ISSN 2347 - 3983
Volume 7, No. 12 December 2019
International Journal of Emerging Trends in Engineering Research
Available Online at http://www.warse.org/IJETER/static/pdf/file/ijeter247122019.pdf
https://doi.org/10.30534/ijeter/2019/247122019

A Formal and Enriched Framework for Testing Distributed Embedded Systems

K Chaitanya[1], Dr. K Rajasekhra Rao[2], Dr. JKR Sastry[3]
[1]Scholar, Department of CSE, JNTU Hyderabad, kilaru.chaitanya84@gmail.com
[2]Director, Usha Rama College of Engineering, Vijayawada,
[3]Professor, ,Koneru Lakshmaiah Education foundation University, Vaddeswaram, AP, India

ABSTRACT

Distributed embedded systems frequently employed for implementing many of the applications such as home automation, Automobile systems, Air surveillance, and quite recently as subnets forming an IoT (Internet of things).

Distributed Embedded systems are quite complex due to the existence of heterogeneity among hardware and software and due to the existence of variance between the message flows that should happen from the protocol uses and flow requirements of the application concerned. The distributed embedded systems must be tested considering hardware, software, and the system used for networking the individual embedded systems. For undertaking testing the distributed embedded systems, many gadgets, tools, methods, and mechanisms required. Testing communication that happens among the individual embedded systems is complex. Continuous availability of the entire distributed embedded system along with the testing system is a critical requirement for undertaking comprehensive testing, which as such cannot be guaranteed. In this paper, a framework proposed for undertaking the testing of the distributed embedded system.

Key words: Distributed Embedded System, Testing embedded systems, scaffolding, Heterogeneous embedded systems, Instruction set simulator, in-circuit emulator, logic Analyzer, assert macros, Comprehensive testing

1. INTRODUCTION

Testing Distributed embedded systems comprehensively is required for developing fail free software. An embedded system is developed using specific purpose hardware and software. Both hardware and software need to be tested, individually, and also in conjunction with each other. Testing the proper functioning of the embedded systems that are networked is also required. Testing an embedded system involves testing hardware-dependent code, independent hardware code, and testing environment required for a specific code segment.

In a distributed embedded system, several embedded systems are developed using different microcontroller-based systems which are generally heterogeneously requiring the use of middleware for doing data marshalling. Different kinds of interfaces such as RS232C, RS485, CAN, I2C, etc. are provided on the Microcontroller board, for effecting communication between the Microcontroller based systems. In a distributed embedded system network, individual embedded systems must communicate with others for implementing an application.

Each of the Microcontroller based systems is a location and as many such locations exist within a distributed embedded system. Testing must be carried at each of the locations to test the proper functioning of the local functionality and also concerning the functions running at other locations. Different kinds of methods which include scaffolding, assert macros. Instruction set simulator, logic analyzers, in-circuit emulation, etc. required for undertaking different kinds of testing such as testing for device functioning, response time, throughput, etc.,

The proper Testing environment required for undertaking testing of an embedded system. Establishing the required test environment is complex. The test environments set at each of the locations must be in working condition in conjunction with each other. Testing of embedded systems requires that the communication interfaces be working properly. Testing of embedded systems is also complex due to the existence of heterogeneity among the microcontroller- based systems used for building the individual embedded systems that act as nodes within the distributed embedded system

Scaffolding method used for testing hardware-independent code, assert macros used for testing the existence of the required environment for proper processing within the embedded system, instruction set simulator for testing the hardware in simulation mode. Logic analysers used for testing hardware, and in-circuit emulators, are used for testing the Target in communication with the HOST.

Different types of testing methods required for undertaking the testing of the embedded systems. Logic Analysers used for testing the hardware. Hardware independent code tested

using scaffolding; assert macros used for testing hardware-independent code. Instructions set simulators are used for testing hardware-independent code and testing the devices in a simulated manner and testing the hardware-dependent code undertaken through the use of in-circuit emulators. Test cases initiated from PC transmitted to target board or the Logic Analyzer for undertaking the testing and the test results are transmitted back by the Target or the Logic Analyzer back to the PC for storing and analysing the test results.

In a distributed embedded system, both hardware and software distributed into different processing nodes connected through a network. The information communicated among the processing nodes using the network to which the nodes are connected. The kind of networking system used is the key to affect communication among the individual embedded systems. The individual embedded systems are heterogeneous that they differ in many ways, which include coding systems, parity, endian, number systems, and the interfaces. The heterogeneity among the computing nodes is also due to the existence of different interfaces requiring conversion to a specific communication standard. The issues of heterogeneity are primarily due to variances existing among different networking standards that include CAN, USB, I2C, RS485, etc. Most of the times serial, bus-based communication systems used for networking distributed embedded systems.

Use of a specific networking standard dictates the kind of testing done. The kind of testing to be done largely varies as the communication protocol changes. The interfaces available on each of the microcontroller-based system may not be suitable for communicating using a specific standard. Most of the times, there is a need to convert the native interface to the required interface so that a Microcontroller- based system participates in the network as a computing node.

Many methods presented in the literature for testing standalone embedded systems which can be investigated further to see how best these methods can be used for undertaking testing of the distributed embedded systems. The issue of setting the environment required for undertaking the testing of the distributed system needs investigation since it is one the most complex issue. Availability of the entire distributed system in working condition is another important issue that needs consideration. It is not possible to test any system unless the entire system is in working condition. Simultaneous testing at different locations in an isolated manner and an integrated manner is complex and challenging

Many processes, methods, techniques used for undertaking testing of the distributed tems in a most standard manner. A framework which encompasses all elements of testing will help to undertake to test a distributed embedded system formally. The existing methods used for a testing stand-alone embedded system must be modified, extended and included into the framework so that the framework used for undertaking the testing of distributed embedded system,

The test cases defined at system level must be broken into elementary test cases so that the elementary test cases used for undertaking testing at the individual location and the results obtained at each location are merged to arrive at over test results. One can customize the framework for suiting to the requirements of specific distributed embedded systems.

The entire distributed embedded system must be in working condition for testing a system-level test case. It is not possible to meet this kind of requirement due to the existence of many subnets within a distributed embedded network. A strategy thus is required for carrying testing without the need for an entire distributed embedded system in working condition. The strategy leads to a model which help testing carried at individual locations and the test results merged to get the overall status of the entire distributed embedded system

The system-level test cases are decomposed to elementary test cases to arrive at test cases tested at a specific location. The elementary test cases are such that they can be tested using a specific method at a specific location. The test results obtained at each location when merged will project the overall test status of the entire system. System-level test cases derived from the requirement specification of the distributed embedded system.

Many aspects considered when a distributed, embedded system tested. The aspects include interfaces, process flow, heterogeneity protocols, response time, throughput, device status, the existence of proper environment, etc. the methods required for undertaking the testing must be identified considering every element of testing carried.

A testing framework useful for testing any distributed embedded system presented in this paper. The framework, as such is extendable. To meet the testing requirements of individual distributed embedded systems.

