Defence Science Journal, Vol. 56, No. 5, November 2006, pp. 693-700
© 2006, DESIDOC

SHORT COMMUNICATION

Range Safety Real-time System for Satellite Launch Vehicle Missions–Testing Methodologies

R. Varaprasad and V. Seshagiri Rao

Satish Dhawan Space Centre SHAR, Sriharikota–524 124

ABSTRACT

A real-time system plays a critical role in the range safety decision-making in a satellite launch mission. Real-time software, the heart of such systems, is becoming an issue of criticality. Emphasis is being laid on the development of reliable, robust, and operational system. This paper purports to delineate prudent testing methodologies implemented to test the real-time system.

Keywords: Real-time system, range safety, inertial navigation system, polynomial filter, Kalman filter, testing, network simulations

1. INTRODUCTION

A real-time system (RTS) in a satellite launch mission helps the range safety officer to monitor the behaviour of the vehicle at each instant after the lift-off. Abnormal deviant behaviour of the vehicle entails its destruction during mid-flight. This flight monitoring and mission-abort action preempts loss of precious human lives and property. Radars around the launch site as well as down the range, and telemetry from onboard systems provide range safety officer with data describing position, velocity, instantaneous impact point and performance of onboard systems to enable to execute requisite action within the specified time. A well-designed, systematically developed, and thoroughly tested real-time system is of utmost necessity for the mission. Towards this end, prudent testing methodologies are implemented at the Satish Dhawan Space Centre [SDSC SHAR], which are simple, cost-effective, flexible, and repeatable in their scope.

2. DESCRIPTION OF REAL-TIME SYSTEM

The range safety real-time system is a time-triggered RTS that offers predictable execution of hard real-time tasks even under peak-load conditions. The following salient features have influenced the design of this RTS:

- Range safety philosophy
- Redundancy to provide fail-safe and reliable operation
- Fault tolerance based on data diversity
- Stringent maintenance of response time
- Distributed computers configuration
- Standard modular hardware and software.

Figure 1 depicts the configuration of range safety RTS. Principal components of the instrumentation of RTS are:

