MILEYE - Secure Military Information System

Prepared by

Capt Zeeshan Malik

NC Zulqarnain Butt

NC Waqas Bhatti

Submitted to the Faculty of Computer Software Engineering

National University of Sciences and Technology, Islamabad in partial fulfillment for the requirements of a B.E Degree in Computer Software Engineering

June 2013

CERTIFICATE

Certified that the contents and form of project report entitled **"MILEYE – Secure Military Information System"** submitted by 1) Zulqarnain Butt, 2) Capt. Zeeshan Malik, and 3) Waqas Bhatti have been found satisfactory for the requirement of the degree.

Supervisor: _____

Lt. Col Dr. Adnan Rashdi

ABSTRACT

With the progressing IT revolution throughout the world, being nuclear power and 15th largest population of internet users, Pak Army is not so much advance in use of IT related equipment like other nuclear powers. Pak Army still uses old methods of command and navigation.

This thesis proposes architecture of a system responsible for Real-time situational awareness in mission critical military operations. System will not only replace obsolete manual situation reporting process but also bring the Tactical Picture to the handheld tablet which can be supported by Robust Planning Tools, Decision Aids and Display Capabilities. System will be capable of displaying the COP from tactical data received from other workstations. Whereas authenticated user can view and edit the COP, apply overlays, display imagery, send and receive tactical messages and gain overall battlefield situational awareness.

DECLARATION

No portion of the work presented in this dissertation has been submitted in support of another award or qualification either at this institution or elsewhere

Dedication

In the name of Allah, the Most Merciful, the Most Beneficent.

To our Parents, Teachers and Heroes of Pak Army who have sacrificed their lives for our country.

Acknowledgement

We would like to thank Allah Almighty for His incessant blessings which have been bestowed upon us. We are also thankful to our families for their continuous moral support which makes us what we are.

We are extremely grateful to our project supervisor Lt. Col Dr. Adnan Rashdi from MCS who in addition to providing valuable technical help and guidance also provided us moral support and encouraged us throughout the development of the project. We are highly thankful to all of our teachers and staff of MCS who supported and guided us throughout our course and research work. Their knowledge, guidance and training enabled us to carry out this research work. Special thanks to Sir Umar Mahmud who gave us critical and valuable advises.

Finally we are grateful to the faculty of Computer Science Department of the Military College of Signals, NUST.

In the end we would like to acknowledge the support provided by all our friends, colleagues and a long list of well-wishers whose prayers and faith in us propelled us towards our goal.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

1.1 Introduction

MILEYE is a Secure Military Information System in response to criticality of real time situation awareness in military operations.

This document is to present a detailed description of requirements and specifications of MILEYE. It will explain the functional features of the system along with interface details, design constraints and related considerations such as performance characteristics.

1.2 Background

With the progressing IT revolution everything is being digitized such as commerce and banking. IT is also affecting the military tactics supporting it by supplying modern day gadgets that work efficiently. But Pakistan being Nuclear Power still lag behind in adaptation to modern day technologies. Problems have been noticed as of Pak Army's command and navigation system. We still use obsolete paper map for tactical planning and navigation. The maps are not only obsolete but are also faulty. Also all of this is done manually with no technical support i.e. decision tools etc.

All of these are very sensitive issues and needed to be taken care of. At the same time, making everything computerize is not a practical approach. Instead of giving complete control to the machines, the specific task should be assigned to the machines. MILEYE will provide a way of enhancing and improving Pak Army's capabilities in order to make the invincible.

1.3 Problem Addressed

The problems that are being focused by the project are:

a) Replacement of obsolete and faulty paper maps with up to date and easily maintainable digital maps.

b) Revolutionize Pak Army's command and navigation system with the modern day gadgets.
c) Replacing the manual situation awareness process with our system.

The system shall focus on the following aspects:

a) SitRep Service
b) COP Service

1.4 Objectives

The objectives of our project include:

a) To learn Embedded System Development/Interfacing on WinCE OS
b) To understand the configuration and architecture of SDR.
c) To learn different techniques for SDR communication.
d) Put Pak Military in front row with modern technology gadgets to lead the world.
e) Make real-time situation awareness efficient, secure and maintainable.
f) Providing tool based help with decision making and tactical planning.
g) Allow user to view and edit the COP, apply overlays, display imagery, send and receive tactical messages and gain overall battlefield situational awareness.

1.5 Deliverables

Deliverables of the project are:

a) Embedded Client
b) .Net Server
c) Client/server Application over SDR network
d) Documentation/User Manual

1.6 Technological Requirement

Hardware:

a) Handheld tablet (Beagleboard-xM)
b) Garmin 18x PC GPS device
c) System with Core 2 Duo or above processor, 1GB RAM and 20GB Hard Disk
d) 2 × SDR Radio

Software:

a) Visual Studio 2oo5 & 2010
b) Gmap.Net API
c) Windows XP operating system
d) WinCE 6.0 BSP
e) WinCE 6.0 OS image
f) Integrated database server (SQL Server 2008) on server side.

Chapter 2: Literature Review

The project MILEYE – Secure Military Information System is very new idea in the area of command and navigational history of Pak Army.

Nearly no considerable work is done in this domain. Pak military still lag very much behind in technology in comparison to other nuclear countries. Pak military still uses the obsolete methods of situation reporting and battle field tactics planning, which is manually designing overlays on paper maps which are also obsolete and erroneous.
The system in use is inefficient, time consuming and highly insecure. Maps used to plan tactics are also faulty. No way to change or correct the plan once drawn, to cater changes we have to draw new plan.

This system is designed to be a replacement for old obsolete system for situation reporting and tactical planning using obsolete paper maps. This system will provide real-time situation awareness and take it over to handheld tablet and will use updated most recent digital maps supported by Robust Planning Tools, Decision Aids and Display Capabilities thus making it more efficient and secure as encoding of data and encryption over communication will be there to make system secure. In this regard, old systems can be a disaster waiting to happen, often with no recovery plan in place!

Chapter 3: System Requirements Specification

System can be divided into three Modules:

a) SitRep Service
b) COP Service
c) Messaging and Planning Service

3.1 System Feature 1: SitRep Service

3.1.1 Description and Priority

Real-Time situation awareness is critical to military operations done through SitRep, where the client send his whereabouts and other tactical parameters to inform the upper command tier about his current situation, currently done manually. This module will make this process efficient in terms of time and resources.

Priority: Priority is **High** for this particular feature as output sent by this module is used to formulate COP on receiving end.

3.1.2 Stimulus/Response Sequences

Stimulus: user on client side plugs in GPS device.
Response: system retrieves GPS coordinates from GPS data.
Stimulus: user on client side first encodes then load GPS coordinates along with station ID in situation awareness software.
Response: system makes packets of the data to send.
Stimulus: user on client side plugs in data cable to the serial port of radio.
Response: Radio starts transmitting data packets.

3.1.3 Functional Requirements

3.1.3.2: Radio should be working in full capacity and configured to the setting required for field communication i-e. Frequency, Bandwidth, Communication mode, Communication type.

3.1.3.3: Both tablet and radio must be able to communicate on serial port for asynchronous communication.

3.1.3.4: GPS device must be working and providing GPS data on plug in.

3.1.3.5: system must be able to encode the GPS coordinates before sending.

3.1.3.6: situation awareness software must accept the encoded GPS coordinates along with ID of the station and make packets of the data to send.

3.1.3.7: All systems must have same settings to communicate on network successfully.

3.1.3.8: In error situation like GPS device incompatibility or serial port malfunction system must generate alert.

3.2 System Feature 2: COP Service

3.2.1 Description and Priority:

From the data received from different field teams the command officers try to formulate a tactical picture or a field map by plotting the data received on paper maps and plan activities through overlays. All this is done manually and take huge amount of time and most of the maps are faulty and obsolete and require intense care and are difficult to maintain. This module will automate this process and make it efficient and maintainable and reduce a considerable amount of time required to plan tactical strategies which can be supported by tools. This will show us a tactical picture presentation of data received from field stations.

Priority: Priority is **High** for this particular feature as it provides a tactical picture of data received from other end to provide command tier real-time situation awareness about tactical teams in field.

3.2.2 Stimulus/Response Sequences

Stimulus: Radio receives data packets from field station and buffer them until stream complete and then send it to system.

Response: System retrieves GPS coordinates and station ID from data then decodes them and plots them to corresponding area on the map.

Stimulus: user on server side selects COP view.

Response: after authentication system will plot data from all field stations with ID and link station details from database to view a complete operational picture.

Stimulus: user on server side select particular area of COP to view or edit.

Response: after authentication system will allow user to see or edit the details of selected area in COP.

Stimulus: user on server side select particular object to see details.

Response: After authentication system will view the details retrieved from database.

3.2.3 Functional Requirements

3.2.3.1: System should be in working condition with full functionality of windows XP OS and database containing digital maps, personnel profiles and details of administrative set up .

3.2.3.2: Radio should be working in full capacity and configured to the setting required for field communication i-e. Frequency, bandwidth, communication mode, communication type.

3.2.3.3: Both tablet and radio must be able to communicate on serial port for asynchronous communication.

3.2.3.4: system must be able to decode and plot the received data on digital map along with information from database.

3.2.3.5: User on server side must be able to view COP.

3.2.3.6: User on server side must be able to view and edit particular area in COP.

3.2.3.7: User on server side must be able to view administrative detail of particular area in COP.

3.2.3.8: In error situation like Database connectivity or serial port malfunction, system must generate alert and in case to prevent invalid access system must authenticate user for particular operation like editing the map.

3.3 System Feature 3: Messaging and Planning

3.3.1 Description and Priority

Along with real-Time situation awareness system will also provide critical messaging service and support for robust planning tools that can help in creating overlays and planning tactical routes

Priority: Priority is **Medium** for this particular feature as it is an add-on providing additional support for system to work efficiently not a basic system feature.

3.3.2 Stimulus/Response Sequences

Stimulus: user from both sides can select messaging interface to communicate.

Response: System takes textual input from user and sends it over to receiving end.

Stimulus: user on server side selects planning interface to simulate overlay or tactical route.

Response: after authentication system will take input parameters and simulate the request of overlay or tactical route.

3.3.3 Functional Requirements:

3.3.3.1: system should be in working condition with full functionality of windows XP OS and database containing digital maps, personnel profiles and details of administrative set up .

3.3.3.2: Radio should be working in full capacity and configured to the setting required for field communication i-e. Frequency, bandwidth, communication mode, communication type.

3.3.3.3: Both tablet and radio must be able to communicate on serial port for asynchronous communication.

3.3.3.4: system must support robust planning tools.

3.3.3.5: User on both sides must be able to use radio communication module.

3.3.3.6: User on server side must be able to use planning tools.

3.3.3.7: In error situation like planning tools incompatibility, communication module or serial port malfunction, system must generate alert and in case to prevent invalid access system must authenticate user for particular operations.

Chapter 4: System Design Specifications

4.1 Architectural Style

System uses Client/Server architecture model. On the server end, COP is formed with the help of location of units from SitRep received through each workstation along with its ID. On the client side unit location is sent and critical messages exchange is also incorporated. Radio is used for communication purposes.

