
IBUC 97

4th International Blaise

Users Conference

Paris France

May 1997

Table of Contents

Automated Paper Documentation of Blaise III Datamodels <i>Steve Anderson, Social Survey Division, Office for National Statistics , UK</i>	1
Bascula : Current Status and Future Developments <i>Jelke Bethlehem, Statistics Netherlands</i>	20
Advantages and disadvantages of migrating Blaise 2.5-based survey systems to Blaise III <i>Leif Bochis, Statistics Denmark</i>	68
New approaches to data processing integration in North Rhine-Westphalian Statistics based on BLAISE III <i>Markus Broose, Frank Merks, Thomas Pricking, data processing and statistical office of the State of North Rhine Westphalia - Germany</i>	73
An evaluation of computer-assisted occupation coding : results of a field trial <i>Diane Bushnell, Office for National Statistics, UK</i>	90
Le questionnaire d'une enquête en deux phases avec tirage d'un échantillon représentatif. <i>François Clanché, INSEE, France</i>	100
Interactive coding of economic activity using trigram search in BLAISE III <i>Omar S. Hardarson, Statistics Iceland</i>	117
Experiences of Two Capi-Papi -- Comparisons in Finland <i>Markku Heiskanen, Kirsti Ahlqvist, Statistics Finland</i>	128
An Office Management System in Blaise III <i>Merilyn Henden, Fred Wensing, Ken Smith, Mano Georgopoulos Australian Bureau of Statistics</i>	147
Integrating Surveys with Blaise III <i>Fred Heuvelmans, Frans Kerssemakers, Jeroen Winkels, Statistics Netherlands</i>	163
Some Thoughts About a Metadata Management System <i>Jean-Pierre Kent and Maarten Schuerhoff - Statistics Netherlands</i>	175
Documenting questionnaires <i>Jean-Pierre Kent and Leon Willenborg, Statistics Netherlands</i>	197
Manipula & ManiPlus Usage at the National Agricultural Statistics Service <i>David Knopf and Roger Schou, National Agricultural Statistics Service, USA</i>	206
An Object-Oriented Case Management System for CAPI Surveys <i>Vesa Kuusela, Anssi Parviainen, Statistics Finland</i>	216
Cati Call Management <i>Marien Lina, CBS, Nederland</i>	225
CapiPlus, un système intégré pour la réalisation des enquêtes auprès des ménages <i>Gilles Luciani, INSEE, France</i>	234
An integrated household survey in the UK: The Role of Blaise III in an Experimental Development by the Office For National Statistics <i>Tony Manners and Kirsty Deacon, Social Survey Division, Office for National Statistics, UK</i>	243
The Transition from Cases and BLAISE 2 to BLAISE III at the National Agricultural Statistics Service <i>Asa Manning, National Agricultural Statistics Service, USA</i>	264
SICORE, un outil et une méthode pour le chiffrage automatique à l'INSEE <i>Eric Meyer et Pascal Rivière, INSEE, France</i>	280
The Application of Blaise III to the Israel Labour Force Survey <i>Edith Noy and Gad Nathan, Central Bureau of Statistics, Jerusalem, Israel</i>	294
Making Blaise 2.5 Do Things It Was Never Meant to Do <i>James M. O'Reilly, Research Triangle Institute, USA</i>	303
Optimal Screen Design in Blaise <i>Mark Pierzchala, Westat, Inc., USA</i>	314

Automated Paper Documentation of Blaise III Datamodels

Steve Anderson, Social Survey Division, Office for National Statistics , UK

Abstract

CAPI questionnaires are ideally “self-documenting” - that is, someone who is familiar with Blaise can understand them. However, there are many users of the information contained in a datamodel, and some are more comfortable with paper documents similar to PAPI questionnaires.

Existing Blaise tools cannot create such documents, since they do not make routing information available. So each project within SSD has managed the problem in a different way, some of which involve a heavy manual element.

The paper discusses the different kinds of documentation required, the methods used to manage the problem in the past, and the development of a tool which automates the task.

1. Introduction

The use of CAPI has simplified many parts of the survey process. Using the Blaise language a datamodel specification can be built up from small blocks into a very large instrument - often much larger than might have been attempted in a non-CAPI project.

As the specification grows in size, the relationships between variables can become more complex and difficult to understand - until the project fails because the data is no longer comprehensible. It could be said that CAPI becomes a victim of its success.

In many respects, the Blaise language is self-documenting. The basic language elements are easily mastered, and there is a facility to add an explanatory text wherever a cryptic expression might appear, such as compound conditional statements and elaborate checks.

Even so, sheer size can defeat the attempt to understand the details of a specification even for an expert with the Blaise language, and such expertise is not always available to the end users of the data.

The client or end user is not the only person in need of some form of additional documentation of the data collected using CAPI methods.

The researcher or other specialist who creates the datamodel specification will wish to verify that variables have been collected under appropriate conditions. Trainers and Field support staff will require reference material based on the survey instrument, but usually in paper form. Secondary analysts and data archives will wish to have the same information as the original clients.

Because of this the provision of paper documentation is always a high priority, and is often a mandatory condition of contracts with clients.

The Blaise metadata tools (Setup Generator in Blaise 2, Cameleon in Blaise III) do not provide access to information on data relationships or routing. In Social Survey Division (SSD) of the UK Office for National Statistics, therefore, each project has adopted a different approach to the problem of documenting their CAPI instruments.

One of the commonest methods used was semi-automated: starting with the output from the Blaise 2 'Paper Questionnaire' (or 'PAPI') utility, if routing was required the researcher would add it in a word processor after inspecting the specification source.

Another popular method started directly from the specification source. All unwanted material was removed. Usually this meant removing everything except the contents of QUEST paragraphs, but if routing was required it could be added by editing the ROUTE paragraph to make it more readily understood by people who did not know Blaise.

In both these methods, so much effort would be required to document the routing that it was often left out altogether.

Very often the full procedure was only used at the start of the project. For subsequent versions of the instrument changes were made directly to the corresponding document. This invites the possibility that the documentation will drift out of date with the instrument, even without the inherently error-prone nature of manual preparation.

These methods are expensive in skilled personnel. The work is often impossible to delegate to clerical staff, because of the need either for Blaise expertise or familiarity with the particular datamodel specification.

They are also very expensive in time, and - if any further argument against manual preparation is needed - it is often the case that the documentation is required for discussion with clients at key phases of the project, or for briefing of interviewers. The up-to-date paper document must be available soon after the datamodel specification is finalised.

All of the information required is buried in the datamodel specification, so it is natural to look for an automated method of extracting and presenting it.

SSD and others encouraged Statistics Netherlands to develop a suitable automatic documentation tool, but because of the Blaise III development

there was insufficient resource to take on the project. Statistics Netherlands did offer to provide some help and advice if the tool was developed within SSD and so on that basis the project was opened.

The analysis for the project involved two strands: determining the desired output format, and understanding the mechanics of locating and extracting the information to be output from the metadata. Development would then proceed by translating this algorithm into software and adding a user interface to control it.

In the Blaise 2 system, routing information is not stored in an accessible form within the metadata files; it becomes part of the intermediate source code which is generated by the syntax checker and then compiled by the Turbo Pascal compiler. It would be necessary to walk through the datamodel specification source files to gather the routing information, in effect replicating the Blaise 2 syntax checker but storing the routing information in a usable format.

Initial discussions suggested that, whereas for Blaise 2 this would be unavoidable, for Blaise III it would be much easier and more future-proof to access the metadata directly using the library of object handlers already created for the Control Centre, the Data Entry Program, Cameleon, and the rest of the Blaise III system toolset.

Because the methods required to implement the documenter tool for the two versions of Blaise would have been quite different - to the extent of being two separate tools with very little in common between them - it was decided to leave aside the development of a Blaise 2 tool. Separate development would have been too expensive, and in any case it was expected that most projects would be migrating to Blaise III as soon as possible.

SSD accepted the offer of assistance from Statistics Netherlands, and I spent two days at Voorburg discussing the new object-oriented methods of manipulating metadata in Blaise III. Thanks are due to Maarten Schuerhoff for facilitating the discussions and to Jean-Pierre Kent for patiently explaining the mechanics of the metadata objects and for providing the expression-to-text delegator object.

During analysis some important features were noted as desirable. Users of the tool should not have to concern themselves with the internals of the metadata files, the tool should not be sensitive to changes in the Blaise language; finally, the output format should be flexible and configurable.

The actual implementation breaks the tool into 2 separate programs, an Object Pascal program linked with the Statistics Netherlands object library, which collects the metadata required, and a Clipper program which presents it. A set of intermediate datafiles is used to communicate between the two programs.

This separation of functions makes the tool easier to maintain. There is more expertise with Clipper than with Object Pascal within SSD, and whereas the metadata reader is likely to be quite stable, the presentation software will need to be flexible to meet future requirements.

Initially the project was concerned with data dictionaries and code re-use. However, it was quickly redirected towards the production of paper documentation. One aim of the original project was retained, however: a variable should be understandable in isolation, with all of the conditions under which it is asked collected together.

2. Technical description

In the metadata files, the syntactic elements of the datamodel specification are stored as objects. Looking at the RULES section, for instance, an IF statement is stored as an MetaIfStatement, which is derived from a more general MetaStatement object.

The MetaIfStatement object is built from other objects, such as the IfExpression expression object which represents the condition part of the IF statement, and the ThenPart statement object which represents all the statements to be followed if the expression is true.

With this in mind, the algorithm to collect routing information for a particular variable can be summarised as follows :

- Read the syntactic objects from the metadata files in order - they are retrieved in the same order that they were declared in the RULES sections of the datamodel specification.
- Whenever a condition is found - an IF object - make a note of the condition expression on a push-down stack. When the condition statement falls out of scope - i.e. at the point where the ENDIF keyword would be found in the specification - remove the corresponding condition from the stack.
- Whenever a variable is found record the details along with a record of all the conditions still on the stack, which are therefore still in force at the time this variable appears in the RULES.

To document a variable, dump the expressions or substitution texts of all of the conditions under which it is asked, plus the variable label, variable type, and if applicable the value labels.

The current presentation program allows the user to select: specific blocks to document; which components of the RULES should appear in the document (variables, checks and signals, computations/edits), whether fully-qualified variable names should be used, use of short value labels; an alphabetical index to all variables, an appended structure diagram.

3. Concluding notes

Using this tool, a task that used to take a week and was error-prone now takes less than five minutes and is accurate, and it is easy to keep in step with changes to the specification. The document is derived directly from the metadata so it can be used to verify that the requirements of the client have been accurately coded into the datamodel. All relevant information

about each variable is collected together, making it ideal for archiving alongside the data.

This program creates plain text files. Work has been started on another presentation program which creates a source file for the Windows Help system. Once compiled, the resulting help file can be used to explore the structure of a datamodel in a number of different ways, using the Windows Help engine.

Another variant will create a Rich Text Format file suitable for direct import into Microsoft Word. This allows greater control over formatting - for example allowing the use of the bold attribute, or different font sizes.

The documentation tool has been tested by a small number of organisations outside SSD. Future development, and the provision of similar functionality within Blaise by Statistics Netherlands, are matters currently under discussion.

Appendix - Sample output

Documentation of Questionnaire/Module 'OMN9511'
Created on 12-01-1997 at 10:30

ask always:

OMN9511.MainMods.M130.m130_1

THIS MODULE IS FOR THE DEPARTMENT OF HEALTH
Which of these do you think causes the most premature deaths in
the UK each year?

SHOW CARD C130.1

- | | |
|-------------|-------------------------|
| (1) roadacc | Road accidents |
| (2) workacc | Accidents at work |
| (3) AIDS | AIDS |
| (4) qlsmoke | Smoking |
| (5) murder | Murder and manslaughter |
| (6) drugs | Illicit drugs |
| (7) alcohol | Alcohol misuse |

ask always:

OMN9511.MainMods.M130.m130_2

Do you smoke cigarettes at all nowadays?

- | | |
|---------|-----|
| (1) Yes | Yes |
| (2) No | No |

ask if: m130_2 = Yes

OMN9511.MainMods.M130.m130_3

How many cigarettes a day do you usually smoke at weekends?

0..200

ask if: m130_2 = Yes

OMN9511.MainMods.M130.m130_4

How many cigarettes a day do you usually smoke on weekdays?

0..200

ask if: m130_2 = Yes

OMN9511.MainMods.M130.m130_5

Do you mainly smoke filter tipped cigarettes, or plain cigarettes or hand-rolled cigarettes?

- | | | |
|-----|--------|---------------|
| (1) | filter | Filter tipped |
| (2) | plain | Plain |
| (3) | hand | Hand-rolled |

ask if: NOT (m130_2 = Yes)

OMN9511.MainMods.M130.m130_6

Have you ever smoked cigarettes regularly?

- | | | |
|-----|-----|-----|
| (1) | Yes | Yes |
| (2) | No | No |

ask if: NOT (m130_2 = Yes)

and: m130_6 = Yes

OMN9511.MainMods.M130.m130_7

About how many cigarettes a day did you smoke when you smoked regularly?

0..200

ask always:

OMN9511.MainMods.M130.m130_8

Do you smoke at least one cigar of any kind per month nowadays?

- | | | |
|-----|-----|-----|
| (1) | Yes | Yes |
| (2) | No | No |

ask if: HHGrid.HH1.Hg[HHGrid.RESP].Sex = Male

OMN9511.MainMods.M130.m130_9

Do you smoke a pipe at all nowadays?

- | | | |
|-----|-----|-----|
| (1) | Yes | Yes |
| (2) | No | No |

ask if: m130_2 = Yes

OMN9511.MainMods.M130.m130_10

How soon after waking do you smoke your first cigarette of the day?

- | | | |
|-----|----------|------------------------------|
| (1) | less5 | Less than 5 minutes |
| (2) | n5to14 | 5-14 minutes |
| (3) | n15to29 | 15-29 minutes |
| (4) | n30to1hr | 30 mins but less than 1 hour |
| (5) | n1to2hrs | 1 hour but less than 2 hours |
| (6) | n2ormore | 2 hours or more |

ask if: ((m130_2 = Yes) OR (m130_8 = Yes)) OR (m130_9 = Yes)

OMN9511.MainMods.M130.m130_11

Would you like to give up smoking?

- | | | |
|-----|-----|-----|
| (1) | Yes | Yes |
| (2) | No | No |

ask if: ((m130_2 = Yes) OR (m130_8 = Yes)) OR (m130_9 = Yes)
and: m130_11 = Yes

OMN9511.MainMods.M130.m130_12

How much would you like to give up smoking...

RUNNING PROMPT

- | | | |
|-----|-------|------------------|
| (1) | litle | a little |
| (2) | fair | a fair amount |
| (3) | qulot | quite a lot |
| (4) | vmuch | very much indeed |

ask if: ((m130_2 = Yes) OR (m130_8 = Yes)) OR (m130_9 = Yes)
and: m130_11 = Yes

OMN9511.MainMods.M130.m130_13

What are your main reasons for wanting to give up?

SHOW CARD C130.13

SET [3] OF

- | | | |
|-----|--------|---|
| (1) | beter | better for my health not to smoke |
| (2) | doctr | doctor said I should stop |
| (3) | famly | family/friends wanted me to stop |
| (4) | finan | financial reasons (couldn't afford it) |
| (5) | preg | pregnancy |
| (6) | worr | worried about the effect on my children |
| (7) | q13oth | other (specify) |

ask if: ((m130_2 = Yes) OR (m130_8 = Yes)) OR (m130_9 = Yes)
and: m130_11 = Yes
and: q13oth IN m130_13

OMN9511.MainMods.M130.spec13

Please specify other reasons

OPEN

ask if: ((m130_2 = Yes) OR (m130_8 = Yes)) OR (m130_9 = Yes)
and: m130_11 = Yes

OMN9511.MainMods.M130.m130_14

If you tried to give up in, say, the next three months, how likely do you think you would be to succeed?

SHOW CARD C130.14

- | | | |
|-----|---------|-----------------|
| (1) | vlike | Very likely |
| (2) | flike | Fairly likely |
| (3) | funlike | Fairly unlikely |
| (4) | vunlike | Very unlikely |
| (5) | q14nsur | Not sure |

ask if: ((m130_2 = Yes) OR (m130_8 = Yes)) OR (m130_9 = Yes)

OMN9511.MainMods.M130.m130_15

Have you made a serious attempt to give up smoking in the last five years, that is since ^DAT.THISMTH ^DAT.YEAR5?

- | | | |
|-----|-----|-----|
| (1) | Yes | Yes |
| (2) | No | No |

ask if: ((m130_2 = Yes) OR (m130_8 = Yes)) OR (m130_9 = Yes)
and: m130_15 = Yes

OMN9511.MainMods.M130.M130_16

How many times have you tried to stop (in the last 5 years)?

0..10

ask if: ((m130_2 = Yes) OR (m130_8 = Yes)) OR (m130_9 = Yes)
and: m130_15 = Yes

OMN9511.MainMods.M130.M130_17

What is the longest period you've managed to give up for (in the last five years)?

- | | | |
|-----|---------|-------------------------------|
| (1) | fdays | A few days - less than a week |
| (2) | aweek | About a week |
| (3) | n2to3wk | About two or three weeks |
| (4) | amth | About a month |
| (5) | mmth | More than a month |

ask if: ((m130_2 = Yes) OR (m130_8 = Yes)) OR (m130_9 = Yes)
and: m130_15 = Yes
and: M130_17 = mmth

OMN9511.MainMods.M130.SPEC17

Please specify the number of months or years

STRING[20]

ask if: m130_6 = Yes

OMN9511.MainMods.M130.m130_18

How long ago is it since you stopped smoking cigarettes?

PLEASE RECORD WHETHER YEARS MONTHS OR WEEKS AGO AND ACTUALLY
HOW LONG AT NEXT QUESTION
IF LESS THAN A WEEK RECORD AS ZERO WEEKS

- | | | |
|-----|--------|--------|
| (1) | yeers | years |
| (2) | munths | months |
| (3) | weaks | weeks |

ask if: m130_6 = Yes

OMN9511.MainMods.M130.INT18

How many ^M130_18 ago was this
ASK OR CODE THE NUMBER HERE

1..99

ask if: m130_6 = Yes

OMN9511.MainMods.M130.m130_19

Was that your first serious attempt at stopping, or had you
tried to give up smoking cigarettes before then?

- | | | |
|-----|-------|---------------|
| (1) | first | First attempt |
| (2) | tbe4 | Tried before |

ask if: m130_6 = Yes

OMN9511.MainMods.M130.m130_20

On the whole, did you find it easy or difficult to give up
smoking cigarettes? Was it...
RUNNING PROMPT

- | | | |
|-----|---------|-------------------------|
| (1) | veasy | Very easy |
| (2) | feasy | Fairly easy |
| (3) | fdiff | Fairly difficult |
| (4) | vdiff | Very difficult |
| (5) | q20nsur | Not sure/can't remember |

ask if: m130_6 = Yes

OMN9511.MainMods.M130.m130_21

What were your main reasons for wanting to give up smoking cigarettes?

SHOW CARD C130.13

SET [3] OF

- | | | |
|-----|--------|---|
| (1) | beter | better for my health not to smoke |
| (2) | doctr | doctor said I should stop |
| (3) | famly | family/friends wanted me to stop |
| (4) | finan | financial reasons (couldn't afford it) |
| (5) | preg | pregnancy |
| (6) | worr | worried about the effect on my children |
| (7) | q13oth | other (specify) |

ask if: m130_6 = Yes

and: q13oth IN m130_21

OMN9511.MainMods.M130.spec21

Please specify other reasons

OPEN

ask if: (m130_2 = Yes) OR ((m130_6 = Yes) AND (QUITAGO < 5))

OMN9511.MainMods.M130.m130_22a

In the last 5 years, have you been given advice on smoking by your GP?

- | | | |
|-----|-----|-----|
| (1) | Yes | Yes |
| (2) | No | No |

ask if: (m130_2 = Yes) OR ((m130_6 = Yes) AND (QUITAGO < 5))

OMN9511.MainMods.M130.m130_22b

(In the last 5 years, have you been given advice on smoking by someone else who works at the surgery or health centre?)

- | | | |
|-----|-----|-----|
| (1) | Yes | Yes |
| (2) | No | No |

ask if: (m130_2 = Yes) OR ((m130_6 = Yes) AND (QUITAGO < 5))

OMN9511.MainMods.M130.m130_22c

(In the last 5 years, have you been given advice on smoking by a pharmacist?)

- | | | |
|-----|-----|-----|
| (1) | Yes | Yes |
| (2) | No | No |

ask if: (m130_2 = Yes) OR ((m130_6 = Yes) AND (QUITAGO < 5))

OMN9511.MainMods.M130.m130_22d

(In the last 5 years, have you been given advice on smoking by)
any other medical person?

(1)	Yes	Yes
(2)	No	No

ask if: (m130_2 = Yes) OR ((m130_6 = Yes) AND (QUITAGO < 5))
and: m130_22d = Yes

OMN9511.MainMods.M130.SPEC22d

Please specify who the other medical person was

OPEN

ask if: (m130_2 = Yes) OR ((m130_6 = Yes) AND (QUITAGO < 5))

OMN9511.MainMods.M130.M130_23

(On any of these occasions) Did you have a discussion about
giving up smoking, or were you just given something to take
away and read?

IF BOTH CODE 1 FOR DISCUSSION

(1)	discus	discussion
(2)	liter	literature only

ask if: (m130_2 = Yes) OR ((m130_6 = Yes) AND (QUITAGO < 5))

OMN9511.MainMods.M130.m130_24

Did you find the advice helpful?

(1)	Yes	Yes
(2)	No	No

ask if: HHGrid.HH1.Hg[HHGrid.RESP].Marstat IN [Marr, Cohabit,
SameGen]

OMN9511.MainMods.M130.M130_25

Does your partner smoke?

(1)	Yes	Yes
(2)	No	No

ask if: HHGrid.HH1.Hg[HHGrid.RESP].Marstat IN [Marr, Cohabit,
SameGen]

and: M130_25 = Yes

OMN9511.MainMods.M130.M130_26

How do you feel about your partner smoking?

SHOW CARD C130.26

- | | | |
|-----|---------|--|
| (1) | astrong | Approves strongly |
| (2) | approv | Approves |
| (3) | napprov | Neither approves nor
disapproves/doesn't mind |
| (4) | dapprov | Disapproves |
| (5) | dstrong | Disapproves strongly |

ask if: HHGrid.HH1.Hg[HHGrid.RESP].Marstat IN [Marr, Cohabit,
SameGen]

and: ((m130_2 = Yes) OR (m130_8 = Yes)) OR (m130_9 = Yes)

OMN9511.MainMods.M130.m130_27

How does your partner feel about you smoking?

SHOW CARD C130.26

- | | | |
|-----|---------|--|
| (1) | astrong | Approves strongly |
| (2) | approv | Approves |
| (3) | napprov | Neither approves nor
disapproves/doesn't mind |
| (4) | dapprov | Disapproves |
| (5) | dstrong | Disapproves strongly |

ask always:

OMN9511.MainMods.M130.m130_28a

Do you think the government should or should not allow tobacco
advertising...
on boardings and posters?

- | | | |
|-----|--------|------------------|
| (1) | sallow | should allow |
| (2) | nallow | should not allow |

ask always:

OMN9511.MainMods.M130.m130_28b

(Do you think the government should or should not allow tobacco
advertising...)
in places where cigarettes are sold?

- | | | |
|-----|--------|------------------|
| (1) | sallow | should allow |
| (2) | nallow | should not allow |

ask always:

OMN9511.MainMods.M130.m130_28c

(Do you think the government should or should not allow tobacco advertising...)
in newspapers and magazines?

- | | | |
|-----|--------|------------------|
| (1) | sallow | should allow |
| (2) | nallow | should not allow |

ask always:

OMN9511.MainMods.M130.m130_28d

(Do you think the government should or should not allow tobacco advertising...)
of any kind at all?

- | | | |
|-----|--------|------------------|
| (1) | sallow | should allow |
| (2) | nallow | should not allow |

ask always:

OMN9511.MainMods.M130.m130_29a

How do you feel about tobacco companies being given publicity in return for sponsoring or supporting...
sports events?

SHOW CARD C130.26

- | | | |
|-----|---------|---|
| (1) | astrong | Approves strongly |
| (2) | approv | Approves |
| (3) | napprov | Neither approves nor disapproves/doesn't mind |
| (4) | dapprov | Disapproves |
| (5) | dstrong | Disapproves strongly |

ask always:

OMN9511.MainMods.M130.M130_29b

(How do you feel about tobacco companies being given publicity in return for sponsoring or supporting...)
events such as pop concerts?

SHOW CARD C130.26

- | | | |
|-----|---------|---|
| (1) | astrong | Approves strongly |
| (2) | approv | Approves |
| (3) | napprov | Neither approves nor disapproves/doesn't mind |
| (4) | dapprov | Disapproves |
| (5) | dstrong | Disapproves strongly |

ask always:

OMN9511.MainMods.M130.M130_29c

(How do you feel about tobacco companies being given publicity in return for sponsoring or supporting...)
opera or other arts events?

SHOW CARD C130.26

- | | | |
|-----|---------|---|
| (1) | astrong | Approves strongly |
| (2) | approv | Approves |
| (3) | napprov | Neither approves nor disapproves/doesn't mind |
| (4) | dapprov | Disapproves |
| (5) | dstrong | Disapproves strongly |

ask always:

OMN9511.MainMods.M130.m130_30

How strongly would you approve or disapprove if the government increased the tax on cigarettes?

SHOW CARD C130.26

- | | | |
|-----|---------|---|
| (1) | astrong | Approves strongly |
| (2) | approv | Approves |
| (3) | napprov | Neither approves nor disapproves/doesn't mind |
| (4) | dapprov | Disapproves |
| (5) | dstrong | Disapproves strongly |

ask always:

OMN9511.MainMods.M130.M130_31

Do you think the government should increase the tax on cigarettes...

RUNNING PROMPT
CODE ONE ONLY

- | | | |
|-----|--------|--------------------------------------|
| (1) | mmore | much more than the rate of inflation |
| (2) | jabove | just above the rate of inflation |
| (3) | inline | only in line with inflation |
| (4) | natat | not at all |

ask always:

OMN9511.MainMods.M130.M130_32A

Do you think that living with someone who smokes does, or does not, increase a child's risk of...
asthma

- | | | |
|-----|---------|------------------------|
| (1) | increes | increases risk |
| (2) | notinc | does not increase risk |

ask always:

OMN9511.MainMods.M130.M130_32b

(Do you think that living with someone who smokes does, or does not, increase a child's risk of...)
glue ear?

- | | | |
|-----|---------|------------------------|
| (1) | increes | increases risk |
| (2) | notinc | does not increase risk |

ask always:

OMN9511.MainMods.M130.M130_32c

(Do you think that living with someone who smokes does, or does not, increase a child's risk of...)
diabetes?

- | | | |
|-----|---------|------------------------|
| (1) | increes | increases risk |
| (2) | notinc | does not increase risk |

ask always:

OMN9511.MainMods.M130.M130_32d

(Do you think that living with someone who smokes does, or does not, increase a child's risk of...)
cot death?

- | | | |
|-----|---------|------------------------|
| (1) | increes | increases risk |
| (2) | notinc | does not increase risk |

ask always:

OMN9511.MainMods.M130.M130_32e

(Do you think that living with someone who smokes does, or does not, increase a child's risk of...)
chest infections?

- | | | |
|-----|---------|------------------------|
| (1) | increes | increases risk |
| (2) | notinc | does not increase risk |

ask always:

OMN9511.MainMods.M130.M130_32f

(Do you think that living with someone who smokes does, or does not, increase a child's risk of...)
other infections?

- | | | |
|-----|---------|------------------------|
| (1) | increes | increases risk |
| (2) | notinc | does not increase risk |

ask always:

OMN9511.MainMods.M130.m130_33a

Do you think that breathing someone else's smoke increases the risk of a non-smoker getting... asthma?

- (1) increes increases risk
- (2) notinc does not increase risk

ask always:

OMN9511.MainMods.M130.m130_33b

(Do you think that breathing someone else's smoke increases the risk of a non-smoker getting...) lung cancer?

- (1) increes increases risk
- (2) notinc does not increase risk

ask always:

OMN9511.MainMods.M130.m130_33c

(Do you think that breathing someone else's smoke increases the risk of a non-smoker getting...) diabetes?

- (1) increes increases risk
- (2) notinc does not increase risk

ask always:

OMN9511.MainMods.M130.m130_33d

(Do you think that breathing someone else's smoke increases the risk of a non-smoker getting...) heart disease?

- (1) increes increases risk
- (2) notinc does not increase risk

ask always:

OMN9511.MainMods.M130.m130_33e

(Do you think that breathing someone else's smoke increases the risk of a non-smoker getting...) bronchitis?

- (1) increes increases risk
- (2) notinc does not increase risk

ask always:

OMN9511.MainMods.M130.m130_33f

(Do you think that breathing someone else's smoke increases the risk of a non-smoker getting...)
coughs and colds?

- | | | |
|-----|---------|------------------------|
| (1) | increes | increases risk |
| (2) | notinc | does not increase risk |

ask always:

OMN9511.MainMods.M130.m130_34a

How far do you agree or disagree that there should be restrictions on smoking...
at work?

SHOW CARD C130.34

- | | | |
|-----|----------|--|
| (1) | agstrong | agrees strongly |
| (2) | agree | agrees |
| (3) | nagree | Neither agrees nor
disagrees/doesn't mind |
| (4) | dagree | Disagrees |
| (5) | dgstrong | Disagrees strongly |

ask always:

OMN9511.MainMods.M130.m130_34b

(How far do you agree or disagree that there should be restrictions on smoking...)
in restaurants?

SHOW CARD C130.34

- | | | |
|-----|----------|--|
| (1) | agstrong | agrees strongly |
| (2) | agree | agrees |
| (3) | nagree | Neither agrees nor
disagrees/doesn't mind |
| (4) | dagree | Disagrees |
| (5) | dgstrong | Disagrees strongly |

ask always:

OMN9511.MainMods.M130.m130_34c

(How far do you agree or disagree that there should be restrictions on smoking...)
in pubs?

SHOW CARD C130.34

- | | | |
|-----|----------|--|
| (1) | agstrong | agrees strongly |
| (2) | agree | agrees |
| (3) | nagree | Neither agrees nor
disagrees/doesn't mind |
| (4) | dagree | Disagrees |
| (5) | dgstrong | Disagrees strongly |

ask always:

OMN9511.MainMods.M130.m130_34d

(How far do you agree or disagree that there should be
restrictions on smoking...)
in public places such as banks and post offices?

SHOW CARD C130.34

- | | | |
|-----|----------|--|
| (1) | agstrong | agrees strongly |
| (2) | agree | agrees |
| (3) | nagree | Neither agrees nor
disagrees/doesn't mind |
| (4) | dagree | Disagrees |
| (5) | dgstrong | Disagrees strongly |

ask if: (M1.PaidWork = Yes) OR (M1.Away = Yes)

OMN9511.MainMods.M130.M130_35

What sort of restrictions are there on smoking where you work?

SHOW CARD C130.35

- | | | |
|-----|--------|---|
| (1) | nosmok | No smoking at all on the premises |
| (2) | smok | Smoking only allowed in designated
smoking rooms |
| (3) | norest | No restrictions at all. |
| (4) | dwork | Don't work in a building with
other people |

Bascula : Current Status and Future Developments

Jelke Bethlehem, Statistics Netherlands

1. Introduction

1.1. About errors in surveys

Our complex society experiences an ever growing demand for statistical information relating to social, demographic, industrial, economic, financial, political, and cultural situation of the country. Such information enables policy makers and others to take informed decisions for a better future. Sometimes, statistical information can be retrieved from administrative sources. More often there is a lack of such sources. In that case, the sample survey is a powerful instrument to collect new statistical information.

A sample survey collects information on only a small part of the population. In principle, this sample only provides information about the selected elements of the population. Nevertheless, if the sample is selected using a proper sampling design, it is also possible to make inference about the population as a whole. Data about the sample elements can be used to compute estimates of population characteristics.

Estimates will never exactly equal to the population characteristics to be estimated. There will always be some error. This error can have many causes. Two broad categories can be distinguished: sampling errors and non-sampling 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. The sample is selected by means of 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 magnitude of the sampling 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, the error in the estimate can be reduced.

Non-sampling errors occur even if the whole population is investigated. Non-sampling errors are errors made during the process of recording the answers to the questions. An important source of non-sampling errors is

non-response. It is the phenomenon that sample elements do not provide the required information. There may be various reasons for this: refusal to co-operate, not at home at the time of the visit of the interviewer, or not able to co-operate due to illness or other circumstances.

Due to non-response the sample size will be smaller than planned. This may result in increased variances of population estimates. Another, more serious, consequence is that non-response may be selective. This happens if non-response causes some groups in the population to be over- or under-represented in the sample, and these groups behave differently with respect to the population characteristics to be investigated. For example, if people with high incomes refuse to co-operate in the survey, the estimate of the mean income in the population will be too low.

1.2. Weighting sample surveys

There is ample evidence that non-response often causes estimates to be biased. This means that something has to be done to correct for this bias. A frequently used technique is *weighting*. Weighting is based on the use of *auxiliary information*. Auxiliary information is defined as a set of variables that have been measured in the survey, and for which information on the population distribution is available. By comparing the population distribution of an auxiliary variable with its sample distribution, it can be assessed whether or not the sample is representative for the population (with respect to this variable). If both distributions differ considerably, one must conclude that non-response has resulted in a selective sample.

The auxiliary information can also be used to compute adjustment weights. Weights are assigned to all records of observations. Estimates of population characteristics can now be obtained by using the weighted values instead of the unweighted values. The weights are defined in such a way that population characteristics for the auxiliary variables can be computed without error. Then the weighted sample is said to be *representative* with respect to the auxiliary variables used.

If it is possible to make the sample representative with respect to several auxiliary variables, and if these variables have a strong relationship with the phenomena to be investigated, then the (weighted) sample will also be (approximately) representative with respect to these phenomena, and hence estimates of population characteristics will be more accurate.

The Blaise environment contains a tool for computing adjustment weights. It is the program *Bascula*. Using the file with sample data and a file with the population distribution of auxiliary variables, *Bascula* can compute these adjustment weights using a number of different techniques. The most simple and straightforward one is *post-stratification*. Due to lack of a sufficient amount of population information, or empty or small sample cells, post-stratification is not always applicable. For such a situation, *Bascula* offers two alternatives: *linear weighting* and *multiplicative weighting*.

The purpose of this paper is to describe the current status and the future developments of *Bascula*. Section 2 gives an overview of the theoretical

background. Section 3 describes how Bascula can be used within the Blaise environment. Section 4 gives discusses some future developments that will lead to a new, enhanced version of Bascula.

Bascula is being developed by two departments of Statistics Netherlands. The Department of Statistical Methods is responsible for the theoretical framework, and specific weighting algorithms. The Statistical Informatics Department is in charge of the user-interface and the integration in the Blaise system.

2. Weighting techniques

2.1. Basic concepts of sampling

In order to be able to explain the theoretical background of weighting, first some basic definitions and notations of sampling theory are introduced.

Let the finite *target population* U consist of a set of N identifiable elements, which may be labelled $1, 2, \dots, N$. Associated with each element i is a unknown value Y_i of the *target variable*. The vector of all values of the target variable is denoted by

$$Y = (Y_1, Y_2, \dots, Y_N)' \quad (2.1.1)$$

Objective of the sample survey is assumed to be estimation of the population total :

$$Y_T = \sum_{i \in U} Y_k \quad (2.1.2)$$

To estimate this population parameter, a sample of size n is selected. It is assumed throughout this paper that samples are selected without replacement. In this case, the sample can be represented by a subset s of U ($s \subset U$). The sample size is equal to the number of elements in s . The *first order inclusion probability* of element i in the target population is defined as :

$$\pi_i = P(i \in s) \quad (2.1.3)$$

It is the probability that the sample s contains element i . In case of a simple random sample, all inclusion probabilities are equal to n/N .

A straightforward way to estimate the population total of the target variable is to apply the estimator defined by Horvitz-Thompson (1952) :

$$y_{HT} = \sum_{i \in s} \frac{Y_i}{\pi_i} \quad (2.1.4)$$

The Horvitz-Thompson estimator (2.1.4) is an unbiased estimator of the population total. This estimator does not make any use of auxiliary information. However, such information can be used to improve

estimators. Bascula offers three ways of incorporating auxiliary information in the estimation procedure. These three approaches are post-stratification, linear weighting and multiplicative weighting. They will be discussed in the following subsections.

2.2. Post-stratification

Post-stratification is a well known and often used weighting method. To be able to carry out post-stratification, one or more qualitative auxiliary variables are needed.

Suppose, we have an auxiliary variable X having H categories. So it divides the population into H strata. The strata are denoted by the subsets U_1, U_2, \dots, U_H of the population U . The number of population elements in stratum U_h is denoted by N_h , for $h=1, 2, \dots, H$. The population size N is equal to $N=N_1+N_2+\dots+N_H$.

A sample of size n is selected from the population. The set of sample elements in stratum h is denoted by s_h . If n_h denotes the number of sample elements in the sub-sample s_h (for $h=1, 2, \dots, H$), then $n=n_1+n_2+\dots+n_H$. Note that the values of the n_h are the result of a random selection process. So, they are random variables.

Post-stratification assigns identical correction weights to all elements in the same stratum. The correction weight d_i for an element i in stratum U_h is in its most general form defined by

$$d_i = \frac{N_h}{\sum_{j \in s_h} \frac{1}{\pi_j}} \quad (2.2.1)$$

where the sum is taken over all sample elements j in the stratum U_h , i.e. all elements in sub-sample s_h . In case of a simple random sample, all inclusion probabilities π_i are equal to n/N , and the correction weight d_i for an element i in stratum U_h reduces to

$$d_i = \frac{N_h n}{N n_h} \quad (2.2.2)$$

If the values of the inclusion probabilities and correction weights are imputed in expression (2.2.1), the result is the well known post-stratification estimator

$$y_p = \sum_{h=1}^H N_h \bar{y}_h \quad (2.2.3)$$

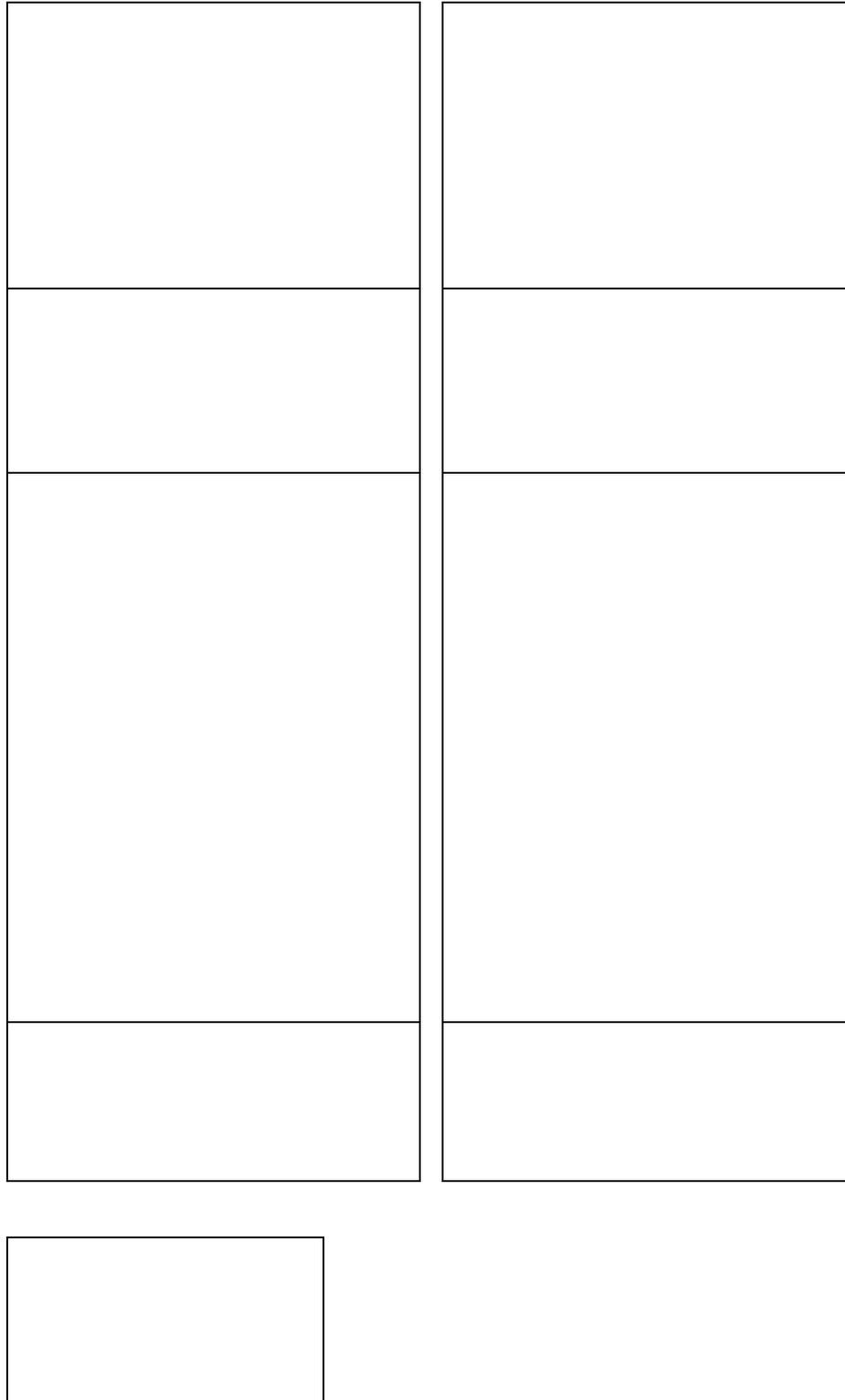
where

$$\bar{y}_h = \frac{1}{n_h} \sum_{i \in s_h} Y_i \quad (2.2.4)$$

is the sample mean of the observations in stratum U_h . So, the post-stratification estimator is equal to a weighted sum of sample stratum means.

A simple example shows how weighting works in the case of a simple random sample. From a population of size 1000 a sample of size 100 is selected. There are two auxiliary variables: Sex (with two categories Male and Female), and AgeClass (with three categories Young, Middle and Old). Figure 2.2.1 contains both the population and the sample distribution of these variables.

Figure 2.2.1. Post-stratification with two auxiliary variables



The sample is not representative for the population. For example, the percentage of young females in the population is 20.9%, whereas in the sample the percentage is 15.0%. The sample contains too few young females. The third table in figure 2.2.1 contains the correction weights as computed by means of expression (2.2.2). For example, the weight for young female is equal to $(209/1000) \times (100/15) = 1.393$. Young females are under-represented in the sample, and therefore get a weight larger than 1. People in over-represented strata get a weight less than 1.

The adjustment weights w_i are obtained by multiplying the correction weights d_i by the inclusion weights c_i . In this case, all inclusion weights are equal to $N/n=10$. Suppose, we use the weights to estimate the number of young females in the population. The weighted estimate would be $15 \times 10 \times 1.393 = 209$, and this is the exact population total. Thus, application of weights to the auxiliary variable results in perfect estimates. If there is a strong relationship between the auxiliary variable and the target variable, also estimates for the target variable will be improved if these weights are used.

The weighting model obtained by crossing the variables AgeClass and Sex is denoted by

AgeClass x Sex.

The idea of crossing variables can be extended to more than two variables. As long as the table with population frequencies is available, and all sample frequencies are greater than 0, weights can be computed.

However, if there are no observations in a stratum, the weight can not be computed for that stratum. This leads to incorrect estimates. If the sample frequencies are very small, say less than 5, weights can be computed, but estimates will be unstable.

As more variables are used in a weighting scheme, there will be more strata. Therefore the risk of empty strata or strata with too few observations will be larger. There are two solutions for this problem. One is to use less auxiliary variables, but then a lot of auxiliary information is thrown away. Another is to *collapse strata*. This means merging a stratum with too few observations with another stratum. It is important to combine strata that resemble each other as much as possible. Collapsing strata is not a simple job, particularly if the number of auxiliary variables and strata is large. It is often a manual job.

Another problem in the use of several auxiliary variables is the lack of a sufficient amount of population information. This is illustrated in figure 2.2.2. The population distribution of the two variables *AgeClass* and *Sex* is known separately, but the distribution in the cross-classification is not known. In this case the post-stratification *AgeClass* x *Sex* cannot be carried out, because weights cannot be computed for the strata in the cross-classification.

Figure 2.2.2. Lack of population information

One way to solve this problem is to use only one variable, but that would mean ignoring all information with respect to the other variable. What is needed is a weighting technique that uses both marginal frequency distributions simultaneously. There are two weighting techniques that can do that: linear weighting and multiplicative weighting. These two techniques are described in the next two subsections.

2.3. Linear weighting

The technique of linear weighting is based on the theory of general regression estimation. This section only gives an overview of the theory of

linear weighting. More details can be found in Bethlehem and Keller (1987).

Suppose, there are p auxiliary variables available in the survey. For each element i in the population, there is a p -vector $X_i = (X_{i1}, X_{i2}, \dots, X_{ip})'$ of values of these auxiliary variables. The p -vector of population totals of the auxiliary variables is denoted by

$$X_T = (X_{T1}, X_{T2}, \dots, X_{Tp})', \quad (2.3.1)$$

where X_{Tj} denotes the population total of the j -th auxiliary variable :

$$X_{Tj} = \sum_{i \in U} X_{ij}, \quad (2.3.2)$$

where the sum is taken over all elements i in the population U . The $N \times p$ -matrix of all values of the auxiliary variables is denoted by X . The i -th row of this matrix corresponds to the vector X_i .

If the auxiliary variables are correlated with the target variable, then for a suitably chosen p -vector $B = (B_1, B_2, \dots, B_p)'$ of regression coefficients for a best fit of Y on X , the residuals in the N -vector $E = (E_1, E_2, \dots, E_N)'$, defined by

$$E = Y - XB, \quad (2.3.3)$$

vary less than the values of the target variable itself. Application of ordinary least squares results in

$$B = (X'X)^{-1} X'Y = \left(\sum_{i \in U} X_i X_i' \right)^{-1} \left(\sum_{i \in U} X_i Y_i \right) \quad (2.3.4)$$

Of course, the vector B cannot be computed in practice, because the required information is not available. Instead, this quantity is estimated using the sample data. Let the p -vector b be the sample estimator for the p -vector B . A good choice for b is

$$b = \left(\sum_{i \in S} \frac{X_i X_i'}{\pi_i} \right)^{-1} \left(\sum_{i \in S} \frac{X_i Y_i}{\pi_i} \right) \quad (2.3.5)$$

The estimator b is an asymptotically unbiased (ADU) estimator of B . Expression (2.3.5) can be used to compute the *general regression estimator*. This estimator is defined as

$$y_R = y_{HT} + (X_T - x_{HT})'b \quad (2.3.6)$$

where y_{HT} is the Horvitz-Thompson estimator for the population mean of the target variable and x_{HT} the p -vector of Horvitz-Thompson estimators for the population totals of the auxiliary variables. The general regression estimator adjusts the simple Horvitz-Thompson estimator for differences

between the true values and estimated values of the totals of the auxiliary variables. The estimator is asymptotically unbiased.

It can be shown, see e.g. Bethlehem and Keller (1987), that the variance of estimator (2.3.7) is small if the residual values in E are small. Hence, the use of auxiliary variables which can explain the behaviour of the target variable, will be rewarded with a precise estimator.

Use of the general regression estimator implies a form of weighting. The estimator can be rewritten as

$$y_R = \sum_{i \in S} \frac{d_i Y_i}{\pi_i}, \quad (2.3.7)$$

where the correction weight d_i is defined by

$$d_i = X_i' v \quad (2.3.8)$$

and the p-vector v of weight coefficients is equal to

$$v = \left(\sum_{i \in S} \frac{X_i X_i'}{\pi_i} \right)^{-1} X_T \quad (2.3.9)$$

Post-stratification is a special case of linear weighting. This is illustrated with the example of weighting by Sex and AgeClass. The auxiliary variables are dummy variables. There is a dummy variable for each cell in the table obtained by crossing the variables. The weighting scheme is denoted by *Sex x AgeClass*. The available population information is displayed in figure 2.2.1.

