Using APL WebServices, WebTransfer, C# and Windows Presentation Foundation

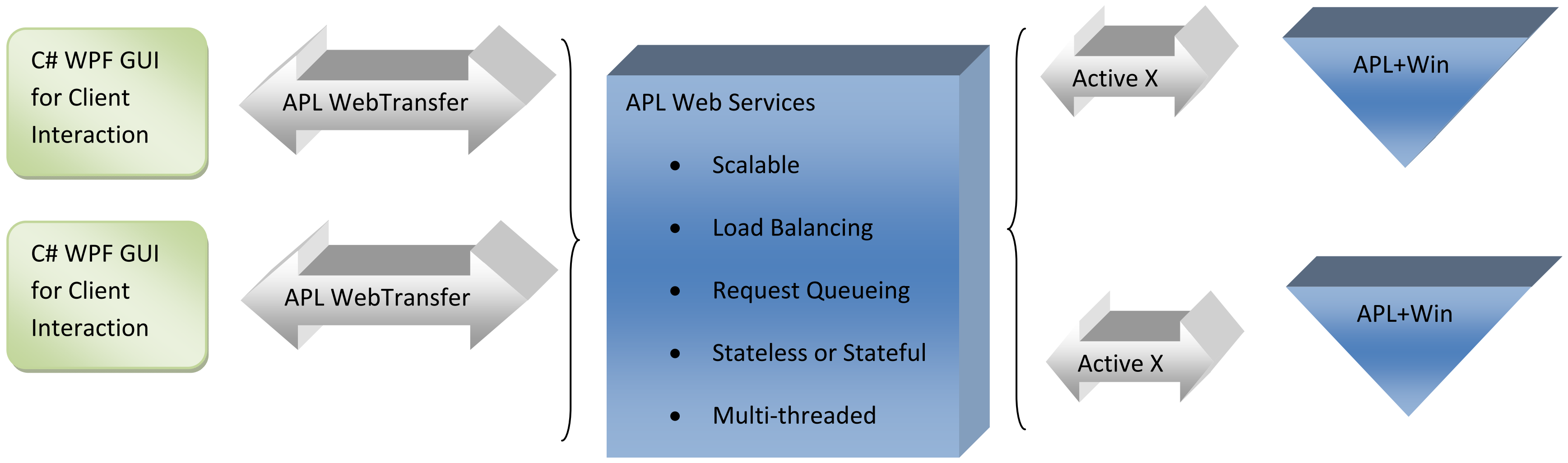
Synopsis:

This example is included in the '\framingham' subdirectory when APL WebServices is installed.

This example illustrates a C# / WPF GUI using APL WebTransfer to communicate with APL WebServices. APL WebServices uses ActiveX to communicate with APL+Win. The APL+Win workspace(s) contain functions which perform application-specific calculations and business rules analysis to satisfy the client request.

