Impressions of Basil

Kathleen O'Reagan, Richard Frey and Karen Robbins, Westat

1. Introduction

Basil is an alternative data entry program for Blaise questionnaires, specifically designed to handle self interviewing situations. Since Basil combines the strength of the Blaise rules engine with the flexibility of Maniplus, which allows programmers of Basil to create tailor-made applications, we have been testing the use of these capabilities for process and workflow management. This paper describes the overall functional requirements, the design and subsequent coding techniques used, including the integration of Maniplus, Basil and SQL Server, and lessons learned when incorporating Basil in a tracing system project.

2. Tracing System Requirements

2.1 General Tracing Systems

In most long term surveys, losing track of some participants is inevitable. Tracing is the process of locating participants who can no longer be contacted using any of the location information on record. A variety of tracing methods and sources of location information are used in an attempt to find a participant. Information gathering sources and methods include phone numbers, emails, letters, relatives, friends, coworkers, neighbors, schools, community organizations, places of worship, directory assistance, and government offices such as voter registration, taxation, department of motor vehicles, and social services. Most of these sources can be researched simultaneously to locate the participant in a quick and effective manner. Timely tracing is a key component in continued data collection. When the participant has been located, all the data on record will be confirmed and updated in the study's survey management system. The SMS will also contain the historical records of the participant since joining the study.

Tracing systems can be can be made more intricate with the addition of a case management component to include supervisor review, roles and permissions to access functions and data, status codes milestones and reporting. For the purposes of this paper, we are addressing the data collection and verification activities of a tracing system to show how Basil is used.

2.2 Westat Tracing System

The functional requirement of a tracing system is to obtain location information about the participant. The Westat Survey Management System (SMS) contains the participant's name, physical and mailing addresses, email address, home, work, and cell phone numbers, and the name, address, phone and email of contacts who would know how to reach the participant in case their phone number or address changed during the course of the study. Tracers have a wide variety of sources to use to produce leads to the missing participant. When the participant is successfully located, they will be asked to review and update their information on record.

In the Westat Tracing Logical Flow (Figure 1), the Westat SMS Workflow issues assignments to the interviewer to conduct a survey at the participant's residence, to the Telephone Research Center (TRC) to make a followup call to the participant's home, work, or cell phone, and to send letters to the participant's physical , mailing, and email addresses .

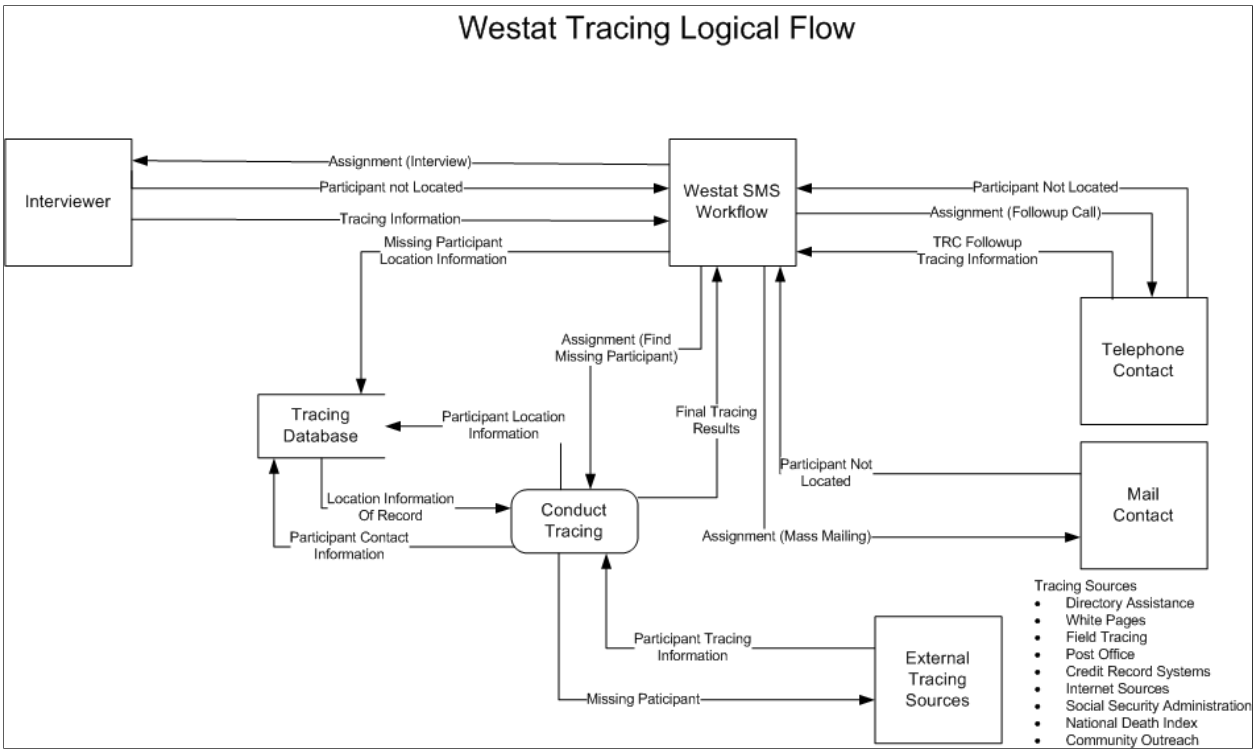

Figure 1. Westat Tracing Logical Flow

The tracing task of the Westat Tracing Logical Flow begins when the participant is not located. Responses from the workflow assignments signal that part of the information on record may no longer be correct. The interviewer might have discovered the participant's address on file is vacant. The TRC could have received a 'disconnected' message when they call one of the provided phone numbers. From the mail contact source, the letters or email could have been returned as not deliverable. With this recognition of having incorrect data for a participant, the complete record of that missing participant, their personal information and their contacts' information, is passed from the SMS to the Tracing Database.

