

# **Proceedings of the 8th International Blaise Users Conference**

## **IBUC 2003**

**Copenhagen May 21st - May 23rd, 2003**



**Proceedings of  
the 8th International  
Blaise Users Conference  
IBUC 2003**

Printed by  
Statistics Denmark

*Address:*

Danmarks Statistik  
Sejrøgade 11  
DK-2100 København Ø

e-mail [dst@dst.dk](mailto:dst@dst.dk)  
www [dst.dk](http://dst.dk)

## Preface

This book contains the papers presented at the 8th International Blaise Users Conference held in Copenhagen, Denmark, on May 21st - 23rd, 2003. The conference included three days of technical papers and presentations on the use of Blaise and related topics.

The conference programme was planned and organized by the Scientific Committee of the International Blaise Users Group. The members of the committee included:

- Vesa Kuusela, Statistics Finland (chair)
- Leif Bochis, Statistics Denmark
- Bill Connett, University of Michigan
- Sal Gara, Statistics Canada
- Lon Hofman, Statistics Netherlands
- Jim O'Reilly, Westat, USA
- Mark Pierzchala, Mathematica Policy Research
- Tony Manners, Office of National Statistics, UK
- Michael Haas, U.S. Bureau of the Census

Statistics Denmark, as the host organization of the conference, has collected, edited and printed the proceedings.

# Contents

## *Metadata and Automatic Generation of Blaise Datamodels*

Blaise Survey Generator .....	7
Blaise Questionnaire Text Editor (Qtxt) .....	17
Automatic Upgrade of Production Blaise Instruments .....	27

## *Standards for design of user interfaces*

Screen Layout Standards at Statistics Finland .....	39
Developing and updating screen layout and design standards .....	45
Developing an optimal screen layout for CAI .....	63
Screen Design Guidelines for Blaise Instruments .....	77

## *Design of Web Questionnaires*

Methodological guidelines for Blaise web surveys .....	91
--	----

## *Applications of Blaise*

The usage of Blaise in an integrated application for the Births Sample Survey in CATI mode.....	125
Blaise at Statistics Netherlands.....	137
Displaying Chinese Characters In Blaise.....	149

## *Data Quality and Testing*

Analyzing Audit Trails in the National Survey on Drug Use and Health (NSDUH) .	155
A Random Walk Application for Blaise Instruments.....	175
First steps along the Audit Trail.....	183

## *Post-processing and tabulation*

Creating Code to Convert Blaise to ASCII for SAS Input (Using the Blaise API within Visual Basic) .....	191
Extracting data from a large instrument using the API.....	199

*Integration of multi mode surveys, Survey Management, Case Management*

The “Multiple Application Interface” with Blaise and Visual Basic.....	205
Electronic Data Reporter .....	215
Integration of CAPI and CATI infrastructures at Statistics Canada .....	225

*CATI and Organizational issues*

Blaise CATI at Mathematica .....	233
Cati Survey Management System .....	241
Interactive training for the interviewers.....	251

*Paper Forms and Blaise*

Blaise for Post-collection Data Editing .....	255
The melting pot - practical experience of scanning paper forms.....	263
Lessons learned on the development of the Unified Enterprise Survey.....	265

*Posters*

The American Time Use Survey Data Collection Instrument .....	269
Using non-Latin alphabets in Blaise.....	281
Automated Dialing - What will it do for you? .....	291

*Appendix*

Developing CASI standards at Statistics Netherlands.....	295
Contacts .....	306



## ***Metadata and Automatic Generation of Blaise Datamodels***

- **Blaise Survey Generator** .....7  
*Carlo J.C. Vreugde, Jacques J.M.J. de Groot,  
Christiaan van 't Hoft, VNG, SGB0, StimulansZ, of the  
Netherlands*
- **Blaise Questionnaire Text Editor (Qtxt)**.....17  
*Grayson Mitchell, Statistics New Zealand.*
- **Automatic Upgrade of Production Blaise Instruments**...27  
*Lilia Filippenko Joseph Nofziger & Gilbert Rodriguez,  
RTI International*



# Blaise Survey Generator

*Any researcher can make a survey 'without writing Blaise code'.*

*Carlo J.C. Vreugde, Jacques J.M.J. de Groot, Christiaan van 't Hof, VNG, SGBO, StimulansZ, of the Netherlands*

## 1. Introduction

The question that was troubling me for some time has been answered: “Can anybody make a Blaise survey without writing any Blaise code?” This all started with one project that became larger during time.

The Welfare Benchmark of the Netherlands is a project for the Dutch Municipalities to learn and improve their method of applying Municipal welfare programs. The Welfare Benchmark collects data from the Dutch Municipalities using Blaise programs to form a unique central database. The big challenge in 2003 is that this National project has expended itself to not just 1 Benchmark but to a total of 5 Benchmarks. With only 2 Blaise programmers this was an unthinkable task.

How could we use Blaise at its maximum potential? The answer came with the development of the Blaise Survey Generator, the Excel Survey Generator, and the Blaise SPSS Generator: easy survey making code for non Blaise-users, namely our researchers.

It is the development and implementation of these Generators with the use of Blaise and BCP program and all the procedures of creating a Survey and Survey-results that this paper addresses. We were confronted with the following (technical) aspects: easy Blaise use for the Researchers, easy survey making and remaking for the Researchers, easy Survey-question ordering, easy Survey-table making, easy Survey-routing, and easy Survey-results for the programmers.

Finally, this paper will address which important elements made these Benchmarks so successful and the working relationship with the Central Bureau of Statistics Netherlands, which contributed to its successful implementation.

## 2. The target group

The VNG is the national organization for Dutch municipalities. It contains the nonprofit agency StimulansZ that advises all the municipalities about handling the welfare grants from the government for its welfare recipients. With the help of SGBO (research and consultancy department of the Dutch Local Government Association,VNG) and CentERdata (research expert center of the University of Tilburg) it has made a Blaise Internet program for the Dutch municipalities.

Dutch municipalities send their government official, who is responsible for the Welfare program, to a major meeting of StimulansZ. In large municipalities, this person will delegate the Benchmark project to other government officials but in regular municipalities they would have to enter their data through the use of Internet. Through this data, StimulansZ will measure long-term progress in achieving the Government strategic plan goals: municipal cooperation, collaboration, and a result-driven approach to problem solving. The benchmark results from the municipalities will be compared and the results will show how

compatible municipalities as a whole are doing, and by performance measurement systems, which seek to measure how a given municipality is performing.

This was the first year that only the Internet could be used for data-entry. A lot of data has to be entered by the Dutch municipality that it becomes important for the government official to see an overview of all the entered data. Also with the use consistency tables that show a subtraction, addition of certain questions are necessary for accurate data entry. Otherwise the Welfare Benchmark would be a total failure. This year the government official could easily move from one chapter to another and from one question to another through the whole questionnaire.

### **3. Easy Blaise use for the Researchers: “Blaise Survey Generator”**

Imagine a very busy period and that there is a change in the research staff. A new researcher has just been introduced to your colleagues and you want him to write a data-entry program in Blaise immediately. At that moment you would give him the use of the ‘Blaise Survey Generator’.

The new researcher is an expert in a certain research area and can therefore formulate the questions easily. He also knows the technique of what kind of questions he has to formulate to receive the data he needs for his research. Most importantly he wants to make a routing for the respondent so that the respondent can skip certain questions.

By entering only these specifics in the “Blaise Survey Generator” the job is done. This is a straightforward Blaise program. The base of the Blaise Generator is its Blaise database. It is made of two components. The first component is the Blaise Survey that asks standardized questions to the researcher. The researcher fills in these questions and enters the ‘Question text’ for the generating Blaise survey. The second component is the extensive Manipula job. This will use all the data from the Blaise Generators database.

#### **3.1 The Blaise Survey**

All the Survey information needs to go in a Blaise database. It was decided that every question becomes 1 record. Therefore all the variables are made up of all the different types of questions available in a Survey. The Primary Key is for the question number and the order in which the questions are stored.

The next step was the Blaise DEP program itself. A researcher was becoming a data-entry person. He was not familiar with any data-entry program or activities in this field.

Therefore we made the Blaise DEP program super simple. The researcher has to enter project information in the first couple of questions. The first page gives him information how to accomplish this through the standard questions. There are only 4 speed buttons: New question, All the questions at once, Copy the last question, select the appropriate language. With an auto save-interval of 3 minutes and auto-save when finished, all goes automatically, no worries about data saving.

The most important part was to make it look simple and finally to enter data easily. We found out that just thinking about entering the Survey questions in a database is a big change in thinking for a researcher. By using a rough draft, on a piece of paper, the researcher could enter the questions with more ease. This was especially

important for tables. These can be made easily with the Survey Generator but you need to know first what it looks like before you can enter this information.

### 3.2 The Manipula job

Once all the questions are in the database the next step is to generate a Blaise program. This is all done with the Manipula job. All Blaise programmers write Manipula jobs therefore everybody can change this Manipula file to their own specifications. Before this Manipula job is explained it can be downloaded freely at <http://cdata5.uvt.nl/blaiseusers/> at the tools section. You can change everything for your own use, but please let us know how well it works in your organization.

With the help of the Blaise Generator.bdb a Blaise \*.bla will be created. In Manipula a character or string variable will be filled with characters from a question and then exported to a file. To be on the safe side the maximum length of a string was defined; `FIELDS OneLine:STRING[32767]`.

With the statement `OneLine:=` and eventually after that with `OneLine:=OneLine+` the total text for the source of one question will be combined.

When the Manipula file is activated it starts to read the Blaise Generator.bdb. This will proceed from the first record until the last record with the option of checking the first and the last record. It is not possible to select a specific record. The first record consists of project information; project-number, project-name, maximum respondents. With the first record all Blaise code is generated concerning the start of the \*.bla file like: 'DATAMODEL, ATTRIBUTES, PRIMARY, etc...' and with the last record all the Blaise code is generated that consists of the \*.bla final statement like: ' Finaltime:= SYSTIME, ENDMODEL'. At all the other records the information is read because each record is a new variable (a question in the Survey) and all the parameters are checked, what type is it, what information is entered at that specific type etc. All the information is then written as FIELDS- and RULES-information.

Every syntax statement needs a specific location in the \*.bla job therefore a special code is supplied and is written with the statement `[OutFile.WRITE]` Without this special code the whole generated job would be a disaster. This special code is written with every source statement so that the generated job works perfectly.

The special code works as follows: At the beginning of the job the special code is `{a000 }` and in the Manipula File it is written as `'{a000'+sp15+'}'` and for the end of the job the special code is `{z000 }` and in the Manipula File it is written as `'{z000'+sp15+'}'`.

All the pieces of syntax concerning with the text of FIELDS are written with the special code of `{f000 }` and all the questions with `{f01 }`, `{f02 }`, etc. All the pieces of syntax concerning with the text of the RULES are written with the special code `{r01 }`, `{r02 }`, etc. All the pieces of syntax concerning with tables that consists of Blocks are written with the special code of `{b01 }`, `{b02 }`, etc.

For every record information is written once for the FIELDS, secondly for the RULES, and thirdly for the BLOCKS if applied in a table. By using the statement: `SORT OneLine`, all the source is put in the correct order. To make the generated \*.bla job more readable these special codes have 70 spaces in front of them. The system works fine and can be very useful for future adaptations.

The table functionality takes up 50% of the whole Manipula job. The Blaise Generator database is filled with information such as the number of columns and the number of rows, the type of information that has to be entered in each column. Also if there is a different answer option a sub-question will automatically be

generated with 'Different answer is...'(this applies for every question in the Blaise Generator). In the database is also the information if there should be a total of the column(s) or/and a total of the row(s). The maximum of 9 columns (because of SPSS name length of 8 characters) and 60 rows is applicable. At the moment only 99 questions can be entered into the Blaise Generator. Questionnumber 100 will make the table variable more then 8 characrters. This will be solved as soon as the new version of SPSS will arrive where there is no limit on the lenght of the variable name. The numbers of columns can also be enlarged at that moment.

The filling of a table in the Blaise Generator database is made with a loop and the generating of the table source also is made with loop statements. For each row a Block is generated and within a Block the number of columns is generated for FIELDS and for RULES with all the appropriate labels.

Here is an example of the Manipula job where the type (sort=\_7) is the type of a table entered by the Researcher in the Blaise Generator database. The 'R' stands for rows in the table and the 'K' stands for Columns.

```
ELSEIF soort=_7 THEN OneLine:=OneLine+'T'+nummer
  OutFile.WRITE
  OneLine:=sp70+'{b'+nr
  OneLine:=OneLine+sp15+'}' +char(13)+
  'TABLE T'+nummer
  FOR R:=1 TO tabel.rijen DO {Eerste}
    OneLine:=OneLine+char(13)+
    sp05+'BLOCK B'+nummer
    IF R<10 THEN OneLine:=OneLine+'R0'+STR(R)
    ELSE OneLine:=OneLine+'R'+STR(R)
    ENDIF {R<10}
    Oneline:=OneLine+char(13)+
    sp10+'FIELDS'
    FOR K:=1 TO tabel.kolommen DO {Eerste}
      Oneline:=OneLine+char(13)+
      sp15+'T'+nummer
      IF R<10 THEN Oneline:=OneLine+'0'+STR(R)
      ELSE Oneline:=OneLine+STR(R)
      ENDIF {R<10}
      Oneline:=OneLine+STR(K)
      Oneline:=OneLine+' ' / "'+Tabel.kolom[K].ktekst+'":'
```

Here is the result of a simple table with two columns and two rows of which both have a total and Descriptions is used for the final result in the DEP program:

```
TABLE T02
  BLOCK B02R01
    FIELDS
      T02011 " " / "First Column":1..100
      T02012 " " / "Second Column":1..100
      T0201TL " " / "Total":1..200
    RULES
      T02011
      T02012
      T0201TL:=T02011+T02012
      T0201TL.SHOW
  ENDBLOCK {B02R01-First Row}
  BLOCK B02R02
    FIELDS
      T02021 " " / "First Column":1..100
      T02022 " " / "Second Column":1..100
      T0202TL " " / "Total":1..200
    RULES
      T02021
      T02022
      T0202TL:=T02021+T02022
      T0202TL.SHOW
  ENDBLOCK {B02R02-Second Row}
  BLOCK B02RTL
    FIELDS
      T02TL1 " " / "Total First Column":1..400
      T02TL2 " " / "Total Second Column":1..400
      T02TLTL " " / "Total General":1..800
```

```

      RULES
        T02TL1
        T02TL2
        T02TLTL:=T02TL1+T02TL2
        T02TLTL.SHOW
      ENDBLOCK {B02RTL}
      FIELDS
        TB02R01 " " / "First Row":B02R01
        TB02R02 " " / "Second Rows":B02R02
        TB02RTL " " / "Totaal":B02RTL
      RULES
        TB02R01
        TB02R02
        TB02R03
        TB02R04
        TB02RTL.T02TL1:=TB02R01.T02011+TB02R02.T02021
        TB02RTL.T02TL2:=TB02R01.T02012+TB02R02.T02022
        TB02RTL.SHOW
      ENDTABLE {T02}

```

One of the big challenges for the Blaise Generator was the implementation of the routing in a Survey. The Blaise Generator asks the researcher from which answer possibility the routing will take place and at which question the routing has to stop. This means that the generated routing sourcecode in the RULES should account for multiple routings, not only after one another but also nested in one another and with overlapping one another. This was solved with the use of a memory variable. With each record (question) the specified routing-parameters are stored in several memory variables. If that is the case that question will receive the IF-statement in the RULES when that question may be answered with the specified condition. Counters will keep track of the nested IF-statements and the overlapping questions. So that at the end of the RULES statement the right amount of ENDIF-statements are placed.

The first routing code in the Manipula looks like this:

```

      IF filter=_2 THEN
        ActieveFilters:=ActieveFilters+1
        Mvanvraag[ActieveFilters]:=nummer
        Msoortvraag[ActieveFilters]:=soort
        Mvoorwaarde[ActieveFilters]:=voorwaarde
        Mnaarvraag[ActieveFilters]:=naarvraag
      ENDIF

```

When going through the Blaise Generator database every record is checked if routing information from this record or to another record is available. The variable 'ActieveFilters' gives an identification-number to the memory Array-variables (Mvanvraag, Msoortvraag, Mvoorwaarde, Mnaarvraag), so that every routing will be recognized separately with the appropriate conditions and also where to go to or where it comes from. If during the process a routing from one record is finished, the counter in 'ActieveFilters' will be reduced and when it reaches zero there will be no more IF-statement in the RULES.

The results will look like this in the RULES. A code is placed in when the question is not answered and therefore can be coded as a special missing in SPSS:

```

      NEWPAGE
      v04
      {r05      }
      NEWPAGE
      IF v04=_2 THEN v05:= REFUSAL ELSE v05 ENDIF
      {r06      }
      NEWPAGE
      IF v04=_2 THEN v06:= REFUSAL ELSE IF v05=_7 THEN v06:= REFUSAL ELSE v06 ENDIF ENDIF
      {r07      }
      NEWPAGE
      IF v04=_2 THEN v07:= REFUSAL ELSE IF v05=_7 THEN v07:= REFUSAL ELSE v07 ENDIF ENDIF

```

With the Blaise Generator you can put in routing just as easy as making questions. At the moment of writing this paper only ONE answer possibility of a question can be used in the routing. We are working on the option of more answer possibilities.

## **4. Excel Survey Generator**

One researcher loves working with MS Excel and presented us all the questions in a spreadsheet. This researcher liked the overview of seeing everything together by using the Tab-sheet functionality. This meant that the researcher also included all the Table questions and SPSS formulas. Because a lot of discussions are going to take place before the final Survey is ready and all the questions change all the time we made a 'Hand-out' Tab-sheet for the researcher.

At that moment we already had the Blaise survey Generator and we could use that for all the table questions which we put into a separate Tab-sheet. The next step is to re-arrange the researcher's MS Excel spreadsheet into something we could use for Blaise. We started to use this spreadsheet as a database.

At this moment there is no need for Blaise syntax knowledge. The researcher enters the questions in cells of a spreadsheet. It is less advanced as the Blaise Survey Generator but it is more comprehensible for the researcher in larger projects. Most researchers have a lot of experience with Excel and feel comfortable seeing all the questions in a column.

### **4.1 The Question-sheet**

All the Survey information needs to go in a Blaise database. Every row in Excel is a new record and every Column is either part of a variable of a new variable for the Blaise database. Editing the questions is therefore a fast procedure for the researcher. If the researcher wants to change the minimum and maximum of a couple of questions he just copies the values along all the rows in one column. The information that is entered is the following: question- : number, -text, -type, -values.

### **4.2 The Table-sheet**

This part is undeveloped at this moment. The researcher enters all the tables on this sheet. The Blaise programmer then uses the Blaise Generator to enter all the tables. This generates the 'Blaise tables include file' for the project. The Table-sheet provides a nice view of the tables for the researcher when he prints a handout and also it is used for a html file.

### **4.3 The Formula-sheet**

All the Survey information needs to go in a Blaise database but the formulas will be used for SPSS. The researcher can enter definitions of the formulas which will be calculated in statistical software and produce either SPSS tables or a total overview of benchmark participants comparable percentages or numbers in a final Excel sheet with labels.

### **4.4 The Macro-sheet**

All the Survey information needs to go in a Blaise database. Only the questions are put into Blaise. Visual Basic is used to write all these applications. All the cells in the different columns are used and put together in a print command. Creating Blaise chapter Blocks which are then used by a manually created \*.bla file. Using the same method the types.inc file is created. Also a handout document is created in Ms Word.

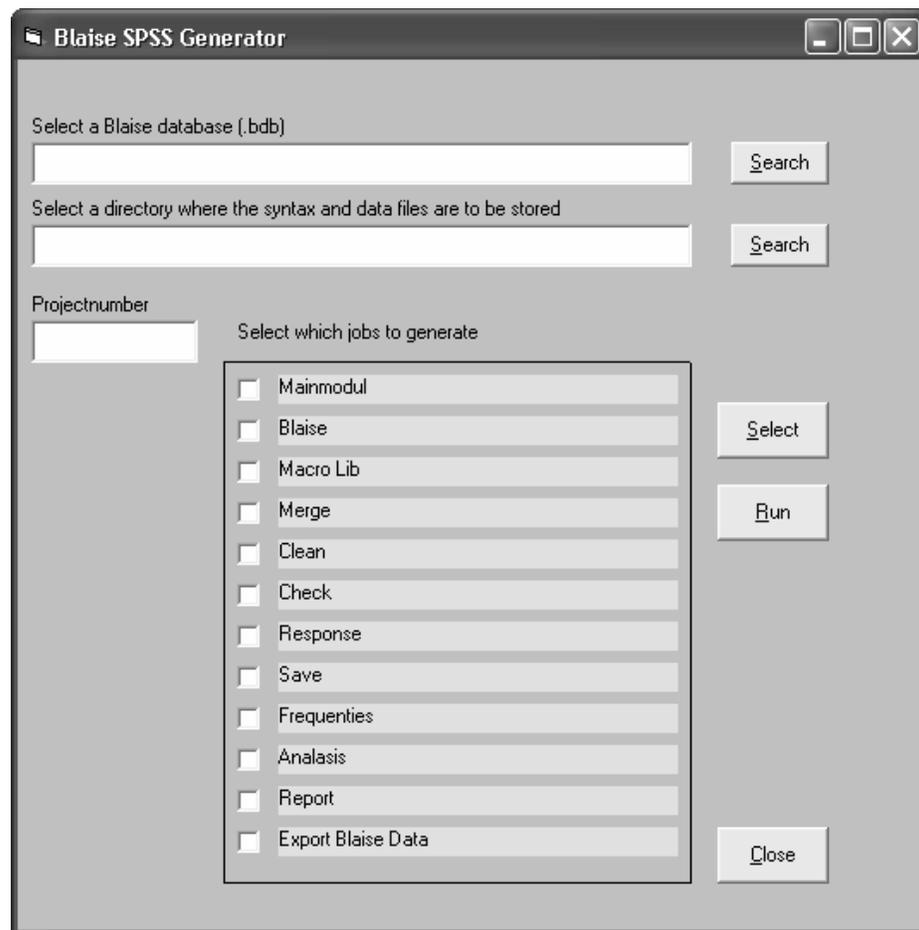
Here is a Code snippet from the Visual Basic macro:

```
Sub ExportBlaiseTypes()
  'declare variables
  Dim file As String
  Dim TypeName As String
  Dim TypeCat As String
  Dim filename As String
  Dim sh As Worksh
  Application.ScreenUpdating = False
  'Activate the excel sheet that holds the type information
  Set sh = Worksheets("Survey")
  sh.Activate
  sh.Cells(2, 1).Select
  ActiveCell.CurrentRegion.Select
  'Determine the number of questions in the survey each question gets it's own type
  NumberOfQuestions = Selection.Rows.Count
  filename = Worksheets("Index").Cells(1, 5).Value
  file = bestandnaam & "_TypesLib.inc"
  ' Open text file for blaise types.
  Open file For Output As #1
  Print #1,
  Print #1, "{TYPE Library}"
  Print #1, "{Survey naam: }"
  Print #1,
  Print #1, " TYPE"
  'Loop through all the questions in the survey
  For i = 2 To NumberOfQuestions
    'TypeName is the concatenation of the datatype en the question number
    TypeName = sh.Cells(i, 4).Value & "_" & sh.Cells(i, 2).Value
    'Type Data Kind is integer or decimal
    If UCase(sh.Cells(i, 4).Value) = "INT" Or UCase(sh.Cells(i, 4).Value) = "DEC" Then
      TypeCat = sh.Cells(i, 5)
      'Type data kind is string
    ElseIf UCase(sh.Cells(i, 4).Value) = "STR" Then
      TypeCat = sh.Cells(i, 5)
      TypeCat = "STRING [" & TypeCat & "]"
      'Type data kind is a set
    ElseIf UCase(sh.Cells(i, 4).Value) = "SET" Then
      TypeCat = sh.Cells(i, 5)
      TypeCat = Replace(TypeCat, ":", " " & vbCrLf & " _")
      TypeCat = "SET OF " & "(" & TypeCat & ")"
      'Type data kind is a single response question
    ElseIf UCase(sh.Cells(i, 4).Value) = "ENUM" Then
      TypeCat = sh.Cells(i, 5)
      TypeCat = Replace(TypeCat, ":", " " & vbCrLf & " _")
      TypeCat = "(" & TypeCat & ")"
    End If
    'Write the type syntax to the text file
    Print #1,
    Print #1, TypeName & " = "
    Print #1, " " & TypeCat
  Next i
  Print #1,
  Print #1, "{End Types}."
  Close
  Application.ScreenUpdating = True
  Set sh = Nothing
End Sub
```

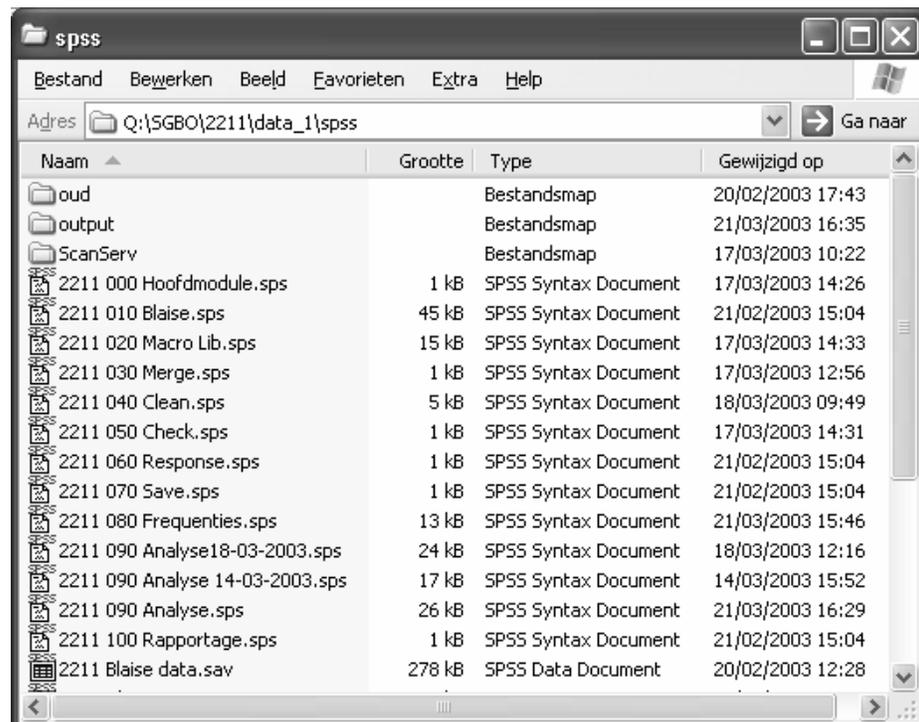
## 5. Blaise SPSS Generator

When all the data is in a Blaise database all the Meta information is available. Every project needs a Blaise database with all the Survey Questions. This is the starting point for SPSS analyses. The Blaise SPSS Generator actually generates all the SPSS syntax in different SPSS files from the Blaise Meta data.

This is what it looks like when you start the Blaise SPSS Generator:



The output files are the following:



The output files are used in order to go through every step in SPSS which is used for statistical analyses.

## 5.1 BCP

The \*.bdb is selected once you enter its location in the Blaise SPSS Generator therefore automatically \*.bmi is used and all the Meta-data of the project is available. The provided Cameleon job of creating a SPSS job has been used as a basis for the development of Visual Basic program that uses BCP.

First the import job of a Spss program is created so that all the data from a Blaise datafile can be imported with all the labels and value labels and missing statements. In Visual Basic especially the Block functionality is complex because of all of its labels. Spss file meta-data is one-dimension and Blaise block-tables have more dimensions because of the layers. You have to go through all these layers to get the right labels and value labels. A loop functionality is used through every cell in the table and after all the loops have been finished the spss syntax will be written.

The check functionality of the functionality descriptive uses all the variables with numeric 'data type with Float and Integer types with the exception of the Set and Enumeration types' that are selected from the Meta-data.

The final Spss job is the frequencies.sps. This creates all the basic tables for the researcher. By looking into the Metadata you can see what type of question is used and therefore the appropriate Spss-macro is written next to the variable. The multi-response information also comes from the Meta-data, automatically two Spss macros; Categorized and Dichotomy.

Export ascii data is done automatically by using a pointer starting in record one until the last record. This creates the ascii.txt data set which can be imported into spss.

The biggest advantage of this generator is that routine jobs are automatically produced and a standard working environment is created.

Before we started with the Blaise SPSS Generator we used Blaise Database description program in BCP that automatically created all the Meta-data information in Excell. This is good basis for understanding the explanation of the Blaise SPSS Generator. See the attachment at the end for the BCP source.

## 6. Future

The Blaise Generator comes to Intranet for our researchers. At the moment we are working on paper output from the Blaise Generator. This means that we soon will have a preview functionality and output to MS Word for a paper Survey. In the end we will have one Generator that will do all our standard work.

## 7. Conclusions

Yes it is possible to write Blaise Syntax without the knowledge of Blaise code. By using the Blaise Generator a researcher without any knowledge of Blaise can make a survey by answering simply questions and selecting the right type of answers, this includes complex tables and routing of the survey. With Manipula the Blaise syntax is made and the researcher can look at a Blaise Data-entry program straight away. The Manipula is known to every Blaise programmer and therefore easily adapted to every organization.

By using the Excel Survey Generator researchers who are comfortable with Excel can put the questions and types in columns. This is far less advanced as the Blaise Generator but it provides a comfortable feeling for the researcher at the moment.

The SPSS generator is wonderful for 'us' programmers. The Blaise Meta-Data provides all the information for making standard tables and graphs in SPSS. With the use of BCP in Visual Basic the source is written and the outcome are all \*.sps files that can be used straight away.

The StimulansZ/SGBO Benchmarks will serve as outcome measures for the Municipalities results-driven approach to problem solving. Blaise is used for the Benchmark at its maximum potential: easy survey-making for non Blaise-users, namely our Researchers. Blaise is the main engine behind Data collection of the Benchmark and proved to be a major success.

# Blaise Questionnaire Text Editor (Qtxt)

Grayson Mitchell, Statistics New Zealand

## 1. Abstract

**Qtxt** is a program designed to reduce the amount of work involved with the production of large questionnaires. **Qtxt** achieves this by enabling the questionnaire designers to edit question text without the need to know about Blaise syntax. This has reduced the amount of communication required between the questionnaire designers and the application developers.

Another key feature of **Qtxt** is its ability to handle multiple languages (including Chinese). The **Qtxt** program always displays English (or a selected base language) as a reference, and the new language is entered in another pane. The aim was to provide external translation agencies with a system to translate question text alternative languages.

**Qtxt** also allows the user to edit question fill text and we have fill text to be set up as category types. This means that all question texts are all stored in one place, and more importantly are easily editable with **Qtxt**. Another benefit of this approach is that our fills can easily be written in multiple languages without needing to maintain the Blaise code for the variables involved.

The question texts are structured. Each question belongs to a module, and each module belongs to a questionnaire (which we store in Microsoft Access format – though they could easily be stored in other formats). After exiting **Qtxt** the files are exported into a Blaise source file which can be compiled.

**Qtxt** allows the user to edit question text in an easy to use environment, similar to many other rich text editors.

The evolution of the Blaise API has allowed for more possibilities in the future (for instance the ability to read the *modlib* using the API). Statistics New Zealand is looking for opportunities to improve our questionnaire design process and **Qtxt** has helped with this.

### Success Criteria

- Reduced involvement of technical staff in text changes
- Ease for non-programmers to change Blaise text
- Ease for external agencies to change Blaise text (such as translators)
- Allow non-programmers to change formats/ layout of text
- Integration into existing tools (e.g. our Survey Management system)
- Management of text changes on stand alone PC's (e.g. distribution to translators)
- Easy to use system for code fills in text
- Export text to Blaise code
- Edit multiple languages (including Chinese)
- A stand alone product

## 2. How Blaise is used in SNZ

Statistics New Zealand (SNZ) use Blaise mainly for Social surveys. These fall into two categories: paper questionnaires and electronic questionnaires. Most of the paper questionnaires have been around for a while, and the Blaise data entry systems were also developed some time ago. In recent times we have been writing much more complicated electronic questionnaires. With these electronic questionnaires we have incorporated new technologies such as the Blaise API. Blaise tools such as Cameleon and Manipula are mostly used with the older systems, or the odd ad-hoc job.

The Questionnaire Design Consistency (QDC) are given 'topic specs' that detail the outputs expected from the questionnaire. They then use these specs to devise their questions. If the questionnaire is an electronic one then this design will always be in the form of a flow diagram. Once they are happy with a module (or section) then they deliver the flow chart to the programmers (Technical Application Services) who write the code. The module is set up so it can be run in isolation. Then QDC test the module. When they find errors they tell the programmers about it, and then the programmers go and fix it.

So in Statistics New Zealand our questionnaire designers do not program Blaise, TAS do all of the programming. One of the disadvantages of this approach is that TAS can spend a lot of time on textual changes, which is a waste of resources. Training our questionnaire designers how to program Blaise might resolve some of those issues, but would introduce a whole lot more problems; managing the different tiers of Blaise programmers for one.

## 3. Brief History of Qtxt

So as you can see a lot of time is spent in communicating small changes to the programmer. The programmer tends to get frustrated at they get a stack of changes, and one page might be specifying that a full stop is required. Then from the QDC persons viewpoint they get annoyed because it takes a week before they get to see the program with that full stop in place.

This is the situation that made me come up with Qtxt. I did not want to make any question text changes, but I did not want QDC staff to be mucking about with our code at the same time. So I invented a tool that enabled QDC to edit text without having to worry about Blaise formatting or code.

Qtxt version 1 was a Visual Basic program. It stored the question data in individual Blaise files, and translated the formatting code into rich text for editing and then back again on saving the document. Then there came a need for us to do a multi-lingual questionnaire. Qtxt proved not to be robust enough to handle all of the languages, and the text manipulation became unwieldy. The scope of the project had grown considerably since its conception.

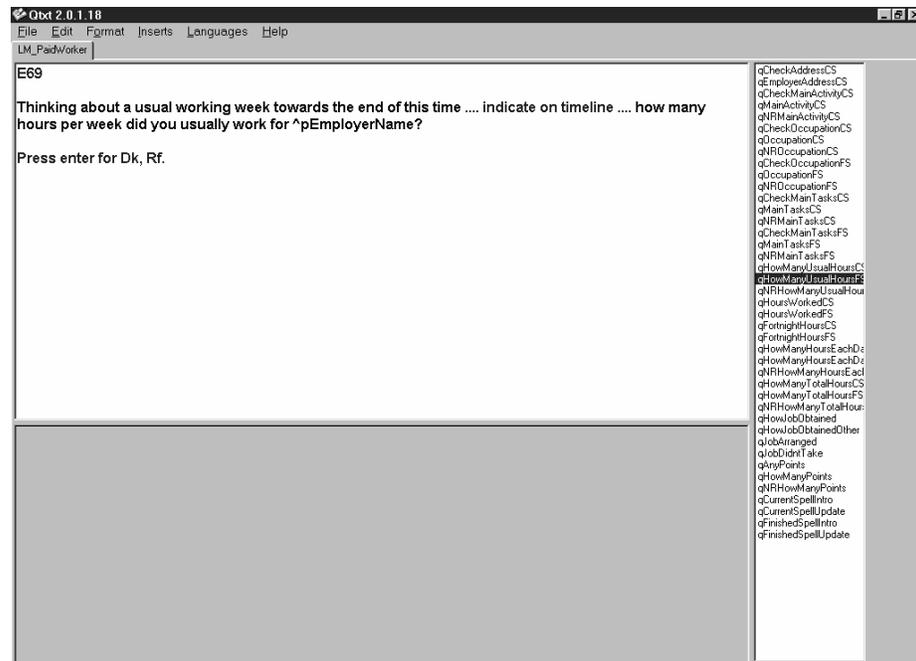
Qtxt version 2.0 is our current version. It is written in Delphi, we changed to Delphi after discovering several shortcomings of Visual Basic 6. It stores all of a questionnaire's text into a database as rich text, and so is far more robust. It allows the editing of fills in multiple languages and can handle Chinese and other languages. In fact a lot of its design and functionality have been put in place for the express purpose to make multi-lingual questionnaires easier to produce.

## 4. Qtxt-User interface

Qtxt main function is a text editor, and inherits a lot of its functionality from the 'TRichEdit' class in Delphi. There have been a quite a few enhancements made to make question text editing more user friendly. On the picture below you will see that on the right hand side of the screen there is a list of question names. This list enables the user to select what question they want to edit, and the text for that question is loaded into the text box.

This means that there is no possibility of the wrong text being edited. Also you will note that no Blaise code is visible. There are no @B's and no type definitions, this makes Qtxt very intuitive to use, and little or no training is required for general text changes. Your average user will be only use the navigating features to move through the questions, typing amendments, or changing the text colours. As you navigate through the questions changes are automatically saved to the questionnaires database, and when you close the Qtxt program a Blaise formatted file is exported to the appropriate directory.

There are other standard type functions too, like you can add/delete questions and even modules. But these are usually reserved for the programmer rather than the standard user.



## 5. How does Qtxt interact with Blaise?

The text itself is stored, as rich text in tables, so is not accessible to Blaise. To keep the Blaise code always up to date Qtxt always exports a copy of the question text in Blaise format after it has been edited in Qtxt. This means there are essentially two copies of the same text, the master copy in the Qtxt database, and the local Blaise copy.

To keep this export system simple we have separated the question text from the actual code. This means that each module has an extra include file with a .Qtxt extension. This has been adopted as a SNZ standard. E.g. a .Qtxt file:

Fields

```
qCheckChild
"@/.
@/I-have-your-name-as-the-person-who-can-answer-a-few-questions-about-^aWholeName.
holeName.
@/.@B.
@/Press-enter-to-continue@B."
: String,EMPTY

qCheckName
"@/.
@/Can-I-just-check-..Is-that-child's-full-name-^aWholeName?"
: tNRYN
```

**Note:** When we exported to Blaise we chose to export the <ctrl .> character rather than a space. This proved to be a real problem when we came to use Chinese with Qtxt. Chinese (and other Unicode languages) require that key for a character, so I had to make the exporting of <ctrl .> a option. The reason why we wanted to have the dots is because sometimes Blaise removes spaces when the code is run, an example being the that two spaces after a full stop becomes one. The users did not like that.

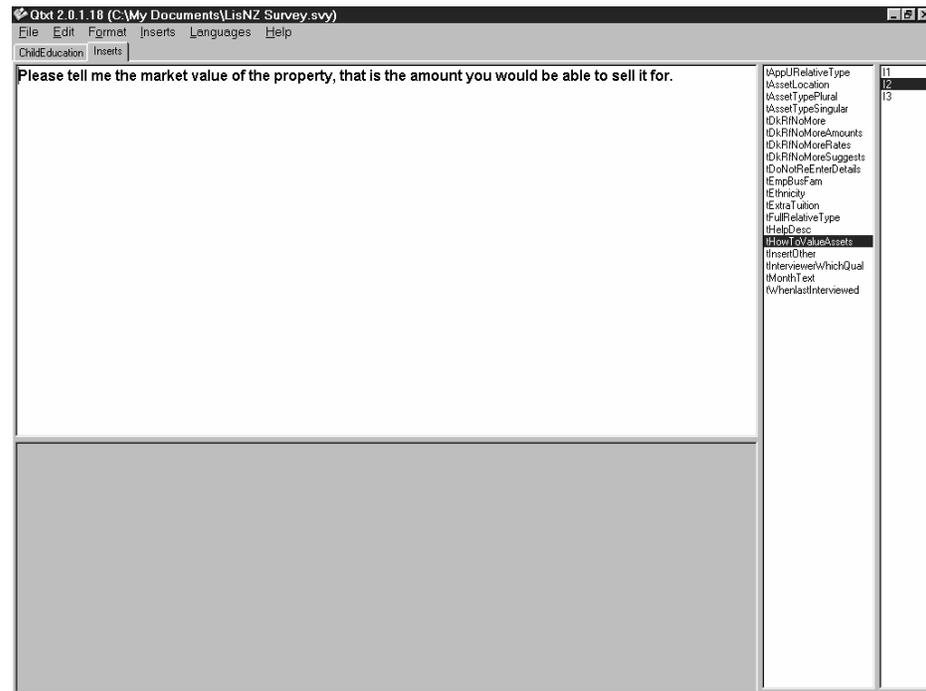
## 6. Advanced Features - Fills

One thing we have found to get out of hand in the past is text fills. With the standard way they are programmed it is not simple for a non-Blaise user to edit the values of fills. So what I came up with is a way to write fills in the Blaise code that does not include the actual text to be displayed. This has several attractive benefits. First we can manage and see the fill values much more easily as they are in one place. And we can also have multilingual fill values. To do this I set up a fill include file that lists all of the fills, e.g.

Type

```
iWasIs =  
  (I1      "was",  
   I2      "is")
```

Then in your code it is just a simple matter of defining a variable, and assigning to either I1, or I2. You don't deal with any text at all within your code. For users to edit the inserts can use Qtxt in much the same way, there is just an extra listbox with the fill definition. Also when encountering an insert in some question text the user can press <ctrl-s> and Qtxt will take them to that insert to edit.



## 7. Advanced features - Multilingual

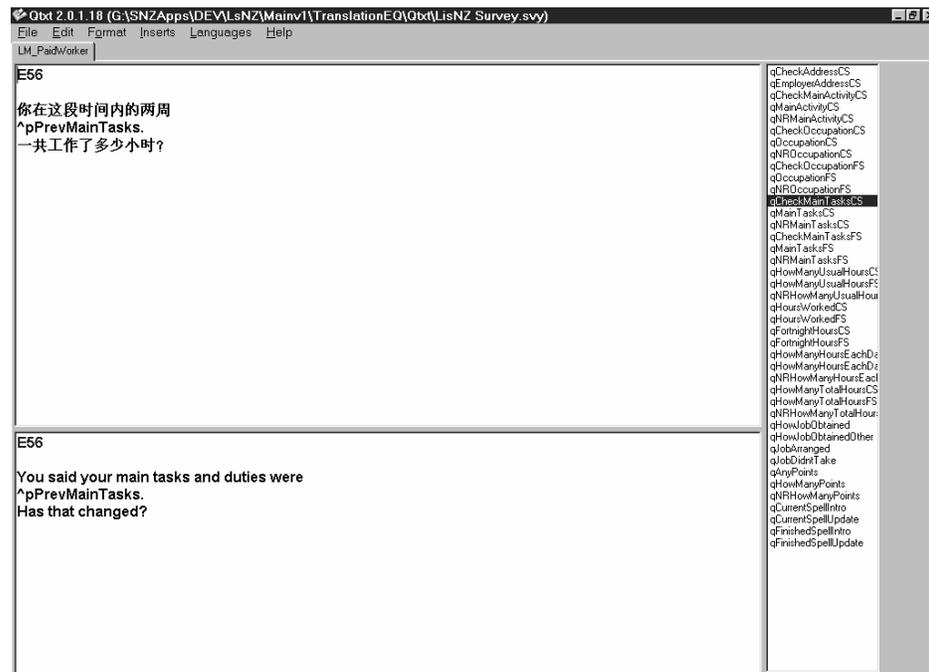
Qtxt has been designed with multi-lingual questionnaires in mind. When editing an additional language the bottom of the screen shows the original language text to aid in translations as you can see below .

Logistically translating questionnaires externally can be a nightmare. I will not go into the problems we have gone through here, but there have been many. Foreseeing technical issues is extremely difficult when most of the developers only speak one language.

In Qtxt you can import and export to rich text format, which the translators can then edit in Word. This solved a lot of technical issues with getting Qtxt to work in various centres around the country/globe. When they are done we then read those files back into Qtxt.

We have several modifications to the formatting of these files to make them as easy to translate as possible. One of these changes is to make inserts as clear as possible. Text in Blaise might be 'you once told me ^name was ill' this would become 'you once told me [name] was ill'. Then we can tell the translators to not edit any text with square brackets around it. And because even then they will translate the odd insert by mistake, we have a checking program that check the English vs. the translation to make sure it still contains the same inserts.

So after changing the text you can click on the recompile button, and the question text will be updated within the instrument.



## 8. Appendix A- Module Manager

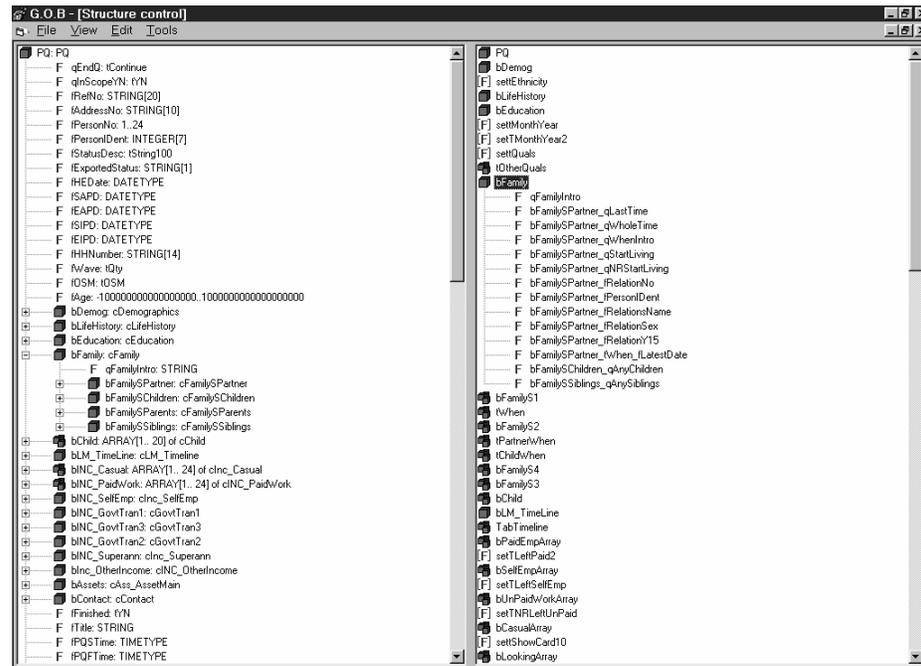
In Qtxt a lot of the time wasted with unnecessary text changes has been alleviated. After looking at how QDC/TAS did their design work we saw a lot of areas where an integrated design tool would help. This is how the Module Manager came about. With it you can do anything required to design a questionnaire. You can run a module, edit its question text, open up its flow charts, and even compile the code. This tool is mostly used by QDC, but my vision is that TAS developers also use it.



## 9. Appendix B- B.O.S. (Blaise Output System)

BOS is the most technically challenging Visual Basic program we have written that utilises the Blaise API. The problem of how we get data out of Blaise has been one that has been at SNZ as long as I have been there. Every questionnaire we have developed has seems to try and devise a new approach to getting the data out of a questionnaire with least amount of effort.

Being the type of person who does not accept doing the same job twice I came up with BOS. What it does is load a Blaise map into a window, you can then generate a normalised view on how BOS will output the data. Then by cutting and pasting you can move fields around, rename them, delete them, etc. When you are happy with how the output tables are going to look, then you can create those tables, and populate them with data.



## 10. Appendix C- Where from here

So far we have managed to do some individual projects that have helped to make streamline some of the processes that have proved to be cumbersome in the past. This year we are looking at the whole survey design process, and how it all fits together. Hopefully after our investigations SNZ will be able to further streamline more of our development processes.

Where I see a lot of room for improvement are in the Module Manager utility. What I want to see here is a management tool that will be able to keep track of all aspects of a questionnaire development, as well as being a tool that allows access to the development environment. Some areas for improvement include.

- keeps track of code changes.
- maintain backups
- easier access to code edit information
- maintain a milestone date list for modules
- keep track of who is editing modules.
- manage rights of users

## 11. Appendix D- Generation of questionnaire code

There has been a lot of interest at SNZ in a project I proposed some time ago to generate code (possibly Blaise) from our flowcharts. The technology is definitely there to do this. Our tool flowcharter 2000 has a VBA backend that allows much to be done with the data stored in a flowchart. It is a direction that we would like to head, but I see it as more an icing on the cake rather than an ultimate solution. To generate questionnaire code from flowcharts requires that all questions adhere to certain standards. The most important thing is that you have those standards in place. Once we have all of our question and module types defined then life becomes that much easier for us. Generating the code from our flowcharts then becomes a logical next step.



# Automatic Upgrade of Production Blaise Instruments

*Lilia Filippenko Joseph Nofziger & Gilbert Rodriguez, RTI International*

## 1. Introduction

Sometimes it is necessary to upgrade instruments when data collection has begun. But for modifications that alter the structure of a Blaise data model, it is difficult to do: an existing production Blaise data files must be upgraded with the new data model when the data model has changed. Although the Manipula setup for this procedure is very simple, many steps need to be taken to avoid loss of data: creation of backup folders, copying of a number of files, the preparation of Manipula setups for the upgrade and so forth. It can be an extremely complex process, especially when instruments reside on multiple computers, possibly with different versions.

At RTI International we have projects that use Blaise data models for CAPI, CATI, and ACASI instruments. A Master database resides at RTI, but individual interview cases are assigned to different Field Interviewers who use laptops to conduct the interviews. The cases are therefore distributed among laptops. When an interview for a case is completed, its data file will be sent back for loading into the Master database. When a case is transferred to another interviewer, its data files are transferred between laptops. An upgrade can of course occur at any point. So applying the upgrade means upgrading the Master database, data files on a number of laptops (it could be hundreds), and - posing a particular challenge - many cases that are in transit.

A Visual Basic program was written to perform these tasks. A key piece of data for the program is the version number of the instrument. Every time the data model is changed, it should be prepared using the corresponding Blaise project with a new version number.

Several additional requirements drove the design of the program:

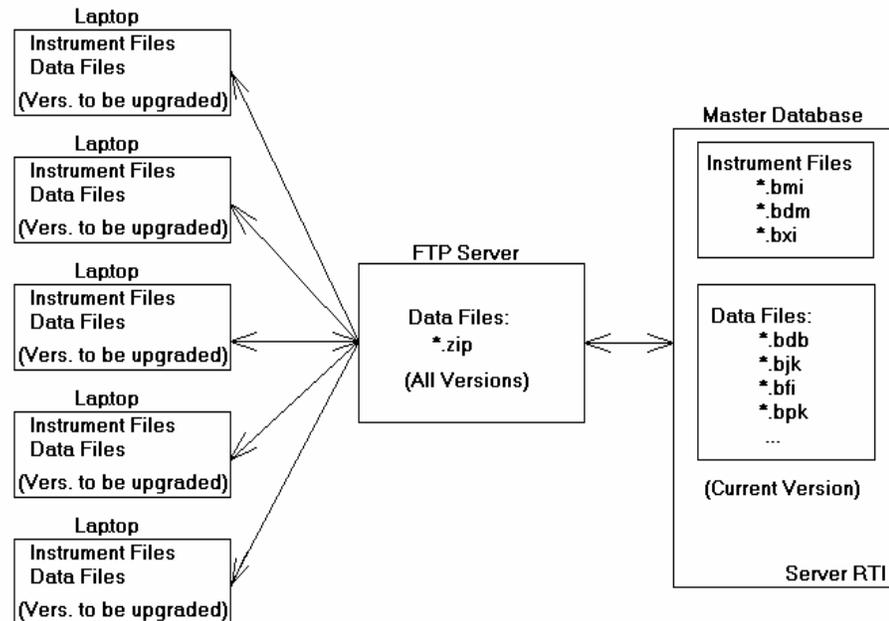
- Upgrades must be reliable - all data and other Blaise files are backed up before upgrading, and restored on failure.
- Manipula setups that depend on the data model must be prepared whenever the Master database is successfully upgraded.
- It must be possible to upgrade from any previously released version.
- And finally, we must be able to monitor whether field laptops have been upgraded successfully.

## 2. General Approach

The usual structure of the project is shown in Figure 1. The Master database resides on the server at RTI and contains a current version of the project: the instrument and the data files (all interview cases). Before project is started, all needed software and instrument files are loaded on laptops at RTI. Laptops are then issued to Field Interviewers. The cases are initialized in the Master database, placed on an FTP server in individual files, and sent to laptops when the project is started. The same server is used to transfer data back to the Master database during data collection.

The structure of the data files is defined in the project design stage and normally could not be modified after data collection has begun. However, such modifications are often desirable. An upgrade of the project could be a result of the customer's late request, hardware modification, or bug found in one of the project components. To do so we have to upgrade the instrument files at each laptop as well as the data files, which already contain collected data.

Figure 1



However, the Blaise software provides a tool to make such an upgrade possible and easy. The main idea is to utilize the version number, which exists for each project, and which is assigned to every element of the project: instrument and data files. When the Master database is upgraded, each component inside the database receives a new version number, and this new number is the key to a successful upgrade of the project components on the laptops even after the data collection has begun.

To process the upgrade a special Visual Basic application, UBI (Upgrade Blaise Instrument) is created and distributed to all laptops. The UBI first handles the upgrade of the Master database components. During the upgrade of the Master database the UBI prepares Manipula setups to be used later in upgrading the laptop's data files. We should, however, remember that versions of the case data files on the FTP server could differ from the current Master database version, as well as from the laptop's files. This is because the previous upgrades (if any) upgraded the data files on the laptops and in the Master database, but not on the FTP server. Therefore, the UBI requires special Manipula setups to handle conversion from any old version to the current one for individual case data files.

### 3. Description of the UBI

#### 3.1 Naming convention for directories and Manipula setups

An important part of the upgrade is maintaining the upgrade history and backup files on each machine. It will be used as a backup copy if the upgrade fails. Also, we need to have all previous instrument files in case we need to upgrade case data files that are significantly older than the current version.

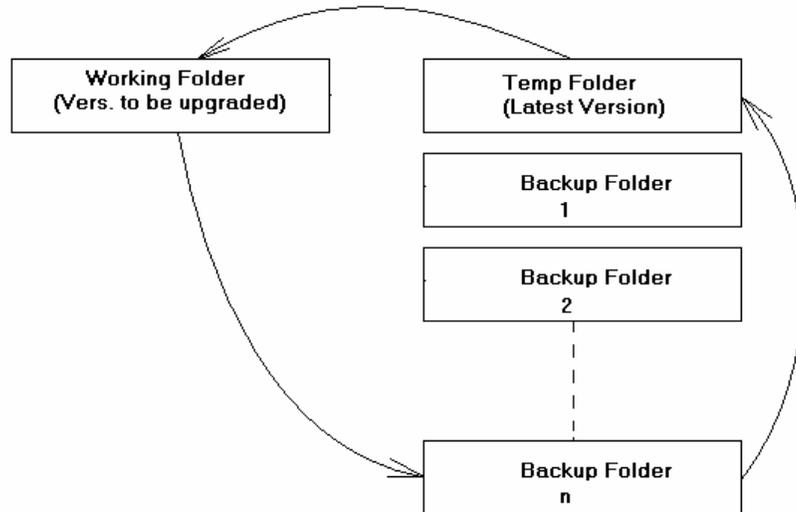
A special rule on the naming of backup directories names provides a way to keep records on the history of upgrades: the name consists of the instrument name and the version number separated by an underscore:

<inst>\_<version>

This allows us to provide the UBI with a simple way to identify the location of the files that are necessary for handling an upgrade from any older version.

A typical directory structure is shown in Figure 2.

Figure 2



Names for the Manipula setups that perform the upgrade include both version numbers: the old version and the latest one. This is used to determine what Manipula setup should be used to do the upgrade:

<SetupName><OldVersion>To<NewVersion>

### 3.2 Modes used with UBI

The UBI is used in three modes depending on the kind of data files that need to be upgraded: Master database, database on laptops, and data files for individual cases.

### 3.2.1 Upgrade of Master database

When the decision has been made to upgrade an instrument that is already in the production stage, the upgrade should be applied first to the Master database.

Our approach for upgrading to the Master database is as follows:

- Determine the version of the current instrument in the working folder
- Create backup folder with all current instrument files
- Upgrade data files from the backup folder to the temp folder
- Copy new instrument files and upgraded data files from the temp folder to the working folder

To perform these major steps, the latest version of instrument files is placed in the temp folder. When the UBI is started, the current version – prior to upgrade - all instrument files, data files, and Manipula setups are copied from the working folder to a new backup folder. This will be the base version for the upgrade, and will remain for future upgrades of individual cases. It also serves as a backup in case the upgrade fails.

The UBI automatically creates the Manipula setup and calls the B4Cpars.exe utility to prepare the Manipula setup. It then invokes the prepared Manipula setup to perform the upgrade. Only after successful completion of this process will the instrument and data files in the working folder be replaced with the new instrument and data files from the temporary folder. This allows us to perform any necessary tests on the new data files before committing the upgrade to the working folder.

Then the UBI creates and prepares Manipula setups to do an upgrade for individual case data files that will be used later to load cases in the Master database or on a laptop.

During execution of the UBI for upgrading the Master database, messages will be displayed and a log file will be created with additional information for future review.

Flowchart 1 presents a diagram for this mode of the UBI in Appendix A.

### 3.2.2 Upgrade of individual cases

For transport of individual cases among laptops or between a laptop and a server, we extract case data to a Blaise data file containing only the data for the case. When it reaches its destination, the data file may be an older version than the one into which it is to be loaded. For such cases we will upgrade the data files directly to the working folder rather than to a temporary folder. But we should bear in mind that some cases could be a few versions behind before they reach their destination. Perhaps they are en route from a laptop that had not yet successfully applied the upgrade at the time the case was extracted.

Our approach is as follows:

- Determine the version of the case data file in the working folder
- Copy the file to the appropriate version folder
- Upgrade from the version folder to the working folder

During execution of the UBI for upgrading case data files, information will be added to the log file for future review. On laptops a special upgrade status flag will be created. If the status is “Failure”, information about this particular case is

written to an Access database to be sent later to RTI for review. Such cases will be re-sent to the field laptop.

Flowchart 2 presents a diagram for this mode of the UBI in Appendix A.

### **3.2.3 Upgrade of database on field laptops.**

After successfully upgrading the Master database we will have the new instrument files, new Manipula setups for upgrading database on laptops, and Manipula setups for upgrading cases while loading them into the databases.

If laptops contain additional Manipula setups that refer to the instrument files, those must also be prepared with the new instrument. We prepare these in house. Finally, we create a command script to invoke the UBI on the laptop with the proper parameters.

All these files are uploaded to laptops, where the command script is invoked automatically using existing mechanisms. Most of the same steps described previously for upgrading the Master database will be performed to upgrade the database on the field laptop.

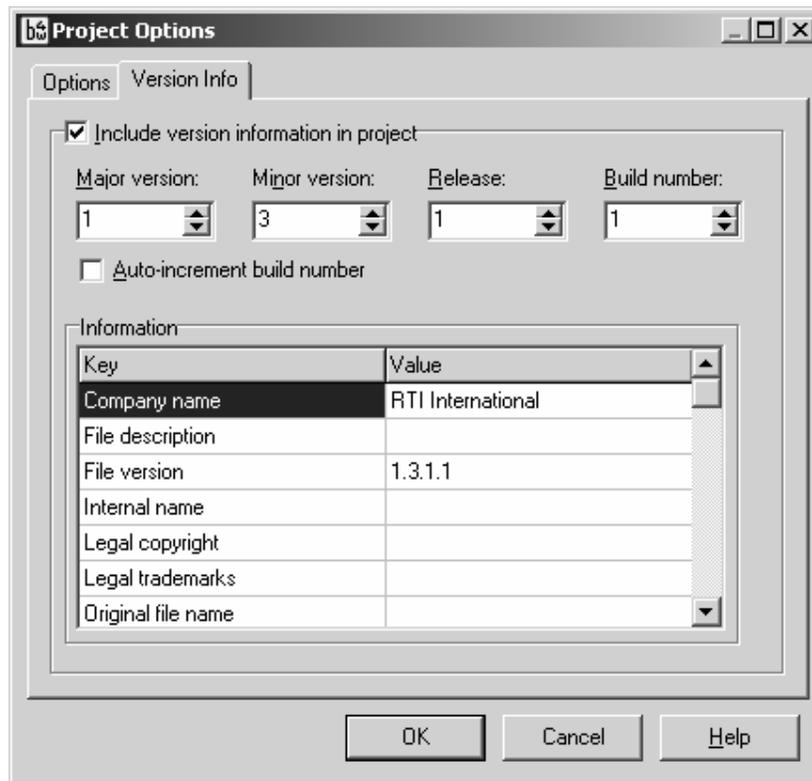
In the laptop environment there will be no user interaction; all messages will be logged. The next time the laptop connects to RTI, the more critical messages will be sent to a server log, including the new instrument version on the laptop. If errors occur and the upgrade fails but the information in the server log is not sufficient to determine what occurred, we can set up a request to retrieve the log file from the laptop for review at RTI.

Flowchart 3 presents a diagram for this mode of the UBI in Appendix A.

### **3.3 Example of Master database upgrade**

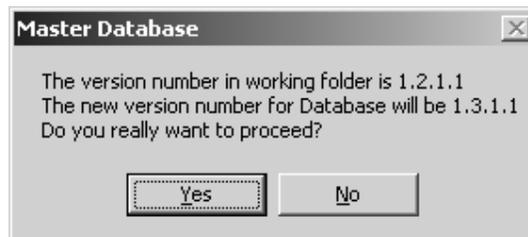
As an example we will use the instrument with the name “hope”. The current version for the instrument “hope” is 1.2.1.1.

When changes to the instrument files require upgrading data files, the new version should be specified for the instrument. This is done in the Control Centre under Project options menu. For this example the new version for the “hope” instrument became 1.3.1.1



New instrument files hope.bmi, hope.bdm, and hope.bxi are copied in the “temp” folder created under the folder where the current (1.2.1.1) version of the “hope” instrument exist.

The UBI accepts the instrument name, working folder name with a full path, and a name for the temporary folder as parameters. When the UBI starts, it will determine the versions of the new instrument in the temporary folder (“temp”) and the old instrument in the working folder.



When we answer “Yes” to start the upgrade, a new backup folder is created under the working folder with the name “hope\_1.2.1.1”. Files from this folder are used for upgrade Master database.

The UBI creates the Manipula setup in the working folder.

#### **BlzUpdateBlz.man:**

USES

Old 'Hope\_1.2.1.1\Hope'

New 'temp\Hope'

INPUTFILE InFile : Old ('Hope', BLAISE)

OUTPUTFILE Outfile : New ('Hope', BLAISE)

MANIPULATE

Outfile.WRITE

Note that upgraded data files are not created in the working folder but in the “temp” folder. The prepared Manipula setup (BlzUpdateBlz.msu) is created by invoking the B4Cpars.exe utility in the “temp” directory.

The UBI runs the upgrade Manipula setup BlzUpdateBlz.msu and the upgraded data files are created in the “temp” folder. The version number assigned to the new data files should become 1.3.1.1. When the UBI determines that the Master database has been upgraded successfully, all instrument files: “hope.b\*” are copied to the working folder. The working folder now contains version 1.3.1.1 of the “hope” instrument.

New Manipula setups are created for upgrade of individual cases. In this example the “hope” instrument is upgraded for the second time and folder “hope\_1.1.1.1” has instrument files for version 1.1.1.1. Two Manipula setups are created and prepared:

#### **BlzUpdCaseBlz1\_1\_1\_1To1\_3\_1\_1.msu**

```
USES
Old 'Hope_1.1.1.1\Hope'
New 'Hope'
INPUTFILE InFile : Old ('Hope', BLAISE)
OUTPUTFILE Outfile : New ('Hope', BLAISE)
MANIPULATE
Outfile.WRITE
```

#### **BlzUpdCaseBlz1\_2\_1\_1To1\_3\_1\_1.msu**

```
USES
Old 'Hope_1.2.1.1\Hope'
New 'Hope'
INPUTFILE InFile : Old ('Hope', BLAISE)
OUTPUTFILE Outfile : New ('Hope', BLAISE)
MANIPULATE
Outfile.WRITE
```

The last step performed by the program is the deletion of old prepared Manipula setups that are used by other processes, such as loading any finalized cases into the Master database. Those Manipula setups will be prepared again for the new instrument files by other programs.



## **4. Summary**

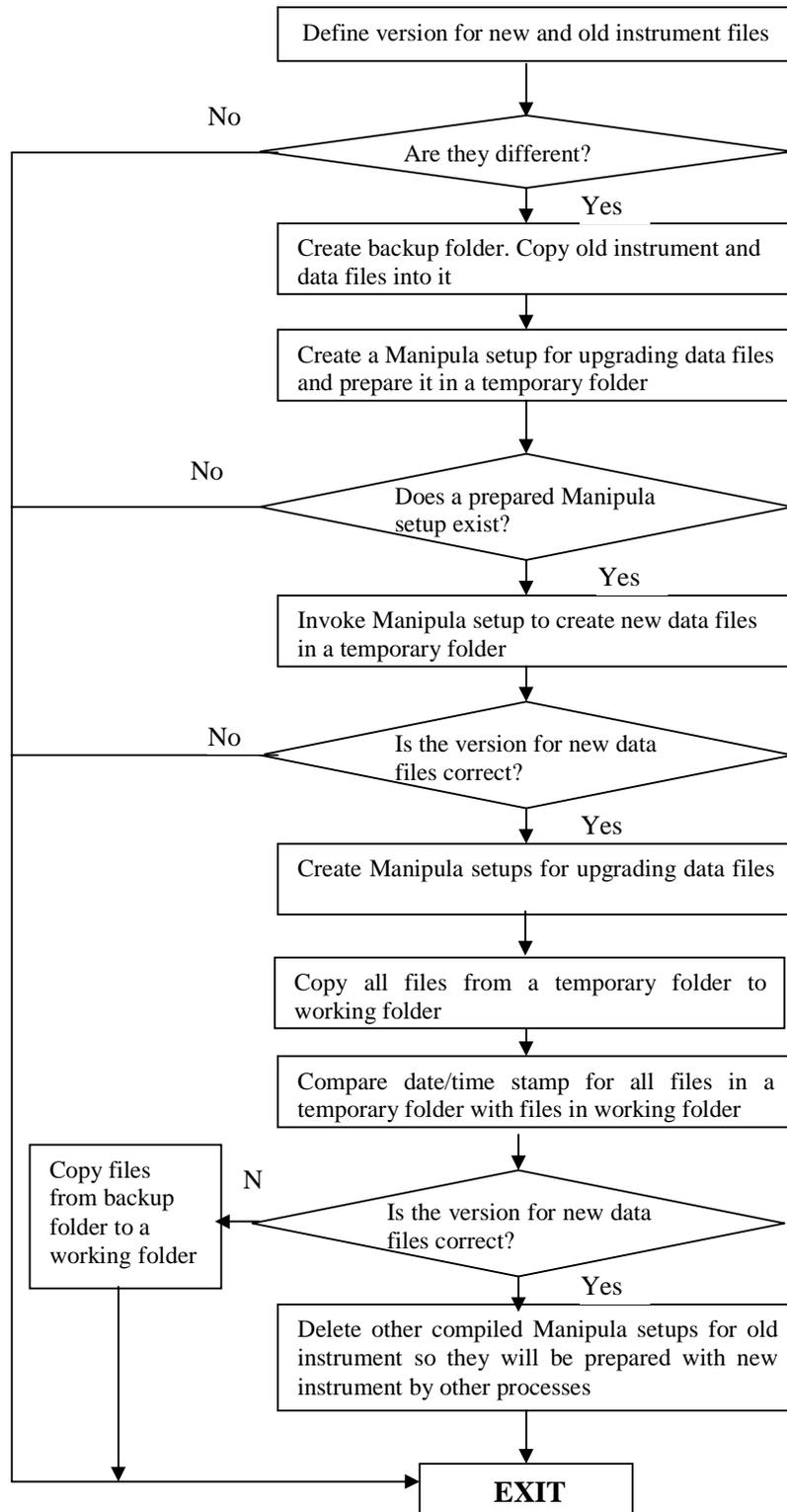
It may be necessary to upgrade a Blaise instrument once production has begun and the upgrade may alter the structure of the instrument, making any previously generated data incompatible with the upgrade. This methodology upgrades a Blaise instrument and data files without any interruption to data collection and can be

used for different projects that use Blaise data models, resulting in time and cost-effective project maintenance.

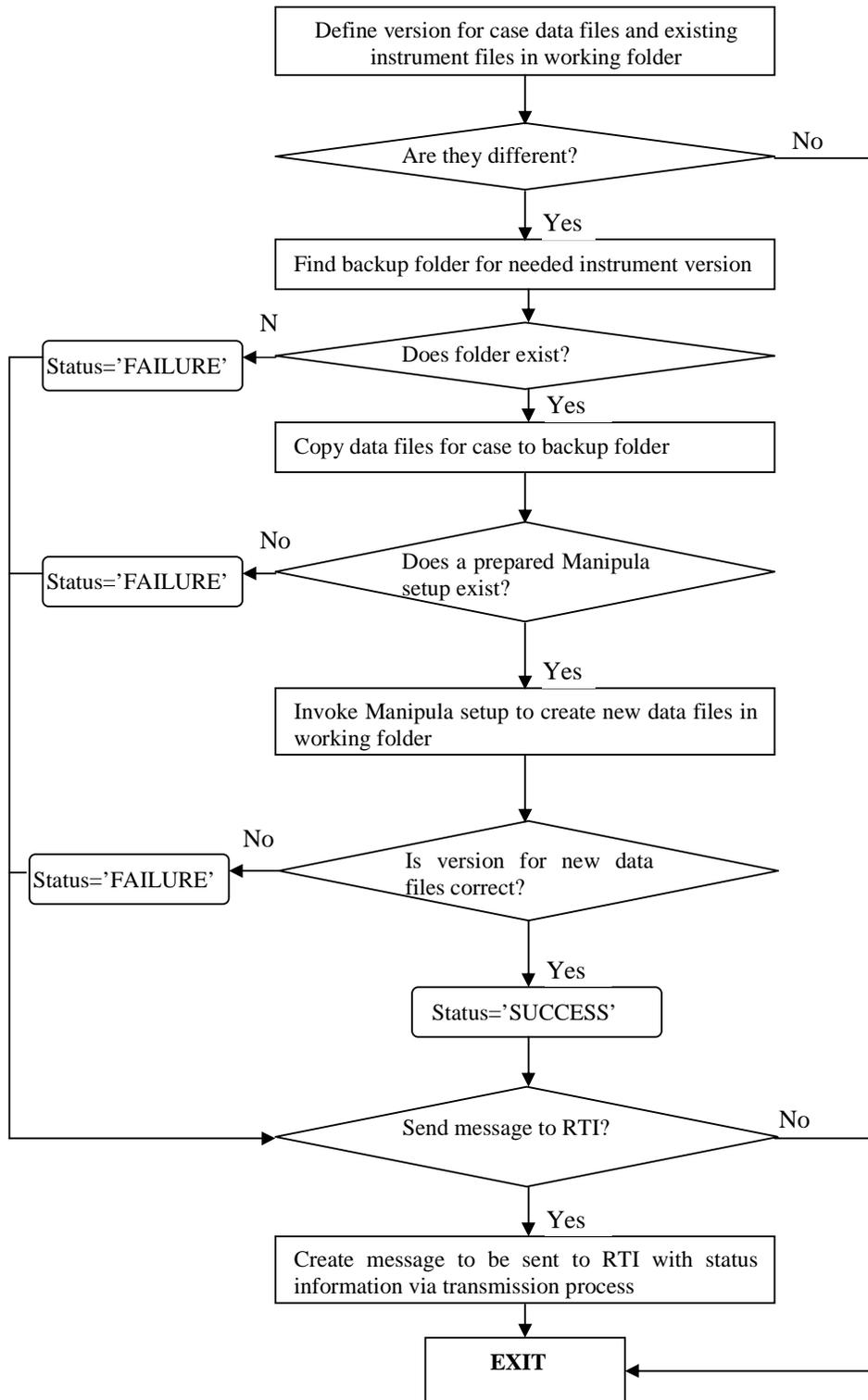
Our approach can even be used while an instrument is under development since if the instrument is upgraded and the structure has changed, deleting data files may require re-inputting data, which may be time-consuming.

## Appendix A.

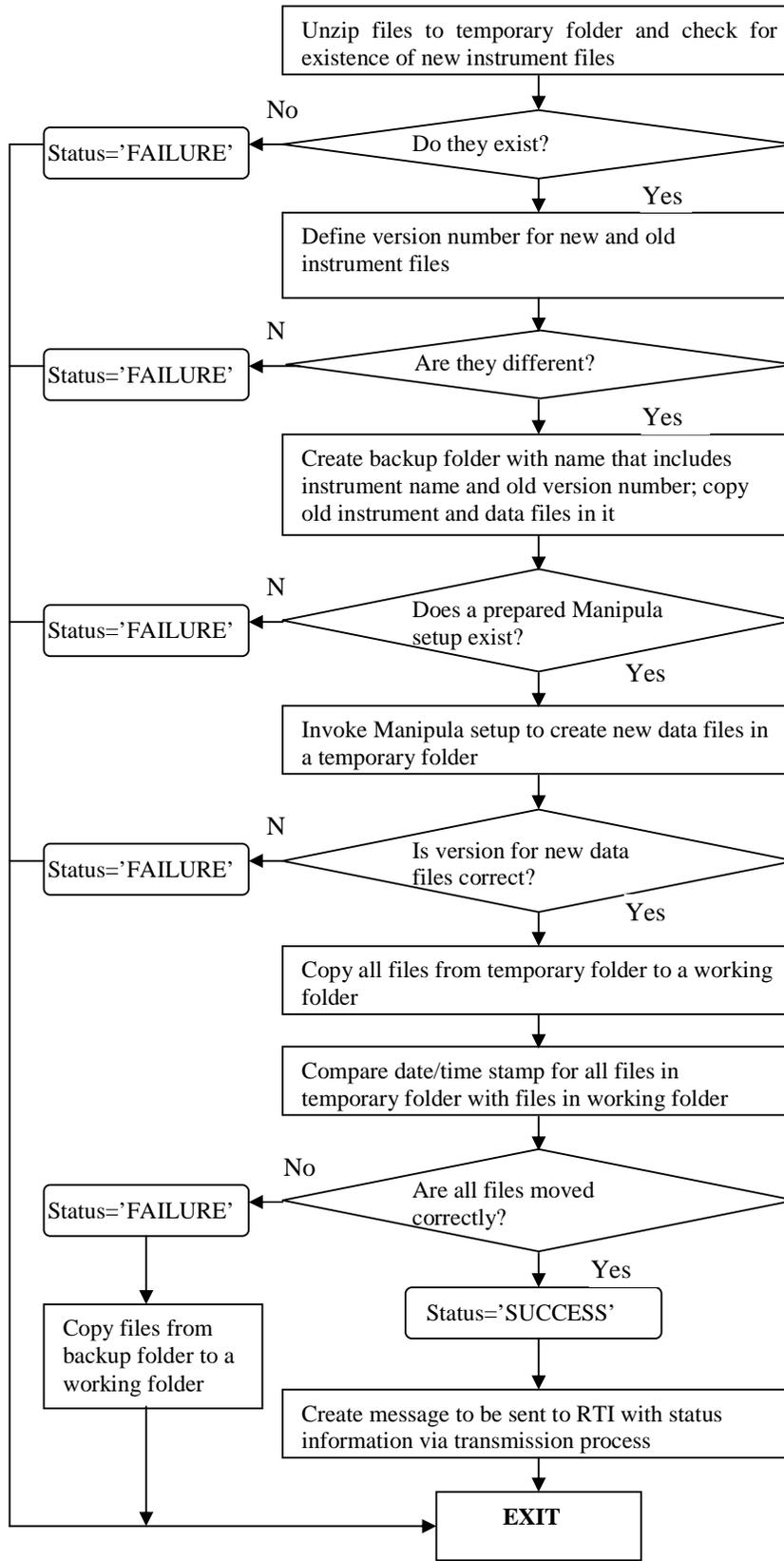
Flowchart 1 - Upgrade of Master database



**Flowchart 2 - Upgrade for individual case**



**Flowchart 3 - Upgrade database on field laptops**





## *Standards for design of user interfaces*

- **Screen Layout Standards at Statistics Finland**.....39  
*Vesa Kuusela, Survey Research Unit, Statistics Finland*
- **Developing and updating screen layout and design standards**.....45  
*Rebecca Gatward, Social Survey division, Office for National Statistics*
- **Developing an optimal screen layout for CAI**.....63  
*Fred Wensing, Jane Barresi and David Finlay, Australian Bureau of Statistics*
- **Screen Design Guidelines for Blaise Instruments**.....77  
*Sue Ellen Hansen & Karl Dinkelmann, Survey Research Center, University of Michigan*



# Screen Layout Standards at Statistics Finland

*Vesa Kuusela, Survey Research Unit, Statistics Finland*

## 1. Introduction

Probably all professional survey organizations apply a standardized interview method in their data collection. A question presented in an interview is supposed to initiate a measurement process and the given answer is the result of measurement. However, a survey question may be a very sensitive meter affected by various disturbances. For example, the result may vary considerably depending on how the question is presented. There is a wealth of examples how even small, almost unnoticeable, changes in the wording of a sentence may change its meaning. In surveys, where many interviewers ask questions, error variation due to different interviewing styles is always a serious threat to the credibility of surveys. The main purpose of the standardized interview method is to minimize inter-interviewer variation but it reduces intra-interviewer variation, as well.

The interviewing method is composed of several different topics but in brief, it means that questions are written in a specific format and interviewers should follow the format as instructed. One point here is that all questions should be asked exactly as they have been written. In other words, interviewers should follow strictly the wording of a question and use auxiliary information only on a specified style. An implication of the method is that interviewers have been trained to present – and understand – questions in similarly basing on this writing style. Another implication is that the questionnaires have to be written and formulated in such a style that interviewers may be expected to follow the specifications. One should bear in mind that an interview is an interactive discussion between two people and in that situation, a badly phrased question cannot be asked as such.

An essential precondition in carrying out standardized interviews is an explicitly defined writing style of the questions. That is the most important reason, why the layout and style of CAI questionnaires should be standardized. Reversibly, without a standard layout standardized interviewing cannot be expected. If the questions on different questionnaires are presented in varying styles, interviewers probably ask questions in varying styles, as well. That will increase inter-interviewer variation immediately and in the long run it will gradually lead to corruption of the method interviewers have been trained to apply.

Interviews' working ergonomics is another reason to standardize screen layouts. If several authors make questionnaires and all were allowed to make the screens as they like, probably screens would be composed basing on different principles. In practice that would mean that interviewers needed to learn different styles. Coping with them might become very demanding and in any case, it would be waste of interviewers' time – and nerves.

The layout and presentation standards also facilitate authors' work. When there is a good and easy to apply standard the authors don't have to design the screens anew every time they make a questionnaire. Sometimes a problem also is that authors do not have at their disposal similar laptops as interviewers have. Computer displays may differ considerably and a design, which looks good on one screen, may be nearly useless on another. It is difficult only to imagine what a certain design would look like on another screen but imagination is not needed if one can apply a ready-made and tested solution.

A minor, but not meaningless, reason to standardize question layout is the external image of the agency. Respondents only rarely see the screen of the computer but sometimes they do, and they form partly their opinion on what they see. Interviewers also embrace the idea of the organization and its style of working by the tools they are given.

The need for standardization of questionnaires is not new issue, but the need becomes more prominent with graphical user interfaces (GUI). The paper questionnaires have already been standardized for a long time and on text-based screens, partly the same standards could be applied as on paper questionnaires. Therefore, the need of standardization of text-based screens was much lesser than on graphical screens, and on the other hand, there were not many options in designing text-based screens. Graphical user interfaces have changed the situation dramatically. For example, Windows Blaise provides almost endless amount of possibilities for variations

In year 2000, at the same time when Statistics Finland acquired new laptops for interviewers, the entire CAI information system was converted to Windows and renovated comprehensively. Naturally, also Windows Blaise was taken in use. Soon the need to define new layout and presentations standards to comply with implication of the new environment became obvious. A group of few people with different backgrounds was collected to design a new standard.

## **2. Aims of design**

At Statistics Finland, a standardized interview method was adopted already long before CAI started, as probably in all professional survey organizations, and the interview method has been an inherent part of interviewer training all the time. In other words, the some presentation standards were already defined (and applied) when CAI was introduced, and interviewers were experienced in applying it. The standards were designed mainly for paper questionnaires but they could be followed reasonably well still when using text based screens and DOS software.

The new layout and presentation standards were outlined to compose rather a new design than only a refinement of the previous one. Graphical User Interfaces provide plenty of new features and possibilities, which could improve the functionality of the questionnaires when applied deliberately. That is, GUI also includes features, which could focus questions better and decrease inter-interviewer variation. An attempt to apply only the existing methods had been a waste of those possibilities. In addition, some of previous solutions were too laborious and cumbersome to accomplish in designing CAI questionnaires.

The one of the most important aims of the design work was that result should function well on the field interviewers' laptops. The LCD displays of laptops are more sensitive instruments than CRT displays of desktop computers. In addition, laptops are often used in settings that are more difficult to control. Three other general aims and preconditions were also set for the designers

1. to provide interviewers with easily comprehensible screens were important parts are easy to find
2. to keep amount of work reasonable required from the authors
3. to retain as much as reasonable the features of the older standard

A danger, when having all the possibilities that the Windows environment provides, is to get too exited of them, or to exaggerate the number of elements which need special attention. The resulting screen would be restless if it contains many different colors and several fonts, in addition to other graphical elements. On

an incoherent and tangled screen, even important parts may be difficult to notice or they may be even lost amidst the great number of different details.

