# A Model based on Priority Scheduling and State Transitions for the Automation of Integrated Spacecraft Testing

Viswanathan.P.C, Sheena Jose, Usha Bhandiwad

*Spacecraft Checkout Group*

*ISRO Satellite Centre, Bangalore*

*viswan@isac.gov.in,sheena@isac.gov.in,usharb@isac.gov.in*

***Abstract* — This Paper presents the design of a mechanism for the automation of the Integrated Spacecraft testing. This model executes the pre-defined sub-system test schedules in parallel with the various activities for the performance evaluation, anomaly monitoring and safety-critical operations on the Spacecraft. It guarantees unmanned testing of the complex spacecraft sub systems in a controlled environment where the sub-system test schedules, auto-actions, safety-critical operations, user-interactions, background activities and all function in harmony. The continuous monitoring and evaluation of the overall spacecraft health will be carried out during all sub-system testing and invokes the appropriate safety-critical test procedure on detection of an anomalous behaviour.**

**There exists only one interface for sending telecommands to the spacecraft. The user submits one or more pre-defined sub-system test schedules for execution. The sub-system test schedules will contain the execution of a number of test procedures requiring commanding the spacecraft. There will be multiple alert and notification tasks and critical-background tasks running in parallel and performing specific monitoring and evaluation of the spacecraft parameters. These tasks will initiate the execution of certain test procedures on detection of an abnormal condition on the spacecraft. These entire spectrums of activities are asynchronous and non-deterministic in nature. This requires a scheduling scheme for test procedure execution in which the 'most important' one will be taken up for execution at any point of time. This will achieve the automated testing of the complex spacecraft sub-systems**

**This paper covers dynamic spacecraft test environment, the various sources of the test procedures, interrupt handling , the priority assignment mechanism, scheduling algorithm based on priority, the cyclic design of the execution control logic, state model, and state transitions of the test procedure executions.**

***Keywords*—Integrated Satellite Testing, Test Schedules, Schedule Command Language, Checkout Command Language**

## I. INTRODUCTION

The Integrated Spacecraft Testing (IST) involves the detailed testing of the various spacecraft sub-systems and interfaces according to a pre-defined test plan. The spacecraft tests will be conducted during various phases such as Thermovacuum, Vibration Tests, Acoustics Tests, and Pre-Launch .The tests will cover the functionalities, interfaces and performance of the spacecraft-subsystems in various environments which resembles the on-orbit conditions. The Integrated Spacecraft is complex and highly unpredictable due to the dynamic behaviour, concurrent tests and the possible deviation in the performance. Many of the anomalies occur on spacecraft during testing can be catastrophic if not handled properly and immediately.

The test plan for each spacecraft sub-system consists of a set of test procedures written in a highly sophisticated Spacecraft-Checkout Command Language (CCL). Each test procedure conducts a specific test on a spacecraft sub-system. The test procedure use the appropriate CCL instructions for sending telecommands to the spacecraft, setting stimuli for simulating various on-orbit conditions during testing, reading and validating certain spacecraft telemetry parameters and so on. In the present scenario, the CCL instructions in the test procedures will be executed by the Spacecraft Checkout Software sequentially as submitted by the Test Engineer. This activity is entirely manual where the test engineer submits the test procedure for execution one after the other. During the execution of the tests, the systems performing the monitoring and safety of the spacecraft will alert the test engineer through audio and visual alarms if any anomaly is detected. When an alarm is generated, the test engineer takes the necessary action by submitting the appropriate test procedure for execution to deal with the anomaly and to put the spacecraft in a safe condition.

This manual mode of operation is prone to multiple inefficiencies like incorrect-sequencing of test procedures by the user, sub-system expertise requirement during testing, delay in attending to an anomaly, probability for ignorance by the user to attend to a safety-critical behaviour, ambiguity in the order of execution when multiple simultaneous anomalies occur and so on.

The model presented in this paper will eliminate all the manual interventions during the sub-system testing and improve upon the draw backs in the conventional way of Integrated Spacecraft Testing.

This model achieves the automation of the Integrated Spacecraft Testing through three components – (1)Automatic execution of pre-defined sub-system test schedules, (2)Critical monitoring and Anomaly detection tasks, attached with safety-critical test procedure and (3)An execution Control Logic having automatic sequencing of test procedure execution based on priority.