2, PROBLEM DEFINITION

Thus the problem is to create a framework that is useful for undertaking the testing of distributed embedded systems without the need to have the entire system in working condition along with the test environment system. The setting should be undertaken considering different segments of the system and the method used for testing the system.

3. RELATED WORK

In literature, many authors have presented the use of standard methods of testing either stand-alone systems or distributed embedded systems. The analysis of the methods proposed in the literature survey reveals that no specific standard methods have been in existence for testing a distributed embedded system using different methods. No comprehensive framework presented in the literature that helps in testing distributed embedded systems.

[Chen-Huan Chiang, et al., 2004] [1] Test architecture aims to transmit JTAG signals over a serial channel. The architecture has been developed to facilitate system testing and automatic field updating of distributed base stations situated in a wireless network. The test architectures assume that the distributed bases stations are on the same back pane and the same chassis. The architecture considers the use of boundary SCAN software, which is run by the processor situated within a wireless sensor node. The nodes configured by the SCAN software receiving instructions from a remote location. The SCAN software will also be able to conduct a system test and find if any system errors exist.

[Dae-Hyun Kum et al., 2006] [2] have presented a model-based system used for the development of an embedded system. The model-based systems are useful as it improves quality, and the development done is the least possible time. Simulating a system is an essay when model-based development undertaken. The system, as such, can be validated in the early stages of the development. Test cases automatically generated when systems are developed using the models. All the test cases required for validating the models and the functions can be generated using the models. Virtual prototypes o the models developed for undertaking the testing.

An electronic communication system is presented by [Eric Armengaud et al., 2005] [3] that connects all the individual embedded systems fitted into an automobile system. Testing of the embedded systems fitted into an automobile system required as the failure of any system may lead to disasters. Test cases are required to test the automobile system under stringent conditions. A method is presented to generate test cases based on the stimulus-response model under tough conditions. They have developed a method that is accurate and flexible, which generates test data for testing the communication within the data link layer. They have used the method for testing robustness and interoperability between the distributed systems.

Changes to the existing applications are needed due to new requirements or due to the introduction of new and sophisticated technologies. The changes made to the software may affect the code area where no changes caused; Regression testing is to be accrued to find whether the changes made to the software affected other areas of the code. Specialized hardware and software are required some times to conduct regression kind of testing. The type of testing tool selected depends on the strategy of the organization concerned. A regression testing tool is needed can be configured by the organizations as per their needs. Manual test processes are complex and time-consuming and therefore needs avoidance. Tool based testing is robust, and the process of undertaking testing rather becomes simple — [G. Walters et al., 1998] [4] have proposed an automated regression test tool that can be configured by the users as per their requirements.

[H. Thane et al., 1999] [5] have presented the testing method used for testing sequential programs by controlling the sequence of inputs fed to the application as input. Sequential test inputs are to be presented in a specific order and during specified time intervals and the time duration during which the concurrent tasks executed. One should not use sequential test techniques as they do not figure out the significance of the occurrence of the tasks.

An efficient architecture helps to develop decentralized, reliable, collaborative, and rapid applications, which need to be highly responsive real-time and distributed embedded systems. These systems have inbuilt processes for undertaking the testing. J. Russell Noseworthy, 2008] [6] have named the architecture as TENA (Testing and Training enablement architecture). A middleware built into TENA helps in code generation that is understandable through easy to understand abstractions. An excellent API included in TENA is capable of detecting programming errors at compilation time. TENA includes software components that can be used to undertake different kinds of testing.

Environment setting becomes very important for undertaking testing of any embedded system [Pei Tian et al., 2009]. [7] The basics of the environment setting must be analysed considering the basic structure, functionalities, and characteristics of distributed software. A three-layer development pattern proposed that facilitates the setting appropriate test environment. This kind of proposal is quite difficult to implement as it is not possible to dictate a structure for the development of individual applications within distributed embedded systems.

Distributed and networked embedded systems are being used heavily in automobile, space, and many other such applications. Testing distributed applications are complex. The distributed applications are generally component-based and exhibit dynamic behavior. Dynamic interaction, structural behavior, run-time configurations, etc. makes the testing of distributed systems complicated. It has been proved time and again that any amount of testing carried on the developed product; some unknown errors noticed during the production time. Therefore, it is necessary to undertake to test, while distributed embedded systems are in the production phase. [Peter H. Deussen et al., 2002] [8]. Therefore, there is a need to develop concepts and methods using which a distributed system tested while the system is in running mode. Online testing of distributed embedded systems is, therefore, necessary. Online testing will help to undertake the testing of functionality under limited time, resources available, complex transactions that performed between the components.

Most of the distributed embedded systems are built using fault tolerance concepts. One of the main challenges is to find the errors occurring while the distributed embedded systems are in run-state. Faults can occur at any level. The faults occurring at the PIN level normally affect the network interfaces and the communication that is built to facilitate

communication between various distributed nodes which are networked. Architecture is proposed [Sara Blanc et al., 2003] [9] which consider the use of a monitor that keeps monitoring the faults occurring at the PIN level. The monitor observes the system behavior and also detects whether any failure has occurred at any of the PIN.

Interconnecting the distributed embedded systems are error-prone due to the presence of many intricate issues. Individual embedded systems as such may be error-free and become error-prone as they get connected to a network. Many methods used for undertaking the automated testing to trace out the bugs existing in the working of distributed embedded systems. [Silvie Jovalekic et al., 2008][10] have proposed cause and effect graphs which are time-dependent to describe the test cases considering the distribution and real-time properties. Tests object structure used for undertaking the testing in-depth. They have proposed a simple language describing test objects consisting of modules and connections. The language enables graphical documentation and context-sensitive protocol analysis. Symbolic representation of received messages facilitates better comprehension of system behavior.

Simulators also are used for Testing distributed embedded systems, [Steven A. Walters 1994][11] has presented a methodology for developing a simulator meant for testing a real-time distributed embedded system. The architecture deals with the various issue that includes reuse, expandability, reconfigurability, and modularity. However, the simulations model found to be inadequate for testing distributed embedded systems as the model as such is not suitable to handle inter-process communication and the requirement for proper scheduling the tasks and the need for establishing communication between concurrent tasks.

It is quite a difficult test distributed system, especially considering the issues that include synchronization, collaboration, concurrency, timing, and interoperability among the concurrent tasks. Lots of time needed for developing code required for testing a distributed embedded system. To address this issue, T. Tsai et al., 2003] [[12] have proposed a method that helps testing a distributed system quite rapidly. They have used methods for modelling test scenarios, state transitions, design, and verification patterns ripple effect analysis, regression testing, executing the test cases automatically.

Testing an embedded system can be carried by finding thin threads which represent END-TO-END testing. END-TO-END testing is an integration testing approach starting from sensing to actuating and development of a historical database. [Tsai W. T et al., 2003] [[13] have presented a method that helps to carry END-TO-END testing. They have used the concept of verification patterns used for undertaking testing. But this approach has not been applied for testing distributed embedded systems.

