On the use of test standardization in communication space applications

A.M. Ambrosio [1] ; E. Martins[2] ; M.F. Mattiello-Francisco[1]; C. S. Silva[1]
N. L. Vijaykumar[1]

[1]INPE - National Institute for Space Research
Av. Dos Astronautas, 1758 - Sao Jose dos Campos -12227-010 - SP - Brazil
FAX: +55-12-3945-6625, E-mail: ana@dss.inpe.br ; fatima@dss.inpe.br ; claudia@dea.inpe.br ; vijay@lac.inpe.br

[2] UNICAMP- University of Campinas
E-mail: eliane@ic.unicamp.br

ABSTRACT

This paper presents some experiences in adapting the concepts, framework and methodology for protocol testing from the standard IS-9646 to validate space application protocols. In order to assess the space applications the conformance tests defined in IS-9646 are combined with the fault injection technique. This solution comprises a test architecture and a test process with elements to assess not only error recovery and exception mechanism but the behavior of the protocol implementation in presence of failures. A research project which includes some already implemented testing tools based on the IS-9646 has supported experiments conducted on real space applications validation. Results have pointed out advantages on the use of test standardization for space applications validation.

Keywords: space protocol testing, IS-9646, system testing architecture, testing process.

1. INTRODUCTION

The quality and cost-effectiveness of the space products is only achieved if strict methods and standards are established and followed by the involved parts, such as the space agencies and industries. Therefore, the Consultative Committee for Space Data Standardization (CCSDS)[1] has been defining, since the 80´s, a set of recommendations mainly for space communication protocols. It is needless to say that these recommendations are extremely valuable and indeed several of these have become ISO standards.

Complementing CCSDS activities, the European Space Agency, through the European Cooperation for Space Standardization (ECSS) has also been working on preparing a set of other standards for the space activities improvement. The objective of the ECSS standards is to facilitate the elaboration of an efficient and cost-effective space management system. These standards cover management, quality assurance and engineering rules for the space projects execution.

Both the standardization committees have extensively worked in standardizing ground-board communication protocol. The CCSDS recommendations have been focused on the end-to-end transport of telemetry and telecommand data from/to ground/board systems, while the ECSS recommendations has addressed a set of services provided by the on-board application processes in which the data field of the telemetry and telecommand packets are standardized [1].

Although there is a great effort in standardizing the protocol behavior and data structures for space communication applications, one may not find much work at engineering level to provide directions to the generation, implementation and execution of tests of a space application software.

In order to conduct testing of a space application software, one may use the standard IS-9646 [2]. In [3] the tests of an implementation of the CCSDS standard Space Link Extension (SLE) services for communication between the ESA and the JPL were based on concepts stated in IS-9646. Besides defining a methodology, corresponding terminology, test specification language, the IS-9646 provides a framework for specifying conformance abstract test suites. Moreover, it guides the realization of means of testing capable of running an executable form of an abstract test suite as well as covering the conformance assessment process.

In short, this standard recommends that sets of tests, called *test suites*, be developed and standardized for all ISO standardized protocol. In some context the IS-6946 standard may also be applied for testing CCSDS specified protocols for space communication applications.

The approach presented in this paper is based on automating some test activities of the IS-9646, under the research project named ATIFS[2]. This project is carried out by INPE and the University of Campinas. Today the ATIFS comprises a set of test tools which supports test generation, test execution and result analysis. Its main characteristic is not to be limited to the IS-9646 conformance tests, but to combine with conformance testing the software implemented fault injection technique (SWIFI), which is an attractive aspect for space application validation. Based on the ATIFS tools and techniques, we have been focusing on understanding the protocol testing standards, tools and techniques when applied on real space application protocols.
Two aspects of protocol testing are addressed here, the test architecture and the test process. The test architecture provides an implementation of the means of test to support the test execution. The test process describes the activities for one to perform the tests. Section 2 presents a brief overview of these branches as stated in the standard IS-9646 while section 3 discusses the solutions given into ATIFS scope. Section 3 also illustrates the use of the test architecture configured to test the implementation of a telemetry reception software of a balloon scientific experiment and used as the test system of the in-house developed Brazilian Payload Computer (BPC).