Revised 18 October 2005

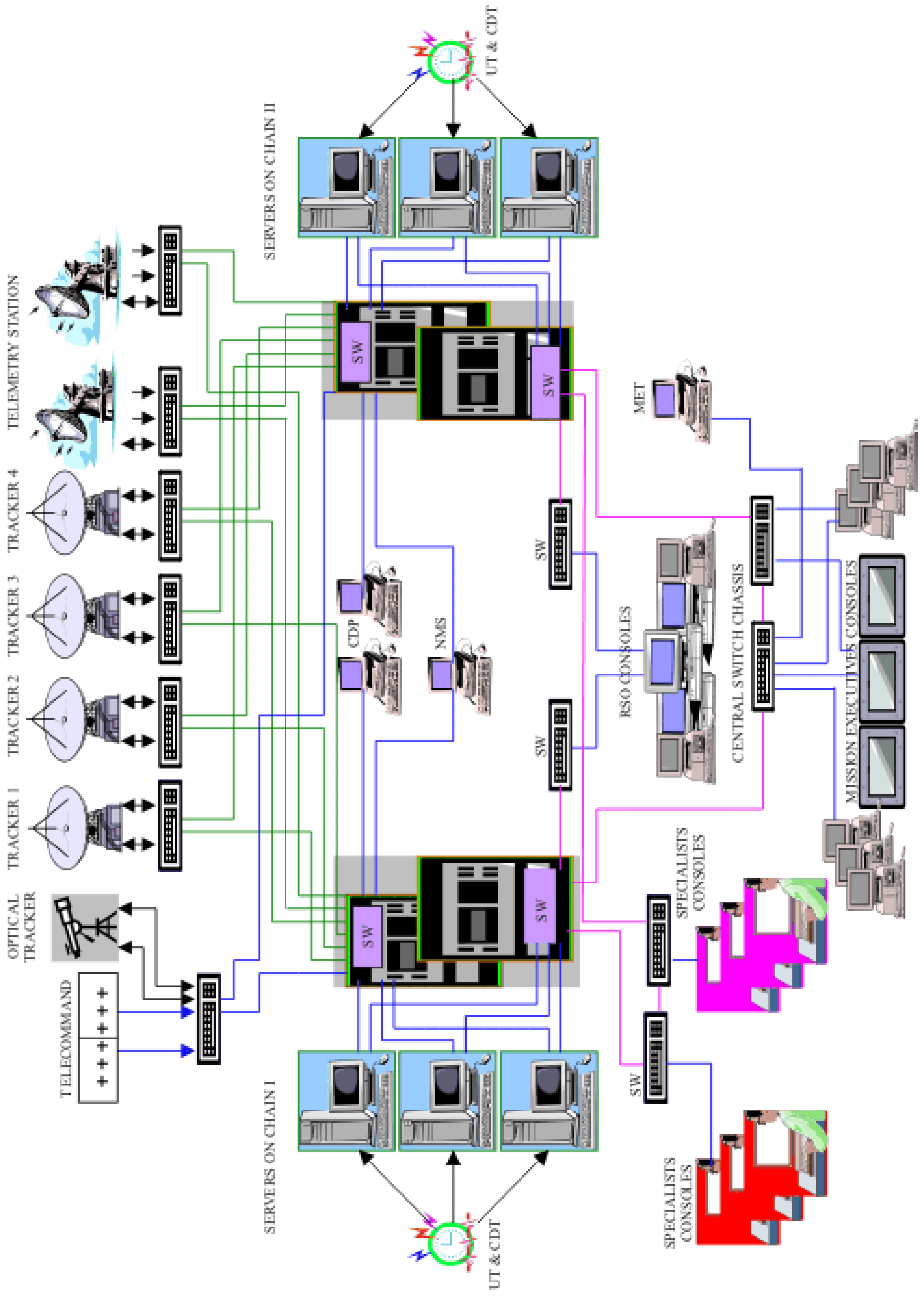


Figure 1. A typical real-time network configuration.

- C-band radars and one S-band radar
- Dual redundant S-band telemetry stations at launch base and down the range connected by spacecraft links
- Host servers on primary chain and secondary chain to process radar tracking data and onboard telemetry data
- Real-time displays including primary and secondary range safety information.

In the light of high reliability requirements for range safety, fault-tolerance capabilities are established by configuring the whole system around two independent chains on a fast fibre-optic Ethernet local area network. Each host server is connected independently to multiple trackers and telemetry ground stations on dedicated lines. On each chain, the host server processes C-band radar tracking data, S-band radar tracking data, and vehicle subsystems telemetry data, including inertial navigation system data. All the processed data such as position, velocity, acceleration, propulsion, and control and guidance-related parameters are transmitted to redundant displays in range safety officer's console at 10 Hz for decision-making.

2.1 Real-time Hardware

In a mission-critical system, custom-built hardware with proven design is preferred to commercially available modules. For performance reasons, high-end proprietary machines were selected for the host servers and high-resolution graphic workstations as range safety displays. All ground stations and mission-related displays were connected to each host server through Ethernet switches over fibre-optic cables. Single mode and multimode fibre-optic cables were used, the former for establishing connectivity to farther ground stations and the latter for those in proximity. Cables and their construction were so chosen to provide for extra mechanical protection and stable fibre characteristics. External ground stations down the range were connected to the servers via 128 / 256 KBPS redundant dedicated satellite links.

2.2 Salient Features of Real-time Software

A real-time software consists of a manufacturer-supplied software, an environment software and an application software. Manufacturer-supplied software includes operating system features like real-time clock, multi-programming facility, inter-process communications, and synchronisation features. For the range safety real-time softwere, DIGITAL UNIX with X-windows user interface was chosen as the operating system on host servers as well as on range safety display workstations. Inter-process communication features supported by DIGITAL UNIX are callable in Fortran programs. The environment software supports the communication between the system and different network elements. The software is developed in C language. The application software is Fortran90 compatible.

