International Journal of Electrical and Computer Engineering (IJECE)
Vol. 6, No. 1, February 2016, pp. 381~398
ISSN: 2088-8708, DOI: 10.11591/ijece.v6i1.9058

Implementing Dynamically Evolvable Communication with Embedded Systems through WEB Services

Smt. J. Sasi Bhanu[1], A. Vinaya Babu[2], P. Trimurthy[3]
[1]Department of Computer Science Engineering KL University, Vaddeswaram, Guntur District, India
[2]Department of computer Science and Engineering, JNTU Hyderabad, India
[3]Department of computer Science and Engineering, ANU Guntur, India

Article Info

Article history:

Received Sep 22, 2015
Revised Nov 11, 2015
Accepted Nov 28, 2015

Keyword:

Communication systems
Dynamic evolution
Embedded systems
Remote communication
WEB services

ABSTRACT

Embedded systems that monitor and control safety and mission critical system are communicated with by a HOST located at a remote location through Internet. Such kind of embedded systems are developed to be dynamically evolvable with respect to syntax, semantics, online testing and communication subsystems. All these systems are to be dynamically evolvable and the components needed for evolution are also to be added into the embedded system. Architectural models describe various components using which dynamically evolvable sub-systems are realised through implementation by using specific and related technologies. Implementation system describes the platform, code units and the interlacing of various processes/tasks to the elementary level of details. WEB services place an excellent platform for implementing dynamically evolvable systems due to the use of open standards. This paper presents an implementation system that is related to dynamically evolvable communication and other sub-systems using web services technologies.

Corresponding Author:

Sasi Bhanu Jammalamadaka,
Department of Computer Science and Engineering,
KL University,
Vaddeswaram, Guntur District, Andhra Pradesh, India 522502.
Email: sasibhanu@kluniversity.in

1. INTRODUCTION

1.1. Background

The overall architecture of dynamic evolution of embedded from the point of view of evolution of syntax, semantics and online testing has been shown in the Figure 1. The first layer in this model is the communication sub-system. Communication between the HOST and the TARGET is initiated from the HOST through commands strings which follow the UNIX like standard. The physical connection between the HOST and the embedded system can be achieved in many ways. Peer to peer communication between a Target and HOST can be achieved using wireless communication methods which include Wi-Fi and Bluetooth [1] [2] which has distance limitations.

Embedded systems can also be connected to a HOST using SPI and interfacing the same with a I^2C, USB, RS425, CAN or mulit port interface [3], [4], [5], [6], [7], [8], [9]. A HOST communication with the TARGET using serial port communications systems such as I^2C, USB, RS425, CAN or mulit port is limited by the distance to a maximum of 1200 Meters.

An embedded system can be connected to an Internet so as to increase its distance from the HOST. Use of Ethernet interface and use of TCP/IP or UDP based communication helps establishing communication

between the HOST and the Target. Both HOST and the TARGET can be situated at longer distances, thus meeting the primary requirement of Safety and Mission critical systems.

Communication between the HOST and the Target using the internet can be achieved through implementation of a Email extension server, WEB server or a WEB service server. WEB services sever allows several communication protocols for effecting communication with the software components which are deployed on to its own server. TCP/IP communication protocol is the most frequently used protocol. Technologies that support OPEN standards are used to develop and implement WEB services. This feature of the WEB services system makes it possible to develop dynamically evolvable communication systems.

Communication between the HOST and the Target can be effected by several protocols such as TCP/IP, HTTP, UDP, X.25 etc. The target can host several services and a different communication protocol is necessary for utilizing a WEB service. Thus the communication system must evolve dynamically based on the type of protocol needed by a service which is initiated by the HOST for execution by the TARGET.

The implementation of dynamic evolution models requires fairly more memory resources and faster processor. The Modern Microcontroller based system no doubt can run a dynamically evolvable embedded system due to the availability of more resources. However, low powered microcontroller systems may not be able to take much load. Sometimes it is worthwhile idea to move some of the processing load to the HOST and keep the embedded system light weighted. The dynamic evolution of the communication system however is needed so that communication with the WEB services can be done as per the protocol used by it. Use of web services technology addresses both the requirements of dynamic communication system and the lightweight embedded system. Implementation of dynamic evolution using the WEB services Technologies is necessary as they support OPEN standards which is the true means of evolving dynamic evolvable systems.

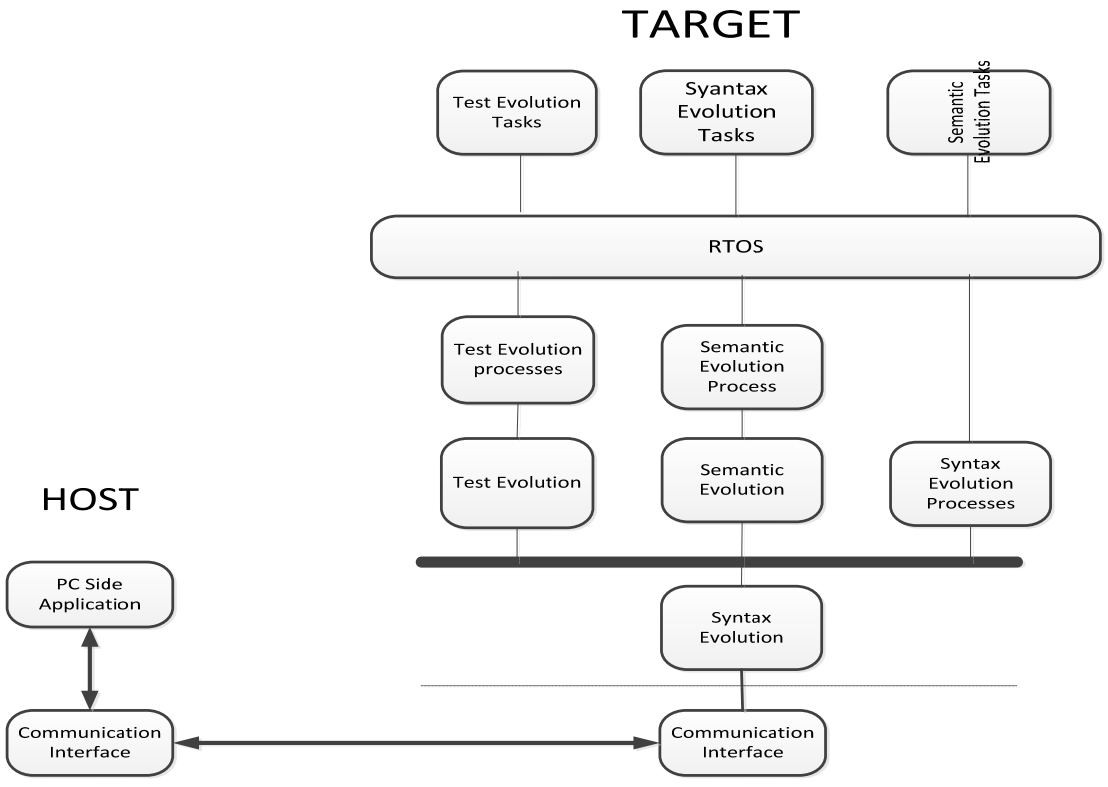

Figure 1. Overall architecture of dynamic evolution of embedded systems

There could be many architectures that support dynamic evolution using the WEB services. It is necessary to find the best of the architectures that support dual purposes of OPEN communication and dynamic evolution. Dynamic evolution is the ability to make changes to any of the software components while the system is up and running.The general web service architecture is shown in the Figure 2. A web

services architecture can be viewed in two ways which include examining the roles of each web services actor and the second is to examine the protocol stack.

