CHAPTER 4

Implementation

4.1 Introduction

In this chapter we explain the actual implementation of the proposed solution .The whole process is based on the selected methodology and the programming language for user interface of mobile devices.

4.2 Steps of the ICONIX Methodology

As we already mentioned that ICONX methodology (more details in Chapter 3 section 3.4) .This methodology is very efficient at solving such mobile related applications. It consists of four main steps and each step has its own secondary parts. All of these steps will be executed to solve the problem according to the proposed methodology.

4.2.1 Step One :Requirement Analysis

This step is divided into three subtasks, these subtasks are:

- **Domain model :** is the Class in its primitive status. In this step there are three classes : Storage Class , MobileController Class and User Class.

  The next Figure illustrates domain model of the application.

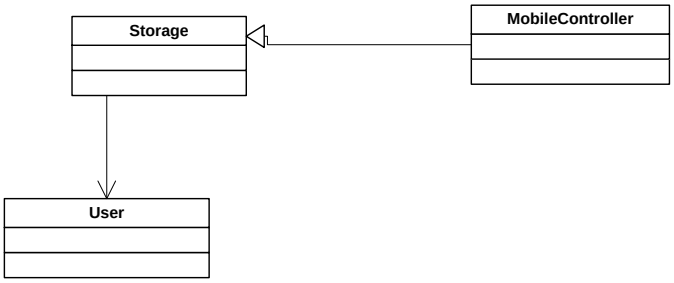


Figure 4.1 : The domain model of the application.

- **Use Case Diagram**

This step illustrates the roles of the proposed actors who interact with the application. This step also shows the most interactive user/s with the general functions of the system.

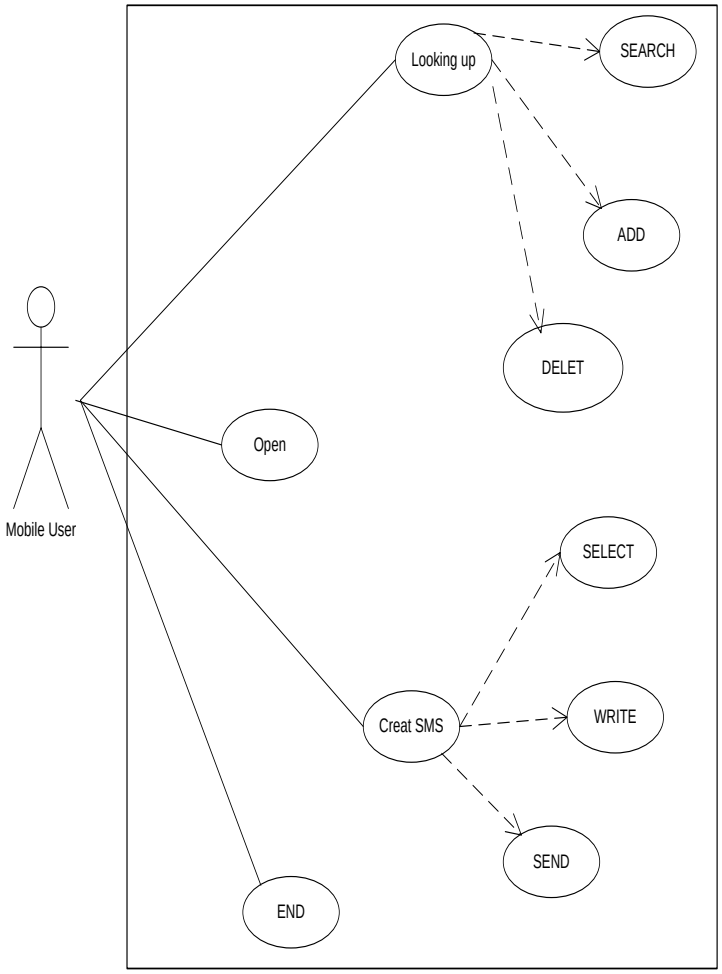


Figure 4.2 :Use case diagram of the application.

- **GUI Prototype**

It shows a prototype of each use case in the application. And these use case are:

1. **Use case: Start**

Figure 4.3:Prototype of the start use case

2. **Use case: Selection process**

Figure 4.4:Prototype of the Selection process use case.

3. Use case: Contacts Setting

Figure 4.5:Prototype of contacts setting use case.

4. Use case: Create a message

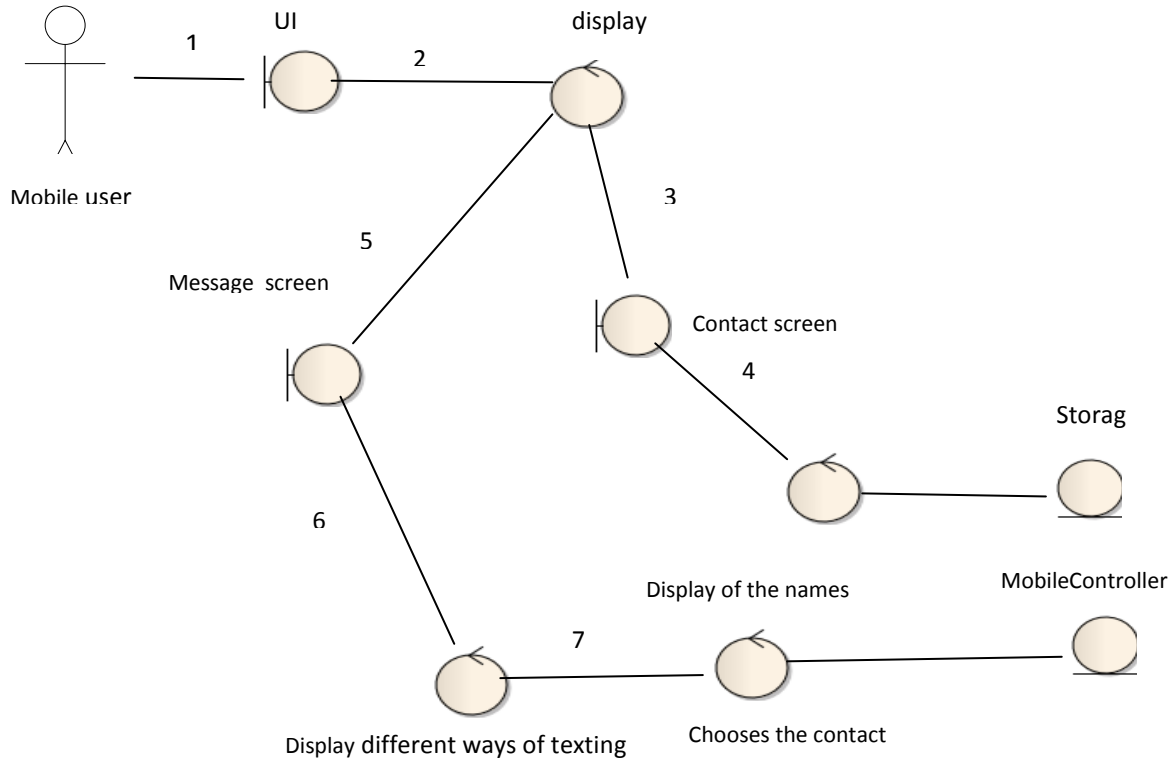
Figure 4.6:Prototype of the Create a message use case.