A massive number of individual embedded systems were tested together for reducing the time required for testing. [Yanfang Wang et al., 2010] [14] has used a master-slave system in which a PC used as a HOST for undertaking the testing. RS485 networking used for undertaking mass device testing. The arrangement used for undertaking individual device testing and not used testing of distributed embedded systems itself.

One can use a logic Analyzer for testing proper working of the hardware of an embedded system by connecting probes to the junction points exposed from embedded systems. Commands are sent to a Logic Analyzer so that the LA does the testing required and forward the test results back to the PC. [David E. Simon, 1999] [15] presented a method using which testing of an embedded system carried with the help of a Logic Analyzer. The testing using logic analyzers carried either is static or timing mode. The way the testing of an FPGA based board, signal integrity and memory devices using Logic analyzers has been presented by [Tektronix, 2006] [16] in their white paper,

Kyeongjoo Kim et al., [17] have presented, an analysis of streaming the data flowing across the systems presented, which viewed as the basis for proper data flow across the network. Sasi et al., [18] has presented a gaming system which used for testing an embedded system

Many frameworks have been presented in literature for the development of either stand-alone embedded systems or distributed embedded systems. The frameworks presented in the literature includes Component Frame work [18], Rapid Application development [19], Scalable analysis and design of system-wide graceful degradation of distributed embedded systems [20], scheduling and optimising distributed embedded systems[21], and development of Hard Real time distributed embedded systems

A framework has been presented that can be used for diagnosing the faults occurring in distributed embedded systems. Model based diagnosis architecture is presented by Gregory Provan et al. [22] which considers distributed sub-systems connected through a Graph. The model computes both local minimum diagnosis and Global diagnosis. The authors focus on to find what errors have happened instead of testing whether the system is properly developed and functions properly as per the functional requirements.

Furthermore contributions have been in presenting the frameworks that are related to Building Time components [23], developing frameworks that are related to development of real-time distributed embedded systems [24], Frameworks relating to either security aware or building security [25]
A novel method has been presented by [J.K.R Sastry et al., 2015] [26] for networking different heterogeneous embedded systems through RS485 and bus-based serial communication system. One of the computing node connected to the network behaves like a master having full access and control of the bus.

[J.K.R Sastry et al., 2015] [27] have proposed an efficient method of networking heterogeneous systems using I2C communication system. All issues related to networking, including synchronization, timing, arbitration, design of data packets, etc. presented in this paper. [Sastry et al., 2015] have addressed the design of USB based network for connecting heterogeneous Microcontroller based system, design of specific communication system as required by the distributed embedded application, address allocation to the salves and configuring the slaves through descriptors for making them adaptable for the implementation of distributed embedded application. The designing of the messages and controlling the flow of messages across the distributed Microcontroller based system has been presented considering a distributed embedded system that monitors and controls temperatures within a Nuclear reactor system.

Networking of a distributed embedded system is achieved through networking using the CAN-based communication system, which is a BUS based serial communication system. Every communication system requires that messages communicated in a specific sequence. The application requires the transmission of messages in a specific sequence. Both message systems must be combined to arrive at a composite communication system. [Sastry et al., 2015] have presented a novel method using which an arbitration method that takes message flows into account has been presented, leading to efficient communication using CAN-based communication system.

Protocols specify the way the messages must flow using the data packets of different types. Applications require the flow of messages in a proper sequence and order. A mapping method is requited that ensure the movement of application-specific messages while following the way a protocol dictates the flow of messages.[Sastry et al., 2017] [28] have proposed a method of organizing the movement of application-specific messages without

[K. Chaitanya et al., 2018] [29] have presented the way testing of a distributed system carried when networked using CAN protocol and using the scaffolding method for testing. They have also presented a method [30] of testing a distributed embedded system using networked through an RS485 network and using the scaffolding method for undertaking testing of the distributed embedded system.

[Chaitanya et al., 2017] [31] have proposed a method using which testing of a distributed system can be undertaken using assert macros to find the existence of the required environment for undertaking specific embedded processing. Assert Macros are inserted into the code dynamically either through the established pointers or through an interactive process. Macros are generated based on test scripts, and the same inserted into the embedded application as in-line code. Instruction set simulators used by [Chaitanya et al., 2017][32] for testing embedded systems, especially testing for throughput and response time while simulating the hardware devices which meant for carrying Input/output. [Chaitanya et al., 2017][33] have used instruction set emulator for undertaking the testing the functioning of the Hardware and software considering the Target with the test cases initiated from HOST. Logical analyzers have been used by [Chaitanya et al., 2018] [34] for testing the proper functioning of the hardware. The testing of the proper functioning of the hardware is undertaken by Logic analyzers, which fed with the test cases initiated from HOST.

[K. Chaitanya et al., 2018] [35] have presented the way testing of a distributed system carried through the different testing method by using a repository of master test cases and integrating them. [Chaitanya, 2013] [36] have explained the complications of Embedded Systems that can occur in Agriculture Technology by using a Customized Software. A frame work proposed by Chaitanya et al. [37] based on the process flows that happen within a framework which can be used for undertaking testing of distributed embedded systems.

4. INVESTIGATIONS AND FINDINGS

Testing Distributed embedded systems is complicated. Several test components, connectivity with external devices, simulators, process flows, system integration is required. Entire distributed embedded system must be in working condition to enable testing. The proper working of communication system is essential. Testing considering entire distribution is complicated as interworking of testing gadgets connected to the distributed embedded system is complicated. Different work flows have to be followed for undertaking testing of different types. For instance the fork flows to be followed for testing using Scaffolding method is different from the workflow to be used for testing through In circuit emulators

Distinct and different components are to be used for undertaking testing distributed embedded systems. The test components must be integrated and formed into a framework which can be used for undertaking testing of any distributed embedded system.

The testing framework must include processes that can be used for extracting the test cases from functional specification of typical distributed embedded system, decomposing the test cases logically such that the test cases be tested using a specific method and selection of a specific embedded system at which testing must be undertaken.

A specific environment is required for undertaking testing of the embedded systems. The testing framework must include process that helps establishing the test environment required for undertaking actual testing of the embedded systems.

Different workflows are followed for undertaking testing using the methods such as testing through scaffolding, assert macros, Instruction set simulator, in-circuit emulator and Logic Analyser. The framework must have a component that supports different workflows needed for undertaking testing of distributed embedded systems.

An integrator is required for integrating test results and the test results that reflects the overall testing of distributed embedded systems is required.
Considering the complexity of testing involved, testing distributed embedded system manually is not possible. A framework that automates various processes that can be used for testing any of the distributed embedded system is required.

A novel framework is developed that has all the stated processes, components and workflows. The framework is shown in Figure 1.0.

The framework includes the following segments.