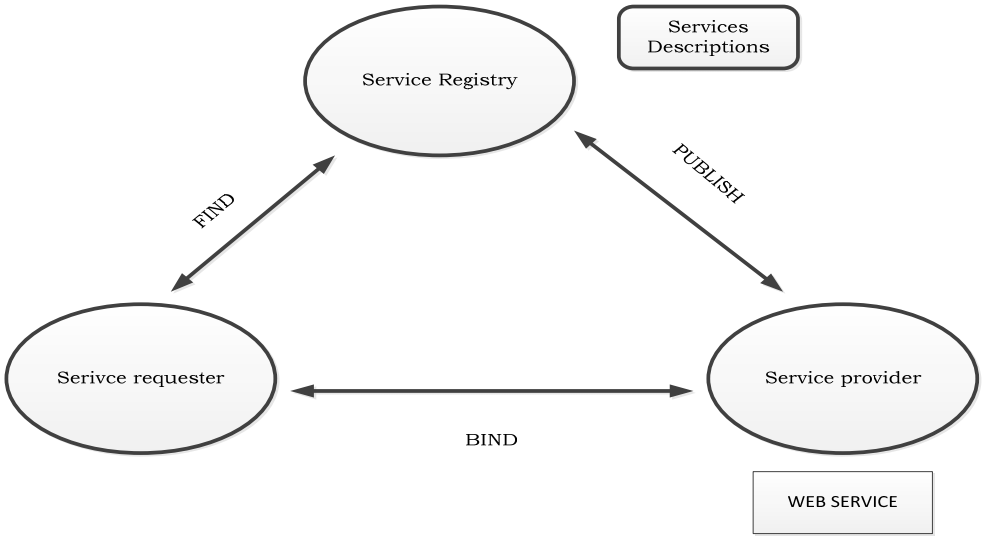

Figure 2. WEB service architecture

Three major roles are to be considered while attempting to develop a web services oriented system. The roles include Service registry, service requester and service provider. Services are deployed on a server by the service provider and the same are provided as and when the service is requested. Service requestor uses several services and builds the applications. The services are included into the application by opening a network connection and sending a message in the XML format. .Service registry is a repository of services which are registered. Every service is described using web service description language (WSDL). The registry is a centralized data repository which is hosted on a server. The services can be published into a registry by a developer by using the interface provided by the registry (UDDI). UDDI is like a centralized clearing service for developers to register and the users to enquire for the existence of such a service. Various types of protocols are used for accessing the WEB services. The protocols are layered and stacked with the protocols in one stack communicating with the protocols in another stack. The protocol stack related to WEB services are shown in Figure 3.

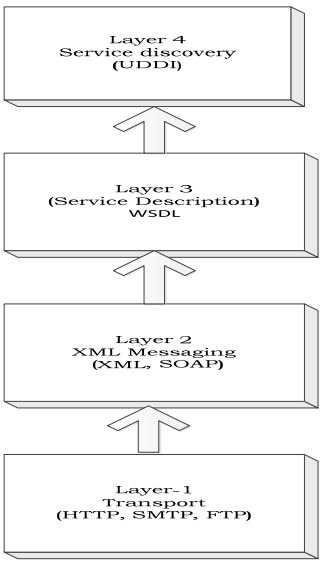

Figure 3. Web service protocol stack

Four layers exists in the stack. The fist layer Transport layer which is responsible for transmitting messages in between the service provider and service requester. In this layer Simple mail transfer protocol (SMTP), hypertext transfer protocol (HTTP), Blocks Extensible Exchange Protocol (BEEP) etc. are used for transporting the messages between the provider and the requester. The second layer is related to encoding the messages in the XML format. In this layer XML-RPC and SOAP protocols are used for encoding the messages.

WSDL (Web service description language) is used for describing the WEB services. The description of all the WEB services is provided in the third layer. This layer provided the public interface using which the services related enquiries can be initiated. UDDI (Universal Discovery and Integration) is the technology that is included into the fourth layer. The technology can be used for publishing the WEB services and also to enquire and obtain the description of the intended web services. All the details required for contacting the server at which the web services are hosted can be obtained through an enquiry to UDDI registry.

Several technologies are being used for implementing WEB services which include UDDI, XML, SOAP and WSDL.UDDI (Universal data discovery interface) has been designed as a standard for building a repository of WEB services. It has been included into the discovery layer of the protocol stack. Each of the WEB service is described by using WSDL (Web services description language) and the same are registered within the UDDI registry. WSDL follows XML for describing a WEB service. UDDI defines an API which can be used for publishing, enquiring and obtaining the response. The communication is achived through transmission of XML messages peretaining to publishing, enquiring and obtaining the response in terms of the description of the WEB services.

The UDDI specification includes API details for searching existing data and publishing new data. UDDI is implemented through building a business registry which fully implements UDDI specification. The UDDI registry captures the data from the XML messages that include general description (Business name, business description, and address), General data description (Company name, Industry, description of the products offered, Geographic codes based on the universal taxonomies) and Technical information (A pointer to an external specification and an address for invoking the WEB service).

XML has become de-facto standard for effecting communication between two computing stations. XML is an extended mark-up language that helps most of the computers to exchange data in an open manner. It is not dependent on any of the OS or programming language. Many tools are in extent today for developing an XML based message and also parsing the same. Tools to deal with XML based documents are available nearly for every operating system and for migrating into many of the languages which include java, C#, C, C++ etc.

Communication between the computing locations can be effected through RPC (Remote Procedure Calls). RPCs can be performed by using XML-RPC based protocol which is simple. Request to execute a procedure at one of computing can be encoded into an XML message which can be sent using a transport protocol such as HTTP. The results obtained out of the procedure execution is formulated into a XML message and transmitted back as a HTTP response. XML responses are embedded into the body of the HTTP response. Diverse applications can communicate using ML-RPC as the protocol as such is platform independent. For example an Application developed in JAVA can communicate with an application developed in PERL language. XML-RPC defines a set of data types that can be used for passing the data as function arguments, return values and error messages, a request related message structure containing a HTTP POST request method and the information and a response related message structure containing a HTTP GET request method and the information. A remote procedure call is made of data, request and response message which are all encoded into a XML message. The XML message can also be included as aa TCP/IP message. Parsers are required for either encoding or decoding the XML messages that are either encoded or decoded.

Information between the computers can also be exchanged by using SOAP (Simple object oriented access protocol). SOAP has been designed originally to implement remote procedure calls and now it has been modified such that variety of messaging systems can be used to deliver the messages using different transport protocols. SOAP thus can be used for a client application to uses the services deployed on a remote serer using a communication protocol implemented by a WEB service and also invoke a specific method that has been coded into the WEB service. SOAP messages can be delivered by suing variety of transport protocols such as HTTP, TCP/IP etc. SOAP specification includes three different parts that include an envelope, data encoding rules, and RPC conventions.

WEB services are defined using WSDL which uses XML Grammar. WSDL is a specification langue used for describing the WEB services. Four different aspects of the WEB services have been included into the WSDL specification that includes information about the interface which gives details about the publicly callable functions. WSDL also provides the description about different kinds of data types that can be included into different request response messages. One can also specify using WSDL, the transport protocols that can be used for communication. The binding information related to binding a Transport

protocol to a WEB service can be specified using the WSDL specification. The TCP/IP related information related to the server at which web services are deployed can also be included into WSDL specification.

The contract between the Client (Service requester) and the server (service provider) is defined by WSDL in the similar way that JAVA interface represents the contract between a java object and the client application. WSDL is platform independent and also is language independent unlike JAVA or any other language and the WSDL is used primarily for describing the WEB Services. A client can locate the existence of a WEB service using WSDL and be able to call the functions that are defined with public access. Tools that have been developed with built-in awareness of WEB services can be used to automate the process of integrating the new services with the applications. WSDL is the nucleus of the WEB services architecture as it includes the language using which the services are described and provides the specification required for automatically interfacing the WEB services into a server based application.

