Reuse of Legacy Softwares in SOA, A Case Study of the “Next Version” Technique

Mohamed Gharzouli
Department IFA
Faculty of New Technologies of Information and Communication
University Constantine
Algeria
gharzouli@gmail.com

ABSTRACT: *The development and the improvement of Information Systems are important factors for the success of the economic development or administrative enhancement of any organization. To define new business models, the enterprises are required to progress their applications and integrate them in one homogeny information system. In this field, the integration of existing applications (especially legacy systems) with the new ones presents a complex challenge. In Algeria, many of Small and Medium Enterprises (SME) and administrative organizations use home developed software to progress their Information Systems. For this reason, the source codes of several legacy applications are generally available, which becomes the reuse of these applications easier and more possible. However, the main problem is related to the techniques and methods that can be used for integrating the existing applications in the new view of the information system. In this context, the most current architectural model is SOA (Services Oriented Architecture, where Web services present the main important technology to develop novel Web Information Systems. In this paper, we present a case study about the use of the “next version” technique to develop a Web service based application by reusing an existent legacy software.*

Keywords: Web Services, Legacy Software, Client Web Application, Reuse of Legacy Software

Received: 2 September 2013, Revised 4 October 2013, Accepted 9 October 2013

1. Introduction

In these last years, the use of Web services is considered to be a good solution for creating a homogeneous environment [11]. The web services architecture is based on three elements: the service provider, the discovering agency and the service requestor [18], [10], [11]. If a company wants to develop a system based on web services architecture, it defines the role of the service provider and those of the service requestor. So, the development of such application requires firstly the preparation and the construction of its contents by re-using existing applications [1], [12]. Among the exciting applications, we found some software those are not directly reused. They need generally a supplementary work to be invoked from new applications (Web applications). Based on our experiences about Web services technologies and their applications in many fields [7], [6], [5], [8], [4] and [3], in this paper, we describe a case study of a development of a web service by re-using an existing application. This example explains how we can develop initially the web service and then how to consume it through a client web application. In the following, we present the most important aspect that is related to the use of interoperable web services for computational Web information systems. After, we describe the various steps of our case study and we finish with a conclusion and some prospectives.

2. Web services for Web information systems

Recently,Web services are used to integrate heterogeneous systems such as e-commerce, e-services and e-learning [9]. To ensure the interoperability between various applications, Web services form an adequate and effective solution. A new information system, for an enterprise, needs to reach heterogeneous information sources and re-uses existing applications. Web services answer these constraints because they offer a high degree of interoperability. The three basic standards WSDL, UDDI and SOAP make possible the description, the publication and the invocation of Web services [16], [2], [17], [10]. The basic interaction of the Web services for a client application is based on a separation between the server that manages the user interface and the one that uses a particular service [13]. In the case of a web client application, the SOAP server can be internal i.e. located on an Intranet. In this case, the provided services can be invoked via SOAP by using the local network area. If the SOAP server is external, the interaction is made via an Extranet or the Internet network [19], [14], [15]. Generally, the internal systems (sub systems) are the departmental Intranets of the enterprise. Each one can contain one or several applications, which use various data sources. The external systems are generally provided by the partner companies. These systems are accessible through an Extranet or Internet, as they can be public web sites, which provide useful services and functionalities for the enterprise (for example, a purse service).

3. Reuse of existing softwares

To progress and to develop a new version of the enterprise information system, it is necessary to compare the functionalities provided by the existing applications; those are considered useful for the information system project and the awaited functionalities. This comparison enables us to specify the services to be added (software components, which realize certain missing functionalities) [8]. Some applications are not directly reusable. These applications require a phase of transformation (migration) or wrapping. The translation can be carried out manually and/or automatically from the source code into another language (java for example). However, the wrapping operation consists to create a transition layer (Wrapper), which allows the interaction with the source code. A good example is the creation of a java wrapper that uses Java Native Interface (JNI). We can apply this approach to FORTRAN, C and C++ code. Another possibility is that the awaited web services will be composed from other ones [7], [8]. This case requires the development of a business process engine. If this system already exists, this situation requires a study for the integration of a service in this system.

4. Development of the client application

Practically, this phase is decomposed into two principal levels: The first level is used to design the dynamic part of the system independently of the platform, which will be used for the implementation. For the external services, we need to develop local presentations (to measure) for certain external services. The second level makes the relation between the result of the first part and the software components used to implement the various presentations. In order to facilitate the task and to express the relationship with the implementation of the presentations, a programming pattern of the web applications should be used. The majority of current works propose the use of the MVC2 (Model, View, Controller) paradigm for designing the web applications, because it presents an important advantage which is the separation between the application model and the user interface.

