Web Service for Calculating the Probability of Returning a Loan – Design, Implementation and Deployment

Julian VASILEV
Varna University of Economics, Varna, Bulgaria
vasilev@ue-varna.bg

The purpose of this paper is to describe the process of designing, creating, implementing and deploying a real web service. A basic theory approach is used to analyze the implementation of web services. An existing profit model is used. Its business logic is integrated within a web service. Another desktop application is created to demonstrate the use of the recently created web service. This study shows a methodology for fast development and deployment of web services. The methodology has wide practical implications – in credit institutions and banks when giving a loan. This study is the first of its kind to show the design, implementation and deployment of a web service for calculating the probability of returning a loan. The methodology may be used for the encapsulation of other business logic into web services.
Keywords: *Web Services, Delphi XE, Web Service Development, Logistics Distribution Function, Probit Model, Source Code*

1 Introduction

Web application development is a difficult task. A lot of books are published on HTML, CSS and other web technologies. Serious practical examples are available. Libraries and book stores provide easy access to really heavy books on ASP, ASP.NET, AJAX, C#, PHP. The race to building web sites starts from writing source code in text editors and exceeds to writing web application in visual development environments.
Web services provide communication between different software applications, written in different languages. Web services are a useful instrument for creating business logic within custom functions. These functions are open to other users and developers. They may integrate the outer business logic without writing source code, but directly using web services. A web service may be used to read data from a database. It may be used to make a calculation. Or it may be used to send data to a database.
Web services use a common interface, described in web service description language (WSDL). Each web service has a description in the form of an XML file. A web service provides an interface. Several functions may be defined within the interface. The Simple Object Access Protocol (SOAP) is used to send data from an application to a web service and vice versa.
The life nowadays means providing not only content by static and dynamic web pages but providing custom business logic in the form of web services. A lot of publications focus on the result of using web services. The purpose of this article is to describe the design, implementation and deployment of a real web service. It provides a technological solution for easy and fast web service development using visual tools for web service development provided in Delphi XE. Web service development with Delphi XE looks like creating a desktop application. The developer does not need to write any HTML or XML code.
Web services allow the complex business logic to be separated into several stand-alone modules. Each module is created separately. All created services may work together in a big information system. Web services allow the creation of distributed web or desktop application. Separating the business logic into different servers and different web services allows programmers to create quickly a complex software application. Big software applications may perform faster than traditional ones when the business logic is separated into different servers and modules. In this way load balancing of hardware and software is achieved.

DOI: 10.12948/issn14531305/18.4.2014.10

Web services may be accessed by any computer connected to internet. Web services may be installed on an intranet server. In this way we get corporate web services. The user of a web service needs to know the input, the output parameters of a web service and its business logic. The business logic may be described as text or by a diagram. Web services work by communicating with simple messages. Messages are described in the SOAP standard. The messages are sent and received mainly through the protocol HTTP.

Some web services are free to use. See the airport weather check. See the example on the Microsoft web site [14].

A lot of examples for building web applications are available in internet. Few examples are given for building web services with Delphi XE. Information technologies provide a powerful instrument for web services development. Web services are used all over the world. That is why end users need knowledge not only on using web services, but creating new ones. This article provides newcomers some secrets of rapid web service development and deployment.

Giving loans is a risky operation. That is why credit institutions (including banks) need metrics to evaluate customers. These metrics are used for the decision making process of giving a loan. Since a loan may be given or not binary models may be used. Several binary models are known in econometrics. One of the most common models is the linear probability model. The logistics distribution function (LDF) describes the probit model. Probit models are widely discussed in econometrics literature. Another article of the author of this study proves that the probability of returning a loan may be calculated with the following formula.

$$P = e^z / (1 + e^z)$$

Where $z = B_1*X_1 + B_2*X_2 + B_3*X_3 = -0.025*X_1 -0.056*X_2 + 0.02*X_3$

X_1 is the month of birth, X_2 is the region and X_3 is the sum of contract.

People who live in the town of the credit institution have region "1" and those who live at the most remote places – "16". So the region coding shows the proximity to the credit institution with a numeric value on an ordinal scale. The previous article of the author proves that: (1) the month of the contract cannot be an independent variable in a binary logistics model, (2) The year of the contract cannot be an independent variable in a binary logistics model, (3) The age cannot be an independent variable in a binary logistics model, (4) The gender cannot be an independent variable in a binary logistics model.

