Using Basil for ACASI at Westat– From custom to COTS

G J Boris Allan, Peter Stegehuis, and Rick Dulaney
Westat,
Rockville, Maryland, USA

Audio Computer-Assisted Self Interviewing (ACASI) is an important data collection mode for many surveys, particularly when collecting very sensitive data. As the use of Basil for ACASI is fairly new, this paper focuses on the possibilities and capabilities of the product to be customized in various ways. We have been able to use Basil in CASI mode, ACASI mode, language-aware ACASI, and with text-to-speech (TTS). There are limitations inherent in the use of human voice recordings to read questions during an ACASI interview ("voice talent"), so we have investigated the use of TTS software and show the possibilities with TTS integration into Basil applications. We conclude some thoughts on lessons learned and next steps.

1. Basic Basil – CASI

Basil can be fully implemented without any additions as a commercial-off-the-shelf (COTS) software and apart from the specific graphics files would run on any computer with Blaise. Once prepared, all that is needed is `Manipula.exe` and command-line parameters. The program is CASI, not ACASI, because we have no audio (no sound files).

In programming Basil for CASI, two characteristics of Basil are of utmost importance as compared to Blaise – how information is displayed and how the RULES are executed:

- *Basil is page-based*, that is, screens are presented one page at a time, where the design of what appears on a page is dependent on
  - a template (an application definition) and
  - the questions that appear on that screen (the field definitions).
- *Basil is event-driven*, that is, though moving between items on a page/screen is by user choice, to move from one page/screen to the next or previous requires selection of a specific item (usually a button).

For example, a basic Basil screen looks like this:

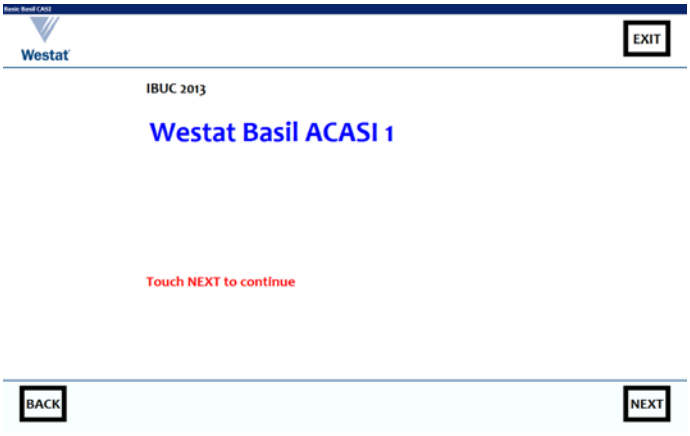

Figure 1-1: WBA 1, title screen

By following the "Touch NEXT to continue", another screen appears (the NEXT page):

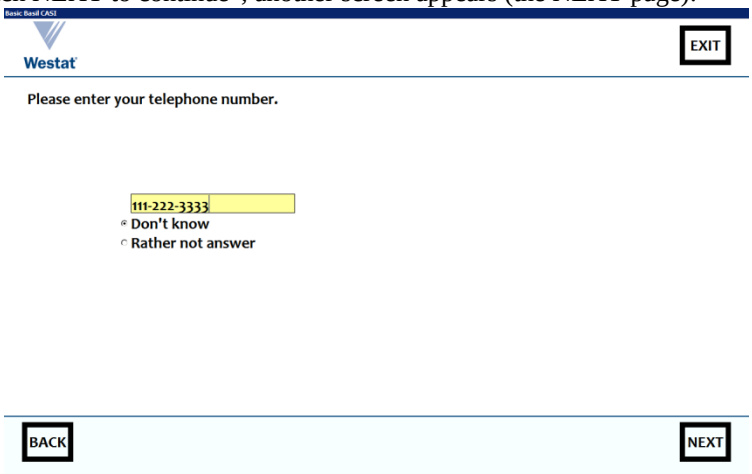

Figure 1-2: WBA 1, data-entry screen (text entry)