General system architecture would look like as below:

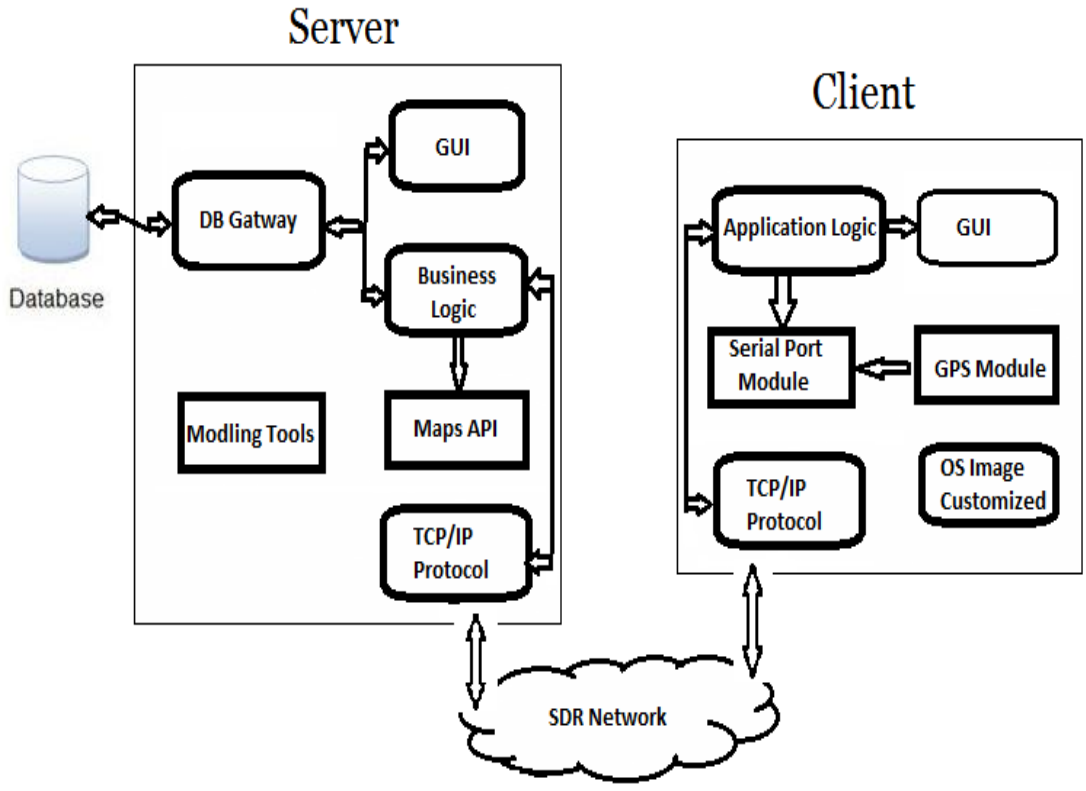

Figure 4.1 Architectural Style

We have used Client/Server architectural style because:

a. It is suitable for systems that are of distributed nature.
b. The structure of the system involves a separate client and server system, and a connecting network.
c. Each client sends updates to the server, and the server act accordingly.
d. The server typically authorizes the user and then carries out the processing required to generate the result.

The main benefits of the client/server architectural style are:

a. Higher security. All data is stored on the server, which generally have offers greater control of security than clients do.
b. Centralized data access. Because data is stored only on the server, access and updates to the data are far easier to administer than in other architectural styles.

4.2 Detailed Design

4.2.1 Class Diagram

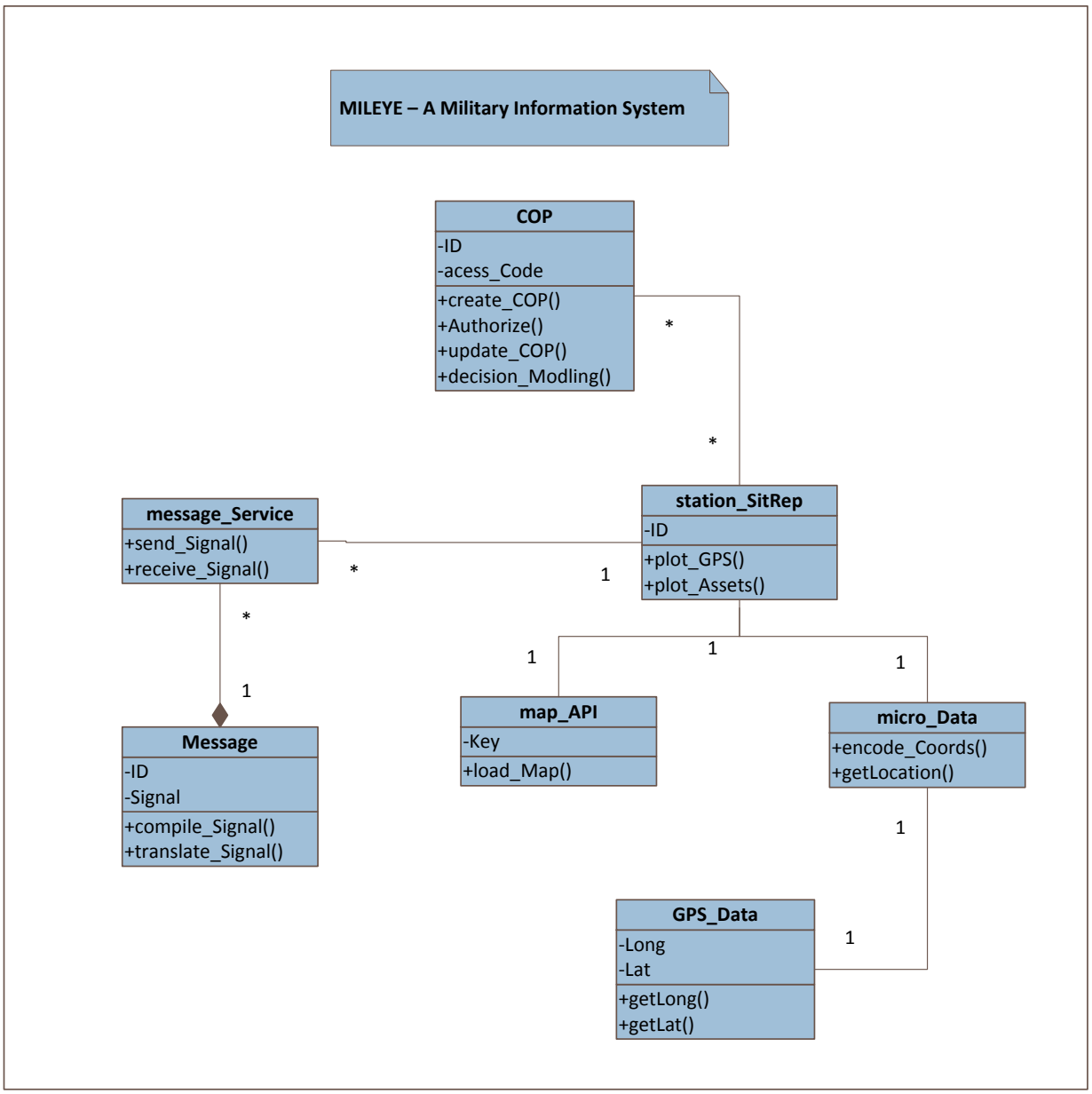

Figure 0.2 Class Case Diagram

Description of Class Diagram:

This class diagram shows classes responsible for system proper functioning.

GPS_Data: This class uses the longitude and latitude as a data with the get function.

Micro_Data: This class uses encode and get_location function to encode GPS data and convert it into coded location.

Map_API: This class uses the Key as data with the load_map function.

Message_Service: This class uses the send and receive Signal functions to exchange signals.

Message: This class uses the station ID and signal as data with the compile and translate signal.

COP: This class uses the basics access data like user name and password with the access function to view, update COP and Decision Modeling functions.

Station_SitRep: This class uses the location and station ID to plot unit's area map on COP.

4.2.2 ER Diagram

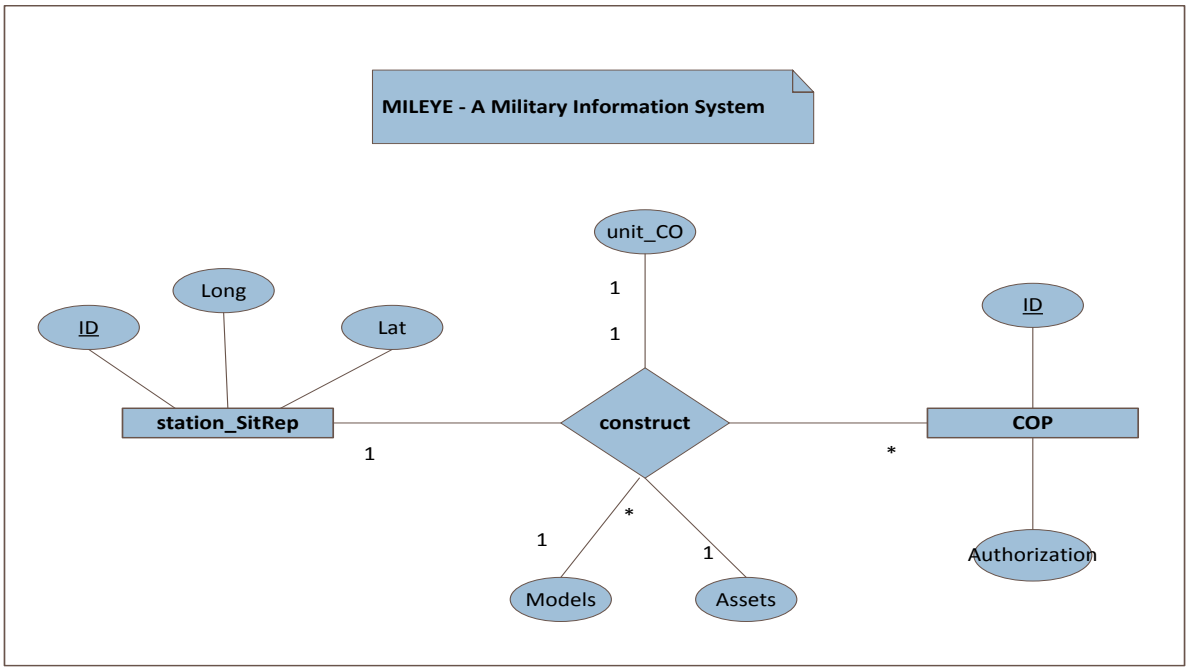

Figure 0.3 ERD Diagram

Description:

Station_SitRep and COP are the two tables of the basic Database. Station_SitRep table uses the StationID, longitude and latitude as columns while COP uses the User name, password and Station Assets information.

