A MODULAR, EXTENDIBLE AND REUSABLE TEST CONFIGURATION FOR SYSTEM-LEVEL MANUFACTURING TESTS

Muharrem TÜMÇAKIR, Çınar YEŞİL and Mert Burkay ÇÖTELİ
Defense Systems Technologies Department
ASELSAN Inc.
Ankara, TURKEY
mtumcakir@aselsan.com.tr, cyesil@aselsan.com.tr, mbcoteli@aselsan.com.tr

***Abstract* –This paper explores an effective solution for a test configuration to perform system-level manufacturing tests without use of system management/user interface software of system under test (SUT).**

A generic, computer aided and modular test configuration, which acts in combination and in an integrated way with the management unit(s) of the SUT over Ethernet connectivity is proposed as a solution. Using enough number of software modules to be installed on the management unit(s), the proposed test configuration provides a number of improvements compared to the existing test configurations. Parallel testing capability is one of these major improvements meaning that test scenarios may be applied to a certain number of SUTs at the same time independently. This feature accelerates the system-level manufacturing test activities both during the planned test operations and during fault localization cases.

The proposed configuration is implemented and used on various systems based on various platforms such as; sheltered C4I platforms, tracked observer or gun platforms, wheeled ground surveying platforms etc.

Keywords – System Manufacturing Tests, Test Configuration, Automatic Test System, XML, Ethernet

I. INTRODUCTION

After design, implementation, verification and validation phases are completed; system-level manufacturing tests become one of the most critical activities for any developed system. In most cases, these tests are applied to every single system product to be delivered, in order to locate and fix system-level defects related to the manufacturing processes such as cabling and assembly of the System Units (SU). Generally speaking, for military electronics systems, main focus of the system-level manufacturing tests is to verify electrical and sometimes mechanical interfaces between SUs and to verify the main functionality of System Under Test (SUT) as well.

Various types of test configuration or test methods might be used for system-level manufacturing tests in the Industry.

Applying specific test procedures to the SUT over its user interfaces (e.g. system application/user interface software, buttons, knobs etc.) might be one of the first methods considered to be applied for manufacturing tests as shown in Fig.1.

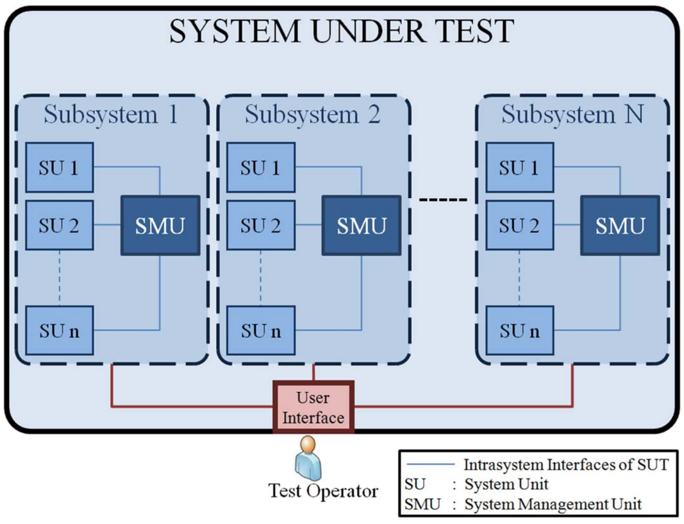


Fig. 1. Extant Test Configuration-1

By this test configuration, the main user interface that test operator may use is the user interface software of the SUT itself. Test operator follows the instructions written in the manufacturing test definition document prepared according to the SUT's user interface capabilities. For some specific verification steps, performing some temporary interface alterations (e.g. disconnect-measure-connect actions, connecting an external test device to an SU etc.) on the SUT becomes unavoidable to be able to verify some specific features of the SUT.

Another manufacturing test configuration alternative that might be taken into consideration is shown in Fig.2.

By this configuration, various test connectors with sufficient number of lines are designated on specific hardware units of the SUT for test purposes. During system-level test activities including manufacturing tests and field tests, these test connectors are effectively used by the test system. With test connector usage, the temporary interface alterations on the SUT mentioned above become less necessary. Therefore, the manufacturing quality of the system is preserved mostly by eliminating the necessity of such temporary alterations for test purposes.

