Experiences with Dynamic Link Libraries

Youhong Liu & Gina-Qian Cheung (University of Michigan, USA)

1. Introduction

Blaise supports the use of Dynamic Link Libraries (DLLs) to perform a process or action that is not currently available in Blaise itself. An example is a DLL used as an external subroutine that can be called by the Blaise Data Entry Program (DEP) or Manipula. The Blaise instrument passes information to the DLL and the DLL acts on the information and passes the modified or new information back to the instrument. Then the Blaise instrument stores or acts on the modified information. There are two types of alien or external references. Those that perform calculations are called alien procedures and those that ask questions are called alien routers.

Previously, DLLs could only be developed using Borland Delphi. The Delphi DLL itself would then communicate with DLLs developed using a different development environment, such as C++ or VB. From Blaise version 4.6 forward, alien procedures and alien routers in data models and Manipula setups also support the use of ActiveX. The external subroutines now may be implemented by using a COM object method.

This paper describes how the University of Michigan's Survey Research Center (SRC) used an alien router technique to meet a study requirement that could not be met directly with Blaise. After describing the problem addressed and the solution implemented, we will discuss the advantages and disadvantages of the approach taken.

2. Background

In the fall of 2005, the decision was made to move the instrument for a project from paper and pencil to Blaise, in order to ensure better data collection quality. The instrument is very straight forward and does not contain very complex logic. But a significant portion of the survey asks respondents about events that happened during their lifetime. There was a need to have a calendar with grids for interviewers to fill out for each event. This required development of special grids to simulate what is called an Event History Calendar (EHC). Blaise does not provide an easy way to implement the interface and functionality of an EHC. Thus, SRC decided to create the EHCs with the alien router technique. The Visual Basic (VB) ActiveX Com option was selected for creating this DLL. VB was chosen since it is a language very familiar to programmers and we already had some experience working with VB and the Blaise Component Package (BCP).

3. How It Works

The questionnaire is designed to ask filter questions that pass control to the external subroutine if the response is affirmative. For example, if the respondent answers "Yes" to the question "Did you ever live on a property where pets frequently went in and out of your home?" (Figure 1), then the alien router EHC window would open (Figure 2). If the response were "No," Blaise would continue to the next field.