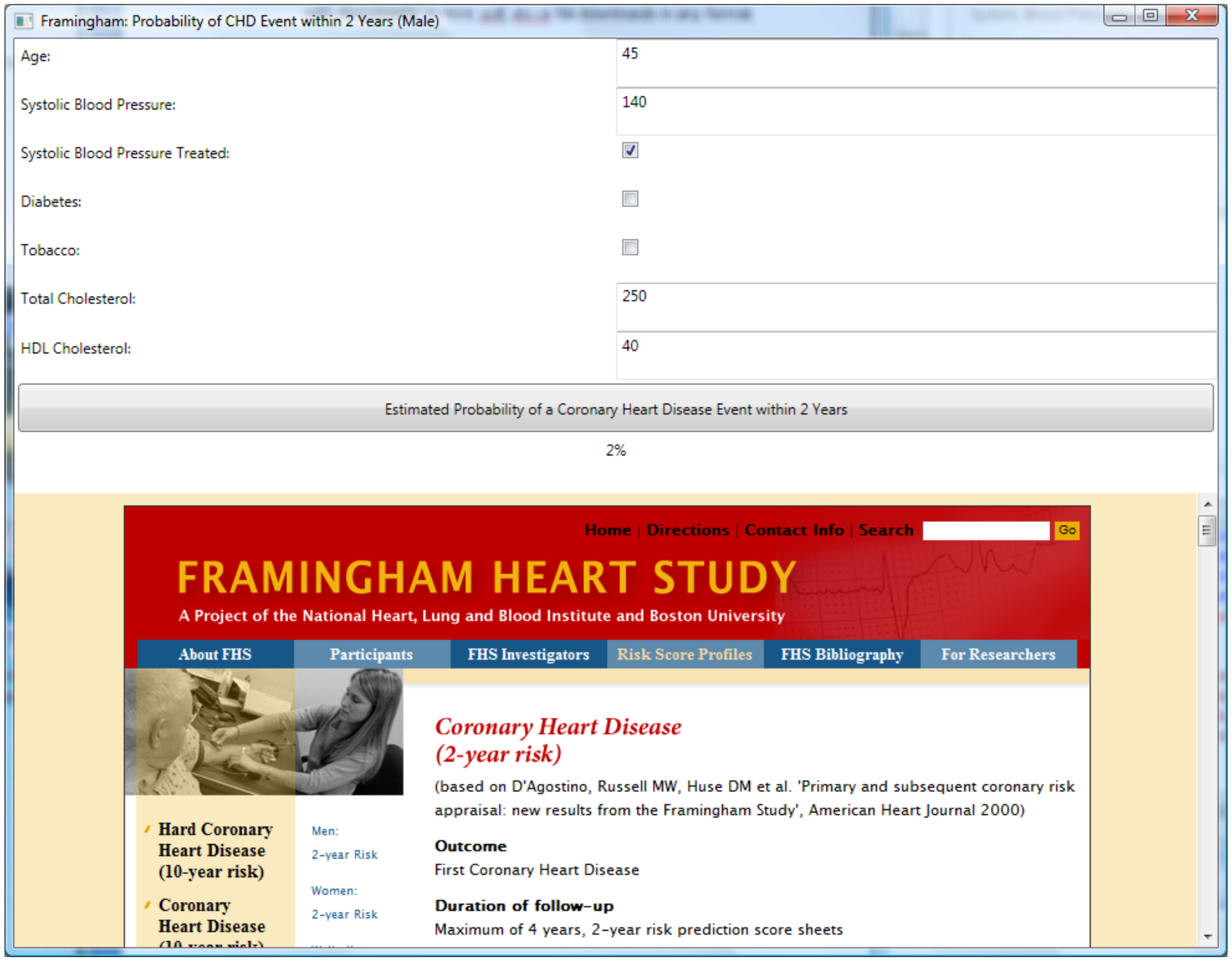
This application example is based upon the long-term Framingham heart disease study. The GUI collects client values for the risk factors identified by the Framingham study. When the client submits the request for analysis, an APL+Win function calculates the risk of a coronary heart disease event within two years, based on the risk factors and disease occurrence profiles identified by Framingham study. When the results are presented to the client, the Framingham study home page (http://www.framinghamheartstudy.org/) is also presented in a WPF browser control to provide the client with interpretation of the results.

The application design provides the most up-to-date GUI for the Windows environment and utilizes APL WebServices technology to efficientlysupport any number of *ad hoc* clients of the application.



In the screen shot below the client has entered the values for the risk factors and then clicked the ‘Estimated Probability…’ button to display the probability estimate.

The GUI for this application system took only a few minutes to develop and is completely scrollable and re-sizable. The calculations are supported by APL+Win functions which incorporate the Framingham parameters and equations. With a bit of additional work by a graphic designer the GUI could certainly be cosmetically enhanced.



Framingham: Probability of CHD Event within 2 Years (Male)

Age:	45
Systolic Blood Pressure:	140
Systolic Blood Pressure Treated:	✓
Diabetes:	☐
Tobacco:	☐
Total Cholesterol:	250
HDL Cholesterol:	40

Estimated Probability of a Coronary Heart Disease Event within 2 Years

2%

Home | Directions | Contact Info | Search
Go
FRAMINGHAM HEART STUDY
A Project of the National Heart, Lung and Blood Institute and Boston University
About FHS
Participants
FHS Investigators
Risk Score Profiles
FHS Bibliography
For Researchers
Hard Coronary Heart Disease (10-year risk)
Coronary Heart Disease
Men:
2-year Risk
Women:
2-year Risk

Coronary Heart Disease (2-year risk)

(based on D'Agostino, Russell MW, Huse DM et al. 'Primary and subsequent coronary risk appraisal: new results from the Framingham Study', American Heart Journal 2000)

Outcome
First Coronary Heart Disease

Duration of follow-up
Maximum of 4 years, 2-year risk prediction score sheets

- A graphical user interface (GUI) has been created using Microsoft Visual Studio 2008 and Windows Presentation Foundation (WPF). WPF is the latest Microsoft-designed, .Net 3.5 GUI methodology. It uses an XML-format ‘XAML’ file to define the GUI control and a C# ‘code-behind’ file for the event handlers. WPF represents a fundamental enhancement to Windows application development environment because it permits the creation of the GUI by a graphic designer. This GUI could also have been created using Microsoft Expression Blend / Silverlight. Expression Blend which does not require a programmer operator to create the GUI.

  Here is the XAML file which describes this GUI. You can see that this is very compact and quite clear due to the used of mnemonic control names, properties and events.

