

# DINO Excel AddIn Documentation

**Date:** 31 August 2009

**By:** Oscar Gomez (iDelft)

**Review:** Erik Vriend (iDelft)

**Version:** 1.0.10

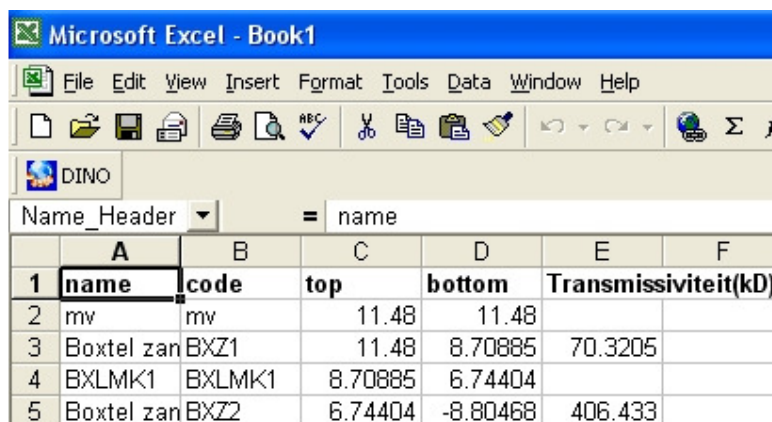
**Customer:** TNO Bouw en Ondergrond, Utrecht

## 1. Introduction

The **DINO WebService** is an Excel extension application that allows the users to retrieve information from any web service following the specifications of the [DINO WebService](#). This web service provides information of the Dutch underground composition on different locations for different models and resolutions.

The application, once installed, creates an extra toolbar in Excel 2003 (see: figure 1) or an extra *Ribbon Tab* in Excel 2007 (see: figure 2). It contains a button that will open the *DINO WebService Form* and that allows the user to specify the necessary parameters to retrieve information from the service.

A number of specific Excel functions are defined too. These functions allow the user to retrieve the web service information cell by cell, unlike the form, which retrieves it in a block of cells.



The screenshot shows the Microsoft Excel 2003 interface. A custom toolbar labeled 'DINO' is visible, containing a button with a globe icon. Below the toolbar, a data table is displayed with columns A through F. The table contains data for different locations and models, including 'mv', 'BxZ1', 'BXLmk1', and 'BxZ2'. The table has a header row and five data rows.

	A	B	C	D	E	F
1	name	code	top	bottom	Transmissiviteit(kD)	
2	mv	mv	11.48	11.48		
3	BxZ1	BxZ1	11.48	8.70885	70.3205	
4	BXLmk1	BXLmk1	8.70885	6.74404		
5	BxZ2	BxZ2	6.74404	-8.80468	406.433	

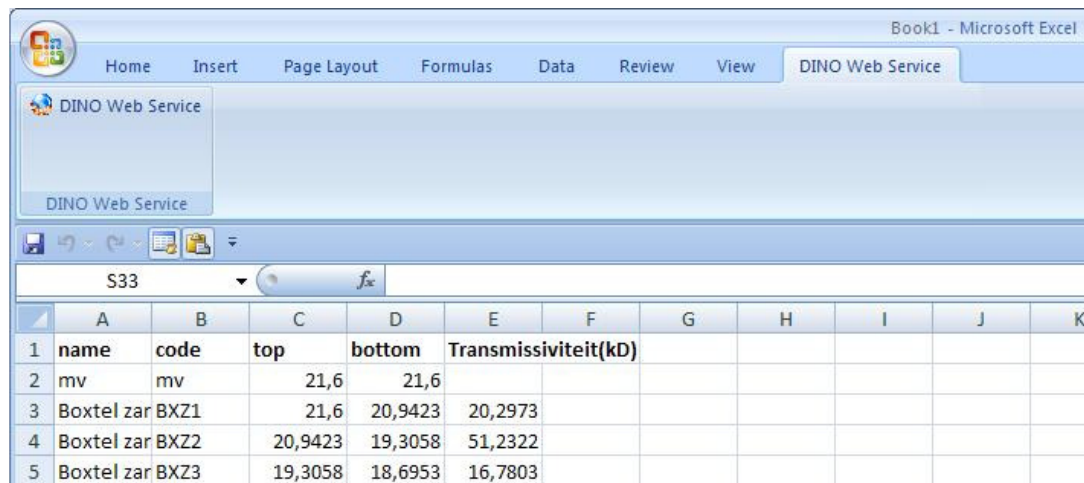
**Figure 1.** DINO toolbar button in Excel 2003

## 2. Installation instructions

The **DINO WebService** installation kit comes with two different installation applications:

- *Setup.exe*
- *DINO WebService Setup.msi*

In most systems, there will be no difference in executing one or the other file, but in Windows Vista, the *Setup.exe* needs to be used under an administrator account and the MSI application for the rest of users.