1. Accessing the test requirements, decomposing the test cases such that testing can be done using a specific method and at a allocations
2. Environment setting
3. Mapping Test cases with the required test environment and generation of test scripts
4. HOST based Application code management
5. Testing through Scaffolding
6. Testing through Assert Macros
7. Testing through Instruction set simulator
8. HOST based Test Application Management
9. Testing Hardware through Logic Analyzer
10. Target Based Application code Management and Testing through In-circuit Emulator
11. Test Result integrator and Audit Trail developer

4.1 Generating test cases

No formal methods till now exist to automatically generate the test cases that can be used for undertaking the testing of the distributed embedded systems. Test cases are to be generated manually using the functional specification of the application system. An interactive GUI can be used for generating the test cases at system level. An interactive GUI can also be used for decomposing the System level test cases elementary test cases such that a test case can be tested using a single method and at a specific location. The framework flow related to generation of test cases is shown in Figure 2.

Test methods includes the methods used for undertaking testing that comprise testing through scaffolding, Assert macros, Instruction set simulator, In-circuit emulator and Logic Analyzer and any other method that the user might like to add. Test locations are the individual embedded systems at which testing must be undertaken. The decomposition of the master test cases and mapping to test methods and test locations can be undertaken using a ladder diagram. The master cases are serial numbered and decomposed test cases are also numbered as an extension to the master test case serial number thus making a link between the decomposed test cases and master test case. Each of the decomposed test cases is mapped to a testing method and the location where the testing should be undertaken. The mapping can be done using interacting GUI based ladder diagram.

4.2 Setting Environment

Environment setting implies the commands and command line arguments that must be used for communicating between the test processes. Environment testing also is related to the function sequences that must be called to facilitate testing using the test cases. The function code sequences that must be executed when testing is to be carried are generated so that the same can be mapped to test cases subsequently. The framework segment related to setting environment for undertaking testing is shown in Figure 3.

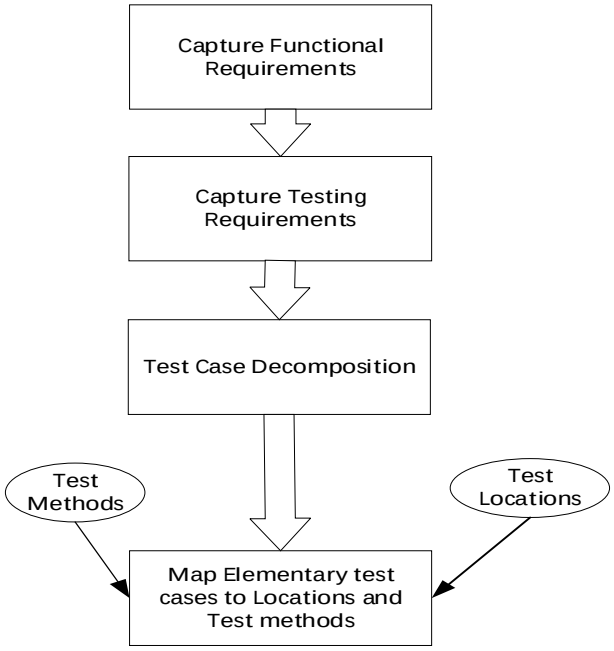

Figure 2: Generating test cases mapped to test methods and Locations

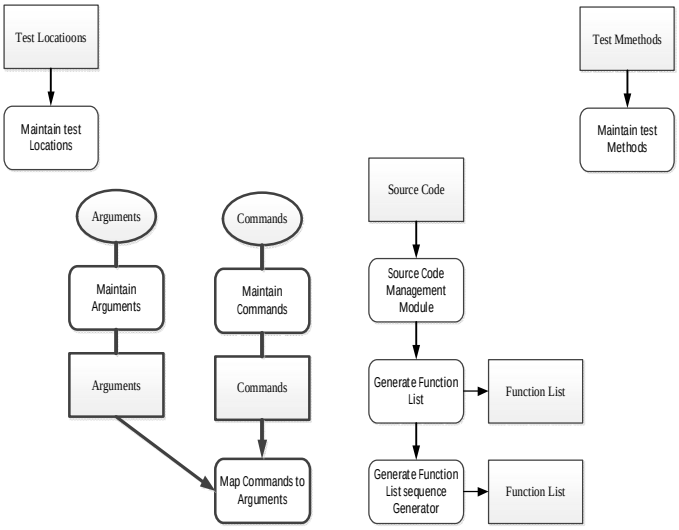

Figure 3: Framework for Environment setting

Commands indicated the type of testing to be carried. Command Line arguments are the inputs that must be fed for undertaking testing also the output variables into which the test results must be stored. Users can interact with the framework component and set the environment required for undertaking testing as per their requirement.
The user defined test methods and Test Location are maintained through USE of a Graphical Interface and storing the contents in a database

4.3 Mapping test cases to test environment

The test cases are to be mapped with its related commands and command line arguments which represents the test input and expected test results. A separate segment of the framework achieves this objective. Figure 4 shows the required framework segment. Test scripts are generated for each of the test case, which contains the test case, command representing the test cases, input and output arguments. From now on words test scripts represents the test cases that can be presented as input to the processes that actually carries testing and produce test results.

4.4 Host based Application code management

Host implies PC side of computing. The firmware code needs to be updated with additional code components that are required for undertaking testing. A code component called parser needs to be added which is responsible for reading test case, test data, assign the test data to internal variables and also responsible for calling functions in a sequence that contribute to executing the test case.

The firmware source must also be updated to cater for testing using scaffolding methods especially to comment hardware dependent code. The firmware source code needs to be added with macros for undertaking environment testing that is required for proper working of different segments of the system. The host based application code management is shown in the Figure 5.