  - The entire GUI window is contained in a ScrollViewer control to facilitate re-sizing by the client.
  - Inside the ScrollViewer control is a 4-row grid:
    - Row #1 of this grid contains a subordinate grid with 7 rows and 2 columns containing the labels, textboxes and checkboxes for client input of the values of the risk factors.
    - Row #2 of this grid contains the button for the client to submit the request for the probability of CHD event. The button event handler function ‘bn_Calc_Click’ is specified as a button control ‘Click=’ property.
    - Row #3 of this grid contains the textblock to receive the calculated probability%.
    - Row #4 of this grid contains the browser control to present the Framingham web page.

```
<Window x:Class="WPF_Framingham_2YrProbCHDEvent.Window1"
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  Title="Framingham: Probability of CHD Event within 2 Years (Male)" Height="432" Width="545" Loaded="Window_Loaded">
  <ScrollViewer Name="SV1" VerticalScrollBarVisibility="Auto" HorizontalScrollBarVisibility="Auto">
    <Grid Name="G1">
      <Grid.RowDefinitions>
        <RowDefinition Height="6*"></RowDefinition>
        <RowDefinition Height="*"></RowDefinition>
        <RowDefinition Height="*"></RowDefinition>
        <RowDefinition Height="8*"></RowDefinition>
      </Grid.RowDefinitions>
      <Grid Name="GInput" Grid.Row="0">
        <Grid.RowDefinitions>
          <RowDefinition></RowDefinition>
          <RowDefinition></RowDefinition>
          <RowDefinition></RowDefinition>
          <RowDefinition></RowDefinition>
          <RowDefinition></RowDefinition>
          <RowDefinition></RowDefinition>
          <RowDefinition></RowDefinition>
        </Grid.RowDefinitions>
        <Grid.ColumnDefinitions>
          <ColumnDefinition></ColumnDefinition>
          <ColumnDefinition></ColumnDefinition>
        </Grid.ColumnDefinitions>
        <Label Grid.Row="0" Grid.Column="0">Age:</Label>
        <Label Grid.Row="1" Grid.Column="0">Systolic Blood Pressure:</Label>
        <Label Grid.Row="2" Grid.Column="0">Systolic Blood Pressure Treated:</Label>
```

```
            <Label Grid.Row="3" Grid.Column="0">Diabetes:</Label>
            <Label Grid.Row="4" Grid.Column="0">Tobacco:</Label>
            <Label Grid.Row="5" Grid.Column="0">Total  Cholesterol:</Label>
            <Label Grid.Row="6" Grid.Column="0">HDL Cholesterol:</Label>
            <TextBox Name="TBX_Age" Grid.Row="0" Grid.Column="1">45</TextBox>
            <TextBox Name="TBX_SBP" Grid.Row="1" Grid.Column="1">140</TextBox>
            <CheckBox Name="cb_SBPT" Grid.Row="2" Grid.Column="1" Margin="5,5,5,5" ></CheckBox>
            <CheckBox Name="cb_DIAB" Grid.Row="3" Grid.Column="1" Margin="5,5,5,5" ></CheckBox>
            <CheckBox Name="cb_TOBAC" Grid.Row="4" Grid.Column="1" Margin="5,5,5,5" ></CheckBox>
            <TextBox Name="TBX_TCHOL" Grid.Row="5" Grid.Column="1">200</TextBox>
            <TextBox Name="TBX_HDLCHOL" Grid.Row="6" Grid.Column="1">40</TextBox>
         </Grid>
         <Button Name="bn_Calc" Grid.Row="1" Grid.Column="0" Grid.ColumnSpan="2" Margin="3,3,3,3" Click="bn_Calc_Click">Estimated Probability of a
Coronary Heart Disease Event within 2 Years</Button>
         <TextBlock Name="tbk_Result" Grid.Row="2" Grid.Column="0" Grid.ColumnSpan="2" Margin="3,3,3,3" TextAlignment="Center"></TextBlock>
         <WebBrowser Name="wb1" Grid.Row="3" Grid.Column="0" Grid.ColumnSpan="2"></WebBrowser>
      </Grid>
   </ScrollViewer>
</Window>
```