**Figure 2.** DINO Ribbon button in Excel 2007

### 3. User's Guide

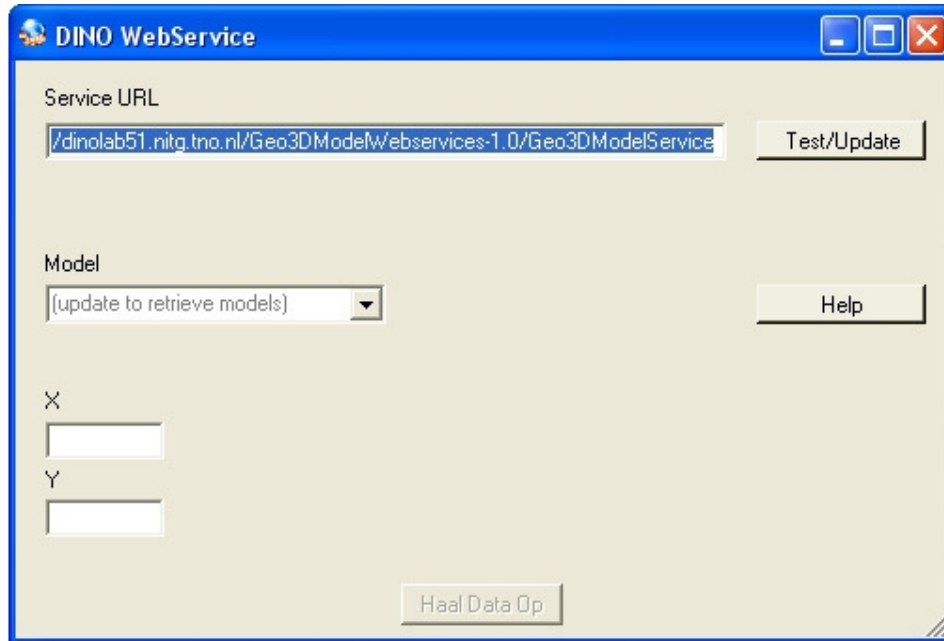
#### 3.1 DINO WebService Form

Once the Excel AddIn is installed and the toolbar/ribbon button is visible, the usage is straightforward. The following steps need to be followed:

- i. **Position selection:** Select an active cell in the worksheet. Note: the top-left corner of the result set will later on be positioned in the active cell.

The number of rows and columns added to the worksheet is unknown as it depends on the number of available layers and the parameters.

- ii. **Open the DINO form:** Click the DINO button to open the DINO Webservice form (See Figure 3). Initially, the form is empty. Only the standard URL of the Web Service is shown.



