International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 4 Issue 03, March-2015

An Automated Test Execution Kernel for Spacecraft Checkout Operation

Abhijith. R
Project Trainee, SCG
ISRO Satellite Centre
Bangalore, India

G. Raghavendra Rao
Professor & Head, Dept. of CSE
The National Institute of Engineering
Mysore, India

Viswanathan. P. C
Scientist/Engineer-SD, SCG
ISRO Satellite Centre
Bangalore, India

Sheena Jose
Scientist/Engineer-SF, SCG
ISRO Satellite Centre
Bangalore, India

***Abstract*— Automation plays a critical role in spacecraft checkout operations. Automatic Checkout Software System (ACSS) is a set of software products developed for the automation of spacecraft checkout operations. The Test Execution Kernel is a part of ACSS that coordinates with testing of spacecrafts during checkout operations. The Test Execution Kernel at the core of ACSS will carry out the actual automatic execution of test procedures. The previous systems support automation of Integrated Spacecraft Test (IST) limiting on one side manual operation requirements. Is the extent of automation sufficient even if the complexity rises in the checkout operations? Hence, developing a completely automated test execution kernel is important in spacecraft checkout operations. Thus, this problem is investigated in this paper. Firstly, complete automation of spacecraft checkout operations is proposed. A critical part in ACSS that should be automated is then analyzed. The Test Execution Kernel is found to be the critical part of automation and it shall be automated to a larger extent that assures automatic execution of all complex test procedures in checkout operations and repetition of tests as required by various phases of testing the spacecraft. The resulting implementation ensures the correct and error free operation of spacecrafts.**

Keywords-Automation; Automatic Checkout Software System; Integrated Spacecraft Test; Spacecraft; Spacecraft Checkout Operation; Testing; Test Execution Kernel

I. INTRODUCTION

Spacecrafts undergo extensive testing by monitoring thousands of parameters as a part of ground checkout before launch. Spacecraft checkout activities are automated to a greater extent through a set of software products under the name Automatic Checkout Software System (ACSS). ACSS is housed in the Spacecraft Checkout Computer (SCC) that runs as a checkout server. ACSS will acquire the spacecraft parameters in real time.

The Test Execution Kernel at the core of ACSS will carry out the automatic execution of test procedures as a part of checkout operation with minimum manual operations. This paper discusses the idea of completely automating the Test Execution Kernel by assigning the execution of all complex test procedures and manual operations to the automatic engine. Further, the automated Test Execution Kernel operates without placing any restriction on the current execution status of ACSS. The spacecraft test procedures are prepared using Checkout Command Language (CCL). CCL provides structured programming constructs and a versatile instruction set tailored to the need of checkout operations.

The Objective of this paper is to describe an idea of completely automating the spacecraft checkout operations and thus, ensuring early detection of anomalies, automatic test report generation, safety of the spacecraft and enabling to achieve error free operation of spacecrafts.

II. EXISTING SYSTEM

A. Overview

The Test Execution Kernel in the first generation Automatic Checkout Software System (ACSS v1.0) was deployed on VAX/VMS system. Later, in second generation Automatic Checkout Software System (ACSS v2.0) was realized on a DEC alpha based hardware platform under digital UNIX operating system [1]. The subsequent versions of the Automatic Checkout Software System (ACSS v2.1 - ACSS v2.9) were realized on an AMD-64 Opteron/Athlon processor based system which is based on the innovative AMD-64 technology [2][3][4]. This provided a multi vendor, low cost and high performance environment, an effective alternative to RISC based DEC alpha platforms.

The existing Test Execution Kernel is an application at the heart of the Spacecraft Checkout System (SCS) provides a set of services to carryout spacecraft checkout operations with a configurable level of automation, limiting on one side manpower requirements and to the complexity of the whole system to automated processes [5][6].

The existing Test Execution Kernel supports automation of Integrated Spacecraft Test (IST) to certain extent. However, there is a scope for more improvements and more automation of most of the manual operations.

B. Disadvantages

- Limited automation and demands manual operations during testing.