The tracer conducts the search for the participant in stages. First, the tracer tries to locate the participant using the various phone numbers, addresses and email that they provided. If the participant was not found, attempts to reach each contact reference provided by the participant will be made by phone, mail and email.

When the tracer has exhausted all the location means provided by the participant and still has not located the participant, tracing focus shifts to investigating external sources. These sources are less cost efficient and are therefore used last due to their history of yielding lower results. Examples of external sources include directory assistance, post office, field tracing, internet searching, community outreach and government offices such as the Social Security Administration, Department of Motor Vehicles, and social services, and local businesses for instance credit card companies.

Whenever the tracer is successful in reaching the participant, the tracer reviews and updates all of the participant's new location information and the participant's contact references information. At study specified intervals, the SMS will retrieve this verified update information and add it to the SMS database.

3. Using Basil for the Westat Tracing System

Blaise components, Basil, Manipula, and the relational database storage capabilities offered by Blaise Datalink, were incorporated into the latest prototype of Westat's Tracing System. This system allows for the tracing of missing participants using a variety of location sources, reporting and processing options. It includes a set of core capabilities and can be customized to meet the needs of specific project application requirements.

3.1 Tracing System Capabilities

At a minimum, the tracing system needed to provide core project requirements allowing for the inclusion of missing participant location information, the management of tracers, the tracking of location changes, and the ability to work external sources within the system.

The tracing functional requirements were matched against the Blaise components and implemented based on the amount of time the user was using the interface. For instance, adding or assigning a tracer is usually done once, therefore, using the DEP to add a tracer and give permissions did not require the sophisticated look and feel Basil would provide to the user interface.

Figure 2 shows the functional tracing requirements and the Blaise component used to implement the requirement.

Functional Requirement	Blaise Component
Controlling system	Manipula
Assign a tracer to a missing participant	Manipula and DEP
Add a tracer	Manipula and DEP
Delete a tracer	Manipula and DEP
Add a participant to the tracing system	Manipula and Datalink
Add a contact for the participant	Basil application and Datalink
Participant Tracing Information Sheet	Basil application and Datalink
Contact Tracing Information Sheet	Basil application and Datalink
External Source Work Sheet	Basil application and Datalink
Radio buttons, check boxes, and buttons	Basil input controls which trigger execution of a Maniplus procedure

Figure 2. Tracing Requirements

For the remainder of this paper, we will focus on the actions that use Basil as the basis for collecting and verifying location information. These include the verification of participant (P-TIS) and contact (C-TIS) information, and the searching of external sources.

The first screen viewed in the Westat Tracing System is shown in Figure 3. This is the main menu screen controlled by the Manipula setup, TracingStratup2. From this Manipula setup, the supervisor manages the tasks of adding a tracer to or deleting a tracer from the system, assigning a tracer to find a particular participant, and adding a missing participant to the system.