Crossing the two auxiliary variables produces a table with $2 \times 3 = 6$ cells. So six dummy variables have to be introduced. The possible values of these dummy variables are shown in figure 2.3.1. The bottom line in the table contains the vector of population totals of the auxiliary variables. The values are equal to the population frequencies in the cells of the population table.

Figure 2.3.1. Values of the dummy variables in Sex x AgeClass

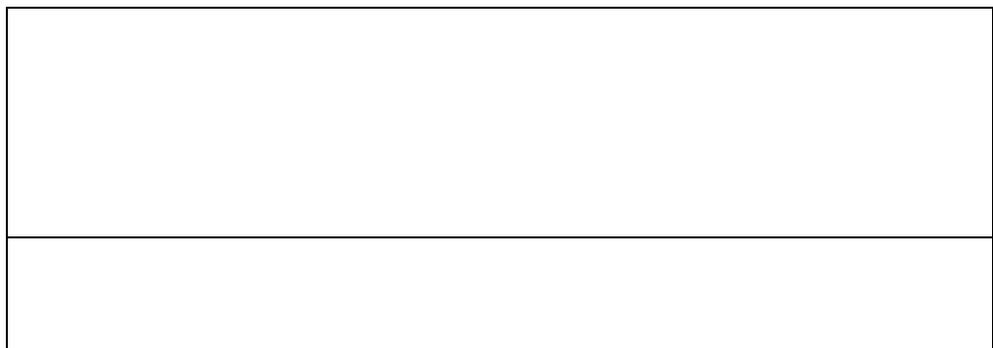


Using this information, the value of the vector v of weight coefficients turns out to be equal to

$$v = (0.983, 0.950, 1.023, 1.393, 0.847, 0.850).$$

Figure 2.3.2 shows how the corrections weights can be computed using the vector v . The weights are identical to the weights in figure 2.2.1.

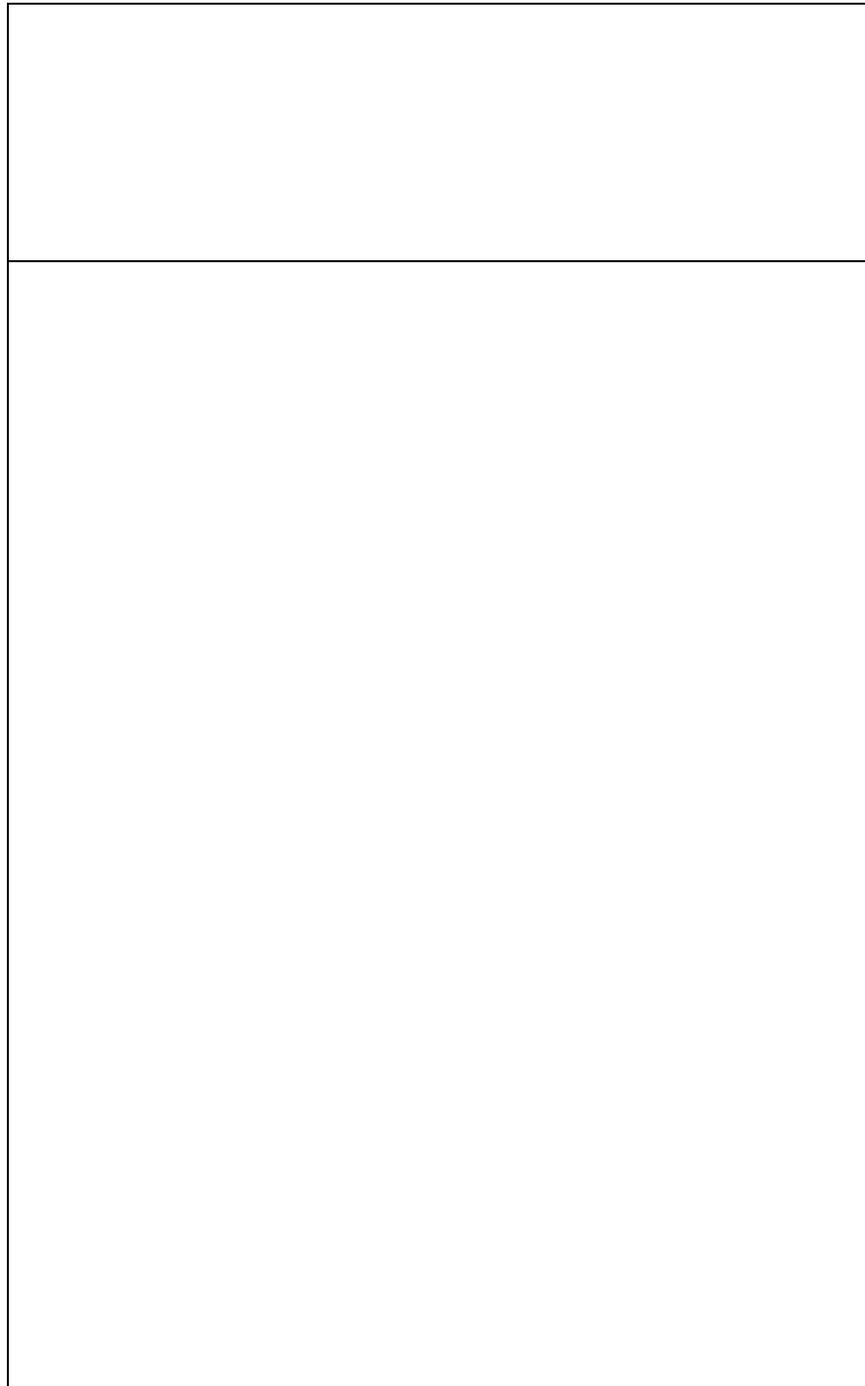
Figure 2.3.2. Computation of weights for Sex x AgeClass

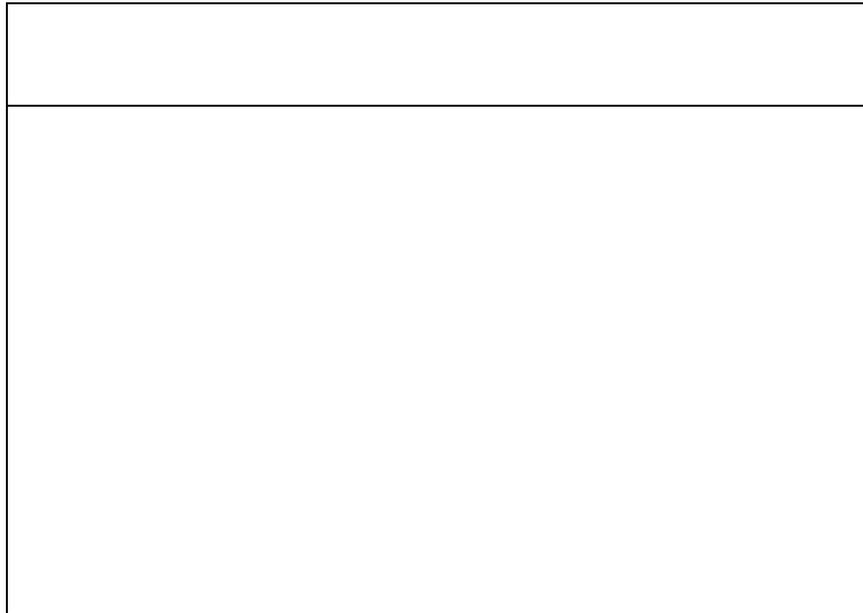


Now, it is illustrated how linear weighting can address the problem of the lack of sufficient population information. Linear weighting offers a possibility to include both variables in the weighting scheme without knowing the population frequencies in all cells. The trick is to introduce a different set of dummy variables. Instead of using one set of $2 \times 3 = 6$ dummy variables for the cells of the tables, use two sets of dummy variables: one set of two dummy variables for the categories of Sex, and another set of 3 dummy variables for the categories of AgeClass. Then

there are $2 + 3 = 5$ dummy variables. In each set, always one dummy has the value 1, whereas all other dummies are 0. The possible values of the dummy variables are described in figure 2.3.3. The first dummy variable represents the constant term in the regression model. It always has the value 1. The second and third dummy variable relate to the two sex categories, and the last three dummies represent the three age categories. The vector of population totals is equal to the frequencies of all dummy variables separately. Note that in this weighting scheme always three dummies in a row have the value 1.

Figure 2.3.3. Values of the dummy variables in Sex + AgeClass





Since no information is used about the crossing Sex by AgeClass, but only the marginal distributions, a different notation is introduced. This weighting scheme is denoted by

Sex + AgeClass.

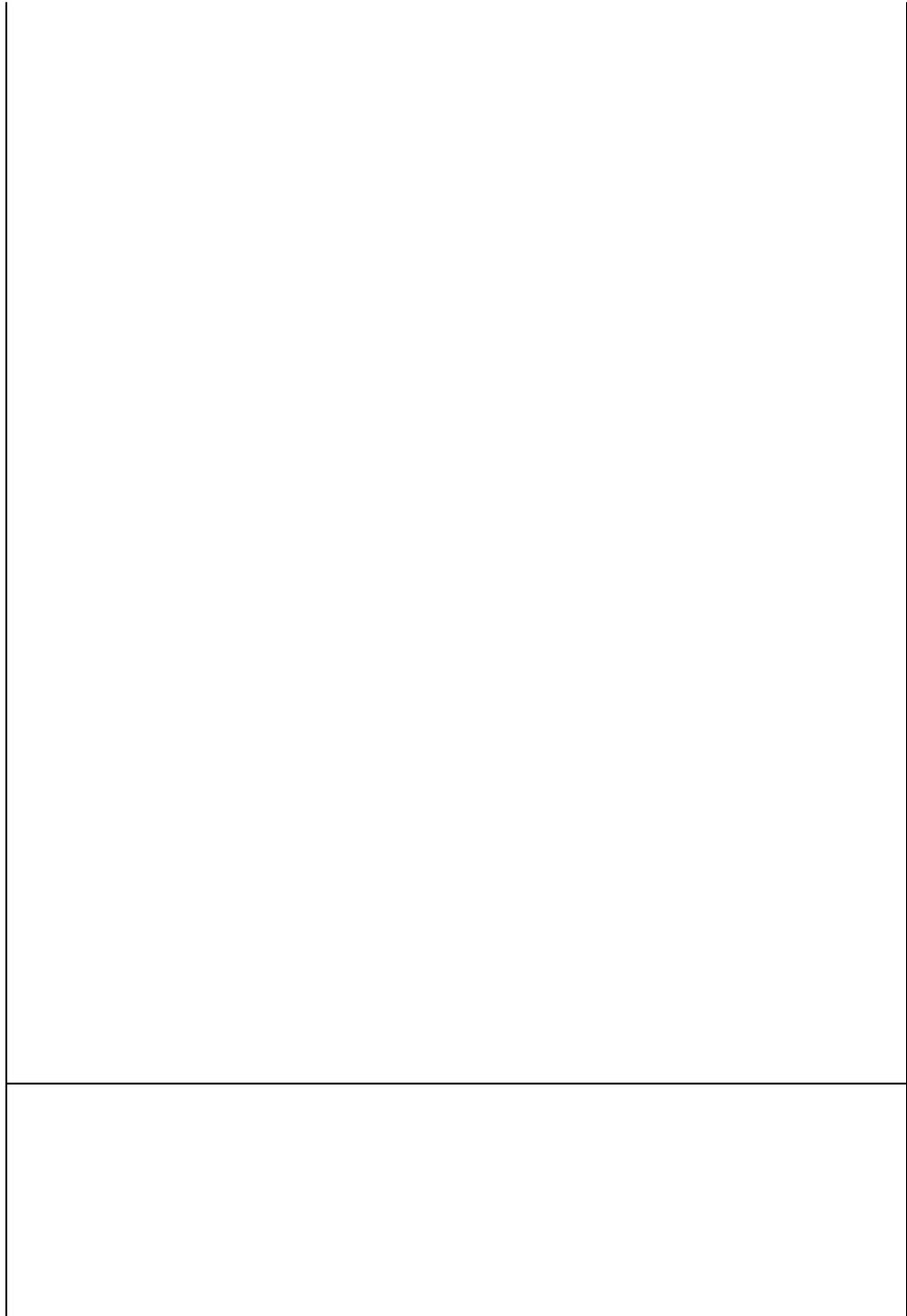
Expression (2.3.9) can be applied for the computation of the weight coefficients. The vector v of weight coefficients turns out to be equal to

$$v = (0.991, -0.033, 0.033, 0.161, -0.095, -0.066).$$

The weight for an element is now obtained by summing the appropriate elements of this vector. The first value corresponds to the dummy X_1 , which always has the value 1. So there is always a contribution 0.991 to the weight. The next two values correspond to the categories of Sex. Note that their sum equals zero. For males, an amount 0.033 is subtracted, and for females, the same amount is added. The final three values correspond to the categories of AgeClass. Depending on the age category a contribution is added or subtracted. Figure 2.3.4 contains the computation of the correction weights for all combinations of Sex and AgeClass. All figures are rounded to three digits.

Figure 2.3.4. Computation of weights for Sex + AgeClass





The weights in figure 2.3.4 are not equal to the weights obtained by complete post-stratification (see figure 2.2.1). This is not surprising, since the model *Sex + AgeClass* uses less information than the model *Sex x AgeClass*.

The examples used to illustrate the theory of linear weighting were based on two auxiliary variables. Of course, it is possible to use more than two variables.

So far, only the applications of linear weighting with qualitative auxiliary variables were discussed. It is also possible to apply linear weighting if only quantitative auxiliary variables are available. Then, linear weighting becomes generalised regression estimation. Moreover, it is also possible to combine qualitative and quantitative variables in a weighting model. This opens new possibilities for weighting. For more details, see the Bascula Reference Manual.

2.4. Multiplicative weighting

If linear weighting is applied, correction weights are obtained that are computed as the sum of a number of relevant weight coefficients. It is also possible to compute correction weights in a different way, namely as the product of a number of weight factors. This weighting method is called *multiplicative weighting*. Sometimes it is also called *raking* or *iterative proportional fitting*.

Generally, multiplicative weighting can be applied in the same situations as linear weighting as long as only qualitative variables are used. An example is the situation in which there is no population information about the complete cross-classification of all auxiliary variables, but there is information about classifications of subsets of these variables. The method of multiplicative weighting computes correction weights by means of an iterative procedure. The resulting weights are the product of factors contributed by all cross-classifications.

There is no general theoretical framework for multiplicative weighting. It is a step-wise process that is continued until satisfactory results are obtained. The general scheme is as follows :

- 1) Introduce a weight factor for each stratum in each cross-classification term. Set the initial values of all factors to 1.
- 2) Adjust the weight factors for the first cross-classification term so that the weighted sample becomes representative with respect to the auxiliary variables included in this cross-classification.
- 3) Adjust the weight factors for the next cross-classification term so that the weighted sample is representative for the variables involved. Generally, this will disturb representativeness with respect to the other cross-classification terms in the model.
- 4) Repeat this adjustment process until all cross-classification terms have been dealt with.
- 5) Repeat steps 2, 3, and 4 until the weight factors do not change any more.

This procedure is illustrated with a simple example. The two qualitative auxiliary variables Sex and AgeClass are used. It is assumed that only the population distribution of Sex (2 categories) and AgeClass (3 categories) separately are available, and not the cross-classification. Figure 2.4.1 contains the starting situation.

Figure 2.4.1. Starting situation for multiplicative weighting

The upper-left part of the table contains the weighted frequencies in the sample for each combination of AgeClass and Sex. They are the Horvitz-Thompson estimates of the corresponding population quantities.

The row and column denoted by 'Weight factor' contain the initial values of the weight factors. The values in the row and column denoted by 'Weighted sum' are obtained by first computing the weight for each sample cell (by multiplying the relevant row and column factors), and then summing the weighted cell frequencies. Since the initial values of all factors are equal to 1, the weighted sums in figure 2.4.1 are equal to the unweighted sample sums. The row and column denoted by 'Population distribution' contain the frequencies of AgeClass and Sex in the population.

The iterative process must result in row and column factors with such values that the weighted sums match the population distribution. This is clearly not the case in the starting situation. First, the weight factors for the rows are adjusted. The result of that exercise is shown in figure 2.4.2.

Figure 2.4.2. Situation after adjusting for AgeClass

--	--	--

The weighted sums for the rows are now correct, but the weighted sums for the columns still show a discrepancy. The next step will be to adjust the weight factors for the columns such that the weighted column sums match the corresponding population frequencies. The result of this operation is shown in figure 2.4.3.

Figure 2.4.3. Situation after adjusting for Sex

--	--	--

Note that the adjustment for Sex has disturbed the adjustment for AgeClass. The weighted sums for the age categories no longer match the relative population frequencies. However, the discrepancy is much less than in the initial situation.

The process of adjusting for AgeClass and Sex is repeated until the weight factors do not change any more. The final situation is reached after a few iterations. Figure 2.4.4 shows the final results.

Figure 2.4.4. Situation after convergence

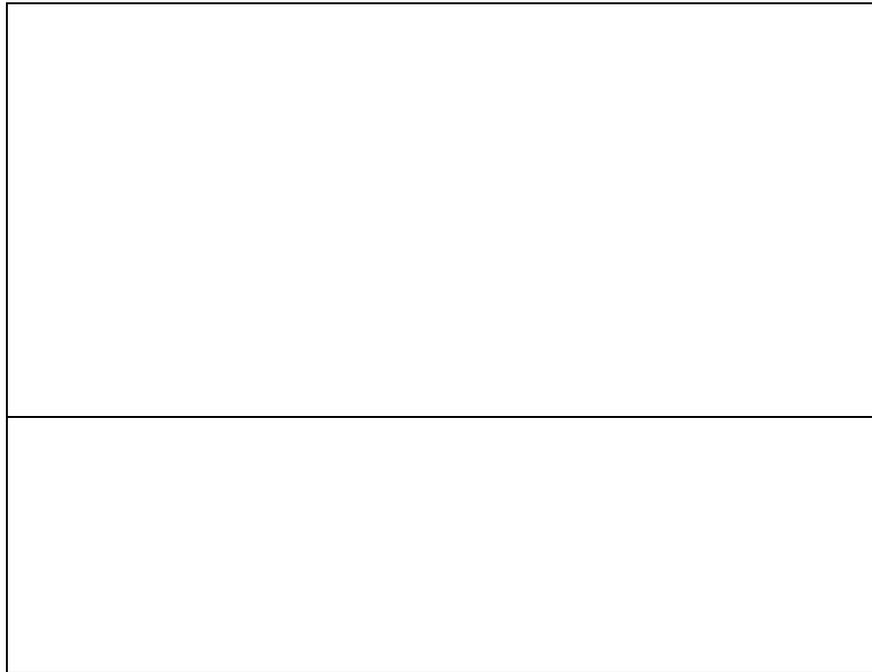
--	--	--

--	--	--

The correction weights for a specific sample element are now obtained by multiplying the relevant weight factors. Figure 2.4.5 contains the resulting correction weights. If the correction weights are multiplied by the inclusion weights, the adjustment weights are obtained. Since the inclusion weights are equal to $N/n=10$ in the example, the adjustment weights are simply 10 times the correction weights.

For this example, the adjustment weights differ only slightly from those obtained by linear weighting, see figure 2.3.4.

Figure 2.4.5. Computation of weights for Sex + AgeClass



In situations where both linear and multiplicative weighting are possible, a choice must be made. To help making this choice, some of the advantages and disadvantages of both techniques are pointed out.

Linear weighting has the advantage that it is based on a model that explains in which situations weighting will work. It is also possible to describe the effect of linear weighting on the variances of estimates.

Linear weighting may result in negative weights. Such weights are not wrong, but simply a consequence of the theory. Usually, negative weights are an indication that the model does not fit too good. Unfortunately, there are some analysis packages (e.g. SPSS) that do not accept negative weights. That might be a reason not to use linear weighting.

Multiplicative weightings lacks a proper model describing the properties of estimators based on these weights. It is also not very easy to compute variance estimators.

Multiplicative weighting always produces positive weights. If this is necessary requirement, this weighting technique should be chosen.

Although linear and multiplicative weighting seem to be very different weighting techniques, it is possible to create a general framework of which both techniques are special cases. This work was done by Deville and Särndal (1992). They proved that estimators based on linear or multiplicative weighting approximately behave in the same way. So, one could use linear weighting in selecting the proper variables, and then apply multiplicative weighting in order to ensure weights to be positive.

3. Use of Bascula in the Blaise environment

Bascula can be used in two ways: as a stand-alone package, or as a tool within the Blaise environment. To use Bascula as a stand-alone package, there must be a sample data file in Ascii format. Furthermore, a description of the variables in the file must be provided. This can be done interactively from within Bascula. This section describes how Bascula can be used as a tool within the Blaise environment. This is the most efficient way to use Bascula, because both data and meta-data are provided by Blaise.

To show how Bascula can be used, a simple example is used. A very small country was constructed. It was called Samplonia. It is a small island with 1000 inhabitants. The country is divided into two provinces: Agria and Induston. The province of Agria consists of three districts: Wheaton, Greenham, and Newbay. Induston has four districts: Oakdale, Crowdon, Mudwater, and Smokeley.

3.1. Data collection with Blaise

Statistics Samplonia has conducted a survey based on a sample of 100 inhabitants. Information was collected about place of residence (province and district), sex, age, employment status, and income. Figure 3.1.1 contains the Blaise data model for this survey.

The data model should contain all information required in the cause of processing the survey data. Therefore, it is important to realise at the design stage which variable will be needed in the ultimate data file. The data model in figure 3.1.1 contains two fields that are not filled during data collection, but afterwards.

The first field is *AgeClass*. This field represents a derived variable. It is a discrete variable, and its values are derived from the continuous variable *Age*. Survey publications often do not contain statistics on continuous variables. One reasons may be protection of confidentiality. Another could be that the values of the continuous variable are not very accurate. Often, it is already known at the design stage of the survey which tables will be published. In such a case, it is a matter of good documentation practice to include required derived variables in the data model. The variable *AgeClass* is a good example of a publication variable that is derived from an interview variable. Figure 3.1.1 shows that *AgeClass* will divide the *Age* in three classes.

To be able to compute unbiased estimates of population characteristics, the sample design must be known. More specifically, the first order inclusion probabilities are required. For a simple random sample without replacement, all inclusion probabilities are equal to n/N , where n is the sample size and N is the population size. To have the inclusion probabilities available, the field *IncWeights* is included in the data model of figure 3.1.1. *IncWeight* represents the inclusion weight, which is defined as the reciprocal of the inclusion probability. The inclusion probabilities in the example are all equal to 0.1, so the inclusion weights are all equal to 10.

Figure 3.1.1. The Blaise data model

```
DATAMODEL   Samplon1   "Samplonian   Population
Survey"
```

```
FIELDS
```

```
    District   "District of residence": (Wheaton,
Greenham, Newbay,
```

```
                                Oakdale,
```

```
Crowdon,
```

```
                                Smokeley,
```

```
Mudwater)
```

```
    Province   "Province of residence": (Agrida,
Induston)
```

```
    Sex         "Sex of respondent"      : (Male,
Female)
```

```
    Age         "Age of respondent"      : 0..99
```

```
    AgeClass   "Age class"                : (Young,
Middle, Elderly)
```

```
    Employ     "Employment status"       : (Job,
NoJob)
```

```
    Income     "Monthly net income"      : 0..6000
```

```
    IncWeight  "Inclusion weight"         :
0.000..1000.000
```

```
    AdjWeight  "Adjustment weight"      :
0.000..1000.000
```

```
RULES
```

```
    Province District Sex Age
```

```
    IF Age <= 30 THEN
```

```
        AgeClass:= Young
```

```
    ELSEIF AGE <= 55 THEN
```

```
        AgeClass:= Middle
```

```
    ELSE
```

```
        AgeClass:= Elderly
```

```
    ENDIF
```

```
    Employ Income
```

```
    IncWeight:= 10.000
```

```
ENDMODEL 7
```

Note that for the case of equal inclusion weights, it is not really necessary to specify these weights. If no inclusion weights are given, Bascula will assume them all to be equal, and can compute adjustment weights without knowing the inclusion weights. So the inclusion weights could have been omitted in the example. However, it is considered to be good documentation practice to include this information.

Most surveys suffer from non-response. Therefore, it is likely that some kind of weighting has to be carried out. The computed inclusion weights must be added to the data file. Again, good documentation practice suggests to account for weight variables in the data model. To that end, the field *Weight* has been included in the data model of figure 3.1.1. Note that initially this field has no value assigned to it. In a later stage, Bascula will replace this value by the true value of the weight. Figure 3.1.2 contains a view at the data file just after the fieldwork has been completed.

Figure 3.1.2. The first 10 records of the sample data file before weighting

--	--	--	--	--	--	--	--	--

After all data has been collected, and as much as possible detected errors have been corrected, a weighting procedure can be carried out. This is the topic of the next section.

3.2. Weighting with Bascula

To be able to compute adjustment weights, auxiliary information must be available. It is assumed that three auxiliary variables can be used: *District*, *Sex* and *AgeClass*. The available population information for these three variables is displayed in figure 3.2.1. There are two tables. The first one contains the population counts for the districts, and the second one has the counts for age class by sex.

Figure 3.2.1. The available population information

If there is only one population table available, simple post-stratification can be applied. This is not the case here. Therefore, a choice has to be made between linear weighting and multiplicative weighting. Linear weighting will be applied.

If Bascula is installed in the Blaise directory, the weighting package can be run from within the Blaise Control Centre. Bascula is one of the options in the *Tools menu*.

The first thing to do is to inform Bascula about the sample data. The *Sample menu* has options for this. The file type is set to Blaise with the option *File type*, and the name of the data file is selected with the option *Sample file*. Note that no meta-data has to be specified. Once Bascula knows the name of the Blaise data file, it can locate the corresponding meta-data file, and extract all required information about the variables from this meta-data file.

Next, Bascula must be instructed which variables to use for weighting. This can be done with the option *Assign variables* in the *Weighting menu*. The *Select button* produces a list of all variables in the Blaise data file. The relevant variables are selected in this list. For the example at hand, it comes down to selecting the auxiliary variables *District*, *AgeClass* and *Sex*, the inclusion weight *IncWeight*, and the final weight *AdjWeight*. After these variables have been selected, Bascula must know what roles they play in the weighting process. The *Edit button* is used for this. The variables *District*, *AgeClass* and *Sex* are assigned the role of auxiliary variable, *IncWeight* is the inclusion weight, and *AdjWeight* is to contain the final weight.

Three auxiliary variables will be used in the weighting model. For these variables the relevant population tables must be entered. This is done with the option *Population tables* in the *Weighting menu*. To be able to use the population information in figure 3.2.1, two tables must be defined. The first table only contains the variable *District*. The second table is obtained by crossing *AgeClass* and *Sex*. This table is denoted by *AgeClass × Sex*.

Tables are defined with the *Define button*. The defined tables are filled with the population distribution by pressing the *Data button*.

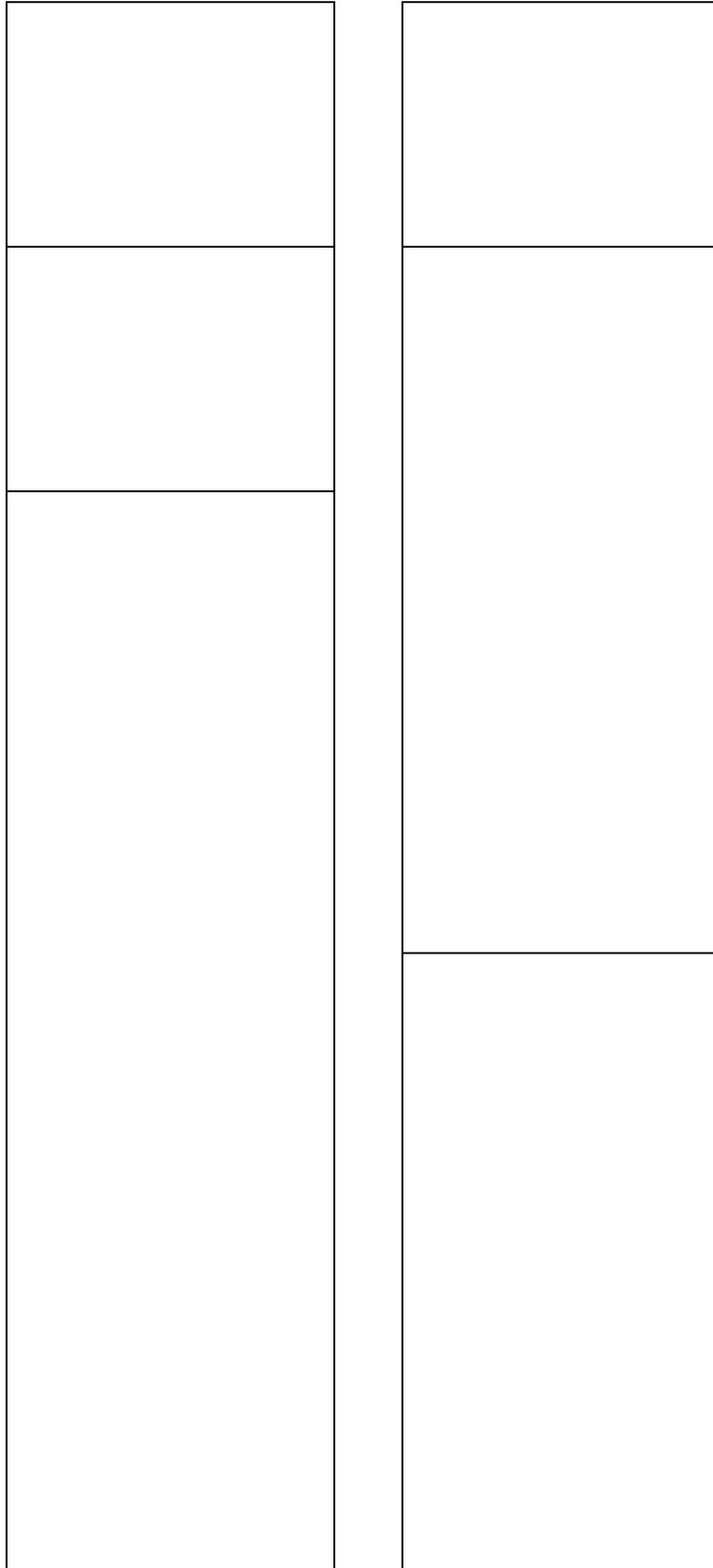
If more than one table is defined, they must all be checked for consistency. For example, the grand totals of all tables must be the same. Consistency is checked with the *Validate button*.

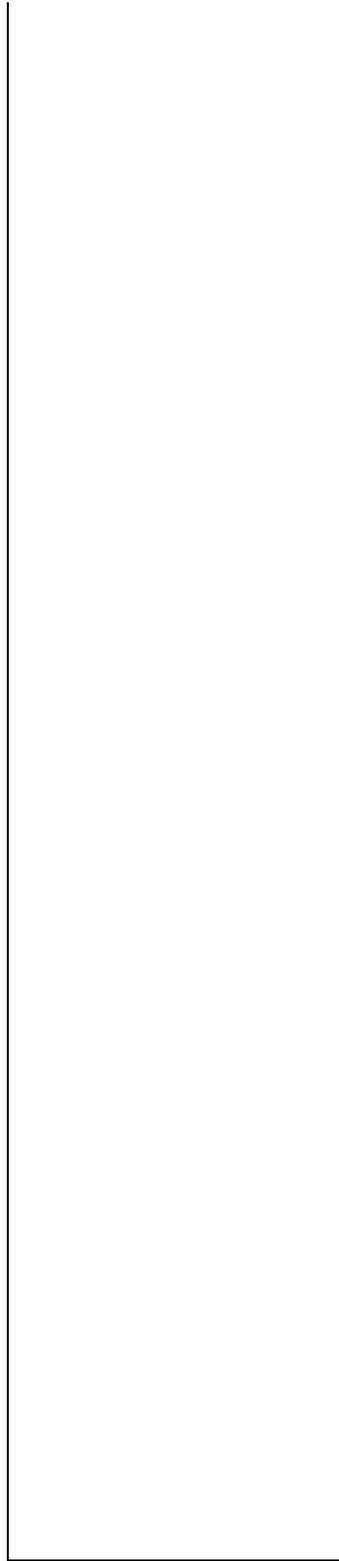
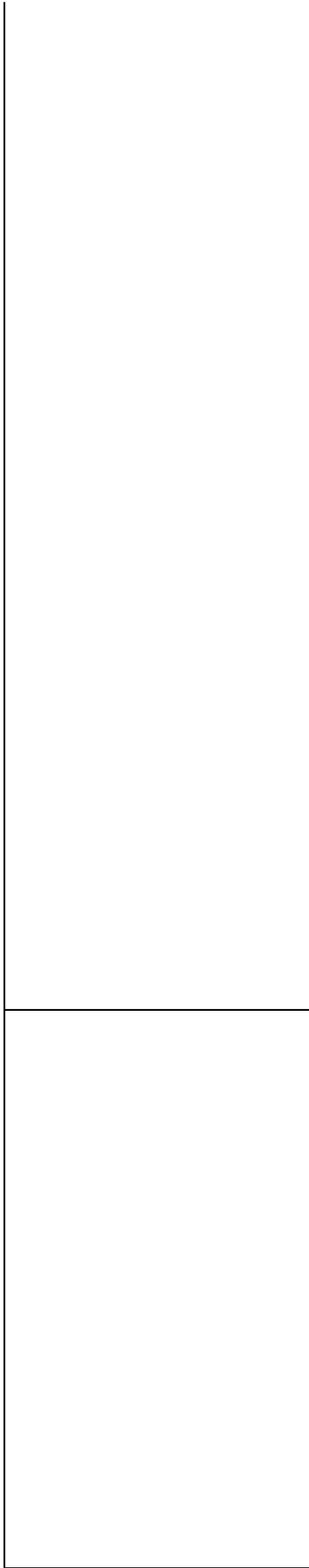
After all required information (sample data and population data) has been specified, the weighting model can be selected. This is done with the option *Select model* in the *Weighting menu*. Usually weighting produces the best results if the maximum possible population information is used. The maximal weighting model is selected by pressing the *Maximal button*. A weighting model must always be checked to see that sufficient sample observations are available in each stratum. This check is carried out by pressing the *Check button*.

If no problems are encountered, the computation of adjustment weights can be initiated through the option *Start weighting* in the *Weighting menu*.

For the example in this paper, two weighting techniques can be applied: linear or multiplicative weighting. Linear weighting is selected. Figure 3.2.2 contains the resulting weight coefficients.

Figure 3.2.2. The weight coefficients





The weight for a specific sample case is computed by adding up the weights in the relevant categories. For example, the weight of an elderly male in Crowdon is equal to

$$1.089 + 0.432 + 0.053 + 0.343 + 0.166 = 2.083$$

A closer look at the weight coefficients shows which categories are over- or under-represented in the sample. Negative weight coefficients denote under-representation and positive weights over-representation.

Note that the weights computed in this way, are correction weights. The final weights are obtained by multiplying the inclusion weights with these correction weights.

The final thing to do in Bascula is to write the final weights to the Blaise data file. The *Write button* takes care of this.

After leaving Bascula, it is a good idea to check the data file. You can take a look with the option *View data* in the *Tools menu* of the Blaise Control Centre. Figure 3.2.3 displays the contents of the first 10 records of the data file.

Figure 3.2.3. The first 10 records of the sample data file after weighting

--	--	--	--	--	--	--	--	--

The only difference with figure 3.1.2 is the contents of the column AdjWeight. The zeros are replaced by the real values of the adjustment weights.

3.3. Tabulation with Abacus

The next step in the statistical production process could be to make some publication tables. It is now shown that it is very easy to make unweighted and weighted tables with Abacus. For the weighted tables, Abacus can use the weights that were computed by Bascula.

First, a simple unweighted table is constructed. Abacus is accessed through the *Tools menu* of the Blaise Control Centre. All fields are selected for tabulation. A two-dimensional table is defined. The fields *Province* and *District* are used for the rows of the table, where *District* is nested in *Province*. The fields *Employ* and *Sex* are used for the columns, and *Sex* is nested in *Employ*. The resulting table is shown in figure 3.3.1.

Figure 3.3.1. The unweighted table

It is clear that this table contains sample frequencies, since the grand total of the table is equal to the sample size, which is 100.

To make estimates of population quantities, weights have to be incorporated in the table. That can be done in Abacus. The *Edit menu* of Abacus has the option *Calculations*. After activating this option, the weight variable can be selected by pressing the *Weight button*. The field *AdjWeight* is selected for this purpose. If the table is now recomputed, the result will be as displayed in figure 3.3.2.

The counts in the table add up to 1000, and that is the population size. Note that the totals for the field *District* exactly match the frequency distribution in figure 3.2.2. This is consequence of the fact that this field was used in the weighting model.

Figure 2.3.2. The weighted table

This section illustrates the use of Bascula as a tool in the statistical production process. Since the Blaise Control Centre handles all data and meta-data aspects, the use of Bascula becomes simple and straightforward. Bascula 2.0 was used to process the example in this section. Although Bascula is very useful to compute weighted estimates, it still has its limitations. Possible extensions of Bascula are discussed in the next section.

4. Future developments

Statistics Netherlands uses Bascula 2.0 to carry out adjustment weighting for several surveys. Experience has shown that this package is not applicable in every situation. Therefore, there is a demand to enhance the package with new functions. A project has been started to accomplish this. Work is in progress that will lead to a new version 3 of Bascula. In this section, an overview is given of some of the new features.

4.1. Limited weights

There are several reasons why a statistician may want to have some control over the values of the adjustment weights. One reason is that extremely large weights are generally considered undesirable. Large weights usually correspond to population elements with rare characteristics. Use of such weights may lead to unstable estimates of population parameters. To reduce the impact of large weights on estimators, a weighting method is required that keeps the adjustment weights within pre-specified boundaries, and at the same time enables valid inference.

Another reason to have some control over the values of the adjustment weights is that application of linear weighting might produce negative weights. Formally, this is not incorrect. The theory does not prevent negative weights. Usually, negative weights are an indication that the used regression model does not fit the data very well. Negative weights may cause problems in subsequent analysis. For example, if analysis is done on a sub-sample of the data containing a high portion of negative weights, the estimates will be inaccurate. Another problem is caused by some statistical analysis packages that can work with weights, but expect these weights to be positive. These packages are not able to properly process data sets containing negative weights.

To force weights within certain limits, several techniques have been proposed. A technique developed by Deville et al. (1993) comes down to repeating the weighting process (either linear or multiplicative) a number of times. First a lower bound L and an upper bound U are specified. After the

first run, weights smaller than L are set to L and weights larger than U are set to U. Then, the weighting process is repeated, but records from the strata with the fixed weights L and U are excluded. Again, weights may be produced not satisfying the conditions. These weights are also set to either the value L or U. The weighting process is repeated until all computed weights fall within the specified limits. Convergence of this iterative process is not guaranteed. Particularly, if the lower bound L and upper bound U are not far apart, the process may not converge.

In Bascula 3.0, a technique developed by Huang and Fuller (1978) will be implemented. Their algorithm produces weights that are a smooth, continuous, monotone increasing function of the original weights computed from the linear model. The algorithm is iterative. At each step, the weights are checked against a user-supplied criterion value M. This value M is the maximum fraction of the mean weight by which any weight may deviate from the mean weight. For example, if M is set to 0.75, then all weights are forced into the interval with lower bound equal to 0.25 times the mean weight and upper bound equal to 1.75 times the mean weight. Setting the value to 1 implies that all weights are forced to be positive.

One iteration of the algorithm consist of computing weights based on the linear regression model, where an adjustment factor g_i applied to every case i . This adjustment factor is the result of a 'bell' shaped function. The adjustment factor is large for vectors $X_i = (X_{i1}, X_{i2}, \dots, X_{ip})'$ close to the mean of the auxiliary variables, and the value of the factor approaches zero as the distance to the mean becomes larger.

The iterative process is repeated until all computed weights satisfy the criterion value. The value of M must be chosen with some care. If the value of M is too small, the algorithm may fail to find a solution. Huang and Fuller (1978) suggest to take a value of M that satisfies the condition

$$0.5 \leq M \leq 1 - \frac{n}{N}. \quad (3.1.1)$$

Huang and Fuller (1978) also prove that the asymptotic properties of the regression estimator constructed with their algorithm are asymptotically the same as those of the general regression estimator. So, restricting the weights has (at least asymptotically) no effect on the properties of population estimates computed with the these weights.

4.2. Household weights

Household surveys are usually based on a hierarchical data model. The highest level in the data model is the household. Households are composed of persons and they represent the next level in the model. Such surveys collect information about both levels in the data model. The collected information can be used to make estimates for two populations: the population consisting of all households, and the population consisting of all individual persons.

If the aim of the survey is to make inference on the population of all individual persons, the process is fairly straightforward. The unit of measurement is the individual person. The data file must be approached

as a file of records with data on persons. Available population information on the distribution of personal characteristics can be used to compute adjustment weights, and these weights are assigned to the individual records.

For making inference on the population of households, the same approach can be used. However, there is a problem. Usually there is no information available on the population distribution of household variables. Even information on simple variables like size of the household and household composition is lacking. This makes it impossible to carry out an efficient weighting procedure.

Since it is possible to compute weights for the members of the household, one may wonder whether it is possible to use the person weights in some way to compute a household weight. Possible approaches could be to take (1) the weight of the head of the household, (2) the weight of a randomly selected household member, or (3) to compute some kind of average weight of the household members. Whatever approach is used, there are always problems :

- If the household weights are applied to the members of the households, the weighted estimates of personal characteristics will not match known population frequencies. This discrepancy will not occur if the personal weights are used.
- Inconsistencies may turn up. For example, an estimate of the total income through the households will not be equal to an estimate based on the individual persons.

Generalised regression estimation offers a solution to these problems. Suppose the population consist of N persons distributed over M households. A sample of m households is selected, and these households contain all together n persons.

A new $n \times m$ -matrix H is formed defining household membership. The element H_{ij} of H gets the value 1 if person i belongs to household j (for $i=1,2,\dots,n$ and $j=1,2,\dots,m$). Otherwise, the value of H_{ij} is equal to 0. Next, the matrix Z is defined as $Z = H'X$. This matrix aggregates the values of the auxiliary variables within the households. For example, if there is one auxiliary variable Sex, then there are two dummies (one for male, and one for female). A row of Z represents a household, and the values in the row the number of males and females in that household.

The theory of linear weighting can now be applied, where the sample person data matrix X is replaced by the sample household data matrix Z . The same vector of totals X_T of the auxiliary variables is used in the computation of the weights. For more details about this technique, see Nieuwenbroek (1993).

4.3. Variance estimation

The current version of Bascula can compute adjustment weights. Use of such weights allows for the publication of better quality population statistics. However, to be able to judge the quality of statistics, some

indication of the quality of the published statistics is required. Therefore, it is important to be able to compute variances of estimates.

The theory of linear weighting provides formulae for the computation of the variances of estimators of population characteristics. However, computation of these variances requires the second order inclusion probabilities to be available. Particularly for large samples, this is a considerable computational effort. For multiplicative weighting there are no straightforward variances expressions.

The Department of Statistical Methods of Statistics Netherlands is now in the process of implementing a general variance estimation module that can be applied for both linear and multiplicative weighting, and that does not require second order inclusion probabilities to be known.

The variance estimation technique is based on the method of *Balanced Half Samples*. The general idea is the following. Assume that a stratified sample has been selected. Two elements have been selected (with replacement) from each stratum. Using the sample data, an estimator t for a population characteristic T can be computed.

By selecting one of the sample elements in each of the L strata, a so-called half sample is formed. Such a half sample can also be used to compute an estimate of the population characteristic. Many half samples are possible. In fact, there are 2^L such samples. Denote the estimator based on half sample α by t_α . Now, the variance of the estimator t can be estimated by the quantity

$$\frac{1}{K} \sum_{\alpha=1}^K (t_\alpha - t)^2, \quad (4.3.1)$$

where K is the number of half samples.

One complication is that not every sampling design is a stratified design with two observations per strata. So, for other designs adjustments have to be made to obtain such a situation. Another complication is that for a large number of strata, the number of half samples can be very large. Since an estimate has to be computed for every half sample, the computational effort can be substantial. In order to avoid this, it is possible to work with a subset of half samples. This subset has to be selected very carefully. The theory of Balanced Half Samples provides means to generate such a set of balanced half samples. For more information on the BHS method, see e.g. Wolter (1985).

Implementation of the theory of BHS method will mean that for each half sample a weighting procedure is carried out, resulting in a set of weights. To compute an estimate of the variance of some statistic, the statistic is computed for all sets of weights, and then expression (4.3.1) is applied.

The final tool will not be built into Bascula. Instead, it will be a separate tool that will repeatedly call the Bascula module responsible for computing the weights.

4. References

Bethlehem, J.G and Keller, W.J. (1987), Linear weighting of sample survey data. *Journal of Official Statistics* 3, pp. 141-153.

Deville, J.C. and Särndal, C.E. (1992) , Calibration estimators in survey sampling. *Journal of the American Statistical Association* 87, pp. 376-382.

Deville, J.C., Särndal, C.E. and Sautory, O. (1993) , Generalized raking procedures in survey sampling. *Journal of the American Statistical Association* 88, pp. 1013-1020.

Huang, E.T. and Fuller, W.A. (1978), Nonnegative regression estimation for sample survey data, *Proceedings of the Social Statistics Section, American Statistical Association*, pp. 300-305.

Nieuwenbroek, N.J. (1993), An integrated method for weighting characteristics of persons and households using the linear regression estimator. Report 8445-93-M1-1, *Statistics Netherlands, Voorburg, The Netherlands*.

Wolter, K.M. (1985), *Introduction to variance estimation*, Springer Verlag, New York.

Advantages and disadvantages of migrating Blaise 2.5-based survey systems to Blaise III

Leif Bochis, Statistics Denmark

1. The use of Blaise at Statistics Denmark

At Statistics Denmark Blaise has been used since 1990. During these years at least 70 applications have been developed, including instruments for CADI, CAPI, CATI and CASI and ranging from input systems for small paper forms to complex questionnaires, such as the Labour Force Survey comprehending instruments for telephone interviewing, data input and data editing.

Most of the applications are CADI or CATI - one CAPI (conducted by interviewers from a market research institute) and two CASI-instruments used as supplement for two surveys where most of the data are collected from registers.

A central data input section was abandoned a couple of years ago, leaving the data input tasks either to be performed by out-of-house bureaus or to be put in and edited via systems developed at Statistics Denmark. A large number of these tasks are now put in and edited via Blaise instruments.

At Statistics Denmark all data input/editing and interviewing are carried out on diskless workstations with all programs, data and external files located on file servers. The main software configuration is based on Windows 3.1 upon which almost all of the basic software - e-mail, wordprocessing, spreadsheets and archives - are based. Thus, Blaise instruments are always run in Dos-sessions under Windows - except for the telephone interviewing.

2. Case : The Forms Input Systems for Hotel and Camping Site Bed-Nights

For the Hotel and Camping Site Bed-Night Surveys paper forms are collected each month from medium-sized or larger hotels and camping sites. These forms were typed in via a data input bureau and later validated in batch - like traditional forms processing.

The planning started in the late autumn of 1995 in order to modernize the data input systems into combined data input and validation. The goals were :

Faster error correction and publication (the old process was relatively efficient though - but a couple of days ..)

Cheaper data input. Since the closing of the central data input section at Statistics Denmark many data input tasks were performed by a data input service company. This traffic was - though fairly efficient - rather expensive, so it was quite a demand that we should make an in-house solution.