5. Case study

In this section, we present an example relating to an application that manages the students Grade Point Average (GPA). Our objective is to re-use this application in the form of Web services which will be integrated in the information system. To achieve this goal, we use the "*next version technique*" to migrate the original software developed in VB6 to the VB.NET.

5.1 Implementation of theWeb service
The function Show_GPA (immat:String) (appeared in the following listing) belongs originally to an existing application developed in VB6. We re-used this application in a way where we carried out an operation of migration (automatically) towards VB.NET, and then we exposed this function like a web service. For this reason, we have used the Visual Basic Upgrade Wizard of visual studio .NET (figure 1).

The following VB code shows the function Show_GPA (immat:String):

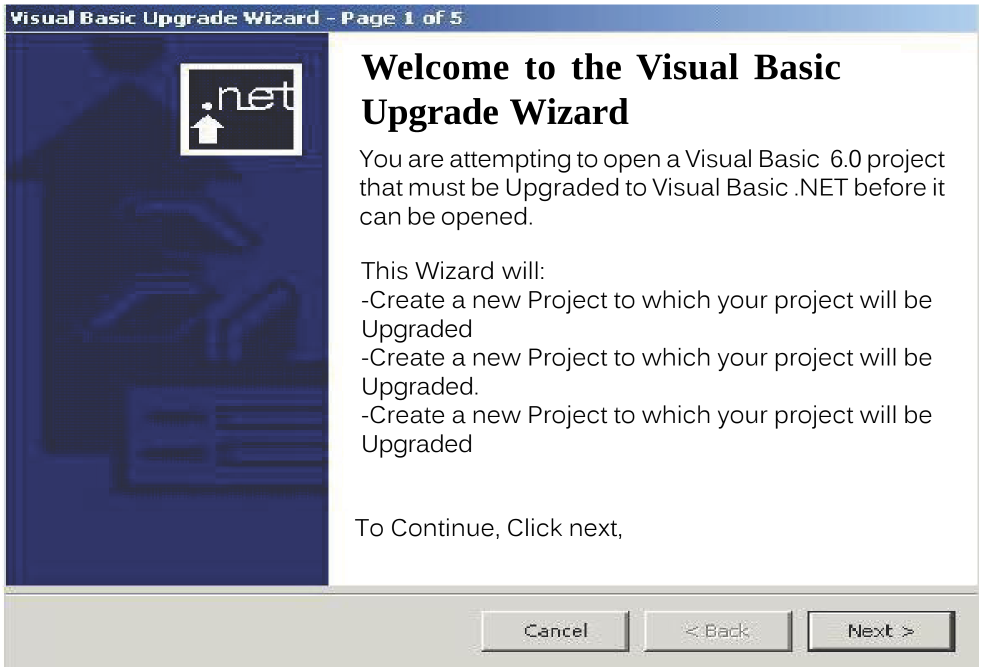
This function uses two ADO objects: "*Connection*" and "*Recordset*". The database connection is realized through the provider

```
Public Function Show\ _GPA (Text1 As TextBox) As String Dim conn As
ADODB. Connection Dim rst As ADODB. Recordset Set conn = New
ADODB. Connection Set rst = New ADODB. Recordset conn . Open
“Provider  = Microsoft.Jet .OLEDB. 4. 0; Data
Source = . . . .\vb6/database.mdb;
Persist Security Info = False”

Sql query . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

Public Sub form_Load ()
Form1.Caption  =  “Show_GPA of a student” Form1. Label2. Caption = Show\ _GPA (Text1)
Form1.Refresh
End Sub
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
Public Sub Command1_Click ()
Unload From1 conn.Close Set conn = Nothing
End Sub
```

Listing 1. VB6 code of Show_GPA function

“*Microsoft.Jet.OLEDB*.4.0”. To transform the VB code toward Vb.NET automatically we can use the visual studio .NET environment, which verifies the VB version and opens the textbox that proposes the migration. This last is realized in five successive steps. The first one is shown in figure 1. The second textbox proposes two kinds of final applications: EXE or DLL. In our case, we choose the EXE one. After third and fourth steps, the last one shows the execution of the conversion. If this operation is finished correctly, the visual studio doesn’t produce any error message. Thus, the result folder of the conversion obtains the VB.NET project, which includes many files (.vb and .vbproj,...) and two subfolders “*bin*” and “*obj*”. Among the output files, we find and HTML one, that contains the detailed rapport of the migration operation. The main file in our case is that contains the VB.NET code of the function Show_GPA. However, this last can include some errors those marked by the following comments:

UPGRADE_WARNING, UPGRADE_ISSUE, UPGRADE_TODO or UPGRADE_NOTE.