On the other hand, some features, which Blaise and Windows environment provide, would bring about very informative and clear screen designs but some features had required too much detailed editing from authors. For example, if those answer categories, which should be read to respondent, were indicated using a different color than in question, individual attributes had to be attached to each answer category. In a long questionnaire, that would considerable increase editing.

As there already existed a standard, which interviewers were trained to apply, it was considered unnecessary to give up all the conventions. Therefore, those elements of the old standards were retained which were found good and working and which were easily applicable.

## 2.1 Prevailing conventions

The prevailing interview method at Statistics Finland already defined some rules for the compilation of questions. The style how a question is written includes, besides the question itself, also instructions for interviewers. They tell in which manner a question should be read to a respondent and how to continue in case the respondent does not understand the question. A major distinction was between presenting a factual and an attitudinal questions. The main rules say in brief:

- 1) A question text should always be read up to question mark;
- 2) First time a question should be read exactly as it is written;
- 3) If respondent does not understand the question,
  - a) in a question concerning a **fact**, interviewer should specify or explain it by his or her own words;
  - b) in question concerning **opinion, attitude, or knowledge**, interviewer should specify or explain the question only using instructions given on the screen.

These rules are obvious and their realization does not necessarily call for any graphical solutions. For example, attitudinal question was indicated with letter M (abbreviation from Finnish) after the question number. This kind of conventions could be copied from the prevailing standard.

## 2.2 Elements of a CAI question

Questions, which appear in a CAI questionnaire, have a relatively common basic structure. There are a limited number of different elements on a question screen, which should be discernible to direct interviewer's attention to them and hence to facilitate and to improve interviewer's performance.

**Question identification and attributes**, like question name or number, type of the question, attributes of the question, use of display card, and previous answer to the question in longitudinal surveys

**Question text and answer categories**. The elements of a question, which should be taken into consideration, are

- the part of the text which should always be read
- the part of the text, which is read conditionally to specify the question
- emphasized parts of the question text indicating key words
- indicator whether answer categories should be read or not

**Instructions for the interviewer** are needed for several different purposes:

- Complementary or specifying instructions for questions
- Definitions and concepts
- Instructions how answers should be marked
- Directive instructions
- Very important directive instructions

The CAI software takes automatically care of some issues, which were essential in paper questionnaires. For example, routing instructions, which were inherent elements in paper questionnaires, are not needed at all. Another example is a multiple answer question, which is handled by the system and does not need additional instructions. In addition, the default screen of DEP already has a lot of information, which is useful for interviewers.

### **3. Layout and presentation standards**

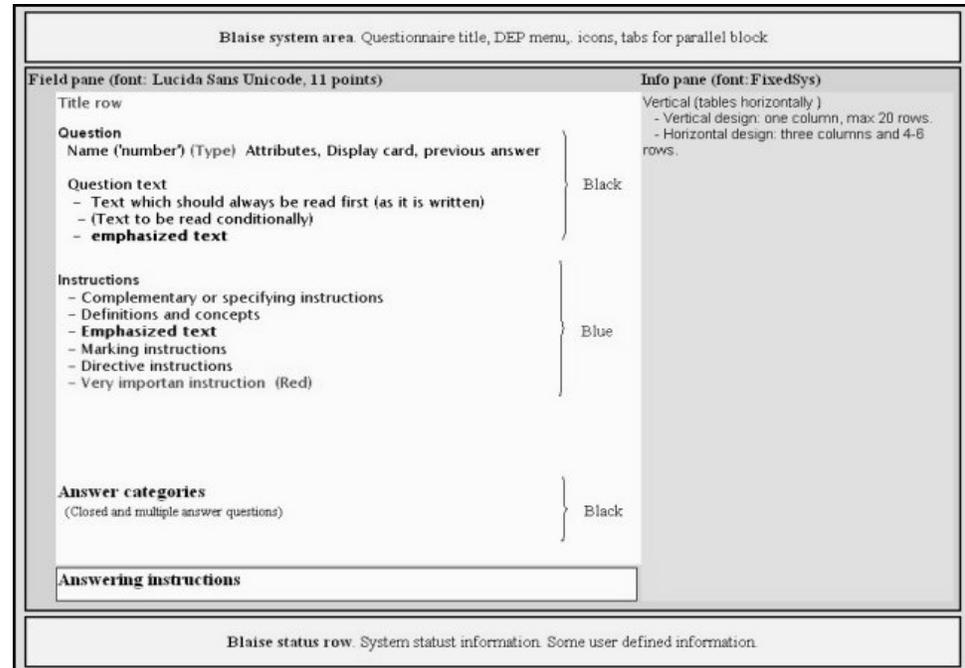
The standard, which was eventually agreed on, defined the division of DEP screen, menu items, writing style for questions and instructions, and the use of colors and fonts (see figure 1).

In most cases, the DEP screen is divided horizontally. Only on large tables, vertical division should be applied. The reason for horizontal division is that on a 1024\*768 pixel screen rows are very long if they go from margin to margin. Horizontal division makes rows shorter depending, of course, on how the division is done.

The reason why a horizontal division was selected was that a short text line takes easier a shape and is therefore easier to read. The horizontal division also provides more lines for the question. This arrangement allows for more possibilities for modifications in question writing and layout design.

All Blaise system texts visible in Dep were translated in Finnish as well as the Dep menu. The default Dep menu was slightly edited to meet the needs of interviewers and to comply with the information system. In practice that meant that only those menu items are shown which interviewers needed. In addition, some specific design features were taken in use. For example, parallel blocks are shown as tabs, and the line between question text and answer categories was left away.

Figure 1: Schematic presentation of a CAI screen designed for layout standard.



Only one font (Lucida Sans Unicode, 11 points) is used throughout the field pane, and another (Fixedsys) is used in the info pane. Emphasized words are written in bold in every instance, including questions, answer categories and instructions. Text that is read conditionally, e.g. if respondent did not understand the question, is surrounded by parentheses.

Whether answer categories should be read to the respondent or not is indicated by placing a question mark after the question text or after the last category to be read, or verbally in the instructions (as previously). Several other alternatives were tested, as well, but they all appeared to be too laborious to carry out in everyday authoring practice.

Blaise default color scheme is used except in Field pane texts. Colors make the distinction between the question text and instructions. Both the question and related answer categories are shown in black.

All instructions, wherever they appear on the screen, are blue. Although there are many different types of instructions, they were not discriminated by any means. The informative value of discriminating between different instruction types was reflected against the graphical image that had resulted. Many different colors would have been too distracting and it was given up. Different fonts might have been possible but that was not considered beneficial enough because of increased need of editing.

However, there is one exception to the previous rule. Red is reserved (only) for very important instruction, and to indicate whether the question is an attitudinal one. Red color should be used very sparingly to preserve its effect.

#### 4. Discussion

It is not sufficient only to define screen layout standards. They also have to be widely known and accepted to become a part of everyday questionnaire authoring. The two critical groups here are those who make the questionnaires and

researchers. Both groups have to understand the importance of everyone applying the same standards. The authors of the questionnaires have to know the technical details and how to apply them – and take them as a natural part of questionnaire design. Researchers, on the other hand, have to understand the importance of the standards and learn to require their use in every questionnaire.

At Statistics Finland, the defined screen layout standards were published on a paper printout, which was presented in a meeting to all those people who deal with CAI questionnaires in a way or another. In addition, a ready-made Modelib was made available for everyone.

The defined standards have been in use for some time in all questionnaires of Statistics Finland. The feedback from interviewers has been mainly positive. In addition, authors workload has remained bearable. Hence, the result may be regarded to satisfy the requirements, which were set. However, the defined standard should not be regarded as a final end result. Rather question is of a continuous process.

The Blaise Modelib editor proved to be a surprisingly cumbersome tool and its behavior remained somewhat vague. Especially, the definition of the place and size of screen elements was difficult. During the designing and testing of different screen layouts a need for a WYSIWYG type modelib editor came up many times.

The screen layout is not the only part of a questionnaire, which needs to be standardized. Some structural and syntactical standards are equally important. For example, then an adequate naming convention of questions and blocks helps navigation within a large questionnaire. These kinds on standards are not as visible as the screen layout but also they could make interviewer performance more accurate and easier.

Finally, one should notice that layout and presentation standards are only a small, but important, factor influencing the survey quality. Concepts, question wording and question order have a much deeper and wider effect on survey instruments and survey process (not to speak about sampling and non-response). However, adequate standards make good instruments work better – and poorly defined standards make inferior instruments work worse.

# Developing and updating screen layout and design standards

*By Rebecca Gatward, Social Survey division, Office for National Statistics*

## 1. Introduction

The importance of computer assisted interviewing (CAI) screen layout and design on interviewers performance in collecting quality data is now widely accepted and documented. The screen layout and design should enable interviewers to navigate quickly and easily around a questionnaire and provide them with an immediate and clear recognition of the task they need to perform.

In the mid-1990s Social Survey Division (SSD) of the Office for National Statistics (ONS) first developed a set of screen standards for Blaise for the organisation. A standard configuration file, based on these standards, was produced for use by all questionnaire developers. This standard configuration file was revised when ONS moved to a windows version of Blaise (B4W).

Screen layout and design standards need to be reviewed regularly both to take advantage of any advances in Blaise and to provide standards for new computer assisted interviewing (CAI) techniques, such as Audio-CASI. ONS are currently carrying out a review of their screen standards.

The aim of this paper is to provide a practical documentary of each stage of this process; it is not to describe screen design or layout guidelines that have not been suggested elsewhere. This paper may be helpful to other organisations that are about to start or are in the process of updating their own screen standards.

Throughout this process we have drawn on the work of other organisations. We have found the research carried out by Survey Research Center, University of Michigan particularly valuable.

## 2. Background

The collection of good quality data should be the key objective of any survey organisation. Interviewers play a major role in helping to meet this objective. Their primary goal is to collect accurate information, quickly and easily and to enter that information into the Blaise questionnaire. Couper (1994) remarks that the potential that interviewers have to impact the data collection process (in terms of both costs and errors), makes it imperative to design systems to maximise interviewer efficiency and minimise errors.

Blaise developers therefore have a responsibility to provide the tools to enable interviewers to meet their goal. It is essential that we help the interviewer in any way possible to do their job well. Interviewers need to be kept happy, the questionnaire should be a 'pleasing tool' to work with and should not hinder them in anyway. This is especially important at a time when maintaining survey response is a particular problem. Kuusela (1995) suggests that the user interface should be transparent so that an interviewer does not notice she or he is using it. This means that using it must be easy.

### 3. Importance of screen standards

Most organisations who develop CAI instruments have now developed their own set of screen standards or guidelines. The scope or main objectives of these standards may vary across organisations. For ONS, the objectives of screen standards are as follows.

- Provide a uniform interface. Interviewers can expect that instructions will always be in the same location on the screen. This means that interviewers can concentrate on interviewing, building a rapport and maintaining eye contact with the respondent.
- Simplify the programming task for developers because decisions about screen design have already been made. This is especially important in ONS where designers of questionnaires are researchers working on projects rather than a dedicated team of programmers.
- Make checking the questionnaire easier.
- Ensure that optimum standards (i.e. tried and tested) are being used.

### 4. Procedure for updating screen layout and design standards

The remainder of this paper will provide a documentary of the steps that have already been taken, or we plan to take, at ONS to review and update our current screen standards.

#### General principles

Before starting this process we developed the following list of key principles. We kept these principles in mind throughout the process.

- Use Blaise defaults and standardise as much as possible within the standard configuration file.
- Keep the number of standards to a minimum and the standards themselves as simple as possible. The implementation of standards should not increase the time taken to develop a Blaise questionnaire. It is desirable that the time should be reduced.
- The standards should be intuitive or easily memorable to enable new interviewers to become accustomed to the new conventions.
- Do not use Blaise features just because they are available.
- Any standard should be tested within ONS before it is introduced.
- Ensure that interviewers who are colour blind can benefit from the screen design standards. Rigden (1999) suggests that in good design, colour should never be the primary cue for information.

#### Stage 1: Review Blaise usage and new capabilities of Blaise

The aim of this first stage was to identify the following gaps and opportunities.

- CAI techniques that are now being used in ONS for which standards have not yet been documented.
- Existing question types or interviewing methods for which there are no standards.

- New Blaise capabilities that could enhance the current screen layout or design, such as symbols and the more flexible use of coloured text and font types.
- Changes to default settings between versions of Blaise, for example the default setting for a 'hard space' changed between Blaise 4.2 and 4.3.
- Facilities available in Blaise which are not currently being used because a decision has been actively made not to or time has not been available to investigate their use. Some examples that ONS identified were:
  - allowing interviewers to use a mouse to move around the questionnaire,
  - a more systematic use of 'NEWPAGE' to separate questions, and
  - the use of more meaningful extended field descriptions which could be used on screen and in data files rather than field names.

### Stage 2: Literature review and experience of other organisations

The next step was to gather information about optimum screen design from variety of sources. We reviewed recent papers and articles and drew together information gathered from workshops and other organisations. One of the most productive tasks for us was to review standards set by other organisations. Using other people's standards is efficient; it is not necessary to spend time discovering for yourself what has already been discovered by others. If it has been tried and tested by others and it meets your needs then use it.

### Stage 3: Feedback from users

For this particular review, we collected feedback from the field managers, who have daily contact with interviewers. The feedback they were able to provide was fairly limited. We feel a more productive method would be to seek feedback directly from interviewers on an ongoing basis.

With this in mind, we would suggest that feedback is actively encouraged during the first few months after implementation of new standards. After this point interviewers will probably have got used to how the screen looks and become too familiar with the questionnaire(s) to be sufficiently annoyed or hindered to provide feedback. For example, on the first few interviews, an interviewer may have missed an onscreen instruction; this problem might be forgotten by the time that they have reached the tenth interview.

### Stage 4: Proposed changes to screen layout and design standards

The next stage was to develop a list of proposed changes to the current screen standards. Examples of some of our proposed additions and changes to the existing standards are detailed in the following section.

#### Examples of new additional standards

- Concurrent interviewing

Concurrent interviewing is carried out on three of the ONS continuous surveys and enables the interviewer to ask a series of questions to more than one person at a time. Interviewers can potentially be interviewing up to the maximum number of respondents in a household, which is 14 for these surveys. Such large households are rare; it is quite common, however, to need to interview 3 or 4 respondents in a household. Interviewers switch between addressing sets of questions to one

respondent and then another. Screen standards for this technique have evolved since the move to CAI but have never been documented or reviewed.

Feedback from interviewers suggests that they have a problem knowing exactly when they should switch between respondents while carrying out a concurrent interview. The switch is, of course, programmed in Blaise, but the interviewer gets no warning to address a different person in the real world except a change in the respondent's name at the top of the screen. Interviewers need a more eye-catching method to tell them they should be turning to the next respondent.

The proposed solution is to change the colour of the question text for each person in the household. For example, the question text for the first respondent would be black, the second red and the third blue, the fourth would revert to black. This solution has already been used successfully on one continuous survey carried out by ONS. Interviewer feedback suggests that this does provide them with the immediate indication they need to prompt them to move to the next person in the household. To ensure that this standard is also helpful for interviewers who are colour blind we also change both the colour and font type. The technique of changing font type will be tested before it is adopted as standard. (*Appendix A*)

- Text Substitution (Text fills)

There are currently no documented standards to specify the extent to which text fills can be used in ONS Blaise questionnaires. However, the developing ONS standard is only to use text fills where a clear case can be made that the pre-CAI method of relying on interviewers to remember information or to provide the appropriate variations can be improved on. For example, we think there is a clear case for displaying information such as the respondent's name on every screen; to carry forward responses from previous questions if that is needed; and to display dates in question text. Unlike many others, we see no case for routinely changing pronouns to agree with the respondent's gender. We only make very limited use of text fills to vary verb tenses. We would do this only for special circumstances where (a) it is essential to a distinction that is being measured and (b) would be ambiguous which tense should be used despite the context of preceding questions.

In our view, there are these arguments against the extensive use of text fills.

- Using text fills increases the complexity of the Blaise programme and so increases the risk of errors (and so development and testing time) without a sufficient benefit.
- What appears in any document produced via electronic documentation software is the text fill rather than the text that the fill represents. This means that the main users of these versions of the questionnaire (customers and interviewers) would not be able to understand the document unless it was manually edited.
- ONS interviewers are trained to be flexible and to deal with text fills. We think it is important to allow interviewers to be actively engaged in the interview and not to be encouraged to slip into a robotic role.

Other organisations do make more use of text fills and some have 'text fill libraries'. We feel that further research is required to justify using text fills more extensively (for example, does the use of text fills improve the quality of the data?). We hope to carry out such a project in the future, hopefully before the next revision of the standards.

- Audio-CASI

Audio-CASI was first used in ONS in March 2001. The current ONS screen standards do not include any guidance on screen layout and design when using audio-CASI.

Screen design is particularly important when using audio-CASI because the end user of the Blaise questionnaire is the respondent. Respondents will have a range of experience using computers so the questionnaire should be easy to understand and complete. Poor design should not provide an excuse to respondents to refuse.

The following standards are based on work carried out before audio-CASI was first used in production in ONS.

#### *Screen layout*

The following guidelines were followed when setting the screen layout standards:

- the screen should be kept uncluttered to avoid distracting the respondent.
- question text should not be displayed on screen. It is more important for the respondent to concentrate on listening to the question (this is especially important for young people who had reading difficulties).
- response categories can be displayed in the answer list section of the infopane in the standard way.

#### *Flexibility to switch interviewing technique*

In an audio-CASI interview the option should be available for the questionnaire to be completed using CASI or as a face to face interview (e.g. if the respondent preferred to do it that way or got into difficulties). To achieve this, we need a simple way to switch between the modes. We have found that using the colour contrast of foreground and background colours provides a simple mechanism. If audio-CASI is to be used the text of the question should be set to the same colour as the screen background. This has the effect of making it invisible, so that the respondent is not distracted from listening to the text (as noted above). For usual CAI or CASI the question text should be set to a contrasting colour so it can be read. More detailed instructions of how this is programmed in the Blaise questionnaire will be included in the ONS standards. A copy is also provided as an appendix to this paper. (*Appendix B*)

Even if the questionnaire is completed using audio-CASI it should still be possible for interviewers to be able to recognise the respondent's whereabouts in the questionnaire. A small identifier should be displayed. It is obviously important not to use an identifier that could distract the respondent such as the question number or "question 1 of 250". The questionnaire variable name may be the most appropriate identifier if it is not itself a mystifying mnemonic.

#### *Labelled keys*

We find it helps if the main navigational keys are colour coded, e.g. using paper stickers. Our standard is that the <ENTER> key is labelled white and the <F10> key, which we assign as the repeat key, is blue. Blue and white have been assigned to the two navigational keys because they are the least likely to be confused by people who are colour blind.

## Examples of proposed changes to ONS standards

The following are examples of revisions we are currently proposing to the ONS standards following a review of some 5 years' experience and information gathered from other sources.

- *Question text format*
  - default font size for question text was increased from 10pt to 12 pt, to improve readability
  - any text which is read aloud by an interviewer should be immediately identifiable: we have opted for bold text,
  - likewise an optional text should be immediately identifiable but should not interfere with the compulsory text: based on work by other organisations our choice is grey text.

(Appendix C )

- *Response categories*

Lists of response categories should be kept to one column for up to eight categories; otherwise the categories should be balanced across the screen to avoid the interviewer missing categories.
- *Interviewer instructions*

Interviewer instructions are currently identified using; 'INTERVIEWER:' followed by the instruction in capitals. It is generally accepted that reading text displayed in mixed upper and lower case letters is faster and easier (Tickner 1963). Hill (1999) explains that this increase in speed is due to a person's tendency to recognise the shape of the word as opposed to each individual letter in a word. When text is in upper case letters, it takes away the characteristic shape of words. The reader is forced to identify the individual letters of each word, which slows the reader down.

Now that an alternative method of highlighting instructions (i.e. colour) is available, we were particularly anxious to change our current standard for interviewer instructions.

Interviewer instructions will now be identified using the following symbol ⓘ ('i' in Webdings pt 16) in red text. The instruction text should be written using 11pt in Comic sans serif (red).

<ul style="list-style-type: none"><li>• Current standard: INTERVIEWER: ENTER TO THE NEAREST £1 (AFTER HOUSING BENEFIT)</li><li>• Proposed standard (in red text): ⓘ Enter to the nearest £1 (after housing benefit)</li></ul>
---

- *Standard action instructions*

Instructions which are telling interviewers to carry out a task, rather than providing them with additional instructions, will be formatted in the same way as interviewer instructions (i.e. red text, mixed case and in font type, comic sans serif). The key action only will be written in capitals. This format has been chosen because we want to draw the interviewers attention to the action that they should take. Some examples:

CODE all that apply  
to SAVE enter [Alt+S]  
to EDIT enter [insert]

- *Symbols*

In the current standards show cards and interviewer instructions are identified by 'SHOW CARD' and 'INTERVIEWER INSTRUCTION'. Opinion questions are currently identified by [\*]. As mentioned previously, we were anxious to avoid using capital letters wherever possible. Symbols take up less room on screen and are more immediately identifiable than capital letters.

The following symbols will be used to represent a show card, interviewer instruction and an opinion question.

Show card		(webding, 20pt, red)
Interviewer instruction		(webding, 18pt, red)
Opinion question		(monotype sorts, 14pt, red)

*(Appendix D)*

- *Colour Scheme*

The colours used for each part of the screen will be changed. The infopane will be a pale yellow, the question information section a pale turquoise and the fieldpane a pale blue. The main reason for this change is to create more contrast between the formpane background and the fieldname text. The current standard is to set the formpane background colour to royal blue and interviewers have commented that they find it difficult to read the fieldnames.  
*(Appendix E)*

## Stage 7: Testing

One of our key principles at ONS is that any proposed screen layout and design changes should be tested within ONS before being introduced.

Before testing started we created a test questionnaire which included examples of all question types. The standard Modelib file was also revised.

Testing of the new screen layout and design standards will be carried out in the following three main stages (stages one and two have been completed):

- Expert review

A small group of people who have a good knowledge of Blaise and Field Co-ordinator (an 'expert user') in ONS were asked to comment on the new screen standards. The screen layouts and designs were amended to incorporate comments their comments and the test questionnaire was prepared for the second stage of testing.

- 'Walkthrough tests'

This stage involved a small group of users (Field managers, experienced and inexperienced interviewers (Face to Face and Telephone interviewers). The testing is taking place in three main stages:

#### *1. Problems with current screen:*

Before revealing the proposed changes to the screen layouts, I asked the interviewers whether the current screen layout and designs caused them problems. Interviewers provided the following comments:

*'it is sometime difficult to distinguish between instructions and question text, it would be nice to have them in different colours'*

*'the space between question and instructions is important'*

*'some newer interviewers read through the question mark, or stop too soon'*  
(the ONS standards is for interviewers to read to the question mark)

*'lights and sun glare on the white screen, but get used to it'*

*'the empty questions at the bottom of the screen are sometimes daunting'*

*'I have trouble reading capitals and text is too small, especially the question names at the bottom of the screen'.*

*'I like the fact that all parts of the screen are different colours'.*

#### *2. Symbols*

Interviewers were shown the range of symbols available in the Webding font type. They were asked to say which one they would choose to represent an interviewer instruction. All but one interviewer picked the 'ⓘ' symbol. Some comments were *'it's simple', 'striking', 'others are too childish', 'not too complicated', 'I for information' and 'it stands out'.*

As the majority of the interviewers were telephone interviewers I did not ask them to pick a symbol for a showcard. However, as they were going through the questionnaire a couple of the interviewers commented that they felt the '☞' was appropriate to represent a showcard. One remarked that it said 'look at' to them.

### 3. Screen layout and design changes

The next stage was to as the interviewers go through the test questionnaire, they were encouraged to 'think aloud' and provide feedback as they did so.

Interviewers liked the new colour scheme, they found it: '*plain*', '*clear*', '*not glaring*', '*soft on the eye*' and '*more restful*'. All interviewers agreed that using red text for the interviewer instructions made them stand out more (*'it hits you'*) and they were easier to read. They were also positive about the use of bold for the question text, they agreed that it was a clear way to highlight text that should be read. One interviewer noticed that the question text was not extending all across the screen and felt it was an improvement.

Some general comments about the overall appearance of the screen were that they '*didn't feel bogged down by it*' and another said they thought it was '*user friendly*'.

- Final large scale test

This final stage of testing has not yet been started. We plan that this stage will involve experienced and inexperienced telephone and face to face interviewers. We will also include some interviewers with visual impairments and colour blindness. The testing will be carried out in a variety of environmental conditions (for example, inside and outside). Outcomes from this test will be evaluated and final changes made to the document detailing the proposed standards.

#### Stage 8: Finalise screen standards document

Once the changes to the standards have been agreed the next task will be to update the current standards document. The changes and additions will need to be disseminated to other Blaise developers within ONS. This will probably be done by holding a number of small workshops. The document and a summary of changes will also be circulated to all Blaise users in ONS.

#### Stage 9: Implementation of the new and revised standards

The final stage of the process will be implementation. The following steps will be taken to help smooth the transition from the old to new standards.

##### Tools for developers

At ONS we use standard blocks. These are comprised of harmonised modules of questions or standard code (for e.g. a block for the interviewer to record details about the administration of the interview, such as outcome and number of calls ). Developers build their questionnaires using these standard blocks and blocks of survey specific questions. These standard blocks are revised on an annual basis and are ready for developers to include them on questionnaires for the beginning of April (although essential minor changes are made through out the year). The new standards will be incorporated into these blocks.

Some templates will also be developed for each type of question and interviewer instruction. These will be available to all developers so they can cut and paste them into their questionnaires and add the question text and response categories. The syntax to format the question in the correct way will already be there.

These templates may be particularly useful when developers are familiarising themselves with the changes. For example a template for a 'Running Prompt' question:

```
Fieldname "@/@/@b INSERT QUESTION TEXT HERE ...@b
          @/@/@rRunning prompt@"
          : (LABEL 1   "@b RESPONSE 1,@b",
            LABEL 2   "@b RESPONSE 1,@b",
            LABEL 3   "@b RESPONSE 1?@b")
```

#### Timing of implementation

Ideally we would like to implement the standards at the same time for all surveys; this probably will not be possible. Most continuous ONS survey run to an annual cycle survey year beginning at the start of April (to match the UK financial year). Adhoc surveys can begin at any point in the year. As many surveys as possible will adopt the new standards at the beginning of the 2004 survey year (April). At this point all surveys will have moved to Blaise 4.6 and the new standards can be incorporated into the updated standards blocks.

#### Guidelines for interviewers and trainers

Before the new standards are introduced we will also produce some guidelines to explain the changes to interviewers. We will also write an instruction document for trainers of new interviewers.

## 5. For the future

Over the next few months we will need to take a closer look at Blaise 4.6 and identify any features that may further enhance screen layout or design. We will then move to the final stage of testing.

Once the changes have been finalised and implemented we will then start thinking about the future. One of the next major jobs for ONS will be to develop screen standards for web surveys. We may also investigate the necessity of tailoring CASI screen standards to the type of respondents, for example, older people, young people and children. Children are generally more experienced computer users than older people and so require fewer navigational aids than older people, but perhaps need a more visually exciting screen design to keep their interest.

Finally, based on our experience of updating and reviewing the screen layout and design standards, we would suggest that the process of reviewing and updating screen standards should be ongoing. We would also recommend that someone within an organisation be given responsibility for screen standards. Amongst other tasks they could keep up to date with current thinking on optimum screen standards and encourage and act upon feedback from interviewers.

## 6. References

Couper, M.P. (1994), 'What can CAI learn from HCI?' *Discussion paper presented at the COPAFS Seminar on New Directions in Statistical Methodology, June 1994*

Hansen, S. E. (2000) 'Designing 'Usable' CAI instruments' *Workshop at The International Field Directors and Technologies Conference, Clearwater Beach, Florida*

Hill, A.L., and Scharff, L.F.V. (1999), 'Readability of websites with various foreground/Background colour combinations, font types and word styles' Stephen F. Austin State University, Department of Psychology, Nacagdoches, TX. unpublished paper.

Kuusela, V. (1995), 'Interviewer interface of the CAPI system of Statistics Finland' *In Proceedings from the Third International Blaise Users' Conference*

Rigden, C. (1999) 'The eye of the beholder - Designing for Colour blind users' *British Telecommunications Engineering, Vol. 17, January 1999*

Tinker, M. (1963) 'Legibility of print' *IOWA State University Press, IOWA*

## 7. Appendices

### Appendix A

- First person in household - question text is black

Blaise Data Entry - \\\SURVEY1\SQL\Projects\Usability\concurrent\mod0304

Forms Answer Navigate Help

MOD0201A General\_Blaise\_Help

**GEORGE**

**What is your date of birth?**

**i** if day not given... ENTER 15  
if month not given... ENTER 6

Enter a valid date

	Name	Sex	Birth	Agelf	DVage	MarStat	LiveWith	Hhdr
QHComp[1]	GEORGE	1	01/06/1967		35	2		1
QHComp[2]	MILDRED	2	04/07/1964		38	2		5
QHComp[3]	VIOLET	1	07/08/1999		3			
QHComp[4]								
QHComp[5]								
QHComp[6]								
QHComp[7]								

4/5 Navigate MOD0201A (QHComp.QHComp[1].Birth)

- Second person in household - question text is blue

Blaise Data Entry - \\\SURVEY1\SQL\Projects\Usability\concurrent\mod0304

Forms Answer Navigate Help

MOD0201A General\_Blaise\_Help

**MILDRED**

**What is your date of birth?**

**i** if day not given... ENTER 15  
if month not given... ENTER 6

Enter a valid date

	Name	Sex	Birth	Agelf	DVage	MarStat	LiveWith	Hhdr
QHComp[1]	GEORGE	1	01/06/1967		35	2		1
QHComp[2]	MILDRED	2	04/07/1964		38	2		5
QHComp[3]	VIOLET	1	07/08/1999		3			
QHComp[4]								
QHComp[5]								
QHComp[6]								
QHComp[7]								

4/5 Navigate MOD0201A (QHComp.QHComp[2].Birth)

- Third person in household - question text is green

Blaise Data Entry - \\LSURVEY1\NSQA\Projects\Usability\concurrent\mod0304

Forms Answer Navigate Help

MDD0201A General\_Blaise\_Help

VIOLET

What is your date of birth?

**i** if day not given... ENTER 15  
if month not given... ENTER 6

Enter a valid date

	Name	Sex	Birth	Agelf	DVage	MarStat	LiveWith	Hhdr
QHComp[1]	GEORGE	1	01/06/1967		35	2		1
QHComp[2]	MILDRED	2	04/07/1964		38	2		5
QHComp[3]	VIOLET	1	07/08/1998		3			
QHComp[4]								
QHComp[5]								
QHComp[6]								
QHComp[7]								

4/5 Navigate MDD0201A (QHComp.QHComp[3].Birth)

## Appendix B

The colour switch variable was set up as a parameter in the top-level block of the self-completion section of the questionnaire. Appropriate colours are assigned to 'W' and 'P' in the modelib editor. The parameter (PColour1 and PColour2 – see example below) was switched between '@W' or '@P' depending on the mode of completion.

```
C3G3  "^PColour1  Have you ever used glue, gas or
      solvents?@/
      PRESS 1 for NO, 2 for YES @/
      PRESS the WHITE key to continue@/ ^PColour1
      ^PColour2@/ Question G3 ^PColour2@/"
      MML "SOUND(C3G3.WAV)
          SOUND(Delay.WAV)
          SOUND(NoYes.WAV)
          SOUND(White.WAV) "
      : NY
```

## Appendix C

- An example of the proposed format for displaying optional text

Blaise Data Entry - \\LSURVEY1\SQA\Projects\Usability\Test questionnaire\Blaise42\SQA101201

Forms Answer Navigate Help

SQA101201 | General\_Blaise\_Help

After the baby was born did you have any help around the house or with the baby, from ...

... your husband?

1. Yes  
 2. No

XWhyStop		GenHlth	1	Good
LastQual2	18	BabyHelp	1	
QualDesc	gcse	HelpDisp	1	Continue
LastQual4	18	HelpHus	1	Yes
VehWork	2	HelpRel	1	Yes
HowFeed	2	HelpFind	1	Yes
NumAL	1	HelpHome	1	Yes
	No			
	breast			
	OneAL			

4/7 | Navigate | SQA101201 (HelpHus)

Blaise Data Entry - \\LSURVEY1\SQA\Projects\Usability\Test questionnaire\Blaise42\SQA101201

Forms Answer Navigate Help

SQA101201 | General\_Blaise\_Help

(After the baby was born did you have any help around the house or with the baby, from ...)

... other relatives?

1. Yes  
 2. No

XWhyStop		GenHlth	1	Good
LastQual2	18	BabyHelp	1	
QualDesc	gcse	HelpDisp	1	Continue
LastQual4	18	HelpHus	1	Yes
VehWork	2	HelpRel	1	Yes
HowFeed	2	HelpFind	1	Yes
NumAL	1	HelpHome	1	Yes
	No			
	breast			
	OneAL			

4/7 | Navigate | SQA101201 (HelpRel)

## Appendix D

- An example of an opinion question

Blaise Data Entry - \LSURVEY1\SQLA\Projects\Usability\Test questionnaire\Blaise42\SQLA101201

Forms Answer Navigate Help

SQA101201 | General\_Blaise\_Help

(Name) HELP [F9]

★  
Over the last month would you say your health has on the whole been good, fairly good, or not good?

1. Good  
 2. Fairly Good  
 3. Not Good

XWhyStop		Genhlth	1	Good
LastQual2	18	BabyHelp	1	
QualDesc	gcse	HelpDisp	1	Continue
LastQual4	18	HelpHus	1	Yes
VehWork	2	HelpRel	1	Yes
HowFeed	1	HelpFind	1	Yes
NumAL	1	HelpHome	1	Yes

4/7 | Navigate | SQA101201 (Genhlth)

- An example of a question with a showcard

Blaise Data Entry - \LSURVEY1\SQLA\Projects\Usability\Test questionnaire\Blaise42\SQLA101201

Forms Answer Navigate Help

SQA101201 | General\_Blaise\_Help

12

Are/were you receiving any of these state benefits in your own right that is, where you are the named recipient?

CODE all that apply

<input type="checkbox"/> 1. Child Benefit	<input type="checkbox"/> 6. War disablement pension or War Widow's Pension (and any related allowances)
<input type="checkbox"/> 2. Guardian's Allowance	<input type="checkbox"/> 7. Severe disablement allowance
<input type="checkbox"/> 3. Invalid Care Allowance	<input type="checkbox"/> 8. None of these
<input type="checkbox"/> 4. Retirement pension (National Insurance), or Old Person's pension	
<input type="checkbox"/> 5. Widow's Pension, Bereavement Allowance or Widowed Parent's (formerly Widowed Mother's) Allowance	

Enter at most 6 values

Ben1Q

6/9 | Navigate | SQA101201 (Ben1Q[1])

## Appendix E

- Current colour scheme

Blaise Data Entry - \\LSURVEY1\SQA\Projects\Usability\Test questionnaire\Blaise42\SQAold

Forms Answer Navigate Help

Is your degree...

CODE FIRST THAT APPLIES

1. a higher degree (including PGCE)?  3. other (eg graduate member of a professional institute or chartered accountant)?

2. a first degree?  4. Don't know (DO NOT PROMPT)

HelpDisp  Continue Likelce2

HelpHus  Yes

HelpRel  Yes

HelpFmd  Yes

HelpHome  Yes

Degree  Higher

Ben1Q

Method1

Method2

Likelce1

5/8 Navigate SQAold

- Proposed new colour scheme

Blaise Data Entry - \\LSURVEY1\SQA\Projects\Usability\Test questionnaire\Blaise42\SQA101201

Forms Answer Navigate Help

SQA101201 General\_Blaise\_Help

Is your degree...

CODE first that applies

1. a higher degree (including PGCE)?

2. a first degree?

3. other (e.g. graduate member of a professional institute or chartered accountant)?

4. Don't know (Spontaneous only)

Degree  Higher

5/9 Navigate SQA101201 (Degree)



# Developing an optimal screen layout for CAI

*Fred Wensing, Jane Barresi and David Finlay, Australian Bureau of Statistics*

## 1. Introduction

This paper describes the development of an optimal screen layout for Blaise to be used for Computer Assisted Interviewing (CAI) at the Australian Bureau of Statistics (ABS).

In making the transition from paper-based interviewing to computer assisted interviewing it is important to consider the screen interface that interviewers will use to administer questionnaires. Apart from ordinary screen useability issues it is also important to minimise the risk of measurement error that might come about from a sub-optimal design. While Blaise provides a good range of screen layout features, the default settings are not necessarily the best settings to use in all cases. Furthermore, the presentation standards that apply to paper questionnaires do not readily translate to the computer screen.

The screen layout recommendations, contained in this paper are based, in part, on papers by Dr Michael Couper and Mark Pierzchala. The design recommendations also take consideration of the hardware and operational circumstances at the ABS at this time.

This paper also describes the methodology employed in evaluating the proposed screen layout for ABS use.

## 2. Approach taken

The objective of developing a suitable screen layout for CAI is to optimise the interviewer's ability to interpret the screen presentation, read out the relevant question text with accuracy and at the same time follow any specified instructions such as showing a prompt card or recording multiple responses. This objective is no different from that which applies in the paper questionnaire environment.

In order to develop a screen layout that suits the ABS environment, the following approach was taken:

- a literature search was carried out to find relevant international material which dealt with screen design for CAI and Blaise;
- contact was made with major statistical agencies who make use of CAI, to identify current screen design practices;
- a sample of Blaise instruments used in some of these agencies was also obtained;
- the Blaise software was scrutinised closely to identify the various adjustments that are possible (also to identify any limitations which may exist);
- the relevant literature was discussed in an internal working paper and a set of draft recommendations for screen layout was prepared;
- comparative survey instruments were prepared using existing screen layout (based on some ABS paper questionnaire conventions and Blaise default settings) and using the recommended screen layout;
- initial useability testing was carried out to assess whether the proposed screen layout was preferable;

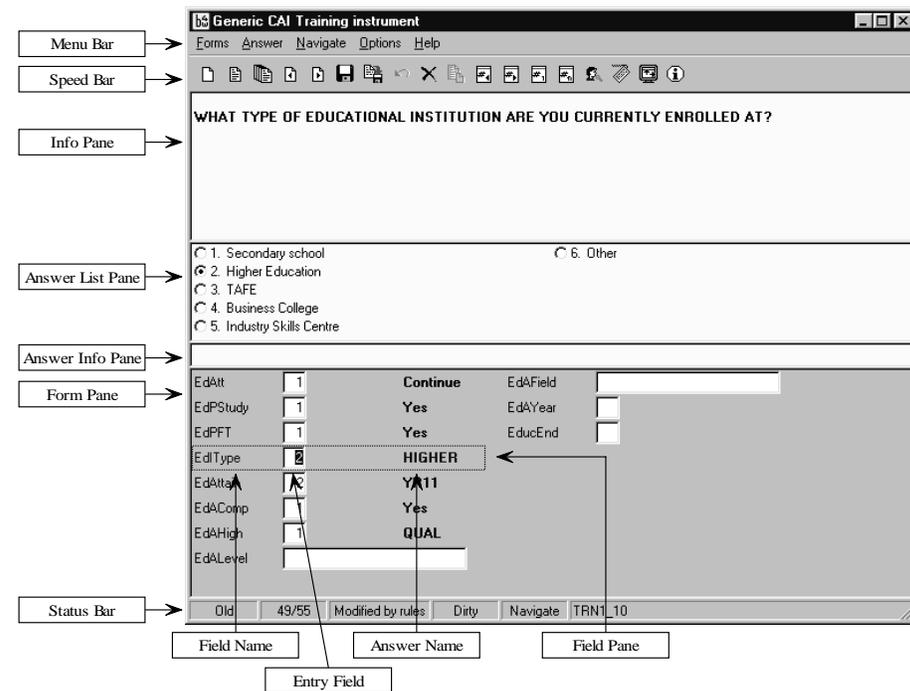
- more intensive useability testing was carried out to refine the proposed screen layout further; and
- a full field test was conducted using the proposed screen layout, as refined.

### 3. Basic elements of a Blaise CAI screen

Overall, the goal of the CAI screen is to focus the interviewers' attention rapidly on the key elements of the task, and allow them to complete those tasks efficiently and with minimal error (Couper et al, 2000). Screens often contain action items (the things that an interviewer needs to administer the questionnaire, such as the question to be read to respondent, the answer to be recorded and interviewer actions like showing prompt cards), information items (the things that facilitate the interviewer's delivery of the question and recording of the answer, but are not directly part of the question/answer process, such as help, question by question specifications, interviewer instructions), and auxiliary items (contextual information, such as case identifier, time/date displays, function key mapping, navigation tools, etc.).

The default Blaise screen is presented in Figure 1. Similar to Couper and colleagues (2000), the authors of this paper consider that the basic/default screen layout provided in Blaise requires several changes to facilitate the goals outlined above. The Mode Library component of Blaise permits a great deal of customisation.

Figure 1: Basic Elements of a Blaise CAI Screen<sup>1</sup>



<sup>1</sup> When parallel blocks are used, tabs can be made to appear between the Speed Bar and Info Pane. For an example of this feature see Figure 4.

The interviewer collects data with the Blaise Data Entry Program (DEP), which features a distinctive split-screen display shown in Figure 1. The screen in Blaise refers to the entire area of the computer window, from the title bar on the top and extending to the status bar on the bottom.

The upper part of the screen is called the Info Pane. It contains question text and other information meant for the interviewer.

The lower part of the screen is the Form Pane or Page. It contains data entry cells and the cursor moves from one data entry cell to another. Question text displayed in the Info Pane corresponds to the position of the cursor in the Page. The Form Pane can be seen as reflecting a page in a paper questionnaire. The term is intuitive to the interviewer, and the Page Up and Page Down keys move backwards and forwards, one page at a time.

#### 4. Screen Design Recommendations

There are many aspects of screen presentation, covering a range of items such as font type, size, colour, emphasis and positioning of text and other objects (icons, buttons etc). Each of these aspects then needs to be considered for each type of field element such as the question text itself, the instructions, the response set and data entry elements.

The screen design settings which follow are those which have been recommended for use as the default settings for all CAI instruments in the ABS. Flexibility exists to modify these settings to suit difficult questions or special data entry needs for particular surveys, in which case these recommendations could be used as a guide to the preferred presentation.

The screen design recommendations have been specified to suit a notebook computer screen set to a resolution of 1024x768, which is the conventional setting used on the current generation of computers at the ABS. Corresponding adjustments would be needed if the resolution settings on computer screens were changed.

A summary of the design recommendations is given in Table 1. A brief discussion of these settings follows.

**Table 1. Recommended screen design settings for screen resolution 1024x768**

Element and attribute	Recommended setting
<i>General</i>	
Blaise window	Use full screen, disallow resizing or minimising
Menu bar	Provide essential items only
Speed bar	Provide a limited number of icons only
Parallel blocks	Use as a navigational aid. Add context and status text to label.
Windows task bar	May remain visible (default)
On-line help	Use Blaise language facility (for convenience)
<i>Info Pane</i>	
Margins	Left - 5mm, Top - 5mm, Right - 100mm
Question text font, size and colour	San Serif, 14 point, black, bold
Question text placement	Left justified within the margins specified
Question text case	Mixed case
Instruction text font, size and colour	San Serif, 12 point, blue, bold
Instruction text placement	Indent text by one tab stop (about 10mm)
Instruction text structure	Use upper case action word. Separate from other instructions by a blank line.
Context information font, size and colour	San Serif, 12 point, blue, bold
Context information placement	Indent text by one tab stop (about 10mm)
Use of symbols instead of words	Explore their use for common actions

Element and attribute	Recommended setting
Fill text	Blend with existing text (use a colour highlight for testing only)
Emphasis	Specific words that require emphasis are to be highlighted using underline (rather than italics or bold)
Borders	Remove borders which produce unnecessary lines across the screen
Answer list	Where possible keep these to a single column
Answer info pane	Do not display
<i>Form Pane</i>	
Margins	Left margin to match the Info Pane (i.e. 5mm)
Text font and size (applies to the whole Form Pane)	San Serif, 12 point
Columns	Two columns
Rows	Enough to fill the pane
Context information text colour	Dark red
Context information placement	Indent text by one tab stop (about 10mm). Add text to give the appearance of a heading line. Leave single line space before (if room).
<i>Field Pane</i>	
Field attributes displayed	Field description (brief), Remarks point, Field value, Answer name
Field text colour	Normal - black, Highlight - blue
Answer name text colour and emphasis	Bold, Normal - grey, Highlight - blue
<i>Status Bar</i>	
Attributes displayed	Relative page number (to know place in the questionnaire) Field Tag (for references to documentation) Field Name (for communication of problems)

## 5. Info Pane settings

When interviewers set their eyes on the screen for the first time, they should be drawn immediately to the key features of the screen needed for successful delivery of the survey question and recording of the response (Couper et al, 2000). To do so requires a consistent design, visual discrimination among the different elements (so interviewers learn what is where, and know where to look), adherence to normal reading behaviour (i.e. start in upper left corner), and removal of unnecessary information (e.g. lines) or other display features that distract the interviewer from the task. Couper et al (2000) proposes that this would result in a cleaner design with more "white space" and a clearer distinction of the key components of the screen.

Increased font size has been chosen as the main way for attention to be drawn to the most important feature on the screen, namely the question text. Consideration was given initially to using a font size of 12 points for the question text on a screen with resolution 1024x768, with all other texts set to 11 points. This was soon changed, however, after some sample screens were prepared. Ultimately, it was considered appropriate to recommend a font size of 14 points for the question (and answer) text, with a size of 12 points for all other texts.

The use of bold rather than normal font for various text elements in the Info Pane and Field Pane has been done to make these important elements more distinguishable, especially under conditions that may include direct sunlight.

Special emphasis for particular words should be done through the use of underline rather than bold or italics. This is in line with conclusions drawn by Couper et al (2000) who found that underline was most distinguishable from normal text under a range of different font settings and lighting conditions. In fact, this convention was already in place for paper questionnaires.

Readability of text can also be adversely affected by line length, which is why the right-hand margin has been set to 100mm. The other margins have been defined to offset the text slightly from the borders of the Info Pane and Field Pane and thereby make the text more readily identifiable.

Some settings have been employed to keep the screen relatively free from "clutter" and enable the interviewer to focus on the important elements necessary to carry out interviewing functions. The main settings associated with this objective are the removal of various borders or lines on the screen, limiting the icons on the speed bar and, where possible, keeping the answer list to a single column.

Couper et al (2000) also recommends that the interviewer instructions and response categories be indented deeper than regular question text, as a way to further distinguish these items from the question. While the recommendation to indent interviewer instructions has been adopted for the ABS screen design, the indentation of response categories is not possible because this feature is not currently available in Blaise.

Colour has been used consistently as an identifying feature of particular text elements (e.g. Blue for instructions and context in the Info Pane, Dark red for context headings in the Form Pane). However, colour is not the only identifying feature, and other aspects such as placement or font size also help to distinguish various text elements. The background colours of cream (for the Info Pane) and light grey (for the Form Pane), which are set as the default colours in Blaise, are considered quite acceptable.

Mixed case has been recommended for screen text as it has been consistently found to facilitate reading speed (Schriver 1997, Galitz 1993). Pierzchala and Farrant (2000) also recommend using mixed case because it is clearer for interviewers to read. The recommendation to use mixed case does differ, however, from the ABS practice in relation to paper forms used by interviewers where upper case is used to signify text to be read out. While this change of practice could be seen as significant, it was considered that the use of CAI was sufficiently different from paper for a new set of conventions to be readily understood. This was one particular issue that was pursued in the useability testing.

One of the main features of CAI is the ability to tailor the question text to the circumstances of the interview through the use of "fill" text. Depending on the circumstances, the pronouns, dates and text references to previous answers in the survey can be "filled" with the appropriate words. In that way the interviewer can simply read the generated question texts and not be worried about adjusting the text as the interview proceeds. In early DOS systems for CAI at the ABS it was conventional to highlight the fill text through the use of colour or style change. Couper et al (2000) argues that fills should generally be indistinguishable from the surrounding text. This argument has been accepted for the ABS CAI screen layout, although a colour highlight is considered to be useful during testing and debugging of the instrument.

Couper et al (2000) recommends that some contextual information relevant to the interview be placed in the top right of the screen, but this is not currently possible in Blaise. Therefore, it has been recommended for the ABS screen layout, that context headings be placed in the Form Pane using dark red text and indented (see more on this below). Other context information, which might be required during the interview, such as the respondent's name or other details, should be treated in

the same way as interviewer instructions (i.e. one font size smaller than question text, blue and indented).

## **6. Form Pane settings**

The Form Pane displays the fields specified in the questionnaire along with spaces for entering the responses. The Form Pane is analogous to a page in a paper questionnaire, it organises and displays related data elements together (Pierzchala and Farrant, 2000). The design and specification of the Form Pane is often neglected but is considered by Pierzchala and colleagues to be just as important as the design of the Info Pane. The authors of this paper agree.

The Form Pane is made up of a Grid, the cells of which form smaller elements called Field Panes. A Field Pane is an area where the field description, data entry cell, and other related elements can be displayed. There are several features that can be specified in the Form Pane that enhance the presentation for the interviewer. Most of the time, instrument designers/programmers end up using the default features and these are not necessarily the most appropriate features for survey interviewing (Pierzchala and Farrant, 2000).

Pierzchala and Farrant recommend the specification of readable Field Descriptions to identify each field (as opposed to the default Field Name), with heading labels that group related questions, using two columns in the Form Pane with data entry cells in each column. Use of the Field Description, as distinct from the default Field Name, gives more flexibility, since spaces can be included in the description. Field descriptions can also be multilingual and used as the field identifier in the edit jump dialog.

For the ABS screen layout, the recommendation of using the Field Description to identify each field (Pierzchala and Farrant, 2000) has been endorsed. The Form Pane should also be enhanced through the inclusion of blank lines to divide the list of fields into groups and the addition of suitable context headings. For the ABS screen layout it has also been recommended that the headings be presented in dark red text, with the addition of some text (five dashes at each end) to give the appearance of a line. The font size for all text on the Form Pane has been recommended to be 12 points (for screen resolution 1024x768) which is the same as for interviewer instructions and context information on the Info Pane.

While it is possible to include up to ten different elements for each field in the Form Pane, the following four elements have been recommended for the ABS screen layout: Field Description (brief), Remarks Point, Field Value, and Answer Name.

Initial implementation of CAI at the ABS considered it sufficient to show only the current question field in the Form Pane. The change to making more extensive use of the Form Pane, as reflected in the recommendations above, was considered an important aspect to assess in the useability testing. For that purpose the Form Pane was configured to show up to 18 fields. Logical headings were added to groups of fields along with blank lines and commands to start new page at logical points in the question flow.

## **7. Other settings**

Couper et al (2000) recommends that the CAI screen be designed for keyboard use as distinct from mouse use. This is a valid recommendation, however in the ABS where some surveys are expected to be administered in both CAPI and CATI modes, and a mouse device is integrated into notebook computers, it is recommended that the CAI screen be designed to accommodate the use of both keyboard and mouse, with slightly more emphasis on the keyboard. This means providing relevant shortcut keys and menu options for keyboard use as well as

convenient mouse operated icons for common functions (e.g. exit, save). Consideration was also given to the fact that other facilities on the computer, being used by the interviewer, may also require use of the mouse. Therefore excluding it from use during the interview may be restrictive for interviewers who are comfortable with a mouse.

In the interest of keeping the screen as "clean" as possible, consideration was given to the removal of various screen elements such as the Speed Bar, Action Info Pane, Status Bar and the Windows Task Bar. The only element that has been recommended for removal (or non use) in the ABS screen layout, however, is the Action Info Pane. The other elements have been retained because they serve useful functions and are generally part of standard Windows software. The justification is, that interviewers will become used to these elements and their functions, and they will "blend" into the background during the interview because they are generally not needed. It has been recommended, however, that these retained elements be kept relatively free of objects and that their content be made relevant to the interview process. In the case of the status bar, the information recommended for display there (i.e. Relative page number, Field Tag, Field Name) provides the interviewer with discreetly placed additional information about the question in focus. These aspects of the screen design were also explored in the useability testing.

There are two ways that on-line help can be provided to interviewers when using Blaise. One method makes use of Winhelp and the other uses an alternate language within the instrument. Couper et al (2000) recommends using Winhelp because it integrates well with the Windows environment. Preparation of Winhelp files can be quite involved and requires software that is not generally part of the ABS software suite. Given that on-line help has not been used for any CAI surveys at the ABS to-date, it is proposed that a Blaise based solution be used for the ABS because it can be readily incorporated into the instrument. Winhelp may be reconsidered after some experience has been gained with the basics of on-line help. Consideration has been given to the use of icons instead of text for certain common instructions in order to simplify the screen further. This can be readily achieved through the use of Windings font, although graphics can also be used. Icons have the advantage of not being confused with question text. This aspect of screen layout is expected to be explored further on the basis that useability testing indicates it has merit.

To avoid the segmentation effect, in which interviewers report losing their place in the instrument, Couper et al (2000) has suggested developing a "master control screen" to facilitate interviewer navigation and flexibility (see also Sperry et al, 1998). They suggest this could present a list of the sections of the CAI instrument. Those sections available for access could be designated as such, while those not yet accessible to the interviewer could be greyed out. The interviewer could then enter any section that was available. Upon completion of the section, the instrument would return to the master control screen for selection of the next section to do. Interviewers could also use it to tell at a glance their progress in an interview.

For the ABS situation it is recommended that consideration be given to using such a control screen for only the more complex surveys, at this stage. Instead, it is recommended that more work be done with the use of parallel blocks, a feature of Blaise that did not receive attention by Couper. The Blaise software has a special feature called Parallel Block that can be used to depart from the serial processing order that is specified in the rules. This feature allows different sections of a questionnaire to be completed in parallel to the main path. Once Parallel Blocks are in place the interviewer can access the parallel blocks through a menu option or through tabs at the top of the interview screen. The text, which appears on the tabs, can be tailored to the circumstances of the interview, showing respondents' names and status of their part of the interview. It is recommended that this feature of Blaise be used where flexibility of navigation is required.

There are many aspects of screen layout that are not discussed here but still need to be considered when preparing for CAI. Irrespective of the reasoning behind particular settings, it is important to conduct some useability tests to ensure that an appropriate level of functionality is achieved.

## 8. Useability Testing

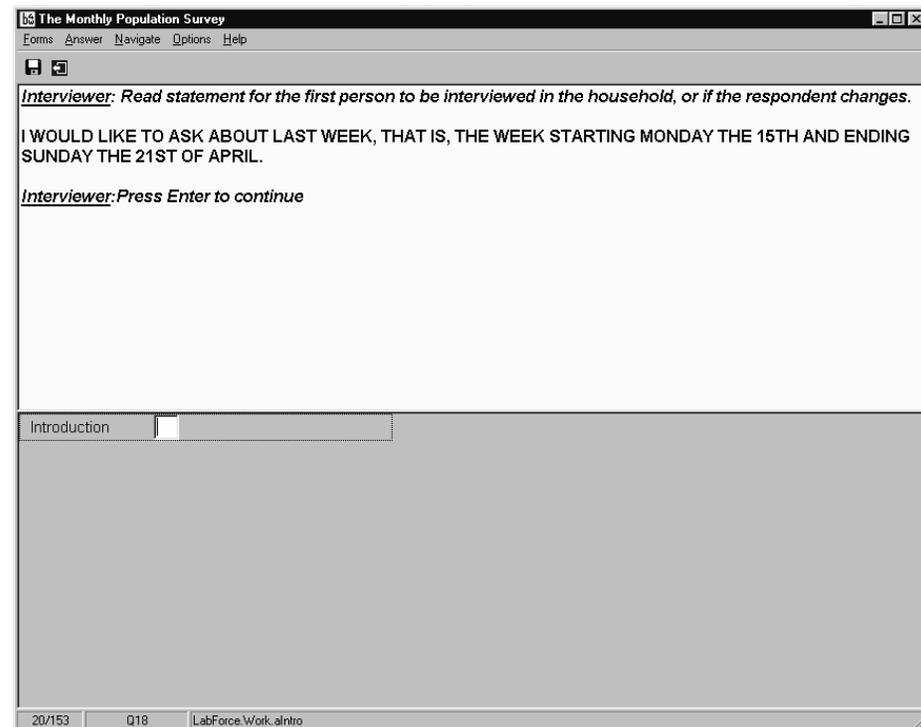
Many of the techniques available for pre-testing paper questionnaires, can be used to evaluate CAI systems and survey instructions, including the effectiveness of CAI layout and design. However, instead of focussing on respondents' understanding of the questions, useability evaluation focuses on interviewer interaction with the CAI system and survey instrument.

To evaluate the proposed screen layout, it was decided to conduct two useability tests using a walk-through technique. This is a quick and flexible method for collecting user feedback regarding the design of user interface mock-ups and prototypes.

During the walk-through, users were presented with various screen designs associated with a workflow scenario and were prompted to respond to the screen design elements.

For the first test, two alternative survey instruments were prepared. The first instrument was developed using an existing screen layout, based in part on ABS paper questionnaires and Blaise default settings. A sample screen from this instrument is shown in Figure 2.

Figure 2: Sample screen based on ABS paper questionnaires and Blaise default settings



The second instrument was prepared using the using the settings in the recommended screen layout. A sample screen from the second instrument is shown in Figure 3.

Figure 3. Sample screen showing new layout features

For the first test, only a few parts of the standard labour force survey were placed into a test instrument.

For the second test, other components of the survey, such as the household roster and supplementary survey questions, were added, along with other features such as help icon and parallel block tabs. A sample screen from the instrument used in the second test is shown in Figure 4.

Figure 4. Sample screen showing enhanced layout features

Each test was conducted using six interviewers who were individually taken through a series of interview scenarios using the sample instruments. For the first test, half of the interviewers were exposed to the default Blaise screen layout first and the other half was given the recommended screen layout first.

The broad objectives of the useability testing were to gain user input to and evaluation of screen design elements.

The detailed objectives were to assess:

- readability of questions and interviewer instructions (i.e. font size, mixed case, bold, indentation/margins, font colour) between the two screen designs;
- use of icons and upper case words in the interviewer instructions;
- use and preferences for fill text to be blended with question text (i.e. in terms of colour, mixed case for respondents names, etc);
- use of field descriptions and section heading in Form Pane;
- use of help pop-up box, and what information interviewers would like to be displayed in the help function;
- use of speed bar and status bar (i.e. whether these should be displayed on screen) and if so, what information would be useful to display in them; and
- ease or difficulty in navigation through the instruments.

The main procedure followed in the walk-through was:

- introduction to the test and its objectives;
- obtain consent to /video tape the session;
- provide instructions on use of the notebook computer;
- review each scenario with the user (respondent played by facilitator, interviewer used the interface); and
- users tell the facilitator about the actions they would take and aspects they liked or did not like (during or after the scenario).

The testing was interactive, where the facilitator could interrupt users to probe behind the issues, being careful to make the participants feel at ease, encouraging and confirming their responses.

The video tapes were reviewed after the sessions were completed and detailed notes were made about the observations. A comprehensive report was produced after each test, containing summaries of the observations and recommendations for changes to be made for the next round.

## **9. Summary of results**

The reports from each test were comprehensive and covered many aspects of the screen design. A brief summary of findings is presented here.

All interviewers reported they preferred the recommended screen layout to the layout that applied default Blaise settings (and paper conventions). The main screen design issues that interviewers preferred were:

- bolded, black text with indentation and margins, allowing for easier reading on the screen;
- interviewer notes in blue and indented because it helped them to stand out from the questions;
- interviewer notes being briefer, more efficient to read;

- contextual information presented in the Form Pane which allowed for easier navigation and the ability to identify and correct mistakes more efficiently;
- the screen looked clearer and wider, thereby aiding visibility.

### **9.1 Readability of questions and interviewer instructions**

All participants reported that they had no difficulty reading the text on the screen, the font size was reported by all to be easy to read and all commented on their preference for the use of colour to differentiate between questions and interviewer notes (i.e. black, bolded for questions and blue for interviewer notes). The majority of interviewers preferred the mixed case question text to that which used all upper case.

### **9.2 Use of icons and upper case words in the interviewer instructions.**

The majority of interviewers reported that they liked the use of symbols and short upper case words to aid the reading of notes to interviewers.

### **9.3 Use and preferences for fill text to be blended with question text**

All participants reported a preference for fill text that was blended within the question. It was preferred, however, that fills match the text from previous answers as much as possible (e.g. questions on child-care should contain the names and ages of the children in the household for whom the questions are relevant).

When optional question text was presented in rounded brackets, interviewers reported that the impulse to read it out was stronger because the brackets did not stand out. However, upon seeing some optional question text enclosed in square brackets, some interviewers noted that these helped to make it stand out and they paused to notice whether it should be read or not.

### **9.4 Use of field descriptions and section headings in the Form Pane**

All participants reported, when probed, that having the context displayed in the Form Pane was useful, especially when navigating back through the instrument or checking that a response had been entered correctly.

Participants also noted that the Form Pane tended to be in their peripheral vision most of the time because they were mainly entering responses using the response categories presented underneath the question in the Info Pane. However, when they were required to enter responses such as dates or numbers, this could only be recorded in the Form Pane yet participants were still looking to record the response in the Info Pane.

More specifically, participants felt they needed to constantly switch their attention up to the questions in the Info Pane and then to record these responses in the Form Pane and then back up to the question. This occasionally led to interviewers making the mistake of reading the next question from the Form Pane before realising that they needed to look back up to the Info Pane to read the question. All participants did note, however, that as they became more familiar with the CAI instrument, it was easier to shift attention back and forth between the two Panes. This was identified as a training issue.

### **9.5 Use of on-line help**

Although the majority of participants did not accurately guess what the help icon referred to, once they pressed the F9 key, shown beside the icon, they accurately described it as some kind of help function. They all reported that a help option would be very useful, especially for new or difficult topics.

The majority of participants did caution that if a help pop-up box was to be used then it needed to be consistently used and presented throughout the instrument (i.e. if definitions are presented in the help pop-up box, then definitions should not be

presented on the screen underneath or along side the question text, as happened for some questions).

### **9.6 Use of speed bar and status bar**

Most participants reported not noticing the status bar, and therefore did not pay attention to the information displayed in it. When it was pointed out to them that page numbers and question numbers were displayed in the status bar, they all reported this could be helpful, especially when reporting issues back to office staff. In relation to the speed bar, participants did not use it during testing, even after being probed about it and having its purpose described to them. All participants did acknowledge that, with greater familiarity and use of the notebook, they might make use of the speed bar in the future.

### **9.7 Ease or difficulty in navigation through the instruments**

All participants tended to ask how to navigate back through the instrument to make a correction. However once directions were given, all participants reported navigation to be fairly easy and straightforward. Some participants noted they believed navigation was easy because contextual information was displayed in the Form Pane (i.e. context of preceding and forthcoming questions).

## **10. Further testing and conclusion**

Useability testing has shown that the recommended screen layout, described in this paper, was functional and has been favourably received by the participants. As a result, the new layout has been adopted for the current series of CAI instruments at the ABS.

Field tests of the current CAI instruments are also likely to lead to further suggestions for improvement, as the issues relating to screen design and layout will continue to be included as discussion topics at debriefing. Where specific screen design issues need to be explored further, the option also exists for laboratory testing to be done.

It is recognised that there are many aspects to screen design and useability and this paper has covered but some of them. The CAI team at ABS will continue to examine the issues surrounding screen layout and functionality and make further improvements as required.

## **11. Acknowledgments**

Recognition is given to all those who have assisted the team at ABS in defining and testing a new screen layout for CAI. In particular we would like to acknowledge the willing cooperation of Dr Mick Couper, Mark Pierzchala and Dr Marek Fuchs, as well as the interviewing staff at the ABS.

## **12. References**

Bushnell, D. (2000). "From DOS to windows: Usability issues for interviewers" *Proceedings of the Sixth International Blaise Users Conference*, Kinsale, Ireland, May 2000.

Couper, M.P., Beatty, P., Hansen, S.E., Lamias, M., Marvin, T. (2000). "CAPI Design Recommendations", *Report submitted to the Bureau of Labour Statistics. Interface Design Group, Survey Methodology Program*, Survey Research Center, University of Michigan.

- Degeral, H. (2000). "The process of making a new CAI-operation in Statistics Norway", *Proceedings of the Sixth International Blaise Users Conference*, Kinsale, Ireland, May 2000.
- Finlay, D. (1999). "International Experience in the Introduction of Computer Assisted Interviewing (CAI) into Labour Force Surveys", *ABS Internal document* (unpublished).
- Fuchs, M. (1999). "Screen design and question order in a CAI instrument Results from a usability field experiment" *Paper presented at the 54<sup>th</sup> Annual meeting of the American Association for Public Opinion Research*, Petersburg, Florida, USA 1999.
- Fuchs, M., Couper, M.P., and Hansen, S.E. (2000). "Technology effects: do CAPI or PAPI interviews take longer?" *Journal of Official Statistics*, 16(3), 273-286.
- Galitz, W.O. (1993) "User-Interface Screen Design", Boston: QED Information Services Inc.
- Hansen, S.E., Couper, M.P., and Fuchs, M. (1998). "Usability evaluation of the NHIS instrument". *Presented at the 53<sup>rd</sup> Annual Meeting of the American Association of Public Opinion Research*, St. Louis, Missouri USA, 1998.
- Hansen, S.E., Beatty, P., Couper, M.P., Lamias, M., Marvin, T. (2000). "The effect of CAI screen design on user performance: Results of an experiment", *Report submitted to the U.S. Bureau of Labour Statistics. Interface Design Group, Survey Methodology Program*, Survey Research Center, University of Michigan.
- Kelly, M. (1998). "Producing an error-free CAI instrument- is it possible?" *Proceedings of the Fifth International Blaise Users Conference*, Lillehammer, Norway, 1998.
- Pierzchala, M. (1998). "Optimal screen design in Blaise". *Proceedings of the Fourth International Blaise Users Conference*, Paris, France, 1997
- Pierzchala, M., and Farrant, G. (2000). "Helping non-Blaise Programmers to specify a Blaise instrument" *Proceedings of the Sixth International Blaise Users Conference*, Kinsale, Ireland, May 2000.
- Pierzchala, M., and Manners, T. (1998). "Producing CAI instruments for a program of surveys". *Computer Assisted Survey Information Collection*. Wiley Series in probability and statistics, New York.
- Pierzchala, M., and Manners, T. (2001). "Revolutionary Paradigms of Blaise". *Proceedings of the 7th International Blaise Users Conference*, Washington DC, USA, 2001.
- Schrivver, K.A. (1997). "Dynamics in document design: Creating text for readers". John Wiley & Sons, New York.
- Sperry, S., Edwards, B., Dulaney, R. and Potter, D.E.B. (1998), "Evaluating Interviewer use of CAPI Navigation Features". *Computer Assisted Survey Information Collection*. Wiley Series in probability and statistics, New York.
- The Hiser Group (1995). "Designing with users: The key to success". Trailmoss Pty Ltd trading as The Hiser Consulting Group, Australia.



# Screen Design Guidelines for Blaise Instruments

*Sue Ellen Hansen & Karl Dinkelman, Survey Research Center, University of Michigan*

## 1. Introduction

Many survey organizations now recognize the need for computer assisted interviewing (CAI) screen design guidelines for Blaise programming. Like many, the Survey Research Center (SRC) at the University of Michigan has developed a set of screen design guidelines. Originally developed in 2000, these guidelines were based on key principles taken from Human-Computer-Interface (HCI) research, which include:

- consistent screen design
- visual discrimination among the different elements (so that CAI users learn what is where, and know where to look)
- adherence to normal reading behavior (i.e., start in upper left corner)
- display of instructions at points appropriate to associated tasks (e.g., the show card or respondent booklet instruction precedes the question and the entry instruction follows the question)
- elimination of clutter and unnecessary information or other display features (e.g., lines and toolbars) that distract users from immediate tasks

The original SRC guidelines specified text font and colors (for questions, response categories, and instructions), the placement of help indicators and instructions, the formatting of variable text and error messages, and so on. Several attempts to apply the guidelines, however, revealed that there were many instances in most questionnaires in which specific guidelines could not be applied consistently.

The guidelines underwent extensive revision during the course of redesign of an existing survey instrument. During that process attempts were made to consistently apply current principles, and to refine them when they were shown to be inadequate. The result is a more extensive set of proposed guidelines that can apply more broadly to the survey instruments SRC develops.

This paper summarizes the major issues faced during the redesign, and provides an overview of the new guidelines with numerous examples of redesigned screens. Major additions include guidelines for “interviewer checkpoints” (non survey question screens), multiple part questions (e.g., MM/DD/YYYY), and design of probes and data entry instructions.

## 2. Developing CAI Screen Design Guidelines

Good CAI screen design respects the natural flow of the user's tasks. It should make the interviewer or respondent more efficient, prevent errors, and reduce training or retraining needs, through making required tasks obvious. On seeing a screen for the first time, the user's eyes immediately should be drawn to the key features of the screen for successful delivery of the survey question and recording of the response. Ideally, a relatively untrained interviewer or respondent seeing a particular screen for the first time should be able to distinguish the action items from the information items, and know what the immediate task should be.

The question is the most important feature of the CAI screen, and it should be the most visible element of the screen, immediately identifiable. In addition, different question types and response input formats (for example, currency, dates, grids, and open text), and other elements of the screen (for example, response options, on-line help indicators, and section and other context information) should all be formatted consistently. Keeping in mind these guidelines and the HCI principles previously mentioned, SRC outlined basic Blaise screen formatting guidelines in early 2000, and have revised them over time, based on attempts to apply them in the development of various survey instruments.

One of the lessons we repeatedly have learned is that we do not always have control over screen design, particularly when current instrument development relies heavily on previously designed questionnaires. This is particularly a problem with longitudinal or panel surveys, or surveys that use modules in whole from other surveys. Examples are the U.S. Health and Retirement Survey (HRS), the World Mental Health survey (WMH), and surveys that use WMH components, such as the U.S. National Survey of Health and Stress.

Figure 1 shows a screen from a survey developed prior to SRC's 2000 guidelines, in which instructions and response options appear in upper case. Prior to the use of Blaise for Windows at SRC, CAI instruments that were developed in DOS-based systems, with limited screen text options, displayed all interviewer instructions and response codes in upper-case or capitalized text. This followed long-standing SRC guidelines for paper-and-pencil surveys. Early surveys in Blaise tend to use these same conventions.

This screen has other features of early SRC surveys in Blaise: (1) the question identifier (CC6\_2) appears at the top of the screen, which clutters the screen; all text, whether instruction or question text, is flush with the left margin; and (3) the respondent booklet instruction appears below the question identifier, distinguished by the "pointing finger" icon, rather than being the first thing the interviewer focuses on. Later guidelines suggested eliminating the question identifier from the top of the screen, since it was also displayed in the form pane, indenting instructions that followed the question, to further distinguish them from the question, and providing a clearer respondent booklet instruction (see Figure 3 in the "Examples" section).

Another lesson repeatedly learned was that our guidelines were not comprehensive enough to cover a great many design decisions questionnaire specification writers and instrument programmers had to make in real life. This was particularly true for display of the many types of interviewer instructions that researchers asked programmers to place on the CAI screen.

Figure 1. Screen formatted using basic screen design guidelines

CC6\_2.

(RB, PG 42)

Which of the conditions on this list or any other conditions resulted from that injury? (Just give me the number from the list?)

RECORD ALL MENTIONS

1. BROKEN OR DISLOCATED BONES  
 2. SPRAIN, STRAIN, OR PULLED MUSCLE  
 3. CUTS, SCRAPES, OR PUNCTURE WOUNDS  
 4. HEAD INJURY, CONCUSSION  
 5. BRUISE, CONTUSION, OR INTERNAL BLEEDING  
 6. BURN, SCALD  
 7. POISONING FROM CHEMICALS, MEDICINES, OR DRUGS  
 8. RESPIRATORY PROBLEM SUCH AS BREATHING, COUGH, PNEUMONIA  
 96. OTHER (SPECIFY)

CC3n	<input type="text" value="1"/>	CC6a	<input type="text" value="1"/>
CC4n	<input type="text" value="1"/>	CC6a1	
CC3o	<input type="text" value="1"/>	CC6_1	<input type="text" value="1"/>
CC4o	<input type="text" value="1"/>	CC6_1a	<input type="text" value="1"/>
CC3r	<input type="text" value="1"/>	CC6_1b	<input type="text" value="1"/>
CC3s	<input type="text" value="1"/>	CC6_1b1	
CC3t	<input type="text" value="1"/>	CC6_1c	<input type="text" value="1"/>
CC6	<input type="text" value="1"/>	CC6_2	<input type="text" value=""/>

11239812321 8:49:41 PM

The SRC 2000 screen guidelines recommended avoiding the use of capitalization, large blocks of which are hard for the interviewer to read. Other basic guidelines were to display question text in 12 point Arial black, and instructions in 11 point Arial blue, capitalizing only action verbs in instructions (for example, ENTER). Such limited use of capitalization was designed to draw attention to the task involved, and to help the interviewer locate a particular instruction type quickly. Figure 2 provides an example of how this guideline was applied in the National Survey of Family Growth (NSFG).

It is obvious in this example that capitalizing any and all action verbs anywhere in an instruction defeats the purpose of the capitalization. In some cases, the capitalization draws the interviewer's attention away from the primary focus of the instruction. This is the case with "REFER" in the first instruction, in which the interviewer's attention is drawn immediately to the action verb, rather than the important conditional clause. Seeing such actual applications of the guideline to capitalize action verbs in instructions led to a new guideline that suggested capitalization of a limited set of verbs associated with key interviewer tasks, for example, ASK, READ, PROBE, and ENTER, and to use capitalization only if the verb appeared at the beginning of the instruction.

Figure 2. Example of old guideline to capitalize action verbs in instructions

Blaise Data Entry - C:\My Documents\WPDOCS\NSFG\blaise files\d\_tem

Forms Answer Navigate Options Help

Calendar

When did you have your tubal sterilization?

If R cannot recall month and year, REFER her to the life history calendar.

If R can recall the year but not the month, RECORD the year and PROBE for season.

NOTE: If R had tubes tied or ovaries removed in 2 separate operations, record month/year for most recent operation. This is the date when she became completely sterile.

1. January       5. May       9. September       13. Winter  
 2. February       6. June       10. October       14. Spring  
 3. March       7. July       11. November       15. Summer  
 4. April       8. August       12. December       16. Fall

DC\_1 DATOPNNR\_ 1 January

DC\_1a DATOPNNR\_ 1999

After review of several CAI instruments in May 2002, SRC decided to update the guidelines, making them clearer, more comprehensive, and easier to apply, and ensuring consistency within and across instruments. The first application of the guidelines was to a newly revised WMH instrument that was to be modularized for easy insertion of components of the survey into other surveys. Using the guidelines in revising a complete instrument helped identify gaps in the guidelines and areas of potential misapplication. The next sections outline some of the resulting proposed changes to the SRC Blaise screen design guidelines, and give examples of how they were applied to the WMH instrument.

### 3. Basic SRC Blaise Screen Design Guidelines

Following are key elements of the SRC guidelines currently under revision.

#### Basic text characteristics

Make the question the dominant element on the screen. Provide sufficient contrast between the question and related text and other elements of the screen:

- question text: light background color (cream) and dark text (mixed case, 12 point Arial, black)
- instructions: smaller font in color (11 point Arial blue)
- response categories (answer lists): use same conventions as for question text
  - categories that could be read to the respondent are 12 point Arial black
  - categories that would not be read to the respondent are 11 point Arial blue

#### Additional text characteristics

- use underline for emphasis, sparingly
- (place optional text in parentheses)
- place in text references to numbers computer keys to type in mixed case within square brackets, for example [Enter], [1], [F12], [Ctrl-R]

### Elements of the screen

Place elements in the order in which the interviewer would need to attend to them, according to interviewing task demands. For example, a show card instruction precedes the question, and a probe instruction follows the question but precedes an entry instruction.

- references to interviewer aids (such as event history calendar or show card) and question appear in the upper left corner of the screen
- instructions that follow the question are indented
- a question-level help indicator ( [F1]-Help ) appears above the question on the right margin

### Additional instruction guidelines

- “bullet” individual instructions with a small blue diamond
- display in order associated with required interviewer tasks
- instructions that precede the question are flush left with the question
- instructions that follow the question are indented
- use icons to distinguish special instructions
  - e.g.,  Page 1, for respondent booklet instruction,  Calendar, for event history calendar instruction, and  Interviewer checkpoint
- include an actual question in interviewer checkpoints
- capitalize only key task-related action verbs, (ASK, READ, ENTER, PROBE), only at the beginning of instructions
- single space within an instruction, double space between instructions
- keep instructions simple and concise
  - put long instructions or those not directly related to asking questions or entering responses into online help
- conditional instructions start with the conditional phrase, not the action verb, and the action verb is not capitalized
- in probe instructions, place text to be read in Arial black
- place references to respondent answers in quotation marks

## **4. Examples**

### Basic screen design

Figure 3 shows a typical question formatted according to the new basic screen design guidelines. It is the same question shown in Figure 2. In this version, the respondent booklet instruction has been simplified, with a new booklet icon, and a clearer page reference, Page 42 rather than (RB, PG 42). The entry instruction is bulleted, in mixed case, with the action verb “enter” capitalized. It is also indented to further separate it from the question text, which is the primary focus of the screen. The response options are mixed case, with the “other – specify” code in blue. This follows the guideline to place only text that could be read to the respondent in Arial black—although the interviewer is not required to read response options, she would have to if the respondent were having trouble reading the respondent booklet.

Figure 3. Question designed using new guidelines (as compared to Figure 1)

WMH2000 - CAPI Modularization Demo  
Eoms Answer Help

Page 42

Which of the conditions on this list or any other conditions resulted from that injury? (Just give me the number from the list?)

- ENTER all that apply

<input type="checkbox"/> 1. Broken or dislocated bones	<input type="checkbox"/> 6. Burn/scald
<input type="checkbox"/> 2. Sprain, strain, or pulled muscle	<input type="checkbox"/> 7. Poisoning from chemicals, medicines, or drugs
<input type="checkbox"/> 3. Cuts, scrapes, or puncture wounds	<input type="checkbox"/> 8. Respiratory problem such as breathing, cough, or pneumonia
<input type="checkbox"/> 4. Head injury/concussion	<input type="checkbox"/> 96. Other -- specify
<input type="checkbox"/> 5. Bruise, contusion, or internal bleeding	

CC6\_2

11239412321 | 7:43:55 PM | Vers. Date: 04/12/2003 | Vers. Time: 5:54 PM | BLCHRONIC.CC6\_211

### Conditional instructions

The greatest limitation of the original screen guidelines was the very basic convention of displaying instructions in mixed case in blue, not bulleted, with just a few simple guidelines about specific types of instructions. This turned out to be completely inadequate for a large number of instructions that appeared in the WMH instrument, particularly for probe and entry instructions. One problem was that discussed earlier (Figure 2), a capitalized action verb drawing the interviewer's attention away from the primary focus, which was the condition for the action. This led to guidelines for conditional instructions. Figure 4 shows an example of a question with a conditional probe instruction formatted under the new guidelines, and Figure 5 shows conditional entry instructions.

Figure 4. Question with conditional probe instruction

WMH2000 - CAPI Modularization Demo

Forms Answer Help

What if you were faced with one of these situations today. How strong would your fear be -- not at all, mild, moderate, severe, or very severe?

- If volunteered "It depends on which situation," probe: What if you were faced with the situation that scares you most. How strong would your fear be -- not at all, mild, moderate, severe, or very severe?

C 1. Not at all  
C 2. Mild  
C 3. Moderate  
C 4. Severe  
C 5. Very severe

AG15  AG19

AG16

AG17

AG17a

AG17b

AG18

11239412321 | 6:54:05 PM | Vers. Date: 04/12/2003 | Vers. Time: 5:54 PM | BLAGORAPHO.AG18

Figure 5. Question with conditional entry instruction

WMH2000 - CAPI Modularization Demo

Forms Answer Help

People do not always take their medicine as they are supposed to. Think of a typical month when you took Adapin in the past 12 months. How many days out of 30 did you typically either forget to take it or take less of it than you were supposed to take?

- If volunteered "Not supposed to take regularly", enter [996]
- If volunteered "Never took for full month", enter [997]

PH16

PH17

PH19

PH19a

PH20

11239412321 | 7:40:34 PM | Vers. Date: 04/12/2003 | Vers. Time: 5:54 PM | BLPHARMACO.PH14\_20.PH16

The conditional probe instruction starts with the condition, "If ...," and does not capitalize the verb "probe," so that the interviewer's attention will not be drawn away from the condition. The referenced respondent answer appears in quotation marks, meant to help the interviewer quickly focus on the actual condition under which she would have to probe. Probe text is provided in Arial black, indicating that it is to be read verbatim to the respondent. Similar conventions are used for the conditional entry instruction (Figure 5), with what the interviewer is to enter, under the specific condition, placed in square brackets. This example also follows

the guideline that keys and function keys that the interviewer is to type are always placed in brackets, and helps the interviewer to quickly focus on what to enter under the specified condition.

### Interviewer checkpoints

In many CAI surveys there are items that ask for information from the interviewer, in order to program the subsequent flow of the interview or to tailor the display of subsequent items. Previous SRC guidelines did not address how to handle such “interviewer checkpoints.” As a result, such items were formatted in a variety of ways across SRC surveys, and most did not actually ask the interviewer a question. Figures 6 and 7 provide “before” and “after” examples of an interviewer checkpoint.