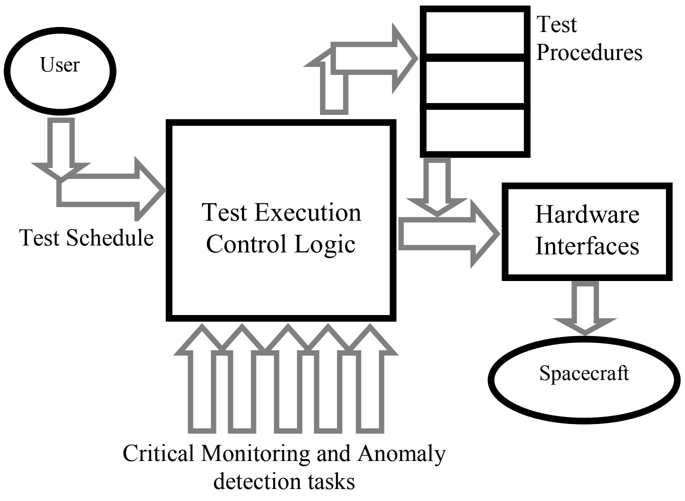
In this method, individual sub-system tests will be bundled in a test-schedule. The test schedule will contain a validated sequence of test procedures. This sequencing and validation will be a preparatory and offline activity carried out by the experts in the concerned sub-system testing prior to the actual execution of the test schedule. This test schedule will then be submitted for execution for conducting the tests on the sub system during Integrated Spacecraft Testing. The test procedures in the test schedule will be automatically taken up for execution during the execution of the test-schedule file. Thus the intervention of the user for submission of individual test procedures will be removed. Moreover, by this approach, the sub-system expertise requirement for sequencing of the tests will also be eliminated.

The various alert and notification tasks and critical anomaly detection background tasks will perform their indented monitoring concurrently while the test schedule execution progresses. An interrupt for the execution of a safety critical test procedure will be generated by these tasks whenever an incorrect behaviour on the spacecraft is detected. The operation of these tasks and their associated safety-critical test procedure specification will be defined before the start of the Integrated Spacecraft Testing. These definitions will elaborately make use of the past-test experiences and sub-system test expertise to model the probable performance deviations and anomalies on the spacecraft and corresponding corrective actions.

For example a background task will read the spacecraft parameters pertaining to the thermal sub system continuously and evaluates them against the expected performance of the Automatic Temperature Controller (ATC). On detection of an abnormal condition against the specified ATC behaviour indicating the mal-functioning of the ATC, the task will initiate the execution of a set of test procedures to shutdown the temperature sensitive elements on the spacecraft.

In similar manner, multiple tasks can asynchronously generate the demands for execution of test procedures. That is, the test procedures to be executed can be any number greater than or equal to one. Since only one interface exists with the spacecraft, the 'active-executing' test procedure can be maximum one at any given point of time. This will be achieved by a state-transition model for the test procedures in which only one test procedure will be in 'Executing' state at any given instant of time. Moreover, each test procedure will be associated with a priority for execution such that the highest priority test procedure will be executed always over lower priority ones This test procedure scheduling will follow a pre-emptive priority scheduling algorithm. A higher priority test procedure can pre-empt a lower priority test procedure in execution. A test procedure can complete its execution in entirety in the executing state if and only if no higher priority test procedures arise during its course of execution.

Figure-1: Model for the automation of Integrated Spacecraft Testing


User
Test Procedures
Test Schedule
Test Execution Control Logic
Hardware Interfaces
Spacecraft
Critical Monitoring and Anomaly detection tasks

## II. SUB-SYSTEM TEST SCHEDULE

Each spacecraft sub-system will have a test schedule. The schedule file logically groups the test procedures to be executed for the sub-system testing in the appropriate sequence. Moreover it also ensures required conditions on the spacecraft and simulation are satisfied before invoking a test procedure execution in the schedule file. This grouping and conditional constructs in the test schedule will be based on the tests, test-conditions, on-board specifications of the spacecraft, test time, test phase and so on. The test schedule will be prepared in advance making use of the sub-system expertise, experiences during the previous spacecraft testing, guidelines, recommendations and standards.

The test schedule will be submitted by the user and executed automatically by the software during the Integrated Spacecraft Testing. By executing the test schedule for a sub-system, all the planned tests on the sub-system and interfaces will be completed. The test schedule will be a text file containing syntactically verified sequence of Schedule Command Language (SCL) instructions. Whenever an instruction to start the execution of a test procedure is encountered in the schedule file, the execution of test procedures will be invoked by passing on to the execution control-logic.

