ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 5, Issue 5, May 2016

Generation and Compilation of Test Procedure Files of a Spacecraft

Priyanka.N.Murthy[1], K.R.Shylaja[2], Sheena Jose[3], Neha Misra[4]

PG Student [CSE], Dept. of CSE, Dr. Ambedkar Institute of Technology, Banglore, Karnataka, India[1]

Associate Professor, Dept. of CSE, Dr. Ambedkar Institute of Technology, Banglore, Karnataka, India[2]

Section Head, Spacecraft Checkout Group, ISRO Satellite Centre, Old Airport Road, Banglore, Karnataka, India[3]

Scientist / Engineer, ISRO Satellite Centre, Old Airport Road, Banglore, Karnataka, India[4]

ABSTRACT: ISRO Satellite Centre(ISAC) is the lead centre of the Indian Space Research Organization in the development and operationalisation of satellites for communication, navigation and remote sensing applications. ISRO is gearing up for challenging and demanding targets in the years ahead in terms of developing advanced workload & complex satellite systems, resulting in an average workload of 10-12 satellite projects per year. Checkout of a satellite consists of extensive tests on the spacecraft subsystems and payloads for ascertaining satisfactory performance during spacecraft integration and the defined checkout phases. Spacecraft checkout software is a set of software products developed for the automation of spacecraft checkout operations residing on Spacecraft Checkout Server (SCC) which is responsible for telemetry data reception, equipment monitoring and control, spacecraft commanding, validation and execution of test procedures, automatic logging, off-line analysis and data presentation. To meet the requirements complete automation of spacecraft checkout activities is needed, so that the spacecraft can be cleared for launch quickly, efficiently and accurately from the disassembled testing phase to environmental simulation and through launch preparation. Automated function will proceed by downloading and executing procedures which are constructed on the database, one after the other.

This paper discusses a system which supports an automated environment for the generation and compilation of error free test procedure files which are ready for execution. An experimental result shows the major comparison between old and new system based on manpower, time, software modules and speed of execution and shows that new system is better than old system.

KEYWORDS: Automation, Integrated Development Environment, Database Loader, Test Procedure Files.

I. INTRODUCTION

Spacecraft checkout operations consist of extensive tests on the spacecraft subsystems and the payloads for the satisfactory performance during various stages of integrated satellite testing before launch. Every spacecraft which is built shall undergo rigorous testing during various phases of spacecraft integration. It has led to a considerable reduction in testing effort and time. The in- house developed software system is a suite of applications at the heart of the Spacecraft Checkout System with the objective of providing functionality that test spacecraft and also to automate the ongoing checkout operations.

A. CHECKOUT SOFTWARE SYSTEM (CSS)

Checkout software is used for evaluating the functional performance of the spacecraft during various stages of assembly, integration, various environmental tests in the thermo vacuum chamber with hot and cold cycles, vibrations and acoustics, and the pre-launch at the launch pad. The main tasks of checkout are to provide controlled testing sessions, to provide means for assessing the overall performance of the integrated spacecraft, to simulate the different on orbit modes & evaluate the performance of various onboard subsystems. Hence, Checkout software can be defined as a set of software products for automation of spacecraft checkout operations. It is organized into various real-time, offline and software support packages. Its salient features are:

a. Telecommand verification, generation, transmission and confirmation

b. Real time TM data acquisition and processing
c. Monitoring and control of checkout support equipments
d. Automatic logging of all the events, raw data and test results
e. Plotting of parameters

Automation is required to reduce human effort and avoid manual entry errors while creating test procedures files using Checkout Command Language. The new system namely Test Procedure File Generator integrates editor and the compiler that provides an Integrated Development Environment (IDE) having a user friendly GUI (Graphical User Interface) for the development of test procedure files which is ready for execution by the software. The way to fulfill the need of making automated checkout operations is to ensure complete automation as it is considered crucial for the efficient and cost effective checkout operations. Extensive tests are performed on spacecraft subsystems using standard test procedures. This Test Plan will be converted into a sequence of operations which will be the Test Procedure for conducting a particular test. The executable Test Procedure is written using Checkout Command Language .These test procedure files then undergo a validation process before being subjected to execution.