2 Literature review

Complex web applications need to communicate through web services. Strimbei [1] proves the need of web services between front-end and back-end services. He offers a common platform to standardize access across heterogeneous data source types. The Service Data Objects (SDO) architecture allows the standardization of communication within the SOA stack.

Dospinescu and Perca [2] say that web services have proved their stability over the years. They offer the adoption of SOAP and REST architectures in mobile applications. Web services are engaged in B2B applications. Even though there are a lot of advantages of both architectures (SOAP and REST) Dospinescu and Perca describe their proper application in business. They prove that their web service may be configured to return a number of different results, depending on the input parameters. The result of a web service may be an array list.

Mobile applications may use special web services, for instance for login to a software system. A software system may interact with a web service. The web service communicates with the database. Such a communication offers Horia [3]. Horia describes several web services. Some of them are oriented to web login, others – for retrieving the list of orders, others – for placing orders. As a result the time of making orders is reduced. Customers of a restaurant may make orders by talking to the waiter or by using their mobile phones.

E-services have different aspect in economics. Walczak and Polkowski [4] analyze the use of e-services in Poland. E-services in medicine are connected with information about medical services and using patients' data online. Walczak and Polkowski offer the use of private and public cloud for the integration of web sites of health institutions. Web services allow the integration between the conventional information systems and the knowledge based information systems.
Cloud and mobile computing technologies may improve the communication between persons and organizations. Khanna and Bhagat [5] introduce virtual law offices. By using web and mobile applications everybody may receive judicial data after personal authentication. Case management is helped by an e-library system, working with web services.
Web services have wide application areas. Pocatilu [6] describes the creation of a mobile application for iOS with web services. Web services are used for authentication of users and retrieving of content. Web services are based on SOAP and WSDL. They are implemented by the using of .NET technology. Several specialized web services are created. One of them is made for getting the number of questions in a test. Another web service extracts the text of a specific question. Web services may be extended to support JSON and XML responses.
Mult-agent systems based on web services are offered by Pan and Mao[7]. They propose a novel multi-agent based semantic web service composition model (SWSCPA). It manages the relationships between different service providers and service users. The integration of semantic web services into agent systems is described. Artificial Intelligence planning approach is used for describing web service ontologies.
Security issues concerning web services are discussed by Uma and Kannan [8]. Password guessing attacks are the most important security issues with web services. A secure authentication procedure has to be applied in web services. The movement of the mouse of the user is used to get a higher level of personalization. The proposed dynamic nonce system calculates different mouse movement and generates unique numbers for each session. Each number consist not only information about the mouse movement but also a random number and time stamp.
Diet-aid web services are created by Su [9]. The proposed system generates a diet plan according to the personal health conditions. The system uses web services and an ontological knowledge database. Several scripts are used for processing data chains.
The probability of returning a loan, given to farmers, is studied by Wongnaa and Awunyo-Vitor [10]. They analyze the factors which are most important for loan payment. The result of their study shows that education, experience, profit, age, supervision and off-farm income have positive effects on loan repayment performance. Gender and marital status have negative effects on loan repayment.
Delphi XE (and its later versions XE2, XE3, XE4, XE5, XE6, XE7,…) is an integrated development environment (IDE), created by Embarcadero Technologies. A great variety of software applications are made with Delphi [11], [12], [13].

3 Creating the web service

As we mentioned there are a lot of IDEs which allow developers to create web services. We have chosen the IDE Delphi XE, product of Embarcadero Technologies. The web service will get as input parameters the month of birth (of the loan owner), the region (the loan owner lives) and the sum of contract. The web service will calculate the probability of returning the loan using the following formula.

$$P = e^z / (1 + e^z)$$

Where $z = B_1*X_1 + B_2*X_2 + B_3*X_3 = -0.025*X_1 -0.056*X_2 + 0.02*X_3$
X_1 is the month of birth, X_2 is the region and X_3 is the sum of contract.
This calculation may be implemented within a dynamic link library (DLL). But we decided to make a web service. The web service will be installed on an Apache web server. The web service will be tested with

another software application.
The creation of the web service in Delphi XE is done following the steps:
a) Building the web service (File/New/Other/Delphi projects/Web Services/SOAP server application);
b) Since Delphi offers several types of World Wide Web server applications, we choose "CGI-Standalone executable" (Figure 1);