- No standard templates for the test results. The approach of testing varies from one tester to another.
- Lacks performance, reliability and maintenance due to the change in environment where the software functions.
- Phasing out of systems running on DEC alpha based platform.
- Outdated hardware components.

III. PROPOSED SYSTEM

A. *Overview*

The proposed Test Execution Kernel for spacecrafts is designed using a multi threaded approach. The proposed system includes open system architecture with all the interfaces standardized. This uses well proven Checkout Command Language (CCL) for preparing test procedures in advance, very versatile data processing, presentation and analyzing tools [7]. The proposed Test Execution Kernel will have interface with checkout equipments such as telecommand encoder, various simulators, payload checkout system and so on.

The overall functionality of the proposed Test Execution Kernel shall be implemented using four different units.

- Controller.
- CCL Execution Unit
- Interrupt Handler
- Background Task

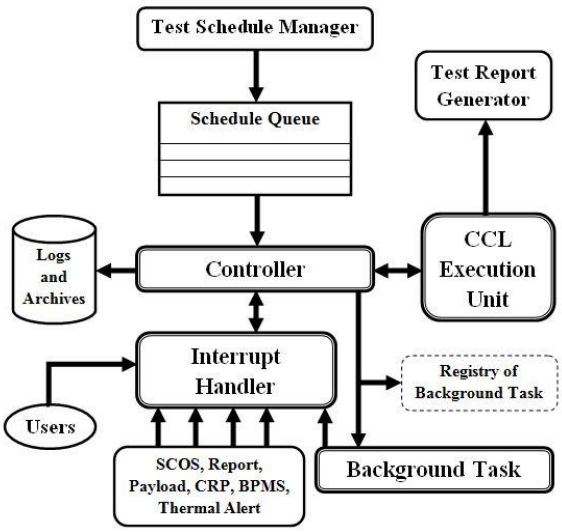

Fig. 1. Architecture of the test execution kernel

B. *Controller*

The controller is the major component of the Test Execution Kernel which coordinates the execution of test procedures and instructions by interacting with the test schedule manager [7], interrupt handler and CCL execution unit. The major activity of the controller is to initiate the execution of test procedures and dispatching it to the CCL execution unit. The controller will receive the inputs for initiating the execution of test procedures from either the test schedule manager or the interrupt handler.

The controller will keep track of the execution status of test procedures for updating the user interface with the current status. Further, the controller has the capability to keep track of the transition of test procedures by saving and retrieving the context of the currently running test procedures, suspending-resuming and terminating the execution.

The controller will also perform the management of background tasks by initiating the task, allocating resources for it, checking its status and terminating the task. The logs of all activities carried out by the controller will be maintained for future reference and prerequisite checks. These logs can be used by test schedule manager and other checkout applications.

The role of the controller will not only be execution sequencing, but it will also do the execution control. The Execution control function ensures veracity of execution of the instructions in the test procedures by proper parse, dispatch and flow-control within the test procedure.

C. *CCL Execution Unit*

The CCL execution unit is responsible for parsing and executing all the CCL instructions of test procedures by interacting with controller and interrupt handler of the Test Execution Kernel.

The major activity of CCL execution unit is to receive the CCL instructions from controller, validate them with respect to syntax and execute the instructions. The execution status will be made available in test execution status region of the user interface.

Upon successful execution, the CCL execution unit waits for the controller to receive next instructions to be parsed. Any error during the execution of a CCL instruction will be sent to the interrupt handler. Further, the appropriate error handling routine for that error condition will be initiated by the interrupt handler.

D. *Interrupt Handler*

The interrupt handler will perform the handling of various alerts, indications and anomalies simultaneously with the execution of the test procedures. The interrupt handler works along with the controller to accomplish its tasks. There will be many sources that generate interrupts. The interrupt behaviors will be asynchronous, i.e. sources can generate interrupts at any point of time. The sources of interrupts can include background tasks, CCL execution unit, user, various anomaly notifying services and so on.

The interrupt handler will communicate with controller for suspending/resuming/aborting the currently executing test procedure. The interrupt servicing will be carried out at the end of execution of the current CCL instruction that is being parsed. Priority will be assigned to every interrupt generated.