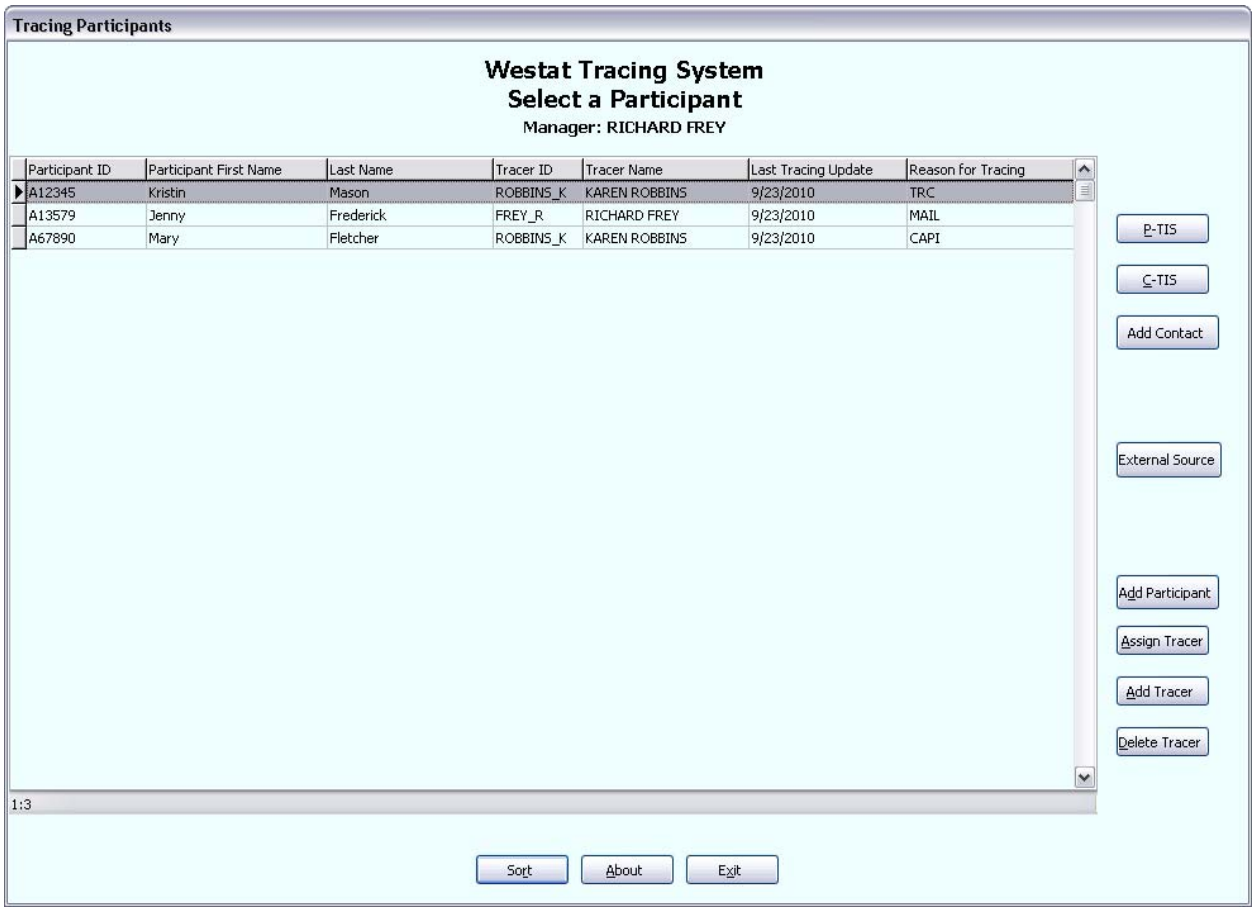

Figure 3. Manipula Setup for Tracing Participants

Tracing and location verification of the missing participant using the information of record utilizes the buttons P-TIS (Participant Tracing Information Sheet), C-TIS (Contact Tracing Information Sheet) and Add Contact.

If the participant and contact reference information on file hasn't led to locating the participant, then the the Manipula procedure launched by the External Source button will provide other means to locate the participant.

3.2 Basil, Manipula and Manipula Extensions

As mentioned in the paper's introduction Basil is an alternative data entry program for Blaise questionnaires, specifically designed to handle self-interviewing situations. Basil combines the strength of the Blaise rules engine with the flexibility of Maniplus. It allows for a layout with a diverse variety of field types. It can readily provide on a single screen one of a large variety of different types of standard data collection input forms. Within the context provided by the screen, users can enter the requested information in whatever order works for them. Moreover, users can take advantage of available buttons to launch an event associated with one or more inputted values.

The Manipula setting that allows for the communication between Basil and Manipula is Interchange. Interchange allows work directly on the record from the calling process, in this instance Basil. This is only possible when the data model of the calling process is the same as the data model used in the Manipula setup. Generally, this is done by using the reserved word VAR in the USES section of the

Manipula setup. Each Basil application identifies the Manipula setups, the datamodel and the update file in the Application section in the Basil datamodel.

Basil relies heavily on the usage of Manipula functions, methods and instruction extensions. The extensions are used in building procedures to enhance Basil controls. In the Westat Tracing application there are two Manipula setups. The one named Generic is found in the Blaise Samples from the Blaise installation. This setup contains generalized procedures for writing to an audit trail, error handling, printing, and exiting the questionnaire. The other setup is TracingProcedures in which procedures and dialogues specific to the tracing application are located. Specific procedures include those to handle button controls for dialing telephone numbers, collecting new address information and launching Basil pages. The Application section code in each Basil datamodel identifies each setup as follows:

```
BASIL "<application title='Westat Participant Tracing System'
            width=950 minimumwidth=800 Height=850 clientheight=825
            resource='Generic.dll'
            icon='BLAISE' sizeable=true fontface='Microsoft Sans Serif' fontsize=8
            fontcolor=#663300 color=#FFFFEF
            setups='TracingProcedures.msu /KdmGeneric=TracingTIS
                    /NufGeneric=TracingTIS;
                    Generic.msu /KdmGeneric=TracingTIS /NufGeneric=TracingTIS'
            oncreate='Generic.Initialisation'
            onclosequery='Generic.StopQuestionnaire'
            onhelp=blaise:exec('Generic.chm')
            onrangecheckerror='Generic.RangeCheckError'
            ongeneralerror='Generic.GeneralError'
            singleinstance=true>
          <panel left=0 align=client>
            <panel top=0 align=client color=#FFFFEF>
              <content-area horizontalscrollbar=false verticalscrollbar=true>
            </panel>
          </panel>
       </application>”
```

3.2.1 How Basil Was Used

On the Participant Tracing Information Sheet (Figure 4), you see a number of fields defined as a Blaise or user defined type according to Blaise syntax. They are each displayed as a label, input box or button using Basil layout controls.