4.2.2 Step two: Preliminary Design Review

This step consists of two subtasks: Perform robustness analysis for each use case and update domain model.

- **Perform Robustness Analysis**

1. Robustness Analysis for Star



1-The application displays the Start Screen as shown in figure(4.3)

2-The user clicks on the open button

3-The application displays the choose screen between processes

4-The application displays Contact screen

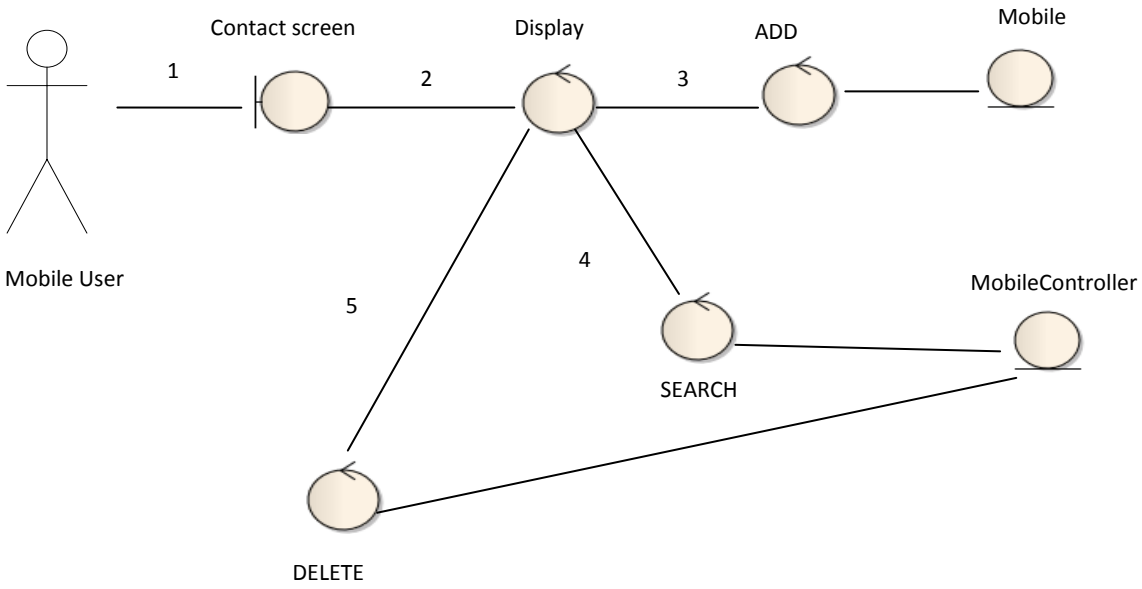
5- The application displays Message Screen

6-Select of typing

7-Choose a contact

Figure 4.7: Robustness Analysis for Star

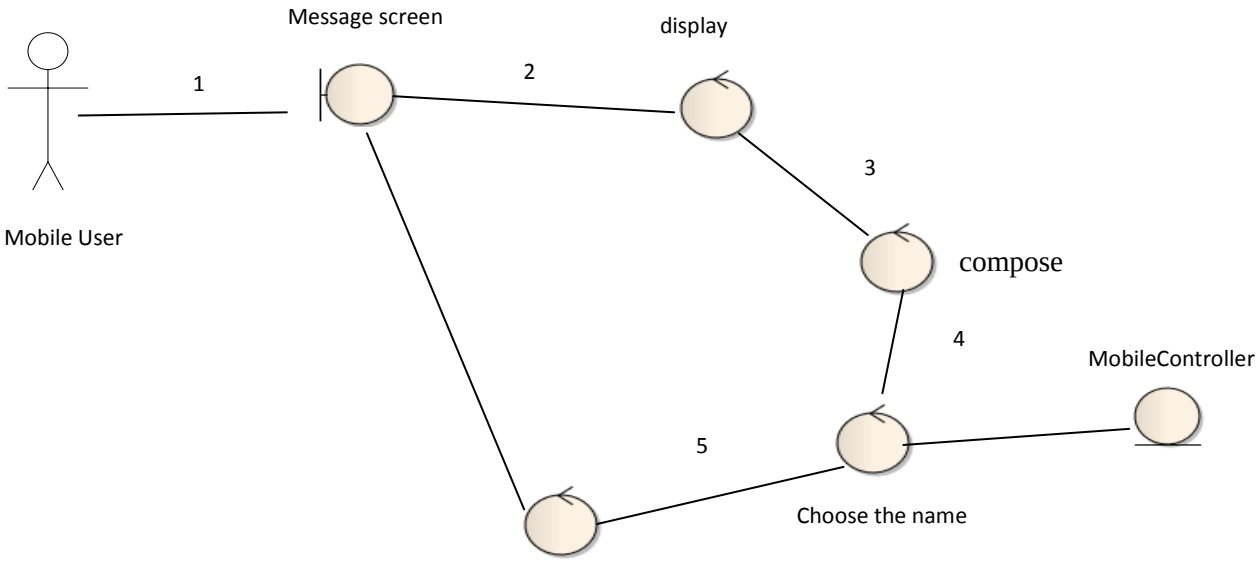
2. Robustness Analysis for contact setting



1- The user clicks on the contact button
2- The application displays the contact screen as shown in figure(4.5)
3- The user selects ADD process
4- The user selects SEARCH process
5- The user selects DELETE process

Figure 4.8 :Robustness Analysis for contacts setting.

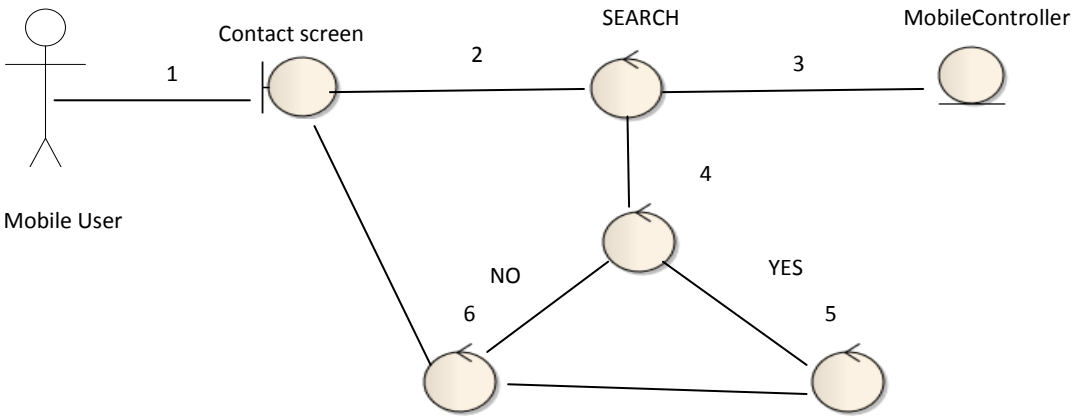
3. Robustness Analysis for Create Message



1- The user clicks the MESSAGE button
2- The application displays the message screen as shown in figure (4.6)
3- Select one of the typing options to compose SMS messages.
4- Choose the contact
5- Send the message

Figure 4.9: Robustness Analysis for Create Message.