- Here is the This C# ‘code-behind’ program, created in Microsoft Visual Studio 2008, to link the GUI button click event to C# event handlers supporting the application’s calculations and business rules.

```
 using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;

namespace WPF_Framingham_2YrProbCHDEvent
{
    /// <summary>
    /// Interaction logic for Window1.xaml
    /// </summary>
    public partial class Window1 : Window
    {
        public Window1()
        {
            InitializeComponent();
        }

        private void bn_Calc_Click(object sender, RoutedEventArgs e)
        {
```

```
double Age;
double SBP;
bool SBPT;
bool DIAB;
bool TOBAC;
double TCHOL;
double HDLCHOL;

if (!Double.TryParse(TBX_Age.Text, out Age))
{
    MessageBox.Show("Enter a valid age");
}
else if (!Double.TryParse(TBX_SBP.Text,out SBP))
{
    MessageBox.Show("Enter a valid SBP");
}
else if (!Double.TryParse(TBX_TCHOL.Text,out TCHOL))
{
    MessageBox.Show("Enter a valid TCHOL");
}
else if (!Double.TryParse(TBX_HDLCHOL.Text, out HDLCHOL))
{
    MessageBox.Show("Enter a valid HDLCHOL");
}
else
{
    SBPT=(bool) cb_SBPT.IsChecked;
    DIAB = (bool) cb_DIAB.IsChecked;
    TOBAC = (bool) cb_TOBAC.IsChecked;
```

```
APL2000.Utils.WebTransfer.WebTransfer wt = new APL2000.Utils.WebTransfer.WebTransfer();
object[] obj;
obj=wt.Open(@"http://localhost:9002");
if (0!=(int)obj[0])
{
    MessageBox.Show("Cannot open web server");
}
else
{
    object[] rarg = new object[7];
    rarg[0] = Age;
    rarg[1] = SBP;
    rarg[2] = SBPT;
    rarg[3] = TOBAC;
    rarg[4] = DIAB;
    rarg[5] = TCHOL;
    rarg[6] = HDLCHOL;
    obj = wt.SendObject(@"/framingham/do_FHS_1Arg", rarg);
    if (0!=(int)obj[0])
    {
        MessageBox.Show("Error transmitting data to web server");
    }
    else
    {
        tbk_Result.Text = obj[1].ToString()+"%";

        string string_uri=@"http://www.framinghamheartstudy.org/risk/coronary2.html#tab1"
;
        Uri uri = new Uri(string_uri, UriKind.RelativeOrAbsolute);
```

```
                        try
                        {
                            wb1.Navigate(uri);
                        }
                        catch
                        {
                            MessageBox.Show("Could not access web site: "+string_uri);
                        }
                    }
                }
            }
        }

        private void Window_Loaded(object sender, RoutedEventArgs e)
        {

        }
    }
}
```

- The C# program receives the GUI events – in this case a calculation button on the WPF window. The event handler function header is:

  ```
  private void bn_Calc_Click(object sender, RoutedEventArgs e)
  ```

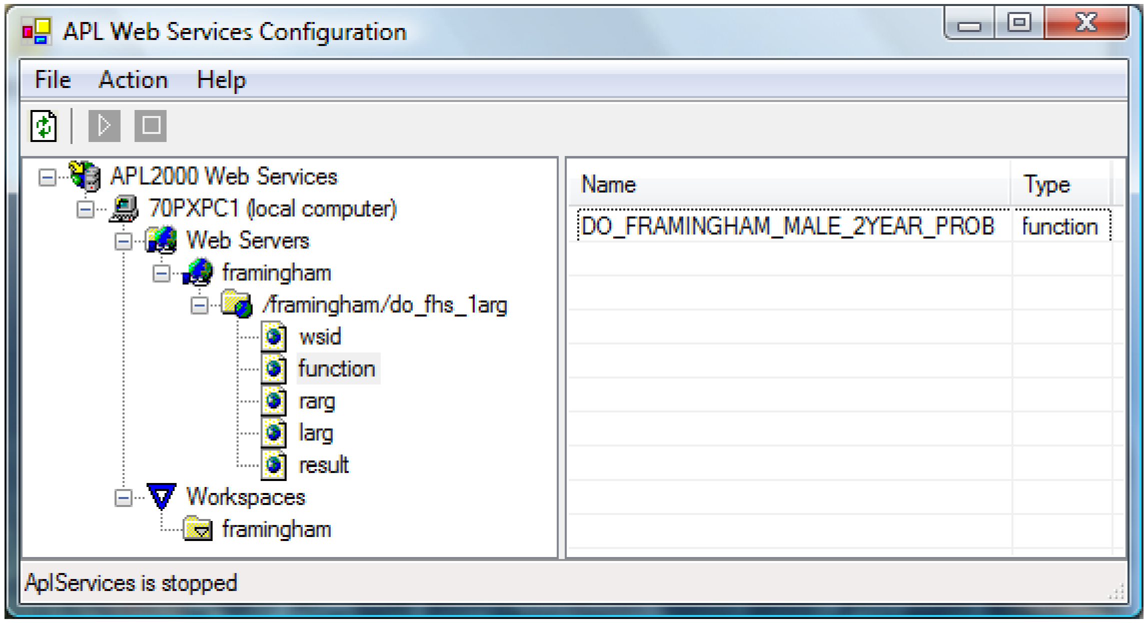
- The C# program validates the user input from the GUI controls on the WPF window using this code:

  ```
  if (!Double.TryParse(TBX_Age.Text, out Age))
  {
      MessageBox.Show("Enter a valid age");
  }
  else if (!Double.TryParse(TBX_SBP.Text,out SBP))
  {
      MessageBox.Show("Enter a valid SBP");
  }
  else if (!Double.TryParse(TBX_TCHOL.Text,out TCHOL))
  {
      MessageBox.Show("Enter a valid TCHOL");
  }
  else if (!Double.TryParse(TBX_HDLCHOL.Text, out HDLCHOL))
  {
      MessageBox.Show("Enter a valid HDLCHOL");
  }
  ```