Figure 1: Filter Question C1 Used to Trigger Alien Router Call to EHC

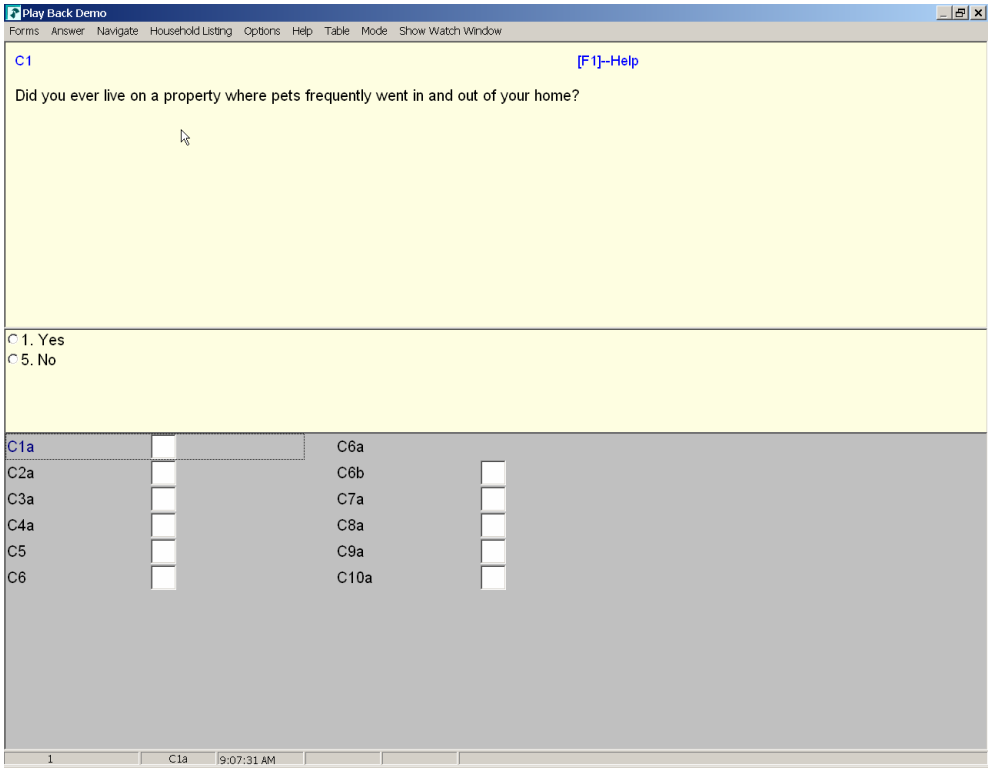
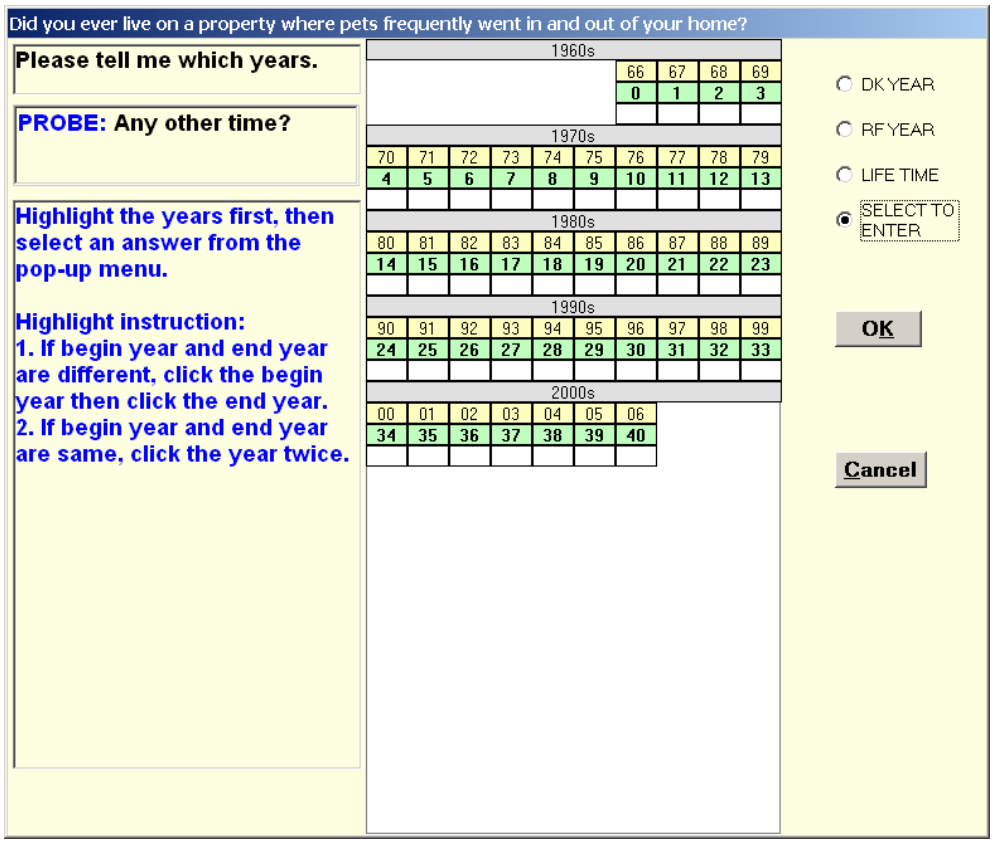