The software is highly modular. This software being complex and critical, each module is meticulously tested for full-condition coverage, branch coverage, besides boundary, off-boundary and boundary-crossing cases. Minutest attention was devoted to prevent situations like exceptions, single fault-failure modes, ie, stop, crash, hang, incorrect data response, etc. Care was taken to dispense with any unused executable code and unreferenced variables. Limit and reasonableness checks were performed on all input and output before the action based on these was taken. Different routines for data validation, co-ordinate conversion, filtering and computation of trajectory were tested separately and verified for their accuracy. Input and output data handling routines were tested individually by transmitting known patterns of data.

To ensure fault tolerance[1,2], N-version programming was implemented with identical software running on the host servers with diverse source priorities. This feature ensured that the software process on each host server was exposed to different data set at any instant. Also software with diversified filters with identical functionality was implemented. All the critical functions were designed as independent processes and each software process was designed with error-resistant and self-checking capabilities.

After preprocessing, the position and velocity components of the vehicle were obtained. A polynomial filter estimated the initial state, and in continuation the recursive linear Kalman filter [LKF] was used. Data of all the sources were processed. Position, velocity, and instantaneous impact point were computed for range safety displays. One of the sources was selected to provide antenna-pointing information to telecommand and other geographically-dispersed trackers.

The real-time software was instrumented with a host of control variables to assess the performance of the range safety real-time softwere. Though identical software existed on both the chains, source priorities were kept different. This enabled selection of data from diverse sources for processing.

3. TESTING OF REAL-TIME SOFTWARE

Testing of real-time software is very complex and time-consuming because of non-repeatability, equipment interaction, inter-task communication, and synchronisation and time criticality. Until now, there does not exist a thoroughly investigated testing methodology for real-time software. In 1980, Glass[3] considered real-time testing as a lost world and nothing has changed since then. However, the available literature reveals that environment simulator is a well-accepted and universally used tool for testing real-time software. Over the past three decades, advent of fast processors, sophisticated computer modelling with graphics, and advances in network technology have gone into building of simulation technology that is just a little short of the reality. Other pertinent advantages provided by the simulator are time flexibility, training, repeatability and the deep insight into the functionality of systems it mimics[4].