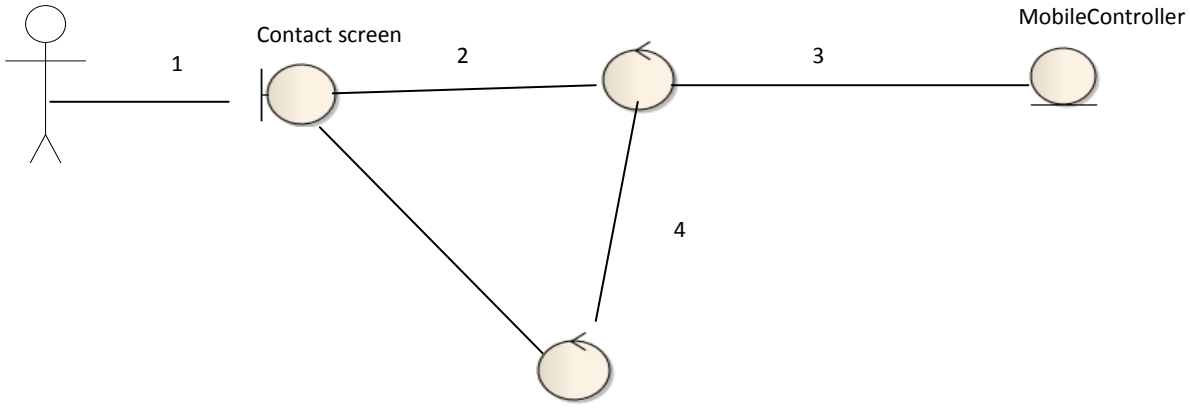
4. Robustness Analysis for Search.



1- The user clicks on the CONTACT button
2- Then, the user searches for Contact by entering contact names
3- The application executes the search of the name from MobileController class
4- The application tests (if the contact name found or not)
5- The contact is SELECTTED if found.
6- If not found, then the keyboard is automatically hidden

Figure 4.10: Robustness Analysis for SEARCH Contact.

5. Robustness Analysis for DELETE Contact



1-The user clicks on the CONTACT button
2-The application displays the Contact Screen
3-Select the contact name from the list box
4-Choose the DELETE option.

Figure 4.11: Robustness Analysis for DELETE contact

4.2.3 Step Tree: Detailed Design Review

This step divided into three subtasks : generate sequence diagram from boundary and entity objects on the robustness diagram ,choose a suitable design pattern and update the domain model into class diagrams as needed. These subtasks are:

- **Sequence Diagram**

Sequence diagrams illustrate the behavior allocation in timeline for each use case.

1. Sequence Diagram of the Start Screen

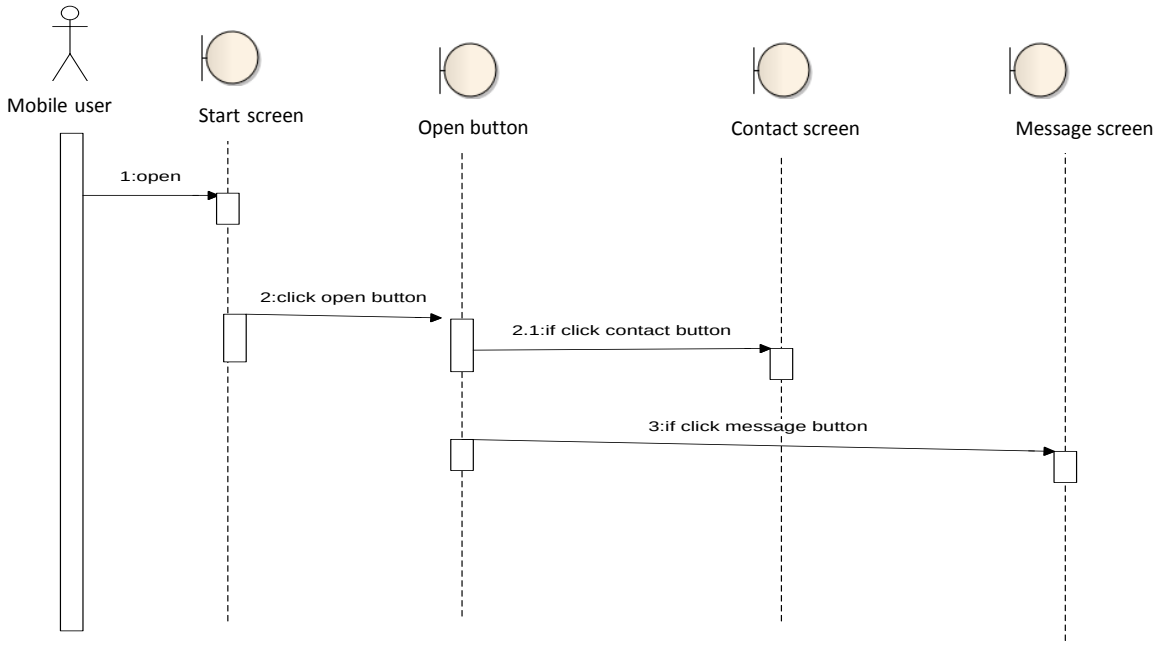


Figure 4.12: Sequence Diagram of the Start Screen.