Figure 2: EHC Displayed by Alien Router Call after Filter Question C1

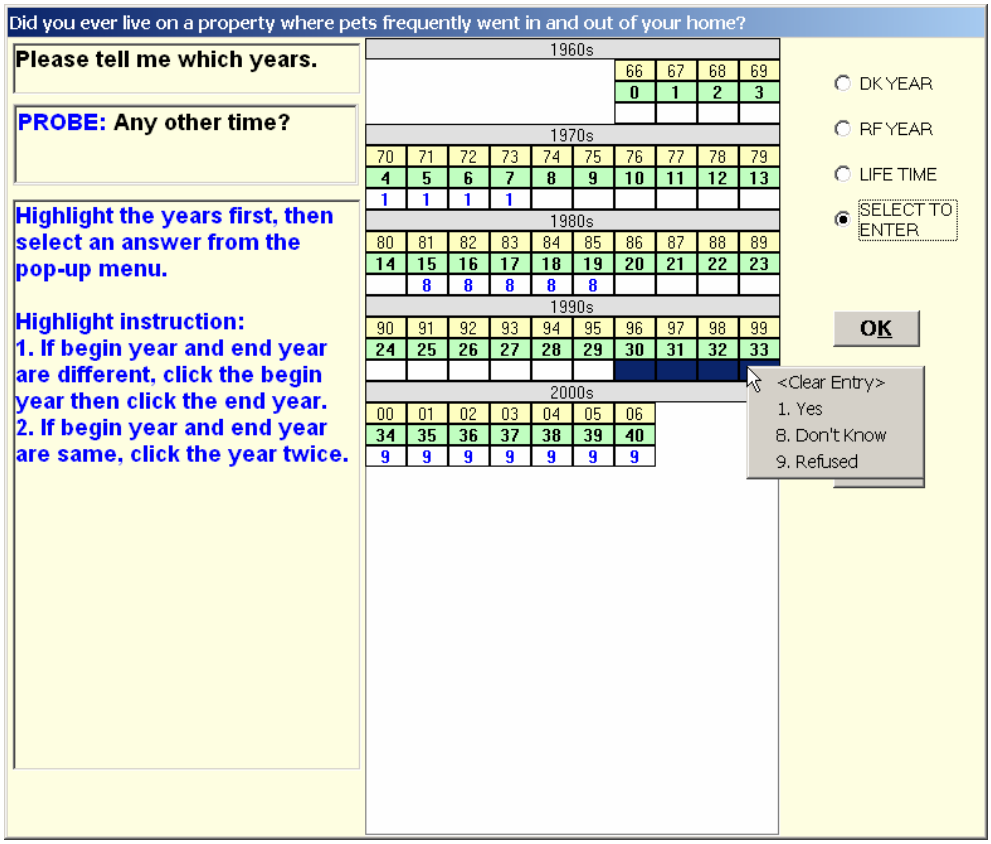


This is a Visual Basic form implemented in the DLL called by Blaise Alien router. The calendar's grids are displayed based on the respondent's birth year. The yellow grid indicates the year, the green grid represents the respondent's age, and the gray bar identifies the decade. All are for display only, so the Interviewer can navigate easily. The white grid is the only one used for data entry. The color grids

are VB label controls. All white grids are cleared when the calendar is accessed the first time. In the background, the DLL will read all information passed from the Blaise database, and save it for later use.