Communication over the internet can be achieved by using the universal datagram protocol (UDP) and Transport control protocol (TCP). TCP and UDP protocols can be used for sending and receiving the messages over the internet. Many applications uses either UDP or TCP for communicating over either a Local area network or Internet. For example a computer sends a URL to a Domain name server (DNS) and gets the TCP\IP address. A computer sends an URL to a WEB server and the WEB server sends the requested WEB page by embedding the WEB page into the TCP segment of TCP data packet. TCP and UDP can be used to transmit messages developed in any other format. Transmitting using UDP is a faster process as the protocol does not support the process of acknowledging the receipt of the message, finding out the proper order of receipt of the message and finding out proper flow of the messages. TCP/IP address identifies the address of both the sender and the receiver. UDP as such need not have to include the senders TCP/IP address as a part of the message that is meant for transmitting. TCP/IP address must be associated with a Net mark address to route the message to a specific local area network. In addition, it is necessary to attach few more addresses to the message which include gateway address, a router address when it acts as gateway and an IP address of a DNS server. All these addresses will help in proper routing of the messages and ensure that messages are received by the designated receiver. The IP addresses used for different purposes can be pre-coded within firmware or a separate DHCP (Dynamic host control program) server can be contacted to get the IP Addresses that can be used for different purposes.

TCP/IP and UDP protocols are used by many application specific protocols such as HTTP which is used for communicating with the WEB server. Every embedded board can be provided with data related to IP address, network mask, logical port number stored in its EEPROM. The Ethernet port can be binded with the TCP/IP address and suit a TCP/IP function calls can be made available as callable functions as regular functions that can be called from a ES application.

1.2. Problem Definition

Implementing WEB services server within an embedded systems leads to extensive automation and reuse. Implementation of WEB services requires huge amount of storage area processing power and porting of the technologies which are required for interacting with internet world. Embedded systems are low in resources and therefore throw a challenge to implement light-weight embedded WEB services that can cater for the user requirements. The main problem is to develop simple WEB services architecture that requires very less resources on the embedded system side which is expected to host all the services required to complete the tasks initiated by the HOST and returns the results achieved out of execution of the tasks.

We have seen in the dynamic evolution model many tasks are to be performed by the TARGET based on the command initiated from the HOST and rerun the results obtained through execution of the tasks back to the HOST. The Tasks that have been identified are related to syntax, semantics and online testing and evolving the tasks dynamically.

The commands initiated from the HOST can be enclosed into a SOAP message which is formatted in XML language and transmitted. The SOAP message is received and parsed and the command is retrieved. Based on the command received, one of the main services semantic evolutions, Syntax evolution, Evolution of online testing or passing the data received to the embedded application can be undertaken. Here the services primarily are related to one of the evolution models or the ES application itself.

An efficient WEB services oriented architecture is required that implements the dynamic evolution and pass data to ES application. It is also important that the real implementation platforms are determined and the same are to be used for development of services considering the evolution modules and the ES application.

Thus the problem is to find an efficient architecture and use the same for effective implementation which is simple to implement and that requires very few resources. One important way is to explore pushing much of the processing to the HOST and just the services are implemented on the target side. The implementation must be done using the platforms that require few resources. To the extent possible it is

necessary to write proprietary platform so that the overheads that one has to face when technologies like TOMCAT, WEBLOGIC SERVER, JBOSS etc. gets completely gets eliminated.

1.3 Related Work

[Michael Sig et al, 2004] [10] Mainly described the applicability of WEB services in distributed embedded systems environment. They have presented the size of the message to be transmitted increases extensively due to addition of many of the abstractions on top of the actual message. The problem is to minimise the overhead as much as possible. They have developed a tool called WSTOOL for generating the WEB services using minimum of overheads. The tool generates WEB services related middleware in C Language. The tool also can be used to develop WEB services that are to be deployed on the server side and the code required to be resident on the client side to effect communication with the server for want of a service.

WSTOOL can also generated the skeleton code required to expose the functionality of the existing application on the server side transparently. The tool also allows generating the client side code (stub) which can be used for requesting the services offered by the server in a transparent manner. The STUB allows for calling a remote function coded into a WEB service using SOAP protocol. The STUB provides a C interface using which a remote function of a service provider can be requested for execution a of the function is executed locally.

The code that acts as a stub on the client side picks the arguments values that needs to be passed to a function and embed the argument values into a SOAP message and the request to the server by transmitting the message that has both the name of the function and the values that must be passed to the arguments of the function. The code that is generated on the server side representing the skeleton is responsible for listening the request from the client (STUB) and invoking the function, the name of which and the arguments that must be passed to the functions are contained in the SOAP message received by the Skelton code.

[Stanislav Sliva, et al, 2004] [11] Stated that distributed processing plays a major role in applications whose parts (Procedures) are executed in local nodes and in remote nodes distributed in a network. Distributed processing applicable to embedded systems describes several possibilities of using distributed computations in an embedded environment. The major part of the distributed processing applicable to embedded systems is focused on a description of WEB services and related protocols like SOAP, XML-RPC.

The main intention of using distributed processing mechanism in designing an embedded system is to improve the possibilities of such a system to have an access to the services and resources present on the other systems in distributed environment. They address the description of several existing distributed processing mechanisms.

They presented thatRPC is a popular and easy model for implementing a communication system in the client – a server mode within a distributed environment. RFC mechanism is triggered by calling –client which sends the remote message to server to execute a certain procedure using sent arguments. The outcome of the resultant message is returned to the caller. There are many other implementations of RPC. However some of the RPC implementations uses different protocols making it difficult to communicate.

They have presented t**hat "**Qnet" is a native network distributed processing mechanism used in the real time operating system. This operating system not only aimed at powerful workstations, servers but also support embedded systems which are built using Microcontrollers. This mechanism also called as Transparent Distributed processing, is realized using passing messages. The message passing mechanism is fundamental part of Inter Process Communication (IPC) of Qnet system. They described the pre-requisites that need to be considered for building distributed processing mechanisms. They have also presented the possibility of using WEB services for building distributed embedded system was discussed.

[Kevin J. et al, 2005] [12] Have presented that SNMP (Simple network management protocol) has been used extensively for integrating WEB services related infrastructure into networking infrastructure. The SNMP has been included in to the firmware of networking devices which are basically the embedded systems used for undertaking networking. Existing applications are being enhanced with inclusion of additional functionality to achieve XML/SOAP inter-operability thereby enabling the existing applications to provide support to access the WEB services. However SNMP or similar networking protocols are installed on legacy hardware and therefore to do not adhere to the OPEN standards.

Kevin J. et al, 2005 have attempted to enhance the capability of the existing protocols so as to implement WEB services based on SNMP protocols. In way they attempted to implemented SNMP for enabling the WEB services. SNMP V2 has been extended through a scheme that enables web services at transport level through a XML encoded SOAP message which is encapsulated and bound to HTTP for transporting the message using the SNMP. They have presented two options for implementing the extensions

to SNMP which include a WEB server and JAVA library, and the other one being the use of light weight HTTP/SOAP stack. Both the options have been implemented by integrating the extensions within SNMP daemon that runs as a background job. They have aimed to achieve interoperability between both the approaches and also to maintain interoperability between the legacy systems.

Both the approaches have their own performance levels and many features and as such both support WEB services with SNMP as the transport protocol. They have presented the impact on the performance when Extended SNMP, especially standard SNMP V2 has been extended with transport related infrastructure. They have implemented the gSOAP protocol stack with Net-SNMP in the first phase and then integrated the modified SNMP into APACHE WEB server, JAVA applets, to provide a browser-based interface to Net-SNMP.