2. Sequence Diagram of SEARCH Contact

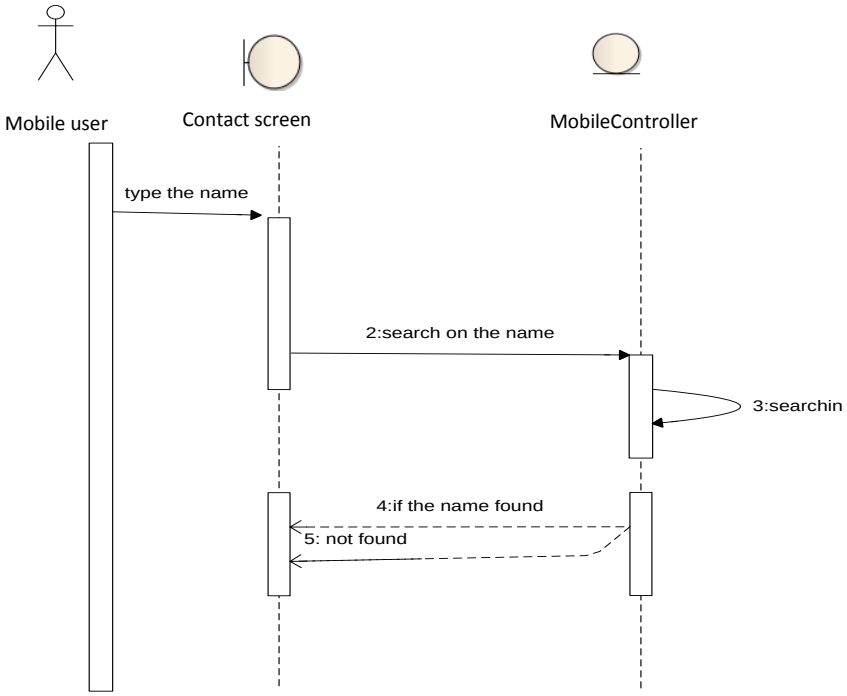


Figure 4.13: Sequence diagram of search contact.

3. Sequence Diagram of ADD Contact

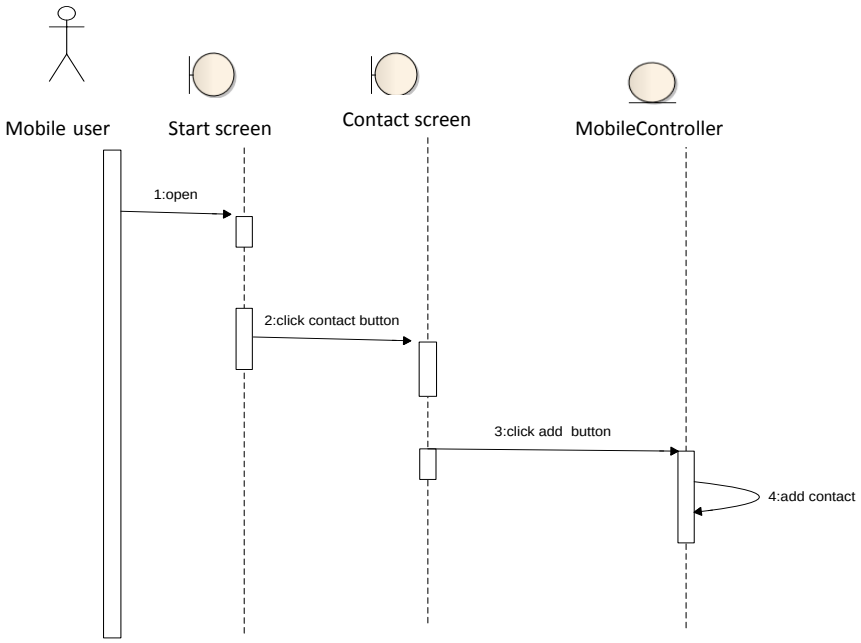


Figure 4.14: Sequence diagram of ADD contact.

4. Sequence Diagram of DELETE Contact

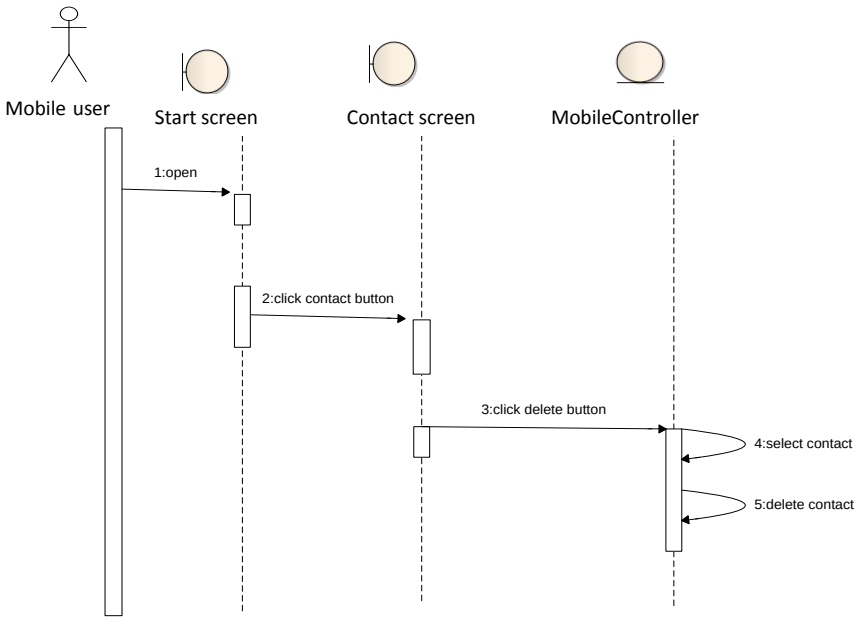


Figure 4.15: Sequence diagram of DELETE contact

5. Sequence Diagram of COMPOSE Message

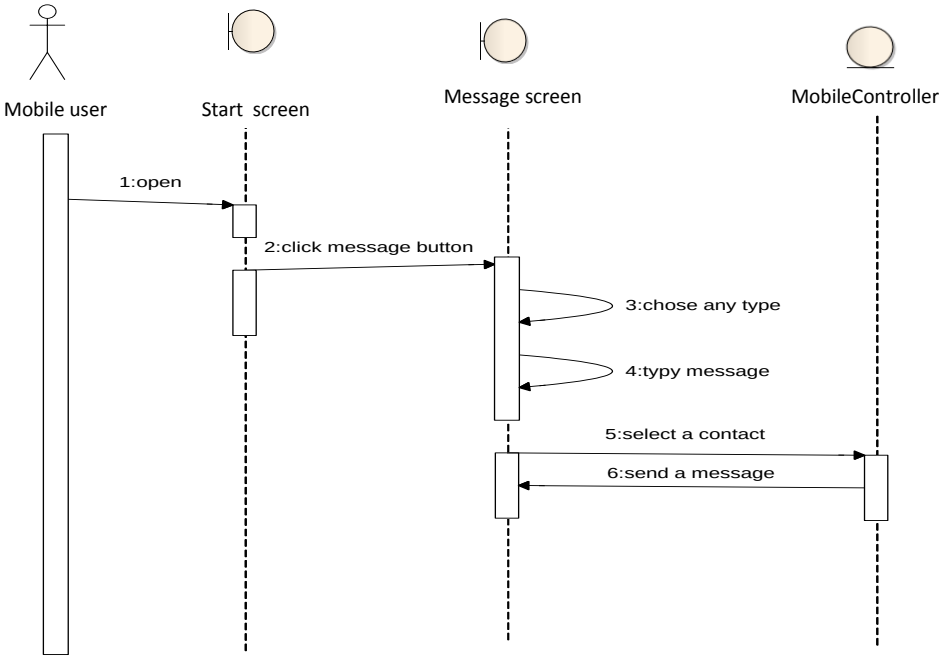
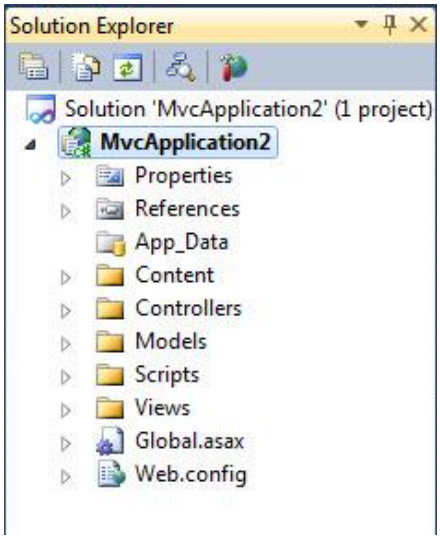