Visual Basic Upgrade Wizard - Page 1 of 5

Welcome to the Visual Basic Upgrade Wizard

You are attempting to open a Visual Basic 6.0 project that must be Upgraded to Visual Basic .NET before it can be opened.

This Wizard will:
-Create a new Project to which your project will be Upgraded
-Create a new Project to which your project will be Upgraded.
-Create a new Project to which your project will be Upgraded

To Continue, Click next,

Figure 1. Basic Upgrade Wizard of visual studio .NET

```
Public Function Show\ _GPA (ByRef Text1 As
System.Windows. Forms. TextBox) As String Dim ADODB As Object
‘UPGRADE\ _ISSUE: ADODB. Connection object was not upgraded.Click for more:
‘ms” help://MS.VSCC/commoner/redir/redirect.htm? keyword = “vbup2068” ’
Dim conn As ADODB. Connection ’UPGRADE_ISSUE: ADODB. Recordset object was not
upgraded.Click for more:
‘ms” help://MS.VSCC/ commoner /redirredirect.htm? keyword = “vbup2068”’
Dim rst As ADODB. Recordset conn = New ADODB. Connection rst = New
ADODB. Recordset ’UPGRADE\_WARNING: Couldn’t resolve default property of object conn.
Open. Click for more:
‘mshelp://MS.VSCC/commoner/redir/redirect.htm? keyword = “vbup1037” ’
conn. Open (“Provider = Microsoft.Jet.OLEDB. 4. 0;
Data Source = . . . .\ vb6\database.mdb; Persist Security Info = False”)
‘UPGRADE\_WARNING: Couldn’t resolve default property of object
rst.Open. Click for more:
‘ms” help: //MS.VSCC/commoner/redir/redirect.htm? keyword = “vbup1037” ’
rst.Open (“SELECT first “name, last\ _name, section FROM student Where
immat = Text1. Text”, conn)’ UPGRADE\_WARNING: Couldnt resolve
default   property of  object rst.EOF. Click for more:
‘ms” help://MS.VSCC/commoner/redir/redirect.htm? keyword = “vbup1037”’
While Not (rst.EOF) ‘UPGRADE_WARNING: Couldn’ t resolve default property of
object rst ().Click for more :
‘ms” help://MS.VSCC/commoner/redir/redirect.htm?keyword = “vbup1037” ’
. . . . . . . . . . . . .
‘UPGRADE\_WARNING: Couldn’t resolve default property of object rst ().Click for more.
MoveNext.Click for more:
‘ms” help://MS.VSCC/commoner/redir/redirect.htm? keyword = “vbup1037”’
rst.MoveNext ()
End While
End Function
```

Listing 2. VB.NET code of Show_GPA function

Every comment is followed by the reason of the error with an URL to the complete help, which explains how we can resolve the problem. The following listing shows the VB.NET of the function Show_GPA with the generated errors.

In this code, it exits two “*UPGRADE_ISSUE*” errors, that occurred during the conversion of the “*Connection*” and “*Recordset*” objects from the ADO to ADO.NET and five “*UPGRADE_WARNING*” errors, those appeared as consequences of the two past “*UPGRADE_ISSUE*” errors. The comment “*UPGRADE_ISSUE* ” signifies that there are some lines, which avoid the compilation of the code. Whereas, UPGRADE_WARNING errors indicate the compiled lines which can cause errors in the runtime. So, it remains to correct the migration errors to obtain the correct VB.NET code of the function Show_GPA. In our case, the problem is solved as follows:

• Replace the “*Connection*” object by “*OleDbConnection*” associated to the data provider OLE DB.

• Replace the “*Recordset*” object by the ADO.NET object “*DataSet*”. “Add an object “*OleDbDataAdapter*” that makes the connection between the “*DataSet*” object and the database source.

• Import the namespaces “*System.Data*” and “System.Data.OleDb” to can use the new objects “*OleDbConnection*”, “*DataSet*” and “*OleDbDataAdapter*”.