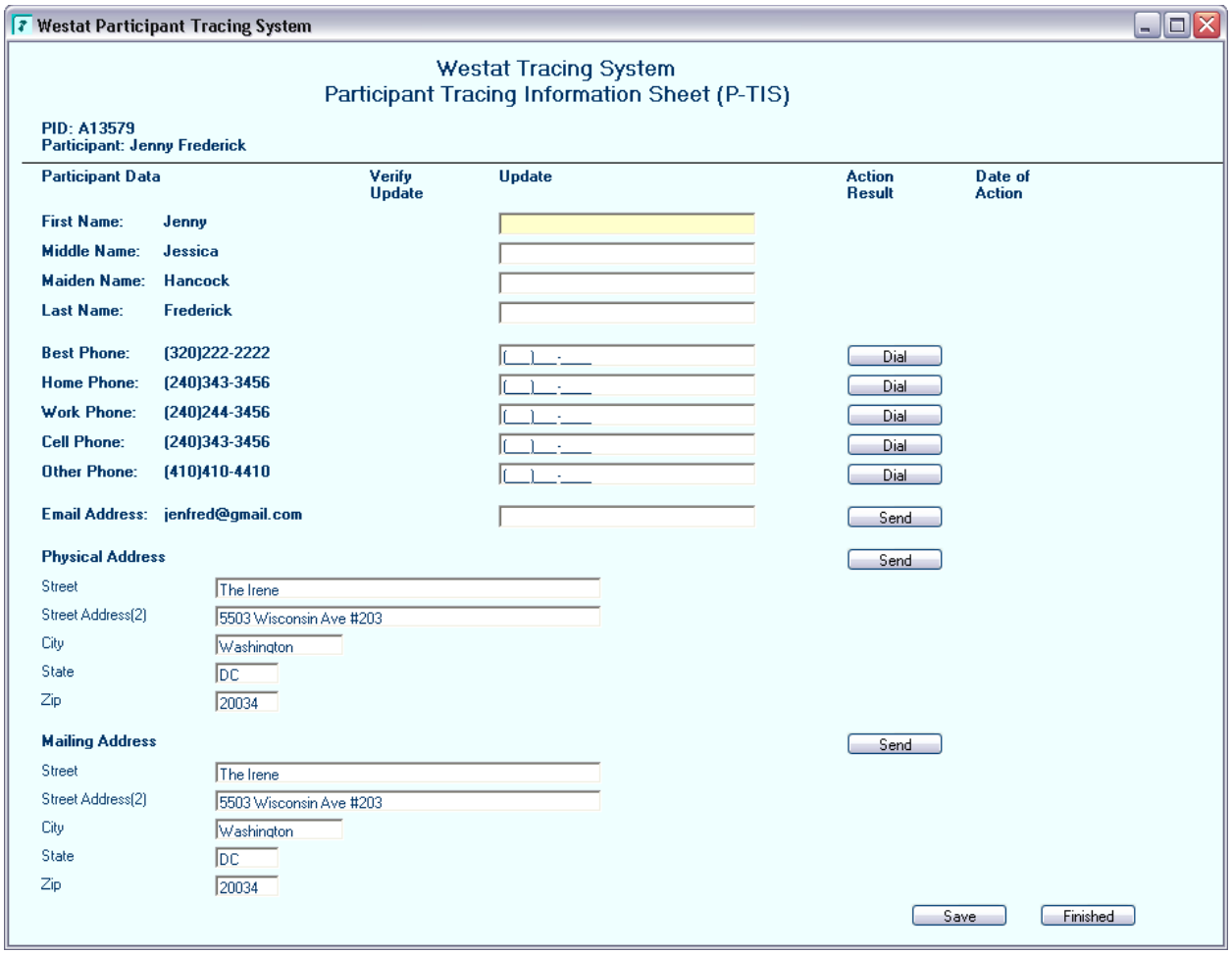

Figure 4. Participant Tracing Information Sheet

These controls are defined in the question text of the field similar to the way multimedia instructions are used in a Blaise multimedia instrument. The following code snippet shows the definition of the First Name field:

```
FirstName
      BASIL "<question span=5>
               <label left=25 width=100 topmargin=5 text='<b>First Name: </b>'>
               <label left=120 width=100 topmargin=5 text='<b>%TIS.FirstName </b>'>
             </question>"
      : STRING[20]
```

The FirstName field is defined as a string. Basil code uses two label controls. The first is used to display the label text ‘First Name:’. The second label control is used to display the current value of FirstName as indicated by the Basil fill string % symbol followed by the fully qualified field name. In the Rules, FirstName uses the .SHOW method.

The question attribute, span=5, indicates, the next five questions, including FirstName are to be grouped horizontally on one line.

The input box is used to hold any update made by the tracer to the FirstName field. The code for this field is below:

```
auxFirstName
      BASIL "<question>
              <input $auxAuxFieldPos>
             </question>"
      : STRING[20]
```

You'll notice the input control is followed by $auxAuxFieldPos. This is an AUXFIELD that holds the formatting of the input box. The $ is a fill string symbol. The following field assignment in the Rules section establishes the location, width, color and the 3D look.

auxAuxFieldPos := 'left=380 top=5 width=200 height=18 focusedcolor=#CCFFFF ctl3d=true'

The auxAuxFieldPos field is used for each name, telephone and email input box. Since we wanted each input box to display in a single column, using the fill string, auxAuxFieldPos, all the formatting attributes of each input box control were set the same.

