

Automated validation of embedded optical network software

Aline Cristine Fadel, Regina Moraes
School of Technology – UNICAMP
R. Paschoal Marmo, 1888 – CEP: 13484-332
Limeira, Brazil
alinecfadel@gmail.com, regina@ft.unicamp.br

Eliane Martins
Institute of Computing – UNICAMP
Av. Albert Einstein, 1251
Campinas, Brazil
eliane@ic.unicamp.br

Abstract - This work discusses automated tests performed on an optical network for high-capacity triple-play services (voice, video and TV). Given that high availability and reliability are important requirements for this kind of network system, it is mandatory to apply regression testing every time it is updated. For automatic execution of the regression testing a test robot was developed. It is responsible for collect the outputs and compare them with the expected results. In order to complement the regression testing, fault injection campaigns were performed, which were based on state machines of the embedded software, seeking a more comprehensive coverage of tests. For this purpose the robot was adapted to send a trigger to the algorithm that controls the fault injection. The fault injection used an optical switch that interrupted the communication among the board's system components. The results show the effectiveness of fault injection in detecting bugs that were not detected during several months when other types of tests were applied.

Keywords: *embedded software validation, fault injection, regression test, GPON*

I. INTRODUCTION

Telecommunications systems must operate without interruption and without loss of data, since these events may cause several financial losses to telecommunications operators and users.

According to Sommerville [1], reliance on computer systems is a property that reflects the degree of confidence that users may have on the system and it depends on reliability and availability of telecommunication system. Moreover, according to Clark and Pradhan [2], availability is the ability to be operational at any given time without failures, while reliability is the ability to operate without failure for a certain period. However reliability cannot be expressed numerically; other classification can be used such as: "unreliable", "very reliable" and "ultra reliable" [1].

Disregarding reliability and availability during software development can result in the occurrence of several failures in the operational phase of the system. Software test can be applied to help in disclosing the faults responsible for these failures.

Software testing is time consuming, therefore the automation of software test is designed to allow performing a large volume of test cases (TC) in a shorter time. Also, automation helps to achieve greater reliability in results and to reduce costs in the testing process.

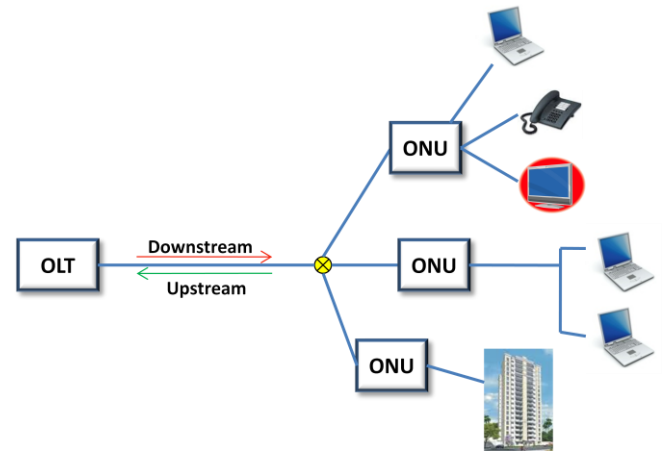


Fig. 1. Operation of GPON network [4].

The use of tools to automate tests is a challenge in embedded systems as each type of embedded software has a specific interface. Thus, each embedded software requires a customization for automation to be successful.

The optical networks are gaining new followers every day, mainly in European and Asian continents [3]. The increase of the number of users occurs due to high rates of transmission of these networks can provide, which can reach 2.5 Gbit/s in the downstream direction and 1.25 Gbit/s in the upstream direction besides the possibility of their diversity of services [4].

Aiming to ensure the reliability and availability of embedded software for optical networks, two experiments will be discussed in this paper. The first one is the regression testing automation and the second one is the automatic fault injection, both applied in GPON (Gigabit Passive Optical Networking), a project of CPqD (Centre for Research and Development in Telecommunications).

GPON networks are solutions for high-capacity triple-play services access (voice, video and TV) using data transmission over fiber [3]. The connections are established between the OLT (Optical Line Termination), located at the service provider, and the ONU (Optical Network Unit), located at residential or corporate areas, as shown in Fig. 1.

GPON consists of software units that are embedded on OLT and ONU's boards. Each OLT fiber must be connected

to almost 128 ONU's and at a distance of up to 12 miles (The standard ITU-T sets 37 miles) [4].