E. *Background Task*

The primary function of background task is to monitor the subsystem telemetry/telecommand parameters during testing and thereby validating them by applying various types of user defined computations/processing on them. These computations include trend analysis, statistical computations, mathematical computations etc. Background tasks will continuously monitor various health parameters of the spacecraft. The background task will put the spacecraft in safe condition by taking predefined actions if the monitored parameters go beyond safe limits. Further, it notifies the user with a message about the action taken.

Background tasks will be activated automatically by the test procedure or by the user. Any number of background tasks can be created and terminated dynamically at any time of execution of test procedures. Once a background task is activated, it will perform all the computations and it will update the result tables for test report generation.

F. Advantages

- Error free spacecraft testing with negligible human intervention.
- Reduction of integrated spacecraft testing time by more than 50%.
- Standard way of testing enforced across the spectrum of spacecrafts.
- Standard templates for test reports.
- Early detection of anomalies and ensures safety of spacecrafts.

IV. IMPLEMENTATION

The Test Execution Kernel was designed and developed for the complete automation of IST. The IST automation made the spacecraft testing faster, error-free and also achieved complete elimination of manual operations. Consider asynchronous activities that will be performed during spacecraft checkout operation.

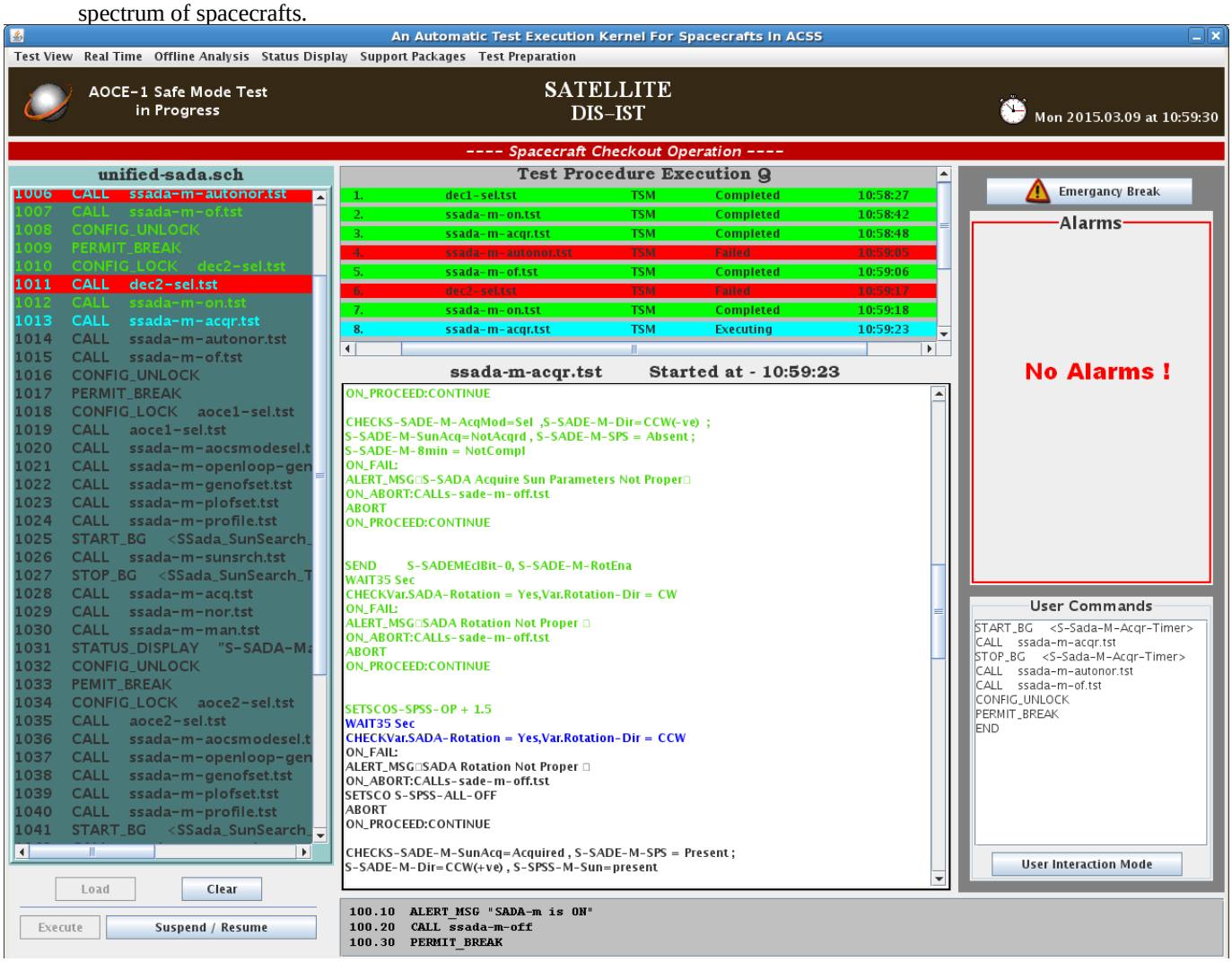

Fig. 2. Implementation of the test execution kernel

A pre-planned sequence of testing activities will be embedded in a test schedule file i.e. unified-sada.sch and it is submitted for automatic execution. The test schedules are organized according to sub-system testing. Each spacecraft sub system will have a specific test schedule for execution. The major component of the test schedule is the names of test procedures to be executed by the Test Execution Kernel for the particular sub system of the spacecraft. The test schedule manager will dispatch the test procedure file names to the controller of the Test Execution Kernel as and when they are required to be executed as per the schedule definition. During the schedule execution, there will be occasions when the user wish to intervene and execute certain commands or test procedures interactively by activating the user interaction mode.