- The C# program creates an instance, ‘wt’, of the APL WebServices WebTransfer class. Opens a connection to the APL WebServices server using the ‘wt.Open’ method of the APL WebTransfer class instance. Prepares the argument ‘rarg’ using the client input from the GUI. Sends the object, ‘rarg’, containing the user-entered information to APL WebServices using the ‘wt.SendObject’ method of the APL WebTransfer class instance. The ‘SendObject()’ method creates a synchronous request to the APL WebServices server, so that the C# event handler waits for a response from APL WebServices.

```
 … APL2000.Utils.WebTransfer.WebTransfer wt = new APL2000.Utils.WebTransfer.WebTransfer();
object[] obj;
obj=wt.Open(@"http://localhost:9002");
if (0!=(int)obj[0])
{
   MessageBox.Show("Cannot open web server");
}
else
{
   object[] rarg = new object[7];
   rarg[0] = Age;
   rarg[1] = SBP;
   rarg[2] = SBPT;
   rarg[3] = TOBAC;
   rarg[4] = DIAB;
   rarg[5] = TCHOL;
   rarg[6] = HDLCHOL;
   obj = wt.SendObject(@"/framingham/do_FHS_1Arg", rarg);
 if (0!=(int)obj[0])
 {
    MessageBox.Show("Error transmitting data to web server");
 }
 else
 {         …
```

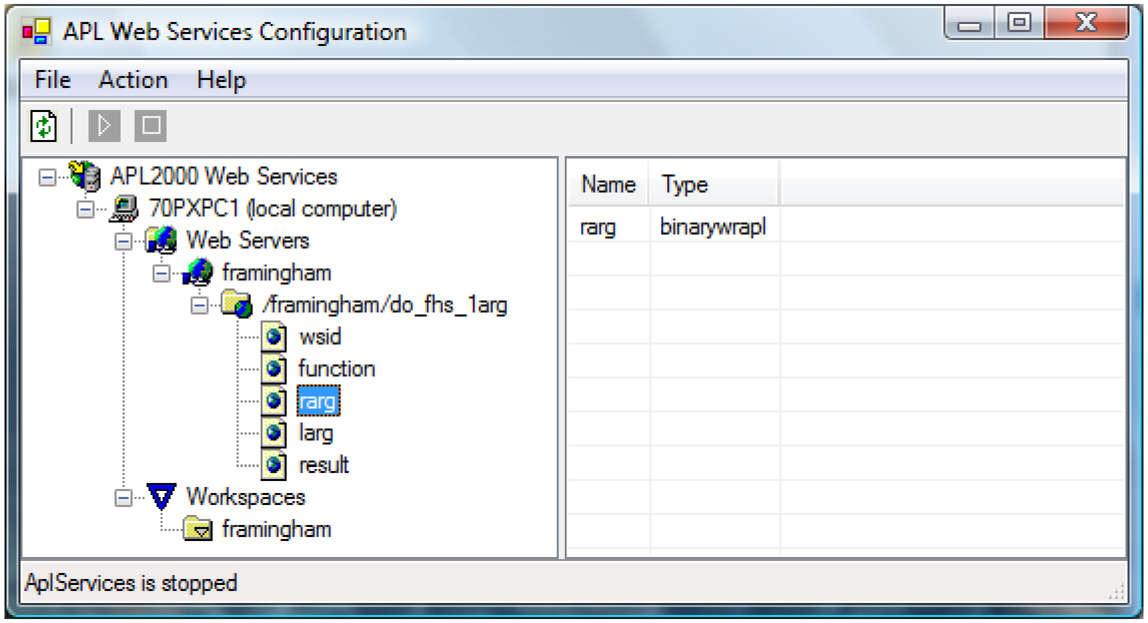
APL WebServices receives the client request and provides it to an APL+Win function in an APL+Win workspace behind APL WebServices. The machine, '70PXPC1', has APL WebServices installed. The 'framingham' web server is configured with the '/framingham/do_fhs_1arg' virtual path. This virtual path points to the APL+Win function 'DO_FRAMINGHAM_MALE_2YEAR_PROB' which will perform the calculations necessary to satisfy the client request. This function is contained in the 'framingham.w3' workspace.