Figure 4.16 :Sequence diagram of COMPOSE message.

- **Design Pattern**

At this stage , we determine the appropriate pattern. As we stated earlier that the MVC pattern is the most suitable and most effective for the proposed methodology in this thesis. Also, the ASP.NET is selected because MVC is one of three main ASP.NET programming models. The most important components of MVC in ASP.NET are as follows:

MVC Folder: A typical ASP.NET MVC web application has the following content of all folders.



Application information

Properties
References

Application folders

App_Data Folder
Content Folder
Controllers Folder
Models Folder
Scripts Folder
Views Folder

Configuration files

Global.asax
Web.config

Figure 4.17: The MVC Folders.

The folder names are equal in all MVC applications. The MVC framework is based on default naming. Controllers are in the Controllers folder. Views are in the Views folder, and Models are in the Models folder. You do not have to use the folder names in your application code. Standard naming reduces the amount of code to make it easier for developers to understand MVC projects.

The table below describes each single folder and its function.

The Name Folder	Description
App Data folder	is for storing application data
Content Folder	is used for static files like style sheets (CSS files), icons and images. the file **Site.css** in the content folder. The style sheet file is used to edit or change the style of the application.
Controllers Folder	contains the controller classes responsible for handling user input and responses. MVC requires the name of all controller files to end with "Controller".
Models Folder	contains the classes that represent the application models. Models hold and manipulate application data.
Scripts Folder	stores the JavaScript files of the application.
Views Folder	The **Views** folder stores the ASP.NET files related to the display of the application (the user interfaces). And also it contains one folder for each controller.

Table 4.1: The Application Folders

The main operations that MVC contains :

1- Adding a Model

Folders of **Models** contains the classes that represent the application model.

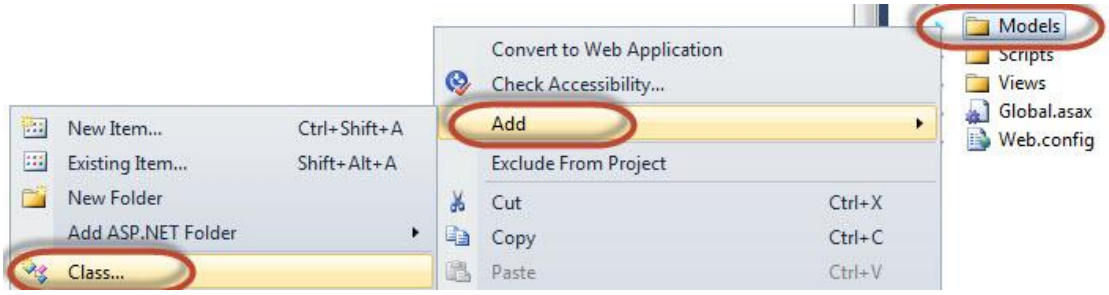


Figure 4.18: Adding a class into a Model

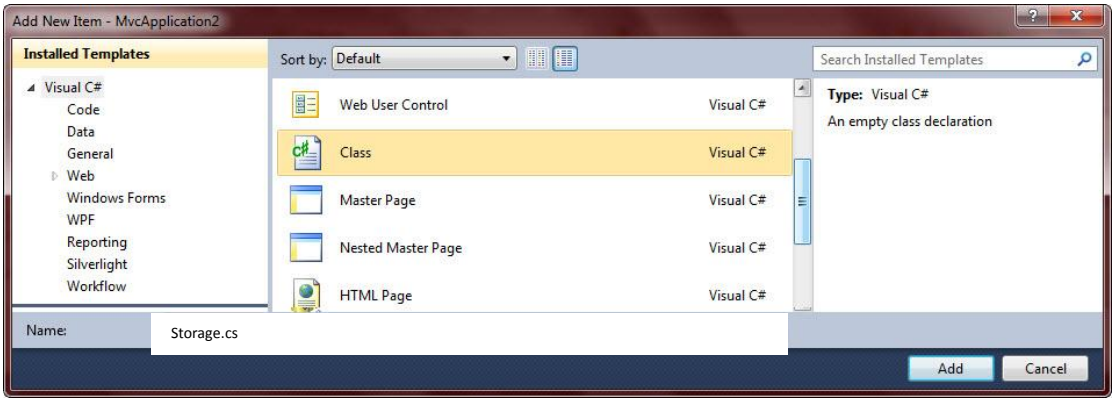


Figure 4.19: Create a Storage class into a Model .

2- Adding a Controller

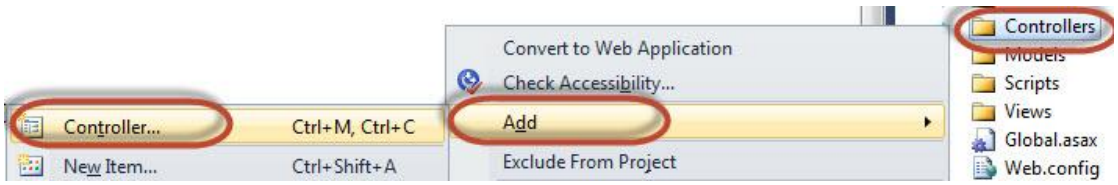


Figure 4.20: Adding a Controller.