**Figure 3.** DINO Web Service Control Form initial state

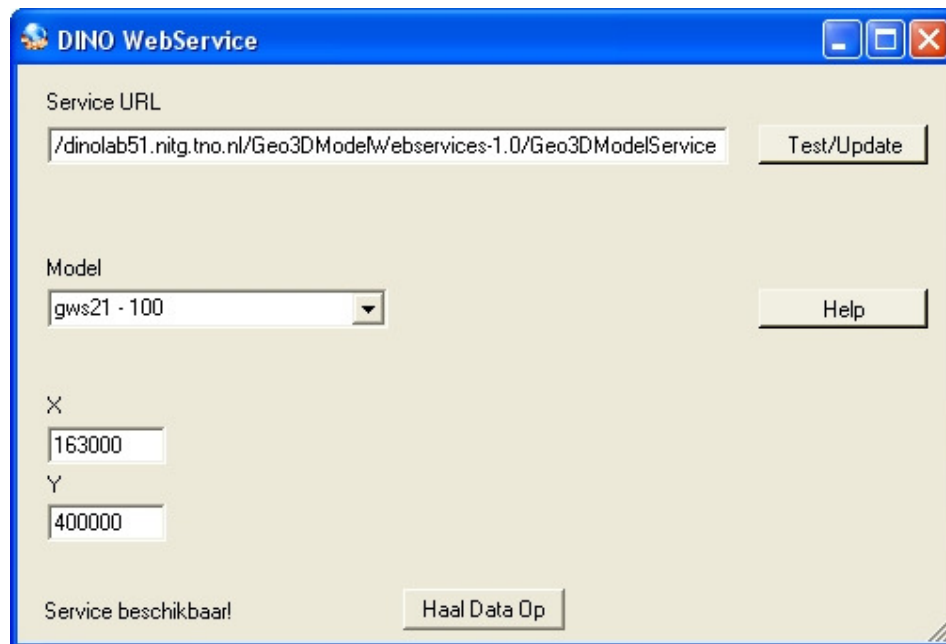
- iii. **Test the service:** the *Test/Update* functionality ensures that the *Service URL* is correct. If the WebService is up and running the software retrieves the available model-resolution pairs. To test the WebService click the "Test/Update" button. The Test/Update button should be used after opening the form and after editing the Service URL box (if necessary).

There are two options to test the service:

- a. Empty XY: the software will retrieve all the model/resolution pairs available in the system.
- b. Specific XY value: the software will retrieve only those model/resolution pairs available in the specified location.

If the test was successful the *Model* combo box will be filled. A "Service beschikbaar" message will appear and the Run button ("*Haal Data Op*") will be enabled. Otherwise an error/warning message will be shown.

- iv. **Fill the parameters:** Before running the service, an X and Y coordinate value must be specified in RD projection system. These values need to be numbers to be accepted by the system. Not all the XY pairs have associated geo-data information (!) For testing purposes, enter X as 163000 and Y as 400000 (see Figure 4).
- v. **Run the service:** Figure 4 shows the form when the software is ready to make a data request to the service (see Figure 4). By clicking the "Haal Data op" button, the data is requested and retrieved via the Webservice. If the request is successful, the form will be hidden and the information will be loaded into the worksheet using the active cell as upper-left corner. Otherwise, an error message will pop up.



The screenshot shows a Windows-style dialog box titled "DINO Webservice". It contains the following elements:

- Service URL:** A text box containing the URL "/dinolab51.nitg.tno.nl/Geo3DModel Webservices-1.0/Geo3DModelService". To its right is a "Test/Update" button.
- Model:** A dropdown menu showing "gws21 - 100". To its right is a "Help" button.
- X:** A text box containing the value "163000".
- Y:** A text box containing the value "400000".
- Status:** At the bottom left, the text "Service beschikbaar!" is displayed.
- Action:** At the bottom right, there is a button labeled "Haal Data Op".

**Figure 4.** Example request parameters

Optionally, the *Help* button opens a form with some information about the DINO service and the AddIn functionality and utilization.

### **3.2 DINO Excel Functions**

In addition to the Webservice form, the addin package includes a number of Excel formulas which allow the user accessing the web service at cell level.

Predefined URL functions list:

```
DINO_Layer_Name(model, resolution, x, y, layerIndex)
DINO_Layer_Code(model, resolution, x, y, layerIndex)
DINO_Layer_Top(model, resolution, x, y, layerIndex)
DINO_Layer_Bottom(model, resolution, x, y, layerIndex)
DINO_Layer_Parameter(model, resolution, x, y, layerIndex,
parameterName)
```

Flexible URL functions list:

```
DINO_Layer_Name2(serviceURL, model, resolution, x, y, layerIndex)
DINO_Layer_Code2(serviceURL, model, resolution, x, y, layerIndex)
DINO_Layer_Top2(serviceURL, model, resolution, x, y, layerIndex)
DINO_Layer_Bottom2(serviceURL, model, resolution, x, y, layerIndex)
DINO_Layer_Parameter2(serviceURL, model, resolution, x, y,
layerIndex, parameterName)
```