B. CHECKOUT COMMAND LANGUAGE (CCL)

Checkout Command Language (CCL) is the in-house designed and developed test language which is used for the checkout purposes. CCL is the command language with simple English like instructions which are developed for writing test procedures to be executed by the software for performing checkout operation on the spacecraft. CCL has been developed keeping in mind all the requirements for spacecraft checkout and provision has been provided to define new instructions as and when the need arises for future enhancements. CCL is designed in such a way that the test procedure files created using CCL can serve both as an input file to the computer as well as a document for the test procedure. Each CCL instruction has a defined purpose and syntax. This syntax has to be followed properly for writing error-free test procedures. New instructions can be added easily as per the requirements of the spacecraft checkout operations.

II. TEST PROCEDURE FILE GENERATOR

To test the satellite before launch, Test Plans are designed and Test Procedure Files are generated. All the operations to be performed on the spacecraft for conducting a particular test are specified in the form of a Test plan. This Test Plan will be converted into a sequence of operations which will be the Test Procedure for conducting a particular test. The executable Test Procedure is written using Checkout Command Language. Test Procedure File is generated by the software residing on Checkout software. Algorithm of Test Procedure File generator is shown in Algorithm1.

Test Procedure File Generator that is a component of the Test Preparation Package used in checkout software system. It utilizes the present system database for executing and validating the test procedures. Test Procedure File Generator follows object-oriented design. Figure 1 shows the architecture of test procedure file generator. It is comprised of three design entities: GUI Based Editor, Compiler and Database. The software system named as CCL Execution Unit provides an integrated development and execution environment for CCL test procedures while ensuring safe automation of checkout operations. Test Procedure File Generator shall provide an Integrated Development Environment (IDE) with user-friendly Graphical User Interface (GUI) and editor features to:

- Generate new Test Procedure Files.
- Edit and Format existing Test Procedure Files.
- Parse and validate Test Procedure Files to make ready for execution by Test Execution Engine.

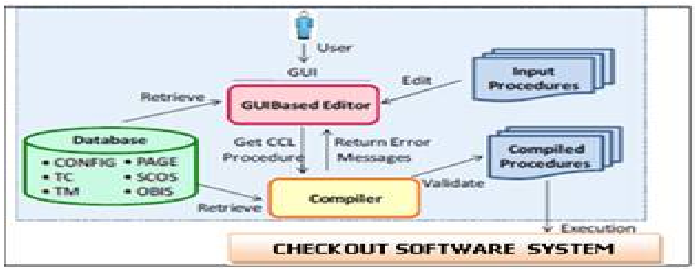


Figure 1: Architecture of Test Procedure File Generator and Executor

Algorithm 1: Test Procedure File Generator

```
Step 1:    Start
Step 2:    Reads checkout configuration files of all kinds of information
Step 3:    Establishes a connection with the test procedure database
Step 4:    Connection not successful goto step 15
Step  5:   Checks for available database records in the memory
Step 6:    If user selects to create a new test file then goto step 8
Step 7:    Allows user to input ccl instructions with appropriate arguments and
Step 8 :   mnemonics from the database
Step9:     Select either New or Open
Step10:    If it is New goto step 11.
Step11:    If it is Open goto step 12.
           Editor screen gets displayed having a header field, checkout
Step12:    command instructions are inserted using  Insert Instruction option
Step13:    (specified from the database)
           Open the existing Test Procedure Files and goto step 11.
           Stop
```

III. DATABASE LOADER

Database Loader (DBLoader) is a component of the application developed to perform core processing tasks. The component defines the data structures for all the databases which are shared by multiple tasks. These data structures can be utilized for loading database contents into shared memory by Database Loader or into process memory by other tasks of checkout software system. It is held responsible for reading tables for databases which are shared by multiple tasks, and loads the required data to shared memory. The data stored in the shared memory is accessed by both real time tasks and the offline tasks respectively.