4.3 UML Diagrams

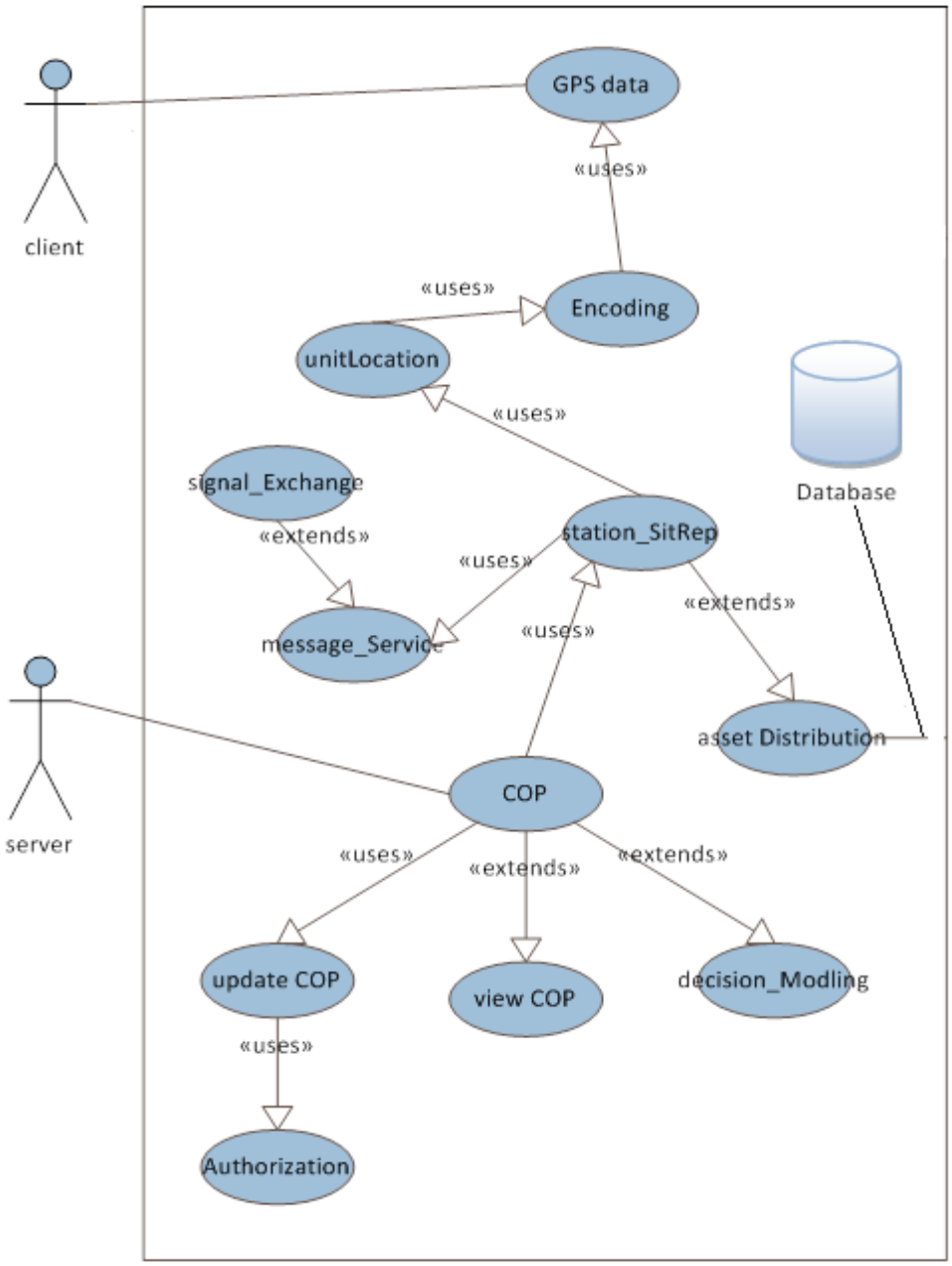

Figure 0.4 Use Case Diagram

4.4 Logical View

4.4.1 Sequence Diagram

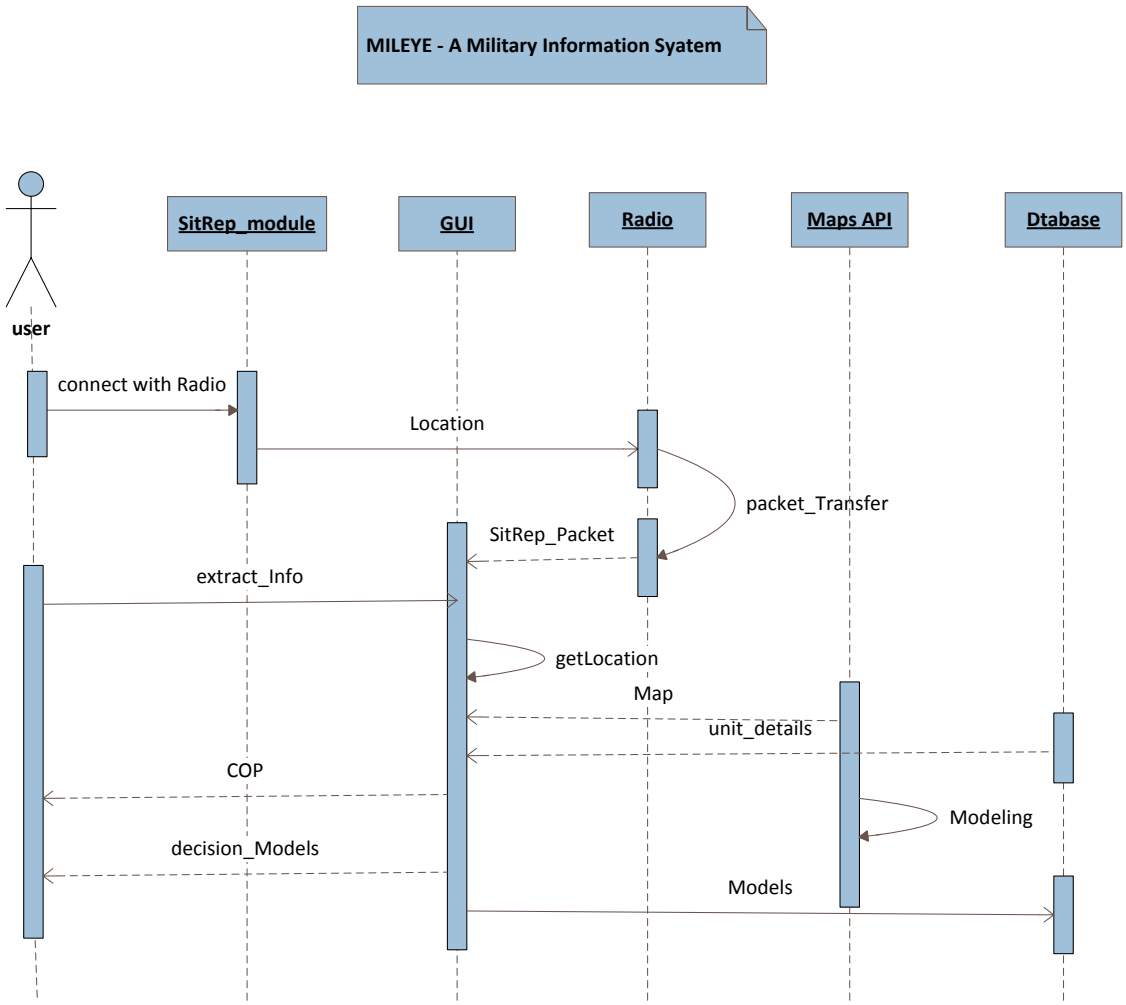

Figure 0.5 Sequence Diagram

4.5 Dynamic View

4.5.1 Activity Diagram

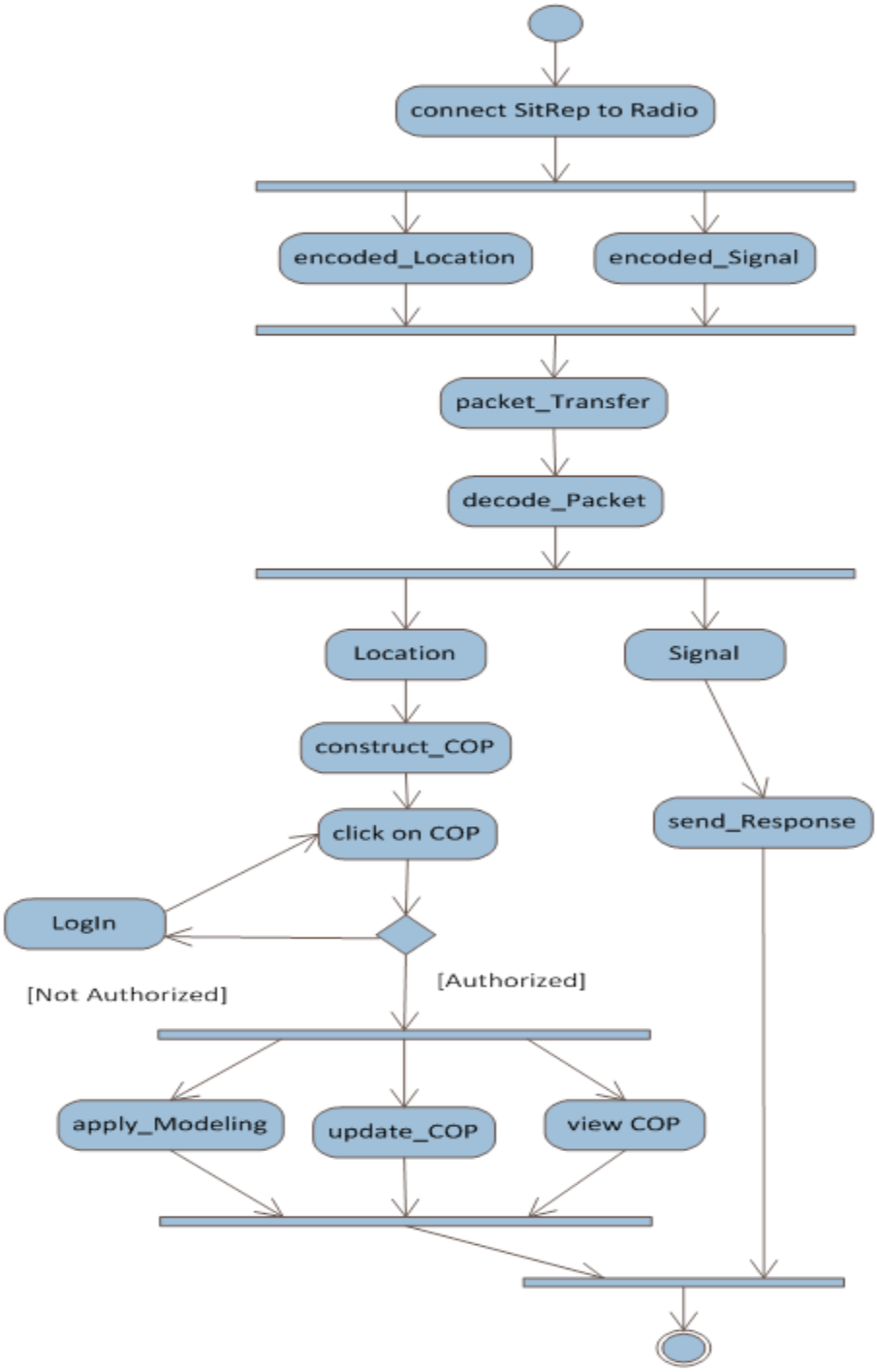

Figure 0.6 Activity Diagram

Description:

First of all SitRep module is interfaced with the Radio and afterwards we have two options send encoded coordinates of location with Station ID or encoded signal, the packet is then sent while on receiving packet it is decoded and checked that it is a signal or the coordinates, in case of signal response is sent and if coordinates are received, they are plotted on COP according to the station ID. In the next Step we can view and update COP or apply Modeling after authorization.

4.5 Implementation View

4.6.1 Component Diagram

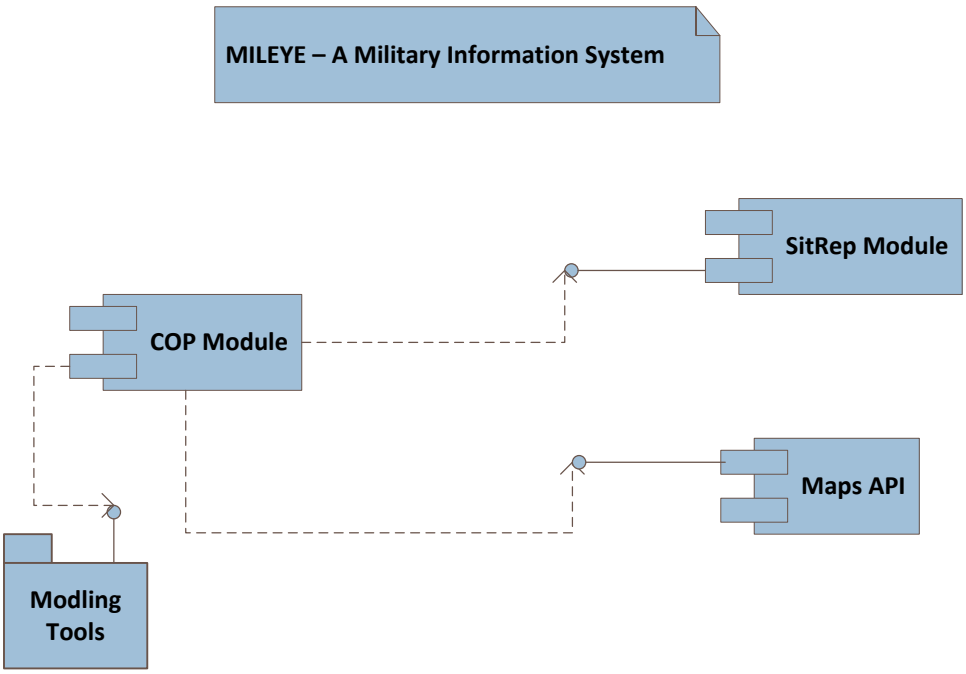

Figure 0.7 Component Diagram

Description:

Component diagram above shows the three main components of the system plus package supported. COP module consists of the common operational picture (COP) build from integrating SitRep module with Database together using the Maps API. COP also support modeling tools.

Chapter 5: System Implementation

5.1 Devices, Development Tools &API's

Microsoft Visual Studio is a complete set of development tools for building ASP.NET Web applications, XML Web services, desktop applications, and mobile applications. Visual Basic, Visual C++, Visual C#, and Visual J# all use the same integrated development environment (IDE), in both native code together with managed code for all platforms supported by Microsoft Windows, Windows Mobile, Windows CE, .NET Framework, .NET Compact Framework and Microsoft Silverlight.

Visual Studio includes a code editor supporting IntelliSense as well as code refactoring. The integrated debugger works both as a source-level debugger and a machine-level debugger. Other built-in tools include a forms designer for building GUI applications, web designer, class designer, and database schema designer. The Visual Studio integrated environment also includes tools for targeting devices such as PDAs and Smartphones.