Name	Type
DO_FRAMINGHAM_MALE_2YEAR_PROB	function

AplServices is stopped

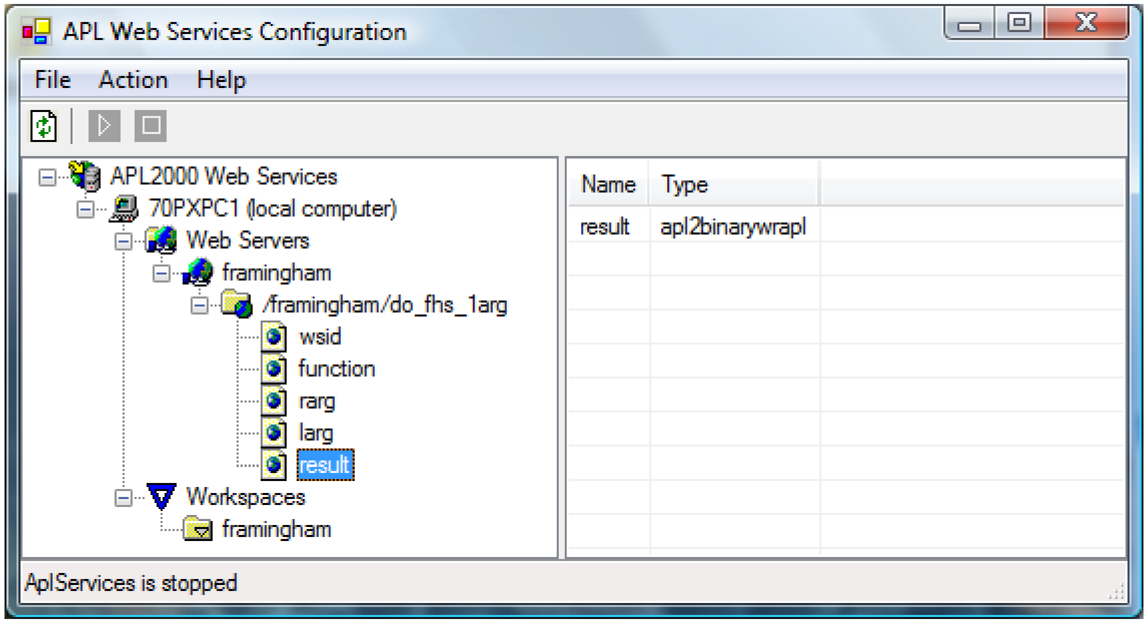
The function ‘DO_FRAMINGHAM_MALE_2YEAR_PROB’ has been configured with a single right argument (‘rarg’), because the information required by this function is transmitted to APL WebServices by the GUI as a C# 7-element array of objects.



Name	Type
rarg	binarywrapl

ApIServices is stopped

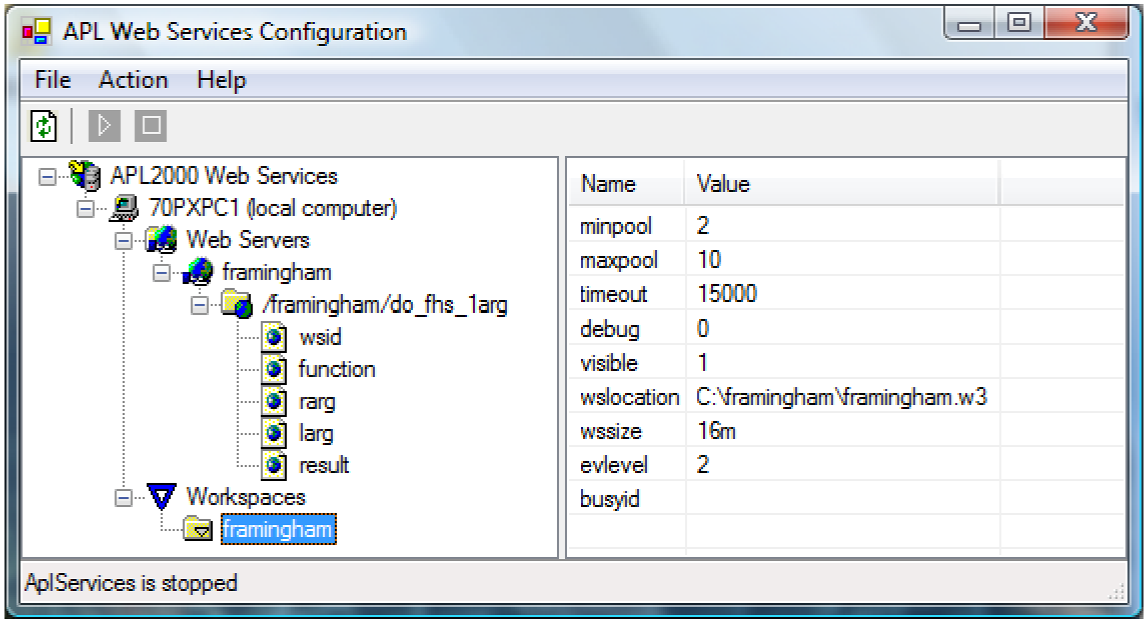
The function ‘DO_FRAMINGHAM_MALE_2YEAR_PROB’ has been configured with a single result (‘result’), because the information returned by this function is transmitted to the GUI as a C# 2-element array of objects. The first element of this array is provided by APL WebServices with value 0 if there were no errors, and an error code otherwise). The second element of this array is the error message provided by APL WebServices or the return value of the ‘DO_FRAMINGHAM_MALE_2YEAR_PROB’ function.