The old version of the item has upper case text that is neither an instruction nor a question, and the interviewer is left to determine what must be done at the screen. The new version use the new checkpoint () icon, followed by “Interviewer checkpoint“. This provides both graphical and text clues for the interviewer that this item is not a question to be read to the respondent. There is a question, but it is displayed in blue, also indicating that it is not to be read. Response categories are also in blue, reinforcing that this is not a survey question.

There are some screens that are neither interviewer checkpoints nor survey questions. They display information for the interviewer to review, or instructions to the interviewer prior to proceeding in the interview. Figures 8 and 9 show “before” and “after” examples of such an item.

In this item there are three statements or instructions, which appear in Figure 8 in upper case, not bulleted, and therefore not clearly distinguishing three separate instructions. The first displays filled text that gives the name, age, and gender of the selected respondent. The others are instructions for what to do if the information is incorrect, and what to do if it is correct.

The redesigned screen in Figure 9 makes these three distinct bulleted statements or instructions, and separates and indents the filled text, so that the interviewer can quickly focus on it when the screen is displayed. The final instruction, which originally seemed unrelated to the response option, was changed to the more appropriate “ENTER [1] to continue.“

Figure 6. An interviewer checkpoint before guideline revisions

The screenshot shows a software window titled "NATIONAL SURVEY OF HEALTH AND STRESS" with a menu bar containing "Forms", "Answer", and "Help". The main content area is divided into three sections. The top section contains the text "HU10" and "HU LISTING OBTAINED FROM:". The middle section contains a list of radio button options: "1. HU Member", "2. Neighbor", "3. Apt. Mgr.", "4. Landlord", "5. Observation", and "7. Other". The bottom section contains a "SampleID" field with the value "11239812321" and several input fields for "HU9", "HU10", "HU10S", "HU8", and "HU11", each with a small square next to it. The status bar at the bottom displays "11239812321" and "8:19:58 PM".

Figure 7. An interviewer checkpoint after guideline revisions

The screenshot shows a software window titled "WMH2000 - CAPI Modularization Demo" with a menu bar containing "Forms", "Answer", and "Help". The main content area is divided into three sections. The top section contains a checked checkbox labeled "Interviewer checkpoint" and the text "Who gave you the household listing?". The middle section contains a list of radio button options: "1. Household Member", "2. Neighbor", "3. Apartment Manager", "4. Landlord", "5. Observation", and "7. Other -- specify". The bottom section contains an input field for "HU10" with a small square next to it. The status bar at the bottom displays "11239412321", "Forms", "Vers. Date: 04/12/2003", "Vers. Time: 5:54 PM", and "BLN\_HHLHU10".

Figure 8. Interviewer confirmation screen before guideline revisions

The screenshot shows a window titled "NCS-R / WMH2000 CAPI Application - DEMO". The main text area contains the following instructions:

THE RESPONDENT FOR THIS CASE IS tobis, 70, MALE.  
IF THIS IS NOT CORRECT PLEASE USE YOUR PAGE UP KEY TO MOVE BACK TO THE HOUSEHOLD ROSTER AND CORRECT THE HOUSEHOLD LISTING.  
PRESS "\*" TO VERIFY THE HOUSEHOLD LISTING.

Below the text is a section labeled "C 1. CONTINUE".

The bottom section contains a list of variables and their corresponding values or options:

DeleteIntro		SC3	<input type="checkbox"/>
EndRoster	<input type="checkbox"/>	SC3a	<input type="checkbox"/>
AllAdolHH		SC4	<input type="checkbox"/>
ChosenPeopl	<input type="checkbox"/>	SC4a	<input type="checkbox"/>
SC0_1		SC5	<input type="checkbox"/>
SC0_3		SC7	<input type="checkbox"/>
SC0		SC9c	<input type="checkbox"/>
SC1	70	SC9d	<input type="checkbox"/>

The status bar at the bottom shows "22222222" and "9:19:39 PM".

Figure 9. Interviewer confirmation screen after guideline revisions

The screenshot shows a window titled "WMH2000 - CAPI Modularization Demo". The main text area contains the following instructions:

- The respondent for this case is:  
TOBIAS, 70, Male
- If TOBIAS is not the correct respondent, use the [Page Up] key to move back into the roster and correct the household listing
- ENTER [1] to continue

Below the text is a section labeled "C 1. Continue".

The bottom section contains a list of variables and their corresponding values or options:

EndRoster	<input type="checkbox"/>	SC3	<input type="checkbox"/>
AllAdolHH		SC3a	<input type="checkbox"/>
SC0_1		SC4	<input type="checkbox"/>
SC0_3		SC4a	<input type="checkbox"/>
SC0		SC5	<input type="checkbox"/>
SC1	70	SC7	<input type="checkbox"/>

The status bar at the bottom shows "12113542761", "8:43:36 PM", "Version Date: 06/22/2002", and "Version Time: 2:10 PM".

### Multi-part questions

Some questions require entry of multiple items for one response. For example, "What is your date of birth?" requires entry of month, day, and year. SRC refers to these as multi-part questions. The SRC 2000 guidelines did not provide recommendations for such questions. A usability evaluation of the National Health Interview Survey suggested that a screen design in Blaise that facilitates the interviewer associating the parts of such multi-item questions can lead to greater

usability of the instrument and improved data quality (Hansen 2003). Therefore, the SRC proposed guidelines make a recommendation for screens for multi-part questions.

Figures 10 and 11 shows “before” and “after” examples of the first part (feet) of a multi-part question, “How tall are you?” Figure 12 shows the “after” example of the second part of the question (inches). There is no information in original version of the question (Figure 10) that indicates that the item is the first of two associated with the actual question, except with the “RECORD FEET FIRST” part of the capitalized instruction, which is hard to read. In the revised counterpart (Figure 11), the first thing the interviewer sees on entering the screen is ① of ② , indicating that there are two parts to the question. In addition, \_\_\_feet and inches follows the question, with the boldfaced blank and “feet” indicating that this part calls for entry of how tall, in feet. The first thing the interviewer sees on the second question part is ② of ②, and 6 feet and \_\_\_ inches follows the repeated question, associating it with the prior item, but in parentheses to indicate reading it is optional.

Figure 10. Example of first part of multi-part question before guideline revisions

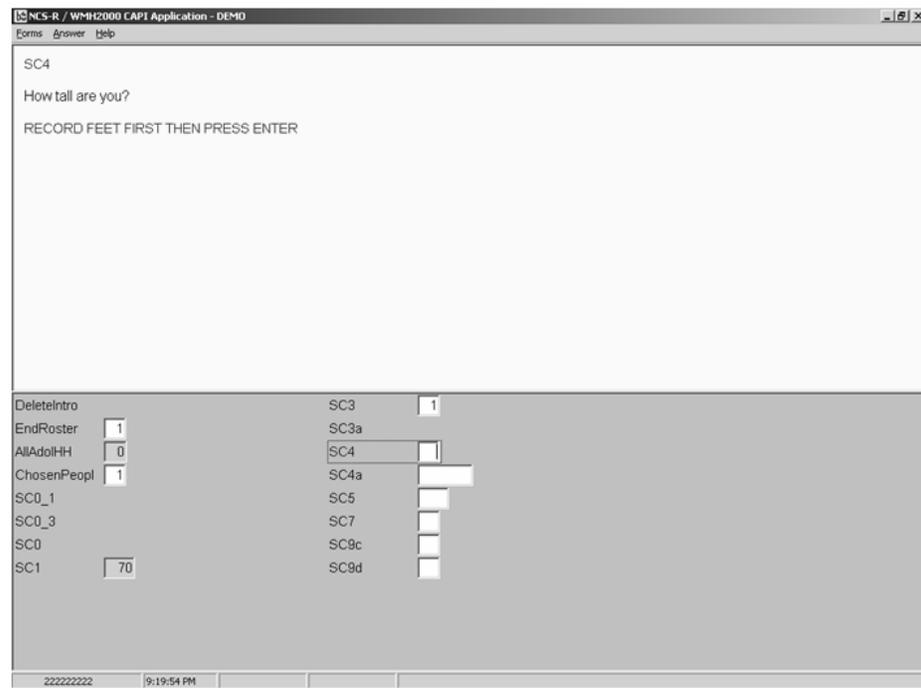


Figure 11. Example of first part of multi-part question after guideline revisions

WPHZ2000 - CAPI Modularization Demo  
Forms Answer Navigate Options Help

① of ②

How tall are you?  
\_\_\_feet and inches

EndRoster	1	SC3	1
AllAdolHH	0	SC3a	
SC0_1		SC4	6
SC0_3		SC4a	
SC0		SC5	
SC1	70	SC7	

12113542761 8:43:51 PM Version Date: 08/22/2002 Version Time: 2:10 PM

Figure 12. Example of second part of multi-part question after guideline revisions

WPHZ2000 - CAPI Modularization Demo  
Forms Answer Navigate Options Help

② of ②

(How tall are you?)  
6 feet and \_\_\_inches

EndRoster	1	SC3	1
AllAdolHH	0	SC3a	
SC0_1		SC4	6
SC0_3		SC4a	1
SC0		SC5	
SC1	70	SC7	

12113542761 8:44:00 PM Version Date: 08/22/2002 Version Time: 2:10 PM

## 5. Summary

Developing comprehensive screen design guidelines for programming CAI instruments has been an iterative process, evolving as we gain experience programming instruments in Blaise. This process has been enhanced by communication with other Blaise users, which has led to adoption of conventions used in other organizations that have design guidelines (for example, the U.S. Bureau of the Census and Statistics Canada). We hope that the discussion and

examples presented here will help other organizations think about the application and possible future enhancement of their own guidelines.

Such guidelines are meant to improve CAI screen design, making it easier to program consistently designed screens and survey instruments, and making it easier for respondents and interviewers to use those survey instruments and to record quality data. The SRC proposed guidelines are currently under review, and may change further. We will be happy to share them with other organizations when they are complete. We also welcome comments and suggestions for improvements.

## **6. Reference**

Hansen, S.E. (2003). "Evaluation of the 2003 Redesigned NHIS Blaise Instrument: Results of a Usability Test." Report Submitted to the National Center for Health Statistics. Interface Design Group, Survey Methodology Program, Survey Research Center, University of Michigan.



## *Design of Web Questionnaires*

- **Methodological guidelines for Blaise web surveys.....91**  
*Jelke Bethlehem, Statistics Netherlands & Adriaan  
Hoogendoorn, Free University, Amsterdam*



# Methodological guidelines for Blaise web surveys

*Jelke Bethlehem, Statistics Netherlands & Adriaan Hoogendoorn, Free University, Amsterdam*

## 1. Introduction

Carrying out a survey is a complex, costly and time-consuming process. Traditionally, surveys were carried out using paper forms (PAPI). One of the problems is that data collected in this way usually contain many errors. Therefore, extensive data editing is required to obtain data of acceptable quality. This consumes a substantial part of the total survey budget. Rapid developments of information technology in the last decades of the previous century made it possible to use microcomputers for computer-assisted interviewing (CAI). A computer program containing the questions to be asked replaced the paper questionnaire. The computer took control of the interviewing process, and it also checked answers to questions on the spot. Application of computer-assisted data collection had three major advantages: (1) It simplified the work of interviewers, because they did not have to pay attention any more to choosing the correct route through the questionnaire, (2) It improved the quality of the collected data, because answers could be checked and corrected during the interview, and (3) it considerably reduced time needed to process the survey data, and thus improved the timeliness of the survey results.

The rapid development of the Internet in the last decade had lead to a new type of computer-assisted interviewing: *Computer Assisted Web Interviewing* (CAWI). The questionnaire is designed as a website on the Internet. By visiting the website, respondents can answer the questions. This type of survey is usually called a *web survey*.

It is not surprising that survey organisations use, or consider using, web surveys. At first sight, web surveys seem to have some attractive advantages:

- Now that so many people are connected to the Internet, a web survey is a simple means to get simple access to a large group of potential respondents;
- Questionnaires can be distributed at very low costs. No interviewer are needed, and there are no mailing costs;
- Surveys can be launched very quickly. No time is lost between the moment the questionnaire is ready and the start of the fieldwork;
- Web surveys offer new, attractive possibilities, such as the use of multimedia (sound, pictures and movies);

Thus, web surveys are a fast and cheap means of collecting large amounts of data. According to Couper (2000) this has opened the possibility for many organisations, other than the traditional survey organisations, to conduct surveys. Many of these organisations are not aware of potential methodological dangers of web surveys. Therefore there are many good many bad surveys on the Internet. And for the respondents it is not always easy to distinguish between both types of surveys. Do they decide to participate because of societal relevance, or are they misled by the entertainment value?

An abundance of good and bad surveys on the Internet creates a serious risk of lower response rates. Potential respondents are overcrowded, and pull out. The effect is similar to that which can be observed for CATI surveys, where the survey climate is spoiled by telemarketing activities.

This paper attempts to present a number of practical guidelines for the design of web surveys. These guidelines focus on methodological aspects. Section 2 presents a general taxonomy for survey errors, and discusses which errors are relevant for web survey design. Section 3 discusses a set of guidelines that aim at reducing survey errors as described in section 2. These guidelines are, for a large part, based on a literature review. Section 4 applies the guidelines to Blaise. It is examined to what extent they can be implemented in a Blaise web survey. A proposal is made for a default questionnaire layout that as much as possible follows these guidelines.

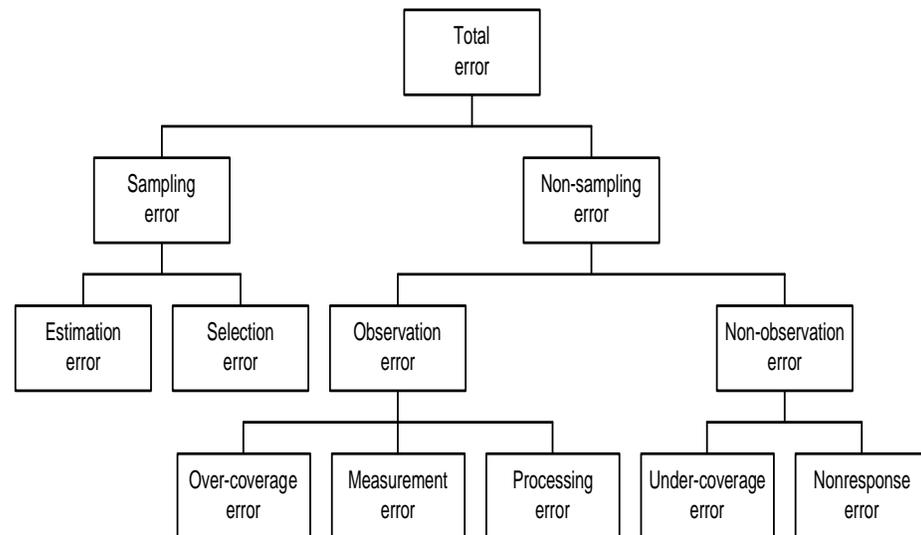
## 2. Survey errors

### 2.1. A taxonomy of errors

Usually, one of the main objectives of a sample survey is to compute estimates of population characteristics. Such estimates will never be exactly equal to the population characteristics. There will always be some error. This error can have many causes. Bethlehem (1999) gives a taxonomy of possible causes. It is reproduced in figure 2.1. The taxonomy is a more extended version of one given by Kish (1967).

The ultimate result of all these errors is a discrepancy between the survey estimate and the population characteristic to be estimated. This discrepancy is called the *total error*. Two broad categories can be distinguished contributing to this total error: sampling errors and non-sampling errors.

Figure 2.1. A taxonomy of survey errors



*Sampling errors* are introduced by the sampling design. They are due to the fact that estimates are based on a sample and not on a complete enumeration of the population. Sampling errors vanish if the complete population is observed. Since only a sample is available, and not the complete data set, to compute population

characteristics, one has to rely on an estimate. The sampling error can be split in a selection error and an estimation error.

The *estimation error* denotes the effect caused by using a sample based on a random selection procedure. Every new selection of a sample will result in different elements, and thus in a different value of the estimator. The estimation error can be controlled through the sampling design. For example, by increasing the sample size, or by taking selection probabilities proportional to some well-chosen auxiliary variable, you can reduce the error in the estimate.

A *selection error* occurs when wrong selection probabilities are used. For example, the true selection probabilities may differ from the anticipated selection probabilities when elements have multiple occurrences in the sampling frame. Selection errors are hard to avoid without thorough investigation of the sampling frame.

*Non-sampling errors* may even occur if the whole population is investigated. They denote errors made during the process of recording the answers to the questions. Non-sampling errors can be divided in observation errors and non-observation errors.

*Observation errors* are one form of non-sampling errors. These denote errors made during the process of obtaining and recording the answers. An *over-coverage error* means that elements are included in the survey not belonging to the target population. A *measurement error* occurs when the respondent does not understand the question, or does not want to give the true answer, or if the interviewer makes an error in recording the answer. Also, interview effects, question wording effects, and memory effects belong to this group of errors. A measurement error causes a difference between the true value and the value processed in the survey. A *processing error* denotes an error made during data processing, e.g. data entry.

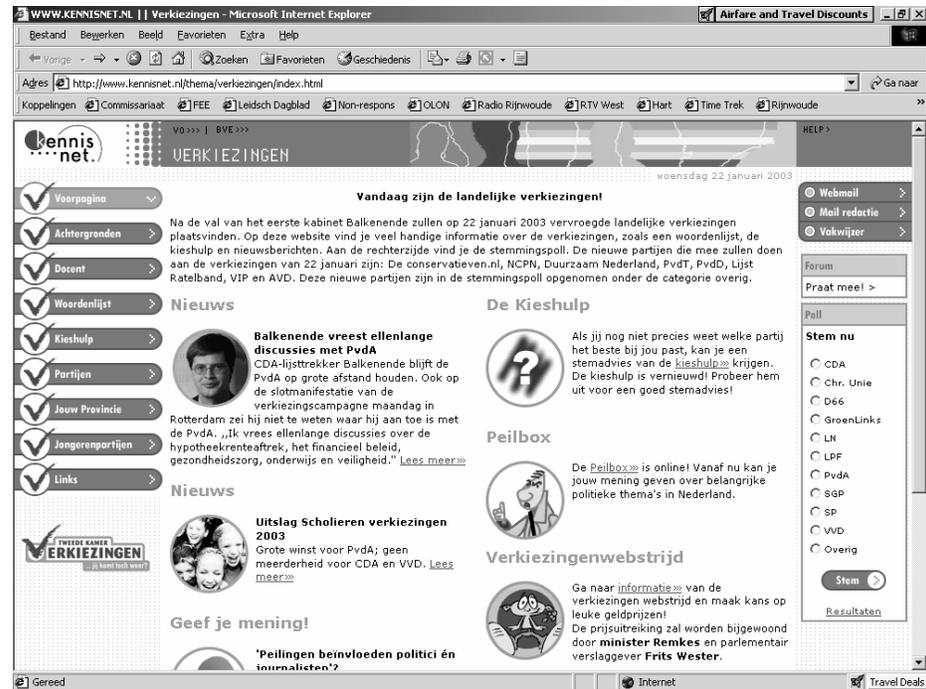
*Non-observation errors* are errors made because the intended measurements could not be carried out. *Under-coverage* occurs when elements in the target population do not appear in the sampling frame. So, representatives of these elements can never be contacted. Another non-observation error is *nonresponse*. It is the phenomenon that people selected in the sample do not provide the required information.

The taxonomy above makes clear that a lot can go wrong during the process of collecting survey data, and usually it does. Some errors can be avoided by taking preventive measures at the design stage. However, some errors will remain. Therefore, it is important to check the collected data for errors, and where possible, to correct detected errors. This activity is called *data editing*. Data editing procedures are not able to handle every type of survey error. They are most suitable for detecting and correcting measurement errors, processing errors, and possibly over-coverage. Phenomena like selection errors, under-coverage, and nonresponse require a different approach. This approach often leads to the use of adjustment weights in estimation procedures, and not to correction of individual values in records.

## **2.2. Probability sampling**

In their fundamental paper, Horvitz and Thompson (1952) show that only unbiased estimates of population characteristics can be computed if every element in the population has a non-zero probability of selection, and that the researcher knows these probabilities. Furthermore, only under these conditions, the accuracy of estimates can be computed.

Figure 2.2.1. A survey just for entertainment



Many surveys on the web are not based on probability sampling. Couper (2000) distinguished three groups of such surveys:

The first group consists of surveys that are primarily intended for entertainment purposes. They often take the form of simple polls. There is often no control over who responds, and even how many times one responds. A typical example is the survey on Kennisnet (Knowledge net). This is a Dutch website for all involved in education. More than 11,000 schools and other educational institutes use this website. Figure 2.2.1 shows the page about the general elections of 22 January 2003. The right-hand part of the screen contains a small survey where visitors can vote.

Table 2.2.1 contains both the results of this poll on the morning of the Election Day, and the official results of the election.

Table 2.2.1. The results of an 'entertainment' survey

Party	Survey result	Election result	Difference
CDA (Christian democrats)	19.2 %	28.6 %	- 9.4 %
LPF (populist party)	12.0 %	5.7 %	+ 6.3 %
VVD (liberals)	16.1%	17.9 %	- 1.8 %
PvdA (social democrats)	8.8 %	27.3 %	- 18.5 %
SP (socialists)	14.4 %	6.3 %	+ 8.1 %
GL (green party)	17.0 %	5.1 %	+ 11.9 %
D66 (liberal democrats)	3.1 %	4.1 %	- 1.0 %
Other parties	9.4 %	5.4 %	+ 4.0 %

The survey figures are based on 17,574 people. This is not a small sample. Nevertheless, it is clear that the survey results are no way near the true election results.

The second group of non-probability sample surveys is self-selected web surveys. This approach uses open invitations on portals, frequently visited web sites, or dedicated survey sites.

An example of this approach is the election site of the Dutch Television channel RTL 4, see figure 2.2. It resembles to some extent the ‘entertainment’ web survey, but has a slightly more serious appearance. Again, the survey researcher has no control at all over who is voting. There is some protection, by means of cookie, against voting more than once. However, this also has the drawback, that only one member of the family can participate.

The conclusion from this analysis can be that probability sampling is an important prerequisite for making reliable inference on a target population.

Figure 2.2. A self-selection survey



Table 2.2 shows the survey results at noon on the day of the general elections. Figures are based on slightly over 10,000 votes. Deviations between estimates and true figures are large, particularly for the large parties. Note that even the large sample size of over 10,000 people does not help to get accurate estimates.

Table 2.2.2. The results of a self-selection survey (percentages)

Party	Survey result	Election result	Difference
CDA (Christian democrats)	16 %	28.6 %	+ 12.6 %
LPF (populist party)	8 %	5.7 %	-2.3 %
VVD (liberals)	25 %	17.9 %	+ 7.1 %
PvdA (social democrats)	27 %	27.3 %	+ 0.3 %
SP (socialists)	7 %	6.3 %	- 0.7 %
GL (green party)	6 %	5.1 %	- 0.9 %
D66 (liberal democrats)	5 %	4.1 %	-0.9 %
Other parties	6 %	5.4 %	-0.6 %

The third group of non-probability sample surveys distinguished by Cooper (2000) consists of volunteer panels of Internet users. This approach creates a volunteer panel by wide appeals on well-visited sites and Internet portals. At time of registration basic demographic variables are asked. A large database of potential respondents is created in this way. For future surveys, samples are selected from this database. Only registered people can participate in these surveys.

**Table 2.2.3. The results of a self-selection survey (seats in parliament)**

Party	Survey result	Election result	Difference
CDA (Christian democrats)	42	44	- 2
LPF (populist party)	6	8	- 2
VVD (liberals)	28	28	
PvdA (social democrats)	45	42	+ 3
SP (socialists)	11	9	+ 2
GL (green party)	6	8	- 2
D66 (liberal democrats)	5	6	- 1
Other parties	7	5	+ 2

An illustrative example of this approach was the general election survey of the Dutch commercial television SBS6. People were invited to participate in the survey. Those visitors of the site accepting the invitation were asked a number of questions related to socio-demographic characteristics and voting behaviour in the previous election. From the set of people that was obtained in this way, samples of size 3000 were selected. Selection was carried out such that the sample was representative with respect to the social-demographic and voting characteristics. And these were asked for their voting behaviour in the coming election. Table 2.2.3 shows some results. The survey took place on the day before the general elections.

**Table 2.2.4. The results of a probability sample survey (seats in parliament)**

Party	Survey result	Election result	Difference
CDA (Christian democrats)	42	44	- 2
LPF (populist party)	7	8	- 1
VVD (liberals)	28	28	
PvdA (social democrats)	43	42	+ 1
SP (socialists)	9	9	
GL (green party)	8	8	
D66 (liberal democrats)	6	6	
Other parties	7	5	+ 2

Although attempts have been made to create a representative sample, the results differ still considerable from the final result. A much better would have been obtained with a true probability sample. Table 2.2.4 shows the results of a survey based on a true probability sample. The television channel Nederland 1 carried it out. A sample of size 1200 had been selected by means of random digit dialling. Compared to table 2.2.3 the results in table 2.2.4 are much better. The *Mean Absolute Deviation* (MAD) in table 2.2.3 is 1.75. In table 2.2.4 the values of this quantity has reduced to 0.75.

The conclusion from the analysis above can be that probability samples are a vital prerequisite for making proper inference about the target population of a survey. Probability sampling starts with the choice of a sampling design. This is the

definition of the selection mechanism applied. It should give a non-zero chance of selection to every element in the target population. A straightforward example of a sampling design is *Simple Random Sampling*, where each element has the same probability of selection.

Since elements are drawn at random in probability sampling, each new selection of a sample will result in a different set of elements. Therefore, the value of the estimator will also differ. This results in the *estimation error*. It only depends on the sampling design used, and not on the mode of data collection. So, the choice for a web survey has no impact on the selection error.

To actually select the sample according to the specifications of a sampling design, a *sampling frame* is required. This is a list, map, or other complete specification of all elements in the target population. The sampling frame must provide information on how to contact selected elements. Examples of sampling frames are a list of addresses, a list of phone numbers, a random digit dialling procedure, or a map containing the geographical positions of households.

If a proper sampling frame is used, the actual selection probabilities of the elements are equal to the planned selection probabilities as planned in the definition of the sampling design. However, there are situations in which is not the case. For example, when a household survey sample is selected from an address list, small business owners appear twice in the frame: once with their home address, and once with the business address. If no correction takes place for this phenomenon, these households will have a probability of being selected that is twice as big as planned. Using the wrong probabilities in the estimation procedure will result in biased estimates of population characteristics. This results in a *selection error*.

The situation is even worse if the Internet is used as a selection mechanism. This happens when surfers on the World Wide Web are invited to participate in a survey through a banner, popup window, or other announcement they happen to see on a website. The probability of being confronted with such an invitation is unknown, and the probability of accepting it is unknown. Therefore, it is impossible to compute unbiased estimates of population characteristics. The opinion poll of the Dutch television channel RTL 4 that was mentioned in section 2.2 is a good example of such a self-selection survey.

### 2.3. Coverage problems

The collection of all elements that can be contacted through the sampling frame, is called the *frame population*. Since the sample is selected from the frame population, conclusions drawn from the survey data will apply to the frame population, and not to the target population. Coverage problems can arise when the frame population differs from the target population.

*Under-coverage* occurs when elements in the target population do not appear in the frame population. These elements have zero probability of being selected in the sample. Under-coverage can be a serious problem for web surveys. If the target population consists of all people with an Internet connection, there is no problem. However, usually the target population is more general than that. Then, under-coverage occurs due to the fact many people do not have access to the Internet.

An analysis of data (source: [www.cbs.nl](http://www.cbs.nl)) about the availability of an Internet connection at home indicates that this is unevenly distributed over the population in November 2001. Figure 2.3.1. shows the distribution by sex. Clearly, more males than females have access to the Internet.

Figure 2.3.1. Having an Internet connection at home by sex.

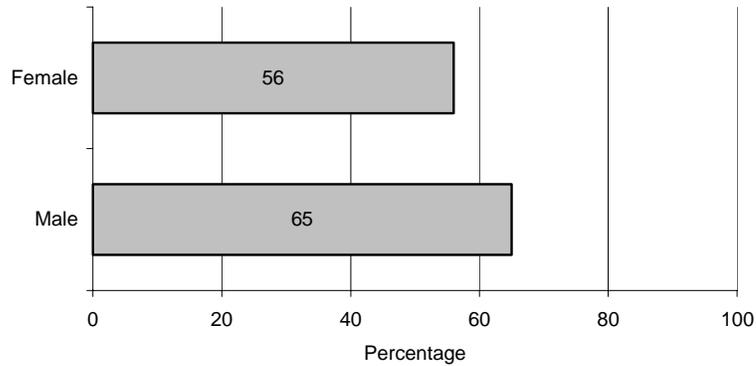


Figure 2.3.2 contains the percentage of people having an Internet connection by age group. The percentages of people having access to Internet at home decreases with age. Particularly, the people of age 55 and older will be very much under-represented when the Internet is used as a selection mechanism.

Figure 2.3.2. Having an Internet connection at home by age.

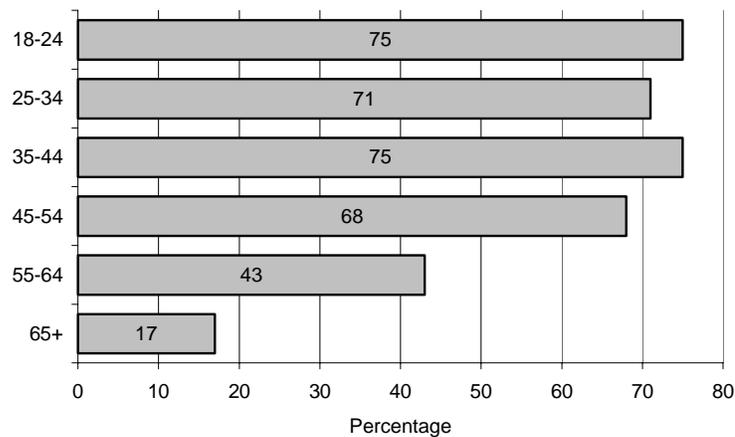
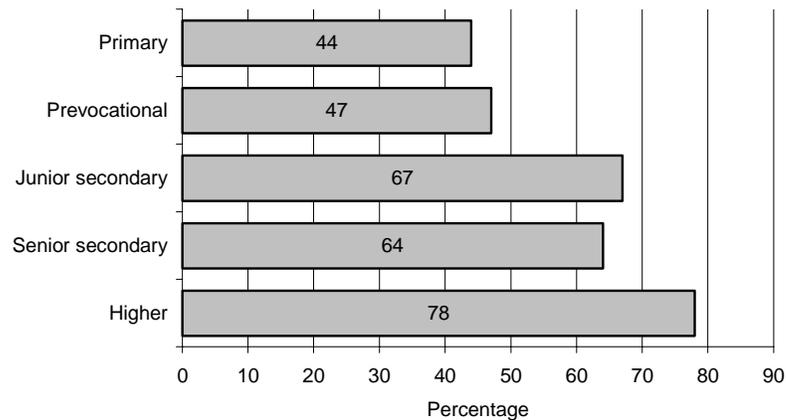


Figure 2.3.3 contains the percentage of people having an Internet connection by level of education.

Figure 2.3.2. Having an Internet connection at home by level of education.



It is clear that people with a higher level of education have more access to the Internet than people with a lower level of education.

The Social and Cultural Planning Agency (SCP) carried out a survey about the use of digital media among a sample of students in secondary education, see De Haan et al. (2002). It turned out that 97% of the students have a PC at home, and that 84% have an Internet connection. However, these percentages are much lower among students with a foreign background. 15 % of the students with a Moroccan or Turkish background have computer at home. And 12% of the students with a former Dutch Indies background have no computer at home.

The results described above are in line with the findings of authors in other countries. See e.g. Couper (2000), and Dillman and Bowker (2001). It is clear that when the Internet itself is used as a selection mechanism, certain specific groups will be under-represented.

#### **2.4. Nonresponse problems**

Nonresponse can be defined as the phenomenon that elements (persons, households, companies) in the selected sample do not provide the requested information, or that the provided information is useless. The situation in which all requested information on an element is missing is called *unit non-response*. If information is missing on some items only, it is called *item non-response*.

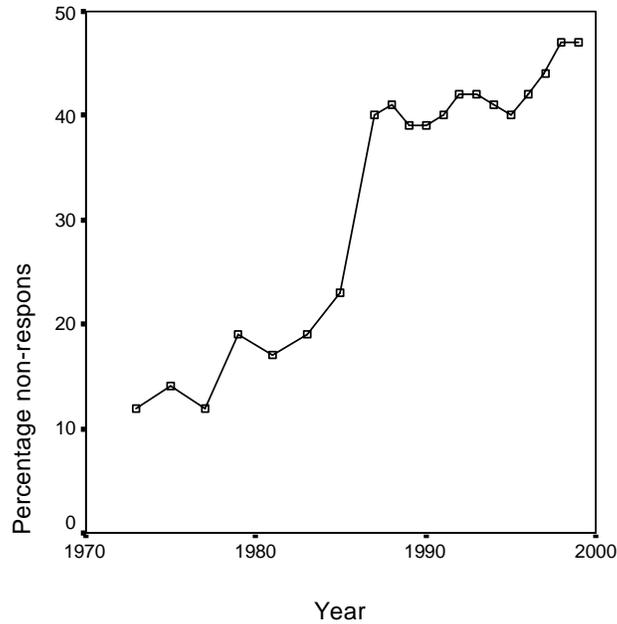
Due to nonresponse the sample size is smaller than expected. This leads to less accurate, but still valid, estimates of population characteristics. This is not a serious problem. It can be taken care of by taking the initial sample size larger. A far more serious problem caused by nonresponse is that estimates of population characteristics may be biased. This situation occurs if, due to non-response, some groups in the population are over- or under-represented, and these groups behave differently with respect to the characteristics to be investigated.

Nonresponse has a negative impact on the quality of population estimates. One of the most serious effects of nonresponse is that these estimates may be biased. This means, that in fact the wrong conclusions are drawn from the survey results. Nonresponse constitutes a serious threat for the quality of survey results, particularly when nonresponse rates are high.

Nonresponse rates are high in The Netherlands. As an example, figure shows the rate of nonresponse for the Dutch Labour Force Survey. The percentage of nonresponse has been rising ever since 1970, and now approaches the 50%.

The magnitude of the non-response is determined by a large number of factors, including the topic of the survey, the target population, the time period, the length of the questionnaire, etc. The mode of data collection is also an important factor. Generally, face-to-face surveys and telephone surveys have lower nonresponse rates than surveys in which respondent are confronted with self-administered questionnaires. The role of interviewers is crucial in this. They can play an important role in persuading people to participate, and in assisting respondents to answer questions. Such assistance is not available for self-administered questionnaires.

Figure 2.4.1. Nonresponse rates of the Dutch Labour Force Survey



A web survey questionnaire is a form of self-administered questionnaires. Therefore, web surveys have to potential of high nonresponse rates. An additional source of nonresponse problems is technical problems of respondents having to interact with the Internet, see e.g. Couper (2000), Dillman and Bowker (2001), Fricker and Schonlau (2002), and Heerwegh and Loosveldt (2002).

Slow modem speeds, unreliable connections, high connection costs, low-end browsers, and unclear navigation instructions may frustrate respondents. And this often results in respondents discontinuing the completion of the questionnaire. In order to keep the survey response up to an acceptable level, every measure must be taken to avoid these problems. This requires a careful design of web survey questionnaire instruments.

### 3. Design of web survey questionnaire instruments

#### 3.1. Introduction

A proper design of a survey questionnaire instrument is crucial for the quality of the data that is collected. Layout of the questionnaire, question wording, type of expected answer, ease of navigation, speed of processing, and browser compliance are all factors that may have an impact on measurement errors and response rates. Therefore, careful attention must be paid to these aspects in the design stage of the web survey instrument.

This section addresses a number of these aspects. Based on a literature review a number of guidelines are formulated. It is shown what these guidelines would lead to in a practical example. Then it is explored to what extent such a design can be realised within the Blaise System. As an example we use a simple questionnaire that attempts to survey local radio listening behaviour. Appendix A contains a paper copy of the questionnaire form.

### 3.2. General design issues

Web questionnaires are created with HTML. The ongoing development of this language has provided designers with all kinds of possibilities to use colours, sound, graphics, animation, and embedded programs (in Flash, JavaScript, Java, PHP, VBScript, etc). This allows for creating very fancy questionnaires. However, there are also drawbacks. One is that it makes the questionnaire program much bigger. Downloading these programs from servers to respondents takes much more time. This may lead to dropout of respondents with slow modems, unreliable connections, and high connections costs. Moreover, not every browser in use may support all these fancy features.

Dillman et al. (1998) compared a fancy web questionnaire design with a plain one. The plain questionnaire gave better results. There was a higher response rate. And a smaller number of people dropped out of questionnaire somewhere halfway its completion. Also, it took respondents less time to complete the plain questionnaire. This all means that we should apply the KISS principle (Keep it Simple, Stupid) to web questionnaire design. The focus should be on the essential parts of the questionnaire. All kinds of *questionnaire junk* should be avoided. Here is an analogue to statistical graphics. For example, Tufte (1983) describes how too much *chart junk* makes interpretation of graphs much more difficult. He introduces the *data-ink-ratio* as a measure of the amount of chart junk in a graph. It is defined as the amount of ink used to draw the essential part of the graph divided by the total amount of ink used. For a good graph, the data ink ratio should be close to 1. Likewise, the data-ink-ratio could be applied to the design of a web questionnaire. Also here, it could be a good idea to strive for a data-ink-ratio of 1.

Application of the KISS principle, going for a data-ink-ratio of 1 means that colours, graphics, animation, sound, and embedded programs are only used where it is really functional and cannot be avoided. Particularly in mixed-mode surveys, where the web questionnaire is just one of the modes of data collection, it is important that questionnaire forms of the various modes resemble one another. Therefore, it would be a good idea to let the web questionnaire form resemble the paper questionnaire form as much as possible.

This all leads to a guideline to the guideline that web survey questionnaires should be designed plain and sober. They should look like their paper analogues. Unnecessary and distracting questionnaire junk like picture, sound and animation must be avoided.

The texts on the screen should be well readable. The choice of font is important. Generally, sans-serif fonts like Arial, Verdana and Helvetica are preferred. The font size should be large enough to keep text readable on screens of various resolutions. At least, they must be readable for the most common screen resolutions (800 × 600 and 1024 × 768).

The layout of the questionnaire screens must not depend on the screen resolution used. Dillman and Bowker (2001) describe web surveys where a different screen resolution results in different ways in which the information is presented on the screen. This may easily lead to measurement errors. One way to get a resolution independent design is to make use of the <table> tag in HTML. Simply put all questionnaire information in a fixed-width table. This trick has been applied in figure 3.2.1. The questionnaire form is designed as a white page on a grey background. The change in screen resolution from 800 × 600 to 1024 × 768 only causes the information to be displayed enlarged. It has no impact on the layout.

Figure 3.2.1. The same web questionnaire screen for different screen resolutions

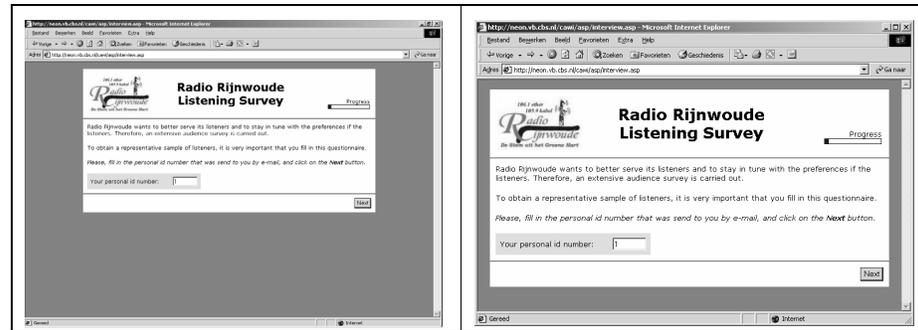


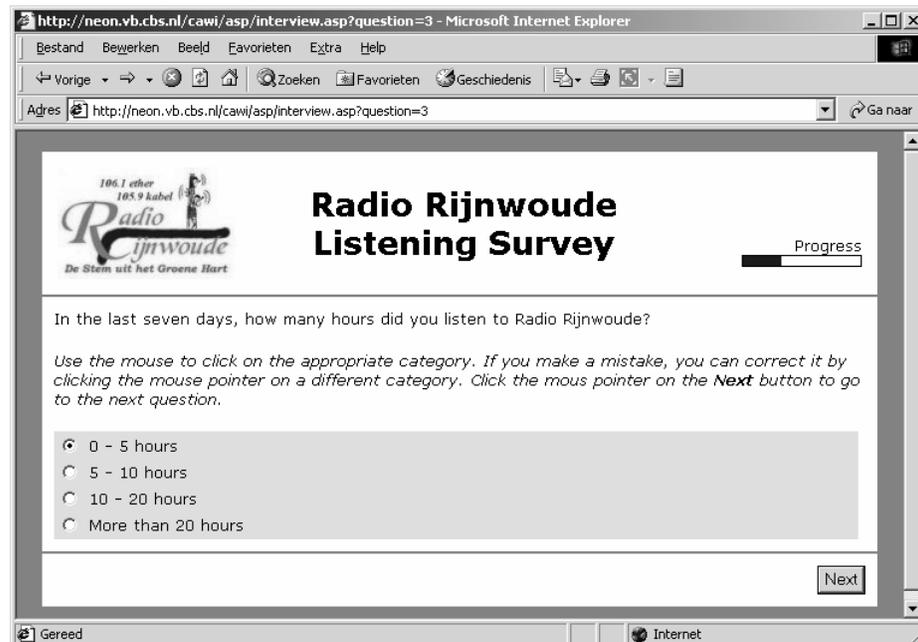
Figure 3.2.1 also shows another important guideline. That is that the use of scrollbars should be avoided as much as possible. Completion of the questionnaires should be as simple as possible. The actions required to answer a question, and to go to the next question, must be kept to a minimum. And it also must be clear what actions are required to realise this. For example, having to scroll down to find a **Next** button may lead to confusion, frustration, and even to dropping out of the session.

### 3.3. Design of item screens

The layout of the various screens of the web questionnaire must be clear, easy to understand, and easy to handle. The various items on the screen must be presented in a standard way. We advocate a structure that consists of three parts.

The top part contains a limited amount of general information. There can be a logo at the upper left indication the organisation that is carrying out the survey. There may be contact information (telephone number or e-mail address). The upper middle part contains the name of the survey.

Figure 3.3.1. A question screen of web questionnaire



The upper right is reserved for some kind of progress indicator. This progress indicator shows respondents where they are in the completion process. It can be a simple text like a percentage, or some kind of graphical display like a progress bar.

Couper et al. (2001) use a pie chart. They show that such a device reduces item nonresponse. Care must be taken that including fancy progress indicators do not increase downloading times, which can have a negative effect on response. Also no use must be made of embedded programs that are not supported by browsers, or require downloading and installing special viewers. In our example we use a simple progress bar, see e.g. figure 3.2.1 or 3.3.1.

The middle part of the screen contains the current question to be answered. Generally, the question information consists of three parts:

- The text of the question;
- Instructions how to answer the questions, and how to proceed to the next question;
- The answer to the question. The format of this part depends on the type of question (open, closed, check-all-that-apply, numeric, etc).

The *text of the question* is displayed in a normal font. To give the text of the question more emphasis, and to distinguish it from the rest of textual information on the screen, one may want to consider displaying the text in boldface, or the use a different colour. To emphasize certain parts (words) in the text one may consider using a different colour. However, one should always be aware of the fact that the colour-blind persons may not be able to distinguish different colours.

It must always be clear for respondents which actions they have to take in order to answer a question and to proceed to the next questions. Dillman et al (1998) describe several ways of doing this.

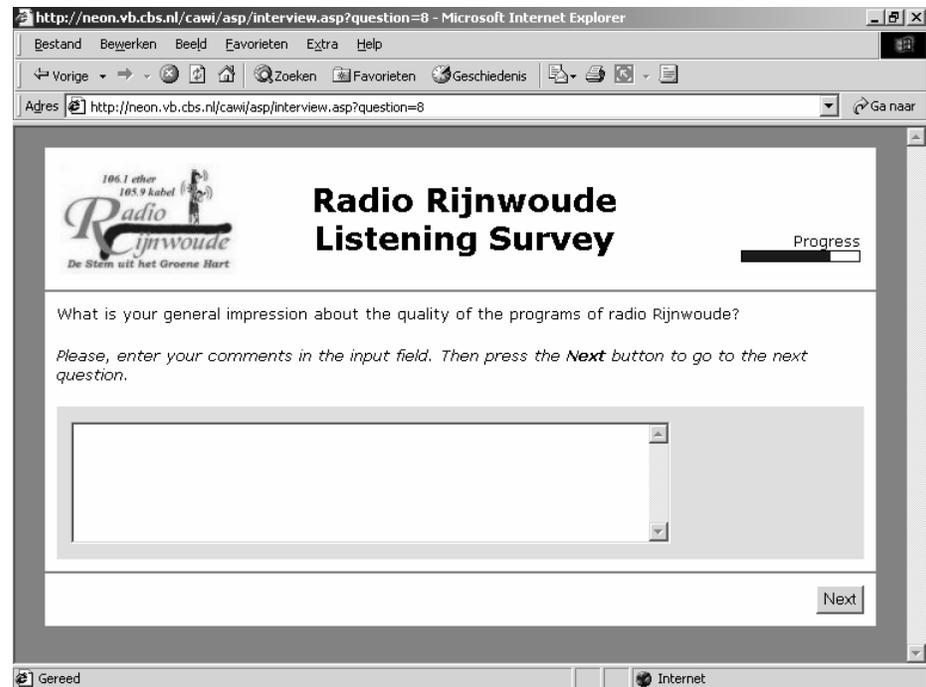
- Add instructions to every question on the screen. An example of this approach is given in figure 3.3.1. To distinguish instructions from the question text, the instructions are displayed in italics. This seems to be more or less a convention. However, one should realise that text in italics may not always be very readable on low-resolution screens. Another option could be to display the question text in boldface and the instruction in normal typeface.
- Only add instructions to the first few questions of the questionnaire. In any case, there must be instructions for any new type of question that appears for the first time. This will lead to less cluttered question screens. But it may lead to problems later in the questionnaire for those that have forgotten the instructions.
- Add an initial screen to the questionnaire where respondents are asked whether or not they are experienced computer users. For experienced users, this is followed by a questionnaire without instructions. And those who indicate they are not experienced will get a questionnaire with instructions.
- Add a window that ‘floats’ on top of the questionnaire screen, and that contains instructions for answering the question. This is a feasible option but may require (too much) additional processing time.
- Add an initial screen with general instructions on answering questions. The respondents will see this screen only once. Dillman et al. (1998) do not consider this desirable practice from the point of view of cognitive or learning practice.

The third part of the question consists of the area in which the respondents must give their answers. The structure and layout of this part depends on the type of question.

For an *open question*, there will be an area in which some text can be entered. See figure 3.3.2 for an example. It is not always clear for respondents what kind of input is expected from them: only a few words, a paragraph, or maybe even a small story. Sometimes, they only see a small input field, and they are not aware of the possibility that a lot of text can be entered by simply scrolling the text. It could help to give the input field a size (in terms of numbers of rows and columns) that more or less reflects the size of the expected answer.

In figure 3.3.2 some text is expected that may not exceed 300 characters. Since the field is 60 characters wide, the number of rows is made equal to 5.

Figure 3.3.2. An open question



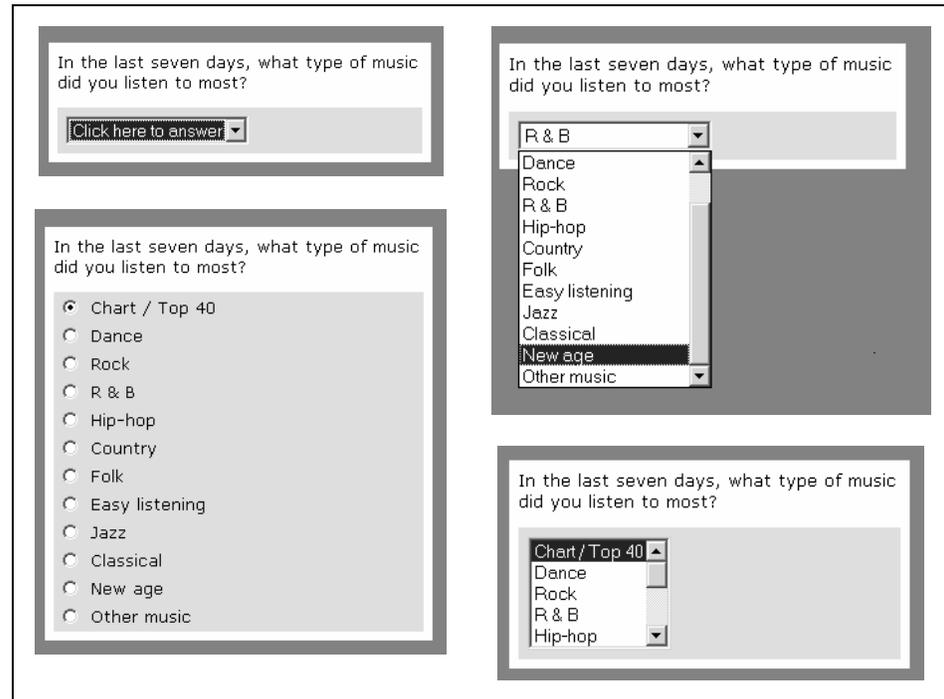
For a closed question, HTML offers various ways of displaying the possible answer categories, and selecting one of them. Examples are displayed in figure 3.3.3. The solution on the lower left uses radio buttons, and the other ones use drop-down boxes. The advantages and disadvantages of both approaches are discussed by Dillman (2000), Heerwegh and Loosveldt (2002), and Couper (1999).

The advantage of radio buttons is that a closed question looks very much like its paper questionnaire analogue. An answer is selected by clicking the appropriate radio button. Clicking another radio button can change an answer. But an answer can never be erased any more. When using radio buttons, all possible answers are visible. This helps the respondent to select the proper answer. A disadvantage of radio buttons is that they can take a lot of space when the list of possible answers is long.

Drop-down boxes usually show only one item (the first one, or the selected one). To select an answer, the respondent must click on the drop-down box. Then the list of possible answers becomes visible. If this list is very long, it only becomes partially visible. In this case, the respondent may have to use the scrollbar to display that part of the list containing the appropriate answer. This is illustrated in figure 3.3.3. The top left drop-down box initially only shows the first item of the list ('click here to answer'). After clicking the drop-down box, the list of possible answers unfolds. The result is display top right. A number of items becomes

visible, but not all of them. The bottom right show a drop-down box with a different initial state. Always five items are visible. The scrollbar must be used to show another part of the list.

**Figure 3.3.3. Various ways of displaying a closed question**

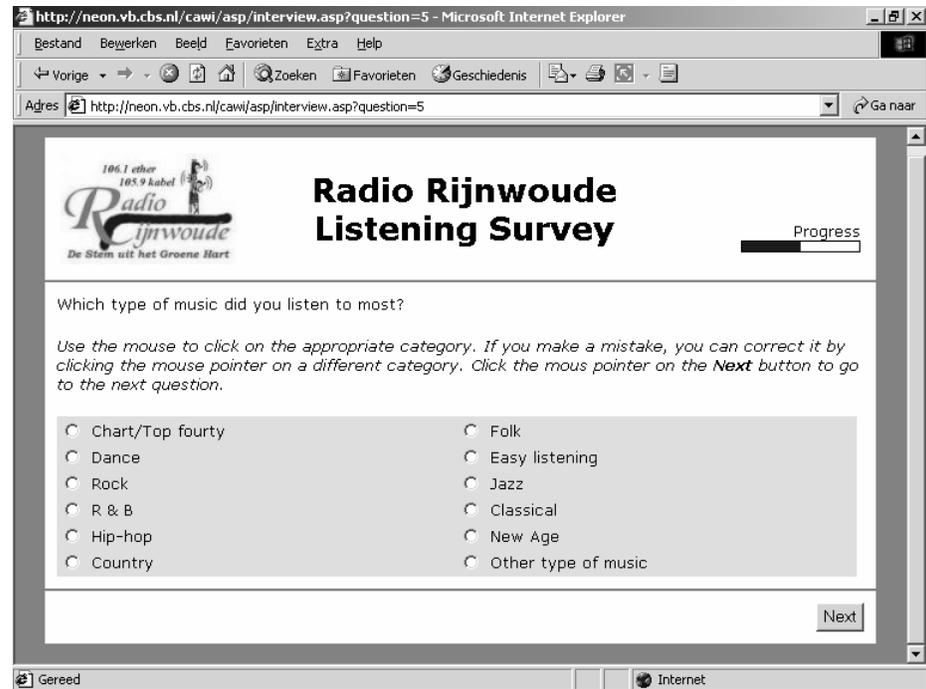


Drop-down boxes have the advantage over radio buttons that it takes less time to download them from the server. However, drop-down boxes also have severe disadvantages. One is that they are more difficult to handle by respondents. They have to perform three actions to select an answer: clicking the box, scrolling to the right answer, and clicking the right answer. Selecting the right answer from a set of radio buttons only requires one click.

Another serious disadvantage has been investigated by Couper (1999). He shows that respondents tend to select more the initially visible options than the non-visible ones. Also Heerwegh and Loosveldt (2002) express concerns about mode effects caused by list boxes. This all indicates that one should use radio buttons as much as possible.

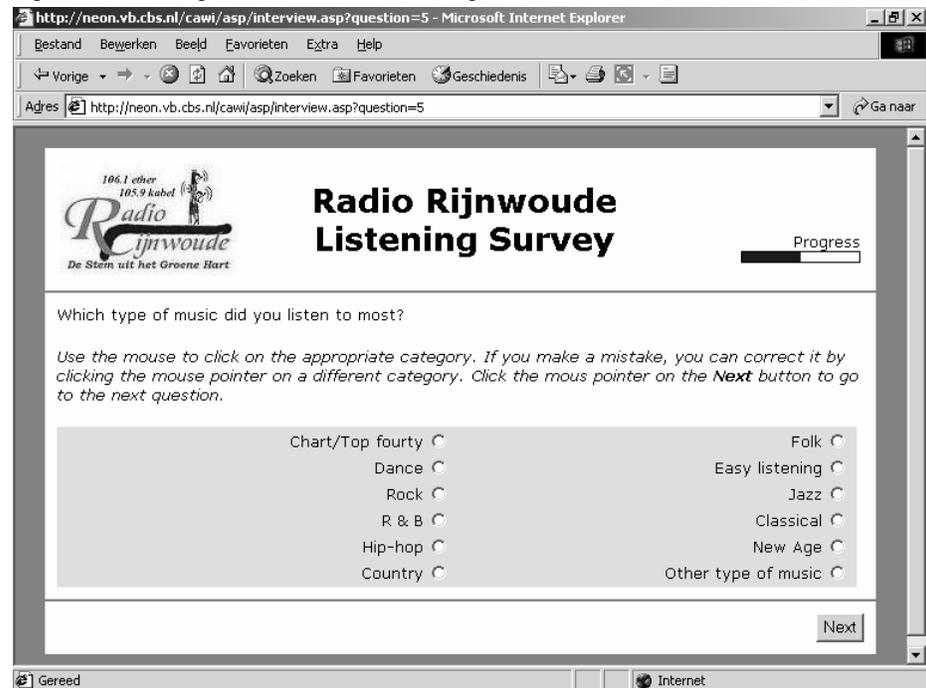
Attention should be paid to displaying radio buttons when the list of possible answers. As was mentioned earlier, all question information should be visible on the screen, and not require any scrolling. Therefore, it is better to split a long list of radio buttons over a number of columns. Figure 3.3.4 gives an example. Here the answers are distributed over two columns. To give a visual clue that these two lists belong together, Dillman et al. (1998) suggest to put them in a kind of box. This is realised in figure 3.3.4 by means of a grey background.

Figure 3.3.4. A closed question with many possible answers



It is very common for paper questionnaires to have the answer boxes to the left of the descriptions of the possible answers. This principle is also being implied in many web surveys, see e.g. figure 3.3.4.

Figure 3.3.5. Putting the answer boxes to the right



However, having to answer closed questions with a mouse in this layout requires substantial mouse movement. This not only takes time, but may also be a source of errors. Bowker and Dillman (2000) describe a small study investigating effects of placing answer boxes to the right of the text. The basic idea is that this layout would reduce the efforts required to answer the question, and to proceed to the next

question. Figure 3.3.5 shows an example of such an approach. Results of their study seem to indicate that changing the position of the answer boxes has no effect on item nonresponse rates and quality of the answers given. However, less-experienced computer users seem to be more confused about how to answer the questions. Although their study had some limitations, results show no clear advantages of putting answers boxes to the right.

A question type that is known to have measurement error problems, is the “check-all-that-apply” question. Respondents are asked to check all items that apply them in a sometimes very long list. The problem is that often respondents stop checking items when they think they have checked enough answers. There is substantial evidence that respondents do not read all possible answers before going to the next question, see e.g. Dillman et al. (1998).

Figure 3.3.6 shows an example of a check-all-that-apply question. Particularly when the list of possible answers is long, it is easy to make mistakes, like checking the wrong category, or forgetting to check categories.

Figure 3.3.7 shows a different format for a check-all-that-apply question. Each check box is replaced by two radio buttons, one for “yes” and one for “no”. This approach forces respondents to perform an action for each category. It encourages them go down the list item-by-item, and give an explicit answer for each item. This approach reduces measurement errors, but also has a drawback: it requires a lot more time to answer the question, and this may increase nonresponse rates.

Figure 3.3.6. A check-all-that-apply question that uses check-boxes

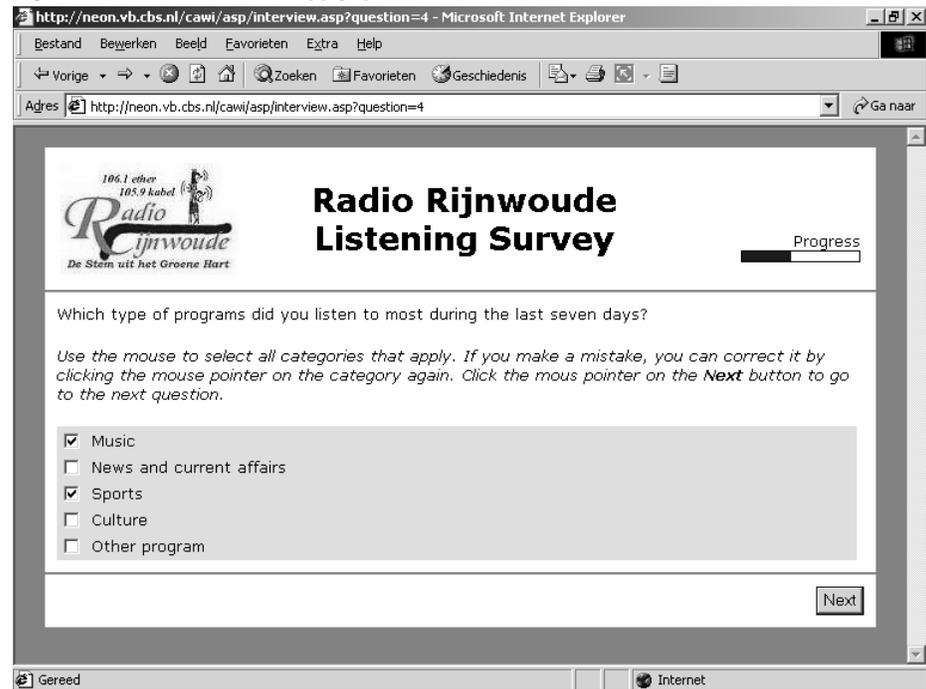
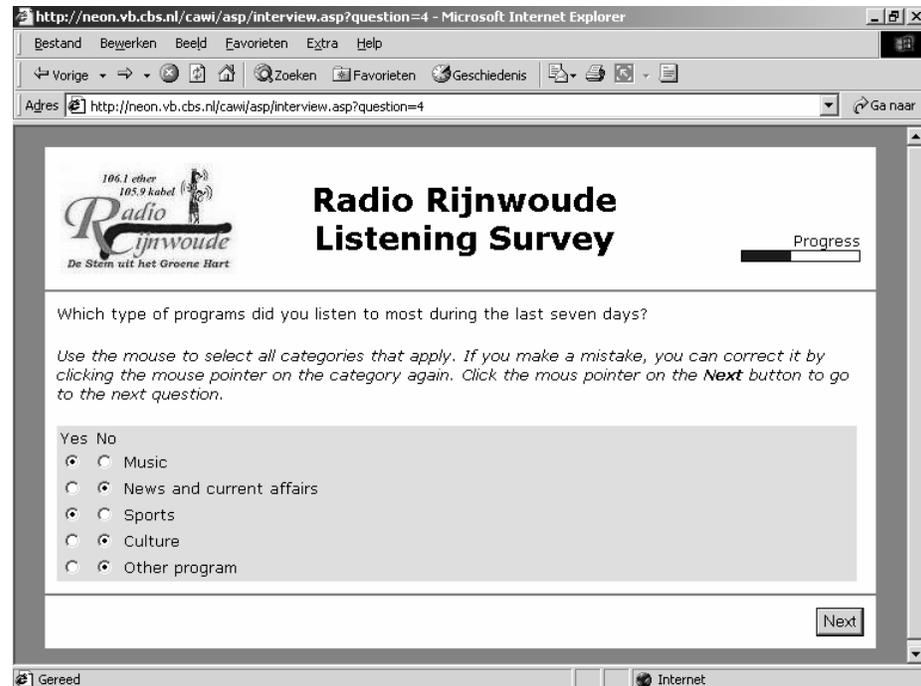


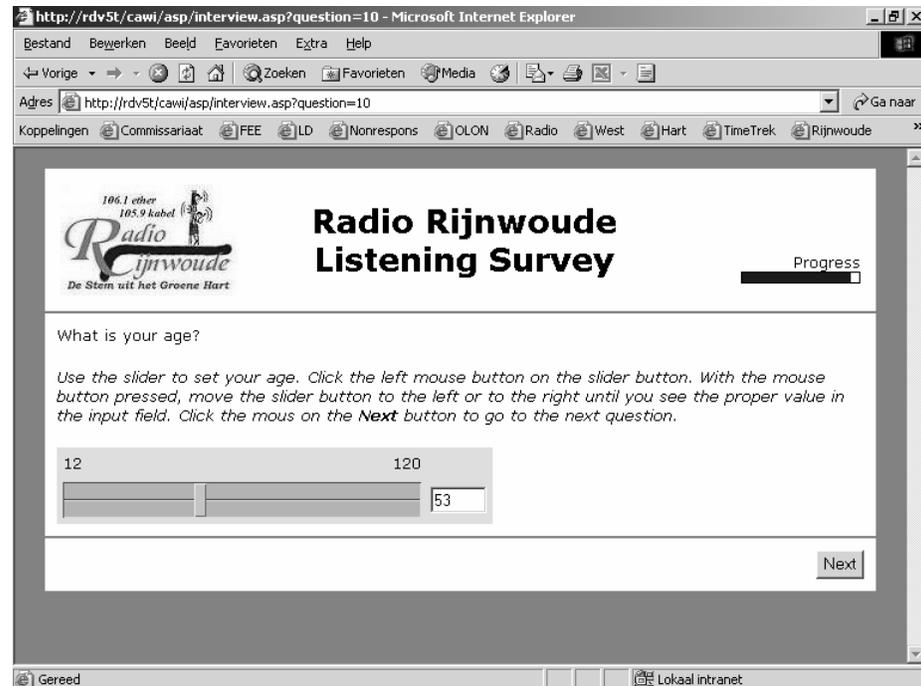
Figure 3.3.7. A check-all-that-apply question that uses radio buttons



A final question type that is briefly mentioned is the *numerical question*. The display format of such a question depends upon the range of valid answers. If the range is limited, the numerical question can be treated as a closed question. This is the easiest way to answer, and reduces measurement errors as much as possible. This approach can typically be applied for questions like the size of the household. If the value range is so large it cannot be handled by a set of radio buttons, an open question type input format has to be used. This means offering an input field in which the respondent must type the answer. Typing is a source of errors. It is easy to enter non-numerical information or values out of the defined range. This requires extensive error checking. Couper et al. (2001) showed that use of input fields causes higher item nonresponse rates.

One can think of different ways of letting respondents specify numerical answers within a specified range. One way is making use of so-called slide bars. Figure 3.3.8 contains an example of such an approach. The respondents specify their answer by dragging the slide bar button to the left or right. One advantage of this technique is that no typing errors can be made. Another advantage is that no out of range values can be entered. One disadvantage is that a starting value has already been specified. This makes it very easy for the respondent to click the Next button without paying attention to the question and its answer. Another disadvantage is that the slide bar is no standard feature in HTML. It has to be programmed, and the code has to be embedded in the HTML page. The slide bar in the example of figure 3.3.8 was programmed in JavaScript. Not every browser may support JavaScript, and it also increases transmission times.

Figure 3.3.8. Entering a numerical answer using a slide bar



In a mixed-mode survey, questionnaires for the various modes should resemble one another as much as possible. This helps to avoid mode effects. This would imply to implement numerical questions in the traditional way, using a simple input field.

The third, and final, part of the questionnaire screen is the bottom part. It contains the navigation buttons. Our example contains just one such button, and that is the Next button. If this button is clicked, the system first carries out various checks. Only if a question has been answered, and the answer is within the defined domain, the respondent is allowed to go to the next question. In case something is wrong, a clear error message will appear, explaining what is wrong, and how the error can be corrected.

### 3.4. Incorporating checks

Experience with various types of CAI surveys has shown that the quality of the collected information can be improved by carrying out *active error checking*. It means that all kinds of range checks and consistency checks are carried out the moment the relevant information has been entered by the respondent or interviewer. This feature should also be implemented in web surveys.

For closed questions, one radio button must have been checked. If not, it is not possible to proceed to the next question.

For numeric questions with an input field, it must be checked whether a numerical answer has been entered, and that this answer is within the valid range. For numeric questions with a slide bar or similar device it must be checked whether the bar was moved.

For check-all-that-apply questions with checkboxes, no checks can be carried. For check-all-that-apply questions with yes/no radio buttons, it must be checked that for each answer one of the two radio buttons has been checked.

For open questions, no checks are carried out. However, it is possible for specific open questions to require the respondent to enter at least something.

Experience with e.g. Blaise surveys has shown that data quality can be improved even more by carrying out consistency checks. Whenever relationships between survey variables impose restrictions on the values of answers given, this should be checked. If an inconsistency is detected, it should be reported on the screen. The respondent should be given the possibility to modify one or more of the answers of questions involved in the check. Usually, this means jumping back to the point in the questionnaire to a previous question. The interview software must have the facilities to do this. In case an error has been detected, a list of questions involved must appear on the screen. Then the respondent must be able to select the question in the list to jump back to.

### **3.5. Navigation issues**

With respect to navigation through the questionnaire, two approaches are possible. One is called active routing, and the other one passive routing.

*Active routing* means that the routing through the questionnaire is forced upon the respondents. They are not free in answering the questions in the order they like. Once they have answered a question, they can only go to the next question on the routing path through the questionnaire. It is not possible to go to questions that are not on the route. The only exception is that they can go backwards to the previous question. This allows for correcting mistakes that might have been made in answering a question.

The active routing approach guarantees that respondents only answer relevant questions, and that they are not confronted irrelevant questions. This also avoids dropout of respondents just because they see the mere quantity of potential questions that can be asked. A disadvantage of this approach is that respondents may feel too limited in their movements, and they may not like that feeling.

The passive routing approach lets the respondents completely free in their movements through the questionnaire. They can go to any question they like and answer any question they like. Obviously this gives more freedom. But there also is a big risk of accidentally skipping relevant questions, and answering irrelevant ones. This may lead to item nonresponse, and also dropout because of getting the impression that too many questions must be answered.

From a data quality point of view, we advocate the active routing approach. This means that the web interviewing software must be in complete control of navigation. Going to a next question, or possibly to a previous one, should only be possible to click on the appropriate buttons at the bottom of each question screen. One should take care to de-activate other navigation facilities of the browser, like the previous and next icons in the browser toolbar.

## **4. Implementing web surveys in Blaise**

### **4.1. Blaise IS**

This section explores the possibility of applying the guidelines described above in developing Blaise web survey instruments. To that end Blaise IS 2.0 was installed on Windows platform running Windows 2000 Server SP3 and Microsoft IIS 5.0. The Local Radio Listening Survey example was used to test the system.

The Blaise IS 2.0 system consists of three programs: the *Workshop*, the *Package Builder* and the *Server Manager*. Next to these programs, the *Online Assistant* provides the necessary information how to use these programs in order to set up a

web survey, starting from a Blaise data model (.bmi file). According to the *Online Assistant*, the three programs correspond to three roles in the survey process.

The *Workshop* is a tool for questionnaire designers. They can use this program, together with the *Modelib editor*, as a tool to define the layout of the web survey instrument. The *Workshop* can also be used to configure data entry behaviour (Interviewing, Edit-Check, Edit-No Check), response pages (web pages that are shown to respondents when the respondent submits or aborts a survey, or when an error occurs) and server settings (e.g. the location of the data file on the server).

The *Package Builder* is a program intended for survey managers. This program deals with survey management issues. It ensures that files concerning 'respondent authentication' and the 'interview start templates' are stored on the server. The *Package Builder* creates a package with all information to run the web survey.

Finally, the program *Server Manager* is intended for the server manager who monitors the technical aspects of the web survey (disk usage, security, and performance issues). Although, as in our case, it is possible that one person takes care of all these tasks, for large surveys it may be useful to have different people for these roles.

Traditionally, as introduced in Blaise 3, the layout of a Blaise survey is controlled by *Modelib* settings. For a Blaise web survey, the layout is controlled by the combination of *Modelib* settings and a 'style sheet'. The information required for a Blaise web questionnaire is stored in an XML file. The way in which the questionnaire is displayed on the screen is controlled by means of an XSL file. XSL (*eXtended Stylesheet Language*) is a language defining layout rules for the elements in an XML file. The beta-release of Blaise IS 2.0 provides two such style sheet files: one for a default web layout (*biHTMLWebPage.xsl*), and one for a default DEP layout (*biHTMLDEPPage.xsl*). It is possible to alter the existing style sheets, or even to write your own. However, this is not an easy task. The two style sheets both contain almost 3000 lines of XML code!

The style sheet *biHTMLDEPPage.xsl* is written with the purpose of creating a web survey instrument that is as close as possible to the *Data Entry Program (DEP)* that runs under Windows. Hence, this style sheet supports the settings of *Modelib*, and supports the menu settings of the DEP menu manager. A single exception is that it does not support the speed buttons of the menu.

The style sheet *biHTMLWebPage.xsl* is written as an example of a style sheet that is suitable for the purpose of a web survey. This prototype is supposed to be changed according to the wishes of the survey organization. By changing the style sheet, the web survey could include the logo of the organization. The style sheet *biHTMLWebPage.xsl* is written such that it only handles information in *Field panes*. The information in *Info panes* is not used. This explains why the default settings in the *Rijnwoude0* example (see section 4.2) resulted into a web page that only showed field names and input lines. No question texts or answer lists are displayed. In order to include these, we have to specify this using the *Modelib Editor* (see the *Rijnwoude2* example in section 4.4).

Section 4.2 describes the default web layout, and section 4.3 the default DEP layout. It is possible to create a totally different screen layout, and this is the topic of sections 4.4 and 4.5.

## 4.2. The default web layout

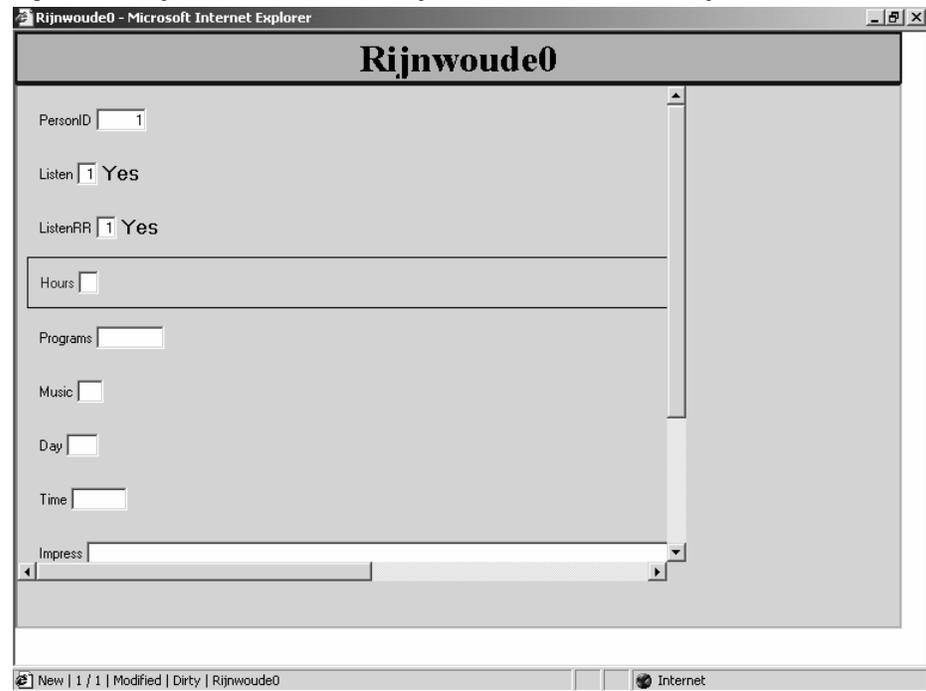
The first step in creating a web survey instrument with the default web layout was a traditional one: We programmed the sample questionnaire in the Blaise language,

and prepared it in the *Blaise Control Centre* to create the necessary files (e.g. the *bmi* and the *bdb* file). Next, we ran the three Blaise IS programs successively accepting default settings only. In the *Workshop* tool we merely selected the appropriate metadata file (*bmi* file), and saved the interview specification as *Rijnwoude0.bis*. We used the *Package Builder* tool to create a new web survey package, selecting the proper input file (*Rijnwoude0.bis*) and saving the new package as *Rijnwoude0.bip*. Finally, in the *Server Manager* tool we uploaded the package *Rijnwoude0.bip* to the Blaise Internet Server.

The result of choosing default settings only is a bare-bone interviewing instrument that cannot be used in a regular web survey situation. Figure 4.2.1 shows an example of the screen layout. The instrument has a number of major problems:

- There is no welcoming message;
- There are no question texts;
- There are no instructions on how to answer questions and how to proceed to the next question (or previous question);
- There are no navigation buttons, or other visual navigation means;
- For closed questions and all-that-apply questions, the possible answers are not listed;
- The answer to every question, including closed questions and all-that-apply questions, must be entered by means of a text input field;
- Horizontal and vertical scroll-bars must be used to navigate to other parts of the questionnaire;
- The questionnaire does not resemble its paper analogue;
- There is no clear survey description in the top part of the screen;
- There is no progress indicator;

Figure 4.2.1. Layout of a Blaise web survey instrument in default web layout



This first version has also some positive points: domain checks are carried out on entered answers, dynamic routing is applied, and the screen display is independent of the screen resolution. However, it is clear that the default version cannot be used for a web survey. It should be seen as a starting point for developing a more

realistic survey instrument. In section 4.4 we will show how we can use this style sheet and add question texts and navigation buttons.

### 4.3. The default DEP layout

As a next step in developing a Blaise web survey instrument, we used the default DEP layout. This leads to something that looks like the *Data Entry Program (DEP)* in *Interview* mode. This layout can be realized with minimal efforts.

We started with the prepared questionnaire in the *Blaise Control Centre*. Then we used the *Workshop* to alter one of the *Interview Specifications*. In the sub-tab *Stylesheet* of the tab *Layout*, we changed *biHTMLWebPage.xml* into *biHTMLDepPage.xml*. Again we used the *Package Builder* to create a web interviewing package. This package was added to the system with the *Server Manager*.

The result of these settings was indeed an instrument that resembled the Blaise *DEP* in CAPI/CATI mode. Figure 4.3.1 shows an example of the screen layout.

Figure 4.3.1. Layout of a Blaise web survey instrument in default DEP layout

The screenshot shows a web browser window titled "Rijnwoude1 - Microsoft Internet Explorer". The main content area contains a question: "In the last seven days, how many hours did you listen to Radio Rijnwoude?". Below the question are four radio button options: "1. 0 - 5 hours", "2. 5 - 10 hours", "3. 10 to 20 hours", and "4. More than 20 hours". Below the radio buttons is a text input field labeled "Enter a value". Below the text input field is a data entry form with several fields: "PersonID" (value: 1), "Impress" (empty), "Listen" (value: 1), "Sex" (empty), "ListenRR" (value: 1), "Age" (empty), "Hours" (empty), "Edu" (empty), "Programs" (empty), "Music" (empty), "Day" (empty), and "Time" (empty). The status bar at the bottom shows "New | 1 / 1 | Modified | Dirty | Rijnwoude1" and "Internet".

The screen layout has improved with respect to a number of aspects. We now have question texts. There are limited answer instructions. These are Blaise system texts. They may be changed but this is not a straightforward procedure. More extensive answer instructions could also be included as part of the question text, see section 4.5.

Radio buttons are used to present the possible answers to closed questions. Closed questions can be answered in two ways: by clicking on the proper radio button, or by pushing the corresponding numerical key. A long series of possible answers is distributed over several columns.

For check-all-that-apply questions, check boxes are used. Also here, a question can be answered in two ways: by clicking on the proper check-box, or by pushing the

corresponding numerical key. A long series of possible answers is distributed over several columns.

The screen displays one question at the time. No scroll-bars are required to make different parts of the questionnaire visible.

In principle, the top part of the screen cannot be used for displaying a logo, questionnaire title and a progress indicator. However, it is possible to include the survey title as the first item in each question text.

The bottom part of the screen presents a form view of the questionnaire. This is uncommon for web surveys. This view might help in giving the respondents some sense of where they are in the questionnaire form. If it works, a progress indicator would not be needed. And if one prefers to get rid of the form pane that can be taken care of in the *Modelib Editor*.

Probably the main problem with this layout is that it gives no clues as to how to navigate through the questionnaire form. It is not possible to include buttons for going to the next or previous question.

This instrument applies dynamic routing. Both domain checks and consistency checks can be carried out.

Although the default DEP layout is superior to the default web layout, it still is totally unacceptable as a web survey. While some trained interviewers may easily find their way with this layout, the average respondent of a web survey certainly will not. In addition we found that the style sheet with the DEP layout does not support navigation buttons. It is therefore less suitable as a basis for improvements. In the next section, we will proceed by improving the web layout of section 4.2.

#### **4.4. Improving the web layout**

Both the default web layout and the default DEP layout turned out to be insufficient for use in web surveys. In this section we will show how the default web layout can be changed in an attempt to come closer to an instrument satisfying the guidelines presented in section 3. Starting from the default web layout, we will combine question text and answer fields of a question on a single web page, and add navigation buttons to it.

In order to improve the layout of the web questionnaire form, we used the *Modelib Editor* to change a limited number of *Modelib* settings in order to put question texts and answer lists on separate pages. Figure 4.4.1 contains an overview of the new settings. The Data Model must be prepared in de *Blaise Control Centre* by setting the *Mode Library* to *Rijnwoude2.bml*. This can be done with *Project | Options*.

Figure 4.4.1. Modelib settings

Style	Form pane font: 10 (was 8)
Layout / Grids / DefaultGrid	BackGround: WHITE (was GREY) Cell width: 78 (was 38) Page width: 1 (was 2)
Layout / Field panes /DefaultFieldPane	Width: 78 (was 38) Height: 12 (was 1)
Controls / Field Name	Visible = FALSE (was TRUE)
Controls / Field Text	Visible = TRUE (was FALSE) Border = FALSE (was TRUE) Width = 78 (was 0) Height = 6 (was 0) Background = WHITE (was YELLOW)
Controls / Input Line	Left = 0 (was 12)
Controls / Answer List	Visible = TRUE (was FALSE) Border = FALSE (was TRUE) Top = 7 (was 0) Width = 78 (was 0) Height = 6 (was 0) Background = WHITE (was YELLOW) Columns = 1 (was 2) Codes = FALSE (was TRUE)
Controls / Answer Name	Visible = FALSE (was TRUE)
Controls / Remark point	Visible = FALSE (was TRUE)

Blaise's navigation buttons are available in the *Speed buttons tab* of the DEP menu. The style sheet for the DEP layout of the beta-release of Blaise 2.0 (i.e. the file *biHTMLDEPPage.xml*) does not support the DEP speed bar. However, the style sheet for the web layout (i.e. the file *biHTMLWebPage.xml*) supports a series of buttons: *Exit, Don't know, Refuse, Make remark, Previous question, Next question, Sub forms, Next page, Previous page, First page, Last page, Form language, Question help* and *Info*). We will limit ourselves to using two of these buttons: one for *Next question* and one for *Previous question*.

We used the *Dep Menu Manager* to add navigation buttons. Next, we created a Dep Menu file in the *Dep Menu Manager*. On the tab *Menu items* we de-selected the menu entries *Forms, Answers, Options, and Help*. Also, on the tab *Menu items*, we opened the menu *Navigate*, and de-selected all sub-menu items. We saved the menu specifications under the name *Rijnwoude2.bwm*.

An example of the result is shown in figure 4.4.2. The screen layout has improved with respect to a number of aspects.

We now have question texts. This version still has no answer instructions, although these could be added in the question text.