The .NET Framework is an application development platform. It offers services for building, deploying, and running desktop, web, and phone applications as well as web services. It comprises of two major constituents: The common language runtime (CLR), an application virtual machine that provides services such as security, memory management, and exception handling. The class library and the CLR together constitute the .NET Framework.

The .NET Framework's Base Class Library provides user interface, data access, database connectivity, cryptography, web application development, numeric algorithms, and network communications.

Microsoft Windows CE 6.0

Windows Embedded CE is a 32 - bit, native, hard, real - time, small - footprint operating system developed by Microsoft to address the needs of handheld, mobile, and embedded devices. With support for multiple processor architectures, Windows Embedded CE can be adapted to a variety of devices like Smartphones, PocketPCs, set - top boxes, thin - client terminals, digital cameras, DVRs, VoIP, point - of - sale, point - of - information, network routers, wireless projectors, industrial automation, home and building automation, robotics, data acquisition, and human – machine interfaces.

CE 6.0 Features:

a) The system components which now run in kernel have been converted from EXEs to DLLs, which get loaded into kernel space.
b) New virtual memory model. The lower 2 GB is the process VM space and is private per process. The upper 2 GB is the kernel VM space.
c) New device driver model that supports both user mode and kernel mode drivers.
d) The 32 process limit has been raised to 32,768 processes.
e) The 32 megabyte virtual memory limit has been raised to the total virtual memory; up to 2 GB of private VM is available per process.
f) The Platform Builder IDE is integrated into Microsoft Visual Studio 2005 as plugin (thus forcing the client to obtain Microsoft Visual Studio 2005 also), allowing one development environment for both platform and application development.
g) Read-only support for UDF 2.5 file system.
h) Support for Microsoft's exFAT file system.
i) 802.11i (WPA2) and 802.11e (QoS) wireless standards, and multiple radio support.
j) CE 6.0 is compatible with x86, ARM, SH4 and MIPS based processor architectures.

k) New Cellcore components to enable devices to easily make data connections and initiate voice calls through cellular networks.

Microsoft SQL Server 2008

SQL Server 2008 is used as data storage backend for different varieties of data: XML, email, time/calendar, file, document, spatial etc. as well as perform search, query, analysis, sharing, and synchronization across all data types.

The GPS 18x PC interfaces to a serial port of a computer via a DB-9 connector and receives power through a 12-volt cigarette lighter adapter. The unit accepts TIA-232-F (RS-232) level inputs and transmits voltage levels that swing from zero V (ground) to 5 V TIA-232-F (RS-232) polarity.

Figure 5.1 Use Garmin 18x PC

GMap.NET is a free .NET control which enables mapping functionality to application. It can be used in Windows form, WPF and Mobile applications and supports several maps including;

a) OpenStreetMap
b) Yahoo Maps
c) Bing Maps
d) ArcGIS
e) Google Maps

Here are a few to understand when using the control:

a) GMapControl renders the map.
b) GMapOverlay is a layer on top of the map control. We can have several layers on top of a map, each layer representing, say, a route with stops, a list of military installation etc.
c) GMapMarker are the points on a layer, each representing a specific geo location (Lat,Lon) e.g. each unit location.
d) GMapRoute is the path or direction between two or more points.

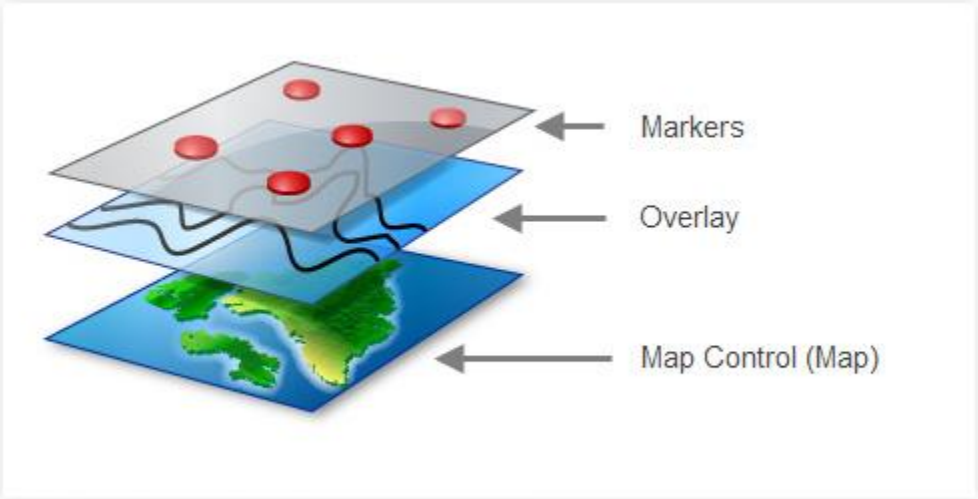

Figure 5.2 Gmap API Architecture

BeagleBoard-xM

BeagleBoard-xM delivers extra ARM ® Cortex ™ -A8 MHz at **1 GHz** and extra memory with **512MB** of low-power DDR RAM, Designed with the community inputs in mind, this open hardware design improves upon the laptop-like performance and expandability, while keeping at hand-held power levels. Direct connectivity is supported by the on-

board four-port hub with 10/100 Ethernet, while maintaining a tiny 3.25" × 3.25" footprint.

Figure 5.3 Beagleboard-xM

Specifications:

a) Package on Package POP CPU/memory chip.

   i. Processor TI DM3730 Processor - 1 GHz ARM Cortex-A8 core
   ii. 512 MB LPDDR RAM
   iii. 4 GB microSD card supplied with the BeagleBoard-xM

b) Peripheral connections[11]

   i. DVI-D (HDMI connector - maximum resolution is 1400x1050)
   ii. USB OTG (mini AB)
   iii. 4 USB ports
   iv. Ethernet port
   v. MicroSD/MMC card slot
   vi. RS-232 port
   vii. JTAG connector
   viii. Power socket (5 V barrel connector type)
   ix. Camera port
   x. Expansion port

c) Development

   i. Boot code stored on the uSD card

Software Defined Radio

A software defined radio system, or SDR, is a radio communication system where components that have been typically implemented in are instead implemented by means of software on a personal computer or embedded system.

Software radios have significant utility for the military and cell phone services, both of which serve a wide variety of changing radio protocols in real time.

5.2 Client Implementation

5.2.1 Windows CE 6.0 Development Environment

Prerequisites:

a) Install Visual Studio 2005.
b) Install Windows Embedded CE 6.0.
c) Install Visual Studio 2005 SP1 URL
d) Install the Windows Embedded CE 6.0 SP1
e) Install the Windows Embedded CE 6.0 R2 update

A typical CE device development project involves following steps:

a) Identify the hardware platform and hardware vendor for the project.
b) Identify additional hardware peripherals needed for the project.
c) Acquire or develop the CE BSP for the hardware platform.
d) Acquire or develop the CE device driver for the additional peripherals.
e) Develop the OS design and generate the CE runtime image to run on the hardware.
f) Develop the application.
g) Build the prototype.
h) Debug and test.

5.2.2 DEVELOP AN OS DESIGN

This section works through the exercises to create an OS design project, stepping through the process to customize the OS design and generate a customized OS run-time image for the target device.

To build a CE 6.0 runtime image, we will go through the following steps:

a) Create the OS design project.

b) Select the Board Support Package.
c) Select additional needed components from the CE 6.0 component catalog.
d) Generate the OS runtime image from the OS design project.

Screenshots:

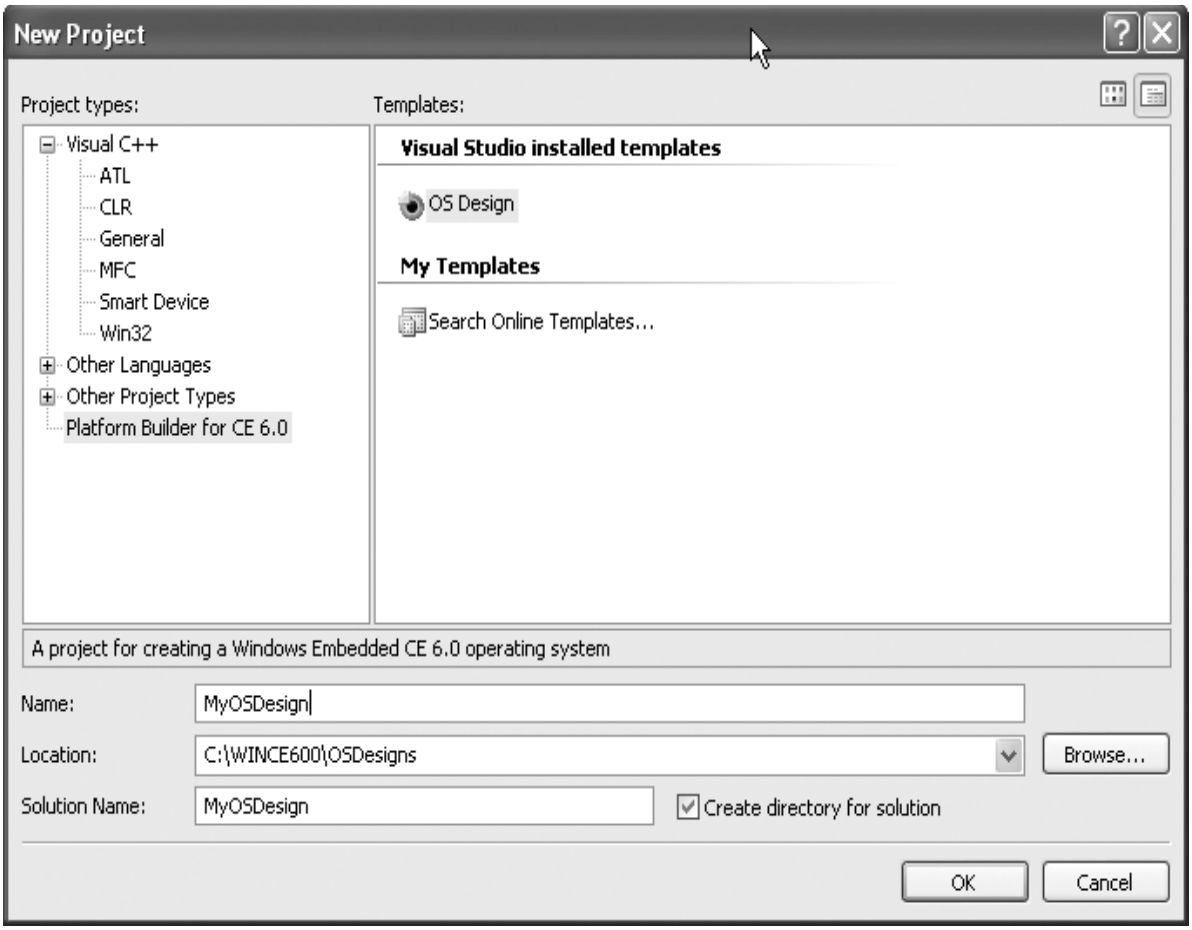

Figure 5.4 OS Design Step1

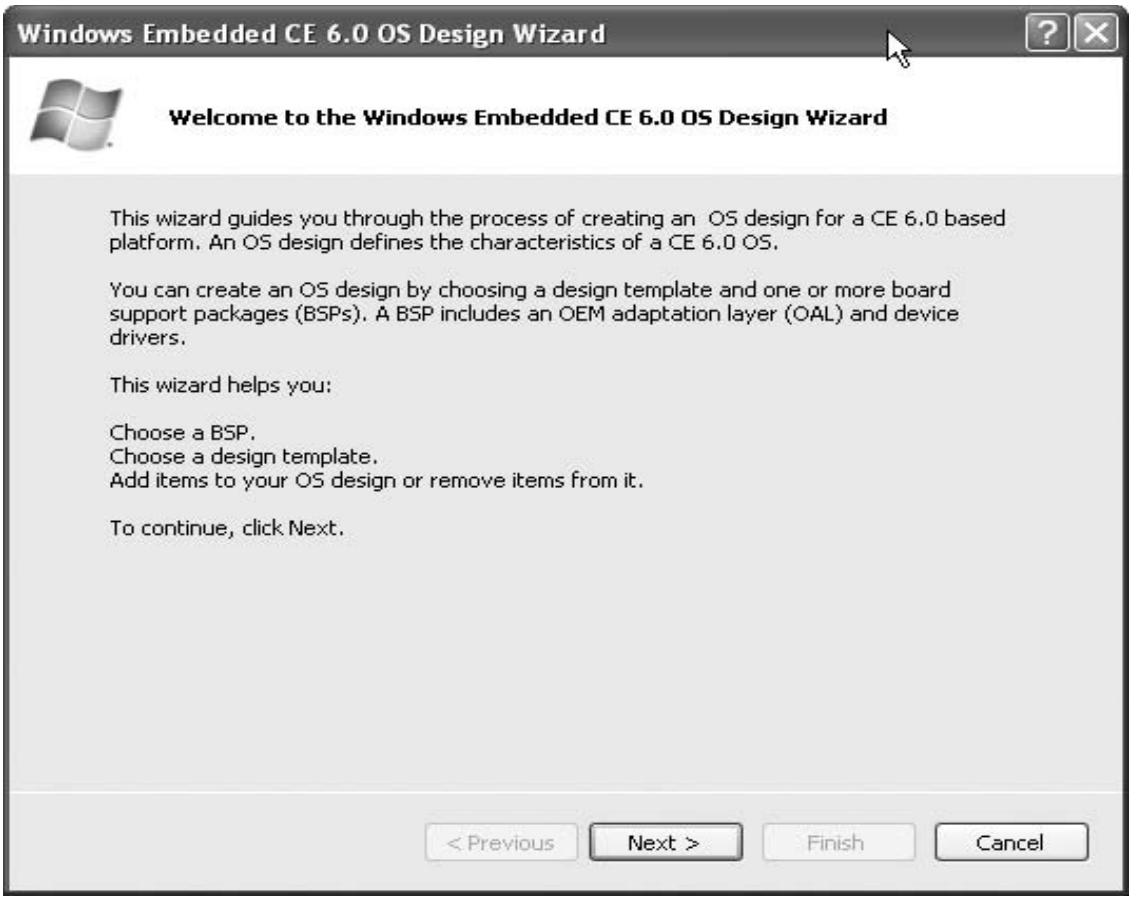

Figure 5.5 OS Design Step2

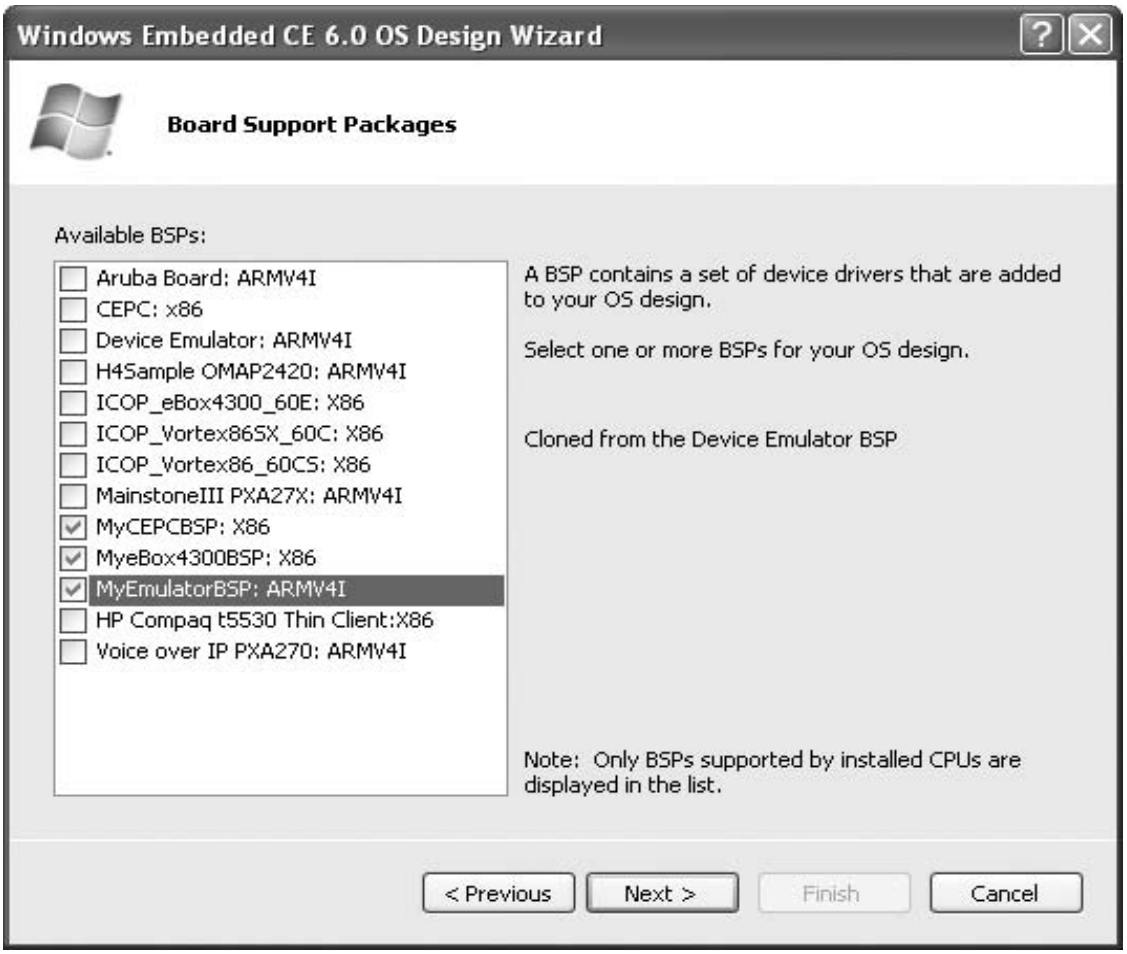

Figure 5.6 OS Design Step3

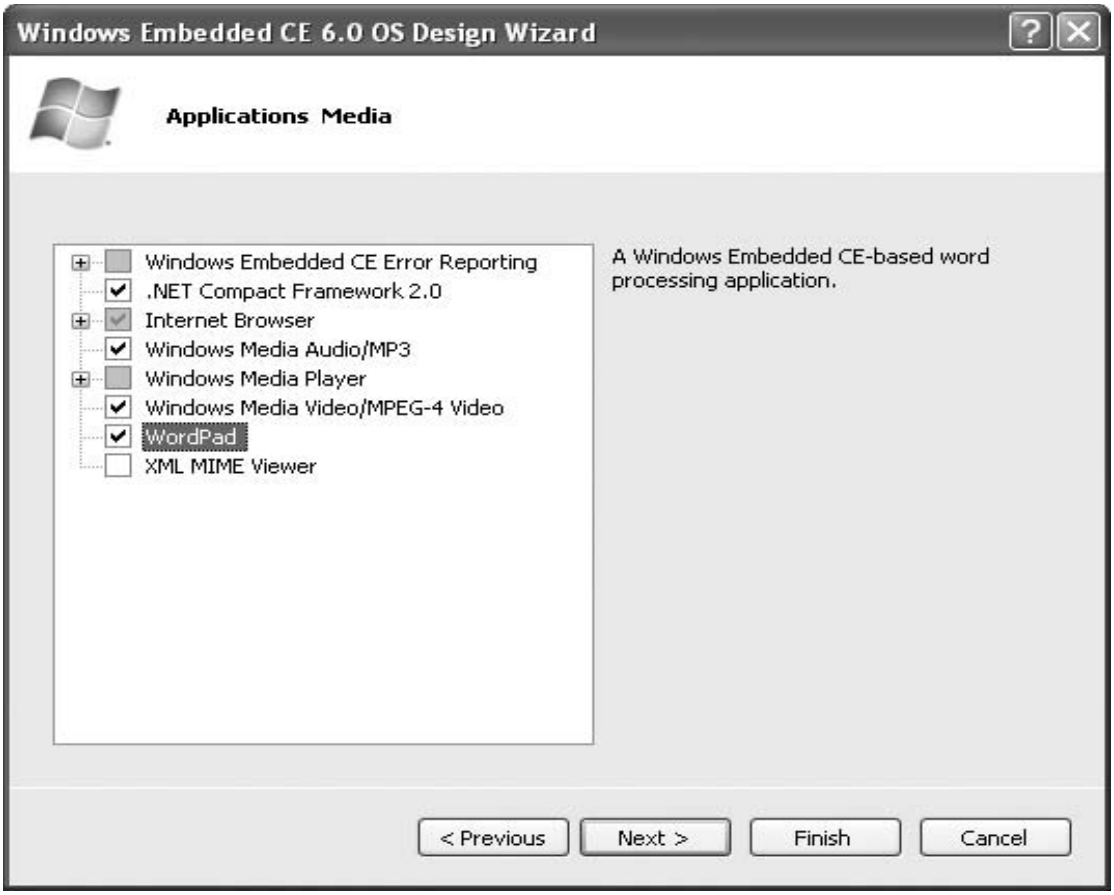

Figure 5.7 OS Design Step4

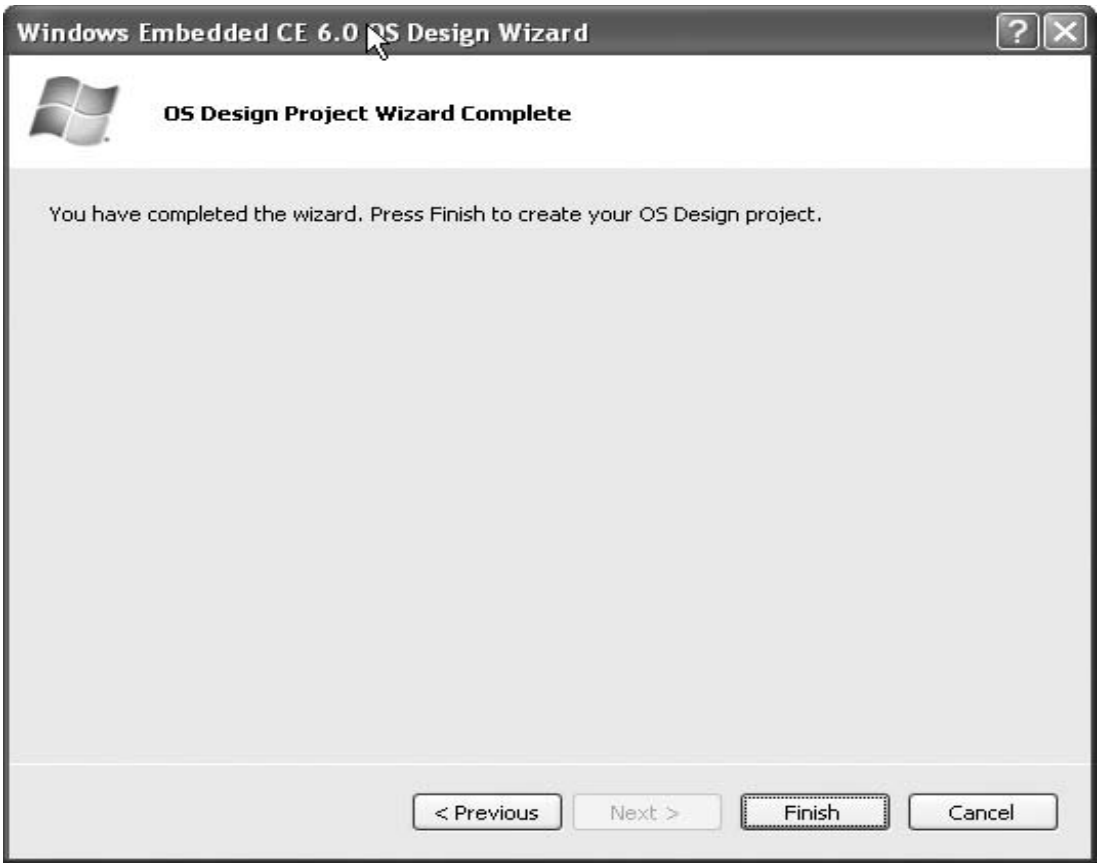

Figure 5.8 OS Design Step5

5.2.3 SDK Development (Windows Embedded CE 6.0)

An SDK is a set of headers, libraries, connectivity files, run-time files, OS design extensions, and Help documentation that developers use to write applications for a specific OS design. The contents of an SDK allow developers to create and debug an application on the run-time image built from OS design.