Section 4 discusses the challenges and the lessons learned. Section 5 concludes the paper.

[1] http://www.ccsds.org/CCSDS/recommandreports.jsp
[2] Test Environment with software fault injection – http://www.inpe.br/atifs

2. ISO IS-9646

The ISO IS-9646 is a standard for protocol conformance testing. It defines a methodology and a framework to guide the test specification, the test execution and the analysis of the results, so that the test of an implementation of a standardized protocol may be applied by different laboratories and they should get the same results.

This standard addresses conformance testing, which consists of checking whether an Implementation Under Test (IUT) satisfies all the specified conformance requirements. These requirements are checked by running a number of tests against the IUT (these tests are referred to *test cases*).

Abstract Test Architecture

The means for conducting the conformance test are based on the recommended *abstract test methods* (i.e., an abstract architecture of a system executing the conformance tests; it consists of a configuration of the functions: Lower Tester (LT), Upper Tester (UT), Lower Tester Control (LTCF) and Test Coordination Procedure (TCP)). The test method may be applicable to a test context (environment where the IUT is embedded): either *Single-Party Testing context* if the IUT requires only one communication line for test execution or *Multi-Party Testing context* otherwise. The test methods vary in the extent of control and observation points (PCO) of an IUT that they provide. Each PCO has a tester associated to it. Whether the IUT is in the same environment as the System Under Test (SUT) the method is local or remote. The choice of the test method influences the execution of the test cases. Figure 2.1 illustrates the local test method in Single-Party context.

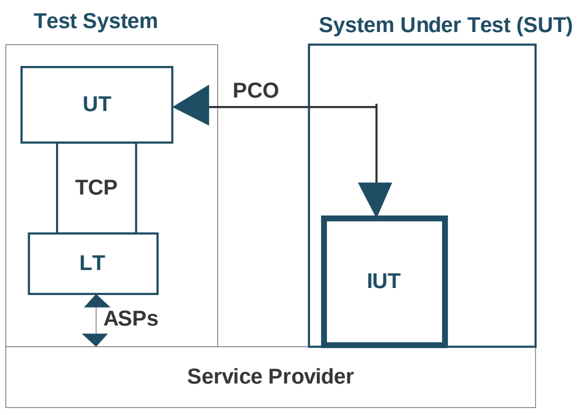

Figure 2.1: Local Test Method in Single-Party context

Test Process

The IS-9646 also defines the conformance test process, which is depicted in figure 2.2. In short, this process may be considered as three-phased. The first phase is the *test case generation*: starting from the Standard Protocol Specification, (going left side of the figure) the test purposes are defined and they will originate an *Abstract Test Suite* (ATS - specification of test cases that are implementation independent) taking into account the test methods, that may restrict the test suite. The ATS is written in the TTCN[3] test notation, according to IS-9646.

The second phase is the *test implementation*. It consists of creating the means to execute the tests. The test cases to be implemented (those implementation-based) are selected from the ATS and they are transformed into the *executable test suite* taking into account the environment of the IUT and the provided implementation information.