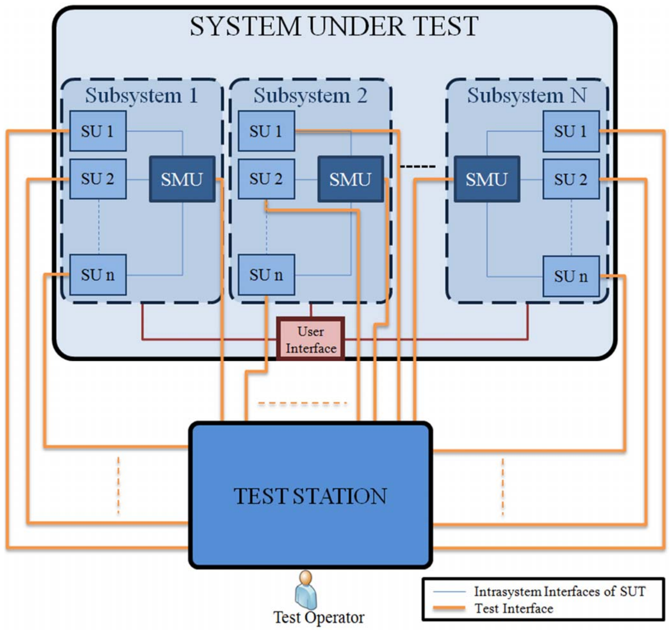


Fig. 2. Extant Test Configuration-2

Using an automatic-based test management software running on an independent computer (i.e. test PC @ TEST STATION) instead of the user interface unit(s) of the SUT as shown in Fig.2, increases the portion of automatic test steps among the complete set of test scenarios. Since, the system management units (SMU) are usually "Mission Computer" or "Command Control Computer" type units working on a specific operating system, one might think of installing the test management software onto the SMU(s) in order not to use a separate test PC for cost efficiency. However, this may cause compatibility problems between the SMU(s) and the test management software desired to be installed on them when various types of SMUs (with a variety of operating systems) are taken into account. In other words, having an independent test computer, the configuration given in Fig.2 can be used more flexibly on various systems. Additionally, this type of flexibility also provides decrease in setup durations of the test equipment before testing which is very important issue for serial manufacturing tests and field test activities.

The test configuration shown in Fig.2 also enables test operator to perform test scenarios without user interface software of the SUT. In other words, there is no need to wait for software development process to be completed before starting test development activities which makes these two processes independent from each other enabling parallelization as well. Eliminating the necessity of user interface of the SUT brings software privacy concerns to an end especially when the test activities are performed outside the company (e.g. in subcontractor facilities).

A. *Limitations of Extant Test Configurations*

Besides the good features of the test configurations mentioned above, there are still some limitations and disadvantages that are worth being discussed here:

The primary limitation of the test configuration given in Fig.1 is that it builds up software privacy concerns especially when the test activities would be carried out at a third party partner or subcontractor facilities while the real operation scenarios and special mechanisms of the SUT are required to be kept private. This limitation is the main starting point of the practical part of the work explained in this paper.

Another limitation of this configuration is that the observability and controllability of the SUT for a desired test scenario might be difficult by the use of user interface software of the SUT. Being designed for different functionalities and features, the user interface software of the SUT might not meet the requirement of manufacturing test scenarios for some cases. For example, the user interface software might not have a designated observation screens that display the values of some measured quantities desired for test purposes only. Similarly, the user interface software of the SUT might not have desired control interfaces for the SUT that are necessary for the test scenarios. That was another limitation we faced before developing our proposed configuration as well.

Another limitation of this configuration which is already mentioned was being forced to wait for the development process of the SUT software before the start of test development activities.

Being forced to apply temporary interface alterations on the SUT for test purposes is another disadvantage of this configuration which is mentioned above as well. Applying such kind of alterations to a completely produced system might both increase the testing duration and degrade the manufacturing quality because restoring the changes applied during tests might require more professional operations by a manufacturing specialist instead of a test operator. Therefore, this situation might inevitably cause either additional loss of time or quality degradation.

Based on extra test interfaces designed for test and an independent test station, the test configuration shown in Fig.2 has some superior features compared to the configuration shown in Fig.1 as mentioned before. However, the test configuration shown in Fig.2 has a few limitations as well:

Since, the complexity of this test configuration is higher with extra test interfaces, test setup durations might increase undesirably. Those extra test interfaces require extra test cables to be included into the test equipment resulting in less portable and less usable test station.

Another disadvantage of the test configuration given in Fig.2 is related to the fault isolation and self-test features. Since there are relatively more interfaces on the test station, when self-test is required on the test equipment it takes more time. Furthermore, the self-test development activities might require more effort and developed self-test equipment might become more complex and heavier than the desired levels. In case of fault occurred during the test scenarios, the fault isolation process might become more difficult with this test configuration. In other words, it might be hard to determine whether the fault originates from the SUT or from the test equipment that has a complicated interface with the SUT.

B. *Our Test Configuration Approach*

The proposed test configuration is based on the one shown in Fig.2 with some improvements to overcome the limitations and disadvantages explained so far. The concept diagram of the proposed test configuration is given in Fig.3.