## III. CRITICAL MONITORING AND ANOMOLY DETECTION TASKS

During the Integrated Spacecraft Testing, it is not only the sub-system being tested but many more will be in operation as part of long duration tests or for creating the indented test environment for the test phase. For example, the thermal subsystem will be continuously operational during the 'thermovacuum' tests to evaluate its performance through various thermal cycles and to provide a controlled thermal environment for the other sub-systems on the spacecraft. Moreover, since the spacecraft sub systems are interfaced to attain much functionality, the activities and events on one particular sub-system will have impact on others. This kind of multiple sub-system operation and subsystem interactions makes the spacecraft highly vulnerable to catastrophe if its overall behaviour is not continuously monitored. Such monitoring will detect the problems on any of the sub-system or interface and takes corrective actions immediately to avoid any disaster.

During the automated spacecraft testing, the critical monitoring and anomaly detection tasks will ensure safety action is taken immediately following anomaly detection without any user intervention. These tasks continuously acquires spacecraft parameters and monitors them against one or more pre-defined validation criteria such as limit-violation, stability, toggling, spikes, various updating trends such as linear, increase, decrease etc. If the parameter(s) being monitored shows any abnormal behaviour, immediately an alarm will be raised and interrupt will be generated to execute the safety-critical test procedure attached with the task.

There will be various types of critical monitoring and anomaly detection tasks which will be triggered for execution depending on the test conditions. All the tasks belongs to any one of the categories in the table below

| **Type of Task** | **Purpose** |
|---|---|
| Emergency Break | An unanticipated vital emergency during the test and a pre-defined test procedure to put off all spacecraft sub system will be executed. |
| Critical Parameter Monitoring | The critical parameters of the spacecraft like battery voltage and their expected values which need to be monitored always. |
| Status Monitoring | The list of parameters whose status not expected to change during this particular testing. |
| Behaviour Monitoring | The expected behaviour of certain sub system parameters (like constant, linear, increasing...) |
| Thermal System logic validation | The continuous validation of the temperature sensors, heaters and thermal control system for their expected behaviour and operation. |
| Limit Check | The important analog parameters and their acceptance limits during the testing. |
| Event Monitoring | The sequence of events and timings to be monitored as part of testing of certain on-board autonomy features. |
| Sub-System Specific | Certain complex logics and behaviours unique to a particular sub-system. |

## IV. EXECUTION CONTROL LOGIC

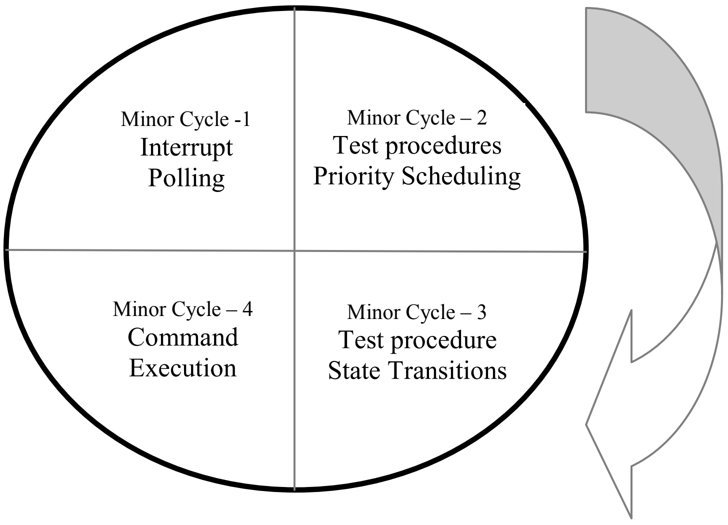
The test procedure execution requests will arise from the test schedule and from various anomaly detection tasks. The execution control logic accepts all these requests and follows a pre-emptive priority scheduling mechanism for test procedure execution.

