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 Hoft, 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 al 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 known 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. Te 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 adapt ions.

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 chararcters. 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:

```
                                        {b02            }
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 challanges for the Blaise Generator was the implementation of the routing in a Survey. The Blaise Generator asks the researcher from which answer possibilty the routing will take place and at which question the routing has to stop. This means that the generated routing sourccode in the RULES should account for multiple routings, not only after one another but also nested in one another and with overlaping 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 vairables. If that is the case that question will receive the IF-statement in the RULES when that question may be anwered with the specified condition. Counters will keep track of the nested IF-statements and the overlaping 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 seperately with the appriopiate conditions and also where to go to or where it comes from. If during the proces 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 al 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 al 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.

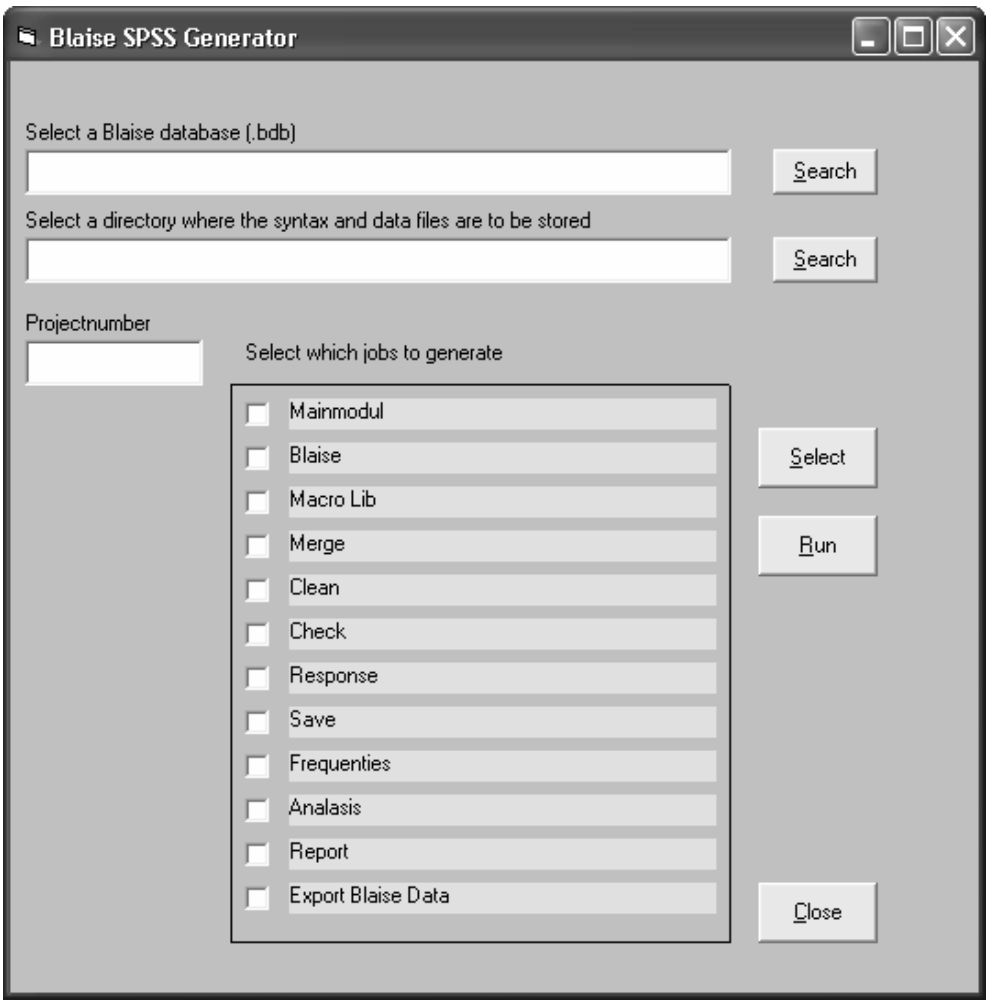
Here is a Code snippet from the Visual Basic macro:

```
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 Libary}"
    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, TypeNr & " = "
   Print #1, " " & TypeCat
Next i
Print #1,
Print #1, "{End Types}."
Close
Application.ScreenUpdating = True
Set sh = Nothing
End Sub
```

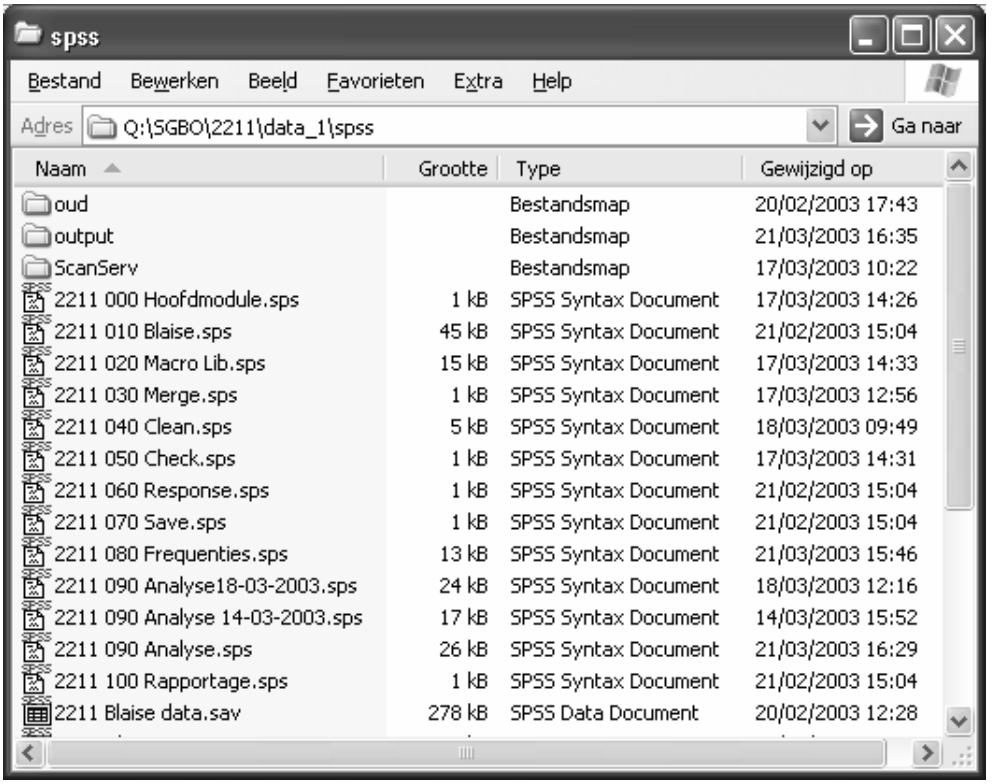
5. Blaise SPSS Generator

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 whichs 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 al 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 asci data is done automatically by using a pointer starting in record one until the last record. This creates the asci.txt data set wich can be imported into spss.

The biggest advantage of this generator is that routine jobs are automaticallyt 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-datag 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.