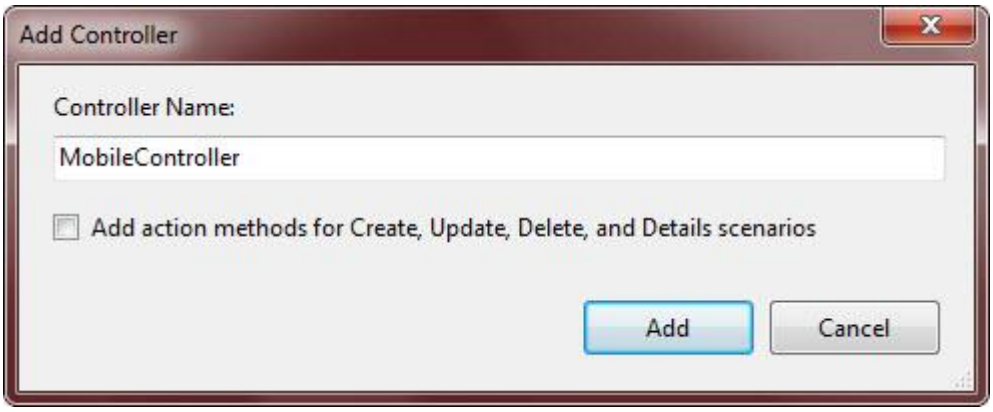


Figure 4.21: Adding a MobileController into Controller.

3- Adding Views for Displaying the Application

The Views folder stores the files (ASP files) related to the display of the application (the user interfaces). These files may have the extensions html, asp, aspx, cshtml, and vbhtml, depending on the language content. The Views folder contains one folder for each controller. Create a Mobile folder, and a Shared folder (inside the Views folder).

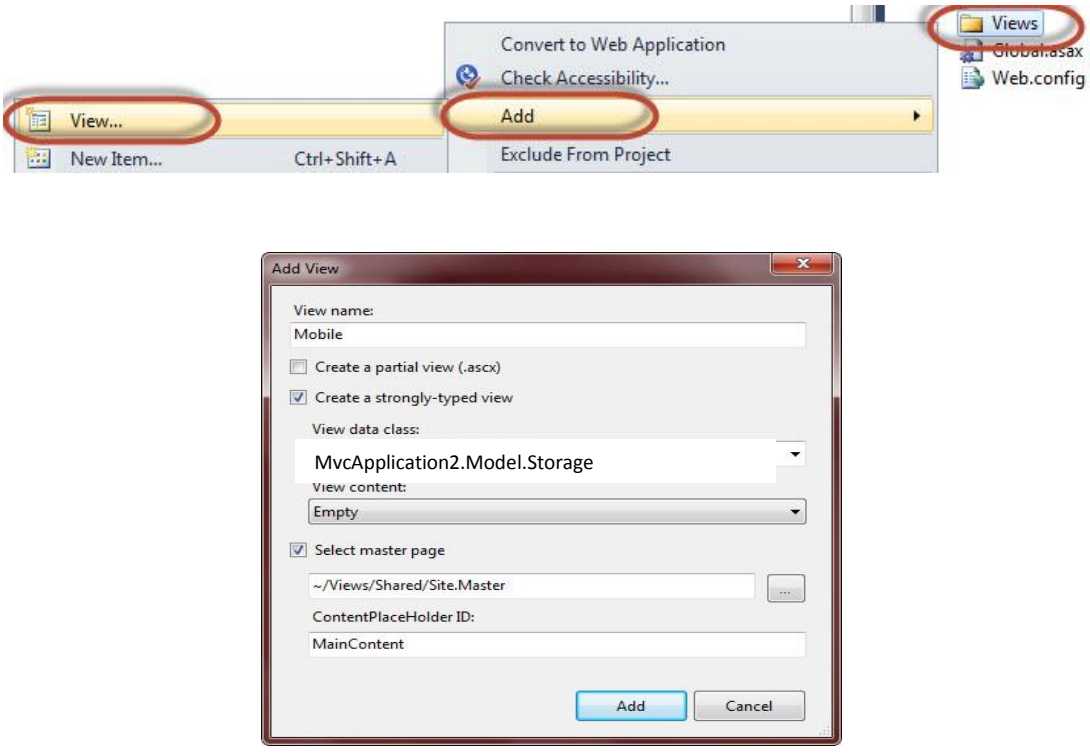


Figure 4.22: Adding a view.

The Mobile folder contains pages for Index. The Shared folder is used to store views shared between controllers (master pages and layout pages).

- **Class Diagram**

The system is analyzed based on the point of view of each single user. The most important classes in this application:

1. Storage Class

In this class, all variables are identified which will be used in the application . It is important that this class is inside the model folder.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace MvcApplication2.Models
{
    public class Storage

        {
            public List<string> NAMES { get; set; }
    }
    }
```

2. Mobile Controller Class

This class contains this function (ActionResult Index()) , which connects the Controller with the View. Also, an object is created from the variables defined in the class model folder which means adding a value to those variables.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using MvcApplication2.Models;

namespace MvcApplication2.Controllers
{ public class MobileController : Controller {

        public ActionResult Index()
        { Storage obj = new Storage();
            obj.NAMES = new List<string>()
        {
            "MOHAMED",
```

```
            "SANA",
            "SOHIB",
            "ABDALRAHMAN",
             "BASMA",
            "MUSTAPHA",
            "SAMAR",
            "AHMED",
            "NORA",
             "REIAM",
            "NONA",
            "MAHA",
            "FARH",
            "MAMA",
            "BABA",
            "ALI",
            "SOSO",
            "MEDO",
            "AHLAM",
            "SAMAR",
            "SALAH",
            "AHMAD"

        }; return View(obj); } }}
```

3. UI Class

This class is responsible for designing the user interfaces of the application. The designing of UIs relies on the HTML language. The HTML contains many functions which are written by JavaScript .This UI class and Site Class are both in the View folder , It is responsible for the action in the application**.**

4. MVC Application Class

This class contains the function of the RegisterRouter which helps in the routing process. This function helps identify URL structure and map the URL with the Controller. Example: http://localhost:2014/.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using System.Web.Routing;

namespace MvcApplication2
{ public class MvcApplication : System.Web.HttpApplication
    {
        public static void RegisterRoutes(RouteCollection routes)
        { routes.IgnoreRoute("{resource}.axd/{*pathInfo}");
```

```
            routes.MapRoute(
                "Default", // Route name
                "{controller}/{action}/{id}", // URL with parameters
                new { controller = "Mobile", action = "Index", id =
UrlParameter.Optional } // Parameter defaults
            );

        }

        protected void Application_Start()
        {
            AreaRegistration.RegisterAllAreas();

            RegisterRoutes(RouteTable.Routes); } }}
```

5. Site-Page_Load Class

This class provides an instant online application at (www.mobiletest.me)

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
 namespace MvcApplication2.Views.Shared
     {public partial class Site1 : System.Web.UI.MasterPage
    { protected void Page_Load(object sender, EventArgs e){} }
```

The next figure describes Classes Diagram of the application.