The Execution Control logic will be the core for handling multiple 'parallel' test procedure executions. The execution control logic will have four components

1. Interrupt interface and Polling mechanism
2. Priority scheduling of Test procedures
3. Test Procedure State Transition
4. Command Execution interface with the spacecraft

All these four components executes four activities in a cyclic manner as in the figure-2

Figure-2: Cyclic Design of the Model



### 1. Interrupt interface and Polling mechanism

An interface will be provided by the Execution Control Logic for raising the interrupts. This interface will be used by the test-schedule-execution and anomaly detection tasks to generate interrupts for executing test procedures. The interrupt requests will have to indicate the source and time. The interrupts will be placed in an interrupt-Q as and when it arrives.

The Execution control logic carries out the interrupt polling during the execution of the minor cycle-01. The interrupt polling reads the interrupt-Q for any pending interrupts and places the test procedure to be executed in the test-procedure-Q along with the time (at which the request arrived) and source. If there are no pending interrupts, then information will be passed on to the minor cycle-2 that there are no 'new' entries in the test procedure-Q

Except for the first time, the interrupt polling will always be carried out at the end of execution of an individual CCL instruction (one undividable operation on spacecraft such as sending of a telecommand), that is after the execution of Minor Cycle-4.

This ensures that the current test procedure can be pre-empted by a higher priority test procedure after the execution of every individual CCL instruction. This accomplishes predictability with respect to the maximum amount of time between an interrupt and the start of the execution of the test procedure required by the interrupt.

## 2. Priority Scheduling of Test Procedures

All the test procedures which are to be executed will be inserted in a Test-Procedure-Q by the Minor Cycle-1. The priority scheduler organizes the Test Procedure-Q and makes the highest priority test procedure at the top of the Q, and initiates its transition to the 'Executing' state.

The Execution Control will have priority for itself which will be same as the priority of the currently executing test procedure. In the beginning, the Execution Control will assign itself a priority lower than the lowest priority in the priority table.

The Test Procedure Priority Scheduling Algorithm

i. Check whether the cycle-1 has indicated that there are no new entries in the Test-Procedure-Q.

ii. Check whether there is event handler notification about the completion of 'wait' by any test procedure in the Q( That is, a test procedure in the Q entered into the 'Waiting' state as a result of execution of a CCL instruction in minor cycle-4 and the event occurred for the wait completion. Waiting will start always in Minor Cycle-4)

iii. If the above checks return true and false respectively, then exit this execution of the activity cycle since there is no need to schedule a different test procedure for execution. The Minor cycle-4 can continue with the execution of the next CCL instruction in the currently executing test procedure.

iv. Otherwise, the test procedures-Q will be sorted in the descending order of priority

v. If more than one test procedure in the Q has the same priority, a 'First Come First Serve' strategy will be followed by the execution control to order the test procedures. The 'time of interrupt' information will be used for this purpose.

vi. If the Current priority of the Execution control is equal to or greater than the priority of the top of the Test-Procedure-Q, then, current test procedure execution will be continued.

vii. If the Execution Control priority is lower than the top of the Test-Procedure-Q, Then the Execution Control assigns itself a priority same as that of the top of the test-procedure-Q and then the test procedure at the top of the Q will be taken up for execution as below

   a. A transition from 'Executing to Suspended' will be specified for the current test procedure in the execution of Minor Cycle-3.

   b. If the test procedure at the top of the Q is in 'Ready' state( Entering to Execution for First time , or finished a "Wait") , then

      i. A transition from 'Ready to Executing' will be specified for the test procedure in the execution of Minor Cycle-3.

   c. If the test procedure at the top of the Q is in 'Suspended' state( Execution was started earlier but was suspended by the Scheduler) , then

      i. A transition from 'Suspended to Executing' will be specified for the test procedure in the execution of Minor Cycle-3.