The telephone input box has a mask to guide the entry of the telephone number. To display a mask the editmask attribute was added to the telephone input control.

```
<input $auxAuxFieldPos editmask='(999)999-9999' >
```

The last field of the First Name line is FirstNameDateLastAction. This field has the DATETYPE type and a Basil label control.

```
FirstNameDateLastAction
    BASIL"<question>
            <label topmargin=5 $auxDateLastAction
             text='<b>%tis.FirstNameDateLastAction </b>'>
          </question>"
     : DATETYPE
```

When the FirstName is updated, this field is also updated with the SYSDATE function. The Rules method is a .SHOW.

The last control to be discussed on the P-TIS screen is the button control on each telephone line. The button for the Best Phone is associated with the field named BestPhoneWhy which has the user defined type typPhoneStatus. The phone statuses include response, busy, no answer, answering machine, disconnected, no contact, dial and appointment.

```
BestPhoneWhy
    BASIL"<question>
           <button top=15 $auxDialButton
            caption='%tis.BestPhoneWhy'
            onclick='TracingProcedures.procUpdateDialStatus(""%tis.BestPhone"","
                     "BestPhone"","^TName"")'>
          </question>"
     : typPhoneStatus
```

The button control for this field uses the attributes, caption and onclick. The caption is the text that is displayed on the button and for this implementation uses the category code of the typPhoneStatus. At startup, the field is set to Dial. The onclick attribute indicates the Maniplus procedure that is to be called

when the button is clicked. In this case the procedure, procUpdateDialStatus in theTracingProcedures setup is called.

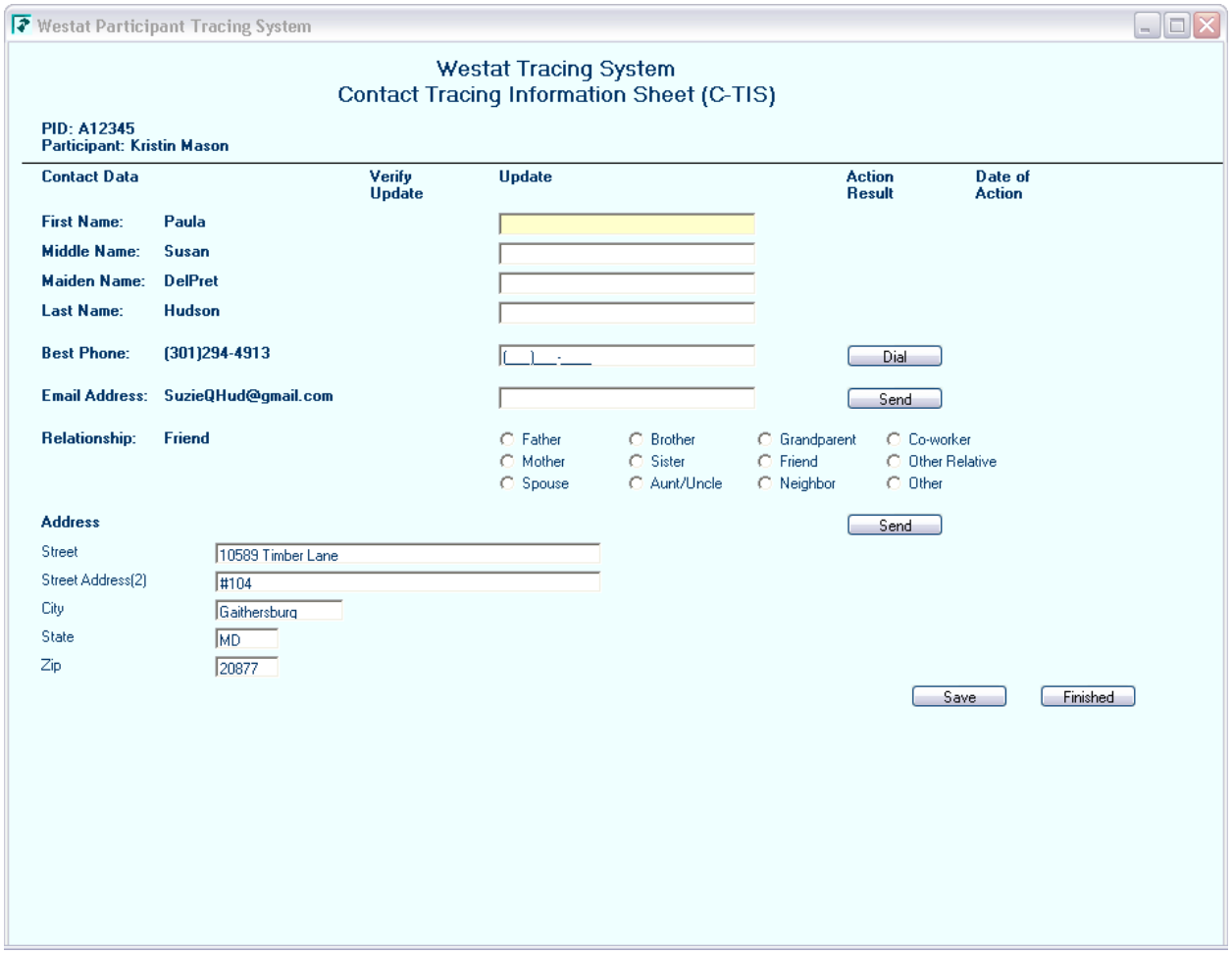

Figure 5. Contact Tracing Information Sheet

The Contact Tracing Information Sheet, as show in Figure 5, uses the same Basil instrument as the P-TIS and is stored in the same database. To differentiate between the participant and a contact, the primary key includes a contact number. A contact number of zero identifies the participant, any number greater is considered a contact.