Radio buttons are used to present the possible answers to closed questions. For check-all-that-apply questions, check-boxes are used. In this version a long series of possible answers is not distributed over a number of columns. This requires more *Modelib* settings.

The input field for an open question always consists of one line. There are no simple instructions to create an input field consisting of more lines. Another problem with open questions is that entered text does not automatically wrap to a new line.

Figure 4.4.2. An improved version of the web layout



The screen displays one question at the time. No scroll-bars are required to make different parts of the questionnaire visible.

The top part of the question screen still displays its default characteristics. There is no logo, no longer title, and no progress indicator. Moreover, we see an ugly background colour, and a wrong text font. Sans serif fonts should be preferred.

This layout contains navigation buttons: one for going to the next question, and one for going to the previous question. However these buttons are not in an ideal position. One would prefer to have them at the lower right.

This instrument applies dynamic routing. Both domain checks and consistency checks can be carried out.

#### 4.5. An even more improved version of the web layout

The layout of the web survey instrument in section 4.4 was still not as we would like to have it. So, we explored ways to improve it even further.

On our request, the Blaise team sent us two adapted versions of the style sheet for the web layout (i.e. the file *biHTMLWebPage.xml*): one that allowed to add graphics to the title bar, and one that placed navigation buttons at the bottom of the screen. Based on the two style sheets, we created a new style sheet that combined these two functionalities, and that picked a different font for the header text. A further improvement was obtained by adding 'answer instructions' to the question texts, and by defining more detailed *Modelib* settings. We created different *Field panes* for different types of questions and added *Layout sections* to the Blaise questionnaire.

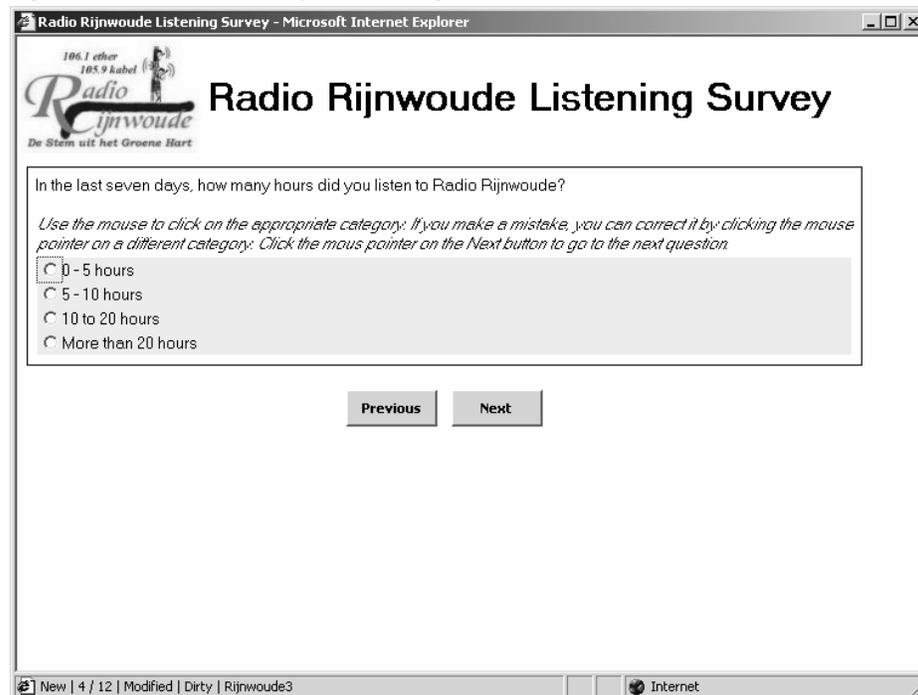
As a result, the top part of the screen has improved considerably. We now have a logo and a survey title. However, there is still no progress indicator.

Answer and navigation instructions have been included as part of question texts. These instructions are set in italics.

Radio buttons are used to present the possible answers to closed questions. It is now also possible to distribute a long series of possible answers over a number of columns. The same applies to check-all-that-apply questions.

The screen displays one question at the time. No scroll-bars are required to make different parts of the questionnaire visible.

Figure 4.5.1. An even more improved web layout



The screenshot shows a web browser window titled "Radio Rijnwoude Listening Survey - Microsoft Internet Explorer". The page features the Radio Rijnwoude logo with the text "106.1 ether 105.9 kabel" and "De Stem uit het Groene Hart". The main heading is "Radio Rijnwoude Listening Survey". The question is: "In the last seven days, how many hours did you listen to Radio Rijnwoude?". Below the question is a set of instructions: "Use the mouse to click on the appropriate category. If you make a mistake, you can correct it by clicking the mouse pointer on a different category. Click the mouse pointer on the Next button to go to the next question." There are four radio button options: "0 - 5 hours", "5 - 10 hours", "10 to 20 hours", and "More than 20 hours". At the bottom of the question area are two buttons: "Previous" and "Next". The browser's status bar at the bottom shows "New | 4 / 12 | Modified | Dirty | Rijnwoude3" and "Internet".

This layout contains navigation buttons: one for going to the next question, and one for going to the previous question. These buttons are neatly placed at the bottom of the screen.

This instrument applies dynamic routing. Both domain checks and consistency checks can be carried out.

#### 4.6. Domain and Consistency checks

We mentioned that all web surveys based on Blaise IS are able to carry out domain and consistency checks. We like to comment briefly on these topics.

In the Blaise questionnaire that we programmed we have domain checks. The domain check is a result of the field definition, as in:

```
Age "What is your age? " : 0..120;
```

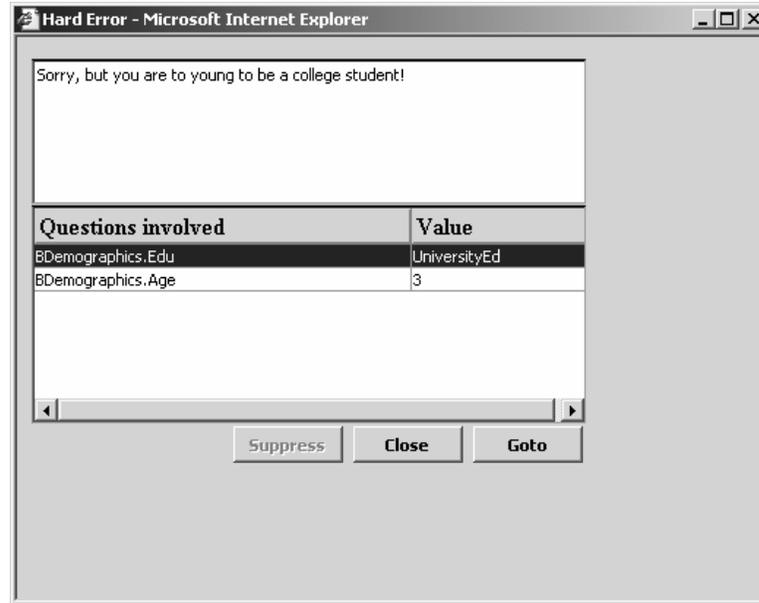
If a respondent provides an answer outside the domain, an error message pops up: "Input Invalid. Value not in range 0 to 120". The respondent can only click the *OK* button and returns to the text box that contains the out of range answer.

As an illustration we also added a consistency check:

```
NOT((Age < 16) AND (Education = EdUniversity))
"Sorry, but you are too young to be a college student"
```

In words the check means that we do not allow a person with an age below 16 to have university education. If a respondent violates this condition, an error message will pop up, see figure 4.6.1.

Figure 4.6.1. A consistency error has been detected



The top field displays the error message. The bottom field contains a list of questions involved in this error. The respondent is supposed to go to either the question *Edu* in the block *BDemographics* or to the *questionAge* in the same block *BDemographics*, and change its value.

Although the actions to be taken will be clear for experienced interviewers, we expect that naïve respondent will have large difficulty in dealing with this error message. For use in web surveys, we suggest to change this message in three respects:

- Change the name of the window (*Hard error*) in to something less frightening. Respondents will not understand what Hard error means. They could even think there is something wrong with their computer.
- The question names (e.g. *BDemographics.Edu*) and question answers (e.g. *UniversityEd*) will be cryptic for respondents. They will hardly be able to make a link between this information and questions already answered. A solution could be to use question labels and answer labels.
- The error message window should contain clear instructions about what to do. It must be clear for respondents that they must select a question, and go to that question to change its answer. Possibly, it is also a good idea to indicate that, after changing a question, the interview will continue from there. This could mean that respondent will see questions they already answered.

## 4.7. Conclusions

In this section we explored the possibilities of Blaise to create a layout for a web survey that satisfies as much as possible the guidelines presented in section 3. Blaise IS 2.0 contains two default layouts: (1) The DEP layout, which attempts to mimic the CAPI/CATI layout as much as possible, and (2) a default web layout.

In section 4.3 we discussed the shortcomings of the default DEP layout. Navigation is a fundamental problem here. Navigation with the Enter key and cursor keys works very well in a CAPI/CATI situation, where the interviewers are in charge of the instrument. However, for a web survey this type of navigation differs too much from web navigation. Therefore, it is by no means intuitive for respondents, and will be a source of problems, like dropping out of an interview in an early stage.

In section 4.2 we showed the default web layout. This layout is far from useful in a regular survey situation. It should be noted that the Blaise developers did not mean this layout to be used. They see it as just a starting point for questionnaire designers who will always want to adapt this layout for their own purposes.

By changing a *Modelib* file we were able to make a number of changes that made the instrument look more like a web survey instrument. This result was presented in section 4.4. To really come close to satisfying the guidelines in section 3, more substantial changes were required. The result of these changes can be found in section 4.5. This instrument comes close to what we would like to see on the web.

Finally we'd like to conclude with a set of recommendations and points of concern for the Blaise future.

### *Better and more style sheets*

In our opinion, the adapted versions of the style sheet for the web layout we used in section 4.5, were essential for the success of our efforts. Producing a web survey instrument requires a lot of work. Although the style sheet approach allows a survey designer to tailor the web layout, adapting a style sheet involves extra work that should be done by specialists. We feel it would help many survey designers if the default web layout addresses as many problems as possible. Therefore, we would like to make a plea for an improved default web layout, or offering a number of default web layouts.

### *Navigation issues*

We remarked that in the default web and DEP layout navigation is problematic. Although the addition of *Next* and *Previous* buttons improved navigation, the web survey still may show unexpected behaviour for naïve respondents. Problems occur when the respondent uses the keyboard to give answers. In the case of a closed question it is common Internet practise to use arrow-up and arrow-down to change the category. In the Blaise web survey, however, arrow-up and arrow-down are linked to the next or previous question. This may confuse respondents. We recommend the web survey to have the behaviour of a standard HTML form.

### *Open questions*

We feel that the input field of an open question should have more flexibility. It would be nice if the questionnaire designer could set the number of rows and columns of the field, and thus given an indication of the size of answer that is expected. Moreover, text should wrap to a new line. We used the *Modelib Editor* to create a *Text box* by defining a *Field pane* for the open text question with an *Input line* with a height of 6. However, the resulting text box showed improper behaviour: the text typed remained on the top line of the text box, and scrolled horizontally.

### *Progress indicator*

We did not succeed in including a progress indicator in the web survey instrument. We suggest making available a number of types of progress indicators in future versions of Blaise IS (text, bar, pie).

### *Check-all-that-apply questions*

In section 3 it was suggested that answer possibilities of a check-all-that-apply question could be presented as a set of yes/no radio buttons. It is possible to redefine a set question in the Blaise questionnaire as a list of yes/no questions. This would however restructure the data model. It would be desirable to make it a layout matter, so that we can choose between a 'check-all-that-apply' question and a series of yes/no questions. Maybe this point reflects Blaise in general, and not only to Blaise IS.

### *Active routing and checking facilities*

Blaise IS inherits a number of strong points from its CAPI/CATI version. One is that it applies active routing. And another strong point is the checking facilities. However, handling of error messages is a point of concern. Literature suggest using these checks sparsely. We expect that the error message caused by a domain check will not give problems, especially when it is clear to respondents why the answer they gave is not a valid one. The error message caused by a consistency check is too complex. The layout of the error message can be handled using a style sheet called *biHTMLErrorDialog.xsl*. We recommend carrying out research in order to find out how respondents react to different forms of error dialogues. And if a suitable format is found, it should be incorporated into the error dialogue style sheet.

### *Some programming issues*

We'd like to make some points with respect to the programming and development of a Blaise web survey. A point of concern is that the road to create a web survey is longer than the road to create a CAPI/CATI windows application. As a result, it takes more effort of a developer to find out what the effects of programming decisions are. In the case of a CAPI/CATI application, the developer can simply prepare and run the data model to find out if the application works, and how it looks. In the case of a web survey, such a test takes more steps. Although development tools as the *Modelib Editor* and the *Blaise IS Workshop* have preview facilities (we are very happy about that!), these are not always sufficient. In the case of the *Modelib Editor* the preview of the *Field pane* gives little information on how the result will look on the Web. The preview in the *Blaise IS Workshop* is much better, but requires more steps (saving *Modelib* settings in the *Modelib Editor*, and loading the bmi file in the *Blaise IS Workshop*. In case one uses dynamic text substitution, as we did to incorporate respondent instructions into the question text, the only way to see the result is to create a package of the *Interview specifications*, to upload the package to the Blaise IS server, and to test it.

Although we produced a list of points of concern and recommendations, we believe that the Blaise IS 2.0 edition is a very promising step to add web surveys to the set of data collection modes in the Blaise Family of Software.

## 5. References

- Baker, R.P., Crawford, S. & Swinehart, J. (2002): Development and testing of web questionnaires. Paper presented at QDET.
- Bethlehem, J.G. (1999): Cross-sectional Research. In: H.J. Adèr and G.J. Mellenbergh, *Research Methodology in the Social, Behavioural & Life Science*, Sage Publications, London, pp.110-142.
- Bowker, D. & Dillman, D.A. (2000). An experimental evaluation of left and right oriented screens for Web questionnaires. Presentation to Annual Meeting of the American Association for Public Opinion Research.
- Couper, M.P. (1999): Usability evaluation of computer assisted survey instruments. Proceedings of the Third ASC International Conference, Edinburgh, pp. 1-14.
- Couper, M.P. (2000): Web surveys: A review of issues and approaches. *Public Opinion Quarterly* 64, pp. 464-494.
- Couper, M.P., Traugott, M.W. and Lamias, M.J. (2001): Web survey design and administration. *Public Opinion Quarterly* 65 (2), pp. 230-253.
- De Haan, J., Huysmans, F. and Steyart, J. (2002): Van huis uit digitaal: verwerving van digitale vaardigheden tussen thuismilieu en school. Sociaal en Cultureel PlanBureau, Den Haag.
- Dillman, D.A. (2002): *Mail and Internet surveys. The Tailored Design Method*. Wiley, New York.
- Dillman, D A., Tortora, R.D. and Bowker, D. (1998): Principles for construction web surveys. Technical report 98-50, Social and Economic Sciences Research Center, Washington State University, Pullman, WA.
- Dillman, D A., Tortora, R.D. , Conradt, J. and Bowker, D. (1998): Influence of Plain vs. Fancy Design on Response Rates for Web Surveys. Presented at Joint Statistical Meetings, Dallas, Texas. August 1998.
- Dillman, D A. Bowker, D. (2001): The web questionnaire challenge to survey methodologists. In: Reips, U.D. and Bosnjak, M. (eds.), *Dimensions of Internet Science*, Pabst Science Publishers, Lengerich, Germany.
- Fricker, R.D. & Schonlau (2002): Advantages and disadvantages of Internet research surveys: Evidence from the literature. *Field Methods* 14, pp. 347-367.
- Heerwegh, D. and Loosveldt, G. (2002): An evaluation of the effect of response formats on data quality in web surveys. Paper presented at the International Conference on Improving Surveys, Copenhagen, 2002.
- Kish, L. (1967): *Survey Sampling*. Wiley, New York, USA.
- Magee, Straight and Schwartz (2001): Conducting web-based surveys: Lessons learned and keys to success with known populations. Paper presented at the 56-th Annual Conference of the American Association of Public Opinion Research (AAPOR), May 17-20, 2001.

Sangster, R.L. & Kennedy, J. (2002): Structured and flexible navigation within XML web surveys. Paper presented at the International Conference on Improving Surveys, Copenhagen, 2002.

Schneidermann B. (1997): Designing the user interface -- strategies for effective human-computer interaction. Addison-Wesley (3rd edn).

Tufte, E. (1983): The Visual Display of Quantitative Information. Graphics Press, Cheshire, Connecticut.

## Appendix. The sample questionnaire

Radio Rijnwoude wants to better serve its listeners and to stay in tune with their preferences. Therefore, an extensive audience survey is carried out.

To obtain a representative sample of listeners, it is very important that you fill in this questionnaire.

1. The last seven days, did you listen at least 5 minutes to the radio?

Mark one box.

- Yes  
 No → Skip to 9

2. In the last seven days, did you listen at least 5 minutes to Radio Rijnwoude?

Mark one box.

- Yes  
 No → Skip to 9

3. In the last seven days, how many hours did you listen to Radio Rijnwoude?

Mark one box.

- 0 – 5 hours  
 5 – 10 hours  
 10 – 20 hours  
 More than 20 hours

4. Which type of program did you listen to most during the last seven days?

Mark all boxes that apply.

- Music  
 News → Skip to 6  
 Sports → Skip to 6  
 Culture → Skip to 6  
 Other program → Skip to 6

5. Which type of music did you listen to most?

Mark one box.

- Chart / Top 40  
 Folk  
 Dance  
 Easy listening  
 Rock  
 Jazz  
 R & B

- Classical  
 Hip-hop  
 New age  
 Country  
 Other music

6. At what type of day did you listen most to the radio?

Mark all boxes that apply.

- Weekdays  
 Weekends

7. At what time of day did you listen most to the radio?

Mark all boxes that apply.

- Morning (6 – 12)  
 Afternoon (12 – 18)  
 Evening (18 – 24)  
 After midnight (24 – 6)

8. What is your general impression about the quality of the programs of Radio Rijnwoude?

Print comments.

9. Are you male or female?

Mark one box.

- Male  
 Female

10. What is your age?

Print number in boxes.

11. What is your highest level of education?

Mark one box.

- Primary education  
 Secondary education  
 Tertiary education

This is the end of the questionnaire. Thank you for participating in the survey.



## *Applications of Blaise*

- **The usage of Blaise in an integrated application for the Births Sample Survey in CATI mode** .....125  
*Massimiliano Degortes & Stefania Macchia, Manuela Murgia, ISTAT, Rome*
- **Blaise at Statistics Netherlands**.....137  
*Marien Lina, Statistics Netherlands*
- **Displaying Chinese Characters In Blaise** .....149  
*Gina-Qian Cheung & Youhong Liu, Institute for Social Research, University of Michigan*



# The usage of Blaise in an integrated application for the Births Sample Survey in CATI mode

*Massimiliano Degorte & Stefania Macchia, Manuela Murgia ISTAT, Rome*

## 1. The new strategy for CATI surveys

A lot of CATI surveys are realised in Istat (Italian National Statistic Institute) and, before the experience described in this document, for all of them the same strategy was adopted: the content of the survey, made clear in the questionnaire, was designed in Istat, but the entire data capturing procedure was carried out by an external private company, selected through an administrative procedure. Even if this strategy always succeeded in concluding the data capturing phase, it often presented different problems. The main difficulties concerned the fact that private companies in charge of Istat surveys were often very experienced in telemarketing or opinion polls, but:

- had never faced in advance the development of electronic questionnaires so complicate in terms of skipping and consistency rules between variables
- had never put in practice strategies to prevent and reduce non response errors
- had not at their disposal a robust set of indicators to monitor the interviewing phase.

Consequently, the product obtained always made it possible to make the interviews, but did not often satisfy all the pre-defined requirements.

That is why a new strategy, called the '*in house strategy*', was tested for a very important survey (the Birth Sample Survey): we relied on a private company who provided the call centre, selected the interviewers and carried out the interviews, but we gave it all the software procedure to manage the data capturing phase, concerning: the calls scheduler, the electronic questionnaire and the indicators to monitor the interviewing phase.

This strategy not only guaranteed the complete correspondence between the planned requirements and the developed procedure, but also allowed a strict collaboration between the survey responsible and the CATI experts in defining the characteristics the package should have and in improving it according to the results obtained step by step during the developing phase. In addition, as the Birth Sample Survey has to be repeated every two years, it will not be necessary to face again all the planning and developing activities in case a different company is charged with this job.

Blaise is the core in this strategy as it has been used for the most important modules of the software procedure.

## 2. The Birth Sample Survey

A new CATI sample survey on births has been designed in Istat and it substitutes the previous exhaustive survey based on administrative sources. A sample of 50.000 mothers is selected from the universe of born alive babies, registered during a period of 12 months which precedes of at least 6 months the interviewing phase. The main purpose of the survey is to collect information regarding basic elements on living births and, for the first time in Italy, to go deep into familiar and social aspects in order to identify the determinants of fertility. The designed questionnaire comprehends four sections, regarding: *i*) the births, the delivery and the household

members, *ii*) mother's job before and after having the baby, *iii*) the baby's care and the management of daily tasks inside the family; *iv*) the house and the socio-economic context. As very long interviews were expected, it was decided to submit the complete questionnaire only to a sub-sample of mothers and a short version, comprehending only sections one and four, to the whole sample; in this way we will obtain estimates respectively at national and regional level.

Three interviewing periods have been defined along one year, according to the babies date of birth.

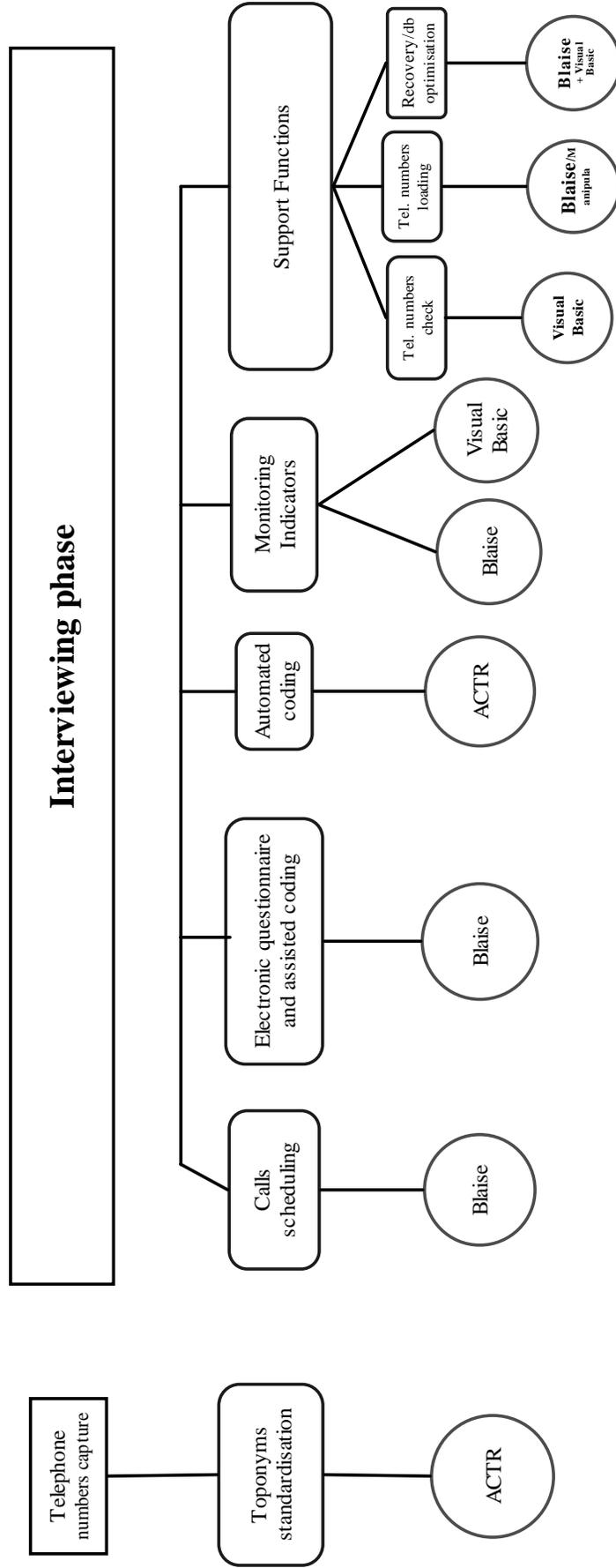
Being the first time this survey was carried on and the new strategy adopted, a pre-test was made, with the following purposes: *i*) testing the whole operating process (selection of sample units, matching with telephone numbers, training of interviewers, development of electronic questionnaire...), *ii*) testing the questionnaire, *iii*) estimating the long form interviews' length. Interviews were performed at Istat, in the CASIC (Computer Assisted Survey Information Collection) laboratory, for a four working days period, by four female interviewers chosen among internal employees.

Following the satisfying results of this test, the '*in house strategy*' was adopted for the full-fledge survey.

### **3. The integrated CATI application for the Birth Sample Survey**

As it can be seen in Figure 1, the software application is composed of various modules for different functions, developed using different software packages.

Figure 1



In particular, the **first phase** concerns the improvement of matching between names and addresses of sample units, extracted from administrative registers, and telephone registers, in order to find telephone numbers. It must be said that the matching activity was delegated to the private company, but, as the results were not satisfactory, we thought to make the matching rate higher by standardising the toponyms<sup>1</sup>, that were a very variable element in the address information. For this purpose, we adopted **ACTR** (Automatic Coding by Text Recognition), a generalised system, developed by Statistics Canada and already used in Istat to code a certain number of variables. As a matter of fact, the main purpose of this software is the matching between the texts of variables captured through open-ended questions and the texts of an internal reference file, in order to assign a code of a predefined classification. As this matching phase is preceded by a text standardisation phase, we used this function to reconduct toponyms expressed in different ways (with synonymous or abbreviations) to the words used in telephone registers. Following this activity the matching rate grew up of 5%.

For the **interviewing phase** different software packages have been used, but **Blaise** has been adopted for the most important functions, like:

- the calls scheduling functions
- the development of electronic questionnaire
- the assisted coding
- the production of some indicators to monitor the interviewing phase
- the telephone numbers loading
- the recovery and the database indexes optimisation.

The combined usage of other software has been necessary mainly for two purposes:

- the production of a large set of indicators to monitor the interviewing phase
- the automatic coding of a complex variable.

### 3.1 Functions implemented with Blaise

#### 3.1.1 Calls scheduling

For the calls management function the ‘CATI management’ of Blaise was adopted. All the potentialities were exploited, concerning:

- the random attribution of telephone numbers to interviewers
- the definition of different interviewers groups depending on the interview language (Italian or German)
- the routing back of appointments to the interviewers who took them
- the definition of *Parallel Blocks* to insert the call result during the interview (*Refuse*, *Interruption* and *Disconnected*). In these blocks we included variables, whose descriptions (in the ‘*type*’) allowed us to register the main causes of *Refuse* and *Interruption* and the kind of *Disconnected* (‘*not existent number*’, ‘*wrong number*’ and ‘*household moved to another address*’).

#### 3.1.2 The electronic questionnaire

As already said, the electronic questionnaire is the core of the application and its quality constitutes the main reason for adopting the ‘*in house strategy*’.

---

<sup>1</sup> The toponyms represent the part of the address that specifies the type of street (i.e. street, route, road, etc.)

The *long form* version of the survey questionnaire has 212 questions, while the *short* version 79, but, after the interviewing phase, we estimated that the average number of asked questions was 110 with the *long* and 45 with the *short*.

The complexity is demonstrated by the checking plan: 195 checking rules have been defined for the *long form* and 63 for the *short*. These rules have been treated both in *soft* and *hard* mode (for instance, in the *long form* version, 101 rules are *soft* and 94 are *hard*).

A lot of other functions offered by Blaise have been used, so the questionnaire turned to be very flexible and user friendly, and the quality of captured data has been guaranteed:

- **data from an external administrative archive, concerning a certain number of variables, have been loaded in Blaise** to compare them with responses provided during the interview, in order to verify their correctness (for example: mother's and father's date of birth, parental status, etc.)
- **internal routines** have been developed in order to calculate the birth order of the sample baby and to re-compute the sample stratum of the unit according to the given answers
- the **assisted coding** function has been used for City/Municipality of birth and of residence and for Country of birth variables
- **don't know** and **refusal** attributes have been associated to the majority of variables
- **questions wording has been customised** according to already known administrative data or answers given to previous questions
- **standards in the screen management** have been defined in order to make the interviewer's job easier (for instance, different colours have been used for different purposes: **black** for texts to be read to the respondent, **green** for questionnaire sections titles and **red** for all the helping texts useful for interviewer but not to be read to the interviewed person)
- **error messages** have been identified by sequential numbers and error texts have been customised according to responses already given.

### 3.1.3 Telephone numbers loading

The sample units file is loaded in Blaise with **Manipula**, using the "*update*" instruction. As in this way we were not able to individuate duplicated keys, we had to make this check using a **Visual Basic** programme.

### 3.1.4 Utilities for recovery and database indexes optimisation

As it will be described in chapter '*Problems encountered*', after the first interviewing days it happened a lot of times that interviews were stopped due to Pc's freezing. It has been hard to overcome these problems, but one of the actions which contributed in solving them was to check daily the database status and to perform a database de-fragmentation.

This has been done by:

- running daily Hospital
- running daily Blaise to Blaise.

In order to make the management of the two procedures (Hospital and Blaise to Blaise) more user friendly, we created a **.WSH** file that execute them automatically and, after processing them, launches an e\_mail message to the Istat informatics experts, concerning their normal or abnormal end.

## 3.2 Functions implemented with other software packages

### 3.2.1 Production of some indicators to monitor the interviewing phase

As far as **indicators** are concerned, we needed to keep daily under control different aspects, like:

- the productivity of interviewers
- the proceeding of contact results
- the number of sample units still at disposal and their movements from a stratum to another.

For this purpose, we prepared different tables aimed at monitoring these three main aspects of the survey:

- the productivity of interviewers
  - telephone contact results (daily and cumulative)
  - refusals causes (total and per interviewer)
  - definitive interruptions causes (total and per interviewer)
  - definitive interruptions per questionnaire section
  - interview length (total and per interviewer)
  - cumulative response and non response rates (total and per interviewer)
- the proceeding of contact results
  - last contact results distribution per type and time slice
  - last contact results distribution per type and per day of week
- the number of sample units still at disposal and their movements from a stratum to another
  - last contact results
  - number of sample units still at disposal (never contacted or already contacted)
  - no more available sample units (because the maximum number of contacts has been reached) or still available sample units classified per stratum
  - matrix of movements of interviews from a stratum to another.

The majority of these tables have been developed using **Visual Basic**, based on an Access database, which produces output Excel files, that can be easily managed by statisticians.

### 3.2.2 Automated coding

The other function implemented with another software is the automatic coding of Occupation. As a matter of fact, three variables were open-ended questions:

- City/Municipality of birth and of residence
- Country of birth
- Occupation

It was decided to code during the interview only the first two variables and not the third one, since its very complex classification would have dramatically increased

the interview length. Therefore the first two variables have been coded with the **Blaise coding function**, (respectively City/Municipality with the alphabetic search and Country with trigram search) while the third one has been processed at the end of the field-work in batch with **ACTR**, the already mentioned coding system.

## 4. The interviewing phase

### 4.1. The informatics architecture

As already said, according to this strategy, the interviews were performed by a private company.

The call centre devoted to this survey was composed of 57 clients (Pentium II with Windows NT) and 1 server (Pentium II, NT4 Service Pack 6 – 512 Mb Ram – 2 disks with 40 Gb each).

The physical structure of the network was the following one:

- 100 Mbit network
- Switch layer 3 Cisco
- Catalyst 3500, series XL
- Switch hub allied telesim
- Centre come FH824u
- LAN class G
- Domain server DSN

A CATI users profile Blaise application has been installed on the server PC, while all the clients had a link to the server. With this organisation, all the data were stored on the server, apart from two files, containing the classifications dictionaries used for assisted coding (*Lookup files*), which have been stored on the clients, because this solution guaranteed better performances.

### 4.2. The main CATI results of the survey

Interviews were made in three periods along 2002 (April – May, September – October, November - December).

In each period, almost 17,000 interviews were completed. The cumulative results are shown in Table 1, while the average number of completed interviews per hour by interviewers was 3 for the *Long* questionnaire and 5 for the *Short* version, with an average interview length of 13.4 minutes for *Long* and 6 for *Short* version.

**Table 1: final contact results**

Contact results	Short		Long	
	<i>nn.</i>	%	<i>nn.</i>	%
1) Concluded interviews	33838	68	16597	67
2) Disconnected	10937	22	5640	23
3) Refusals	1774	3.6	974	3.9
4) Definitively interrupted interviews	692	1.4	359	1.4
5) No more available sample units because the maximum number of contacts has been reached	2551	5.1	1327	5.3
<b>TOTAL</b>	<b>49792</b>	<b>100</b>	<b>24897</b>	<b>100</b>

These results are really satisfying and coherent with those obtained in other CATI surveys on households realised by Istat, apart from the too high percentage of wrong phone numbers, which depends on other elements that do not regard Blaise.

### 4.3 The impact of the usage of Blaise

The impact of the usage of Blaise was very positive from different points of view: that of statisticians, of developers and of the users.

As a matter of fact:

- the electronic questionnaire was really performing and the quality of captured data very high, thanks to the complex set of controls which could be inserted without making the interview too heavy
- the questionnaire was developed by programmers not already experienced in Blaise, which demonstrates that it is not so difficult to be trained in Blaise for a person who already has a programmer's experience, even if not for such a long time
- interviewers found the questionnaire very pleasant and had no problems in using a tool different from that they used in advance
- the response time after each 'enter' was very short and absolutely acceptable.

### 4.4 Problems encountered

A lot of problems have been encountered, but all of them were solved during the first interviewing phase.

In general, the first difficulties concerned the lack of a sufficiently structured information about the CATI management system and about the way contact results are stored: it has been necessary to perform some empirical trials in order to understand when a certain telephone number is proposed again and in which field of the Blaise database results are stored.

In particular, the main problems we faced were the following ones:

#### a) System freezing

During the first interviewing days it happened too frequently (more than once a day) that all the PCs (one after the other) froze. As it was not clear how it could happen, the first advice received by CBS was to change a **Windows parameter** (to put 'EnableOpLock' = 0) and to adopt the last Blaise release (4,5,3, build 692) in which a lot of possible freezing problems had been solved.

As this solution did not overcome the impasse, we followed the Westat advice to try to run a '**Blaise to Blaise procedure**' (after Hospital) in order to perform a database de-fragmentation.

Following the daily running of these two procedures (Hospital + 'Blaise to Blaise'), freezing did not happen anymore, but it is still not clear if and why it is necessary to execute daily 'Blaise to Blaise'.

Could it depend on the amount of data or on the network architecture?

In the first case it would be interesting to compare different experiences of Blaise questionnaires which produce very long records, while in the second case the characteristics of the network system on which Blaise is installed should be defined in details.

#### b) Creation of more history files

After the mentioned freezing, Blaise produced more than one history file, with different extensions (.bth(2), .bth(3)...). As nobody else apart from Blaise was

using the history file in those moments, we could not understand why these files have been produced.

**c) Not enough flexibility in the updating of CATI parameters.**

In this survey it was necessary to manage appointments in a different way depending on the time of calls.

As a matter of fact, interviews were conducted in the afternoon above all, while in the morning only appointments had to be dealt (apart from Saturday morning); therefore all the interviewers were working in the afternoon, while only a certain number of them were present in the morning. Consequently, it was considered an efficient solution for the afternoon to *route back* the appointments to the same interviewers who took them, while for the morning to *'route back to anyone'*.

What we noticed is that Blaise does not update the parameters values in the daybatch, thus managing the delivery of forms according to the previously stored set of parameters. So, to meet our needs, we had to treat the morning parameters manually using the *'treat form'* function.

**d) System bugs**

Some of the problems listed below were originally caused by our errors, whose consequences were unpredictable or were not pointed out by Blaise with ad hoc error messages.

**- Lacking of an error message**

A relevant problem was caused by the following action: as already said, we decided to run daily 'Hospital' + 'Blaise to Blaise' to solve our freezing problems. It happened that, due to an insufficient disk space, Manipula, in running the 'Blaise to Blaise' procedure, created a database containing less forms than the original one, causing the loss of a certain number of records.

We realised it only casually during the following interviewing day, and we had to immediately re-build a file containing all the missing information.

**- Duplication of records in the day-batch file**

In order to overcome the already mentioned not sufficient flexibility of the updating of CATI parameters, we solved manually the appointments management with the *'treat form'* module: the supervisor, after executing the *day-batch* each evening for the following morning, put value *'everyone'* in the field containing the interviewers' name to whom the appointment had to be routed back (*whomade*), after choosing the option *'call as soon as possible'*. This option was evidently wrong (we should have chosen that of appointment), but the further problem was that we observed that Blaise duplicated the form in the *day-batch file*: one concerning the original appointment and a second concerning *'call as soon as possible'*.

**- Zero value in dialresult**

In order to avoid the frequent wrong usage of a parallel block by interviewers, we forced the parallel block call with the following instruction:

```
if sez.Qinizio = no then catimana.caticall.regscalls[1]:= Nonresponse
```

As a consequence of this action, we obtained the attribution of a zero value to the 'dialresult' field and, as this value does not correspond to any call result, it created a certain number of problems to our application.

The Blaise technical support suggested us to use the field CALLRESULT and not REGSCALLS to execute this instruction, specifying that it is strongly recommended not to use the *'register of calls'*.

So, even if we made a mistake as we did not get sufficient information in advance, we wonder if it were useful to display a 'warning' saying that the field is not to be updated.

**- Errors in assigning values to fields**

If a mistake is made when assigning a value to a field, using (=) instead of (:=), each row of the program which follows this instruction is considered as the text of an error message.

This could be avoided by checking that an error message has always to be delimited by inverted comma.

**e) PCs synchronization**

We had some problems since sometimes not all the client pc's were synchronized.

In details, the problems we encountered were:

- a wrong delivery of forms
- a wrong sequence of dial results per phone number: concluded dial results (*Interview* or *Refusal*) were registered before not-concluded dial results (i.e. *Busy*, *Appointment*, etc.)
- 'Start time' of some phone contacts was postponed with respect to the relative 'End time'.

As all of the PCs read the same database on the server, we think it would be useful to perform the synchronization automatically.

In addition, when using the history file and the database to produce some reports, we observed that the contact time in these two files is not identical, because in the database it is 5 minutes rounded.

**f) Access violation**

During the data entry activity it sometimes appeared a message declaring an 'access violation', in different points of the interview. It has been possible neither to understand the reason of this message, nor to verify a cause common to all the moments this message has appeared.

**g) Lack of information about the telephone contact during the interview**

After leaving the "Make dial" screen and entering the data entry-program, Blaise does not provide any information about the contacted form. As a consequence, we created a .dll in DELPHI, able to make the display on the screen of the telephone number and the name of the contacted household, necessary to re-contact it when the line goes off.

**h) Lack of synthetic information about interview length**

If an interview is completed in more than one contact (for instance it is started, interrupted but not definitively and completed in a subsequent appointment), it is necessary to sum up the length of each contact if we want to know the total length of the interview. We think it would be useful, after the last definitive contact result, to store in an additional field the complete interview length

## 5. Consideration on the adoption of Blaise

Following this very important experience, reflections have been made in Istat on the feasibility of this new strategy and, above all, on the adoption of Blaise for a lot of CATI surveys planned in Istat.

The conclusion which has been drawn is very positive. The main items which made us think that it is worth going on are those already described in paragraph 4.3. In addition, we presume that all the mentioned problems faced during the first interviewing phase provided us with such an experience to prevent us from unpredictable situations in the future.

Consequently Istat decided to change its Blaise contract: we owned a simple licence for a certain number of developers and users and turned to a 'Corporate licence', so to be able to provide software for other surveys which could adopt this strategy, without having to face administrative and economic problems.

## 6. References

Blaise for Windows Developer's Guide 4.1

Wenzowski M.J. (1988). ACTR – A Generalised Automated Coding System. *Survey Methodology*, vol. 14: 299-308.

Groves R. M., Biemer P. P., Lyberg L. E., Massey J. T., Nicholls II W. L., Waksberg J. (Eds.) (1988) *Telephone Survey Methodology*, Wiley, New York.

Macchia S. and Murgia M. (2002). "Coding of textual responses: various issues on automated coding and computer assisted coding". *JADT 2000: 6es Journées Internationales d'Analyse Statistique des Données Textuelles*, St Malo 13-15 marzo 2002

Muratore M.G., Quattrociochi L. and Sabbadini L.L. (2000 -in corso di pubblicazione) "Indagini sociali telefoniche: metodologia ed esperienze della statistica ufficiale", Istat



# Blaise at Statistics Netherlands

*Marien Lina, Statistics Netherlands*

## 1. Introduction

Statistics Netherlands developed Blaise as a system for data entry and survey design. Being the producer of the system, Statistics Netherlands is also one of the bigger Blaise users. This contribution focuses on Blaise related applications at Statistics Netherlands that use the Blaise system.

The starting point is Blaise data entry and interviewing. At Statistics Netherlands Blaise has been the standard environment for developing and executing CAPI and CATI interviews for 15 years now. Surveys, carried out by Statistics Netherlands differ in size and complexity of the questionnaire, willingness to respond, accessibility of the sample, the need for personal interaction, respondent burden, response sensitivity and subjectivity of question content. Different surveys ask for a different solution, different data entry types, different ways to address respondents and different techniques, such as applying external data files for typing and coding, or running external procedures while entering data in a Blaise data entry application.

The “traditional” Blaise data entry modes, used at Statistics Netherlands are data entry of paper forms (CADI), Face-to-face interviewing (CAPI) and Telephone interviewing (CATI). They keep playing a significant role in the data entry process at Statistics Netherlands. The first part of this paper will review surveys and sample sizes for various data entry modes.

Recent developments in Blaise offer new technical solutions developed for specific demands. New technology enables new functionality (for example, Internet interviewing) and simplified solutions for complex functionality (for example, combining information from different surveys or interview modes). New Blaise developments were both a challenge and a necessity for carrying out various projects successfully. A number of innovative projects will be reviewed here. The selected themes are related to: *Web interviewing, automated data imputations, mixed mode surveys* and the administration of the survey process of these mixed mode person and household surveys.

This contribution will review:

- “Traditional” data entry and data manipulation: An overview of Blaise interviewing, data entry and Manipula applications at Statistics Netherlands for several person, household and establishment surveys.
- Technological Blaise innovations
- Innovative projects at Statistics Netherlands
- The SSA System for Survey Administration for person and household surveys
- The IMPECT project
- The Electronic Data Reporter for establishment statistics
- Web interviewing in a survey to measure establishment turnover.

## 2. “Traditional” data entry and data manipulation

At Statistics Netherlands, Blaise has been the standard environment for developing and executing CAPI and CATI interviews for 15 years. In 2003, Statistics Netherlands will perform approximately 175,000 field and 175,000 telephone interviews in Blaise.

### 2.1 CATI interviewing for person and household surveys

Most of the CATI interviews at Statistics Netherlands are carried out with Blaise. The Labour Force Survey is the largest household survey.

CATI-interviews with Blaise for person and household surveys:

Survey	CATI interviews *)
Labour force survey	122,000
CCO (Consumers prosperity)	18,000
EDISENT reminders	5,000
Price observation petrol	< 1,000
Non-Response analysis survey	8,000
School-leavers panels	8,000
Woonkosten A (house rent survey)	< 1,000
Experiments Budget survey	1,000
Budget Survey first contact	11,000
<b>Total Blaise CATI interviews</b>	<b>175,000</b>
OVG – traffic research	43,000
Environmental costs survey	<1,000
Experimental	6,000
<b>Total Non Blaise CATI interviews</b>	<b>50,000</b>
<b>Total</b>	<b>225,000</b>

\*) approximate planned numbers + / - 20 %

Most of the person and household CATI interviews at Statistics Netherlands are conducted with Blaise. An exception is the OVG, using paper booklets to enter moves and means of transportation. The special Neu Kontiv Design that has been used for the OVG survey implied specific experimental methods that asked for a specific approach. Nearly 80% of the person and household CATI surveys use Blaise for data entry.

### 2.2 CADI interviewing for person and household surveys

CADI interviews with Blaise for persons and households planned for 2003:

Survey	CADI interviews *)
CPI – price observations	117,000

\*) approximate planned numbers + / - 20 %

In the price-observation survey data are gathered on paper and data entry is done afterwards in Blaise. The questionnaire is very small and it concerns just the prices of a number of goods in shops. In fact the survey population are goods here and not persons or households. Data entry is carried out by the same organisational group as for person and household surveys.

### 2.3 Blaise CAPI interviews for persons and households

Face to face Blaise interviews - persons and households. Laptop / 2003:

Survey	CAPI interviews *)
EBB (labour force survey)	96,000
Gezinsvorming (Survey Family Settling)	16,000
POLS – health survey	17,800
POLS – living conditions and behaviour	8,900
POLS - legal protection and security	8,900
POLS - the young	7,100
ECP - Eurostat corrections	1,000
Budget Survey – first contact	5,000
Budget Survey 2003	2,400
Budget Survey 2004	14,200
Experiments	400
Total Blaise surveys	175,700

\*) approximate planned numbers + / - 20 %

Blaise data entry is used for every field survey among persons and households.

### 2.4 Blaise interviewing and data entry in establishment statistics

Compared to person and household questionnaires, establishment statistics use more secondary data and the surveys usually are much smaller. Some of the establishment statistics do not depend entirely on data entry, but they merely are in need of additional figures to adjust the data readily available, when the bulk of the information is retrieved through other channels.

The following overview gives numbers of data entry forms, entered yearly in Blaise. Some data collection activities may not be listed as there is no central administration of these surveys.

Data entry records per year – establishment statistics.

Statistical topic	Forms entered per year *)
Impect Ps (long list)	80,000
Impect Ks (short list - additional questions)	400,000
International trade **)	600,000
Financial enterprise statistics	40,000
Bus transportation	50,000
Road transport (freight)	300,000
Canal shipping trade	60,000
Sea shipping trade	10,000
Fire Departments statistics	500

\*) approximate numbers + / - 20 %

\*\*) the 600,000 Blaise records are additional to 74,000,000 from secondary data.

The table shows that the number of records entered with Blaise is about 1,5 million each year, about 5 times more than the 350,000 person and household interviews.

One should keep in mind that data collection for establishments is usually data entry for a limited number of data fields, and the time needed to fill in a questionnaire or data form is limited.

There are more establishment statistics and some of them don't use Blaise data entry. For example, data entry of about 280,000 forms for a survey on job vacancies and absence through illness is done with other data entry software. There may be more surveys of this kind that have very small data records. They may be retrieved from and/or combined with specific database formats, not related to Blaise.

Apart from the mentioned figures, there are many records entered in Blaise by establishments outside Statistics Netherlands. In the Iris project, this concerns about 21 million each year.

The tables are not complete (especially the one with the data concerning establishment surveys) but the most extensive surveys are listed here.

## **2.5 Data manipulation at Statistics Netherlands**

Apart from data entry, another "traditional" Blaise functionality is data manipulation. The relevant software parts in Blaise are Manipula, Maniplus and Cameleon. The three tools are applied to execute procedures for cleaning data, typing and coding, file management and preparing data for analysis in other software applications.

The number of calls per year of the Manipula program at different parts of Statistics Netherlands is more than a million. More than 40% of the Manipula calls are relatively "old fashioned" and designed to process ASCII files. More and more of the old systems are being replaced by new technology using new Blaise 4 functionality or Blaise Component Pack applications. COM technology ensures communication options between Blaise and other applications.

The Blaise Component Pack may carry out Manipula-like procedures without activating Manipula. The million calls to Manipula exclude the calls to VB (including Blaise Component Pack) for similar purposes. Some of these (in fact the applications in the SSA project) replace previous applications that used Maniplus and Manipula setups.

## **2.6 General conclusions on traditional data entry and data manipulation**

The practical usage of Blaise for data entry at Statistics Netherlands, derived from the mentioned figures above illustrates the strong quality of the Blaise system for data entry and manipulation.

The choice for Blaise as a data entry machine is obvious when questionnaires and data structures are larger and more complex. For very complex data entry machines like the Labour Force Survey and POLS (life situation survey) using Blaise has become the de facto standard at Statistics Netherlands.

Smaller data records are found at establishment statistics. Statistics Netherlands has a standardisation policy for data collection methods. This policy encourages the use and development of Blaise data entry machines. The aimed practical merit of such standards is increasing exchangeability of data in the future.

## **3. Technological Blaise innovations**

At Statistics Netherlands, Blaise plays and keeps playing a significant role in the survey process, including applications for data entry, typing, coding, manipulating data and preparing files for analysis. The "traditional" functionality is very useful to perform these tasks.

### **3.1 Blaise developments “the old way”**

Over the years, new functionality has been added to the Blaise system in numerous tools and utilities. For example, at the beginning of Blaise developments in the late 80s, the CATI system has been added. Over time it has been extended and developed as a stable toolbox with “out-of-the-box” standard procedures. The CATI modules allow for specifying many parameters in standard dialogues and parameter files. Yet it remains a standard tool you may use “as-is”. As a result, the CATI management system works fine for all standard CATI interviewing. One can imagine, special requirements ask for tailor-made adaptations on-site. Specific non-standard functionality may be added with the programming tools Manipula, Maniplus and Cameleon. Functionality has been added to Manipula to increase control on the CATI management system (for example the daybatchadd keyword). This illustrates how Manipula, Maniplus and Cameleon enable creating tailor-made applications around Blaise. The building of applications in these languages requires specific programming skills. With the programming facilities of Manipula, Maniplus and Cameleon, meta and data manipulations are handled separately (meta information via Cameleon and data via Manipula or Maniplus).

Different institutions and different survey types using Blaise asked for specific features not implemented in the standard tools. As there are many users, this called for more flexible tools, enabling Blaise users to create “home-made” applications to meet their own special needs.

### **3.2 Blaise Component Pack**

Blaise Component Pack (BCP) 2.0 was necessary to carry out the projects that will be reviewed below. The projects in turn have been a driving force for developing and implementing specific required functionality in BCP 2.0.

New Blaise Component Pack technology opens a flexible way to create Blaise and Blaise related applications. BCP did not only increase flexibility for user-made additions to the system, it has also been used for developing new standard tools by the Blaise development team. Among others, the BCP technology has been used to develop the Blaise Internet Services application (Blaise IS).

Some elements of this new approach with BCP are:

- Creation of VB applications that can be activated from within Blaise data entry applications or Manipula jobs (alien procedures and alien blocks).
- Using the OLEDB approach and data exchange via BOI file enables easy communication between the Blaise data format and other data formats.
- BCP enables the creation of Visual Basic applications that read and write Blaise data. VB applications that run outside the Blaise environment can be developed to include and combine DEP, Manipula, Maniplus and Cameleon functionality.

Both developments are continuing:

- adding standard tools to the existing Blaise system
- developing new technical options in Blaise Component Pack 2.0.

## **4. Innovative projects at Statistics Netherlands**

New Blaise technology has been used in a number of projects at Statistics Netherlands. Some use BCP technology, others use new Blaise 4.6 functionality.

For the person and household surveys, the need for controlling and administrating the survey process (on a record level) has resulted in the SSA system. This System

for Survey Administration implements the control and organisation of subsequent steps for processed records in person and household surveys. It combines many surveys in one database and controls data flow. The system is built with VB tools that use Blaise Component Pack technology, especially the Manipula functionality.

For establishment statistics the IMPECT project at Statistics Netherlands combines Blaise with VB and SQL-Server applications. The global outline of IMPECT will be reviewed below. The IMPECT project has a number of sub-projects, each with their own functionality and relations with Blaise, some of which will be treated here, such as LogiQuest, controlling the organisation of data collection.

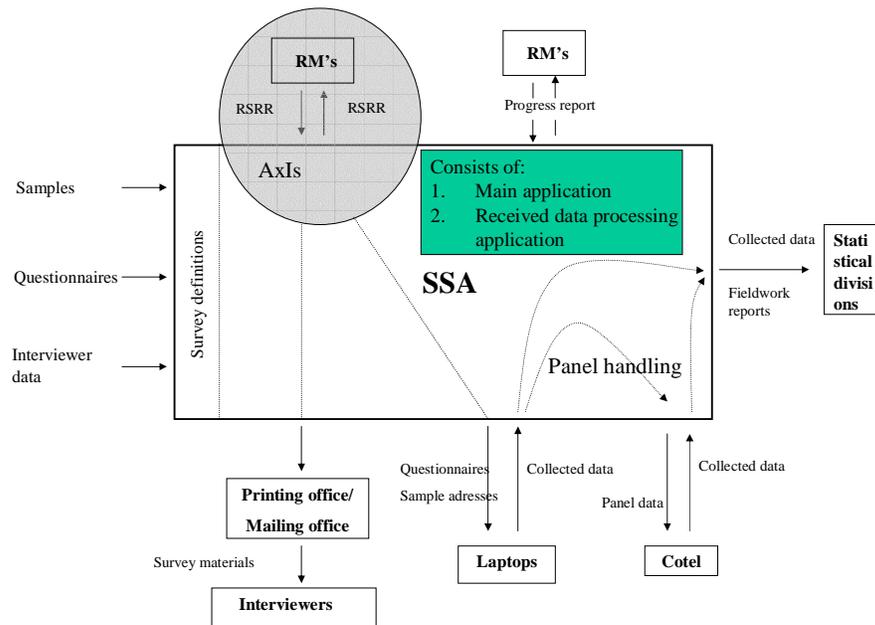
Data may be collected in two new ways: by e-mail or by Web-interviewing. The Electronic Data Reporter organises data collection via e-mail and will be reviewed briefly.

One statistic within the IMPECT project measures yearly turnover figures. Data collection for this statistic is executed with Web interviewing.

## 5. The SSA administration for person and household surveys

In the past, administrations have been a mix of Oracle, Paradox, Blaise and other applications. The putting out of interviews to laptops of field interviewers and the CATI server at Statistics Netherlands has been a process that needed a lot of manual actions.

The SSA (System for Survey Administration) project puts an end to the multitude of administrations and manual actions. The two biggest surveys: POLS (life situation survey) and EBB (labour force survey) are CAPI surveys that have a follow up with CATI. These and other surveys have been accommodated in SSA.

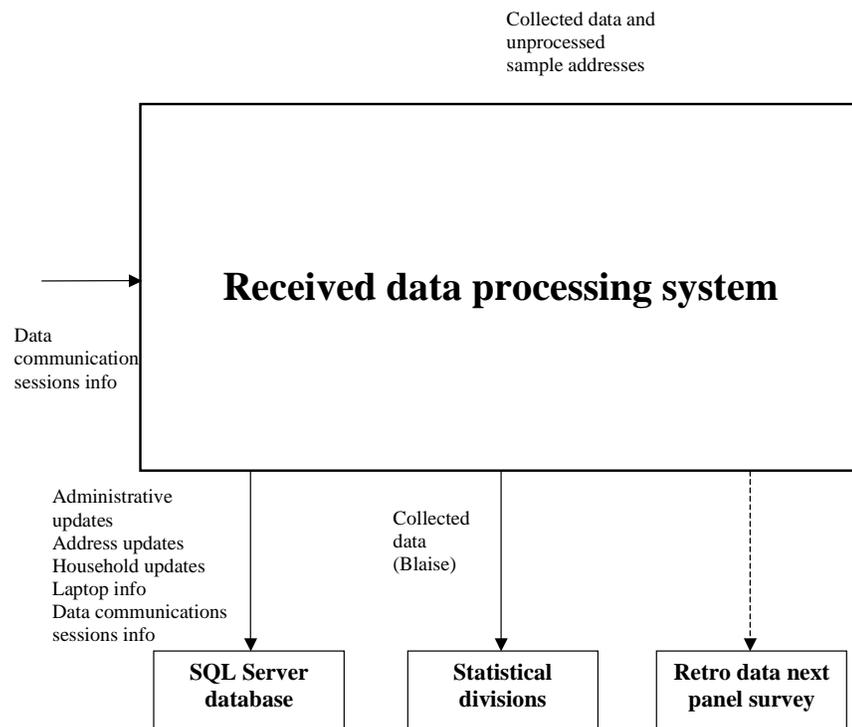


The previous blueprint of SSA shows how it is organised. The survey design of every survey is included in SSA.

The system input consists of the sample (elements), the questionnaire and the interviewer data. Based on the survey design, the system determines when a record is to be activated. It does not only cover the organisation of activating a record, it also executes the sending of interview materials to the interviewers. It loads data on the laptop, collects and processes information that comes back from laptops in the field, creates a regional overview about carrying out interviews, generates fieldwork progress reports and organises that interview forms are guided from CAPI to CATI. After finishing data entry, records are guided to classification procedures (if needed).

The SSA functionality to put out sample elements to interviewers (CAPI and CATI) is written in VB, using Blaise Component Pack (BCP version 1). Completed interview records are guided to and stored in the central SQL-Server database.

When data are received from an interviewer there is a complex process that generates parallel actions on various levels, updating information in the SQL-Server database, producing data for statistical analysis and passing on records to additional panel waves or coding machines.



At this moment, most person and households surveys are part of the SSA system. Statistics Netherlands is still working on accommodating all person and household interviews in SSA.

The system controls the flow of CAPI and CATI interviews from one wave to another. Redirecting cases to systems for coding and typing is included in this system.

## 6. New developments for establishment statistics

The traditional way of collecting establishment statistics used to be by sending out paper forms, receiving the filled forms from the establishments and then entering the data in a Blaise or relational database, depending on the complexity of the data model. Blaise offers new additional methods for entering data by the establishments.

Software has been developed to use Blaise data entry on-site at the establishments. Establishments use EDR to enter the data 'at home' in electronic questionnaires and send them in by e-mail to Statistics Netherlands. The recently developed **EDR** (Electronic Data Reporter) replaces *EDISENT*. It uses some of the new Blaise 4.6 technology and is operational since the beginning of 2003.

Another data entry mode that is expected to be helpful is Blaise IS. Blaise Internet Services is in use now for collecting data on behalf of turnover statistics of establishments.

There are a lot of different establishment statistics. The IMPECT project is the integral project that aims to integrate these projects into one framework. To achieve the goals of IMPECT, Web-interviewing and using the Electronic Data Reporter for establishment statistics is on the wish list. Before focusing on EDR and Web interviewing, a short introduction of IMPECT might be helpful.

### 6.1 The IMPECT project

The full name of the IMPECT project is: "IMPlimentation of EConomical Transformation process". It combines a large number of establishment statistics and the tasks that are performed. IMPECT is a program for a number of projects. Some of the main goals are to speed up the whole process of making statistics and lowering the respondent burden.

#### 6.1.1 LogiQuest

The sub-project LogiQuest is the central data collection system. It generates paper questionnaires and electronic contact items. It generates data entry tools and organises data-entry and statement selection.

LogiQuest connects to a *Central Observation Database*. The *Contact Registration* of the LogiQuest system keeps track of all contacts with establishments. Whenever there is a telephone contact with an establishment, the caller can have a view on all relevant previous contacts.

LogiQuest generates establishment questionnaires in various modes. Information requests can be sent out either on paper, as an electronic Blaise questionnaire, or as a Blaise Web interview (for the time being, an HTML-form only).

LogiQuest manages the questionnaire server, sending the information request to the establishments and handling the returned questionnaires. These actions use new Blaise technology to generate questionnaires electronically.

#### 6.1.2 E-mail and Web interviewing

A number of projects deal with data entry modes for IMPECT: The Electronic Data Reporter replaces EDISENT. It enables suppliers of information to send in data with e-mail facilities. Another sub-project uses Blaise HTML forms for data entry. The current off-line HTML version is realised with Blaise IS and is in production now. It enables suppliers of information to fill out a form "at home". Once the form

is completed, it can be submitted using an internet connection. An on-line version of a questionnaire is not yet operational in production.

### **6.1.3 UniEdit**

A second IMPECT project called UniEdit has been realised to automate data checking and cleaning. It manages error detection in data, automated data corrections, automated editing and (if other methods failed) manual editing. Suspect data may be imputed on estimations or may send a signal to replace the data at the source. Automated rules checking of data, cleaning up data and imputing missing or inconsistent data save time, replacing manual procedures.

In this process of data checks and imputations, Blaise comes in on many stages: for statement selection, error detection and correction, automated editing and interactive editing, analysis, and not in the last place for generating new Blaise source for sub-data models, based on selective fields of existing data models. The *SLICE* tool checks on irrational data combinations and corrects them automatically (if possible). It is written in C++ and uses Blaise Component Pack to link to the Blaise Data and Blaise rules. If the automatic corrections are not successful, manual corrections are carried out with Blaise data editing machines.

### **6.1.4 The analysis project of IMPECT**

Analysing establishment data is the last step before publishing statistics. However, it is an illusion to think that analysing is a straightforward separate process *after* completing data collection, data checks and corrections. The analysis is an ongoing process that starts as soon as the first results of data collection are available. The analysis results show stepwise advancements, based on intermediate results of the last response and the last corrections that have been made in the data. Statistical corrections of the data involve improved methods for applying imputations (automatically *and* manually), weighing, detecting outliers and checking them with external data such as value added tax data from the tax office. A part of the checks and corrections are done with Blaise related tools. *Bascula* is the Blaise related tool for weighing. The acceptance uses SPSS procedures. After correcting, data are transformed into statistical variables.

The required speed for accurate production of statistics asks for this approach in which intermediate results are available in “stand-by” mode. Finally, if the data are accepted, the “final touch” is put and the data are ready for statistical publication.

## **6.2 EDR: the Electronic Data Reporter**

More and more, data collection makes a shift from paper questionnaires to the Electronic Data Reporter and Internet interviewing. With new Blaise technology these methods have been applied successfully. Statistics Netherlands is also experimenting with touch-tone data entry for establishments.

The Electronic Data Reporter (EDR) is an advanced data collection module, and is mainly used for establishment statistics. The module uses new Blaise 4.6 technology. Except for the launcher (a small C++ program), the EDR module has been developed completely in Blaise. It is operational at Statistics Netherlands since the beginning of 2003.

The Electronic Data Reporter has been developed to replace the EDISENT module. EDISENT had a number of restrictions, it was build for a 16-bits environment and therefore, it is difficult to maintain and difficult to extend. EDR has been set up in a more flexible way. EDR applies the Blaise language for questionnaire definitions. This increases the flexibility of questionnaire design. The Electronic Data Reporter must be installed on-site at the establishment (or other supplier of information).

The installation can be delivered on a CD-ROM and includes the Blaise run-time engine for data entry.

The software can be extended and updated by e-mail. Different statistics may use the same Electronic Data Reporter. The questionnaires can use different code lists e.g. for looking up answer values.

As code lists can be shared, they will be stored only once, also when they are used in more surveys. This reduces the claim on the disk space of the supplier of information when a code list is large. Tests with code lists containing over 35.000 elements did not affect the performance in any way.

Every questionnaire may now contain its own hierarchy of keys ten levels deep. All kinds of key types (predefined using a lookup or just open) are supported. Predefined values for keys can be maintained by the data collector when sending questionnaires. Routing and checking is available for open keys. The Electronic Data Reporter offers complete statement management including duplicates.

### **6.3 An establishment statistic featuring Blaise IS**

One of the survey themes for economical establishment statistics is about the yearly turnover of establishment code groups. The survey used to be carried out with paper and pencil forms sent to the establishments. The establishments included in the sample used to be addressed by ground mail, if needed with reminders by telephone.

New Blaise technology made it possible to experiment with Web interviewing. This is done in the IMPECT project with Blaise IS 1.1. Establishments are asked to supply information for statistics on volume of trade. Compared to most of the person and household studies, these questionnaires are very small, about 5 or 6 questions, and in some cases the questionnaire may exist of one question only, in simplified terms: "what was your turnover in 2003".

If an establishment does not reply on the first electronic request to fill in a form, 2 reminders may follow to win them over to participate in the survey and become a member of the responding population. If this is not successful, then the specific cases are handed over to the survey management and they might use "old-fashioned" methods to address the establishment in another way to get the data, or they may decide to ignore the non-respondent and impute values based on estimations. This depends on the amount of non-response for establishments in the same code-group and the weight of the expected turnover of the specific establishment.

Blaise IS is not to be regarded as being the replacing methodology, but merely an addition to data collection by mail, telephone and e-mail. It takes away a lot of reminding and paper work. With the Blaise IS method, the response comes in faster than when using ground mail and telephone only. After one week, 30% of the establishments responded and response at the end became 70 to 75%.

### **6.4 Conclusion on Blaise IS and EDR for establishment statistics**

It is hoped that these new methods decrease the total amount of time needed to produce statistical figures. For specific establishments, the paper and pencil method remains a popular means of passing on information. Reminders are still necessary to get the required amount of response and only time can tell what response rates the new Web and e-mail way of data entry will have in the future.

So far, the traditional CAPI, CATI and CADI data entry of paper forms are being used as long as a certain proportion of response is not coming in via EDR or Blaise IS applications.

Nevertheless, the basic conclusion on EDR and Blaise IS is that more and more establishments are moving away from the old paper and pencil method and that they welcome EDR and Blaise IS for data entry. The merit for Statistics Netherlands is that this decreases the amount of data entry, paper work and correspondence and it may help to have response in a smaller amount of time.

## **7. Summary and Conclusion**

At Statistics Netherlands, Blaise is the standard tool for data entry. Yearly, over a million and a half data records are entered with Blaise CAPI, CADI and CATI machines.

New Blaise technology has been used successfully to integrate surveys for person and households into one system. The Blaise Component Pack made it possible to construct the System for Survey Administration in VB, connecting to Blaise meta and data. This system substantially reduced the amount of manual work in the organisation of the person and household surveys.

Data manipulations to prepare data for analysis are performed a lot with Blaise tools like Manipula and Manipulus (there are about 1 million calls to Manipula each year). These Manipula procedures may be replaced with BCP applications in the future.

New technology and new functionality of Blaise 4.6 has been used successfully in the IMPECT project and other projects to achieve a more efficient production of establishment statistics.

Relevant data entry applications here are the Electronic Data Reporter as well as Internet applications created with Blaise IS. Both methods (e-mail and Web interviewing) are used to collect data for short term establishment statistics.

Data corrections, checking plausibility, weighing, checking for outliers, and many other things are arranged in Blaise applications, Bascula and the SLICE application that uses BCP technology in a C++ application.

The options to combine Blaise applications with others, like C++, SQL-Server and SPSS are increased with Blaise Component Pack. The Blaise software with its specific characteristics for statistical surveys, its data model properties, data entry machines and dynamic routing and checking mechanism keeps playing an important role at Statistics Netherlands.



# Displaying Chinese Characters In Blaise

*Gina-Qian Cheung & Youhong Liu*  
*Institute for Social Research, University of Michigan*

## 1. Introduction

A Blaise data model can be multilingual. This is important for interviewing multi-language populations. One of the projects conducted by the Institute for Social Research at the University of Michigan is the National Latino and Asian American Study (NLAAS). There are five eligible languages in the study – English, Spanish, Chinese, Vietnamese, and Tagalog (Philippine native language).

While three of the languages are based on Latin characters, Vietnamese and Chinese are much different and more difficult to handle. For Vietnamese, it is required to install a Vietnamese font on the computer and then set a font representation in Blaise mode library that point to this Vietnamese font. For example, you can set @V =Vptimes. In Blaise field text, we can just put @V around the Vietnamese text. There are many details about how to prepare a Vietnamese document to be used on Blaise field text and how to do Vietnamese text fills etc. This will not be discussed in this paper.

In this paper, we will concentrate on Chinese language, one of the five languages utilized in the study. We will present how to use the resources available to add Chinese capability to the Blaise Environment. Please note that we only describe one of the possible approaches on the Windows platform (English version), other alternatives also exist.

## 2. Chinese Encoding Methods

First, we need to give some background about the basis of Chinese encoding system.

Encoding involves mapping a character to a numeric value so that the character can be identified through its associated numeric value. Computer systems process data in terms of bits, the most basic units of information processing. Bits are mapped to the 1 (on) and 0 (off) and are grouped together into units called bytes. Bytes can be composed of 7 or 8 bits. 7-bit bytes can allow up to 128 unique combinations while 8-bit bytes up to 256 unique combinations. While these numbers are good enough for encoding most writing systems of Western languages (such as the ASCII character set), with tens of thousands of distinct characters, they are far from enough to represent the writing system of Chinese (and those of other East Asian languages such as Japanese and Korean). The solution to this problem is to use multiple bytes to represent a single character. For example, an 8-bit 2-byte system can encode up to 65,536 (256x256) characters. This is known as a Double-byte Character Set (DBCS).

Two major computer encoding methods are used for Chinese: Big5 and GuoBiao (GB). Big5 encodes traditional characters and is used in Hong Kong and Taiwan, while GB encodes simplified characters and is used in Mainland China and Singapore. Another encoding method, Unicode, can be used in most of the world's major languages, including both simplified and traditional Chinese.

The computer does not know how to display the Double Byte characters. If you open a document that was created using one of these encoding methods on your

English Windows computer, all you will see is a bunch of gibberish. To display the characters properly, you will need a localized version of Windows (e.g. Chinese Windows System) or a software program (such as TwinBridge or NJStar), which acts like an interpreter – watching for double-byte characters and jumping in and converting the encoded characters into Hanzi (Chinese characters) as they come up.

### **3. Our Approach to Displaying Chinese in Blaise**

There are several approaches to handle Chinese on computers. One technique is to have the entire operating system support Chinese. This is the most popular option in which the user only deals with Chinese and no other languages. Microsoft sells both traditional and simplified Chinese versions of its Windows operating system. In this case, it was not feasible to use a Chinese operation system, because our users are primarily English based. Our approach is to use a program to add Chinese capability to Blaise. We chose NJ Star, but there are many other programs available in the market, such as Twin Bridge and Chinese Star, etc.

### **4. Steps to Input Chinese Word Specs to a Blaise Source File**

Suppose we have an existing Blaise English source file, if we have a corresponding Chinese word document, how do we input Chinese into the source file? We cannot copy and paste the Chinese text directly from the word file to the Blaise editor. If we do a direct copy and paste, all the Chinese characters will become “????????” in Blaise file editor because of the encoding problem. In order to input a Word file into a Blaise source file, we need to do the followings:

1. Converting the MS-Word document to an encoded text file
  - a. On the File menu, click Save As
  - b. In the File name box, type a new name for the file
  - c. In the Save As type box, select Encoded Text
  - d. Click Save
  - e. Click Yes to discard any formatting and save as text
  - f. Select Other Encoding, and then select “Chinese Simplify (GB2132)” or “Chinese Traditional (Big 5)” in the list. Please note that you should choose one of the two encodes according to the file encoding used, i.e. the input encoding used for the file. If you select “Chinese Simplify (GB2132)”, but the file uses Big 5 encoding, the result will be wrong. You can also preview the text in the Preview area to check whether it makes sense in the encoding standard you have selected (If the Preview area is not visible, click Show Preview).
2. Adding space between Chinese characters

The Blaise DEP expects to find spaces to break the text into words and lines. Since the encoded Chinese file converted from MS-Word does not use spaces, the Blaise DEP may break up the Chinese text in inappropriate places, including in the middle of a double-byte character, or not break the line at all, leaving a large run of text off beyond the edge of the window. Actually, the early versions of Netscape and Internet Explorer had this kind of problem. To help proper formatting of Chinese text, a space is needed between each character; this will allow Blaise Dep to find an appropriate

place to break the text into lines. Most Chinese software programs have a function to add a space in between the characters. We used NJ Start Communicator to conduct this task.

### 3. Inputting the Chinese text into the Blaise source file

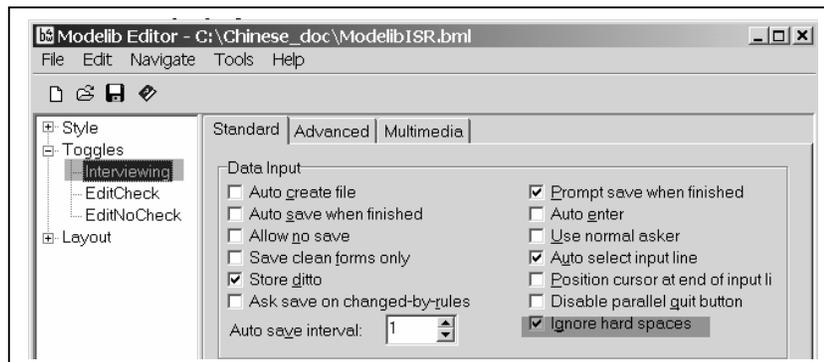
Since it is time consuming to copy field by field to the Blaise source file, UM developed a utility - Foreign Language Merge Utility (FLMU) - that helps to insert Chinese into the Blaise source code. This utility can also be used for any other languages. Using this utility can save a substantial amount of time.

### 4. Handling Blaise Control Characters

In the Blaise Dep, character “@” is a font or format indicator and “^” precedes the field or the variable name whose value will be included in the field text. In the Chinese encoding, “@” and “^” both can be part of a Chinese character. In this case, we must type them twice. We can do “Find and Replace All” to the text file before we use the FLMU to insert text into the Blaise source code. It is still difficult to remember typing these two characters twice when manually inserting Chinese text. Later, it was found that “@” and “^” are used in Big 5 encoding, but not in GB encoding. So we decided to use GB encoding for our instrument. This made our job far easier. If the documents we received use Big 5 encoding, we can use NJ Start Communicator to convert them from Big 5 to GB encoding.

To use a hard space in Blaise field text, normally we can type <Ctrl-period> in the Blaise editor. Unfortunately, this character is also used by some of Chinese characters, both in Big 5 and GB encoding system. If we type <Ctrl-period> twice, it will only instruct Blaise Dep to display two spaces. To solve this problem, in Blaise Mode Library, we can choose to ignore hard spaces (Figure-1). So this means, in a Blaise instrument that includes Chinese characters or any character set that has <Ctrl-period> character, we can no longer use hard spaces in field text.

Figure 2 – Ignoring hard spaces in Blaise Mode Lib



## 5. Chinese Fills in Blaise

In Blaise, fill holds the value of another field, and inserts it into the question text. You can test the current language with the key word *ACTIVELANGUAGE*. You might use the following example to determine which fill to use in language text:

```
IF ACTIVELANGUAGE = CHN THEN
  Fill1 := '你好, 世界!'
ELSE
  Fill1 := 'Hello, world!'
ENDIF
```

Where Fill1 may be a fill in a question text:

```
HelloWorld "^Fill1 This is English."
"^Fill1 This is Spanish."
"^Fill1 This is Tagalong."
"^Fill1 This is Vietnamese."
"^Fill1 这是中文。" : STRING[10]
```

The Chinese question text display for the field HelloWorld will be:

```
你好, 世界! 这是中文。
```

## 6. Switch Language During Interview

Blaise provides a default menu entry for language switching. When a case is suspended during the interview, it will use the first language the next time the case is invoked again not the language you used last time. This is not desirable. We decided to use a parallel block to switch languages. In this parallel block, *SETLANGUAGE* command is used to switch between languages. This works very well, not only we can keep the suspended language setting, but also store the language being used during interview in a array so that the data can be used for analysis later.

It is possible that the parallel language-switching block may create a conflict with the default menu language switching. To avoid this conflict, the default language switching can be disabled (Figure 2). A menu item can be added that invokes the language switching parallel block (Figure 3).

Figure 3 – Disabling Default Language Switch in Blaise Menu

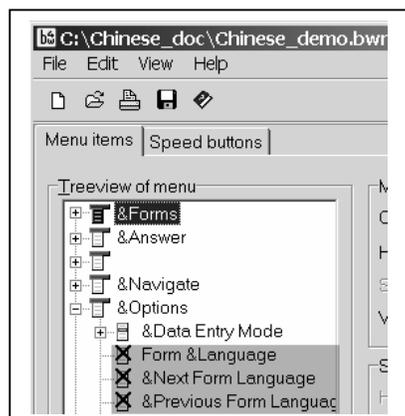
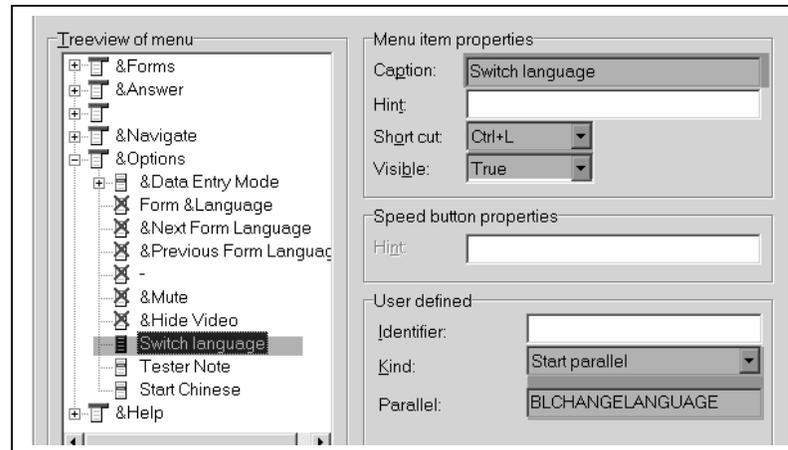


Figure 3 – Adding Parallel Language Switch Block in Blaise Menu



## 7. Setting Menu Item For Chinese View

During a Chinese interview, the interviewer needs to switch the interview language to Chinese, but also has to turn the Chinese Viewer on. Figures 4 and 5 show the difference with and without a Chinese Viewer. To allow quick access to the Chinese Viewer, a menu item and a hot key can be added, e.g. F6 to enable the Chinese viewer (Figure 6).

Figure 4 – Displaying Chinese without Chinese Viewer



Figure 5 – Displaying Chinese with Chinese Viewer

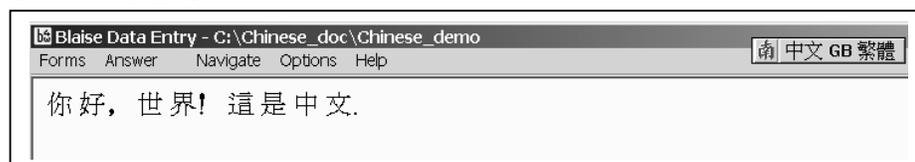
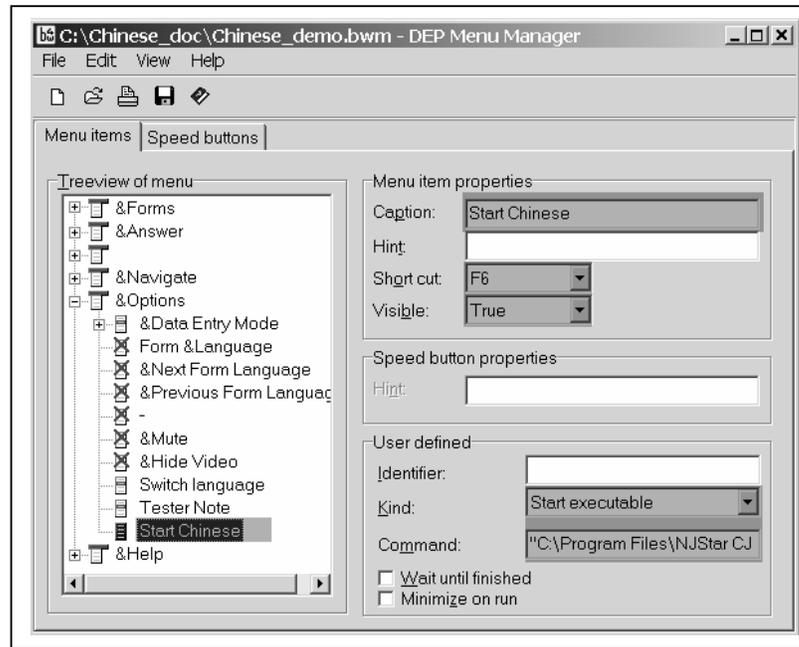


Figure 6 – Adding Chinese Viewer Access to Blaise Menu



## 8. Summary

Displaying Chinese in Blaise is different from other languages because it requires a two-byte encoding system, but it is not a difficult task if the proper tools are used and the right procedures are followed. At the University of Michigan, the multi-language study, National Latino and Asian American Study (NLASS) was deployed in May 2002. The interface was well received by both field staff and the study managers.

## 9. References

Statistics Netherlands Blaise Developer's Guide  
<http://users.erols.com/eepeter/chinesecomputing/encodings/>  
<http://www.njstar.com>

## *Data Quality and Testing*

- **Analyzing Audit Trails in the National Survey on Drug Use and Health (NSDUH) .....155**  
*Michael Anthony Penne & Jeanne Snodgrass, RTI International & Peggy Barker, SAMHSA*
- **A Random Walk Application for Blaise Instruments ...175**  
*Gilbert Rodriguez & Jay R. Levinsohn, RTI International*
- **First steps along the Audit Trail.....183**  
*Tim Burrell, Office for National Statistics*



# Analyzing Audit Trails in the National Survey on Drug Use and Health (NSDUH)

*Michael Anthony Penne & Jeanne Snodgrass, RTI International & Peggy Barker, SAMHSA*

## Abstract

The National Survey on Drug Use and Health (NSDUH) (formerly the National Household Survey on Drug Abuse) is one of the largest government-sponsored CAI household surveys conducted in the United States on a continuous basis. Beginning with the 2002 NSDUH, Blaise audit trail files were retained for each interview conducted. For each screen encountered during the interview, these files retain the date and time of entrance and exit, the response entered and the keystroke used to move to the next screen. Currently, we are using these audit trail files to examine the following data quality issues: 1) instances where respondents back up and change previously recorded answers and the potential effects on estimates 2) areas within an interview where respondent break offs occur and 3) the amount of time required to complete various sections and individual questions in the interview, focusing primarily on unusual response times and possible explanations for these outliers. We analyzed the data on a flow basis as a means to rectify processing problems as they occur and to ultimately streamline our audit trail processing and analysis methods for future years. During data processing and these analyses, we have discovered future areas of analysis that may be of interest to those examining data quality through audit trails.