The following illustration shows the relationship between Platform Builder, SDK, and Microsoft development software in the SDK development process.

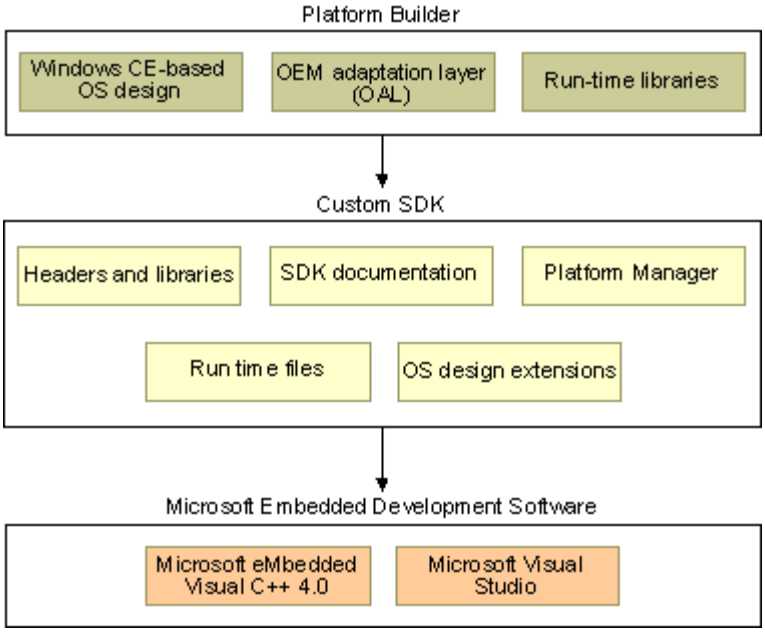

Figure 5.9 OS WinCE SDK

5.2.4 WinCE Application

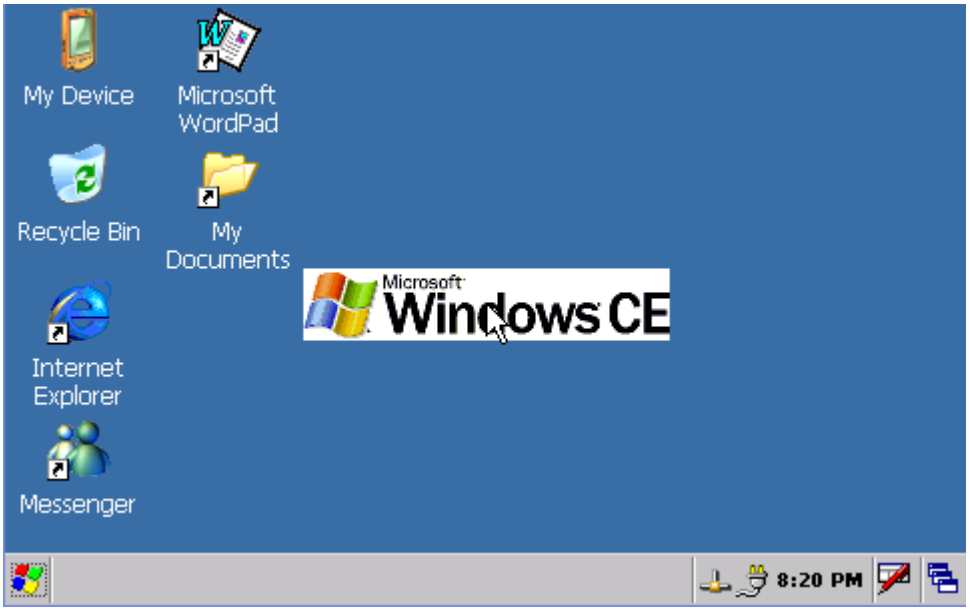

Figure 5.10 WinCE Home Screen

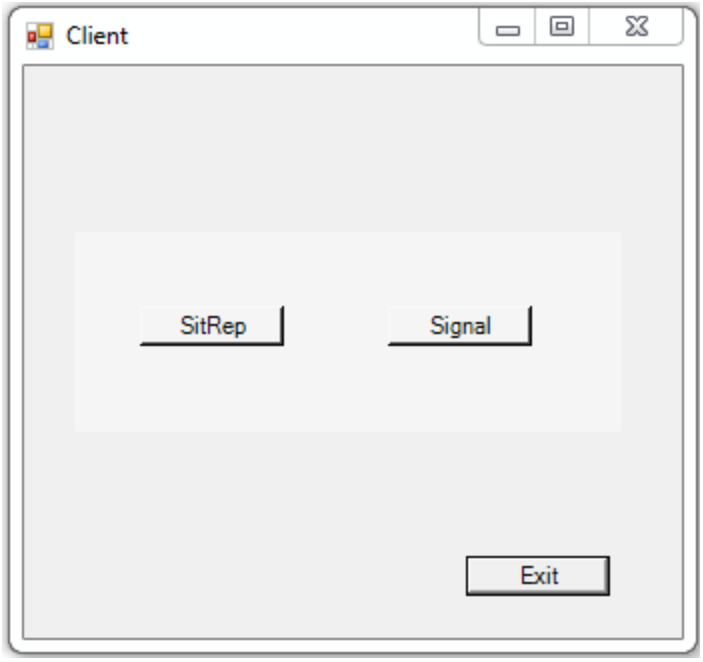

Figure 5.11 Client Application

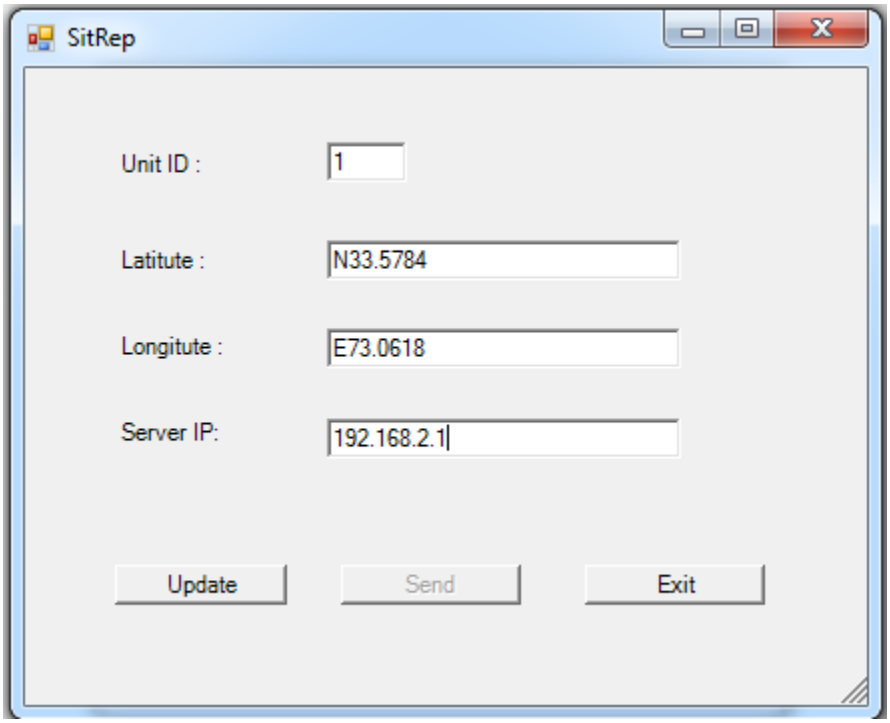

Figure 5.12 SitRep Module

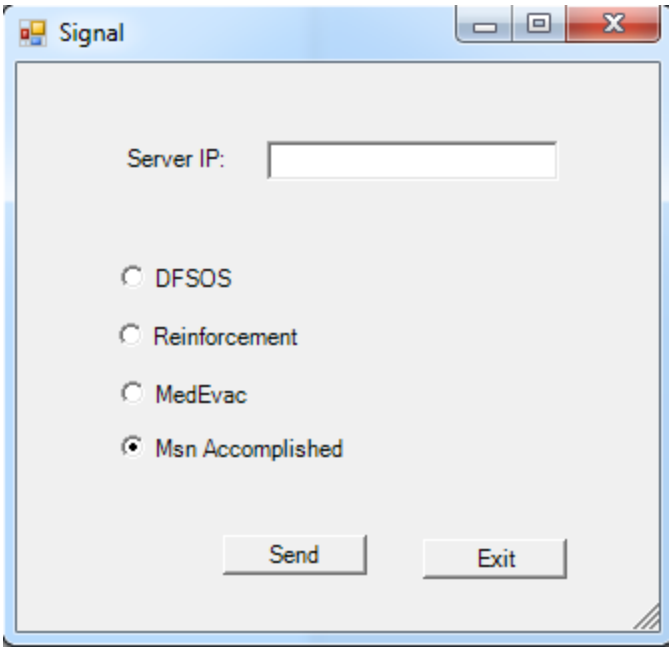

Figure 5.13 Signal Alert Module

Reading Latitude and Longitude from GPS data received over client serial interface:

```
if (serialPort1.IsOpen)
  {
    string data = serialPort1.ReadExisting();
    string[] strArr = data.Split('$');
    for (int i = 0; i < strArr.Length; i++)
    {
      string strTemp = strArr[i];
      string[] lineArr = strTemp.Split(',');
      if (lineArr[0] == "GPGGA")
      {
        try
```

```
        {
            //Latitude
            Double dLat = Convert.ToDouble(lineArr[2]);
            dLat = dLat / 100;
            string[] lat = dLat.ToString().Split('.');
            Latitude = lineArr[3].ToString() +
            lat[0].ToString() + "." +
            ((Convert.ToDouble(lat[1]) /
            60)).ToString("#####");
            //Longitude
            Double dLon = Convert.ToDouble(lineArr[4]);
            dLon = dLon / 100;
            string[] lon = dLon.ToString().Split('.');
            Longitude = lineArr[5].ToString() +
            lon[0].ToString() + "." +
            ((Convert.ToDouble(lon[1]) /
            60)).ToString("#####");
            //Display
            textBox1.Text = Latitude;
            textBox2.Text = Longitude;
            button2.Enabled = true;
        }
 catch
        {    }
} } }
```