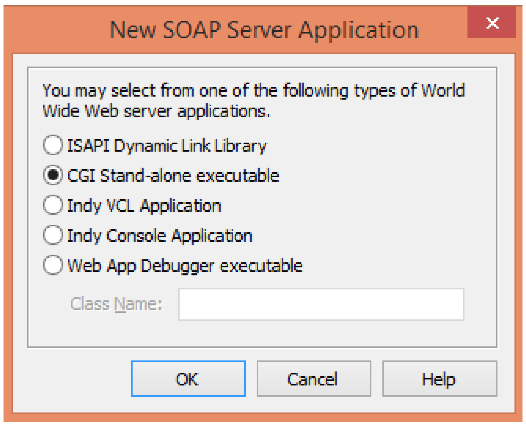
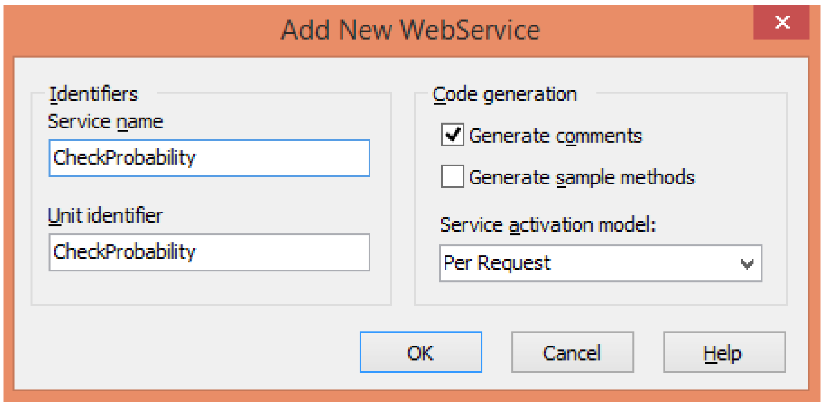

New SOAP Server Application
You may select from one of the following types of World Wide Web server applications.
ISAPI Dynamic Link Library
[✓ CGI Stand-alone executable]
Indy VCL Application
Indy Console Application
Web App Debugger executable
Class Name:
OK
Cancel
Help

Fig. 1. Choosing the type of WWW server application

c) The SOAP server interface is created (File/New/Other/Delphi projects/Web Services/SOAP server interface). We give the name of the interface "CheckProbability". The unit identifier has the same name "CheckProbability". Delphi XE automatically creates two files: "CheckProbabilityIntf.pas" and "CheckProbabilityImpl.pas". The first one describes the interface of the web service, and the second one – its implementation (Figure 2).


Add New WebService

Identifiers
Service name: CheckProbability
Unit identifier: CheckProbability

Code generation
Generate comments [✓]
Generate sample methods [ ]
Service activation model: Per Request

Fig. 2. Adding a new web service

d) Describing the interface of the web service. The unit "CheckProbabilityIntf.pas" is opened. After the row "ICheckProbability = interface(IInvokable)" the declaration of our custom function is defined:

```
function Calc_prob( aSum,
aMonth_of_birth, aRegion : Integer ) :
Extended; stdcall;
```

The unit is saved.

e) Describing the implementation of the web service. The unit "CheckProbabilityImpl.pas" is opened. After the "pubic" statement the interface of the function is defined:

```
function Calc_prob( aSum,
aMonth_of_birth, aRegion : Integer ) :
Extended; stdcall;
```

We write its description after the "implementation" statement in the unit.

DOI: 10.12948/issn14531305/18.4.2014.10

```
function                    TCheckProbabil-
ity.Calc_prob(aSum, aMonth_of_birth,
  aRegion: Integer): Extended;
var
  z : Extended;
begin
  z := 0.02 * aSum -0.056 * aRegion -0.025
* aMonth_of_birth;
  Result := Exp( z ) / ( 1 + Exp( z ) );
end;
```

The unit is saved.
f) The project is saved (File/Save Project As). We give the following name of the project "ws_calc_prob.dproj". Computer programmers usually lose half of their time in the creation of names of variables, names of units and names of projects.

g) The project is compiled (Project/Compile ws_calc_prob). Delphi XE creates an executable file "ws_calc_prob.exe".

h) Deploying the web service to the web server Apache. The created file "ws_calc_prob.exe" is copied into the "cgi-bin" folder of Apache. If Apache is not started it needs to be started. For the uploading of newer versions of "ws_calc_prob.exe" the web server should not be restarted.