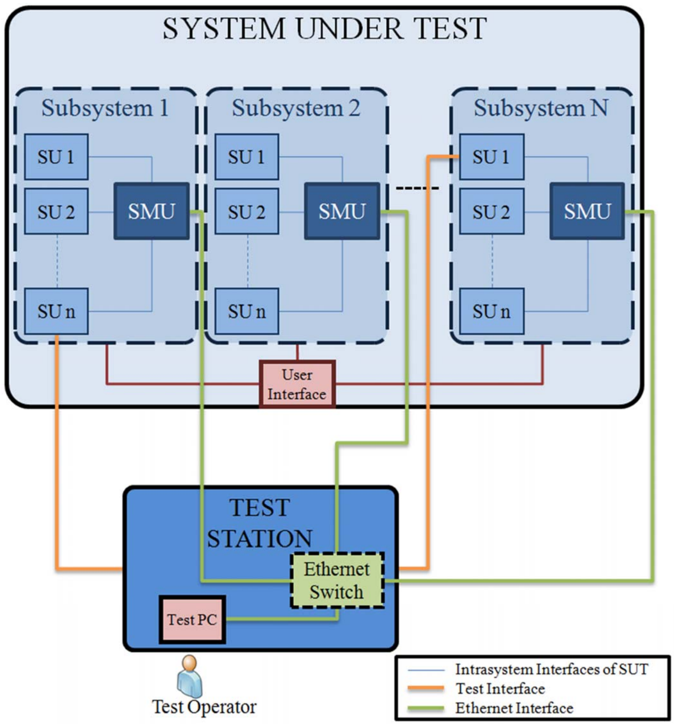


Fig. 3. Proposed Test Configuration

One of the major improvements is that the test interfaces between the SUT and the test station is minimized at the system design phase in order to reduce the complexity of the configuration and improve the portability, setup duration and self-test features. In other words, design-for-test methods are applied to the SUT in order to determine but minimize the required external test interfaces. Instead of external interfaces with test station, test-aimed interfaces are established between SUs and SMU(s) internally via data ports (i.e. RS-232, RS-422, CAN etc.) as much as possible. Furthermore, built-in test (BIT) features of the SUs are either improved or newly defined at design phase of the SUT. Since the SUT types we focused on (i.e. sheltered C4I systems, tracked observer and target identification system, wheeled ground surveying system etc.) do not have intense signal interfaces (either analog or digital), the remaining signal interfaces are directed to the test station over relatively much less external interfaces and test cables.

From design-for-test point of view, "system/subsystem-level built-in self test" feature is one of the most important features aimed for any developed system; on the other hand, the cost of this feature might sometimes become much higher than the expected levels for a given system. In those cases, the BIT features might be defined at SU-level instead of system/subsystem-level. The proposed test configuration is suitable for using the SU-level BIT features and including their results into the system-level test scenarios whenever needed, in a quite effective way. As a result, for system-level test activities including manufacturing tests and field tests, integrating the SU-level BIT features to the system-level test scenarios improves the test conditions in a quite cost-effective way, still under the shades of the ideal solution (i.e. "system-level built-in self test").

Since, the internal interfaces are designed so that the most of the useful test data coming from the SUs would be provided to the SMU(s) of the SUT, an Ethernet interface is included to the test configuration as seen in Fig.3 to benefit from this feature. The main idea behind this configuration is that if the management unit(s) could provide the intrasystem interfaces as the core of the SUT and data exchange between the SUs is under the scheduling of this management unit(s), then test scenarios are carried out by connecting the test computer to this management unit(s) over Ethernet and mainly observing and controlling the SUT over this management unit(s) with the desired detail levels. With this added Ethernet interface, the test configuration has gained some nice features that are worth mentioning here:

Having a proved structure and quite flexible applications, Ethernet is widely used data exchange technology in a variety of industries. Integrating TCP/IP, socket communication and eXtensible Markup Language (XML) on Ethernet basis for the proposed test configuration, the data exchange protocol requirement is met in a quite flexible, less time consuming and simple way. One socket connection is established between the test PC and each SMU. A communication software module is installed on each SMU (see Fig.4) before tests in order to manage the data exchange operations at SMU side.

Whenever access to an SU or a subsystem is required during the test scenario, related test commands and their related parameters are sent from the test management software installed on the test PC which is also the main user interface with test operator, to the related SMU's communication software module. Test operator may follow the test sequence on the user interface of the test management software easily during the test activities whenever needed.

A "hardware management software module" and enough number of "hardware adapter software modules" are installed on each SMU (see Fig.4) before tests, as well. The hardware management software module parses the commands and the parameters coming from the communication software module and engages the related hardware adapter software module to perform the required operation on the related SU.

After the required operation is completed on the related SU, related hardware adapter software module returns the required test data that is predetermined at design phase such as the result of operation, the values of specified parameters obtained during the operation, error codes, BIT results etc. to the hardware management software module to be evaluated and transferred to the test management software via the socket connection. Finally, test management software automatically makes decisions by using these results for pass/fail conditions of test steps and appends these results to the test report that would be finalized after the last step of the related test section or test group.