<b>model:</b>	String	Model name
<b>resolution</b>	Integer	Resolution value
<b>x:</b>	Decimal	X coordinate in RD coordinate system
<b>y:</b>	Decimal	Y coordinate in RD coordinate system
<b>layerIndex:</b>	Integer	Layer index of the desired parameter
<b>parameterName:</b>	String	Parameter name (exactly as it is called in the web service)
<b>serviceURL:</b>	String	URL of the web service

Here are some examples of correct usage of the functions:

```
DINO_Layer_Name("gws21",100,163000,400000,1)
DINO_Layer_Name2("http://anyURL?wsdl","gws21",100,163000,400000)
DINO_Layer_Parameter("gws21",100,163000,400000,1,"Transmissiviteit(kD)
")
DINO_Layer_Parameter2("http://anyURL?wsdl","gws21",100,163000,400000,
1,"Transmissiviteit(kD)")
```

The name of the function parameters is self-explanatory and it is the same information that needs to be inserted in the Webservice form. On the other hand, the method of accessing the data forces a conceptual change. In 3.1 it was explained how to retrieve all the underground information for an specific coordinate in just one call. Now, as Excel functions can only give value to a unique cell, every function call retrieves the value of one cell only. To obtain the same table as we would through the table, we would have to make at least 4 function calls per geoLayer (e.g. at least 32 calls in a coordinate with 8 geoLayers). The result, as it can be seen in Figure 5, is the same.

FIXED URL EXCEL FUNCTIONS					FLEXIBLE URL EXCEL FUNCTIONS					EXCEL Addin FORM				
name	code	top	bottom	Transmissiviteit(kD)	name	code	top	bottom	Transmissiviteit(kD)	name	code	top	bottom	Transmissiviteit(kD)
0 mv	mv	11.48	11.48		0 mv	mv	11.48	11.48		0 mv	mv	11.48	11.48	
1 Bxstel zan BXZ1		11.48	8.70885	70.3205	1 Bxstel zan BXZ1		11.48	8.70885	70.3205	1 Bxstel zan BXZ1		11.48	8.70885	70.3205
2 BxLMK1 BxLMK1		8.70885	6.74404		2 BxLMK1 BxLMK1		8.70885	6.74404		2 BxLMK1 BxLMK1		8.70885	6.74404	
3 Bxstel zan BXZ2		6.74404	-8.80468	406.433	3 Bxstel zan BXZ2		6.74404	-8.80468	406.433	3 Bxstel zan BXZ2		6.74404	-8.80468	406.433
4 Bxstel zan BXZ3		-8.80468	-12.2659	89.6775	4 Bxstel zan BXZ3		-8.80468	-12.2659	89.6775	4 Bxstel zan BXZ3		-8.80468	-12.2659	89.6775
5 Formatie v BEZ1		-12.2659	-13.6551	52.5136	5 Formatie v BEZ1		-12.2659	-13.6551	52.5136	5 Formatie v BEZ1		-12.2659	-13.6551	52.5136
6 BEZ3 BEZ3		-13.6551	-23.0268	556.552	6 BEZ3 BEZ3		-13.6551	-23.0268	556.552	6 BEZ3 BEZ3		-13.6551	-23.0268	556.552
7 Formatie v STZ1		-23.0268	-50.9125	901.598	7 Formatie v STZ1		-23.0268	-50.9125	901.598	7 Formatie v STZ1		-23.0268	-50.9125	901.598
8 Formatie v STZ2		-50.9125	-77.0238	1369.46	8 Formatie v STZ2		-50.9125	-77.0238	1369.46	8 Formatie v STZ2		-50.9125	-77.0238	1369.46
9 Formatie v SYZ2		-77.0238	-77.2101	2.46299	9 Formatie v SYZ2		-77.0238	-77.2101	2.46299	9 Formatie v SYZ2		-77.0238	-77.2101	2.46299
10 Formatie v SYZ3		-77.2101	-78.3406	16.8026	10 Formatie v SYZ3		-77.2101	-78.3406	16.8026	10 Formatie v SYZ3		-77.2101	-78.3406	16.8026
11 Formatie v SYK3		-78.3406	-78.4475		11 Formatie v SYK3		-78.3406	-78.4475		11 Formatie v SYK3		-78.3406	-78.4475	
12 Formatie v WAK1		-78.4475	-95.9477		12 Formatie v WAK1		-78.4475	-95.9477		12 Formatie v WAK1		-78.4475	-95.9477	
13 Formatie v PZWAZ3		-95.9477	-107.059	397.369	13 Formatie v PZWAZ3		-95.9477	-107.059	397.369	13 Formatie v PZWAZ3		-95.9477	-107.059	397.369
14 Formatie v WAK2		-107.059	-120.078		14 Formatie v WAK2		-107.059	-120.078		14 Formatie v WAK2		-107.059	-120.078	
15 Formatie v PZWAZ5		-120.078	-165.603	2516.77	15 Formatie v PZWAZ5		-120.078	-165.603	2516.77	15 Formatie v PZWAZ5		-120.078	-165.603	2516.77
16 Formatie v WAK3		-165.603	-174.989		16 Formatie v WAK3		-165.603	-174.989		16 Formatie v WAK3		-165.603	-174.989	
17 Formatie v PZWAZ7		-174.989	-182.325	222.355	17 Formatie v PZWAZ7		-174.989	-182.325	222.355	17 Formatie v PZWAZ7		-174.989	-182.325	222.355
18 Kiezeloote KIK1		-182.325	-189.562		18 Kiezeloote KIK1		-182.325	-189.562		18 Kiezeloote KIK1		-182.325	-189.562	
19 Kiezeloote KIK2		-189.562	-213.89	547.918	19 Kiezeloote KIK2		-189.562	-213.89	547.918	19 Kiezeloote KIK2		-189.562	-213.89	547.918
20 Kiezeloote KIK2		-213.89	-228.841		20 Kiezeloote KIK2		-213.89	-228.841		20 Kiezeloote KIK2		-213.89	-228.841	
21 KIZ5 KIZ5		-228.841	-237.009	221.75	21 KIZ5 KIZ5		-228.841	-237.009	221.75	21 KIZ5 KIZ5		-228.841	-237.009	221.75
22 OOC OOC		-237.009	-323.726		22 OOC OOC		-237.009	-323.726		22 OOC OOC		-237.009	-323.726	
23 NO DATA NO DATA		NO DATA	NO DATA	NO DATA	23 NO DATA NO DATA		NO DATA	NO DATA	NO DATA	23 NO DATA NO DATA		NO DATA	NO DATA	NO DATA