At CPqD, the GPON pilot project has been placed on trial in conjunction with the Experimental Design High-Speed Network - GIGA, also developed at CPqD [3], and has been in operation for at least six months. This pilot also includes mobility network of Wimax, WI-Fi and Ad Hoc. In this project, the GPON network proved to be an efficient mode of transmission, and was able to carry data over long distances with high transmission capacity and quality [5].

Due to the large number of features of GPON system and also due to GPON design has over a million lines code, a technique of regression testing was used. The regression tests are often used when some correction is made in the software or when some functionality is inserted or deleted. It is a very popular technique and extensively used, since all features can be reviewed in new versions. It is currently the best technique to be applied for this purpose. According to Rothermel [6], regression testing is a technique performed on a modified program to ensure that changes are correct, and without damage of unchanged portions of the program. In this project, the regression testing were automated and executed in the GPON system, ensuring reliable results in a shorter execution time. In order to succeed in it a customization was necessary and for this purpose a robot was developed to perform the tests.

A complementary test technique is also used, that is a validation that automates the fault injection in GPON system. . Fault injection is applied based on state machines that describe the transition among the possible system states, which are used to define TC's in order to achieve greater coverage of the system. The greater the amount of coverage of the code exercised, the greater the quality of software (less failure). However, to increase the code coverage implies to improve the mechanism for fault tolerance [7].

This later experiment has used finite state machines, which consist of a single set of states. They have an initial state, and one or more final states, depending on their execution flow. The state transitions occur when an event is generated, and when this happens, the states are updated [9].

The bugs found by these experiments are classified according to the following criteria: (i) low priority bugs - are trivial flaws or improvements to be made, which do not affect the operation of the system; (ii) average critical bugs - are defects that can affect network performance, but do not lead to crash or interruption of its operation; (iii) highly critical bugs - can interrupt the system or compromise network performance; (iv) very highly critical bugs - can totally undermines the functioning of the system or some basic functionality.

The test automation experiments have brought significant results, requiring great effort to run only at the beginning of each implementation for the generation of

TABLE 1: Preparation of Test Cases	
Test Type	Test
Within the limits (success)	al 0 0, al 7 7, al 1 5, ...
Outside the limits (error)	al -1 -1, al 8 8, al 8 4, ...
Errors test	Non-execution of pre-conditions, re-executing the same command, ...

TC's and tests the adequacy of the robot. Later, this effort was rewarded by the possibility of running the tests on each new version, only updating TC's.

After this introduction, Section II discusses the regression tests executed manually and automatically through experiment executed in the GPON network. Section III comprehends the fault injection experiments and its automation. And Section IV discusses the advantages achieved in the implementation of the automation of regression testing and automated fault injection and the conclusions.

II. REGRESSION TESTING

The GPON project development has been taking four years, and at least once a month a new functionality is released. Due to the extensibility of its functionality and project size, it is necessary to check each new version in order to verify if new errors were not inserted in areas of the system that were previously tested.

The features of the GPON project can be represented by commands. Currently, the project has 110 commands with more than 850 TC's.

The development of TC's in GPON was created from a test plan document, and they are based on submitted inputs, and the outputs (black-box testing). The inputs are sent to the software, and each command has its own respective parameters and preconditions. The results of these commands are displayed to the operator. For example, an input command to enable the ONU is sent, and the expected output is the activation and deployment of this device.

Another instance is the "activate_link" command that has two parameters, device_id and link_id, each parameter can vary from 0 to 7. Based on these conditions, as shown in Table 1, TC's "activate_link" (al) command may be designed: within these limits, outside the limits, or error conditions, among other cases. These conditions were held for the creation of TC's of all available commands.

A. MANUAL REGRESSION TESTING

In the execution of manual tests, the commands are sent by the operator to the OLT software through an interface, as shown in Fig. 2. This interface displays the output, and

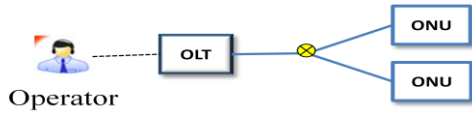


Fig. 2. Execution of manual tests

events and alarms generated by the application can also be seen. All TC's stored in spreadsheets had: the preconditions, the test itself, the expected result and the priority of the test run. A defect is observed when the result of TC is not equal to the expected output.

However, when regression testing is performed manually they are usually very repetitive and their conduction requires considerable effort. Due to the volume of TC's developed, the execution has become unfeasible. It required the development of tools that automate it, as discussed in the next topic.