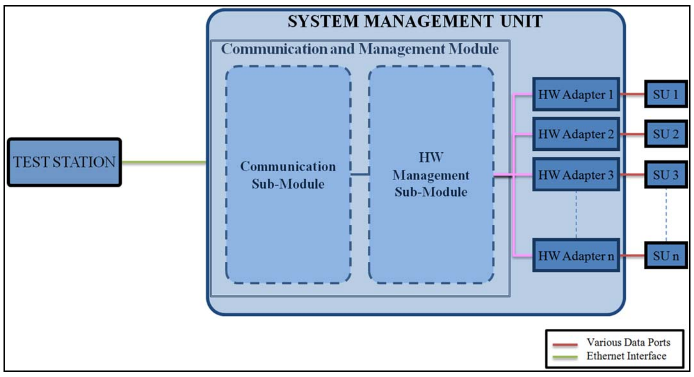


Fig. 4. Software Modules To Be Installed Onto SMU(s)

The hardware adapter software modules installed on the SMU may be reused as long as the related SUs are reused inside other systems while the SMU compatibility requirements are satisfied. From this point of view, the proposed test configuration can easily be adapted to newly designed systems by the help of ready-to-use adapter software module library. Development of new modules would be required only for new SUs which reduces the software development costs and durations significantly.

II. TEST CONFIGURATION DESIGN AND IMPLEMENTATION

A. Architecture

Details of the test configuration architecture are designated by taking various system architectures into account. The block diagram of the test configuration is given in Fig.5 while it is used for manufacturing tests of one of the aimed systems; a target identification and observer system based on a tracked land platform.

The SUT may have either more than one SMUs as shown in Fig. 3 or only one SMU as shown in Fig.5 according to its complexity and design issues. Since the socket connections are established over an Ethernet network regardless of the number of machines in the network, the proposed test configuration can easily be adapted to both conditions.

One external test interface is designated on the power distribution unit (PDU) which is the core unit of the power distribution subsystem. Measurements related to the power distribution subsystem are performed over this test interface at the test station whenever needed. In addition, some test scenarios of power distribution subsystem are carried out over SMU by using the data interface between the PDU and SMU as well.

Another external interface between the SUT and the test station rather than Ethernet is the wired telecommunication interfaces to verify the wired part of the telecommunication subsystem.

As the last external interface, all the data ports of the SMU are mirrored for monitoring. This interface is not used for manufacturing tests but reserved to be used in the field for fault localization tests which are in the scope of logistics support services.

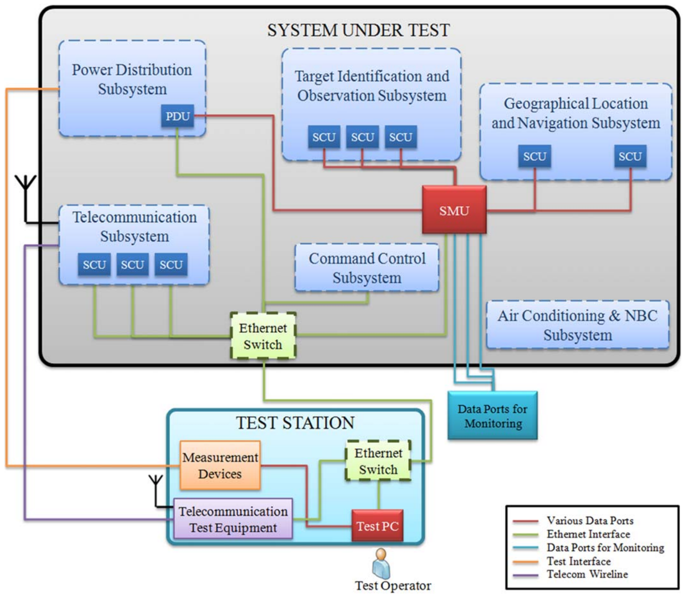

SYSTEM UNDER TEST
Power Distribution Subsystem
PDU
Target Identification and Observation Subsystem
SCU
SCU
SCU
Geographical Location and Navigation Subsystem
SCU
SCU
SMU
Telecommunication Subsystem
SCU
SCU
SCU
Command Control Subsystem
Ethernet Switch
Air Conditioning & NBC Subsystem
Data Ports for Monitoring
TEST STATION
Measurement Devices
Ethernet Switch
Telecommunication Test Equipment
Test PC
Test Operator
Various Data Ports
Ethernet Interface
Data Ports for Monitoring
Test Interface
Telecom Wireline

Fig. 5. Proposed Test Configuration Used With A Real SUT

Internal data ports to SMU and BIT features of all the subsystem core units (SCU) are used as they specified at the system design phase.

A few test groups are assigned for the test scenarios. Each test group is determined so as to correspond to one subsystem. Each test group may be run independently from the first step to the last one, over test management software, in a sequential order. The functional diagram of test operation environment at software level based on the proposed test configuration is shown in Fig.6.