In the P-TIS and C-TIS, an AUXFIELD is used to display the column headings.

```
EventLabel
    BASIL "<question>
            <line left=10 top=3 width=975 height=1 color=black>
            <label left=25 top=7 width=100 height=30 text='<b>^PartConTitle Data</b>'>
            <label left=280 top=7 width=100 height=30 text='<b>Verify<br>Update</b>'>
            <label left=380 top=7 width=100 height=30 text='<b>Update</b>'>
            <label left=650 top=7 width=200 height=30 text='<b>Action<br>Result</b>'>
            <label left=750 top=7 width=200 height=30 text='<b>Date of<br>Action</b>'>
          </question> “
    : STRING [1]
```

A separate label control was used for each heading in order to make it easier to align the headings with the fields. The line control adds the line above the headings just to give visual separation between the title information and the column headings.

The display of the Relationship field and of only one address is determined by the Rules when evaluating the ContactNumber. The Relationship field has the user defined enumerated type, typRelToParticipant. The input attributes for this control include tfillorder and columns.

```
auxRelationship
    BASIL "<question>
            <input  left=280 height=50 width=400 top=15 tfillorder=vertical columns=4>
           </question> "
    : typRelToParticipant
```

3.3.2 Interchange

Interchange and the Manipula extensions bring together a powerful combination allowing for real time access to data presented in a truly customized format. In the tracing system, there is a requirement to use external sources, such as directory assistance or the white pages, when all currently know sources have been exhausted. The tracer would record the numbers obtained from the external source and begin calling the numbers. We will use this requirement to show the relationship of Basil and Manipula.
Selecting the External Source button on the Select a Participant screen launches a Manipula dialogue , Figure 6, from which a tracer can select an external source.

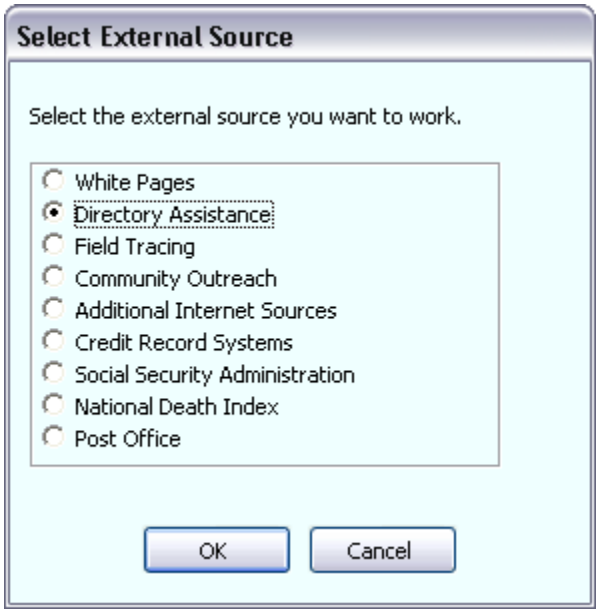

Figure 2. Select External Source

It was decided that the recording of the telephone numbers would be in a separate Basil instrument. The External Source Work Sheet was formatted to look like a popup and contained twelve rows for up to twelve telephone numbers. The primary key was consistent with the P-TIS and the Tracing master database, but also included a secondary key to identify the external source from which the telephone numbers were received.

After the selection of a source, a simple EDIT function call is used to launch the External Source Work Sheet instrument, Figure 7. This code follows:

```
auxResult := EDIT( 'TracingTWS /FTracingTWS /K' + auxTWSKey + ' /G /N ' +
                   '/Q=ParticipantName=' + '"' + auxParticipant + '"' +
                   ';tName=' + '"' + auxTName + '"',BASIL)
```