The data input system should not replace the tabulation programs, but on the other hand, be open to a later replacement, i.e. the solution should not exclude certain ways of replacing the tabulation programs when resources to redevelop showed up. A demand that is easily met by the fine data and metadata export facilities in either versions of Blaise.

3. The Blaise 2.5 Version

The data input system was developed in January 1996 using Blaise 2.5. The necessary administrative tasks (dunning letter draw outs etc.) were programmed in Manipula and packed into a Menu system programmed in Visual Basic. The data input system was ready for use when the first 1996-forms arrived in the beginning of February and the administrative programs were added in the period from February to April. In April the system was adapted for the Camp Site Forms.

The administrative tasks and utility programs included :

- Dunning letter draw outs,
- Lists for telephone dunning,
- Data output to the tabulation systems,
- Names and Adresses draw out for arbitrary searching purpose,
- File Browsing,
- Miscellaneous utilities such as lock removal and backup.

The first three were developed as quite simple Manipula setups, while choice of parameters was done using Visual Basic forms and passed over to the Manipula setups through BAT files.

The names and adresses were put into a file in order to search arbitrarily without knowing the key of the hotel or camping site - in lack of the more flexible lookup-facilities of Blaise III.

File browsing and lock removal utilities were developed entirely with few lines of Visual Basic using the Windows Notepad program as file browser for smaller files.

4. The Blaise III Version

Blaise III was already under consideration when development started, but due to some experiences of instability of the then existing version, lack of some promised - and promising - facilities and the developer's lack of experience with Blaise III we did not dare to base the system upon it.

Later, with the release of version 1.1j in the summer of 1996 which proved stable and comprised the promised facilities, the Blaise support started to reconsider the data input system - though mainly as a study project.

The experiences of this project were later used as a basis for developing a Blaise III instrument for Maritime Statistics Data Input and Validation. In the winter of 1996/97. This instrument is not quite finished when writing these lines, but as this project is only the second major Blaise III project at Statistics Denmark the experiences are closely connected to the former and have influenced also the conclusions of this paper.

The study project on Hotels and Camping Sites consisted of a number of steps :

Automatic conversion of the Blaise-instrument into Blaise III and the Manipula setups into Manipula 2.1 using 22III.EXE and MAN2III.EXE. Followed by the necessary changes in the code due to the new way certain things are implemented in Blaise III. This conversion took a couple of days and exposed a lot of the benefits of Blaise III - among them the ease of manipulating dates and the valuable secondary key concept introducing lookups through trigram search.

The Visual Basic program and the BAT files were then transformed into a Maniplus setup. This process lasted another four or five days - while still inexperienced - and served as a solid introduction into the way these things are done using Maniplus.

The result of this process was a program with the same functions as the version built in Visual Basic and Blaise 2.5 except for some minor differences - as, for example, the built - in possibility of printing files from the Notepad program.

The result was also, however, a program that performed considerably slower due to certain changes in the way rules are processed in Blaise III.

Conclusions

The case once again proved the good properties of Blaise 2.5, Manipula and Visual Basic as tools for rapid application development.

The study project showed that although a rather large investment should be made before starting real life Blaise III projects, Blaise III and Manipula/Maniplus have the same potential - except for the ease of visual development in VB.

The greater advantages of using Blaise III comprehend :

- one integrated tool for all: administrative tasks, data input and menu systems,
- advanced lookup facilities which made alternative search programs obsolete.

We also realized, however, that it takes quite an effort to implement Blaise III as a new standard software in the house. Many things are carried out in Blaise III in a quite different way than in Blaise 2.5, which makes the necessary education of experienced Blaise 2.5 developers into Blaise III developers quite a job. And besides, that a large amount of work is needed to get to the starting point of use - among the trivial but nevertheless time-consuming tasks are translation (when you are using a 'small' language as mothertongue) and integration with other tools such as making the necessary CIF-files for exporting data and metadata (unfortunately a STP2CIF program hasn't been delivered).

Among the disadvantages are the remarkable use of memory - 8 MB is still widespread for our workstations, and when the demands rise to 4-6 MB for the application - including DEP and Maniplus - plus the eventual demands of the external files, these workstations had to be put in front of the queue of upgradings (which the users approved, of course).

We experienced the problem of very slow performance of lookups due to the inability of Blaise III to use lists in memory. As it is stated in the README.DOC file of version 1.1j :

"The MEMORY attribute for external files will not be implemented. The need for this option has become very limited with the usage of disk caching systems."

When using diskless workstations it is - with the current versions of our operating systems - not possible to use disk caching. We have until now solved the problems by using RAM-disks for external data instead. This works and the performance is satisfactory, but we are not happy with the inflexibility the use of RAM-disks imposes on our configuration.

In contrast, the Visual Basic + Blaise 2.5 solution provides the necessary performance and lower usage of memory due to the ability of Visual Basic to share memory (common DLL's etc.) with other Windows applications, and the ability of Manipula 1.6 to perform tasks without the need for extended memory.

Another disadvantage is that users may be biased towards the old dos interface of Maniplus, if they are used to Windows shells built on top of the data input instruments. This will be slightly harder to 'sell', if it is not followed by remarkable gains in the integration of the Blaise III tools. These gains may not be as obvious for users as well as for developers. So for this reason, too, we are looking forward to the Windows version.

At Statistics Denmark we have started development in Blaise III at a low level. There is no doubt, however, we will make good use of the large benefits of this version in the coming years.

On the other hand, quite a number of Blaise 2.5 applications - mostly applications using (larger sized) lists - should be evaluated carefully before moving them to Blaise III.

New approaches to data processing integration in North Rhine-Westphalian Statistics based on BLAISE III

Markus Broose, Frank Merks, Thomas Pricking, data processing and statistical office of the State of North Rhine Westphalia - Landesamt für Datenverarbeitung und Statistik Nordrhein-Westfalen - (LDS NRW), Germany

1. Tasks and structure of the statistical network in the official statistics of the Federal Republic of Germany

Official statistics in the Federal Republic of Germany are supplied by the federal and state statistical offices, which compile regional, supra-regional and national data. All European, federal and state statistics are compiled in accordance with the prescriptions of European, national or state law. Federal statistics comprise an annual approx. 180 sets of statistics, supplemented in North Rhine Westphalia by a further 30 so-called "coordinated sets of state statistics". These are compiled by all states of the Federal Republic with the federal statistical office usually exercising a coordinating function, as comparability of results is in both the federal and state interests. In addition to this, specific state surveys are carried out in the various states: in North Rhine Westphalia these amount to approx. 30 sets of statistics.

Federal statistics are compiled in a division of labour between the 16 state statistical offices and the federal statistical office. In close consultation with the state statistical offices the federal statistical office undertakes the methodological and technical preparations, coordination of the various sets of statistics and compilation and publication of the end results. The state statistical offices undertake the acquisition and processing of the statistical data together with evaluation and distribution of the results relating to their own particular state. Exceptions to this rule are a few sets of statistics which are dealt with centrally by the federal statistical office. The state statistical offices act in the capacity of independent state authorities which are not subject to control by the federal statistical office. They are funded by the states themselves and accordingly bear a large part of the financial burden of compiling federal and European statistics.

Given this structure, both federal statistics and the coordinated state statistics must be compiled in a uniform manner by all concerned so as to assure regionally and technically comparable results. That means that the

same methodological principles and procedures must be employed in all the statistical offices. In order to meet this requirement, the federal and state statistical offices have formed a "network" to develop and implement the necessary measures in close cooperation with one another. This statistical network relates not only to methodological and technical issues but also to collaboration in the creation of IT applications for processing the data acquired in surveys, ie. so-called "network programming".

Network programming aims at the achievement of two goals :

- *Standardization of methods wherever and in whatever division of labour the statistics are compiled.*
- *Optimum cost effectiveness through division of labour.*

In order to attain these goals, the statistical network has laid down criteria for the organization and programming of IT applications which must be adhered to by all parties involved :

- *The technical rules are laid down by the relevant technical committees.*
- *Specification guidelines for validation and tables in the form of a semi-formal language and for the technical/organizational implementation of the rules define a common basis for procedure.*
- *The employment of IT is controlled and coordinated by the information technology working party AKIT. All the statistical offices are represented in this working party.*
- *Standardized working procedures are employed in all statistical offices.*
- *At each stage of the work process the same program is used everywhere.*
- *The programs must be easily portable, as a variety of different platforms are in use within the network.*
- *As a rule, the requisite software is created, documented and maintained by one statistical office. All the other offices then receive the applications free of charge and load them into their systems. Platform-specific functions (job control, printer drivers etc.) must be adapted by the user.*

The federal and state statistical offices together allow an annual capacity of 80 man years for network programming. The synergy effects achievable by means of this division of labour are enormous. If the network did not exist federal statistics in their present form, ie. produced in an organized division of labour between the statistical offices, would scarcely be feasible.

2. Technical basis and perspectives of network programming

Network programming has been in existence since the beginning of the 1960s. Accordingly, mainframe computers (IBM/MVS and SNI/BS2000) have constituted the backbone of the system for more than three decades. For reasons of portability and efficiency the statistical offices have in the past used assemblers and assembler macros as programming tools. Such applications, which are still extensively used today, are run exclusively as batch jobs. In recent years many assembler programs have been successfully replaced by a 4GL language for evaluation and tabulation of results (SPLV, developed by the statistical network) which is tailored to the technical jargon of statisticians. This software is also job driven.

Whereas, in the past, batch oriented procedures were largely equal to the task of effectively supporting the processing of statistics, today's demands can only be met using modern methods. Statisticians' most urgent demands are for interactive solutions, more versatile and more decentralized technology together with faster and better presentation of results. Given increasingly stringent public sector budgets, cost cutting has clearly also become an important factor in the field of statistics.

Germany's public sector statisticians are making efforts to achieve these goals in two ways :

- *The ADABAS data base system using the programming language NATURAL has been introduced as a development tool for interactive mode programming. ADABAS is primarily a mainframe system but can also be run on Unix and PCs. ADABAS was quick to prove its merit in register applications for the production industry, trade and agriculture and also for statistical information data bases. In addition to this, ADABAS applications are used to run statistics processing including data validation in interactive mode.*
- *Micro-computers with their extensive range of capabilities are to constitute the other main pillar of our drive to modernize computerized statistics processing. Thanks to local area networks PCs can nowadays communicate with one another and, if necessary, be quickly connected to a mainframe, ensuring efficient data transfer at all times. The software industry has a wide range of development tools and ready-to-use applications on offer which are of great service in the preparation and processing of statistics. However, these programs all have the disadvantage that they are not, or are only to a limited extent, designed to cope with the special needs of official statistics. For example, they lack tools for error handling, coding of plain text and job sequencing. With regard to these points in particular, CBS Netherlands' BLAISE survey processing system is far ahead of the field and accordingly became standard in network programming in 1991, since when it has been used in a variety of surveys.*

There is now the evidence of a number of surveys to show that switching from batch oriented procedures to interactive mode solutions has substantial advantages.

Owing to organizational considerations beyond the scope of the present article, the 1994 micro-census, the most important sample survey for

population and social statistics¹, was carried out in North Rhine-Westphalia using approx. 45% conventional methods, ie. manual handling of paperwork and data registration and validation in batch runs on a host computer, and 55% using a BLAISE interactive program which already contained most of the validation checks. The "raw data" gained by both methods was subjected to final validation using batch programs on the host computer of the statistical network. This showed that only half the number of runs was necessary for the BLAISE material and that the fraction of implausible cases in the first validation run was only 3% of the error quota ascertained for the data produced by conventional means.

Running under ADABAS the metadata driven application DAMAST, used for processing building trade statistics, also demonstrated the usefulness of interactive solutions.

Similar results have also been obtained by other statistical offices. Nonetheless, it is not at present anticipated that all statistical offices will be switching quickly and completely to processing all their statistics in interactive mode. The conventional procedures will continue to be used in the statistical network for a certain time to come. This begs the question as to why this should be so, despite the advantages of interactive solutions and the favourable response of users to them. The reasons are complex and closely connected with the autonomy of the 17 statistical offices making up the statistical network.

- *As a rule, modernization projects require high initial capital expenditure and this poses a problem in times of budget cutbacks. The budgets of the statistical offices come out of federal or state funds and there is strong pressure to economize. Accordingly, investment in the new data terminals required for the implementation of interactive solutions has to be medium-term.*
- *As yet there are not enough programmers trained in the new techniques available. Resources for software development are accordingly limited.*
- *The planning of modernization projects in the statistical network presupposes agreement in detail between all parties involved. Coordination of the various specialist departments and IT staff easily results in project timescales of several years, during which statistics will necessarily continue to be produced by conventional methods.*
- *A massive introduction of interactive techniques will fundamentally change the statisticians' workplace. Work flows, organizational structures and areas of responsibility will all have to be redefined. The effects on personnel, ranging from reassignment and transfer to redundancy, may well be considerable. On the other hand, no statistical office will voluntarily accept losses of qualified staff, particularly as the demands made on statistics are becoming increasingly heavy.*

¹ *Micro-census is a 1% area random sampling procedure. In North Rhine-Westphalia in 1994 76.000 households were polled.*

Modernization measures must accordingly be a matter of long-term planning.

3. BLAISE in the data processing and statistical office of the State of North Rhine-Westphalia (LDS NRW)

Since the beginning of the 1990s LDS NRW has been making strenuous efforts to improve the compilation of statistics. At first concerned chiefly with better and faster tabulation of corrected data (statistical problem solving procedure), the measures envisaged by LDS are now aimed at the changeover to interactive procedures. As a member of the statistical network LDS must continue to participate in common procedures and is unable to "go it alone". On the other hand, LDS is reluctant to miss the opportunities offered by modern technology. Accordingly, a way out of the dilemma has had to be found.

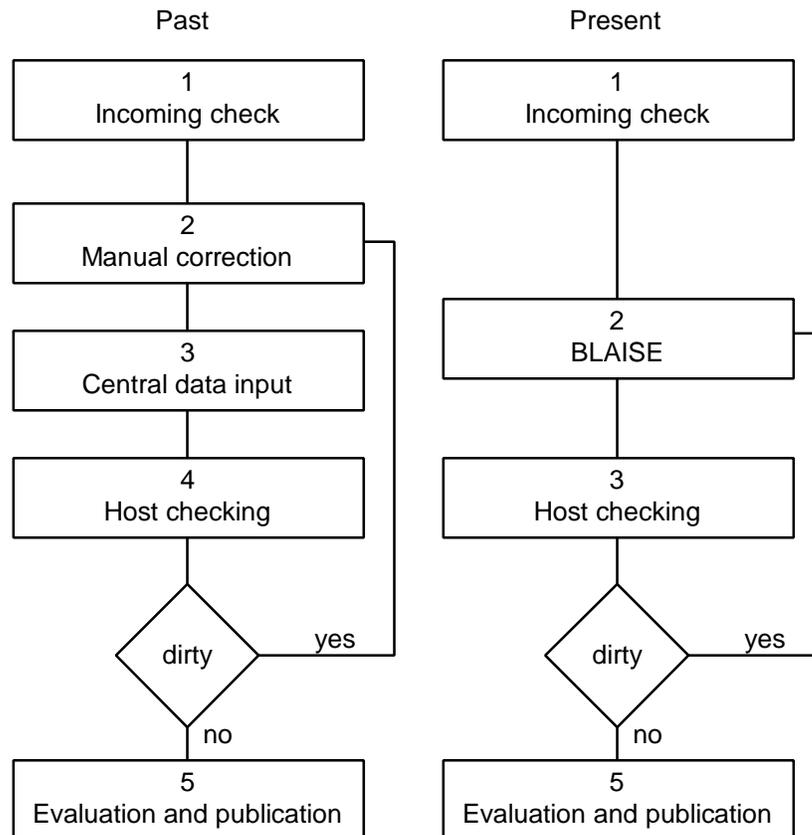
For one thing, LDS urges the increased use of interactive solutions in the statistical network. An example of this is the development of the ADABAS/NATURAL procedure for various surveys coordinated by LDS and its implementation within the statistical network. Whenever completely new procedures are developed for the statistical network consideration is to given to the question of whether PC solutions are technically and economically feasible. In the relevant committees LDS supports programming with BLAISE.

For another, it is urged that a particular way of using BLAISE be adopted which combines PC and conventional procedures. This intermeshing of PC and host computer applications will be referred to henceforth as "data processing integration". The following kinds of statistics are especially well suited to this procedure :

- *Statistics which have hitherto been validated, coded and registered using conventional methods.*
- *Statistics that require extensive coding.*
- *Statistics which, in accordance with statistical network planning, are processed manually up until batch validation.*
- *Statistics for which no changeover to interactive procedures is planned by the statistical network.*

The task of the BLAISE program will be to effect validation and coding during input itself with the result that, in theory at least, error-free material is available as soon as the last questionnaire has been entered. This material - where applicable, together with similar data from external sources (EDI, tapes, floppies) - will then be subjected to a final validation on the host computer using the standard procedures laid down by the statistical network. In this way the material produced with the aid of BLAISE will be integrated in the standardized process of the statistical network. This procedure is illustrated in the chart given in Fig. 1.

Fig. 1 : Chart showing data processing integration using BLAISE III



Two objects can be achieved :

- *Full exploitation of the technical advantages of BLAISE III while preserving the standardized method of the statistical network.*
- *Faster and more cost efficient processing*

The 1995 survey of salary and wage structure will serve as an example of the integration process and its resultant benefits.

4. 1995 survey of salary and wage structure (SWS)

The survey of salary and wage structure provides information on employees and their earnings, the nature and amount of statutory deductions and training and qualification characteristics. The survey is carried out on a random sampling basis in firms of more than 10 employees in the production industry, trade, banking and insurance. In 1995 this involved approx. 4,000 firms in North Rhine-Westphalia. Three different lists were used in the survey:

- *the company data sheet*
- *a list for wage earners*
- *a list for salaried employees.*

Prior to the main survey a preliminary poll was carried out in the firms concerned. The preliminary poll provided information on, for example, the collective bargaining agreements applicable to the firms, the number of employees and the way in which the firms wished to provide the information (questionnaires or data media). This information was processed into coding lists by LDS NRW and the federal statistical office which then formed the basis for parts of the validation procedures employed in the main survey.

Every firm was required to fill in a company data sheet with no omissions. A number of lists distributed for wage earners and salaried employees varied with the total number of people employed by the firm: The bigger the firm the lower the proportion of employees selected. On the forms used for the lists data for a maximum of 10 employees could be entered, with the result that medium sized and, particularly, large firms needed considerably more than one form.

This necessitated the creation of a system of data modelling and data input control that was versatile enough to cope with the varying nature and scope of the lists. Table 1 and Fig. 2 provide an illustration of the structure and scope of data. The flow chart given in Fig. 3 illustrates the individual steps of the procedure.

Tab.1 : Technical description of the model HSWS1_0X

- *BBetriebssatz*
 - *BArbeiterbogen*
 - *BArbeitertabelle*
 - *BArbeitersatz*
 - *BAngestelltenbogen*
 - *BAngestelltentabelle*
 - *BAngestelltensatz*
-

Overall counts

<i>Number of uniquely defined fields²</i>	109
<i>Number of elementary fields³</i>	102
<i>Number of defined data fields⁴</i>	4788
<i>Number of defined block fields⁵</i>	7
<i>Number of defined blocks</i>	7
<i>Number of embedded blocks</i>	0
<i>Number of lock instances</i>	282
<i>Number of key fields</i>	1
<i>Number of defined answer categories</i>	3
<i>Total length of string fields</i>	1964
<i>Total length of open fields</i>	0
<i>Total length of field texts</i>	2819
<i>Total length of value texts</i>	405
<i>Number of stored signals and checks</i>	6168

<i>Data fields</i>	<i>Number</i>	<i>Length</i>
<i>Integer</i>	3719	2010
<i>Real</i>	808	4428

² All the fields defined in the FIELDS section

³ All the fields defined in the FIELDS section which are not of type BLOCK

⁴ Number of fields in the data files (an array counts for more than one)

⁵ Number of fields of type block

<i>Enumerated</i>	37	37
<i>Set</i>	0	0
<i>Classification</i>	0	0
<i>Datatype</i>	0	0
<i>Timetype</i>	0	0
<i>String</i>	224	1964
<i>Open</i>	0	-
<i>Total in data model</i>	4788	18 439

Fig. 2 : Survey of salary and wage structure, data model (schematic)

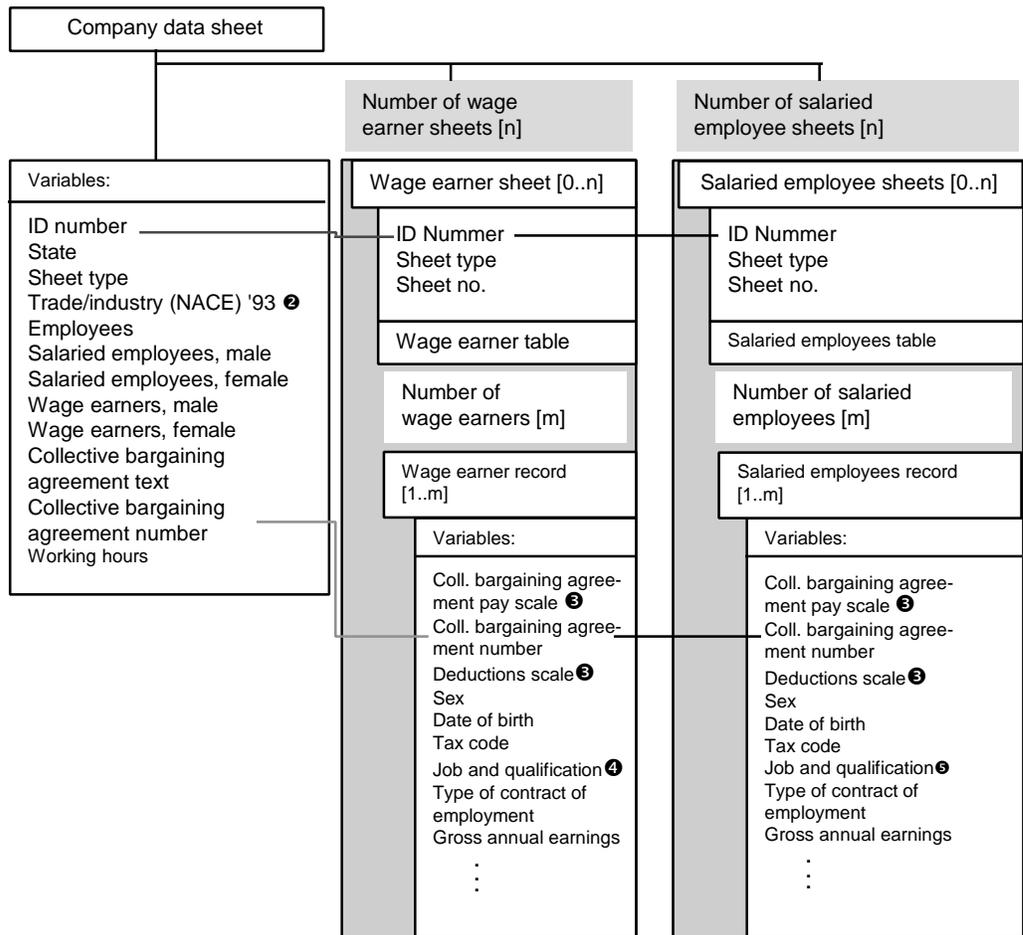
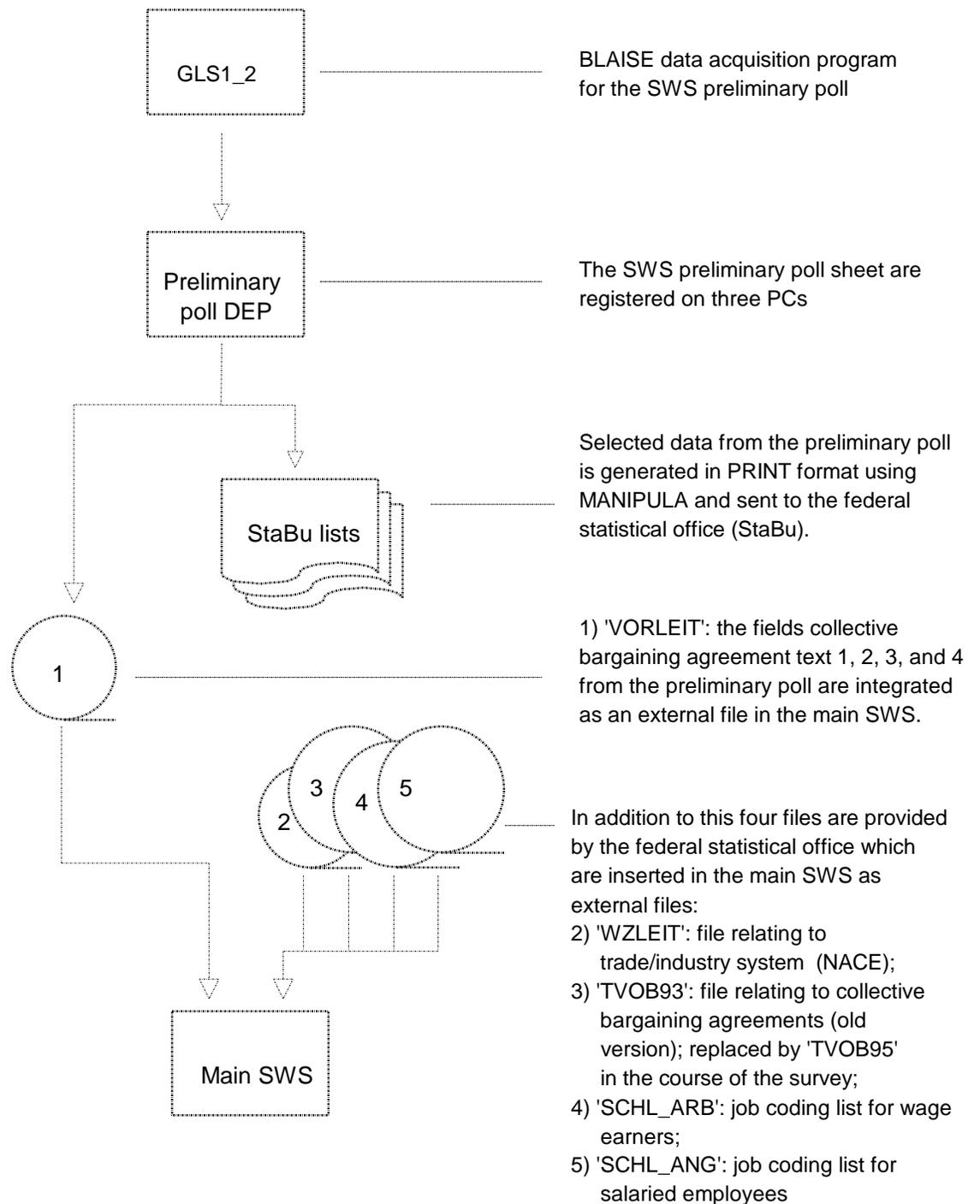
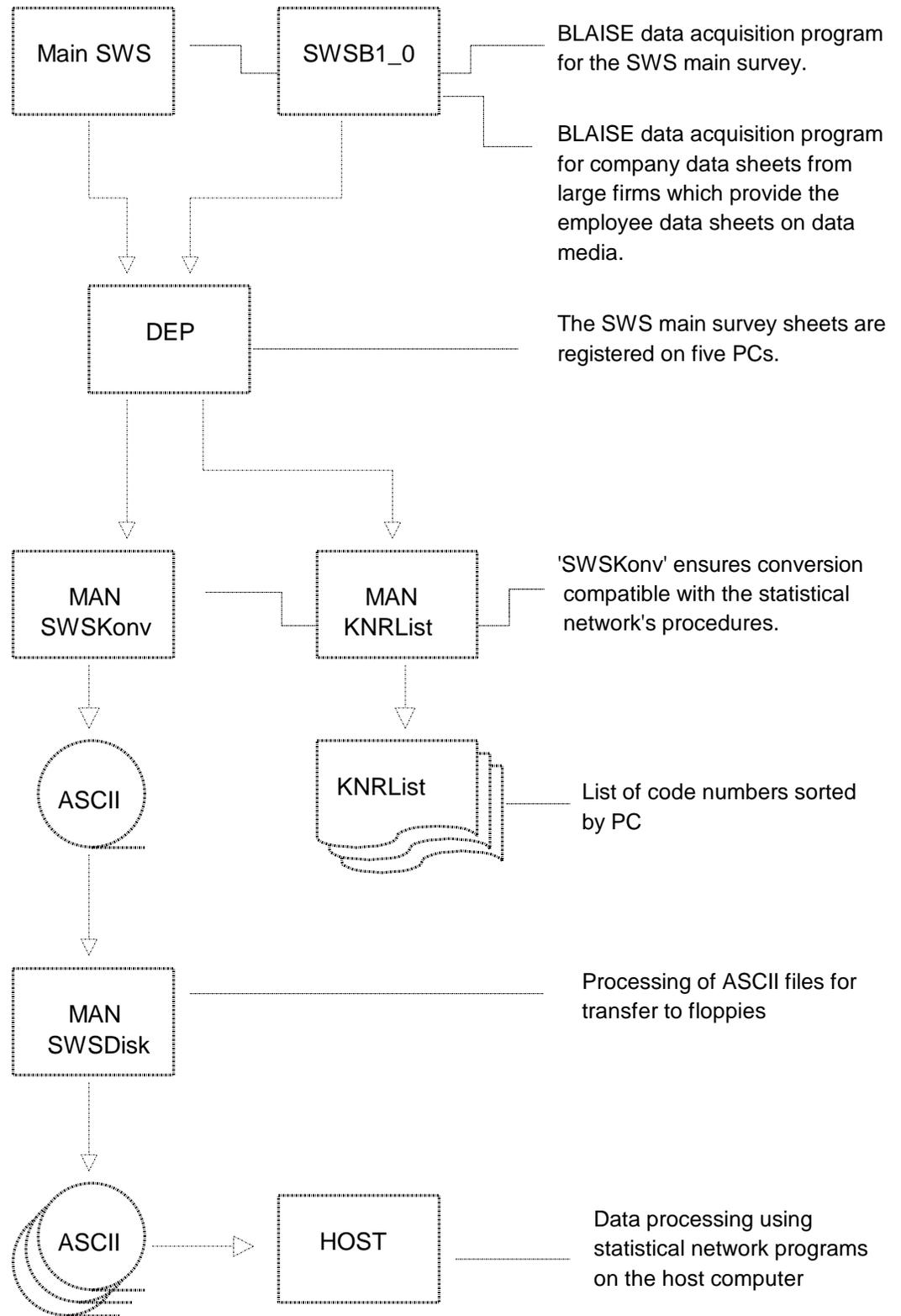


Fig. 3 : Survey of salary an wage structure, data flow chart



... Fig. 3



The preliminary poll

According to statistical network planning, data acquisition in the states was not to commence until the main survey. The data acquired in the preliminary poll was to be sent in to the federal statistical office in the form of lists for central compilation of a collective bargaining agreement key table. Already at this stage, LDS NRW used a BLAISE data registration program (GLS1_2). This facilitated the achievement of two objectives: firstly, to print out the data acquired and send it to the federal statistical office using a standard paper size suitable for clear presentation of information; secondly, to compile a key file (VORLEIT) specific to North Rhine Westphalia which, taken together with the 1993 version of the collective bargaining agreement key tape, would assure an early start for processing of the main survey data.

The main survey

The data entry program was configured in such a way that the necessary support for validation and coding was provided by a number of externals. Unfortunately the latest collective bargaining agreement key file, which was to supersede the older version, was not provided by the statistical network until data registration was already at an advanced stage. The effects of this delay on error frequency will be referred to later.

The enormous quantity of data from each firm quickly proved to be a problem. At the beginning of data registration work was mostly carried out using PCs with DX386 processors. When data records reached maximum size (e.g. for a firm with one company data sheet, approx. 50 wage earner sheets and approx. 50 salaried employee sheets) the PCs frequently crashed. Subsequent reorganisation runs (MONITOR) occasionally revealed that data had been lost. The systems were not stabilised until the switch was made to PCs with DX486 processors and 8 MB RAM and the number of employee sheets was restricted to 35 per firm.

The changeover from conventional to interactive processing with BLAISE necessitated the creation of an interface to transfer data in a compatible format to the host computer. The federal statistical office's data record descriptions for SWS served as a basis for the MANIPULA program SWSKonv and ensured a smooth link with the statistical network's standard program. The conversion procedures required considerably more time than originally estimated. For various reasons it was not possible to network the PCs and the data had to be transferred using floppies. Although the MANIPULA Tools (SWSDisk) proved very useful in splitting the data into manageable chunks, the length of time required was nonetheless unsatisfactory.

5. Results

Having considered the organizational processes and program structures, the question of the benefits of integration and the conclusions that can be drawn is naturally of great interest. This can be assessed by comparing the latest survey of salary and wage structure with that of 1990.

The present article was written approx. 5 months before the planned date for completion of processing and can accordingly provide no final assessment of all the work procedures and effects involved in the use of BLAISE. However, a few initial comments can be made on the basis of findings so far.

Any meaningful interpretation of a comparison of the 1990 and 1995 surveys should, however, be prefaced by one or two remarks on method.

- *The catalogue of statistical characteristics was extended by a further three characteristics.*
- *The work groups were reorganized.*
- *The preliminary poll for the survey of salary and wage structure was also carried out using BLAISE.*
- *Incoming checks and response were effected with computer support.*
- *In 1990 there was a delay of several months before the batch validation programs could be made available by the statistical network and manual validation was accordingly much more extensive than planned.*

The following tables provide an initial overview of the scope, expenditure of time and effort and error frequency of the two surveys.⁶

Tab. 2 : Scope of the survey in terms of the number of firms polled and number of employees

Year	Firms	Employee records	Information received on data media in %	
			Firms	Records
1990	3,000	135,000	14.3	27.4
1995	4,000	155,000	13.9	27.7

⁶ *Our thanks to Rolf Schmidt for his contributions and comments.*

Tab. 3 : Expenditure of time and effort and savings

<i>Procedures</i>	<i>Man months required</i>		<i>Savings in man months</i>
	<i>1990</i>	<i>1995</i>	
<i>Polling, processing, evaluation</i>	<i>290</i>	<i>240⁷</i>	<i>50</i>
<i>Registration of data</i>	<i>10</i>	<i>-</i>	<i>10</i>
<i>Support of statistical network programs</i>	<i>2</i>	<i>2</i>	<i>-</i>
<i>BLAISE programming</i>	<i>-</i>	<i>10</i>	<i>-10</i>
<i>PC care and maintenance</i>	<i>-</i>	<i>1</i>	<i>-1</i>
<i>Total</i>	<i>302</i>	<i>253</i>	<i>49</i>

Tab. 4 : Error frequency in first validation batch run (statistical network program)

<i>Year</i>	<i>Firms checked</i>	<i>Errors</i>	
		<i>Total</i>	<i>Per firm</i>
<i>1990</i>	<i>2,634</i>	<i>481,000</i>	<i>170</i>
<i>1995</i>	<i>3,141</i>	<i>162,000</i>	<i>52</i>

The scope of the 1995 survey is just about 15% greater than in 1990. As the proportion of information received on data media remained constant, the number of cases that had to be processed manually was correspondingly larger. This must be taken into consideration when looking at Table 3. Despite the fact that there was more data to process in 1995 the work required for polling and processing was cut by 50 man months. The savings in the central data input were, however, made up for by the work required for BLAISE program development. If the salary and wage structure survey were to be carried out on a continuous basis the programming required for 1996 would not need to be repeated in the subsequent years. Only one or two man months would have to be allowed

⁷ *Estimated on the basis of present state of present level of working process until now.*

for maintenance and any necessary adaptation of the BLAISE programs and savings in data acquisition would become really appreciable.

Taking the 1990 figures as a basis, the man months required for the more extensive survey carried out in 1995 would amount to around 330, ie. 27% more than was actually needed. Even when the above-mentioned methodological aspects are taken into account and the effective savings are reduced by a few percent, the benefits of BLAISE are obvious. The cost of buying the necessary PC hardware is hardly significant when compared to the savings in manpower.

This also becomes clear when one looks at the quality of the data undergoing validation. The criterion applied is the number of errors detected in the first run of the statistical network program on the host computer. Any number of errors can be registered per data record. The absolute number of errors was cut to a third and the number of errors per firm to 30%. That a lower error quota results in faster and less costly correction must be clear to anyone.

It is nonetheless surprising that the error frequency was so high in spite of BLAISE. This is partly to do with the organization of data in the process as a whole, because certain things cannot be checked until all the data and preliminary information has been brought together. For example, initial studies show that around 30,000 errors are attributable to the fact that an important coding list⁸ was provided by the statistical network so late that a large part of the material could not be compared with it during input. A not inconsiderable number of the errors are soft errors that are recorded for reasons of quality assurance but need not result in corrections in all cases. Account must also be taken of the fact that not all the recognized and conceivable validation checks were implemented in BLAISE so as not to delay sending out the programs to the statistical departments and possibly jeopardize the schedule. For the same reason things concerning validation that were learned in the course of processing the data could be included in the batch programs but not in BLAISE. It is only logical that this resulted in an increased number of error messages.

The bottom line is that PC/host computer integration based on in-house LDS BLAISE programs and statistical network programs has proven its merit. The goals that had been set were achieved in full. The survey was carried out at low cost and a high standard of quality while preserving the methodological uniformity of practice of the statistical network. These impressive results mean that the procedure will be applied to other surveys. Programming with BLAISE can make an efficient contribution to promoting a smooth changeover from conventional batch oriented processing to modern, metadata driven interactive processing of statistics.

Summary

⁸ ie. the so-called "collective bargaining agreement key file".

For many years official statistics in Germany have been compiled by a network consisting of the federal and state statistical offices. The necessary data processing software is developed jointly and used by all. Today's demands on the statistics production process have given rise to projects aimed at the modernization of data processing as used in the field of statistics. The present article describes how conventional and modern procedures have been combined in the compilation of official statistics in North Rhine Westphalia so as to cut costs while preserving standardized methods within the statistical network. The technical basis for this is the use of PCs and BLAISE III.

An evaluation of computer-assisted occupation coding : results of a field trial

Diane Bushnell, Office for National Statistics, UK

1. Introduction

At the last International Blaise Users' Conference held in Helsinki, I described some work being carried out in the Social Survey Division (SSD) of the Office for National Statistics (ONS) on computer-assisted occupation coding.¹ This paper describes the results of our work.

Over the last two years we have been investigating the use of computer-assisted occupation coding by conducting trials of different coding systems. From the results of the trials, we have concluded that it is feasible to use computer-assisted occupation coding without loss of data quality, and that using a straightforward word-matching system can be better than using a knowledge-based system. However, there are some questions remaining about whether unacceptable levels of bias will be introduced into occupation estimates when using computer-assisted coding.

2. Background

The majority of the surveys in SSD are now carried out using computer-assisted personal or telephone interviewing (CAPI/CATI). Computer-assisted coding is used routinely for most straightforward coding frames, such as country of birth, food items or journey destinations. However, occupation coding is still carried out using a clerical method. As we collect occupation details and code occupation on all our surveys we are very interested in assessing whether it is possible to use computer-assisted coding while maintaining, and ideally improving, data quality.

In the mid-1980s, SSD made a decision to code occupation in the field: the interviewers would collect the occupation details as usual and then code occupation at home, after the interview. Previously, in common with most other survey organisations, we employed specialist coders based in the office. Studies that were carried out at that timeⁱⁱⁱⁱ comparing office and field coding showed that although interviewers could not achieve the same levels of inter-coder consistency as office based coders, their smaller workloads meant that any systematic bias demonstrated by individual coders had less impact on the precision of the results. Moreover, they

learnt what occupation details were important to the coding process and became better at eliciting appropriate information in the interview. Finally, the cost and time savings were large enough to offset any worries about a reduction in data quality.

More recently another trial was conducted comparing office and field coding, since the interviewers had become more experienced^{iv}. Although inter-coder consistency was still lower for field coding compared to office coding, the accuracy of coding for the two groups was very similar (Table 1). As in the previous trials, individual coder biases resulted in a greater loss of precision for occupation related estimates when office coded, compared to coding carried out by interviewers.

Table 1 Results of a study comparing office and field coding of occupations

	Reliability*		Accuracy*	
	Office	Field	Office	Field
SOC	0.82	0.74	80.3	76.9
SOC Major group	0.90	0.86	87.8	87.4

* Reliability (inter-coder consistency) was estimated from average agreement between pairs of interviewers (perfect reliability=1). Accuracy was estimated from the percentage agreement between coders and an expert coder (perfect accuracy = 100).

Even though the data quality obtained by field coding has clearly improved since its introduction, we felt that there was scope for further improvement. We hoped that computer-assisted coding would contribute to that improvement.

3. Aims of study

The main aims in carrying out the study were to determine whether computer-assisted coding of occupation by interviewers could

- improve the quality of coded occupation data,
- reduce costs, and
- decrease interviewer burden.

Another, less important, aim was to assess the feasibility of coding occupation during the interview (to further aid the interviewers in assessing which was the most appropriate code).

We designed an experiment to compare two different coding systems to our usual clerical coding method. A discussion of our initial work to decide on suitable systems can be found in my previous paper for the 1995 Blaise conference¹. We asked the interviewers who took part in the experiment to give us feedback on the two systems, using a semi-structured questionnaire.

4. The study

4.1 Occupation coding

In the United Kingdom, the Standard Occupational Classification (SOC) (OPCS, 1990)^v is the most widely used occupational coding scheme for social research. The SOC was developed for use on the 1991 Census of Population. Its major use is for assigning individuals to various social class and socio-economic classifications.

The SOC consists of 371 occupational unit groups (SOC codes) which may be aggregated into minor and major groups. The system is hierarchical so, for example, SOC code 270 (Librarians) is in minor group 27 (Librarians and Related Professionals), which is in major group 2 (Professional Occupations).

In SSD the interviewers collect verbatim details on the job title, main duties and responsibilities of the job, qualifications required and industry, as well as asking specific questions about employment status (e.g. whether respondents are self-employed and the number of employees at their workplace). At home, the interviewers go back into the questionnaires, review the occupation details and select the SOC code which they think is most appropriate from a paper coding index, using a set of standard rules and procedures. Finally, they type the SOC code into the Blaise instrument.

4.2 The experiment

We carried out the experiment in three phases. 300 sets of occupation details were extracted from real data collected on the UK Labour Force Survey (LFS). In the first phase, 24 interviewers coded these using the standard clerical method. In the second phase, the same interviewers coded the same set of occupations using a specialised occupation coding system from within a Blaise 2.5 questionnaire. To minimise memory effects, we allowed nine months to elapse between the first two phases and the occupation details were presented to the interviewers in a randomly rearranged order. In the third phase, a different set of 24 interviewers coded the same 300 occupations using Blaise III.

At the first and second phases Blaise 2.5 was our standard interviewing software for the LFS. By the third phase, Blaise III had become standard; all the interviewers had been trained in its use and had been using it for interviewing for a few months. Therefore, at each phase of the experiment, the interviewers were using the same version of Blaise as they used every day for interviewing. This meant that we only needed to provide a minimum of training in the coding procedures.

4.3 The coding systems

- Clerical coding

The system used in the study was designed to resemble the normal coding process as much as possible. A Blaise 2.5 program was written which displayed each set of occupation details (job title, job description, industry, employment status) in the standard way. Using standard

procedures, the interviewer searched through the paper coding index for a suitable match to the job title, using the other occupation information when appropriate. When a match was found the interviewer entered the code into the Blaise instrument. After entering a code, the next set of occupation details were presented until all 300 were completed.

- CASOC/Blaise 2.5

A customised version of the specialised occupation coding system CASOC^{vi} (Computer Assisted Standard Occupational Coding) was integrated with Blaise 2.5 for use in the study.

CASOC is a sophisticated knowledge-based occupation coding system, first developed in 1986 and based on the Standard Occupational Classification (SOC).

The matching algorithm takes several factors into account before proposing suitable codes, including a word match comparison and the results of a weighting procedure. Weights are attached according to how commonly the job occurs in the population and how many other possible matches there are for the job title entered. It is possible to use employment status information to further refine the search (e.g. if the respondent is self-employed the job titles which are suggested first are those which are most appropriate for the self-employed).

Our customised version of CASOC was incorporated into Blaise 2.5 (it was not possible to call external programs from Blaise III at the time we developed the system). It was specifically designed to look and behave like Blaise 2.5 so that the interviewers would think they were using the usual Blaise coding facilities.

For the experiment, the job title and employment status information was passed to the matching algorithm; the best matches were displayed in the bottom half of the screen; the other occupation details were displayed in the top half of the screen. The interviewers had the option to search through the list of suggestions, extend the search to include more unlikely codes and to edit the job title to add or change the information used for matching.

- Blaise III

As we knew that Blaise III would become our standard interviewing software we were very interested to see how coding using the facilities provided by Blaise III compared to coding with the knowledge-based system.

Blaise III coding is based on word matching (using trigrams) and/or on stepping through hierarchies of a coding frame. We decided that the Blaise III system would use the job title only for matching but that the other details would be displayed on the screen for information. We had some doubts about how a straightforward word-matching system would cope with the complexities of occupation data but, nevertheless, thought it was worth investigating.

A combination of trigram and hierarchical coding was used in the study. The job title was used as a starting point for the matching process and the other occupation details were displayed in the top half of the screen. The best matches were displayed in the bottom half of the screen; the interviewers could scroll up and down through the suggestions, edit the job title and move up and down the coding frame hierarchy to get more information on the suitability of the codes.

5. Results from the experiment

When we designed the study we had certain expectations and theories about the possible outcomes. We thought that computer-assisted coding would be

- more reliable/consistent,
- possibly more accurate,
- less biased,
- quicker, and
- easier

than clerical coding. In addition we thought that

- a knowledge-based system would be 'better' than a simple word-matching system.

Reliability

Inter-coder consistency, also referred to as reliability, is the extent to which coders will assign the same code when supplied with the same information. If reliability is low then we cannot be sure that codes are not being assigned at random and would have little confidence in any survey estimates derived from the data. Perfect reliability would be obtained if all the interviewers assigned the same code when supplied with the same information. In this study we estimated reliability from the average agreement between all pairs of interviewers: a value of 1 indicating perfect reliability.

We hypothesised that reliability would be higher for computer-assisted coding compared to clerical coding as a computer-assisted system would provide the interviewers with a more consistent set of options to choose from. We thought that a knowledge-based system would present a more suitable shortlist of codes to select from than a word-match system and that therefore the interviewers would pick a code from that list more often, resulting in higher agreement amongst interviewers.

We estimated reliability for occupation codes at the most detailed level of the classification (SOC codes, 371 codes) and at the most aggregated level (Major group, 9 codes).

Table 2 Reliability for clerical and computer-assisted coding methods

	Clerical method	CASOC	Blaise III
--	-----------------	-------	------------

SOC	0.73	0.74	0.76
SOC Major group	0.85	0.86	0.88

We found no significant difference in reliability between the CASOC system and clerical coding. However, Blaise III coding was significantly more reliable than coding clerically (Table 2). These results were rather surprising and showed no support for either of our hypotheses. Although the Blaise III system was better than clerical coding, we could not conclude that computer-assisted coding, in general, would be more reliable.

Accuracy

By itself, reliability is not sufficient to assess data quality: reliability may be high even when the codes assigned are incorrect. For example, reliability will be perfect if all coders assign one code to all the occupations, regardless of whether the code is suitable or not. It is, therefore, important to measure the accuracy of coding.

We were not able to go back to respondents to check whether codes assigned to occupations were actually the most appropriate but we were able to look at the agreement between the codes assigned by the interviewers in the experiment and an expert occupation coder. The expert coder had coded thousands of occupations on the 1991 Census of Population and was very experienced in applying the coding rules and resolving queries. We defined coding accuracy as the percentage agreement between the interviewers and the expert.