• Replace the passage mode and the parameters type in the input “*ByRef Text*1 *As System.Windows.Forms.TextBox*” by “*ByVal matriculation As String*” and the output parameter by “*DataSet*”.

The data provider “*Microsoft.Jet.OLEDB*.4.0” and the database source will be reused without modifications. The corrected VB.NET code of the function Show_GPA is the following:

```
System. Data. OleDb . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
Public Function Show_GPA (ByVal matriculation As String) As DataSet
Dim conn As New OleDbConnection (“Provider = Microsoft. Jet. OLEDB. 4. 0;
        Data Source = . . . . . . \ vb6 \ student.mdb ;
        Persist Security Info = False”)
Dim mycommand As New OleDbDataAdapter (“SELECT. . . FROM student Where
immat = ‘“ + matriculation + “ ’ ”, conn)
        conn.Open ( )
        Dim rst As New DataSet (“student”)
        mycommand. Fill (rst, “student”)
        Return rst
End Function
```

Listing 3. Corrected VB.NET code of Show_GPA function

5.2 Reusing and exposition of the function as a Web service

The reutilization of the VB.NET code needs to copy the corrected code to new file with an extension “*.asmx.vb*”.Thus, the last step is to expose the function by the new Web service. To realize this operation, we just put the attribute <WebMethod()> before the function Show_GPA. The following listing shows the exposition of the function Show_GPA by the Web service “*service*1”:

```
Imports System.Web.Services
Imports System Imports System.Data
Imports System.Data.OleDb
<WebService (Namespace := “http://temp uri.org/”)> _
Public Class Service1 Inherits System.Web.Services.WebService
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
<WebMethod (Description := “Show\ _GPA . . . . .”, Enable Session := False)>
Public Function Show\ _GPA (ByVal matriculation As String ) As DataSet
Dim conn As New OleDbConnection (“ Provider = Microsof t.Jet.OLEDB. 4. 0;
Data Source = . . . . . . \ vb6\student.mdb ;
Persist Security Info = False”)
Dim mycommand As NewOleDbDataAdapter (“ SELECT . . . . . FROM student
Where immat = ‘ “ + matriculation + “ ’ ”, conn)
conn.Open ()
Dim rst As New DataSet (“student”)
mycommand.Fill (rst , “student”)
Return rst
End Function
End Class
```

Listing 4. Exposition of the function Show_GPA as a method of the Web service

The listing 5 contains the WSDL description of the implementation of the Web service “*sevice*1”. This description is made by two tags: <service> and <port>.

5.3 Implementation of the Client Web service

For the requestor role, there are two principal steps:

• Generation of a proxy class from the WSDL description of the service.

• Development of a presentation of the service that uses the proxy class to invoke the methods exposed by the service.

5.3.1 Generation of the proxy

The proxy is a bridge between the Web application and the Web service. To develop a proxy, we can use some tools of the .NET SDK or the “Web service Proxy Generator of the WebMatrix environment. The .NET SDK proposes the program wsdl.exe, which is used in command invite as the following:

```
<service name = "Service 1"> <port name = "Service 1 Soap" binding = "s0:Service 1 Soap">
<soap:address location = "http://myserver.com/alias/Service1.asmx"/>
</port>
<port name = "Service 1 HttpGet" binding = "s0: Service1HttpGet">
<http:address location = "http: //myserver.com/alias/Service 1.asmx"/>
</port>
<port name = "Service 1Http Post" binding = "s0: Service1Http Post">
<http:address location = "http://myserver.com/alias/Service1. asmx"/>
</port>
</service>
</definitions>
```

Listing 5. WSDL implementation description of the Web service

```
wsdl.exe http://myserver.com/alias/service 1.asmx?WSDL/n: Proxy
/l:cs/o: Proxy.cs/n:Proxy (namespace), /l:cs (the used language is
CSharp), /o:Proxy.cs (the output file is Proxy.cs).
```

Listing 6. First step of the generation of the proxy

Then, to call the proxy by the clientWeb application, the resulted file (Proxy.cs) will be compiled to a DLL library. To do this operation, we use the .NET SDK C Sharp Compiler (CSC.exe) as follow:

```
csc.exe/target : library/out : Proxy.dll Proxy.cs
```

The same past operations can be realized by using the Web service proxy generator of WebMatrix environment. The following figure shows its interface.