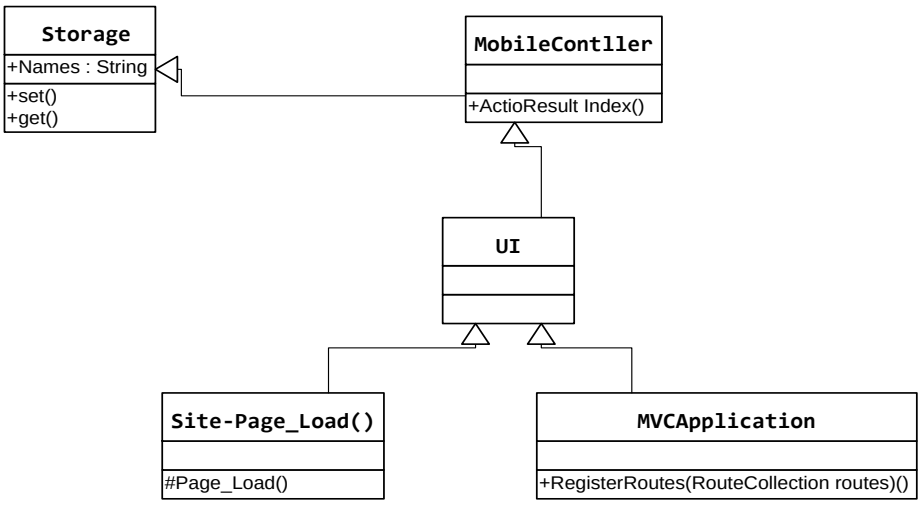


Figure 4.23: The class diagram of the application

4.5 Step 4: Implementation

This step introduces all the screens of the application as shown below. Samples from screenshots for the application**.**

Figure4.24:The Start Screen of the application

Figure4.25:The Select Process Screen of the application

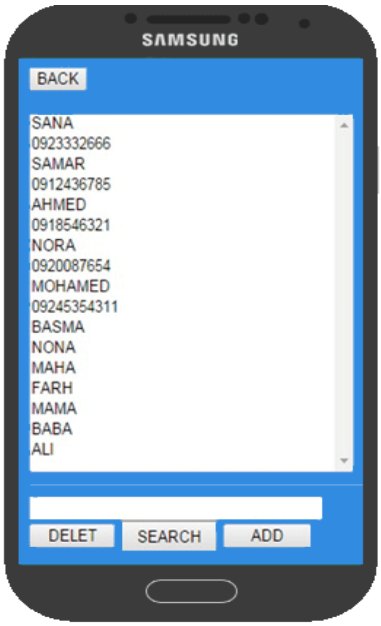


Figure4.26: Contacts Settings Screen of the application

Figure4.27:Message Screen of the application

Figure4.28: SMS1 Screen of the application

Figure4.29:SMS 2 Screen of the application

Figure4.30: SMS3 Screen of the application

CHAPTER 5

Conclusions and Future work

5.1 Conclusions

One of the most important findings in this research is that Design Patterns can solve specific design problems and make object oriented designs more flexible and reusable. They also help designers reuse several designs, including design alternatives to avoid other alternatives that might compromise reusability.

It was also concluded that the main objective of design patterns is to reuse good practice in the design of newly developed applications. Another important objective of using design patterns is to develop common applications and better understanding of the overall designing process which is performed by reusing the same generic names for implemented solutions.

This thesis has concluded that MVC patterns are considered as pioneering patterns for synchronizing user interfaces with domain data. It is actually an excellent choice for Web-based applications .In fact, Web structures naturally support the division of responsibilities of the components of MVC patterns . However, these patterns suffer from poor handling of view state logic, and assume decoupled View and Controller which does not match with many state of the frameworks in project.

Robustness Analysis helps discover objects for each use case and identify the main classes before designing or implementation , also it is the best way to analyze MVC because it represents three objects :Entity objects present classes, Boundary objects present links between the system and the external environment and Controller objects present logical software functions.

Finally, we used ASP MVC .NET framework to solve the problem. After applying ICONEX and MVC.

5.2 Future Work

The researcher will attempt to implement all of the proposed work on "actual mobile phones connected to Internet which can allow for more options for users such adding and editing photos, looking up contacts via emailing. Another future goal is to research for more up-to-date design patterns and find what other UI related problems can be solved by using design patterns.

Reference

[1] Bettina Biel, Thomas Grill, Volker Gruhn, “Exploring the benefits of the combination of a software architecture analysis and a usability evaluation of a mobile application”, *Journal of Systems and Software (JSS)* Vol 83(11), pp 2031-2044, 2010.

[2] Dragos Manolescu, Markus Voelter, and James Noble, "Pattern Languages of Program Design ", 1st ed.: Addison Wesley, 2006, vol. 5.

[3] www.gofpatterns.com/sitemap.php Acceded in [April_2013].

[4] wiki.sdn.sap.com/wiki/display/ABAP/UnitSu.GoF+Design+Patterns Acceded in[April_2013]

[5] Christopher Alexander, Sara Ishikawa, and Murray Silverstein, *"A Pattern Language: Towns, Buildings, Construction (Cess Center for Environmental)*", NewYork: Oxford University Press, 1977.

[6] Erich Gamma, Ralph Johnson, Richard Helm, and John Vlissides, "Design Patterns: Elements of Reusable Object-Oriented Software",Boston, MA, USA: Addison-Wesley Longman Publishing, 1995.

[7]www.intechopen.com/books/human_computer_interaction_new_developments/hci_design_patterns_for_mobile_applications_applied_to_cultural_environments Acceded [April_2013].

[8] www.wiki.sdn.sap.com/wiki/display/ABAP/UnitSu.GoF+Design+Patterns Acceded in [March _2013].

[9] www.javagyan.com/blogs/design-patterns Acceded in [March_2013].

[10] msdn.microsoft.com/en-us/library/ff649643.aspx Acceded [May _2013].

[11] Hojat A. Hasanvand and others ," Mobile Computing:Principles, Devices and Operating Systems ", World Applied Programming, Vol (2), Issue (7),pp 399-408 ,July 2012.

[12] Kot, Chelsea , "A Brief History of Tablets and Tablet Cases". Tablets2Cases. http://www.tablet2cases.com/wiki/about/history/.Retrieved December 10, 2011. of-personal-digital-assistants1.

[13] "History of the HP 95LX computer". HP Virtual Museum. Hewlett Packard. http://www.hp.com/hpinfo/abouthp/histnfacts/museum/personalsystems/0025/0025history.html.

[14] www.webopedia.com/quick_ref/mobile_OS.asp Acceded [May_2013].

[15] **Nilsson Erik G,**"Design guidelines for mobile applications", [Report] : SINTEF Report STF90 A06003. - Oslo : SINTEF Telecom and Informatics, 2005. - ISBN 82-14-03820-0.

[16] Eric Magnuson, "design patterns in user interface design"[project], Worcester Polytechnic Institute, Project Advisor Project Matthew Ward and Jeffrey LeBlanc ,2010.

[17] Amin A.Rasooli," Design patterns for user interface", Topic paper, CS5760 by Prof. Pastel, Spring 2012.

[18] Astahovs Ilja,"Use of design patterns for mobile game development"[project], Project Advisor Project Johan Eliasson ,spring 2012.