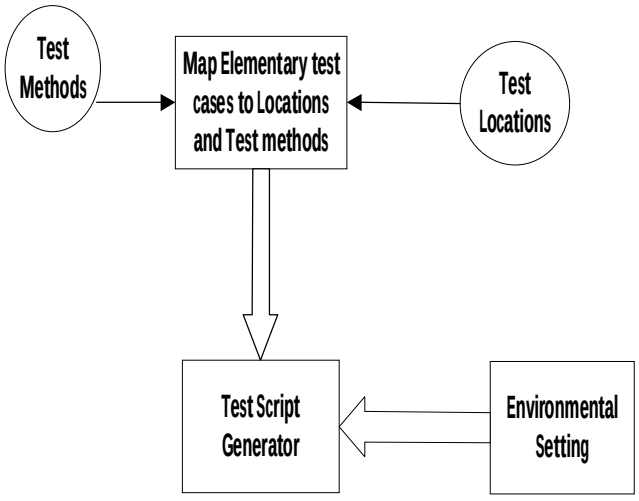

Figure 4: Mapping test cases to test environment

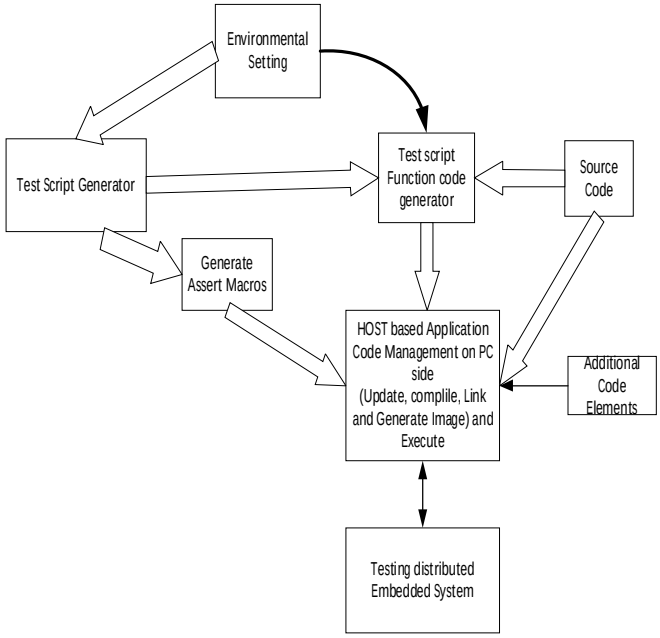

Figure 5: HOST based Application management

The PC side application code management component is responsible for maintaining the source code, make changes to the code and add software components required for undertaking the testing.

This component also have functions required to update the code, link with library and compile the same and produce an image that can be executed.

The parser code is responsible for undertaking testing. Each test case is tested by calling the related functions in a specific sequence. This component fetches the sequence of functions to be called for undertaking testing relevant to a test case and includes the same within the parser code. The association between a test case and sequence of functions to be called is established as a part of environment setting

4.5 Testing through Scaffolding

Testing through scaffolding can be undertaken once the application management on the HOST is complicated. Testing through Scaffolding is undertaken at each of the distributed embedded system. The HOST based application management module will be installed on all the distributed embedded systems and the application is initiated for execution through a separate Module “Testing distributed Embedded System” which is designed to centrally manage the entire testing process. The Framework segment used for testing using scaffolding method is shown in Figure 6. The HOST based application reads the test script one after the other and the test results are sent are written to a data base resident on the HOST.

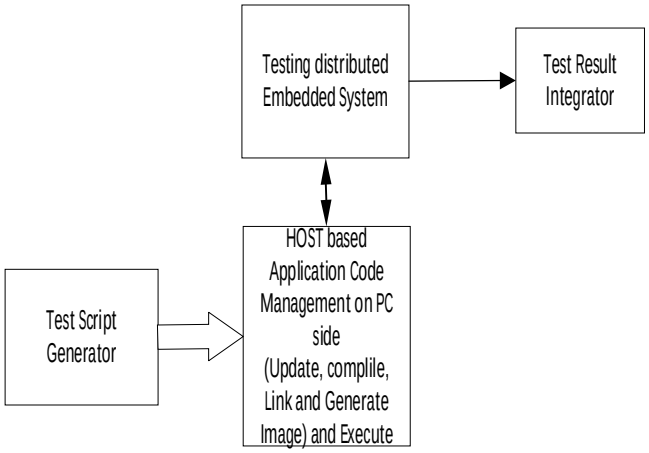

Figure 6: Framework for testing through Scaffolding

4.6 Testing through Assert Macros

Testing through Assert macros can be undertaken once the application management on the HOST is complicated. Testing through Asset macros is undertaken at each of the distributed embedded system. The HOST based application management module will be installed on all the distributed embedded systems and the application is initiated for execution through a separate Module "Testing distributed Embedded System" which is designed to centrally manage the entire testing process. The Framework segment used for testing using Assert macros method is shown in Figure 7. Assert macros are the test cases and therefore there is no need for the HOST based application to read the test cases.

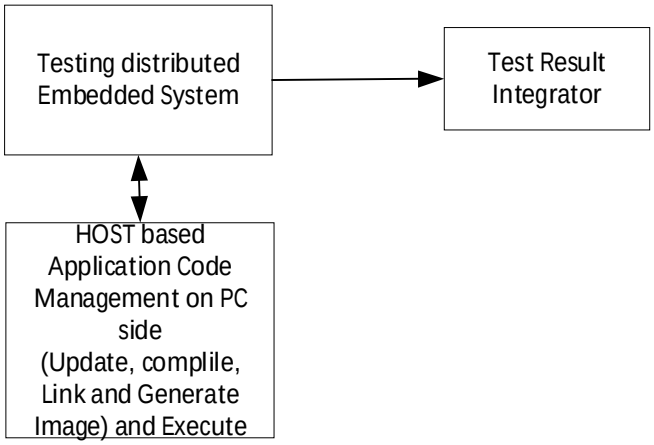

Figure 7: Framework for testing through Assert Macros

The HOST based application is executed and the test results are written to a database resident on the HOST

4.7 Testing through Instruction set simulator

The framework relating to testing through Instruction set simulator is shown in the Figure 8. The testing process is initiated through centralised component that manages entire testing process. Instruction simulator reads the source code instruction by instruction related to the functions that must be executed which are related to a specific test case. The Instruction stimulates the execution of the source code as if the execution is done on target Micro Controller. The test results are stored in a Database existing on the HOST.

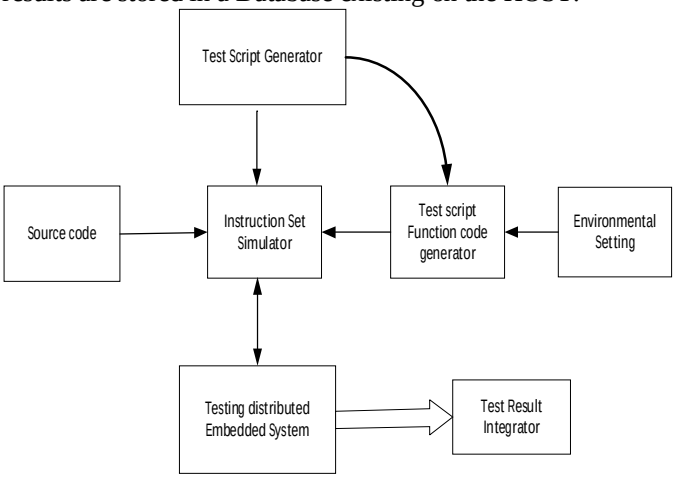

Figure 8: Framework for testing distributed embedded systems through instruction set simulator

4.8 HOST based Test Application Management

A separate application is built on the HOST which is used for undertaking testing using logic Analyser and In-Circuit emulators. The application communicates with the Logic Analyzer and the In-Circuit emulator for submitting a test case and receives the test results which are written to a centralised database. The frameworks design for developing a HOST based application used for undertaking testing using logic analyser and In-Circuit emulator is shown in Figure 9.