The current paper will present the results of our analysis of the 2002 NSDUH audit trail files regarding impact on estimates of prevalence and recency of substance use; break offs and timing data; methods for streamlining both data processing and analysis; and potential for further data quality analysis.

## 1. Introduction

Early research on the National Household Survey on Drug Abuse (NHSDA) computer-assisted interview (CAI) demonstrated the value of using audit trail files to evaluate the ease with which respondents navigated through the audio computer-assisted self-interview (ACASI) portions of the questionnaire (Caspar 1997; RTI, 1997). Building on this initial work, in 2002 we used audit trail files from the first six months of the 2002 survey to further investigate data quality issues. In 2003, we conducted the same analyses on all 2002 NSDUH audit trail files to determine if increasing the sample size created differences in our data. In this paper, we briefly describe the early work, discuss possible methods for streamlining the data processing portion, then focus on use of audit trails in the 2002 survey to investigate three aspects of data quality: question timing, respondent break-offs, and changes in estimates of recency and prevalence of substance use.

### 1.1 Background and Preliminary Analysis

In 2002, the survey name was changed from the National Household Survey on Drug Abuse (NHSDA) to the National Survey on Drug Use and Health (NSDUH). The NSDUH, sponsored by the Substance Abuse and Mental Health Services Administration (SAMHSA), is the federal government's primary source of information on the magnitude and correlates of substance use and abuse in the United States household population. The NSDUH produces annual estimates of

rates of use, numbers of users, and other measures related to alcohol, tobacco, illicit drugs and the non-medical use of psychotherapeutic drugs in the civilian, non-institutionalized population of the U.S. The NSDUH collects data by administering questionnaires to a representative sample of persons aged 12 and older at their place of residence. The questionnaire contains both respondent self-administered sections and interviewer-administered sections (SAMHSA, 2002). Research Triangle Institute (RTI) is under contract to SAMHSA to conduct the NSDUH.

In 1999, the NHSDA data collection method was changed from paper-and-pencil (PAPI) to CAI. The interview sections on substance use and other sensitive topics were changed from self-administered PAPI to ACASI. These changes were prompted by research which showed that CAI questionnaires reduced input errors. Research also showed that use of ACASI increased comprehension for less literate respondents, and, by increasing privacy, resulted in more honest reporting of illicit drug use and other sensitive behaviors (Lessler et al, 2000). At the same time, the sample size was increased almost three-fold, to approximately 70,000 persons per year.

In preparing for the CAI conversion, in the fall of 1996, RTI conducted a field test with 435 respondents to compare two CAI versions of the 1996 PAPI. As part of this experiment, an MS-DOS Blaise program was modified to capture each keystroke made during the course of the interview by both the respondent and the field interviewer (FI). Additionally, the program captured timing data for each screen encountered and an indicator of whether the audio or screen component was turned on or off. Analysis of these keystroke files centered on the respondents' interaction with the ACASI program. Keystroke file analysis from both CAI versions indicated that roughly 60% of all respondents utilized a function key at least once, with very little difficulty. Overall, the analyses suggested that respondents would not experience any difficulty while interacting with the NHSDA ACASI portion of the instrument (RTI, 1997).

With the conversion of the NHSDA questionnaire to CAI, the Blaise programming language was also changed from an MS\_DOS based version to a Windows-based version, and the keystroke file application was no longer compatible. This version of Blaise did, however, contain code that would capture data similar to the previous keystroke file application. These new files were called "audit trails".

Interest in capturing this type of data was renewed after the first nationwide administration of the CAI NHSDA instrument in 1999. Issues surrounding changes in data collection mode prompted renewed interest in audit trail files analysis, and the decision was made that some portion of audit trail files should be retained for analyses. Since a sample size of approximately 25,000 audit trail files would be more than adequate to answer any proposed questions, one in every three transmitted audit trail files was retained starting with the 2000 survey year<sup>1</sup>.

Analysis of the 2000 NHSDA CAI instrument audit trail files was done to validate the findings from the 1996 keystroke file analysis. Sample sizes for this analysis incorporated a little over 12,500 audit trail files (one third of the data from two calendar quarters). The results validated the 1996 findings that respondents were having little difficulty completing the ACASI portion of the questionnaire (Caspar, 2000).

---

<sup>1</sup> A single transmitted file does not imply either a single interview only or an interview in its entirety. Field Interviewers are required to transmit any completed work nightly for every day that they work. This implies that a single transmission may contain more than a single interview. Additionally, if an interview experienced a breakoff, the first portion of the interview would reside in one transmission file while, assuming that the interview was eventually completed at a later date, the remaining portion of the interview would reside in an entirely different transmission file.

Through the 2000 survey year analysis, we encountered problems with only retaining one-third of the transmissions. Most notably, when a breakoff occurred within an interview but was completed at a later date, there were instances where we retained the last portion of an interview, but not the initial portion, and vice versa. As a result, the level of partially completed interviews for audit trail analysis was high, which made it difficult or sometimes impossible, to use the data. This problem was large enough, and the interest for further data quality analysis was strong enough, to warrant retaining all transmitted audit trail files. Conclusions about the difficulties associated with retaining one in every three transmitted files were not made until well into the 2001 survey year, and since the questionnaire structure could not be changed to retain all transmissions at that time, the change was implemented at the beginning of the 2002 NSDUH survey year.

In 2002, attention turned toward an investigation of audit trail analysis to use for possible data quality measures. To assist us in preparing for this analysis, we used the 2001 audit trail data to plan, develop, and test both our analysis methods and programs. We then tested on the first six months of audit trail data from the 2002 NSDUH (Penne, Snodgrass and Barker, 2002).

## 2 Current Issues and Methods

All current analysis methods and results presented in this paper are from the entire 2002 NSDUH survey. This analysis includes all completed and partially completed interviews, regardless of final response or usable status. An interview is classified as “usable” if the respondent provided data on lifetime use of cigarettes and at least nine other substances (SAMHSA, 2002). Since our analysis issues deal only with respondent interaction with specific variables within the questionnaire, as opposed to the entire questionnaire, we felt that all data, no matter what its case status, should be used for each analysis. Prior to analyzing the 2002 NSDUH audit trail data, we outlined each method for examination.

### 2.1 Timing

As was the focus of the timing analysis in our earlier studies of keystroke/audit trail data, our initial purpose for this analysis was to monitor respondent difficulties in utilizing ACASI. Though not presented in this paper, our results were the same as before: respondents were not having any more than expected difficulties in navigating the questionnaire. This reassurance allowed us to turn our focus towards FI performance and what possible ramifications it might have on respondent behavior.

With audit trail data, timing can be calculated for any screen or group of screens within the interview. It can also be calculated up to a hundredth of a second. Timing data collected with timestamp data within the Blaise program is limited to entire sections or modules, and is only calculated to a tenth of a second. We chose 6 measures within the NSDUH instrument to examine FI behavior with regard to timing:

Title	Screen Name	Description
Introduction to CAI	INTROCAI	The interviewer begins rapport with the respondent, and reads them the introduction to the study, if applicable
Calendar Set-up	CALENDAR	The FI sets up the reference calendar for the respondent to use throughout the interview.
ACASI Set-up	INTROACASI	The FI explains the Audio CASI set up to the respondent, adjusts the headphones and starts them on the tutorial.

End of ACASI	ENDAUDIO	The respondent finishes the ACASI portion, and turns the laptop over to the FI, who enters a 3-letter code to continue.
Verification Form	TOALLR3I	The FI interacts with the respondent to complete the verification Completion form.
Ending Interview with Respondent	INCENT01	The FI ends the interview with the respondent, and gives the respondent a cash incentive for their participation.

These single question measures were present in every interview, as opposed to all other questions in the CAI that are asked dependent upon skip logic, and we felt it would give us insight into how long FIs are spending on these important aspects of the questionnaire. Our analysis entailed calculating the amount of time spent on a question deemed appropriate for a respondent to fully comprehend the information on that particular screen, and then comparing the percentages of FIs falling below this amount of time. This allowed us to get a sense of the magnitude of shortcutting. The Gold Standard (GS) timing was developed for each of these six measures, at RTI, with 3 NSDUH staff members administering the NSDUH CAI instrument to 3 non-NSDUH staff members who ranged in age from 27 to 32. The interview files were sent to a Blaise programmer, who calculated the timing for each of the 3 interviews. In all six measures the highest timing across the three interviews was taken as the GS.

## 2.2 Breakoffs

In the breakoff analysis, we wanted to explore, 1) magnitude of breakoffs overall, 2) if there is any particular section of the interview where more than the average number of breakoffs are occurring, possibly indicating that respondents are having trouble or are being offended by the questions, and 3) if there are any interviewers who account for a large percentage of the overall number of breakoffs. We looked at all instances where a breakoff occurred within the interview. We examined the overall number of occurrences (total interviews with at least one breakoff), the number of breakoffs associated with individual FIs, and the location within the interview of the breakoffs. With regard to location, we classified breakoffs into three main categories: ACASI, FI administered sections and lastly, instances where the CAI system itself crashed. The ACASI sections were further categorized as: Tutorial, Main Core Drugs, Psychotherapeutic Drugs and remaining Non-Core sections. FI-Administered sections were categorized as those with some respondent interaction (instances where the FI is actually administering the questionnaire to the respondent) versus no respondent interaction (e.g., confirmation of FI identification number, FI debriefing questions, etc).

## 2.3 Changes in Prevalence and Recency of Use

When the NHSDA was conducted in PAPI, respondents were asked every follow-up question in the core drug modules; even if they indicated they had never used a particular drug. Data editing procedures used answers from all follow-up questions to classify a respondent as a drug user or non-drug user<sup>2</sup>. With the switch to CAI, routing logic was implemented that skipped the respondent out of a drug module if they indicated at the beginning of the module that they never used a particular drug. With this implementation, the previous editing procedure was no longer possible. Though not a direct correlation with the previous editing procedures, the audit trail data does allow us the benefit of observing instances where a respondent

<sup>2</sup> Since skip patterns did not exist within the PAPI, a respondent was directed through all follow-up questions about their use of a particular drug, regardless of the respondent's actual use. Any indication of use of a particular drug classified the respondent as at least a lifetime user of that substance.

backs-up to a previously answered question and changes their answer. Our first step was to determine if in fact there were a sufficient number of instances where this occurred to have an impact on drug use estimates. Since true reasons for a respondent changing their answers are speculative at best, we took a worst case scenario approach to our analysis. In other words, we presumed that all changes in answers were reflective of the respondent's desire that no one know their true drug usage. With this in mind, we examined the audit trails to find instances where a respondent at any time indicated a positive response to using a particular drug. We wanted to know how much drug use estimates would increase when assuming that if a respondent ever said 'Yes' to using a drug, they were in fact a drug user.

We focused on two aspects of interest. First, we analyzed all individual lifetime use gate questions<sup>3</sup> to view the overall effect on lifetime drug use prevalence. Second, we examined the effects these changes might have on recency of use of a drug. In this analysis, we sought to determine the most recent period ever that a respondent indicated they had last used a drug regardless of their final response. For example, if a respondent initially reported past 30-day use of a drug, but later changed their response to past year, this analysis would finalize that respondent as a past 30-day user.

## **2.4 Data Management**

As noted earlier, starting with the 2002 survey year, all transmitted audit trail files were retained. Unfortunately this did not alleviate all of the data management difficulties. Specifically, the potential for the same FIs to replicate the same unique identification (ID) number for different individual interviews, or for two separate FIs to use the same ID, still existed. Additionally, since the audit trail analysis is currently separate from the questionnaire data analysis, corrections of incorrect ID numbers and subsequent linking of follow-up portions of an interview from an earlier interview breakoff were not applied to their respective audit trail files. Ultimately, this resulted in approximately 6.2% (~4,457 records) of our 2002 records having at least one duplicate ID represented on our analysis file. For processing expedience, and since we still retained a sufficient sample size to produce reliable estimates, we decided to remove these records with duplicate IDs from the timing, lifetime and recency analyses (n = 63,811). However, all records were considered within the breakoff analysis (n=68,268).

## **3. Results and Discussion**

The timing data in Table A1 shows that when measured against a Gold Standard (GS) time, FIs are spending approximately the correct amount of time with the very beginning of the interview, at the introduction to the CAI instrument screen. However, once past this point, they spend less time than the GS on several important aspects of the questionnaire, such as setting up the calendar, setting up the ACASI tutorial, completing the verification form and ending the interview with the respondent. Conversely, they are taking longer than the GS in ending the ACASI portion of the interview. There are at least two reasons for these results. The first might be that the FIs are not performing their jobs correctly. The shorter times might indicate that FIs are reading through the questions too quickly for the respondent to understand or get the full meaning of the question.

The second possible reason could lie with the method of the GS calculation itself. Since GS timing was calculated using simulated interviews at RTI, it did not capture the situations that affect timing in many real field situations. The GS

---

<sup>3</sup> Hallucinogens, inhalants and psychotherapeutic drugs (pills) each contain multiple gate questions. Each question focuses on particular substances classified as a hallucinogen, inhalant or pill. For instance, hallucinogens ask questions on LSD, PCP, peyote, angel dust, mescaline, psilocybin, "Ecstasy" and any other hallucinogen not already asked about.

calculation, then, could be seen as a little higher than what is realistic. Another limitation to the GS calculation is that it was not calculated by age. The ages of the mock respondents (27, 29, and 32) were not representative of the entire NSDUH survey population and may not serve as a realistic benchmark measure across respondent age groups. Means for determining the validity of these GS measures would entail either modifying the measuring criteria for each aspect or conducting field tests, with an adequate sample and appropriate age range. On another note, the longer time taken to end the ACASI portion of the instrument could simply be accounted for by rapport between the interviewer and the respondent. At this point in the interview, the computer is handed back to the interviewer, and the interviewer may stay on this screen (ENDAUDIO) while talking to the respondent, or answering any of the respondent's questions about the ACASI.

Though the reasons for these timing results are somewhat speculative and GS times require additional validation, we did notice a trend across our analysis age groups. As the age group of interest goes up, we tend to see a decrease in the percentage of FIs below the GS time. This indicates that FIs are taking more time with older respondents, who tend to be less comfortable with the CAI.

To further assist us in drawing proper conclusions about the interviewers' time to complete, we produced graphical distributions of the audit trail timing data for the 12 and older population<sup>4</sup>. These graphs show detail which Table A1 does not. For instance, in Table A1, it appears that the interviewers are equally distributed around the GS for the introduction to the CAI, but from the graph we see that the distribution is in fact bimodal. Further, we observe that roughly 1/3 of the sample is taking less than 2 seconds to complete this screen (1/3 of the GS time). We see these same patterns in later screens also (Verification Form Completion and Ending the Interview With the Respondent). These results indicate that short-cutting by the interviewer is a possibility. The data in Table A1 will be investigated further by the NSDUH data quality staff, along with many other indicators of data quality, for flagging interviewers whose work may warrant a closer look.

Tables A2.1 and A2.2 illustrate that interview breakoffs within the ACASI are minimal. It does not appear that there are specific areas in the questionnaire where respondents are breaking off, and the majority of the breakoffs are within FI administered portions of the interview. This suggests that while breakoffs do happen, we can't identify any particular questions that generate a significant number of breakoffs. Within the ACASI portion, the breakoffs are clustered in the tutorial and non-core sections, suggesting that if a breakoff does happen during the ACASI, it probably will occur right at the beginning or towards the end where respondents may be getting tired of answering questions. Respondents' reluctance to continue with a long interview may also explain why a majority of breakoffs are occurring in the FI administered sections, which are at the beginning and the end of the entire interview. It should be noted that as a result of not being able to resolve all issues of duplicate ID numbers, and retaining all files for the breakoff analysis, the potential for double counting of interviews and instances per FI does exist.

An interesting finding is in Table A2.3. This shows that there are 90 interviewers with 5 or more instances of a breakoff, and 21 interviewers with 10 or more instances, which might suggest possible interviewer problems. In this instance, the audit trail files could be used as an FI monitoring tool, to alert project staff of

---

<sup>4</sup> Initially, we produced graphs for each of the six timing measures for each age group (including an overall). Upon careful inspection, we noticed only negligible differences in the shapes of the graphs among the different age groups. The only noticeable differences occurred in the magnitude of the distribution, which was expected as a result of the varying sample sizes in each age group. Hence, only results for the 12 and older population are presented.

interviewers who may have problems interacting with the CAI instrument. In the NSDUH, this breakoff data from Tables A2.1-3, along with the timing data, will be used by NSDUH data quality staff in monitoring FI adherence to study protocols.

**Table A1. 2002 NSDUH Audit Trail Timing Analysis: 12+ Year-olds**

Modules of Interest	Gold Standard Time (Minutes)	Respondents 12+ (n = 63,811)			Respondents 12-17 Years-old (n = 22,221)			Respondents 18-49 Years-old (n = 36,645)			Respondents 50+ Years-old (n = 4,945)					
		Median	% Below	% Above	Median	% Below	% Above	Median	% Below	% Above	Median	% Below	% Above			
Introduction to CAI	0.10	0.10	49.81	2.00	0.08	53.32	2.07	44.61	0.10	48.47	1.92	49.61	0.17	43.88	2.29	53.83
Calendar Set-up	1.55	1.22	67.93	0.00	1.23	67.34	0.00	32.66	1.20	68.89	0.00	31.11	1.27	63.42	0.00	36.58
ACASI Set-up	2.23	1.48	74.62	0.00	1.52	74.21	0.00	25.79	1.45	75.95	0.00	24.05	1.70	66.63	0.00	33.37
End of ACASI Verification Form Completion	0.20	0.48	3.97	0.00	0.50	2.78	0.00	97.22	0.47	4.30	0.00	95.70	0.48	6.86	0.00	93.14
Ending Interview with Respondent	0.42	0.08	75.87	0.00	0.07	75.61	0.00	24.39	0.10	76.26	0.00	23.74	0.10	74.07	0.00	25.93
	0.52	0.05	84.98	0.00	0.05	84.42	0.00	15.58	0.05	85.44	0.00	14.56	0.05	84.11	0.00	15.89

**Table A2.1 Distribution Summary Of Breakoff Analysis**

<b>Total Audit Trail Files</b>	68,268
<b>Total Breakoff Instances</b>	1,563 (2.3%)
<b>Total Respondents</b>	1,411
<b>Single Instance</b>	1,293 (91.6%)
<b>Two Instances</b>	92 (6.5%)
<b>Three Instances</b>	20 (1.4%)
<b>Four Instances</b>	4 (0.3%)
<b>Five Instance</b>	2 (0.1%)
<b>Total Field Interviewers</b>	523

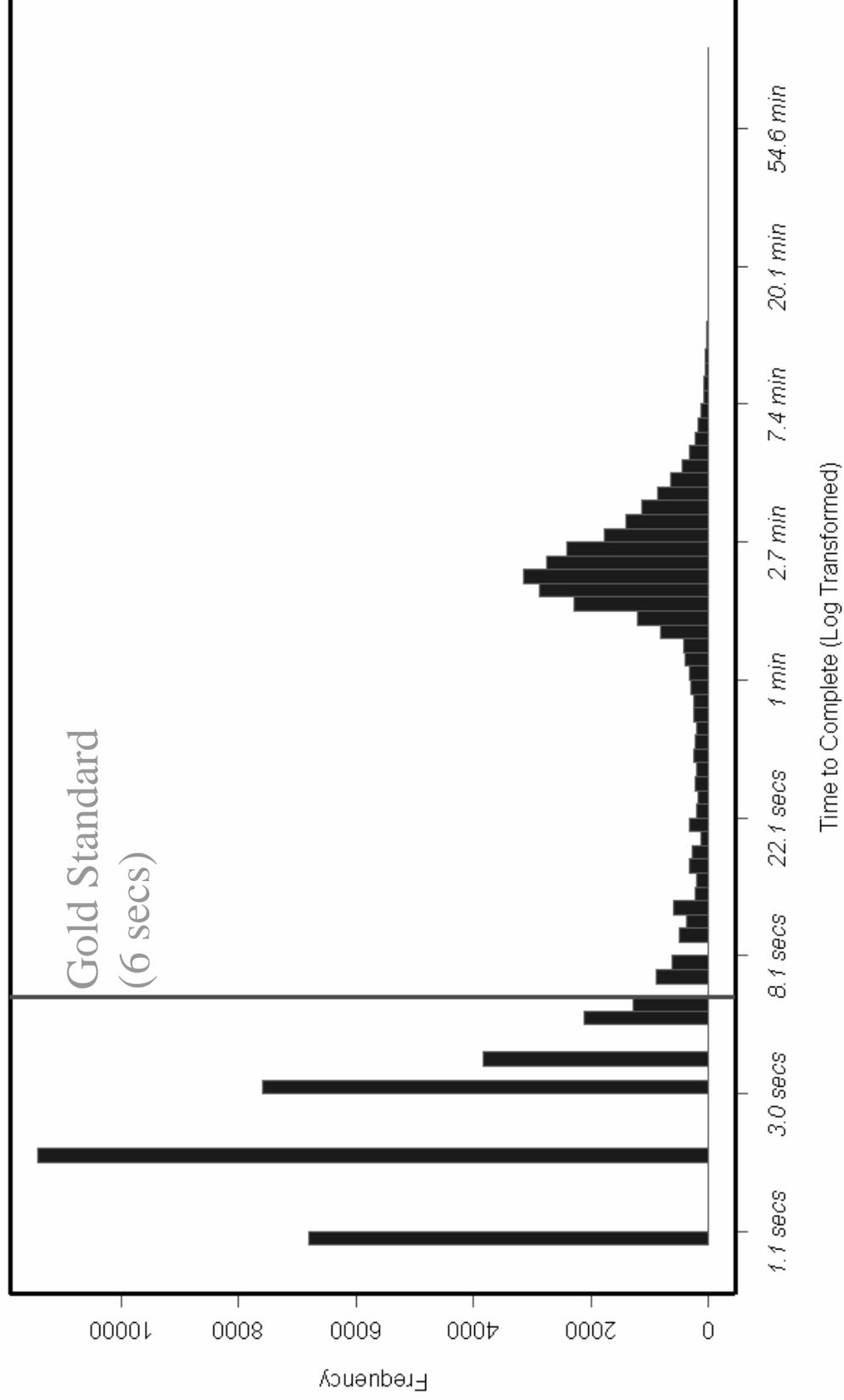
**Table A2.2 Distribution of Breakoffs by Field Interviewer Interaction**

Domain of Interest	N	% of Total Breakoffs	% Within Domain
Field Interviewer Administered	926	59.25	58.64
Respondent Interaction	543		41.36
No Respondent Interaction	383		
ACASI	631	40.37	29.95
Tutorial	189		16.16
Main Core Drugs	102		8.24
Psychotherapeutic Pills	52		45.64
Non-Core Sections	288		
Systems Crash	6	0.38	

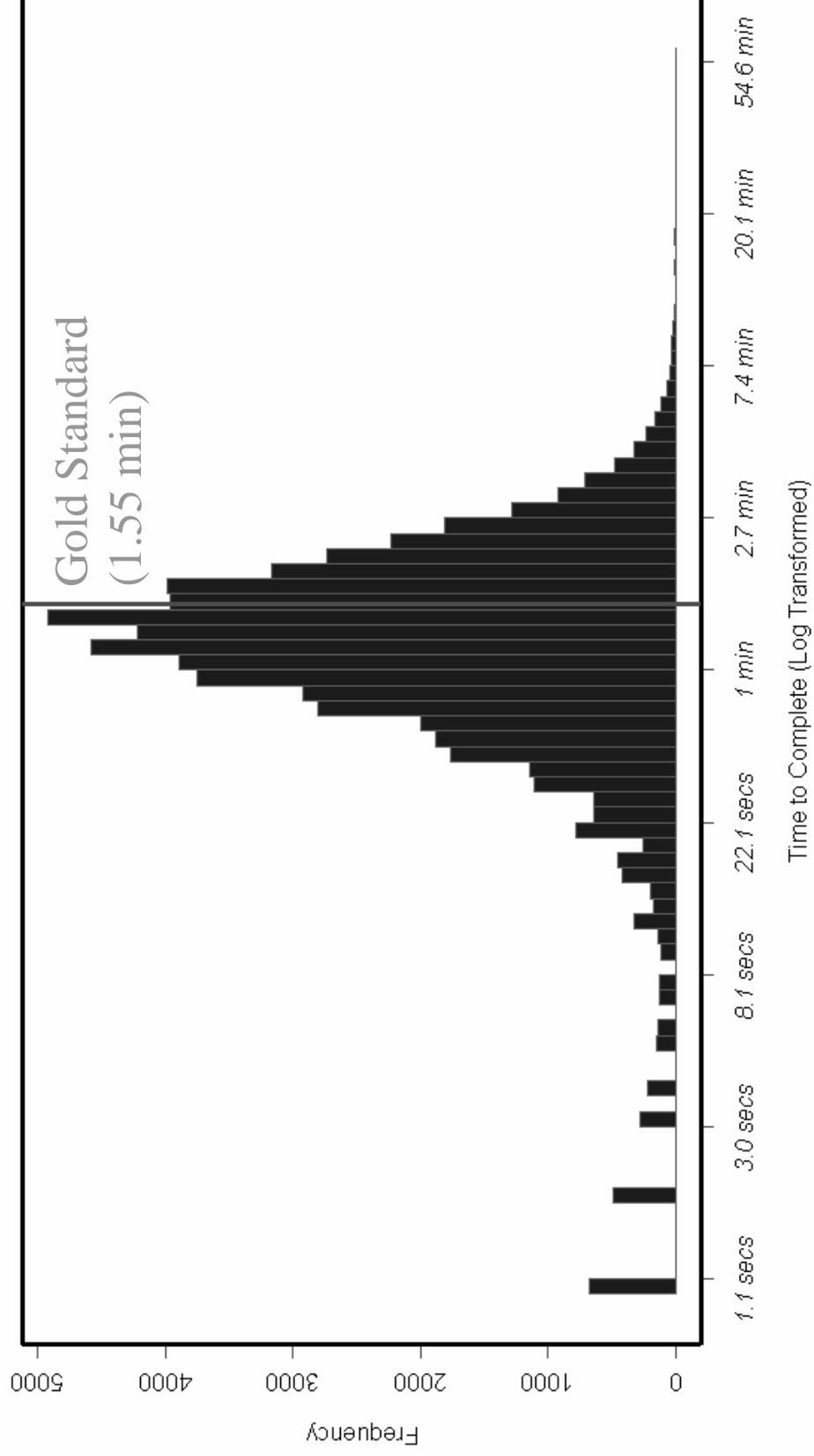
**Table A2.3 Distribution of Breakoffs Attributable to FIS**

# of Breakoff Instances	N	%
1	222	42.45
2	107	20.46
3	68	13.00
4	36	6.88
5	30	5.74
6	16	3.06
7	14	2.68
8	6	1.15
9	3	0.57
10	7	1.34
11	3	0.57
12-45	11	2.10

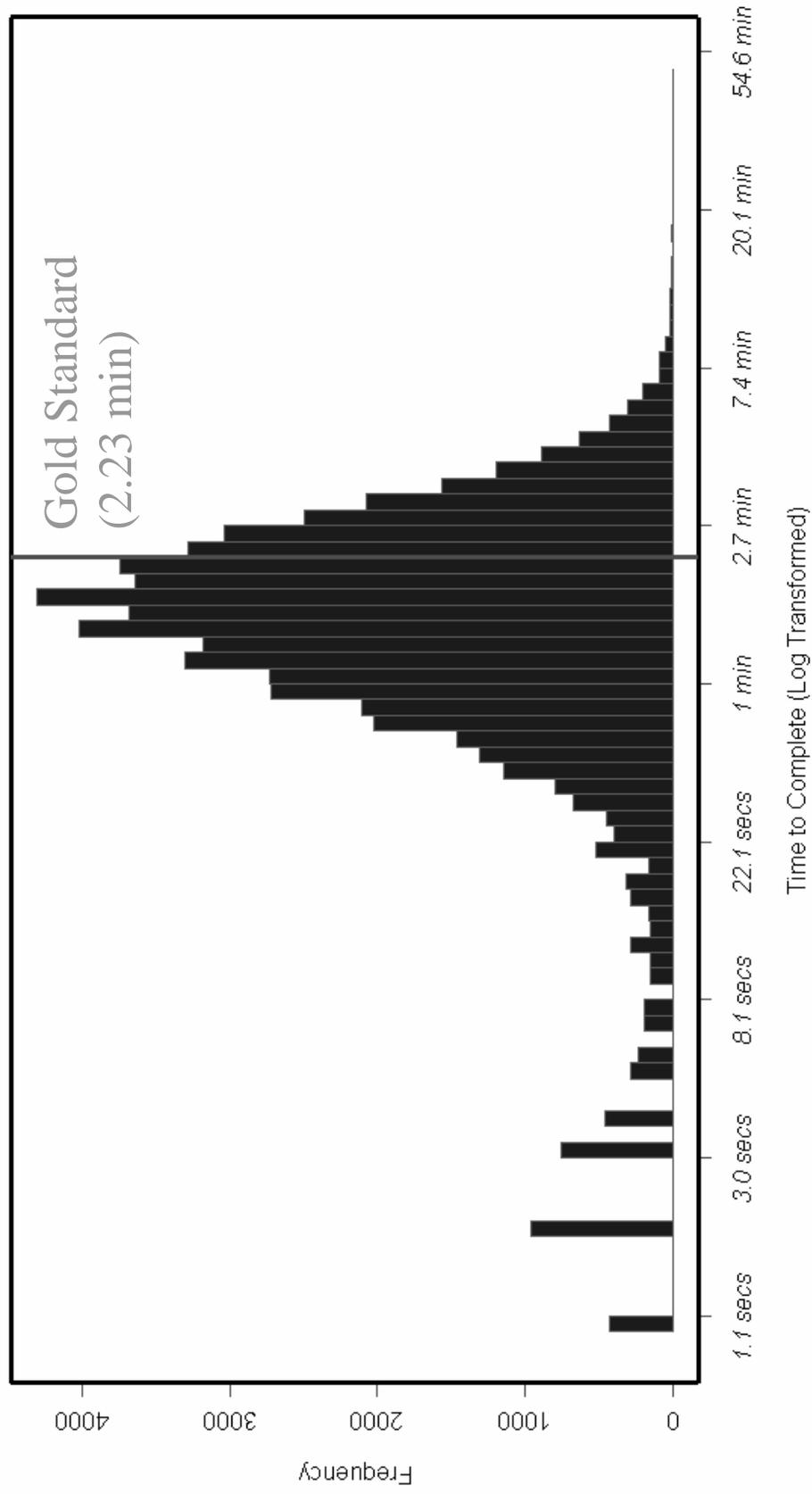
## Distribution of Audit Trail Timing Data: Introduction to CAI



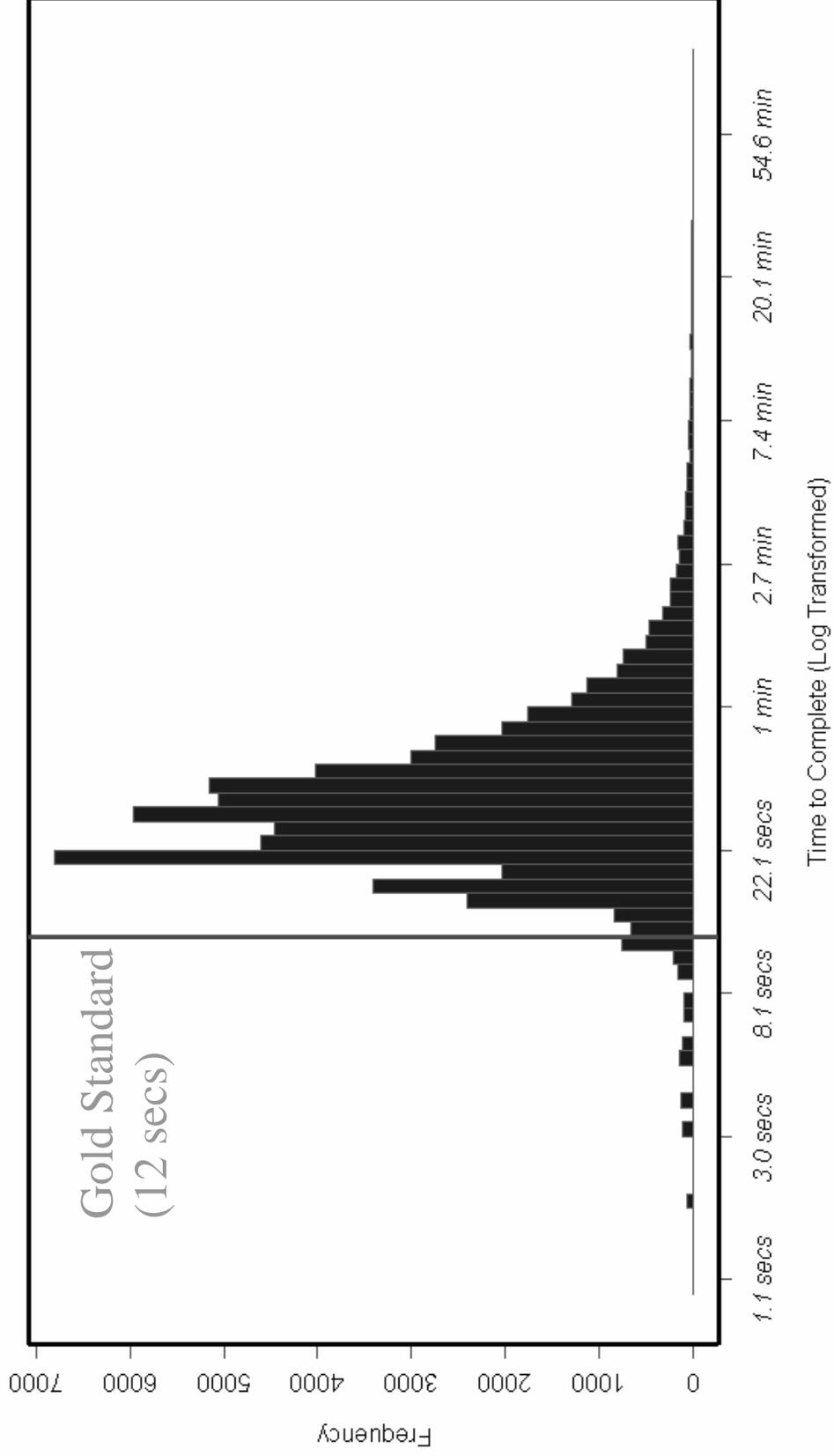
## Distribution of Audit Trail Timing Data: Calendar Set-up



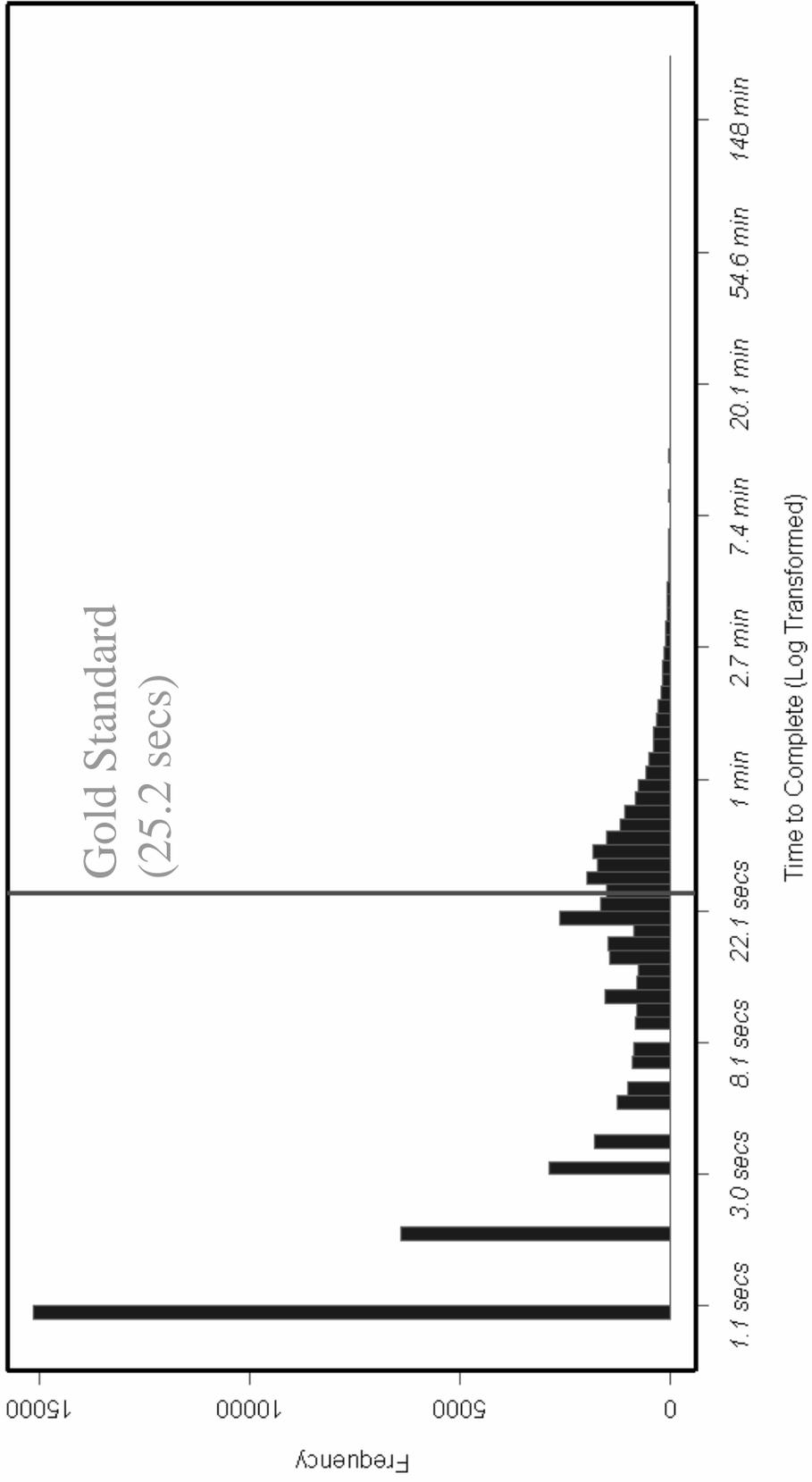
## Distribution of Audit Trail Timing Data: ACASI Set-up



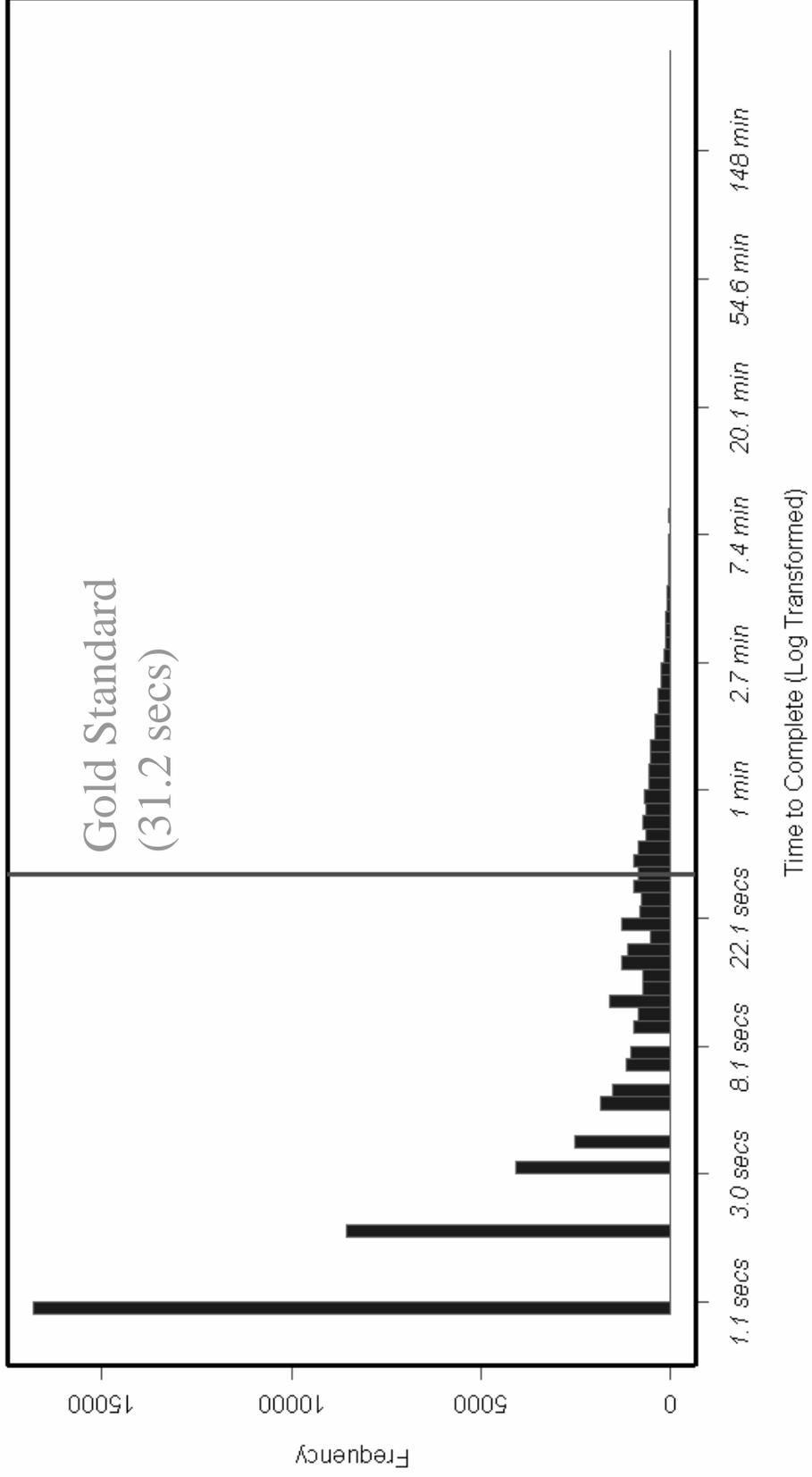
# Distribution of Audit Trail Timing Data: End of ACASI



# Distribution of Audit Trail Timing Data: Verification Form Completion



# Distribution of Audit Trail Timing Data: Ending Interview With Resp



**Table A3. Relative Percentage Change in Estimated Prevalence for Lifetime Drug Use Measures: Raw Questionnaire Data Vs. Using Additional Audit Trail Information**

Drug of Interest	Relative Percentage Change			
	12+ Years-old	12-17 Years-old	18-49 Years-old	50 + Years-old
Cigarettes	1.06	3.15	0.54	0.63
Alcohol	0.94	3.55	0.22	0.43
Marijuana	1.69	3.73	1.03	5.18
Cocaine	4.06	16.24	2.87	8.30
Heroin	10.48	27.59	7.69	23.68
Any Hallucinogen	2.46	9.77	1.24	6.67
Other Hallucinogen	30.41	51.25	23.15	23.52
Any Inhalant	5.62	11.67	2.78	15.04
Other Inhalant	32.30	38.90	23.42	*
Any Pain Reliever	11.72	17.51	9.11	30.00
Other Pain Reliever	34.29	39.07	31.15	52.77
Any Tranquilizer	5.69	13.84	3.72	18.13
Other Tranquilizer	28.20	38.32	24.35	*
Any Stimulant	6.13	14.26	4.26	6.03
Other Stimulant	27.09	40.13	22.61	*
Any Sedative	11.78	44.50	6.84	12.60
Other Sedative	27.07	38.37	20.93	*

\* Low Precision, no estimate reported (initial sample < 40).

Tables A3 and A4 present the results of our analysis on the impact of using audit trail data in estimating both lifetime drug use prevalence and recency of use. With regard to the lifetime prevalence measures, we see a good deal of variation across various drugs and age groups. In some instances, particularly for more prevalent substances such as cigarettes and alcohol, these slight increases in prevalence would have little or no effect. However, for some of the rarer drugs, such as heroin or cocaine, even the smallest increases could have large effects. For instance, the 0.14 percentage increase in heroin accounts for a 10.48% relative increase from the estimate based on raw questionnaire data alone.

Additionally, some drug categories (Hallucinogens, Inhalants, and all psychotherapeutic drugs, or “pills”) contain multiple gate questions with a last, “Other, specify” question. These questions prompt the respondent to indicate any additional drugs used within a particular category besides those specific drugs indicated in the preceding multiple gate questions. These questions, alone, account for more than a 25% relative increase in their respective prevalence rates. Though the “Other, specify” category represents a large relative increase when considered

by itself, it's impact on its respective overall drug category must be calculated with regard to all the other respective gate questions before any specific conclusions can be drawn. Though not shown here, the impact of the "Other, Specify" category only on the overall drug category, on average, accounts for about 26 percent of the lifetime prevalence increase (high of 41.8% for Hallucinogens and a low of 14.7% for Inhalants). These results direct the focus of the "Other, specify" analysis towards questionnaire methodology, in an attempt to discover why a large number of respondents are backing up and changing their answers to these specific questions.

**Table A4. Relative Percentage Change in Recency of Drug Use Measures: Raw Questionnaire Data Vs. using Additional Audit Trail Information**

Drug of Interest	Usage Classification	Relative Percentage Change			
		12+ Years-old	12-17 Years-old	18-49 Years-old	50 + Years-old
<b>Cigarettes</b>	Past 30 Day User	0.95	1.74	0.65	2.99
	Past Year User	0.07	0.12	0.09	0.92
	Used 1-3 Years Ago	0.29	1.10	0.21	1.02
	Confirmed Lifetime	0.76	0.70	0.67	1.06
	At least Lifetime User	712.11	657.64	*	*
<b>Alcohol</b>	Past 30 Day User	0.30	1.12	0.15	0.36
	Past Year User	0.78	1.22	0.41	1.80
	Confirmed Lifetime	1.19	1.45	1.00	1.38
	At least Lifetime User	509.70	537.10	*	*
<b>Marijuana</b>	Past 30 Day User	0.74	0.59	0.68	11.32
	Past Year User	0.65	0.66	0.65	0.00
	Confirmed Lifetime	0.37	1.43	0.26	0.40
	At least Lifetime User	1150.35	*	895.29	*
<b>Cocaine</b>	Past 30 Day User	3.02	1.72	3.28	*
	Past Year User	0.80	1.37	0.58	*
	Confirmed Lifetime	0.11	0.00	0.14	0.40
	At least Lifetime User	1236.39	*	*	*
<b>Heroin</b>	Past 30 Day User	7.45	*	5.30	*
	Past Year User	1.73	3.03	1.16	*
	Confirmed Lifetime	0.15	0.00	0.00	2.85
	At least Lifetime User	*	*	*	*
<b>Hallucinogens</b>	Past 30 Day User	2.73	1.63	3.29	*
	Past Year User	1.58	2.63	1.15	*
	Confirmed Lifetime	0.25	1.36	.18	0.00
	At least Lifetime User	766.30	*	*	*
<b>Inhalants</b>	Past 30 Day User	3.88	3.80	4.02	*
	Past Year User	2.00	1.77	2.37	*
	Confirmed Lifetime	0.26	0.29	0.26	0.00
	At least Lifetime User	419.14	355.43	566.48	*

\* Low Precision, no estimate reported (initial sample size < 20).

For the recency results displayed in Table A4, we see relatively the same results as for lifetime prevalence; however, the magnitude of the change is now dispersed across the various recency periods. In the recency of use categories, “Confirmed Lifetime User” means that all of the respondent’s questionnaire responses indicated only use prior to the past year, or in the situation of cigarettes, prior to the past three years. “At least a Lifetime User” stems from situations where a respondent has indicated some lifetime use within their audit trail data, but their questionnaire data indicates no use, so they were not subsequently ever routed through any of the respective recency of use questions. In short, there is neither questionnaire data nor audit trail data available on recency of use for these respondents. Though the prevalence and recency analyses have shown some interesting results, they have not shown any conclusive evidence that there are any problems with the reliability of the NSDUH estimates.

All data in Tables A1-4 were compared to the results from the six month 2002 NSDUH analyses. There were no significant differences at all, nor was there any shift in any of the final analyses. This suggests that, for future exploration with any large audit trails data set, half of the data set would be sufficient from which to draw conclusions.

Some discussion should be noted on our findings concerning data management of the Audit Trail files. Though the size of the NSDUH study is fairly large and at first glance appears to be burdensome, it was relatively easy to manage the data. With today’s advances in computer processing abilities and relative low cost of storage space, we were able to store and process all the files with minimal down time of any analyses (usually on average about 8 hours to reprocess all the 68,000 files depending on the extent of the modifications requested and about 1.5 GB of storage space). This should provide some assurances to anyone who is considering performing a study of this size and wishes to utilize audit trail data.

Before using audit trail data on any large study, however, a significant change needs to be made to the processing methods used in this analysis. This entails processing the incoming transmitted files on either a daily or at most weekly basis. Though some automated machine editing procedures can be implemented to capture and resolve a bulk of the audit trail idiosyncrasies, it will never be able to resolve all of them. This may be relieved by implementing some automated process that will at least flag certain files that need further investigation or flag situations that indicate a problem. Some examples would be: 1) identifying files with duplicate IDs and resolving them; 2) linking breakoff interviews with subsequent follow-ups; 3) detecting instances where the FI is having technical difficulties and taking appropriate action to correct them.

Ultimately, this hands-on processing would enable project staff to immediately deflect any problems that they see occurring in the field and would produce a fairly clean data file that could be processed quickly. We suggest having an analyst clean and process the data as it is coming in. It is the strong opinion of the authors that when the bulk of audit trail data reaches the magnitude of the NHSDA/NSDUH and covers such a lengthy time-span (i.e., 1 calendar year, divided into 4 quarters) that the data files be streamline processed (e.g., as close to real-time processing as is feasible). There are several reasons for this. First, in post-processing after the end of a quarter of data collection, the data is manageable, but when dealing with the sheer number of records, detection and resolution of inconsistencies in file management become difficult. Hence, the probability of not being able to recover a complete audit trail increases the longer the duration between transmission and processing. More importantly, the closer to real-time that processing can occur, the better opportunity to utilize the data in a manner effective to resolving problems or to developing an enhancement/modification to continue to improve data quality.

As an example, if a new FI is experiencing technical difficulty at the beginning of the data collection period, it would be better to rectify this situation early on, instead of discovering this situation after numerous interviews have been conducted.

#### **4. Future Analyses**

The authors' experiences working with the 2002 NSDUH audit trail data has been an enlightening one. Although time and cost constraints limited the amount of analysis we could do with the 2002 data, we discovered many potential analyses that we could perform. One further analysis would be to utilize the timing information on the initial routing through a prevalence or recency question of interest and determine the number of subsequent questions answered prior to the respondent backing-up and changing their answers. For instance, if a respondent takes two seconds to respond to a drug lifetime use question with 'Yes' and then after going through one or two more questions, backs-up and changes that answer to 'No', we might be more inclined to concede that the respondent had rushed through the initial question and did not realize their initial response was incorrect. Taking an opposite approach, if a respondent takes a relatively long time to answer the initial question and then proceeds through five or more questions before returning to change their answer, we might be more inclined to believe that the respondent is hiding the truth of their drug use. Furthermore, a comparison could be made between the outlying timing data of respondents who change their answers within the interview, with those who do not. This approach is still speculative, but it does provide another means to utilize additional data from the audit trail files, and based on the criteria used to determine what is a long time or a sufficient number of questions to go through prior to backing-up, may provide a more realistic assessment of the effect of changing answers.

Also, with advances in data mining techniques and audit trail file data management procedures, this would be an excellent opportunity to model fraudulent cases from the audit trail data of cases already proven to be fraudulent. Though the NSDUH currently conducts a random 15% verification of all interviews, it might be useful to calculate a predicted probability of being a fraudulent interview and specifically designate these cases to be verified within that FI's 15%. This method would require 1) identifying fraudulent cases and using their audit trail data to establish response patterns and 2) continually updating the predictive model to include new data. We plan to continue to utilize audit trail data to monitor our survey methodology and search for additional avenues of research for which audit trails would be an indispensable aspect.

#### **5. References**

- Penne, M., Snodgrass, J. and Barker, P. (2002)  
Analyzing Audit Trails in the National Survey on Drug Use and Health: Means for Maintaining and Improving Data Quality. Presented at the International Conference on Questionnaire Development, Evaluation, and Testing Methods. Charleston, South Carolina. Nov. 14-17, 2002.
- Caspar, R. and M. Couper (1997)  
Using Keystroke Files to Assess Respondent Difficulties with an Audio-Casi Instrument. Survey Research Methods – Proceedings of the American Statistical Association, pp 239-244.

Caspar, R. (2000)  
Using Keystroke Files to Identify Respondent Difficulties With an ACASI Questionnaire. Presented at Fifth International Conference on Social Science Methodology. Cologne, Germany. Oct. 3-6, 2000.

Lessler, J.T., R. Caspar, M. Penne and P. Barker (2000)  
Developing Computer Assisted Interviewing (CAI) for the National Household Survey on Drug Abuse. Journal of Drug Issues 30(1), 9-34, 2000.

Research Triangle Institute (1997)  
1997 National Household Survey on Drug Abuse: Results from Analyses of the Keystroke File for Field Test No.1. Prepared for SAMHSA. Contract: 283-96-0001. October 9, 1997

Substance Abuse and Mental Health Services Administration (SAMHSA). (2000)  
Development of Computer-Assisted Interviewing Procedures for the National Household Survey on Drug Abuse (Office of Applied Studies, Methodology Series M-3, DHHS Publication No. SMA 01- 3514). Rockville, MD.

Results from the 2001 National Household Survey on Drug Abuse: Volume I. Summary of National Findings (Office of Applied Studies, NHSDA Series H-17, DHHS Publication No. SMA 02-3758). Rockville, MD. (2002)



# A Random Walk Application for Blaise Instruments

*Gilbert Rodriguez & Jay R. Levinsohn, RTI International*

## 1. Introduction

Social science data collection has seen a significant movement toward the use of automated data collection procedures, using increasingly sophisticated tools. The Blaise programming language and data software are the kind of tools used in this movement. At RTI International we have been using Blaise to develop large, complex survey questionnaires for several years. While there have been significant advances in the power of the tools for development and deployment there has not been the equivalent development of tools to provide for quality assurance. For the last two years we have been seeking improved methods for Blaise questionnaire validation and quality assurance. We have developed a system for automating some aspects of the testing process, a software package we call RoboCAI (Levinsohn and Rodriguez, 2001). The RoboCAI process allows one to automatically execute predefined test scripts, compare the data from the executed scripts, and log and report results. A significant issue in this process is the level of effort required to develop meaningful scripts in sufficient quantity to provide adequate test coverage. As an extension to the capabilities afforded by RoboCAI we have developed a second process that will automatically generate scripts with relative little user effort. This automated procedure uses the concept of the random walk.

A random walk can be defined as a process consisting of a sequence of steps each of whose characteristics is determined by chance (Merriam-Webster online dictionary). The random walk as applied to a Blaise questionnaire implies that at each choice point in the questionnaire (each point where input is required) a random decision can be made to produce an input. As one steps through the questionnaire instrument based on the Blaise program and all previous responses each new choice point receives a new random input. The input is selected from the valid choices for a given questionnaire item. The output can then be examined to verify that the results agree with the questionnaire specifications and requirements. In addition, RoboWalk is designed to allow the developer to combine a mix of random inputs with pre-specified inputs. This process allows fairly rapid generation of test scripts and test scenarios. It allows the developer to build test scenarios that he feels will cover the breadth of cases that define the most likely paths as well as others.

At RTI International we have used RoboWalk to create test scenarios, which are replayed later with RoboCAI to test routing and to generate test data. The scenario logs and data are checked for correctness and consistency using utility programs. Also, by fixing values for particular gate questions in a RoboWalk script, one may create scenarios that emphasize the testing of particular blocks.

The random walk application, named RoboWalk, consists of two components that operate together. First, a WinBatch script starts a Blaise CAI instrument and feeds keystrokes to the CAI instrument and second, a Visual Basic program determines a random response for a given field. The Visual Basic program utilizes the Blaise API library and the data model files for a given questionnaire in order to determine the possible range of valid values for a particular field. It then performs a random draw based on the range of values.

An input file is used to specify a case id number and a few other items, such as questions and responses for which a random choice should not be made or in cases where a developer wants to specify part of the test script. Random choices for some questions can be problematic, since some questions may require specific types of responses in order to advance through the instrument.

An accompanying utility has been created that writes out the CAI instrument screens (in sequential order) to an ASCII text file using the Blaise data file as input. This file can then be examined for logic errors, flow errors, and range errors, as well as correctness of the screen text. Also, a random walk script file can be created from the data file so that a particular scenario can be kept to be used as a “test” script in the RoboCAI application.

## 2. The methodology

The RoboWalk process uses the following tools:

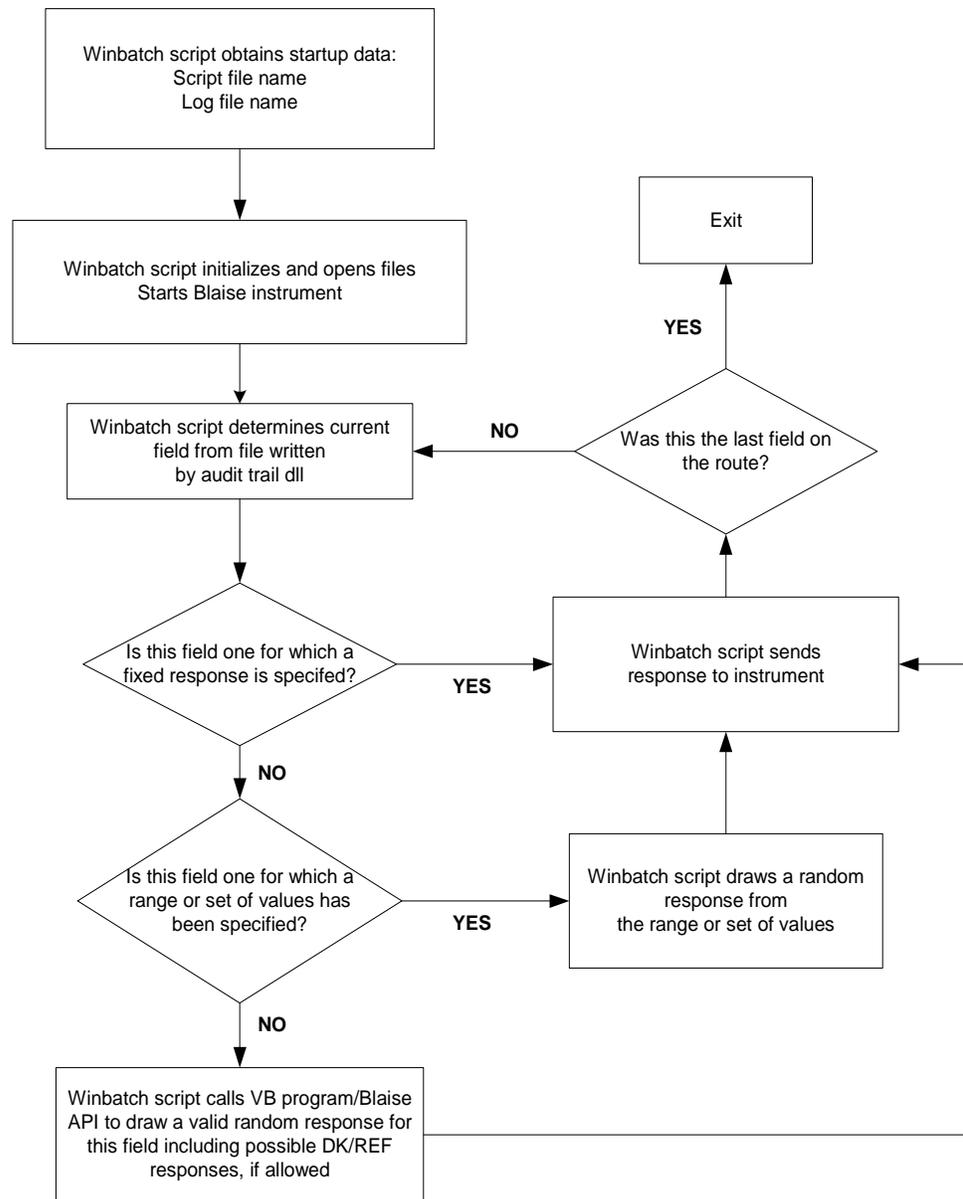
- WinBatch - a batch language interpreter for Windows. Batch files are written using WIL, the Windows Interface Language. A batch file can contain commands to run programs, read and write directly to files, send keystrokes to Windows applications, and perform a number of other tasks. Using the WinBatch compiler, an executable file can be created from a batch file. WinBatch is a proprietary product developed by Wilson WindowWare, Inc.
- Blaise 4.5 – CAI instruments created with different versions of Blaise 4.5 have been used with RoboWalk. Currently, Blaise 4.5.2.666 is being used.
- A modified audit trail DLL, audit.dll – This DLL, written in Delphi, writes out the name of the current field in an instrument to a text file where it can be read by a WinBatch batch file or executable.
- A Visual Basic application that uses the Blaise API objects to determine a valid, random response for a given field and is called as a shelled process from the WinBatch script
- A Visual Basic utility that creates print files of the path through the questionnaire and will also build a script for later use by RoboCAI.

When executed, RoboWalk prompts the user for the name of a text file containing data to be used for the case (which is called a script file) and then prompts the user for the name of a log file to which diagnostic data is written. All problems and cases where more than one random draw is required to proceed to the next screen, such as if a hard or soft error occurs, are logged in the log file. RoboWalk operates as follows:

- starts the CAI instrument,
- reads the script file,
- determines a random response,
- sends the response as keystrokes to a screen of the CAI instrument,
- proceeds to the next screen and then writes the name of the CAI screen to a text file (currentfield.txt) so that RoboWalk can determine the current field for the instrument.
- It then repeats this process for each screen of the CAI instrument.

Figure 1 presents a flow diagram of these steps.

Figure 1 - Random Walk Process



The flow and logic of this process is simple. There is one complication of coding to allow for synchronization of the activity between the Blaise audit.dll and the WinBatch code. These two processes, the random walk process and Blaise, run in parallel under the Windows operating system and RoboWalk must wait for the Blaise code to complete. The random walk process must allow the Blaise audit.dll enough time to open, write into, and close the currentfield.txt file before it can check to see what the next field is in the instrument. This is handled by the

WinBatch script. If the file currentfield.txt file does not exist, then the WinBatch script waits 1 second, and then checks again. This continues until the file exists.

An example will help to make the program operation more clear. The following sections present a short example. In Tables 1 and 2 we present the information that serves as input into RoboWalk (the script file) and the output. In this example scenario the respondent's date of birth is June 4, 1961.

## 2.1 Sample script

The following is an example of the contents of a typical script. In a RoboWalk script you would specify answers or a specific range of allowable answers (to be selected at random) for only a subset of the questions, the balance of the answers would be generated by RoboWalk. Table 1 presents such a script where only a part of the possible questionnaire items are given answers in the script.

QUESTID	2000042
DKPROB	0.01
REFPROB	0.02
DOB	6-4-1961
STATE	{ 26, 41, 46, 50 }
EDUCATION	[ 12, 15 ]
ENDAUDIO	out
VERIFID	T10-0008
CASEID	TX01010111A
FIEXIT	1

Note that you can specify probabilities for Don't Know or Refusal responses which are used to randomly select a Don't Know or Refusal response for fields for which they are allowed. You may also specify a set of values or list from which a response is randomly selected. For example in the case of the STATE question

```
STATE      {26, 41, 46, 50}
```

indicates that the response for STATE should be drawn as one of the items from the list 26, 41, 46 or 50 with equal selection probability. Don't Know or Refusal would not be selected for this question since a specific list is provided, only 26, 41, 46, or 50 would be selected. You may also specify a specific range of values from which a response is randomly selected. For example,

```
EDUCATION [12, 15]
```

indicates that the response for EDUCATION should be drawn only from those integer values between 12 and 15 (inclusive) with equal probability. The random draw, from a specified range or list, is performed within the WinBatch script rather than in the Visual Basic program.

Also, the WinBatch script attempts to work itself out of any hard or soft errors that may occur. If a hard or soft error dialog box pops up, then the WinBatch script will send a carriage return to clear it and then make another random draw for a response. This problem will be logged for the questionnaire author to review. The

WinBatch script will stop execution if a specific number (a user settable parameter, the default value is 20) of consecutive hard errors occur.

## **2.2 Sample log**

The text displayed in Table 2 is the output log file generated by the random walk program file. The log file contains the responses that have been sent to the CAI instrument either from the input script or from the randomly generated selections made by RoboWalk.

Table 2 RoboWalk Log

Random Walk Log  
Thu 1-30-2003 12:02:43 PM

	Screen	Value
0	QUESTID	2000042
1	DOB	6-4-1961
2	CONFDOB	2
***** HARD ERROR ENCOUNTERED *****		
3	DOB	6-4-1961
4	CONFDOB	1
5	CONFIRM	1
6	STATE	26
7	CONFSTATE	2
***** HARD ERROR ENCOUNTERED *****		
8	STATE	50
9	CONFSTATE	1
10	GENDER	5
11	CONFGENDER	4
12	HISPANIC	1
13	HISPGROUP	!
14	RACE	1 2 3 6
15	RACEASIA	1 5 6
16	MARSTAT	4
17	EDUCATION	13
18	HEALTH	2
19	INTROACAS1	1
20	HEADPHONE	
21	INTRO1	
22	INTRO2	
23	HEAROFF	
24	GOTDOG	1
25	EYECOLOR	2
26	ALLAPPLY	1 2 4
27	NUMBER	27
28	STOPLIST	1
29	DOAGAIN	2
30	BACKUP	
31	RANGEERR	2
32	INCONSIS	
33	ANYQUES	
34	ENDAUDIO	out
35	THANKR	
36	VERIFID	T10-0008
37	CASEID	TX01010111A
38	FIEXIT	1

=====  
End of Log  
Thu 1-30-2003 12:03:40 PM

The developer would need to review the log, shown in Table 2, to insure that the flow of the questions (the exact sequence) given the sequence of inputs is one that is consistent to the questionnaire specifications. If this review indicates any inconsistency it likely indicates an error in the Blaise program or a need to revise the questionnaire specifications.

### **3. Summary and conclusions**

There is a growing understanding that as CAI questionnaires get larger, more complex and use increasingly complex techniques that issues of quality control become a serious issue. There is significant cost in the current testing process and there is a significant failure rate in removing all errors prior to release of the production code. The industry is looking hard for better methods. We believe that the Robo tools that we have developed at RTI International help with this problem. In the past we have presented the RoboCAI tool that will automate the testing process from user crafted “test” scripts. The RoboCAI tool is labor intensive for large applications but extremely useful for smaller ones. In an effort to help with the effort of generating large test scripts we moved on to the random walk application – RoboWalk. A random walk application can be useful as a means of generating test scenarios, especially for very large questionnaires. There is significant effort required in going back and reviewing the resulting data afterwards and this level of effort increases as the size and complexity of an instrument increases. But, we feel that these tools taken in concert make good progress toward providing the questionnaire designer with tools to increase quality. Used together the Robo tools can provide excellent testing, documentation, and regression testing tools for small (<100 items) to medium questionnaires (<250). As the questionnaires increase beyond this size more and more effort is required to create test scripts or to interpret the results of a random walk. Very large questionnaires require very large amounts of effort to understand, debug, and document. These tools certainly can help that process but still leave hard work for the developers and designers.

### **4. References**

Levinsohn, Jay R., and Rodriguez, Gilbert (2001). Automated Testing of Blaise Questionnaires. *Proceedings of the 7th International Blaise Users Conference 2001*, Washington, D.C, USA.: Westat.



# First steps along the Audit Trail

*Tim Burrell, Office for National Statistics*

## 1. Introduction

In the last few years the Office for National Statistics (ONS) has started to examine the potential of the Blaise Audit Trail facility. An Audit Trail produced within Blaise records interactions made by the interviewer during computer assisted interviewing (CAI) or the respondent during computer assisted self-interviewing (CASI). The Audit Trail can provide a detailed history of the sequence and timing of navigation, field entries, changes to field entries and other events.

The paper will outline the practical application of Audit Trails on two projects:

- 1) to monitor respondent behaviour during CASI on the General Household Survey (GHS); and
- 2) to observe interviewer actions on a survey of income and living conditions (EU-SILC).

The paper will describe how we have been able to analyse the data produced, with reference to surveys mentioned above, and what progress we would like to make in the future. It will also consider some of the implications and concerns we have on using this tool.

With CAI, and Blaise in particular, the industry standard for the collection of official survey statistics, there is an increased availability of data for the researcher to investigate interviewer and respondent behaviour during an interview. With the help of a package provided by Statistics Canada named ATLAS, we have been able to enhance our use of the Blaise Audit Trail facility and widen the extent to which we have implemented it on surveys. This new software has also led to much easier analysis of the extensive data produced.

## 2. Previous use of the Audit Trail within ONS

ONS has previously used the Blaise Audit Trail facility on a small-scale project to evaluate the utility of this monitoring method. We chose an ONS survey of children and adolescents in the care of local government authorities, known as the survey of Looked After Children (LAC) as the vehicle for this trial. The Blaise instrument for the survey was designed to collect information about the mental health of children and included a substantial CASI section relating to sensitive or illegal behaviour. The main interest of the trial was to monitor respondent behaviour in the Audio Computer Assisted Self-Interview (A-CASI) section of the interview. (Bumpstead 2001)

This study confirmed in ONS earlier findings (Bumpstead 2001, Hansen and Marvin 2001) that the Blaise Audit Trail can provide useful data about respondent use of CAI instruments which it would not otherwise be possible to capture. The processing and analysis of audit trail data is not generally straightforward. However, it was possible to formulate a strategy for dealing with audit trail data which was not unduly cumbersome or time consuming.

### 3. ATLAS - A tool provided by Statistics Canada

ATLAS has been used to aid in the analysis of the vast amounts of data produced within an Audit Trail. It has the ability to read and average out across cases the times of individual questions or sets of questions from the Blaise Audit Trail. This proved useful as we could separate out the time spent in particular parts of the interview, such as the interview section of the questionnaire and in the administrative section. Measurement of elements of the interview was the main point of the projects mentioned in this paper.

Before acquiring the ATLAS package from Statistics Canada, ONS had only been able to carry out limited analyses of Audit Trail data. We had looked at information case by case. Data was produced like that shown in Figure 1, which shows the details from one case. This type of data is hard to read and Audit Trail data is hard to analyse in its raw form. Figure 1 shows the interviewer's progress through 4 questions. In this particular case, the interviewer took 11 seconds to pass through this set of questions.

Figures 2 and 3 show how the information is presented using the ATLAS. Figure 2 shows information from the same case as Figure 1. Figure 3 shows the average time over a number of cases (153 cases from the UK's General Household Survey, GHS ). It shows that, on average, interviewers took 7 seconds to pass through these questions.

**Figure 1. Audit Trail raw data**

```
"19/06/2002 18:16:31","Enter Field:QSignIn.StartDat","Status:Normal","Value:"
"19/06/2002 18:16:37","Leave Field:QSignIn.StartDat","Cause:Next
Field","Status:Normal","Value:20020619"
"19/06/2002 18:16:37","Enter Field:QSignIn.DateChk","Status:Normal","Value:"
"19/06/2002 18:16:40","Leave Field:QSignIn.DateChk","Cause:Next Field","Status:Normal","Value:1"
"19/06/2002 18:16:40","Enter Field:QSignIn.IntEdit","Status:Normal","Value:"
"19/06/2002 18:16:41","Leave Field:QSignIn.IntEdit","Cause:Next Field","Status:Normal","Value:1"
"19/06/2002 18:16:41","Enter Field:QSignIn.WhoHere","Status:Normal","Value:"
"19/06/2002 18:16:42","Leave Field:QSignIn.WhoHere","Cause:Next Field","Status:Normal","Value:1"
```

**Figure 2. Analysing the Audit Trail using ATLAS for one case**

BLOCK	INITIAL		SUBSEQUENT		TOTAL	
	HITS	SECS	HITS	SECS	HITS	SECS
QSIGN	1	11	0	0	1	11

**Figure 3. Analysing the Audit Trail using ATLAS for a number of cases**

BLOCK	SUBSEQUENT		TOTAL	AVG	
	HITS	SECS		HITS	SECS
QSIGN	4	7	0	4	7

It can be seen from the screen shots above that ATLAS summarises data in an aggregated form which is much more convenient as well as being in a much more readable form . This has enhanced our ability to analyse data produced by the Audit Trail. The data can be shown at the case level (figure 2) or the average over a number of cases (figure 3).

The data produced from ATLAS can be easily transformed into SPSS format where data can be analysed further. Other information available in the tool was the ability to single out particularly long hits, or unusually long amounts of time spent on questions. This feature means it is possible to single out any questions where a very long time was taken for an answer to be filled out. This could be due to a break in an interview for a number of reasons. It is then possible to remove such an outlier from the analysis if it would bias the measurement of the overall time spent on that question.

Data can be analysed both at an individual level, looking at the data case by case, or by looking at averages over all cases. Additional features include the ability to look at the number of fields which were answered *don't know* and *refusal*, and changes or edits to questions.

#### **4. A pilot to monitor respondent behaviour during CASI on the General Household Survey (GHS)**

The GHS is a multi-purpose survey providing the government with annual information about the major social fields of Population, Housing, Employment, Education, Health and Income to supplement the more specialised best national sources for estimates on these topics, such as the Labour Force Survey. Because all these topics are covered in one survey, it is possible to examine the relationships between them.

The General Household Survey interview comprises two parts: the household and individual questionnaires. The household questionnaire contains questions on demographic characteristics of household members, tenure and accommodation, consumer goods, migration and ethnicity. Questions relating to income are asked in the individual questionnaire on the GHS and the results aggregated for the household. ATLAS has been helpful in measuring the amounts of time spent in these different sections of the questionnaire.

The GHS carries topics which are asked to all members of the household. Some of these topics, such as smoking, drinking and contraception are sensitive, particularly for adolescents if there is a risk of being heard by their parents and other family members. Due to the sensitive nature of these topics they have always been carried out as self-completion sections. In previous years the self-completion sections have been entered onto paper by the respondent and then keyed in by the interviewer at a later stage. This paper-based method has the advantage, compared with CASI, that it allows all members of the household to fill out their self-completion forms at the same time rather than in turn, passing the laptop to each person.

It is possible that some information was being withheld due to this method of data collection. The respondent may see it as less confidential than if they key the data into the laptop themselves. A pilot study was undertaken to find out how long respondents would take to complete these questions by a CASI section within the Blaise questionnaire.

A pilot study was conducted out using CASI for sections of the questionnaire which previously had been completed through the paper form. The Audit Trail was enabled on the questionnaire to allow these sections to be timed. These sections included smoking, drinking, family information and contraception. Interviewers had stated that using the paper form usually took between 1 and 4 minutes for a household to complete. Self-completion also provided a welcome break to the respondent, giving them an active task to break up the long sequences of interviewer questioning .

The GHS pilot achieved 153 interviews. The results are displayed in Table 1.

**Table 1. Average times for self-completion sections in the GHS**

	Paper (concurrent)	CASI (sequential)	Excess of CASI (sequential) over Paper (concurrent)
Family Information	2 minutes	5 minutes	+3 minutes (150%)
Contraception	2 minutes	6 minutes	+4 minutes (200%)
Drinking	4 minutes	7 minutes	+3 minutes (75%)
Smoking	1 minute	4 minutes	+3 minutes (400%)

Obviously the time spent by household in each of these sections depends very largely on the size of the household and how much they have to say for each section. For example, a household where an individual smokes 20 cigarettes a day will take longer to complete the form than a household of non-smokers.

Due to the nature of CASI, each respondent must enter information sequentially on to the laptop. When the sections are conducted via paper, all members of the household can enter information concurrently. This has a large bearing on the increase in time for respondents to complete these sections. It also means that the two methods of data collection are not directly comparable.

It can be seen that, on average, it took an extra 3 minutes to conduct each section by CASI rather than on paper. However the relative increases are large, as much as 400% for smoking. Over the length of the entire GHS pilot interview, CASI added 13 minutes to the length, taking the mean time from 80 minutes to 93 minutes. This would be a large increase in respondent burden if implemented on the production survey.

ATLAS was used to calculate the times in the Blaise Audit Trail from the instrument. This information was used when drawing conclusions from the pilot study. With the help of ATLAS these times were able to be measured very quickly and in a form which enabled research staff to run more in-depth analysis than would have been possible without this package. Times were calculated both for sets of questions and individual questions.

One issue that the analysis investigated was if there was any time saving in using computer-assisted coding (CAC) in Blaise for large coding frames rather than coding from a reference manual. We were particularly interested in occupation coding, which our interviewers code at home after the interview. We did not have a direct comparison of the two methods for the same coding frame, so we explored the issue by an indirect argument. The Audit Trail told us that it took almost exactly the same mean time for interviewers to code occupation by CAC (their normal method) as to code industry from a reference manual (also their normal method): 52 seconds and 53 seconds respectively. However, the cognitive task involved in coding occupation from a reference manual is known to be much greater than the cognitive task in coding industry: occupation is coded to 5 digits as against the 3 for industry, and the coding index for occupation is over 50 times longer and less structured. Therefore we were able to conclude that it would take longer to code occupation than to code industry by the same method of referring to a manual. Since coding occupation using CAC took the same time as coding industry from a manual, it follows that coding occupation using CAC is quicker than coding occupation from a reference manual.

The Audit Trail analysis also allowed us to make design decisions about CASI for the GHS. As the pilot survey showed that CASI required additional time, it was decided that not all self-completion sections of the GHS could be asked as CASI to all sets of individuals. CASI is to be reserved for those sections which are particularly sensitive and, in particular, where confidentiality from the interviewer as well as other household members might be an issue. The Smoking and Drinking sections are mainly sensitive for adolescents in the presence of their parents so, while self-completion remains mandatory for people aged 16 & 17, and optional for all other respondents, it seemed reasonable to continue to use paper rather than CASI. The family information section is potentially more sensitive and for more people: it asks about former as well as present relationships and about all births/abortions and contraception. From 2003, it will be carried out in CASI. Paper self-completion documents are also available for this section for the rare occasions when a respondent is unhappy about using a laptop, or there are several eligible respondents in the household. In the latter scenario, some people can use the paper self-completion forms while others complete the section on the laptop. This will save time in the interview.

### **A pilot to measure the total time spent on new questions woven into appropriate places throughout an existing Blaise questionnaire**

The European Union is planning, through its statistical office, Eurostat, a mandatory survey on income and living conditions (EU-SILC) to be carried out by all member countries. The aim of the survey is to provide information on poverty and social exclusion in the UK that can be compared with the situation in other EU countries. In the UK, the cross-sectional element of the survey will be met by adding questions to the multi-purpose General Household Survey (GHS) which already covers many of the required topics. The information for EU-SILC will be picked up from many individual questions scattered throughout the GHS instrument.

This part of the paper describes the role of the Blaise Audit Trail in the pilot work for the version of the GHS which will deliver the requirements of EU-SILC in addition to the normal GHS requirements<sup>5</sup>. To test the cross-sectional component of the EU-SILC in a pilot study, a CAPI instrument was prepared that integrated the GHS with the EU-SILC primary target variables that it did not already cover. Detailed timing of questions was required for this study to provide Eurostat with information on how well the questions had worked as well and, in particular, how burdensome the sections were to respondents. We needed to isolate and measure the burden that the EU-SILC questions comprised in total. This was a difficult task since the questions were in many groups scattered throughout the combined GHS/SILC pilot instrument. We decided to use the Blaise Audit Trail to provide the level of detail that was needed.

The sample design of the study involved twenty interviewers, covering the range of levels of experience expected in the live survey, to work on the EU-SILC pilot study. Probability sampling methods were used to select 20 Primary Sampling Units (PSUs). Twenty addresses were randomly sampled within the PSU and interviewers were instructed to obtain interviews at 10 addresses in order to achieve 200 household interviews. 203 individual interviews were achieved.

The EU-SILC pilot study was conducted using Blaise, like all ONS social surveys. Table 2 provides information about the length of the combined GHS and EU-SILC

---

<sup>5</sup> The GHS is published each year as *Living in Britain*. *Living in Britain 2001* is a web publication at <http://nswebcopy/lib2001/index.html>

interview as well as the length of time taken by the EU-SILC questions alone. The combined GHS and EU-SILC timings include the interview and post-interview administration time (such as calls information and coding at home, e.g. occupation and industry). The time shown as spent on EU-SILC questions, summed by ATLAS from all over the questionnaire, represents only time actually spent in the interview.

**Table 2. Length of interview and of EU-SILC module**

Pilot study elements	Household Questionnaire	Individual Questionnaire	Complete interview
<b>GHS &amp; EU-SILC*</b>	17 minutes*	1 hour and 48 minutes*	2 hours and 5 minutes*
EU-SILC	4 minutes	42 minutes	46 minutes

\*Including administration.