4. REAL-TIME SIMULATION TESTING

4.1 Environment of Simulation Tool

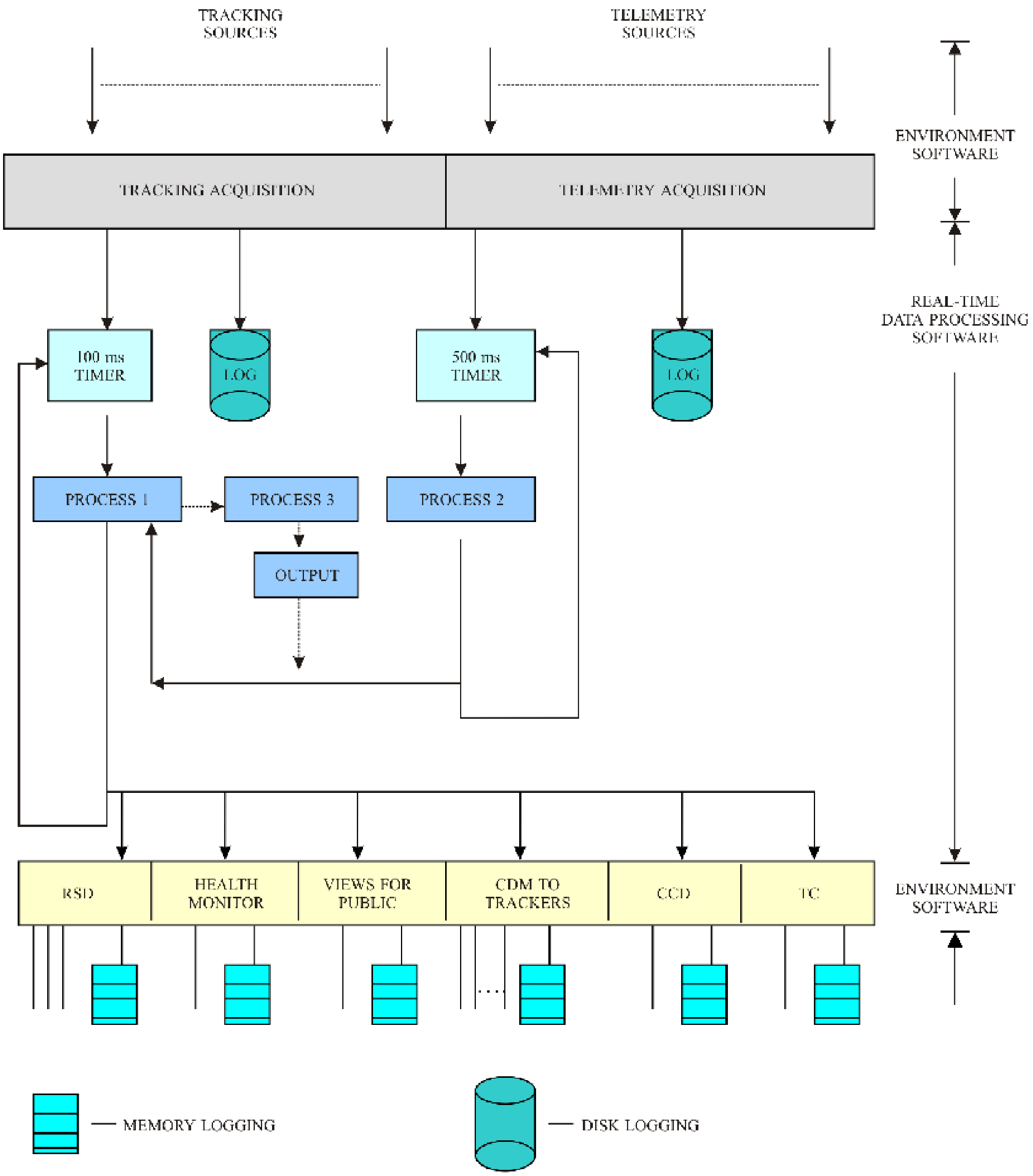
A real-time simulation tool simulator for range safety real-time system [SRRTS] was developed consisting of a build of the entire trajectography chain with the acquisition and transmission part of the network replaced by file interfaces. Data flow in SRRTS is described in Fig.2. Development of this real-time simulator is based on the conventional host/target system approach. The host system is a workstation with real-time environment features identical to those of the target servers. This commonality has facilitated completing the modular tests and integration tests on the host system itself.

Specification of the simulator is derived from the description of the environment to be simulated as well as from the application to be tested. Functionality and temporal requirements of the simulator are arrived at by inverting of the functionality and timing constraints of the application software.

Interactive iconised windows and their execution supported by selective input capability facilitated better testing environment. POSIX 1003.1b features of DIGITAL UNIX[5] provide the communication and synchronisation necessary between environment software and application software. File interfaces provide tracking data of range radars and external stations at 10 Hz and vehicle subsystem telemetry data at 2 Hz. Processed data are transmitted as Ethernet packets over real-time network to the range safety displays. Antenna-pointing information transmitted to trackers and ground stations is logged. This tool has added advantages. It provides control over the simulated environment and facilitates testing in an easier way. Good observability in terms of evaluating functional correctness and temporal behaviour is provided.

4.2 Test Cases and Criteria for their Adequacy

The key issue of testing is to find out the possible test cases that have the highest probability of detecting most of the errors [6]. Test cases are generated based on the functional specifications of each software process using blackbox methods. These test cases are supplemented by the ones from whitebox methods. Overall strategy for generation of adequate test cases is based on cause-effect graphing, boundary value analysis followed by equivalence partitioning. Flow graphs derived from the procedural representation of the software provided their cyclomatic complexity giving the upper bound of the number of independent paths to be tested.



RSD	:	RANGE SAFETY WORKSTATION DISPLAYS
CDM	:	COMPUTE DESIGNATE MODE DATA
CCD	:	CONTROL CENTRE DISPLAYS
TC	:	TELECOMMAND RECEIVING CDM

Figure 2. Data flow in the range safety real-time system.

