Design of a Reconfigurable and Self-contained Automated Checkout System

Jaya G. Nair, C. Radhakrishna Pillai, S. Hemachandran, Dr. P. P. Mohan Lal
ISRO Inertial Systems Unit, Indian Space Research Organisation,
Vattiyoorkavu, Thiruvananthapuram
g_jaya@vssc.gov.in

Abstract:
Computerized Checkout Systems are used for the calibration and performance evaluation of navigation systems. These checkout systems are designed to support fool-proof operations, are self-contained, require minimum intervention by the operators and it is possible to use them for different missions without change in operating procedures. The paper discusses the features in the design that lead to the development of such an automated checkout system.

Key words: Checkout Systems, Automation, Safety, Fault-tolerance.

I. INTRODUCTION

Navigation systems used in launch vehicles are complex and hence require computerized Checkout Systems for conducting calibration and performance evaluation tests. Design and development of such an automated checkout system for a Redundant Strap-down Inertial Navigation System (RESINS) which is used for all ISRO-LV missions is described in this paper. The navigation system comprises of sensors, electronics, interfaces and computers that do the navigation computations. The checkout systems provide all the necessary interfaces to connect the test articles and implement the different test procedures. Interfaces for powering, data acquisition and health monitoring are provided. This translates to implementing 2 Mil-1553 bus interfaces, 64 analog input channels, 16 analog output channels, 48 channels of digital input and outputs, 22 pulse simulator channels, 96 multiplexer channels, 20 powering relays, 2 serial interfaces and 7 DC power supplies.

The test and evaluation schedule for a navigation system comprises of a sequence of tests under different environmental conditions. The checkout systems are typically operated by test personnel. The checkout systems are required to be available to test navigation systems belonging to different missions concurrently. The checkout systems are typically used for many years which translate to having a long maintenance phase after the initial design and deployment. To efficiently implement the different requirements, the checkout systems must have features of self-containment, reconfiguration and automation. The elements of design and mechanisms of implementation which lead to the realization of these features are discussed here.

Motivation for the design features are given in section II. Design of mechanisms is covered in section III. Details of implementation are presented in section IV. The development approach and conclusions are given in sections V and VI respectively.

II. REQUIREMENT AND MOTIVATION FOR FEATURES

The navigation systems for different missions have broadly the same test requirements but there are differences in many specifics. These include changes in the format of data, software used in navigation processors, test conditions, methodology used for computation of test results etc. The checkout systems must support the different missions concurrently.

Different units of checkout systems are deployed to cater to tests at different test stations including environmental test-beds. Differences in the hardware configuration between the different units need to be supported seamlessly. Any failure and reallocation of hardware interfaces during the long maintenance phase must be absorbed. The feature of ***reconfiguration*** is needed in the checkout system to support all the above requirements.

The checkout systems are operated by people who are not necessarily navigation experts. Hence the system should be equipped to support fool-proof operations with minimum intervention required from the test personnel. The feature of ***automation*** is needed to realize such a requirement. Automation means that issues of security, safety and fault tolerance must also be addressed without need for conscious action from the operator. Another requirement is for the operating procedure to remain the same irrespective of the changing functionality with respect to missions and test-beds. This requirement gives rise to the need for both the features of reconfiguration and automation.

Efficiency of usage motivates the need to have all facilities to conduct tests and analyze the results using the checkout system itself. Ease of use of the checkout system is of prime concern since there is need to physically move the system between test-beds during the course of the performance evaluation of the test articles. The need for the checkout systems to be ***self-contained*** both in terms of functionality and physical configuration is the offshoot of these concerns.

Fig [1] shows the context under which the checkout system operates. The requirements that lead to features of reconfiguration, automation and self-containment are indicted by letters R, A and S respectively on top of the boxes.

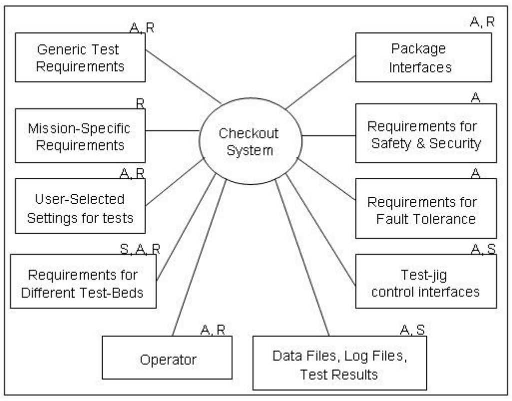
A, R
Generic Test Requirements
R
Mission-Specific Requirements
A, R
User-Selected Settings for tests
S, A, R
Requirements for Different Test-Beds
A, R
Operator
A, S
Data Files, Log Files, Test Results
A, S
Test-jig control interfaces
A
Requirements for Fault Tolerance
A
Requirements for Safety & Security
A, R
Package Interfaces
Checkout System

Fig [1]: Context diagram for Checkout System

III. DESIGN OF MECHANISMS TO REALISE THE FEATURES

The different mechanisms used to realize the features identified are detailed here.