Table 2 shows that the EU-SILC components (whether already covered in the GHS or covered by additional questions) accounted for an average of 46 minutes of the total interview length in the pilot study. However, it should be noted that because (as we knew from other data from the study) respondents had so much trouble understanding the EU-SILC household income questions<sup>6</sup> the interviewers did not feel able to do what they normally do on ONS income surveys and encourage respondents to find documentation. Hence the questions did not take as long as they would have done if implemented in a production survey. As a result, the final report to Eurostat stressed that these data underestimated the true burden of the EU-SILC components and recommended changes necessary to reducing it. It also stressed that the GHS component was over-estimated in these results, partly through consequential effects of poor EU-SILC questions and partly through the inclusion of administration time. These findings point to the importance of careful interpretation of the results. The Audit Trail and ATLAS can provide measurement but not analysis.

Detailed timing of the entire interview and of the selected areas of interest was provided by the Audit Trail. ATLAS enabled the research team to easily divide the EU-SILC module from the rest of the questionnaire.

The ATLAS tool proved invaluable when analysing data over a number of cases with the way it aggregates timings. In this manner we were able to look very quickly at the length of the interview and make rapid changes where necessary. Once again, the data in this readable form was able to be transferred into SPSS where further investigation could be conducted. As mentioned above, it was important to look at specific sections and how each section related to each other. This can give an indication of whether particular sections are influenced by other parts of the questionnaire.

## 5. Conclusions

The Audit Trail facility has become a more usable and adaptable tool which can now be used for analysis. When we first began to use the Audit Trail there was a fear that the amount of data produced would have a negative effect on the performance of the laptop. There has been no evidence of this. We are, therefore, continuing to use this function where detailed timing information will prove useful when analysing data or there is a pressing need from the client for detailed times in a questionnaire.

---

<sup>6</sup> Not designed by ONS!

ATLAS is a very useful analysis tool but further development of the software (and other systems if necessary) would be required to fit ONS requirements for routine/automatic use.

These developments are:

- ability to distinguish between different types of case (e.g. responding, non-responding cases);

- outputs should have min, max and range of values (field, block, subsections, datamodel) as well as averages;

- data should be able to feed into other information management systems (e.g. response rate, interviewer performance monitoring both for individually and divisionally);

- the user interface could be improved, so that you can look at different reports at the same time (e.g. different months of same survey).

Despite our wish to see these desirable additions, ATLAS has already greatly enhanced our ability to utilise the Audit Trail facility and we will continue to use it for future analysis. We are very grateful to Statistics Canada for permission to use it. Audit Trails have proved useful in both the timing and testing of questionnaires as well as trying new procedures and ways of carrying out data collection. It has been interesting to see what sections take the interviewer a particularly long time to get thorough and to assess where, if any, changes can be made to make interviews less burdensome to the interviewers and respondents.

As the Audit Trail has, as yet, had no visible adverse effects on data collection, there is the ability to collect information routinely on all SSD surveys. With the help of ATLAS this data can now be quickly transformed into a readable form for analysis. The fact that data can be quickly extracted has led to its increased use on designing surveys and questions. We have recently used it to aid the design of the *People, Families and Communities Survey* which is intended to find out about the role individuals and families play in their local community and to explore issues related to social capital.

The Audit Trail has been used on the pilot of this survey to give detailed timing information to the research team. From this information they were able to provide feedback to clients about if there were any particularly long sections or any subject matter where it took more time than expected to collect data.

There is also the potential to use Audit Trail data to give an indication of interviewer performance. For example, it is possible to compare the times interviewers take to complete individual sections of an interview. with due attention to our previous warnings about the need for careful interpretation, this method can be used to judge an interviewer's effectiveness. Audit Trail data can be used more straightforwardly for probity checking.

We would also like to take further advantage of some of the other features of ATLAS such as looking at the number of fields which were answered *don't know* and *refusal*, and at changes or edits to questions. It will be interesting to find out if there is a certain type of question which often has its value changed or edited.

We are continuing to increase our use of the Audit Trail facility. It has become a standard tool for the design of surveys.

## **6. References**

Bumpstead, R., (2001) "A practical application of Audit Trails", 7<sup>th</sup> International Blaise Users Conference

Hansen, S. and Marvin T., (2001) "Reporting on item times and key strokes from Blaise Audit Trails", 7<sup>th</sup> International Blaise Users Conference

*Post-processing and tabulation*

- **Creating Code to Convert Blaise to ASCII for SAS  
Input (Using the Blaise API within Visual Basic).....191**  
*Roger Schou, National Agricultural Statistics Service,  
USA*
  
- **Extracting data from a large instrument using the  
API.....199**  
*Lois Steinfeldt, ARS, USDA*



# Creating Code to Convert Blaise to ASCII for SAS Input (Using the Blaise API within Visual Basic)

*Roger Schou, National Agricultural Statistics Service, USA*

## 1. Background

At the National Agricultural Statistics Service (NASS), we often have a need to create an ASCII file from a Blaise data set, in order to read it into SAS. We very rarely want to read out every field from the Blaise data set, so a very tedious, hand-coding effort was required to create a Manipula setup as well as a SAS input statement. The introduction of the Blaise API provided tools to create SASOut.EXE, an application using Visual Basic to easily create these two files that were once so labor intensive. Throughout this paper, this application will be referred to as SASOut.

SASOut has much of the same basic functionality as the Manipula Wizard, which will be part of Blaise 4.6. However, there are many additional functions available with SASOut.

## 2. NASS Standards

SASOut assumes that certain NASS standards are being used. If the standards should change, minor changes to the Visual Basic code would be necessary. The following discussion addresses the standards that SASOut uses.

### 2.1 NASS Folder Structure

At NASS, we have a standard folder structure for all CASIC applications. The instrument is placed in the `h:\hqapps\casic\surveyfolder` folder, the survey specific Manipula setups are in the `h:\hqapps\casic\surveyfolder\Manipula` folder, the Blaise data set is in `h:\data\casic\surveyfolder`, and any generated ASCII output is placed in the `h:\data\casic\surveyfolder\asciout` folder, where *surveyfolder* reflects the survey. These four folders play a role in SASOut, because the optional SAS input statement will be placed in the instrument folder. The Manipula setup will be placed in the Manipula folder. This Manipula setup will look for the Blaise data set in the data set folder and place the output in the asciiout folder.

### 2.2 NASS Field Description

A coding standard has been introduced into the Blaise instrument to ease the processing steps needed with SASOut. The standard is simply the addition of a field description to each field to be included in the output. The syntax is SASVar: followed by the variable name (the colon is required). An example of a field in Blaise follows:

FIELDS

ID "Please enter the id." / "SASVar: RecordID" : 1..5000.

This standard is by no means required, it just simplifies the SASOut processing steps. SASOut will use the Variable name following the SASVar: notation, if it is present in the instrument. If it is not present, SASOut will prompt to use the Blaise field name for each field chosen. There is also a button available that will use the Blaise field name as the Variable name for all of the remaining selected fields.

## 2.3 Output Files

There are potentially two output files created by SASOut: SASOutVB.MAN and *Instrumentname*VB.SAS, where *Instrumentname* is the name of the Blaise instrument.

The Manipula file is always created, and the SAS file is optionally created. Two small examples of these output files are included at the end of this paper.

## 2.4 Field Conversions

It may be desirable to convert some fields when they are output. SASOut addresses some of these conversions. “Don’t Know” and “Refusal” responses may be converted to -1 for all of the non-string fields (integer, real, and enumerated fields). If the field to be converted to a -1 is a one-digit field, then it will automatically be output as a two-digit field, in order to handle the minus sign. This conversion is an all or none conversion, meaning that it will be true for all non-string fields.

Any string field may optionally be converted to an integer. This conversion is based on each individual string field. A prompt will appear for each string field, asking whether it should be converted to an integer.

Date type and time type fields are output as strings.

SASOut will not output open fields. It simply gives a warning that it cannot output open fields, and then continues.

## 3. The SASOut Interface

The buttons and windows of SASOut are dynamic, only appearing when they are valid. It was written to be straightforward, while obtaining all the necessary information to create the Manipula setup and the optional SAS input statement.

### 3.1 Selecting the Blaise Meta File

When invoking SASOut, a window is displayed that has only one active green button. It is used to identify the Blaise data model to be used. Clicking on the green button will allow the user to point to the Blaise meta file (BMI file). Once the BMI file has been selected, a prompt asking for the Blaise data set name appears. The name of the Blaise data set should be entered without an extension.

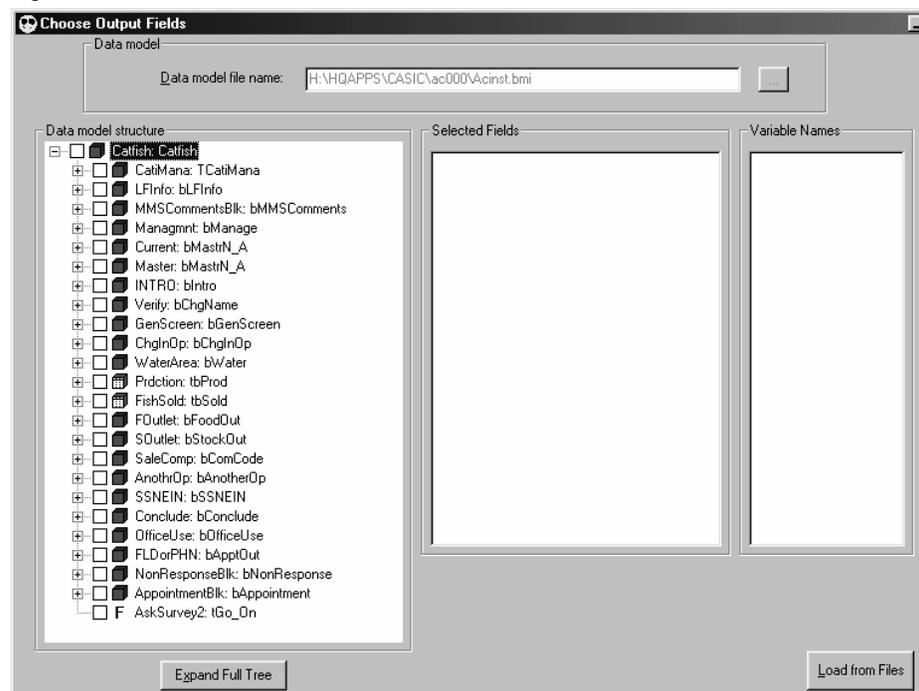
Figure 1. Blaise Data Model



### 3.2 Selecting Fields

Once the Blaise data set name has been identified, the structure browser of the data model is displayed. The structure is collapsed to the data model level and may be manually expanded block by block. The *Expand Full Tree* button is also available, which will expand the full structure tree. However, there is no Collapse Full Tree button, so once the full tree is expanded, it can only be collapsed manually. A *Load from Files* button will only be visible if the selected fields and variable names were saved from a previous session. This will be discussed later, but it is worth pointing out that the view (or field selections) may be saved and reloaded at a later time.

Figure 2. Select Fields from Structure Browser



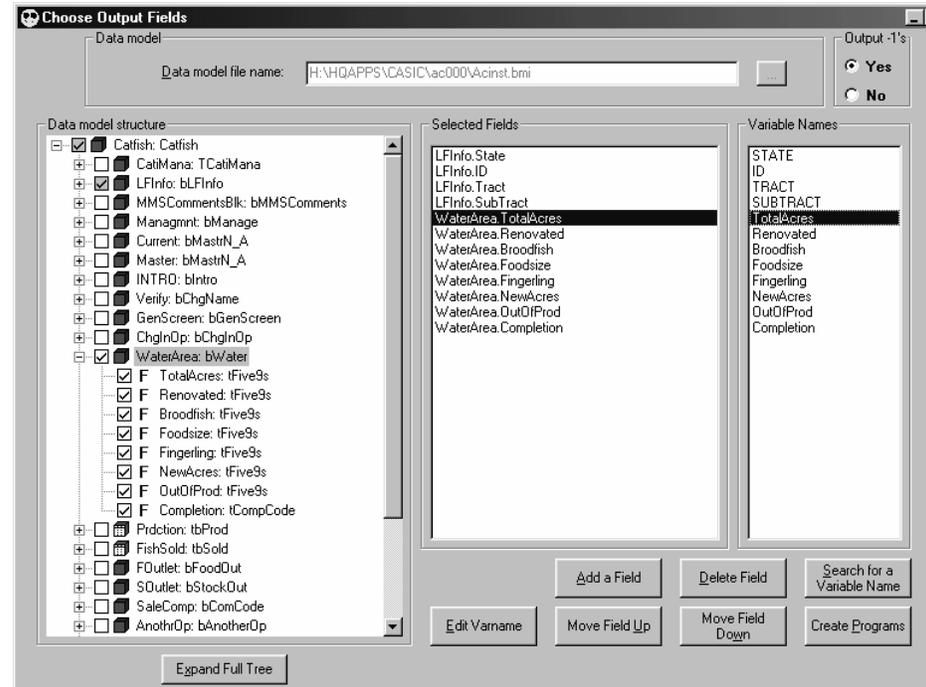
When the first field is selected from the structure browser, a prompt to output a –1 for all “Don’t Know” and “Refusal” responses for all non-string fields appears. Once that question has been answered, it will be noted at the top right of the window. This answer may be changed by clicking on either the *Yes* or *No* option. The *Create Varnames* button also becomes visible. The selection of fields may continue, until all desired fields have been selected. The order in which the fields are selected will determine the order the variable names will be listed; however, this order may be changed.

### 3.3 Creating Variable Names

The next step is to click on the *Create Varnames* button. All selected fields that have the SASVar: notation in the field description in the instrument will appear with the designated name in the *Selected Fields* and *Variable Names* list boxes. The full Blaise field name will be in the *Selected Fields* list box, and the variable name will be in the *Variable Names* list box. If the selected field has a string type, a prompt will be displayed asking if the value should be converted from a string to an integer. For every selected field that does NOT have the SASVar: notation, a prompt will be displayed asking for the variable name to be used for the selected field. This variable name will default to the Blaise field name, but may be changed. In fact, there is an *Accept ALL Blaise Field Names* button available which will use the Blaise field name for all of the remaining selected fields. Once the *Create Varnames* button is clicked, it will no longer be visible, and the structure browser will also be disabled.

There are two new buttons that become visible. The *Create Programs* button will create the Manipula setup and the optional SAS input statement. This button will be discussed later. The second new button is the *Add a Field* button. If a field was omitted from the original selection, clicking the *Add a Field* button will enable the structure browser so more fields may be selected. After clicking the *Add a Field* button, the *Create Programs* button will become invisible, and the *Create Varnames* button will become visible again.

Figure 3. Manipulating the Variable Names



### 3.4 Manipulating the Variable Names

Once the *Create Varnames* button has been clicked, it is possible to select a field in the *Selected Fields* or the *Variable Names* list boxes. Selecting a field in either of these list boxes will cause up to five new buttons to be visible. The *Edit Varname* button offers the ability to change the selected variable name. The *Delete Field* button will remove the selected variable name from the list, and thus it will not be output. The *Search for a Variable Name* button provides the ability to search the *Variable Names* list box. The search can be from the beginning of the list down or from the selected field down. The search is a “whole word search,” but it is not case sensitive. The *Move Field Up* button will be visible for all fields except the first one, and it will move the selected field up one. The *Move Field Down* button will be visible for all fields except the last one, and it will move the selected field down one.

### 3.5 Creating the Output Files

Once all of the desired fields to be output have been selected and the variable names have been assigned, the *Create Programs* button should be clicked. Once clicked, SASOut will check the variable names to insure that each one is unique and contains no spaces. If it finds a duplicate name or a variable name that contains a space, a message is displayed and the programs are not created. Once SASOut determines that there are no duplicate variable names and none contain a space, a few additional prompts are displayed.

#### 3.5.1 Saving the Selected Fields

The first prompt presents the option to save the selected fields and variable names. Clicking *Yes* will cause SASOut to save a series of little ASCII files so that the selected fields and corresponding variable names may be reloaded at a later time. Clicking *No* will not save the selected fields and variable names, and the next time someone selects the same instrument, they will have to reselect all of the desired fields and assign the variable names again.

### **3.5.2 SAS Input Statement Option**

The second prompt presents the option to create a SAS input statement in addition to the Manipula setup. Clicking *No* will result in only the Manipula setup being created.

### **3.5.3 Delimited File Option**

The third prompt presents the option to create an ASCII delimited file instead of a fixed field file. Clicking *Yes* indicates that an ASCII delimited file is desired, and a dialog box will be displayed, where the delimiter of choice may be selected.

### **3.5.4 Naming the SAS Data Set**

The fourth prompt will only appear if a SAS input statement was requested by answering *Yes* to the second prompt (the SAS Input Statement Option). This last prompt asks for the name of the SAS data set to be created. This will be the name used in the Data step in SAS.

## **4. Conclusion**

Examples of a Manipula setup and a SAS input statement, both generated by SASOut follow. These selected examples were shortened in order to conserve space, but it is evident that these programs could become very large.

Before SASOut existed, creating these large programs was very time consuming. Cameleon was used to generate the SAS input statement, but all of the fields in the instrument were included in the input statement. This file then had to be pared down by hand. Next, a corresponding Manipula setup had to be written by hand that would be used to create the ASCII file for SAS to read. This entire process took hours to complete, and the programs often contained typographical errors. With the new SASOut application, the same process can be completed in 15 minutes to ½ hour depending on the number of variables desired in the output. Without the Blaise API, SASOut would have never been possible.

### Example 1. SASOutVB.MAN – A generated Manipula setup

```
SETTINGS
  DATEFORMAT = MMDDYY
USES
  InstFmt '\HQAPPS\CASIC\ac000\Acinst'

  DATAMODEL SASOutFmt
  FIELDS
    STATE : INTEGER[2]
    ID : INTEGER[9]
    TRACT : INTEGER[2]
    SUBTRACT : INTEGER[2]
    TotalAcres : INTEGER[5]
  ENDMODEL

Inputfile BlzData : InstFmt ('\data\CASIC\ac000\ac000', BLAISE)

Outputfile SASOut : SASOutFmt ('\data\CASIC\ac000\asciiout\BlzData.ASC',
ASCII)
  SETTINGS
    SEPARATOR = ','

MANIPULATE

  SASOut.STATE := BlzData.LFInfo.State
  SASOut.ID := BlzData.LFInfo.ID
  SASOut.TRACT := BlzData.LFInfo.Tract
  SASOut.SUBTRACT := BlzData.LFInfo.SubTract
  IF (BlzData.WaterArea.TotalAcres = DK) OR
    (BlzData.WaterArea.TotalAcres = RF) THEN
    SASOut.TotalAcres := -1
  ELSE
    SASOut.TotalAcres := BlzData.WaterArea.TotalAcres
  ENDIF

  SASOut.WRITE
```

**Example 2. InstrumentNameVB.SAS – A generated SAS input statement**

```
TITLE Catfish;
DATA CATFISH;
INFILE 'data\CASIC\ac000\asciiout\BlzData.ASC' LRECL=62 DELIMITER=';';
INPUT
  STATE           /* LFInfo.State */
  ID              /* LFInfo.ID */
  TRACT          /* LFInfo.Tract */
  SUBTRACT       /* LFInfo.SubTract */
  TotalAcres     /* WaterArea.TotalAcres */
;
LABEL
  STATE = 'LFInfo.State'
  ID = 'LFInfo.ID'
  TRACT = 'LFInfo.Tract'
  SUBTRACT = 'LFInfo.SubTract'
  TotalAcres = 'WaterArea.TotalAcres'
;
```



# Extracting data from a large instrument using the API

*Lois Steinfeldt, ARS, USDA*

## 1. Introduction

The Automated Multiple Pass Method (AMPM) instrument, developed by the Food Surveys Research Group, Agricultural Research Service, U.S. Department of Agriculture, collects 24-hour dietary recall data. The data are used to address economic, nutrition and food safety issues. For example, the data are used to evaluate the nutritional adequacy of the American diet and the impact of food assistance programs. The data are also used to estimate exposure to pesticide residues and to study the impact of food fortification, enrichment, and food labeling policies.

The steps in the AMPM interview are shown in Table 1. The multiple pass method improves the collection of 24-hour dietary recalls. Individuals recall the foods and beverages that were consumed the day before the interview. Details about each food and beverage are collected as well as an estimate of the amount consumed. Information is also collected on the time of day the food was eaten, the name of the eating occasion, whether the food was eaten at home or away from home, and where the food was obtained. AMPM contains more than 2500 questions and more than 20,000 responses. Ninety-five percent of the questions are about specific food details, including the amount of the food eaten.

**Table 1. Automated Multiple Pass Method (AMPM)**

	Pass	The respondent:
Step 1	Quick List	... reports an uninterrupted listing of all foods and beverages consumed in a 24-hour period the day before the interview.
Step 2	Forgotten Foods List	... answers a series of 9 food category questions probing for any further food items which may have been forgotten.
Step 3	Time and Occasion	...answers the time they began eating or drinking the food reported and what they would call the eating occasion for this food.
Step 4	Detail Cycle	...answers standardized questions developed by USDA to probe for detailed information about each food reported and the amount of the food eaten. Additional information is elicited about where the food or most of the ingredients was obtained and where each eating occasion was eaten. ...reviews the eating occasions and times between occasions to see if additional foods are remembered.
Step 5	Final Review Probe	...answers a final probe for anything else consumed.

Foods are grouped into 132 categories each with a unique code. Each category has questions specific to the foods in the group. For example, a respondent reporting orange juice is asked if it was 100% juice and whether it was freshly squeezed, made from frozen concentrate, or came from a bottle, a carton, or a can. If the orange is from frozen concentrate, then the respondent is asked about the amount of water used to dilute the juice. When soda is reported, the respondent is asked if the soda contained caffeine and whether it was diet or regular. In addition to food details, AMPM provides the respondent with a number of ways to quantify the amount consumed. There are weight measures, volume measures such as cups and

liters, and item descriptions such as 1 slice. In addition there are two-dimensional models of dishes such as glasses, mugs, and bowls, and shapes for measuring rectangular, round, and wedge-shaped foods.

The complexity and the size of the AMPM instrument make efficient data extraction and organization for food coding and analysis both a challenge and a necessity. In order to accomplish this, a program was needed which could quickly and accurately identify the fields with data values and extract the value and other selected field properties. Because of the flexibility and ease of programming, the Blaise Application Programming Interface (API) and Visual Basic were used to develop this program.

## **2. Background**

There is a large amount of variation across respondents in both the numbers and the types of foods consumed. While the number of foods reported during an interview is usually less than 20, there have been a few respondents who have reported more than 30 foods. And while one respondent may have coffee many times during the day and no vegetables, someone else may have a lot of fruits and vegetables and no coffee. The AMPM instrument must balance the collection of complete and accurate intakes against the size and complexity of the data model. Setting array sizes to accommodate the greatest numbers of foods per day, and foods per category per day, produces a very large model. The limit for the number of foods per day is currently set at 40. So far this limit has not been exceeded. The number of foods per category per day is set at either 5 or 10 depending on how often foods in that category were usually reported. About one quarter of the 132 food categories accommodate up to 10 reports per day, the rest of the categories allow up to 5 reports per day. If the foods reported exceed any of the limits, the information is stored in a remark. Although the category limits have been exceeded a few times, for example with infant formulas, the limits in general have been adequate.

The AMPM data model contains information at the level of the interview, the food, and the food details. The majority of the data is stored in block and field arrays of foods and food details. For example, the food array, which allows for up to 40 foods, contains 53 fields including the name of the food, the time it was eaten, the name of the meal, and the food category. Because the food detail arrays contain the details and the amount consumed for the foods in that category, each is a different size. The milk category, which allows for up to 10 reports per day, has 33 fields including 5 of which record additions to the milk, such as chocolate syrup. The green salads category, which allows for up to 5 reports per day, has approximately 270 fields including the ingredients in the salad, the amounts of each ingredient and the type and amount of salad dressing.

Although there are over 140,000 defined data fields, as shown in the technical description of AMPM in Table 2, only 6,655 are elementary fields. Also shown in Table 2 there are 18,837 instances of 1,020 blocks. This shows that the size of the data model is due to the need to allow for multiple reports of foods and food ingredients, including their descriptions and amounts. For an individual interview, most of the fields in a record will be empty. From an average interview, there are fewer than 500 fields that contain data values needed for food coding and nutrient analysis. From an interview with more than 30 foods, there could be as many as 1000 fields needed, while from an interview for an infant there may be as few as 100. But because each respondent consumes different numbers and types of food, different data fields need to be extracted for each record. The difficulty lies in finding the data values that are needed without having to check every one of the

140,849 fields and in maintaining the link between the food detail data and other food information. Then once the data are extracted, they need to be organized for food coding and analysis.

**Table 2. Technical description of AMPM - Overall Counts**

	Value
Number of uniquely defined fields*1	7,675
Number of elementary fields*2	6,655
Number of defined data fields*3	140,849
Number of defined block fields*4	1,020
Number of defined blocks	1,020
Number of embedded blocks	234
Number of block instances	18,837
Number of key fields	7
Number of defined answer categories	1,923
Total length of string fields	239,2781
Total length of open fields	0
Total length of field texts	356,878
Total length of value texts	126,603
Number of stored signals and checks	127,397
Total number of signals and checks	127,397

\*1) All the fields defined in the FIELDS section

\*2) All the fields defined in the FIELDS section which are not of type BLOCK

\*3) Number of fields in the data files (an array counts for more than one)

\*4) Number of fields of type block

### 3. Data Extraction

Since less than 1% (~500/140,000) of the fields are going to be extracted, excluding large groups of fields as early as possible increases the efficiency of both the extraction and the subsequent processing. The main approach to extracting data is to use the information that is stored in the food array for each food reported to identify which food category array and which instance in the food category array contains the food detail information for that food. When a respondent reports a food, it is selected from a food list. Each food in the list is linked to a food category that determines which questions are asked for the food. Information about the food is stored in a food block array. Because the same category of food (e.g. fruits) can be eaten more than once a day, the answers to the food detail questions are stored in arrays for each food category block. Both the food category and the instance in the food category block array are stored in the food block array for each food reported. Using these fields, the extraction program can find and read only one food category detail block instance for each food. This greatly reduces the number of fields in the AMPM database that must be read.

Once the correct block instance is found, the extraction uses the basic recursive function 'For Each objField in ParentField Fields' expanded to reference a set of exclusion criteria. The exclusion criteria were added because even limiting the extraction to the specific food category block instance, the program would still extract many fields that aren't needed. For example, the meat sandwich category must allow the respondent to report multiple meats, cheeses, and vegetables in a sandwich. Here again, the variation in food consumptions across individuals requires that enough space be allocated to record the sandwich with three meats,

two cheeses, and five vegetables, as well as the sandwich with one meat, one cheese, and no vegetables. For most sandwiches this results in a lot of empty fields. Although the fields must be accessed to determine if they meet the exclusion criteria, subsequent steps in the processing benefit by not extracting the fields that are not needed. The exclusion criteria used are collections of Field Tags, Field Values, and Field Names, which are evaluated in that order. The order is based on the likelihood that the criteria will exclude a field or a group of fields. That likelihood is based upon the AMPM data model and the nature of the data collected. The number of items in each of these collections is kept as small as possible to reduce the amount of time the program spends comparing the field properties in the database to the items in the exclusion collections.

The exclusion criteria used most often are Field Tags and Field Values. The consistent naming and use of Field Tags in AMPM made possible this quick and efficient method for eliminating fields that contain fills, which have a Tag Name of INTFILL and fields that are used to control and monitor the flow of the instrument which have a Tag Name of INTFLOW. Then because such a large percentage of the fields are empty, exclusion based on empty Field Values greatly reduces the amount of data extracted. Field Names are the least utilized because of the amount of work to specify and maintain a Field Name collection and the time it would take the program to compare each Field Name to a large collection of Field Names. For AMPM, the Field Name collection contains a single entry that occurs in a large number of the food category block arrays and which cannot be eliminated using any other criteria. This field contains the answer to the question, "Did you add anything to this food?" While the subsequent food coding process does need to know what foods were added, it does not need to see the yes or no answer.

Before the exclusion criteria are evaluated, the presence of a remark is checked. There are a few fields in AMPM where the field can be empty, but the interviewer is able to record a remark. If there is a remark attached to a field, that overrides the exclusion criteria and the field is extracted.

In addition to the exclusion criteria listed above, there are additional criteria used at the data model level and at the first and second block levels. They are Include Root level fields (yes or no) and a collection of Block Names. The Root level fields criteria either includes or excludes the fields at the level of the object passed to 'For Each objField in ParentField Fields'. The block name collection allows the exclusion of blocks that are used to control the flow of the instrument and do not store any food information. It is also used to exclude blocks that only contain interview instruction fields.

During extraction, each field is written into a Microsoft Access database as a separate record with a unique identifier. The unique identifier is a sequential number representing the order the field was extracted from the Blaise databases. This number is important for keeping the food detail fields in order, which is needed for accurate food coding. The Blaise database name and primary keys further identify each record. Since AMPM contains food level data in addition to interview level data, the food number further identifies the food specific fields. Each record also contains the fully qualified name, local name, display text, value, type, tag, and remark for the field.

## **4. Conclusion**

The extraction program developed with the Blaise API and Visual Basic successfully identifies and extracts the intake and food data values that are required from each record. Parameters were created to allow data fields to be excluded from the extraction based on block name, tag name, field name, and/or field value. Once the blocks that contain food detail data are identified, the program quickly steps through the fields within each block testing for exclusion criteria. The presence of a remark, even on an empty data field, overrides the exclusion criteria so that all fields with remarks are extracted. The program has proved to be an efficient and accurate method to extract the small amounts of intake and food data from the large AMPM database.



*Integration of multi mode surveys, Survey Management, Case Management*

- **The “Multiple Application Interface” with Blaise and Visual Basic** .....205  
*Jim Hagerman & Hemant Kannan, The University of Michigan*
- **Electronic Data Reporter** .....215  
*G.W. de Bolster, Statistics Netherlands*
- **Integration of CAPI and CATI infrastructures at Statistics Canada**.....225  
*Jacqueline Mayda & Pamela Ford, Statistics Canada*



## The “Multiple Application Interface” with Blaise and Visual Basic

*Jim Hagerman & Hemant Kannan, The University of Michigan*

In the course of planning the transition of a long-term economic panel study from a DOS-based CAI application to Blaise v4.5, the project staff elected to integrate an additional interviewing method into the mix; the use of an Event History Calendar (EHC) programmed in Visual Basic. The Panel Study of Income Dynamics (PSID) is used as an example in this paper.

The challenge lay in how the sample management system (SurveyTrak) would be able to handle and manage what essentially turned out to be five separate applications: three Blaise applications and two VB applications.

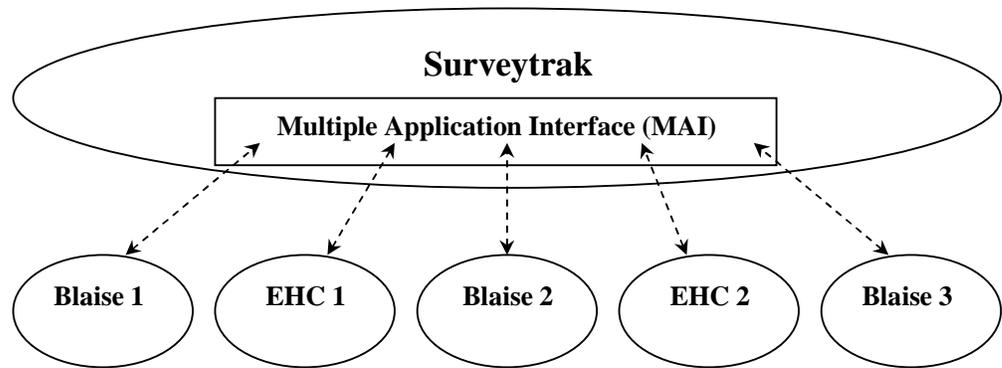
The applications would be implemented in an alternating sequence: Blaise 1 (which included the Coverscreen), EHC 1 (employment history questions about the head of the household utilizing the Event History Calendar), Blaise 2 (additional employment history questions about the head of the household), EHC 2 (employment history questions about the wife, if one was present, utilizing the Event History Calendar), and Blaise 3 (additional employment history questions about the wife as well as sections on consumption, income, wealth, pensions, health, marriage and children, background and education, philanthropic practices, and questions about public assistance and welfare).

Each application had its own preload needs, involving both common and unique variables. The initial preload file would only satisfy the preload needs of the first Blaise application. The other four applications would need to have their preload built “on the fly” during the interview itself. The application deployment would occur in a downward cascading fashion. Therefore, data from Blaise 1 would feed the preload needs of EHC 1; data from Blaise 1 and EHC 1 would feed the preload needs of Blaise 2; data from Blaise 1 would feed the preload needs of EHC 2; and data from Blaise 1, EHC 1, Blaise 2, and EHC 2 would feed the preload needs of Blaise 3 (which comprised the majority of the interview).

SurveyTrak needed a method to extract data, prepare the preload for the next application, and deploy the next application at the appropriate time. This led to the use of the “Multiple Application Interface” (MAI) developed in the Survey Research Center at the University of Michigan. The MAI had been previously used on a small study that involved two Blaise applications and one VB application.

---

The authors wish to acknowledge the grateful assistance, information, and feedback provided to them by members of the DCS Systems Programming Team, the DCS CAI Programming Team, and staff members from the Panel Study of Income Dynamics (PSID) project; all located in the Survey Research Center in the Institute for Social Research at The University of Michigan.



## 1. Surveytrak

Surveytrak is a windows-based sample management system developed, using Powerbuilder and Sybase, by the staff at University of Michigan's Survey Research Center located in the Institute for Social Research. SurveyTrak is designed to support interviewing efforts, as well as those of the survey management team. SurveyTrak performs the following critical functions:

- Delivers sampled households to the interviewer;
- Controls entry into the interview instruments;
- Moves other SurveyTrak data from the interviewer's laptop to ISR;
- Transfers data to ISR;
- Receives data from ISR and program files.

## 2. Event History Calendar (EHC)

The Event History Calendar is an interviewing method that does not follow the standard q-list style of interviewing. It is a method that uses more of a "free-flow" approach to interviewing. It relies on events in the respondent's life triggering the respondent's memory in relation to such subject matter as employment, housing, health, etc.

The EHC was programmed in Visual Basic and its interface is set-up primarily as a mouse-driven environment. This is a major difference from the Blaise applications, which were developed as a keyboard-driven environment only; as the SRC interviewers are trained only to use the keyboard and not the mouse. This difference has created some usability issues that are currently under study.

The EHC was first implemented with Blaise in a very small study, as was mentioned earlier. This involved only one instance of the EHC and two Blaise applications. This small study was successful in utilizing the MAI to control the different applications. This helped lead to the integration of the EHC into the much larger panel study, PSID.

### 3. Multiple Application Interface (MAI)

The MAI is a service component, developed in Powerbuilder, for the sample management system (SurveyTrak). It allows Surveytrak to access to multiple applications during an interview session.

The MAI was developed with an open design, which allows taking future requirements into consideration. The current known entities (applications) are Blaise and EHC. However, using an open design will allow the inclusion of any unknown entity in the future with very little change to the MAI. These entities can be any application with an executable and a related database or a proprietary product like Blaise. The strong handling of object-oriented programming by Powerbuilder makes it easy to implement the open design, which facilitates the use of the MAI for projects in the future with very little additional programming involved to accommodate new applications.

The primary job of the MAI is deploying the correct application at the correct time. It also plays an active role in the development and passing of preload variables for whichever application it is due to invoke. While it has no actual “user interface” per se, the MAI works as an interface between Surveytrak, Blaise, and the EHC. Because the MAI works in a downward cascading fashion, one of the constraints is once a module or application has been completed and moved past, it cannot be re-invoked or backed up to during that particular interview session.

The need for the MAI arises from projects, which use multiple applications (Blaise, EHC using VB) to comprise a single interview. The MAI was chosen as the preferred method over Alien Routers or Procedures for the following reasons:

Alien routers/procedures weren't able to pass as many variables back to the Blaise applications; the third Blaise application and the last application in the sequence to be deployed was also the largest requiring preload data that could go as high as 1,100-plus variables.

- Did not want to have Surveytrak, Blaise, and EHC running at same time on the laptops as this was creating problems with memory usage.
- Experienced a lot of problems trying to implement alien routers/procedures with very large applications. Stability of the DLL was a concern as it could cause an entire windows system crash.

Given a project requirement that uses multiple applications as a part of an interview, the MAI can be setup in a database (making it data driven) to handle all the applications in the sequence in which they need to be executed for a given project. The current design allows the execution of project-specific computations or business logic before and after the completion of each application. This makes sure all the interview requirements are done by the current application and data is passed from one application to the next if needed. The MAI can be customized to project-specific requirements if this pre-defined deployment sequence needs to be overridden for any reason.

The MAI provides a handle to control and build the preload for the consecutive applications at the end of each current application. For example, using Blaise, a preload string is to be built for the Event History Calendar (EHC); it needs to create a MDB database and populate it with the data collected in the Blaise instrument. The MAI makes extensive use of the BCP function to pull data out of Blaise and normal SQL to update the MDBs.

The MAI was designed, initially, for a small project where the communication between each application was very minimal and the preloaded data that was

required by each was small and already available in Surveytrak and not being pulled from one application or other. Surveytrak could easily, via the MAI, pass the information required by each application. The interdependence between applications was very small.

However, with the PSID project, the problem with the design became very apparent due to high interdependence between the five applications. The redesign involved creating the ability to pull information from completed portions of the interview and pass it along to the next application in the form of preload as needed. The MAI, as it existed prior to the PSID project, did not have this capability. Thus, this functionality had to be programmed into the MAI because the PSID interview created this environment of interdependence between the various applications (Blaise and EHC). For the basic MAI logic flow, please refer to *Appendix A* on page 12.

Currently, the MAI handles the various computer-based components used to collect data at SRC. It remains to be seen how the system will accommodate new data collection methods in the future. All SRC computer-assisted data collection methods that could be used in the system have been accommodated for and the goal is that the introduction of any new methods will be met with very little change to the basic design of the MAI.

## 4. MAI Options

### *Complete, Start New, or Readback Interview (Non-Finalized Interviews)*

- “Complete Interview” is an option in MAI projects that instructs the MAI to pick up the interview at whatever point it was suspended and deploy the correct application.



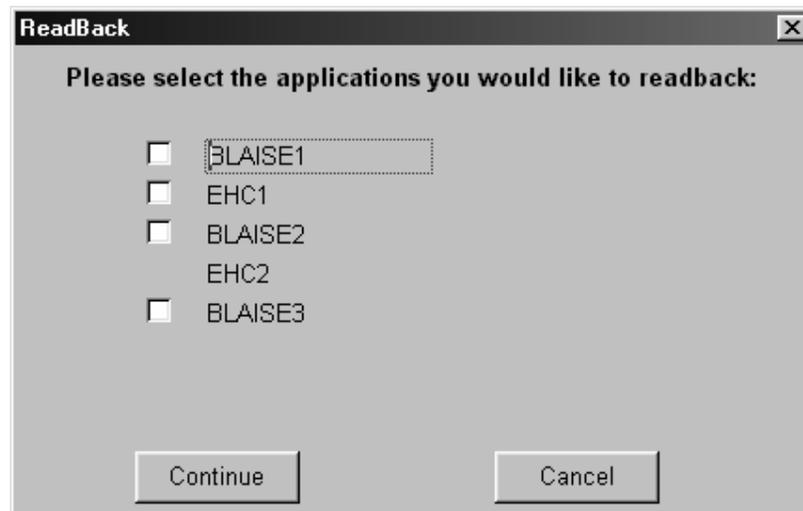
- “Start New Interview” is another option in MAI projects when a user tries to open a case that is a partial (suspended) or completed interview. Choosing this option will wipe out all the data as a part of the interview and will start the interview as a new interview. Internally, the MAI backs-up each application that is complete or suspended. This is helpful to restore the data if the user chooses this option by mistake.



- Readback is also an option provided in MAI projects when user tries to interview a partial or completed interview.

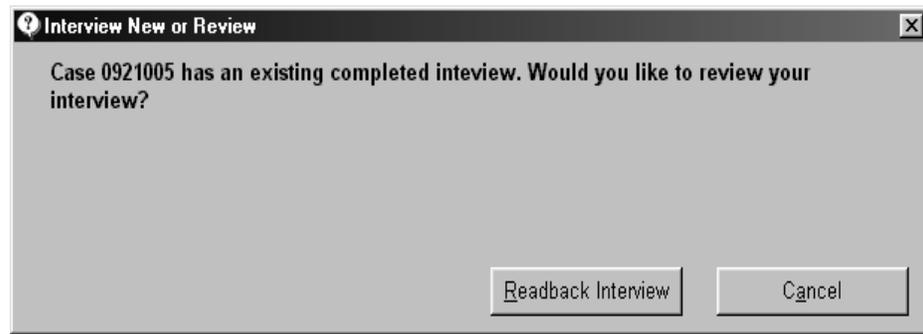


Readback is a read-only mode allowing the user to go through any case that is partial or complete. Choosing this option will list all the applications that are a part of one interview and enable selections for only those applications that are eligible for readback. For an application to be eligible for readback, it should have a partial or complete status and should have a related dataset of the same case in Surveytrak. The MAI sends the indicator to the application telling it that its mode of access is readback only. The respective application that is a part of readback should then handle readback within itself.



#### 4.1 Completed Interviews (Finalized)

The option below is provided to the user when the interview is complete and result code is coded as a final complete (e.g. 01 or 1001).



#### 4.2 Partial Interviews (Finalized)

The option below is provided to the user when the interview is suspended and result code is coded as partial final (e.g. 05 or 1005).



#### 4.3 MAI Data Merge

- Takes the interview information from an individual case and appends it to a master database.
- The number of datasets corresponds to the number of applications used to comprise the Interview. For example, in PSID, there were a total of 5 applications. The merge produced 5 separate master datasets: three Blaise databases and two MDBs from the EHC applications.
- Updates the SurveyTrak database with the interview length. The length of each application is calculated and a total length, involving all applications, is computed and all of this information is stored in the Surveytrak database.
- Reports errors if any are encountered.

#### 4.4 Merge criteria

Each application will have its own unique merge criteria. The following are the criteria required for each merge set-up:

- Name/description of merge
- If merge should be run automated or on demand
- Whether merge is active
- Path to the master database
- Paths to other merge-related files

Description	Frequency	Days Run	Active	Begin Date	End Date
NSHS-SR obs merge	Weekly	1	Active	04/09/2002	01/01/2003
NSHS-SR Production	Daily	5	Active	04/09/2002	01/01/2003
NSHS-SR Training 1	On Demand		Active	04/09/2002	01/01/2003
Pitt SuperMerge	On Demand				
Pitt1 Merge	On Demand				
Pitt2 Merge	On Demand				
PSID EHC 1 Merge	On Demand		Active	09/01/2002	01/01/2004
Test of Admin and Merge	On Demand		Non-Active		
testing - Jingyi	Daily	5		02/05/2000	02/21/2000
Train2 Main	On Demand				

Cases that are successfully merged into the master database are recorded in the “*tMerge\_Track table*” of the SurveyTrak database. Any errors that are encountered are output to the status window and recorded in the “*tMerge\_Error table*” of the Surveytrak database.

**Interview Data Merge -- ON DEMAND MODE**

```

---Project Id: "SRC.DST.PSID03.TEST" -- Merge Id: "EHC1"
Sample line 0009001: Interview data is null.
Sample line 0921004: Interview data is null.
Sample line 0921011: Interview data is null.
Sample line 0921312: Error code -24
Sample line 2933003: Interview data is null.
Sample line 2933302: Interview data is null.
Sample line 9904001: Interview data is null.
Sample line 9904002: Interview data is null.
Sample line 9905001: Interview data is null.
-----
... Project-Merge done
-----

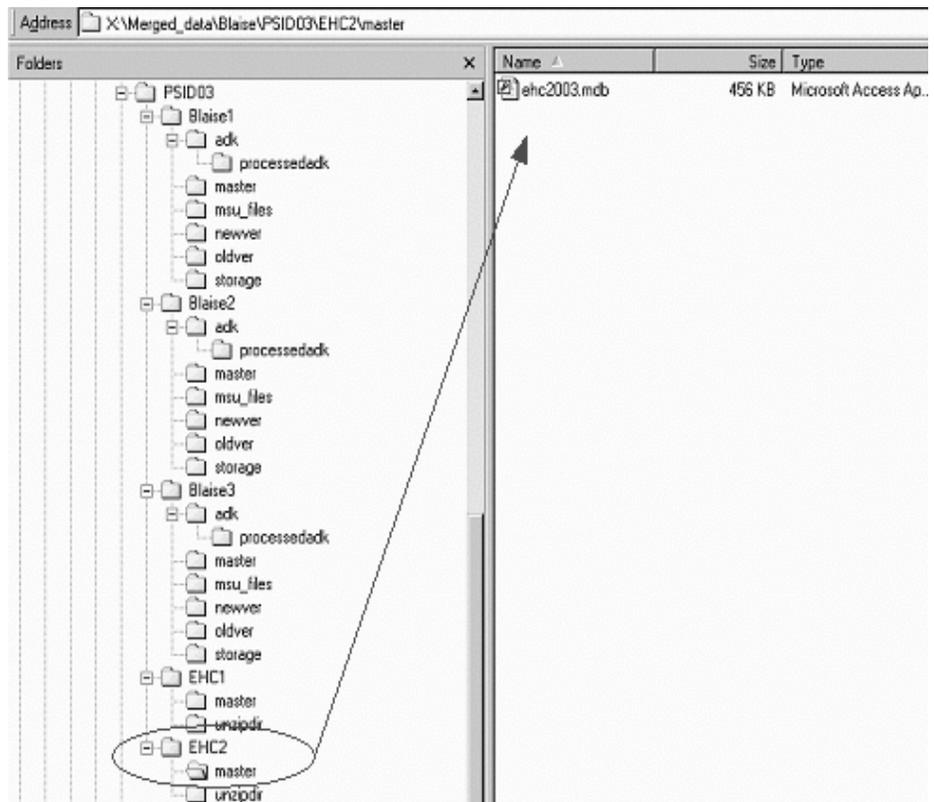
```

**Sample Lines to Process:**       **Merged:**       **Select**  
**Processing Sample Line Id:**       **Error Cases:**       **Exit**  
**Processed:**

Ready

#### 4.5 Data Set Production

Takes the interview information from an individual case and appends it to a master database. In this image, you can see the directory structure of five different instruments and the data contained in one of those instruments:

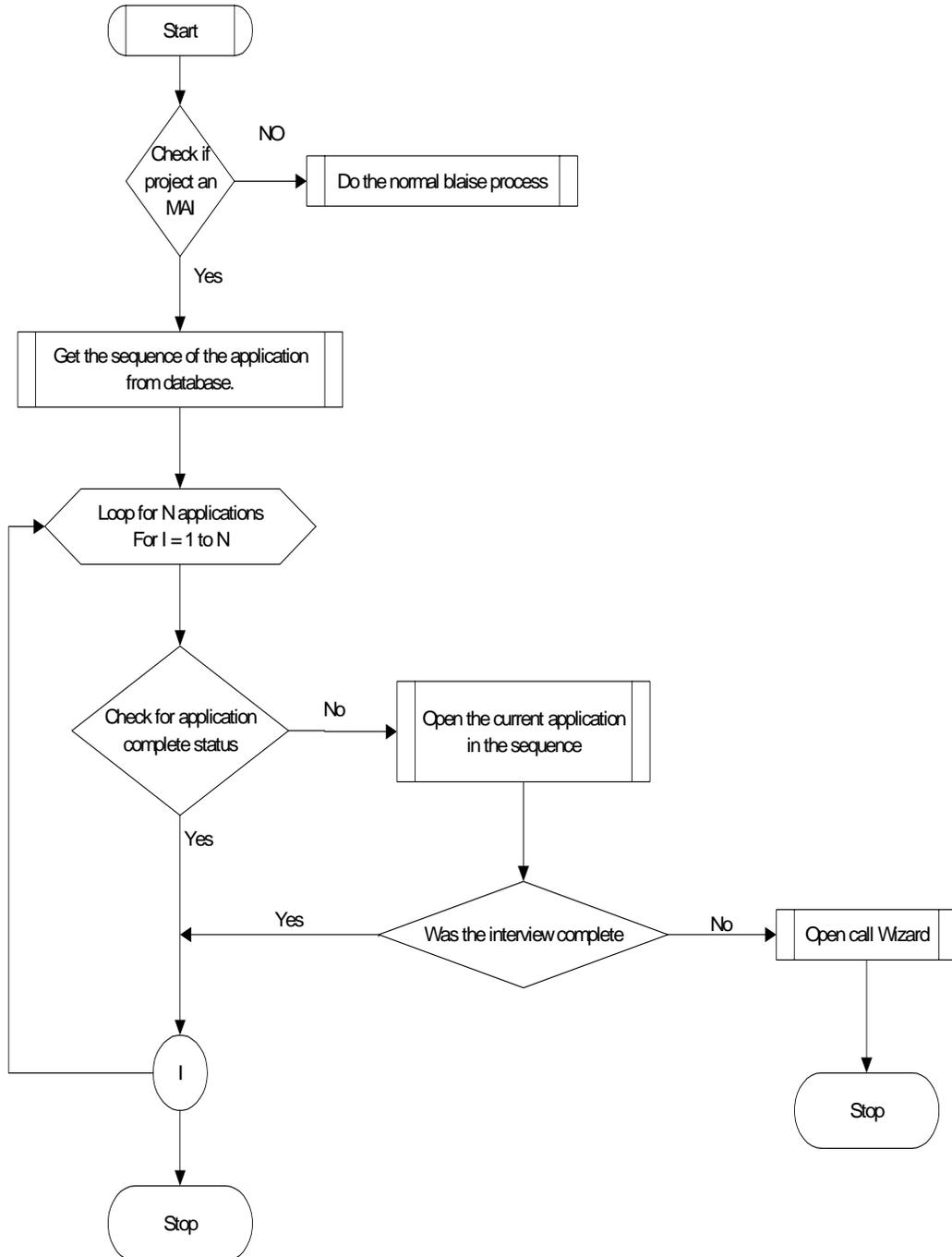


## 5. Summary

Even though it was challenged by the various preload needs, the MAI has stood up to the task and has consistently delivered. It has demonstrated a robustness and capacity to handle any number of applications consistently and successfully.

## 6. Appendix A

### MAI Logic Flow





# Electronic Data Reporter

*G.W. de Bolster, Statistics Netherlands*

## 1. Introduction

The Electronic Data Reporter is a 99.99% Blaise build data collection module. It's build in Blaise 4.6 using part of its new functionality created mainly because of this development. Only the launcher was created as a small C++ program. The module is multi data collector, multi survey, multi questionnaire, multi data communication, multi user, multi language, multi statements and whatever multi ... you can think of. It is built to be as user friendly as possible and it's operational for CBS since the beginning of this year.

The Electronic Data Reporter has been developed to replace the EDISENT<sup>8</sup> module. This module, also a development by Statistics Netherlands, was build for a 16-bits environment and therefore difficult to maintain and extend. Although it could handle numerous questionnaires it also had several restrictions. With the introduction of the Electronic Data Reporter most of these restrictions are eliminated. Especially the use of the Blaise language for the definition of the questionnaires extended the possibilities for questionnaire design enormously as all the flexibility of Blaise can be applied. Just like EDISENT the Electronic Data Reporter must be installed at the respondent. After installation surveys can be send by the data collector e.g. by e-mail and automatically imported in the Electronic Data Reporter environment. Every data collector can supply its own data communication software which can be added separately. Even the software itself can be extended and updated by e-mail.

Different data collectors can use the same installed Electronic Data Collector. A level called *survey* has been introduced to create the possibility of hierarchical questionnaires. Surveys are grouped by data collector. The questionnaires can use different code lists e.g. for the lookup of answer values. With the possibility of shared code lists voluminous code lists can be stored only once while used in the surveys of many data collectors. Tests with code lists containing over 35.000 elements had almost no influence on the high performance.

Every questionnaire can contain its own hierarchy of keys ten levels deep. All kind of key types (predefined using a lookup or just open) are supported. Predefined values for keys can be maintained by the data collector when sending questionnaires. Routing and checking is available for open keys. The Electronic Data Reporter provides storage facilities for statements.

## 2. History

To lower the administrative burden of the respondents Statistics Netherlands has been active in the field of electronic data collection since the beginning of the nineties. For this purpose several tools have been developed, always applying Blaise as much as possible.

As the borders in the EU opened for trade in 1993 the reporting burden related to this trade shifted from customs to the enterprises. To limit this burden as much as

---

<sup>8</sup> EDI between Statistics and ENTerprises.

possible IRIS<sup>9</sup>, a dedicated tool for the collection of data for the Foreign Trade survey, was developed. The first version was built in Blaise 2.5 and operational since January 1, 1993. It was quite successful from the start and therefore it was further developed during the years. The current version is build with Blaise 4.5. For special functions a Delphi DLL is added. Besides manual data entry a semi-fixed import facility is available. I refer to it as “semi” because although the fields are fixed their place in the record can be defined by the user.

In 1993 a pilot project called EDI Pilot1 was started to develop a more generic process of collecting data from enterprises for all kind of surveys. This pilot project resulted in 1994 in the proto type of the EDISENT tool. This tool was based on the concept of flexible, tuneable import functionality and it supported multiple questionnaires. The proto type was build using Turbo Vision 6 in combination with Blaise 2.5. With this module a small pilot was performed involving 10 enterprises. As the results were positive the management gave the assignment to start a second pilot project, called EDI Pilot 2. The goal of this project was to develop a more robust version of EDISENT and test it out with a larger number of respondents. The project started in 1995 and the development of the new version of the EDISENT module was placed under the same responsibility as Blaise development because of the available knowledge of such tools in this department. It was build completely in Turbo Vision using several parts of the Blaise sources. In 1996 the pilot phase was ended with some 150 enterprises using this tool for reporting statistical data to Statistics Netherlands. In 1996 Statistics Netherlands introduced the EDISENT concept in the European funded TELER<sup>10</sup> project. In this project a Windows 3.1 version of EDISENT with improved functionality was built in Delphi 1 still using parts of the Blaise sources. It was tested out in 8 countries (Sweden, Finland, Germany, Italy, Spain, Portugal, Slovenia and The Netherlands). Today it is still in use in Slovenia and The Netherlands.

Given the limitations within the project there were still a number of restrictions left in the functionality of EDISENT. One of them was the limited flexibility concerning the design of the questionnaires. The lay-out was rather fixed and only a basic set of elements could be applied to construct questions. There were hardly any rules possible, only range checks on numeric fields were allowed. The identification of the questionnaires consisted of *unit* and *reporting period*. To create questionnaires a program was developed using an ASCII definition as input. It appeared to be difficult to integrate this program in a data collection system. After the project has ended the EDISENT module was improved several times but the limitations of the programming language (Delphi 1) soon were felt. As Statistics Netherlands was in the possession of Blaise it was considered not opportune to put more effort in further development of EDISENT.

Once installed and tuned at the respondent EDISENT fulfilled its promise to reduce substantially the administrative burden. However, as the initial costs of tuning were to high the management of Statistics Netherlands decided to abandon the use of EDISENT. In stead a new tool was needed to replace EDISENT but limited to support only the data entry functionality. For this purpose the Electronic Data Reporter was developed. The challenge was to apply as much as Blaise as possible. Not only the integration of underlying parts would be as optimal as possible, also the tool will then be very light to install. Thanks to additions in Blaise 4.6 this goal was achieved: with the exception of the launcher the Electronic Data Reporter is completely built in Blaise. Besides the Electronic Data Reporter (and still IRIS and EDISENT) Statistics Netherlands is also using Blaise IS for its electronic data collection.

---

<sup>9</sup> Interactive Registration of International trade Statistics.

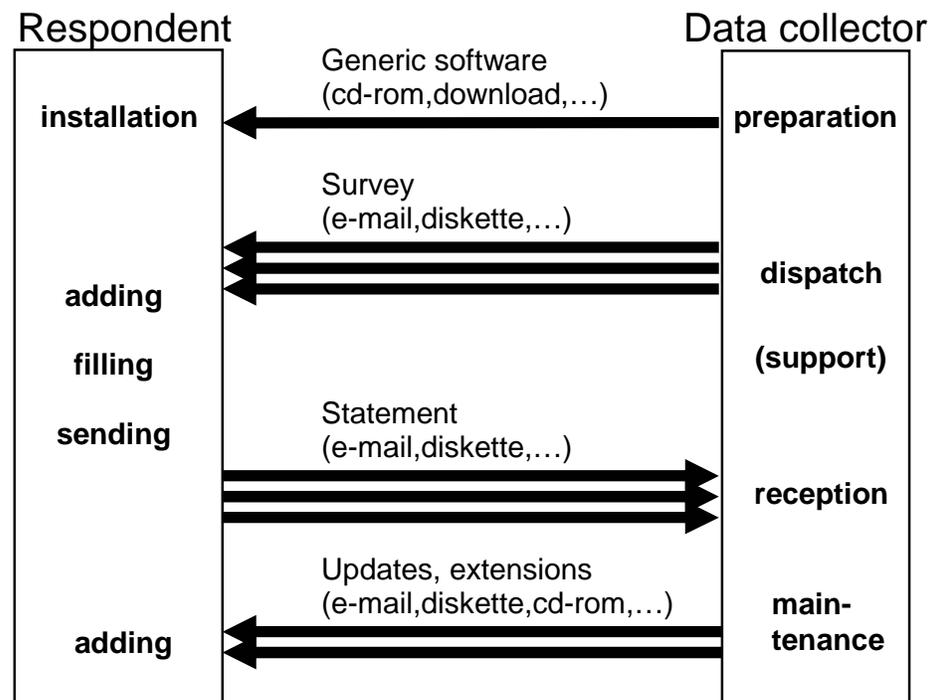
<sup>10</sup> TELematics for Enterprise Reporting

### 3. General process

Just like EDISENT the Electronic Data Reporter must be installed at the respondent. The installation is only necessary for a generic part of the tool. Although this is done by sending an installable version on a CD-rom the whole tool is light enough to be sent by e-mail. Given the right compression it can be reduced to a little bit over 2 MB. As it does not contain any components, registration is not necessary. As a matter of fact it can even be distributed as a self-extracting executable or zip-file.

The Electronic Data Reporter has been developed to support the electronic data collection by a data collector (the organisation that is requesting the data like a NSI<sup>11</sup> or the Tax authority) at respondents (the organisations that are providing the data like enterprises and institutes). It has not been developed for typical statistical data collection, it can be used for the collection of all kind of data. An installed version of the Electronic Data Reporter can be used by different data collectors simultaneously.

Figure 1. The general process



After installation surveys can be sent by the data collector to the respondent e.g. by e-mail. These surveys are in reality Blaise files (bmi, bdm, bxi, bdv) and Maniplus set-up's (msu) packed in Microsoft cabinet files (cab). If predefined data is included too (e.g. lookup files) it can be in the shape of Blaise databases (bdb+) or ASCII files. Finally a small XML-file is added containing information about the survey and the data collector such as identifiers and names. These so called supplement files are given the extension *esf* (Electronic Data Reporter Supplement Files). Using a registered file association the Electronic Data Reporter can be invoked by merely clicking on the supplement file. As a result the survey is automatically imported in the Electronic Data Reporter environment and, depending on a tag in the XML-file, the included questionnaire is opened and the

<sup>11</sup> National Statistical Institute

user is invited to fill it in and send the data (statement) directly to Statistics Netherlands using the included data communication software.

Every data collector can supply its own data communication software which can be added separately packing it in the a supplement file just like the surveys. Again the XML-file will inform the Electronic Data Reporter about the content. Even the software itself (including the Blaise run-time engine Manipula.exe) can be extended and updated using the same procedure. The concept of sending the surveys separately by mail was already implemented before in one of the last versions of EDISENT. As a data collector can control the whole functioning of the data collecting tool by just changing the meta data on a distant this concept is called Remote Meta Data Control or RMDC. This concept was once proposed as part of a EU-funded project (METER) but the project was eventually not accepted for organisational reasons.

#### 4. Logical structure

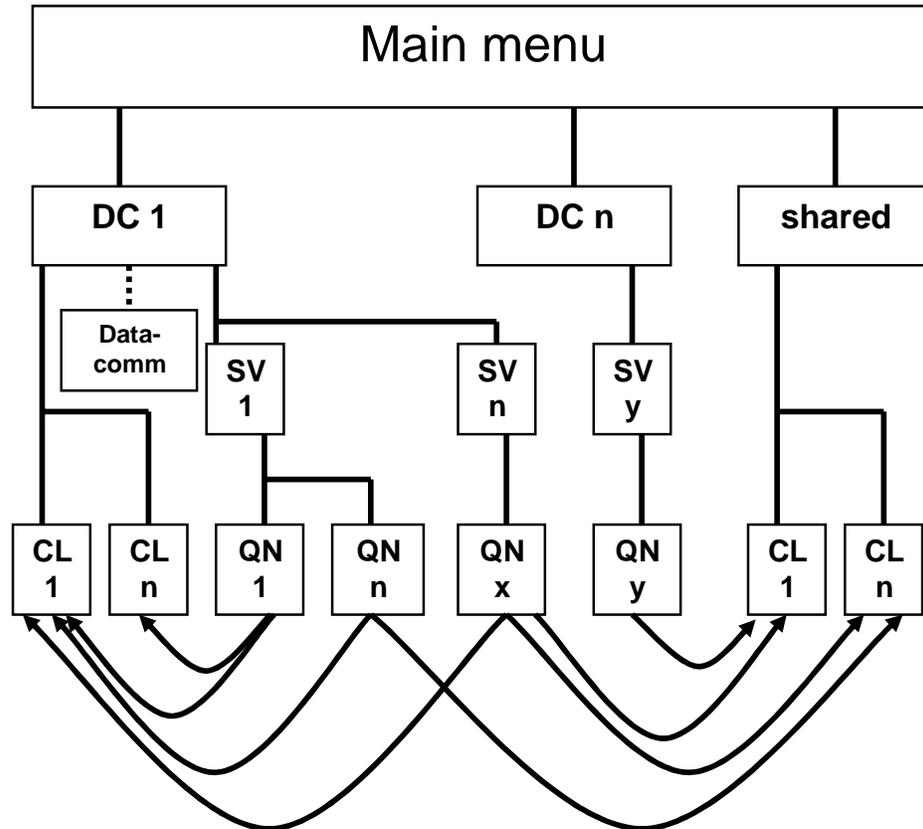
The general (and generic) access to the Electronic Data Collector is through the main menu. Below this main level the logical structure is divided per *data collector* as the tool has been developed for the simultaneous use by various data collectors. Each data collector should therefore have a unique identifier consisting of maximal 8 characters. The identifiers are also used as file names. To support still older environments as Novell 3.11 these names should be limited to 8 characters. The data collector level is parent for the next two levels: *surveys* and *codelists*. The survey level has been introduced to make it possible to apply a (hierarchical) set of questionnaires. For the survey on Road Transport a questionnaire set consisting of 4 linked questionnaires was created: *enterprise*, *vehicle*, *journey* and *load*. The survey is uniquely identified only within the set of the data collector. In the same line the identifiers of the questionnaires only have to be unique within the boundaries of a survey. Placing the code lists next to the surveys allows you to use the same code list in different surveys. Not only will it save disk space but it also guarantees more consistency. The code lists have there own set of unique identifiers within the domain of the data collector, separately from the surveys. Parallel to the data collector a special entry *SHARED* was introduced for the storage of code lists that can be used by all the data collectors.

The user is accessing the different functionalities through the main menu. The two most important entries are *Statement* and *Survey*. Starting up the Electronic Data Reporter the user is more or less forced to select one of the surveys of one of the data collectors as the active one. The name and code of this survey is clearly visible in the main window so the user is constantly aware which survey he is working on. Every option within the *Statement* sub-menu is now related to actions on statements belonging to this survey, with the exception of *send* (all statements for the current active *data collector* are sent in one message if the option is activated).

Within the *Survey* sub-menu options to manipulate surveys are available like selecting another survey. The level of questionnaires is not available to the user. A user is expected to create statements for a survey. Questionnaires are only considered as the means to do so. As it can differ from survey to survey the routing between different questionnaires within a survey must be controlled by the survey itself. Therefore this functionality is included in the processes that are typical to a survey and not in the generic menu. In most of the cases a survey will consist of a single questionnaire so activating a survey is almost the same as activating a questionnaire.

Normally the code lists are fixed lists and accessible through a questionnaire, e.g. as a lookup. A data collector can also include code lists that can (or must) be edited by the user before using the resulting data in a questionnaire. This can be useful if local user codes have to be translated to standard data collector codes. In the case a data collector has defined a code list as editable it is accessible through the option *Code list* in the *Survey* sub-menu.

Figure 2: The logical structure (DC = data collector, SV = survey, QN = questionnaire, CL = code list)



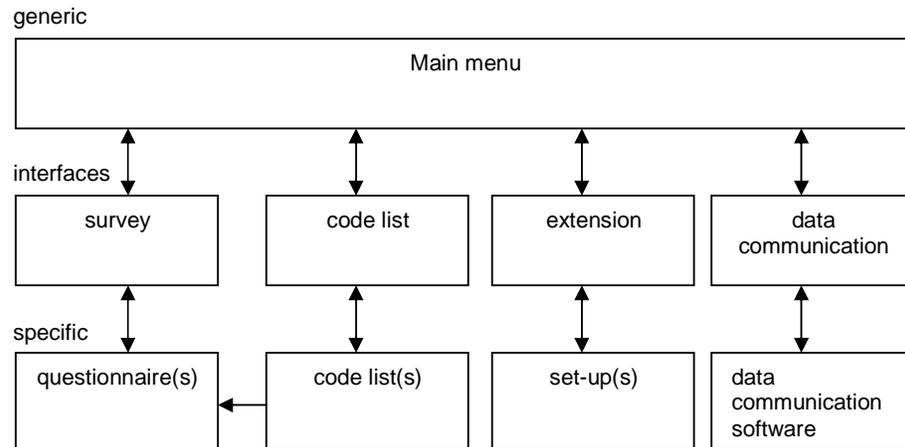
## 5. Technical structure

From a more technical point of view the architecture of the Electronic Data Reporter consists of three layers. The first layer is the generic main menu. The main menu is controlling the next layer which contains the interfaces with the specific data models of the questionnaires and the code lists. The latter one is only present if the code list is editable. Furthermore this layer contains the interfaces with the data collector dependant data communication software. Consequently the third layer contains the questionnaires, the code lists and the data communication software. The Electronic Data Reporter tool can be extended with extra functionality (like imports) using add-ons. They are also using an interface being part of the second layer.

An alternative way of looking at the structure is to see the combinations of an interface with its specific parts (questionnaires, code lists, data communication software and extensions) as objects added to the generic main menu.

In the next paragraphs a small overview is given of the way the different parts of the Electronic Data Reporter are working and communicating with their environment.

Figure 3 – The 3 layers



### 5.1 The main menu

The main menu itself is a Maniplus set-up with a fixed set of Blaise files for administrative purposes. In these Blaise files the codes and names of data collectors, surveys and code lists are registered including some attributes. The menu itself and all display texts (including button texts) are read from an external ASCII file. For different languages different ASCII files are used, the language code is part of the file name. This language code is stored in the registry. Under Win2000 and XP the language setting (and other settings) is even user dependant.

To communicate with the registry the main menu simply ‘runs’ the standard Windows program *Regedit.exe* with command line options. Depending on the command line options it can be used to write to or to export from the registry. The data that is interchanged between the *Regedit.exe* program and the main menu is written to and read from a so-called *.reg* file.

In many other cases standard Windows or ‘user’ programs are used to perform special functions. In stead of supporting all kind of printers the Electronic Data Reporting produces reports as RTF or ASCII files and invokes then the associated text editor by calling the standard Windows program *RunDLL.exe* with the right set of parameters. It is left to the text editor to provide the printing facilities. Another standard Windows program called *Extrac32.exe* is used to extract files from Microsoft cabinet files. These cabinet files are used as supplement files mentioned in chapter 3. To create these cabinet files a free Microsoft program called *CabArc.exe* is available on MSDN<sup>12</sup>. This program is used at the data collector’ to create the supplement files.

The main menu is using the standard defined temporary map to store all kind of working files. The path of the standard temporary map is obtained from the environment variables TEMP or TMP or the equivalent registry entries. A user setting is available to select another temporary map.

The Electronic Data Reporter is working with a fixed folder structure relative to the main installation folder. The unique codes identifying the data collector, the survey and the code list are used as folder names. The launcher, the only C++ program within the Electronic Data Reporter called *EDataRep.exe*, is making the main installation folder the current folder. By so doing the set-ups and data models can use relative file paths.

<sup>12</sup> Microsoft Developer Network



In the interface set-up the key values are asked using Maniplus dialogs. In case of creating a new form the interface set-up determines if the key already exists and if it does it offers the user the possibility to create another version of the form. The right version number is automatically created. Using Maniplus dialogs makes it possible to create a very neat user interface and even a very sophisticated (hierarchical) selection of predefined key values.

#### 5.4 The data communication software

The data communication software is data collector dependant and normally written in languages like Delphi or C++. It is invoked by a special interface with the fixed name *EDRsend.msu*, which deals with the special commands to invoke the different functions of this program and to interpret the feedback. Statistics Netherlands uses its own program (*CBScript.exe*) that applies Triple-DES and RSA for encryption (according to the PKI<sup>13</sup> concept). It uses the local mail client for sending the statements over the Internet by SMTP<sup>14</sup>.

### 6. Blaise 4.6. features

The Electronic Data Reporter was more or less developed parallel with Blaise 4.6. Many of the new features in Blaise 4.6 (at least the ones in Maniplus) were added based on needs emerging from the development of the Electronic Data Reporter. Consequently Blaise 4.6 was tested automatically while testing the Electronic Data Reporter. This implicit co-operation appeared to be very fruitful for both developments.

The new features of Blaise 4.6 listed below are used in the Electronic Data Reporter:

- Main window display (text and logo). The name and the code of the current active survey are displayed in the main window and the logo of the related data collector is shown at the right bottom corner.
- Hourglass display in message boxes while an action is performed.
- Fills in dialogs (including the buttons) to make them multi-lingual applying text read from external files.
- Improved shared functions for Blaise files creating a stable multi-user environment.
- Integration of help files so the same help file is used in Maniplus set-ups (interfaces) as well as data models (questionnaires).
- User-defined name for help files to make them multi-lingual.
- System-text-file parameter (/#) so even the system messages are multi-lingual.
- Version information string used in the 'about' box and status report.
- Database views for lookups in data models creating customised and multi-lingual lookups.
- Common stop option (Alt-F4) providing a more consistent user interface.
- Extended RUN-commands avoiding the use of non-Blaise programs for file handling.
- User-defined HALT values for better and easier communication between set-ups.
- Select folder dialog for user-defined work folder setting.
- Save file dialog for interactive user controlled storage of output files.
- Default NO in Confirm dialog avoiding user errors.
- IN- and OUTPUTPATH function for better sharing of data between set-ups.

---

<sup>13</sup> Public Key Infrastructure

<sup>14</sup> Simple Mail Transport Protocol

Other new features of Blaise 4.6 like the dialog boxes will be included in future releases of the Electronic Data Reporter as well as a better use of the other features.

## 7. Current use

The Electronic Data Reporter was first put in production for the Traffic and Transport survey at January 2003. The tool was not disseminated by Statistics Netherlands itself but by SIEV, an organisation for the transport branch. They cooperate with Statistics Netherlands collecting the data for both data collectors at the same time. NIWO, another branch organisation will join soon. The independency of the Electronic Data Reporter from the data collector appeared to be useful from the beginning. As mentioned before, the survey consisted of 4 hierarchical questionnaires with large code lists for lookups. Some of these code lists contain about 40.000 records and even a trigram search gave no performance problems.

In March the survey of International Trade in Services started with the use of the Electronic Data Reporter. For this survey the import extension was added. It is now used for a dedicated ASCII record set but an XBRL alternative already has been developed.

Soon the Electronic Data Reporter will be put in action for the Short Term Production survey. Unlike the other surveys the Blaise questionnaires for this survey will be generated from a questionnaire server (part of the LogiQuest data collection system) and send automatically by e-mail to the respondents.

More surveys are testing the tool at this moment.

## 8. Future developments

The basic architecture of the Electronic Data Reporter in combination with the Blaise 4.6 language makes it very easy to extend its functionality. Some features that are on the list to be added:

- Extended and flexible import facilities
- Complete multi-lingual environment
- Archive facilities
- Export facilities
- XML messaging (ebXML<sup>15</sup>, XML4DR<sup>16</sup>). This feature is developed in cooperation with members of the SOS<sup>17</sup>-group.
- Certified distribution packages

With the growing use of the Electronic Data Reporter more useful features are expected to be identified and thereupon added to the list.

---

<sup>15</sup> Electronic Business XML

<sup>16</sup> XML for Data Reporting

<sup>17</sup> Statistical Open Standards



# Integration of CAPI and CATI infrastructures at Statistics Canada

*Jacqueline Mayda & Pamela Ford, Statistics Canada*

## 1. Background and motivation for the integration

Statistics Canada's numerous social surveys have been conducted for many years now using Computer Assisted Interviewing applications, some in person (Computer Assisted Personal Interviewing, CAPI) and others by telephone (Computer Assisted Telephone Interviewing, CATI). The telephone component can be conducted by the field staff from their homes using the same application as used for CAPI, or can be conducted in the call center in each Regional Office. The Blaise software is currently being used to author all of these collection applications. Workload planning and allocation is done differently depending on the mode of collection. Field interviewers have specific cases assigned to them through case-management software that was developed at Statistics Canada and interacts with the Blaise survey applications. Telephone interviewing by field staff from their home does not require the same functionality as a full CATI environment in a Regional office where the Blaise call scheduler allocates the cases amongst many CATI interviewers depending on the case criteria, respecting appointment times and the like.

A few years ago the introduction of a new large-scale survey (the Canadian Community Health Survey) dramatically increased the number of personal interviews. In order to meet the capacity requirements, a decision was made to consolidate the telephone interviewing portion of all surveys into the existing Regional Offices throughout the country. Due to the short time frame for implementation of this initiative, a formal redesign of the software was not possible. As an interim measure, survey applications previously used solely by field staff were quickly set up in a pseudo-CATI environment in the workstations in the call centers, with the software functioning exactly as it did on the laptops. Consequently this did not allow for any of the features of a CATI environment, such as the use of the Blaise call scheduler.

Management problems in the Regional Offices (in terms of assigning cases to interviewers and workload planning) arose as a result of not having a call scheduler facility in this environment. Pseudo-CATI and CATI (call scheduler) applications share workstations. When a workstation is being used for CATI surveys, its pseudo-CATI cases are completely unavailable. When a workstation is being used for pseudo-CATI, it cannot be used for CATI surveys. Managers must print lists of cases with appointments and ensure that both workstations and interviewers are available at the appropriate times. In addition, transferring cases from Regional Office servers to laptops for current full-CATI (call scheduler) surveys is not possible due to the difference in software configuration; we can copy files to LAN servers for CATI, but there is a rich mix of software and files involved in CAPI. To change from CAPI to CATI or vice versa would require a significant investment in additional development time.

Given all of these issues it was felt by all stakeholders that the best approach would be to undertake a redesign of the infrastructure for the CAPI and CATI collection applications with the goal of one system that could be used in both environments, and the ability to transfer cases between the two environments. The full integration option in terms of cost and implementation issues was investigated, but the time

and resources needed to develop the components required for full integration, the impact of a “big bang” approach for clients, the need for a lengthy implementation and testing window, as well as a high cost, made this option less desirable.

Instead, a partial integration approach was adopted. This approach promises to render collection operations more efficient by introducing a true CATI environment with the utilization of a call scheduler starting with the Labour Force Survey (LFS) and its supplementary surveys, and progressively converting each of the other CAPI/CATI surveys to the new model. While the partial integration does not provide for one fully integrated CAPI/CATI environment, it does maintain a transfer process between CAPI and CATI, albeit a manual one. An additional benefit of the integration relates to the programming of the applications. Since the original conversion of CAPI surveys to Blaise, there has been a move towards more coding standards, and the development has become more efficient. These standards can now be applied to the Labour Force and supplementary survey applications as part of the redesigned infrastructure.

## 2. Description of the Existing System

In the current CAPI environment, cases are stored in individual databases in individual directories and one database is created *per case, per laptop*. For surveys that share a sample, such as the LFS and its Supplements, each survey has physically separate cases. Case-management software, called CASEMAN and developed in Sybase by Statistics Canada, is used to send and retrieve data to and from the laptop computers and to workstations in the regional offices. The case management (assignment planning, reporting) is done on one database for all surveys. Survey Collection applications consist of a combination of Blaise and Visual Basic components. Most of the questionnaire modules have been developed in Blaise, with a few left in Visual Basic. All components are controlled by a Visual Basic program manager that has the ability to 'communicate' with CASEMAN. The pseudo-CATI environment is the same as the laptop CAPI environment, except that the applications reside on Network Workstations in a Regional Office.

The full CATI (call scheduler) environment has quite a different configuration than that of the CAPI infrastructure. All survey data are stored in a single database, and there is one database *per survey per CATI Server*. In each region, each survey has its own Blaise call scheduler. Case Management (call scheduling, reporting) is done on the survey database. All components of all full CATI surveys have been developed in Blaise.

## 3. Objectives of the redesign

To attain the overall goal of providing an environment to effectively manage interviewing resources, respondent contacts, and respondent burden, the project team delineated the following requirements as objectives that must be met and that drive the design of the integrated system:

- ❑ There is a need for one collection system useable in both the CAPI and CATI environments.
- ❑ All CATI surveys should be administered using the Blaise call scheduler.
- ❑ There should be a means of transferring cases between platforms.
- ❑ Surveys must be able to share samples, but receive data and have their progress tracked individually. In addition, surveys sharing a sample should be permitted to have varying collection periods.

- There should be a mechanism to effectively manage interviewing resources (interviewers, workstations, workloads, etc....) in both CAPI and CATI environments.

#### 4. Conceptual Model

The existing CASEMAN system already provides functionality for CAPI assignment planning and for transmitting cases to CAPI laptops. As any significant modification or redesign would come with great cost, it was determined that using this software (with the least change) to move cases between platforms should be an additional requirement.

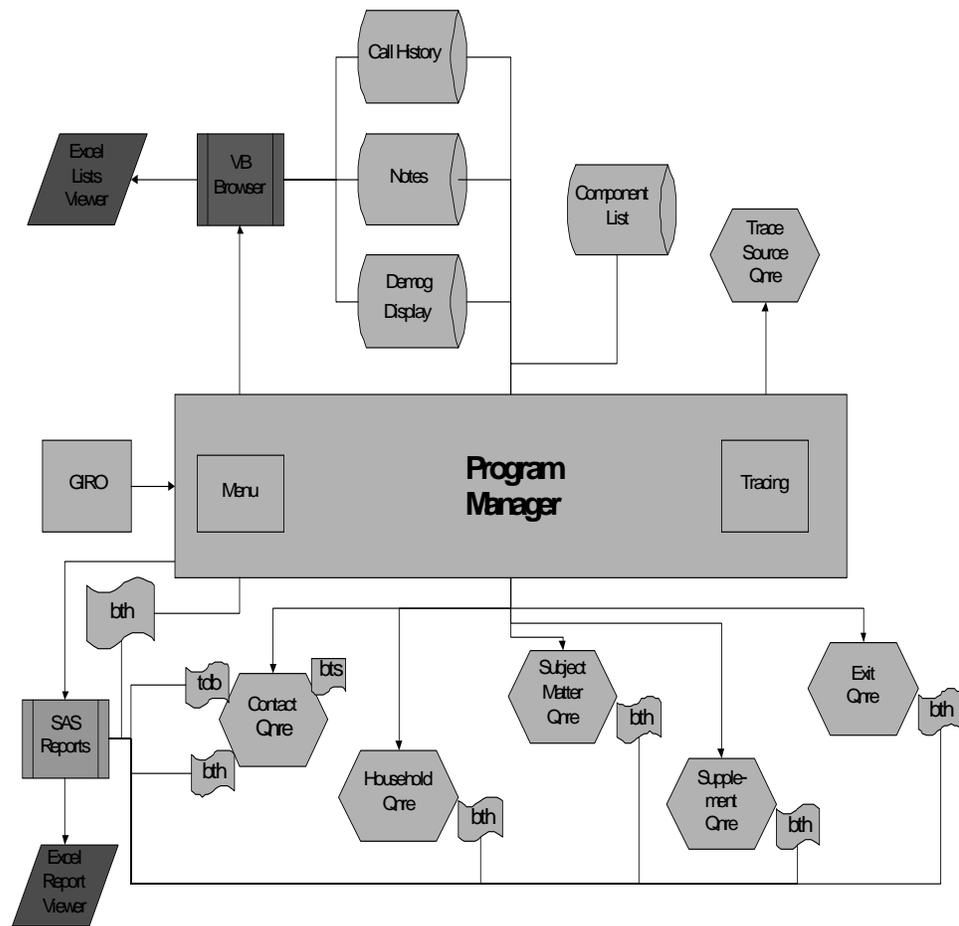
Furthermore, it was determined that there would be additional savings in using the call scheduler included in the Blaise software package which is already in use for existing full CATI surveys, rather than adding this functionality to CASEMAN. This last requirement implies that the existing CATI Survey application model, or one similar to it, must be implemented on the laptops. As the existing CATI model is not suitable for surveys sharing a sample, such as the LFS and its supplements, a new model has been proposed.