If you look at the two screens, there are certain common display features (from the logo to the NEXT button). These common features are defined in a Basil application section.
Here is the application section used to define the common features for the above displays (underlined text is for emphasis):

```
DATAMODEL BasicBasil

  "
  <application name='app' title='Basic Basil CASI' canclose=false
windowstate=maximized width=1280 minimumwidth=1280 clientheight=800
minimumheight=800 resource='' icon='westat.ico' sizeable=false
fontface='Candara' fontbold=true fontsize=20 fontcolor=black
singleinstance=true>
   <panel name='overall' align=client color=white>
     <panel name='global' height=100 align=top>
      <rectangle left=1160 top=10 height=70 width=90 color=#000000>
      <img src='Westat_Standard_Vert.gif' left=34 top=4 width=92 height=81
stretch=true onclick='http://www.Westat.com' hint='Westat website'>
      <speedbutton name='finish' left=1170 top=20 height=50 width=70
color=#000000 caption='EXIT' onclick='blaise:save();blaise:quit()'
hint='Finish interview'>
      <line left=0 top=98 width=1280 height=2 color=#7E4500>
     </panel>
     <content-area horizontalscrollbar=false verticalscrollbar=false >
     <panel name='navigation' height=100 align=bottom color=#FFFFF0>
      <line left=0 top=0 width=1280 height=2 color=#7E4500>
      <rectangle left=30 top=10 height=70 width=90 color=#000000>
      <rectangle left=1160 top=10 height=70 width=90 color=#000000>
      <speedbutton name='back' left=40 top=20 height=50 width=70
color=#000000 caption='BACK' onclick='blaise:previouspage()' hint='Previous
page'>
      <speedbutton name='forward' left=1170 top=20 height=50 width=70
color=#000000 caption='NEXT' onclick='blaise:save();blaise:nextpage();'
hint='Next page'>
     </panel>
   </panel>
  </application>
  "
```

```
 LANGUAGES = BASIL "Basil"
```

Note that we have declared only one language, the *machine* language `BASIL` – we have not declared *spoken* languages such as English or Spanish.

Within the application section we define certain standard features such as font characteristics and colors. The screen is divided into three "panels". One panel (`overall`) contains two other sub-panels (`global`, `navigation`) and a `content-area` that is used to display questions.

The `global` (top) sub-panel has two items:

- the Westat logo gif, and a link to the Westat web site
- an EXIT button that uses standard Basil commands to save the data, and exit the instrument

The `navigation` (bottom) sub-panel has two items, `BACK` and `NEXT` buttons that use standard Basil commands to save the data, and change the page (using `BACK` does not save the data, but we could modify `onclick` to add a `blaise:save()`).

Screen locations and other design aspects are given within sub-panels.

In the "content-area" we turn off scroll bars, and this screen area is where we show instrument fields. Basil provides for considerable flexibility in the development of screens. For example, the first screen (the description "WBA 1") is defined by:

```
_StartField
  BASIL "<question>
    <label fontsize=20 halign=center width=1000 topmargin=20 text='IBUC 2013
<br><br> <font color=blue size=40>Westat Basil ACASI 1
</font><br><br><br><br><br><br><br><br> <font color=red size=20>Touch NEXT to
continue</font>'>
  </question>"
  : STRING[1], EMPTY
```

The `_StartField` field definition uses HTML-style formatting commands (from `<question>` to `</question>`). The `label` control has a `text` attribute which uses HTML-style formatting:

```
'IBUC 2013 <br><br> <font color=blue size=40>Westat Basil ACASI 1</font>
<br><br><br><br><br><br><br><br> <font color=red size=20>Touch NEXT to
continue</font>'>.
```

This is the formatted text that appears in the first screen (Figure 1-1).

The second screen (which asks for a telephone number) shows an example with an extra control (`input`) to indicate that a data value is expected (most fields will have an `input` control). The `input` control has various Basil-specific formatting commands (including an edit mask):

```
Quest41
  BASIL "<question>
    <label left=40 width=1200 topmargin=20 text='Please enter your telephone
number.'> <input visible=true  halign=center left=200 top=200 width=300
height=100 editmask='000\-000\-0000' fontcolor=black focusedcolor=#99FFFF
showinputlineradiobutton=false fontcolor=black radiobuttonsize=40
showdontknow=true showrefusal=true showfocusrect=false>
  </question>
  "
  : STRING[10], DK, RF
```

Additional example screens show fills and rules.