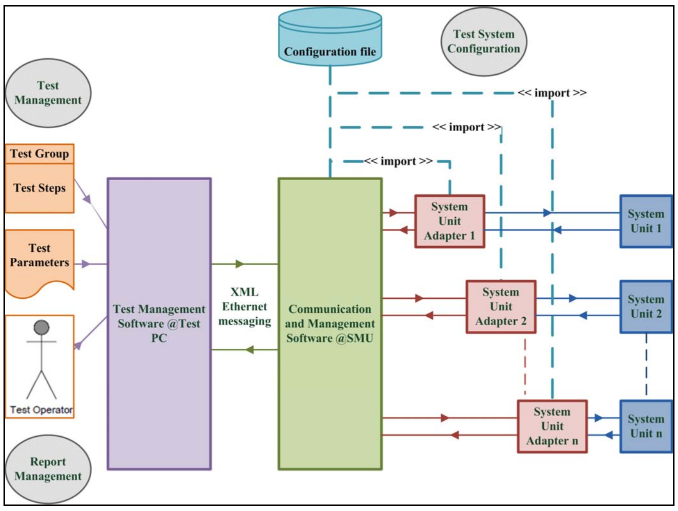


Fig. 6. Test Operation Environment

B. Test Methodology

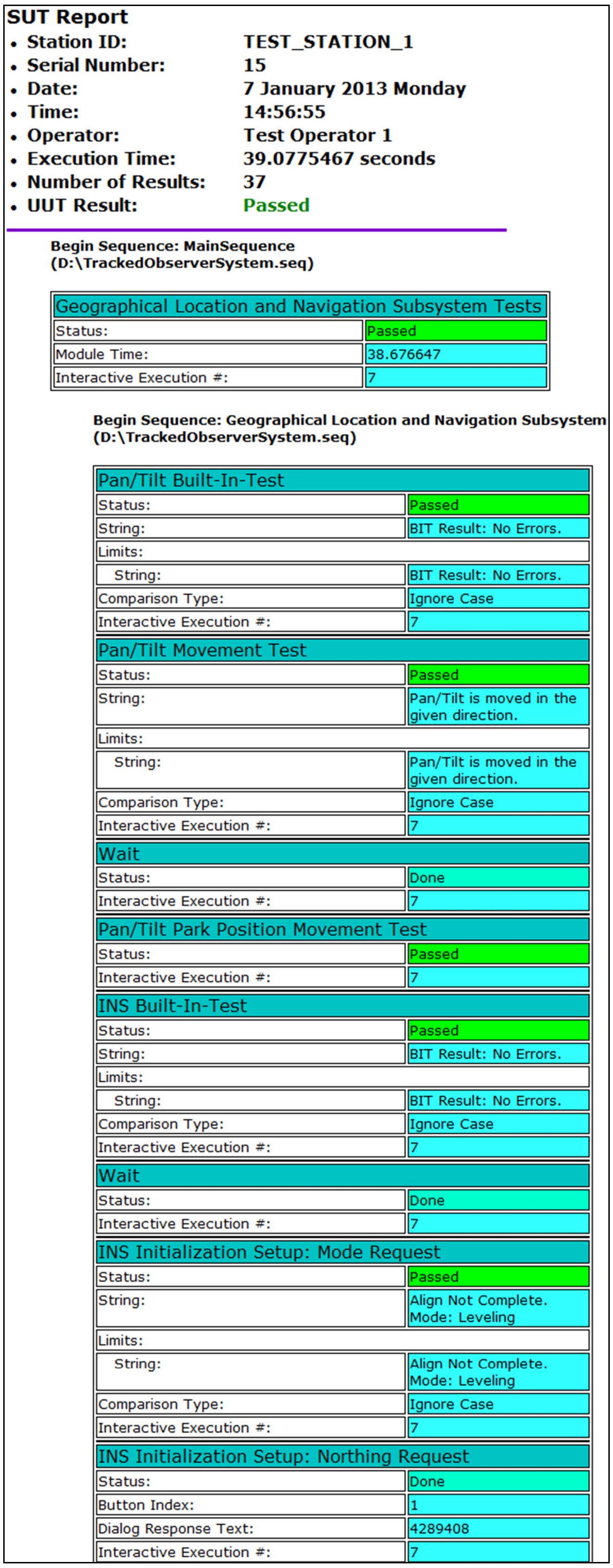
The specified test groups are executed by a test operator using a graphical user interface (GUI) of test management software. One test report is created for one test group by the test management software and at the end of each test group execution; the test management software finalizes the report of the related test group automatically in XML or HTML format which is defined as “Report Management” in Fig.6. Since the executed test steps might require various types of results (e.g. various numbers, arrays, matrices, texts, responses from system units, error codes etc.) to be logged according to the test scenario, the logging/reporting behavior of the test management software is specified at test step level during the design phase of the test steps sequencing. A part of an example test report created by the test management software is given in Fig.7.