[19] Erik G. Nilsson,"Design Patterns for User Interface for Mobile Applications", Computer-Aided Design of User Interfaces VI, pp 307-312, 2009.

[20]www.techterms.com/definition/smartphone Acceded [March_2013].

[21] S. Alpaev .” Applied MVC Patterns. A pattern language”, presented at the Viking PLoP conference,2005.

[23] Joydip Kanjilal ,"Implementing the MVC Design Pattern in ASP.NET",article,31 Jan 2008.

[24] Brian Fling, "Mobile Design and Development: Practical Concepts and Techniques for Creating Mobile Sites and Web Apps", O’Reilly, 2009 .

[25] Marek Stępień, "WAP dla każdego," Helion, 2001.

[26] K.Wseem Abrar , Prof. R.M.Noorullah ," Comparative Study In Utilization Of Creational And Structural Design Patterns In Solving Design Problems", International Journal of Information Technology (IJIT), Volume – 1, Issue – 1, August 2012

[27] S. S. Suresh, Prof. Dr. M. M. Naidu and S. Asha Kiran.” Design Pattern Recommendation System (Methodology, Data Model and Algorithms)”. presented at International Conference on Computational Techniques and Artificial Intelligence (ICCTAI'2011).

[28] G. E. Krasner and S. T. Pope, “A cookbook for using the model-view controller user interface paradigm in smalltalk-80,” J. Object OrientedProgram., vol. 1, no. 3, pp. 26–49, Aug. 1988.

[29] Artem Syromiatnikov , Danny Weyns," A Journey Through the Land of Model-View-* Design Patterns",Topic paper , October-2013.

[30] M. Potel, “MVP: Model-View-Presenter The Taligent Programming Model for C++ and Java,” Taligent Inc, 1996.

[31] “GUI architectures,” http://martinfowler.com/eaaDev/uiArchs. html, 2006, [accessed October-2013].

[32] E. Gamma, R. Helm, R. Johnson, and J. Vlissides," Design patterns: elements of reusable object oriented software", Boston, MA, USA:Addison-Wesley Longman Publishing Co., Inc., 1995.

[33] G. E. Krasner and S. T. Pope, “A cookbook for using the model-view controller user interface paradigm in smalltalk-80,” J. Object Oriented Program., vol. 1, no. 3, pp. 26–49, Aug. 1988.

[34] ” Software design pattern”. Available :http://en.wikipedia.org/wiki/Software_design_pattern

[35]] S. S. Suresh, Prof. Dr. M. M. Naidu and S. Asha Kiran.” Design Pattern Recommendation System (Methodology, Data Model and Algorithms)”. presented at International Conference on Computational Techniques and Artificial Intelligence (ICCTAI'2011).

[36] The Taligent Programming Model for C++ and Java (1996) by Mike Potel, http://www.wildcrest.com/Potel/Portfolio/mvp.pdf.

[37]Introduction to Model/View/ViewModel pattern for building WPF apps (2005) by John Gossman, http://blogs.msdn.com/johngossman/archive/2005/10/08/478683.aspx

[38]WPF Apps With the Model-View-ViewModel Design Pattern (2009) by Josh Smith, http://msdn.microsoft.com/en-us/magazine/dd419663.aspx.

[39] S. Danturthi, “Comparative Study of Web Application Development with SQL Server and Db4o,” Master’s Thesis in Computer Science, Mlardalen University, Vasteras, 2011.

[40] S. Rakibul Hasan . “Developing an online store for a startup apparel business”. Bachelor's Thesis. Business Information Technology. April 2013.

[41] Tomáš Chlouba "Design Patterns in Mobile Architectures", Topic paper , University of Hradec Kralove, Rokitanskeho 62, Hradec Kralove, 500 03 Czech Republic, October 31, 2010..

[42]] P. Argall1, R. J. Sica1.” Development of a new Lidar Data Analysis Program”. The University of Western Ontario London ,2013.

[43] G. E. Krasner , S. T. Pope. “A description of the model-view-controller user interface paradigm in the smalltalk-80 system”. Journal of Object Oriented Programming, , 1988.

[44] A. Kolu.” MVC FRAMEWORKS IN WEB DEVELOPMENT”. Master thesis, University of JYVSKYLN, 2012.

[45]] E. Gamma, R. Helm, R. Johnson, and J. Vlissides , “Design Patterns: Elements of Reusable Object Oriented Software”, Addison Wesley, Boston, 1995.

[46]] L. D Í E Z.” Secure, scalable and component based Web shop using Struts and Hibernate”. Master of Science Thesis Stockholm, Sweden 2006.

[47] ” ICONIX Process “. available : http://iconixprocess.com/iconix-process/.

[48] ” Mastering UML with Enterprise Architect and the ICONIX Process”. available : http://www.iconixsw.com/eaiconixprocess.html [**2013].**

[49] F. Cover. D. Rosenberg, M Stephens .”*Use Case Driven Object Modeling* with *UML*: Theory and Practice” . Apress Berkely, CA, USA ©2007, Jul 31, - 472 pages.

[50] S. Mukhtar. 22 Aug 2004 .” Applying Robustness Analysis on the Model–View–Controller (MVC) Architecture in ASP.NET Framework, using UML”. Available: http://www.codeproject.com/KB/architecture/#General [1999-2013].

[51] J. Denham, George Heineman. “Entity, Boundary, Control as Modularity Force Multiplier”, 01/2009.

[52] K. Scott and D. Rosenberg. March 01, 2001” Successful Robustness Analysis”. Available : http://www.drdobbs.com/successful-robustness-analysis/184414712.

[53] Helen Sharp, Finkelstein and Galal Galal, ” Stakeholder Identification in the Requirements Engineering Process”, Publisher: IEEE, Conference: Florence, pp 387 – 391, No 6359086, 1999.

[54] G. E. Krasner , S. T. Pope. “A description of the model-view-controller user interface paradigm in the smalltalk-80 system”. Journal of Object Oriented Programming, , 1988.

[55] Ashish Shukla, “Overview of ASP.NET MVC”, article, Apr 25, 2011.

[56] Walter Zimmer," Relationships between Design Patterns",[topic paper], Forschungszentrum Informatik, Bereich Programmstrukturen, 2000.

[57] Anuar Lezama," Introduction to the mobile application development"[Master Thesis], Advisor: Josep Solé Pareta, Fotis Christodoulopoulos, November 2010 .

[58] E. Gamma, R. Helm, R. Johnson, and J. Vlissides , “Design Patterns: Elements of Reusable Object Oriented Software”, Addison Wesley, Boston, 1995.

[59] *K. A. Robbins.* “User Interfaces and Events”. Available: http://vip.cs.utsa.edu/classes/cs4773s2004/lectures/cs4773week4.html [*February 2, 2014*].