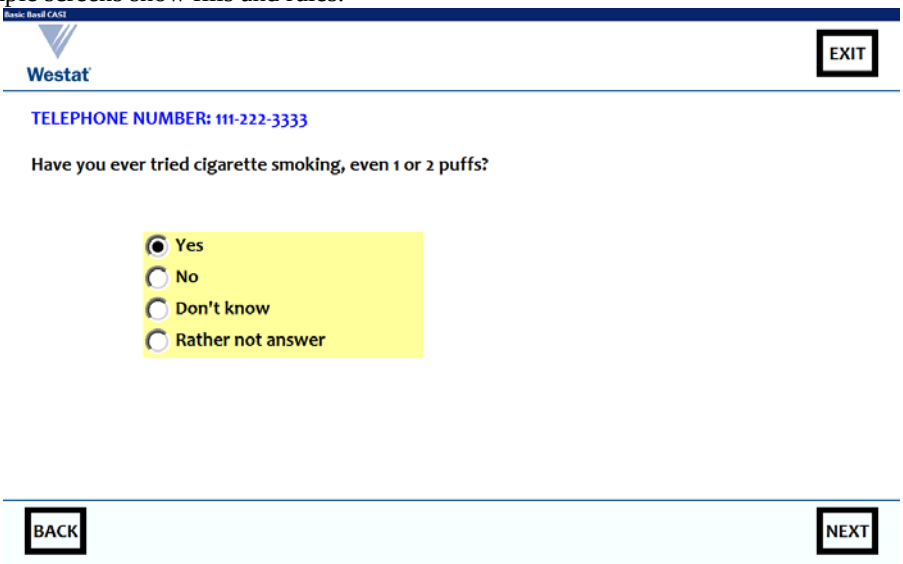

Figure 1-3: WBA 1, data-entry screen (enumeration)

```
Quest1
  BASIL "<question>
    <label height=600 width=1200 top=20 left=40 text='<font
color=blue>$Telephone</font>Have you ever tried cigarette smoking, even 1 or
2 puffs?'> <input visible=true left=200 top=200 width=400 fontcolor=black
radiobuttonsize=40 focusedcolor=#99FFFF showdontknow=true showrefusal=true
showfocusrect=false>
  </question>"
  : (Yes, No), RF, DK
```

The response categories are highlighted (`#99FFFF`) and we will look at the focus rectangle in the context of the next screen.

In the `text` attribute for the `label`, `$Telephone` is a fill (just as `^Telephone` would be a fill in ordinary Blaise). Basil has two types of fill (`$` and `%`) where the difference between them is not always clear in practice – the use of standard Blaise (`^`) fills is restricted to fills in type definitions.

The value of the `Telephone` field is given in the `RULES`, and depends on the value entered for the telephone number:

```
Quest41
IF (Quest41 = RESPONSE) THEN
  Telephone := 'TELEPHONE NUMBER: ' + SUBString(Quest41,1,3) + '-'
     + SUBString(Quest41,4,3) + '-' + SUBString(Quest41,7,4) + '<br><br>'
ELSEIF (Quest41 = REFUSAL) THEN
  Telephone := 'TELEPHONE NUMBER: Rather not say<br><br>'
ELSEIF (Quest41 = DONTKNOW) THEN
  Telephone := 'TELEPHONE NUMBER: Do not know<br><br>'
ELSE
  Telephone := ''
ENDIF
```

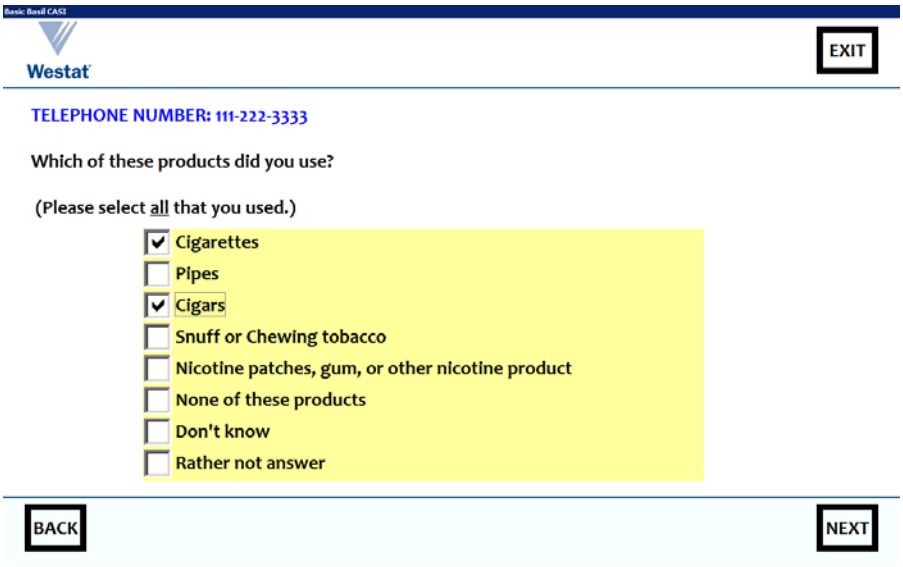

Figure 1-4: WBA 1, data entry screen (set)

```
Quest60
  BASIL "<question>
    <label left=40 width=1200 topmargin=20 text='<font
color=blue>$Telephone</font> Which of these products did you
use?<br><br>(Please select <u>all</u> that you used.)'>
    <input visible=true left=200 top=200 width=800 checkboxsize=40
focusedcolor=#99FFFF showdontknow=true showrefusal=true>
  </question>
  "
  : SET OF TCigTypes, DK, RF
```

In `Quest60` (Figure 1-4) there is a focus rectangle around `Cigars` for illustrative reasons. There is no rectangle around `Yes` for `Quest1` (Figure 1-3) because we have switched off the focus rectangle for that question (`showfocusrect=false`). The final screen:

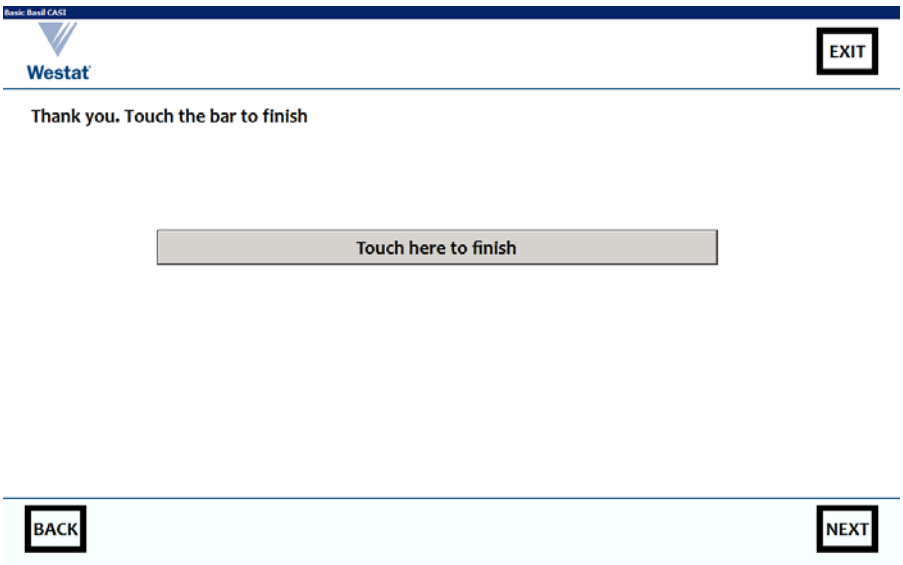

Figure 1-5: WBA 1, confirmation screen (button)

The interview ends when the button is touched (`onclick`).

```
EndInstrument
  BASIL "<question>
    <label left=40 width=1200 topmargin=20 text='Thank you. Touch the bar to
finish'> <input type=button left=220 top=200 height=50 width=800
caption='Touch here to finish' onclick='blaise:save();blaise:quit()'>
  </question>"
  : (Yes), EMPTY
```

2. **Basil CASI to ACASI with Maniplus added**

The next enhancement to Basil is completed by adding sound to each input field (using WAV files) and activating the error-checking features of Basil. At this point Basil CASI becomes ACASI.

We make a small change to the beginning of the application definition:

```
"<application name='app' title='Basil ACASI -- Audio files' canclose=false
windowstate=maximized width=1280 minimumwidth=1280 clientheight=800
minimumheight=800 resource='' icon='westat.ico' sizeable=false
fontface='Candara' fontbold=true fontsize=20 fontcolor=black setups='WBA.msu
/Kmeta=WBA02;' singleinstance=true>
…
```

The only real difference is the use of a metadata-aware Maniplus executable named `WBA.msu` (**W**estat **B**asil **A**CASI). This is the generic WBA Maniplus driver program, and individual projects can add additional programs for special purposes. The program is declared by `setups='WBA.msu /Kmeta=WBA02;'`, that is, there is a program `WBA.msu` that has a variable metadata definition, and in this case the metadata is that provided by `WBA02` (`WBA02.bmi`).

Another change is to the operation of the `NEXT` button, which actually uses a procedure from the Maniplus program declared in the `setups`:

```
<speedbutton name='forward' left=1170 top=20 height=50 width=70 color=#000000
caption='NEXT' onclick='WBA.ErrorCheck;blaise:save();blaise:nextpage();'
hint='Next page'>
```

The `onclick` attribute has an extra action, that is, the `ErrorCheck` procedure from `WBA.msu`.

The Maniplus program is aware of the current data and the current metadata of the instrument in memory by a file setting `INTERCHANGE = SHARED`.

The program performs the following actions:

- Check the rules (`CHECKRULES`) for the record in memory (a result of the setting `INTERCHANGE = SHARED`)
- Save the name of the currently active field for the record in memory to `theActive`
- Set the scope of error checking to include only those fields on the currently active page
- For the currently active page, get the number of Blaise/Basil errors and – if the count is greater than zero – for each error, get the type (`KIND`) of error

Much of the power of Basil in ACASI comes from the use of metadata-aware Maniplus (metadata such as `tempFile.GETERRORCOUNT`) even though we use only one procedure in this example.

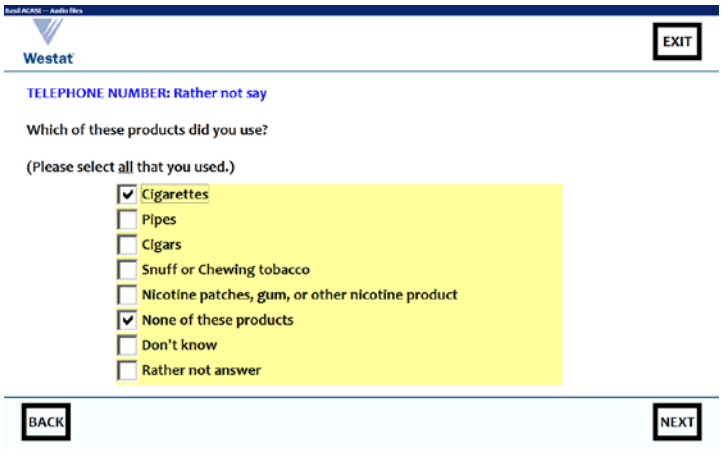

Figure 2-1: WBA 2, data-entry error (incompatible selections) – the question text and responses categories are spoken