“Test System Configuration” is another activity which deals with the SU adapters imported to the test environment according to the type of SUT. A configuration file describes the test environment components by dealing with the adapters. When a newly implemented adapter is imported to the test system, it is added as a system component to the test environment by using the configuration file. Available XML test classes connected to the SU adapter are also inserted to the test environment. There is a test class pool for the whole configuration. Each newly implemented SU adapter results test class insertion to the test class pool of the whole configuration. These newly inserted test classes would also be selected and executed using the test management software via XML messaging interface.

Execution characters of the test groups are either manual, semi-automatic or full-automatic according to the related test scenarios. For example, the test group related to the air conditioning and NBC (nuclear, biological and chemical) subsystem is manual due to the fact that it unavoidably requires user interactions and manual measurements during the test scenario. On the other hand, almost all the other test groups are semi-automatic or full-automatic. The execution characters are illustrated in the use case diagram of the test configuration in Fig. 8.

Test step execution would be considered as a finite state machine (FSM) [3] especially when access to an SU is performed. When a test step execution is started, the test management software goes into waiting state till the desired response arrives from the related SU. Responses coming from other SUs are not evaluated during the waiting state. Following test steps are not executed either during the waiting state. The synchronization between the test management software and the related SUs is provided by this state change management. The state changes that may occur during the testing of different SUs are shown in Fig.9.

When a function is called during a test scenario, the software goes from path (1) to path (4) in the sequential order. After path (2) is called as a query to an SU, the software waits for the SU’s output response which is the path (3). Then, the function output is sent to the test management software.



SUT Report

• Station ID:	TEST_STATION_1
• Serial Number:	15
• Date:	7 January 2013 Monday
• Time:	14:56:55
• Operator:	Test Operator 1
• Execution Time:	39.0775467 seconds
• Number of Results:	37
• UUT Result:	Passed

Begin Sequence: MainSequence
(D:\TrackedObserverSystem.seq)

Geographical Location and Navigation Subsystem Tests	
Status:	Passed
Module Time:	38.676647
Interactive Execution #:	7

Begin Sequence: Geographical Location and Navigation Subsystem
(D:\TrackedObserverSystem.seq)

Pan/Tilt Built-In-Test	
Status:	Passed
String:	BIT Result: No Errors.
Limits:	
String:	BIT Result: No Errors.
Comparison Type:	Ignore Case
Interactive Execution #:	7

Pan/Tilt Movement Test	
Status:	Passed
String:	Pan/Tilt is moved in the given direction.
Limits:	
String:	Pan/Tilt is moved in the given direction.
Comparison Type:	Ignore Case
Interactive Execution #:	7

Wait	
Status:	Done
Interactive Execution #:	7

Pan/Tilt Park Position Movement Test	
Status:	Passed
Interactive Execution #:	7

INS Built-In-Test	
Status:	Passed
String:	BIT Result: No Errors.
Limits:	
String:	BIT Result: No Errors.
Comparison Type:	Ignore Case
Interactive Execution #:	7

Wait	
Status:	Done
Interactive Execution #:	7

INS Initialization Setup: Mode Request	
Status:	Passed
String:	Align Not Complete. Mode: Leveling
Limits:	
String:	Align Not Complete. Mode: Leveling
Comparison Type:	Ignore Case
Interactive Execution #:	7

INS Initialization Setup: Northing Request	
Status:	Done
Button Index:	1
Dialog Response Text:	4289408
Interactive Execution #:	7

Fig. 7. A Part of An Example Test Report

The testing duration is an important issue for the test methodology. To give an example, the average testing duration to complete all the manufacturing test scenarios of the SUT exemplified in this chapter, is about 4 hours (including setup and test preparation durations) per system unless any manufacturing defects occur. Otherwise, the identification and repair of each defect unavoidably increase the given testing duration. Assuming that total identification and repair duration of defects for one SUT is not more than the defect-free manufacturing tests duration (i.e. 4 hours), the given testing duration is acceptable for the specified system since the manufacturing rate of the system is approximately "50 units in 8 months".