Name	Type
result	apl2binarywrapl

ApIServices is stopped

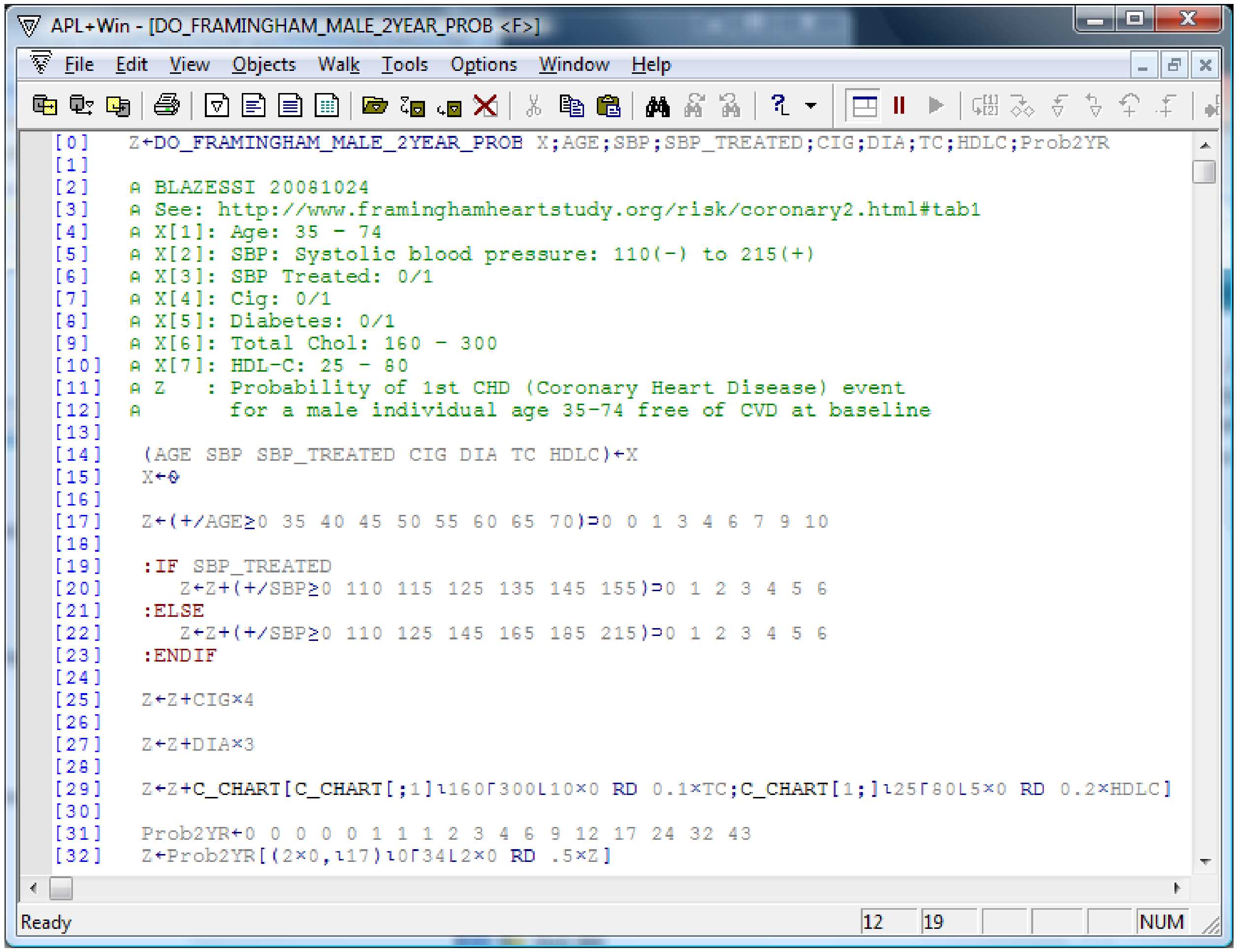
For load balancing purposes, this workspace on the server has been configured in APL WebServices to maintain 2 instances of the workspace ready to satisfy requests at all time and up to 8 additional instances of the workspace when user load increases. The busyid field has been left empty, but can be filled with the url of an additional web server to handle ‘overflow’ requests. This busyid field provides for complete scalability of an APL WebServices-based application system. In addition APL WebServices will automatically queue requests until they are satisfied by the application system, up to a programmer-specified timeout limit.