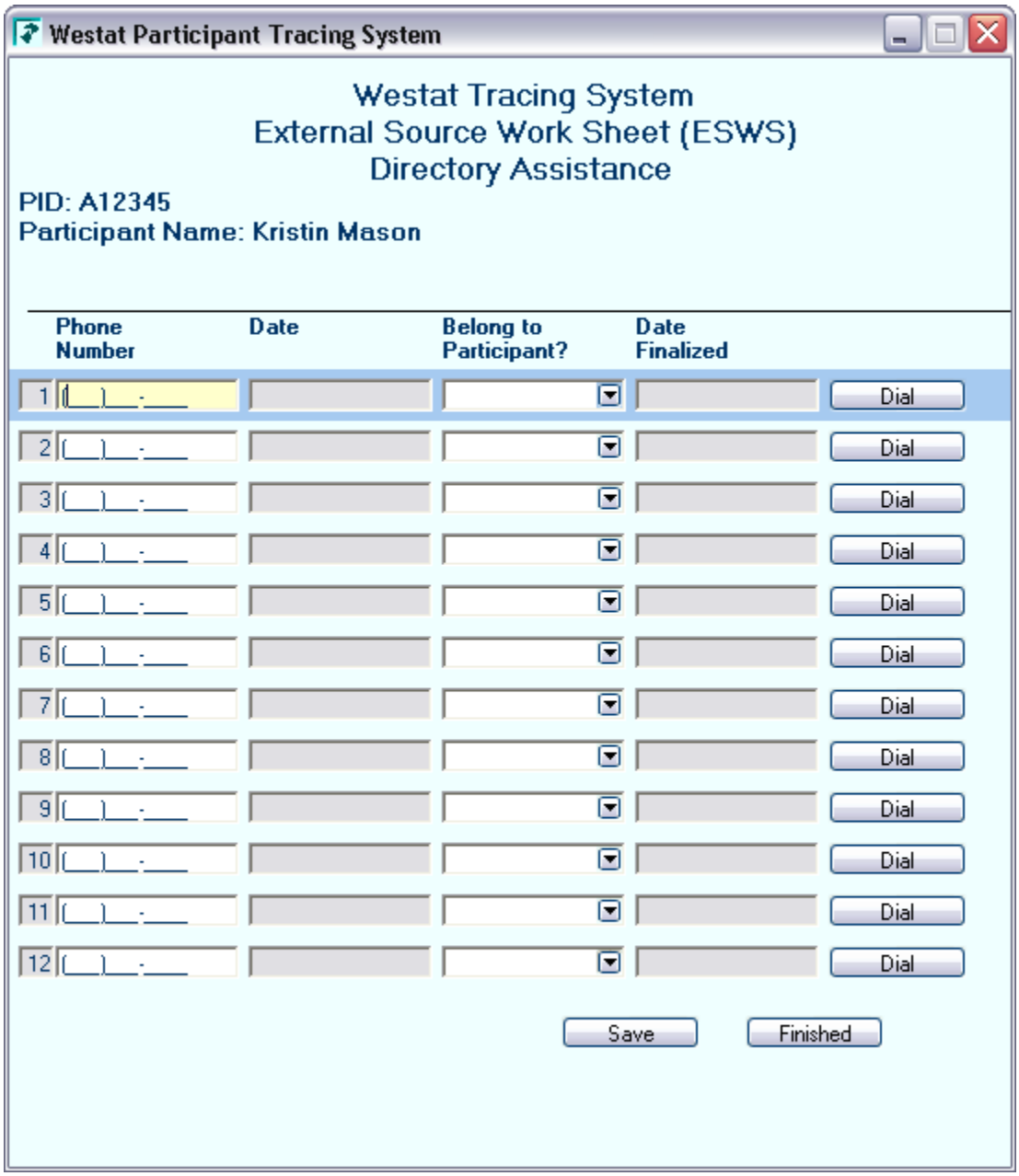

Figure 3. External Source Work Sheet

After collecting telephone numbers from the external source, the tracers begin calling each number. They will enter the telephone number and then select the Dial button for each number. The Maniplus procedure, procDialPhone, is called from which a dialogue will display the number to call, predefined text and a call

result. See Figure 8. As can be seen from the following code, the DialNumber uses a button control to call the procedure procDialPhone in the TracingProcedures Manipula setup:

```
DialNumber
    BASIL"<question>
            <button $auxDialButton
              caption='%TWS.ContactSummary[^pSubScript].DialNumber'
              onclick='TracingProcedures.procDialPhone
                       (""%TWS.ContactSummary[^pSubScript].Telephone"","%TName"")'>
          </question>"
      : typPhoneStatus
```

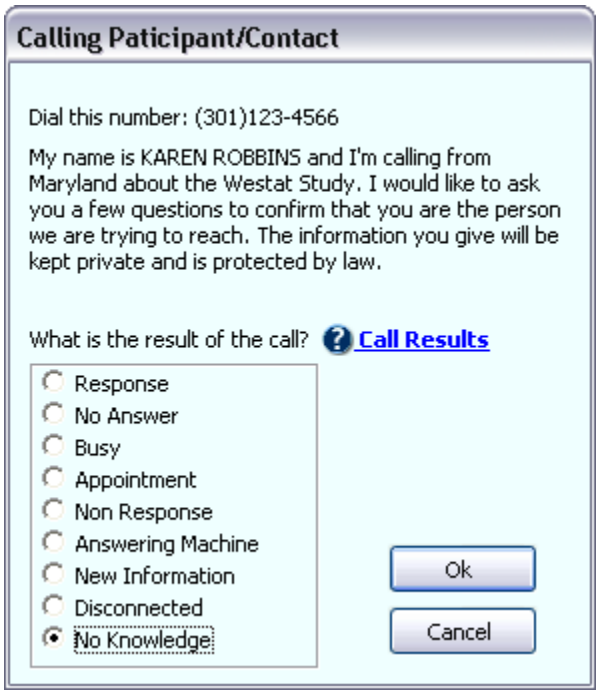

Figure 4. Dial Screen

After the call has been made and a result selected, the procedure will then use Manipula extensions to update the work sheet with the call result and the date. The Dial button will be changed to reflect the call result and the date field will now contain the current date. See figure 9.