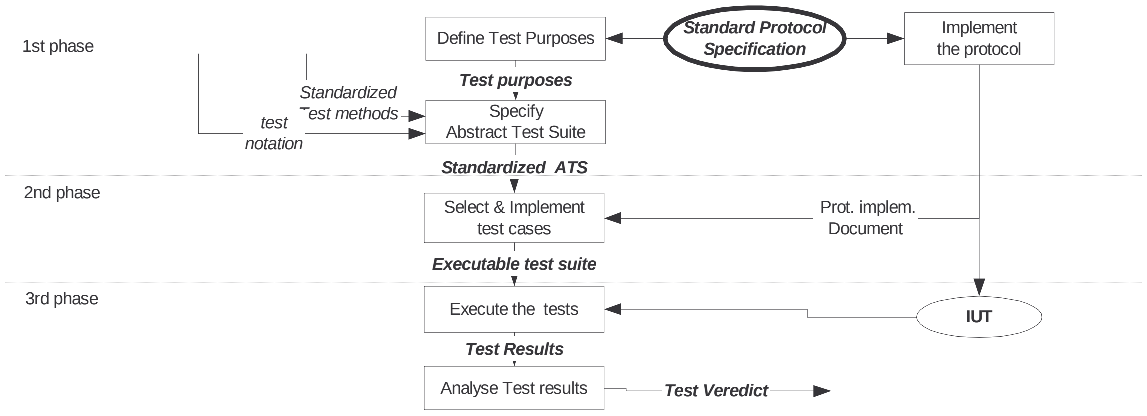

Figure 2.2.: Overview of the IS-9646 Conformance Testing Process

Finally, the third phase consists of the *test execution* against the IUT. The behavior of the IUT, i.e., its generated outputs, are observed and the results are analyzed leading to a verdict about the conformance of the IUT with respect to the standard protocol specification.

3. THE ATIFS RESEARCH PROJECT

The ATIFS project comprises a toolkit to help one to execute conformance tests based on the IS-9646 concepts. Additionally, it adds to conformance testing the possibility of combining tests by injecting faults in the IUT. So the behavior of the IUT may be observed also in presence of environment faults [4].

The *conformance tests* aim at determining if an implementation meets its specification. These tests are applied to an implementation, the IUT. The test outcomes are compared with the expected outcomes. Only observable aspect of the implementation, e.g., their observable input and outputs are taken into account. In ATIFS the conformance test are automatically derived from a specification given in a finite state machine, in which the state transitions are represented by the inputs and their corresponding expected outputs. For details one may see [5].

The *fault injection* consists of the deliberate insertion of faults or errors into a system aiming at observing its behavior. This technique is useful to validate the implementation of error recovery and exception mechanisms, as well as to determine the system behavior in the presence of environment faults. The fault injector implemented in ATIFS mimics communication faults, like message loss, duplication, corruption and delay.

[3] Tree and Tabular Combined Notation

Both these test aspects are useful and valuable in the validation of a space application leading us to further use the ATIFS tools and techniques and taking its advantages for testing INPE's in-house implementations. For the test execution phase, the ATIFS provides the user with a test architecture supported by a tool whose description and uses are given in Section 3.1.

Regarding the test process, the ATIFS organizes the activities to conduct the experiments for communication system validation taking into account both conformance and fault-injection-based tests in all test phases. The current state of the process definition has yet many things to be established in order to satisfy space software system needs. This topic is discussed in Section 3.2.

3.1. Ferry-Injection Architecture

The ferry-injection architecture, depicted in figure 3.1., is an extension of the ferry-clip architecture as defined in [6] yet satisfying the ISO methods. It has the three most important components of the ferry-clip: the active and passive ferries and the ferry channel, which are responsible only for the communication between the Test System and the SUT during the test execution. This solution allows the Test System to be as independent of the IUT as possible, so changes in the IUT will cause the minimum changes in the rest of the systems used for testing. In the ATIFS project, Martins [4] has added the ferry-clip architecture with a fault injector divided into two modules, the Fault Injector Controller (FIC) and the Fault Injection Manager (FIM) modules, following the idea of reducing extra processing on SUT. In this architecture the *Ferry Channel* is used only for test purposes and the *Test Channel* is an implementation of the *service provider* of figure 2.1. The lower and upper testers (LT and UT) are part of the Test Engine module, which also includes the Test Coordination Procedure (TCP). (With only one LT and UT the Lower and Upper Tester Control Function (LTCF) is not required).