**Figure 5.** Output of 3 different web service access methods

This conceptual change explains the need of the *layerIndex* parameter (from 0 to any integer number). It specifies the layer index from which a given parameter wants to be retrieved.

The functions are divided in two main groups. The first group (predefined URL) makes a call to the web service located in the URL described in the section 3 of this documentation, while the second group (flexible URL) accepts a different URL as parameter.

The return value of these functions can be:

<b>Value</b>	Value of the requested parameter
<b>No Value</b>	Only in the <code>DINO_Layer_Parameter</code> or <code>DINO_Layer_Parameter2</code> functions, if the specified parameter is not present in the layer.
<b>NO DATA</b>	If layer index is not present
<b>Invalid URL</b>	If the specified URL is not valid
<b>Error</b>	Unknown error

The file *DINOFunctionsExample.xls* in the documentation folder contains a sample of how to create a complete layers table with flexible and fixed URL functions.

### **3.3 VBA Interface**

As requested, the most important functionalities provided by the web service are accessible from *Visual Basic for Applications* (VBA). An interface offers access to the information provided by the *DINO Web Service*. This interface can be divided in two conceptually different access methods.

The first method mimics the mechanism of the *DINO toolbar button*. By providing the request parameters, it connects to the web service, retrieves the information and fills the worksheet with it. A sample of this method utilization can be seen in *VBASampleA.xls* and in the following code:

```
Dim DinoModule As Object
Set DinoModule =
Application.COMAddIns.Item("DINO_AddIn.AddinModule").Object

'Parameter population
DinoModule.popModel ("gws21")
DinoModule.popResolution (100)
DinoModule.popX (163000)
DinoModule.popY (400000)
DinoModule.popCol (4) 'Initial cell coordinates
DinoModule.popRow (5) 'for the result table
'Optional
'DinoModule.popUrl
('http://dinolab51.nitg.tno.nl/Geo3DModelWebservices-
                                1.0/Geo3DModelService?wsdl")
'Function firing
DinoModule.DINOatVBA
```

Only the `popUrl` function is optional, the rest are required to call the `DINOatVBA` function successfully. A more formal description of the functions defining the VBA interface can be found at the *Appendix B.1*. No error handling is provided on this access method, so the request arguments need to be carefully checked on the VBA code.