The possibility of using WEB services architecture (WSA) for implementing the distributed embedded systems has been investigated by [Nikolay Kakanakov, et al, 2005] [13]. They have explained that WSA integrates different component based technologies with the World Wide Web.

According to [R. Pallavi, et al, 2015] [14] a service is: "a software system identified by a URI, whose public interfaces and bindings are defined and described using XML. Its definition can be discovered by other software systems. These systems may then interact with the WEB services in a manner prescribed by its definition, using XML based messages conveyed by Internet protocols."

WEB services are accessed by the Applications through use of WEB related protocols and different kinds of formatting standards. HTTP is the most frequently used protocol for transporting form one end to the other. XML is universally being used for coding the data to be transported over the internet. The internet supports a structure that facilitates the program to user interaction while WSA supports program to program interaction [Kerger et al, 2001] [15].

Several standards are used for building the applications using WSA. XML is used for data representation, accessing the WEB services is implemented through SOAP, describing the WEB services is achieved through WSDL, registering and discovery of the WEB services is achieved through UDDI [R. Pallavi, et al, 2015] [14] [Kerger et al, 2001] [15].

SOAP (Simple Object Access Protocol) is a light weight protocol meant for data exchange undertaken in a distributed environment. SOAP is developed using the XML. A request to a server is sent by naming the function to be invoked on the server side and the server sends the response containing the data in XML format.

XML grammar is used for describing a web service using yet another language called WSDL (WEB Services Description Language). All the properties of a WEB service are described using WSDL The properties of a WEB service include its location, description of the functionality, the methods that can be invoked and the parameters that must be passed. WSDL is similar to CORBA, and DCOM which are all related to defining and working with the interfaces.

UDDI (Universal Description, Discovery and Integration) is an application that maintains a registry of services. UDDI provides API using which the WEB services are published, enquired and responses sent. Interaction with UDDI server is achieved through XML messages that have standard formats. The details of a business, industry, product and its related services, interfaces, communication protocols etc., are stored in the registry. Many public API sites exists today and they can be used for publishing the WEB services. Once published, the details of those services can be obtained through use of related API.

Embedded devices which include the smartphones provide flexibility in accessing the data through support of many of the communication systems. A set of smartphones can also collaborate for exchange of the data among them. Heterogeneity is the fundamental issue that must be addressed when it comes to distributed computing. Heterogeneity is due to use of different operating systems, hardware platforms, real-time behaviour and performance characteristics and the ability of the devices in a neighbourhood to communicate with in a seamless manner [Daniel Schall, et al, 2005] [16]. The interoperability between heterogeneous applications could be achieved by using the WEB services which are nothing but XML based protocols. The inter-operability between the applications can be addressed by porting WEB services into the embedded devices. Several tool kits exists that help investigating the performance of the embedded devices into which the WEB services have been ported. The main goal as on today is to implement WEB services on the mobile devices and then to estimate the performance of such devices into which the WEB services have been ported.

Heterogeneity and Interoperability among different systems can be addressed by using the WEB services and a framework involving the WEB services will be of a great help. It is necessary to consider limitations on the performance and constraints that are imposed due to the availability of limited resources while implementing the WEB services. WEB services are being sued as enabling infrastructure for achving interoperability between the heterogeneous mobile devices. The performance of the applications is the most important issue that must be considered while attempting to implement WEB services in those systems.

Several platforms exists today using which the WEB services can be implemented. Comparison of the performance of such platform is needed. One can compare the performance using the technologies JAVA/J2ME, C++ using which the WEB services can be implemented.A framework has been presented using which the performance at various layers of WEB services stack can be measured. The performance measurements take into the consideration creation of SOAP messages, processing of the messages etc. The framework provided a method to trace the flow of messages between the client and the server and computes the time spent in transmitting the messages.

Most of the embedded systems are built as standalone systems. However, these days most focus is on integrating the embedded systems with other embedded systems and even with the loaded systems [Guilherme Bertoni et al, 2006] [17]. The embedded systems are connected with other systems by using many types of communication systems. Embedded systems are being interfaced with internet so as to be able to communicate with any of the system that is part of internet. TCP/IP is predominantly used for effecting the communication among the systems that are interface with the internet. WEB services are heavily in use for supporting distributed applications that run on the internet. WEB services oriented architecture supports interconnecting different types of systems through use of TCP/IP driven networks. Many business applications communicate seamlessly when the applications are implemented by using the WEB services which are deployed on a HOST server. But no cases as such is seen which considers interconnection between the heterogeneous embedded systems using WEB services. The feasibility of such a interconnection between the heterogeneous embedded system and making the embedded systems to communicate with each other by using the WEB services has been explored. The feasibility of interconnecting the heterogeneous embedded systems and make them to communicate with each other through web services has been tried on SHIP board which is interfaced with many other systems using Internet as connecting backbone.

It has become a reality of interconnecting [Guilherme Bertoni Machado, et al, 2006] [17] and integrating various types of computing systems and platforms including the embedded systems. WEB services oriented architectures and technologies can be used to develop middleware using which independent systems can be made to communicate with each other. WEB services oriented technologies are being employed successfully for integration of different business applications. In factory system several devices/equipment exists that needs to communicate with each other and as such the communication is achieved through customised solutions. The devices and the equipment are generally heterogeneous in nature and even follow different communication systems. Such systems must be integrated using the WEB services related technologies as communicating among heterogeneous systems is quite possible using WEB services. The use of Web services has been investigated for integrating various devices that are used in the manufacturing of a smartphone. The application has been investigated from the point of re-configurability and working of the same under the re-configuration situation. The ability of the devices to re-configure under changed situations will be able to adapt themselves to the changes. Recommendations have been made to make changes to the firmware loaded into the embedded systems so that the devices can be integrated through WEB services.

The service oriented architecture (SOA) is being used off late for implementing the distributed embedded systems [Mitko Petro Shopov, et al, 2007] [18]. There is a possibility that some of the systems that are vendor specific will be replaced by the systems that use open standards and moving towards implementing distributed embedded devices which are connected on to enterprise wide networks. Many experiments have been conducted for evaluating the WEB services that are launched on to ARM based systems which run under the control of LINUX operating system. Many WEB services oriented code generation tool kits are in use and port the same to be installed within the embedded devices to make them used as WEN services oriented devices. gSOAP is one of the frequently used tool kits which is being used for generation WEB services related code that can be ported in the Embedded devices. Two WEB services (Echo and Temperature) have been developed which are deployed on to APACHE WEB server using CGI interface. The services that are deployed are tested using gSOAP and .NET WEB clients.

These days information gets generated from diversified sources **[**David E. Culler et al, 2007] [19] and integrate this same and providing the unified picture is the necessity. The data gets generated extensively through the communication media which allows the exchange and control of data. The partners which are exchanging data using the communication media must agree on the data representation, standards, formats, models (Data/Object) etc. WEB services are to be discovered and used so that all the basic issues can be addressed for being able to exchange the data by the partners using the media for communication.

There is urgent need for using different technologies for making heterogeneous embedded device connected on to a network to communicate effectively **[**Danilo J et al, 2008] [20]. It is necessary that service oriented technologies be used for making diverse and heterogeneous embedded devices to communicate effectively over the networks to which the devices are connected. XML and WEB services have been put into the use within the embedded devices. The strong interaction power of the WEB services makes it essay to

access different kinds of resources which generally heterogeneous in nature. The Middleware developed using the WEB services makes the semantics of an application more abstract leading to the hiding the complexity of the application systems.