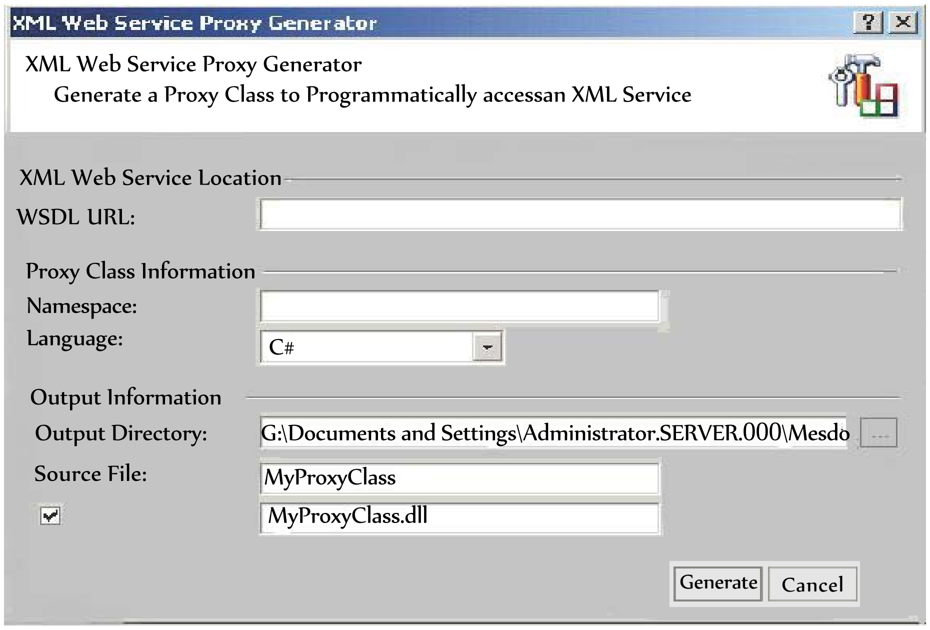
5.3.2 Development of the client Web application

In this step, we will develop a client web application for the precedent Web service by using the generated Proxy. For this, we have copied the file Proxy.dll in the bin directory of the Web application "*webclient*". The code of thisapplication is presented as follow:

```
<\%@ import Namespace =  "Proxy"\%>
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
<script run at = "server">
string mat;
public void Submit _Click (Object sender, EventArgs E) {
mat = (matriculation.Text) ;

Proxy.Service 1 myService = new Proxy. Service 1 (); DataSetdt = myService. Show_GPA (mat);
student _DataGrid. DataSource = dt.Tables ["student"]. DefaultView;
student _DataGrid. DataBind ();
}
```

Listing 8. Code of the Client Web application

The precedent listing describes how the client web application uses the generated proxy. The directive <%@ import Namespace = "*Proxy*" %> permits the importation of the local proxy library from the "*bin*" directory. This importation we permits the use of the class proxy "*service*1" implanted by the namespace "*Proxy*" (Proxy.Service1 myService = new Proxy.Service1 ();). Thus, we can invoke the function Show_GPA described in the proxy class "*service*1". Finally, we write the HTML source file, which is the interface used by the end user. The following listing presents thisHTML code:


XML Web Service Proxy Generator
XML Web Service Proxy Generator
Generate a Proxy Class to Programmatically accessan XML Service
XML Web Service Location
Proxy Class Information
Output Information
Generate
Cancel

WSDL URL:
Namespace:
Language: C#
Output Directory: G:\Documents and Settings\Administrator.SERVER.000\Mesdo
Source File: MyProxyClass
✓ MyProxyClass.dll

Figure 2. Web service proxy generator of WebMatrix

```
<html> . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . <body>
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . <form id = “Form1” run at = “server”>
<div. . . . . . . .>
immatriculation number : . . . . . . . .
<br/> <? xml: namespace prefix = asp//>
<asp: TextBox id = “matriculation” run at = “server” Text = “ ”> </asp:TextBox>
<br/>
<input id = “Show\ _GPA” type = “submit” value  = “Show\ _GPA”
name = “Show\ _GPA” run at = “server” on server click = “Submit\ _Click”/>
<p>
<ASP: DataGrid id = “student _DataGrid ” run at = “server”
. . . . . . . . </ASP: DataGrid>
. . . . . . . . . . . . . . . .
</ html>
```

Listing 9. View of the client web application

Finally, the MVC2 model in this web client application is presented as follow: the View is implanted by the file WebForm.aspx, the controller is the code behind included in the fileWebForm1.aspx.cs and “*Proxy.DLL*” is the model.