Database Loader defines a main class and different classes to represent each database of the software system. Figure 1 shows the loading of database as needed by different processes and also loaded database can be accessed from the shared memory.

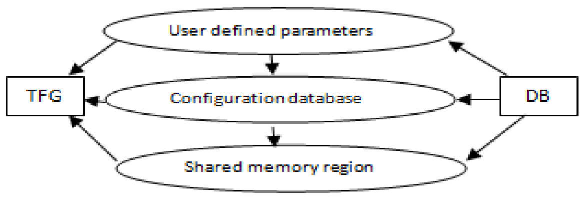


Figure 2: Loading of database

Database Loader is responsible for loading the tables required by multiple tasks (not the task-specific ones) into shared memory using defined data structures. The structures in shared memory are then read/updated by various checkout software online tasks and read by offline tasks. Task-specific database tables are read/updated by ACSS tasks directly from MySQL database or loaded into their own process memory for use. Database Loader will read-lock the required MySQL main tables and then loads them into shared memory.

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 5, Issue 5, May 2016

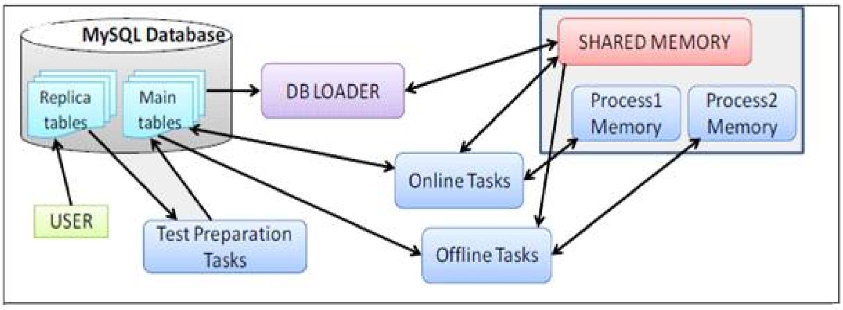


Figure 3: Overall System Design of Database Loader

The main functionalities of Database Loader are described in brief below:

- MySQL_DB_connect-This method is responsible for establishing a connection to the MySQL database using the required driver.

- Shared_memory_create_and_attach–This method is responsible for creating the shared memory region for each database using the defined data structures and attach to these regions.
- Shared_memory_load_and_update- This method is responsible reading the data from MySQL tables and populating the shared memory.
- MySQL_DB_update - It removes the read lock from MySQL tables and puts a write lock to copy the updates from replica tables. It avoids inconsistent data in MySQL tables. After updating main tables to the new version, it also updates the shared memory regions, through the above described module.

IV. USER INTERFACE FOR TEST PROCEDURE FILE GENERATION

Some of the salient features provided by the GUI based editor are as follows.

- **Start up and Configuration**

The software shall start its execution by establishing a connection to the system for which IP address, user name and password are specified in the software configuration file. Software shall establish a connection with MYSQL database of the connected system to read and load all database tables into memory. MYSQL database contains the configuration database tables-'Configuration' and a template table specifying its fields named as 'Configuration Template'. The software shall read the spacecraft configuration from configuration table using the configuration template table, set up the required variables and path and launch the GUI.

- **Database Validation and Loading**

These database tables shall used by the compiler for validation of CCL instruction syntax. Editor shall utilize the database tables loaded into memory to produce display list of arguments for a command instructions so as to automate editing of Test Procedure Files. User shall be able to view the database tables as files through a menu option on the GUI.