WS4D is a common open source platform for developing and using device profile WEB services (DPWS) [Elmar Zeeb, et al, 2008] [21] that can be used in different environments. SOA is changing the view of many of the enterprise applications. SOA helps in developing the software architected that considers the modular approaches which aims at high degree of interoperability and reusability. SOA is making it possible to make small resource constrained devices to become powerful by using SOA and networking the embedded devices.

The open source platform WS4D is used with the web service tool kit gSOAP for the development of SOAP based web services using C/C++. The gSOAP tool kit has in it, the development and the run-time environment. The tool kit aims at the development of web services that run on small footprint. gSOAP has the functionality in it generate code for implementing the WEN service using its own built-in service description language which is based on C like syntax. The web services description is stored in gSOAP files that are similar to C header files. A translation module also is contained in gSOAP for translating the WSDL files into gSOAP files and vice versa. gSOAP has a code generation tool that generates code required for implementing the web services. The gSOAP generates XML schema that include C data binding which is also capable of generating stub and Skelton interface modules for describing the WEB services.

The XML schema generation process creates a map between C data types and XML schema structures. Marshalling and de-marshalling functions are included into the code generated for conversion one type of data to other. The Skelton and stub codes that are generated maps WSDL operations to C functions. The runtime of gSOAP includes the generated code and the runtime environment of gSOAP.

[RistoSerg, et al, 2008] [22] Have explained that most embedded systems are seldom used alone. Systems that communicate between each other are much more common in real world ubiquitous applications. If the embedded sensors and devices could directly work together and with other computing devices, they would add value to each other, and enable new consumer application. Present requirements of cyber-physical systems are usually too high for implementing them on single, non-networked units. Using service oriented architecture is one of the solutions to achieve interoperability and possible future scaling of the system. They have explained that a limited subset of XML WEB service protocols can be implemented in very limited environment. Surprisingly they found that limited XML WEB service implementation introduces only minimal overhead.

Using XML WEB services as the communication layer for embedded systems facilitates standardization of communication and helps increase interoperability between different types of communicating systems. According to Metcalfe's law the value of the system is larger than the sum of its components. Limited scale SOAP implementation introduces inly insignificant overhead and simplifies several ubiquitous computing scenarios. Memory constraints can often be dealt with by using offline analysis tools and fixing code and data as much as possible so as to put into the ROM memory area.

Modern engineering education requires extensive use of laboratories which are quite limited due to requirement of huge capital cost [Yin-Wei-Feng, et al, 2009] [23]. The laboratories are quite used for data collection, analysis and design of data and provide a platform experimenting. The laboratories are being used in traditional manner and always extensive operational planning is required to use for the purposes for which the laboratories are provided. Creation of virtual environment or remote access to the laboratories is also not helping to access the laboratories in the traditional manner. With the availability of broad connectivity, people started to think to use web based e-learning so that the e-learners can learn at their own convenience.

New type of Laboratories are being put in place that provides real time remote access remote to all the devices and be able to conduct experiments as per the users design specification. Embedded systems related laboratories which are quite limited in resources are being provided with remote access for conducting the experiments using actual hardware and the related tools. WEB services oriented architectures are being implemented to be able to share the laboratories by multiple users.

The use of IP (Internet protocol) is not going to be limited to accessing of the Internet and computer networks [Elmar Zeeb, et al, 2010] [24]. IP as such is going to be used for facilitating communication in between enormous heterogeneous applications, devices and systems. Many new communication architectures, technologies, protocols, concepts etc., which cater to thousands of devices and communication points are expected to be evolved. These new elements must provide extensive support for effecting interoperability across different domains. The DPWS provides the support required for cross domain access.

2. METHODS

2.1. Effective Architectural Models

Architectural models provides the basis for developing and installing the implementation system that considers the platform, technologies, programming languages, operating systems, program units and interlacing between several sub-systems.Two alternative architectures have been proposed [25] which are effective for implementing WEB services oriented, dynamically evolvable syntax, semantics, online testing and communication related sub-systems. WEB services have been found to be the way the dynamic evolution of communication sub-system can be achieved and also implement the other sub-systems and seeing to it that the entire embedded software on the TARGET (Embedded system) is made to be light-weight and dynamically evolvable.

2.2. Dynamic Evolution through Web Services Architecture – Alternative -1

This architecture is shown in the Figure 4. This is essentially a two tier architecture. Tier-1 is the HOST which is the client itself. At the client, UDDI registry is implemented, thus eliminating the requirement of another server. This also reduces the communication delays quite drastically. The client application uses the UDDI for registering the WEB services through publishing the WEB services into the registry. The UDDI is an application by itself that implements the UDDI protocol through use of SOAP/XML messaging. The client application uses UDDI interface both for publishing and enquiring the details of the WEB services. WSDL scripts are written for all the WEB services and the same are published into UDDI.

The client application parses the details of the WEB application after obtaining the WSDL scripts from UDDI. The command string that the client needs to transmit is queried from the data base or obtained through a user interface is developed into a XML script and the same is encapsulated into SOAP message and transmitted to the Embedded systems. On the embedded systems side, different types of evolution components are recognized as tasks operating under the influence of a Real time operating system. The tasks are treated as WEB services. However the service execution is undertaken through tasks that together accomplish a command initiated from the HOST as a HTTP request. From the architecture diagram it can be seen that service execution through tasks is always achieved through the syntax evolution task.

This architecture implements all the dynamic evolution models and also archives dynamic evolution of communication system. The communication is undertaken through SOAP messages into which the commands that are coded as XML messages have been encoded. Even the results are also sent as XML messages, thus requiring the parsing to obtain the actual results to be presented using the GUI implemented by the client. The architecture is excellent as long as the embedded system has as much resources as required. This is definitely a light weight web services system as the Embedded system is not burdened with the need for a WEB server or Application server which is definitely a major advantage.

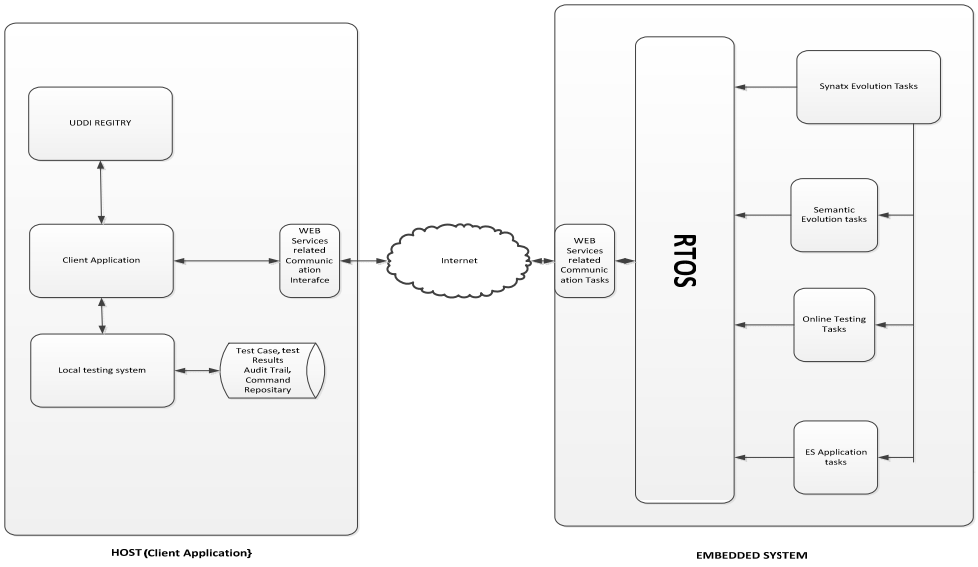

Figure 4. Dynamic Evolution of embedded system oriented - Web Services Architecture – Alternative-1

The Communication block implements the Communication related web services tasks. The Communication related WEB services task receives the message, parse the message to retrieve the command, validate the command and passes the same to the command processors which validates the command for its semantics and passes the command to it related task which is executed as a service. The service task places the output in global variables and the results are returned by the web service oriented communication services by retrieving the data from the global variables, forming into a XML script, enclose into SOAP message and transmit.