B. REGRESSION TESTING AUTOMATION

Automatic execution of regression testing allows the reduction of the execution time of the TC's, and may lead to increased coverage of the software as the testers are able to conduct a large number of tests. Moreover, testers are free to focus their efforts on other types of test or tests that cannot be automated.

A test robot was developed at CPqD to automate regression testing. It was implemented in C language, Linux operating system and it communicates with the OLT equipment through TCP/IP sockets. The robot is responsible for collecting the outputs and comparing them with the expected results.

TC's that were manually developed were stored in a database consisting of text files. These files have the test script to be executed, and the expected results. Only one of the commands cannot be automated, due to the hardware characteristics.

After the execution of TC's, the robot generates a report composed by these executed TC's and the execution statistics, such as TC's that were successful and the presented errors. This report can be sent to the entire team automatically.

C. RESULT OF REGRESSION TESTING AUTOMATION

During one year of automatic execution of regression tests in GPON, we found 206 failures (distributed during the time), as shown in Fig. 3. In this chart, one can see that the period of deployment of the robot (June-October, 2009) a large number of bugs were found. Another period when the number of bugs has increased was the one between March and June 2010, which was the time when critical new features were implemented. In Fig. 4, the reported bugs were separated by criticality.

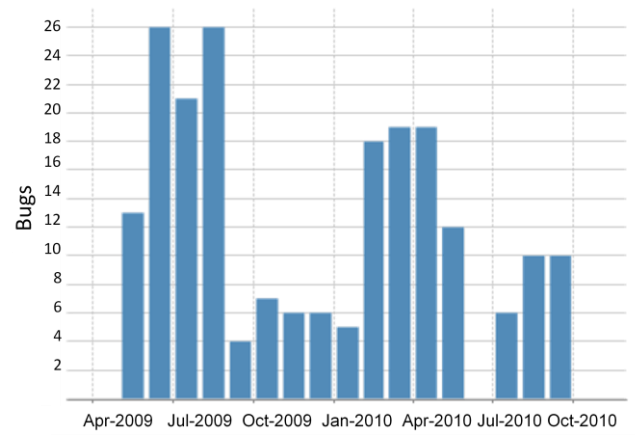


Fig. 3. Total of found bugs

The execution time of regression testing has been reduced from 4 days to 10 hours. This resulted in time saving and allows executing the tests overnight. During business hours it is only necessary to check the generated report, the registration bugs and updating TC's. With the time saving it was possible to think of our features for the robot, allowing the creation of another experiment that will be addressed in Section III.

III. FAULT INJECTION

Telecommunications systems must have high availability and they must be able to provide the requested services, even in adverse conditions. Thus, one way to validate these systems is to verify if they are fault tolerant, i.e., they are able to deliver the service correctly even in the presence of faults [8]. Fault-tolerance is one of the essential characteristics for systems that need to ensure high dependability.

By developing systems that require high dependability, just implementing fault-tolerance mechanisms is not enough. It is also equally important to validate them in order to ensure they are correctly implemented, i.e., that all the services offered by the system are provided according to their specifications. To validate the implementations one can use any means designed to achieve dependability, such as: prevention, tolerance, removal and fault forecasting [8].

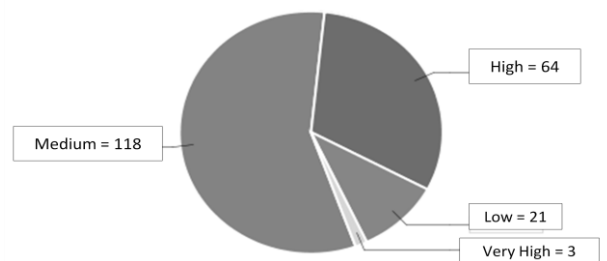


Fig. 4. Criticality of found bugs

A technique that can be used to check whether the system is fault-tolerant or not is the application of fault injection, which aims to observe the behavior of the system in the presence of faults that were deliberately included in order to validate the system under analysis.

A. MANUAL FAULT INJECTION

The tests performed attempted to emulate the interruption of communication between the OLT and the ONU's. In an operational system, these failures may occur due to breakage of optical fibers or loss of signal, for instance. In an ideal system, when such events occur, after the fibers are replaced or the signal is recovered, the components must be reconnected. The OLT and the ONU's connected to this OLT should return to their previous state without human interaction.