One should not be able to answer "Cigarettes" and "None of these products", and – when `NEXT` is touched – a defined error checking routine goes into operation:

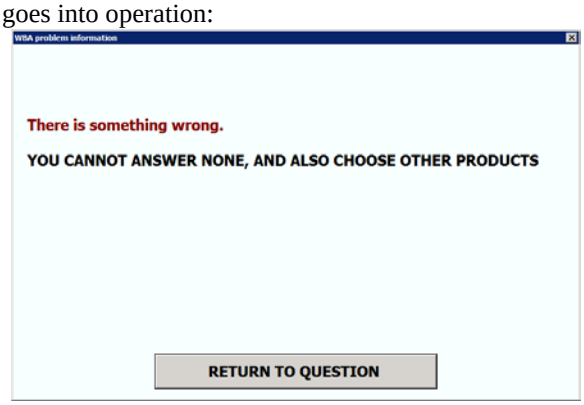

Figure 2-2: WBA 2, error dialog box (incompatible selections) – the error text is not spoken

This is a Maniplus dialog box, and the displayed text is taken from a `CHECK` in the Basil/Blaise rules:

```
IF (None IN Quest60) AND (Quest60.CARDINAL > 1) THEN
  CHECK ERROR
    BASIL "YOU CANNOT ANSWER NONE, AND ALSO CHOOSE OTHER PRODUCTS"
ENDIF
```

Note that our Maniplus procedures prepend "There is something wrong" to the defined error text in the displayed text. If none of the answers are chosen (a common user action), we get a different dialog box, that is, the dialog box for a `ROUTE` error:

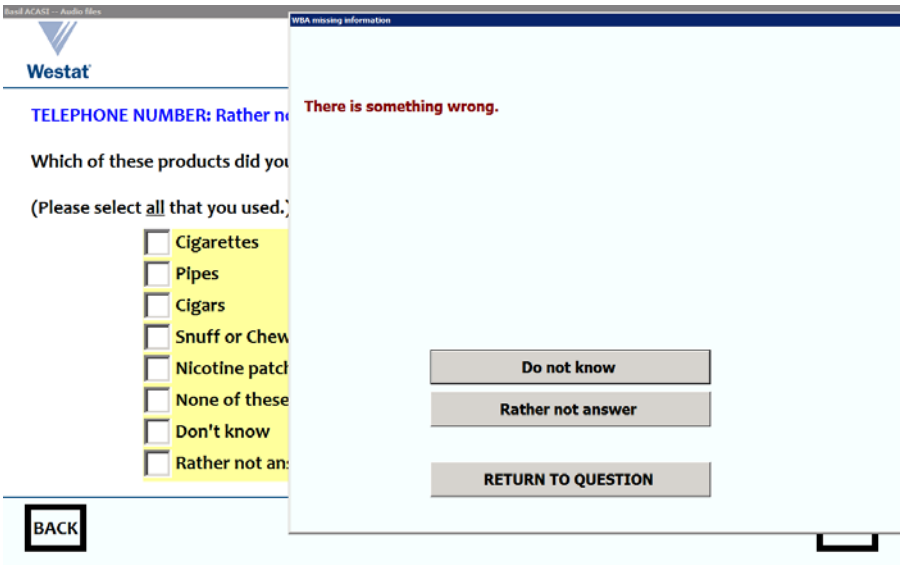

Figure 2-3: WBA 2, error dialog box (missing data) – the error text is not spoken

It is clear that we can change the text displayed, depending on the type of error (`ROUTE SOFT HARD`), and `ErrorCheck` has different dialog boxes for each type (we based our procedure on a sample provided with the Basil distribution).

We have also declared audio output as `Quest60Audio`, composed of many sound files in a folder `Media\English`. We have the main question file (`Quest60.wav`), files for each option, and files for `DK` and `RF`, plus a silence file. The sound files are played in a continual loop (including silence), and the audio loop is stopped when a key is pressed. We start the audio by attaching it to the `<input … onenter= … >` control/attribute.

3. Basil ACASI becomes language-aware

So far we still have only one language (`BASIL`) which we conveniently express in English. Many of our studies are multi-language, and so we need to be able to switch spoken languages. The Basil programming language always operates with one machine language (`BASIL`) with the question text being displayed by the `<label … text= …>` control/attribute in whatever spoken language is provided by the fill. At Westat, we have resolved this problem of changing spoken languages by using procedures defined in `WBA.msu` that fill the `<label … text= …>` control/attributes, and select folders, depending on the currently chosen spoken language.

A language option is added to the screen as shown below.

Figure 3-1: WBA 3, title screen (with language option)

After the user selects the language, the screen text is changed.