On the Clint side the Web services oriented communication application receives the SOAP messages, extracts the XML message and parse the same into data which is displayed either on the user interface or the same is used to update the database.

2.3. Dynamic Evolution using WEB Services Architecture – Alternative -2

Another WEB services oriented architecture that suits to the dynamic evolution of the embedded system that is implemented on a Low powered low storage embedded system is shown in the Figure 5.

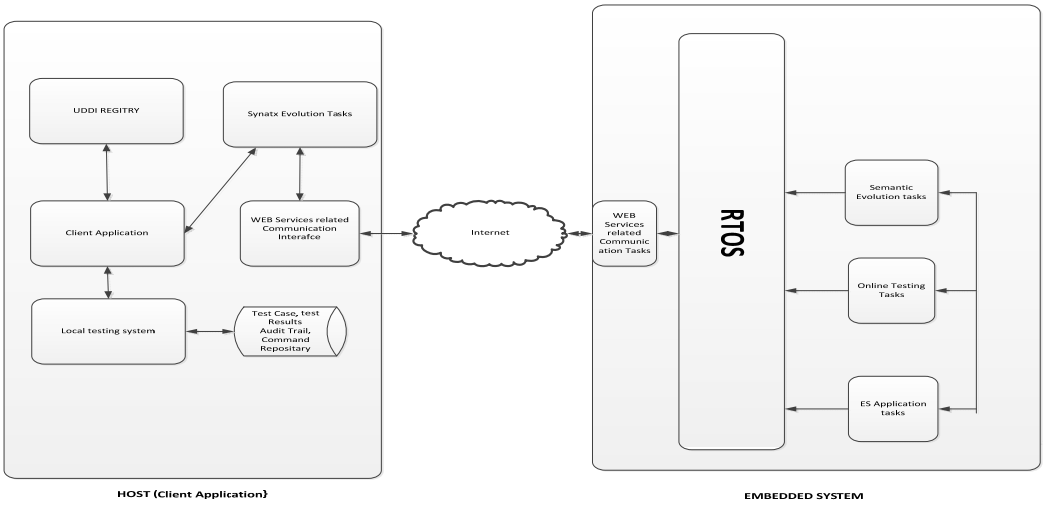

Figure 5. Dynamic Evolution of embedded system oriented - Web Services Architecture – Alter-2

In this architecture the entire block related to syntax evolution of the embedded system is shifted to the HOST. The command language evolution, syntax and sematic verification when the commands are inputted from the user screen are carried on the HOST. If the commands are directly fetched from the database, the syntax and semantic evaluation are bypassed leading to processing of the commands. This architecture implements web services as further light weight services as no web server or Application server is used for the implementation of dynamic evolution of the embedded systems. The syntax evolution module now placed in the client takes care of all the tasks/processes that are related to syntax evolution after finding the correctness of the command, hands it over to WEB service oriented communication system. The web service oriented communication system converts the command string into XML specification which in encapsulated into a SOAP message and the same is sent to the embedded server where the SOAP is messaged is received parsed and handed over to the service oriented task that runs under the influence of the RTOS.

2.4. Implementation of Alternative-2 for TMCNRS

The details of the implementation of Alternative-2 in respect of an embedded system that monitors and controls the temperatures within a Nuclear reactor system are provided in the following sections.. Following steps of implementation have to be followed when web services oriented architectures are to be used for implementation of dynamically evolvable embedded systems.

{1} Identification of WEB services and development of WSDL scripts
{2} Posting the WSDL scripts to UDDI server
{3} Identifying the commands and development of XML messages
{4} Enquiring UDDI server for a service and get the response
{5} Developing SOAP messages out of XML messages
{6} Transmitting the SOAP messages

{7} Receiving the SOAP messages, extracting the XML message, parsing the XML message and invoke the task concerned after placing the data to be processed by the Task in global variables.

2.5. Identifying and Developing the WEB Services

To start with, all the tasks that are related to syntax, semantics, online testing and the ES application that must be dynamically evolvable must be identified have to be identified.. Table 1 shows the list of tasks that must be supported for making the embedded system which is meant for monitoring and controlling the safety and mission critical system dynamically evolvable.

Table 1. Tasks for dynamic evolution of TMCNRS system

Serial Number	Evolution System	Task to be converted as WEB service
1.	Semantic Evolution	SEMEVOL
2.		INVOKEEXTTSK
3.		DELETEEXTTASK
4.		ADDRULE
5.		CODEREPOBUILD
6.		CODEGEN
7.		COPYTASKCREATE
8.		COPYTASKUPD
9.	Online Testing	TESTEVOLUTION
10.		BLDTSTTSKREL
11.		BLDTSKDLY
12.		BLDTSK
13.		INVOKEEXSTTASK
14.		DELETETESTTASK
15.	ES application	INITIALIZE
16.		RECVREF1
17.		RECVREF2
18.		AUTHENTICATE
19.		SENDTEMP1
20.		CONPUM1
21.		SENDTEMP2
22.		CONPUMP2
23.		CONBUZ
24.		COMMUNICATION
25.	Syntax Evolution	SYNTAX EVOLUTION
26.		SELF-ADAPTION
27.		CMDADD
28.		RULEADD
29.		ADDCMDPROCESSOR
30.		WRITENEWCMDPROCESSOR

All these tasks are recognized as services for which WSDL scripts have been developed. Every WSDL service must be bound to a communication protocol and the port on which the protocol must listen to. Data that must be passed to the script is defined through port tag. Each service can be binded with a transport protocol and the location where the service is deployed. The scripts related to all the services will have same structuring except for parameters to be passed, communication protocol which should be binded and the port on which the protocol must activated. The WSL script for each of the service can be generated through a program which can be passed with content specific to the WEB service. Some WSDL scripts generated are placed below

```
# SEMEVOL Script for semantic Evolution service / Taskd

<definitions>

  name="SEMEVOL"
targetNamespace="http://www.examples.com/wsdl/SetEnv.wsdl"
xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:tns="http://www.examples.com/wsdl/SetEnv.wsdl"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">

<message name=" SEMEVOLRequest">
<part name="CMDSTRING"   type="xsd:string"/>
</message>
```

```
<message name=" SEMEVOLResponse">
<part name=" SEMEVOLStat" type="xsd:string"/>
</message>

//linking messages to an operation in an abstract manner

<portType name=" SEMEVOL_PortType">

<operation name="RecvSEMEVOL ">
<input message="tns: SEMEVOLRequest"/>
<output message="tns: SEMEVOLResponse"/>
</operation>

</portType>

  //binding a port to a named binding that define concrete implementation of the abstract port type

<binding name=" SEMEVOL_Binding" type="tns: SEMEVOL_PortType">

        <soap:binding style="rpc"
                        transport="http://schemas.xmlsoap.org/soap/http"/>
        <operation name="RecvSEMEVOL">

                        <soap:operationsoapAction="SEMEVOL"/>

                        <input>

        <soap:body
                                                encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
           namespace="urn:examples: SEMEVOL service"  use="encoded"/>

                </input>

                <output>

        <soap:body

                encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
           namespace="urn:examples: SEMEVOLservice"
           use="encoded"/>

</output>

            </operation>

</binding>

<service name=" SEMEVOLService">

        <documentation>WSDL File for SEMEVOLService</documentation>

                <port

                        binding="tns: SEMEVOL_Binding" name="SEMEVOL_Port">
                <soap:address location="http://192.168.32.1/SEMEVOL/">

        </port>

</service>
</definitions>
```

The word “SEMEVOL” in the above script can be replaced with another services name / task name to get the script for other tasks. For example the Script for the Task RECNREF1 can be generated and from the script above by adding REF1 parameter to the script while the service location protocol for communication and the port which should be used for effecting the communication being the same, The command that must be used by a task for processing the same on the receiving side can also be sent as a parameter to the script.