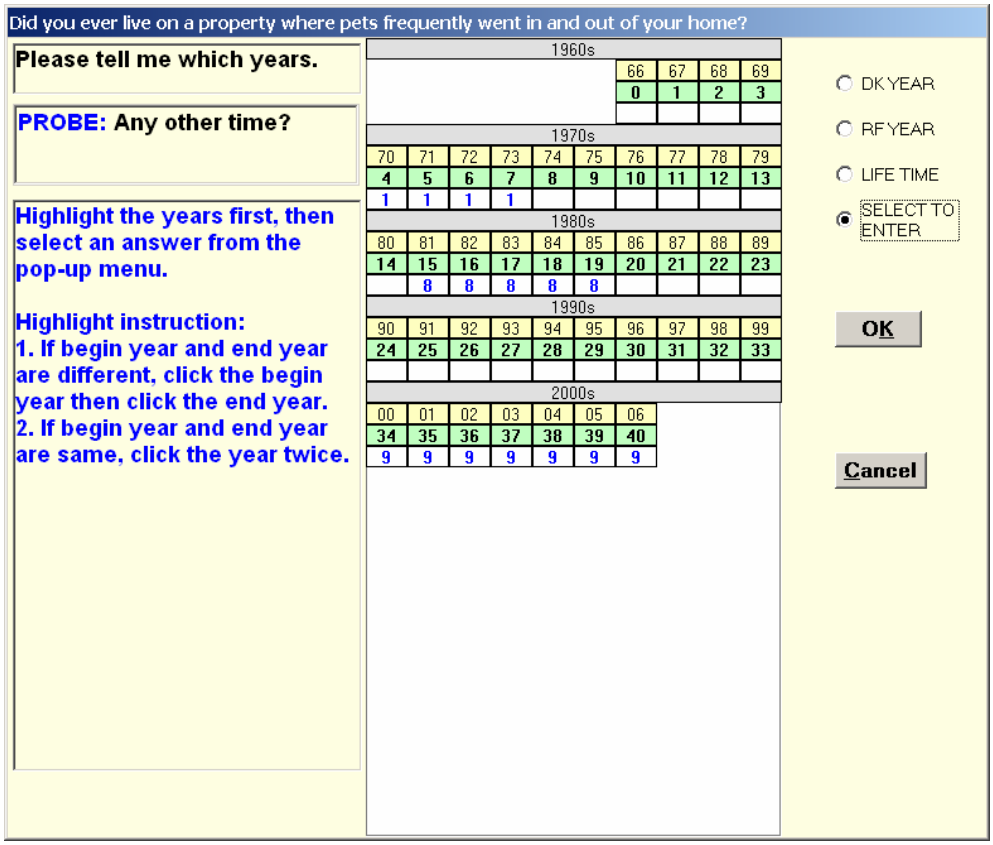
Normally, an Interviewer will highlight a few input fields and choose an option from the pop-up menu (Figure 3). If an Interviewer selects "DK Year", "RF Year," or "LIFE TIME", the response grids will be filled as 8, 9, or 1 respectively. After selections are completed, and "OK" is clicked, the DLL will send data to the Blaise database. Then the calendar is closed and the control is passed back to Blaise.

Figure 3: EHC Highlighted Period for Which Interviewer Can Record That Event Occurred



If the same calendar is accessed later, the DLL will read the Blaise database and fill the appropriate response grids in the form load event (Figure 4). The Interviewer can start to modify the data on the calendar and save it back to the Blaise database.