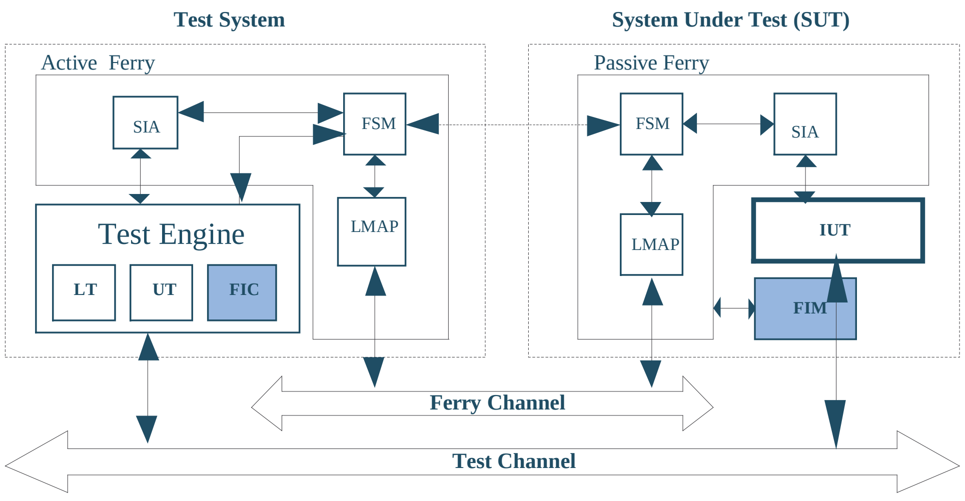

Figure 3.1. Ferry-injection architecture

The ferry–injection architecture in a single-part context succeeded to test a real communication system developed at INPE, the TMSTATION software. This architecture has also been studied to support the tests of an on-board computer requiring multi-party context. These experiments are briefly presented as follows.

Experiment 1: TMSTATION testing

For supporting the test execution of the Telemetry Reception Software (TMSTATION) [7], the ferry-injection architecture configuration shown in figure 3.2., required a single-party context. This software (IUT) implements (in C / Windows) a ground entity of the ground-board communication protocol receiving, in real time, through a single channel (serial RS232 interface), the data acquired from the X-ray sky imager installed in a balloon. It separates the frames (sequentially acquired data) into distinct files, based on the identification of an end-of-frame pattern. In the test architecture, the Lower tester, the TCP and the FIC that comprises the Test Engine were written as a test script in C/LINUX. The test channel is a serial line while the ferry-channel is a LAN whose communication mechanism was implemented in sockets. The FIM, which injects the communication faults, is located as close as possible of the IUT, then it was implemented in C / LabWiew for the Windows environment.

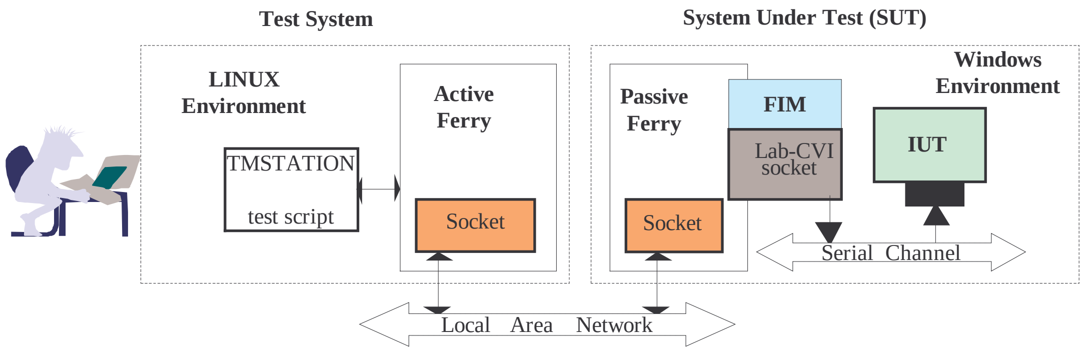

Figure 3.2. Ferry-injection architecture for TMSTATION Tests

This architecture succeeded in the test of the TMSTATION for two different test phases. Various scripts were developed for conformance testing and for fault injection. For details consult [4].