2.6. Implementing Client Side Application

An application has to be written at the client that is capable of publishing WEB services, enquiring for WEB service and receiving results from the EMBEDDED systems which serves as WEB services server.

The client application should be able to communicate with either the Local server or a UDDI server that has been hosted on the net SEMNET.COM.

A UDDI registry is maintained by a UDDI software which is available as freeware on the WEB that can be downloaded and installed on the client as per the architecture. In addition an alternative exists for publishing a request to be initiated from the client to a UDDI server that has been located on the WEB.

A publishing request has to be initiated from the client application. The client application that post the WEB services, enquires for a particular WEB service and receives the response. A UDDI server that has been posted on the SYSNET.COM is used for the client application. An XML parser that has been posted on to Sun Micro system site is used for development of XML script through which WSDL scripts can be hosted.

2.7. Communicating with UDDI Registry from Client

The client application must enquire about the availability of a WEB services and the UDDI registry must respond. The client application must bind with a protocol for being able to communicate with UDDI registry For communicating with the UDDI registry, messages are to be constructed in XML language which can be enclosed into a SOAP message.

```
# Binding with Communication protocol for communicating with UDDI Registry
<uddi:find_bindingsserviceKey="uuid: " generic="2.0">
        <findQualifiers>

                <findQualifier>
                        SEMEVOLBinding
                </findQualifier>
        </findQualifiers>
</uddi:find_binding>

# Enquiring for the WEB services
<uddi:find_service generic="2.0">
                <findQualifiers>
                        <findQualifier>
                                caseSensitiveMatch
                        </findQualifiers>
                </findQualifiers>
                <uddi:name>
                        SEMEVOLService
                </uddi:name>
<uddi:find_service>

# Getting response from UDDI registry
bindingDetails generic="2.0" operator="SYSTINET" XMLNS="Urn:uddi-rg:api_v2">

<bindingTemplate
bindingKey="uuid:acd5sdf-1235-6a3f-c34a-bf798dab124a"
serviceKey="uuid:1245sdef-af35-6a3f-c34a-bf798dab65a">

        <accessPointURLType ="http">
                http://www.semnet.com/       SEMEVOLservices_entry/
        </accessPoint>
        <tModelInstanceDetails>
<tModelInstanceInfotModelKey="uuid:acd5sdef-1235-6a3f-c34a-bf798dab124b">
        <description>
            provides SOAP interfac.described by SEMEVOL_WSDL.wsdl
        </description>
        </tModelInstanceInfo>
</tModelInstanceDetails>
</bindingTemplate>
</bindingDetails>
```

2.8. Communicating with Embedded System that Hosts the WEB Services

Messages are transmitted from the client to the ES WEB services server for commencing the Semantic Evolution Task and also contain the status of commencement of the services. The messages are to be formulated into XML script which are encapsulated into SOAP messages. The following are the XML

scripts for communicating with Embedded WEB services server and obtaining the response that the service has been commenced.

 # Sending request for commencing SEMEVOL Task from client

```
<?xml version="1.0"?>
<soap:Envelope
        xmlns:soap="http://localhost:8080/webservice/soap-request"
        soap:encodingStyle="http://localhost:8080/webservice/soap-request">
        <soap:Bodyxmlns:m="http://localhost:8080/webservice/soap-request">
                <m:GetSEMEVOLValue>

                        <m:Actuate>Get  SEMEVOLStatus</m:Actuate>
                </m:GetSEMEVOLValue>
        </soap:Body>
</soap:Envelope>
```

 # Get SEMEVOL commencement status from ES web server

```
<?xml version="1.0"?>
<soap:Envelope
        xmlns:soap="http://localhost:8080/webservice/soap-response"
        soap:encodingStyle="http://localhost:8080/webservice/soap-response"
        <soap:Bodyxmlns:m="http://localhost:8080/webservice/soap-response">
                <m:GetSEMEVOLStatus>
                <m:SEMEVOLStatus>SEMEVOLSTATUS</m: SEMEVOLStatus>
                </m: GetSEMEVOLStatus>
        </soap:Body>
</soap:Envelope>
```

2.9. Implementing ES System Side WEB Services Application

The WEB services communication application will be running on the ES server side for receiving and sending the messages to the client. Code has been developed to implement WEB services component on the ES server side. When a message is received, the command contained in the message is parsed and the Task that is related to the command is executed by way of invoking an event related to the task which is waiting for the event to be triggered. If the message has the name of the task, then an event is set by signing on to the event through a RTOS function. The request response messages are pre-coded in the XML Language and the same are used to form into a SOAP message using which the messages are either received or transmitted.

3. EXPERIMENTATION AND RESULTS

The dynamic evolution model has been tested by feeding different inputs from the client by way of sending message to start a WEB service / Task and the result obtained out of execution of the task is sent back to the client through an SOAP message. The results obtained are shown in the **Table 1, Table 2 and Table 3.** It has been observed from the test results that the system behaved exactly they have behaved as per the expected execution.

4. CONCLUSIONS

Web services based architectures can be used for implementing syntax, semantic, online testing and communication related dynamically evolvable sub-systems. WEB services uses OPEN standards and therefore gives us the perfect platform for implementing dynamically evolvable systems. On the embedded system / embedded server side, the code has been made light weight because the uses of WEB server or WEB services server has been avoided. The tasks are used as services and the services are invoked as and when required through invocation of the related events by calling the functions supported by a RTOS. Timing and memory management supported by a real time operating system have been used for achieving the dynamic evolution.

The code on the embedded system side has been made further light weight by shifting UDDI registry code and Syntax evolution sub-system to the client side. The communication between the HOST and the TARGET has been made dynamically evolvable by using the binding scripts to various WEB services

and enquiry/ response scripts. Web services based dynamic evolution thus has led to implementation of total dynamic evolution within the embedded systems.

REFERENCES

[1] Sastry JKR, Venkataram N, Srinivasa Ravi K, Pradeep G, Reddy LSS, "On Dynamic Configurability and Adaptability of Intelligent Tags with Handheld Mobile Devices", *International Journal of Electronics, systems and circuits (IIJECS)*, Vol. 1, Iss. 2, pp. 114-123, 2012-5.

[2] Sastry JKR, Venkataram N, Srinivasa Ravi K, Pradeep G, "Software Architecture for Implementing Dynamic Configurability and Adaptability of Intelligent Tags with handheld Mobile Devices", *Research Journal of Computer Systems Engineering – RJCSE*, Vol. 3, Iss. 2, pp. 393-398, 2012-6.