Figure 4: Prior Responses to Question C1 EHC Displayed Upon Reentry



4. Blaise Alien Router Block

Following is the source code for the Blaise alien router block that creates an EHC calendar grid, in this case for Question B1a, about ever living in households where members usually removed their shoes.

```
BLOCK RouterB1
      AUXFIELDS
            B1
            "BSec_B.B1.B1b~BSec_B.B1.B1_DK_Year~BSec_B.B1.B1_
            LifeTime~BSec_B.B1.B1_RF_Year" /
            {This is the field text of B1 which is used to pass field
            name list to the DLL, so DLL knows which field values need to
            be updated. The reason we want to pass the field list to the
            DLL is that we can use one calendar for multiple events.}
            "Did you ever live on a property where most
            household members usually removed their shoes
            before coming into the house?: STRING
            {This is the field description. It is used to be populated as
            EHC's title.}
      FIELDS
            B1a (B1a) "Did you ever live on a property where
            most household members usually removed their
            shoes before coming into the house?" : TYesNo

            ROUTER     Calendar     ALIEN('Dioxin.clsDioxin',
            'Calendar')
            {Calling the DLL from here. The DLL name is Dioxin, the class
            name is clsDioxin, and the sub name is Calendar. B1a is the
            field to access the alien router, and the EHC will be opened
            up if "Yes is answered.}
            B1b (B1b) "Please tell me which years?" :
            ARRAY[1..111] OF 0..1
```

```
            {First parameter in B1, corresponding to the white grids in
            the EHC, e.g. B1b[1]=>1900, B1b[2]=>1901… B1b[111]=>2010}
            B1_DK_Year (B1_DK_Year) : (Yes, No)
            {Second parameter in B1, DK option on the EHC}
            B1_LifeTime (B1_LifeTime ) : (Yes, No)
            {Third parameter in B1, LifeTime Option on the EHC}
            B1_RF_Year (B1_RF_Year ) : (Yes, No)
            {Fourth parameter in B1, RF option on the EHC}
      LOCALS
            I : INTEGER
      RULES
            B1a
            IF B1a <> Yes THEN
            {Empty all values; we need to do this here to clean up if the
            root question is not "Yes"}
                  FOR I := 1 TO 111 DO
                        B1b [i]:= EMPTY
                  ENDDO
                  B1_DK_Year := EMPTY
                  B1_LifeTime := EMPTY
                  B1_RF_Year := EMPTY
            ELSE
            {Values updated by the DLL that need explicit keep.}
                  FOR I := 1 TO 111 DO
                        B1b[i].KEEP
                  ENDDO
                  B1_DK_Year.KEEP
                  B1_RF_Year.KEEP
                  B1_LifeTime.KEEP
            ENDIF
ENDBLOCK
```

5. VB ActiveX DLL

There are many functions in the VB ActiveX DLL, and we show two of them below. The first is the function in the clsDioxin class. It is the connection point between Blaise and the DLL. The database or DB parameter gives access to the data and metadata that are currently present in the Data Entry Program (DEP). db.Field.([FieldName]) can be used to access any field in the database. DS is the DepState object that gives access to the state of the Data Entry Program. For example, one can use the DepState object to determine the name of the currently active Field in the Data Entry Program.

```
Public Sub Calendar(db As BlAPI4A2.Database, ds As
BlAPI4A2.DepState)
Set dbp = db
Set dsp = ds
Set Layout =
dbp.Screens.LayoutSetCollection(dsp.LayoutSetIndex)
Set Parallel = Layout.ParallelCollection(dsp.ParallelIndex)
Set Page = Parallel.StoredPageCollection(dsp.StoredPageIndex)
Set Quest = Page.QuestionCollection(dsp.QuestionIndex)
Dim PassedNamestr() As String
PassedNamestr =
Split(Quest.Field.Parent.Fields(blfkAuxField).Item(1).Questio
nText(1, False), "~")
      {B1's question text contains the list of fields that will be modified
      by the DLL}
DKYearName = ""
AllLifeName = ""
RFYearName = ""
If UBound(PassedNamestr) >= 0 Then
  RootName = PassedNamestr(0)
```

```
End If
If UBound(PassedNamestr) >= 1 Then
        DKYearName = PassedNamestr(1)
End If
If UBound(PassedNamestr) >= 2 Then
        AllLifeName = PassedNamestr(2)
End If
If UBound(PassedNamestr) >= 3 Then
     RFYearName = PassedNamestr(3)
End If
formCaption                                                    =
Quest.Field.Parent.Fields(blfkAuxField).Item(1).DescriptionTe
xt(1, False)
{B1's description contains the EHC's title text}
YearBorn = dbp.Field("YearBorn").Value
{"YearBorn" is in the datamodel; the EHC grids are displayed according to
it's value}
If ds.AlienRouterStatus = blrsPostEdit Then
      If Quest.Field.Value = 1 Then
            Form1.Show vbModal
            {Form1 is the EHC form that only opens the EHC, when question
            value is "Yes"}
      End If
End If
If ds.AlienRouterStatus = blrsEdit Or _
      ds.AlienRouterStatus = blrsPreEdit Then
      ds.AlienRouterAction = blraEditQuestion
      {This is necessary here, so the alien router questions can take
      input}
End If
End Sub
```