Each test procedure that is encountered in the schedule file is stored in a test procedure queue. Later, the test procedures from the test procedure queue are loaded one by one onto the execution area for automatic execution. E.g. ssada-m-acqr.tst file is currently executing as the active test procedure on the execution area. Every line of the test procedure is passed to the CCL execution unit for executing the CCL instructions. The status of execution is updated in the test procedure queue after the completion of execution

of the corresponding test procedure. The next test procedure in the queue is then loaded for execution. This process repeats until the completion of execution of all the instructions in the schedule file.

There will be multiple alert and notification tasks such as background tasks running in parallel with the scheduled spacecraft testing. These notifications are made visible to the user on the Alarm region of the user interface. All the implemented test execution activities are asynchronous. These activities take place until all the schedule files are completely executed.

V. CONCLUSION

In this paper a completely automated test execution kernel was proposed for spacecraft checkout operations. The automation of the same was designed and developed. It comprehensively enabled error free operation of the spacecraft. Automatic test report generation was also made possible by executing the test procedures using automated test execution kernel. Further, better performance, reliability with respect to change in environment of testing the spacecrafts was achieved.

ACKNOWLEDGMENT

The authors would like to thank Mrs. Usha Bhandiwad, Division Head, Checkout Simulator and Automation Division, Spacecraft Checkout Group and Mr. K.B.Anantha Rama Sarma, Group Director, Spacecraft Checkout Group, ISRO Satellite Centre, Bangalore, India for their generous support. This work was supported in part by a grant from ISRO Satellite Centre.

REFERENCES

[1] Automatic Checkout Software System (ACSS v2.0) and test document for IRS-P4, ISRO-ISAC-IRS P4-TE-0256, August 1998.a

[2] Automatic Checkout Software System (ACSS v2.4) features for INSAT-3A, ISRO-ISAC-T04-4102-02, December 2001.

[3] Software Requirements Specification for Automatic Checkout Software System (ACSS v2.8), ISRO-ISAC-Chandrayan1-PR-1013, October 2006.

[4] Automatic Checkout Software System (ACSS v2.8) updates for Chandrayan-1, ISRO-ISAC-Chandrayan1-PR-2017, March 2009.

[5] Work proposal of automated database manager, ISRO-ISAC-T04-4105-02, May 2013.

[6] System Requirements Specification for ADMS v1.0, ISRO-SCG-CSAD-SW-ACSS-1302, June 2013

[7] Software Requirements Specification for Automatic Checkout Software System (ACSS v3.0), ISRO-SCG-CSAD-SW-2013-06 v1.0, October 2013.