The Test Procedure Priority Table

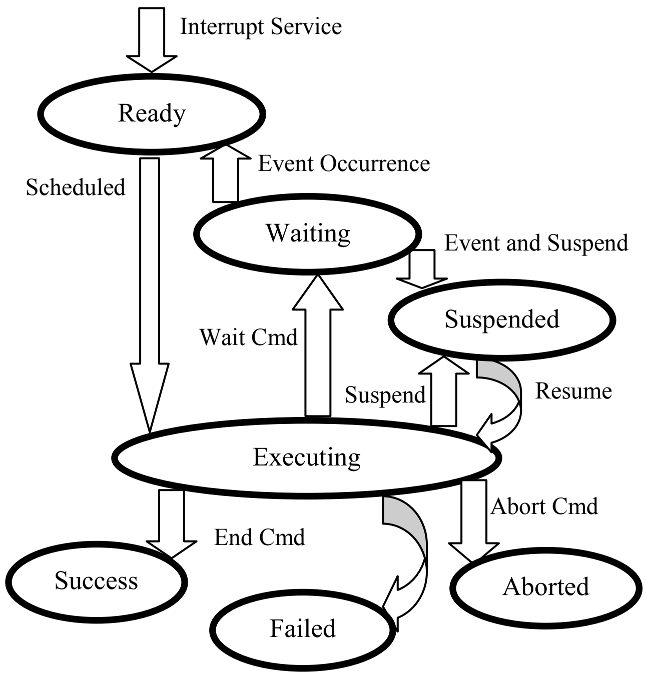
| **Test Procedure Source** | **Priority** |
|---|---|
| Emergency Break | 09(Highest) |
| Critical Parameter monitoring | 08 |
| Status Monitoring | 07 |
| Thermal System | 06 |
| Behaviour monitoring | 05 |
| Limit Check | 04 |
| Events Monitoring | 03 |
| Sub-System Specific | 02 |
| User Submitted Test Schedule | 01(Lowest) |

## 3. Test Procedure State Transitions

A test procedure undergoes various state transitions from the time it is requested for execution till the completion of the execution. The state transition diagram for test procedures is shown in figure-3.

The Test procedure occupies any one of the state from the point it enters the execution control through interrupt till the completion of its execution. The state transitions will be appropriately carried out by the Execution Control Logic

Figure3: Test Procedure State Transition Diagram



State Description Table

| State | Description |
|---|---|
| Ready | Indicates the test procedure has been requested through an interrupt and will be scheduled for execution |
| Executing | The test procedure is currently being executed by executing the commands in Minor Cycle-4 |
| Suspended | The test procedure was entered the 'Executing' state but currently the execution is suspended by the Control Logic |
| Waiting | During the Execution of the test procedure a wait was encountered. The test procedure is waiting for the event to occur |
| Aborted | The execution of the test procedure was forcefully aborted by executing the 'ABORT' instruction |
| Success | The execution of the test procedure has been completed and it meets the success criteria of execution |
| Failed | The execution of the test procedure has been completed and it does not meet the success criteria of execution |

In the beginning, the test procedure will be in the 'Ready' state once interrupt is polled and the test procedure is added in the test-procedure-Q. The Test procedure will go to the 'Executing' state once the scheduler decides to take it up for execution according to the priority scheduling logic in Minor Cycle-2. While in execution of a test procedure, any of the following four things can happen this changes its state from 'Executing'

i. The execution of the test procedure can reach the end (the END instruction) without any pre-emption by a higher priority test procedure. In this case, if the test procedure satisfies the criteria for success (Execution of all CCL instructions in the test procedure was successful and no anomalies detected), its state changes to success; otherwise the state will become failed. This state transition will be detected and set by Minor Cycle-4 which executes the CCL instructions.

ii. An abrupt termination of the test procedure can occur as a result of the execution of the ABORT instruction. The ABORT instruction will be placed in the test procedures to handle error conditions such as spacecraft telemetry check failure which are the pre-requisites for conducting the test. This state transition also will be set by the Minor Cycle-4

iii. The test procedure contains a CCL instruction which results in waiting for an on-board event. These events will be one more conditions on the spacecraft indicated in the spacecraft telemetry. On execution of such 'wait' instructions, the test procedures enters into the 'Waiting' state and waits for the event to occur. This state transition also will be set by the Minor Cycle-4

iv. A higher priority test procedure can pre-empt the execution of the test procedure. For executing the higher priority one, the test procedure will be suspended. The suspend operation brings the test procedure to the suspended state from the executing state. A test procedure in the suspended state will hold all its properties required for resuming its execution at a later stage.

<u>The 'Waiting' State of the test procedure</u>

The test procedure will contain instructions which generates wait. These instructions will be either for waiting an absolute quantum of time (like setting a stimuli and waiting the prescribed amount of time in the test procedure for the simulation environment to stabilize before executing the next instruction) or waiting for an event to occur (certain conditions on the spacecraft or ground systems indicated in one or more telemetry parameters). In either case, the test procedure will be entering into the 'waiting' state immediately.