This ultimately led to the number of test cases to be designed and executed to guarantee the coverage of all program statements. Intuition and experience in the real-time software also helped expose some of the possible errors and eventually supplemented the above set of test cases for their adequacy.

4.3 Testing on Host System

The principal requirements of hard RTS are response time, accuracy, reliability, and response to undesirable events [7].

The RTS was tested with a combination of nominal and non-nominal input. Selective input facility provided in the simulator enables exercising the RTS with a chosen input configuration. Incremental addition of ground stations to the input configuration led to different input combinations for a thorough testing of the software. For each mission, a number of input configurations consisting of nominal and non-nominal data, were exercised. Each input configuration led to exercising the software for a particular branch of execution. In each simulation, processor utilisation or the time taken to complete a processing cycle was noted. It was found that the average response time for a processing cycle is around sampling period with negligible variance. Also, each sample transmitted to the range safety displays was stamped with system time and block-id. It was found that all the samples were received at range safety displays without I/O queue.

Once the application software was activated, a standard time was generated which was incremented by 100 ms in each processing cycle. Synchronisation of sources was verified by comparing the program standard time with that of individual sources.

Ground trace, trajectory and instantaneous impact point were computed using radar tracking data of various trackers and were as expected and as seen from range safety displays.

Also the pointing information transmitted to various trackers for target acquisition/reacquisition was logged and checked for its accuracy, frequency and continuity.

Logic of selection of sources for trajectory display was verified by changing priorities and terminating some sources during processing. Displays were monitored to verify the sequence of source selection.

The entire trajectography chain was tested with minimum and maximum load configurations and it was found that system performance does not degrade with increasing load conditions.

5. TESTING ON TARGET SYSTEM

After completing integration test on the host system, a real-time software was installed on the target processor and was exercised in network simulations, balloon-tracking exercises, satellite tracking campaigns, etc.

5.1 Network Simulations

Before any satellite launch mission, the real-time network is exercised a number of times with the intention of validating the functionality of network elements and interfaces, verifying correctness of the software elements, and assessing the performance characteristics and reliability of the RTS, including the wide area network elements.

Tracking data of radars from a local simulator and telemetry data from geographically distributed ground stations were transmitted to target computer. Initial phase of network simulations consists of autonomous checks of each ground station. All combinations of input streams from that station were input to the RTS. The real-time software was exercised and range safety displays and trajectory displays were monitored. Antenna-pointing information at respective stations were received and stored to assess their correctness.

Later ground stations were added incrementally to the real-time network to verify I/O load on the network, source selection, and functional and temporal correctness of each element in the RTS. These are termed as integration checks.

Full-scale network simulations consist in exercising each hardware, software, and interface elements of the real-time network in the launch configuration. Input configurations consist of nominal and non-nominal trajectories. Time taken for each cycle of operation was stored in memory at the systems. In each simulation, its mean was computed and verified. Input and output data were logged at each system to verify continuity of time and validity of parameters. Snapshots were taken during critical periods of real-time to confirm that there was no I/O queue. Time synchronisation among sources was checked. Instrumented variables were verified for expected values. Repeatable environment provided during the simulations helped assess the reliability of the system. In addition simulation exercises familiarised the station personnel with nominal and deviant trajectories and trained them to properly operate the systems during launch.

5.2 Balloon-tracking Exercises

Satish Dhawan Space Centre has a meteorological facility that releases balloons thrice a week throughout the year. Real-time application software is exercised to support the balloon tracking. Tracking data are acquired at 10 Hz and the data are processed to provide trajectory as well as pointing information to the trackers.

5.3 Satellite Tracking for Radar Calibration

Before each launch, satellite-tracking campaign is conducted for calibrating the C-band radars. C-band transponder onboard Indian remote sensing satellite is tracked and the tracking data is used for calibration. The RTS is similarly exercised for these campaigns with expected performances.