Name	Value
minpool	2
maxpool	10
timeout	15000
debug	0
visible	1
wslocation	C:\framingham\framingham.w3
wssize	16m
evlevel	2
busyid	

AplServices is stopped

- The APL+Win function calculates the application-specific values and provides the result to APL WebServices. The function uses a global variable C_CHART in the workspace containing a table of cholesterol risk profiles from the Framingham study.


APL+Win - [DO_FRAMINGHAM_MALE_2YEAR_PROB <F>]
File Edit View Objects Walk Tools Options Window Help
Ready
12
19
NUM

```
    Z←DO_FRAMINGHAM_MALE_2YEAR_PROB X;AGE;SBP;SBP_TREATED;CIG;DIA;TC;HDLC;Prob2YR

    ⍝ BLAZESSI 20081024
    ⍝ See: http://www.framinghamheartstudy.org/risk/coronary2.html#tab1
    ⍝ X[1]: Age: 35 - 74
    ⍝ X[2]: SBP: Systolic blood pressure: 110(-) to 215(+)
    ⍝ X[3]: SBP Treated: 0/1
    ⍝ X[4]: Cig: 0/1
    ⍝ X[5]: Diabetes: 0/1
    ⍝ X[6]: Total Chol: 160 - 300
    ⍝ X[7]: HDL-C: 25 - 80
    ⍝ Z    : Probability of 1st CHD (Coronary Heart Disease) event
    ⍝        for a male individual age 35-74 free of CVD at baseline

     (AGE SBP SBP_TREATED CIG DIA TC HDLC)←X
     X←⍬

     Z←(+/AGE≥0 35 40 45 50 55 60 65 70)⊃0 0 1 3 4 6 7 9 10

     :IF SBP_TREATED
        Z←Z+(+/SBP≥0 110 115 125 135 145 155)⊃0 1 2 3 4 5 6
     :ELSE
        Z←Z+(+/SBP≥0 110 125 145 165 185 215)⊃0 1 2 3 4 5 6
     :ENDIF

     Z←Z+CIG×4

     Z←Z+DIA×3

     Z←Z+C_CHART[C_CHART[;1]⍳160⌈300⌊10×0 RD 0.1×TC;C_CHART[1;]⍳25⌈80⌊5×0 RD 0.2×HDLC]

     Prob2YR←0 0 0 0 0 1 1 1 2 3 4 6 9 12 17 24 32 43
     Z←Prob2YR[(2×0,⍳17)⍳0⌈34⌊2×0 RD .5×Z]
```

- APL WebServices transmits the APL+Win-determined result information back to the C# program.
- The C# program refreshes the WPF GUI to illustrate the results.

```
tbk_Result.Text = obj[1].ToString()+"%";
```

- The C# program displays the Framingham web page in the text block below the probability% result:

```
string string_uri=@"http://www.framinghamheartstudy.org/risk/coronary2.html#tab1" ;
                Uri uri = new Uri(string_uri, UriKind.RelativeOrAbsolute);
                try
                {
                    wb1.Navigate(uri);
                }
```

- This application can be easily incorporated into a traditional installer file for local installation on the end user's machine.
- With Visual Studio 2008, it is also easy to publish the GUI of this application system to a Windows web server as a '1-click install' application. When this is done, the end user navigates to the web site where the application system GUI was published and it will be automatically installed on the user's machine. The publisher can elect to have it permanently installed on the user's machine or only transiently installed, just like any other temporary Internet content file. The 1-click install process is unusually quick because only the application-specific code need by downloaded and installed, since the end user has already installed the Microsoft .Net framework and the application's calculations and business rules are in the APL+Win workspace behind APL WebServices and need not be installed on the user's machine.

All the code necessary for this application is included in the '\framingham' subdirectory. The C# code is in Unicode format, so it can be viewed in a Unicode-compatible text viewer such as Microsoft Notepad in case Visual Studio 2008 is not installed. The APL+Win workspace requires APL+Win v8.x.