6. Conclusion

In this paper, we described a case study of an application, which integrates a legacy system as a web service. This process is based on two levels. The first level concerns the preparation and the construction of the Web service within exciting applications. The second level describes the development of a client application. For the first part, we Re-use an existing application, developed in VB6, as a Web service. The objective of this phase is to demonstrate how we can use a code of a legacy system to develop a new one, which has a compatibility with the web services technologies. The second part shows how to develop the client web application to consume the new developed Web service. We described the different steps for generating a proxy, and we explained how this last cab be used by the client application to communicate and invoke a distant web service. We used in our work the visual studio .NET platform and other Microsoft technologies like WebMatrix. But there remain some aspects that are not studied in this work especially those that are related to the business processes of the enterprise. In this case, in more with SOAP, UDDI and WSDL we can use other standards dedicated to the realization ofWorkflow like BPEL. Moreover, we will demonstrate, in the future, the installation of our client application on different mobiles devices.

References

[1] Brittenham, P. (2001). Web services development concepts. http://www.csd.uoc.gr/ hy565/newpage/docs/.../wsdc.pdf.

[2] Chauvet, J. M. (2002). Services Web avec SOAP, WSDL, UDDI, ebXML. Eyrolles.

[3] Benmerzoug, D., Gharzouli, M., Zerari, M. (2013). Agent interaction protocols in support of cloud services composition. In HoloMAS 2013, LNAI 8062, p. 293–304.

[4] Gharzouli, M., Boufaida, M. (2004). A development process of enterprise portal based on the web services. *In:* Proceedings of the The international Arab conference on Information Technology (ACIT).

[5] Gharzouli, M., Boufaida, M. (2009). A generic p2p collaborative strategy for discovering and composing semantic web services. *In*: Proceedings of the Fourth International Conference on Internet and Web applications and Services (ICIW), p. 449–454.

[6] Gharzouli, M., Boufaida, M. (2010). A distributed p2p-based architecture for semantic web services discovery and composition. *In:* Proceedings of the 10th Annual International Conference on New Technologies of Distributed Systems (NOTERE), p. 315–320.

[7] Gharzouli, M., Boufaida, M. (2011). Pm4sws: A p2p model for semantic web services discovery and composition. Advances in Information Technology, Special Issue on Advances in P2P Technology, 2 (1).

[8] Gharzouli, M., Boufaida, M., Seinturier, L. (2008). Reuse of existing applications during the development of enterprise portals integrating web services. *In*: Proceedings of the 10th Maghrebian Conference on Information Technologies (MCSEAI).

[9] Guillermo, J., Javier, E. (2006). Implementation of an e-services hub for small and medium enterprises. *In*: Proceedings of the AICT/ICIW.

[10] Kreger, Heather. (2001). Web services conceptual architecture (wsca 1.0). http://www.cs.uoi.gr/zarras/mdwws/WebServicesConceptualArchitectu2.pdf.

[11] Newcomer, E., Champion, M., Ferris, C., Orchard, D. (2002). Web Services Architecture.http://www.w3.org/TR/2002/WD-ws-arch-20021114.

[12] W-J Van Den Heuvel M. P. (2006). Papazoglou. Service-oriented design and development methodology. *International Journal of Web Engineering and Technology,* 2 (4).

[13] Fox, G. M. , Kurt, Steve M., Pierce, MYoun, C., Ozgur, B. (2002). Interoperable web services for computational portals. *In*: Proceedings of the IEEE/ACM Conference on High Performance Networking and Computing.

[14] Parsons, D., Newnham, J (2006). A web services architecture for rich content mobile learning clients. *In:* Proceedings of the 17th Australasian Conference on Information Systems.

[15] Prinz Satish, Wolfgang, Srirama, Narayana, Jarke, Matthias. (2010). Mobile web service discovery in peer to peer networks. http://arxiv.org/ftp/arxiv/papers/1007/1007.3631.pdf.

[16] SOAP specification. (2007). http://www.w3.org/tr/soap/ .

[17] WSDL specification. (2007). http://www.w3.org/tr/wsdl20 .

[18] Tellmann, R. (2003). Analysis of web service solutions and frameworks. http://swws.semanticweb.org.

[19] Wu, T., Zheng, P. (2006). Master of Science Thesis: Developing a Web Service Based Application for Mobile Client. PhD thesis, Stockholm, Sweden.