i) Initial testing of the web service. We write in a browser http://localhost/cgi-bin/ws_calc_prob.exe. If we have software such as FlashGet or BitCommet, they may try to download the executable from the mentioned link, but it is not possible. We should see the following information.

```
ws_calc_prob - Service Info Page
      ws_calc_prob - PortTypes:
• ICheckProbability [WSDL]
      o     Calc_prob
• IWSDLPublish [WSDL]
Lists all the PortTypes published by this Service
      o     GetPortTypeList
      o     GetWSDLForPortType
      o     GetTypeSystemsList
      o     GetXSDForTypeSystem
      WSIL:  Link to WS-Inspection document of Services here
```

The web service name is "ICheckProbability" with one built-in function "Calc_prob". The web service ICheckProbability may contain other functions which implement different business logic. When we click on the WSDL link right to ICheckProbability, we see the automatically created description of the web service in a XML format.
In intranet or intranet environment instead of "localhost" we have to write the IP address of the web server.

```
<definitions name="ICheckProbabilityservice" targetNamespace="http://tempuri.org/">
<message name="Calc_prob0Request">
 <part name="aSum" type="xs:int"/>
 <part name="aMonth_of_birth" type="xs:int"/>
 <part name="aRegion" type="xs:int"/>
</message>
<message name="Calc_prob0Response">
 <part name="return" type="xs:double"/>
</message>
<portType name="ICheckProbability">
 <operation name="Calc_prob">
  <input message="tns:Calc_prob0Request"/>
  <output message="tns:Calc_prob0Response"/>
 </operation>
</portType>
<binding name="ICheckProbabilitybinding" type="tns:ICheckProbability">
<soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
<operation name="Calc_prob">
<soap:operation    soapAction="urn:CheckProbabilityIntf-ICheckProbability#Calc_prob"
style="rpc"/>
```

```
<input>
 <soap:body  use="encoded"  encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn:CheckProbabilityIntf-ICheckProbability"/>
</input>
<output>
<soap:body   use="encoded"   encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn:CheckProbabilityIntf-ICheckProbability"/>
</output>
</operation>
</binding>
<service name="ICheckProbabilityservice">
 <port name="ICheckProbabilityPort" binding="tns:ICheckProbabilitybinding">
  <soap:address location="http://localhost/cgi-bin/ws_calc_prob.exe/soap/ICheckProb-
ability"/>
 </port>
</service>
</definitions>
```

The description is made using the web service description language (WSDL).

4 Using the web service from a desktop application

The web service may be called from a web application as well as from a desktop application, as well as from a mobile application. In this article we will show how to create a simple desktop application and how to connect it to the web service. The desktop application has its own user interface. But the business logic of calculating the probability of returning a loan stays outside the application. If the business logic changes in the future, the desktop application does not need to be recompiled. Only the web service should be recompiled and uploaded to a web server. The web server should not be restarted.
The creation of the desktop application in Delphi XE is done in the following steps:
a) A folder “WS Client” is created. A new project in Delphi XE is created (File/New/VCL forms Delphi). VCL stays for visual components library. The project is saved as “WS_Client.dpr” (File/Save Project As).
b) A link between the desktop application and the web service should be made (File/New/Other/Delphi projects/Web Services/WSDL importer). In the location of the WSDL write: http://localhost/cgi-bin/ws_calc_prob.exe/wsdl/ICheckProbability. In an internet or intranet environment instead of “localhost” use the real IP address of the web server. A new unit “ICheckProbability1.pas” is automatically created. In this unit there are two rows referencing the web service.

```
defWSDL     =     'http://localhost/cgi-
bin/ws_calc_prob.exe/wsdl/ICheckProba-
bility';
defURL      =     'http://localhost/cgi-
bin/ws_calc_prob.exe/soap/ICheckProba-
bility';
```

We have to change “localhost” with the IP address of the server.
c) The user interface is done. The file Unit1.pas is opened. Five labels, one edit box, two combo boxes are put on the form (Figure 3).

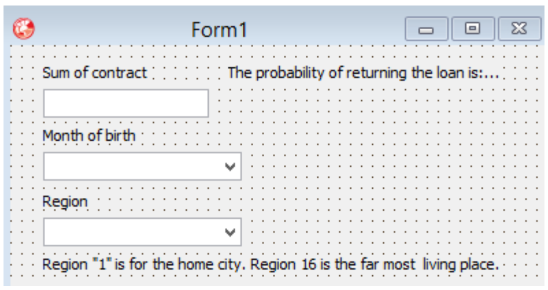

Form1
Sum of contract
The probability of returning the loan is:...
Month of birth
Region
Region "1" is for the home city. Region 16 is the far most living place.