Accuracy of coding will depend on whether the interviewers are influenced by the coding system used or whether they continue to code as they do clerically. In the first instance, accuracy will depend on how good the system is at suggesting appropriate codes; in the second instance, accuracy will remain the same as for clerical coding. We expected, by definition, that a knowledge-based system must be more accurate than a straight word-matching system.

Table 3 Percentage agreement with expert coder

	Clerical	CASOC	Blaise III
SOC	76.5	75.2	79.5
SOC Major groups	86.9	87.3	90.0

The accuracy of codes assigned using the CASOC system was not significantly different from codes assigned clerically but the Blaise III accuracy levels were significantly higher than both the clerical and CASOC levels (Table 3). Again, this was a surprising result: we had expected the CASOC system to produce the most accurate codes. Moreover, the accuracy of coding using the Blaise III system was as high as that obtained by office coders in the earlier study (Table 1), even at the SOC code level.

Bias

Bias occurs in the data where there is a systematic deviation from the true answer. If accuracy is very high then there is little room for bias in the results. However, where accuracy is not high we must examine whether

the incorrect codes are distributed randomly or form a pattern which may lead to bias in the resulting occupation estimates.

If a coder has a tendency to favour certain codes this will lead to individual coder bias. We normally assume that these biases will cancel out when aggregated over all the coders. This will result in an increase in the variance around the estimates (i.e. a decrease in precision) but no overall bias. When using a computer-assisted coding system, we might assume that the individual coder bias would decrease as interviewers have less scope to favour particular codes. Thus, the impact on the variance would be less than for a clerical process. However, we can no longer assume that there is no overall bias as the system itself may be influencing all the coders in the same way.

Assessing bias is usually very difficult, as the data we collect is usually subject to some sort of error (measurement, recall, sampling, etc.). In this study, we made the assumption that the codes assigned by the expert coder were the correct codes. We looked for evidence of bias by comparing the distributions of the occupation codes found in the experiment with the distribution obtained from the expert coder. Sample sizes from the experiment were not large enough to allow comparisons between distributions at SOC code level but it was possible to examine SOC Major groups.

Table 4 Distribution of codes allocated to Major groups

SOC Major Group		Expert	Clerical	CASOC	Blaise III
		%	%	%	%
1	Managers & Administrators	14.0	15.7	15.9	13.9
2	Professionals	6.3	6.5	6.2	7.1
3	Associate Prof. & Technical	12.0	11.5	11.6	11.7
4	Clerical & Secretarial	14.0	14.0	14.1	14.2
5	Craft and Related	12.0	12.2	12.2	12.1
6	Personal & Protective Service	10.0	9.8	9.5	9.4
7	Sales	9.0	8.0	7.6	8.4
8	Plant & Machine Operatives	9.7	9.7	9.7	9.6
9	Other Occupations	13.0	12.5	13.3	13.6

Each of the methods of coding demonstrates some bias, compared to the distribution of Major Groups derived from the expert coder (Table 4). However, the pattern of bias for each method is different so we cannot draw any conclusions about whether computer-assisted coding is more biased than clerical coding.

Speed

The interviewers were asked to record the time they spent coding the 300 occupations so that we could compare the relative speeds of the methods.

Computer-assisted coding was significantly faster than clerical coding and the Blaise III system was slightly faster than the CASOC/Blaise 2.5 system. Interviewers using the CASOC/Blaise 2.5 system and Blaise III system respectively took 13% and 23% less time to code the 300 occupations, than those using the clerical method.

Although it appears that significant time savings (and therefore cost savings) can be made by using computer-assisted coding, in practice codes will be assigned one at a time, rather than in a batch of 300. Therefore, time savings may not be so substantial in the field.

Ease of use

The interviewers taking part in the trials were asked to fill in questionnaires to provide us with feedback on the use of the systems. Most of the interviewers found using a computer-assisted system easier than coding clerically, although some found ambiguous occupations slightly more difficult to code. Most of the problems which arose were due to deficiencies with the coding frame and index, rather than the coding systems. Despite some reservations, all the interviewers thought that occupation coding should be computerised and looked forward to using such a system in the future.

6. Conclusion

The results of the experiment enabled us to draw some conclusions about the aims of the study.

- Improving quality of coded occupation data

Computer-assisted coding of occupation by interviewers does not necessarily improve the quality of coding, compared to clerical methods. Coding using the CASOC/Blaise 2.5 system developed for our trial was no more reliable or accurate than coding clerically. However, there were significant improvements in reliability and accuracy of occupation coding when using the Blaise III system.

These findings were contrary to our expectations, as we had assumed that the codes suggested by the knowledge-based CASOC system would be correct more often than for the Blaise III system, and that therefore the interviewers would be more likely to select from the suggested codes. We thought this behaviour would lead to higher reliability (through interviewers picking codes from the suggested list, rather than searching around for their own preferences) and to higher accuracy.

Further analysis of the data revealed that the CASOC system was more likely to suggest correct codes than the Blaise III system. When we looked at the first code suggested by the two systems for each occupation, we found that 62% of the CASOC first codes agreed with the expert-assigned codes, compared to only 50% of the first codes suggested by Blaise III. However, the interviewers did not necessarily select these codes, even when the first code suggested was the correct one. Using either system, interviewers selected the first code about 90% of the time when the code was correct. However, interviewers selected the first code, when incorrect, in 23% of cases for the CASOC system and in only 8% of cases for the Blaise III system. A possible explanation for this behaviour is that incorrect first codes suggested by the Blaise III system are more obviously implausible than those suggested by the knowledge-based CASOC

system, leading the interviewers to reject them more often and to search for better solutions.

All the coding systems (including clerical coding) showed some evidence of bias, compared to the codes assigned by the expert coder. There was a tendency for the interviewers using the CASOC/Blaise 2.5 system to agree with the first code suggested, regardless of whether it was correct or not (codes assigned by interviewers corresponded with the first code suggested in 64% of cases, compared to only 50% of cases in the Blaise III system). Although it was clear that interviewers were still using their judgement to assign codes and not just agreeing with suggestions, they may become more dependent on the system suggestions over time and so bias may increase. However, a computer-assisted system will allow us to have more control over the coding process than at present and will allow us to monitor coding quality more easily.

- Reduction in costs

Using either of the computer-assisted systems is likely to result in cost savings due to a reduction in the time interviewers spend coding. Coding with the Blaise III system was slightly faster than the CASOC/Blaise 2.5 system.

There are other cost considerations which must be taken into account besides field coding costs. There will be one-off costs associated with a change to computer-assisted coding, whatever system is used. These costs will include work on development of a system, training of interviewers, changing existing management systems, updating coding instructions and so on. There are also the costs of using and maintaining a system. The cost of introducing either of the systems used in our experiment would be fairly minimal (one of the reasons why we selected them in the first place). In the long term, therefore, it is quite likely that using a computer-assisted coding method will be cheaper than our existing method.

- Decreased interviewer burden

There was a definite decrease in interviewer burden when using the computer-assisted systems. The elements of the systems which the interviewers did not like are relatively easy to change (screen layout, speed and so on). We will be conducting some usability trials to find the best combination of layout, behaviour and speed.

- Ability to code during the interview

All the interviewers thought that coding during the interview, using the systems in the trial, would not be possible, even though they recognised the advantages that it would bring. Their main objection was that the systems were too slow and that spending a lot of time coding during the interview would interrupt the flow of the interview and disrupt the rapport they had with the respondent. They also felt that it was not appropriate to spend a lot of time on occupation when it may appear to the respondent to have little relevance to the survey topic.

In conclusion, we will need to carry out some further work to identify the factors which are relevant to an increase in the data quality, for example whether there was some aspect of the Blaise III coding system, not included in the CASOC system, which could be responsible for the increase in coding quality. We also need to analyse the results on bias in more detail, before we feel confident that we can make a change to our coding procedures.

However, this experiment clearly demonstrates that computer-assisted coding of occupation by interviewers is feasible, without a reduction in data quality, where coding is carried out after the interview. Interviewers can code occupation more consistently and accurately, faster, more cheaply and more easily using computer-assisted coding than using the standard clerical procedures.

Le questionnaire d'une enquête en deux phases avec tirage d'un échantillon représentatif.

François Clanché, INSEE, France

1. Résumé

L'enquête « Propriétaires bailleurs » a été réalisée en 1996/97 par l'INSEE, auprès de 3 000 ménages qui avaient déclaré, au cours de l'enquête nationale Logement, posséder un patrimoine immobilier locatif ou en avoir détenu un depuis 1993. Son but est de faire décrire au ménage l'ensemble des logements locatifs qu'il possède, les revenus qu'il en tire, la façon dont il a constitué son patrimoine et ses projets.

La difficulté technique de cette enquête est la suivante : Les "gros" bailleurs, ceux qui possèdent plus de deux logements, représentent à peine 15 % des ménages propriétaires, mais ils possèdent près de la moitié du parc. On ne peut évidemment pas leur demander de décrire dans le détail la totalité de leur patrimoine, sinon l'enquête durerait plusieurs heures. On a donc mis au point le système suivant : au début du questionnaire, le ménage énumère la totalité des logements qu'il possède et en donne les caractéristiques principales. Avec ces quelques indications, 2 logements "représentatifs" de l'ensemble sont choisis par le logiciel, que le ménage décrit dans le détail. Grâce à cette procédure, l'enquête, qui dure moins d'une demi-heure quand le ménage possède un seul logement, dépasse rarement les 50 minutes, même dans le cas de patrimoines importants.

Dans le programme en Blaise-III, on a procédé en 3 phases :

- la première partie de l'enquête lance des blocks identiques, courts, qui permettent de noter une description sommaire de chaque logement (adresse, date de construction, situation juridique, taille). Pour les appartements situés dans des immeubles, on travaille à un double niveau : on décrit une seule fois l'adresse, la date de construction et la situation juridique du bâtiment, puis on pose, dans les petits immeubles, 3 questions pour chaque appartement (dont la taille et des précisions permettant de retrouver un appartement), et pour les grands immeubles on demande une répartition par taille des appartements. Parmi les logements ainsi repérés, seuls certains sont susceptibles de faire l'objet d'une description détaillée dans la deuxième partie (les logements « éligibles »).

- à la fin de cette première partie, le programme crée en AUXFIELDS une table ayant autant de lignes que de logements éligibles (il y a donc des duplications dans le cas des immeubles) et comportant en colonne les variables caractéristiques de la représentativité souhaitée. Les "individus/logements" sont ensuite triés selon ces caractéristiques grâce à un algorithme optimisé pour ne pas trop ralentir la collecte sur un ordinateur peu performant. Enfin vient le tirage lui-même par une méthode systématique utilisant la fonction RANDOM.

- Au début de la deuxième partie du questionnaire, le programme récupère les informations de la première partie, essentielles pour filtrer les questions et adapter les textes par la suite. Cette information récupérée permet en particulier d'indiquer au ménage quels logements ont été désignés, avec un message du type "nous allons maintenant décrire l'appartement de 3 pièces 1er étage gauche avenue Charles de Gaulle à Lyon". La difficulté de cette phase vient de ce qu'il n'y a aucun lien systématique entre les rangs des blocks décrivant un même logement entre les parties 1 et 2.

Nous allons examiner plus en détail ces trois phases de l'enquête en nous attachant aux syntaxes Blaise « originales » et à leurs justifications. Les extraits de programme proposés ne sont que des illustrations, extraites d'un programme global beaucoup plus long et complexe, que l'auteur tient à la disposition des personnes intéressées.

2. Première phase : description rapide de tout le patrimoine

Cette phase comporte en fait deux parties : les logements « isolés » et les immeubles.

1. Les logements isolés sont décrits de façon simple dans des blocks de type BIsolé dans lesquels on demande notamment la commune où est situé le logement, son adresse, sa taille, et son occupation actuelle. Ces réponses permettent de renseigner, toujours à l'intérieur du block Isolé, la variable Eligi (qui distingue les logements susceptibles de faire l'objet d'une description détaillée) et la variable Tri qui servira plus tard pour s'assurer de la représentativité de l'échantillon des logements décrits en détail. A la fin de chaque logement « Isolé », la question Autre permet de savoir si on doit une nouvelle fois entamer une description ou si l'on doit passer à la partie Immeubles. Des compteurs de logements décrits (Mlo) et de logement isolés éligibles (Mis) sont incrémentés.

FIELDS

```
{ ** blocks de description des logements ** }  
Isole : ARRAY[0..10] OF BIsolé  
Mlo : 0..999 { nbre total de logt repérés }  
Mis : 0..10 {nbre d'isolés éligibles }
```

RULES

```
Isole[0]  
FOR n:=1 TO 10 DO  
  IF Isolé[n-1].Autre = oui  
  THEN Isole[n]  
    Mlo := Mlo + 1  
    IF Isole[n].Eligi = Oui  
    THEN Mis := Mis+1  
  ENDIF
```

```
ENDIF
ENDDO
```

2. Pour ce qui est des immeubles, le principe général est le même : on lance successivement un ou plusieurs blocks de type Bimme où certaines questions propres au bâtiment sont similaires au cas isolé. Si l'immeuble comporte moins de 10 appartements, l'occupation de l'appartement et sa taille sont détaillées appartement par appartement (sous-block Immeub.Appart), et on calcule une variable Tri pour chaque appartement. S'il compte plus de 9 appartements, on pose des questions sur la répartition entre les différentes tailles et on génère, toujours à l'intérieur du block, autant de variables Tri qu'il y a d'appartements dans l'immeuble en utilisant les données sur la répartition entre tailles. On calcule également le nombre de logements éligibles de l'immeuble (NbElig).

```
FIELDS
```

```
Immeub : ARRAY[0..3] OF Bimme
Elilmm  : 0..400 { nbre d'éligibles en immeubles }
Nim     : 0..3 {nombre d'immeubles repérés }
Neligi  : 0..350 { nbre total de logements éligibles : Mis + Elilmm }
```

```
RULES
```

```
Nelig := Mis
```

```
Immeub[0]
FOR z:= 1 TO 3 DO
IF Immeub[z-1].Autre = oui THEN
    Immeub[z]
    Nim := Nim + 1
    Mlo := Mlo + Immeub[z].Inolo
    IF Immeub[z].Eligi = Oui THEN
        Elilmm := Elilmm + Immeub[z].NbElig
        Neligi := Neligi + Immeub[z].NbElig
    ENDIF
ENDIF
ENDIF
ENDDO
```

3. Deuxième phase : choix des logements à décrire en détail

1. Pour rendre possible un tri et un tirage parmi tous les logements, on renseigne un vecteur d'auxfields comportant, pour chaque logement éligible (qu'il soit isolé ou situé dans un immeuble), la variable ATri. Cet ARRAY d'auxfields est indicé par i, i allant de 1 à Neligi, et la variable Aclasst est initialisée à i. On remplit cet ARRAY en deux temps, d'abord avec les logements isolés, ensuite avec les appartements.

```

AUXFIELDS
AClasst  : ARRAY[0..350] of 0..350
Anbcorr  : ARRAY[0..350] of 0..350
ATri     : ARRAY[1..350] of 1..7000
Anum     : ARRAY[1..350] of 0..2
AChoix   : ARRAY[1..350] of OuiNonType

```

RULES

```

FOR n := 1 TO 10 DO
  IF Isole[n].Eligi = Oui THEN
    FOR i:= 1 TO Mis DO
      IF Isole[n].k = i THEN
        ATri[i] := Isolé[n].Tri
        AClasst[i] := i
        ANbCorr[i] := Isolé[n].Nbc
      ENDIF
    ENDDO
  ENDIF
ENDDO

IF Nim > 0 THEN
  FOR z:= 1 to Nim DO
    IF Immeub[z].Eligi = Oui THEN
      For i:= Immeub[z].Debut TO Nelig DO
        IF i > (Immeub[z].Debut)
          AND i < (Immeub[z].Debut + Immeub[z].NbElig + 1)
          THEN
            AClasst[i] := i
            w := (i - Immeub[z].Debut)
            ATri[i] := Immeub[z].Tri[w]
          ENDIF
        ENDDO
      ENDDO
    ENDIF
  ENDDO
ENDIF

```

2. Tri des logements : un algorithme fait en sorte que la variable AClasst de chacun des Nelig logements corresponde au classement selon ATri. Pour cela on utilise la méthode de l'insertion successive des individus/logements, dont les principes sont les suivants:

Au début de chaque étape les (i-1) premiers individus sont triés, et on introduit le ième individu avec le classement i.

On compare la valeur de la variable ATri de l'individu i avec celle de l'individu classé (i-1)ème, c'est-à-dire dernier à l'issue de l'étape précédente.

Si elle est supérieure, le classement est alors entièrement bon et on peut passer à la phase suivante (introduction du (i+1)ème individu).

Sinon, on inverse les classements des deux derniers individus (i est classé (i-1)ème), et on compare son ATri avec celui de l'individu classé (i-2)ème, etc...

On ne passe à la phase suivante que quand l'individu i a trouvé sa « bonne » place.

```

RULES
FOR i:=2 TO nelig DO { évolution du niveau d'entrée : étapes }
  h := i -1
  FOR k:=1 TO h DO { évolution de la profondeur de comparaison }
    FOR q := 1 TO h DO { recherche du j avec qui comparer }
      IF AClasst[q] = (AClasst[i] - 1) THEN
        IF (ATri[q] > Atri[i])
          THEN devant := Aclasst[q]
              Aclasst[q] := Aclasst[i]
              AClasst[i] := devant
          ENDIF
        ENDIF
      ENDIF
    ENDDO
  ENDDO
ENDDO

```

Remarque : la syntaxe de l'algorithme est ici simplifiée. Pour gagner en performance on introduit des tests qui limitent le nombre de boucles parcourues aux seules comparaisons pertinentes.

3. Le choix des logements à décrire se fait ensuite par un tirage systématique, dont le pas est calculé en fonction du nombre de logements éligibles et le niveau de départ tiré au hasard par la fonction RANDOM. Notons la nécessité de « fixer » le valeur de Alea en utilisant l'ordre KEEP pour éviter qu'elle se modifie à tout moment.

Toujours dans la table d'auxfields, on parcourt l'ensemble des indices i de 1 à Nelig, et les logements « choisis » se distinguent par des variables Achoix renseignée à oui, et leur ordre de tirage est indiqué dans la variable Anum.

Si il y a moins de 3 logements éligibles, le nombre de logements détaillés décrits (NbDet) est égal à Nelig et le renseignement de Achoix et de Anum est plus aisé.

```

FIELDS
Alea , Pas , Debut : REAL , EMPTY
NbDet : 0..2 {nombre de descriptions détaillées}
Deta : ARRAY[1..2] of 0..350 {numéros des logements choisis}

RULES

{ ** Tirage de alea ** }
Alea.Keep
IF Alea = Empty THEN
  Alea := Random
ENDIF

{ ** calcul du nombre de blocks détaillés nbdet ** }
IF Nelig < 3 THEN NbDet := Nelig
  ELSE NbDet := 2
ENDIF

{ ** détermination des logements à décrire ** }
IF nelig < 3 THEN
  FOR i:= 1 TO nelig DO
    AChoix[i]:= oui Anum[i] := Aclasst[i]
    IF Deta[1] <> RESPONSE THEN Deta[1] := i
    ELSE Deta[2] := i
  ENDIF
ENDDO
ELSE
  Aclasst[0]:=0
  Pas := (Nelig/2)
  Debut := Pas*Alea
  FOR i:= 1 TO nelig DO
    IF ((Aclasst[i]-1 <= Debut) AND ( Debut < Aclasst[i]))
      THEN AChoix[i]:= oui Anum[i]:=1 deta[1] := i
    ELSEIF ((Aclasst[i]-1 <=(Debut+pas)) AND ((Debut+pas) < Aclasst[i]))
      THEN AChoix[i]:= oui Anum[i]:=2 Deta[2] := i
    ELSE AChoix[i]:= non Anum[i] := 0
  ENDIF

```

```
ENDDO
ENDIF
```

4. Troisième phase : lancement des descriptions détaillées

1. Une message paramétré indique à l'enquêté (et à l'enquêteur) le nombre de logements qu'il va maintenant avoir à décrire en détail.

```
FIELDS
```

```
Intro "Nous allons décrire en détail ^tt vous venez d'évoquer. ": STRING[1]
```

```
RULES
```

```
IF NbDet = Mlo THEN
  IF Mlo = 1 THEN tt := ' le logement que'
  ELSE tt := ' les '+str(NbDet)+' logements que'
ENDIF
ELSE
  IF NbDet = 1 THEN tt := ' un des '+str(Mlo)+' logements que'
  ELSE tt := ' '+str(NbDet)+' des '+str(Mlo)+' logements que'
ENDIF
ENDIF
Intro
```

2. Remarque : Il est absolument nécessaire de «verrouiller» le choix des logements au début de cette deuxième partie. En effet, si après avoir commencé la partie détaillée on modifie dans la première partie ne serait-ce qu'une caractéristique des logements décrits, les paramètres du tirage des logements vont être modifiés, et éventuellement le choix des logements tirés. Il y a donc un risque que deux variables d'un même block détaillé se réfèrent en fait à deux logements différents ! L'utilisation de l'ordre KEEP et de filtres du type « IF Intro = Empty THEN » fait en sorte que le choix des logements à décrire n'est plus remis en cause une fois la deuxième partie entamée.

3. Avant de poser les questions du block BDetaill (questions beaucoup plus nombreuses que dans le block BIsol: environ 20 minutes d'entretien contre moins de 5), il faut récupérer et noter dans ce block les informations recueillies dans la première partie concernant ce logement.

La difficulté de cette opération vient de ce qu'il n'y a aucun lien systématique entre les rangs des blocks décrivant un même logement entre les parties : le premier logement décrit en détail peut être le second, ou le troisième, décrit dans la première partie. Ce peut être un logement isolé, un appartement situé dans un petit immeuble, ou encore situé dans un grand immeuble.

Pour chaque indice j (j étant égal à 1, puis éventuellement à 2), on va donc d'abord parcourir les indices i (i allant de 1 à Nelig) pour trouver celui qui, à l'issue du tirage, a été choisi pour être décrit en jème (chercher i tel que Achoix[i] = oui et ANum[i] = j).

Si l'indice i concerné est inférieur ou égal au nombre d'isolés éligibles ($i < Mis$), on va aller chercher dans la première partie, parmi les logements isolés, celui qui a été décrit en nème position, n tenant compte des logements non éligibles éventuellement décrits avant ce logement. Ce

sont les caractéristiques de ce logement isolé qui seront affectées au block détaillé j. En particulier, son adresse et sa taille seront indiquées au tout début du block détaillé, lorsqu'il faudra indiquer au ménage sur quel logement portera la description (variable Detail[j].tex).

Si $i > \text{Mis}$, c'est que le logement choisi est issu d'un immeuble. On va alors parcourir les immeubles (z allant de 1 à Nim) pour repérer celui qui contient le logement en question. Une fois l'immeuble repéré, il faut trouver à l'intérieur de l'immeuble l'appartement concerné (indice k). Quand l'immeuble est petit, on est ramené à la situation des logements isolés et la récupération des caractéristiques ne pose pas de difficultés. Quand l'immeuble est grand, on détermine la taille de l'appartement à décrire en fonction de la répartition par tailles donnée dans la première partie et on génère un message du type « Nous allons maintenant décrire le 2ème appartement de 4 pièces de l'immeuble situé.... ».

Une fois ces opérations réalisées, le lancement du block détaillé lui-même est à la portée d'un programmeur BLAISE débutant !

FIELDS

Detail : ARRAY[1..2] Of BDetail

RULES

FOR j:= 1 TO Nbdet DO

{ ** récupération des données de la première partie ** }

FOR i:= 1 TO nelig DO

IF AChoix[i] = oui AND ANum[i] = j THEN

IF $i \leq \text{Mis}$ THEN { données issues de logements isolés }

n := (i+ ANbCorr[i])

Detail[j].num := j

Detail[j].numlo := n

Detail[j].tex := 'Nous allons maintenant décrire '+Isole[n].tex+

Isole[n].ladres+ 'à '+Isole[n].Commune+Isole[n].LCom+'.'

Detail[j].app := Isole[n].Appma

Detail[j].Inopi := Isole[n].Nopi

ELSEIF $i > \text{Mis}$ THEN { données issues d'immeubles }

FOR z := 1 TO nim DO

IF ($i > \text{Immeub}[z].\text{debut}$) AND ($i < (\text{Immeub}[z].\text{debut} + \text{Immeub}[z].\text{Nbelig} + 1)$)

THEN

Detail[j].num := j

Detail[j].numlo := (Mis+z)

k := (i - immeub[z].debut)

IF $\text{Immeub}[z].\text{Inolo} < 10$ THEN { petits immeubles }

FOR y := 1 TO 10 DO

IF $\text{Immeub}[z].\text{Appart}[y].\text{ClassEli} = k$

THEN v := y

ENDIF

ENDDO

IF $\text{Immeub}[z].\text{Appart}[v].\text{INop} = 1$ THEN tx := 'le studio'

ELSE tx := 'l'appartement de+STR(Immeub[z].Appart[v].INop)+' pièces'

ENDIF

Detail[j].tex := 'Nous allons maintenant décrire '+tx+' situé '+

Immeub[z].appart[v].lbatesc + ' de l'immeuble situé '+ Immeub[z].MAdres+ 'à '+

Immeub[z].Commune+

Immeub[z].LCom+'.'

Detail[j].LNopi := Immeub[z].Appart[v].INop

ELSE { grands immeubles }

y := INT(k/Immeub[z].max)

IF $y \leq \text{Immeub}[z].\text{Inop1}$ THEN v:= y tx:= 'studio' Detail[j].LNopi := 1

ELSEIF $y \leq \text{Immeub}[z].\text{Inop22}$ THEN

v:=(y-Immeub[z].Inop1) tx := 'appartement de 2 pièces' Detail[j].LNopi := 2

```

ELSEIF y <= Immeub[z].Inop33 THEN
  v:=(y-Immeub[z].Inop22) tx := 'appartement de 3 pièces' Detail[j].Lnopi := 3
ELSEIF y <= Immeub[z].Inop44 THEN
  v:=(y-Immeub[z].Inop33) tx := 'appartement de 4 pièces' Detail[j].Lnopi := 4
ELSEIF y <= Immeub[z].Inop55 THEN
  v:=(y-Immeub[z].Inop44) tx := 'appartement de 5 pièces' Detail[j].Lnopi := 5
ELSE v:= (y-Immeub[z].Inop55) tx := 'appartement de plus de 5 pièces'
ENDIF
IF (v) = 1 THEN ième := 'er '
  ELSE ième := ' ième '
ENDIF
Detail[j].tex := 'Nous allons maintenant décrire le '+str(v)
+ième+tx+' de l'immeuble situé '+Immeub[z].MAdres+' à '
+Immeub[z].Commune+Immeub[z].LCom+'.'
ENDIF { fin du filtre sur la taille de l'immeuble }

Detail[j].Linet := Immeub[z].Inbet
ENDIF { fin du filtre sur z }
Detail[j].imm := Oui
Detail[j].App := Ap
ENDDO {fin de la boucle sur z}

ENDIF { fin du filtre la différence isolés / immeubles }

ENDIF { fin du filtre sur i = j }

ENDDO {fin de la boucle sur i }

{** Lancement du block de description détaillée **}
Detail[j]

ENDDO {fin de la boucle sur j}

```


Le questionnaire d'une enquête en deux phases avec tirage d'un échantillon représentatif.

François Clanché, INSEE, France

1. Résumé

L'enquête « Propriétaires bailleurs » a été réalisée en 1996/97 par l'INSEE, auprès de 3 000 ménages qui avaient déclaré, au cours de l'enquête nationale Logement, posséder un patrimoine immobilier locatif ou en avoir détenu un depuis 1993. Son but est de faire décrire au ménage l'ensemble des logements locatifs qu'il possède, les revenus qu'il en tire, la façon dont il a constitué son patrimoine et ses projets.

La difficulté technique de cette enquête est la suivante : Les "gros" bailleurs, ceux qui possèdent plus de deux logements, représentent à peine 15 % des ménages propriétaires, mais ils possèdent près de la moitié du parc. On ne peut évidemment pas leur demander de décrire dans le détail la totalité de leur patrimoine, sinon l'enquête durerait plusieurs heures. On a donc mis au point le système suivant : au début du questionnaire, le ménage énumère la totalité des logements qu'il possède et en donne les caractéristiques principales. Avec ces quelques indications, 2 logements "représentatifs" de l'ensemble sont choisis par le logiciel, que le ménage décrit dans le détail. Grâce à cette procédure, l'enquête, qui dure moins d'une demi-heure quand le ménage possède un seul logement, dépasse rarement les 50 minutes, même dans le cas de patrimoines importants.

Dans le programme en Blaise-III, on a procédé en 3 phases :

- la première partie de l'enquête lance des blocks identiques, courts, qui permettent de noter une description sommaire de chaque logement (adresse, date de construction, situation juridique, taille). Pour les appartements situés dans des immeubles, on travaille à un double niveau : on décrit une seule fois l'adresse, la date de construction et la situation juridique du bâtiment, puis on pose, dans les petits immeubles, 3 questions pour chaque appartement (dont la taille et des précisions permettant de retrouver un appartement), et pour les grands immeubles on demande une répartition par taille des appartements. Parmi les logements ainsi repérés, seuls certains sont susceptibles de faire l'objet d'une description détaillée dans la deuxième partie (les logements « éligibles »).

- à la fin de cette première partie, le programme crée en AUXFIELDS une table ayant autant de lignes que de logements éligibles (il y a donc des duplications dans le cas des immeubles) et comportant en colonne les variables caractéristiques de la représentativité souhaitée. Les "individus/logements" sont ensuite triés selon ces caractéristiques grâce à

un algorithme optimisé pour ne pas trop ralentir la collecte sur un ordinateur peu performant. Enfin vient le tirage lui-même par une méthode systématique utilisant la fonction RANDOM.

- Au début de la deuxième partie du questionnaire, le programme récupère les informations de la première partie, essentielles pour filtrer les questions et adapter les textes par la suite. Cette information récupérée permet en particulier d'indiquer au ménage quels logements ont été désignés, avec un message du type "nous allons maintenant décrire l'appartement de 3 pièces 1er étage gauche avenue Charles de Gaulle à Lyon". La difficulté de cette phase vient de ce qu'il n'y a aucun lien systématique entre les rangs des blocks décrivant un même logement entre les parties 1 et 2.

Nous allons examiner plus en détail ces trois phases de l'enquête en nous attachant aux syntaxes Blaise « originales » et à leurs justifications. Les extraits de programme proposés ne sont que des illustrations, extraites d'un programme global beaucoup plus long et complexe, que l'auteur tient à la disposition des personnes intéressées.

2. Première phase : description rapide de tout le patrimoine

Cette phase comporte en fait deux parties : les logements « isolés » et les immeubles.

1. Les logements isolés sont décrits de façon simple dans des blocks de type BIsolé dans lesquels on demande notamment la commune où est situé le logement, son adresse, sa taille, et son occupation actuelle. Ces réponses permettent de renseigner, toujours à l'intérieur du block Isolé, la variable Eligi (qui distingue les logements susceptibles de faire l'objet d'une description détaillée) et la variable Tri qui servira plus tard pour s'assurer de la représentativité de l'échantillon des logements décrits en détail. A la fin de chaque logement « Isolé », la question Autre permet de savoir si on doit une nouvelle fois entamer une description ou si l'on doit passer à la partie Immeubles. Des compteurs de logements décrits (Mlo) et de logement isolés éligibles (Mis) sont incrémentés.

```
FIELDS
{ ** blocks de description des logements ** }
Isole : ARRAY[0..10] OF BIsolé
Mlo   : 0..999 { nbre total de logt repérés }
Mis   : 0..10  {nbre d'isolés éligibles }

RULES

Isole[0]
FOR n:=1 TO 10 DO
  IF Isolé[n-1].Autre = oui
    THEN Isole[n]
      Mlo := Mlo + 1
      IF Isole[n].Eligi = Oui
        THEN Mis := Mis+1
    ENDIF
  ENDIF
ENDDO
```

2. Pour ce qui est des immeubles, le principe général est le même : on lance successivement un ou plusieurs blocks de type Bimme où certaines

questions propres au bâtiment sont similaires au cas isolé. Si l'immeuble comporte moins de 10 appartements, l'occupation de l'appartement et sa taille sont détaillées appartement par appartement (sous-block Immeub.Appart), et on calcule une variable Tri pour chaque appartement. S'il compte plus de 9 appartements, on pose des questions sur la répartition entre les différentes tailles et on génère, toujours à l'intérieur du block, autant de variables Tri qu'il y a d'appartements dans l'immeuble en utilisant les données sur la répartition entre tailles. On calcule également le nombre de logements éligibles de l'immeuble (NbElig).

FIELDS

```
Immeub : ARRAY[0..3] OF Bimme
Elilmm  : 0..400 { nbre d'éligibles en immeubles }
Nim     : 0..3 {nombre d'immeubles repérés }
Neligm  : 0..350 { nbre total de logements éligibles : Mis + Elilmm }
```

RULES

```
Nelig := Mis

Immeub[0]
FOR z:= 1 TO 3 DO
IF Immeub[z-1].Autre = oui THEN
    Immeub[z]
    Nim := Nim + 1
    Mlo := Mlo + Immeub[z].Inolo
    IF Immeub[z].Eligi =Oui THEN
        Elilmm := Elilmm + Immeub[z].NbElig
        Neligm := Neligm + Immeub[z].NbElig
    ENDIF
ENDIF
ENDDO
```

3. Deuxième phase : choix des logements à décrire en détail

1. Pour rendre possible un tri et un tirage parmi tous les logements, on renseigne un vecteur d'auxfields comportant, pour chaque logement éligible (qu'il soit isolé ou situé dans un immeuble), la variable ATri. Cet ARRAY d'auxfields est indicé par i, i allant de 1 à Neligm, et la variable Aclasst est initialisée à i. On remplit cet ARRAY en deux temps, d'abord avec les logements isolés, ensuite avec les appartements.

```

AUXFIELDS
AClasst  : ARRAY[0..350] of 0..350
Anbcorr  : ARRAY[0..350] of 0..350
ATri     : ARRAY[1..350] of 1..7000
Anum     : ARRAY[1..350] of 0..2
AChoix   : ARRAY[1..350] of OuiNonType

```

RULES

```

FOR n := 1 TO 10 DO
  IF Isole[n].Eligi = Oui THEN
    FOR i:= 1 TO Mis DO
      IF Isole[n].k = i THEN
        ATri[i] := Isolé[n].Tri
        AClasst[i] := i
        ANbCorr[i] := Isolé[n].Nbc
      ENDIF
    ENDDO
  ENDIF
ENDDO

IF Nim > 0 THEN
  FOR z:= 1 to Nim DO
    IF Immeub[z].Eligi = Oui THEN
      For i:= Immeub[z].Debut TO Nelig DO
        IF i > (Immeub[z].Debut)
          AND i < (Immeub[z].Debut + Immeub[z].NbElig + 1)
          THEN
            AClasst[i] := i
            w := (i - Immeub[z].Debut)
            ATri[i] := Immeub[z].Tri[w]
          ENDIF
        ENDDO
      ENDDO
    ENDIF
  ENDDO
ENDIF

```

2. Tri des logements : un algorithme fait en sorte que la variable AClasst de chacun des Nelig logements corresponde au classement selon ATri. Pour cela on utilise la méthode de l'insertion successive des individus/logements, dont les principes sont les suivants:

Au début de chaque étape les (i-1) premiers individus sont triés, et on introduit le ième individu avec le classement i.

On compare la valeur de la variable ATri de l'individu i avec celle de l'individu classé (i-1)ème, c'est-à-dire dernier à l'issue de l'étape précédente.

Si elle est supérieure, le classement est alors entièrement bon et on peut passer à la phase suivante (introduction du (i+1)ème individu).

Sinon, on inverse les classements des deux derniers individus (i est classé (i-1)ème), et on compare son ATri avec celui de l'individu classé (i-2)ème, etc...

On ne passe à la phase suivante que quand l'individu i a trouvé sa « bonne » place.

RULES

```
FOR i:=2 TO nelig DO { évolution du niveau d'entrée : étapes }
  h := i -1
  FOR k:=1 TO h DO { évolution de la profondeur de comparaison }
    FOR q := 1 TO h DO { recherche du j avec qui comparer }
      IF AClasst[q] = (AClasst[i] - 1) THEN
        IF (ATri[q] > Atri[i])
          THEN devant := Aclasst[q]
              Aclasst[q] := Aclasst[i]
              AClasst[i] := devant
        ENDIF
      ENDIF
    ENDDO
  ENDDO
ENDDO
```

Remarque : la syntaxe de l'algorithme est ici simplifiée. Pour gagner en performance on introduit des tests qui limitent le nombre de boucles parcourues aux seules comparaisons pertinentes.

3. Le choix des logements à décrire se fait ensuite par un tirage systématique, dont le pas est calculé en fonction du nombre de logements éligibles et le niveau de départ tiré au hasard par la fonction RANDOM. Notons la nécessité de « fixer » le valeur de Alea en utilisant l'ordre KEEP pour éviter qu'elle se modifie à tout moment.

Toujours dans la table d'auxfields, on parcourt l'ensemble des indices i de 1 à Nelig, et les logements « choisis » se distinguent par des variables Achoix renseignée à oui, et leur ordre de tirage est indiqué dans la variable Anum.

Si il y a moins de 3 logements éligibles, le nombre de logements détaillés décrits (NbDet) est égal à Nelig et le renseignement de Achoix et de Anum est plus aisé.

FIELDS

```
Alea , Pas , Debut : REAL , EMPTY
NbDet : 0..2 {nombre de descriptions détaillées}
Deta : ARRAY[1..2] of 0..350 {numéros des logements choisis}
```

RULES

```
{ ** Tirage de alea ** }
Alea.Keep
IF Alea = Empty THEN
  Alea := Random
ENDIF

{ ** calcul du nombre de blocks détaillés nbdet ** }
IF Nelig < 3 THEN NbDet := Nelig
  ELSE NbDet := 2
ENDIF

{ ** détermination des logements à décrire ** }
IF nelig < 3 THEN
  FOR i:= 1 TO nelig DO
    AChoix[i]:= oui Anum[i] := Aclasst[i]
    IF Deta[1] <> RESPONSE THEN Deta[1] := i
    ELSE Deta[2] := i
  ENDIF
ENDDO
ELSE
  Aclasst[0]:=0
  Pas := (Nelig/2)
  Debut := Pas*Alea
  FOR i:= 1 TO nelig DO
    IF ((Aclasst[i]-1 <= Debut) AND ( Debut < Aclasst[i]))
      THEN AChoix[i]:= oui Anum[i]:=1 deta[1] := i
    ELSEIF ((Aclasst[i]-1 <=(Debut+pas)) AND ((Debut+pas) < Aclasst[i]))
      THEN AChoix[i]:= oui Anum[i]:=2 Deta[2] := i
    ELSE AChoix[i]:= non Anum[i] := 0
  ENDIF
ENDIF
```

```
ENDDO
ENDIF
```

4. Troisième phase : lancement des descriptions détaillées

1. Une message paramétré indique à l'enquêté (et à l'enquêteur) le nombre de logements qu'il va maintenant avoir à décrire en détail.

```
FIELDS
```

```
Intro "Nous allons décrire en détail ^tt vous venez d'évoquer. ": STRING[1]
```

```
RULES
```

```
IF NbDet = Mlo THEN
  IF Mlo = 1 THEN tt := ' le logement que'
  ELSE tt := ' les '+str(NbDet)+' logements que'
ENDIF
ELSE
  IF NbDet = 1 THEN tt := ' un des '+str(Mlo)+' logements que'
  ELSE tt := ' '+str(NbDet)+' des '+str(Mlo)+' logements que'
ENDIF
ENDIF
Intro
```

2. Remarque : Il est absolument nécessaire de «verrouiller» le choix des logements au début de cette deuxième partie. En effet, si après avoir commencé la partie détaillée on modifie dans la première partie ne serait-ce qu'une caractéristique des logements décrits, les paramètres du tirage des logements vont être modifiés, et éventuellement le choix des logements tirés. Il y a donc un risque que deux variables d'un même block détaillé se réfèrent en fait à deux logements différents ! L'utilisation de l'ordre KEEP et de filtres du type « IF Intro = Empty THEN » fait en sorte que le choix des logements à décrire n'est plus remis en cause une fois la deuxième partie entamée.

3. Avant de poser les questions du block BDetaill (questions beaucoup plus nombreuses que dans le block BIsol: environ 20 minutes d'entretien contre moins de 5), il faut récupérer et noter dans ce block les informations recueillies dans la première partie concernant ce logement.

La difficulté de cette opération vient de ce qu'il n'y a aucun lien systématique entre les rangs des blocks décrivant un même logement entre les parties : le premier logement décrit en détail peut être le second, ou le troisième, décrit dans la première partie. Ce peut être un logement isolé, un appartement situé dans un petit immeuble, ou encore situé dans un grand immeuble.

Pour chaque indice j (j étant égal à 1, puis éventuellement à 2), on va donc d'abord parcourir les indices i (i allant de 1 à Nelig) pour trouver celui qui, à l'issue du tirage, a été choisi pour être décrit en jème (chercher i tel que Achoix[i] = oui et ANum[i] = j).

Si l'indice i concerné est inférieur ou égal au nombre d'isolés éligibles ($i < Mis$), on va aller chercher dans la première partie, parmi les logements isolés, celui qui a été décrit en nème position, n tenant compte des logements non éligibles éventuellement décrits avant ce logement. Ce

sont les caractéristiques de ce logement isolé qui seront affectées au block détaillé j. En particulier, son adresse et sa taille seront indiquées au tout début du block détaillé, lorsqu'il faudra indiquer au ménage sur quel logement portera la description (variable Detail[j].tex).

Si $i > \text{Mis}$, c'est que le logement choisi est issu d'un immeuble. On va alors parcourir les immeubles (z allant de 1 à Nim) pour repérer celui qui contient le logement en question. Une fois l'immeuble repéré, il faut trouver à l'intérieur de l'immeuble l'appartement concerné (indice k). Quand l'immeuble est petit, on est ramené à la situation des logements isolés et la récupération des caractéristiques ne pose pas de difficultés. Quand l'immeuble est grand, on détermine la taille de l'appartement à décrire en fonction de la répartition par tailles donnée dans la première partie et on génère un message du type « Nous allons maintenant décrire le 2ème appartement de 4 pièces de l'immeuble situé.... ».

Une fois ces opérations réalisées, le lancement du block détaillé lui-même est à la portée d'un programmeur BLAISE débutant !

FIELDS

Detail : ARRAY[1..2] Of BDetail

RULES

FOR j:= 1 TO Nbdet DO

{ ** récupération des données de la première partie ** }

FOR i:= 1 TO nelig DO

IF AChoix[i] = oui AND ANum[i] = j THEN

IF $i \leq \text{Mis}$ THEN { données issues de logements isolés }

n := (i+ ANbCorr[i])

Detail[j].num := j

Detail[j].numlo := n

Detail[j].tex := 'Nous allons maintenant décrire '+Isole[n].tex+

Isole[n].ladres+ 'à '+Isole[n].Commune+Isole[n].LCom+'.'

Detail[j].app := Isole[n].Appma

Detail[j].Inopi := Isole[n].Nopi

ELSEIF $i > \text{Mis}$ THEN { données issues d'immeubles }

FOR z := 1 TO nim DO

IF ($i > \text{Immeub}[z].\text{debut}$) AND ($i < (\text{Immeub}[z].\text{debut} + \text{Immeub}[z].\text{Nbelig} + 1)$)

THEN

Detail[j].num := j

Detail[j].numlo := (Mis+z)

k := (i - immeub[z].debut)

IF $\text{Immeub}[z].\text{Inolo} < 10$ THEN { petits immeubles }

FOR y := 1 TO 10 DO

IF $\text{Immeub}[z].\text{Appart}[y].\text{ClassEli} = k$

THEN v := y

ENDIF

ENDDO

IF $\text{Immeub}[z].\text{Appart}[v].\text{INop} = 1$ THEN tx := 'le studio'

ELSE tx := 'l'appartement de+STR(Immeub[z].Appart[v].INop)+' pièces'

ENDIF

Detail[j].tex := 'Nous allons maintenant décrire '+tx+' situé '+

Immeub[z].appart[v].lbatesc + ' de l'immeuble situé '+ Immeub[z].MAdres+ 'à '+

Immeub[z].Commune+

Immeub[z].LCom+'.'

Detail[j].LNopi := Immeub[z].Appart[v].INop

ELSE { grands immeubles }

y := INT(k/Immeub[z].max)

IF $y \leq \text{Immeub}[z].\text{Inop1}$ THEN v:= y tx:= 'studio' Detail[j].LNopi := 1

ELSEIF $y \leq \text{Immeub}[z].\text{Inop22}$ THEN

v:=(y-Immeub[z].Inop1) tx := 'appartement de 2 pièces' Detail[j].LNopi := 2

```

ELSEIF y <= Immeub[z].Inop33 THEN
  v:=(y-Immeub[z].Inop22) tx := 'appartement de 3 pièces' Detail[j].Lnopi := 3
ELSEIF y <= Immeub[z].Inop44 THEN
  v:=(y-Immeub[z].Inop33) tx := 'appartement de 4 pièces' Detail[j].Lnopi := 4
ELSEIF y <= Immeub[z].Inop55 THEN
  v:=(y-Immeub[z].Inop44) tx := 'appartement de 5 pièces' Detail[j].Lnopi := 5
ELSE v:= (y-Immeub[z].Inop55) tx := 'appartement de plus de 5 pièces'
ENDIF
IF (v) = 1 THEN ième := 'er '
  ELSE ième := ' ième '
ENDIF
Detail[j].tex := 'Nous allons maintenant décrire le '+str(v)
+ième+tx+' de l'immeuble situé '+Immeub[z].MAdres+' à '
+Immeub[z].Commune+Immeub[z].LCom+'.'
ENDIF { fin du filtre sur la taille de l'immeuble }

Detail[j].Linet := Immeub[z].Inbet
ENDIF { fin du filtre sur z }
Detail[j].imm := Oui
Detail[j].App := Ap
ENDDO {fin de la boucle sur z}

ENDIF { fin du filtre la différence isolés / immeubles }

ENDIF { fin du filtre sur i = j }

ENDDO {fin de la boucle sur i }

{** Lancement du block de description détaillée **}
Detail[j]

ENDDO {fin de la boucle sur j}

```

Interactive coding of economic activity using trigram search in BLAISE III

Omar S. Hardarson, Statistics Iceland

Abstract

This paper presents the results of a study at Statistics Iceland on the feasibility of using trigram-coding to code economic activity. The results show that the total success rate, i.e. the rate of producing acceptable codes, of trigram-coding and residual manual coding can be expected to range between 89.1% and 97.7%, depending on the classification level and type of population. The total time of such coding is less than the total time of exclusively manual coding. The paper also shows that while there was no difference in the coding success between experienced and inexperienced interviewers, the efficiency of trigram-coding is mainly determined by the quality of the index of descriptions.

1. Introduction

Any survey of the population can be seen as a systematic codification of verbal data from respondents. Answers are sometimes coded as ayes or nays but more often than not the response categories are more varied than that. In computer assisted interviewing, however, it is not advisable to exceed more than one screenful of alternatives for any one question. The step towards a more complex structure of alternatives is nevertheless only a question of degree, albeit highly dependent on the available software algorithms as well as coding dictionaries.

Versions 2.0 to 2.4 of the BLAISE CAI system provided facilities for complex classifications with hierarchical coding or alphabetical search in lists. Version 2.5, and later BLAISE III 1.1, offer trigram-coding in addition to the former two.

Roessingh and Bethlehem (1993) compare the three types of coding facilities provided by BLAISE when coding articles in the Dutch Expenditure Survey. They conclude that at least for this subject matter the trigram-coding is better than the alphabetical, but hierarchical coding is by far the best. They point out, however, that their coders were very experienced, knowing many of the codes by heart, as well as the fact that hierarchical coding is largely dependent on the availability of a "workable classification of items" (p. 132). They also find that trigram-coding can be successful even if the coders have no experience with it.