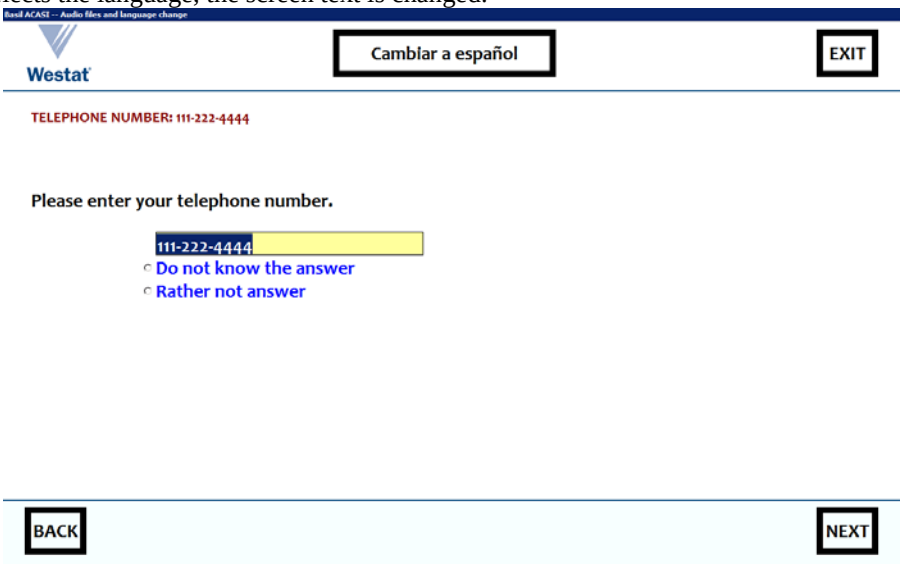

Figure 3-2: WBA 1, data-entry screen with English display (text entry)

Touch the button that changes to Spanish:

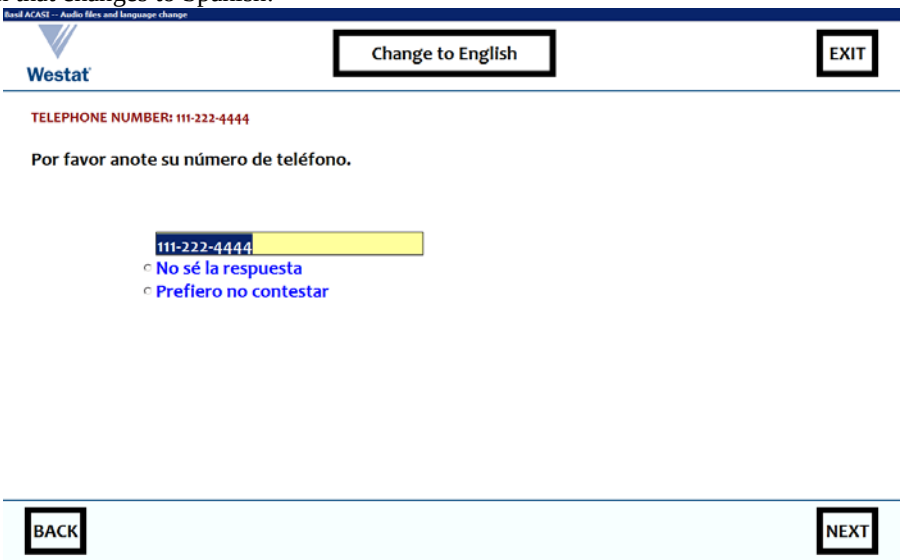

Figure 3-3 WBA 4, data-entry screen with Spanish display (text entry) – the question text is spoken in Spanish

Our question is now displayed in Spanish.

The field (`Quest60`) that asks about types of tobacco product has an attached type:

```
Type
  TCigTypes =
    ( Cigarettes (1)  BASIL "^TCigTypes_Cigarettes" ENG "Cigarettes" ESP
"Cigarillos",
      Pipes (2)  BASIL "^TCigTypes_Pipes" ENG "Pipes" ESP "Pipas ",
      Cigars (3)  BASIL "^TCigTypes_Cigars" ENG "Cigars" ESP "Cigarros",
      Snuff (4)  BASIL "^TCigTypes_Snuff" ENG "Snuff or Chewing tobacco"
          ESP "Rapé o tabaco",
      Other (5)  BASIL "^TCigTypes_Other"
          ENG "Nicotine patches, gum, or other nicotine product"
          ESP "Parches de nicotina, chicle, u otro producto de nicotina",
      None (9)  BASIL "^TCigTypes_None" ENG "None of these products"
          ESP "Ninguno de estos productos"
    )
  ENG "Which of these products did you use?<br><br>(Please select <u>all</u>
that you used.)"
  ESP "¿Cuál de los siguientes productos usó usted?<br><br>Por favor elija
<u>todos</u> los que usó."
  : SET OF TCigTypes, DK, RF
```