- **Creation/Import of Test Procedures**

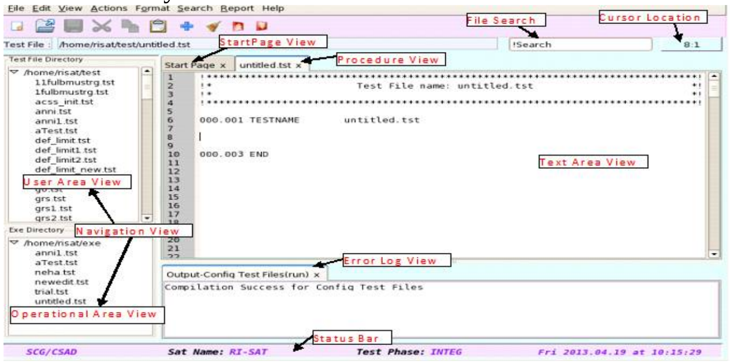
The software shall provide functionality for Creation of a new Test Procedure Files, Import/opening an existing Test Procedure File, Define and set attributes for created Test Procedure file, Fill Automatic insertion of header in newly created files and in imported files detected with improper header. To create /modify/updates the test Procedure Files, Document Editor is used in which basic editor features as well as some additional features are made available.

- **Insert instruction option**

Automatic insertion of new step number with the appearance of a pop-up list is seen which shall contain the name of all the CCL instructions in order. User can refine the list by typing characters at the cursor location. It lists selections and user-typed special characters shall trigger automatic display of pop-up lists for further completion of instruction step.

- **Navigator**

View of user home directory as a tree structure. View of "exe" directory. Successfully compiled Test Procedure Files are time-stamped and sent to this directory.



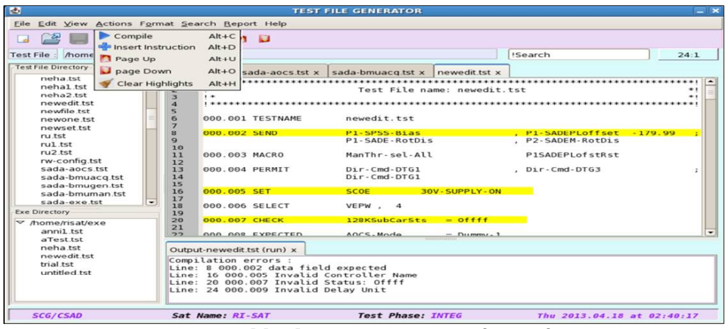
```
!*************************************************************************!
!*                       Test File name: untitled.tst                    *!
!*                                                                       *!
!*************************************************************************!

000.001 TESTNAME        untitled.tst


000.003 END
```

```
Compilation Success for Config Test Files
```

Figure 4: GUIBasedEditor Screen for Test Procedure File Generator

- **Tabbed Document Interface and View**

Multiples files opened as tabs for editing and formatting as shown in figure 4. Tab can be activated by clicking. Only one tab active at a time. Tab switch asks for saving the file in previously active tab. Any tab closed by clicking a 'x' mark button on right-top corner of the tab. User can view the selected database put in the shared memory region.



```
!*************************************************************************!
                       Test File name: newedit.tst                       *!
!*                                                                       *!
!*************************************************************************!

000.001 TESTNAME        newedit.tst

000.002 SEND            P1-SPSS-Bias              , P1-SADEPLoffset  -179.99  ;
                        P1-SADE-RotDis            , P2-SADEM-RotDis

000.003 MACRO           ManThr-sel-All              P1SADEPLofstRst

000.004 PERMIT          Dir-Cmd-DTG1              , Dir-Cmd-DTG3              ;
                        Dir-Cmd-DTG1

000.005 SET             SCOE          30V-SUPPLY-ON

000.006 SELECT          VEPW ,   4

000.007 CHECK           128KSubCarSts     = offff

000.008 EXPECTED        AOCS-Mode         = Dummy-1
```

```
Compilation errors :
Line:  8 000.002 data field expected
Line:  16 000.005 Invalid Controller Name
Line:  20 000.007 Invalid Status: offff
Line:  24 000.009 Invalid Delay Unit
```

Figure 5: Tabbed Document Interface of GUI

- **Compile**

Parses and validates all the CCL instruction steps at a time. After each compilation, each of the detected errors is listed along with line number and instruction step number at which it occurred. Each line of the Test Procedures file containing error is highlighted. File selected to compile is automatically saved before compilation. Algorithm for compilation of created Test Procedure Files is given in Algorithm 2.