Experiment 2: Brazilian Payload Computer testing

The Brazilian Payload on-board Computer (BPC) software and hardware has been developed at INPE. The system to test the BPC comprises three different equipment. This system is connected to BPC via different communication lines: (i) an OS-link line connected to the development board, (ii) an analogic line connected to a PC that provides the analogic telemetry from the payload equipment (iii) a RS-232 from which telemetry and telecommand packets arrive and leave to/from the on-board main computer simulator. Then, the test system requires a multi-party context method where more than one line is required to conduct the tests. In configuring the ferry-injection architecture, the following questions are posed: (i) Where to locate the active and passive ferries in order to provide the communication with the various lines? (ii) Where to locate the FIM to avoid as much intrusion as possible in the IUT? A solution being considered (illustrated figure 3.3.) is to triple each module implementing a communication line drive (AFs and PFs); to implement the PF in the development board, as close to the IUT as possible, though avoiding the intrusion in the on board computer.

Although, this architecture has shown us that testing on-board computer usually requires a multi-party methods and that to divide the ferries is a good solution, further studies are needed to conclude the configuration of the ferry-injection architecture to comply with the multi-party methods.

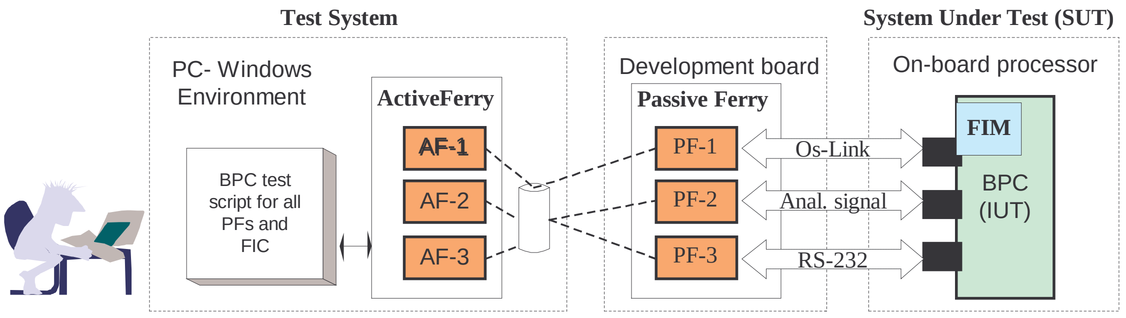

Figure 3.3. Ferry-injection architecture for Brazilian On-board Computer Tests

3.2. Conformance and fault injection testing process

The IS-9646 defines the test process illustrated in figure 2.2, while in ATIFS we have been working in the definition of a process that addresses the automation aspects and merges the conformance and fault-injection testing approaches, still complying with the IS-9646 standard. An overview of the ATIFS tests process is given in figure 3.4