Figure 5. Updated Work Sheet

The procDialPhone procedure uses two simple Manipula extensions, ACTIVEFIELD and PUTVALUE as the following code snippet shows.

```
aFieldName := ACTIVEFIELD
ufGeneric.PUTVALUE(aFieldName,STR(typPhoneStatus))
ufGeneric.PUTVALUE(aFieldNameDate,DATETOSTR(SYSDATE,DDMMYY))
```

Using the INTERCHANGE=SHARED Manipula setting allows for this communication (the transfer of data) between the Manipula setup and the Basil instrument.

Further examination of the Button control of the Basil shows the prefix $ in front of an AUXFIELD used for formatting and a % in front of the fully qualified field.

```
<button $auxDialButton
     caption='%TWS.ContactSummary[^pSubScript].DialNumber'
```

The Caption attribute of the DialNumber button is updated when the value of the DialNumber field from the procDialPhone is updated. The % in front of the field name ensures the caption attribute is updated with the current value of the field.

The standard Blaise fills (field names preceded by the ^ symbol) can be used but are evaluated during the execution of the rules. In Basil there are two alternative fill symbols available: the $-fill and the %-fill. They are both evaluated when Basil decides that they are needed and are both replaced by the value of the indicated field at the moment of evaluating.

4 Relational Database Storage

4.1 Blaise Datalink

The Blaise Datalink facility allows the data used by a Blaise data model to actually be stored in a non-Blaise format, such as Microsoft SQL Server, Oracle, or other relational databases. Datalink implements this functionality using Microsoft's OLE DB (Object Linking and Embedding, Database) technology. This means that it is possible to implement Datalink for any data source for which an OLE DB driver

exists. As a practical matter, the type of Datalink system that we required is much easier to implement for a subset of the most common relational data sources, for which Blaise takes into account the characteristics of the particular data source, such as the maximum number of columns it can have in a single table. Blaise does provide such support for SQL Server, which is the database we are using for our tracing system.

4.2 Tracing History and Versioning

Management is always looking at how a tracer is progressing in the search for a missing participant. And a big part of this is looking at what has been done to date. In the past Blaise instruments were written to handle historical data as well as current data. This often led to unwieldy instruments. Using the Datalink Versioning feature removes the need for an instrument to keep track of historical information and uses the database to keep track.

We chose to use the Generic feature of Datalink with flat blocks and versioning as our MS SQL database configuration. Flat blocks mean each block in our Basil tracing instruments will correspond to a table in the SQL database. Generic allows for the sharing of tables across instruments. Instrument data is stored in a centralized way in only a few predefined and fixed table structures. Versioning is used to store multiple versions of a Basil record in the database. The versions are either current or historical. Each time a changed record is written to the database, the previous record is marked as 'historical' and the new record is inserted in the database and marked as 'current'.

The tracing system had a requirement to view all participant and contact address changes. When using Manipula to extract data or the Blaise Database viewer to inspect the data, only the current data is extracted or displayed. To get the historical records you need to use other tools like SQL report writers or the Blaise Data Center. But each by itself is impractical in that they are not easily incorporated into the tracing system.

An alternative is to use the OLE DB Tool Box to generate a Blaise datamodel based on a SQL view. In this case a SQL view was created showing all the address changes for each participant or contact. Using the OLE DB Tool Box and the SQL view as input, a Blaise datamodel and a corresponding Boi file were generated. A Manipula lookup dialogue was then programmed to display the address history for each participant or contact. This technique would be used to view other record changes such as name and telephone numbers.

5. Conclusions

5.1 Lessons Learned

In the process of taking forms designed for a manual paper process and converting the process into computer data entry system using Basil and SQL databases we learned the following:

- When designing the input screens we learned the importance of creating Auxfields for the common formatting of Basil controls. For example, if you have a column of similar fields you probably want them all to be vertically aligned. You would create an Auxfield with the common attributes, such as, left, top, width etc. So in the Rules you would have:

  ```
  auxColumnAlign := 'left=10 width=100, top=15'
  ```

  where auxColumnAlign would then be used by the control for each field in the column. Thus, any changes are made in one place instead of in each control.

- The concept of a tracing system is different than the collection of survey data. Survey data are essentially collected at one time from one source, whereas, the tracing system must allow for the collection of disparate data at different times. Storing data in different Basil instruments allows modularizing code for easy maintenance and testing.
- During the development phase use Blaise databases instead of a SQL database. Until the Basil metadata is finalized there is no reason to do the extra work that would be involved in the frequent regeneration of the Datalink files and SQL tables.
- At the same time you are designing the instrument you should be building in lockstep the database load utilities. As you add/delete questions to the Basil instrument the load utilities should be modified accordingly. Thus, in the case of the tracing system, when development is finished the data sources for each participant are identified, the load utilities are up to date, and are ready to go at the click of a button.

5.2 Summary

Our impressions of using Basil to build the Westat Participant Tracing System prototype are overall, very good. Formatting the Basil instruments takes a little getting used to, but the ability to do your own formatting can result in a rich and cleaner looking user interface. Basil also features free form data entry and the ability to freely move from one module to another.