To conduct this test manually, the fiber connected to the ONU's and the OLT was removed and the behavior of the system was observed. However, there were two main difficulties: when a failure was discovered: it is difficult to reproduce it since the exact moment of the interruption is unknown and it is difficult to ensure that all these moments are being covered by tests.

To minimize these difficulties and increase test coverage, it was decided to perform the fault injection based on state machines of the embedded software, which will be detailed in the next subsection.

B. AUTOMATIC FAULT INJECTION

The experiment is detailed in Fig. 5 that is composed by OLT and ONU's, a Test Robot containing OXC (Optical Cross Connection) equipment and controller software, and a switch. The switch performs the connections between network devices.

The OXC was inserted into the environment to control the communication between the ONU's and the OLT. It is composed by optical switches that are able to connect and disconnect the fibers by commands, and it was used to inject faults in the test environment.

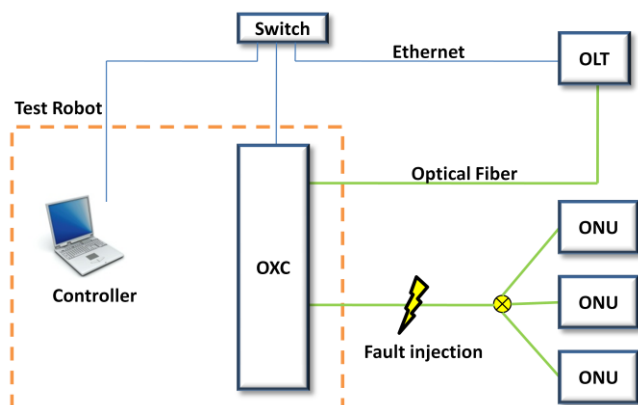


Fig. 5. Operation of the experimental tests

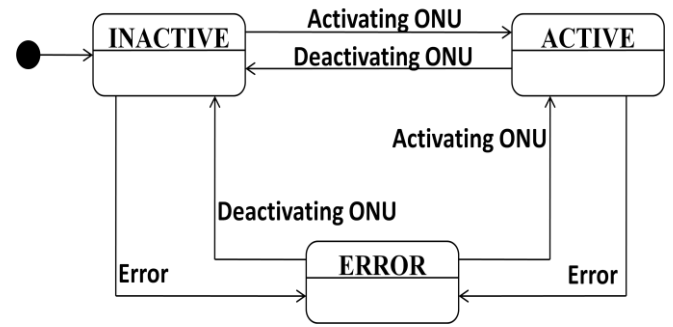


Fig. 6. Example of ONU state machine

The robot used in regression testing has been adapted to run this experiment. In this case it is responsible for:

- sending commands to the OLT;
- receiving the system logs (the logs have communication information between the OLT and the ONU's);
- sending commands to the OXC and;
- Running the TC's and analyzing their results.

The TC's were derived from the OLT's state machine, therefore it is in the state transition that a greater likelihood of unexpected events normally takes place. In Fig. 6, there is an example of the ONU state machine. Specific commands are required to enable or disable an ONU. In case of failure during activation or deactivation of the ONU, its status can be changed to Error. In the case the failure is tolerated, the ONU returns to the state prior to the failure event. In Fig. 6, due to confidentiality reasons; the state machine was modified in this paper.

Automated tests will run from the receipt of logs by an instance of the tests robot. As shown in the sequence diagram in Fig. 7, the robot sends commands to an instance of GPON. When a specific state log is received, the instance of the robot will send a command to the instance of the OXC to interrupt the communication between the OLT and the ONU, injecting the fault in the system. After a period of time, the communication is re-established, and the logs are analyzed in order to verify the behavior of the OLT and the ONU's. While the robot does not receive these logs, the system continues processing the normal execution flow.

TABLE 2
Scenario Explored on Tests

Scenario Testing	Situation
Scenario 1	1 ONU connected without flow
Scenario 2	3 ONU's connected without flow
Scenario 3	1 ONU connected with 1 flow
Scenario 4	2 ONU's connected with 1 flow
Scenario 5	1 ONU connected with 5 flows
Scenario 6	2 ONU's connected with 5 flows

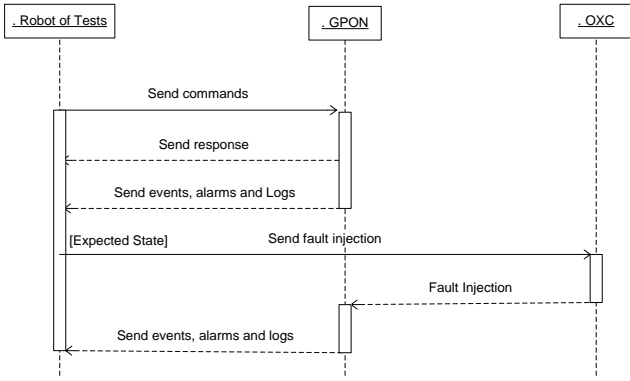


Fig. 7. Sequence Experiment Diagram

The period that the application was inoperative, i.e., with no critical event, the test results obtained were the same if interruption duration is 2 seconds or 2 minutes.