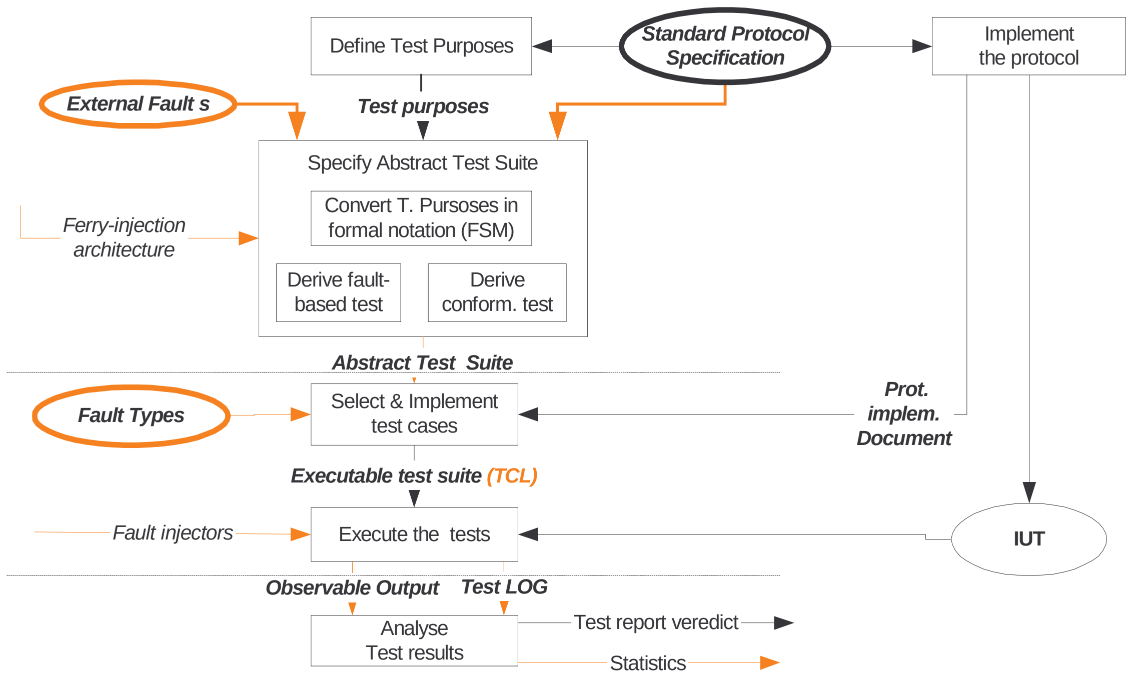

Figure 3.4: Overview of the Conformance and Fault Injection Testing Process

The ATIFS test process comprises the three phases of the ISO process, covering the same test activities except that for automating some steps and improve that process with the fault-based tests the following differences may be highlighted:

(i) The ATIFS process defines in smaller steps the derivation of the abstract test suite (ATS): the *test purposes* (defined by studying the protocol specification) are translated into a finite state machine from which path transitions are taken as test cases by using graph traverse algorithms [5], so the activity represented in the "*Derive conform test"* box is automated.

(ii) Continuing in the "Specify Abstract Test Suite" activity, this process allows the tester to specify the possible external communication faults, which will be used to check the SUT behavior when unmatched, delayed and lost messages occurs.

(iii) In order to avoid the huge number of test cases generated by graph traverse algorithms, which are used to automate the *test case generation* phase (step (i)), the exceptional situations are subtracted from the protocol specification (then reducing the FSM). To support this step, the ferry-injection architecture is taken into consideration and the exceptional situations are then tested by injecting faults.

(iv) the generated ATS is written in a particular language closer to the TCL[4] bridging the automation step to executable test suite, instead of being written in TTCN, as recommended in IS-9646.

(v) In the step represented by the "Select & Implement test cases" box, test cases to be implemented are chosen from the ATS. This chose is based on the documentation of the protocol implementation, such as in the ISO process. But here, the faults to be used for completing the test situations may be chosen among the options: transient (inserted only once), intermittent (inserted periodically) or permanent (inserted in all messages). On the set of chosen test cases, drivers and special information are included, then the test cases are translated into a script in TCL language.

(vi) The test execution is performed under the automated version of the ferry-injection architecture supported by an ATIFS's tool.

(vii) During the test execution, all the test events are recorded in the test LOG and the input and output effectively observed are stored and latter compared with the expected outputs for the corresponding inputs. In this process, besides the conformance verdict report, statistics about the detected errors are generated.

Although this test process is interesting for space applications due to combining fault injection with conformance tests, the test purposes are still defined as ad hoc from the protocol specification. In order to address this situation, we have been working on a test methodology oriented to space area.

The test methodology being considered starts from the TM & TC utilization packets services specification [1] and uses UML-based models. Each service defined in [1] is translated in a use-case, where normal, alternative and exceptional scenarios are identified. After each scenario is, then, modeled in sequence diagrams and finally transformed into the FSM to be automatically executed. Only the normal and alternative scenarios are translated into FSMs. From the exceptional scenarios the faults to be injected are derived. With this work we intend to provide an ATS for the Packet Utilization Services [1] specification. Based on that set of abstract test cases one would select and implement the test cases related to their services implementation.