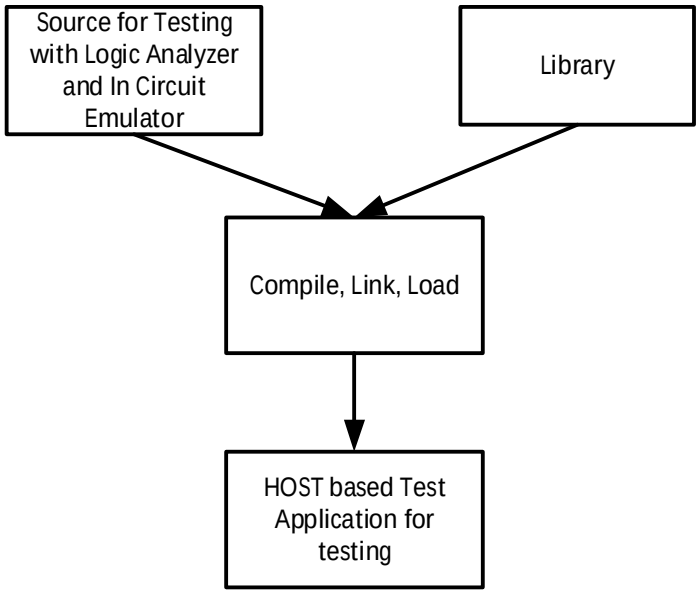

Figure 9: Framework segment for development of HOST based application used for undertaking testing using Logic Analyzer and In-Circuit emulator.

4.9 Testing Hardware through Logic Analyzer

Figure 10 shows the framework segment that deals with testing hardware using a Logic Analyzer. Logic Analyzer is interfaced with the target using probes. The central application triggers the HOST based application when testing using Logic Analyzer is required. The HOST based application reads the test cases and sends commands to Logic Analyzer for undertaking testing and making available test results which are stored in a centralised database.

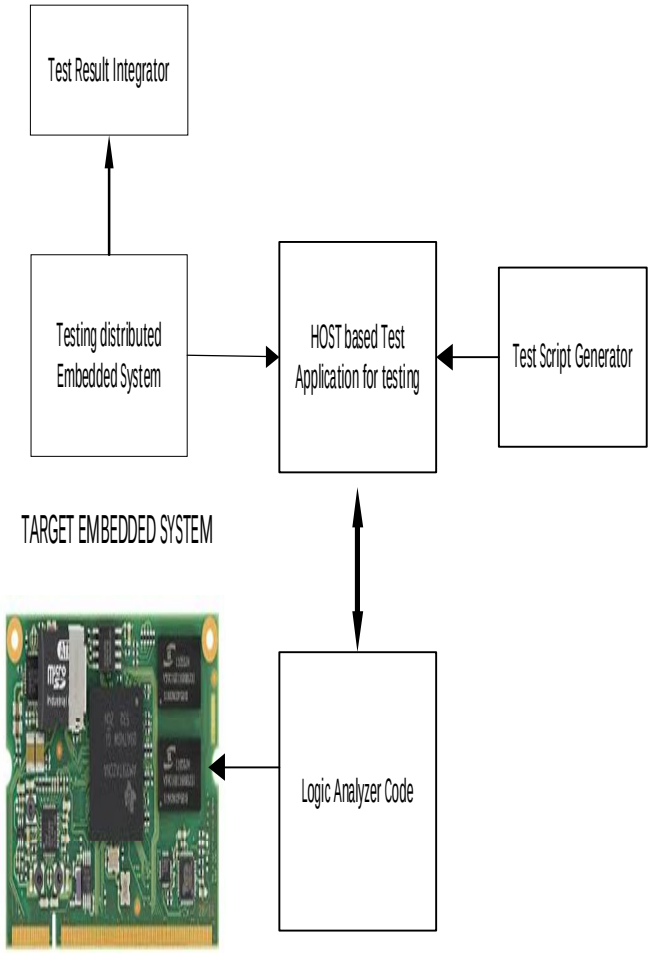

Figure 10: Framework Segment for testing through Logic Analyzer

4.10 Target Based Application code Management and Testing through In-circuit Emulator

The framework related to testing through In-Circuit emulator is shown in figure 11.

The target code is added with the test components and a new application is developed and loaded into the Target Machine through a separate interface. The centralised component triggers a HOST based application for undertaking testing through In-Circuit Emulator. The Host based application reads a test script, forwards the same to the target which then undertakes testing and the test results are sent back to the HOST based Application which then stores the test results in a Centralised database.

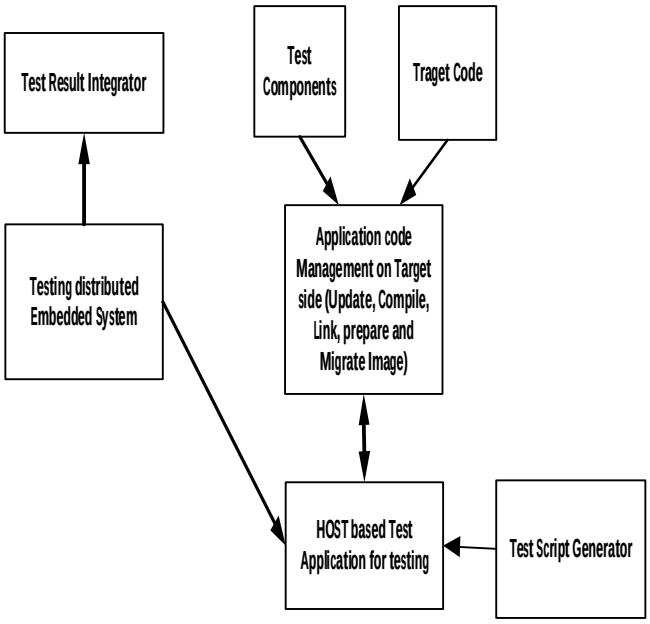

Figure 11: Framework Segment for Testing through In-circuit Emulator

4.11 Test Result integrator and Audit Trail developer

The Framework segment that processes the final test results is shown Figure 12. The central component triggers the Test result Integrator component for combing and integrating the test results that have been written into a centralised database by the individual centralised test components that are resident on different distributed embedded systems. The integrated test results then are processed to find the reliability of the system

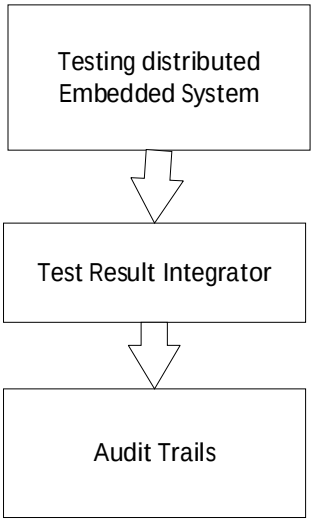

Figure 12: Framework segments for integrating test results and conducting Audit trail.

5. CONCLUSION

Testing Distributed embedded system is complicated as it involves testing at individual locations and also considering entire system as whole.