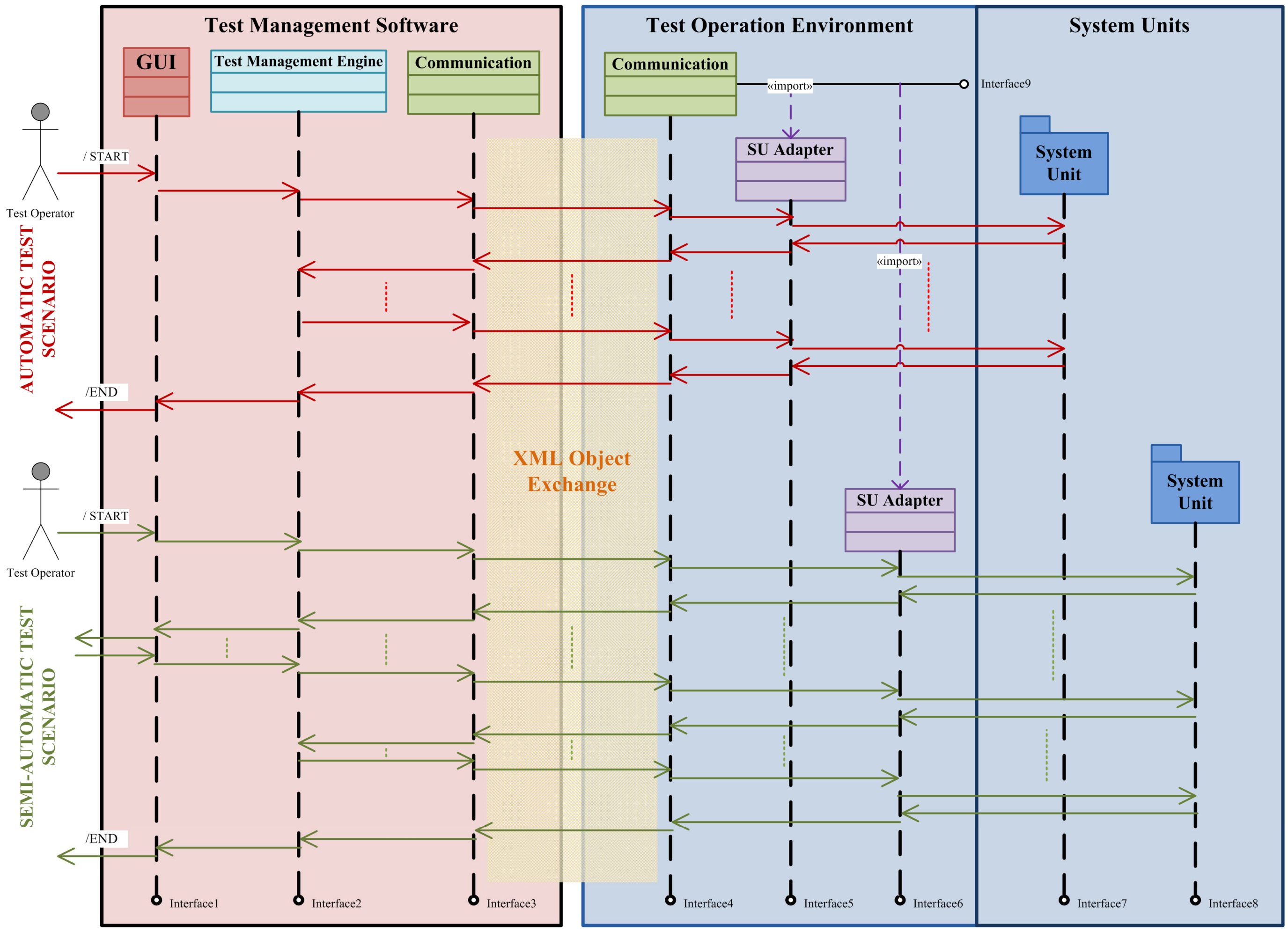


Fig. 8. Use Case Diagram of the Test Configuration

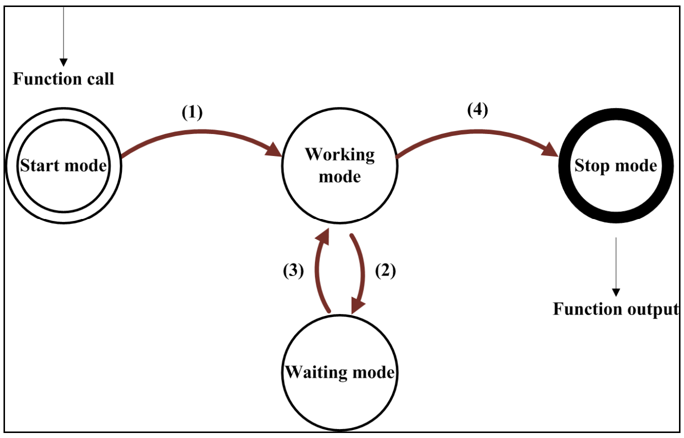


Fig. 9. Stateflow of the Software During Hardware Interactions

C. Hardware

All the necessary hardware of the designed test equipment based on the proposed test configuration is integrated on a portable case as shown in Fig.10.

The test equipment is connected to the SUT over its connection panel. On the other hand, test PC is mounted so that it can be detached from the case whenever the test scenarios require it to be used away from the case (e.g. during the navigation or air conditioning subsystem test scenarios).