4. LESSONS LEARNED AND CHALLENGES

In performing the studies presented in the above section we can summarize some important points, which are reported as follows.

Test Architecture

Firstly, we configured the test ferry-injection architecture to two space applications (e.g., telemetry and telecommand handling system) validation. In short, the lessons learned with these experiments are:

- The ferry-injection architecture used in the single-party context could be easily configured for different test phases and objectives, in the first experiment. We used 2 distinct configurations and both have provided us good results without spending much time in the tool adaptation;
- Standard reusable tools are very welcome by testers. Although, the use of a new technology requires familiarization efforts, improvements in testing tools, techniques and methods compensate the time spent with the familiarization, when they may be reused in the future;
- With the second experiment, we may say that, testers of the on-board computers are more interested in the fault injection techniques. The communication set of faults implemented in ATIFS will be improved to include also typical memory faults. Furthermore, to test on-board computers usually require *Multi-Party Testing* context, so test architecture in this context will be further studied, as the experiment is not yet completed.

Test Process

The ATIFS process activities are currently supported by tools. Two test case types were generated for testing the TMStation: the conformance and the fault-based. On comparing them we found that the fault-based test suite is much efficient than the conformance suite.

5. CONCLUSION

We have posed the solutions provided in the ATIFS research project, which is based on IS-9646 test standard, focusing on the test architecture and process topics.

[4] Tool Command language .

The ATIFS test process combining conformance and fault injection test allows test cases derivation and system robustness to be validated under the presence of faults which is an attractive perspective for space applications. The present work being carried out on ATIFS project has defined steps to direct indications to a tester to find an efficient test suite based on the use of UML[5] diagrams and taking as a starting point the standard specification of Packet Utilization Services [1]. At the completion of this project we will have an abstract test suite (ATS), i.e., test cases specified independently of any real testing device comprising all possible options for it. In this way we may have a test guide independent from the implementation as a basis to different implementations into a particular mission or for any other mission based in the test standard.

The ferry-injection test architecture, proposed in ATIFS project, was configured to support the test execution of some project at INPE so we could evaluate its performance with real space application and better understand the real needs for achieving a standard test architecture.

In conclusion we reinforce that the automation and standardization of test activities allow the tests be scaled up and re-used across one mission. Moreover, they can also be applied to other missions, thus, by saving money and improving the reliability of the testing tool itself.

References:

[1] ECSS-E-70-41^{A} - Ground systems and operations – Telemetry and Telecommand packet utilization – January 2003. - ESA Publications Division

[2] International Organization for Standardization/International Electrotechnical Commission – “CTMF- Conformance Testing Methodology and Framework”, International Standard IS-9646. ISO, Geneve, 1991.

[3] M. Mertens – Advanced Protocol Testing Methods and Tools - SpaceOps2002 (2002)

[4] E. Martins; M.F. Mattiello-Francisco, A Tool for Fault Injection and Conformance Testing of Distributed Systems, Proc. of 1st. Latin-American Dependable Computing Symposim, Lecture Notes in Computer Science, Springer Verlag, September/2003, pp282-302.

[5] E. Martins; S.B. Sabião; A.M. Ambrosio, ConData: a Tool for Automating Specification-based Test Case Generation for Communication Systems, *Software Quality Journal*, Vol. 8, No.4, 303-319, edited by Anna Liu and Paddy Nixon - Kluwer Academic Publishers, 1999.

[6] S.T.Chanson, B.P.Lee, N.J. Parakh, H.X. Zeng, Design and Implementation of a Ferry Clip Test System, Proc. 9th IFIP Symposium on Protocol Specification Testing & Verification, Enscchede, The Netherlands. (1989) 101-118

[7] M.F. Mattiello-Francisco, Software Requirements Specification for MASCO Telemetry Ground Reception, Internal Report MASCO-SRS-001, INPE, (05/2000)

[5] Unified Modeling Language