The testing of distributed embedded systems involves setting testing environment, generation of test cases and creation of test scripts, source code management on the HOST and source code management on the target. Several framework components are required for undertaking different kinds of testing carried through methods such as Scaffolding, Assert macros, Instruction set simulators, Logic Analyzer and In-circuit Emulator. Framework components also are required for testing both hardware and Firmware. The test results obtained at different locations have to be collected and collated at one place so that the same can be integrated and system level test results are produced.

Testing distributed embedded systems requires a framework that can be adapted for testing any type of distributed embedded systems.

REFERENCES

1. Chen-Huan Chiang, Paul J. Wheatley, Kenneth Y. Ho, Ken L. Cheung, Testing and Remote Field Update of Distributed Base Stations in a Wireless Network, IEEE Conference Publications, 2004, page no.711-718
2. Dae-Hyun Kum, Joonwoo Son, Seon-bong Lee and Ivan Wilson, Automated Testing for Automotive Embedded Systems. IEEEConference Publications, 2006, page no. 4414-4418
3. Eric Armengaud, Andreas Steininger, Efficient Stimulus Generation for Testing Embedded Distributed Systems -The Flex Ray Example, IEEE, 2005, page no.763-770
4. G. Walters, E. King, R. Kessinger, R. Fryer. Processor design and implementation for Real-Time Testing of embedded systems. IEEE Conference Publications, vol.1, 1998
5. H. Thane, Real-Time Res. Center, Malardalen Univ., Vasteras, Sweden, H. Hansson, Towards systematic testing of distributed real-time systems, Real-Time Systems Symposium. Proceedings. The 20th IEEE, 1999, page no.360-369
6. J. Russell Noseworthy. The Test and Training Enabling Architecture (TENA) —Supporting the Decentralized Developmentof Distributed Applications and LVC Simulation. IEEE Conference Publications, 2008, page no. 259-268 https://doi.org/10.1109/DS-RT.2008.35
7. Pei Tian, JianchengWang, HuaijingLeng, Kai Qiang. Construction of Distributed Embedded Software Testing Environment. IEEE Conference Publications, vol.1,2009, page no. 470-473 https://doi.org/10.1109/IHMSC.2009.125
8. Peter H. Deussen, George Din, Ina Schieferdecker, An online Test platform for component-based systems. IEEE Conference Publications, 2002, page no.96-103
9. Sara Blanc, Pedro. J. Gil. Improving the multiple errors detection coverages in distributed embedded systems. IEEE Conference Publications, 2003, page no. 303-312
10. Silvie Jovalekic, Bernd Rist, Test Automation of Distributed Embedded Systems based on Test Object Structure Information, IEEE Conference Publications, 2008, page no. 343-347 https://doi.org/10.1109/EEEI.2008.4736543
11. Steven A. Walters, Practical Techniques for Distributed Real-time Simulation. IEEE Conference Publications, vol.2,1994, page no. 890-896
12. Tsai W. T., R Mojdehbakhsh and F. Zhu, Ensuring Systems and Software Reliability in the Safety-Critical Systems, IEEE ASET 98, Dallas, Texas, March 1998, page no.48-53
13. W. T. Tsai, L. Yu, A. Saimi. Scenario-Based Object-Oriented Test Frameworks for Testing Distributed Systems. IEEE Conference Publications,2003, page no.288-294
14. Yanfang Wang, Wandui Mao, Jinying Li, Peng Zhang, Xiaoping Wang, A Distributed Rectifier Testing System Based on RS-485. IEEE Conference Publications, 2010, page no. 779-781 https://doi.org/10.1109/ICIEA.2010.5515241
15. David E. Simon, An Embedded Software Premier, Pearson Publications, 1999, page no.313-319
16. The XYZs of Logic Analysers Primer, Tektronix, A Logic Analyzer Tutorial part1, http:// nutsvolts.texterity.com/, nutsvolts / 200709/?folio=71&pg=71#pg71
17. Kyeongjoo Kim, Jihyun Song, Minsoo Lee, Real-time Streaming Data Analysis using Spark, International Journal of Emerging Trends in Engineering Research, Volume 6, No.1, 2018, pp. 1-5 https://doi.org/10.30534/ ijeter/2018 /01612018
18. Christo Angelov, Krzysztof Sierszecki, Nicolae Marian, and Jinpeng Ma, A Formal Component Framework for Distributed Embedded Systems, Gorton et al. (Eds.): CBSE 2006, LNCS 4063, pp. 206 – 221, 2006.
19. R. Obermaisser, P. Peti, A Framework for Rapid Application Development of Distributed Embedded Real-Time Systems
20. Charles P. Shelton, Philip Koopman, William Nace, A Framework for Scalable Analysis and Design of System-wide Graceful Degradation in Distributed Embedded Systems, WORDS03 – January 2003
21. Adrián Noguero, Isidro Calvo, A Framework with Proactive Nodes for Scheduling and Optimizing

Distributed Embedded Systems, HAL Id: hal-01056492 https://hal.inria.fr/hal-01056492 Submitted on 20 Aug 2014