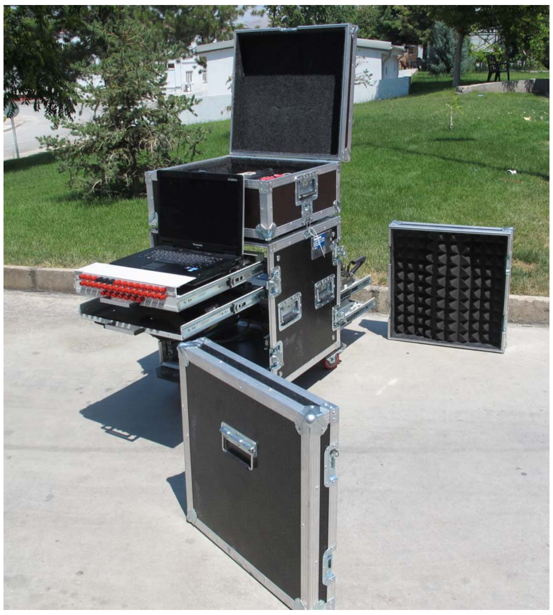


Fig. 10. Hardware of The Test System on Portable Case

D. *Software*

SU adapters are the connection classes between the SUs and the hardware management software. Adapter is a class that connects two different interfaces. It is a design pattern defined in the literature [2]. As explained before, when an SU is inserted into the test system configuration, related SU adapter is also imported to the test environment. As a result, insertion of SU adapters and configuring the software by means of the configuration file provides modularity for the test configuration.

As illustrated in Fig.6, the test environment is divided into three main parts at software level; test management software, communication and management software and SU adapters. Communication and management software runs on SMU side as a background service. Communication part of this service is a socket connection service between the management part of this service and test management software. Connector classes are implemented for each side; the test management software and communication part of the service. The communication between the connector classes is the object oriented messaging protocol which is named as XML. XML message format contains test input data, responses from the SUs and function calls on the SU adapters. XML is mostly used object oriented messaging protocol over TCP/IP [4].

III. CONCLUSIONS AND FUTURE WORK

A generic, modular, reusable, extendible test configuration using Ethernet connectivity and a test system based on this test configuration is proposed. The advantages and improvements coming with this configuration are mentioned. An example application of this test configuration and test system is described with their architecture, hardware, software and test methodology features.

Design-for-test techniques and arrangements are described which make the proposed test configuration to be used effectively. Besides these arrangements, improvement of intrasystem data interfaces and BIT features of the SUT which are determined during system design phase makes the manufacturing tests and field tests of logistic support services more efficient especially with the proposed test configuration since the proposed test configuration aims to maximize the controllability and the observability features of the SUT.

As a system designer, it can be a good foresight to specify enough number of Ethernet interfaces for the core/management units of the system and to design an Ethernet network inside the system for both manufacturing tests and field tests activities.

The most important future work activity for the proposed test configuration is multiplexing the manufacturing tests by the use of Ethernet connectivity. By installing as many virtual machines as possible onto the test PC and by performing the manufacturing test scenarios on multiple SUTs over these virtual machines independently, testing duration will significantly be reduced. As used in the proposed test configuration, the TCP/IP, socket connection and XML integration over Ethernet connectivity has one of the best structures to be used for multiplexing. The total testing duration becomes more critical especially when a manufacturing defect is detected during the test scenarios and must be fixed before taking to the next test step. In that case, with the usage of existing simple sequential structure, the test operator has to wait for the defect to be fixed or he/she may start with another test group only if the new test group to be run seems to be unrelated to the defect, which requires additional test setup/preparation duration as well. On the other hand, the test operator can continue testing on other SUTs or subsystems with the usage of multiplexed structure while the defect is being fixed at the same time, resulting in no bottleneck anymore.

Another future work activity for the test system based on the proposed test configuration is to automate the manual and semi-automatic test scenarios defined for the SUTs by automating the user interactions and manual measurements as much as possible. Before starting this work, it would be a nice approach to perform Return on Investment (ROI) analysis which is defined as cost benefit analysis in the literature [1]. ROI is used, to determine whether automation is cost effective or not for a given test scenario. Automating the manual and semi-automatic test scenarios after determining the level of automation of the test scenarios by ROI analysis, would then be a more acceptable activity for cost effectivity.

REFERENCES

[1] Rehani, M.; Abercrombie, D.; Madge, R.; Teisher, J.; Saw, J.; “ATE Data Collection - A comprehensive requirements proposal to maximize ROI of test”, in ITC International Test Conference, 2004.

[2] Alves, V.; Borba , P.; “Distributed Adapters Pattern: A Design Pattern for Object-Oriented Distributed Applications”, in First Latin American Conference on Pattern Languages Programming, Rio de Janerio, 2001.

[3] Gajski, D., D.; Vahid, F.; “Specification and Design of Embedded Hardware & Software Systems”, in IEEE Design & Test of Computers, 1995.

[4] Han, Q.; Gutierrez-Nolasco, S.; Venkatasubramanian, N.; “Reflective Middleware for Integrating Network Monitoring with Adaptive Object Messaging”, in IEEE Network, 2003.