In all these tracking exercises, functional and temporal characteristics of the range safety RTS are monitored and observations are submitted to the Standing Review Committee for Software Development [STARS] for analysis and review. This Committee oversees all the software activities covering the life cycle and configuration control of real-time software[8] as per ISRO-DOS Software Engineering Standards ISES-92 . After a careful scrutiny of the test results by the Flight Readiness Review Committee (FRRC), the RTS is cleared for mission support.

6. REAL-TIME SUPPORT TO SOUNDING ROCKET, PSLV AND GSLV MISSIONS

The RTS, which is thoroughly tested, has successfully supported a number of major flights, ie, five polar satellite launch vehicle (PSLV) missions, two geosynchronous satellite launch vehicle (GSLV) missions and many sounding rocket missions for atmospheric studies. All the varied real-time requirements of missions are met. There is no failure. This fact amply proves the effectiveness of testing methodologies.

7. CONCLUSIONS

Simple, practical, and effective testing methodologies have been explained. The SRRTS has proved to be a cost-effective and flexible tool to provide a repeatable real-time simulation environment to test the real-time support to sounding. Test procedures cited above are technically viable and practically applicable. The real-time support to sounding explained here has supported a number of satellite launchings without any failure.

ACKNOWLEDGEMENTS

The authors wish to thank Shri M. Annamalai, Director, SDSC SHAR, Sriharikota, for granting permission to publish this work. Thanks are also due to Shri M.K. Sanyal, Dy Director, Range Operations, SDSC SHAR for his constant encouragement. Success in this critical area would not have been possible but for the dedicated support and unstinted cooperation of colleagues in Real Time Data Processing (RTDP) and Mission Analysis and Range safety (MARS) Groups at SDSC SHAR.

REFERENCES

1. Seshagiri Rao, V., *et al.* Fault-tolerant data processing [FTDP] for reliable real-time support for satellite launch vehicle mission. National Workshop on Software Reliability Engineering [WSRS–92], 23-25 November, 1992, BARC, Mumbai.

2. Punnekkat, S.; Burns, A. & Davis, R. Analysis of check pointing for real-time systems. *Real Time Syst.*, 2001, **20**(1), 83-102.

3. Glass, R.L. Real time: The lost world of software debugging and testing. *Comm. ACM*, 1980, **23**(5), 264-71.

4. Schutz, W. The testability of distributed real-time systems. Kluwer, Academic Publishers, 1993.

5. Digital Unix: Guide to real-time programming. Digital Equipment Corporation, Maynard, Massachusetts, March 1996.

6. The art of software testing. John Wiley & Sons, 1979.

7. Stankovic, J.A. & Ramamritham, K. Advances in real-time systems. IEEE Computer Society Press, Washington DC, USA, 1993.

8. ISRO-DOS Software Engineering Standards [ISES-92]: ISRO-HQ-SP-61-92.

Contributors

Mr R.Varaprasad obtained his MSc (Physics) from the Andhra University, Visakhapatnam and joined the Indian Space Research Organisation (ISRO) in 1983. He worked in diverse fields such as radio position determination systems, LAN-based vehicle telemetry real-time monitoring systems and range safety real-time system. Presently, he is Dy Manager, SHAR Computer Facility at the Satish Dhawan Space Centre SHAR, Sriharikota. He has published a number of technical papers in national and international journals. His fields of interest include design of real-time systems for satellite launch vehicle missions, advanced data processing techniques in trajectory estimation and allied fields.

Mr V. Seshagiri Rao obtained his MSc (Electronics) from the Andhra University, Visakhapatnam. He joined ISRO in 1976. He worked extensively in the design and development of real-time systems for SDSC launch base. He has developed numerous digital filtering techniques for state estimation using radar data. He has also contributed towards development of simulators for software testing. Many algorithms developed by him are used in radar calibration, wind profile estimation and telemetry data processing. He is instrumental in enforcing software engineering standards for software development in the Centre. He has published around 18 technical papers in various national and international journals/ conferences. He is currently working as General Manager, SHAR Computer Facility at the Satish Dhawan Space Centre, SHAR, Sriharikota.