22. Christo Angelov, Krzysztof Sierszecki, Feng Zhou, A Software Framework for Hard Real-Time Distributed Embedded Systems, 2008 IEEE
23. Chetan Raj1, Jiyong Park1, Jungkeun Park2 and Seongsoo Hong, CREAM: A Generic Build-time Component Framework for Distributed Embedded Systems, 2008 IEEE.
24. Khaled Chaaban, Paul Crubillé, and Mohamed Shawky, Real-Time Framework for Distributed Embedded Systems, pp. 96–107, 2004. © Springer-Verlag Berlin Heidelberg 2004
25. Hyunsuk Nam and Roman Lysecky, Security-Aware Multi-Objective Optimization of Distributed Reconfigurable Embedded Systems, https://doi.org/10.1016/j.jpdc.2018.02.015
26. Dr. J. Sasi Bhanu, Dr. JKR Sastry, B. Sunitha Devi, and Dr. V Chandra Prakash, Career Guidance through TIC-TAC-TOE Game, International Journal of Emerging Trends in Engineering Research, Volume 7, No.6, 2019, pp. 25-31 https://doi.org/10.30534/ijeter/2019/01762019
27. J. K. R. Sastry, A. Suresh, and Smt J. Sasi Bhanu, Building Heterogeneous Distributed Embedded Systems through RS485 Communication Protocol, ARPN Journal of Engineering and Applied Sciences, issue. 16, vol.10, 2015
28. Sastry JKR, J Viswanath Ganesh, Sasi Bhanu J, "I2C based Networking for Implementing Heterogeneous Microcontroller, based Distributed Embedded Systems", Indian Journal of Science and Technology, Vol. 8, Vol. 15, pp. 1-10, 2015-1 https://doi.org/10.17485/ijst/2015/v8i15/68739
29. Sastry JKR, Sai Kumar Reddy, Sasi Bhanu J, "Networking Heterogeneous Microcontroller based Systems through Universal serial bus," International Journal of Electrical and Computer Engineering, Vol 5, Iss. 5, 2015-2
30. Sastry JKR, Vijaya Lakshmi Machineni, Sasi Bhanu J, "Optimizing Communication between heterogeneous distributed Embedded Systems using CAN protocol," ARPN Journal of engineering and applied sciences, Vol. 10, Iss. 18, Pg. 7900-7911, 2015-
31. JKR Sastry, T. Naga Sai Tejasvi and J. Aparna, Dynamic scheduling of message flow within a distributed embedded system connected through RS485 network, ARPN Journal of Engineering and Applied Sciences, VOL. 12, NO. 9, MAY 2017
32. K. Chaitanya, Dr. K. Raja Sekhara Rao, Testing Distributed Embedded Systems Built over CAN using Scaffolding Method, International Journal of Emerging Technology and Advanced Engineering, issue.12, vol. 8 December 2018, page no. 28-42. https://doi.org/10.30534/ijatcse/2019/84842019
33. J.K.R. Sastry, K. Chaitanya, K. Rajasekhara Rao, D.B.K. Kamesh, An Effective Model for Testing Distributed Embedded Systems using Scaffolding Method, PONTE International Journal of Sciences and Research, issue.8, vol.73, 2017, https://doi.org/10.21506/j.ponte.2017.8.1
34. K. Chaitanya, Sastry JKR, K. N. Sravani, D. Pavani Ramya and K. Rajasekhara Rao,Testing Distributed Embedded Systems Using Assert Macros, ARPN Journal of Engineering and Applied Sciences, 2017, page no.3011-3021
35. Sastry JKR, K. Chaitanya, K. Rajasekhara Rao, DBK Kamesh, Testing Distributed Embedded Systems Through Instruction Set Simulators, PONTE, International Journal of Sciences and Research, issue.7, vol.73, July 2017, page no.353-382
36. JKR Sastry, K. Chaitanya, K. Rajasekhara Rao, DBK Kamesh, An Efficient Method for Testing Distributed Embedded Systems using In-circuit Emulators, PONTE, International Journal of Sciences and Research, issue.7, vol.73, 2017, page no.390-422
37. K. Chaitanya, JKR Sastry, K. Rajasekhara Rao, Testing Distributed Embedded Systems Using Logic Analyzer, International Journal of Engineering and Technology, March 2018, page no. 297-302.
38. Chaitanya Kilaru, K. Rajasekhara Rao, Comprehending Testing of distributed embedded systems, International Journal of Engineering and Technology, issue. 2.7, vol. 7 March 2018, page no. 303-307
39. K. Chaitanya, K. Rajasekhara Rao, Complication of Embedded Systems in Agriculture Technology using Customized Software, International Journal of Emerging Technology and Advanced Engineering, issue. 7, vol. 3, July 2013, page no. 368-373
40. K Chaitanya, Dr. K Rajasekhra Rao, Dr. JKR Sastry, A Framework for Testing Distributed Embedded Systems, International Journal of Advanced Trends in Computer Science and Engineering, Volume 8, No.4, July – August 2019, Pp. 1194-1227

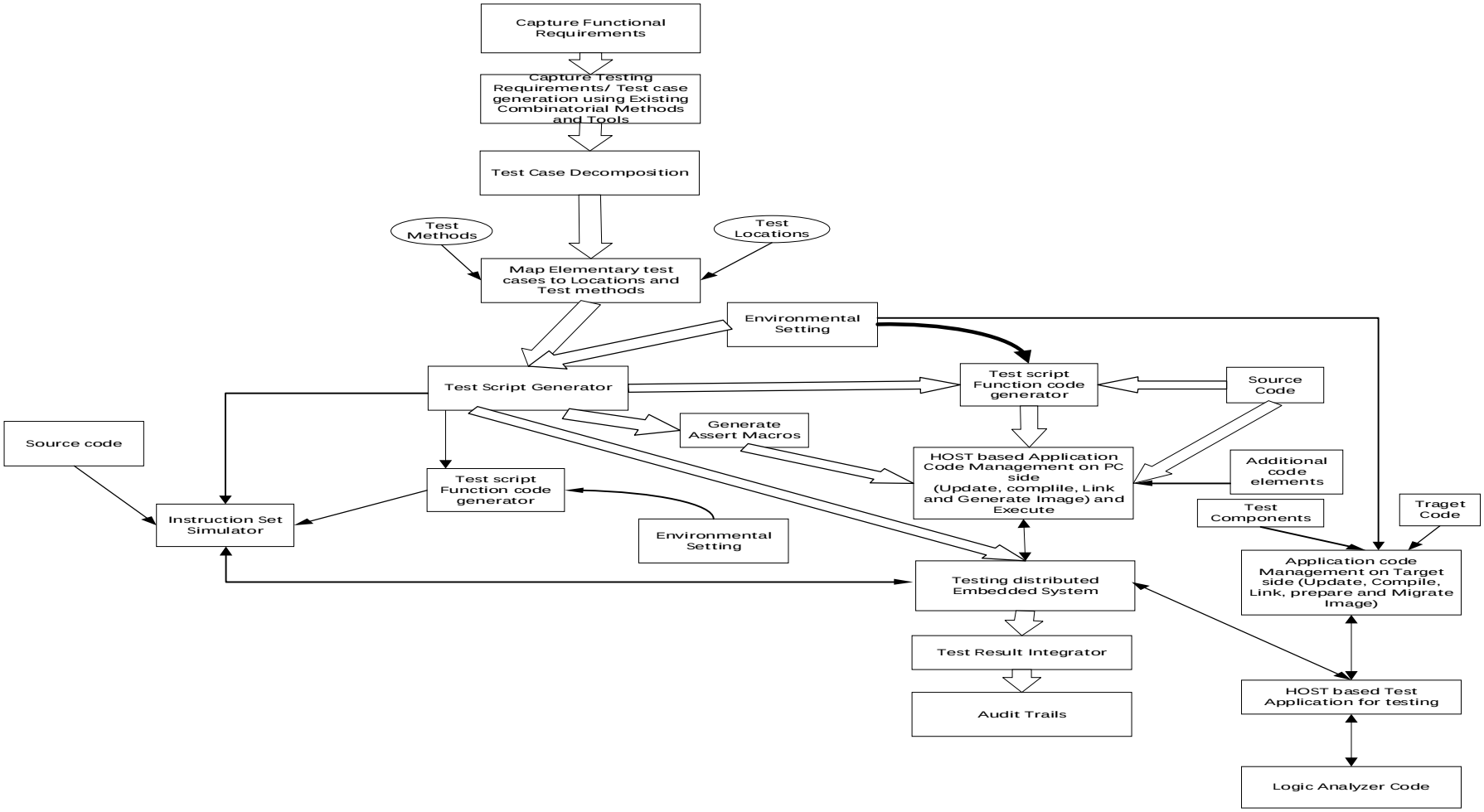

Figure 1: Framework for Testing Distributed Embedded Systems