The other possible method to retrieve information from the web service using VBA code exploits the user defined functions added to Excel by the AddIn. In this case, the VBA code will be used to introduce the Excel functions dynamically, making them more flexible and comfortable (code in *VBASampleB.xls*):

```
Dim i As Integer
For i = 3 To 27
    Cells(i,2)="=DINO_Layer_Name("&Chr(34)&"gws21"&Chr(34)&_
                                ",100,163000,400000," &CStr(i - 1) & " )"
    Cells(i,3)="=DINO_Layer_Code("&Chr(34)&"gws21"&Chr(34)&_
                                ",100,163000,400000,"&CStr(i - 1) & " )"
    Cells(i,4)="=DINO_Layer_Top("&Chr(34)&"gws21"&Chr(34)&_
                                ",100,163000,400000,"&CStr(i-1)&") "
    Cells(i,5)="=DINO_Layer_Bottom("&Chr(34)&"gws21"&Chr(34)&_
                                ",100,163000,400000,"&CStr(i-1)&") "
    Cells(i,6)="=DINO_Layer_Parameter("&Chr(34)&"gws21"&Chr(34)&_
                                ",100,163000,400000,"& CStr(i-1)&","&Chr(34)&_
                                "Transmissiviteit(kD)"&Chr(34)&") "
Next
```

While the first method is more efficient and requires only one call to the web service, it only accepts one result layout possibility. On the other hand, the second method gives total flexibility on which information to retrieve and where to place it. The set back of the flexible method is the high network waste, as it requires one call to the web service per cell. Both methods have fixed and flexible URL access options.



## 4. Technical information

Excel Versions Supported:

Excel 2007, Excel 2003 and Excel 2000 (not guaranteed)

Developed in:

C# .NET

.NET framework 2.0

Web Service Specification Source:

[http://dinolab51.nitg.tno.nl/Geo3DModelWebservices-1.0/  
Geo3DModelService?wsdl](http://dinolab51.nitg.tno.nl/Geo3DModelWebservices-1.0/Geo3DModelService?wsdl)

See also Appendix A and B with additional information.

### **4.1 Technology limitations**

The AddIn has one limitation derived from the technology used to extend Excel's functionality. This limitation applies to the user defined functions (and the processes derived from these). When the computer's or application's regional settings are set to *Dutch*, these functions don't accept cell references as arguments value. The following example will help clarifying the problem:

**OK**

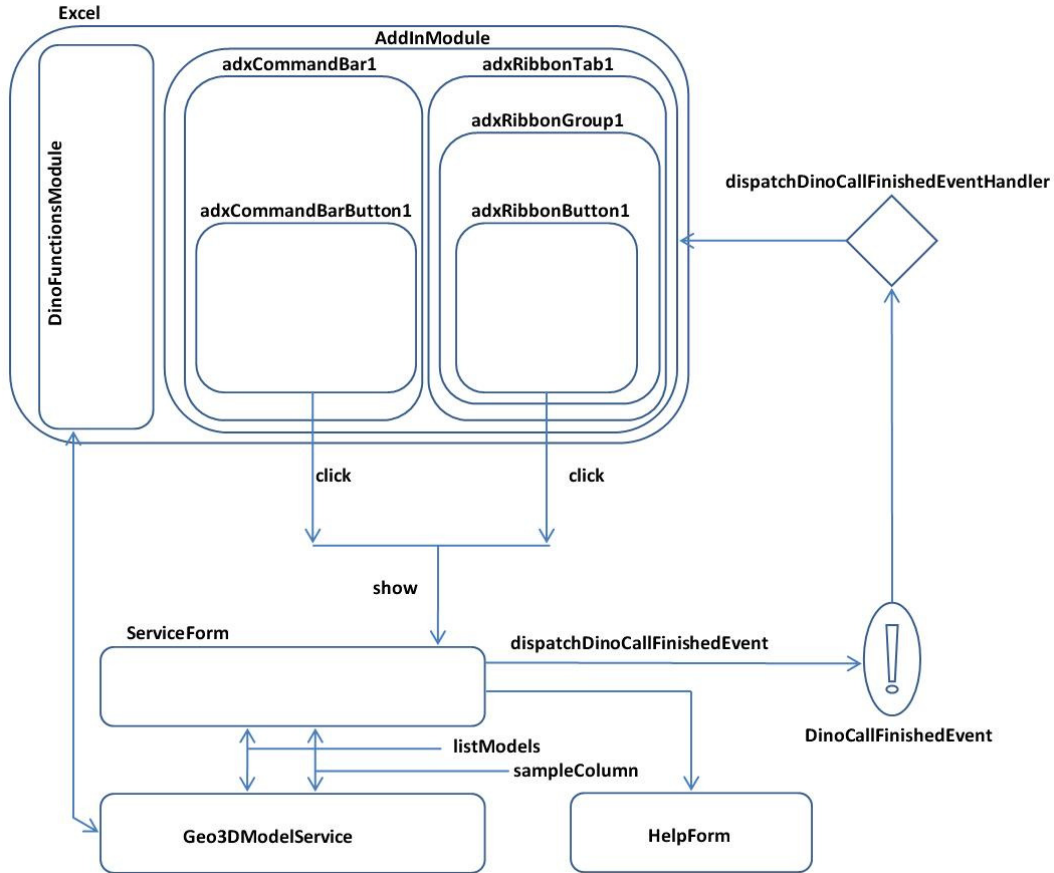
```
=DINO_Layer_Name("gws21", 100, 163000, 400000, 1)
```