Algorithm 2: Compilation of Test Procedure Files

```
Step 1:   Start
Step 2:   End of CCL test procedure file goto step 9
Step 3:   Read first/next line
Step 4:   Extract token from file is not valid goto step 8
Step 5:   Extract succeeding token
Step 6:   Not a valid CCL instruction goto step 8
Step 7:   Call CCL instruction parse and validate function
Step 8:   Update error log and parse status then goto step 3
Step 9:   Stop
```

V. RESULTS AND DISCUSSION

Ccl execution unit provides a automated and integrated platform for creation, validation and generation of test procedure files for any spacecraft checkout. It also enforces automated creation/editing of test procedure files having an defined format in the Checkout operation. This has improved the standard of process of preparation of test procedure files which has also brought down chances of reduction in time and human errors.

A. Experimental setup
This application has been developed using Qt Creator and C++ Language on UNIX/LINUS platform with compiler that supports the same. Test procedure files are created by retrieving the data stored in the database using DBLoader and later compiled using CCL Compiler prior to execution. Processor used for the development is 3.5Ghz 64 bit Quad Core with RAM size about 8GB and Harddisk size approximately 1TB.

B. Comparison
CCL has provided an efficient and improved performance in its execution speed. Recovery is possible where errors occur. The time taken for testing a spacecraft has reduced down at the ratio 4:1 and also manpower has decreased by the ratio 4:1.

VI. CONCLUSION & FUTURE WORK

Procedures from the previous checkouts are provided to new test engineers, as part of their training and to make them understand satellite testing faster. CCL has been evolving with additional features and instructions and the present system is also getting updated with new libraries in this process. CCL is found to be very simple, easy to learn, use from test engineers point of view. Test engineer can prepare all the procedures required for spacecraft testing offline and validate them using CCL compiler. The valid test files can be invoked at the test console during spacecraft testing either in Automode or SingleStep Mode by the test engineer. Though all procedures are validated offline, in case of interactive mode of operations, Test file interpreter itself will provide the user with the syntax errors which can be corrected on-line before execution.

ACKNOWLEDGEMENT

The authors express thanks to Mrs. Muthumani Rajan, Group Head, SCG, ISRO Satellite Centre, Mrs. Vasantha Kumari U.N, Group Director Spacecraft Checkout Group, ISRO Satellite Centre and Mr. Nageswara Rao. M, Deputy Director, Integration and Checkout Group, ISRO Satellite Centre, Bangalore, India for the valuable guidance, support and inspiring innovation throughout the realization of the software. Full support was granted from ISRO Satellite Centre for this work.

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 5, Issue 5, May 2016

ACRONYMS

GUI	Graphical User Interface
IDE	Integrated Development Environment
TC	Telecommand
TM	Telemetry
TFG	Test File Generator
CCL	Checkout Command Language
CEU	CCL-Execution Unit

REFERENCES

1. Dan yu, Gang Ye, Shilong Ma, Naixue Xiong, Yang, The Spacecraft automatic testing system based on workflow, L.T. Asia-Pacific Services Computing Conference,2008 IEEE.
2. Vasantha kumari, U.N. CCL:A test language for automating spacecraft checkout operation Aerospace Conference, 2011 IEEE.
3. Neha Misra, Sheena Jose, N. Hema and Usha Bhandiwad 'Automated Generation, Validation and Execution of Spacecraft Test Procedures For Faster and Efficient Checkout Operations '.
4. Neha Misra, Sheena Jose, N. Hema and Usha Bhandiwad 'Integrated Test File Editor & Compiler for Spacecraft Checkout'.
5. ACSS version-2 system features and Test & Evaluation document for IRS-P4
6. Guidelines for writing Test Sequence files using Checkout Command Language