5.3 Server Implementation

Figure 5.14 COP Plotting

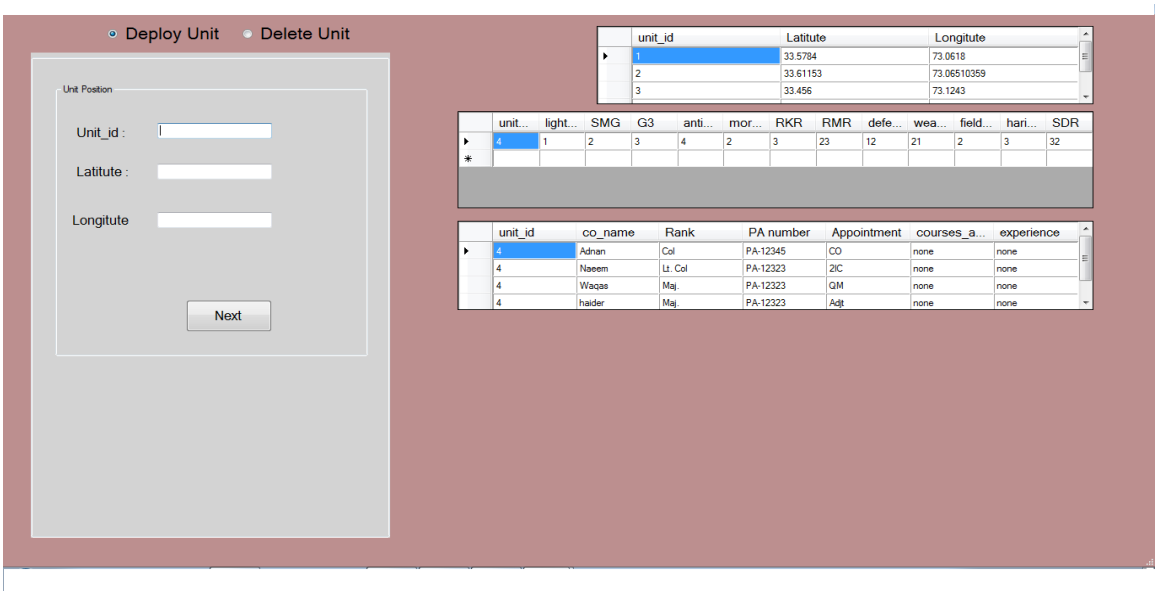

Figure 5.15 Unit Deployment Module

Adding Initial Position Marker

```
private void button1_Click_1(object sender, EventArgs e)
    {
      string lat = null, log = null, unit_id = null;
      int i = 0;
      if (comboBox1.SelectedIndex != -1)
      {
        overLayOwnPos.Markers.Remove(new GMapMarkerBlue(pos));
        gmCon.Overlays.Remove(overLayOwnPos);
      }
      overLayOwnPos = new GMapOverlay(gmCon, "ownPos");
      try
      {
        SqlConnection c = new SqlConnection(@"Data
        Source=.\SQLEXPRESS;AttachDbFilename=|DataDirectory|\info.mdf;Integrated
        Security=True;Connect Timeout=30;User Instance=True");
        string s = "Select * from unit_position";
        c.Open();
        SqlCommand cmd = new SqlCommand(s, c);
        SqlDataReader reader = cmd.ExecuteReader();
        while (reader.Read())
        {
          lat = reader["latitute"].ToString();
          log = reader["longitute"].ToString();
          unit_id = reader["unit_id"].ToString();
          pos = new PointLatLng(Convert.ToDouble(lat), Convert.ToDouble(log));
```

```
            //inititing ownposn overlay

            //initiating onw posn marker

            overLayOwnPos.Markers.Add(new GMapMarkerBlue(pos));

            ownPosn = overLayOwnPos.Markers[i];

            ownPosn.ToolTipText = "Unit ID =" + unit_id + "\n" + ownPosn.Position.Lat.ToString() +
        "\n" + ownPosn.Position.Lng.ToString();

            ownPosn.IsVisible = true;

            i++;

        }

        //putting ownposn overlay on map

        gmCon.Overlays.Add(overLayOwnPos);



        reader.Close();

        c.Close();

    }

    catch

    {     }

    button1.Enabled = false;

    comboBox1.Enabled = false;

}
```

Updating Position

```
public void get()
    {
      byte[] data = new byte[1024];
      ipep = new IPEndPoint(IPAddress.Parse("102.0.124.66"), 8080);
      newsock = new UdpClient(ipep);

      MessageBox.Show("Waiting for a client...");
      IPEndPoint sender = new IPEndPoint(IPAddress.Any, 0);
      string[] a =null;
      string s;
      try
      {
        data = newsock.Receive(ref sender);
        MessageBox.Show(Encoding.ASCII.GetString(data, 0, data.Length));
        string welcome = Encoding.ASCII.GetString(data, 0, data.Length);
        a = welcome.Split(',');
       MessageBox.Show("Your unit id is = " + a[0] + "\nyour Latitude is =" + a[1] + "\n Your
      Longititude is =" + a[2]);
      SqlConnection con = new SqlConnection(@"Data
      Source=.\SQLEXPRESS;AttachDbFilename=|DataDirectory|\info.mdf;Integrated
      Security=True;Connect Timeout=30;User Instance=True");
      string str1 = "update unit_position set latitute='" + a[1] + "',longitute='" + a[2] + "' where
      unit_id='"+a[0]+"'";
        SqlCommand cmd1 = new SqlCommand(str1, con);
        con.Open();
        cmd1.ExecuteNonQuery();
```

```
            con.Close();
            MessageBox.Show("Position Successfully Updated");
            }
        catch
        {
            MessageBox.Show("there is some error..");
            newsock.Close();
        }
```

Adding Overlay

```
private void pictureBox3_Click(object sender, EventArgs e)
    {
        overLayOwnPos = new GMapOverlay(gmCon, "ownPos");
        overLayOwnPos.Markers.Add(new Class1(pos));
        gmCon.Overlays.Add(overLayOwnPos);
    }
```

Chapter 6: Testing and Results Analysis

This system Test Plan is to provide the description of test cases for MILEYE – Secure Military Information System. This document will describe the test cases for different features of the system such as SitRep Module, Plotting COP, Update COP, Add unit etc.

Software Requirement and Specification document of the system supports this Software Test Plan.

6.1 Test Plan

6.1.1 Test Items

Following are test items and their version:

Test Item Name	Test Item Version No.	Test Type
Overall System	Version 1.01	Black Box
SitRep Module	Version 1.01	Black Box
GPS Module	Version 1.01	Black Box
COP Module	Version 1.01	Black Box
Messaging Module	Version 1.01	Black Box
Communication Module	Version 1.01	Black Box
Add Unit	Version 1.01	Black Box
Delete Unit	Version 1.01	Black Box
Update Unit	Version 1.01	Black Box
Update COP	Version 1.01	Black Box
Database Working	Version 1.01	Black Box
SitRep Class	Version 1.01	White Box
GPS Class	Version 1.01	White Box
Maps API	Version 1.01	White Box
GUI Test	Version 1.01	White Box

Table 6.1 Test Items

6.1.2 Features to be Tested

Feature	Parent Component/System	Overview
Overall System	MILEYE – Secure Military Information System	Run the Secure Command and Control Desktop (Server) and WinCE Client. Send SitRep packet from client side and plot COP on server side.
SitRep Module	Station_SitRep	Run the client. Format SitRep packet. Send packet to server.
GPS Module	micro_Data	Run the client. Connect GPS through serial interface. Format the SitRep packet.
COP Module	COP	Run the server. Try to plot and edit the COP.
Messaging Module	messaging_Service	Run the client or server and try to send any signal listed.
Communication Module	MILEYE –Secure Military Information System	Run both client and server and try to establish the connection.
Add Unit	Database	Run the server and try to add a complete unit details by insert query.
Delete Unit	Database	Run the server and try to delete a unit details by delete query.
Update Unit	Database	Run the server and try to update the unit details by update query.
Update COP	COP Class	Run the server. Try to edit COP and again plot the updated COP.
Database Working	Database	Check connectivity, security and working of database system.
SitRep Class	MILEYE –Secure Military Information System	Run the client and format SitRep packet.
GPS Class	micro_Data	Run the client and connect the GPS through serial port and get Lat, Lon from GPS data string.
Maps API	map_API	Run the server. Try to add or delete marker, zoom in or zoom out the map.
GUI	MILEYE –Secure Military Information System	Test GUI items such as buttons, navigational link, color schemes, font compatibility etc.

Table 6.2 Features to be Tested

6.1.3 Features Not to Be Tested

Due to sensitivity of the system all features need testing.

6.1.4 Approach

Gray Box testing since we are going to adopt Black ox testing for output oriented tests and White box testing for internal working. Testing approach will be same for all features of the system. System will be tested in different perspectives such as GUI testing, logical errors, database functionality etc.

No separate tool will be used to test the system. Code will be reviewed on Visual Studio (2005 & 2010) as developed on it. Beagleboard-xM, Garmin 18xPC GPS and DB 9 serial cable for client deployment testing. Testing will be comprehensive so that to ensure high quality of the system. Minimum degree of comprehensive required is that all features should be tested at least 3 times. To judge the comprehensiveness a record will be maintained of the features which have been tested completely i.e. a checkList.

6.1.5 Item Pass/Fail Criteria

Any test item will be declared pass if it conforms to the Pre-Conditions and post-Conditions specified in the SRS and fail if it does not conform to the requirements.

6.1.6 Suspension Criteria and Resumption Requirements

The test activity will be suspended if a feature is declared fail after its testing and will resume again from the start when the error in the feature is removed.

6.1.7 Environmental Needs

Windows based system and WinCE based Beagleboard-xM is required for testing of the server and client respectively. SDR network. Microsoft Visual Studio (2005 & 2010) is required to run and edit the application. Microsoft .Net frame 2.0 for client.exe and 3.5 or higher is required to run server.exe.

6.1.8 Schedule

Testing should be completed within 5 weeks.

6.1.9 Risks and Contingencies

High risk is there if all the features of the system are not completely tested in the given schedule. Not only that we will be violating deadline but system security and quality will be compromised.

Note: There are three types of priorities (Importance of the test case in terms of functionality)

P0: Basic Functionality

P1: General Functionality (Input Domain, Error Handling, Compatibility)

P2: Cosmetic Testing (User Interface)

6.2 Black Box Testing

The software program or system under test is viewed as a "black box". The selection of test cases for functional testing is based on the requirement or design specification of the software entity under test. Functional testing emphasizes on the external behavior of the software entity.

Test Case Id: BB01

1. **Test Case Name:** Overall System
2. **Features to be Tested:** It will test overall working of the system.
3. **Test Suite Id:** Batch 1.01
4. **Priority:**

   P0: Basic Functionality
5. **Test Environment:** System with windows/Beagleboard with WinCE/Visual Studio 2010 & 2005/Garmin 18x PC GPS/SDR.
6. **Test Effort (Person/Hour): 15** minutes on an average.
7. **Test Date:** April 05, 2013. 1030 hours

8. **Test Setup:** System should be in working condition. Application should also be in working condition.
9. **Test Procedure:**

    a) Run the client.
        i. Connect GPS through serial interface.
        ii. Connect Beagleboard-xM to SDR client through Ethernet interface.
        iii. Format SitRep packet.
        iv. Click send button to send packet to the server.
    b) Run server.
        i. Connect server's system to SDR server through Ethernet interface.
        ii. Listen over UDP socket for clients.
        iii. Receive updates.
        iv. Plot COP.

10. **Test Case Pass/Fail Criteria:** Packet sent from client should be received on server side and position of the respective unit should be updated.
11. **Actual Output:** System works as specified in passing criteria.
12. **Status:** Passed.

Test Case Id: BB02

1. **Test Case Name:** SitRep Module
2. **Features to be Tested:** It will test the SitRep packet generation functionality of the system.
3. **Test Suite Id:** Batch 1.01
4. **Priority:**

    P0: Basic Functionality

5. **Test Environment:** Beagleboard with WinCE/Visual Studio 2005/Garmin 18x PC GPS.
6. **Test Effort (Person/Hour): 05** minutes on an average.
7. **Test Date:** April 05, 2013. 1115 hours

8. **Test Setup:** System should be in working condition. Application should also be in working condition.
9. **Test Procedure:**

    a) Run the client.
        i. Connect GPS through serial interface.
        ii. Format SitRep packet.
        iii. Click send button to see SitRep module works.

10. **Test Case Pass/Fail Criteria:** Success message should be there when user hit the send button.
11. **Actual Output:** Message box with success message.
12. **Status:** Passed.

Test Case Id: BB03

1. **Test Case Name:** GPS Module
2. **Features to be Tested:** It will test the fetching of Lat, Lon from GPS data string and their encoding.
3. **Test Suite Id:** Batch 1.01
4. **Priority:**

    P0: Basic Functionality

5. **Test Environment:** Beagleboard with WinCE/Visual Studio 2005/Garmin 18x PC GPS.
6. **Test Effort (Person/Hour): 05** minutes on an average.
7. **Test Date:** April 05, 2013. 1130 hours
8. **Test Setup:** System should be in working condition. Application should also be in working condition.
9. **Test Procedure:**

    a) Run the client.
        i. Connect GPS through serial interface.

ii. Format SitRep packet.
iii. Click send button to see if GPS module works.

10. **Test Case Pass/Fail Criteria:** Success message should be there when user hit the send button.
11. **Actual Output:** Message box with success message.
12. **Status:** Passed.

Test Case Id: BB04

1. **Test Case Name:** COP Module
2. **Features to be Tested:** It will test the plotting of COP from data received from client stations.
3. **Test Suite Id:** Batch 1.01
4. **Priority:**

   P0: Basic Functionality

5. **Test Environment:** System with Windows and .Net framework (3.5 or higher)/Visual Studio 2010/Gmap.Net API
6. **Test Effort (Person/Hour): 15** minutes on an average.
7. **Test Date:** April 05, 2013. 1145 hours
8. **Test Setup:** System should be in working condition. Application should also be in working condition.
9. **Test Procedure:**

   a) Run the server.
      i. Listen on UDP socket for clients.
      ii. Receive updates from clients.
      iii. Process the received packets.
      iv. Update the respective client's location.
      v. Re-draw the COP

10. **Test Case Pass/Fail Criteria:** Success message should be there when user hit the Draw COP button.

11. **Actual Output:** Message box with success message.

12. **Status:** Passed.