**ERROR**

```
=DINO_Layer_Name("gws21", 100, 163000, 400000, $A$3)
```

This problem can be easily solved both by hardtyping the values or by changing the settings to English.

## Appendix A: Class diagram



**Figure 6.** Class Diagram

In the class diagram we can see that the main class, *AddInModule*, is embedded in Excel. The class has two objects. *adxRibbonTab1* serves Excel 2007 while *adxCommandBar1* serves the rest of supported Excel versions. Inside these two objects different subclasses are present, creating menus and buttons.

The buttons show the *ServiceForm* when their *Click* event is fired. It is the form which has direct access to the *Geo3DModelService*, the web service stub class.

After receiving a correct response from the stub to the *sampleColumn* function, the form fires a *DinoCallFinishedEvent* through the dispatcher. The main class handles this event through an event handler and processes the results (passed as event arguments).

From the *ServiceForm* a button opens the *HelpForm* containing a short description of the service and user guide.

A second independent module is embedded in Excel (*DinoFunctionsModule*). It contains the Excel functions defined to be called directly from the Excel cells. This module access the *Geo3DModelService* directly.

## Appendix B: Method Reference

### B.1 AddinModule

```
//
// Sets a neutral Language Settings to avoid generic Excel error
//
private void setNewCulture()

//
// Sets back the original Language Settings
//
private void setOldCulture()

//
// Excel 2007 button clicked
//
private void adxRibbonButton1_OnClick(object sender,
                                     AddinExpress.MSO.IRibbonControl control, bool pressed)

//
// Excel 2000-2003 button clicked
//
private void adxCommandBarButton1_Click(object sender)

//
// Opens the parameter selection form
//
private void openServiceForm()

//
// Event handler serving the DINO form finished event
//
private void Dino_CallFinished(DinoCallFinishedEventArgs

//
// Fills the results table with the info from the DINO WebService
//
private void fillSheet(GeoColumn data)

//
// Prints in the Excel worksheet the header of the result table
//
private void setHeader()

//
// Normalizes the cell name to ensure it is valid
//
private string formatCellName(string cellName)

/*****
Functions accessible from VBA.
*****/

public void DINOatVBA()
public void popModel(string model)
public void popUrl(string url)
public void popResolution(int resolution)
public void popX(double x)
public void popY(double y)
```

```

public void popCol(int col)
public void popRow(int row)
public void clearVBAData()