The next subroutine is behind the "OK" button on the EHC form. It updates the data in the database according to the entries on the form.

```
Private Sub cmdOK_Click()
Dim i  As Integer
Dim FieldName As String
Dim answer As Integer
Dim msgstr As String
If Entry_check(msgstr) = False Then
      answer = MsgBox(msgstr & "Do you like to proceed?",
      vbYesNo + vbExclamation)
      {Make sure at least one grid has value}
End If
If answer = vbNo Then
     Exit Sub
End If
setValueEmpty
{Initialize}
If opt1(0).Value = True Then
      dbp.Field(DKYearName).Text = "1"
      {Update DKYear field in the BDB}
ElseIf opt1(3).Value = True Then
      dbp.Field(RFYearName).Text = "1"
      {Update RFYear field in the BDB}
ElseIf opt1(1).Value = True Then
       dbp.Field(AllLifeName).Text = "1"
      {Update AllLife field in the BDB}
      {OK if AllLife EMPTY then will not come to this path}
End If
For i = YearBorn - 1900 + 1 To Year(Now()) - 1900 + 1
      {Update event array in the BDB according to the caption on the
      labels}
      DoEvents
      FieldName = RootName + "[" & i & "]"
```

```
      If lblResult(i).Caption <> "" Then
            If lblResult(i).Caption = "8" Then
                  dbp.Field(FieldName).Status = blfsDontKnow
            ElseIf lblResult(i).Caption = "9" Then
                  dbp.Field(FieldName).Status = blfsRefusal
                     Else
                  dbp.Field(FieldName).Text                  =
                  CStr(lblResult(i).Caption)
            End If
      End If
Next i
Unload Me
End Sub
```

6. Multiple Instances of the Same Calendar

In the study, there are over 30 EHCs. If the developer programs one subroutine for each EHC, it would require lots of programming time and testing, and the code would be difficult to maintain. The technique here is to use one routine for all instances of the EHC, and to use an auxfield to pass in the field names to the DLL. This way the DLL knows which fields to update.

The DLL can be programmed to be quite flexible. For example the DLL subroutine described here allows passing a special character string from Blaise to the EHS, so the subroutine knows how to process information differently based on filter question or location in instrument. For example, the lifetime options are not applicable to the events in Section D of the Dioxin study. As a result, an “Empty” was passed into the lifetime column. When the subroutine sees this, it makes the “LIFE TIME” option ratio button invisible and when it is time to update the database fields, it skips updating the LifeTime field.

7. Conclusion and Summary

DLLs are external routines that can accomplish many complex tasks in a flexible fashion. At the University of Michigan, there are several studies using this technique. Our experiences with the alien router calls to an external EHC have been very positive and interviewers have found the interface user-friendly. With careful programming between Blaise and the DLL, the quality of data collection has improved significantly over the prior paper-and-pencil EHCs.

However, the DLL could create problems in the application without careful implementation and testing. For example, if we use a DLL in our Blaise program without the DLL physically existing in user’s machine or if it is not properly registered, the Data Entry Program will stop since it cannot find the DLL. Thorough testing of the whole system before production is essential.

Another disadvantage of using the DLL is that the data entered in the DLL program is not recorded in the Blaise Audit trail system. This makes it difficult for later data recovery or examination. We have thought that for future implementations the data should be written to an Access database or a text file, so the data are backed up in case the Blaise database becomes corrupt.

8. Acknowledgement

We would like to acknowledge following SRC programmers for their help on the VB DLL programming and input on this paper: Sujatha Venkatapuram, Peter Sparks and Jim Hagerman

9. References

Blaise OnLine Assistant (release 4.7.1000). (2005). Retrieved February 24, 2006 from World Wide Web:http://www.blaise.com/onlinehelp/