With regard to trigram-coding Roessingh and Bethlehem (1993) point out that its success is determined by the quality of the dictionary or list of descriptions at hand. The coding file for trigram-coding, however, does not have to be as long as the list for alphabetical coding where one needs to have multiple entries of the same items in a different word order. The importance of the coding file is emphasised by others, e.g., Lyberg and Dean (1992), Bushnell (1993) and Martin (1993).

Lyberg and Dean (1992) offer the dictum that “[a]utomated coding and residual manual coding should be less expensive than the same operation being 100% manually coded” (p. 23). This may not hold true for interactive coding. Bushnell (1993) lists 13 characteristics for a computer assisted coding system. An “ideal” system should include as many of these as possible. Cost effectiveness is only one of these criteria.

In theory computer assisted coding should be better than manual coding of data generated by the same interviews. This is because the interviewers can solve any ambiguities by asking the respondent. The manual coder is on the other hand restricted by the information already recorded. This advantage of interactive coding is, however, offset by other factors. The interviewers may not be sufficiently trained, they have less resources than the manual coder and the time to code may be unacceptably long (see Martin 1993). The decision to use interactive coding has thus to take all these factors into account.

From 1992 Statistics Iceland has conducted its Labour Force Survey as well as other surveys with the help of the BLAISE CAI system. Versions 2.3x, 2.4 and 2.5 were used until early spring 1996, when the agency introduced BLAISE III 1.1 in its survey work. Until 1996 the search and look-up facilities in the BLAISE system were used very sparingly and almost exclusively for the coding of labour unions in the Labour Force Survey. This was primarily due to the difficulty of the BLAISE 2.x system in handling non-ASCII characters in strings. This drawback was corrected in the BLAISE III version. In April 1996 we introduced trigram-coding for coding the municipality of the local unit and the ISCED level of completed further education in the LFS. Both of these proved to greatly facilitate the interview process. We were therefore keenly interested in knowing how the trigram-coding module would perform when coding complex concepts like economic activity of local units.

The study of the efficiency of trigram-coding was carried out in two separate steps. The first step was part of a Pilot Survey for the Small Business Survey in the summer of 1996. The second step consisted of an experiment within the November 1996 Labour Force Survey. The first two parts of the study were aimed primarily at estimating the reliability of the trigram-coding vis-à-vis manual coding, whereas the third part focused on examining the relative cost of this type of coding. The Small Business Survey as well as the Labour Force Survey could also be used to assess the results of the Pilot Survey.

2. Design of the study

The Small Business Pilot Survey (SBPS). The first part of the study was based on a pilot survey for the Small Business Survey in July 1996. The

purpose of the pilot survey was partly to examine the feasibility of coding economic activity of the local unit during the interview by using the trigram-coding facility in BLAISE III 1.1.

A random sample of 400 businesses was drawn from a part of the Small Business Survey population. Of these seven were no longer part of the frame, and 26 others were dropped from the sample for other reasons. There were thus 367 individuals in the final pilot survey sample.

The Pilot Survey was conducted on 11, 12 and 14 July between 17.00 to 22.00. Seven interviewers were hired to conduct the survey, four on the first two days and then three in addition on 14 July. The list of descriptions was taken directly from the index of the ISAT-95 manual, with 3,831 entries.

Contact was made with 243 individuals, of which twelve did not have any business activity in the reference period (the year 1995), leaving 231 cases to be coded.

The Small Business Survey (SBS) was primarily aimed at assigning the new ISAT-95 codes⁹ to the activities of unincorporated businesses. The total population of unincorporated businesses exceeded 28 thousand. The majority of these had ISIC¹⁰ codes assigned, which could be directly matched with ISAT-95 codes. A total of 8,059 businesses were coded in the survey. The file of coding descriptions from the Pilot Survey was used for trigram-coding with the addition of professions which could be matched with an economic activity code, along with some other minor changes. The total number of entries was 4,222 coding descriptions.

The assigned codes from either the manual or the interactive coding were checked by a separate coder. If this coder found any ambiguity he would look closer at the data and make necessary changes to the codes. In some instances the code was left unchanged. In this study the results of the review process are used to determine the acceptability of the initial activity codes.

The Labour Force Survey (LFS). The second part of the survey consisted of an experiment within the November Labour Force Survey 1996. The Icelandic LFS is carried out twice a year, in April and November. It is a panel survey with some 4,400 participants with a net response rate of 90%. Approximately three quarters of the respondents are carried to any subsequent survey. The panel design enables the technique of dependent interviews in which the answers from a previous survey are confirmed or changed in the interview. In the survey a total

⁹ The ISAT-95 classification is the Icelandic equivalent of the NACE (rev. 1) classification of industry, where the first 4 digits are more or less compatible with international standards while the fifth digit is exclusively Icelandic.

¹⁰ A three digit code based on the international standard, ISIC, rev. 1, which had been in use at Statistics Iceland since the late sixties.

1,774 cases were previously recorded and confirmed, leaving 1,227 cases to be coded afresh.

The experiment was designed so that half of the respondents answering the question on economic activity of the local unit in their first job were randomly assigned to two groups:

The responses of the first group were recorded by the interviewers, and then the resulting text was used for the initial search in the coding dictionary using the trigram-coding utility in BLAISE III 1.1. If the interviewers did not succeed in finding a relevant code they were asked to record a further description of the business activity in a separate field of the type OPEN.

The second group, or the control group, was only given the question on the main functions of the local unit, which was then coded by an experienced coder, using an MS Access form to review all relevant information provided by the respondent (such as age, gender, education, name of the firm, place of activity, occupation and employment status) as well as access to the Icelandic Business Register and the ISAT-95 dictionary of descriptions. The expert coder also coded the cases where the interviewers failed to assign a code.

In the LFS economic activity is coded to the level of the fourth digit, even if the required precision level need only correspond to the first two digits. Nevertheless the same list of descriptions and five digit codes were used in the LFS as in the SBS. Out of 23 interviewers 10 had participated in the SBS. These were designated as experienced interviewers, even if many of the remaining interviewers had previous experience with the LFS. The total number of cases which were coded either manually or interactively was 1,227.

The inexperienced interviewers were given the same type of training in coding economic activity as the experienced interviewers. The training consisted of a short lecture on the concepts and structure of the ISAT-95 classification frame and hands-on practice, totalling four hours prior to the survey.

The procedure of reviewing and editing the activity codes in the SBS as described above applied also to the LFS.

3. Accuracy of trigram-coding vs manual coding

The assignment of ISAT-95 codes in the SBPS was made with the trigram-coding facility of BLAISE III 1.1. On completion of the coding the respondents were requested to confirm whether they had been allocated the correct code. In addition, the interviewers recorded the relevant information on the activity in a field of the type OPEN. An experienced coder was subsequently given the task of coding the activities afresh without having access to the results of the trigram-coding or earlier ISIC codes.

The two resulting codes were then compared. If the codes were identical, we assumed that they were acceptable. If the codes differed, the author

then reviewed the codes and coded them as acceptable or unacceptable. If the recorded data was too scant in order to code with precision to the fifth digit the former ISIC code, if available, was used to assess the acceptability of the result.

Table 1 shows the results of the review. Using the data recorded by the interviewers, the expert coder could assign 222 codes out of 231 or 96.1% at the five digit level, while the interviewers assigned 213 codes or 92.2%. All of the cases where the expert coder failed to assign a code were also the cases where the interviewers did not record sufficient information to assess the code without resorting to the existing ISIC codes. The acceptance rate of the interactive coding and manual coding at the five digit level is 95.1% and 97.7%, respectively.

Table 1. Results of the interactive and manual coding. SBPS 1996.

	Interactive coding			Manual coding		
	5 digits	4 digits	2 digits	5 digits	4 digits	2 digits
Acceptable	196	197	205	217	220	227
Unacceptable	10	9	5	5	5	2
Insufficient information	7	7	3	0	0	0
Not coded	18	18	18	9	6	2
Total number of cases	231	231	231	231	231	231
Hit rate ¹	92.2	92.2	92.2	96.1	97.4	99.1
Acceptance rate ²	95.1	95.6	97.6	97.7	97.8	99.1
Success rate ³	87.7	88.2	90.0	93.9	95.2	98.3

¹ The number of assigned codes over the total number of cases

² The number of accepted codes over the number of assigned codes

³ Hit rate times acceptance rate

In an actual interview environment the cases that cannot be coded interactively will be referred to manual coding. The above results indicate that trigram-coding and residual manual coding produce a total success rate of approximately 95.1% at the five digit level, 95.6% at the four digit level and 97.7% at the two digit level,.

Table 2 shows how the 1996 Small Business Survey performed. In 91% of the cases the interviewers succeeded in finding an ISAT-95 code, of which 96.4% were considered acceptable after checking for errors. The acceptance rate was somewhat higher if the requirement was only to code to the two digit level, or 97.9%. These results are in line with the results of the pilot survey.

Table 2. Hit rate and acceptance rate using trigram search while interviewing. SBS 1996.

	ISAT-95 5 digits	ISAT-95 4 digits	ISAT-95 2 digits
Hit rate	91	91	91
Acceptance rate	96.4	96.6	97.9
Success rate	87.7	87.9	89.1
Number of cases	8,059	8,059	8,059

In the November 1996 Labour Force Survey the hit rate was similar to the hit rate in the 1996 SBS (Table 3). The acceptance rate was, however, considerably lower, or 88.7% at the four digit level and 92.7% at the two digit level. These results indicate that the total success rate for interactive and residual manual coding amounts to 93.1% at the two digit ISAT-95 code, as compared to 97.7% in the SBPS. At the four digit level the total

success rate in the LFS can be estimated as 89.1%, as compared with 95.6% in the SBPS.

The difference between the experienced and inexperienced interviewers was not statistically significant, in fact the success rate was almost equal (Table 3). The results seem, however, to indicate that the inexperienced interviewers were more inclined to resort to guessing if they could not find an appropriate code immediately.

Table 3. Hit rate and acceptance rate using trigram search while interviewing. LFS November 1996.

	ISAT-95 4 digits		ISAT-95 2 digits	
	Total		Total	Inexperienced interviewers Experienced interviewers
Hit rate	92.5	92.5	93.8	90.5
Acceptance rate	88.7	92.7	91.6	94.3
Success rate	82.0	85.7	85.9	85.3
Number of cases	610	610	357	253

The fact that there was no difference in the success rates between experienced interviewers and inexperienced, indicates strongly that one has to look elsewhere for the determinants of the success rate. Table 4 shows that the interactive coding performed very differently depending on the section of economic activity of the edited ISAT-95 code. In both the LFS and SBS these variations were statistically significant.

Table 4. Success rate of trigram-coding by economic activity, 4-digit level. SBS 1996, LFS November 1996.

<i>Percentages</i>	SBS 1996	LFS Nov. 1996
Agriculture	95.3	94.4
Fishing	96.3	100.0
Manufacturing	76.8	76.8
Electricity and water supply	—	...
Construction	89.9	92.5
Wholesale and retail trade, repairs	85.5	75.3
Hotels, restaurants	88.2	92.9
Transport, storage and communication	95.1	77.8
Financial intermediation	90.5	84.6
Real estate, renting and business activities	85.8	73.9
Public administration	...	71.4
Education	61.9	86.2
Health and social work	92.9	89.6
Other services	88.8	73.6
Total	87.9	82.0
Number of cases	8,050	610
χ^2	324.10	28.61
df	12	13
p	.000	.007

4. The cost of trigram-coding vs manual coding

The length of the interview process was measured in the November LFS by using the inherent time measurement in the CATI environment, whereas the length of the manual coding and post-editing process was measured simply by noting the time of opening and closing the relevant forms in MS Access. This proved to be a crude measurement. In both cases a better procedure would have been to measure the time it took to process individual cases.

An experienced coder carried out the manual coding, while the review and editing were handled by an inexperienced coder who only had knowledge of the basic concepts and structure of the ISAT-95 classification but no prior practical experience. This coder was given insufficient instructions so he did not always close the forms while taking lunch hours or working on other things. For the purposes of this paper the idle time that could be verified was eliminated.

Table 5 shows the main results of the length of interview measurements. The control group (manual coding) has a shorter average length of interview than the interactive group. This difference is statistically significant at the .085 level. When the interactive interviews are broken down by the success of the trigram-coding, the “no-hits” stand out as considerably prolonging the interview. The difference between the “no-hit” average length of interview and other “interactive” interviews is significant at the .005 level. Even though the success rate is independent of the experience of the interviewers, Table 5 shows clearly that the inexperienced interviewers were far more likely than the experienced ones to spend time in a futile search in the trigram-coding dictionary.

Table 5. Average interview time in seconds by method and success of coding economic activity. LFS November 1996.

Averages	Length of interview in seconds			
	N of Cases	Total	Inexperienced interviewers	Experienced interviewers
Manual coding	617	443	485	375
Interactive coding	610	469	513	407
<i>Not coded</i>	46	601	777	439
<i>Not acceptable¹</i>	64	474	501	417
<i>Acceptable¹</i>	500	456	495	402
Total	1,227	456	498.6	392

¹ 4-digit level

We would, of course, expect that the average length of interviews would increase when adding an interview item to the questionnaire. On average the number of interview items, i.e., the number of recorded questions, trigram-codes and remarks, was 1.2 items more for the interactive group than the control group. As shown in Table 6 the effects of interactive coding disappear when controlling for the number of interview items.

Table 6. ANOVA Interview length by Manual/Interactive coding with Interview items

		Unique Method				
		Sum of Squares	df	Mean Square	F	Sig.
Covariates	Interview Items	11.314.590	1	11.314.590	193,7	0,00
Main Effects	Manual/Interactive coding	43.271	1	43.271	0,7	0,39
Model		11.515.136	2	11.515.136	98,6	0,00
Residual		71.505.736	1224	71.505.736		
Total		83.020.872	1226	83.020.872		

Table 7 shows the results of the measurements of time consumed while coding manually and while reviewing and editing both the interactive and manual codings. In addition, the estimated time of coding in BLAISE is taken into account, using the difference in average interview time between interactive coding and manual coding in Table 5.

**Table 7. Time used in manual and interactive coding of economic activity.
LFS November 1996.**

<i>4-digit level</i>	Manual Coding		Interactive Coding		Total hours
	N of Cases	Hours	N of Cases	Hours	
Coding	663	6.13	610	4.33 ¹	10.46
Checking	663	2.60	565	2.22	4.82
Editing ²	34	1.36	73	2.92	4.28
Total		10.09		9.47	19.56

¹ Estimated time.

² In a number of cases the editing resulted in unchanged codes.

Table 8 shows the estimated time of interactive and residual coding for all 1,227 cases, using the hit rate estimates in Table 3 and the time measurements in Table 7. The table also shows the estimated time if the cases were exclusively manually coded. The table shows that one can expect that the time of actual coding will decrease by 21% while the time of editing will increase by the 39%. The total coding time using interactive coding as a primary method of coding is 8% less than if manual coding were used exclusively.

Table 8. Estimated time of coding economic activity using trigram and residual manual coding.

<i>4-digit level</i>	Manual Coding		Interactive Coding		Total hours	Excl. manual hours
	N of Cases	Hours	N of Cases	Hours		
Coding	92	0.85	1135	8.06	8.91	11.34
Checking	92	0.36	1135	4.46	4.82	4.82
Editing	5	0.19	83	3.31	3.50	2.52
Total		1.40		15.83	17.23	18.68

Source: Tables 3 and 7

We represent in this paper the cost estimates only in terms of time consumed. Analysis of the pay structure at Statistics Iceland revealed that the average cost per hour of interviewer work is approximately the same as the average cost per hour of a professional with a university degree. This is because the interviewers worked predominantly night and weekend hours, and the fact that university education in Iceland is relatively poorly remunerated (Institute of Economic Studies 1997).

5. Discussion

In this study we have established that using interactive or trigram-coding as a primary means of assigning economic activity codes would produce an total success rate ranging from 89.1 to 97.7%, depending on the classification level and type of survey. This is in line with other studies on the accuracy of this type of coding (ILO 1990). In the post-editing process the total success rate can be increased substantially.

The results of the SBPS indicated that the interviewers were less successful than the manual coder in assigning ISAT-95 codes to the economic activity. The failures of the manual coder to assign a code was primarily due to imperfect data recorded by the interviewers.

In the SBS the trigram-coding performed as expected from the results of the pilot study. This was not the case for the LFS, where the success rate of the interactive coding was by far lower than expected. There were at least four significant differences between the two surveys to account for this.

First, in the SBS the respondents were asked a specific confirmation question, which was not the case in the LFS.

Second, in the SBS the time factor was of no importance, as each interview only covered a handful of questions. Many of the interviewers preferred to have the ISAT-95 manual available in order to be able to code the economic activities with greater accuracy.

Third, the respondents from the two surveys did not come from the same population. The SBS population is by far better informed of the functions of their businesses than the LFS population, which for the most part consists of employees with often only a sketchy knowledge of the functions of the firms with which they are employed.

Fourth, the two populations differed in the fact that the SBS population was more homogeneous than the LFS population. The economic activities of the small business owners tended to concentrate in certain sectors of the economy, and they were more likely than the firms in the LFS to cover only one possible ISAT-95 code.

The interesting result of this study is that there is no difference between experienced and inexperienced interviewers with regard to the success rate of trigram-coding. There was a slightly greater tendency for the inexperienced interviewers to have a higher hit rate but this was offset by more unacceptable codes, indicating that they may resort more to guessing when they do not find an appropriate code. These differences are, however, not statistically significant. This does not mean that training interviewers will have little if no effect. The interviewers in the LFS only differed in terms of the amount of hands-on experience. Training them specifically in terms of understanding the concepts and structure of the classification system itself may prove to be advantageous.

Analysis of the success of trigram-coding by section of economic activity indicates that the list of descriptions is the main factor in determining whether this type of coding returns acceptable codes or not. The importance of the coding file is also emphasised by other authors.

The results show that trigram-coding of economic activity prolongs the interview, mainly because it is an added item in the questionnaire. There is also reason to believe that this type of coding takes somewhat longer to record than the average interview item. Our crude measurement did not, however, enable us to verify this. As borne out by the study of Roessingh and Bethlehem (1993), failures to assign a code result in considerably prolonging the interview. In this there was a marked difference between the experienced and inexperienced interviewers, clearly showing that better training may lead to better results.

In this study the interactive coding performed only marginally better than manual coding in terms of costs, as measured by time. Different pay structures may change the odds more distinctly in favour of trigram-coding. We did not take into account in this study the costs of developing the trigram module, nor the cost of training the interviewers. It is, however, not clear whether the training cost should be included. If there is a reason to train the interviewers to record better descriptive data for trigram-coding,

there is certainly a reason to do so in order to improve the quality of the data for manual coding. Furthermore, neither the investment in developing a computer-assisted manual coding facility in MS Access nor the investment in training the expert coder were taken into account.

6. Conclusions

The results of this study show that as a means of coding economic activity during the interview, the trigram-coding facility in BLAISE III is a viable instrument. The total success rate, prior to editing, is at acceptable levels and the cost of trigram-coding is less than the cost of exclusively manual coding. The differences in costs are, however, not great. The study indicates that the performance can be considerably improved, both in terms of the success rate and in terms of costs.

First, the list of coding descriptions can be refined in order to increase the likelihood of hits, especially within certain sections of economic activity.

Second, training of interviewers is important. The training should both consider interview techniques, such as knowing when there is time to quit searching in the trigram dictionary, and the concepts and structure of the classification system. The latter, however, was not tested in this study.

Finally, interpretation of these results as valid for all circumstances of trigram-coding should be made with caution. Classification systems can be vastly different as the underlying structures may be different in how they lend themselves to codification. These differences can also be across language and cultural boundaries. Interactive coding of a concept such as economic activity may thus not be feasible in all languages.

References

Bushnell, D. (1995). Computer-assisted Occupation Coding. In Essays on Blaise 1995, Proceedings of the Third International Blaise Users' Conference, V. Kuusela (ed.), pp. 23-33, Helsinki, Statistics Finland.

ILO (1990) *Surveys of economically active population, employment, unemployment and underemployment*. An ILO manual on concepts and methods. Geneva.

Institute of Economic Studies (1997). Menntun, mannaður og framleiðni [Education, human capital and productivity] (in Icelandic). Hagfræðistofnun Háskóla Íslands, Report no. C97:02, Reykjavik.

Lyberg, L. and Dean, P. (1992). Automated Coding of Survey Responses: An International Review. R & D report 1992:2, Green series, Statistics Sweden.

Martin, B. (1993). The Blaise Coding Facilities - a closer look. Paper presented at the Second International Blaise Users Conference, London, October 1993.

Roessingh, M. and Bethlehem, J. (1993). Trigram-coding in the Family Expenditure Survey of the CBS. In Essays on Blaise 1993, Proceedings of the Second International Blaise Users Conference, pp. 118-132, London, Office of Population Censuses and Surveys.

Experiences of Two Capi-Papi -- Comparisons in Finland

Markku Heiskanen, Kirsti Ahlqvist, Statistics Finland

One of the cornerstones of quantitative survey interviewing is the standardized measuring situation. Survey questions are often considered as certain kinds of fixed stimuli to similarly rouse the expected responses (concerning facts, opinions, values) in every population group studied. The survey dogma assumes that the measurement situation can (and must) be firmly controlled. On the other hand, there are many studies indicating that even the slightest alteration of the details may cause dramatic effects in the results. Features like clothing of the interviewer, his/her behaviour and attitudes, voice emphasis, small changes in question wording, interview modes, sampling and so on provide a jungle of sources to spoil the results. Could the portable be one of them ?

The influence of technology and engineering has become a more decisive factor in the interview process with the CAI-system. Routing errors disappear, all relevant questions are asked, coding systems are incorporated, no basic data entry is needed afterwards. These means naturally improve the control of measurement, but they have not been major problems in our surveys so far. The main problem is whether the respondent gives "the right" answer to the questions, or are there obstacles in the interviewing situation that prevent the respondent from revealing him/herself.

When our interviewers first began computer assisted interviewing in 1993, the question of the contents of the Capi data was widely discussed. Two larger Capi/Papi comparisons have been made up to the present. The first one was a comparison of our Survey on Living Conditions (SLC) in 1993, the second one the Household Expenditure Survey (HES) in 1995. Both are large surveys with a face to face interview. Some of the main results of these mode comparisons are reported in this article.

1. The Survey on Living Conditions

The pilot Survey on Living Conditions was carried out in the autumn of 1993 to compare Capi and Papi results. For all interviewers it was their first contact with the respondent using Capi- method and for most interviewers with computer assisted interviewing in general (a couple of Cati studies had been compiled at a small Cati-center). This may account

for the fact that the average duration of the Papi interviews was 65 minutes and that of Capi interviews 75 minutes.

A random sample of Finnish population consisting of 1 000 persons aged 25-54 years was drawn from the Finnish Central Population Register. One half of the sample was interviewed by Papi-method, the other half by Capi-method. Since 140 interviewers participated in the mode comparison, they only had a few interviews of both types. Thus the interviewer effect on personal level was kept small. The type of the starting interview was randomised to each interviewer. The rate of nonresponse was 24% in both samples, 3 percentage units lower than in the actual survey conducted in spring 1994. Most of the background variables, such as gender, age and place of residence, did not show any significant differences in participation rates. People with a higher level of education were, however, slightly overrepresented in the Papi-dataset, a fact that was not corrected with the weighting of the results by gender, age and region. Even so weights are used in the comparison.

The main topics of the SLC were household/family structure, living conditions in childhood, residence and housing conditions, social activity and leisure activities, contacts and relations to relatives and friends, education and training, work and working relations, health and security, and evaluation of own life.

A cursory comparison of most questions was made using chi-square and t-tests. Most variables were categorical. Of the 329 questions tested (the complicated question sets of household structure and sickness/disease lists were omitted from the overall comparison), there was a statistically significant difference ($p < 0.05$) in 31 questions.

Table 1. Differences between Capi and Papi results in the Survey on Living Conditions, 1993.

Topic		

Total		
Background		
Everyday life		
Activity in organisations		
Social support		
Work		
Health		
"Values of life"		

Hence nearly one out of ten questions produced different results. This was quite an surprising result; more particularly since due to small sample sizes the tests do not give significant differences except in cases where the results are also primarily interpreted as significant in contents and meaning; in extreme cases not even a difference of 8 percentage units (a dichotomised variable) did exceed the significance level. In addition, some variable classes were combined because of zero observations. In relative terms, most differences between Capi and Papi results were found in questions under the topics Everyday life and Activity in organisations (20% of the results in each topic differed).

Yet larger differences have been reported in mode comparisons. E.g. in the study of Baker et al. the overall per cent of Capi-Papi differences was

13.5 (Baker et al. 1995, 419; they compared 445 questions in the US National Longitudinal Survey of Labor Market Experience).

It is not possible to say which method yielded more reliable results. E.g. the Labour Force Survey of Statistics Finland indicates that in 1993 85% of the people with earned income were wage earners. The corresponding number was 82% in Capi results and 88% in Papi. With most variables there exists no comparison possibilities. Nevertheless, we can look at the trends in the differences: does one or the other method produce more or less certain activities or positive stands.

The topic Everyday life includes questions of different leisure time activities. The Papi results showed in four cases more activity than the Capi results. In one variable (sports activities) there were more active participants in Capi results. The results of the topic Activity in organisations were in line with the leisure time questions above: the Papi results reported more activity than the Capi results.

Unofficial social aid and support (from relatives, friends, neighbours, fellow workers) is received and given more often according to the Capi results. The number of friends was also greater in Capi. But in four out of 12 cases social aid was received or given more often in Papi results. In work-related questions fear connected to working conditions was more common in Papi in three cases, in Capi in two cases.

It is quite common to carry out mode comparisons by comparing differences in the results on total level with certain key variables. The need for details in comparison depends on the purpose the results are used for. Total counts are often an adequate accuracy in statistics production. But in social research the behaviour of different population groups often needs examination. Do female respondents differ from males? Or different age groups, or different social strata ?

We chose two topics of the SLC for a more detailed analysis. The questions concerning social support and health are also used by many other Finnish researchers, not merely by those of Statistics Finland. These questions may to some degree be sensitive or delicate to the respondent. E.g. in social relations, revealing that one does not have friends, can rouse unpleasant feelings in the respondent. On the whole, neither of the topics is extremely delicate (like drug use or sexual behaviour) but represent the middle range sensitivity in social surveys.

Although the health responses on the main distribution level had only one significant mode difference in the 57 questions tested, the detailed analysis of background variables showed that education, which as a variable reflects the social status of the respondent, was a key determinator causing differences in the mode comparisons that were not found in the total distributions (Ahola 1996). The common formula of mode differences was as follows: people with the highest level of education report more health deficiencies in Capi than in Papi results, whereas respondents with a low level of education reported more health deficiencies in Papi interviews. The mode differences were also larger in the low education group. One explanation to this might be that the change in the interview situation had raised the status of the interviewer compared

with that of the interviewee. Or that the interaction situation had become more official and formal, and thus preventing the information mediation process. The differences were smaller in health symptoms when socially desirable questions were compared (e.g. fatigue, over-exertion) than in questions which were socially less desirable (e. g. questions concerning depression).

The result that population groups with a lower level of education give a more positive report of their health in Capi seems somewhat contrary to Baker et al.'s statement that Capi yields greater respondent willingness to disclose sensitive information (Baker et al. 1995, 413).

Certain response structures remained the same in both modes. Both in the Capi and Papi results women reported more health symptoms than men. The structure of commonness of different symptoms was also maintained by men and women.

An analysis by gender, age group and education was also made with certain key variables in the topic Social support. The conclusion drawn of that comparison was that statistically significant, to certain direction consistent, mode differences could not be found, although older respondents seemed to give somewhat more positive picture of their social relations in the Capi results than in the Papi responses (Heiskanen 1995).

As a result of these mode comparisons, the question was raised that the international results, where in general no significant differences between Capi and Papi modes were found (e.g. de Leeuw 1992, Martin et. al., 1993), might possibly not hold in all cultures. Finnish society has been characterised by a small social distance between different social stratas, thus small changes in the interaction situation might influence their responding stand, particularly among people with the lowest level of education.

2. The Household Expenditure Survey

When the second mode comparison study was carried out the interviewers of Statistics Finland had worked with computer assisted interviews for about a year. The HES had formerly been conducted every five years with the Papi method, but from 1994 onwards it was carried out on an annual basis and with a smaller sample size. The one-year sample was divided into two subsamples of equal size in 1995; one half was interviewed by Capi, the other half by Papi.

The interviewers are trained in the subject matter more carefully than in many other surveys of Statistics Finland. The survey consists of a face to face interview and a two weeks' diary of household expenditures. The datasets were collected during the whole year 1995. Sample sizes were 1721 households in Capi and 1729 in Papi. Response rates were nearly the same (68% in Capi, 67% in Papi).

The length of the interview was 72 minutes in Capi and 70 in Papi. Of the 7 interviews that lasted over three hours only one was made by Papi.

Here the two research questions were: 1) Are there significant differences between the Capi and Papi results and 2) Does the social status of the household (education of the head of the household) affect the responses in a similar way as in the SLC? For the question 1) we chose preliminarily 13 commodities or groups of commodities (or payments) for the mode comparison. Three commodities from the diary datasets were also included in the comparison.

The 13 chosen expenditure items cover 50 per cent of the consumer expenditures obtained from the interview, and 20 per cent of all consumer expenditures. When questions concerning acquisition of durable commodities, such as furniture, car, household or entertainment equipment, pc-computer or sports and hiking equipment, were asked response cards were used. The interviewee could see the commodity names in these cards.

In most of the questions the answering proceeded in two phases. In the first phase the interviewee was asked whether the household had acquired certain commodities. If this question was answered with yes-alternative, then the price of the commodity (or payment) was inquired. The comparisons are presented in a similar way: Table 2 shows first the percentage of households who had had the questioned consumption activity during the reference period, then yearly mean values of expenditures calculated on those who had the activity, and standard errors of the means. The results were calculated considering the two stage sampling frame.

The results in Tables 2 and 3 have been weighted to correspond with the Finnish household structure. Thus they differ slightly from the nonweighted ones, e.g. the proportion of one person households increases. Using weighted results may be somewhat problematic in mode comparisons, as mode differences can change when weights are used. If we look at the results from a statistical standpoint, then weighting gives a more accurate description of the population studied, and is also the "final truth" published in reports.

Table 2. Differences between Capi and Papi results in the Household Expenditure Survey, 1995¹¹.

	Capi	Papi
Goods (variable code, reference time in months)		
Source: interviews		
Charges for owner-occupied dwelling (300,1)		
Rent of dwelling (301,1)		
Electricity (31000,12)		
Furniture (4000-,12)		
Household		

¹¹We would like to thank Pertti Kangassalo for his help with Sudaan programming.

appliances and
equipment (420,12)

Radio, television,
tape recorder etc.
(700,12)

Personal
computers and
peripherals
(70600,12)

Sports equipment,
hiking gear (7170,
12)

Purchase of car
(600,12)

Physicians' fees
(5100, 3)

Interest payments
on household loans
(302,12)

Amount of loan for
dwelling (asuntvel,
interview)

Amount of
consumption loan
(kulutvel, interview)

Source: diaries

Meat (101, ½)

Alcohol beverages
(121,½)

Outer
garments/outdoor
clothes (200,½)

The interview results show no statistically significant differences between the modes in the main distributions (% of sample). Neither do the means reveal surprising results. Some smaller differences in the results however exist. In six Papi responses the prevalence rate was slightly higher (the widest difference was 3 percentage units), in three questions the Capi mode yielded more commodity purchases. The same holds for the consumption means; in nine cases they were somewhat higher in the Papi results. These results are in line with the results of the British consumer expenditure mode comparison (Manners 1995). Differences of same magnitude also exist in the diary comparison, with the exception of the only statistically significant difference in Outer garments/outdoor clothes.

The second research question was whether the different social strata react differently to the mode change. The respondents were classified into three groups according to the education of the head person of the household. This variable is supposed to describe roughly the habitus of the family. The classification has been made according to the length of education of the head of family (low=education less than 10 years, middle=10-12 years, high=13 years and more). Age of the head person is expected to describe the phase of life of the household.

Table 3, in which the consumption means of the mode comparison variables are presented as an index ratio by education (Capi/Papi*100), shows one significant difference ($p < 0,05$) both among the interview variables and the diary variables. On the total consumption level, differences are in practice non-existent. By looking at the trends in mode differences - including those that did not exceed the significance level - we find however that in the interview variables by both lower and middle education the means are slightly higher in Capi in 5 variables, and as to Papi in 8 variables. In families with a higher level of education, higher consumption was measured in 9 Capi variables, and in 3 Papi variables. Mode differences in the highest education group are quite small except in physicians' fees ($p = 0,11$), but show nevertheless a slight difference compared to the lower education groups.

Families, where the head person is aged 60 years or over, seem to have somewhat higher total consumption in Papi than in Capi ($p = 0,09$). In loan and sports equipment variables, considerable differences were also found.

This may also indicate an incoherent consumption structure within the group (a couple of outliers, but only on the total level, were omitted from the analysis).

There existed also some statistically significant differences in the age group 40-49 years. In age groups over 30 years the interview variables in the Papi results give somewhat more often higher consumption means, while in the group under 30 years the consumption is higher in the Capi results.

The consumption level and structure differ when education increases. The attitude to consumption probably differs as well. In Finland a sparing consumption style has been traditionally appreciated. At the end of the 1980s a consuming lifestyle and living involving debts entered the traditional consuming patterns, and mostly in younger and higher educated families. The recession in the 1990s ended the "cheerful consumption celebration" and consumers were seen guilty of the recession. This might stand at the background of prevailing consuming attitudes, so that the people having consumed more in the 1990s, might be a sensitive group regarding the mode comparison (Ahlqvist 1996).

3. Discussion

Dufour et al write in their report of the conversion of the Canadian Labour Force Survey to CAI: "The first lesson learned was that any major change in a complicated process requires time to stabilize". In this respect the timing of the Finnish SLC mode comparison as the first Capi project of Statistics Finland was not the best possible. The longer interviewing average in the Capi mode already indicates in this direction.

The interviewers had a very positive attitude to the CAI from the beginning (90% were for, 9% don't know; an inquiry immediately after the SLC). Some drawbacks were nevertheless experienced. 4% of the interviewers reported that the interviewee was shy of the computer, 16% had found that the computer made the interview situation rigid. 21% of the interviewers evaluated that the computer hampered the interaction, and 25% that the computer program restricted the conversation. These results show that the interviewers (and the researchers/programmers) were not yet sufficiently acquainted with the CAI-system. In the actual SLC compiled in spring 1994 the interviewers did not complain about interaction troubles any longer, but on the contrary felt that the computer facilitated the interview situation (a discussion with the interviewers of the Helsinki region, Manderbacka et al. 1994).

The mode differences in the SLC were quite numerous. There might have been more of them in case the samples had had a wider age distribution. If the assumption of different reactions to computer in different educational positions would hold good, it might have become stronger if the young and the old ones had been contrasted.

It might be argued that differences caused by inexperience will balance with time. Some comparisons with the mode study results and the main

SLC 1994 results were made. In some questions in some population groups the results in the main SLC in 1994 did get closer to the Papi results. But in some questions they did not, which reminds that each survey measures situations that are inconstant and subject to variation (here seasonal change in interviewing).

The Houshold Expenditure Survey, where the questions are more fact-like, seems to produce fairly comparable results with different modes. The same structure as in the SLC seems to be at the background though: in the group of people with a high level of education the consumption is slightly higher in many variables by Capi, while people having a lower level of education report higher consumption in Papi.

Table 3. Differences between Capi and Papi results in the Household Expenditure Survey by education of the head of the family, 1995.

Goods (variable code, reference time in months)	Education / Mode index: Capi/Papi * 100
---	--

Source: interviews

Charges for owner-occupied dwelling (300,1)

Rent of dwelling (301,1)

Electricity (31000,12)

Furniture (4000-, 12)

Household appliances and equipment (420,12)

Radio, television, tape recorder etc. (700,12)

Personel computers and peripherals (70600,12)

Sports equipment, hiking gear (7170, 12)

Purchase of car (600,12)

Physicians' fees (5100, 3)

Interest payments on household loans (302,12)

Amount of loan for dwelling (asuntvel, interview)

Amount of
consumption loan
(kulutvel, interview)

Source: diaries

Meat (101, ½)

Alcohol beverages
(121, ½)

Outer
garments/outdoor
clothes (200, ½)

Total consumption

* after the number p
<= 0,05

Goods

Lo
w
edu
cati
on

Hig
h
edu
cati
on

Source: interviews

Charges for owner-
occupied dwelling
(300,1)

Rent of dwelling
(301,1)

Electricity
(31000,12)

Furniture (4000-,12)

Household
appliances and
equipment (420,12)

Radio, television,
tape recorder etc.
(700,12)

Personel computers
and peripherals
(70600,12)

Sports equipment,
hiking gear (7170,
12)

Purchase of car
(600,12)

Physicians' fees
(5100, 3)

Interest payments
on household loans
(302,12)

Amount of loan for
dwelling (asuntvel,
interview)

Amount of
consumption loan
(kulutvel, interview)

Source: diaries

Meat (101, ½)

Alcohol beverages
(121,½)

Outer
garments/outdoor

clothes (200,½)

Table 4. Differences between Capi and Papi results in the Household Expenditure Survey by age of the head of the family, 1995.

Age / Mode index: Capi/Papi * 100

Goods (variable code,
reference time in
months)

Source: interviews

Charges for owner-
occupied dwelling
(300,1)

Rent of dwelling (301,1)

Electricity (31000,12)

Furniture (4000-,12)

Household appliances
and equipment (420,12)

Radio, television, tape
recorder etc. (700,12)

Personel computers
and peripherals
(70600,12)

Sports equipment,
hiking gear (7170, 12)

Purchase of car
(600,12)

Physicians' fees (5100,

3)

Interest payments on
household loans
(302,12)

Amount of loan for
dwelling (asuntvel,
interview)

Amount of
consumption loan
(kulutvel, interview)

Source: diaries

Meat (101, ½)

Alcohol beverages
(121,½)

Outer garments/outdoor
clothes (200,½)

Total consumption

* in front of the number:
n < 20 ; - n < 10

*, ** or *** after the number: T-Test significance $p \leq$
0,05, 0,01 or 0,001

References

Ahlqvist Kirsti, 1996, Kulutustutkimus 1995. Menetelmävertailun (capi-papi) esitutkimuksen raportointia (manuscript, in Finnish).

Ahola Anja, 1996, Responses to health questions with a new interview mode. In "The Role of Statisticians in Today's Information Society". The 20th Conference of Nordic Statisticians. Statistical Reports of the Nordic Countries 66. Copenhagen, 255-270.

Baker Reginald P, Bradburn Norman M and Johnson Robert A, 1995, Computer-assisted Personal Interviewing: An Experimental Evaluation of Data Quality and Cost. Journal of Official Statistics. Vol 11, No. 4, 413-431.

Dufour Johane, Simard Michelle and Mayda Frank, 1996, First year of computer-assisted interviewing for the Canadian labour force survey: an update. In Laaksonen Seppo (ed.): International perspectives on nonresponse. Proceedings of the sixth international workshop on household survey nonresponse. Research reports 219. Statistics Finland: Helsinki, 53-71.

Heiskanen Markku, 1995, Tietokoneavusteisella ja perinteisellä käyntihaastattelulla saatujen tulosten vertailu (Comparing Capi and Papi Results). In "Elinolotutkimus 1994. Aineiston keruu." Tilastokeskus Muistio 1995:2, 58-68 (in Finnish).

Laaksonen Seppo (ed.), 1996, International perspectives on nonresponse. Proceedings of the sixth international workshop on household survey nonresponse. Research reports 219. Statistics Finland: Helsinki.

de Leeuw E D, 1992, Data Quality in Mail, Telephone and Face to Face Surveys. TT-Publications: Amsterdam.

Manderbacka K, Huuhka M, Lahelma E and Rahkonen K.
1994, Haastattelijoita haastattelemassa, Hyvinvointikatsaus
4/1994, 52-54 (in Finnish).

Martin Jean, O'Muircheartaigh Colm and Curtice John,
1993, The Use of CAPI for Attitude Surveys; An
Experimental Comparison With Traditional Methods. Journal
of Official Statistics, Vol 9, No.3, 641-661.

An Office Management System in Blaise III

*Meryl Henden, Fred Wensing, Ken Smith, Mano Georgopoulos
Australian Bureau of Statistics*

1. Introduction

Collection of data using computer assisted interviewing (CAI) does not stop with the field work. After the data is returned to the office it needs to be downloaded, unpacked, registered, checked and coded before it can be handed over for further processing and output production. The arrival of Maniplus in the tool set for Blaise III has made it possible to develop a comprehensive system to manage all these processes.

This paper describes the Office Management System (OMS) developed in Blaise for the first production application of computer assisted interviewing at the Australian Bureau of Statistics (ABS) and discusses some of the problems that were encountered.

2. Household survey operations in the ABS

The ABS operates a substantial household survey program comprising :

- *Monthly Labour Force Survey* - 30,000 households involving an eight month rotating sample using any-responsible-adult methodology, face-to-face interview in the first month and telephone interview from interviewer's home in subsequent months,
- *Supplementary surveys* - generally conducted as an extension to the Monthly Labour Force Survey, covering a range of short topics,
- *Special supplementary surveys* - 15,000 households, carried out once or twice in a year using personal interview methodology, face-to-face, and covering a range of topics in depth,
- *Quarterly omnibus surveys* - 3,000 households covering a range of user-pays items and generally involving any-responsible-adult methodology,
- *Other user-pays surveys* - varying size, content and methodology depending on demand.

The first two components of the survey program are generally known as the *Monthly Population Survey*.

Most of the surveys in the program are developed and coordinated by central management (located in Canberra) with the field phase being conducted by eight regional offices around Australia. These regional offices are responsible for managing the interviewer work force and carrying out the input processing of survey data (receipt, coding, data entry and editing). Subsequent processing is usually managed by central processing and subject areas.

3. Use of BLAISE in the ABS

Blaise was first used at the ABS in 1994 in a pilot test to assess the viability of computer assisted interviewing in the context of population surveys. That test made use of an early Beta version of Blaise III and concluded that computer assisted interviewing was indeed viable and could deliver quality and timeliness improvements with possibly some cost savings. A technical assessment of the Blaise III software concluded that it was a suitable product, although there were some concerns about integration with other ABS computer facilities given that it is a DOS product.

Blaise software has subsequently been endorsed by ABS management as appropriate for the collection of survey data where that collection involves the use of computers in the field. Given that many other facilities within the ABS make use of Windows and the client-server architecture, the use of Blaise has been restricted to field collection and limited office processing (sufficient to clean up the data and deliver it to other systems).

The test program which was put in place for CAI also included a major test of the *Monthly Population Survey* conducted over a six month period in 1996 which made use of 80 interviewers and a rotating sample of 4,000 households per month. This major test provided significant experience which contributed to the development of systems.

The first production application of CAI at the ABS has been the second wave of the longitudinal Survey of Employment and Unemployment Patterns, conducted in September 1996 with a sample size of 8,500 and interviewer work force of around 200, for which the systems described below provided essential facilities.

4. Office Management System Requirements

A new office management system developed to handle CAI was expected to provide, at the very least, the same degree of management and flexibility which is currently experienced with the well established non-CAI systems at the ABS.

Two key features of the old systems which the new developments would have to maintain were :

- ability to manage the work of interviewers from a network of regional offices (ie. the existing organisational arrangements should continue),

- the concept of a batch of work for each interviewer (known as a *workload* and well understood by interviewers).

In addition, the office management system would need to provide the following functionality :

- assign selected addresses to workloads,
- assign workloads to interviewers,
- prepare respondent records for each workload from data collected previously,
- collate workload data for transmission to each interviewer,
- collate associated instrument software for transmission to each interviewer,
- manage transmission to and retrieval of data from interviewers,
- provide for reallocation of workloads or individual respondents from one interviewer to another,
- enable the examination of received data to clean up 'dirty' records or to resolve queries raised by interviewers,
- provide for the coding of some data fields in the office (eg. occupation, industry),
- provide different levels of functionality for central and regional staff,
- monitor survey progress and extract management information,
- export a clean data file for further processing.

5. Description of the Office Management System

5.1 Overview

The core of the office management system developed for CAI surveys at the ABS is a series of Blaise facilities which handle three main office activities associated with surveys :

- preparation of workloads for transmission,
- receipt of data from interviewers,
- office processing (clearing of records).

The system also includes Oracle based facilities where selected addresses are allocated to workloads and assigned to interviewers.

The office management facilities are complemented by a system on the interviewers' notebooks (also written in Blaise) which manages the receipt of workloads from the office and the working needs of the interviewer.

Figure 1 shows the Blaise components of the Office Management System and their relationships to the Oracle and notebook processes.

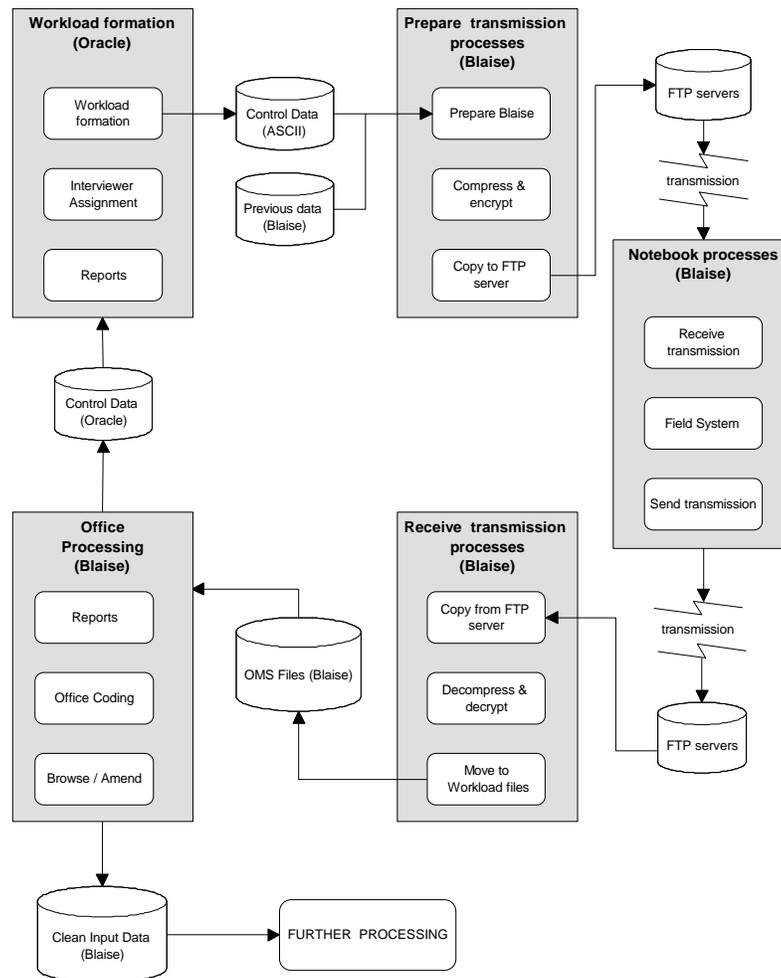


Figure 1. Office Management System showing Blaise components

5.2 Workload formation and interviewer assignment (Oracle)

The underlying feature of this component of the OMS is a series of relational tables stored in Oracle which contain the addresses and other respondent details for the survey sample. For the longitudinal Survey of Employment and Unemployment Patterns these other details included contact information to assist with tracking the respondents over a three year period.

The main reason for developing this component in Oracle (accessed via SQL Windows) is that the functions to be carried out are highly interactive and lend themselves well to that environment and relational tables are a good medium for storage of this kind of data. In addition, Oracle/SQL Windows is the standard client-server environment used at the ABS for most other applications and using this environment gives it a similar look and feel to other applications and position it well to be integrated with other ABS facilities (eg. general sample management facilities) in the future.