*A model had to be devised for survey collection applications so that they could be implemented in both CAPI and CATI environments. The approach currently in use on the laptops is not appropriate for call scheduling; a call scheduler would have to be implemented into CASEMAN. The approach currently in use on CATI (call scheduler) workstations can be used on laptops, but the interviewing application would be independent from CASEMAN and an interface between the two systems would have to be developed if CASEMAN were to be used for transmitting cases.*

Furthermore, surveys that share a sample must be treated as a single entity to enable the use of a call scheduler. For LFS and its supplements, this means that a single entity must be created which encompasses the parent (LFS) and supplement client records for the sample unit. This entity is being referred to as the master sample entity (MSE). The CATI call scheduler will schedule MSE cases rather than individual LFS or supplement cases.

Until such time that an interface between CASEMAN and the new survey application model can be developed and implemented, it will be necessary to maintain the existing survey application model in the CAPI environment. It will nevertheless be possible and advisable to implement the actual Blaise questionnaire components being used in the CATI application into this model. Figure 1 shows the conceptual model of the CAPI/CATI environment:

Figure 1: Conceptual Model CAPI/CATI environment



#### 4.1 Program Managers

A program manager is the driver of a collection system. It controls the flow between questionnaire components and launches the questionnaire components when appropriate.

In the CAPI environment, each survey will have its own Visual Basic program manager with supplement program managers being launched at times by the parent program manager. CAPI program managers also consist of the functions required to send and receive information to and from the Sybase CASEMAN database. When collection of the parent survey is complete, interviewers access each of the supplement surveys directly through their individual program managers.

In the CATI environment there will be one Maniplus program manager that controls all parent and supplement questionnaire components.

For both program managers, much of the code surrounding the launching of the questionnaires themselves is generic and does not require modification from survey to survey. The benefits of this design are obvious.

## 4.2 Questionnaire Components

There are four types of questionnaire components. The same components will be used in each of the CAPI and CATI environments.

The Contact questionnaire has three roles. The first is to record whether contact has been made at the dwelling or telephone number level and to indicate which treatment should be applied in the event of non-contact. The second role is to guide the interviewer in making contact with an appropriate respondent for the survey. Finally the contact questionnaire is also used to provide introductory information for the respondent, to obtain the respondent's language of preference, and to inform them, when applicable, that the interview may be monitored.

The Household questionnaire is used to collect household-level data. Normally the information collected is used to determine which subject matter detail should be collected and who should be interviewed further.

Subject matter questionnaires are used to collect the main content of a survey. The Labour Force Survey has two subject matter components, with various supplements each contributing additional components when required. For multi-datamodel surveys such as LFS, subject matter questionnaires are normally accessed via a component list.

The Exit questionnaire is used to wrap up an interview session. It provides interviewers with a means of assigning an outcome code to the survey and includes text used to thank respondents for their participation and to inform them of any potential future contacts they may have with Statistics Canada as a result of their participation.

## 4.3 Sample Files

Clients will prepare a sample file for each environment that serves as input to the case management and collection systems. The file for the CAPI environment consists of two parts, the first part containing information used solely by the CASEMAN case management system and the second part used solely by the survey collection application. The structure of the input to the CAPI collection system will be identical to that of the CATI collection system.

## 4.4 Output Files

The structure of the output files will be the same for both CAPI and CATI environments. Records from the CAPI environment will be returned in individual files per case, while the CATI environment will return multi-case files on a daily basis.

## 4.5 Reports

Being able to report on the progress of individual surveys is a key requirement. In the CAPI environment, collection reports are associated with the CASEMAN case management system and survey-level reports already exist.

In the CATI environment, standard reports are created from historical case event data that exists at the scheduled case level. It will be necessary, then, to ensure that historical case event information is kept at the individual survey level as well, so that the required reports can be produced. Keeping historical case event information for each questionnaire component will facilitate this.

## 5. Implementation plans, testing and other considerations

As this project has been foreseen for months, many steps towards implementation were initiated behind the scenes in preparation for the redesign. A repository of standard question blocks (used for all Social Surveys at Statistics Canada) has been created. The existence of this repository should reduce development time in the Contact, Household, and Exit components of the applications. A prototype of the household roster blocks has been developed, as well as a proof-of-concept prototype of the new multi-datamodel program manager. A generic browser (in Visual Basic) used by interviewers to select specific cases, as well as standard reports (in SAS) have been developed and tested. They will be implemented in production for several surveys before this redesign takes place. Having this preparation work done has allowed us to focus on the user-defined aspects.

In order to manage the project effectively, a detailed list of activities and tasks has been created. This schedule leads to an implementation date of July 2003 for the new CAPI environment, and January 2004 for the CATI implementation. High level requirements were gathered from the key stakeholders, and further discussions within the multi-disciplinary project team have lead to detailed specifications for all components. The programming and testing of key components is well underway. At each step team members ensure that documentation of the process, as well as the product, is not forgotten.

Prior to development, a specifications review process was established whereby a team comprised of members from the survey operations, subject matter area and application developers work together to develop specifications. This has resulted in superior specifications that have helped to expedite development.

To ensure a high quality product, development and testing at progressive levels is promoted for all collection applications and this project is no different. Once the initial authoring is complete, blocks of the collection application are passed to team members from the survey operations area as well as the other stakeholder divisions for initial testing. After at most three rounds of block level testing, the blocks are put together into questionnaire components incorporating the flow logic between the blocks. Next, the questionnaires are integrated together with a program manager and integrated testing of the entire application begins. Once the integrated application has been approved, end-to-end testing takes place in the survey operations area, where a mock-up of Regional Office environments allows the simulation of the actual interviewing environments. This final round of testing involves the input and output files generation, as well as simulating the transfer of data to and from the regional offices.

Another key section of the schedule is the tasks and timeframes allowed for development and training of Regional Office staff. Training and reference material for all levels of personnel will be prepared, and site visits from knowledgeable team members will help to clarify outstanding issues.

A capacity test is planned in one regional office well ahead of the production date, to ensure that there will be no issues with the volume of cases being processed on the Blaise servers, now that a larger volume of true CATI cases will be managed. To allow sufficient time for field tests and the subsequent evaluation, two adjacent months of field testing have been set aside, after which a decision will be made as to whether the project should proceed to phase-in the remaining Regional Offices one at a time over a period of several months.

## 6. Conclusion

For a project of such a grand scale, and far-reaching implications if not done well, a phased approach with step-by-step implementation is the best scenario. Some of the hardest potential defeats we have come across are a commitment in terms of personnel and monetary resources for the project from all stakeholders. The greatest challenge is to balance new development and production at the same time. Many of the key players are the same, and have conflicting priorities. This has caused delays in a schedule that seems appropriate. Regular team meetings, as well as a steering committee that has the authority to make decisions and re-orient priorities as required, are essential to the success of such a project.

The move towards a true CATI environment using the Blaise call scheduler for the CAPI/CATI surveys should provide a more efficient way of working in the call centers, as compared to the manual selection of cases that occurs at the moment in pseudo-CATI. One concern does remain with the client area and survey operations division: the first survey being converted, the Labour Force Survey, has a 7 day collection window each month. The notion of best time and day to call are very important to this survey, as the collection period is short and the sample size is large. We will be experimenting with new processes surrounding the setting of the call scheduler parameters, as well as simulating time slices during the day in order to try to maximize the number of calls to a respondent during their "best time." This may result in several daybatch updates during each day for the first several days of collection, and different procedures during the last few days of collection. Extensive simulations are being planned, to assure the developers and the clients that the new model, as well as the Blaise call scheduler, will be effective and perform as expected.



## *CATI and Organizational issues*

- **Blaise CATI at Mathematica** .....233  
*Leonard Hart & Linda Bandeh, Mathematica Policy  
Research, USA*
- **Cati Survey Management System** .....241  
*Leif Bochis, Statistics Denmark*
- **Interactive training for the interviewers** .....251  
*Hilde Degerdal, Statistics Norway*



# Blaise CATI at Mathematica

*Leonard Hart & Linda Bandeh, Mathematica Policy Research, USA*

## 1. Abstract

With more than 250 workstations at two locations, Mathematica Policy Research, Inc. (MPR) has one of North America's largest computer-assisted survey capabilities for primary data collection supporting public policy research. To insure that these survey operations run efficiently MPR has undertaken the development of reusable components for major aspects of the Blaise system. Three major parts of this overall capability are covered in this paper: (1) Blaise instrument shells for list, random digit dialing (RDD), and establishment surveys; (2) supervisor review utilities; and (3) overnight batch processes.

## 2. Introduction

Most survey organizations conduct surveys using some form of Computer Assisted Telephone Interviewing (CATI). CATI procedures, however, vary greatly from company to company, or even survey to survey within the same organization. Some organizations are able to use the built-in CATI features that come with their Computer Assisted Interviewing (CAI) package, while others have to build complex CATI systems to meet their needs. The approach an organization chooses depends on the level of management desired in managing a survey. In this paper, we describe how Mathematica Policy Research, Inc. (MPR) is adapting the standard Blaise suite CATI tools and how MPR has gone beyond these standard Blaise tools to do a CATI survey.

Blaise is a Commercial Off The Shelf (COTS) CAI package that can perform almost any survey needs an organization might need. Over the past couple of years, the survey community has seen great advances from Statistics Netherlands, they have added new features and extended the capabilities to the current Blaise suite. It is safe to say that no two survey organizations are the same; each has their own special needs. In the competitive contracting world of survey data collection, CATI management is the one capability that sets the various organizations apart.

For more than 20 years, MPR used a competing CAI package. During this time, the CATI management system at MPR developed into a highly complex management environment that met the needs of MPR clients. In the late 1990s, MPR began to see a change in clients' requests about how they wanted a survey managed. There were also changes in the technical environment and concerns about support and "product life" of the current CAI package. Clients started demanding a COTS package that would allow them to move data collection operations from one survey organization to another without having to do a major rewrite. After researching the different types of CAI packages on the market and after listening to client requests, MPR decided to move to Blaise in late 1999. Since that time, MPR invested in moving its CAI operations to the Blaise environment. MPR has completed four major surveys in Blaise. It has three major projects scheduled for early 2003, with more scheduled for later in the year.

Since MPR started using Blaise, we have learned much about the Blaise system, including some limitations. Blaise is a powerful COTS product that is hard to beat for easily getting a survey going with basic CATI management features. For some CATI management features, however, additional infrastructure is needed. This is

not due to a lack of support from Statistics Netherlands. As was mentioned, no two survey organizations manage a CATI survey exactly the same; each organization has its own specific needs. Because of this, it is not feasible for Statistics Netherlands to build a product that meets everyone's needs. To address this problem, Statistics Netherlands provides the user with a suite of tools that enable the company or organization to build, fairly easily, its own CATI management system.

This paper describes three key areas MPR felt additional features were needed to go beyond the basic CATI management facilities provided in Blaise in order to build a complex multimode CATI management system. These are: (1) Blaise instrument shells for list, RDD, and establishment surveys; (2) supervisory review utilities; and (3) overnight batch processing. One may think of these areas as the supporting legs that support CATI management. MPR calls the suite "MPR CAI Plus." This paper also touches briefly on a test environment that insures the quality and reliability of these pieces.

### **3. MPR Generic Blaise Instrument Shell**

After fielding four projects with Blaise, MPR decided that it was the appropriate time to reevaluate the features available in Blaise, as well as the list of features that were needed in order to fielding CATI surveys cost effectively. Although the Blaise software is robust and has many features our previous CAI software did not, as with any COTS software, it did not cover all the features MPR desired. There were still areas that needed to be fine-tuned to meet MPR's needs and management style. After gaining some experience with Blaise, and developing an early version of a generic shell, MPR decided to put off additional Blaise surveys until we had had a chance to review what we learned, to write and update requirements, and to further develop our own "front-end shell." For the purposes of this paper, the generic shell is defined as a set of programs and processes that manage cases, that enable supervisors to review problem cases, apply interim and final status codes, and produce monitoring reports such as appointment and daybatch analysis reports.

Besides the features MPR wanted to add or change in the Blaise system, there were several primary goals for the new shell. We wanted to:

- Develop a flexible outcome-coding scheme that allows projects to add and customize codes to specific needs, while maintaining a standard set of codes that can be processed consistently across projects.
- Create a standard set of variables to be used for reporting and analyzing call attempts.
- Develop an interactive method of allowing project staff and programmers to select outcome codes and set related parameters without changing source code or recompiling the program.
- Allow the flexibility to change parameters for cases that need special handling, or for experimental design.
- Give supervisors more tools with which to review problem cases and monitor samples.
- Improve the history file display and access for interviewers and supervisors.

- Provide an easy-to-use parameter-driven system.
- Reduce programming and implementation costs.

To accomplish this, MPR organized a group of people, with representatives from each of the five main survey groups. Called the “Front-End Working Group” (FEWG), this group consisted of Karen Cybulski, a Survey Researcher; Barbara Carlson, a Senior Sampling Statistician; Larry Snell, Senior Manager of the Princeton Operations Center; Barbara Kolln, a Systems Analyst; and Linda Bandeh, Assistant Director of System Development, the chair of the FEWG.

The FEWG met once a week for a period of about eight months. The group wrote requirements without consideration for any particular platform. In other words, the intention was to not limit requirements according to what was available in either Blaise or our current software; instead, the requirements based on those that MPR staff felt were important and necessary. As the requirements were written, the Blaise development team reviewed them and asked questions. The team consisted of Carlo Caci (Programmer), Leonard Hart (Systems Analyst), Doug Dougherty (Senior Systems Analyst), and Mark Pierzchala (Senior Analyst for Computer Assisted Interviewing Methodology). As a result of this review, some specifications were modified in order to accommodate the existing limitations of the software, while other specifications were put on hold until the next version of the shell. Some of the goals are listed below.

### **3.1 Develop a Flexible Coding Scheme**

The first requirement the FEWG tackled was to determine a flexible coding scheme for outcome variables. This requirement was, by far, the largest task; but, without it, many other tasks could not be started. To accomplish this, the group reviewed the status codes used on numerous projects over the past 10 years and borne out by the American Association of Opinion Research (AAPOR) scheme. First, the FEWG defined a structure of three-digit codes that would allow for expansion—for example, 200-299 would be used for refusal codes. Once this was done, the FEWG defined how this group of codes would be treated. Every code within this group would have a general meaning and a method of processing. For example, cases with refusal outcomes would be part of a refusal group controlled by number of refusal calls, days between refusal calls, and refusal letters, which would be assigned to refusal interviewers. This reduced the handling of specific codes and allowed new refusal codes to be added with minimal programming; for some processes, no additional programming would be required. The major outcome groups included completes, locating, refusals, contact calls (including appointments), no-contact calls, barriers, and phone problems. The most common outcomes were given a code that could not be changed; codes were grouped by type and function within the major groups and gaps were left between codes for projects to define any additional codes necessary.

Once the outcome codes had been defined, the FEWG examined each code and discussed how it should be processed, how and if the case should be delivered for the next and future calls, and how it should be reported.

### **3.2 Define a standard set of variables**

The FEWG also defined the related variables that would be needed to control or monitor the case. As an example, an eligibility code—denoting eligibility for inclusion in relevant study—was removed from the outcome codes and a separate set of variables developed. These variables can be customized for a project’s needs—thus allowing projects to define and track multiple levels of eligibility

without having to create an extensive list of outcome codes to cover all situations. For example, on an RDD study, there may be several levels of edibility, such as working phone, residence and household members over 18. Standardized reports are then developed that use the eligibility variables, along with outcome variables.

### **3.3 Develop an interactive method of selecting outcome codes and setting related parameters**

MPR wanted an easy way for survey staff and programmers to select outcome codes and options from this list in order which to customize the shell according to their project's needs. The Blaise development team designed an interactive process that allows users essentially to go through a series of screens and pick or change options. The screen was set up with the recommended default options, but users can select outcome codes and change such parameters as: the number of days to hold the case before calling back after a refusal; the number of refusals to allow; whether an outcome should be final status or sent to a supervisor for review; the number of times to call before leaving a message; the number of days to hold a case between leaving messages; the number of phone problems before sending a case to locating; and so on. These options are implemented as external Blaise files, one for status codes and one for other survey settings

A Blaise shell was developed that reads and is driven by these parameter files. The external parameter files accommodate experimental design. There can be an external parameter record for each treatment, thus allowing projects handle cases differently depending on their experimental design.

### **3.4 Implement a set of supervisory tools**

To define a set of tools and screens for supervisors, MPR reviewed the tools available in the Blaise system, requests from phone center supervisors, and feedback from other groups. MPR's system allows supervisors to review individual cases, groups of cases, final and interim status cases, history, all counters, dates, parameters, flags, and outcome related to the cases, as well as change those parameters, if necessary. (See figure 1 below.)

### **3.5 Maintain history files**

Supervisors and interviewers requested a more extensive history than that which is currently provided by the Blaise system. They wanted a record of every call to be available on a screen, which would allow them to scroll through calls and see the time, date, interviewer, outcome code, appointment information, and notes from interviewers.

Using the native history file would not accommodate the needs of the supervisor or interviewer. To overcome this problem, the FEWG, working with their users, decided which key information would be maintained in the data model as part of the shell. The variables were then added to a block called "history" which was then arrayed for up to 100 calls. Next, code was added to all possible exits to the main shell, that would update this array each time a person or process touched a case.

### **3.6 Maintain adequate documentation**

Writing and maintaining the requirements document was an important responsibility of the FEWG. Each member was responsible for representing his or her group's interests. During the requirement-gathering phase, several meetings were held. The staff were given copies of the requirements and invited to discuss them and ask questions. By having a representative group and conducting company-wide meetings, the group was able to ensure that all interests were

addressed. The requirements document became a crucial document for programming, testing, and documenting the new system.

After the review of the documentation by the Blaise development team, the requirement documentation was updated and sent back to the FEWG for review and comments. This process continued until FEWG had completed the requirements for version 1.0, and for several months into testing, to ensure that requirements had been interpreted correctly and that questions from the developers had been answered.

### **3.7 FEWG products**

The FEWG outlined the requirements for calling routines and developed a set of standard outcome codes, counters, flags, timing variables, and history requirements. They not only looked at the information needed to conduct the survey, but at the requirements for data collected for reporting, monitoring, and analysis. When the FEWG was first formed, the intention was to write requirements for List, RDD and Establishment surveys. The focus was on developing a flexible architecture that would allow for other types and modes of surveying, and to develop a source code that was independent of project-specific screening and contact blocks. It quickly became clear that this was too large a task. List surveys are the most common types of surveys for MPR and the first project that was targeted to use the next version of the shell was a list survey.

The shell was developed so that the contact section, the introduction and the screens are separate blocks from the rest of the standard code. This allows projects to customize the screener and introduction to their project needs, in addition to making it the starting point for RDD or Establishment shells. While there will be different modules for the three shells, they will share much of the same source code.

## **4. Supervisor Utilities**

Once the new version of the list shell was developed and was in the testing phase, the emphasis switched to developing supervisory tools. Blaise CATI management tools are probably the weakest link in the Blaise suite. The suite does come with a few easy-to-use utilities—for example, CATI Specifications, CATI Management, and CATI History Viewer or built in features like Time Slices, Time Zones and Groups. These, however, are insufficient for most survey organizations. As mentioned in the introduction above, every survey organization will have its own special need for tools with which to manage a survey, and it would be impossible for Statistics Netherlands to provide a single product that would accomplish all these tasks.

Blaise suite does provide other tools with which to write individual, in-house applications. In the past, one could use Manipula or Maniplus or a dll link into the DEP or Manipula/Maniplus through a Delphi hook provide by Statistics Netherlands. But, in the past few years, Statistics Netherlands has put great effort into developing the Blaise Component Pack (BCP). The BCP, a collection of several COM components written specifically for Blaise, can be used in applications or programming languages, so that the organizations can create the user's own customized tools for integration into the Blaise CATI management.

MPR Supervisor Utilities are part of a package called “MPR CAI Plus.” MPR CAI Plus is built on the philosophy of extending the current Blaise suite of utilities to meet the needs of MPR and its clients. MPR CAI Plus is a generic system built by using Visual Basic, Manipula, Maniplus, Blaise suite programs, and third-party

applications. This package provides users with an easy-to-use menu system, control over granting access to certain tools, and various utilities for managing a multimode survey.

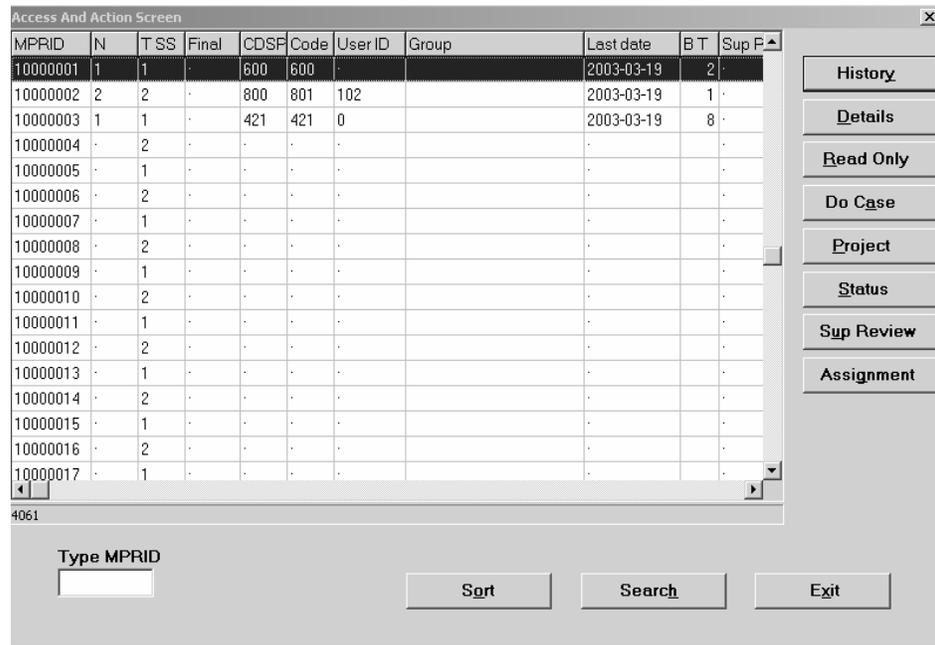
The first part of the Supervisor Utilities is the MPR CAI Plus menu system. This is a generic, easy-to-use menu system built by MPR with built-in survey management tools. A key feature of this package is the ability to set up a new survey and grant Supervisor or administrative privileges to certain utilities. As long as key survey files reside in the right place, a person can have a new survey setup and be ready to go within a few minutes at one of the call centers. In addition to these features, there will be reports that are ready to run, as well as case management tools for reviewing case progress.

The other part of the Supervisor Utilities consists of the generic Maniplus applications. These standard Maniplus programs are written to use the standard shell, with only a few minor changes needed for the Maniplus code; basically, for any survey, changing a few uses sections, with input or output statements pointing to the right folder structure, and these programs will fit right in with the MPR CAI Plus menu system

Figure 1 is a screen shot of one of the many Supervisor Review programs called “Access and Action Screen” that are part of the MPR CAI Plus system. For this particular Maniplus program, a CATI Supervisor has control over a case or a series of cases, review case information, set flags; it can even override certain flags. From here, it can sort, search, or go retrieve a particular ID. Also, once an ID has been selected, it can change a case, or “re-status” it, see predefined case details, or see the case history on a user-friendlier screen.

Several other prepackaged Maniplus programs round out MPR CAI Plus—for instance, a better Daybatch viewer, or a detailed appointment viewer or mark cases for review. This series of Maniplus programs, which come as part of the generic MPR CAI Plus suite, can, with little modification, be used for any survey. All the survey programmer needs to do is make a slight change, if needed, to the Maniplus program, re-prepare, test the finished MSU file in the test environment, then move the new MSU file into the live production. The MPR CAI Plus menu will automatically add it to the supervisor menu.

**Figure 4 - Access and Action Screen**



## 5. Overnight process

No CATI system would be complete without an overnight process. The user might want to process cases each night—assign treatment groups to cases, re-status cases, reset missed appointments, apply new rules, run reports, back up key data files, do maintenance of files or update outside data sets, and so on.

Like the other two pieces of MPR CATI management mentioned above, the overnight process was built on the foundation of a standard Blaise shell and Supervisor Utilities, to work efficiently. Like the Supervisor Utilities, the overnight process is a highly generic system that can be up and running within a few minutes. This generic process backs up key files multiple times throughout the process. It can tie into the Survey Management System (SMS) built with Microsoft SQL Server, run maintenance on files, and create/update files that are used by the Supervisor Utilities.

Currently, MPR feels that data manipulation can better be done as an overnight process that does not affect live calling. Over time, we feel that applications will become faster, hardware will get faster and cheaper, and software applications to do perform these tasks as real-time process will all become cheaper and better. MPR will continue to look at the options to do perform these tasks in real time as they become available.

## 6. Testing environment

As MPR moved from its old CAI package to Blaise, MPR realized that a testing environment closely resembling the live environment was a very important part of the move to Blaise. This test environment played an important role in building a generic CATI system. This environment became a proving ground for determining whether various features would work.

MPR sees this testing as an important part of CATI management. We feel that every effort should be made to replicate the production environment for this testing. Local machine testing is not adequate since there are so many variables that can affect how things will operate: different servers, the operating system, the wrong version of a dll, a network switch, and so forth. The purpose is to minimize the changes in moving from the test to the production environment.

MPR has tried to overcome this problem by replicating its test environment using the exact hardware, software, switches, and client machines. The test environment uses the same equipment and software as does production; it just maps to a different network folder. By doing this, we have caught things that went unnoticed when testing on local machines. We have also found differences in equipment and software that we might not have found until we moved into production. A good test environment is well worth its cost.

## **7. Summary**

Over the past two years, MPR has invested heavily in laying down a sound foundation for a new CATI system. Many people have been involved, with many hours devoted to writing specifications and coding generic systems, investing in new equipment and software, and retraining people at all levels of the company. We feel that we are beginning to see this investment pay off. We can now get a new Blaise survey into production in very little time, and people have the utilities to manage their CATI surveys. We continue to look at new ways to improve our Blaise components and to integrate them into our outside applications. While no Commercial Off The Shelf software can meet every need, Blaise does provide the tools with which to adapt Blaise CATI to individual needs, just as MPR has done.

# Cati Survey Management System

*Leif Bochis, Statistics Denmark*

## 1. Introduction

From the beginning of 2000 Statistics Denmark upgraded all CATI surveys to Blaise 4 Windows. At the same time emerged the need for a general Survey Management System that could make deployment of CATI surveys easy and fast and possible to carry out for non-technical staff and with a high degree of automation.

The first survey converted to Blaise 4 Windows was the Labour Force Survey with a large number of interviews and carried out on a weekly basis. The LFS is characterized by a large number of standard activities in the survey life time and the solution to LFS was a tailored management system integrating all tasks from importing the data for the interview process through interviewing and post editing (coding etc.) to the export of edited data.

Later, for the rest of the CATI surveys, Statistics Denmark has developed a generalized Survey Management System, in order to carry out similar tasks for the larger number of more heterogenous surveys.

Common for the two systems are:

- Standardization of folder structures and processes
- Automatic generation of standard procedures for import of telephone numbers, addresses etc.
- Automatic export of collected data into SAS
- Almost entirely written in Maniplus

This paper will describe the systems and discuss some success criteriae for evaluation of the systems.

## 2. History

Statistics Denmark has carried out the LFS and Omnibus surveys using Blaise CATI since 1991/1992. A number of tasks were automated with a combination af the available tools in Blaise 2.x and Dos batch files, however, a real generalized Survey Management System hasn't been developed before the advent of Maniplus made this easier to accomplish.

1997	Pilot studies in Blaise III/Maniplus
1998-1999	First experiences with Blaise III Maniplus Survey Management on the Immigrant survey
Autumn 1999	Pilot studies in LFS and studies in the needs of a SMS
Jan. 2000	LFS Survey in Blaise 4W launched
March 2000	Upgrade to the Ansi version of B4W (Blaise 4.3)
July/August 2000	Conversion of the Holiday Survey (quarterly) and the Omnibus Survey (monthly) into B4W – Administration carried out with a slightly modified version of the Immigrant Survey Management System – The SMS was extended with a Blaise database used to administration of the menu in order to ease inclusion of new surveys and removal of old ones

Until 2001	"Maturing" process of the LFS management system
Spring 2001	Development of a Generalized Survey Management System studying the needs of the running Cati surveys (apart from LFS)
August 2001	Omnibus Survey (monthly) moved to the new generalized SMS
Autumn 2001	The rest of the surveys were moved while maturing the SMS processes
Jan. 2002	Transport survey (monthly) – the last survey to be moved from Blaise 2.5 to B4W
2002	Further improvements
July 2002	Addition of a simple tabulation tool

### 3. The LFS experience

Characteristical for the LFS is a high degree of standardization in a large number of tasks:

Each week a new sample is drawn, telephone numbers collected and interviewing carried out, after interviewing data are automatically coded and passed to the post editing system. The need was a large number of fixed procedures aiding the integration between interviewing, automatic coding, post editing and export procedures. This system is supplied with a number of standard reports to help administering the survey.

The LFS itself uses a very complex questionnaire which requires specialized Blaise expertise in the development. In order to reduce the need for programming work – and reserve it for the development of the datamodel – procedures were developed that carried out automatic generation and preparation of utility programs, e.g. Interview to post editing conversion plus a number of other conversion and reporting programs.

This automation was helped by the routineful execution of tasks, that made it possible together with a strict naming standard for datamodels, databases and setups to calculate a number of parameters from the system date: What is the path and file names of the actually needed datamodels and databases, and what is the most probable action to carry out on these data on this day.

### 4. The need of a Generalized Survey Management System

In contrast to the LFS the need for the rest of the surveys was flexibility in setting up surveys and a much smaller number of utility functions – while the LFS is using a very complex questionnaire which requires a high level of Blaise expertise in the development, most of the other surveys are simpler and less standardized with respect to e.g. interviewing periods

And while the LFS needs a system that integrate interviewing, post editing and management of both, few of the other surveys requires post editing at all.

The need emerged for the development of a Generalized SMS that should provide clerks with sufficient tools to set up and manage a variety of surveys

Aims and purpose of the development may be described as:

- To automate as many tasks as possible
- To make deployment of CATI surveys easy and fast

- To make it possible for non-technical staff to carry out (almost) all work in setting up surveys

By standardization of the processes it should thus make the management of surveys faster, easier, better and more effective.

## 5. Standardization process

The process of standardization comprised a number of standardizations:

- Standardization of file and folder structures
- Standardization of datamodels by use of templates
- Standardization of background data
- Automation of standard procedures
- Standardization of user roles
- Other standardization efforts

### 5.1 Standardization of file and folder structures

On the basis of the recommendations in the Developers' Guide a folder structure was defined:

```
CATIROOT          - DATA          - SurveyNameX
                  -                 - SurveyNameY
                  -                 - SurveyNameZ
                  - DATAINIT
                  - INST
                  - MANI          - INIT
```

The individual subfolders under the DATA folder contains (of course) the data files belonging to each survey plus a number of reports, e.g. DAY files from Manipula etc. In the DATAINIT folder templates like a general CATI Survey Definition file is placed and from where it is automatically copied to the survey data folder when needed. The INST folder contains all prepared survey datamodels and the MANI folder contains all prepared Manipula setups. The MANI\INIT subfolder contains templates for standard Manipula setups.

This folder structure is combined with a naming standard for Datamodels and Manipula setups.

### 5.2 Standardization of datamodels by use of templates

A basic template for surveys was defined:

```
DATAMODEL MySurvey

INCLUDE "Standard_NonResponse_Treatment.INC"
INCLUDE "Standard_Appointment_Treatment.INC"
INCLUDE "Standard_BackgroundInfo_Treatment.INC"

INCLUDE "MySurvey_Questions.INC"

(...)

ENDMODEL
```

The developer of the survey should only care about the contents of the block defined in the file 'MySurvey\_Questions.INC', i.e. the questionnaire itself. All the administrative tasks are taken care of in a standardized way. Of course it is possible to change the Non-Response Treatment and supply the Background Info if necessary, but the standard blocks defined was developed in order to support most surveys.

### 5.3 Standardization of background data

A standard datamodel describing input data including telephone number, address and names of the household members etc. was defined and made up the basis of templates for initialization manipula setups. An initialization Manipula Setup could be described in a template:

```

SETTINGS
    { ... various settings defined ... }
USES
    InputMeta

    { Variable part of Setup: }

        OutputMeta 'MySurvey'

    { ----- End of variable part of Setup }

INPUTFILE inp : InputMeta ('', ASCII)

OUTPUTFILE outp : OutputMeta ('', BLAISE)

{ ... rest of the setup,
  e.g. checking correctness of the input-data ... }

```

It turned out – not surprisingly – that it was not possible to use the same datamodels to represent input data for all surveys. The standard solution, however, could easily be supplemented by support for variant datamodels and variant initialization Manipula setups.

### 5.4 Automation of standard procedures

With the template defined above and a strict naming standard for Datamodels as well as Manipula setups it became also possible to generate the actually needed Manipula setups automatically on request.

Example:

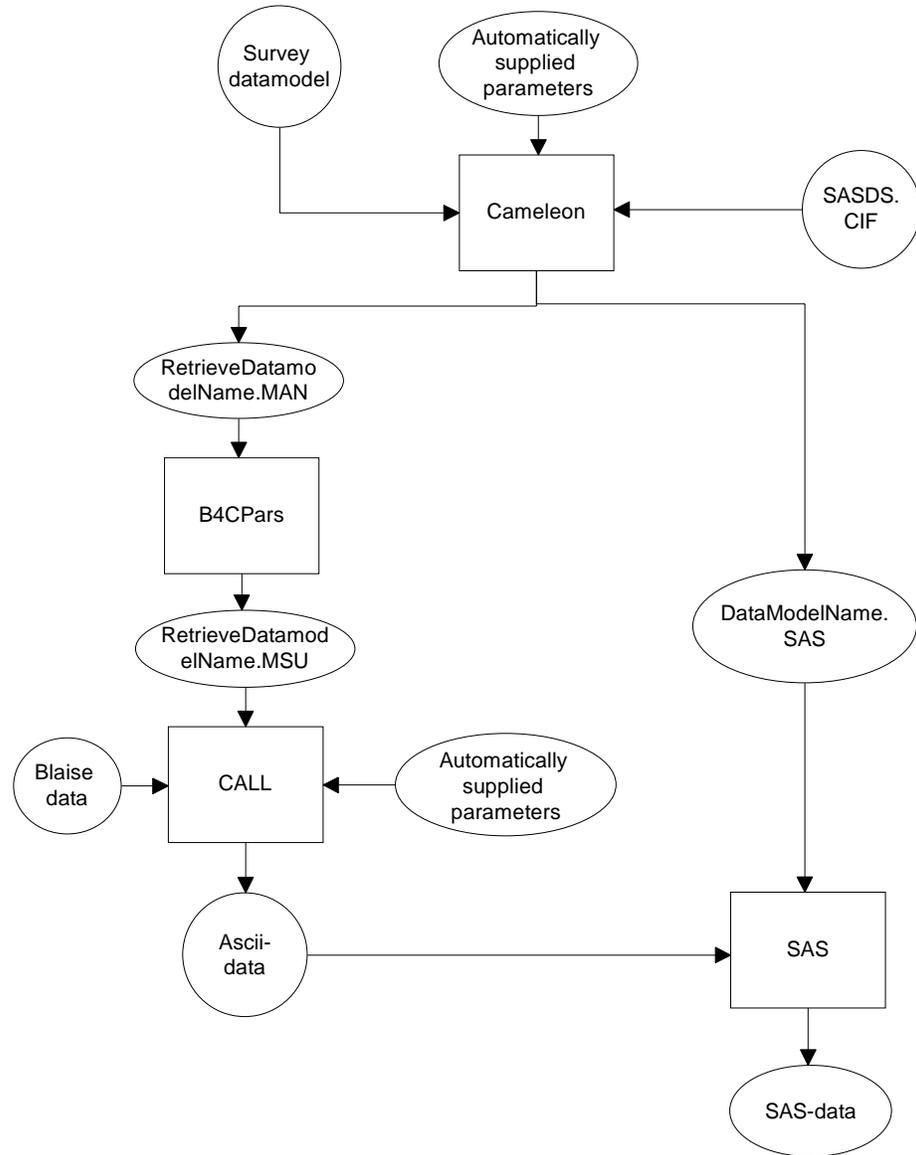
To a datamodel defined in the Source file 'MySurvey.BLA' corresponds an Initialization Setup 'InitMySurvey.MAN' which is produced automatically the first time it is needed – simply by concatenation of:

- the first few lines of the template above
- a generated line: OutputMeta 'MySurvey'
- the rest of the template above

giving the new file the name 'InitMySurvey.MAN' and preparing it with the B4CPars utility.

This model is used for initialization of data as well as export of data. And among the benefits of this part of the standardization are reduction of errors by the automation of program generation while it the same time makes it possible for non-programmers to carry out the proper procedures at any time after the datamodel is finished.

Figure 5.1: Automatic Export of data.



### 5.5 Standardization of user roles

To simplify the tasks needed four different standard user roles were defined:

Role	Usual tasks
Interviewer	Interviewing
Supervisor	Interviewer + Access to Cati-management
Researcher	Testing, a range of listings, analysis and retrieval procedures, access to archive
Administrator	All, including setting up new surveys and updating telephone numbers etc.

The definition of these roles made it possible automatically to set up a personalized user interface though all of the users share the same entry into the Survey Management System: Interviewer administrators maintain a Blaise database with

the usernames of all roles (except interviewers), i.e. defined roles as Administrators, Supervisors and Reserarchers. All other users with access to the Survey server will be treated as Interviewers, i.e. users with access to a limited set of functions.

### **5.6 Other standardization efforts**

The SMS should enforce documentation of processes (what has happened when and by whom to which database). The logging mechanism of the CATI subsystem carries out a great deal of this documentation automatically. By providing all Manipula templates with settings defining dayfiles the rest of the need for documentation of production processes was achieved.

Whenever a datamodel is defined, deployment should be a matter of a few point and click operations, so the SMS should be able to retrieve all necessary information and provide the Cati administrators with the proper functionality.

Tailored Cameleon scripts were developed helping fully automatized export to SAS. By defining a set of standard parameters to our localized SAS Cameleon script we were able to provide fully automatic SAS export ruled by the knowledge managed by the SMS.

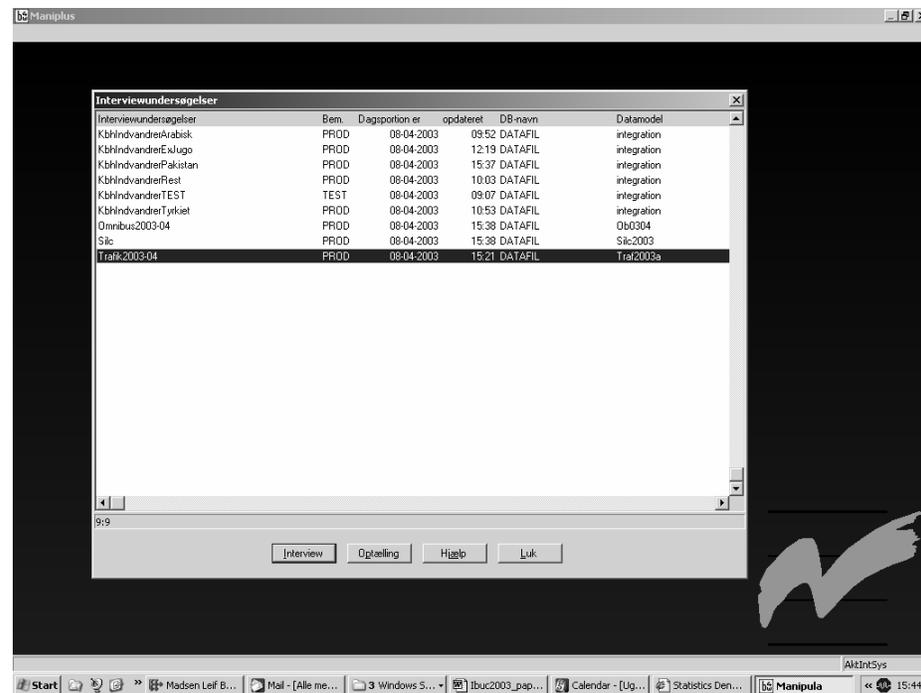
## **6. The Generalized Survey Management System**

The core of the system is a Maniplus procedure that scans the folders for installed surveys and presents a list of available surveys and a set of functions for the user while taking into account the role of the user and the status of surveys.

Surveys are defined by the existence of a subfolder naming the survey or survey portion. Active surveys are defined by the existence of a Blaise database and a recently updated daybatch file plus the non-existence of a "SystemStop" flag.

The interviewer role needs a list of active surveys and a few functions, namely "Start Interviewing", "View personal results" and "Exit". The generalized SMS thus detects all active surveys and presents a list of the surveys together with the proper set of functions that should be provided for interviewers.

Figure 6.1: The Interviewer Selection Screen

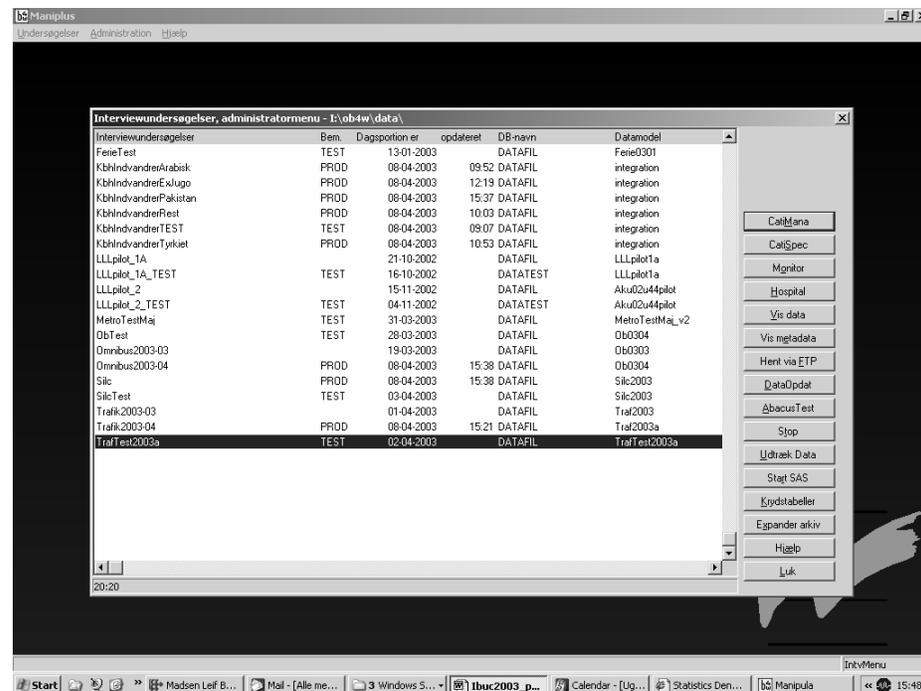


The supervisor role needs an extended list of surveys (to be able to activate an inactive survey), so the list presented to supervisors comprise all the installed surveys. Additional functions provided for the supervisors include “Cati Management”, “History”, “Monitor” and “Hospital”.

The researcher role is granted access to “Dataview”, “Metaview”, “Export data to SAS”, “Start SAS” and “Simple Cross Tabulation” functions. Researchers also have access to archived surveys, i.e. surveys that are removed from the interview system but still kept in the environment.

The administrator role has access to all the functions mentioned above plus functions for import of telephone numbers, Cati specification (Setup survey) and activation/deactivation of a survey.

Figure 6.2: The Administrator Selection Screen



The generalized SMS is now an approx. 1300 lines long Maniplus program (including comments) with a modular design which makes it easy to amend new functionality, or even new roles if necessary.

Almost all functionality is written in Maniplus. The exceptions are two Delphi functions essential to the system: A filedate routine (stolen from the samples collection) and a routine that scans a .BFI-file and retrieves the datamodel name.

## 6.2 Future developments

With the new features of Blaise 4.6 (e.g. auxfields sections in procedures, external activeX procedures) modularity can be improved and it will be easier to integrate functionality made possible by use of BCP. The core will still be kept as an easy-to-maintain Maniplus program.

## 7. Conclusions

With the generalized SMS it has definitely become easier to deploy surveys. The procedures for the deployment of a survey now consists of the following steps:

- 1) Development of a datamodel. This is definitely the least automated process, but a few templates supports the very few people that is involved in this process and makes it possible for non-programmers to define datamodels relatively easy
- 2) Copying the datamodel to the folder that contains all instruments
- 3) Ordering a sample for the survey from the sampling unit
- 4) Point and click through the rest of the processes, including automatic generation of the necessary Manipula Setups

Deployment of surveys this way have been made independent of assistance from IT staff. Some 120 surveys have been deployed 2001-2003, most of the surveys have solely been carried out by office workers from authoring of the Blaise datamodel

on the basis of specs from the researchers to the delivery of data in a SAS dataset. Only a few more complicated questionnaires required Blaise expertise on a higher level.

The Blaise Support at Statistics Denmark (IT staff) is not involved in daily work in the Cati section (and actually didn't know about the number of surveys, before writing this paper...). Thus our knowledge of current surveys has decreased significantly, as Blaise Support is only involved in the maintenance of a few very complex questionnaires. This clearly states that the goal has been achieved.



## Interactive training for the interviewers

*Hilde Degerdal, Statistics Norway*

In Statistics Norway we started our first distributed CAI-operation in 1995. By then we trained our old interviewer staff to use laptops for their work. Quite many of them had low skills, if some at all, in using computers. We did worry about the training, but it turned out to be successful. Most of the inexperienced computer-users found it very interesting and useful to learn to use a computer. They appreciated to get this kind of knowledge, which was more and more a common knowledge in the society. And they found it very useful in their work as interviewers. As I have told on earlier conferences we gave them a laptop with full Windows environment and they also had the possibility to send and receive e-mails.

One of the main reasons to the success was the fact that we divided the training in sessions.

They stayed at a hotel twice in groups of 10-20 participants for 3 days. We offered two teachers and two teacher assistants. In the period between the two sessions they had 2 weeks at home, getting familiar with the equipment and they also had some tasks to do. After this period they went back to a hotel again and had some other 3 days of training. Afterwards it was also some tasks to do.

This two-sessions model was later adopted by the ordinary training of new interviewers. So after 1995 all new interviewers have been trained in using computers, Windows, e-mail and so on, mixed up with interviewer specific themes, all the time with a period at home between two residential courses. This way of training has been very useful for those who had low skills in using computers.

The part of inexperienced computer users has decreased dramatically during the years from 1995 to 2003. Norway is a country with quite big distances, and the traveling costs for two meetings does cost a lot. Early in 2002 we decided that we couldn't go on with this. The most of the participants doesn't need this "Getting Familiar" – period anymore. Spring 2002 we quitted the two sessions model, due to the costs. We started up with a training model with one residential course of 4 days. We quitted the computer teaching and expected that the interviewers had the needed skill to handle the laptop in their work. When they applied for the interviewer job it is a question in the form on what they themselves think about their skill of computer knowledge. We know from experience that this information does not give a correct picture. So we developed a computer skill questionnaire (using Blaise of course) in which we asked 27 questions about use of computers. On this bases we decide if the applicant would gain of some pre training on the computer. Due to travel costs this one-day pre-training is kept the day before the residential course. Until now just two participants have needed this extra training before two of the courses.

In fact the reduction from two to one training session didn't meant a big reduction in number of hours available for the training. We use four whole days. But to get through all the contents in so short period of time, makes the training very compressed. Of course this reduce the quality of the training. We started to evaluate alternative ways of training to reduce the loss caused by the compression. It needed to be done from their home. Not all the participants have access to a computer before they got their laptop. So if the training should include use of computer it had to be after the residential course. So what we had to do was to make some exercises for the purpose of repetition after the course. There were

already some homework after the course, consisting of interviewing of other interviewers, people in the office and so on. We made up a list of the subjects we want to repeat, some computer tasks and some interviewer topics. We want to make something more motivating than just reading some pages, something requiring action from the user, and also giving us some results back. We want the possibility to check that they really go through the repetition session, and perhaps also with which level of difficulty.

What tool could offer this functionality? What we want was much the same, as we want from a questionnaire, so why not use Blaise? It has also the advantage of being a known environment for the users.

So what we did was to make a questionnaire with sound and pictures picked from the themes we want to repeat. For the time being it has been within following subjects

- First contact with the respondent
- How to meet different ways of answering
- Use of Outlook Express and Internet Explorer - the Interviewers Homepage
- How to look at/correct your wage demand
- Specific questions in the Labour Force Survey

**Examples:**

**Example 1: The first contact with a respondent on the Labour Force Survey**

<b>FIELDS</b>	
<b>Kont1</b>	<i>"You call a respondent on the Labour Force Survey You get hold of the respondent, introduce yourself and say that you are calling from Statistics Norway in occasion of the survey in which he is chosen to take part @F Press &lt;Enter&gt; to hear the respondents answer. @F" : STRING[1], EMPTY</i>
<b>Kont1Lyd</b>	<i>"@GYou are calling a respondent on the Labour Force Survey You get hold of the respondent, introduce yourself and say that you are calling from Statistics Norway in occasion of the survey in which he is chosen to take part. @G How would you respond on this?" "^Lyd" : (Slippe "No, of course you may be let off. It is voluntary", Maalkke "No,it is voluntary, but we want you take part, otherwise we lose the information about all persons in the group of which you are chosen to represent.", Maa "Yes, in fact persons are obliged to take part in this survey.")</i>
<b>Oppf1</b>	<i>"^Oppftxt Press &lt;Enter&gt;" : STRING[1], EMPTY</i>
<b>RULES</b>	<i>Kont1 Lyd := 'SOUND(Sound/Kont1.wav)' Kont1Lyd IF Kont1Lyd = Maa THEN OppfTtxt := '@KCorrect@K/ LFS is in fact the only interview survey where a person is obliged to take part'</i>

	<p>ELSE</p> <p>OppfTxt := '@LError@L,@/LFS is in fact the only interview survey where a person is obliged to take part'</p> <p>ENDIF</p> <p>Oppf1</p>
<b>Kont1.wav</b>	"Oh I don't know. I don't take part in all the surveys going on. They are voluntary, so I don't have to take part. Do I?"

**Example 2: How to meet different ways of answering – "in the middle"**

<b>FIELDS</b>	
<b>Spm2_Innl</b>	<p>Imagine a scale from 1 to 10, where 1 means never and 10 means always. At what point would you place yourself concerning when to justify the following action:</p> <p>Avoid to report a damage that you have involuntary caused on a parked car</p> <p>Press &lt;Enter&gt; to listen to respondent's answer"</p> <p>: STRING[1], empty</p>
<b>Spm2</b>	<p>Imagine a scale from 1 to 10, where 1 means never and 10 means always. At what point would you place yourself concerning when to justify the following action:</p> <p>Avoid to report a damage that you have involuntary caused on a parked car</p> <p>@F If you can't manage to pick out an answer, you may press Enter!@F"</p> <p>"^Lyd"</p> <p>: 1..10, EMPTY</p>
<b>Oppf2b</b>	<p>"Well certainly this is not easy to decide. What would you do.."</p> <p>: (Spm "Repeat the whole question",</p> <p>Svar "Repeat the scale, for example.: Yes, would you say 2,3,4,5,6 7 ,8 or 9",</p> <p>Tall "Ask for what number she would choose",</p> <p>Sint "Ask the respondent if she is stupid - you need an accurate answer")</p>
<b>Oppf2</b>	<p>"^OppfTxt</p> <p>Press &lt;Enter&gt;"</p> <p>: STRING[1], EMPTY</p>
<b>RULES</b>	
	<p>Spm2_Innl</p> <p>lyd := 'SOUND(sound/midt.wav)'</p> <p>Spm2</p> <p>IF Spm2=EMPTY THEN</p> <p>Oppf2b</p> <p>ENDIF</p> <p>IF (Spm2 &gt; 4) and (spm2 &lt; 7) THEN OppfTxt :=</p> <p>'Well, actually it won't be quite correct. Won't both 5 and 6 both be in the middle?'</p> <p>ELSEIF spm2 = DONTKNOW THEN OppfTxt:=</p> <p>'It was perhaps to stop some quickly!'</p> <p>ELSEIF spm2 = REFUSAL THEN OppfTxt:=</p> <p>'Would not exactly mean that she refused to answer.'</p> <p>ELSEIF Oppf2b = Spm THEN OppfTxt :=</p> <p>'Yes perhaps, but wouldn't it be better to ask for an accurate number"</p>

	<pre> ELSEIF Oppf2b = Svar THEN OppfTxt :=   'Yes certainly it will be. Then let us hope that she'll understand that you   need   an answer' ELSEIF Oppf2b = Sint THEN OppfTxt :=   'Does not think this was any good idea. Wouldn't it be better to ask for an   accurate number' ENDIF </pre>
<i>midt.wav</i>	<i>"Oh. In the middle, I guess"</i>

By now, we have used these interactive training for repetition after four courses, and the users have told they find this way or repeating of the subjects very useful.

We have also started to think of using this technique for repeating subjects and for further training of experienced interviewers. Some subjects, for instance, how to reduce non-response, could be made as a small course with some cases, and send to interviewers that have been in the field for a while. As a test of the interviewers knowledge it will be useful to let them return the answers in the questionnaire, and perhaps also to take advantage of the audit trail feature.

For some surveys, which require special knowledge we have used different methods of training, in addition to the obligatory instruction paper. Sometimes we have made a videocassette with some examples. Sometimes when it is a very large survey we have arranged residential courses. We now discuss to use the interactive training technique to make the adequate training as a sort of briefing before the survey goes in the field.

This spring Statistic Norway bought a dedicated tool for making interactive courses, Rapid Builder. The interview training is being picked out to be a test project using this tool. So I have now started the work of making this repeat part of the training in Rapid Builder to find out if it will be better to make these features in this tool. Of course we lose the advantage of using the well-known Blaise environment.

We have also discussed to use the technique for helpdesk purposes. For more technical issues Rapid Builder, with the possibility to record the cursor's movement on the screen will be useful.

Sending Blaise questionnaires together with pictures, sounds, and movies to the interviewers using the ordinary lines will be impossible. Our communication is based on ISDN and the communication will time out before all the large files are transferred. So we have to use CD's for this purpose.

As a conclusion, it is clear that we are just in the initial period of using this training method. We hope to develop it much further. It is a cheap, and we think, efficient way of training. It gives new possibilities and is much more motivating than just to read some more instructions. To be very optimistic, we can imagine a possibility just to send the interviewer a laptop with a complete self-study program when they are hired as interviewers. This will mean that we could avoid periods of areas lacking interviewers. On the other hand it is a lot of subjects that is more suitable for face-to-face training. Another matter to concern is the social aspect. To be interviewer out in the field is a lonely employment, with few possibilities to have contact with colleagues. The residential courses are essential for the purpose of making contacts between the interviewers and also to improve the contact between the interviewer and the office. So more realistic is it to think about the interactive training as an efficient complement to the residential courses.

## *Paper Forms and Blaise*

- **Blaise for Post-collection Data Editing** .....255  
*Pavle Kozjek, Statistical Office of the Republic of Slovenia (SORS)*
- **The melting pot - practical experience of scanning paper forms** .....263  
*Marianne Troelsen & Lars Pedersen, Statistics Denmark*
- **Lessons learned on the development of the Unified Enterprise Survey** .....265  
*Rock Lemay, Statistics Canada*



# Blaise for Post-collection Data Editing

## *Building general data editing system based on Blaise*

*Pavle Kozjek, Statistical Office of the Republic of Slovenia (SORS)*

### 1. Introduction

The Blaise system is mostly and traditionally used to support surveys, where data collection and editing are integrated in the process called computer assisted interviewing (CAI). Since the system is also able to support other modes of survey processing, SORS is trying to extend its usage to support administrative data editing in general. It means that data editing system based on Blaise should be ready to receive and process any data coming from the input, regardless of the data collection mode.

Compared to CAI surveys, processing of data collected on paper forms is usually organized in a different way. A typical process begins with high speed data entry as the first phase, and data editing (reviewing-checking and correction) as the second phase. This kind of 'separate' data editing requires a different approach when preparing survey applications. Some characteristics:

- more batch processes
- different administration and user interfaces
- sources of additional information for data editing not always available
- different (data editing) screen layouts
- needs for (many) generated reports, etc

In the past, different ad-hoc data editing applications were developed at SORS to cover the needs of individual surveys. But for continuous and efficient processing of a large number of different surveys a standard and generalized solution is necessary, with some of the key requirements:

- automation, efficiency and reliability
- integration into complete statistical process
- acceptability for survey methodologists, application developers and application users
- complete LAN infrastructure support (system and user administration, archiving etc)
- openness for upgrading
- long-term support

During the development of the new data editing system we tried to use best practices to fulfil these requirements as much as possible. Knowledge about existing data editing methods and tools was used as a basis for the new solutions.

### 2. Background and starting points

Statistical Office of the Republic of Slovenia (SORS) has been using Blaise for data entry and editing of CAI surveys for almost ten years. Since 2001 Blaise also supports all high speed data entry from paper forms, making use of GEntry, in-house developed generator of data entry applications, and using Blaise also for

editing of these data seems to be a natural next step in the process: at last (but not least) we don't need to bring another software (and license) into the office.

Data editing at SORS was traditionally supported by different tools and applications (Cobol, Pl-1, Godar) running on a mainframe platform. Many of them are still in production, organized as circulating processes, with printed error messages and corrections re-entered in each cycle. Other solutions (Godar) are more efficient, based on relational technology (Rapid), combining batch and interactive work. But there's a big problem about their support in the future.

The new system is bringing many changes to survey editing process: developers and users are migrating to a new platform (LAN with Windows XP as the operating system) and they are using the new software. And often, the survey contents and methodology are using the 'opportunity' to change when migrating to a new environment.

Our previous experience with GEntry was good and helpful from many aspects, and Blaise with VB interfaces was proven as a good combination to build systems, robust and easy to maintain. We learned how to:

- build efficient systems on LAN
- enforce standard solutions
- design simple and friendly user interfaces
- educate users
- organize maintenance of applications etc

But the data editing process is more complex: it requires much more interaction with 'environment': address registers, classifications, other reference files. There are needs for different formats of input and output data and metadata, there's a large variety of complexity of applications, needs of different users etc. There were also some doubts at the office: is Blaise, known as a CAI tool, capable to support large traditional 'paper form' surveys?

### **3. Preconditions for development of a new data editing system**

The idea about the data editing system was to define it as a relatively independent module which can receive data (and as much as possible metadata) in Blaise or different ASCII formats, and produce clean (at administrative level) Blaise or ASCII data with basic technical metadata for further processing

Training of new developers and users was one of the first issues. For new and inexperienced developers of Blaise applications it was crucial to have standardized and user friendly environment for developing and testing applications. And for users of applications we need to prepare clear and easy access to all functions necessary for operational work – data editing. User interfaces - shell around both environments - were built by Visual Basic. Similar to Gentry, templates and automated generation of code were used wherever possible, to standardize work in development and production environment.

One of the often-asked questions was: can Blaise obtain all data editing functionalities of the old mainframe system? If something already used suddenly becomes unavailable, the users won't be very satisfied. The mainframe Godar system, based on Rapid RDBMS was a kind of reference, with the relatively automated covering of complete survey data editing process. Needs were identified, and some additional solutions were developed and implemented:

- a survey administration system was developed, supporting user access to editing system. Access is based on parameters (password, survey code and period...) and user rights (Windows XP/NT user groups).
- each error in the survey editing application is numbered with its own error code, to optimize searching and to report data correctness
- set of standard fields was added to each editing data model:
  - array of Boolean values to handle errors
  - alternative key (statistical ID)
  - some status fields about survey process and contents
- a number of standard batch templates were prepared: ASCII to Blaise and vice-versa, restructuring data (record to form and vice-versa), comparing between data models, importing and exporting data etc.

With these additions we were ready to prepare and test applications in the new system. Developer's and user's environment were defined as separate shares on the data editing server, and survey applications were located in sub-folders.

#### 4. The system - developer's perspective

As mentioned before, most of developers had no previous experience with Blaise or development on a network environment. So training was necessary as the first step. A short developer's manual was prepared, and three-day in-house course for developers was organized in February 2002. Contents of the manual and the course were limited to the needs of data editing, and practical work on examples was emphasized. It would be wise to continue with the real work immediately, but it took another six months before the new data editing system became operational.

It was not surprising that migration to a new platform was the main problem for most of application developers. Blaise itself was well accepted: syntax is not much different from other programming languages, and some key features of Blaise (modularity, re-use of code etc.) were immediately visible in practice. It is very encouraging for developer when the extent of application code is reduced up to ten times... Templates (for data models and Manipula setups) were designed to guide developers and to support some in-house conventions about developing applications.

To help beginners on a LAN environment to concentrate on applications, a developer's user interface was prepared. It supports:

- direct access to development environment, based on parameters (code of survey, developer etc.)
- copying all templates and standard setups (modelib, depmenu...) to survey development folder, to begin the new application
- import and export data
- implementation - sending complete application to production environment

Perhaps this interface will become obsolete (or less important) when developers became familiar with the new environment. But it is proved as a good start for new developers.

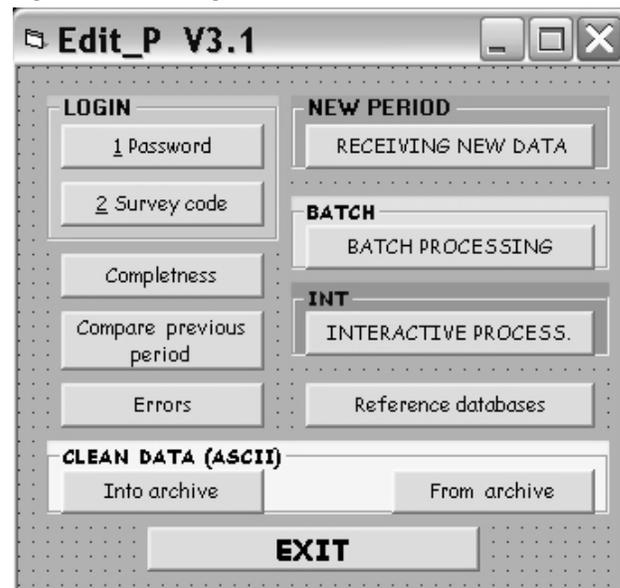
One of the main questions was how to design data models in the editing system: form-based or record-based. Old mainframe applications were record-based, but we decided to use (with exceptions allowed) form as object of observation: it makes editing much more effective and organized, and it's also better supported by Blaise. But our high speed data entry is record based as well, so we need to convert data first. This was solved by Manipula template record to form; another template

(form to record) is applied where record structure is needed for further processing. Despite templates, restructuring is time consuming and error-prone, and in the future we will try to tend towards single (form based) data model in the process of survey data entry and editing.

## 5. The system - user's perspective

Data editing user environment is physically defined as a share on the data editing server, and survey applications are located in sub-folders. Access to applications is enabled through general VB user interface. A short manual was prepared to explain users how to use interface and how to edit data, and a course was organised before the first application was implemented.

Figure 1: Data editing user interface



Access to survey applications is defined by parameters entered into the user interface: password, survey code, period of survey, names of reference databases etc. The password also defines the level of user: survey administrator or data editor. Some buttons (e.g. receiving data, archiving) are active for administrator only. Other parameters (e.g. /g - get form; /P2 - data editing page layout) are always fixed. After the login procedure, the user begins with receiving ASCII files (data, references etc) into Blaise, runs batch checking and then goes to interactive review and correction. Blaise (and other Windows) executables are called through VB Shell function, using Blaise command line parameters. Example:

Figure 2: Interactive editing

```
Shell (potDEP & potLK & "M" & SCode & " /g /p2 /t2 /Ydsn /e" & potLK & " /m" & potLK & "depmenu.bwm"), vbMaximizedFocus
```

Figure 2 shows commands behind *Interactive editing* button: call of survey data editing application in user environment, where *potDEP* is path to DEP, *potLK* is path to survey data editing folder, and *SCode* is numeric survey code. Get form mode, data editing page layout and data editing - dynamic checking (/g /p2 /t2) are default settings in the data editing system, and clean records are not brought on the screen (Ydsn). Depmenu and modelib files are designed for the common needs of data editing and can be adapted for the single survey if necessary.

In practice up to 20 users (data editors) work interactively on a survey database, editing their own part of data defined by key values. We tried to avoid mixing batch and interactive processes as much as possible, so batch checking is performed on complete database at the beginning, and later only when necessary. With some surveys, batch checking process is relatively slow and we need to discover reasons: hardware, applications (performance issues), network processes etc. But this does not seem to be a critical point of the system.

## **6. The system - statistician's perspective**

In the old system some statisticians were lacking insight into clean data after data editing. Now the access is no longer a problem any more: user just need to be added into the correct group of users by Windows XP system administrator. Another question is if they really need to access Blaise databases: in most cases a generated ASCII (CSV) file would be quite enough to make a quick overview of final data in Excel (or some other tool, where a short macro analysis is possible).

Other data formats (e.g. ASCII-relational) and various generated reports are also available now for each survey. The contents can be adapted to the user's needs.

## **7. Implementation**

The first survey was implemented into the new system in June 2002. Real work - development and implementation - started in September and October. New Blaise users and developers were supported by experienced colleagues when developing their first applications. Templates and user interfaces were (and still are) developed parallel with applications, considering user needs. Periodically they are replaced with the newer versions.

By the end of March 2003, 16 survey applications were implemented into the new system, including some comprehensive monthly surveys (Monthly Industrial Report, Report on Construction Activities, Report on Purchase of Agricultural Products etc.). No serious problems were perceived and users accepted them well. They are satisfied with better overview of the data editing process (forms instead of records), screen designs, accessibility to applications and flexibility of work (users - data editors - are no longer 'fixed' to their physical part of data). There are some comments on slow batch checking processes and we are working on solution of the problem. From developer's side, a total control and access to the user environment is an excellent way to improve applications, remove eventual errors and help users when necessary. Step by step we are also making improvements on organizational aspects of survey data processing.

Although the system is new (with parts still under development), there are already many lessons learned from the experience. The feedback from developers and users is in general positive. Successful implementation means:

- a step forward in building and implementing standards for application developers (modularity, re-use of components, common templates, using metadata ) and application users (common interface, editing screens, commands, etc.)
- automation of processes
- efficient and uncomplicated administration: overview and control of processes, data flow and users involved
- intuitive usability

- ability to include (and combine) multiple modes of data entry and editing
- increased production efficiency – concentration on quality
- easier maintenance

Administration of some processes (e.g. preparing everything necessary to start new survey period) is not yet completely automated, and with complex surveys some auxiliary tasks will probably remain 'manual'. For such cases good documentation of complete procedure is even more important.

## 8. Conclusions

About half a year after implementation it seems that the main objective is reached: the new system successfully supports data editing of surveys that were traditionally running on a mainframe, supported by various tools. The system also covers CADI surveys: integrated data entry and editing from paper forms, and - probably most important: it brings common in-house standards of development and usage of data editing applications. Some advantages of new approach are already visible now; the others are expected later, when users and developers get more experience. The system can be considered as a part (or module) of general system for data collection and editing at SORS.

With successful start we should not consider the job as finished: new users and developers will need a lot of help and support, and it will take some time and efforts until all the components of the system will be completely harmonized. We should also analyse and find the way of connection between editing system and Metis - SORS metadata base, which is under development.

We believe that lessons learned with the data editing system will also help us to approach to some other important issues and challenges of data collection at SORS. One is building an effective system for administration and automation of survey processes with special requirements - most complex and demanding surveys (e.g. Labour Force Survey, Monthly report on Earnings, etc.) where 'all possible' modes of data collection (including internet) should be enabled and combined. New features of Blaise (version 4.5 or higher) should probably be included to get the proper results.

## 9. References

Edgar, M., and Turner, M. J., Software Quality Framework, Meeting on the Management of Statistical Information Technology, Geneva, 17-19 February 2003

Katic, M., Klanjscek, M., and Kozjek, P., Management and Monitoring of the Statistical Process and Impact on Data Quality, Meeting on the Management of Statistical Information Technology, Geneva, 17-19 February 2003

Kozjek, P., Blaise Generator for High Speed Data Entry Applications, Proceedings of the 6<sup>th</sup> International Blaise Users' Conference, Cork, 2000

Pierzschala, M., and Manners, T., Revolutionary Paradigms of Blaise, Proceedings of the 7<sup>th</sup> Blaise User's Conference, Washington D.C., 2001

SORS and Statistics Sweden, Feasibility study on the architecture of information systems and related equipment issues, Study implementation and Hardware/Software Specification for Tendering, September 1997

Sundgren, B., An Information Systems Architecture for National and International Statistical Organizations, CES/AC.71/1999/4, Meeting on the Management of Statistical Information Technology, Geneva, 15-17 February 1999



# The melting pot - practical experience of scanning paper forms

By Marianne Troelsen & Lars Pedersen, Statistics Denmark

## 1. Abstract

History has proven paper as the number one carrier of questionnaires with many hard-to-beat features. Responding to a questionnaire is in many cases based on the good will of a respondent. Even with the best technological solution, some respondents will always prefer paper to any other solution. And pressed to any other solution will affect the overall response rate.

On the other hand, just because a proportion of our respondents prefer paper to electronic carriers – the NSI's do not have to bare the burden of paper. With that in mind we set out to define a system that makes the need for paper redundant. The basis is a result of a scanning process, where an image and a data file is created for each scanned questionnaire. A prototype was created, that visualized the key feature, which is, the ability to combine image and data in one screen as seen in screenshot 1.

**Beskæftigede ved bygge- og anlægsvirksomhed** 12 15-41-36-89

Medarbejdere og entreprenører

1. Bygningsarbejde: nybyggeri og tilbygning, herunder grundarbejde, væg, jord- og kloakarbejde, planering og vejledning i forbindelse med tilføjelse af forpligter

2. Bygningsarbejde: reparation og vedligeholdelse, herunder også vask og distribution samt arbejde i forbindelse med ombygning og forandring af bygninger

3. Anlægsarbejde: oplysning og håndtering af anlæg, samt reparation og vedligeholdelse af anlæg, herunder f.eks. vand og kloakarbejde, postboks, der af enhver art (se dog punkt 1), brolægning, bro, havnearbejde o.l.

4. Andet arbejde, der ikke er direkte forbundet med oplysning og reparation af bygninger, herunder arbejde, administration, indkøbsarbejde, byggearbejde og materialer, såvel som maskinarbejde o.l.

5. Ikke på arbejde som følge af dårligt vejr

6. Ikke på arbejde som følge af ferie, sygdom, straffe, uddannelse o.l.

7. Antal arbejdere og medarbejdere målte i alt (1-2-3+4-5-6)

	Medarbejdere	Faglærte	Uden faglært	Lærings-/Uddannelses	Beskæftigede d. 1.8. febr. 2002
1					9
2					2
3					19
4					
5					3
6					3
7					33

8. Andre relevante oplysninger (herunder ikke defineret i håndbogens afsnit 12.1)

Øvrige ansatte

Mænd

Kvinder

9

2

19

3

3

33

1

1

Journr. 15413689

TædDate 13-02-2002

NjBygning 9

Reparation 2

Anlægsarb 19

AndetArb

Veiflg 3

Ejfab 3

AntArbejdere 36

auxantalArb

AnsvIndeh 1

Funktioer 1

AndreAns

DevigAns 2

auxantalDev 2

Ansatt 35

auxantal 38

Also the active entry point is marked simultaneously in both image and the editable data field. To ease the error-handling process symbols for warnings and absolute errors are used. In the prototype the search facilities are very limited.

## 2. The prototype – step by step

The prototype was written entirely in Blaise:

- 1) Define a data-model for the questionnaire, including the field-definition and rules-section (where the error-checking is defined)
- 2) Make a definition of the questionnaire in the ZoneDef program. This is where the link between data and the precise zone in the questionnaire is done.
- 3) Make a manipula-program, which combines the data-file and the image file. The key to the match is in this case an identification number.
- 4) Now all there is to do is running the manipula-program. This results in a database, filled with scanned data. Now it is possible to see each questionnaire on screen with the corresponding data in the dep-window (Screenshot 1).

In the prototype it wasn't interesting to see all the clean questionnaires, therefore some additional manipula-programs were used.

- 1) One that runs through the questionnaires checking the rules.
- 2) One that counts all questionnaires, to find clean, suspect, dirty, notchecked.
- 3) One that moves clean questionnaires to a clean database

Now, all you have left is the questionnaires, which need to be checked. It is possible to correct obvious faults, ie. the scan, respondents writing in the wrong columns. Then run the check-programs again, and move the (new) clean questionnaires to the clean database, thus reducing the dirty database to the questionnaires, that need direct action with the respondent.

## 3 First version

With the key functionality in place a system was planned to handle all tasks usually needed to handle paper questionnaires. The tasks in question are error handling and normal everyday retrieval of a specific questionnaire.

Some of the requirements for the solution are:

- The user interface must be menu driven. It shall be possible to get a status on a chosen dataset as to number of records, how many errors and a count on error types.
- Multi user system.
- Possibility to add (also without an image file) and delete a questionnaire.
- Filter. Example: Search for records with specific error-type and possibility to auto-jump between records with the same error-type. Or retrieve all questionnaires from a postal code. Or ..
- Search. In all fields – with wildcard characters.
- Easy access to relevant additional metadata in list boxes.

# Lessons learned on the development of the Unified Enterprise Survey

*Rock Lemay, Statistics Canada*

## 1. Introduction

The Unified Enterprise Survey program incorporates several annual business surveys into a single framework, using questionnaires with a consistent look, structure and content.

The 2002 version of the Unified Enterprise Survey encompasses 41 surveys, collecting standard data points from both large and small businesses across multiple industries, resulting in more coherent and accurate statistics on the economy, particularly at the provincial/territorial and industry levels.

The collection process is done at Statistic Canada's headquarters with a mailout/mail-back approach of over 40,000 questionnaires, using a multi phase mail-out system based on the respondent's fiscal year end, to a maximum of 3 phases.

During the course of this paper, I will take you through the whole process of our system and in particular, the reasoning behind each software and design approach that was adopted. The goal of the system design was to reduce the amount of time required for maintenance and development, but to also give the collection team tools to improve their day-to-day operations, such as analytical reports, to help in their decisions.

A plan to solely use Blaise for all of these processes and controls would not allow us to grow with the increasing demand to streamline the collection operations and the complexity of the questionnaires being designed to reduce response burden.

For UES, we have determined that Blaise's biggest strength is the processing of records at the micro level. It allows for extremely efficient ways of keeping track of edits behavior. However, the speed at which it processes vast amounts of records has pushed us to investigate the use of other software to complement the whole UES collection system process.

The system can be broken down into 6 different areas, each with its own software approach.

## 2. Mailout System

The first step in our mailout/mail-back approach to survey taking is the preparation of questionnaires, address inscriptions, and control mechanisms. The approach taken for this particular process was to use SAS/AF, due to its analytical power of large files and the speed at which it can process data. The system generates mailout files that are sent for addressograph printing and questionnaire stuffing into envelopes. It also facilitates separating the files and sending them to a repository area awaiting the load of these cases into various Blaise applications.

### 3. Blaise applications, Frame Feedback and Central Document Control (CDC)

Each of the 41 questionnaires is a distinct Blaise application, but all records are loaded into a Central Document Control and a Frame Feedback Blaise database. Even with the consistent framework approach to the questionnaire, there are enough differences to require us to build a different Blaise application for each questionnaire. Each of them has on average 100 edits, with approximately 60 of them being generic to all.

The design allows us flexibility during the collection period to change or fix individual applications without disrupting others. However this is not true for the CDC and the Frame Feedback, since all applications contain external calls to these. Should an enhancement be required for either of the two main components after deployment or even during testing, it requires us to recompile all 41 applications. Another drawback is the fact that we have 41 call schedulers (CATI Specs), which makes it difficult for the Collection area to manage, since some of their interviewers work on more than one questionnaire.

One of the approaches we are considering is the use of SAS as a CDC, being the only point of entry for all cases. The Collection area has requested a greater control for case management and follow-up. Because of the mailout/mail-back approach, two kinds of follow-up are required; one for non-response and the other for edits both requiring different types of staff. Using SAS/AF with a built-in selection grid would allow the Collection Managers a greater flexibility in targeting cases depending on the time of collection (i.e.: beginning of collection - non-response follow-up). Using this strategy would necessitate the use of the BCP to allow for the transfer of data between the Blaise application and the SAS dataset. With the BCP, any changes to the CDC would not require major recompiles of the 41 Blaise applications. In using SAS, the creation of MIS reports would be sped up, compare to the share mode utilized in Blaise.

### 4. Analysis tools

One of Blaise's features is the creation of a file with the extension .BTH, which adds an observation every time someone exits a case in true CATI mode. The BTH and edit evaluation features present in Blaise lend themselves well to the survey data processing and historical case access recording. There is a need to analyze this type of data to improve questionnaire design and reduce collection cost.

Analyzing the BTH files has allowed us to measure the efforts done by the collection area. This however was not fully representative of the total effort since it only recorded entries accessed by the DEP. Entry into the CDC is via Maniplus and therefore not recorded. We have created Manipula processes that are called on exit of cases, regardless of their access point. These processes create entries in a pseudo-BTH file. Using SAS, we gather all BTH information for a particular questionnaire and are able to prepare daily progress reports in real-time. This type of data is also extremely useful in debugging, since it demonstrates the full history of case access. In addition, we have started to compile and analyze using SAS, the Blaise Audit Trail data to provide additional information on the collection process.

### 5. Data Capture

During the initial years of development, application design was based on a CATI approach, with skip patterns and **".Show"**. After a few collection cycles, we

noticed that only 15% of all data was captured during CATI sessions. The remainder was from questionnaires, which were later keyed using a non-interactive edit mode. The application was a reflection of the questionnaire and the interviewer would have to go through all cells to typically capture a small number of items. To bring efficiency to this costly process, a string key approach was developed using Blaise, which we named Quick Data Entry (QDE). Only cells with data are captured and subsequently re-integrated into the application. A Quality Control System was later deployed to ensure reasonable accuracy during capture, using Visual Basic to create reports and manage QC parameters such as 100% or batch verification.

This data capture process led us to develop QDE Edit Reports. The Collection areas needed a way to quantify the amount of edits that were failing from the initial capture or respondent data, prior to any intervention from the interviewer. After re-integration of the QDE, which is done in overnight batches, the **CheckRule** is applied and an extraction of the database is done with all the failed edits. To bring meaning to this information, SAS with HTML outputs are used to compile the data into reports for further analysis. This is done for every application, which enables the Subject Matter Areas to review their questionnaire design with the information provided.

The same process can be used to measure the quality of any data needing re-integration in Blaise, such as electronic data reporting or key from image.

Problems with data retention occurs during the re-integration process when data on the questionnaire does not follow the skip patterns set out in the CATI Blaise application. The solution is to show all cells and eliminate the skip pattern flow. We have lost some efficiency in navigating through the application, but have ensured data retention regardless of what respondents fill out.

## 6. Follow-Up

To optimize resource utilization during collection, a better targeting of units had to be employed. A **Dynamic Score Function** was developed. The logic behind this approach is to stop follow-up when your weighted response rate has been reached and transfer resources in strata where more data is required to produce adequate estimates. Using SAS, a calculation on key variables from clean records is used to recalculate the priority of each case. This is then re-integrated into the Blaise application. The CATI specifications give priorities to those cases and ignore others where response rate targets have been met.

## 7. Outputs

The Estimation and Imputation system for the UES is programmed in SAS. A need to output the data into SAS datasets was required to allow quick and efficient process. All data outputs are done from Blaise to ASCII to SAS. Programming the output for all 41 applications would be quite time consuming. By utilizing a standard output specification format in Excel and Visual Basic, we developed a **Code Generator** to produce the Manipula and SAS programs. The same Code Generator is also used to create the QDE re-integration programs.

## 8. Conclusion

As you can see from the steps above, the development of our business model in a modular approach has given us the flexibility to grow with the ever-changing requirements from all stakeholders. Since a large percentage of monetary resources of a survey are allocated to the collection area, a need to understand how and where they are spent is of the utmost importance. Not only do we need to analyze this data, but we must also be in a position to re-tool the systems from the findings of that analysis.

We have determined that Blaise and SAS provide us great tools to ensure micro level data management as well as versatile macro data manipulation.

Further research is being done to improve the UES collection process, such as customized questionnaires generated in XML using previous year's respondent data. Other components requiring data summation are being redesigned with the approaches learned from the UES, such as using SAS for dataset manipulation and Blaise for record level validation and verification.

Blaise will remain as a major player in our collection systems, while we continue to integrate components utilizing whatever software is deemed the most appropriate. This process will continue to evolve, due to the nature of technology and the ever-changing needs of all players involved in Survey taking.

## *Posters*

- **The American Time Use Survey Data Collection Instrument** .....269  
*David Altvater, William Dyer, Christian Borillo, Roberto Picha, United States Bureau of the Census*
- **Using non-Latin alphabets in Blaise** .....281  
*Rob Groeneveld, Statistics Netherlands*
- **Automated Dialing - What will it do for you?**..... 291  
*Dan J. Bernard, Marketing Systems Group*



# The American Time Use Survey Data Collection Instrument

*David Altvater, William Dyer, Christian Borillo, Roberto Picha  
United States Bureau of the Census*

*This paper reports the results of work undertaken by the United States Census Bureau staff. It has undergone a US Census Bureau review more limited in scope than that given to official Census Bureau publications. This report is released to inform interested parties and to encourage discussion of work in progress.*

## 1. Introduction

The American Time Use Survey (ATUS) Blaise data collection instrument, sponsored by the U.S. Bureau of Labor Statistics, developed by the U.S. Bureau of the Census, and conducted in Computer Assisted Telephone Interviewing (CATI) mode, is designed to provide nationally representative estimates of time that Americans spend in various activities. The data are used to measure the value of unpaid, productive work such as housework and childcare, and to measure nonproductive activities such as waiting in line and commuting to work.