If a higher priority test procedure pre-empts the 'Waiting' test procedure as a result of the subsequent execution of Minor Cycles 1 and 2, the test procedure will be suspended and moved to the suspended state. The difference between the 'suspended' and 'waiting' states is that in the 'Waiting' state, the test procedure will still be holding the 'execution control' and its properties will not be saved. In the suspended state, the properties of the test procedure will be saved to give way for execution of another one. If a test procedure enters into suspended state from waiting, then the wait for the event will be continued in the suspended state. On resumption to the executing state, the status of the wait will be checked, and if it's incomplete, then the test procedure again will be put in the 'waiting' state

If no higher priority test procedure arises during the waiting period and the event for wait completion (or timeout) occurs, then the 'Waiting' test procedure will go the 'Ready' state and will be continuing with the execution as decided by the scheduler.

The Properties of the test procedure saved during suspend

While suspending the execution of a test procedure, all the properties at that instant will be saved to ensure the accurate resumption at a later time. This includes

i. The status of previously executed CCL instructions in the test procedure and a reference to the next instruction to be executed
ii. Time and the State of the spacecraft sub system as in the recent telemetry check and the values of the critical parameters to be checked on resumption
iii. State of ground support equipments like stimuli settings etc
iv. Level of execution in case of nested test procedure invocation and if the level greater than one, then the reference to the 'properties' of the parent test procedure

### 4. Command Execution

The test procedures will contain a list of CCL instructions. The CCL instructions will be intended to perform certain operation on the spacecraft. These include sending one or more telecommands to the spacecraft sub-systems, reading certain spacecraft telemetry parameters and checking their values against the expected numbers. The CCL also include instructions for decision making, branching and looping (IF, WHILE etc) and for setting and controlling the ground support equipments such as simulators

The execution of the test procedure means the execution of the CCL instruction in the test procedure in the sequence. The execution of every CCL instruction will interface with certain hardware system such as telecommand encoder or data acquisition system. These systems provide the interface with the spacecraft.

In minor cycle-4, the 'current' CCL instruction in the currently executing test procedure will be executed in entirety as an atomic entity. Then the instruction pointer will be incremented by one to point to the next instruction. If the executed instruction is WAIT or END or ABORT then appropriate state transition will be set for the test procedure during the execution of Minor Cycle-3 in the next major cycle.

## V. SIMULATION AND TESTING

The model has been designed and prototype software has been developed. Simulations are made in the lab using the previous spacecraft test procedures, test-data and various simulation tools (for simulating spacecraft telemetry, Test environment parameters and anomalies). The functioning of the prototype was satisfactory yielding unmanned operation covering the entire test spectrum of the spacecraft sub systems. Moreover it provided a deterministic behaviour with respect to the maximum amount of time taken for executing the safety-critical test procedures in case of an anomaly. The overall test efficiency also improved by having standard test sequences and reduced test execution time.

## VI. CONCLUSION

The model based on priority scheduling and state transitions of the test procedures will eliminate the probable manual errors and omissions in the Integrated Spacecraft Testing. Moreover it results in a standalone system which utilizes the sub-system expertise and previous test experiences prior to the starting of the testing for sequencing the test procedures and for modelling probable anomalies and corrective actions. Testing will be conducted independently by the software without requiring expertise on the spacecraft sub-system. This reduces the man power and expertise requirements and dependencies and improves the testing time by a large amount.

## VII. ACKNOWLEDGMENT

The authors would like to thank Shri.K.B Anantha Rama Sharma, Group Director, Spacecraft Checkout Group, ISRO Satellite Centre for his continuous support and motivation.

## VIII. REFERENCES

**[1]** Dan Yu 1, Gang Ye 1, Shilong Ma 1, Naixue Xiong 2, Laurence T.Yang. "The Spacecraft Automatic Testing System based on Workflow", IEEE Asia-Pacific Services Computing Conference 2008
**[2]** "The Design and Implementation of Workflow Engine for Spacecraft Automatic Testing", JOURNAL OF COMPUTERS, VOL. 6, NO. 6, JUNE 2011
[3] Guohua Wangn, YanCui,ShuoWang,XiaofengMeng. "Design and performance test of spacecraft test and operation software", , Beijing University of Aeronautics and Astronautics,Beijing,China Acta Astronautica 68 (2011) 1774–1781
[4] Martin S. Feather & Ben Smith. "Automatic Generation of Test Oracles- From Pilot Studies to Application", IEEE International Conference on Automated Software Engineering, October 1999. IEEE ComputerSociety,pp-63-72