Having this component in a client-server environment while other components are in Blaise/DOS does, however, introduce a platform/environment problem. This is discussed further in section 6.

The workload formation facilities take care of the following processes :

- assign selected addresses to workload numbers,
- assign interviewers to workloads,
- transfer records from one workload to another,
- export workloads to ASCII to be prepared in Blaise for transmission to the field,
- update address details for households (as required),
- produce collection control reports including workload and response status summaries.

Once the workloads for a survey have been formed and assigned to interviewers, the Oracle processes produce a series of ASCII files that are used to prime Blaise instruments and prepare transmission files for the interviewer. This is described in the next section.

5.3 Prepare transmission processes (Blaise)

Preparation for transmission involves the following steps :

- prepare Blaise records for each selected addresses in an interviewer's workload using the ASCII data received from Oracle,
- load the Blaise records with any additional data obtained in previous cycles of the survey,
- collate any software required to update the field systems,
- compress and encrypt all the files into a single file (for each survey) to be sent to each interviewer,
- move the transmission files from the LAN to the corresponding FTP server for interviewers to collect,
- check the FTP server to see whether transmission files have been collected.

The transmission preparation processes have been developed using a series of Blaise datamodels, Manipula programs and DOS utilities which are activated from a Maniplus interface. A diagram of the Maniplus menu system to control the transmission preparation process is shown in Figure 2.

Included in this subsystem (and the other Blaise-based subsystems) are collection control files which are maintained as Blaise datamodels. Where possible the system programs have been developed as generic processes which make use of parameters and standard naming conventions to identify each survey and its associated files. To assist with the location of data, metadata and program files, standard directory structures have been set up on both the LAN and the FTP servers.

While the processes may appear simple, some important features were found to be necessary :

- the need know whether software updates were to be included (eg. for a new survey) as additional installation procedures would be needed,
- the need to be aware of whether a transmission was to be an update of a previously transmitted workload (eg. to add a new respondent) in which case a Manipula update procedure would be needed once the transmission was received at the notebook rather than a DOS copy procedure,
- support for one-off transmissions which may be required to fix problems on notebooks in the field (in some cases this may be required for all notebooks in the field),
- the need to keep a record of all preparation events in case problems occur and re-preparation is required,
- the ability to handle multiple transmissions to the same interviewer on the same or a different survey,
- the ability to operate the facilities in both batch and interactive mode.

Some of these features are discussed further in Section 6.

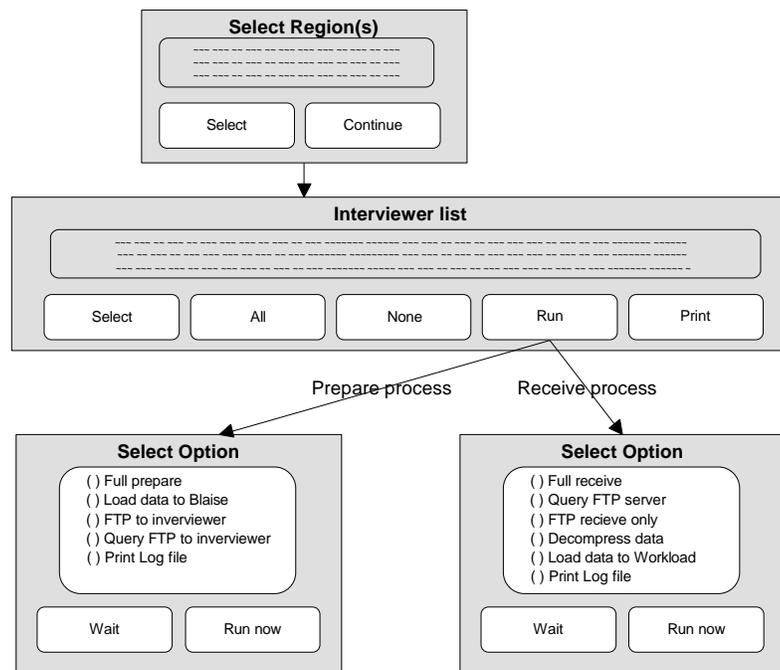


Figure 2. Menus for Preparation and Receive processes

5.4 Receive transmission processes (Blaise)

Receipt of transmissions involves the following steps :

- search the FTP servers for any transmitted data,
- transfer the transmitted data from the FTP servers to the LAN,

- decompress and decrypt the transmitted data for each interviewer for each survey and load the Blaise data into separate workload files.

The receive transmission processes have also been developed using a series of Blaise datamodels, Manipula programs and DOS utilities which are activated from a variation of the same Maniplus interface used for preparation of transmissions (see Figure 2).

Important features that were found to be necessary for receipt of transmissions include :

- the ability to check whether there were any transmissions from an interviewer,
- the ability to handle multiple transmissions from the same interviewer for the same survey,
- the ability to handle transmissions for more than one survey at a time,
- the ability to operate the facilities in both batch and interactive mode.

Some of these features are discussed further in Section 6.

5.5 Transmission (FTP)

Transmission to and from the interviewers is managed through a series of regional FTP (file transfer protocol) servers connected to the LAN and protected with a "firewall" which certifies the user (both internal and external) and controls access to the facilities on the server (see Figure 4). Dial-in access to the server is only possible through encrypting modems which ensure that all communications are authorised and protected. Access is further controlled through a technique known as "strong authentication" which requires the interviewer to carry a "smart card" that generates a single-use password with a limited validity time that can be validated by the server.

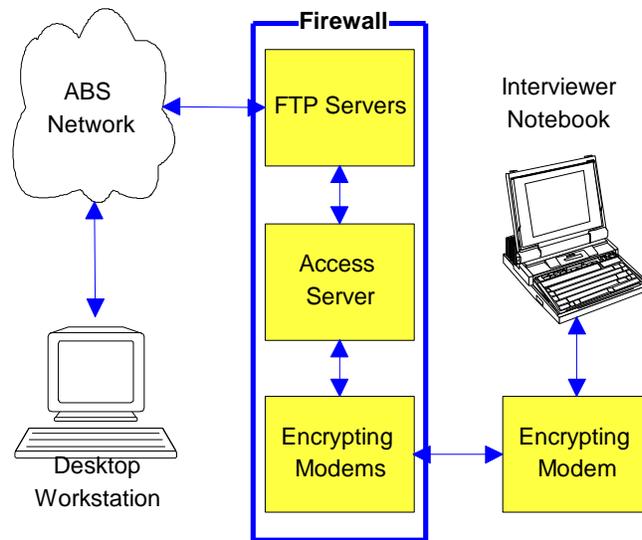


Figure 3. ABS transmission facilities

Each interviewer is generally registered to one of the regional servers and can only access data within his/her own directory on the server. Transmissions are interviewer initiated and make use of a 'pick up' and 'drop off' methodology, in which all external transactions with the FTP server are program controlled from the notebook. Email access is not available to interviewers although this may be considered later.

5.6 Notebook transmission processes (Blaise/DOS)

As mentioned above, transmissions are controlled through programs on the Notebook which handle the following processes :

- prepare response data on the notebook for transmission,
- send response data to the FTP server,
- receive transmission files from the FTP server,
- decompress and decrypt the transmission files,
- execute installation batch files.

The notebook transmission processes have been developed as a series of batch and executable files (containing DOS or FTP commands and Manipula programs) which are called from a Manipulus interface. Interviewers can select whether to send or receive, or both. Facilities are also available through the Manipulus interface to reconfigure the transmission parameters (eg dial-in telephone number) or to switch transmission to floppy disk.

The notebook transmission facilities are completely generic and will act on any or all surveys that are on the notebook. Control of which data files are to be transmitted is managed by including with every survey a preparation batch file (with the same standard name) that contains DOS commands and other instructions to be carried out (to encrypt and compress the data

into a single transmission file). The prepare process simply searches all directories on the notebook for preparation batch files and executes them before transmission. In this way the sending and receiving of data is left in control of the programmers developing the survey since they can include relevant commands in the corresponding batch file that is sent out with the survey instrument.

A significant feature of the transmission processes that have been developed at the ABS is that transmissions to the office will generally contain all records from the interviewer's workload (for a survey) whether interviews are completed or not. This was a deliberate decision, and provides a higher degree of management since transmission files will contain all records for a workload at all times. Transmission of whole workloads also provides a convenient means of data backup (within the notebook as well as from the notebook to the FTP server). There is of course a marginal additional cost associated with longer transmission times.

To handle multiple submissions, a simple naming convention is used for the transmission files. After the respondent data is compressed and encrypted, a single transmission file is produced for each survey and given a name that identifies the survey, followed by a numerical suffix. The suffix number is incremented for successive transmissions using a C program utility. A copy of each transmitted file is kept on the notebook (for backup purposes) and is deleted after the survey is well and truly completed. Recovery of old transmitted files can be made once the notebook is returned to the office. In the office, the latest transmission received for any one interviewer is the file with the highest numbered suffix. Earlier transmissions are kept in the office system and can also be processed if problems arise with the latest transmission file.

Once transmission has been carried out, individual records in the workload which were previously identified as completed by the interviewer (through a "response status" field) are marked as "transmitted" by a Manipula program which is included in the transmission batch file. All records in a workload which are marked as "transmitted" are no longer accessible to be changed by the interviewer, although they can be listed on the screen (for information).

5.7 Office processes (Blaise)

Once the respondent data is successfully transferred from the transmission facilities the first phases of office processing are concerned with :

- following up queries raised by interviewers,
- coding which could not be done in the field,
- clearing up any outstanding data problems identified by the Blaise instrument (eg remaining edit failures or route problems),
- reporting the processing status of workloads for the survey,
- changing the status of individual records,
- combining released workloads into a file for the next stage(s) of processing.

The office processing facilities have been developed using a Maniplus interface, a series of Manipula programs and Blaise datamodels (control files) as well as the main survey datamodel. A diagram of part of the Maniplus menu system to control the processes is shown in Figure 4.

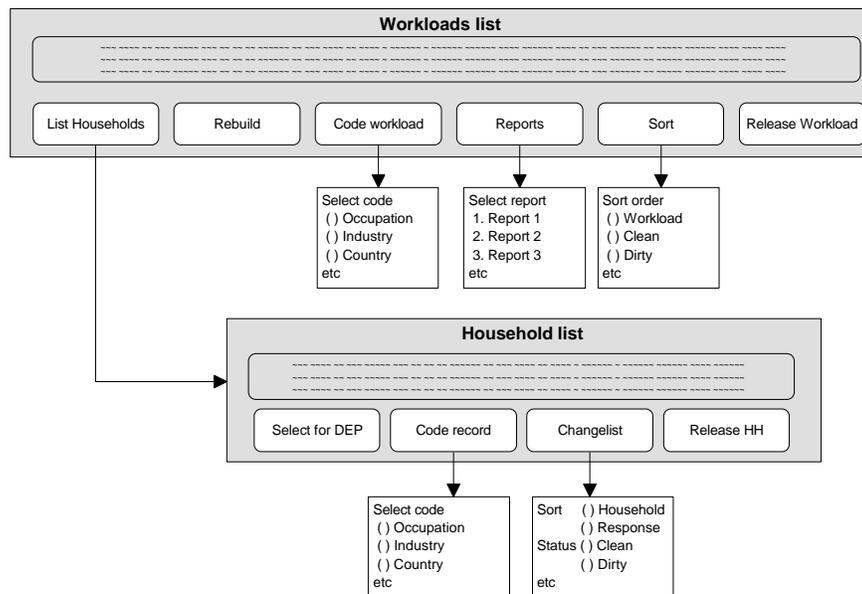


Figure 4. Menus for office processes

A key element of the office processes is knowing the completion status of a survey record. A completed record is one that has been cleared by the interviewer, has had coding carried out (if applicable), is considered clean by Blaise and has also been released by the office staff. Status information is stored either within the survey record or in workload summary control files. The initial status of a record is determined by the system but release of that record requires manual intervention through the Maniplus office interface, and can be done at the record or workload level. Rules are also applied within the system to ensure that there are no inconsistencies in the status elements.

The field system developed for CAI at the ABS provides a facility for interviewers to annotate any respondent record with comments to be followed up in the office. These comments, known as queries, are made available to office staff through the OMS who can enter the respondent record using the Blaise Data Entry Program (DEP) to clear up the identified problem.

Although coding can be carried out in the field using Blaise coding facilities there are various reasons why some coding may be done in the office: the complexity of some classifications (eg occupation) may make it impractical to carry out coding in the field; special systems may already exist in the office to provide for computer assisted coding; or it may be desirable to use existing office coding procedures to maintain comparability with previous surveys. All these reasons were applicable at the time that the OMS was being developed. In addition, coding facilities were not fully functional in the version of Blaise that was available at the time. The OMS, as it has been developed, supports office coding by displaying

relevant record details and allowing a code to be typed directly into the record. It was not possible to link the system to corporate computer assisted coding facilities at the ABS because they make use of SQL Windows. Such links may be possible once Blaise is moved to the Windows platform.

6. Some Technical Issues and Solutions

6.1 Reallocation of respondents from one interviewer to another

One of the more difficult issues to handle has been the reallocation of respondents to other interviewers or for further enumeration by someone in the office. This issue was particularly prevalent in the longitudinal Survey of Employment and Unemployment Patterns where many of the respondents have moved address in the time between the waves of the survey. Some of the situations which have arisen are :

- respondent moves to another interviewer's area,
- respondent moves to a location which is under the control of a different regional office,
- refusal or non response requiring the office or another interviewer to follow up,
- reallocation of a partly completed workload to another interviewer.

Early versions of the OMS dealt with these situations only partially. Although respondents were successfully reallocated, it would result in duplicate respondent records (one from each interviewer), because the record would then be in two workloads. Deletion of the record from the original workload was not practical because the original interviewer may have transmitted a partially completed workload to the office before the reallocation took place. The record would then need to be deleted from all copies of the original workload (both in the office and on the interviewer notebook) as well.

Where duplicate records exist between interviewers (or regions) it is not possible to simply accept the latest data received because the original interviewer may transmit again and after the record was re-allocated to another interviewer. Instead, it was considered sufficient to accept the record which was more complete based on a "response status" code applied by the interviewer. This approach was not fully effective because some response status codes do not indicate which one is more complete.

To overcome this duplicate data problem, the latest version of the OMS contains a Blaise based *control file*. Each time a respondent record is moved to or from any of the processes (ie. Prepare Processes, Notebook Processes, Receive Processes), the *control file* is first checked to determine whether that process has control of the record. If it does then the process is activated and a record is written or modified on the *control file* to indicate the change in control (See Figure 5).

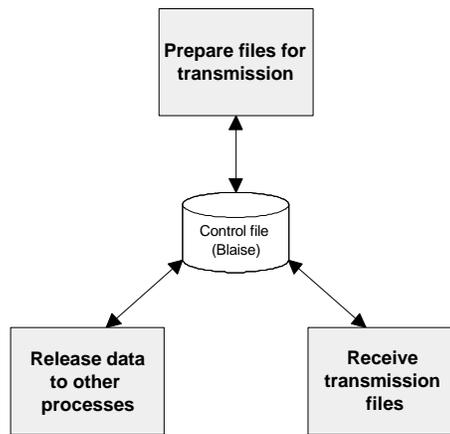


Figure 5. Controlling the movement of records

With this design, a record for any respondent may be located in more than one place but only one is considered to be the active record at any one time. This form of control was only needed in the processes which handle transmissions to and from the field because that is the place where multiple copies can arise. It was not necessary to have a similar arrangement for the pre or post-field processes since there would be only one copy at that time.

An example of the way in which the control file is used (for reallocation) is as follows :

- A record is prepared for a respondent and transmitted to an interviewer in Region 1. The Prepare Transmission process adds a record to the Control file to indicate that interviewer now has control for that record.
- The interviewer discovers when he/she attempts to make contact with the respondent, that he has moved to Region 2. The interviewer marks the response status on the record for that respondent accordingly and it subsequently is transmitted back to the office as part of the interviewer workload.
- The Receive Transmission system processes the transmission and loads the record to a Workload Blaise File. The Control file is updated to reflect that the Blaise OMS now has control of that record.
- An office person reviews the record transmitted from the interviewer and releases the record to be sent back to Oracle. The Blaise OMS Export process moves the record to Oracle and the Control File is updated to reflect that Oracle now has control (actually done by removing the control record for that respondent).
- Someone in the office for Region 2 uses an Oracle process to reallocate the record to an interviewer in their region. The Prepare Transmission process then prepares a transmission for the new interviewer. A new record is created in the Control file to reflect the new interviewer in region 2 now has control.
- Any further transmissions of that respondent record from the first interviewer will be ignored because the control file will indicate that it is no longer with the first interviewer.

While this control process has the potential to increase the time taken to move a record/workload from one interviewer to another, the benefit of knowing exactly where a record is outweighs this disadvantage.

6.2 Running batch jobs in Blaise

The Blaise processes for preparing and receiving transmission were found to be time consuming when run in interactive mode. There were also performance and access problems when other users were accessing the same data files. Arranging for these processes to be run overnight also had problems because :

- being interactive LAN processes, the computer needed to be left logged into the network overnight, causing a potential security problem,
- often these processes needed to start at a much later time (eg after midnight) when all users are out of the system or after interviewers have transmitted their work to the office.

To overcome these problems a WAIT function has been developed in our Maniplus interface for the processes concerned. The function calls a WAIT DOS utility (developed at the ABS) which "locks" the computer until a specified time. After the specified time is reached the computer is freed to run the selected process. At the end of the selected process, WAIT is invoked again to "lock" the computer until the same time the next day. Weekend processes can be activated to start on Sunday by allowing the entry of start date into the interface as well as a start time. The WAIT utility is then invoked in a loop until there is a match with the specified date.

The WAIT utility requires the user to enter a password at the time of submission. If at any time the WAIT process is interrupted the same password is required to allow processing to continue. The only way to interrupt the WAIT process (or the delayed processing) without the password is to reboot the computer (after which normal LAN security prevents unauthorised access).

A problem also arose with delayed processes that produce screen messages which require the user to press the <enter> key to continue but an option to turn off screen messages is now available in the latest releases of Blaise.

6.3 Performance and data volume issues

With the Blaise Office Management System running on the ABS network we have encountered a number of performance problems/issues. In the early version of the system, transmissions that were received from the field were loaded into a single Blaise data file for each region and placed on a Central office server to make them more accessible to survey management staff. With the longitudinal Survey of Employment and Unemployment Patterns this created data files of around 40 Mb each and a national file of approximately 112 Mb.

The problems we encountered with data files of this size were that extraction or updating of workloads during office processing was taking far too long (anywhere from 3 to 20 minutes); large files seem to be more susceptible to "corruption" caused by users aborting processes or from network dropout; some processes required exclusive access to the file (preventing others from using it); and rebuilding the files when they became "corrupt" could take hours to perform.

Two important improvements have now been incorporated in the OMS to deal with these :

- locating the processing files on a regional server;
- maintain a large number of small files during office processing.

Moving the data files to a local server made a significant difference to performance in the regional office because it reduced the delays caused by network traffic. While this improved regional performance there was a consequential reduction in the performance of Central office processes (such as prepare workloads and receive data) but these were not interactive and could be run overnight using the WAIT facility described above.

The most significant improvement to the OMS design has been to maintain a large number of smaller data files during the office processes, rather than a single regional file. By using a separate local file for each workload, it is no longer necessary to "extract" a workload from the main file for local processing, corruption of a workload file no longer affects other workloads and rebuilding such a file can be carried out quickly and does not interfere with other processes. In addition, the prepare and receive processes could now be run during the day without interrupting other processes. Keeping a larger number of smaller files does, however, necessitate a very organised and maintained directory structure on the network and is assisted by the existence of a workload summary file (such as that mentioned in section 5.7) to enable workloads to be located and report on the status of each.

6.4 Management of processes

An important requirement of an office management system is to be able to manage the separate processes, particularly where there is a work force of interviewers that operate remotely from the office. Each interviewer and the each of the processes which supply and receive work to them require separate control procedures. At any one time it must be possible to see which interviewers are expecting work, who has received work, who has returned work and whether the processes which carry out the various send and receive tasks have succeeded or failed.

The OMS contains a control screen in both the prepare and receive processes which provide this functionality and makes them easier to manage (see Figure 6).

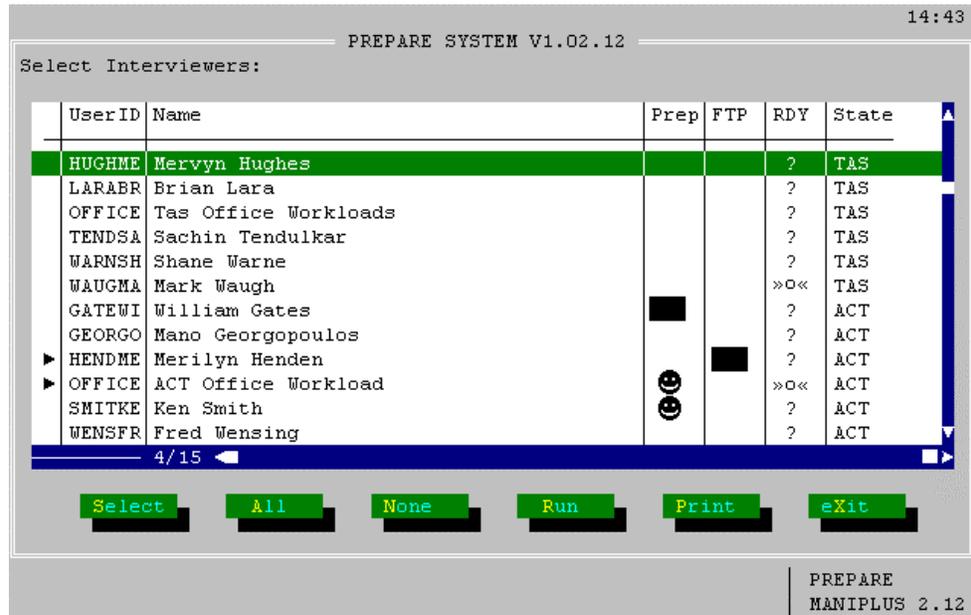


Figure 6. Typical control screen for prepare processes

The control screen presents a list of interviewers and a number of columns which report the status of each process. To make the status easier to interpret, the status report is presented using graphic symbols rather than words or numbers to indicate whether a process is ready to run, has run successfully or failed to run successfully during the current session.

The control screen includes indicators (arrows on the left) that identify when an interviewer is "selected" for a process to be applied, and buttons for selection of "all" or "none". Once a selection of one or more interviewers is made, the required process(es) can be selected using the "run" button which presents a list of processes that can be activated to run immediately or later (using the WAIT facility described in 6.2). The "print" button produces a printed report of the screen details.

The main feature is that all the management information and action options are presented in a single screen. A similar screen been developed for the receive processes with a different set of status columns and underlying processes.

6.5 Clean and dirty records

Another important requirement of an office management system is the ability to determine the "clean" status of individual records. While Blaise provides a useful system-generated status indicator on each record (with codes for Clean, Suspect, Dirty and Not checked), this status can change with even a small adjustment to a record or the questionnaire instrument.

In developing the OMS it became important to understand the way in which the record status indicator is adjusted by Blaise. In that way it can be used to assist (and not hinder) the clearing of records received from the field.

Some important lessons which were learned from the behaviour of the status indicator are :

- as far as possible ensure that the field instrument contains all the office management fields because moving the data between even slightly different datamodels in the office will cause all suppressed edits to become un-suppressed,
- make use of the "check rules" option while running Manipula processes on the data file otherwise all changed records will become "Not checked",
- a "dirty" record may indicate data has been added to fields that are not on the path, so it is important to carefully check the behaviour of all processes.

Since the development of the office management system described in this paper an additional option has been added to Blaise to enable the software to "force" the rules in a Manipula process and remove data that is not on a valid path through the instrument. This feature should assist office processes in the future.

7. Conclusion

The system described in this paper highlights some of the logistic, technical and data handling issues that have to be overcome in office management of computer assisted interviewing. Maniplus has provided the means to develop a series of integrated functional procedures to deal with these issues, which can handle the response data while it is a Blaise format, and present the office staff with an easy-to-use interface to carry out their functions.

8. Acknowledgements

We would like to acknowledge the contribution of other staff from the Household Surveys Facilities development team and the Population Survey Operations Section at the ABS who have assisted in the design, development and testing of the systems described in this paper. Special reference must be made to Julie-Anne McDonald and Sharon Baker who were responsible for developing the early versions.

Integrating Surveys with Blaise III

Fred Heuvelmans, Frans Kerssemakers, Jeroen Winkels, Statistics Netherlands

1. Introduction

As in other European countries Statistics Netherlands has designed several surveys to gather information on (aspects) of living conditions. These surveys not only vary in content, but also in the sample frame used, sample size, response, interview length, number of persons to be interviewed, use of proxy respondents, type of data editing, weighing method, etc. Some of these surveys were revised in the beginning of the nineties when Statistics Netherlands introduced computer assisted interviewing (CAPI). These revisions focused both on the harmonization of questions used for demographic and socio-economic classifications and on the translation into CAPI of questionnaires that were originally developed for being asked with paper and pencil (PAPI).

Several developments brought Statistics Netherlands to go a step beyond harmonization of survey variables. These developments especially have to do with the demand for statistics on living conditions, initiatives to improve response in social surveys and further development of information technology. In the next section an overview is given of the most important reasons that lay the new design of a system of social surveys. The name of the system, POLS, refers both to the Dutch abbreviation for a Continuous set of Surveys on the Quality of Life and the Dutch word for pulse ('to keep a finger on the pulse'). The POLS-design is presented in section 3.

Sections 4 and 5 focus on the integration from a Blaise perspective and the new management system respectively. Section 6 goes into more details about the Blaise III questionnaires and includes some remarks on the system of post-processing the collected data. It is followed by a section on the problem of documenting Blaise questionnaires. The last section discusses some practical experiences of using Blaise III as a tool to integrate social surveys.

2. The integration of social surveys

At this moment Statistics Netherlands has started the redesign of its data collection on persons and households. This redesign is, because of the enormous complexity and impact, structured along some stages. In the

first stage the socio-cultural surveys are redesigned. From 1997 onwards the eight surveys in the fields of health, crime and security, political and social attitudes, the general social survey (with a special survey on the youth and the elderly), time budget and housing are being combined. However, the long term developments that steer the redesign of these surveys will also influence the redesign of other surveys inside and (as we see it) outside Statistics Netherlands in the near future: new demands, the fight against non-response and new technology to link data.

New demands

Rather than being concerned with the distribution of aggregate variables over groups of statistical units, social research frequently focuses on relations between variables at the micro-level. This focus on micro-relations has been the driving force behind the development of ever more comprehensive household surveys. These surveys aim to cover as many variables as possible, so that all social relations can be analysed for a comprehensive data set. However, there is a growing awareness of the limits to this approach. The response burden on the sample households becomes too heavy if too many variables are covered in a single survey. This burden forms a serious constraint on statistical agencies to meet all the demands. The type of statistical information needed by policy makers also seems to change. The demand is no longer directed to rather one-dimensional statistics about the well known themes of the political agenda (health, crime, income). The demand for information directed to the more complex (causal) relations between these themes has increased. This demand reflects the policy questions arising from societal developments such as poverty, social exclusion and deprivation. These societal problems often converge in problem areas, like inner cities. Especially these developments need to be monitored by statisticians to fulfil the politicians needs. To state it in 'variable language': not the univariate or bivariate distributions of variables but the multivariate relations between sets of variables is being asked for.

The fight against non-response

Non-response in household surveys is a severe problem. Compared with other countries The Netherlands are not doing very well (De Heer, 1996). Within the domain of the socio-cultural surveys response varies between 50% and 60%. This level of response probably causes serious bias that can not always be neutralized by smart methods of weighing. With respect to the four-yearly National Election Study, for example, there is a small debate in the Netherlands what the figures from this survey really tell us about political alienation when so many people do not participate. Statistics Netherlands has taken several initiatives to reduce non-response. A lot of these initiatives have to do with the way surveys are presented to the public (e.g. introduction letters), the monitoring of interviewers and the optimization of using different data collection methods. These initiatives are always taken under the restriction that the overall respondent burden should be as low as possible.

New technology to link data

Other reasons for changing the social surveys of Statistics Netherlands stem from information technology developments. As described in Van Bochhove and Everaers (1996) there are basically three methods to link micro data from different sources. The first is *exact matching*. In this case

micro data from different sources on *the same individual* are linked. The second way of linkage is *synthetic matching*. In this case micro data from different sources on *the same group* of individuals (for example age group) are linked. A third method of linking micro data is to redesign the sources in such a way that there will be a brief *joint questionnaire* for a large sample that provides succinct information on core variables from all original surveys; and more in depth questionnaires on the separate areas for smaller subsamples. This way a core micro data set is obtained directly (the equivalent of exact matching) and an in depth synthetic data set can be created by combining the in depth data from the various subsamples using the joint core as a synthetic matching key. Because the variables in the core are associated with the in depth variables of the subsamples, this approach picks up relations more easily than synthetically matched data sets that rely on just demographic variables for the matching. POLS has been based on this third method.

Restrictions on the integration of surveys in POLS

The most important restrictions that steered the design of integrating the separate social surveys within the socio-cultural domain were:

1) *Minimization of the burden on respondents. This condition can be viewed from a micro and from a macro perspective. 'Micro' means that the mean interview-time within a household should not exceed 45 minutes. 'Macro' means that the total interview-time of all the modules should be reduced by POLS. This restriction has also lead to efforts to combine separate surveys from other agencies with specific POLS-modules.*

2) *A person-based sample frame. Statistics Netherlands now has the possibility to take samples from a file that is based on the fully automated population register in the Netherlands. This register contains information about nearly all the persons and households in the Netherlands. In the first place the fieldwork department can profit from this information to combat non-response: interviewers know beforehand who is going to be interviewed and some characteristics of the persons (sex, age, nationality) who refuse are known. But data-users can also profit from this information, because oversampling is rather easy, because some information on the non-responding persons is given and because this register will (in the future) be combined with other registers.*

3) *Enough cases to allow description of relatively small subgroups. This condition could only be met if all the socio-cultural surveys should join. However, the estimated net response of 36 000 households (1997) forces Statistics Netherlands to make work of the extension with other surveys to find the smallest ones of the target groups for policy makers.*

4) *No proxy interviewing (for persons who are older than eleven years), unless the empirical relation with other characteristics is well known. This restriction has to do with the efforts to both improve quality of the data (proxy information is in general less reliable) and to decrease the burden on respondents.*

5) *Flexibility in adding modules on certain topics during certain periods or in oversampling specific population groups within a separate survey.*

3. The design of an integrated system of social surveys (POLS)

Shell-structure

The design of POLS is based on a shell-structure. In principle there is no limit to the number of shells, so panel studies can be included. Every shell has its own characteristics (see figure below). A survey always consists of the joint questionnaire (shell 1 and shell 2) and one or more modules from shell 3. However, the order of the questions during the interview can be different, because of the interviewing process. For example, the questions on income that make part of the joint questionnaire are always asked at the end of the interview. The joint questionnaire is designed under the restriction that both telephone and face-to-face interviewing should be possible. The CATI-part will be used to reach certain types of non-respondents from the first fieldwork stage. At the end of this telephone interview the respondents are asked if they are willing to join in a face-to-face interview (to gather information being asked in shell 3).

Shell 1 : (total sample)
Joint questionnaire part 1 : *harmonized classification variables*

This part contains all the questions to be asked on every person in the sample. The questions use the harmonized classification variables, both the demographic ones (age, sex, marital status, nationality, place of birth) and the socio-economic ones (education, socio-economic position, household income). For the future it is planned to collect this information as much as possible via registrations. Apart from the future use of register information, this part of the questionnaire will only change if the concepts and definitions of the harmonized questions are changed.

Shell 2 : (total sample)
Joint questionnaire part 2 : *core questions in the socio-cultural domain*

This shell contains the core questions of the socio-cultural surveys. The reason to name it shell 2 (the interviewer off course does not see these terms) is that it is foreseen that in a later stage core variables from other surveys will be added. This shell is therefore less stable in content than shell 1. Because the questions in this shell are used for the total sample, they allow the quarterly publication of important indicators and the annual publication on a low regional level.

Joint questionnaire part 3 : *screening questions in the socio-cultural domain*

This part varies from year to year. Although the large sample allows finding a lot of special groups, including a lot of screening questions in the joint questionnaire is not possible. In 1997 the screening (of 36,000 persons) is for accidents and injuries. The information gathered will be used to ask a specific subsample to join in a follow-up study (shell 4, see below).

Joint questionnaire part 4 : *Variable questions*

To meet the fifth restriction (see above) POLS has implemented the principle that a small part of the joint questionnaire should be reserved for actual themes or themes that require a large sample. This part can vary twice a year. In 1997 it is used for questions on victimization.

Shell 3 : Socio-cultural modules (subsamples)

In this shell theme-specific questions are planned. However themes are combined in a way that more-dimensional indicators (as described above) can be calculated. This calculation will partly be based on the idea of consistent weighing (see Renssen and Nieuwenbroek, forthcoming). The contents of this shell change regularly. Data on some topics have to be collected once or twice a year during a certain period. Some other topics only need to be analysed once every three or four years. Within this shell also screening questions (on a subsample) are included to trace persons for a follow-up study. The following modules, for which non-overlapping samples are used, will be implemented in 1997. The figures between brackets refer to response and age categories.

Health (N = 10,000, age 0+)

This module uses two modes of data collection: CAPI and PAPI. The paper and pencil mode is primarily chosen because of the sensitive questions on health, for example the 'burnt out' syndrome. The module is continuously being asked (January-December).

Justice and Environment (N = 7,500, age 12+)

Here again CAPI and PAPI are both used. The PAPI is chosen because of the questions about time use (a so called 'yesterday interview'). The module is continuously being asked.

Justice and Participation (N = 5,000, age 12+)

This module only uses CAPI. Part of the questions are identical to the last module in order to fulfil the requirement of having 12,500 responses about themes like victimization, crime prevention etc. The module is continuously being asked.

Youngsters (N = 4,000, age 12-30)

This module also only uses CAPI, but in a slightly different way. Because of the interviewed population (under 30 years) it is possible to ask the respondents to complete parts of the questionnaire on a notebook computer by themselves. The answers to these questions (about sex, drugs and crime) will be of a better quality if self-completion is chosen. The module is scheduled three times in a decade and will be asked from March 1997 till December 1997.

Trends (N = 4,500, age 18+)

This module only uses CAPI and primarily consists of trend questions that have been asked in surveys since 1974, and questions on request of the Social and Cultural Planning Office. The module is scheduled three times in a decade and will be asked from March 1997 till December 1997.

Accidents (N = 5,000, age 0+)

The survey on accidents is part of a new research project of Statistics Netherlands that has been formulated on request of the Ministry of Health. The other part of the data-need of this project has been formulated within the joint questionnaire (screening questions) and a follow-up study in shell 4. This module uses CAPI and PAPI. The PAPI-part is the same as de-

scribed above: a yesterday interview to measure time use. Here it is only to be used on Friday however, in order to supply additional cases for getting a uniform distribution of net responses over the days of the week.

Shell 4 : Follow-up studies (subsamples)

The possibilities both in the joint questionnaire and in the specific modules to screen target groups are used for follow-ups. This shell can also incorporate a panel survey, such as the National Election Study. In 1997 two follow-up studies are foreseen, both based on screening :

- A health examination survey (HES) in a subsample of municipalities,
- CATI-interviews on accidents and injuries, of about 15 minutes.

As the mean interview-time for the joint questionnaire is fixed at 15 minutes, shell 3 may generally take another 30 minutes.

In 1997 a reduction of the total (macro) interview-time of 20% (ie 5,000 interview hours) will be achieved thanks to the introduction of POLS.

An overview of the Integrated System of Surveys (POLS) in the year 1997

(numbers refer to persons sampled from Population Register)

Basic questionnaire	part 1	
N = 36,000	shell 1	CAPI/CATI

- Harmonized questions for demographic and socio-economic classifications (stable)

Basic questionnaire	part 2,3,4	
N = 36,000	shell 2	CAPI/CATI

- Core questions (stable)
- Screening questions (var.)
- Special themes (variable)

Health	Justice		Young	Trend	Accidents
	Environment	Participation			

- Theme-specific modules
 - In depth questions
 - Screening questions
- (modules can change from year to year)

<i>CAPI</i>	<i>CAPI</i>	<i>CAPI</i>	<i>CAPI</i>	<i>CAPI</i>	<i>CAP I</i>
<i>PAPI</i>	<i>PAPI</i>				
10,000	7,500	5,000	4,000	4,500	5,000
<i>shell 3</i>					

<i>Health exami nation</i>	<i>Accidents</i>
	<i>CATI</i>
<i>shell 4</i>	

4. Integrating surveys from a Blaise perspective

To keep things manageable the in principle indefinite number of subject-matter questionnaires should be independent from each other as much as possible. The integration concept rests entirely on the common basic questionnaire. The latter could be duplicated in different datamodels, one for every subject-specific questionnaire. These datamodels could then be put under the umbrella of a single case management system and be handled by Maniplus as a single survey. But especially the basic questionnaire is meant to be relatively stable and independent from the variety of subject-oriented wishes and developments. And if the datamodels are to be recognizable entities linked to a particular subject, changing the basic module should leave them unchanged, if possible. Subject-matter specialists who are responsible for a particular datamodel should not be bothered with things they did not initiate and that may be irrelevant to them. Therefore it was decided to have a separate datamodel for the basic questionnaire on the integrative level next to a set of datamodels for which, as before, independent and non-overlapping samples of persons are drawn.

First, the common basic questionnaire is asked. Included are also themes that require a large sample (eg victimization or accident rates). After finishing this datamodel a Manipula-setup takes care of writing the data that may possibly be needed in anyone of the shell 3-datamodels in a separate 'external' datamodel. Through a dialogue box in Maniplus the interviewer can now open the particular datamodel to which the target person was assigned. After concluding this part of the interview some questions still have to be asked about the household of the target person. Here data are collected from the person with the highest income, such as

income itself, educational and occupation. As this may involve a shift of respondent these questions, which actually belong to the basic questionnaire, are asked near the end of the interview. They are put in a separate datamodel, which is only auxiliary because the data is subsequently added by Manipula (block moving) to the files of the original datamodel. Thus, the latter will finally contain all data from the basic questionnaire. For a particular case only the data from the first model and from the applied shell 3 are sent back by telephone. So, for the actual questionnaires POLS uses a *1-n-1 structure of datamodels*. The two constants represent the datamodels for the generally applied basic questionnaire. The variable n represents the datamodels for the different shells 3 from which one is chosen per interview. Afterwards extra datamodels can be used, as they are, for follow-up studies among screened cases (i.e. shell 4).

The interviewer can always return to preceding datamodels for changing already given answers. However, as this may effect routing, checks and computations in subsequent models, Manipulus forces the interviewer to open these models again so that potential changes will be processed automatically. This could be inconvenient if occurring frequently. In practice it can be coped with by choosing the right, relatively independent modules.

5. A new management system for the interviewer

The interviewer management system on the laptop, which has been used by Statistics Netherlands since 1991 in connection with Blaise, is called LIPS. Written in Pascal, it is a dedicated system for some well-defined repeating tasks such as choosing addresses, making interviewer reports and treating sample elements. Although robust, the system is not easy to adapt or extend. Once the decision was taken to use Blaise III for POLS it was clear that a new and preferably more flexible system was needed.

Inspired by the possibilities of Blaise III and to improve accessibility it was decided to build the new management system entirely with the tools of Blaise III. Especially Manipulus proved to be a useful tool to create an interactive environment for all kinds of action, not only starting a questionnaire but compressing a data file as well. So, the modular design of the POLS-dependent questionnaires, mentioned before, was also used for the new LIPS. Being a management system, it was designed to be as independent from a particular survey as possible. From a Manipulus-setup, called LIPS.MAN, separate Blaise-questionnaires are started to handle addresses, to make an interviewer report, or to pass control to another Manipulus-setup, called POLS.MAN, that for his part steers the survey-specific questionnaires to be used in the actual interview. In a way one could define LIPS now as only taking care of the steering of the respective Blaise-questionnaires and the presentation of the main entities: survey, address and sample element. That is how the core of the system is made resistant to changes. All survey-dependent code is put in separate components which can far more easily be adapted or replaced. Even the interviewer administration can now be geared to the demands of a particular survey or be changed for a certain period of fieldwork within a single survey. General and specific parts have been separated. Adding a new subject can simply be accomplished by writing a new questionnaire

and specifying in the setup the criteria for calling it. Besides, at any point during the interview only the setup that is in control and the particular datamodel the interviewer is using are loaded into the internal memory of the laptop-computer (which was extended from 4 Mb to 12 Mb).

The required Maniplus-setup however is not trivial. Some specialist knowledge is indispensable. Exchange of data between models has to be arranged and a lot of careful checking is needed in order to guarantee a correct and smooth proceeding of the interviewer activities. Nonetheless, the lessened complexity of the modular design clearly outweighs the complexity caused by the introduction of a separate level of control.

6. The Blaise III questionnaires

Although much effort was needed to structure and steer the different data-models, the POLS-questionnaires itself became simpler and easier to handle. Somewhat contrary to expectations there were hardly any problems when the questionnaires of the old surveys had to be converted to Blaise III at the beginning of 1996. For this purpose Blaise III proved to be remarkably stable. Most discussions were about operating the user-interface and about the layout of the screen in particular. Removing the check-paragraph in most cases helped writers to simplify their questionnaires. At least, one could get rid of some persistent problems in the old Blaise pertaining to the interaction, sometimes difficult to grasp, between the routing and the check-paragraph, for instance, when variables in the routing-conditions were imputed in the check-paragraph. Generally, most writers think that specifying questionnaires has become easier. On the other hand, Blaise III offers a lot of new opportunities which have to be mastered first. The use of block parameters, for instance, to optimize a modularly designed questionnaire like the one in POLS takes time. And to fully exploit the new possibilities source codes have to be checked almost line for line. But doing so is in no way a prerequisite for using Blaise III. In POLS old and new live together. Overall, the progress is a result of gradual enhancement. Old parts are often waiting till they will be redesigned by subject-matter specialists. In the meantime some aspects are being improved already, for instance, by introducing new date functions from Blaise III. So, a lot is still to be done to fully use the new possibilities.

Perhaps because of the ample resources of Blaise III, a noticeable shift has already occurred from discussions about Blaise to the more fundamental question of how questionnaires should be organized so that maximum advantage can be taken from the datamodels once they are specified. The goals that are at stake here include subsequent data processing and analysis, maintenance and documentation of questionnaires, and comparability or exchangeability of parts between surveys. These are all paramount issues for a complex integrated design like POLS.

Subsequent data processing has been organized as a multi-stage process. The main stages are *initialization*, *execution* (checking data, construction of composite variables, weighing) and the *output* stage. An important restriction on the process can be formulated as 'non-increasing

complexity'. The processing is organized in such a way that the data can be analysed in all stages. All stages and substages are fully automated. A user-friendly information system has been developed to select variables and micro data (from the several POLS-modules) for internal use.

7. The documentation of the datamodelling

The datamodels of Blaise III can be specified in such a way that they are almost entirely self-documenting for those who are able and willing to read Blaise. At least one should be capable of reading the block structure, not being distracted by technicalities or large pieces with checks and computations. The new date and time functions of Blaise III, by the way, greatly helped POLS to get rid of a lot of such code. Yet, even users with enough other programming experience often lack confidence when they have to use source codes for documentation purposes. We doubt if this has much to do with Blaise. That is to say, probably all computerized questionnaires of some complexity, large sized and with a lot of functionality will raise problems of documentation, not specifically Blaise.

So, one could say that Blaise-questionnaires can and should be made self-documenting for those who are used reading or writing Blaise. Lacking an (semi-)automated solution for those who cannot, POLS has its own hand-made documentation on paper. The problem with this is not so much the making of it as well the maintenance, keeping it up-to-date. Therefore a prototype of an information system has been developed that is going to be used as a tool for reading *all* the relevant POLS-documents, including both questionnaires in an end-user readable format and the 'self-documented' Blaise sources. Probably this will help to keep them well matched.

Meanwhile ideas are evolving within Statistics Netherlands for an automated aid, utilizing the metadata that is read by Cameleon from the data-model. What is needed is a setup that selects metadata and interprets it as language codes with respect to the layout and, next, a program to represent text in a common format so that it can interactively be tailored to the user's own wishes. WordBasic commands could be used for controlling Word. But first the major users have to define the functionalities.

8. Miscellaneous Blaise

Finally, an example will be given of some typical questionnaire items in POLS and the role Blaise III plays in handling them.

- *As a consequence of chosen policy increasingly information is to be used that is already present in administrative registers (eg population, tax or social security). As data do not have to match with the respondent's own perception, imputation in interviews can raise problems. In POLS age, gender and marital status of the sampled person are obtained from the population register. By selecting a certain code the interviewer assigns the external data to the assumed target person and the corresponding variables in the household-roster will be imputed. Then the interviewer checks the imputed answers and may change them. Although a discordant 'age'*

or 'gender' is hardly acceptable, for the time being one error will be accepted. Discrepancies are counted and will be evaluated.

- Each shell 3 should take 30 minutes per interview. The sample size can vary from shell to shell, depending on the subject-modules they contain. Added up, the non-overlapping samples for the different shells should not exceed the fixed total for POLS, which is now 36,000. Given these restrictions it is not easy to find a satisfactory combination of modules in each shell. For instance, some modules cover persons below the age of twelve but most do not. And among the first a module may need two times as many children as the other. In practice, this will result in either a surplus or a shortage of children. Also, certain kinds of events may be underrepresented. In POLS, for instance, previously existing separate modules for different kinds of accidents are combined into one general module. After screening by CAPI, the last accident with injury is selected and asked about more in depth by CATI (shell 4). In the old situation the relative rareness of some kinds of accidents was compensated by screening more sample persons. In the combined module a relatively rare kind of accident has to be overrepresented to gather the same number as before. For these and similar frictions additional sampling mechanisms can show very helpful. A child, if any, is randomly drawn in the first example if the initial target person is one of the parents (who in this case should also respond for the child). And if there are more kinds of accidents a certain kind is randomly drawn first in the second example (with weights 1 and 2 for one or more of a certain kind respectively). In Blaise the resulting random number should be fixed, of course, but the draw should be ignored if the initial conditions change (by correcting an answer). For a set of mutually exclusive combinations (Comb) of conditions to which random assignment applies, our approach looks like :

```

IF (Comb[1]=Yes) OR (Comb[2]=Yes) OR ..... THEN
    { eg Accident1=One and
    Accident2=More }
    IF (Comb[1]=Yes) AND (Fix[1]=EMPTY) THEN
        NrComb[1]:= RANDOM[N]+1
        Fix[1]:= 1 Fix[2]:= EMPTY .....:= EMPTY
    ELSEIF (Comb[2]=Yes) AND (Fix[2]=EMPTY) THEN
        NrComb[2]:= RANDOM[N]+1
        Fix[2]:= 1 Fix[1]:= EMPTY .....:= EMPTY
    .....
    ENDIF
ELSE
    Fix[1]:= EMPTY Fix[2]:= EMPTY .....:= EMPTY
ENDIF

```

- Statistics Netherlands strongly fosters interactive coding of open answers during the interview, both for reasons of efficiency (reduced coding staff) and quality (contact with the respondent). To achieve this quite a few concepts in POLS need a many-branched, hierarchical tree of questions. Here the facility in Blaise III that leads the interviewer through a hierarchy of successive sets of alternatives has shown extremely helpful. It sometimes replaces complete modules of self-specified questions. When

in addition the nested enumerations in the so-called classification type are put in a type library this helps to get the questionnaire well-organized. The application for the part of the body that is mainly injured reads as follows:

```
{1}      Head          (1) "Head, face" = (
{1.1}    Brain          (1) "Brain eg concussion",
{1.2}    Skull          (2) "Skull",
{1.3}    Ear            (3) "Ear",
.....
{2}      NeckThroat    (2) "Neck, throat" = (
{2.1}    Vertebra      (1) "Backbone",
{2.2}    External      (2) "External",
.....
{3}      Chest          (3) "Chest, upper back" = (
{3.1}    External      (1) "External (eg ribs)",
```

- In CATI only the basic questionnaire (shell 1 and shell 2) is asked. Except for two INCLUDE-files for the introduction and the good-bye respectively, the datamodel used with CATI is the same as the one used with CAPI. Contrary to the old Blaise the arrangements for using CATI can now be made completely external to the datamodel.
- Although certainly not making the most of it yet, we already profited from the new features which Blaise III offers in the LAYOUT section. We generally hate needing more screens for one question. Therefore two standard data entry screens were added for optional use, one for long question texts in the upper part and one for large tables in the lower data entry part, simple but very helpful for the sorts of questions typical of POLS. It is also planned to use the interviewer's computer for the increasing number of self-reports in POLS. This will of course place far greater demands on the layout of the data entry screen.