Fig. 3. User interface of the desktop application

d) A custom function “calc_prob” is written within the Unit1.pas. The declaration of the function is in the private section of Unit1.

```
procedure Calc_prob;
```

The description of the function is created within the implementation section of Unit1.

```
procedure TForm1.Calc_prob;
```

```
  var
    aSum, aMonth, aRegion : Integer;
begin
  // initializing the label on the form
 labProb.Caption := 'The probability of returning the loan is:...';
  // converting the text value of the sum to a float value
  aSum := StrToIntDef( editsum.Text, 0 );
  // If the user has not entered a sum or has not chosen
  // values in combo boxes we exit the procedure
  if ( aSum <= 0 ) or
      ( ComboBoxMonthOfBirth.ItemIndex = - 1 ) or
      ( ComboBoxRegion.ItemIndex = -1 ) then
    Exit;
  // converting the text value of the month to an integer value
  aMonth  := ComboBoxMonthOfBirth.ItemIndex + 1;
  // converting the text value of the region to an integer value
  aRegion := ComboBoxRegion.ItemIndex + 1;
  // Calling the web service and showing the probability of returning a loan
  labProb.Caption := format( 'The probability of returning the loan is %.0f%%', [
GetICheckProbability().Calc_prob( aSum, aMonth, aRegion )*100 ] );
end;
```

The call of the web service is made by the last line of the written source code. The declared variables (aSum, aMonth and aRegion) reserve local memory when the procedure “Calc_prob” is started. The memory is automatically freed on the ending the procedure. The programmer does not need to write source code to free these three variables.

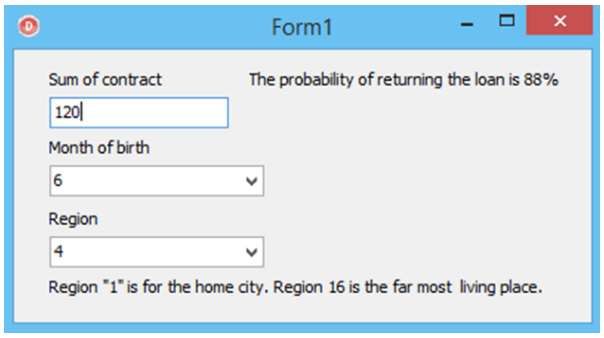
e) The procedure “Calc_prob” should be called when the user changes the sum of contract or changes the month of birth or changes the region of living. That is why we write three event procedures, connected to the edit box and the two combo boxes.

```
procedure TForm1.ComboBoxMonthOfBirthChange(Sender: TObject);
begin
  Calc_prob;
end;

procedure TForm1.ComboBoxRegionChange(Sender: TObject);
begin
  Calc_prob;
end;

procedure TForm1.EditSumChange(Sender: TObject);
begin
  Calc_prob;
end;
```

f) The two combo boxes are filled with values in design time (by using the Items property of each combo box). The combo box “MonthOfBirth” is filled with the numbers from 1 to 12. The combo box “Region” is filled with the numbers from 1 to 16.

g) The project is compiled (Project/Compile WS_Client). An executable file “WS_Client.exe” is created.

For testing the desktop application, the WS_Client.exe is launched (Figure 4).



Sum of contract: 120

The probability of returning the loan is 88%

Month of birth: 6

Region: 4

Region "1" is for the home city. Region 16 is the far most living place.

Fig. 4. Running the desktop application

If the sum of contract is 120 levs (approximately 60 EURO), the customer is born in June and he/she lives not far from the credit institution, the probability of returning the loan is 88% percent. When the end user changes the sum of contract or the month of birth or the region, the desktop application calls the web service and automatically shows the probability of returning the loan. The first call of the web services takes several seconds, but next calls to the web service are executed in less than a second. This desktop application may be used by banks and credit institutions. The web service may be integrated into an existing loan management system.

5 Conclusions

Existing standards for web services such as XML and WSDL allow programmers to write separate parts of business logic and compile them as web services. Since there are a lot of existing IDEs, this study shows a methodology for designing, creating, compiling, deploying and testing web services with Delphi XE (a software product created by Embarcadero Technologies). Encapsulating business logic into a web service allows several applications to use the web service. Moreover, the web service may be created in one IDE, but the client application may be created with another IDE. Calculating the probability of returning a loan is a difficult task. A previous research is used to get the mathematical formula for calculating it.
If the business logic changes, the source code of the web service has to be changed. The web service has to be compiled. The new executable file has to be uploaded to a web server. No restart of the web server is needed. The desktop application will continue working with the new business logic without the need of restarting and its recompilation.
Future research may focus on another web services development. The proposed methodology may be used by other programmers or researchers to build other web services and to extend the functionality of existing software applications.