The ATUS instrument is particularly challenging because of the dynamic reconciliation of complex time relationships represented in the main module of the instrument, the Diary section. In addition, another section of the instrument is dependent on the time data in the Diary and must be updated from that data.

The ATUS Diary is the section of the instrument that is used to collect data about various activities performed throughout the day. People tend to go through a routine but may not remember how long an activity took (duration entries) or exactly what time they ended the activity, unless it is a regularly scheduled event such as leaving work or watching a television show. A typical interview would collect most of a person's activities and times during the day and then insert, delete, or modify entries as the respondent recalled events and times during the course of the interview.

In the Diary section of the ATUS instrument, time data are collected for activities that cover a 24-hour time period, as reported by the respondent. These time data are collected either by duration units (hours and minutes) or by wall clock stop time of the activity. As the data are collected in the Diary module, the time relationships within and across rows are constantly recalculated relative to the time entries in the previous row. Within a row, if duration is entered, stop time is calculated, and vice versa. The calculation is based on the stop time of the previous row.

**Figure A – Activities and Time Relationships in a Partially Populated ATUS Diary Section Grid.**

	Start	ID	Activity	TIME	Hrs	Mins	Stop	Who	Who_2	Where	Where specify
[1]	4:00AM		Sleeping		1	2	6:00AM				
[2]	6:00AM		Grooming		2	0	6:22AM				
[3]	6:22AM		Watching TV/grooming		1		6:37AM	0		1	Respondent's
[4]	6:37AM		Preparing meals and snacks		1		6:42AM	0		1	Respondent's
[5]	6:42AM		Commute to work		1		7:27AM	55		12	Car, truck, or i
[6]	7:27AM		Working at main job		2	4	11:30AM			2	Respondent's
[7]	11:30AM		Eating and drinking/working on Blaise		1		12:00PM	55		2	

00000001 WHERE 9:09:49 AM 2/14/2003

Backward movement and changes are possible, as are row insertions and row deletions. Particularly challenging to maintaining the accuracy of the data relationships is the need to track several layers of time data, thereby creating a virtual three-dimensional table (row, column, time [previous and current]). The two layers of time are the “current” data displayed in the Diary (layer 1), and the “previous” responses, data which are not displayed but are retained in “shadow” fields (layer 2).

This paper summarizes the two approaches that were attempted in the development of the ATUS Diary modules and presents some of the lessons learned during the process.

The first approach allowed the interviewers to key data in either the duration fields (hours and minutes) or the stop time field, without specifying which, and the program determined which method was used. This non-specified, or implied, data entry method was not used for production because of programming complexities it introduced.

The second approach, which is currently used for production ATUS interviewing, also allows the interviewers to enter the data using either duration or stop time, but the interviewers must first select the method they will use on any given row. Then, when data are keyed into either the duration or stop time fields, the program uses the selected method of time data entry to determine which way the time calculations are handled by Blaise. At any time, the interviewer can switch methods by toggling the value of a row-level field in the Diary table. This paper will discuss the pros and cons of these two approaches and the avenue used for development of each, including some lessons learned.

## 2. Version 1 – Non-Specified Method of Time Entry

The first attempt for the Diary section was developed as a working demonstration prototype. It allowed the interviewer to enter time data either as duration or as stop time without requiring the interviewer to specify which method was used. The code was written to allow free navigation, insertion, deletion, and sandwich insertions (splitting an activity). The code reconciled all of the time entries across the rows of events. This presented a problem because the program first needed to determine which entry method was used. Based on the method used by the interviewer, the program then used the appropriate section of code to perform calculations on subsequent time entries past the current Diary section row.

This prototype was developed to demonstrate the kinds of movement, time entry method flexibility, and accounting that might be possible for a Diary.

When entries in the Diary were added from start to finish, everything worked as expected for time calculations. But when the grid was already populated and changes were made (adding, deleting, inserting rows or modifying the content of rows), the time calculations had to be recomputed. Reconciling the two types of entries, stop time and duration, was difficult because the program needed to remember which method of time calculation was used on any given row. This information was not tracked in early Version 1 Diary code. Techniques to record the nature of the entries were introduced but quickly became unmanageable because of the need to retain previously entered data that is not included in the Diary display (since it shows only current data entries). Data fields which are not displayed but which are retained to help determine the nature of respondent activity and previous responses are called shadow variables.

When a new row was inserted, hour, minute and/or stop time entries were empty/missing – only the start time was present since it was populated with the previous stop time entry. Users were able to update any field freely. This set up two scenarios: 1) the hour-minute entry may update the stop time field or 2) the stop time entry may update the hour and minute fields. The following examples and pseudo code show the process of making these calculations:

If the time duration was entered, the instrument would calculate the stop time for the present row and adjust entries in the subsequent rows of the Diary.

If the stop time was entered, then the duration time in hours and minutes was calculated, adjusted and propagated to the following rows in the grid.

```

If hourduration = response or minutesduration = response then
    stop = (start + hourduration + minutesduration)
else
    timeresult = stop-start
    hourduration = timeresult.hour
    minutesduration = timeresult.minutes
endif

```

Here is an example to illustrate how the calculations became a challenge for the Diary programmer.

Entry					Calculated				
Row	Start	Hour	Minute	Stop	Row	Start	Hour	Minute	Stop
1) Sleep	4:00	2		6:00 AM	1) Sleep	4:00	2	0	6:00 AM
2) Eat			30	6:30 AM	2) Eat	6:00	0	30	6:30 AM
3) Work		9		3:30 PM	3) Work	6:30	9	0	3:30 PM

Reported time entries are in gray.      Rules execution recalculates all of the time entries.

In this example, each stop time entry updates the hours and minutes and the next row's start time. If the time duration was entered, the instrument calculates the stop time for the present row and adjusts entries in subsequent rows. Notice that once the fields are calculated we cannot easily identify the method used to derive the entries. This is due to the fact that using the "response" query only works the first time through the Diary as the interviewer enters times for the first time. The second and later time(s) cannot rely on the "response" test to determine which method of time entry was used, because both duration and stop fields would then contain entries. In the example above, gray areas represent the data as reported by the respondent and will drive which type of time calculations are done by the program. To continue using the program to calculate the row times, shadow

variables were used to hold information about the past iteration of the rules process.

Blaise performs calculations described in the code from the top down. These “Rules” are re-executed each time a field is exited after a data change is recorded. This is the nature of interactive data collection and is what the language is designed to do, but potentially causes us some problems with the time calculations in certain circumstances.

Now, lets go back and modify the previous example by adding another activity. The respondent remembers that they spent 45 minutes commuting to work. The interviewer inserts a row for commuting, which takes place between eating breakfast and arriving at work, i.e. insert a new row - #3.

Entry					Calculated				
Row	Start	Hour	Minute	Stop	Row	Start	Hour	Minute	Stop
1) Sleep	4:00	2		6:00 AM	1) Sleep	4:00	2	0	6:00 AM
2) Eat			30	6:30 AM	2) Eat	6:00	0	30	6:30 AM
3) Commute			45	7:15 AM	3) Commute	6:30	0	45	7:15 AM
4) Work				3:30 PM	4) Work	7:15	9	0	4:15 PM
Insertion of row #3 for a fixed duration of 45 min.					May cause incorrect reporting of work stop time.				

This will result in a recalculation of the stop time for entry #4, Work, ending at 4:15, when the respondent originally reported leaving work at 3:30 pm. This happens because the evaluation inside the if statement used the stop time of the row #3, Commute, as the start time of row #4, Work, and then added to row #4’s start time the hours and minutes previously computed from a duration entry.

One can see that the stop time calculation is being performed when the hours and minutes are reported and each row is reconciled after any cell is exited for each entry in the table and the stop time is ignored for row #4 because it is in the “else” portion of the “if” statement. Of course we could reverse this condition and make calculations based on stop times, but we will have a similar problem reconciling the duration entries. This demonstrates the need for retaining the type of entry and being able to recalculate start times for successive rows throughout the table.

This is a simple example that illustrates a problem with adding activities. Consider what is involved in adding the overhead within the code to control deletions and sandwich insertions (splitting an event).

To control all the specific scenarios, the initial code had up to seven layers of if-else-endif statements to invoke edits. There was a great deal of logic for covering all the possible entries that could be made in the time calculation fields. This code employed numerous on/off flags, which in turn would set other shadow variables depending on conditions in the entry. The example shows that reconciling the time entries correctly in previous rows is difficult no matter which method is used to solve the problem. Accordingly, maintaining and enhancing the code became more

and more difficult as the programmer tried to implement increasingly more complex requirements and provide maximum flexibility for the interviewer.

Testing uncovered several problems with the time calculations, and the complexity of the code made it virtually impossible to fix in the time remaining to production. Therefore, a decision was made to redesign the Diary, with the goals of documenting the requirements and simplifying the code.

### 3. Version 2 – Specified Method of Time Entry

As explained above, The Diary section for ATUS was not working as expected. It was determined through extensive research, testing, and by evaluating problem reports, that Version 1 could not be easily patched or made production ready in the time for the ATUS Dress Rehearsal live interviewing.

The decision was made to abandon Version 1 of the Diary and build a Diary section from the beginning using a more structured design approach and the knowledge gained during our work with Version 1. This redesigned Diary became known as Version 2. The number one priority assigned to the redesigned Diary code was to keep the time calculations correct and synchronized across all rows in the Diary table.

A new Diary field, “TIME”, was created. Interviewers use it to choose which of the two methods will be used to enter row times, i.e., time duration or stop time. By default, the field is set to “1 – Time Duration” but the interviewer has the flexibility to toggle between methods at any time.

Time calculations move from the data entered forward through the table, and are based on the method selected. For example, if Time Duration is active, from the point where a time duration is entered forward, the code calculates subsequent row time values by adding the hours and minutes to the start time to populate the stop time field. If Stop Time calculations were active, then after the interviewer enters the stop time for an activity, the code calculates the hours and minutes between that row’s start and stop times. Then it populates the Hours and Minutes fields on that row, and moves forward to subsequent rows to recalculate start and stop times for them.

Also important in this method of time calculation is the use of SHOW fields to indicate, on subsequent rows, which method of time entry was used on each row. Since it could change across rows in the table, this method of graying out the “calculated time values” is visually helpful to the interviewers. The method used to enter time data on each row determines which fields are recalculated for each time event recorded in previous rows.

The code below shows how these calculations were performed.

```
{=====//rpicha-cborillo additions, 04-16-02//=====}  
  IF typeduration = 1 THEN  
    Stop := Start + (HourDur, MinDur, 0)  
  ELSEIF typeduration = 2 THEN {Enter system time format }  
    IF (Start = RESPONSE) AND (.Stop = RESPONSE) THEN  
      IF Start <> Stop THEN  
        TimeResult := Stop - Start  
      ELSEIF (Start = Stop) AND PrevStopDayNumber = 1 then  
        TimeResult := Stop - Start  
      ENDIF  
      HourDur := TimeResult.HOUR
```

```

    MinDur := TimeResult.MINUTE
  ENDIF
ENDIF
{=====//END_rpicha-cborillo additions, 04-16-02//=====}

```

This technique is repeated as new time data are entered. The effect on the table is unidirectional, i.e., it is processed in one direction from the first row to the last. This, plus the retained method of entry field (indicating duration or stop time), allows entries to be computed or recomputed based on new data keyed by the user, at any point in the timeline, without changing the type of calculations used in subsequent entries. The method used to compute time entries in each row must be retained regardless of insertions, deletions or modifications made in previous rows. When this method was implemented, the TIME variable (duration time calculation or stop time calculation) was displayed as a SHOW variable for the user to see, assuring the user that the reported time calculation method was retained.

In addition to adding the TIME field to Version 2 of the Diary, this version introduced another field to allow the user to specifically enter A or P for am and pm.

#### 4. The Redesign Process

In addition to redesigning the Diary, we also put more process controls in place. With the agreement of the sponsor/client of the ATUS, the requirements were identified and documented in a systematic fashion. We identified and prioritized functions and asked our end users and the telephone center interviewers and supervisors to test the functionality of the Diary throughout its development. Their input was very important and was used as a guide for further development of a user-friendly instrument. Once the basic Diary section requirements were established, a Change Control Board was created to manage change to the requirements and to manage the addition of new features.

An important part of the redesign effort was that the specification was organized to facilitate an incremental approach to development. This insured that each layer was working as designed, and that it was ready for production and maintainable.

- Grid design was considered first. Read only fields, visibility of fields in the table, and number of rows displayed on the screen at any given time were defined and agreed upon in advance.
- The row level requirements were discussed and functional specifications were developed to support activity at the row level before adding enhancements to table levels.
- Relationship of the row or time entry to following events was addressed. The crux of the matter was to consider only time relationships within the row and not insertion activities when developing and testing this layer of the Diary.
- Perform the calculation of time entries for each row in the grid. This means that we were able to enter any number of rows in the grid and start performing modifications on any given row and allow the code to calculate and reconcile time entries in subsequent rows.
- Deletion and insertion functionality were added last.

- After the core functions were defined, implemented, and thoroughly tested, navigation requirements (including edit signals and checks) were defined more thoroughly and implemented.

The reorganized Diary reduced the number of extra variables by 50, and the number of checks and edits was reduced from fifteen to two. The redesigned Diary section of the ATUS instrument followed a structured design method. This ensured that each stage of development was tested and debugged before the next phase of development was started.

## 5. Lessons Learned

The first version of the Diary was developed as a prototype to demonstrate how time entries in a table can be created, modified, deleted, and kept synchronized. But, partly by design, it was not geared for production use. It was developed to demonstrate the power of Blaise to handle complex time relationships across the rows of a table and the flexibility of Blaise to allow multiple approaches to inserting, deleting, and reconciling row data in a table. It served this purpose very well even though new functionality was requested and introduced more quickly than the code could be debugged and stabilized. As a result, the code became too complex and unstable in too short a time frame to be considered production ready.

We hoped that the demonstration code and prototypes could be used for production with little modification, but the code was developed over several months and was under constant revision. While new features were being added, it was difficult to test and debug a stable set of code, and the code became very complex. Problems in one version were not solved before new functionality was introduced. Consequently, the debugging burden intensified with each new code revision. It was not likely that the Version 1 code could have been thoroughly debugged by the survey production date or that it could be easily maintained due to the fact that it was originally built to demonstrate functionality in a rapid development, single purpose mode (demo only).

We learned a lot from the demonstration Diary Version 1, however, and the research and effort it took to build it were instrumental in our ability to develop the Version 2 Diary in a relatively short period of time.

By using a more structured design approach that documented and controlled requirements and design, we were able to establish a time schedule for each release of the Version 2 Diary code. For each release, we assigned specific functionality and did not introduce new features until the “current” version was fully tested and debugged. As a result, new problems could be solved quicker and the code could be stabilized before we introduced new functionality.

This exercise underscored the need to distinguish between demonstration prototypes and production ready code. It helped us to establish a more structured design approach that we now use to repeat the successes we established during the design and development of the Version 2 Diary.

Screen captures are attached to help the reader understand the ATUS instrument and coding. Screen shots are from an example run of the Version 2 Diary section. Other examples are presented in the poster session at the International Blaise Users Conference 2003, in Copenhagen, Denmark.

## The American Time Use Survey (ATUS)

Figure 1 Introduction Screen for ATUS

American Time Use Survey Release 4.1  
Forms Answer Navigate Options Help

ATUS Household Roster Eligible Days Status FAQ Appointment

◆ CENSUS CATI SYSTEM  
American Time Use Survey  
Case status is: Need control card  
Date is: Friday, February 14, 2003 Time is: 8:36AM

Eligible Days:  
February 14, 2003 February 15, 2003  
February 16, 2003 February 17, 2003  
February 18, 2003 February 19, 2003  
February 20, 2003 February 21, 2003

1. Continue 3. Call in  
2. Skip all notes and go to DIAL 9. Quit

Front

Start  
Ready  
CPS Status  
HH Roster  
Notes

0000001 START 8:37:47 AM 2/14/2003

At the Bureau of the Census we have developed screen standards for use across both Computer Assisted Personal Interviewing (CAPI) and CATI surveys. Field representatives and telephone interviewers expect the parchment colored infopane, light grey fieldpane, function keys as described in the .bwm, parallel tabs and status bar.

Figure 2 Introduction to the Diary, Section 4 for ATUS

American Time Use Survey Release 4.1  
Forms Answer Navigate Options Help

ATUS Household Roster Eligible Days Status FAQ S3 Appointment

Now I'd like to find out how you spent your time yesterday, Thursday, February 13, 2003, from 4:00 in the morning until 4:00 am this morning. I'll need to know where you were and who else was with you. If an activity is too personal, there's no need to mention it.

1. Enter 1 to Continue

Section 4 - Diary

Intro to time diary 1

0000001 CORE\_LEAD 8:43:04 AM 2/14/2003

Figure 3 Introductory Diary Screen Asking about Activity.

**American Time Use Survey Release 4.2**

Forms Answer Navigate Options Help

ATUS Household Roster Eligible Days Status FAQ S3 S5 S8 Appointment

**So let's begin. Yesterday, Wednesday, at 4:00 AM, what were you doing?**

- Use the slash key (/) for recording separate/simultaneous activities.
- Do not use precodes for secondary activities.

- Sleeping
- Grooming (self)
- Watching TV
- Working at main job
- Working at other job
- Preparing meals or snacks
- Eating and drinking
- Cleaning kitchen
- Laundry
- Grocery shopping
- Attending religious service
- Don't know/Can't remember
- Refusal/ None of your business

	Start	ID	Activity	TIME	Hrs	Mins	Stop	Who	Who_2	Where	Where specify
[1]	4:00AM		Sleeping		1						
[2]											
[3]											
[4]											
[5]											
[6]											
[7]											

00000001 ACTIVITY 3:21:50 PM 3/20/2003

Figure 4 Time Field Entry

**American Time Use Survey Release 4.2**

Forms Answer Navigate Options Help

ATUS Household Roster Eligible Days Status FAQ S3 S5 S8 Appointment

**How long did you spend sleeping?**

- Type 1 to enter duration (hours, minutes).
- Type 2 to enter stop time.

	Start	ID	Activity	TIME	Hrs	Mins	Stop	Who	Who_2	Where	Where specify
[1]	4:00AM		Sleeping		1	0					
[2]											
[3]											
[4]											
[5]											
[6]											
[7]											

00000001 TIME 3:22:15 PM 3/20/2003

Time field is highlighted here, showing the default method of time data entry, "Duration".

Figure 5 More Diary Entries

American Time Use Survey Release 4.2

Forms Answer Navigate Options Help

ATUS Household Roster Eligible Days Status FAQ S3 S5 S8 Appointment

**What did you do next?**

- Use the slash key (/) for recording separate/simultaneous activities.
- Do not use precodes for secondary activities.

1. Sleeping	8. Cleaning kitchen
2. Grooming (self)	9. Laundry
3. Watching TV	10. Grocery shopping
4. Working at main job	11. Attending religious service
5. Working at other job	12. Don't know/Can't remember
6. Preparing meals or snacks	13. Refusal/ None of your business
7. Eating and drinking	

	Start	ID	Activity	TIME	Hrs	Mins	Stop	Who	Who_2	Where	Where specify
[1]	4:00AM	<input type="checkbox"/>	Sleeping		1	2	6:00AM				
[2]	6:00AM	<input type="checkbox"/>	Grooming		2	0	6:41 AM				
[3]	6:41AM	<input type="checkbox"/>	Commuting		1		7:11 AM				
[4]	7:11AM	<input type="checkbox"/>			1						
[5]											
[6]											
[7]											

00000001 ACTIVITY 3:25:25 PM 3/20/2003

Note the SHOW fields alternating as a result of toggling to stop time on row 2.

Figure 6 Row Insertion

American Time Use Survey Release 4.2

Forms Answer Navigate Options Help

ATUS Household Roster Eligible Days Status FAQ S3 S5 S8 Appointment

• Enter Minutes

	Start	ID	Activity	TIME	Hrs	Mins	Stop	Who	Who_2	Where	Where specify
[1]	4:00AM	<input type="checkbox"/>	Sleeping		1	2	6:00AM				
[2]	6:00AM	<input type="checkbox"/>	Watching TV		1						
[3]		<input type="checkbox"/>	Grooming		2	6	6:41 AM				
[4]	6:41AM	<input type="checkbox"/>	Commuting		1		7:11 AM				
[5]	7:11AM	<input type="checkbox"/>			1						
[6]											
[7]											

00000001 MINDUR 3:28:20 PM 3/20/2003

Note that all other time fields are grayed out pending entry into this inserted row.

Figure 7 Sample Activities in a Partially Populated Grid for ATUS

American Time Use Survey Release 4.1

Forms Answer Navigate Options Help

ATUS Household Roster Eligible Days Status FAQ S3 S5 S8 Appointment

**Where were you while you were eating and drinking/working on blaise?**

PLACE MODE OF TRANSPORTATION

1. Respondent's home or yard       12. Car, truck, or motorcycle (driver)  
 2. Respondent's workplace       13. Car, truck, or motorcycle (passenger)  
 3. Someone else's home       14. Walking  
 4. Restaurant/Bar       15. Bus  
 5. Place of worship       16. Subway/Train  
 6. Grocery store       17. Bicycle  
 7. Other store/Mall       18. Boat/Ferry  
 8. School       19. Taxi/Limousine Service  
 9. Outdoors away from home       20. Airplane  
 10. Library       21. Other (specify)  
 11. Other place (specify)

	Start	ID	Activity	TIME	Hrs	Mins	Stop	Who	Who_2	Vwhere	Where specify
11	4:00AM		Sleeping	1	2		6:00AM				
21	6:00AM		Grooming	2	0	22	6:22AM				
31	6:22AM		Watching TV/grooming	1		15	6:37AM	0		1	Respondent's
41	6:37AM		Preparing meals and snacks	1		5	6:42AM	0		1	Respondent's
51	6:42AM		Commute to work	1		45	7:27AM	55		12	Car, truck, or
61	7:27AM		Working at main job	2	4	3	11:30AM			2	Respondent's
71	11:30AM		Eating and drinking/working on Blaise	1		30	12:00PM	55		2	

00000001 WHERE 9:09:49 AM 2/14/2003

Figure 8 Abbreviated Keying

[3] 6:22AM [3/grooming] 1

Abbreviated keying for accuracy and speed

[3] 6:22AM Watching TV/grooming 1 15 6:37AM 0 1

Results of previous key entry notice the first activity is automatically filled to match the pre-code 3

Figure 9 Who Code Follow Up Question

**Who was in the room with you? / Who accompanied you?**  
 On HH Roster      NonHH Family      Other NonHH

0. Alone       51. Parents       54. Friends  
 2.       52. Other non-HH family members < 18       55. Co-workers, colleagues, clients  
 3.       53. Other non-HH family members 18 and older (incl. Parents-in-law)       56. Neighbors, acquaintances  
 4.       57. Other non-HH children < 18  
 5.       58. Other non-HH adults 18 and older  
 6.  
 7.  
 8.  
 9.  
 10.  
 50. All household members

	Start	ID	Activity	TIME	Hrs	Mins	Stop	Who	Who_2	Where	Where specify
(1)	4:00AM		Sleeping	1	2		6:00AM				
(2)	6:00AM		Grooming	2	0	22	6:22AM				
(3)	6:22AM		Watching TV/grooming	1		15	6:37AM	0			
(4)	6:37AM			1							
(5)											
(6)											
(7)											

00000001    WHO    8:59:22 AM    2/14/2003

Notice the way the answer list is organized. The sponsor/client requested that we list different categories in columnar display based on the column header listed in blue in the infopane. This is an interesting visual way to present the user with activities for selection.

Who was with you?

Figure 10 Where Code Follow Up Question

**Where were you while you were watching tv/grooming?**  
 PLACE      MODE OF TRANSPORTATION

1. Respondent's home or yard       12. Car, truck, or motorcycle (driver)  
 2. Respondent's workplace       13. Car, truck, or motorcycle (passenger)  
 3. Someone else's home       14. Walking  
 4. Restaurant/Bar       15. Bus  
 5. Place of worship       16. Subway/Train  
 6. Grocery store       17. Bicycle  
 7. Other store/Mall       18. Boat/Ferry  
 8. School       19. Taxi/Limousine Service  
 9. Outdoors away from home       20. Airplane  
 10. Library       21. Other (specify)  
 11. Other place (specify)

	Start	ID	Activity	TIME	Hrs	Mins	Stop	Who	Who_2	Where	Where specify
(1)	4:00AM		Sleeping	1	2		6:00AM				
(2)	6:00AM		Grooming	2	0	22	6:22AM				
(3)	6:22AM		Watching TV/grooming	1		15	6:37AM	0		1	
(4)	6:37AM			1							
(5)											
(6)											
(7)											

00000001    WHERE    9:00:01 AM    2/14/2003

Where were you while you were...

# Using non-Latin alphabets in Blaise

Rob Groeneveld, Statistics Netherlands

## 1. Basic techniques with fonts

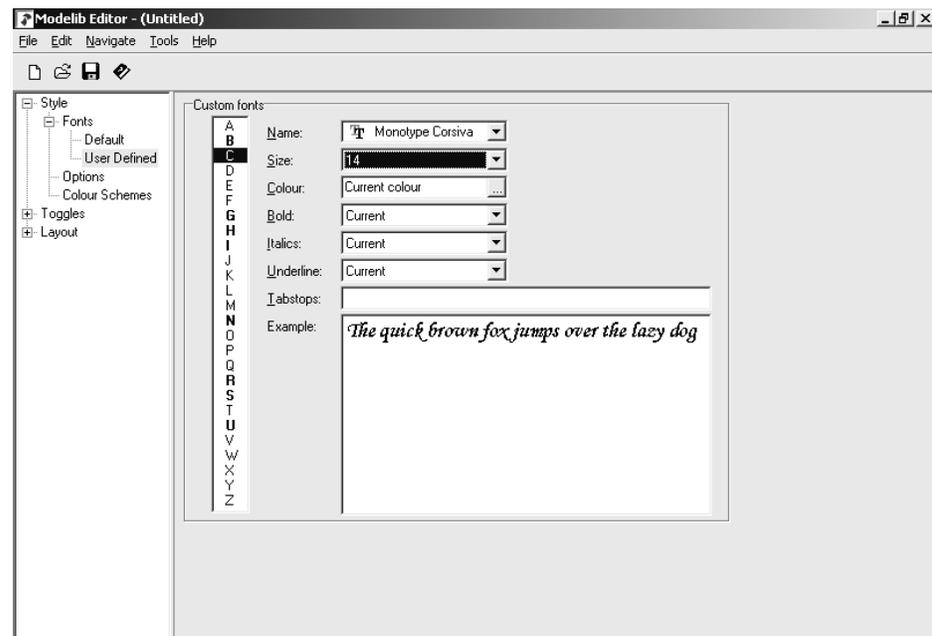
In the Data Entry Program in Blaise, it is possible to use different fonts. Here, we show an example with one font. From the Control Centre in Blaise, open the Modelib Editor:

**Tools | Modelib Editor | File | New**

Now choose:

**Style | Fonts | User Defined**

Select a free letter, for example 'C'. We can associate a font with this letter, e.g., Monotype Corsiva. We also change the size to let's say 14. The Mode Library Editor screen will look like this:



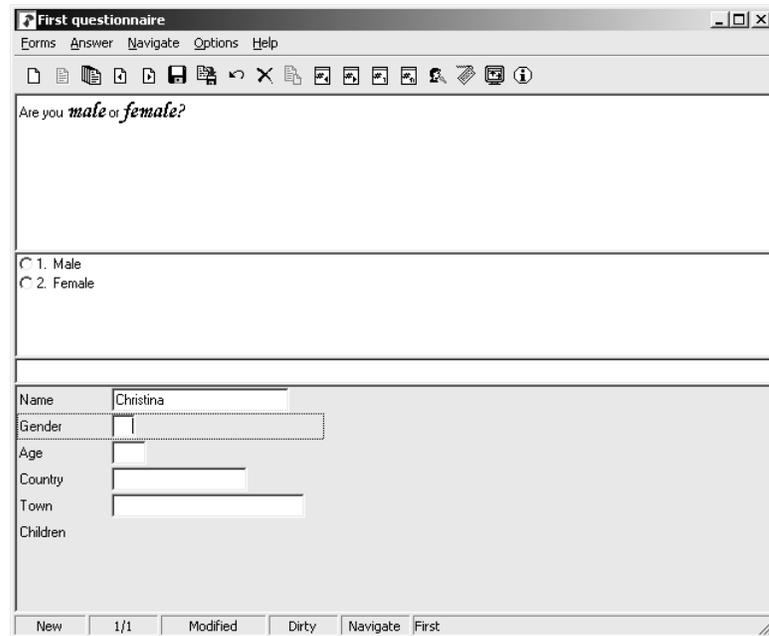
Now we can use the font character C in the text for the questions in a Blaise questionnaire. A question about the gender of the respondent could be, for example:

Gender “Are you male or female?”: (Male, Female)

If we change this into

Gender “Are you @Cmale@C or @Cfemale?@C”: (Male, Female)

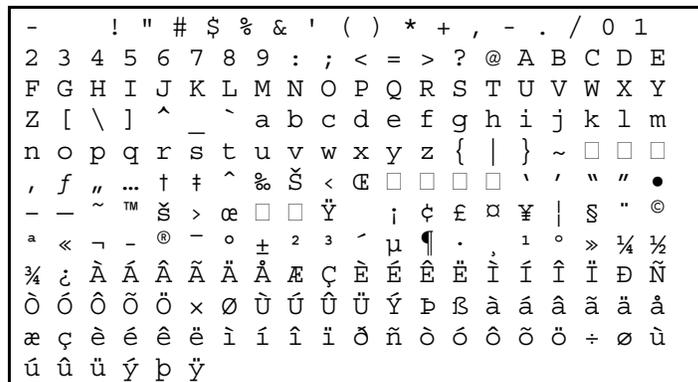
the words ‘male’ and ‘female’ will be shown in the font represented by the letter C. Running this instrument shows the following data entry screen:



The words ‘male’ and ‘female’ now appear in Monotype Corsiva. Languages like English, French, Dutch, German, Swedish, Italian, Spanish, Indonesian, Turkish and a lot of other languages are written in an alphabet technically known as the Latin alphabet because it derives from the alphabet the Latin language was (and is) written in. Readers and writers of these languages will perhaps be surprised to hear that their alphabet is the “Latin” alphabet, because they never thought about it that way. For them, their alphabet is simply the “normal” (English etc.) alphabet.

When using computers, the Latin alphabet is nowadays coded in the so-called ANSI code in MS Windows and other operating systems. This is a coding system which uses a set of 8 bits to represent each character. Hence there is room for  $2^8 = 256$  different symbols. A number of codes are reserved for punctuation marks, the space, arithmetic symbols, mathematical symbols, typographic symbols, a few currency codes and other characters. Letters are represented in both uppercase and lowercase. In addition, there is room in the ANSI code for some variations of the Latin letters used in other Latin-alphabet languages.

These are the symbols represented in an often-used Windows font, Courier New:



Characters that are not printable are represented by a □. As an illustration of special “Latin” letters, see the eth ð (second line from below) and the thorn þ (last line) as used in Icelandic.

## **2. Problems when using non-Latin alphabets**

### **2.1 Left to right writing**

Writers of the Latin alphabet write from left to right. Some other-language writers write from right to left (historically also other ways were used, for example from top to bottom).

### **2.2 Ligatures**

Other alphabets can contain combinations of letters called *ligatures*: two letters (rarely more) are combined into one character. Writing such a combination is easy: you just use your pen to write, for example, the second letter inside the first letter. In printing, you need a separate symbol. When printing books in the Latin alphabet, sometimes the combination “fi” is treated as a ligature (the ‘i’ losing its dot and appearing under the ‘f’).

### **2.3 Combinations of consonants and vowels**

Some alphabets write vowels in combination with consonants. The vowels are usually symbols like strokes or dots. Because one consonant can have various vowels, in printing you need separate symbols for all possible consonant – vowel combinations. Basically this is still an alphabetic system.

### **2.4 Ideographies**

Other ‘alphabets’ (more rightly called ‘writing systems’) use symbols for whole words or syllables. Examples of these are Chinese, Japanese and Korean. Because many words are in common use, each represented by a different symbol, the number of symbols becomes very large, in the order of thousands or tens of thousands. These writing systems are called ‘ideographies’.

## **3. The Arabic alphabet**

The Arabic alphabet has 28 consonants and is written from right to left. The vowels are written as little symbols (strokes and curls) above or below the consonants. There are also symbols for doubling consonants and for denoting the absence of a vowel, there is a consonant written only in combination with vowels (the ‘hamza’) and a few other special symbols. Moreover, the consonants can have up to four different forms: when written initially, in the middle, at the end or isolated (not all consonants have all variations). So here we see a problem not mentioned before: different forms of the letters depending on their position relative to other letters, in addition to problems Nos. 2.1 and 2.3.

All this is really not very complicated and a foreigner can learn to read and write with a pen with not too much trouble, but in printing or typing on a computer one needs many symbols (for the different forms of the consonants, each of them in combination with all possible vowels, etc.). A very elegant system of writing thus leads to a complicated printing and typing task. There is one simplification, however: usually the vowels are not printed or written, only the consonants. The reader knows from the context how to read the consonant-only words. Notable exceptions are the Koran, in which vowels are always written in order to make sure that the pronunciation is always identical, and dictionaries, in which words with the same consonants but different vowels must be distinct. Some other languages, not cognate with Arabic, are also written in the Arabic alphabet with a few additions, for example Farsi, Pashto and Urdu.

## 4. A solution for Cyrillic alphabets

We can use fonts in Blaise to represent non-Latin alphabets that are sufficiently close to the Latin writing system. This applies mainly to Greek and Cyrillic alphabets. The Greek alphabet stands at the origin of the Latin alphabet and is hence similar (more rightly we should say “the Latin alphabet is similar to the Greek”), but uses different letter forms (of course, it would not be a different alphabet if it didn’t). The Cyrillic alphabet has elements from the Greek, Latin and Hebrew alphabets. It has some variations for different Slavonic and other languages. A freeware font called K8 Kurier Fixed implements the Cyrillic alphabet for Russian. Its symbols are:

•	-	!	□	#	\$	%	&	□	(	)	*	+	,	-	.	/	0	1	
2	3	4	5	6	7	8	9	:	;	<	=	>	?	@	A	B	C	D	E
F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y
Z	[	\	]	^	_	□	a	b	c	d	e	f	g	h	i	j	k	l	m
n	o	p	q	r	s	t	u	v	w	x	y	z	{		}	~	□	□	□
□	□	□	□	□	□	I	□	□	□	□	□	□	□	□	□	□	□	□	□
□	□	i	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□
□	□	□	□	□	□	ï	□	□	□	ë	□	□	□	□	□	□	□	□	□
□	□	ю	а	б	ц	д	е	ф	г	х	и	й	к	л	м	н	о	п	я
р	с	т	у	ж	в	ь	ы	з	ш	э	щ	ч	ъ	ю	а	б	ц	д	е
ф	г	х	и	й	к	л	м	н	о	п	я	р	с	т	у	ж	в	ь	ы
з	ш	э	щ	ч	ъ														

Non-printable characters are represented by □. The first 127 characters (up to and including the ~) are the same as in Courier New and many other fonts. From the character ю (4<sup>th</sup> line from below) Cyrillic letters are represented, both lowercase and uppercase. So we must use these characters when we want to write something in Cyrillic letters in Blaise. However, these characters do not appear on the keyboard. When we want to type them in an MS Word document such as this one, we can insert them via **Insert | Symbol**, or type some key combination like ALT + 0192 for ю or ALT + CTRL + SHIFT + Z for ф.

This is inconvenient, although it can be used for small amounts of text. To simplify typing I designed a Manipula program which takes ordinary typed Latin alphabet letters and transforms them into Cyrillic characters. The basic idea is to have a conversion table between Latin letters (single letters or up to four-letter combinations) and Cyrillic characters. A short idea of the conversion table:

A - А  
 B - Б  
 V - В  
 G - Г  
 D - Д  
 E - Е  
 EX - Ё

And so on. The complete table is available upon request. This is not an official transliteration system, by the way. I used characters not appearing in the Cyrillic alphabet like X and H to distinguish letters from one another if necessary. One can then type one’s Russian text on the normal keyboard in the Blaise editor, for instance:

Kak Vashe imax? ('What is your name?')

The principle behind the Manipula program is to recode the Latin letters to the corresponding Cyrillic characters in the Kurier Fixed font, like this:

```
CASE TransChar OF
'A': RusChar := CHAR(223)
'B': RusChar := CHAR(193)
'D': RusChar := CHAR(196)
.
.
.
ENDCASE
```

Here, TransChar is the character to be recoded and RusChar is the transcribed Cyrillic character.

The result of converting the string

Kak Vashe imax?

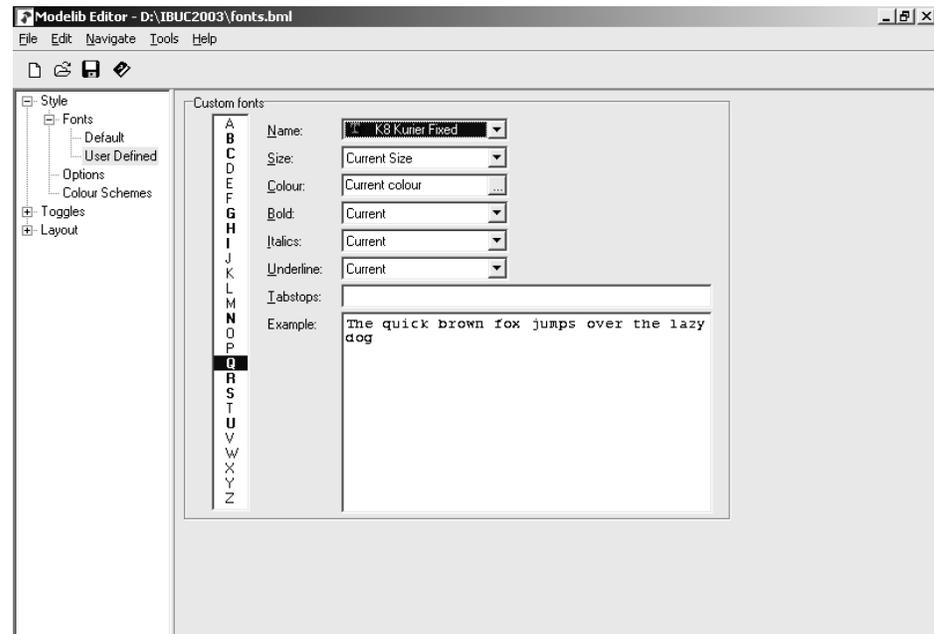
Is:

ëÁË ÷ÁÛÅ ÉÍÑ?

This converted text can be copied and pasted into a Blaise question text, for instance:

Name “ëÁË ÷ÁÛÅ ÉÍÑ?”: STRING[20]

Now the font Kurier Fixed must be associated with a font letter in the Mode Library. Let's take Q:

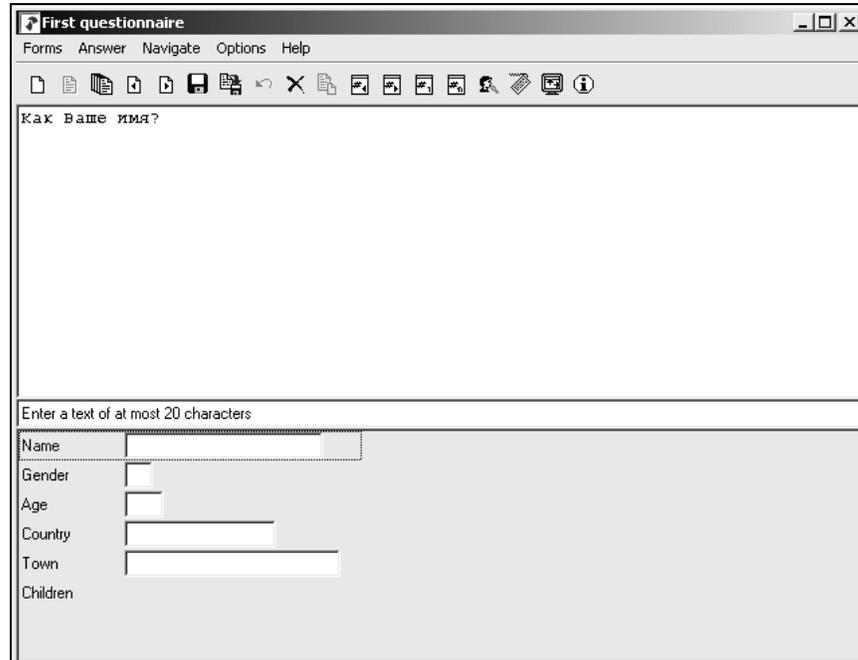


We add the @Q before and after the question text:

Name “@QëÁË ÷ÁÛÁ ÉÍÑ?@Q”: STRING[20]

When we run this questionnaire, the question text appears in Russian:

Как Ваше имя?



All other question texts can be transcribed in the same manner. So the steps to be taken are:

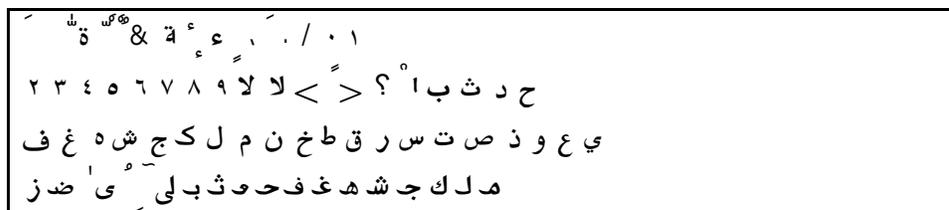
1. Associate the font letter Q in the Modelib Editor with the Kurier Fixed font
2. Type the question texts in normal keyboard characters using the conversion table
3. Transform the text using the Manipula program
4. Copy and paste the question text into the Blaise editor
5. Put the font letter to the left and right of the question text
6. Prepare the Blaise survey using the mode library with the Kurier Fixed font.
7. Run the survey. The Russian text appears in the Data Entry Program.

All this can be done from the Blaise Control Centre.

A similar conversion could be done for the Greek alphabet.

## 5. The Arabic alphabet

When we try to do the same for the Arabic alphabet, we first select a font. There is a freeware Arabic font called Amien 01. This is the character set:





Both the Amien font and the QWERTY utility can be found at <http://user.dtcc.edu/~berlin/font/arabic>  
The Internet address for the QWERTY utility is <http://logics.ghanima.org/en/qwerty.htm>

## 6. Chinese, Japanese and Korean

For Chinese, Japanese and Korean (the so-called CJK languages) a custom solution is available as a commercial product called NJStar Communicator (<http://www.njstar.com>) The use of this product in combination with Blaise is discussed in the paper by Gina-Qian Y. Cheung and Youhong Liu, University of Michigan, entitled 'Displaying Chinese Characters in Blaise' (see the Proceedings of this IBUC). The solution is to use two bytes to represent one symbol. To show the CJK character on the computer screen, the product must be activated. When it is turned off, one sees the two bytes as separate (ANSI) characters.

## 7. Images in Blaise

Yet another solution is to make use of the Blaise capability of showing images in the DEP. One writes or prints the texts in the foreign writing system on paper and scans the images. The texts can then be shown in the DEP as images. This type of solution (along with a discussion of other solutions) is described by Leif Bochis in a paper for the previous Blaise User Conference.

([http://www.blaiseusers.org/ibucpdfs/2001/Bochis--IBUC\\_Paper.pdf](http://www.blaiseusers.org/ibucpdfs/2001/Bochis--IBUC_Paper.pdf)).

## 8. Multi-byte character sets

The problem of representing other writing systems in computer programs was recognized many years ago. Various attempts at solving this problem have resulted in the 16-bit Unicode which is able to represent most writing systems, notably Chinese and related systems, the Arabic and Cyrillic alphabets along with a great number of variations of the Latin alphabet. The Internet, modern operating systems and software use the Unicode system. Hence one can type in MS Word a document in one of these writing systems. Unfortunately, what has been typed in an MS Word document cannot be copied and pasted into the Blaise editor. This has been the source of some confusion. Maybe future versions of Blaise will support Unicode, in which case the problems treated in this paper will be over.

## 9. Fonts in the Blaise editor

The Blaise editor supports only monospaced fonts. These can be True Type fonts or other fonts, but nowadays few fonts are monospaced (Courier is an example). *Proportional* fonts, for example the Times New Roman font this paper is written in, are more pleasing to the eye. In order for a font to even appear as a possible choice in the Blaise editor, it has to be monospaced. The Kurier Fixed font is monospaced, so it can be used in the Blaise editor. Once a font is available in the editor, the field names can be typed in the font. So we are able to make an all-Russian questionnaire.

The author of this paper was unable to find a monospaced Arabic font among available fonts on the Internet.

## **10. Other alphabets**

Some alphabets not mentioned so far but which have turned up in discussions with the Blaise Support Group are the Hebrew, Thai and Vietnamese alphabets. It would be interesting to see how these alphabets can be represented in Blaise.

## **11. Summary**

The problems of using non-Latin alphabets in Blaise were lined out. A solution for a few alphabets close to the Latin alphabet was presented, but many problems still exist for other alphabets, mainly because the number of letters and letter combinations is much greater than the number of characters in the single-byte ANSI character set. For Chinese-like systems, a custom solution is available. Such a solution is not readily available for other writing systems.



# Automated Dialing - What will it do for you?

*Dan J. Bernard, Marketing Systems Group*

## Introduction

Automated dialing has been an increasingly integral part of many market research organizations over the last 10 years. While being adopted as a means of improving interviewer productivity, an unexpected by-product has been quality improvement by taking some of the tedium out of the interviewers job and insuring a consistent application of dialing technique. This paper will address the economic and quality issues surrounding the use of automated dialing – showing the benefits to social research.

## Methods of Dialing

The different terms used for automated dialing can be very confusing. Terms frequently employed are: auto, power, predictive, adaptive, super, progressive, preview, etc. There are three fundamental methods of automated or machine-based dialing:

- **Auto** - telephone number is dialed by a dumb modem
- **Power** - dialer can detect and disposition certain dialing results such as non-working numbers
- **Predictive** - dials more than one number per interviewer using sophisticated statistical algorithms and number knowledge base to deliver a live respondent more quickly

### Auto Dialer

- Dials one telephone number under interviewer control via modem or black box
- Accurate dialing of telephone number
- Dials number much more quickly than manual dialing
- Approximate productivity gains of 3 to 5%
- No abandonment of calls
- No intelligent sensing of dialing result
- No ability to dial “ahead” of the interviewer

### Power Dialer

- Dials one telephone number per interviewer - can be under interviewer control or paced by the programmer
- Builds on all auto-dialer features
- Can automatically detect fax/modem, ring no-answer, non-working, busy signal. Programmatically exchanges messages with CATI system to disposition old number and retrieve new number for dialing
- Conservative productivity gains of 24-50%
- Less tangible benefits of improved working environment
- No abandonment of calls
- No ability to dial “ahead” of the interviewer

### **Performance Gains - Power**

- 1000 23-minute; 94% incidence; Completes per Hour 19% higher
- 950 23-minute; 6.5% incidence; CPH 53% higher
- 2000 10-minute; 95% incidence; CPH 68% higher
- Qualify and transfer to IVR; CPH 100% higher
- Analysis of 1,700,000 dialings: 31 seconds to connect Vs. 56 seconds manually; 43% Reduction
- Large company yields overall 24% increase
- 22 minute; 8% incidence; CPH 96% higher

### **Predictive Dialer**

- Dials telephone numbers in a ratio greater than 1:1
- Builds on all power dialer features
- Uses sophisticated statistical algorithms to calculate quantity of telephone numbers to dial
- Allows adjustment of call abandonment percentage
- Conservative productivity gains of 25% over power dialing
- Can contribute to respondent abuse via call abandonment

### **Performance Gains – Predictive**

- Side-by-side comparisons show 25-50% improvement over power dialing with less than 5% abandonment
- With higher abandonment rates- claims run to 300%

## **What else can Autodialing do for you?**

### **Can replace need for PBX – saving considerable money**

#### **System building blocks provide add-on capabilities**

- Remote Audio Monitoring
- Digital voice capture / playback of open ends
- Whole interview recording – an incredible tool for insuring quality
- Support distributed interviewing
- Integrated IVR (Interactive Voice Response)
- Administrative Features
- Automated inbound/outbound switching - Call blending

#### **Interviewer & Productivity Management**

- Enforces standardized call rules
- Eliminates dialing errors
- Faster dialing means greater throughput
- Dialing modes can be assigned on a study by study and/or station basis
- Real-time graphic and tabular reporting of interviewer productivity
- Full silent monitoring capabilities

#### **Facilities Management**

- Real-time and historical production reporting, by interviewer, study, shift, site, client, and date
- Scheduling module provides information on number of interviewers and supervisors, and those briefed
- Local and remote monitoring capabilities
- Real-time analyses and reporting of trouble on telephone lines

## How Dialers Interface with CATI

- For example, the MSG system is a 20 slot, industrial strength Intel-PC with special telephony hardware by Dialogic
- E1, T1, ISDN, or CO lines plug into boards inserted into backplane
- Lines from interviewing stations are punched onto demarc block and cross connected to lines going to station boards
- Dialer is connected to CATI server via serial connection or ethernet using TCP/IP
- CATI system manages sample file

## What is Heard by the Interviewer?

### Power Mode:

- Some systems can be set to pass call progress tones to the interviewer or just the respondent voice on connects.
- The interviewer will usually hear 'ello'
- The call will sound like a normal call to the respondent

### Predictive Mode:

- No call progress tones can be heard
- The interviewer will usually hear some part of the 'hello'
- The call should sound like a normal call to the respondent unless the call is abandoned

## Research Vs. Telemarketing

- Research has a limited sample frame. The TM supply is comparatively unlimited.
- A primary goal of research is a high response rate
- Researchers cannot afford respondent abuse - on a project OR industry basis
- Predictive dialing works best with more people, researchers often have 5 to 10 people working on a given project

## What is Research doing Different

- We know more about a given telephone number than anyone in the country
- We pay attention to call history
- We will predict the probability of connection rather than predict when an "agent" will be finished
- We are offering predictive dialing with "near zero" abandonment
- Predicting probability of connection works with just a few interviewers
- We can dial numbers in fractional ratios, e.g. 1:1.7 rather than 1:2 or 1:3 like some telemarketing systems which forces high abandonment rates
- Traditional predictive is an optional setting
- We have the flexibility to do it many ways: power, probability of connection, traditional predictive

## **Beyond Productivity – The Improved Environment**

- Improves interviewer retention
- Makes their job easier
- Provides a discipline that isn't innate
- *Improved Job Satisfaction*
- Gives the supervisor time to do things other than push for productivity
- *Improving Project Quality*

### **Social Research is Different**

- It's often heard that social research is different – especially when it comes to length of interview – therefore negating the impact of autodialing. Analysis of dialings shows that 70% of interviews are completed on the first dialing. This allows social research to benefit from the gains of automated dialing.
- Social research call rules often call for double and triple the number of dialings done by market and survey research. This actually gives the advantage to social research of being able to make use of automated dialing. The more dialing of telephone numbers you do, the more productive it can be.

### **Are Dialers Expensive?**

- They are one of the few things in this industry that can demonstrate a return on investment in under a year
- Are you having trouble finding interviewers?
- Would you like to improve project quality?
- You don't need a large phone room realize the productivity gains.

## *Appendix*

- **Developing CASI standards at Statistics Netherlands**..295  
*Frans Kerssemakers, Statistics Netherlands*
- **Contacts**.....306



# Developing CASI standards at Statistics Netherlands

*Frans Kerssemakers, Statistics Netherlands*

## 1. Introduction

Apart from the advantages computer assistance can offer it is above all the expected increase of efficiency in the processing of data that makes CASI (Computer Assisted Self-Interviewing) or, more general, the use of CASQ (Computerized Self-Administered Questionnaires) one of the spearheads of Statistics Netherlands policy. As a consequence there is a rapidly increasing demand for CASI questionnaires. This is particularly true for business surveys where self-administering has always been the common mode of dealing with questionnaires.

One could argue that CASI is not a very precise term. For generally there is little interviewing or interaction involved. However, the focus in this article will be on (Blaise) questionnaires as more or less standardized user-interfaces, which are filled out by a respondent who still has some help desk or fieldworkers' assistance at his disposal. Therefore a term like CASQ would be too broad.

This article will focus on recent CASI developments in business surveys. Some experiments were carried out with persons and households, and although CASI is considered an option here for especially panel studies, there is still a lack of practical experience. Section 2 gives background information on some surveys that represent different types of CASI applications, from simple to complex and from more dedicated to more general. Section 3 takes a closer look at a questionnaire that suffers from space constraints, demanding quite a bit from the layout features of Blaise. Section 4 describes the present efforts to develop standard settings and guidelines on the basis of some kind of form meant to function as a template and test object. Section 5 looks to the future and makes some recommendations concerning CASI for business surveys.

## 2. Background

Statistics Netherlands has already gained a lot of CASI experience with particular kinds of applications. Most prominent are two business surveys where CASI was taken into large scale production some years ago and has been successfully applied since then (next to the paper forms which are still used by most, especially smaller enterprises).

1. One is the use of e-mail facilities for the so-called short-term production statistics on a monthly or quarterly basis. Here very short question forms are used, with often only one or two questions. Up to six questions are put on a HTML-page which has only limited built-in intelligence (e.g. range checks). The page is sent to enterprises by e-mail (e.g. asking for the turnover rate). Many different versions of these short question forms, both on paper and in browser-readable form, can be generated automatically based on a system called Logiquest.

2. The other is a dedicated application for statistical declarations concerning international trade. Here a CD-ROM is sent by mail. It contains mainly a comprehensive (fully Blaise based) system for the coding of imported and exported goods, which can be used stand-alone or in connection with a traditional DEP-based Blaise questionnaire. The latter is also on the CD-ROM, together with facilities for tailoring data collection to the respondent's own needs (such as creating a file with own product codes linked to the official codes, or options for including default values). Although lengthier the fields in the questionnaire take up only one page. The highly technical nature of the questions and the use of question-by-question help for extensive explanations give the DEP more the appearance of what used to be known as a CADI data-entry machine than what is commonly understood by a questionnaire.

### *The Traffic and Transport Survey*

In the spring of 2002 a CASI version had to be developed for a (road) traffic and transport survey. At first sight this survey seemed to have much in common with the international trade survey. In both cases the respondent's task is of a highly repetitive nature. Analogous to every transaction in international trade, every journey of a carrier during a particular week had to be reported. On closer consideration, however, the information that haulage businesses were to provide, mainly by describing their journeys, much more resembled the kind of information asked from private persons, when describing their travel behaviour for instance. The questions asked for only one journey could cover several pages, including some Blaise tables. Besides, information had to be collected on the enterprise and on every vehicle in the fleet. It was decided that a hierarchical design with several DEP's would be easiest. Each unit on which information had to be provided (i.e. enterprise, vehicle and journey) got its own question form. The three questionnaires were linked in a shell which was built in Maniplus. The best way to prevent the respondent from getting lost in a labyrinth of screens was in our opinion to build a strict hierarchy into the design. First deal with the enterprise, then with the vehicles, and only after a vehicle has been dealt with can the respondent go on with the journeys of that particular vehicle. To continue with journeys of another vehicle first that other vehicle had to be selected. Dialogues were used in Maniplus to offer a clear view of respectively vehicles and journeys already dealt with. From here one could open a new question form or take a look at an old form. Because a reported journey, on its turn, may consist of up to four deliveries of goods, we thought it very important to reduce the number of pages in this largest questionnaire (with one form for each journey). At the same time each page should cover a clear-cut task or recognizable subject in the respondent's eyes. What we needed from Blaise was much flexibility to deal with the inevitable space constraints.

### *EDR (Electronic Data Reporter)*

For installing the necessary software on the user's platform, getting access to the different dialogues and data models and sending back data, a Blaise-based management system called EDR (Electronic Data Reporter) was used and further developed. (For further information on EDR, see den Bolster, 2003.) EDR should serve as the central portal for all the self-administered Blaise questionnaires that are DEP-based, as opposed to WEB-based (designed as websites on the Internet).

### *A standard model for testing CASI user interfaces*

Designing and building the traffic and transport survey was rewarding. Much was learned from laying out the question texts, answer lists and instructions. But it was also a time-consuming process. Developing standards or guidelines was therefore desirable. However, both complexity and space constraints forced to specific, tailor-made solutions. These make the survey less suitable as a standard model or

template for the average multi-page CASI form. It is also not suited as a web survey. Especially the screens for a journey show some closely written pages. Also, the filling process may be interrupted several times. Where should data then be stored in a secure way and for how long?

We therefore needed another survey, with a less extreme questionnaire and yet having enough differing style elements, a survey that could alternatively be conducted as a DEP-based survey or as a web survey. Both modes should preferably be based on the same Blaise data model. For a web survey we now can apply two different *style sheets*, written in XSL (eXtended Stylesheet Language). One is meant to reproduce the style of the DEP and another is specially written for web use and what web users may expect from a web site (with buttons instead of a main menu, for instance). See for further explanation Bethlehem & Hoogendoorn (2003).

An opportunity was offered in March 2003 when there was a demand for a CASI version of the survey on Vacancies and Absentee rates. This survey is made up of both short and more extensive modules on the same subjects. On the paper form is a grid with 13 columns, which is too wide to be displayed as a Blaise table without scrolling. It has to be broken up. There is another table which serves as an auxiliary tool for calculating absentee rates. In Blaise it can be handled as a parallel block and displayed on a tab sheet. So we chose this survey to develop our standard model.

About to start is the next stage, consisting of thoroughly testing the draft standards and guidelines.

### **3. A DEP-based style when much space is needed**

As described in the previous chapter we wanted to reduce the number of pages in what was by far the largest questionnaire in the hierarchically designed Traffic and Transport Survey. Each record represents a particular journey. The main body consists of a repeatedly asked set of questions, up to a maximum of four, about each delivery of goods during the journey (where they were loaded and unloaded, the type of the delivered goods, their weight etcetera). The preceding questions are meant to characterize the journey and to determine the routing (depending in particular on whether the journey was empty or loaded, and if loaded, how many times goods were delivered, and whether the vehicle itself had been transported, for instance, by train).

To give the respondent a basic idea of the structure of the questionnaire it was important to have all preceding questions on one page or screen. For then we would have the simple structure of an opening page, followed by up to four identical blocks about the delivered goods, and finally, if applicable, a table for the just mentioned combined transport and another to list transit countries. Companies providing transport for hire and reward, for instance, had to report details on every journey during the reference week. In exchange they would not be bothered the rest of the year. For some this could amount to hundreds of journeys. Eventually we chose the solution that is shown in figure 1.

**Figure 1. The first page of the Traffic and Transport Survey**

**Traffic and Transport Survey: Journey and deliveries**

Journey Answer Navigate Help

Number plate: STX235 Week: 18 (loaded journey with 2, 3 or 4 deliveries) Distance travelled: 1080 km

If the same journey is repeated many times a day, you need only enter the journey details once.  
 Here you can give the **number of extra identical journeys on top of the journey you are entering the details now.**  
 (Identical means: recurring during the day along the same route, and if loaded, carrying the same goods and about the same weight.)

If **no** other identical journeys, leave the field empty (press **[ENTER]**).

Start journey\*  1. Su 27/04/2003  5. Th 01/05/2003  
 2. Mo 28/04/2003  6. Fr 02/05/2003  
 3. Tu 29/04/2003  7. Sa 03/05/2003  
 4. We 30/04/2003

Inside/outside NL\*  
 1. completely NL, without border crossing  
 2. completely abroad, without crossing NL border  
 3. both NL and abroad, with crossing NL border

Type of journey\*  0. empty journey  
 1. journey with 1 delivery  
 2. journey with 2, 3 or 4 deliveries  
 3. journey with 5 or more deliveries  
 (collection-delivery journey)

**double-click or <F8> to start lookup list**

1st Border crossing in NL\*

if **no** freight containers carried, leave field empty (press **[ENTER]**)

Type of freight container carried (max. 2 types)  
 1. 20 ft empty  3. 30-40 ft (and more) empty  
 2. 20 ft loaded  4. 30-40 ft (and more) loaded

if **no** combined transport, leave field empty (press **[ENTER]**)

Number plate of trailer   
 if **no** trailer used, leave field empty (press **[ENTER]**)

Distance travelled in kilometers\*

Combined transport  1. yes, combined transport

Volume (used) in %

if **no** other identical journeys, leave field empty (press **[ENTER]**)

Surface area (used) in %

Number of **other** identical journeys

When using self-administered questionnaires the question text or at least a clear indication of the question should visually be close to the data-entry field. The same is true for important instructions, especially on how to navigate or operate the questionnaire. Our aim with regard to the page shown in figure 1 was to space out a lot of question elements on a single page. We therefore had to avoid the permanent display of complete question texts, instructions and explanations on the screen. The obvious way to achieve this in Blaise is to use the infopane, here on top, for these elements, as they will only be shown if the corresponding field is selected. To keep a good view of the page it is assumed that the usually shortened texts next to the fields can be made clear enough. This is not to be taken for granted. Often in business surveys however, as in our example, concepts and terms are used that have a relatively clear meaning for professionals. On the other hand, there is always the danger that their own conceptions are conflicting with the official, often complex definitions (e.g. as to what should be considered a journey).

When visiting some companies to see whether respondents could get along with the whole system, we got the impression that they were able to follow the route through the two-column design fairly easily. How far they consult the extra information on the infopane remains unclear. The price we had to pay for many fields plus text in the formpane was that the infopane was somewhat pushed away. Combined with a small font this is clearly not ideal for instructions and explanations. The question remains however whether more room for the elements on the infopane would help a lot. In production since January 2003, we have so far not noticed any serious complaints. But we have still to find out what help, if any, the respondent is actually using. By the repetitive nature of the respondent's task and by some built-in checks the questionnaire is to some extent self-instructive. Our hope is that the respondent will also learn that there is something like an infopane. Besides, a call can be made to the helpdesk. If it turns out that some questions or concepts need more explanation a question-by-question help is considered (by setting a second language in Blaise or through the WinHelp utility).

On the first page use has already been made of two lookups. More are to follow. Question items were reshuffled so that near the end of the page we could put some tick boxes for special situations that may apply (but in most cases do not). These

boxes can be left empty. To get familiar with these and other types of questions we use standard instructions above and below the respective fields. If there is a lookup question the respondent can as a rule still enter a description in the field and have it accepted, after a signal has come up if it does not match with the list. But, except as an escape, this is not what we want. By using a recurring instruction we are hoping for an almost automatic double-click or press on F8 to open the list and start the search. A tick box seems appropriate for the question whether combined transport did occur or not. But in Blaise such a box is reserved for set questions. So we had to specify a question with two categories, the second of which got an empty text and is not displayed, thanks to the option 'hide empty categories' in Blaise (see *Project/ Datamodel Properties/ Types*).

We also agreed on the convention to put an asterisk behind the field label, which we use instead of the field name, indicating that the question can not be skipped. Something has to be entered. In some cases, such as with 'Volume (used) in %' and 'Surface area (used) in %' a choice can be made which of the two to use. Here a check is needed to ascertain that at least one of the two is answered. It may be a good idea to have a symbol also for choices of this kind.

If there is a loaded journey the questions continue on the second page which is shown in figure 2 (on the next page). In the first column the name of the village, town or city has to be coded where the delivered goods were loaded respectively unloaded. All together there are 270,000 place-names in the lookup files for about 50 countries. First the country code has to be known. Usually the two letters of this code are entered directly. But use can also be made of a lookup. Depending on this code a lookup list for a particular country will be opened. This is functioning well. In the other column a huge lookup list is used for the coding of the delivered goods or commodities. This list contains about 37,000 sometimes rather lengthy descriptions. For finding the appropriate code a trigram search is used.

In contrast with the first page of the DEP we did not succeed in getting all the questions about a particular delivery on one page. This is mainly due to the rather long answer lists displayed, even though we did not display the list for the site of loading in the same form for the unloading site. Clicking on the latter field will reveal that there is a drop down box which can be opened. There is also an instruction below the field to look up at the displayed categories in the middle. The order of the questions was changed such that a relatively special subject, namely dangerous goods, came on the next page. It only applies to a modest number of businesses and journeys. And if it applies the classification code will very often be known already by means of a link with the general classification code for goods. It usually only has to be confirmed and can be overwritten.

**Figure 2. Subject ‘delivery 1’ on second page of Traffic and Transport Survey**

Number plate: STX235      Week: 18      DELIVERY 1      Distance travelled: 1080 km

What are the carriage returns on this delivery?  
In whole euros, excluding VAT and extras like toll, transport taxes, ferriage etcetera.  
 In case of **combined deliveries of goods** enter the total carriage returns.

enter delivery\*      Distance delivery in km\*      750

**Place of loading (goods) delivery 1 ?**

Land code (FB)\*      NL      double-click or <FB> to start lookup list  
 Nederland

Place-name (FB)\*      Heerlen      Type of goods\*      Ethyleenchlorbromide

ZIP code digits      6412

Type of site\*      Gross weight delivered goods in tonnes      6,8

1. production/extraction       5. railway terminal  
 2. consumption/processing/retail business       6. airport  
 3. seaport       7. bonded warehouse  
 4. inland port       8. distribution/wholesale business  
 9. other (also truck/lorry stand)

**Place of unloading (goods) delivery 1 ?**

Land code (FB)\*      FR      see 1...9 next to Type of site\* above  
 Frankrijk

Place-name (FB)\*      Orange

ZIP code digits

Type of site\*      8

Mode of appearance\*      Carriage returns delivery\*      560

1. liquid bulk (unpackaged)       6. hanging goods  
 2. solid bulk       7. pre-slung goods  
 3. large freight containers (20 ft and more)       8. mobile, self-propelled units, including live animals  
 4. other freight containers, including rolling containers       9. other mobile units  
 5. palletised goods       10. other cargo types

*Working with the Modelib Editor*

Here are some major experiences regarding the use of *Modelib* for creating a flexible design.

- The *Modelib* offered practically all the flexibility we needed, thanks also to some enhancements introduced during the development of Blaise 4.6 (e.g. enriched text now everywhere possible).
- To fully exploit the possibilities a lot of practical exercise is needed.
- Once mastered, spacing out and tuning the question elements on the formpane remains time-consuming because of the often needed reshuffling, rephrasing and stressing of elements.
- For a flexible design quite a number of *Modelib styles* are needed. If properly ordered, however, they can save a lot of time.
- Although flexibility presupposes freedom of design, the availability of guidelines, standard elements and templates can be a great help.

## 4. Developing a standard model for testing purposes

Our next aim was to develop a prototype questionnaire that could suit our needs for testing layout and navigational issues. In short, it should make the carefully prepared testing programme pay off. As a starting point we chose to join the colour settings from the website of Statistics Netherlands. We also wanted to use the company logo as an identifying mark in some way. For a versatile design we had to cope with the fact that the length of both question texts and explanations or instructions can vary a lot. A basic assumption was that help should be at hand, whenever possible. While reading down the page the respondent will usually focus on the data-entry fields. So this help should preferably be close to the fields. Too much text, however, may irritate. We therefore decided to use different panes. For a study on how to position explanations and instructions, see Jansen and Steehouder (2001).

A logical solution would be to use the infopane for explanations etcetera and position it on the right side of the screen. For a quick view of the page the respondent can then focus on the middle. Pane. By horizontally aligning with the data field the explanation or instructions for that field are readily available. The text on the infopane will move up and down, depending on the selected field. This can be a help in offering a visual cue to find the active data field. On the other hand, it may look strange if, with certain questions, there is nothing on the infopane.

The example in figure 3 is made with Blaise IS, using the XSL stylesheet for web use, and with some minor manual adjustments. A progress indicator is already available. What is shown on the (turquoise) infopane on the right is the lengthy explanation that belongs to the first question on the screen. It is only meant as an example of a body of text. The last field is automatically changed from 0 to 1 if the routing rules allow the respondent to continue. This OK-question is used to finish a particular subject. It may in due time be replaced by a button. Not all elements are yet fully balanced, but it's a start.

Figure 3. A webstyle based template, with the infopane on the right

Vacancies and Absentee rates - Microsoft Internet Explorer

### Vacancies and Absentee rates

Progress:

**Exit**

**Next Question**

**Previous Question**

**Calc Sheet**

**Next Page**

**Previous Page**

**First Page**

**Last Page**

**Info...**

Total number of vacancies at the end of the quarter\*

Number of vacancies being filled during the quarter\*

Number of vacancies that occurred during the quarter\*

Number of vacancies withdrawn during the quarter from being opened\*

OK: continue with [ENTER]

**EXPLANATION VACANCIES**  
A vacancy is a job where, inside or outside your company or institute, personnel is looked for, who can be placed immediately or as soon as possible.

The following vacancies also have to be counted:

- vacancies for which candidates have already applied; also if talks have already been going on with these applicants
- vacancies for which the selection procedure will take so much time that the actual entrance is not to be expected soon
- vacancies for temporary workers
- vacancies for students and other persons under instruction, provided that there is an employment contract (not unpaid trainee posts).

If, in case of reorganization or shrinkage, a vacant post can only be filled by employees whose job will disappear, please do **not** count this post as a vacancy.

With temporary and social employment agencies or job creation projects, only vacancies for the own staff should be counted.

In education, please do **not** count as vacation short fill-ins for illness.

As far as the designer is concerned, our first impression is that working with two panes next to each other is relatively easy. The division line between infopane and formpane can be shifted from page to page depending on the space needed. Even more space is available if the column with buttons is removed. This could be achieved by choosing a menu-based style like in the Blaise DEP. But if this space is used for question labels, answer lists etcetera they may not fit the narrower middle pane of the webstyle. If using the same datamodel for either style there will be some room left in the DEP on the left side. If only using the DEP it is worth considering to still use a left column. Financial surveys, for instance, often have questionnaires with modules or blocks that can be dealt with in different orders (likt tab sheets). The column could then be used to show an index and offer a view of the filling progress. Or there may still be some buttons if this proves useful for navigational reasons.

A problem in the DEP for which some kind of solution is required concerns the position of the infopane on the screen. If this position is *Right*, for instance, then the size of the pane on the screen is also determined from the right. As a consequence the pane will shrink to the right if a higher screen resolution is chosen. What is on the formpane, however, will shrink to the left. The information on the two panes will therefore move away from each other. This is shown in figure 4. Instead of the minimum resolution (800 by 600) we support and used so far, the page that is presented in figure 4 has a resolution of 1024 by 768.

**Figure 4. The template from figure 3 in DEP-style and with higher resolution**

GROUP 6	
<b>Total number of vacancies</b>	<b>4</b>
<b>of whom</b>	
work experience not required*	2
20 or more hours a week*	4
for jobs still manned*	1
shorter then one month looked for*	1
between one and three months	3
put employment advertisement*	4
job centre informed*	3
considered difficult to fill*	0

Number of vacancies that you consider as **difficult to fill**  
If not applicable to any vacancy, enter 0.

The questions displayed in figure 4 show the continuation of a Blaise table which was too wide to see all the columns without using the scroll bar. On the paper form the grid has 13 columns. With a minimum resolution there is only place for 5 on one screen. The rows of the table represent groups of vacancies. One row are vacancies for a particular kind of function within a certain municipality. The latter, together with function, required education and the number of people one will be in charge of, are entered in the table. The last question asks for the total number of vacancies in the group. After finishing the table more specific questions are asked per row or group of vacancies, for instance, on the number of vacancies (within the

group) one considers difficult to fill (the active last question in the figure above). As a matter of fact this could well be a much better way of asking these questions.

Finally, a table is presented in figure 5. It is a parallel in the DEP, put on a tab sheet, and meant as an electronic version of a paper sheet on which, as an auxiliary tool, one could calculate the absentee rate. The table can be used arbitrarily, except for obeying the answer types (e.g. date). Its only function is to enable the respondent to write things down to his own needs and have the absence days added up automatically. The parallel still has a second page on which is asked for the average number of employees during the reference period (a quarter). On the basis of the outcomes of the two pages two kind of percentages are calculated. The respondent is free to use these percentages, accepting them, applying a correction factor, or whatever is wanted.

With a wide table the infopane can only be on top or at the bottom of a page. The same is true if there would be added buttons (with a caption). In CASI there seems to be some need for these buttons, especially when using tables. In figure 5 one can see that an extra column is added just to make it possible for the respondent to let the DEP know that the calculations are completed. In the table shown such an explicit button-function, created by an extra column and an extra field, is almost essential. Otherwise the respondent may have used the tab Main Form to leave the table. But then he would not get the percentages. For these he first should have gone to the second page of the parallel. The only escape would have been the use of the next page option in the menu or on the speedbar.

**Figure 5. A calc sheet for the absentee rate as a parallel to the main parallel**

					Number of days in quarter	Absence through pregnancy/delivery Total number of days	Absence through 'normal' illness Total number of days	If calculation completed enter code 1 at the bottom
period [1]	from	21-12-2002	up to	21-01-2003	21	122	230	
period [2]	from		up to					
period [3]	from	21-01-2003	up to	21-02-2003	32	177	366	
period [4]	from		up to					
period [5]	from	21-02-2003	up to	21-03-2003	29	165	266	
period [6]	from		up to					
period [7]	from	21-03-2003	up to	21-04-2003	11	63	118	
period [8]	from		up to					
period [9]	from		up to			10		
period [10]	from		up to					
period [11]	from		up to					
period [12]	from		up to				11	
period [13]	from		up to					
<b>Totals</b>						537	991	1 - completed

If calculation **completed** enter 1 (= completed).  
**To return to the main form without completion** : click on tab **Main Form** or press **[Ctrl] + [Tab]** .

## 5. For the future

As a provisional figure, it seems that half of the transport businesses receiving the CASI software are actually using it for providing their data. As a beginning this is more than we had hoped for, given the heavy burden and the technically complex questionnaire. But it is also known that it may well be difficult to raise this initial

figure (once the decision is taken not to use CASI). Against this background it is important to show a feeling of understanding for what is going on in the world of transport and to give something in return whenever practicable.

While pre-visiting a limited number of businesses we noticed a remarkable willingness to cooperate and think along with us on how to improve CASI. They came out with smart ideas, especially on how the questionnaire could be better customized. The journey questionnaire for the transport survey, for instance, would be easier to understand if it did not have to cover such a wide range of different types of transport. A company that provides transport for a chain of supermarkets makes very similar journeys every day. A haulage business providing international transport, on the contrary, often has very differing types of journeys. Making all kinds of dedicated questionnaires may be impracticable with longer and more complex forms. A user created profile of settings, however, could simplify the filling process and enable us to anticipate better the special problems connected with the kind of business activities. It would be much easier to explain what should be considered a journey or a delivery on the basis of a typical example from the world of supermarkets, for instance. Automatic filling with default values, as already practised with the international trade survey, may also help.

One of the things that could be returned to businesses and what they are repeatedly asking for is a facility to print a report of what they just filled out, e.g. a report giving a good view of the fleet.

Among the obstacles for using CASI those occurring because a transport business has different establishments were the most striking. Sometimes a system manager was needed who was at headquarters far away. Regularly also one wanted to have the truck drivers themselves fill out the questions on their own journeys. And although EDR is multi-user, one apparently found it more difficult to organize the filling process than when paper forms could be distributed.

For the kind of business surveys discussed here we do not expect that the use of internet will expand rapidly. There are still problems of security with regard to data transmission and keeping data available on the less protected external network (to continue filling out forms after interruptions). But as it is, the internet can very well offer services such as detailed and up to date classifications together with sophisticated searching mechanisms. The internet is after all already the central mode to consult statistical information in The Netherlands. For experiences of Statistics Netherlands with different forms of using the internet and e-mail facilities for CASI data collection, see Roos (2002).

Although the DEP-style is menu-based, with CASI there seems to be a need for at least some buttons. As respondents get more experienced with the internet, they may also increasingly expect to find some buttons for navigation. For the more distant future it may even be considered to implement a web look-alike in the DEP, especially for reasons of navigation, using buttons together with captions ready at hand for the respondent. For CASI it is also important to have layouts that are not disproportionally distorted by the screen resolution of the user.

## References

Bethlehem, J. & Hoogendoorn, A. (2003): Methodological guidelines for Blaise web surveys. Paper presented at the 8<sup>th</sup> International Blaise Users' Conference, Copenhagen.

Bolster, G.W. de (2003): Electronic Data Reporter. Paper presented at the 8<sup>th</sup> International Blaise Users' Conference, Copenhagen, 2003.

Jansen C. & Steehouder M. (2001): How research can lead to better government forms. In: Reading and Writing Public Documents, Benjamins, Amsterdam/Philadelphia, pp. 11-36.

Roos, M. (2002): Methods of Internet data collection and implications for recruiting respondents. In: Statistical Journal of the United Nations Economic Commission for Europe, 19 (3), pp. 175-186

## Contacts

### **Survey Generator**

Carlo Vreugde and Jacques de Groot, VNG/SGBO/StimulansZ, The Netherlands  
carlo.vreugde@vng.nl

### **Blaise Questionnaire Text Editor (Qtxt)**

Grayson Mitchell, Statistics New Zealand  
grayson.mitchell@stats.govt.nz

### **Automatic Upgrade of Production Blaise Instruments**

Lilia Filippenko, Joseph Nofziger and Gilbert Rodriguez, RTI International  
grod@rti.org and lfilippenko@rti.org

### **Screen Layout Standards at Statistics Finland**

Vesa Kuusela, Survey Research Unit, Statistics Finland  
vesa.kuusela@stat.fi

### **Developing and updating screen layout and design standards**

Rebecca Gatward, Social Survey Division, Office for National Statistics  
Rebecca.Gatward@ons.gov.uk

### **Developing an optimal screen layout for CAI**

Fred Wensing, David Finlay, Jane Barresi, Australian Bureau of Statistics

### **Screen Design Guidelines for Blaise Instruments**

Sue Ellen Hansen, Survey Research Center, University of Michigan  
sehansen@umich.edu

### **Developing CASI standards at Statistics Netherlands**

Frans Kerssemakers, Statistics Netherlands  
fkr@cbs.nl

### **Methodological Guidelines for Designing Blaise Web Surveys**

Jelke Bethlehem & Adriaan Hoogendoorn, Free University of Amsterdam  
jbtm@cbs.nl and aw.hoogendoorn@scw.vu.nl

### **The usage of Blaise in an integrated application for the New Birth sample Survey in CATI mode**

Massimiliano Degortes, Stefania Macchia, Manuela Murgia, Istat, Italy  
Degortes@istat.it

### **Blaise at Statistics Netherlands**

Marien Lina, Statistics Netherlands  
c.na@cbs.nl

### **Displaying Chinese Characters In Blaise**

Gina-Qian Y.Cheung & Youhong Liu, University of Michigan  
qianyang@umich.edu

### **Analyzing Audit Trail Data in the 2002 National Survey on Drug Use and Health**

Michael Penne and Jeanne Snodgrass, RTI and Peggy Barker, SAMHSA  
LKY@rti.org

**A Random Walk Application for Blaise Instruments**

Gilbert Rodriguez and Jay R. Levinsohn, RTI International  
grod@rti.org

**First steps along the Audit Trail**

Tim Burrell, Office for National Statistics, UK  
tim.burrell@ons.gov.uk

**Creating Code to Convert Blaise to ASCII for SAS Input (Using the Blaise API within Visual Basic)**

Roger Schou, National Agricultural Statistics Service (NASS), USDA  
Roger\_Schou@nass.usda.gov

**Extracting data from a large instrument using the API**

Lois Steinfeldt, ARS, USDA  
lsteinfeldt@rbhnrc.usda.gov

**A Demonstration of the “Multiple Application Interface” with Blaise and Visual Basic**

Jim Hagerman & Hemant Kannan, The University of Michigan

**Electronic Data Reporter**

Gerrit de Bolster, Statistics Netherlands  
gblr@cbs.nl

**Integration of CAPI and CATI infrastructures at Statistics Canada**

Jacqueline Mayda, Operations Research and Development Division, Statistics Canada  
jackey.mayda@statcan.ca

**Blaise CATI at Mathematica**

Leonard Hart, Linda Bandeh, Doug Dougherty, and Carlo Caci, Mathematica Policy Research, Inc. USA  
LHart@Mathematica-mpr.com

**CATI Survey Management System**

Leif Bochis, Statistics Denmark  
lbn@dst.dk

**Interactive training for the interviewers using Blaise**

Hilde Degerdal, Statistics Norway  
hde@ssb.no

**Blaise for post-collection data editing - Building system for editing data entered from paper forms**

Pavle Kozjek, Statistical Office of Slovenia  
pavle.kozjek@gov.si

**The melting pot – practical experience of scanning paper forms**

Marianne Troelsen & Lars Pedersen, Statistics Denmark  
mbt@dst.dk and lap@dst.dk

**Lessons learned on the development of the Unified Enterprise Survey**

Rock Lemay, Statistics Canada  
rock.lemay@statcan.ca

**The American Time Use Survey Data Collection Instrument**

Karen Bagwell & David Altvater, US Bureau of the Census  
karen.ann.bagwell@census.gov and david.edward.altvater@census.gov

**Using non-Latin alphabets in Blaise**

Rob Groeneveld, Statistics Netherlands  
rgnd@cbs.nl

**Automated dialers for CATI systems**

Dan J. Bernard, Marketing Systems Group, USA  
dbernard@m-s-g.com