We were able to set it up so that , an error message for this field, in English and in Spanish, now shows the applicable question text as well as the error message:

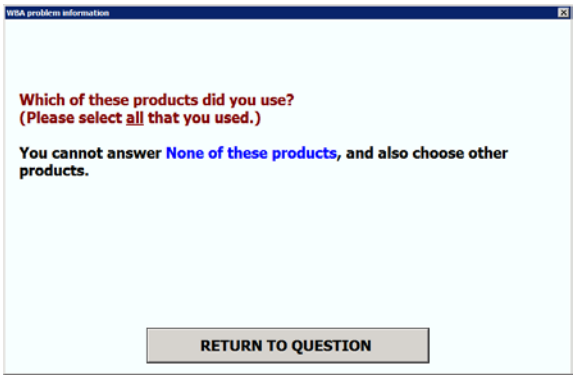

Figure 3-4: WBA 3, error dialog box with question text fill, in English (incompatible selections) – the error text is not spoken

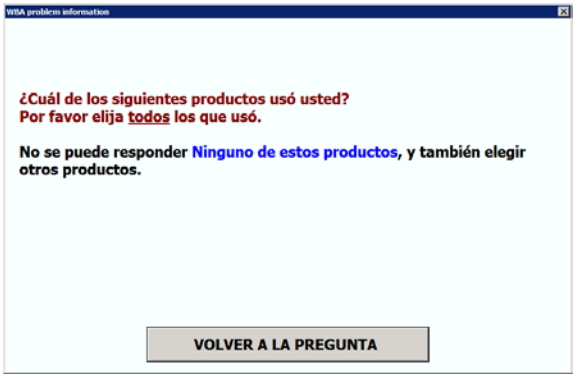

Figure 3-5: WBA 3, error dialog box with question text fill, in Spanish (incompatible selections) – the error text is not spoken

The `CHECK` is:

```
IF (None IN Quest60) AND (Quest60.CARDINAL > 1) THEN
  CHECK ERROR
    ENG "You cannot answer <font color=blue>None of these products</font>,
and also choose other products."
    ESP "No se puede responder <font color=blue>Ninguno de estos
productos</font>, y también elegir otros productos."
ENDIF
```

If we do not make any entries for this question, we get dialog boxes in English and in Spanish:

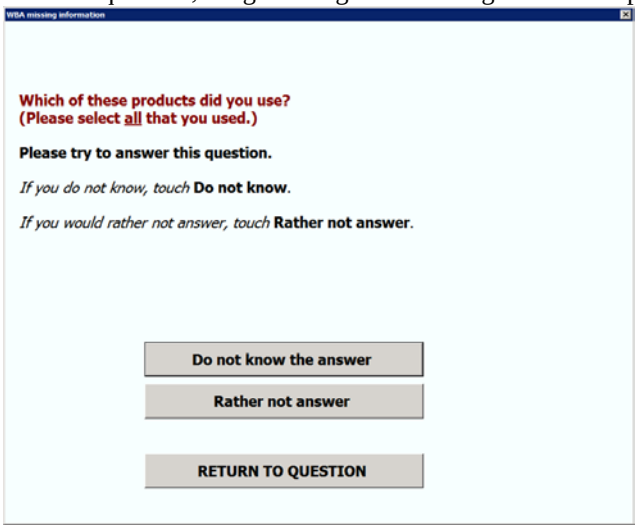

Figure 3-6: WBA 3, error dialog box with question text fill, in English (missing data) – the error text is not spoken

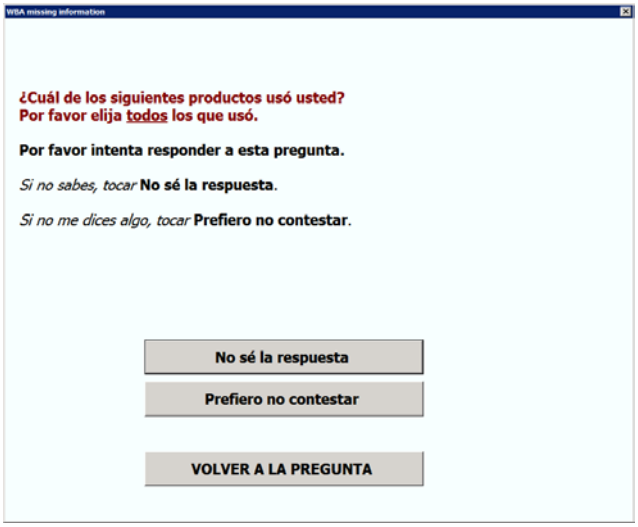

Figure 3-7: WBA 3, error dialog box with question text fill, in Spanish (missing data) – the error text is not spoken

There is no sound attached to these dialog boxes.

4. Adding text-to-speech to language-aware Basil

As a next step, we were able to add text-to-speech (TTS) to Basil. An example screen is shown below.