References

Bochove, C.A van and P.C.J. Everaers, *Micro-macro and micro-micro data linkage in social statistics. Paper for the seminar on 'The Future of Social Statistics'; Mondorf-les-Bains, 1996. Forthcoming in Netherlands Official Statistics, 1997.*

Heer, W. de, *International response trends. Developments and results of an International Survey. Paper for the 7th Workshop on Household Survey Nonresponse, ISTAT Rome, 2-4 Oct 1996.*

Renssen, R. and Nieuwenbroek, N., *Aligning Estimates for Common Variables in Two or More Sample Surveys. Journal of the American Statistical Association, 1997 (forthcoming).*

Some Thoughts About a Metadata Management System

Jean-Pierre Kent and Maarten Schuerhoff - Statistics Netherlands

1. What is Meta-Information ?

In the past quarter century the use of computers has radically changed the data management landscape. Both the volume of collected and analysed data and the speed with which these operations are performed have risen dramatically. New dissemination media, such as CD-ROM and the Internet, have also contributed to making data available to a broader public.

Under the challenge set by the new technical possibilities, the concept of metadata and meta-information has grown in importance in a spectacular way. Congresses and work groups on various aspects of data processing pay attention to it, and it has become the subject of conferences of its own. Software for data management refers to metadata, and there is software specifically designed for metadata management.

It is however increasingly difficult to formulate a definition that covers all the different aspects of the concept. It often happens that two specialists involved in a conversation about meta-information have difficulty in understanding each other because of different backgrounds and different aims. We ourselves happen to be the victims of this phenomenon, both inside Statistics Netherlands and outside. The only sure way to avoid misunderstandings is to start every dialogue about meta-information with a series of agreements about basic concepts.

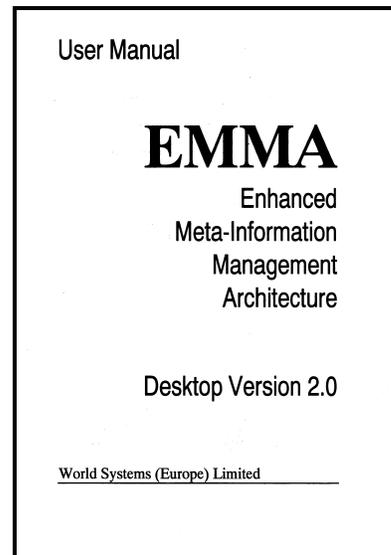
In the first part of this paper we will try to identify the semantic area covered by the words *metadata* and *meta-information* in most contexts. We will not attempt a distinction between the two. In this exploratory phase we will use them as loose equivalents.

1.1 Definition of metadata

The words metadata and meta-information were coined on the model of *meta-philosophy* (the philosophy of philosophy), *meta-language* (a language to talk about language), and the such, in which the prefix *meta-*expresses reflexive application of a concept to itself. This construction is

still productive, as shown by the recent words *meta-network* (a definition of the Internet) and *meta-search engine* (a search engine devised to search search engines). This provides the simplest, but most widely accepted definition of these words : *metadata* are data about data, and *meta-information* is information about information.

Beyond this thin definition, however, it is often difficult to agree upon what



information about information should be captured and managed, and how to use it.

1.2 EMMA vs. Blaise

One of the ways to discover differences in meaning is to confront packages that explicitly state metadata management as one of their functions. This is the case of both EMMA and Blaise.

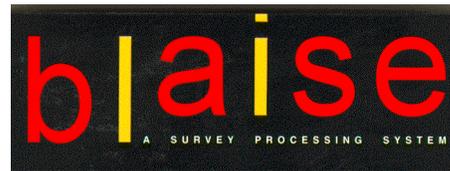
The letters of EMMA stand for **E**nhanced **M**eta-**I**nformation **M**anagement **A**rchitecture. The package is developed in Luxembourg by World Systems Europe Ltd. We will refer here to EMMA Desktop version 2.0.

EMMA provides a data base framework for defining and managing meta-information. It approaches meta-information from a hierarchical point of view and distinguishes three levels :

- *context* meta-information, such as special information, subject matter concepts, language, nomenclature, etc.,

- *resource* meta-information, such as documents, files, authors, etc.,
- *content* meta-information, including surveys, populations, variables and measurement units.

EMMA is a data base management system targeted at meta-information, and supports capturing meta-information, searching and browsing of meta-information, and the production of documentation texts. EMMA is specifically targeted at the management of metadata. It treats metadata in the same way as data are normally treated by a data base management system. Metadata are the data of EMMA.



Blaise is a metadata-driven integrated survey management system produced by Statistics Netherlands. We will refer here to Blaise III, version 1.12.

Defining data models is one of the main tasks one can perform with Blaise. The system offers a hierarchical approach, in which variables can be clustered in blocks, and blocks can be nested. In Blaise, the metadata consist of all the information that can be specified within the system about the variables and their relationships, including the data type of specific variables, their value ranges, classification categories, with a variety of free format texts that can be used for interview questions, variable labels, value labels, documentation texts, etc.

Here are a few characteristic differences between the two packages :

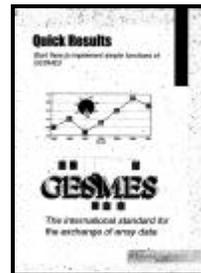
- EMMA focuses on metadata: beyond the fact that metadata refer by definition to data, EMMA has nothing to do with data. In contrast, Blaise focuses on data. The metadata in Blaise are put to work in order to define data, access and manipulate them, and finally transfer them to other packages.
- Blaise metadata are primarily used by software. The system actively uses meta-information during all processes, such as data entry and editing, interviewing, data manipulation and conversion. It can also convert metadata to the format required by other programs, allowing any third-party package to access whatever meta-information is relevant to it. In contrast, EMMA is a user-oriented metadata management system. All the normal user activities on a data base are supported (entering, correcting, searching, browsing, extracting), and the main output of the system is printed text.

- In EMMA metadata are passive bits of information. They can be stored and retrieved. In Blaise, metadata are active. The user enters the metadata, producing a collection of objects that can be put to work for entering, validating, transforming and transferring data.
- EMMA is open to all sorts of information about data. It allows you to define whatever you need as long as it fits in one of the three categories of context, resource and content metadata. In contrast, Blaise supports only those metadata elements that can be put to work by the system.

This comparison shows how different the two packages are in their view of metadata. Their strong points are very different. EMMA is about defining and standardising concepts, co-ordinating data definitions, and managing the meta-information needed by users. Blaise is about automating the survey process, about managing the metadata needed by the software. EMMA is about concepts, and Blaise is about operations. We would like to establish here our first distinction : there is *conceptual meta-information*, and there is *operational meta-information*.

In our opinion this is not a natural contrast. There is no reason why well standardised and co-ordinated concepts should not become operational, or why operational metadata should not be standardised and co-ordinated. The two should merge, and this should be one of the long-term aims of any metadata development project. We will come back to this idea in the second part of this paper.

1.3 Gesmes



We will now extend our comparison to GESMES, a standard that also refers to metadata in its definition. GESMES is defined by the Statistical Office of the European Communities, Eurostat. It is a standard derived from EDIFACT, and it specifies the format in which statistical data and metadata should be transferred electronically from one institution to another. GESMES tells you how to format the information before transferring it.

Here is an example of a data set cast in the GESMES mould :

```

UNA:+.? 'UNB+UNOC:3+STATINSTITUTE+EUROSTAT+950123:1400+
+REF001++GSMES' UNH+001+GSMES:D:95A:UN'NAD+MS+BE001++
++++BE'DTM+137:950511:101'CTA++:Henri de Bakenbourg'
COM+3222567980:TE'DSI+QPROD123'DTM+Z02:1991119923:708'
ARR++BE:101:FR:911:c11+BE:101:FR:912:c12+BE:101:FR:913
:c13+BE:101:FR:914:c14+BE:101:FR:921:c15+BE:101:FR:922
:c16+BE:101:FR:923:c17+BE:101:ES:911:c21+BE:101:ES:912
:c22+BE:101:ES:913:c23+BE:101:ES:914:c24+BE:101:ES:921
:c25+BE:101:ES:922:c26+BE:101:ES:923:c27+BE:201:FR:911
:C31 etc.IDE+5+DFORMAT123'UNT+12+001'UNZ+1+REF001'

```

Before you can use GESMES three conditions have to be met :

- the statistical concepts must be known to both sender and recipient,
- the order of the dimensions and cells must be fixed and known in advance,
- the coding system used for the statistical values must be known to both sender and recipient.

So GESMES is clearly not meant for modelling data or defining concepts. Most of what is called *metadata* in Blaise must already be known before GESMES can be put to use, and GESMES cannot be used to transfer this type of meta-information. However, GESMES allows you to attach a footnote to an individual cell or row, or to the whole data set. Footnotes are free text, so they can convey any human-readable information. This covers most of the domain covered by the word *metadata* as it is used by EMMA.

The GESMES standard itself is all about data, so it also falls within the scope of metadata. It specifies information that cannot be missed in the process of transmitting data: who is the sender, what is the content of the message, and the specific format in which the content should be transferred. This is not about concepts or about data structures, it is about packing data and metadata in such a way as to identify them to the user and to the software. It is about files, storage media and transfer protocols. Let us call this category of metadata logistic metadata.

1.4 Metadata standards

In comparing Blaise and EMMA we established a distinction based on the intended use of metadata : the point was whether the metadata were meant primarily for use by humans or by software. Human users in this context are mainly people involved in designing, managing and executing the operations leading the data through all the phases of collecting, editing, processing, analysing, publishing and disseminating. This picture misses out the most important user of all : the end-user, the person who uses the data in published form. Whether they browse the book, search

their way through the CD-ROM or surf the Net, they must be able to find the data they need, decide whether the data they find are relevant to their questions, and whether the quality of the selected data meets their needs. Finally they must be able to understand the meaning of the data and to use them correctly. We will call the information supporting this type of activities *documentary metadata*. One of the prerequisites for successful communication with the end-user world-wide is international standardisation of metadata. The need for metadata standards has become increasingly acute with the exponential growth of the Internet. The main standard organisations, both national and international, are involved in the definition of metadata standards. For an example of such a standard we will refer to the Federal Geographic Data Committee's "Content Standards for Digital Geo-spatial Metadata", FGDC standard for short. This is a handy example because it is available on the WWW. This makes it easily accessible for our readers, and because it is available in electronic form, it is easy to use parts of the document for quotation purposes.

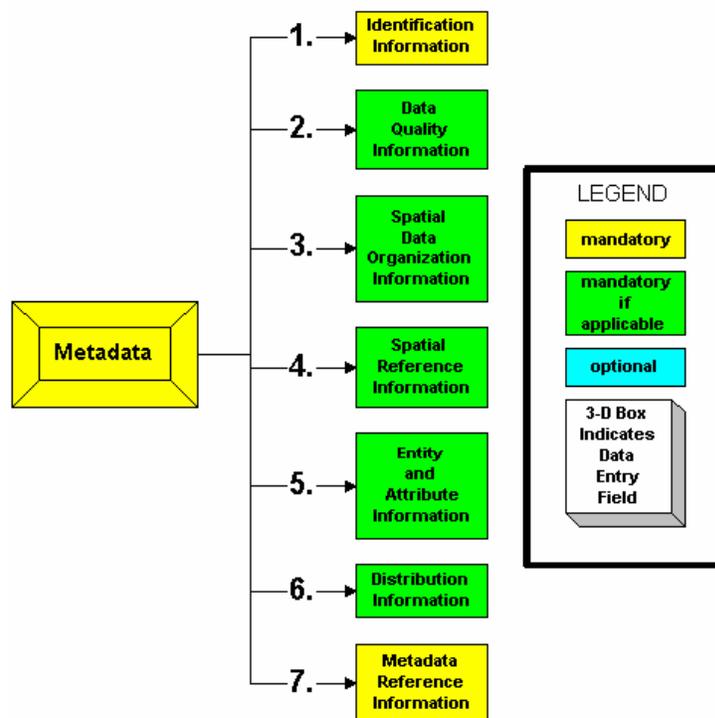


Figure 1 : Content Standards for Digital Geo-spatial Metadata (FGDC)

This figure is a sketch of the structure of a document containing metadata that conforms to the FGDC standard. It can be found on the WWW at <http://www.its.nbs.gov/nbs/meta/meta.htm>. Each box refers to a similar diagram, which in turn contains a number of linked items. At the lowest level the standard specifies elementary fields, each having a name and a data type. This standard prescribes down to the most specific details the information that has to be supplied in order to identify, access, interpret and use a document. It is, in fact, a data model of the documentary metadata. This is far away from the approach to metadata that we have seen in the previous paragraphs. Blaise and EMMA allow you to specify, manage and use your own definitions, GESMES tells you how to format data belonging to a data model specified elsewhere. Metadata standards, on the other hand, prescribe both the form and the content of the

information to be supplied. They provide one specific data model, not a general framework for data models.

They are not tools, but applications. To illustrate what this difference means, one could use EMMA to record and manage the definition of the concepts to be captured according to the FGDC standard, one could use Blaise to build a tool to help in entering and validating documents conforming to the FGDC standard, and one could use GESMES to transfer such documents from one computer to another.

1.5 The whole picture

The areas of metadata covered by the preceding examples do not, of course, have the sharp rectangular edges suggested by figure 2.

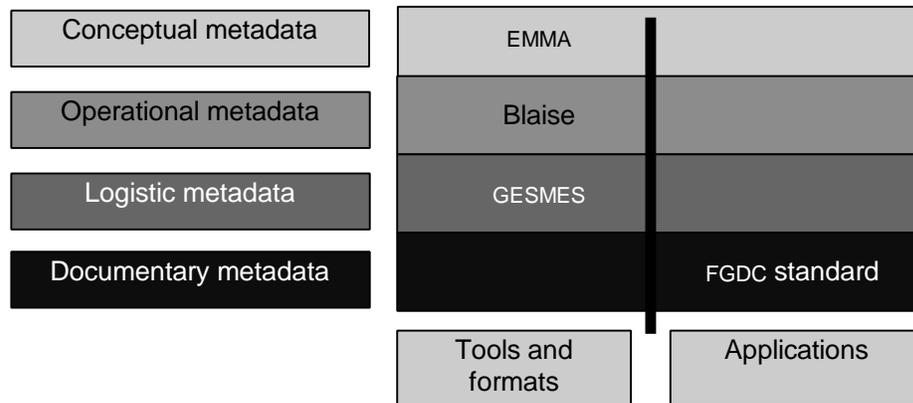


Figure 2 : Semantic area of metadata and meta-information, with a few examples

There is some overlap between them, and wide areas of the metadata concept are not at all covered by any available package. What is missing in this picture is the dynamic aspect: from survey specification to document publication, data and metadata undergo a complex process of specifying, collecting, editing, aggregating and documenting that needs to be analysed to be understood properly, needs to be formalised in order to be managed properly, and eventually needs to be automated. We shall refer to the metadata covering this aspect as *process metadata*.

1.6 Process metadata

In order to illustrate the phenomena that should be controlled by process metadata, let us use the example of a stripped-down survey designed to compute income information. The results have to be published in a table of average income by sex and age class. The size of the population is already known, as well as its distribution over age class and sex. The income information still has to be collected.

This situation implies that a number of concepts have already been provided with a definition, and that both data and metadata are available. Drawing a sample could be one of the first steps of this survey. This process needs to know the population to draw the sample from, and the size of the sample. It produces the sample as its result :

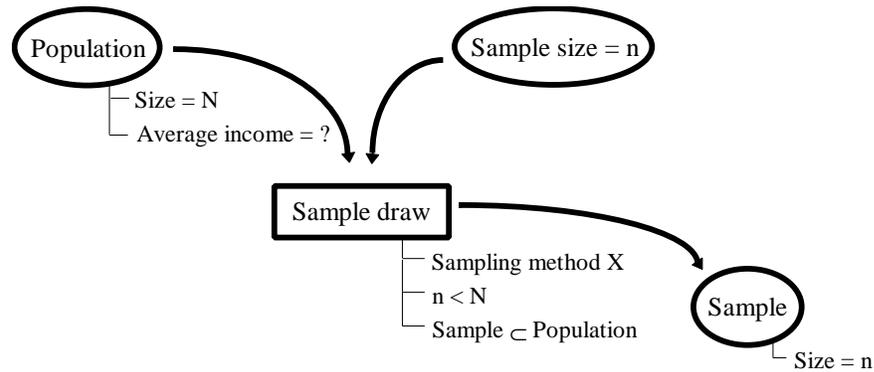


Figure 3 : a sample draw process, defined by its input and output parameters

This diagram represents a process, called Sample draw, which takes two input parameters, Population and Sample size, and produces one output parameter: a Sample of size n. This definition says nothing about the implementation of the process. It could be a manual process performed on a physical card file, an electronic process taking place on computer files, or a mix of the two. The electronic variant could use a relational data base management system or any other type of DBMS, a tailored application, or, again, a mix.

This platform independence of metadata is a very important point. An automated meta-information system must be able to integrate the management of non-automated processes with the management of automated processes. The automated management can trigger a manual process by putting a message on the screen or sending an e-mail message to the person in charge of performing the action. It can then wait for a message telling it the action has taken place. Some processes resist formalisation. The art of carrying them out is passed from old to young, along with the attitude that they cannot be analysed and automated. Such processes will remain manual longer than others, and it is of strategic importance to be able to formalise them in terms of their input and output even if their internal implementation has to wait.

1.7 Modularity of process metadata

An important prerequisite for a concept to become formalised and automated is the possibility of describing it in a modular way. This possibility is present in structured programming languages, where a procedure can be designed as a series of calls to sub-procedures ; in relational databases, where data is analysed in tables through normalisation ; in object oriented design, where an object can be defined as a compound of several smaller objects. It is easy to show that processes, as outlined in the previous paragraph, can also be analysed in smaller processes.

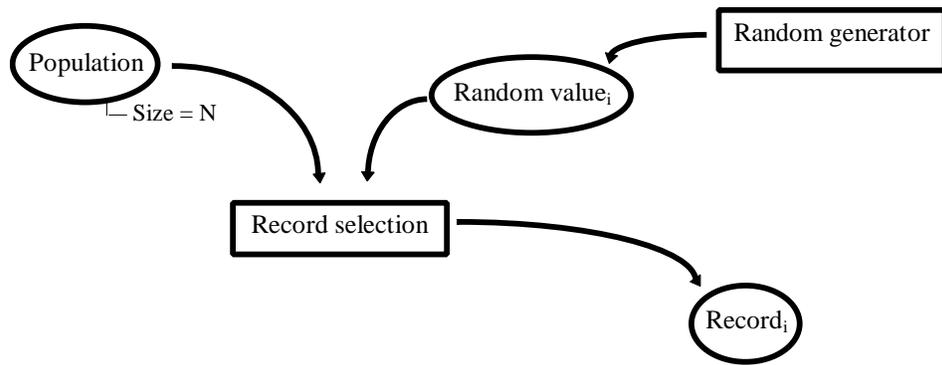


Figure 4 : The record selection process

In the analysis of the sample draw process illustrated in figure 3, one could split the process into n executions of a record selection process. This process would, in turn, need to input a pseudo-random value produced by a generator (see fig 4 and 5).

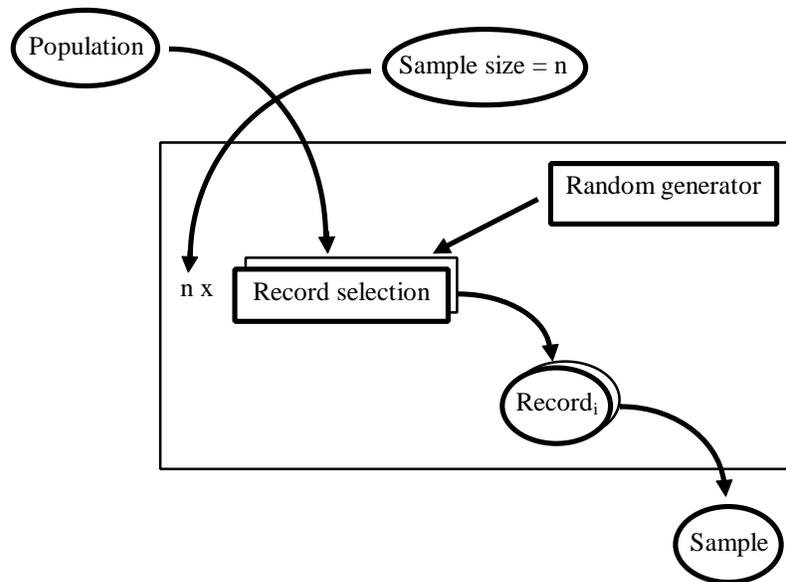


Figure 5 : The sample draw process analysed in n record selections and random generation.

This approach allows us to see a process from two points of view, and concentrate on one of them at a time. Internally, a process has its own structure, which can be designed, understood and maintained without reference to its function within the main process. Externally, a process takes a specific place in the web of sub-processes that contribute to the definition of the whole. While working on one side of the picture, it is safe to forget the other side. At the highest level of the definition, a whole survey can be seen as one compound process.

2. An Integrated Metadata System

In the long term, we wish to have a system that integrates all aspects of metadata (conceptual, operational, logistic and documentary) and covers both data and processes. Specifying the main lines of this system at an abstract level is the first step to take. This general specification can then be used as reference material for short and medium term projects. The second part of this paper states a few points that in our view should guide the general abstract specification of such a long term project.

2.1 A corporate data base ?

Most statistical offices are involved nowadays in the construction of central data bases to manage all their data and metadata. We would like to stress the fact that such a data base is not what we have in mind here. A central data base is an application. It covers the right hand side of the area presented in figure 2. What we are looking at is an integrated system for building, maintaining and using such applications. A tool covering the left hand side of the same area.

2.2 The implementation platform

Considering the high level of abstraction of the system in the first phase of this long term project, implementation aspects should not be taken into consideration. We wish to specify the system independently of the functionality made available by today's data base management systems. We do not know how long the relational model will remain the main standard, whether the object oriented model will take its place, or what implementation platforms will look like in the future. Data base management systems also impose limits on what is feasible. We want to develop our abstract system without taking such limits into account. This should be done in such a way that implementation is possible in any data base management system present or future, or in a general purpose programming language. This platform-independent approach could lead to the conclusion that some parts of the system should use a DBMS, while other parts should be developed in a general purpose programming language.

2.3 Language-based or data-driven ?

As far as specification facilities are concerned, today's systems generally fall into one of two categories: they either offer a specification language allowing to input descriptions and definitions by means of a text file which is processed by a parser, or they offer a series of interactive screens enabling the designer to enter his information by filling in forms. There is, however, no contradiction between the two approaches : a system for managing and retrieving (meta)data can easily have two input channels: a batch-oriented one with a parser, and an interactive one with input forms.

It is an implementation choice whether to supply either or both input methods.

2.4 Active metadata

In the context of operational metadata (1.2), we said metadata should be active. Meta-information should not only be available to the user. It must also actively control the whole statistical process. The following example will illustrate what we mean, and reveal the advantages of such a feature. The table shows journey data in terms of a journey identifier, distance, time and speed.

Journey id	Distance	Time	Speed
	miles	hours	m/s
J1	100	1:00	44.7
J2	45	0:15	80.5
J3	127	1:30	37.8

The speed, of course, is computed from distance and time, but the units of measurement are different. In a language-based system, a formula like the following could be specified :

$$\text{Speed} = (\text{Distance} * 1609.29) / (\text{Time} * 3600)$$

There are two problems with this approach : 1) the system does not know what the constants 1609.29 and 3600 are all about, 2) the system does not know why distance has to be divided by time. It is all in the designer's head. Maybe it has been stated somewhere in a formal design document, or in a meta-information data base, but there is no operational link between those specifications and the use of the formula. If, for example, it is decided that the distance will be presented in kilometres instead of miles, it is the responsibility of the designer to change not only the unit label of the Distance column and its data, but also the first constant of the formula for Speed. On the other hand, it would be difficult to support interactive visual definition and maintenance of this computation.

The approach we suggest allows the system to do the job of defining and maintaining this computation. Measurement units and scaling factors must be part of the metadata. The system must be aware of a measurable concept such as Distance, and it must know all about the units that can be used to express a value of a given concept (meters, kilometres, miles, inches, etc.). If told to convert a value from one unit to the other, such as inches to kilometres, the system must be able to access the definition of the conversion formula. This can involve more than just a scaling factor, as is the case for conversion of a temperature value from Fahrenheit to Celsius. Derived concepts such as Speed are defined by reference to two or more other concepts : Speed is defined as the quotient of Distance by Time, and the measurement units of Speed are defined by reference to the measurement units of Distance and Time. Finally, each numeric field is

specified by reference to a unit belonging to a predefined measurable concept.

Now it is possible to replace the previous formula by a much simpler one :

$$\text{Speed} = \text{Distance} / \text{Time}$$

The system knows all about the relationship between the three concepts referred to by these fields, so it can do the following things :

- check that the quotient of values in Distance and Time units yields a value in a Speed unit: this can lead the system to reject a formula that is not consistent with the available unit definitions,
- select the scaling factors to apply to the computation, in relation to the units used in the three fields,
- automatically convert the values of a column if the unit scale is changed,
- automatically adapt the scaling factors in the computation if the unit scale is changed in one of the columns.

This also makes it easy to support visual interactive specification of the Speed column. The following scenario becomes possible, supposing there is a table containing the first three columns of our example: Journey id, Distance and Time :

- use the menu to add a column,
- select columns 2 and 3 and drag them onto the new column,
- the system looks up its list of derived concepts for one that is defined in terms of Distance and Time. It finds Speed, which is specified as the quotient of Distance by Time,
- the system opens a list box with all defined units of speed,
- select m/s,
- the system fills the new column with computed values,
- the system looks up the list of labels available as column titles for the Speed concept and presents a list box,
- choose the label Speed.

This example illustrates the fact that the language and interactive specification approaches are not contradictory, and shows that a good operational metadata concept can help simplify work in either case.

2.5 Events

Processes take place at a certain point in time, as soon as predefined conditions are met. Such a condition can simply be the availability of all the required input. Reaching a certain date can also trigger a process. Other possibilities are a human decision to launch the process, a state reached by another process, etc. If process meta-information is to be automated, the system needs to know about events triggering processes. These are an integral part of the metadata.

2.6 From concept to documentation

The same concept can play various roles in different parts of a survey. A publication table, for instance, could be supplied with information referring to the non-response rate and the weighting method used to compensate for it. At this stage, it is documentary meta-information. However, both concepts have been present in an earlier phase : non-response has been captured as refused or unknown information and was aggregated to a non-response rate indicator, and the weighting method was implemented in a process that computed the weights. This has to be captured in operational meta-information. But first of all one needs to have a clear definition of the concepts of non-response, weighting and the available algorithms. This is conceptual meta-information.

The system has to know how to supply the individual cases of non-response to the weighting process, how to merge them into a non-response rate, and how to make this available for documentation. The information about the weighting algorithm must be stored only once, and accessed both by the weighting process and the documentation.

3. System Design

As stated in the first part of this paper, the first step will be an abstract specification of the system to be built. The design will have to meet the following four requirements : implementation independence, unlimited coverage of all aspects of meta-information, unity of definition, and object orientation. In the third part we will now concentrate on these four points and conclude with some examples.

3.1 Implementation independence

We have already mentioned that some processes are implementation-resistant. This is partly due to the fact that they have never yet been analysed formally. It is also a question of culture. There are statistics that are generally recognised to be dependent on the specialist's know-how, something that "could never, ever be replaced by software", a handy alibi for the status-quo. So we want to integrate these processes into the system by describing them as black boxes. The system only knows they exist, what input they need, what output they produce, what the relationship is between input and input, and what events trigger them.

Taking the human factor into account in this way implies building a system that runs on two platforms at a time: the computer and the human brain.

Another argument for implementation independence is the fact that some aspects of metadata management already have been implemented in systems that are available on the market. EMMA is one of them. The domain of metadata is so vast that it is not possible to cover it all with one universal package. However, by integrating third-party software into the system we can easily extend its functionality to what is already available. Blaise III has shown that this approach is very powerful. Blaise does not carry out tasks that are made available by other packages. One example is statistical analysis. Quite a few good packages do the job, SPSS is but an example. A Blaise application can offer statistical analysis by integrating SPSS. For instance, there can be a call to SPSS in the menu, which triggers a data selection process, creation of an SPSS set-up on the fly, after which control is transferred to SPSS to create the system file and interact with the user directly. When the user exits SPSS, the application goes on to whatever has been defined as the next step. Any third-party software can be used in this manner to process data managed by the Blaise system. This works for data, and we are convinced that it will also work for metadata. So our choice of platforms is not limited to data base management systems and general purpose programming languages: we also have third-party software to choose from.

Our own software is also a candidate for integration in the metadata system. In the case of Blaise, this could happen in various ways. The system could transfer operational metadata to Blaise and call upon the data management functionality of Blaise for such things as interviewing and data editing. Or we might end up building such a general tool for designing metadata management subsystems that Blaise could be rewritten in terms of the new system. Alternatively, the functionality of Blaise could be extended in such a way that Blaise itself would become the general metadata management system. We do not know yet. And we are not in a hurry to decide.

Presently we are witnessing a rapid evolution of the concept of inter-program communication. In the past, programs could communicate through ASCII files ;later came the dBase format; today, they access each other's data through ODBC drivers, or even communicate with each other in real time through OLE. Software is no longer by definition an executable program. There are a host of formats for software that does not run autonomously : there is DLL, VBX and OCX, and there are more to come. The day will come when those formats will be preferred to the heavy monolithical executables of today. The end-user will build his own private application by clicking diverse modules into place. A word processing module from Microsoft, a spelling checker from WordPerfect, a data management module from Borland, and maybe a metadata management module from Statistics Netherlands. So the system could be implemented as a loose collection of modules selectable by the user.

The system we have in mind might very well never be implemented completely. The first prototypes will be limited subsystems, such as a module for classification definition and maintenance or a tool for co-ordinating and standardising the legacy meta-information scattered over all

the surveys and publications of an institution. The implementation of such loosely related applications could vary greatly, and this is another reason to wish to keep implementation aspects out of the general design.

Finally, the implementation landscape is so dynamic that switching from one implementation form to another between releases needs to be an easy and fast operation. This is not possible if implementation aspects are part of the design.

3.2 Unlimited coverage of all aspects

It is not easy to give a well delimited definition of what metadata will be considered relevant, and what should be left out of the system. Metadata are used to manage data. What do we use to manage metadata ? If they are to be operational, they need to be described, they need to fit their own model. The metadata of the metadata. The meta-metadata. This is also meta-information. Here are a couple of examples that will show that this is also relevant to our system.

The process of data editing has two objects : its first function is to identify and correct errors in the data set. The input of the editing process is a dirty data set, and the output is a clean (or less dirty) data set. The user of the data set considers this to be the primary function of data editing. But the designer has something else in mind : systematic errors can be symptomatic of design mistakes that have to be corrected before the same data are collected again. So the designer also wants data editing to produce an edit trail. This edit trail (which belongs to the output of the data editing process) will be used as input by the maintenance process. The output of the maintenance process is a new data entry process that produces better data and needs less editing. The new entry process cannot take place before the maintenance process has executed.

In this example, the survey design process has become an integral part of the designed survey process. There is no a-priori reason to exclude this type of reflexivity. This, of course, does not mean that we have the ambition of automating every single sub-process that we identify. But we want to see the whole picture before we start implementing some parts of it, and we do not want to exclude any relevant aspect because it might turn out to be overkill.

We would like to draw our next example from the problem of multilingualism. Some countries have two or more official languages, and maybe also one or more widely used non-official ones. In such cases good communication requires the use of more than one language, and sometimes the use of multilingual documents is imposed by law. This applies to every bit of natural-language meta-information : question texts of an interview, titles and footnotes of a publication table, etc. This poses a heavy maintenance problem if the task of warning all translators is the responsibility of any one entering new texts or updating old ones, or if it is the responsibility of translators to check whether there are new or updated meta-information texts. The whole operation of maintaining texts in multiple languages is much easier if the system is designed to keep multilingual texts synchronised by sending a warning to translators at every change of the original set of texts. Here again, we have something that is marginal to the survey process and belongs to the design process, something, nevertheless, that we do not wish to exclude from the possible functionality of our future system.

Another point that falls under this paragraph is the range of applications of a meta-information system. We want to be able to use it for all activities involving data and/or metadata. It must be able to control the co-ordination

and standardisation of concepts, to run and monitor processes, to manage resources, to produce documentation, etc.

3.3 Unity of definition

A concept can be present at many different points of a project. Most concepts have a definition (conceptual meta-information), a set of operations (operational meta-information), and user-oriented documentation (documentary meta-information). We do not want these different aspects of one concept to be scattered all over the system. One must be able to access all aspects of a given concept through the unique object representing it. The rule is : one concept, one polyvalent definition.

3.4 Object orientation

In this paper we have expressed the idea that meta-information should not only be stored and retrieved, managed and maintained, but that it should have tasks to perform and know how and when to perform them. In the previous section we have also demanded that every concept should concentrate all elements of its definition. This combination of data and behaviour is a typical result of an object oriented analysis. It is our conviction that object orientation can be a significant contribution to a powerful and intuitive metadata management system. This does not mean that we intend to use an object oriented language or object oriented data base management system: these implementation decisions will have to be taken later. By an object oriented system we mean a system in which resources, data and processes are represented as objects combining information and behaviour.

The importance given here to objects may come as a surprise to the reader : object orientation is often seen as a programming style, an attitude towards implementation techniques, something that does not concern the user. It is also often considered to be something hard that needs a considerable time investment in order to be applied successfully. We do not agree with this point of view. Object orientation is first of all a design attitude, which in our view contributes to better control of system complexity. It is the next logical step after structured design. A system whose users are involved in designing surveys should help them to apply the most efficient design methodology.

By managing all information and behaviour of one concept, a unique object representing that concept is a contribution to the overall unity and coherence of the system. In this view of things there are no barriers between conceptual, operational and documentary metadata.

What is the behaviour of a metadata object ? It is anything the object can do to supply information or to contribute to one or more processes. An object should be able to supply its name or identifying key, its definition, a list of all the objects it is related to ; it should be able to write itself in Pascal or C code or in an SPSS set-up, to store itself in a relational data base, detect that its documentation in its original English form is more

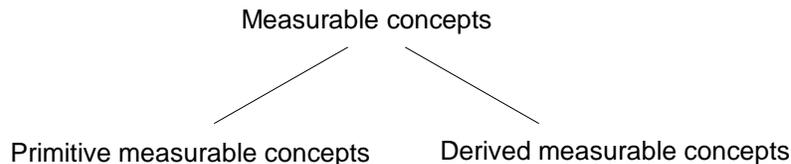
recent than its French translation and trigger a translation process (in which the human black box will be involved, of course), perform computations and selections, take decisions, and carry out any other relevant tasks.

3.5 Examples

To illustrate these ideas, let us follow the life cycle of the concept of Speed as represented by an object in the system. We will refer to the table represented in section 2.4.

Before we can use Speed in a survey, we need a definition of the concept of speed. This definition must be common to all surveys : we want to be able to relate speed information over surveys. So we need an object to represent the concept of speed. Speed is a concept that can be measured, it can have values expressed in units belonging exclusively to speed. The system offers a predefined class of objects, the Measurable Concepts, which is designed to model things like distances, times and speeds.

There are two sorts of measurable concepts: primitives, like distance and time, which are defined independently of any other measurable concept, and derived, which are defined in terms of one or more other measurable concepts. The two have in common the possibility of defining and using a list of units in which to express values. The different units belonging to one concept can be interrelated in order to provide both their definition and their conversion formulae.



Primitive and derived measurable concepts can be defined in terms of what they share and in terms of their differences. We model them with two different classes of objects, but when we are interested in their common features, we can pretend they belong to the same class and forget their differences. They are interrelated in the following way:

Any Primitive measurable concept is a Measurable concept, and so is any Derived measurable concept. The relationship, however, does not hold the other way round. The class of Measurable concepts defines all features common to both: a natural-language definition, a set of units, and an operational definition of the relationship between the available units.

Derived measurable concepts add to this a list of related measurable concepts and an operational definition linking them in a formula. In the case of Speed, the operational definition is Distance/Time. The Distance object, which, let us assume, has already been defined, provides the following units: *mile*, *yard*, *inch*, *km*, *m*, and *cm*. One of them, e.g. *m*, is

defined as the primary unit. All others are defined by reference to the primary unit and a multiplication factor. The Time object is very similar. Its primary unit could be chosen as being the *hour*, which can be expressed in *minutes*, *seconds*, *days* and *weeks* (months and years address the issue of variable scaling factors, which we will not discuss here). Theoretically all combinations of Distance/Time units could yield a Speed unit. But in practice only a few are used. Only three of them will be taken in the definition of Speed : *mph* (miles/hours), *kmh* (km/hours) and *m/s* (m/seconds). Note that the symbol of a derived unit is not always identical to its operational definition : see *mph* vs *m/s*. A derived measurable concept like Speed has no primary unit : all units are derived from those of the concepts referred to.

In order to collect and publish the information shown in the table of section 2.4, we need to define a data model containing fields for a journey identifier, and for Distance, Time and Speed. The mechanism for defining fields is the same as the one for measurable concepts : the system provides a predefined class to model field objects. For the definition of the field object Distance, there is no need to rewrite or copy the definition of the concept object Distance and its units : the Field class provides a reference to a concept for its definition, and knows how to access the functionality of its concept. If the Distance field object is asked to convert its value from miles to kilometres, it will pass the request over to the Distance concept object along with the value to be converted. This mechanism, by which an object draws upon the functionality of another object, is called delegation, and is one of the most powerful features of object orientation. Note that the Measurable concepts have no values : they have only definitions and operations. Field objects have values, and they can use the Concept objects they refer to in order to perform the required operations.

The functionality of the Speed object is a good illustration of the power of the delegation concept. This is what allows us to design the Speed column by dragging the other two onto it (see 2.4). Adding a column to the table creates a new field, but without any definition. Dropping the other two fields onto it triggers the automatic definition process, which works as follows : the new field asks the other two to supply information about what they mean together. They send the question back to their respective concepts. The two find out that they only collaborate in the definition of Speed. A reference to the Speed concept is sent back to the new field. The field inserts the reference to the Speed concept in its definition. From now on this is the Speed field. It asks the Speed concept for a list of available units and asks the user to choose. It passes the user's choice, *m/s*, back to the Speed concept and asks it to return the Distance and Time units that support *m/s*. The Speed concept returns *m* for Distance and *seconds* for Time. The Speed field asks the Distance field to supply its value in *m*. Because the Distance field knows it is using the *miles* unit, it sends its value to the Distance concept and asks it to convert from *miles* to *m*. On receiving the result it passes it back to the Speed field. The same happens to the Time value, which the Time field will send to the Speed field after having it converted to seconds by the Time concept. Having the two required values, the Speed field can send them to the Speed concept and request the computation of a Speed value. It is the responsibility of the Speed concept to perform the division and send back the result.

At first view this looks complicated, and it would be if it had to be implemented in a procedural way. Objects, however, allow the user to define this interaction as a series of messages and responses exchanged by the objects. Most of this behaviour can be implemented in the abstract objects available in the system. So the preceding scenario turns out to be quite simple to design. When the developer comes to defining specific measurable concepts and their units, the abstract classes defining measurable concepts and their features, including their interaction with field objects, are already present in the system, with all their functionality. Fields and concepts already know how to communicate with each other. All the designer has to do is supply definitions for new concepts. The task of defining Distance, Time and Speed will take only a few minutes, after which the objects are operational to define Distance, Time and Speed fields throughout the whole survey, or rather for all surveys.

What we want to offer our users is all the advantages of object orientation without the burden of object oriented theory and rules. Blaise has shown that this was possible for structured design: Blaise offers a framework in which developers work in a structured way without ever having learned to do so the hard way. One of our users told me one day about a software specialist who was looking at his work and said he was doing structured programming: being told such a thing made him feel like Mr Jourdain who had been producing prose for his whole life without knowing so. We are convinced that this can work for object oriented design too. The only way to prove it is to build the system and look what happens. It is a bet, but we are confident that the odds are on our side.

Notice the definition of fields by reference to concepts such as Distance and Time, and the definition of the concepts themselves are two different design sub-processes. Keeping them separate promotes both modularity and standardisation : it makes the definition of concepts independent of their use. This approach enables to separate tasks that are normally performed by different persons: defining concepts and maintaining classifications is usually not the responsibility of people in charge of defining and maintaining questionnaires or publishing tables. By integrating those tasks in different sub-processes we ensure that they do not interfere with each other.

Conversion of Age to Age class will offer another example of the possible contribution of object orientation to making metadata more readily available both to software and to the user. Age information is usually present in microdata in the form of an age field. In cross tables, age information is often present in the form of age classes defining the rows or the columns. This could be modelled by the definition of an Age concept and an Age class concept, which are interrelated. The Age class concept should know what the relationship is between an Age value and an Age class value. It would be expected to know the answer to questions such as : what age class does the age value 45 belong to ? Is the age value 45 greater than (all the values belonging to) the age class 31-40 ? What is the highest age value belonging to age class 61-69? In other design methodologies, where processes are defined far away from the data, such questions would be implemented in separate functions. An object oriented system will allow to manage all these aspects as part of one object : the Age class object.

4. Conclusion

In this paper we have tried to present some very general ideas about metadata. The aim is not yet to provide solutions, but to define the field for further research. It has become clear that the concept of metadata is very diverse, and that a general approach is needed in order to provide generally applicable solutions.

We have learned some basic principles about metadata that will help us in further research :

- Meta-information is interpreted in almost as many ways as there are implementations of meta-information concepts.
- In order to talk about a system covering all meta-information aspects, one must observe meta-information from a broad perspective.
- Meta-information should be defined in such a way as to become operational. That is : its usage should not be limited to one task, say documentation, but it should remain open for implementing other tasks, even some that may not yet be known at design time.
- Design principles such as object orientation should play an important part in the functionality of a meta-information system.
- The ultimate meta-information system will not be one system covering all meta-information aspects, but a loose combination of other systems and tools, sharing the same meta-information principles.

Guided by these principles we will go on searching for ways to handle metadata that will enable us to compare and, more importantly, combine existing metadata tools. Once we reach a metadata concept of a sufficient level of generality and abstraction, we can start designing a system that will contribute to easing metadata management.

Documenting questionnaires

Jean-Pierre Kent and Leon Willenborg, Statistics Netherlands

1. Introduction

The growing possibilities of Blaise III have led designers to build very complex survey instruments. It has become increasingly difficult for users to keep control of the content and structure of questionnaires. Although the Blaise language was designed with self documentation in mind, it takes some knowledge and experience to understand a large questionnaire.

It has become possible to develop and administer data collection instruments so complicated, that researchers, even the original instrument and data producers, have difficulty comprehending them in their entirety. It has become difficult to fully understand the process that leads to responses to each of the items as they ultimately appear on data files.

This rises the question of the feasibility of a tool to represent the content and logic of a questionnaire in a human-readable way. This concern is not limited to the Blaise user community: we read the following in the APDU Newsletter, November, 1993: "how are users going to get access to and develop a thorough understanding of the new survey instrument when it is so complex it requires 6 megabytes or more of memory on a 486 computer to execute?" (quoted by Doyle 1996).

Such concerns are both for the user of the data and for the user of the Blaise system: the former needs to know how the data are structured and how they were collected; the latter needs to keep under control the costs of producing this documentation.

The present paper is primarily intended as a presentation of the state of the problem. It also presents the issues and the costs involved in implementing documentation functionality in Blaise.

2. Historical perspective

When surveys were conducted with PAPI (Paper And Pencil Interviewing) technology, there was no need for separate documentation: the paper form, developed for the interviewer's use, contained all the information available about the questionnaire. Moreover, routing information was kept simple, in order to allow the interviewer to process branching conditions himself. Routing logic was kept simple, and the questionnaire form was the documentation.

The advent of CAI (Computer Assisted Interviewing) brought about two changes in this picture: the paper document was replaced by a computer program, and it became possible to implement more complex branching logic. So users were faced with data collected with a more complex instrument and less documentation to understand it..

In the early times of CAI, when questionnaires were developed for XT computers, in which the operating system, the software and the data had to share 640 KB of memory, instruments were limited to a size that made it possible to document them by hand. We have examples of flow charts created and maintained by hand. The progress of hardware and software technology, however, have opened the way for very large questionnaires: the cost of creating and maintaining the documentation by hand has become prohibitive.

It must also be mentioned that manual documentation can be a source of errors. Whatever is used to express the content and logic of a questionnaire in a way accessible to a broad group of users, whether it is made of flowchart icons, indented paragraphs or natural language references, the information is conveyed in a form essentially different from the Blaise language. Although automatic translation is possible between the two (it should be, otherwise the documentation formalism is poorly designed), the human translator cannot perform the task without extracting the meaning from the Blaise source and re-expressing it in the documentation formalism. This process is subjective and error-prone. It can spawn subtle errors that will usually go by unnoticed, causing the reader to form an erroneous picture. Therefore, automatic documentation is a must.

3. Paper, screen, or both?

In the early times of CAI, developers sometimes asked for the possibility of creating a paper version of the questionnaire. At the beginning, it was not clear why such a document was needed. Interviews were only used in their electronic form. One clearly did not have a multi-mode survey in mind: the intention was not to use CAPI and PAPI side by side.

It eventually became clear that developers needed a document to help them build a picture of the instrument, and to convey that picture to their users. This information contributed to the ideas that led to the creation of the Structure Viewer (hereafter: sv). The sv does more than could be expected from a paper document. It allows the user to picture the data model at various levels of the hierarchy, and to concentrate on the aspects that are relevant for answering any specific question about the structure of the model.

The sv, however, does not respond to the need for a way of documenting the *questionnaire*: the questionnaire is not just the data structure of the model, it is the sum of all features that contribute to making the interview: routing and computations play an important role, and there is nothing in Blaise yet to answer questions relative to these aspects.

In our view, the best response to the expressed documentation needs would be a tool allowing to approach the RULES information in the same way as the SV approaches the FIELDS information. Something that could be called the Route Viewer, or the Rules Viewer (RV). Such a tool should be able to represent the route through the questionnaire as a network of paths, to fold and unfold them on user command, and maybe to create a printed report of the information selected by the user.

In this perspective, screen interaction is dominant: much more relevant information can be obtained by zooming into the area of interest than by visually inspecting a paper document.

It can be argued that the screen document is the only relevant one. Nowadays data are more and more available on line. For on-line access of data, you need to be able to access the metadata on line as well. This type of work implies every user has a computer. In this context, an interactive tool will always be preferred to a passive paper document. We therefore expect that a time will come when paper documentation will no longer be asked for.

In the mean time, we think, the best solution is an interactive tool capable of producing paper output. Although the TELPERION project discussed in this paper (see paragraph 5) was not meant as a documentation tool, it offers an interesting example of what could be achieved.