[3] SasiBhanu J, Sastry JKR, J Viswanath Ganesh, "I^2C based Networking for Implementing Heterogeneous Microcontroller based Distributed Embedded Systems", *Indian Journal of Science and Technology*, Vol. 8, Iss. 15, pp. 1-10, 2015-1

[4] Sasi Bhanu J, Sastry JKR, Sai Kumar Reddy, "Networking Heterogeneous Microcontroller based Systems through Universal serial bus", *International Journal of Electrical and computer Engineering*, 2015-2

[5] Sasi Bhanu J, Sastry JKR, Vijaya Lakshmi Machineni, "Optimizing Communication between heterogeneous distributed Embedded Systems using CAN protocol", *ARPN Journal of engineering and applied sciences*, 2015-3

[6] Sasi Bhanu J, Sastry JKR, Suresh A, "Building Heterogeneous Distributed Embedded Systems through RS485 Communication Protocol", *ARPN Journal of engineering and applied sciences*, 2015-4

[7] Sasi Bhanu J, Sastry JKR, Mounica, "On Testing Distributed Embedded Systems through Scaffolding", *Journal of Embedded systems* - Inder Science, 2015-5

[8] Sastry JKR, Neeraja N , Naga Teja K, Devi Kavya Priya M, Vineela D, Immanuel K, "An Approach towards Development of Communication Standard around CAN Protocol Suite for Networking Embedded Systems", , *International Journal of Advances in Science and Technology*, Vol. 4, Iss. 2, pp. 36-42, 2012-1

[9] Sastry JKR, Neeraja N, Pushpa A.G, Satya Prakash G , Tejaswi G, Muni Kumari T, "An Open Specification for Fire wire Standard to Extend its suitability to diversified Applications", *International Transactions on Electrical, Electronics and Communication Engineering*, Vol. 2, Iss. 2, pp. 1-8, 2012-2

[10] Michael Sig Birkmose, Lars Haugaard Kristensen. Producing Efficient Web Services for Distributed Embedded Systems. Thesis - AALBORG UNIVERSITY, 2004

[11] Stanislav Sliva, Vilem Srovnal. Distributed Processing Applicable To Embedded Systems. *WSEAS International Conference on Automatic Control, Modeling and Simulation*, 2005.

[12] Kevin J. Ma, Radim Bartoˇ. Performance Impact of WEB Service Migration in Embedded Environment, *IEEE*, 2005.

[13] Nikolay Kakanakov, Grisha Spasov, Adaptation Of WEB Services Architecture in Distributed Embedded Systems. *International Conference on Computer Systems and Technologies*, 2005.

[14] R. Pallavi, C. Veeranna. Implementation of TCP/IP Ethernet WEB services On ARM7LPC2148 for Embedded Systems. *IJARCSSE*. Volume 5, 2015.

[15] Kreger H, Austin, D., A. Barbir, C. Ferris, S. Garg. Austin, D., A. Barbir, C. Ferris, S. Garg. Web Services Conceptual Architecture (WSCA 1.0)", *IBM Software Group*, May 2001, www.redbooks.ibm.com

[16] Daniel Schall, Marco Aiello, Schahram Dustdar. WEB Services on Embedded Devices. J.*WEB INFOR. SYST.*, Volume 1, 2005.

[17] Guilherme Bertoni Machado, Frank Siqueira, Roninson Mittmann, Carlos Augusto Vieira. Integration of Embedded Devices through Web Services: Requirements, Challenges And Early Results, 2006.

[18] Mitko Petro Shopov, Hristo Matevmatev, Grisha Valentino Spaso. *Evaluation of WEB services Implementation for ARM Based Embedded Systems*. Proc of ELECTRONICS, volume 7, 2007/9.

[19] David E. Culler, Gilman Tolle. Embedded WEB services: Making Sense Out Of Diverse Sensors. Arch Rock Corp. 2007.

[20] Danilo J. Oklobdzija, Bransislav T. Jevtovic. Using XML WEB services as Platform for Remote Access and Control of Embedded Systems. Volume 28, 2008.

[21] Elmar Zeeb, Andre Pohl, Ingo Luck. WS4D: SOA-Toolkits Making Embedded Systems Ready for WEB services, 2008.

[22] Risto Serg, Johanner Helander. Using XML WEB services For Embedded Systems Interoperability World's Smallest WEB 2.0 Server Demo, 2008.

[23] Yin-Wei-Feng, Sun Rong-Gao, Wan Zhong. Distributed Remote Laboratory Using WEB services For Embedded Systems. CCISST, 2009.

[24] Elmar Zeeb, Guido Moritz, Dirk Timmermann. WS4D: Toolkits For Networked Embedded Systems Based on the Devices Profile for WEB services, 2010.

[25] Sasi Bhanu J, Vinaya Babu A, Trimurthy P, "Web Services oriented architecture for dynamic evolution of communication with Embedded systems", *ARPN Journal of Applied Engineering Sciences, Paper accepted for publication*.

Table 1. Experimental results – Symantec Evolution through WEB services

	Test Case				Test Results	
Experiment Number	Command Sent	Command Argument-1	Command Argument-2	Command Argument-3	Argu-1	Argu-2
1.	RULEADD	REF1 (Name of the Command)	If Temp1>RefTemp1 + 2	Rule is added to the command	Ref1 =35	Temp1 = 32
2.	CMNDADD	TEMP12 (Command)	SEMANTICEVL (Processor name)	ADDCMND(Task name)	CommndTEMP12 is added	-
3.	ADDCMNDPROC	EMAIL (Process to add Email Extension processor	-	-	EMAL Extension processor is added	#93186 (Address Location at which the email processor is added
4.	RULEADD	REF2	If Temp2 >RefTemp2 + 3	-	Ref2 =35	Temp2 = 32
5.	CMNDADD	TEMP123 (Command)	SEMANTICEVL (Processor name)	ADDCMND(Task name)	CommndTEMP123 is added	-
6.	ADDCMNDPROC	WEBSER (Process to add Web server for effecting communication between the HOST and the ES system	-	-	WEBSER processor is added	#13170 (Address Location at which the email processor is added

Table 2. Experimental results – Semantic Evolution of ES software through WEB services

	Test Case					Test Results	
Experiment Number	Command Sent	Command Argu-1	Command Argu-2	Command Argu-3	Command Argu-4	Argu-1	Arg-2
1.	INVKEXSTTASK	COMPTMP1TMP21				COMPTMP1TMP21 Task is invoked	
2.	DELEXSTTASK	COMPREF1				COMPRFE1 Deleted	-
3.	INVOKERULE	(Temp1 – Temp2} > 3	-	-		COMPTMP1TMP2 Activated	
4.	CREATESTDFUN	LCD-BUSSY-CHK()	Code String	Variable String		LCD-BUSSY-CHK() created	
5.	CREATENONSTDFUN	I2c_start-temp()	Code String	Variable String		I2c_start-temp() created	
6.	CREAYEHDWFUN	readkey()	Code String	Variable String	Function Calling Sequence string	readkey( ) created with built-in function calling sequence	
7.	CREATESWCLASS	TEMP3SWCLASS1	SW Member Function Key String	Task Function Key String		TEMP3SWCLASS1 created	
8.	CREATEHWCLASS	TEMP3HWCLASS1	HW Member Function Key String			TEMP3HWCLASS1 created	
9.	CREATESPCLASS	TEMP3SPCLASS1	Supporting Function key string			TEMP3SPCLASS1 created	
10.	CREATEENTRYSTATE	Entry String				Entry into state Transition Table created	
11.	GENCODE	READKEY				READKEY TASK Created	
12.	COPYTSKUPD	TEMP1UPD	TEMP1	Code string		Update Task created	

Table 3. Experimental results for online testing through WEB services

HOST Imitation Number	Command	Argument-1 Test Code	Argument-2 Task Code	Argument-3	Argument-4	Argument-5 Output Variable	Output Result
1.	TESTCASE	RESPONSE	TEMP-1			T1RES	12
2.	TESTCASE	RESPONSE	TEMP-2			T2RES	11
3.	TESTCASE	GRADTIME	TEMP-1			T1GRAD	2
4.	TESTCASE	GRADTIME	TEMP-2			T2GRAD	2
5.	TESTCASE	THRUPUT	TEMP1			THTEMP1	10
6.	TESTCASE	THRUPUT	TEMP1			THTEMP2	9
7.	TESTCASE	PUMP1CNT	PMPCONTR			PUMP1-STA	ON