Figure 4-1: WBA 4, title screen (with language option) – "Touch NEXT to continue" is spoken

And you hear "Touch next to continue". The definition of this field shows a new procedure, `WBA.Speak`, being called.
Because we have to be able to stop and start speech on demand, the application section refers to many TTS-related procedures:

```
DATAMODEL BasilACASITextToSpeech

  "<application name='app' title='Basil ACASI -- Text to Speech'
canclose=false windowstate=maximized width=1280 minimumwidth=1280
clientheight=800 minimumheight=800 resource='' icon='westat.ico'
sizeable=false oncreate='WBA.ChangeLang(''ENG'');' fontface='Candara'
fontbold=true fontsize=20 fontcolor=black setups='WBA.msu /Kmeta=WBA04;'
singleinstance=true>
    <panel name='overall' align=client  color=white>
      <panel name='language' height=100 align=top>
        <speedbutton left=480 top=20 height=50 width=300
caption='$__Language_Fill'
            onclick='WBA.StopSpeaking("""); WBA.GotoStart();
            WBA.ChangeLang(''OTHER''); WBA.GotoOldActive();'
            hint='$__Language_Fill'>
        <rectangle left=470 top=10 height=70 width=320 color=#000000>
        <img src='Westat_Standard_Vert.gif' left=34 top=4 width=92 height=81
            stretch=true
onclick='WBA.StopSpeaking(""");http://www.Westat.com'
            hint='Westat website'>
        <speedbutton name='finish' left=1170 top=20 height=50 width=70
color=#000000
            caption='EXIT' onclick='WBA.StopSpeaking("""); blaise:save();
            blaise:quit()' hint='Finish interview'>
        <rectangle left=1160 top=10 height=70 width=90 color=#000000>
        <line left=0 top=98 width=1280 height=2 color=#7E4500>
      </panel>
      <<content-area horizontalscrollbar=false verticalscrollbar=false>>
```

```
      <panel name='navigation' height=100 align=bottom color=#FFFFF0>
        <line left=0 top=0 width=1280 height=2 color=#7E4500>
        <rectangle left=30 top=10 height=70 width=90 color=#000000>
        <rectangle left=1160 top=10 height=70 width=90 color=#000000>
        <speedbutton name='back' left=40 top=20 height=50 width=70
color=#000000
            caption='BACK' onclick='WBA.StopSpeaking("""");
blaise:previouspage()'
            hint='Previous page'>
        <speedbutton name='forward' left=1170 top=20 height=50 width=70
color=#000000
            caption='NEXT' onclick='WBA.StopSpeaking("""); WBA.ErrorCheck;
            blaise:save();blaise:nextpage();' hint='Next page'>
      </panel>
    </panel>
  </application>
  "
```

You will notice that we make great use of `WBA.StopSpeaking("""")`, otherwise we are liable to continue speaking old text when we move on to new text, because you have asynchronous speech when using alien Visual BASIC procedures.

Basil can handle complex field definitions and rules execution paths. We found that Basil gives an impressive inventory of features for customizing instruments.

5. The potential of Basil for ACASI

ACASI provides a viable new methodology for self-interviewing, and – especially with text-to-speech – great flexibility in what users hear. Blaise Basil has shown to be adaptable as an ACASI platform and to enable Blaise users to develop sophisticated custom applications to meet study needs. Some observations from the process include:

- Apart from the general look and feel defined in the application section, the only instructions for displaying a field in a Basil instrument are those given as `BASIL "<question> … </question>"` where the actual text displayed is given in the `text=` attribute. Other interview languages (say, `ENG ESP`) do not affect Basil.
- Though other interview languages are not recognized by Basil, we can use metadata-aware Maniplus to look at those other languages, and that is the way we can change the content of a `BASIL text=` attribute.
- Because Basil has most of the flexibility of Blaise, with some useful extra features, appearance is very customizable and we can react to the desires of clients and methodologists.
- If we use Basil for ACASI, then there are cost savings for text-to-speech as opposed to using pre-recorded sound files (sometimes known as “voice talent”) – there are valuable time savings in the development cycle by not having to record all questions, answers and fills, especially when using multiple languages. It is also simpler to use TTS with dialog boxes.
- With Basil/Maniplus we control layout and can respond to events within one control environment and do not need to use other features such as the alien router or special DLLs to complete these operations.
- With TTS technology improving at a rapid pace its use for ACASI will become ever more sensible and cost-effective. While there are still some remaining challenges with regard to attaining the full flexibility we would like to achieve with integration of TTS within Basil ACASI, we believe that Basil/Maniplus already allows for a very satisfactory level of control over TTS events .

[1] We would like to thank Joseph Allen and Bill Copeland of Westat for their great help in the preparation of Westat Basil ACASI. The Blaise team were always exceedingly helpful and prompt in their assistance – thanks to Lon Hofman and Roger Linssen in particular.