References

[1] C. Strimbei. (2013). SOA based Data Architecture for HTML5 Web Applications. *Informatica Economică Journal* [Online]. 2(17). Available: http://www.revistaie.ase.ro/content/66/07%20-%20Strimbei.pdf

[2] O. Dospinescu and M. Perca. (2013). Web Services in Mobile Applications. *Informatica Economică Journal* [Online]. 2(17). Available: http://www.revistaie.ase.ro/content/66/02 - Dospinescu, Perca.pdf

[3] D. Horia. (2013). Orders management by using smart phones. *Annals of the University of Oradea: Economic Science* [Online]. 1(22). Available:

[4] M. Walczak and Z. Polkowski. (2013) The e-health system in Poland. *Scientific Bulletin: Economic Sciences* [Online]. 2(12) Available: http://economic.upit.ro/repec/pdf/2013_2_6.pdf

[5] R. Khanna and P. Bhagat. (2013) Introducing Virtual Law offices in the Existing Judiciary. *Database Systems Journal* [Online] 1(4). Available: http://www.dbjournal.ro/archive/11/11_3.pdf

[6] P. Pocatilu. (2013). Developing an M-Learning Application for iOS. *Informatica Economică Journal* [Online] 4(17). Available: http://www.revistaie.ase.ro/content/68/07%20-%20Pocatilu.pdf.

[7] S. Pan and Q. Mao. (2013). Case Study on Web Service Composition Based on Multi-Agent System. *Journal of Software* [Online]. 4(8). Available: http://ojs.academypublisher.com/index.php/jsw/article/view/7804

[8] E. Uma and A. Kannan. (2013). The dynamic nonce based authentication scheme to defend intermediary attack for web services. Journal of Computer Science [Online] 8(9). Available: http://thescipub.com/PDF/jcssp.2013.998.1007.pdf

[9] C. Su et. al. (2013). Personalized ubiquitous diet plan service based on ontology and web services. International Journal of Information and Education Technology [Online]. 5(3). Available: http://www.ijiet.org/papers/329-K012.pdf.

[10] C. Wongnaa and D. Awunyo-Vitor. (2013). Factors Affecting Loan Repayment Performance Among Yam Farmers in the Sene District, Ghana. AGRIS on-line Papers in Economics and Informatics [Online]. 2(5). Available: http://online.agris.cz/files/2013/agris_on-line_2013_2_wongnaa_awunyo-victor.pdf
[11] J. Vasilev. (2012). Guidelines for Improvement Information Processes in Commerce by Implementing the Link Between a Web Application and Cash Registers. Theory and Applications of Mathematics and Computer Science [Online]. 2(2). Available: http://www.uav.ro/applications/se/journal/index.php/TAMCS/article/download/63/46.
[12] J. Vasilev. (2013). Application of Skype API to control working time. TEM Journal – Technology, Education, Management, Informatics [Online]. 4(2). Available: http://www.temjournal.com/documents/vol2no4/Application%20of%20Skype%20API%20to%20control%20working%20time.pdf
[13] Embarcadero Technologies. Available: http://www.embarcadero.com
[14] UDDI: an XML Web Service. Available: http://msdn.microsoft.com/en-us/library/ms950813.aspx

Julian VASILEV has graduated the Faculty of Informatics at Varna University of Economics in 1998. He holds a PhD diploma in Informatics from 2008 and he had gone through all didactic positions since 1999 when he joined the staff of the Varna University of economics, teaching assistant in 1999, senior assistant professor in 2003, assistant professor in 2007 and associate professor in 2011. Currently he is an Associate Professor of Informatics within the Department of Informatics at Faculty of Informatics from the Varna University of Economics. He works also as an IT expert within the Centre for Social Surveys. He is the author of more than 10 books and over 25 journal articles and over 30 articles in conference proceedings in the field of ERP systems, GIS, information logistics, e-government, accounting software, statistics, business analysis, e-logistics, network security. Part of his work focuses on the application of quantitative and qualitative methods in social research. Another part of his fork focuses on the web application development with Delphi XE. He has participated in several research projects. He is a member of the editorial board of the journal Review of Arts and Humanities. He has organized several conferences at VUE. He is an expert in ICT-FP7 at the European Commission. He was a visiting professor in universities in Maribor (2006), Koper (2007), Lithuania (2008) and Katowice (2009).