```

```

/*****
End VBA Functions
*****/

```

## B2. ServiceForm

```

//
// Clear error labels and hide/close the form
//
private void Form1_Closing(object sender,
                           System.ComponentModel.CancelEventArgs e)

//
// Performs a URL test and model/resolution pairs retrieval
//
private void testButton_Click(object sender, EventArgs e)

//
// Retrieves geo information from web service in the specified URL,
// model, resolution and XY coordinate
//
private void RunButton_Click(object sender, EventArgs e)

//
// Checks if all the parameters of the form are correctly formatted
//
private bool correctParameters()

//
// Launches the form "finished" event to be caught by the AddIn
// It is a notification of "geo info ready"
//
private void DispatchDinoCallFinishedEvent(GeoColumn geoColumn)

//
// On URL changed, service needs to be tested and model/resolution
// retrieved. To force it, the RUN (Haal Data Op) button is disabled
// until test is clicked (and successful)
//
private void urlTextBox_TextChanged(object sender, EventArgs e)

//
// X coordinate format check (done on X changed)
//
private void xTextBox_TextChanged(object sender, EventArgs e)

//
// Y coordinate format check (done on Y changed)
//
private void yTextBox_TextChanged(object sender, EventArgs e)

//
// Opens the Help form
//
private void helpButton_Click(object sender, EventArgs e)

```

### B.3 DinoCallFinishedEvent (arguments and delegate )

```
//Class for the event arguments
public class DinoCallFinishedEventArgs : EventArgs
{
    // Include properties for the account number, current balance,
    // required minimum balance, a message describing the event,
    // and a transaction ID.
    // You can include any information that would be useful for you
    // application.
    private GeoColumn GC;
    public GeoColumn geoColumn
    {
        get()

        // AccountBalanceEventArgs constructor
        public DinoCallFinishedEventArgs(GeoColumn gc)

    }

    //Finished Event Delegate
    public delegate void DinoCallFinishedEventHandler(
        DinoCallFinishedEventArgs dinoCallFinishedEventArgs);
```

### B.4 HelpForm

```
//
// Opens the Initial Help page
//
private void goToIniPanel(object sender, EventArgs e)

//
// Opens the DINO WebService Test Help page
//
private void goToTestPanel(object sender, EventArgs e)

//
// Opens the DINO WebService "Ophalen" Help page
//
private void goToHaalPanel(object sender, EventArgs e)

//
// Opens the DINO WebService "Uitvoer" Help page
//
private void goToUitvoerPanel(object sender, EventArgs e)
```

## B.4 DinoFunctionsModule

```
//
// Retrives the geoLayer name to the cell where the function is
// invoked
//
public Object DINO_Layer_Name(string model, int resolution, double x,
                             double y, int layerIndex)

//
// As the previous, but indicating web service URL
//
public Object DINO_Layer_Name2(string serviceURL, string model,
                              int resolution, double x, double y, int layerIndex)

//
// Retrives the geoLayer code to the cell where the function is
// invoked
//
public Object DINO_Layer_Code(string model, int resolution, double x,
                             double y, int layerIndex)

//
// As the previous, but indicating web service URL
//
public Object DINO_Layer_Code2(string serviceURL, string model,
                              int resolution, double x, double y, int layerIndex)

//
// Retrives the geoLayer top high to the cell where the function is
// invoked
//
public Object DINO_Layer_Top(string model, int resolution, double x,
                             double y, int layerIndex)

//
// As the previous, but indicating web service URL
//
public Object DINO_Layer_Top2(string serviceURL, string model,
                              int resolution, double x, double y, int layerIndex)

//
// Retrives the geoLayer bottom high to the cell where the function
// is invoked
//
public Object DINO_Layer_Bottom(string model, int resolution,
                                double x, double y, int layerIndex)

//
// As the previous, but indicating web service URL
//
public Object DINO_Layer_Bottom2(string serviceURL, string model,
                                 int resolution, double x, double y, int layerIndex)

//
// Retrives a geoLayer parameter value to the cell where the function
// is invoked
//
public Object DINO_Layer_Parameter(string model, int resolution,
                                   double x, double y, int layerIndex, string parameterName)
```

```
//  
// As the previous, but indicating web service URL  
//  
public Object DINO_Layer_Parameter2(string serviceURL, string model,  
                                     int resolution, double x, double y, int layerIndex,  
                                     string parameterName)  
  
//  
// Function in charge of the web service call  
//  
private GeoColumn retrieveData(string serviceURL, string model,  
                               int resolution, double x, double y)  
  
//  
// Centralized error processing function  
//  
private string errorProcessing(string errorMessage)
```