1. *Reconfiguration*

To support reconfiguration, both hardware and software mechanisms are employed.

Modular and re-configurable hardware interfaces are used. Additional spare channels are provided in the hardware interfaces. Device drivers are designed to implement common APIs independent of the features of any particular hardware. These mechanisms make it possible to support hardware failures, at both channel and card level. Updates to hardware interfaces during the long maintenance phase are also taken care of in this manner.

All factors which may undergo change are captured as parameters to the checkout software. The values of these parameters are specified using a set of configuration files. These files capture all the different scenarios of usage with respect to missions, checkout units and test beds. The test-configuration for a particular invocation of the software gets specified by a combination of the identity of the user, command-line arguments and selections made by the user from a list of options shown. Once the configuration of test is specified, the software uses the set of configuration files applicable to define the values of all parameters. Hence the software is able to reconfigure itself. It is possible to completely change the look and feel of the checkout software in this manner.

The reconfiguration applies to fields shown in the Graphical User Interface (GUI) displayed, the menu of options shown, channels used in data acquisition, data processing methodology used, reference values used to check test results etc. This feature also enables support for different modes of operation: like test and evaluation mode, package designer’s mode and checkout system diagnostic mode.

2. *Self-containment*

Self-containment is achieved by making available all the necessary tools for completing the tests in the checkout system itself. Periodic display of the status of the test articles and tests in progress is provided. Tools are used for data processing and analysis with automated checking of results. These mechanisms enable decision-making during tests without the need of any additional equipment.

Organization of the checkout system in a single rack with all associated power supplies, interfaces and computers provide ease of use and portability. Implementation of mechanisms for fault-tolerance and the availability of tools for diagnostics of the checkout interfaces add to self-containment.

3. *Automation*

Different mechanisms are designed to implement the feature of automation. Automating the intermediate steps in test procedures is the basic mechanism. The parameters for the steps are captured in the software in such a way that minimum intervention is needed from the test operator. Calibration tests involving a number of steps are automated using configuration files that capture the parameters of intermediate steps.

Some tests need the test-jig to be commanded for necessary orientation and rotation. Commands to the controller of test-jig are generated automatically from the checkout system by building an interface and command protocol between the two systems.

Automatic verification of initial conditions necessary for tests, verification of correctness of the operation of intermediate steps, real-time display and checking of critical parameters during tests, data processing and verification of results at the end of tests etc are some of the mechanisms used to enable automation. Automation creates an identical condition for the system all along the calibration process, especially taking into account the thermal characteristics while taking the measurements.

4. *Design for Safety*

Mechanisms for ensuring safety of the test articles are paramount while designing automation.

Interlocks that control the method and sequence of applying power to the test articles are implemented in the hardware interface. In addition, initial conditions and power-supply voltages are checked by software before powering packages. Continuous monitoring of the power-supply voltages and the currents drawn by the packages is done. In case of any anomaly, automatic actions are undertaken to ensure safety. Critical parameters related to the health of packages under test are also acquired and subjected to continuous surveillance. Operator is alerted about any error conditions by audio signals and display. A watch-dog circuit is designed which alerts with loud alarm if the checkout software is not working as intended. Any user-input provided during test is verified before use.

Some of the automatic safe-actions themselves are disabled during critical tests when a number of initial conditions are necessary to be satisfied. Provision for by-pass control of power allows temporary work on the checkout system without affecting the test articles. Clear indication of any error by display and alarm is the method used in such situations. There is also provision for an emergency switch to cut off power to test articles.

Precautions for ensuring safety of test-log and data are also of importance. Automatic re-routing of printer data to a file in case of any error in the printer is an example. The operator is alerted, and once the printer becomes ready, the stored data is printed and then normal printing mode is restored. No user action is required to activate this re-routing.

5. *Design for Security*

Security is ensured by defining a set of users for the checkout system and clearly demarcating the pathname space accessible to each user. The users are categorized with respect to missions. User authentication by username and password, allotment of privileges to users only as required and use of binary data format for configuration files are some of the other mechanisms designed for security.

6. *Fault-tolerance*

Fail-safe is the condition of fault tolerance required in checkout systems. Automatic safe-actions are undertaken with continuous health surveillance of the test articles throughout the time of tests. In case of anomalies during tests, tests are aborted and the setup brought back to safe conditions.

