reference of test case sets, since they ensure the correct implementation of the machine.

One of the most important gains with the use of FSM modeling was obtained during the model creation phase because this activity allowed feedback and improved the requirement at the design phase.

In this experience of using FSM and the JPlavisFSM, the modeling allows us to identify what a system should do and what the real expected result should be. Whereas the model is correct, it is extremely valuable for the generation of test cases [5].

In the evaluated SUT, the domain of FSM were the system requirements with a high degree of fidelity, verified by conformance tests applied against the real implementation and also all tests were performed at the implementation. As the modeled part of the SSM was simple, no errors were found in the implementation.

8. Acknowledgments

We thank Dr. Adenilso Simão and Arineiza Pinheiro by providing us with the JPlavisFSM platform.

9. References

[1] A.S. Simão, Contribuição para Teste de Software, Janeiro de 2011, Instituto de Ciências Matemáticas e de Computação, USP, São Carlos, São Paulo, Brazil.
[2] A.T. Endo, A. Simao, Evaluating Test Suite Characteristics, Cost and Effectiveness of FSM-base Testing Methods, Information Software Technology (2013).
[3] Binder, R. V, Testing Object-oriented Systems: Models, Patterns and Tools. Addison-Wesley, 1999.
[4] Chow, T.S. Testing software design modeled by finite-state machines. IEEE Trans. Softw. Eng., 4, 178-187, 1978.
[5] Delamaro, M. E.; Maldonado, J. C.; Jino, M, Introdução ao Teste de Software, Elsevier, p. 1-7, 2007b.
[6] ECSS-E-ST-10-02C. Space Engineering - Verification. March, 2009.
[7] Pinheiro, A. C. Subsídios para a aplicação de métodos de geração de casos de testes baseados em máquinas de estados. Dissertação de mestrado. Arineiza Cristina Pinheiro; orientador Adenilso da ICMC-USP, São Carlos, 2012.
[8] Sabnani, K.; Dahbura, A. A protocol test generation procedure. Computer Networks and ISDN Systems, v. 15, n. 4, p. 285-297, 1988.