Test Case Id: BB05

1. **Test Case Name:** Messaging Module
2. **Features to be Tested:** It will test the exchange of signals between client and server stations.
3. **Test Suite Id:** Batch 1.01
4. **Priority:**

   P0: Basic Functionality

5. **Test Environment:** System with Windows/Beagleboard-xM with WinCE/Visual Studio 2010 &2005
6. **Test Effort (Person/Hour): 05** minutes on an average.
7. **Test Date:** April 10, 2013. 1230 hours
8. **Test Setup:** System should be in working condition. Application should also be in working condition.
9. **Test Procedure:**

   a) Run the server.
      i. Select the signal to send.
      ii. Hit the send button
      iii. Wait for response.
   b) Run the client.
      i. View the signal received from server.
      ii. Interpret the signal.
      iii. Select appropriate response
      iv. Send response.

10. **Test Case Pass/Fail Criteria:** Server should get response within the wait limit.

11. **Actual Output:** Error: “No Response from Client”.

12. **Status:** Failed.

Test Case Id: BB06

1. **Test Case Name:** Communication Module
2. **Features to be Tested:** It will test the exchange of signals between client and server stations.
3. **Test Suite Id:** Batch 1.01
4. **Priority:**

   P0: Basic Functionality

5. **Test Environment:** System with windows/Beagleboard with WinCE/Visual Studio 2010 & 2005/Garmin 18x PC GPS/SDR.
6. **Test Effort (Person/Hour): 15** minutes on an average.
7. **Test Date:** April 10, 2013. 1530 hours
8. **Test Setup:** System should be in working condition. Application should also be in working condition.
9. **Test Procedure:**

   a) Run the server.
      i. Listen for client over UDP socket.
      ii. Take appropriate action.
   b) Run the client.
      i. Select the signal to send.
      ii. Hit the send button.
      iii. Wait for response.

10. **Test Case Pass/Fail Criteria:** client should get response within the wait limit.
11. **Actual Output:** Client receives response signal
12. **Status:** Passed.

Test Case Id: BB07

1. **Test Case Name:** Add Unit
2. **Features to be Tested:** It will test the feature of adding a new unit using server station.
3. **Test Suite Id:** Batch 1.01
4. **Priority:**

   P0: Basic Functionality

5. **Test Environment:** System with windows/Visual Studio 2010/Gmap.Net API.
6. **Test Effort (Person/Hour): 10** minutes on an average.
7. **Test Date:** April 15, 2013. 1000 hours
8. **Test Setup:** System should be in working condition. Application should also be in working condition.
9. **Test Procedure:**

   a) Run the server.
      i. Click the Add Unit button.
      ii. Fill the field values for unit details.
      iii. Click ok to run the insert query.

10. **Test Case Pass/Fail Criteria:** New marker should be created as new unit is added to database.
11. **Actual Output:** New marker is appeared.
12. **Status:** Passed.

Test Case Id: BB08

1. **Test Case Name:** Delete Unit
2. **Features to be Tested:** It will test the feature of deleting an existing unit using server station.
3. **Test Suite Id:** Batch 1.01
4. **Priority:**

P0: Basic Functionality

5. **Test Environment:** System with windows/Visual Studio 2010/Gmap.Net API.
6. **Test Effort (Person/Hour): 10** minutes on an average.
7. **Test Date:** April 15, 2013. 1030 hours
8. **Test Setup:** System should be in working condition. Application should also be in working condition.
9. **Test Procedure:**

    a) Run the server.
        i. Enter the ID of the unit to be deleted.
        ii. Click the Delete Unit button.
        iii. Click ok to confirm deletion from confirmation box.

10. **Test Case Pass/Fail Criteria:** Marker should be removed as unit details are removed from database in response to delete query.
11. **Actual Output:** Marker is removed.
12. **Status:** Passed.

Test Case Id: BB09

1. **Test Case Name:** Update Unit
2. **Features to be Tested:** It will test the feature of updating an existing unit using server station.
3. **Test Suite Id:** Batch 1.01
4. **Priority:**

    P0: Basic Functionality

5. **Test Environment:** System with windows/Visual Studio 2010/Gmap.Net API.
6. **Test Effort (Person/Hour): 10** minutes on an average.
7. **Test Date:** April 15, 2013. 1200 hours
8. **Test Setup:** System should be in working condition. Application should also be in working condition.

9. **Test Procedure:**

    a) Run the server.
        i. Enter the ID of the unit to be updated.
        ii. Update unit details.
        iii. Click ok to confirm from confirmation box.

10. **Test Case Pass/Fail Criteria:** Updated details should reflect in database.

11. **Actual Output:** Unit details are updated in database.

12. **Status:** Passed.

Test Case Id: BB10

1. **Test Case Name:** Update COP
2. **Features to be Tested:** It will test the feature of updating COP using server station.
3. **Test Suite Id:** Batch 1.01
4. **Priority:**

    P0: Basic Functionality

5. **Test Environment:** System with Windows and .Net framework (3.5 or higher)/Visual Studio 2010/Gmap.Net API
6. **Test Effort (Person/Hour): 15** minutes on an average.
7. **Test Date:** April 15, 2013. 1500 hours
8. **Test Setup:** System should be in working condition. Application should also be in working condition.
9. **Test Procedure:**

    a) Run the server.
        i. Load the map cached from API.
        ii. Click COP button to plot COP.
        iii. Edit COP.
        iv. Click ok to confirm update.
        v. Re-draw the updated COP

10. **Test Case Pass/Fail Criteria:** Success message should be there when user hit the ok button to confirm and COP should reflect updates.
11. **Actual Output:** Message box with success message.
12. **Status:** Passed.

Test Case Id: BB11

1. **Test Case Name:** Database Working
2. **Features to be Tested:** It will test the connectivity, security and working of database system.
3. **Test Suite Id:** Batch 1.01
4. **Priority:**

   P0: Basic Functionality

5. **Test Environment:** System with Windows/Visual Studio 2010/SQL Server 2008
6. **Test Effort (Person/Hour): 25** minutes on an average.
7. **Test Date:** April 15, 2013. 1500 hours
8. **Test Setup:** System should be in working condition. Application should also be in working condition.
9. **Test Procedure:**

   a) Run the server.
      i. Select any database related feature such as Add Unit feature.
      ii. Input field values.
      iii. Hit Add Unit button.

10. **Test Case Pass/Fail Criteria:** Server should show success message only all fields contain valid values.
11. **Actual Output:** Success message.
12. **Status:** Passed.

6.3White Box Testing

The software entity is viewed as a “white box”. The selection of test cases is based on the implementation of the software entity, on module by module basis.

Test Case Id: WB01

1. **Test Case Name:** SitRep Class
2. **Features to be Tested:** It will test the SitRep Class.
3. **Test Suite Id:** Batch 1.01
4. **Priority:**

   P1: General Functionality (Input Domain, Error Handling, Compatibility)

5. **Test Environment:** Beagleboard with WinCE/Visual Studio 2005/Garmin 18x PC GPS.
6. **Test Effort (Person/Hour): 15** minutes on an average.
7. **Test Date:** April 20, 2013. 1130 hours
8. **Test Setup:** System should be in working condition. Application should also be in working condition.
9. **Test Procedure:**

   a) Run the client.
      i. Connect GPS on DB9 port through serial interface.
      ii. Get the GPS data string from COM port.
      iii. Parse the GPS data to get Lat, Lon.
      iv. Encode Lat, Lon.
      v. Append client ID to complete packet.

10. **Test Case Pass/Fail Criteria:** GPS should be able to transmit data over COM port and client application should be able to parse GPS data string to form SitRep packet.
11. **Actual Output:** Encoded Lat, Lon with client ID.
12. **Status:** Passed.

Test Case Id: WB02

1. **Test Case Name:** GPS Class
2. **Features to be Tested:** It will test the working and compatibility of GPS.
3. **Test Suite Id:** Batch 1.01
4. **Priority:**

   P1: General Functionality (Input Domain, Error Handling, Compatibility)

5. **Test Environment:** Beagleboard-xM with WinCE/Visual Studio 2005/Garmin 18x PC GPS.
6. **Test Effort (Person/Hour): 10** minutes on an average.
7. **Test Date:** April 20, 2013. 1530 hours
8. **Test Setup:** System should be in working condition. Application should also be in working condition.
9. **Test Procedure:**

   a) Run the client.
      i. Connect GPS on DB9 port through serial interface.
      ii. Check the COM port for compatibility and connectivity.
      iii. Get the GPS data string from COM port.

10. **Test Case Pass/Fail Criteria:** GPS should be compatible with Beagleboard-xM and WinCE. GPS should be able to transmit data over COM port.
11. **Actual Output:** GPS transmitting data over COM port.
12. **Status:** Passed.

Test Case Id: WB03

1. **Test Case Name:** Maps API
2. **Features to be Tested:** It will test the working and compatibility of Gmap API.
3. **Test Suite Id:** Batch 1.01
4. **Priority:**

P1: General Functionality (Input Domain, Error Handling, Compatibility)

5. **Test Environment:** System with Windows and .Net framework (3.5 or higher)/Visual Studio 2010/Gmap.Net API.
6. **Test Effort (Person/Hour): 10** minutes on an average.
7. **Test Date:** April 25, 2013. 1130 hours
8. **Test Setup:** System should be in working condition. Application should also be in working condition.
9. **Test Procedure:**

    a) Run the server.
        i. Load Gmap.Net API.
        ii. Check functionality like Zoom in and zoom out, add or remove marker and try to apply overlays to the map.
        iii. Try to cache the maps.
        iv. Try to connect the API with database.

10. **Test Case Pass/Fail Criteria:** API should work properly with no compatibility issues or errors.
11. **Actual Output:** All API features are working normal, maps are cached and database connection is established successfully.
12. **Status:** Passed.

Test Case Id: WB04

1. **Test Case Name:** GUI Test
2. **Features to be Tested:** It will test the aesthetic and compatibility of GUI.
3. **Test Suite Id:** Batch 1.01
4. **Priority:**

    P2: Cosmetic Testing (User Interface)

5. **Test Environment:** System with Windows and .Net framework (3.5 or higher)/Visual Studio 2010/Gmap.Net API.

6. **Test Effort (Person/Hour): 30** minutes on an average.
7. **Test Date:** April 30, 2013. 1130 hours
8. **Test Setup:** System should be in working condition. Application should also be in working condition.
9. **Test Procedure:**

    a) For screen validation it will check
        i. Aesthetic Conditions
        ii. Navigation Conditions
        iii. Usability Conditions
        iv. Specific Field Checks
        v. Date Field checks
        vi. Numeric Field Checks
        vii. Alpha Field Checks

    b) For each Window in the Application it will test
        i. Text Boxes
        ii. Option (Radio Buttons)
        iii. Command Buttons

10. **Test Case Pass/Fail Criteria:**

        i. Resizing of window
        ii. All windows and dialog boxes have a consistent look and feel
        iii. The screen is accessed correctly from the menu, toolbar and control list.
        iv. Move the Mouse Cursor over all Enterable Text Boxes. Cursor should change from arrow to Insert Bar.
        v. Selection should also be possible with mouse. Double Click should select all text in box.
        vi. Click each command button once with the mouse should activate the feature.

11. **Actual Output:** All GUI are working normal and aesthetic features are also consistent.
12. **Status:** Passed.

Chapter 7: Conclusion and Future Work

MILEYE is a Secure Military Information System that totally improvise Pak Army's command and navigation process using our own embedded device as client and secure command and control desktop as server over SDR network so that quality of command and navigational tasks in terms of delay and maintenance can be improved a lot by with extra security provided by SDR as in our system situation awareness will be Real-Time and SDR use AES as encryption to secure communication.

In Beagleboard-xM, we have support for audio and video communication. There are different modes of SDR communication and compression techniques to support video transfer over SDR. Decision aid and planning tools can be integrated to enhance the decision making capabilities.

.

APPENDIX A: GLOSSARY

SitRep	Situation Report; kind of a summary of team's recent situation in the field
GPS	Global Positioning System
COP	Common Operational Picture of all unit deployed to give Real-Time situational awareness on every command tier
SDR	Software Defined Radio
WinCE	Windows Compact and Embedded
OS	Operating System
API	Application Programming Interface
SDK	Software Development Kit
TCP	Transmission Control Protocol
UDP	User Datagram Protocol

APPENDIX B: REFERENCES