IV. SELECTION OF IMPLEMENTATION MECHANISMS

Some of the selections made during the implementation of the design also contribute to the realization of the features of reconfiguration and automation.

1. Choice of Operating System

The requirements to the checkout system fall into functional and non-functional requirements and need many actions with varying degrees of criticality and periodicity. Complex tests needing simulation of inputs to the test articles require simulation and acquisition of data from multiple interfaces with millisecond periodicities and hard real-time deadlines. It is decided to implement the requirements by designing the checkout system as a real-time system using a hierarchy of co-operating processes with different priorities and scheduling policies. To facilitate such a design, it is decided to use a real-time operating system (RTOS) and QNX is selected. QNX has fast context-switch and interrupt latency times and many mechanisms for inter-process communication and synchronization. The choice of operating system also enables the design for reconfiguration and security using the features of soft-links, user authentication and pathname space management.

2. Software Architecture

The software is designed to have client-server architecture. This extends the client-server nature of the underlying operating system (QNX). This architecture enhances the capacity for fault-tolerance and re-configuration as the clients and servers can be updated seamlessly as long as the interfaces remain constant. Servers implementing generic functionality are deployed with different input conditions to simulate similar behavior on different devices. The network-aware nature of QNX makes it possible to deploy servers in any computer in the network if needed, without changing the software or the operating procedure.

The software is designed to have a graphical user-interface using the standard Open Look interface. This aids automation.

V. DEVELOPMENT APPROACH

The initial development path follows a bottom-up approach. The hardware interfaces and servers implementing device driver functions are realized. The client modules are then developed, adding features incrementally. The software is realized with version numbers and new versions are released to incorporate updates and changes during the long period of use of the checkout system. The checkout system is subjected to different levels of a formal verification and validation process to ensure correctness and completeness. After deployment, updates on the configuration setup are required to support changes and new requirements with respect to missions.

VI. CONCLUSION

Need for concurrent support of missions with requirements which are sometimes at odds with one-another, long periods of usage and fool-proof operations is the hallmark of checkout systems. Implementation of features of automation, self-containment and reconfiguration makes it possible to realize these requirements effectively and efficiently. Implementing features in the design that aid in realizing the core functionality required goes a long way in providing an efficient maintenance phase. RESINS checkout system has successfully been in operations for more than a decade supporting various missions.

ACKNOWLEDMENT

Mr. P. S. Veeraraghavan, Director, IISU, is gratefully acknowledged for encouragement and support of the work detailed here. The authors would also like to acknowledge Mr. S. Gopakumar, Head, Computer Division, Vikram Sarabhai Space Centre, Thiruvananthapuram, for his original contributions in the design of RESINS Checkout System.

BIO DATA OF AUTHORS

Jaya G. Nair received B.Tech degree from University of Kerala, in 1988 and M.E. degree from Indian Institute of Science, Bangalore, in 2001, both in Computer Science and Engineering. Currently working in ISRO Inertial Systems Unit. Her areas of interest include Real-time systems, Operating Systems and Software Engineering.

C. Radhakrishna Pillai received B.Tech degree in Electronics and Communication Engineering from University of Kerala in 1988. Currently pursuing M.Tech degree in Microwave Communication Engineering from University of Kerala. Presently working in ISRO Inertial Systems Unit. His areas of interest include

Embedded Systems, Data Acquisition, Automation and Digital communication systems.

S.Hemachandran received B.Sc (Engg) degree in Electronics and Communication Engineering from University of Kerala, in 1983 and M.Tech degree in Electrical Engineering from IIT, Kanpur in 1989. Currently working as Dy. Division Head, NSSD-COS, of ISRO Inertial Systems Unit. His research areas are Embedded systems, Real-Time systems, GPS and Signal Processing.

Dr. P.P. Mohanlal received B.E degree in Electronics and Communication Engineering from Madras University, in 1976, M.Tech degree in Applied Electronics and Instrumentation from University of Kerala in 1995 and PhD degree in Computer Science from University of Kerala, in 2005. He is currently heading Navigation Software and Simulation Division in ISRO Inertial Systems Unit. He is a member of the post-graduate board of studies of University of Kerala. His research interests include intelligent control methods for nonlinear systems (Neural, fuzzy and neuro-fuzzy methods), Optimal filtering & control, DSP & Digital filter design using feedback neural network and Integrated Navigation Systems. Senior member of IEEE for 10 years. He has more than 12 research Publications in International Journals and IEEE Conferences. He is associate editor of a book on Neuro–Fuzzy Control, published by Narosa Publishing House, New Delhi, 1998.