We used nine state machines with presented from 3 to 24 transitions states, as shown in Table 3. In this way, some tests were performed with the interoperation of the states, when two or more state machines operate concurrently by threads, being processed in a seemingly simultaneous way, as shown in Fig. 8.

In Fig. 8, the ACTIVE state of Fig. 6 is detailed. This state is composed by other state machines as CONTROL and VERIFY. The execution of these "sub-state machines" occurs concurrently until all state machines reach the final state. When the CONTROL and VERIFY states reach their final states, the controls of the two sub-states competitors come together again in a single stream, and the state is updated to ACTIVE.

The model states represent the possible behaviors of the system, and the test scenarios derived from them. For the tests execution, 95 TC's was executed that comprehended all the transitions of states machines listed. In each execution of TC's, the scenario was changed, the number of ONU's connected and the number of these flows connected to ONU's could be altered, as shown in Table 2.

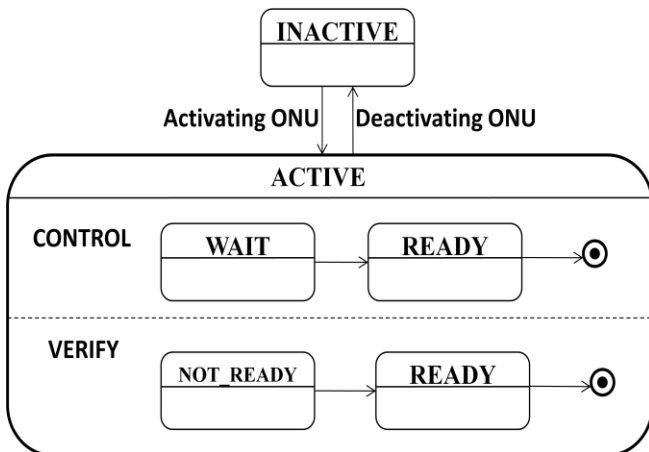


Fig. 8. Example of interworking between state machines

TABLE 3
RESULTS OBTAINED WITH EXPERIMENT

State Machine	Number of transitions	Medium criticality bug	High criticality bug	Total bugs
A	12	2	0	2
B	4	3	0	3
C	3	0	0	0
D	22	6	2	8
E	12	11	0	11
F	4	2	0	2
G	6	0	0	0
H	8	0	0	0
I	24	0	0	0

C. RESULTS OF THE AUTOMATIC FAULTS INJECTION

After assembling the scenario shown in Fig. 5, the three TC's created for the experiment were applied in different scenarios (Table 2), and the results are shown in Table 3. The first column of Table 3 shows the state machines. The number of transitions of each state machine is presented at the second column. The number of bugs found in medium criticality is shown in the third column and the number of bugs with high criticality is in the fourth column. The last column shows the total number of bugs found regardless of their criticality. Most of the problems found were bugs of medium criticality. However, high criticality bugs were also found, which could have stopped the system or compromised its normal performance, making the system work partially, if the faults had been inactivated.

In Table 3, the state machines identified as D and E presented more bugs than any other state machines. A possibility is that they have a higher frequency of use, and therefore are the most critical. In the state machine D, highly critical bugs were found, leading the system to crash.

It is noteworthy that the system has been operating in a pilot project for six months, and prior to this experiment, white box tests were also carried out in the code of implementing each state machine, in order to check if the algorithm was consistent with the proposed diagrams during the development project. Moreover, during previous phases of testing during the development, manual fault injection tests were performed, in which the fibers were simply removed and replaced in the equipment at randomly.

All bugs found in the fault injection experimental tests reported in this paper had not been previously observed. These faults in a commercial system in operational phases put at risk the reliability and availability of the system and must be corrected. Besides the system reliability, the organization credibility is also at stake as the system is critical for the client.

IV. CONCLUSIONS

With the aim of achieving greater reliability and availability of systems, the use of automation in testing optical networks is essential. Test automation can improve the coverage of tests, reduce redundant manual test execution, maximize the accuracy of test results and increase repeatability.

The results of the experiments provided from automation of regression test revealed a large number of bugs. Moreover, automation of regression testing saves time that allowed the test team to think of making new types of tests. This resulted in the automation of fault injection testing to be included in the process.

The use of automation of fault injection validation in the state machines brought improvements to the development and debugging process of the organization. Without using these techniques developers would have much more work trying to reproduce manually failures found by testers. Besides, the preparation of TC's from state machines allowed a better coverage once all states of the OLT can be covered.

The joint use of the techniques to automate tests and to inject the faults allows validating the operation of state machines in the presence of unexpected situations, improving the quality of the tests performed.

The failures reported by the tests that used automatic fault injection technique could not be identified with the manual techniques, once this technique has a wider breadth of coverage of state machines when compared to previously used techniques. After the failures detection, their removal was facilitated by their knowledge of the exact location of the bug, and the exact time the crash occurred. This information is a precious one for the development team and greatly facilitated the system traceability.

This technique increases the probability of finding faults that are difficult to reproduce manually. The transitions from one state to another are short-lived (few milliseconds) and due to the short transition interval the manual test is impracticable, causing an inadequate coverage of the system.

The bugs found had not been reproduced manually and are more critical than those who had already been found by manual testing.

For the team responsible for the GPON system, the use of this technique was very suitable as failures could be reproduced as they occur in the field. Before the use of this technique it was not known if the system was able to treat them. The results of these experimental tests showed that the system must still be improvement to meet the needs of systems with high dependability. Thus, one of the next steps is to inject faults in the communication among components, in order to adjust the system so that it does not behave in unexpected ways, making it tolerant to such failures.

The fault injection experiment can be applied in any software application that can be represented by a state machine, and where communication can be interrupted.

Also, in future work, the objective is to observe the behavior of the GPON system when faults are inserted in the communication protocol called OMCI (ONU Management and Control Interface), which controls communication between the OLT and ONU's.

ACKNOWLEDGEMENTS

This research was conducted with the support of the Graduate Program of FT/UNICAMP - School of Technology and the Centre for Research and Development in Telecommunications (CPqD). Also, the work is partially supported by CAPES.

REFERENCES

- [1] Sommerville, I.: Software Engineering. Sixth Edition. Pearson Addison Wesley. (2003).
- [2] Clark, J., Pradhan, D.: Fault Injection: A Method for validating computer-system dependability. IEEE. Computer Society Press, Los Alamitos, CA, USA (1995).
- [3] Tendências em Redes Ópticas de acesso e Tecnologia GPON – “Trends in Optical Networks and Technology Access GPON” – CPqD. Available at: <http://www.cpqd.com.br/file.upload/p-3_cpqd-giga_atilio-e-regiane_14-05-08.pdf>. Last accessed on August 1, 2010.
- [4] ITU-T G.984.1: Gigabit-capable passive optical networks (GPON): General characteristics (03/2008). Available at: <<http://www.itu.int/rec/T-REC-G.984.1-200803-I/en>>. Last accessed on August 1, 2010.
- [5] Martins, L.; Pozzuto, J.; Mokarzel, M.; Giolo, F.; Bonon, E.; Freire, M.; Junqueira, I.: Fornecimento de Acesso em Banda Larga com Solução Híbrida GPON, WiMAX, WiFi-Adhoc e Mesh CPQD – “Provision of Broadband Access Solution with Hybrid GPON, WiMAX, WiFi-Mesh and Adhoc CPQD”. Infobrasil. (2010).
- [6] Rothermel, G., Harrold, M.: Framework for evaluating regression test selection techniques. Proc. of the 16th Int'l. Conference on Software Engineering, Sorrento, Italy, p. 201-210. (1994).
- [7] DeMillo, R., Li, T., Mathur, A.: Architecture of TAMER: A Tool for Dependability Analysis of Distributed Fault-Tolerant Systems (1994).
- [8] Avizienis, A., Laprie, J. C., Randell, B.: Fundamental Concepts of Dependability. UCLA CSD Report n. 010028, LAAS Report n. 01-145, Newcastle University Report n. CS-TR-739 (2001).
- [9] Thomas, D.; Hunt, A.: State Machines, IEEE Software, v.19 n.6, p.10-12 (2002).