4. A paper documentation generator

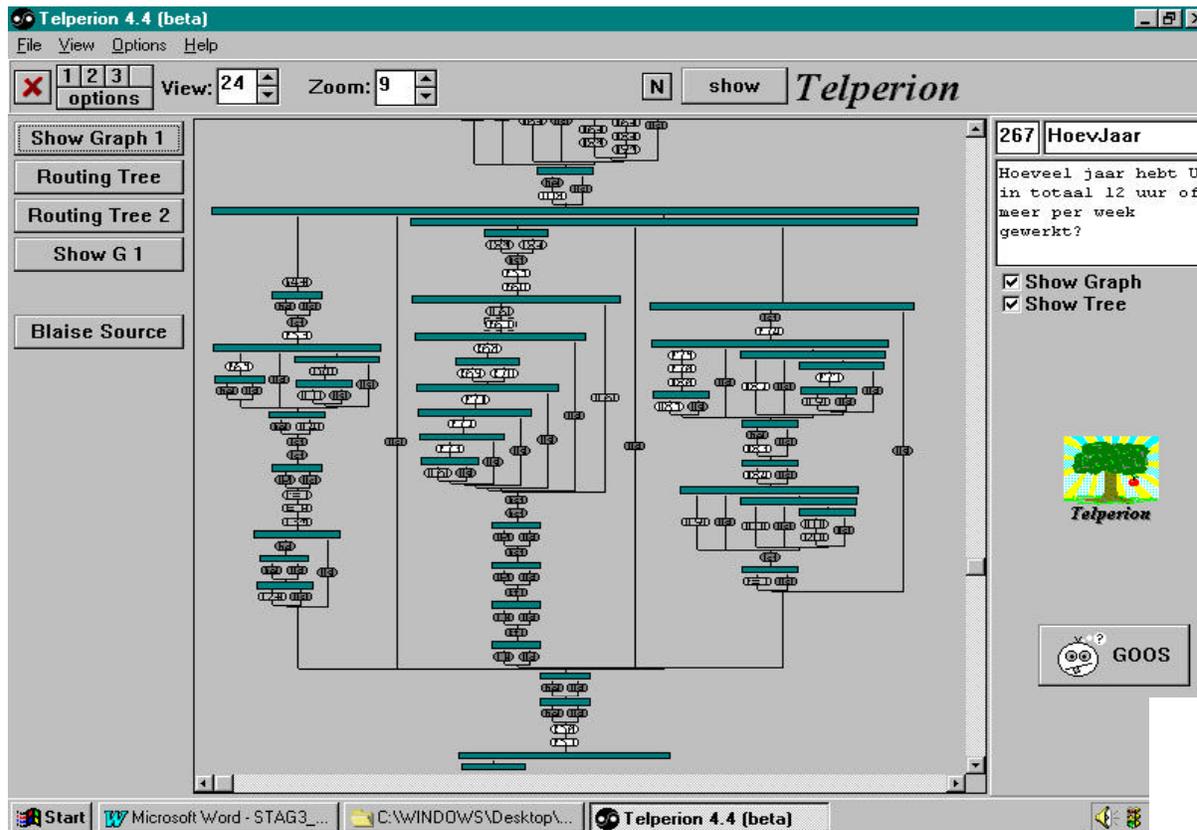
Elsewhere in this volume Steve Anderson presents the project that was developed in the United Kingdom under the acronym BAD (Blaise Automatic Documentation). The reader is referred to Steve's article for a discussion of the problems involved.

BAD is an example of the paper-oriented approach to questionnaire documentation. The following section shows what a screen-oriented tool can look like.

5. An interactive RULES Viewer

TELPERION is a computer program that can be used to plot routing graphs of questionnaires. The currently available version, Telperion I, is only a prototype. Telperion was developed at Statistics Netherlands in 1996 by Dam Backer, then a student of Computer Science of the University of Utrecht, supervised by Leon Willenborg and Goos Kant (University of Utrecht and ORTEC Consultants) and Jan van Leeuwen (University of Utrecht). The initial goal was to develop a program that was able to plot an arbitrary routing graph. Soon after the project was launched, the decision was taken to focus exclusively on Blaise routing graphs. The reasons for this were manifold: time considerations played a role, the fact that Blaise questionnaires were available at Statistics Netherlands and offered concrete examples of routing structures, and also the fact that Blaise questionnaires have a rather simple routing structure, namely series-parallel routing graphs which have the nice property of being planar.

Figure 1



TELPERION I is able to plot routing graphs on a computer screen, but not on paper. Figure 1 shows a screen dump of such a plot (taken from Backer, 1996). A feature of the program is that it allows the definition of blocks, that can either be plotted as a single point or as a more detailed structure consisting of points, blocks and transitions. This allows one to view a routing structure, globally or locally, at various levels of detail.

The currently available version of TELPERION should be viewed as a first prototype. We will now discuss a few features that could be considered for incorporation in a future version. The following list is not intended to be complete: suggestions and feedback from users are welcome.

Routing structure on paper. The current version of TELPERION only yields images of routing structures on the monitor screen. For the production of publishable documentation, however, printer output could be required. As a short-term solution one could work with screen dumps, but for a proper document one needs to rethink the representation problem. Some of the functionality for screen prints is not needed (e.g. folding and unfolding blocks), and some features should be implemented in a different way – one typical example being the link between a question on the routing graph and a full representation of the information referring to it (name,

type and various texts). The use of colour is another dimension that could be investigated.

Functionality of a routing graph tester. The current use of TELPERION is rather passive: displaying in a visually appealing form what is already in the Blaise questionnaire. One could also try to develop it into a tool for testing questionnaires while they are being developed. In order to be able to plot a “questionnaire under construction” one needs a syntactically correct questionnaire, which need not be complete. A questionnaire could be constructed through a process of stepwise refinements, by progressively replacing empty blocks by more elaborate structures, with the possibility of viewing the routing structures at each stage of construction. With a few additional features, TELPERION would allow a questionnaire designer to view a sub-graph between a given starting point and end point, to select a path from the routing and inspect a list of conditions. This would turn it into a rich complement of the SV.

A RULES editor. Originally, the SV was meant to become a structure editor: a tool for the interactive construction of questionnaires. It is not yet clear whether and when it will be possible to assign this project sufficient priority. An alternate possibility would be to integrate TELPERION into the Blaise Control Centre and give it editing functionality: this would make it possible to build blocks interactively by working on the routing structure, instead of taking the data structure as the starting point. The ideal solution, of course, would be to build editing capabilities into both tools.

Graphical representations of survey results. For a completed survey it could be interesting to see the flow through the questionnaire, by plotting the routing graph with edges of varying degrees of thickness: the thicker (or the redder, if colour is permitted) an edge, the more individuals have responded to the corresponding questions. This would give a quick view of the main streams through the questionnaire and also draw attention to the parts where nothing happens. This information would be useful for testing questionnaires. It may require redrawing the graph, in order to plot the main streams as much as possible in the middle of the page.

6. Documenting from the Blaise Control Centre ?

The process of documenting a Blaise data model requires both an interface to the data model, and the capability of presenting the information in the desired form. It also needs a way to communicate with the designer, in order to determine how each element of metadata will be displayed. In the case of an interactive tool with edit functionality, these two aspects need to be integrated. This requirement need not be met by a read-only tool like those mentioned in this paper: it is possible to extract metadata in the first step, and to represent this information in the second step without further accessing the metadata file. This two-step approach is the best way to implement questionnaire documentation as a short-term project. We will concentrate here on the first step: extracting metadata.

6.1. The interface to the metadata

A prepared Blaise data model is available in two different forms: the Blaise source and the metadata file (the `~MI` file). The advantage of using the source is evident: the Blaise language is fully documented, writing a parser for it is the only step needed for access to the data model. Implementing such a parser can be done without help from the Blaise team. However, this method could entail a substantial maintenance problem, in the eventuality of future changes in the specification of the Blaise language. Although such changes are excluded for the short term, this consideration has led developers to discard this option.

Another possibility is the use of the Blaise metadata engine library to access the `~MI` file. Changes in the format of the metadata file would have a negligible maintenance impact, because the only step needed for upgrading an application using the library is a recompile with the new version.

On the other hand, this is an awkward approach, because the library was exclusively designed for internal use. It does not have a simple API, but a complex family of objects. Use of this library requires some coaching from Statistics Netherlands. Although the two projects mentioned here show that this approach can be successful, we do not wish to recommend it. In our view, the documentation developers should be able to do their work without being dependent on the Blaise team.

The normal way of extracting information from the metadata file is to use CAMELEON. For historical reasons, however, the functionality of CAMELEON has been limited so far to the extraction of information from the `FIELDS` and `TYPE` sections of the data model. In spite of the attention that has been paid to the problem of extracting information from the `RULES` section, this work was never assigned the priority it would need in order to reach completion. We would like to suggest here that if we are asked to offer documentation functionality within the short term, our best answer would be to extend the functionality of CAMELEON to the `RULES` section.

6.2 Authoring with documentation in mind

A questionnaire developer works primarily to produce a data model, and a data entry application to collect the data. If these are the author's sole concerns, an automatic documenting tool is doomed to fail. Whether metadata are extracted with CAMELEON or any other tool, hints will be needed to guide the program in its choices and to make conditions readable to a user not involved in development or collection. If, for instance, a series of questions falls under the following condition:

```
IF Age >= 65 AND Pension = No THEN
```

the documentation should mention something like "elderly people without a pension". This wording could be different from what is needed in error messages. So, clearly, the documentation needs a text that can be extracted neither from the conditional expression, nor from the error text.

The documentation tool also has to know what parts of the `RULES` are relevant for output and which are not. What about computations? Most of

them should not make it into the documentation, but some are relevant to the meaning of the data. For instance, complex routing conditions can be pre-computed in local variables: these calculations should appear in the documentation.

The solution to both problems like in the use of the multi-language functionality of Blaise. Originally the languages feature was built into the system in order to support multilingual questionnaires. It can, however, be used for the specification of any text. We know projects in which “languages” are defined and used to specify alternate variable names and labels for analysis and tabulation packages.

CAMELEON can be tuned to use any of the defined languages, and to switch in the middle of a setup. In order to use this feature for documentation, it is sufficient to define a documentation “language” in the data model, and to switch to this language in the setup.

We cannot give here an example of a CAMELEON setup, because the syntax for accessing RULES information is not defined yet. The example in Figure 2 shows how a data model can be defined with documentation texts.

7. Literature

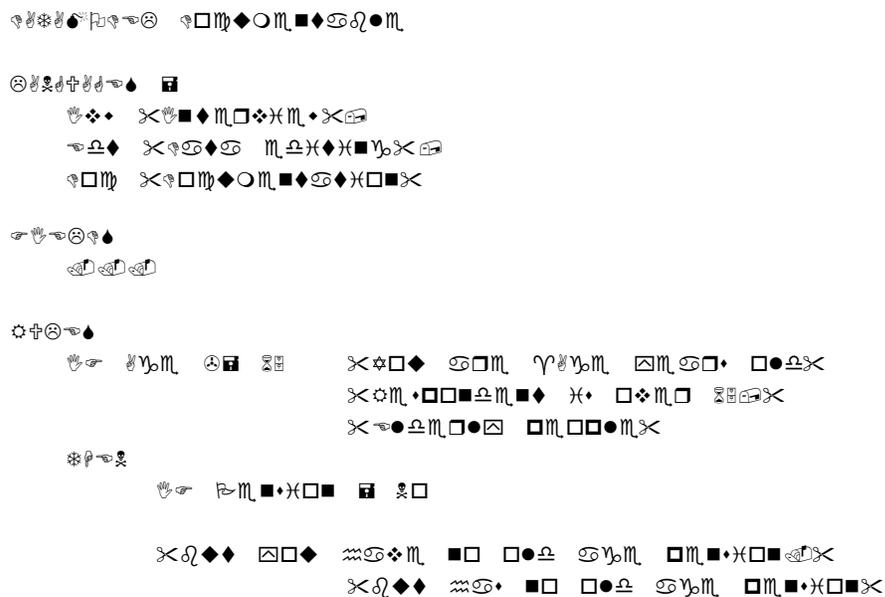
Anderson, S., 1997: “Automated Paper Documentation of Blaise III Datamodels” (elsewhere in this volume).

Backer, D., 1996: *Drawing Questionnaire Routing Graphs*, Report, Department of Statistical Methods, Statistics Netherlands, Voorburg.

Doyle, P., 1997: “Documenting cai instruments: a blueprint for the future.” (Paper to be presented at IASSIST, MAY 1997).

Willenborg, L.C.R.J., 1988: *Computational Aspects of Survey Data Processing*, CWI Tract 54, Centre of Mathematics and Computer Science, Amsterdam.

Figure 2



၂<•*◆ဆီဝ◆◆ ဝ●ဝ ဖွဲ့မ ဝမ■•*ဝ■၂<

✱၉✱

▲၉✱✱✱

၉◆ဆီမဝ၉■၉ဝဝမ ■ ☆မ•

၂<၉ဝမ ၈ဝ◆ •◆ဝမ ၈ဝ◆ ဆွဲဖွဲ့မ ■ဝ ဝ◆ဆီမဝ
•ဝ◆ဝ၉မ ဝ⁹ *■၉ဝဝမ⁹၂<

၂<ဝဝ ဝ◆ဆီမဝ •ဝ◆ဝ၉မ ဝ⁹ *■၉ဝဝမ⁹ ၉• ၉•◆ဆီ*
၉ဝဝဝမ၉⁹⁹၂<

၂<ဝဝမ ၉၈ဝမ၉⁹⁹မဝ ◆ဝ ဆွဲဖွဲ့မ ဝ◆ဆီမဝ *■၉ဝဝမ
•ဝ◆ဝ၉မ•⁹⁹၂<

✱၉✱၉

✱၉✱၉

✱၉✱●၉၉၉

Manipula & ManiPlus Usage at the National Agricultural Statistics Service

David Knopf and Roger Schou, National Agricultural Statistics Service, USA

1. Introduction

The National Agricultural Statistics Service (NASS) has developed CASIC systems using Blaise III. These systems integrate data collection and interactive data editing. In addition to data collection and editing, many survey processes must be performed. Examples of these processes include file creation, data movement and report generation. Each process must be easy to execute and run efficiently. Nearly all of the survey system processes are performed by Manipula and ManiPlus. This paper will describe the many roles played by Manipula and ManiPlus, and how the setups have been written to optimize performance.

2. Uses of Manipula

Creating Data Files

One of the first survey steps is the creation of Blaise data files. Five different data files are created using four Manipula setups. One data file created in this step is the primary file used for CATI data collection. Three others are External data files read by the primary data model. ASCII data files containing sample unit information are used as input. Names, addresses, telephone numbers and other sample information is output into Blaise files.

Updating Data Files

At different points during the survey process Blaise data files are updated using Manipula. Setting the Time Zone indicator for data collection purposes is one example. Surveys that NASS conducts may include forms for sample units located across different time zones. In this Manipula setup only an Update file is defined. Address information in the form is used to compute the Time Zone indicator and to update the Time Zone field in the form. The Time Zone field is later used by the CATI Call Scheduler.

Another common updating practice is computing a value for a Secondary Key field. Other Manipula processing will then be done by the Secondary

Key. ProcessSwitch is an example of a Secondary Key field that we use. ProcessSwitch will be assigned a value based on a selected process and a subsequent Manipula setup will process by ProcessSwitch.

Moving Data

Manipula is used to move data from one data file to another data file during the survey process. Survey data resides in one of two data files. Initially all forms are in the CATI data file. During the survey, data is collected using CATI and paper questionnaires. A process we call Transition is run on a regular basis to move forms with completed data to the EDIT data file. Data from CATI moves with the form while data collected on paper questionnaires will be read into the Blaise data file from an ASCII item code data file. The code data file was prepared with a high speed, heads down data entry software. Conceptually, Transition is simple. The actual process, though, is very complicated utilizing several different Manipula setups to accomplish the operation.

Checking Forms (Integral Check)

Another function of Manipula is to perform a check on all forms in the data file or just a portion of the data file. CHECKRULES marks the forms Clean, Dirty or Suspect for the subsequent interactive edit step. Forms which are Dirty or Suspect are reviewed and modified until Clean. Usually, no further review is done for Clean forms.

Figure 1. *Manipula setup for checking rules.*

```
USES
    InstFmt      'AgSurv'

UPDATEFILE

    EDITData : InstFmt ('AgSurvE', BLAISE)

MANIPULATE

    IF (VAL (PARAMETER) = Managmnt. Batch) THEN
        EDITData. CHECKRULES
    IF EDITData. CHANGED THEN
        EDITData. WRITE
    ENDIF

    ELSEIF Managmnt. Batch > VAL (PARAMETER) THEN
        READY
    ENDIF
```

ENDMANIPULATE

A group number called the Batch Number is assigned to each form. The Batch Number is used in this process to check only a specified group of forms. Normally, checking forms is done for a group after processing through Transition. While the entire data file can be checked, the normal operating procedure is to check by group. Figure 1 contains a setup example.

Creating Reports

Manipula is also used to create several reports. These reports provide information to facilitate the survey process. During a survey period up to thirteen different reports could be generated. Typically five to seven different reports are generated, some of them several times. Input files are Blaise files and output files are ASCII. Use of the PRINT paragraph is common. Content of the reports vary.

Several reports are designed to provide information about survey progress and status at a given point in time. An example of this is the Edit Status Report in Figure 2. This report provides a count of the forms in the interactive edit data file by form status.

Figure 2. Output from Edit Status Report

```
-----  
-----  
FORMS IN THE EDIT DATASET      STATUS   OF  
02/14/1997                     DATE     :  
  
AS703                          TIME : 08:58:12  
-----  
-----  
FORMS                          FORM STATUS   NUMBER OF  
=====
```

FORMS	FORM STATUS	NUMBER OF
CLEAN		600
SUSPECT		45
DIRTY		155

NOTCHECKED	200
CONVERTED TO CODE DATA	1250
NOT ACTIVE	88
=====	=====
TOTAL IN EDIT	2338

Other reports are listings that provide selected information about specific forms. A frequently used listing is the Missing Reports. This is a listing of incomplete reports and the report status. Action can be taken based on the information in this report.

Creating Code Data Files

Following interactive edit, it is necessary to convert the Blaise data to an appropriate ASCII file format for summarization. Code data is the most common format. In this process each field that must have data written to the code data file is identified by a Tag. The Tag is used by Cameleon to create part of the setup code. During this process Manipula reads the Blaise file and writes to the code data file.

3. Maniplus

ManiPlus is playing an ever increasing role in NASS survey systems. Currently ManiPlus is used for a Transactions Processor and Interactive Editing Interface. We anticipate using it to generate Manipula setups based on a specifications data model.

The Transactions Processor updates fields in the form to either withhold or reactivate a form for CATI. Fields are updated in the appointment block (CATIMana) as well a field called DayBatchSwitch. Fields in the appointment block are updated to provide CATI summary and calling history information. DayBatchSwitch is used as a Select Field in the Call Scheduler definition to exclude forms from a Day Batch. Forms are also removed or added from a current Day Batch using the DAYBATCH_DEL or DAYBATCH_ADD methods.

A second use of ManiPlus is an Interactive Editing Interface. Form level information is displayed in a list and the Data Entry Program can be executed for a form selected from the list. Identification, geographical and form status information are among the items displayed. Form status is set during the Integral Check. An editor typically reviews the list for form status Suspect and Dirty, and modifies those forms using the Data Entry

Program. The interface can display the entire data file or a portion chosen by the editor using Batch Number.

4. Improving Manipula Performance

NASS survey processing using Manipula setups with large data models, large data files and complicated survey design proved to be a time-consuming task. In the most extreme example it took nearly five hours to run one process. To make the survey applications practical to use, reducing processing time was necessary. This has been accomplished using Manipula SETTINGS and various other techniques.

Many Manipula setups don't require copying data from one data file to another or don't involve many fields of a data model. A good example of this is a report. In those setups it is common to read five to ten fields and write a similar number. In these setups CONNECT=NO is used. With this setting, only fields specified in the MANIPULATE paragraph will receive a value. Time is saved during initialization only, so this is most advantageous for large data models and small data files.

A significant amount of time has been saved processing by a Secondary Key field. Earlier we mentioned updating the ProcessSwitch field. ProcessSwitch can have the value of (1) Paper Complete, (2) CATI Complete, (3) Default, (4) Sent. When a selected process is run, ProcessSwitch will be set by a Manipula setup like ProcSw1

(see Figure 3.). A second Manipula setup will process by ProcessSwitch and then halt when forms meeting some specified criteria have been processed. Two Manipula setups are used, but the net effect is reduced processing time. This is possible because the initial Manipula setup (ProcSw1) runs extremely fast because of FILTERS and the second one then reads only a portion of the data file using STARTKEY and HALT.

Figure 3. *ProcSw1 Manipula setup*

```
USES

InstFmt 'AgSurv'

DATAMODEL MarkItFmt

FIELDS

    State :    INTEGER (2)
    ID :    INTEGER (9)
    Tract :    INTEGER (2)
    Subtract :    INTEGER (2)

ENDMODEL

INPUTFILE MarkIt : MarkItFmt ('CODEDATA. IDS', ASCII)

UPDATEFILE CATIDData : InsFmt ('AgSurvC', BLAISE)

FILTER

LFInfo

Managmnt

LINKFIELDS

    LFInfo.State =    MarkIt.State
    LFInfo.ID =    MarkIt.ID
    LFInfo.Tract =    MarkIt.Tract
    LFInfo.Subtract =    MarkIt.Subtract

UPDATEFILE EDITData : InstFmt ('AgSurvE', BLAISE°
```

```

FILTER

LFInfo
Managmnt

LINKFIELDS

    LFInfo.State = Marklt.State
    LFInfo.ID = Marklt.ID
    LFInfo.Tract = Marklt.Tract
    LFInfo.Subtract = Marklt.Subtract

MANIPULATE

IF RESULTOK (CATIDData) THEN
CATIDData.Managmnt.ProcessSwitch := PAPICmpl
WRITE (CATIDData)
ELSEIF RESULTOK (EDITData) THEN
EDITData.Managmnt.ProcessSwitch := PAPICmpl
WRITE (EDITData)
ENDIF

ENDMANIPULATE

```

The FILTER section is used to read only a portion of a data record, rather than the entire record. Filtering can be done for an individual field or for an entire block.

In the FILTER section one or more fields can be specified. Only the specified fields are available in the setup. Increased processing speed can be realized for large data models and large data files. In Figure 3, two blocks are read from the input file. These two blocks account for less than 5 percent of the data record.

STARTKEY is another setting used extensively to save processing time. This is used when only a portion of the data file will be processed. STARTKEY is used with the setting KEY. The processing order of forms is determined by KEY. Processing begins with the first form with a KEY value equal to or greater than the STARTKEY values. When all forms that match the STARTKEY value have been processed the setup is conditioned to HALT. This is most beneficial when a data file with a large number of forms is being read. Applying this technique at NASS has reduced the processing time for some steps 50-75 percent. Figure 4 shows the use of KEY and STARTKEY. A typical setup like this one will read 100-400 forms of a 2,500 form file.

Figure 4. Manipula setup showing STARTKEY

```

SETTINGS
AUTOREAD = NO

USES
InstFmt 'AgSurv'

UPDATEFILE EDITData : InstFmt ('AgSurvE', BLAISE)

SETTINGS
KEY = SECONDARY (BatchNumber)
STARTKEY = (100)

OUTPUTFILE BATCHData = EDITData
('BATCH\AgSurvE', BLAISE)

AUXFIELDS (GLOBAL)
SomeSent : INTEGER (1)

PROLOGUE
EDITData. GET (VAL (PARAMETER))

```

```
MANIPULATE
REPEAT
IF (EDITData. RESULTOK) AND
(EDITData.Managmnt.Batch = VAL (PARAMETER) THEN
... other instructions“
ELSE
HALT
ENDIF
UNTIL EDITData.EOF
ENDMANIPULATE
```

A number of processes use ASCII files as input. Efficiency can be improved when processing an ASCII file against a Blaise file by defining the ASCII file as the first input file as shown in Figure 5. In this way, only the forms in the Blaise file that match the ASCII file are read. In the Figure 5 example, the ASCII file will typically have 100-400 records and the Blaise file will have 1000-3500 forms. This eliminates reading several hundred forms.

Figure 5 : *Defining ASCII file before Blaise file*

```
INPUTFILE MarkIt : MarkItFmt ('CODEDATA.IDS', ASCII)
```

```
UPDATEFILE TrackOut : TrackFmt ('Tracking', BLAISE)
```

5. Conclusion

NASS has found Manipula and ManiPlus to be valuable tools for its CASIC survey systems. Using features to improve processing speed, NASS has been able to process large volumes of data very efficiently.

An Object-Oriented Case Management System for CAPI Surveys

Vesa Kuusela, Anssi Parviainen, Statistics Finland

1. Introduction

Case management is one of the most important parts of a CAPI information system. It is also one of the most difficult ones in case something more than a trivial design is required. It is important since it deals with just what the system is all about, i.e. the collection and transferring data concerning sample points (which may be persons, households, companies etc.). The most important part, of course, is the CAI-software but that is usually a package, such as Blaise, which no longer needs any specific designing. Case management is difficult since, in order to be functional, it involves some fairly complicated manoeuvres with the cases - and since it has to be faultless.

By case management we mean the part of the CAPI information system that includes the distribution of sample points to interviewers, their delivery to interviewers' laptops, transfers of cases from one interviewer to another, collection of the sample points back to the office into a single file, and in panel surveys, the transfer of the sample points to the next wave.

In some papers case management has had a somewhat wider meaning than the one described above (see e.g. Nicholls and Kindell, 1993). However, it is easier to analyse and design the CAPI information system if the case management system is considered an inherent but separate part of the two user interfaces : the interviewer interfaces in interviewers' laptops; and the survey management interface or supervisors' interface at the office. There are also several other tasks besides case management embedded in the user interfaces both in the office system and in the laptop system. User interface provides many activities that support the case management, e.g. telecommunications and launching interviews, etc. We conceive the user interface rather as a platform where case management may be set.

Ideally the case management system should be a transparent part in a transparent interviewer interface which should facilitate the interviewer's work rather than makes it more difficult.

Most of the difficulties in the design of the case management system arises from the fact that the sample data base has to be distributed to interviewers' laptops and later the interviewer files have to be collected

into one single data base again. This is one of the major distinctions compared to the CATI system where only one database is needed.

The distribution of the sampled cases has to be exhaustive and it should be exclusive, i.e. all cases are forwarded to one of the interviewers and no case should be available for two (or more) interviewers. However, it is not absolutely necessary that the data bases in interviewers' laptops are exclusive as long as interviewers know which sample point they should interview. However, many annoyances can be avoided if the distribution of the data base is exclusive.

There is still another aspect which puts extra demands on the CAPI system: Survey organisations have quite different ways of undertaking surveys in practice. In addition, the number of different surveys for which the system must be prepared (and of course, their size and schedule) vary. Some organisations may be set up to only few (maybe only one) surveys and the sample will not change too often. In the other end are those organisations which have several surveys going on with different sampling schemes most of the time and some surveys may have very tight deadlines.

The latter of the two 'opposite' survey environments mentioned above naturally sets more stringent requirements for the design of the system. In that case the system must be very versatile in order to enable the handling many different surveys. The installation of new surveys has to be easy and the completed interviews should be available for further processing as soon as possible. The system must stand for high standards of reliability, however. In any case in a case management system there should not exist a danger of losing data. On the other hand, if the information system is designed for a lively survey organisation it can be easily adopted in less lively ones, too.

Roughly speaking there are two major approaches to design the case management system : a data base -oriented information system and an object-oriented system.

In the data base -oriented approach a file, part of the master data base, containing all sample points for a particular interviewer is sent to his/her laptop. All interviews are collected into one data base. In a trivial case the entire interview data base is sent back to the office, maybe several times. In a more sophisticated system the completed interviews are selected from the data base and they are sent only once.

In the object-oriented system each sample point is presented as a single object, which means that all necessary information to conduct an interview is encapsulated in one file. An object contains a unique identification, the name of the survey (i.e. the method), status, sample data to contact the interviewee or household, additional data of the sample point to be used in the questionnaire, messages, etc. When the object is returned it includes updated data (e.g. status) and the answers to the questionnaire attached. The object may be a message, or a questionnaire, or something else.

The object-oriented approach provides many assets compared to data base -based approach as James Gray (1995) has put forward recently

(see also Rumbaugh, et. al, 1991). For instance the assignment of cases to interviewers becomes fairly straightforward and there is no danger that the same sample point goes to two interviewers ; and case transfers from one interviewer to another may be handled reliably. If one data file is corrupted only one case is lost in the object based system. In a corresponding situation in the data base system all interviews completed so far will probably be lost.

Once the object oriented approach is adopted it also makes many other activities rather simple. For instance, by the same telecommunications session several different objects may be sent. Each object, e.g. message, new questionnaire, a batch job, etc. will call for a specific method. Moreover, in the office system, each case may be processed separately in a so-called 'flow basis' (see Gray and Anderson, 1996). In addition, new objects and new methods are easy to install.

2. The Case Management system

The information system for CAPI surveys has been constructed gradually at Statistics Finland. The first version and the principles of the interviewer interface were described in IBUC'95 (see Kuusela, 1995). The case management system in the first version was, however, rather trivial: all interviewers had a copy of the entire sample file. The system was cumbersome and strained telecommunications. The conversion of surveys with large samples, such as Labour Force Survey, to CAPI made it necessary to design a more sophisticated system.

At Statistics Finland, an interviewer may have up to ten different surveys active simultaneously with quite a different production scheme. Some of the surveys share the same sample and some have a totally different sample. It is fairly usual, for instance, that after completion of an interview of the LFS the interviewer may have to do the interviews of a barometer survey and maybe still a third (very short) interview. At the same time the interviewer may have the household budget survey active and that has a separate sample as most of the continuous surveys do. In addition, there may still be some ad hoc surveys, with separate samples. The case management system in this situation presents a complicated designing task.

At the beginning of the design both approaches were scrutinised but quite soon, however, the object oriented approach was adopted. The reason was that the new system was expected to be both sturdy and flexible. The object based approach was considered to offer more bricks to build the system. In addition, it was fairly easy to implement the system in the existing interviewer interface. The greatest modification in the user interface was the change of the programming language: previously it was programmed by QuickBasic and now by Visual Basic for Dos. The idea is that the next version of the user interface will be done by Visual Basic for Windows. The supervisory system had to be made totally anew.

The basic idea is that a sample point is a single object that carries with it all necessary information needed in an interview. The attributes of an object may include e.g. information needed to contact a person e.g. name,

address, phone number, comments from previous interviews etc.; the name of the questionnaire to be interviewed and/or answers to interviews ; and possible external background data needed in the questionnaire, e.g. register data and/or data from a previous wave.

Each object has a unique identification code, a four digit diary code. A sample point which is to be interviewed for two surveys will yield two objects and hence two different identification codes. Each object makes a separate file and the diary code is included in the name of the file. The attributes of the object are fields in the file and they are separated by special characters. Different classes are indicated by the extension of the file name. Different classes of course have different attributes.

The name of each file carries some information of the case, just like an envelope carries the information to whom the letter should be delivered. For instance, if the file contains an interview the extension of file name indicates the status of the case and its panel rotation number. This naming convention is very useful because it enables the delivery of the cases to right places without opening them. In addition, some statistics for monitoring the field work may be produced on the basis of the file names, as well.

03. Interviewing

The installation of a new survey is 'automatic', i.e. when the interface software finds a specific file (questionnaire object) in the input stream it launches the method which performs the required tasks to initiate the survey. This happens right after telecommunications without a specific prompt. The installation of a new survey is told by a message to the interviewer.

The processing of sample points produces a list of all cases for the survey and external file. In the object there is the external data needed and the Extern -command to perform the indexing of the external file.

The interviewer has to select the survey first, of course. When the survey has been selected the sample of the survey is displayed. The sample points are sorted by the status code of the case.

The status codes and their meanings are as follows

New

The sample point is still intact.

Unfinished

Interview started but not completed

Non-response

The case has been marked as non-responding and copied to the out queue. A non-response case may still be interviewed later.

Interview completed

Treatment of the case was completed and the case has been sent to the central unit to be forwarded to subsequent processing. It is not possible to open the case anymore except by a case specific key which may be attained from supervisors upon request. However, it is possible to browse the answers in the training interview session.

Transfer in

Previously, the case has been assigned to another interviewer who was not able to interview it. No interviewing was done.

Transfer out

Transferred to another interviewer. The sample point has been sent to the central unit to be forwarded to another interviewer.

Returned

Interviewers may return a case by this option if it is unclear what to do. Option may be used only by a permission from supervisors. A returned case must have an attached message.

Special operation

Special operation is meant for exceptional situations. For instance, when the sample point for some reason does not belong to the sample.

All cases which have been sent to an interviewer have status new (or transfer in). When the interviewer closes an interview she/he has to mark the new status of the interview. It can be either unfinished, interview completed or non-response. Rest of the status codes are used in exceptional case movements. If the status was changed either to completed interview, transfer out, returned, or special operation, the case is moved to send queue and the interviewer cannot open it anymore except by a case specific key which may be attained from supervisors upon request. They may still open the non-response cases later on although the case had been sent to the office.

Opening a case for any operation starts with a display of the sample information of the case (see figure below). In the screen the Interviewer can change most of the fields (address, phone number, etc.) and attach a remark to the case. All this information will follow the case as long as the case is active in the system.

+| Survey TEST Case number 1190 contact
information ++|

1190

JANATUINEN

SEPPÖ TAPANI

PELTOMÄENKUJA 144 A 1

HELSINKI 10

0913601092

Options

Remark

+-----+ +-----+
-----+ ||

| Interview |

| Change contact information |

| Open remark |

| Transfer |

| Return to office |

| Special operation |

| Return (Esc) |

+-----+ +-----+
-----+ ||

F1 H099 F2 F3 F4 F5

-|
+-----+
-----| |

|||||

4. Discussion

Ideally the case management in a CAPI system might include roughly the same functionality as in a CATI system. The techniques of to-day does not make it possible, however. Another aspect is that the CAPI and CATI organisations may be so different that it is not reasonable to try to design a similar system.

The data base -oriented information system is a natural choice in CATI system because every work station is uninterruptedly connected to the file server. In a CAPI system there is no direct and continuous connection to the master file system in the office. The data base is distributed to interviewers' laptops and the connection to the office is based on telecommunications at long intervals. The object based system provides a more firm basis for design and more possibilities, at the same time.

The basic idea in the object oriented case management system is that each sample point is stored in a separate file and the interview is stored in a separate file. The object based information system is a challenging design and programming task. Maybe the same functionality could be achieved by the data base -oriented information system but it would probably be a more demanding task. And should the system be as sturdy and flexible it would a far more demanding task.

When each interview is a single object (and a file) the complete and exhaustive distribution is quite straightforward. That is manageable even in a data base based system but dealing with transfers and panels (and transfers in and between panels) are far more easy to accomplish in an object based system and it will be far more reliable.

An object based case management system is easier to install if the sample points of the survey are known in advance. This can be accomplished even in a household survey if the households are identified by a (sampled) person. This makes the use of a unique case identifications fairly easy and hence it is possible to keep track of the movements of an object throughout a panel survey. Assigning a unique identification code to each object when they are brought forth during an interview session is a somewhat more difficult task but manageable for instance by allowing each interviewer to produce identification codes within a specified range.

The object based case management system has been in use for some months now and the basic design has proved to be the right choice. It enables implementation on many new options both in the user interface and in the case management. At the moment, only Blaise 2.4 is used. Next task is to implement also Blaise III in the system so that both Blaise versions may be used simultaneously. Surprisingly, it seems that Blaise 2.x gives better support to object-based system design than Blaise III.

The requirements a case management system has to fulfil are rather common to all CAPI information systems. Therefore, especially the design of a case management system for the interviewer laptops may be

regarded as a general purpose module attached to CAI packages, whereas, the user interfaces may have entirely different requirements in different survey organisations and it is probably more difficult to find a common core for those.

References

Gray J : An Object Based Approach for the Handling of Survey Data. In Kuusela V. (Ed.) : *Essays on Blaise 1995*. Statistics Finland, 1995.

Gray J., Anderson S. : The Data Pipeline - Processing Survey Data on a Flow Basis. In Banks R., Fairgrieve J., Gerrard L., Orchard T., Payne C., Westlake A. (Eds.) : *Survey and Statistical Computing, 1996*. Association for Survey Computing, 1996.

Kuusela V. Interviewer Interface of the CAPI-system of Statistics Finland. In Kuusela V. (ed.): *Essays on Blaise 1995*. Statistics Finland, 1995.

Nicholls W.L., Kindell K.K. : Case management and communications for Computer Assisted Interviewing. *Journal of Official Statistics*. 1993, 9, pp. 623-639.

Rumbaum J., Blaha M., Premerlani W., Eddy F., Lorensen W. : Object-Oriented Modelling and Design. Prentice Hall, 1991

Cati Call Management

Marien Lina, CBS, Nederland

1. Cati call management at Statistics Netherlands

In and around the Blaise system, the CATI-call management system has been developed over the last decades. At Statistics Netherlands the system is a helpful tool to schedule, perform and evaluate telephone interviews. Different units of Statistics Netherlands use the system for various purposes. The division 'Gegevensverzameling' or in English 'Data collection' uses the system to optimise the data collection process when doing phone interviews.

The interviewers at Cotel (Cati unit at Statistics Netherlands) get their phone numbers "served on a disk". There is no need to update paper lists of phone numbers manually. The Cotel management uses the system to schedule interviewers day by day and use the results to analyse how much time an average interview takes (including time consumed by dials that did not result in a response). The survey divisions use the Cati call management system to analyse non-response figures, and to evaluate if the non-response has been measured satisfactory. The engineers use the system to adapt the contents of dial screens and interviews to enhance the production of non-response statistics.

Different ways to look at the Cati management system

	<u>involved units</u>	<u>using the system for</u>
A	Cotel	Phone interviewing
B	Cotel management	Scheduling - process guiding
C	Account management	Analysing interview time
D	Research	Analysing non-

The table shows that there are more ways to use the Cati call management system.

2. Cotel interviewing

Within the Cotel department, the system makes life easy to the interviewers when making appointments and to handle other events when dialling a number.

The dial screen.

When trying to call a respondent, the Cati system shows up with a call screen. This screen shows the phone number. The developers of the interview mostly put more information in the dial screen. For example, the name of a key person in an organisation, the name of an institution, an address or a town name. This information may help the interviewer to check if the dialled number is correct. It may help to find a specific person within an institution. The concrete information varies with the target groups of the interview.

Forms Answer Navigate Window Options Help		-OLD	16:22
[■] Make dial			
Dial menu		Zoom...	Help
(*) Start interview			
() Nobody home			
() Busy		Dial	OK
() Answering service			
() Disconnected		Edit	Cancel
() Fax/Modem			
Questionnaire data			
NieuwTnr	06-932-132		
Contact	Jimmy Goodfellow		
BezTel			
CorTel	BE		
Naam	FOOT LOCKER NETHERLANDS B V		
Plaats	4131 PN VIANEN ZH		
Opmerk	dat is de accountant		
F1-Help		RAPDET	
F10-Menu			

Usually the information in the dial screen contains the following information (if available) :

usual information in dial screens

Person surveys

Phone number

Person or
family name

Address

Town

Remarks

Original phone
number

Firm/Institution
surveys

Phone Number

Name of the
firm/institution

Department

Person to contact

Address

Town

Remarks

Original phone
number

The phone number may be changed by the interviewer. This can not only be done in the dial screen, but also during the interview, when making an appointment. When interviewing institutions it is common practice to allow interviewers to change the name of the addressed person. For each survey, specific information can be added, for example, in a specific survey about financial reports they can add an extra phone number (for example the phone number of an accountant of a firm).

After dialling the number, the interviewer selects one of the options in the upper part of the screen. If the call is answered, the interviewer selects 'Start interview'. Non-response treatment and making appointments is handled in the interview (see below). If nobody answers the call, the interviewer does not start the interview but selects one of the following options in the call screen :

- Nobody home (when no one picks up the phone),
- Busy (when the line is used by another call),
- Answering service (when connected to an answering service),
- Disconnected (after a specific message of the telephone company),

- Fax or modem sound (after squeezed arias and digital ballads).

If one of these options is selected, the dial screen for the next phone number appears and the system updates the administration of this form, defining at what time the same number should be dialed again, or, that it does not have to be dialed again.

Making appointments.

At the beginning of the interview period, the Cati-call management system creates and constantly updates a history file. In this file, each dial is a record.

After dialling a number of a possible respondent, the interview enters the interview. If the addressed person is not available for the interview at that time, the interviewer can make an appointment. To do so, he or she enters the 'appointment block' which by default is available in each Cati survey at Cotel, using the standard appointment screen of the Blaise system :

An appointment may vary from a concrete time (or part of day) and date to an appointment without any time or date (just should be called again). The selected options are updated in the system.

After specifying the appointment, in most of the surveys there is room to specify additional information, such as the name of the firm and the person that should be addressed for the interview, his or her personal phone number and remarks.

Forms Answer Navigate Window Options Help		-OLD δ 16:23
afspraak: 1/1		
<p>>>ENQ: hier kunt U eventueel een naam of een toestel-nr intikken waarnaar gevraagd moet worden bij de afspraak. <<</p> <p>Indien geen opmerking : Druk <ENTER></p>		
Plaats	4131 PN VIANEN ZH	
Naam	FOOT LOCKER NETHERLANDS	
Contact	Jimmy Goodfellow	
NieuwTnr	06-932-132	
opmerk	dat is de accountant	
F1-Help Alt : X-Quit Alt-F1-Previous help F3-Close		RAPDET

The additional information is useful, for example in this survey, in which financial data are gathered. The addressed firm name may refer to an accountant. The name of the accountant may be added, his phone number and a remark, in this case possibly 'it's the accountant'. The fields may also be left empty.

After finishing this appointment block, the appointment have been stored in the data and when the time is right the form will automatically show up. The added information about the firm will be displayed in the dial screen (see figure 1).

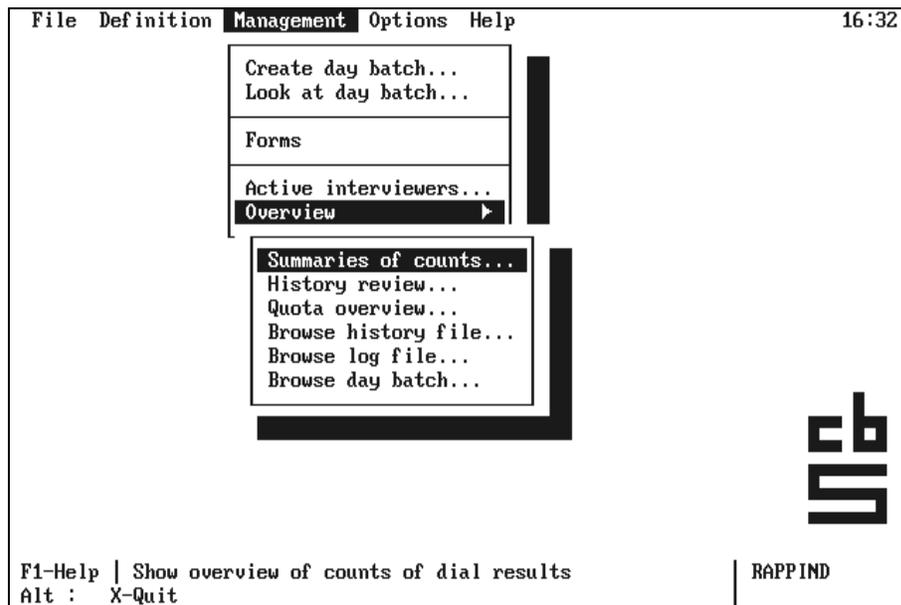
Scoring non-response.

If, for whatever the reason is, an interviewee does not (any longer) participate in the interview, the non-response block will be activated by the interviewer. The interviewer will determine the reason for non-response. The non-response block shows a number of categories, such as 'refuses to corporate', 'language problem'. These categories are not the same in each survey. Apart from that, there is room for the interviewer to make a remark. After the non-response has been scored, it will automatically be updated in the call-management system and the person or firm will not be called again. The dial result, stored in the data is 'non-response'.

The system keeps track of the number of call-attempts in the *Cati call management system*. Additional information about non-response and appointments may be available in the *appointment block* and the *non-response block*, which are integrated in the interview. During the interview sessions, the system also updates a *history file* in which each dial result is a record. These components are the main ingredients for the Cotel manager and the analysts.

3. Cotel management

When the Cotel manager has started up his computer, the first thing to do is to check the results of the past day, and to prepare the day portion for today. This is done for every research separately. After that the over the current researches interviewers can be scheduled for the day. After that the manager is stand by when the interviewers are calling. It may be that a new research has to be installed for Cotel. In that case he checks whether the right options for Cotel are installed.



Depending on the realised response and the target response the Cotel manager concludes how many interviewers should be activated that day. The crucial information is available in the Cati management system. After selecting the "overview" option in the management menu, the following information appears :

Dial results		Appointments	
No call	622	No appointment	1036
No answer	311	No preference	32
Busy	7	Date and time	12
Appointment	68	Period and day part	31
Answering	105	Weekday and daypart	0
Not yet done	1113	Only period	2
Response	797	Only week day	0
Non response	82	Only day part	0
Disconnected	1		
Others	4		
Done	884		

F1-Help
 F3-Open F10-Menu

A97_EURO

The manager prints these overviews for each survey, checks the planning and, makes up today's schedule for each survey and, if needed, activates the right number of interviewers.

4. Account management

When a client (a research Division of Statistics Netherlands, or sometimes an external research lab) wants to use the facilities of Cotel at Statistics Netherlands, their will have to be made a calculation about the size of the job in terms of interview time. One reason is to make a adequate planning. The available number of trained interviewers must be appropriate to handle the job in a certain amount of time. The other reason is to calculate how much the data collection will eat from the clients budget. At first the impact on the interviewers capacity of a survey is calculated with an available method. Different question types take a different amount of time. If the number of questions is known, then the calculation can be done. Later on, when the interview is started, it may come out the interview times are more or less.

At Statistics Netherlands, the duration of one interview is available in two ways. First, the time the interviewer really is interviewing is measured in the questionnaire. Second, for each form, the overall time the interviewer spends in dialling a number, making appointments, scoring a non-response, repeated dialling and other events is available in the history file.

During the interview period, the Cati-call management system creates and constantly updates this history file. In this file, each dial is a record. For each dial, the key of an interview form, the used time and the dial result are available. Hence, for each form the number of dials and the overall consumed time is available.

For the account managers the second way of measuring time is important to answer the question: how much time does on average one interview

take. This is relevant for evaluating and updating the calculation method, getting a better picture of the amount of resources, required for the job.

5. Analysing non-response and the Cati process

If the data entry part of a research has been completed, all dial results are available in the history file and the Cati call management block of the data file. The data enable the researchers to evaluate non-response. If they use the standard Blaise options, extended with a self-defined Non-response block, the data are available. At the end of a data collection period, there are more or less two administrations in the data.

The first is the Cati call management block. In this block, the dial results have been stored, including the scores for 'Nobody home', 'Busy', 'Answering service', 'Disconnected' or 'Fax or modem sound'.

The second is the Non-response block, designed by developers of the survey. In the non-response block, the interviewer can add a code for the reason of the non-response. The non-response block is part of the interview, and can only be activated if an interviewer has started the interview. Hence, if a 'busy' dial result was the last attempt, this cannot be present in the non-response block. However, the researchers often want these cases listed in the non-response statistics.

When analysing non-response, the researchers combines the two administrations to get non-response statistics. If we know that, literally none of the Cati-surveys at Statistics Netherlands have a similar non-response block, there is no standard tool to integrate these figures into a standard non-response-statistic. The expectations for the future are to pay some more attention to this topic of non-response statistics and to develop tools to increase the uniformity of non-response statistics, as used by the researchers in publications and by the data collection department to make trends in the non-response over all surveys more comparable.

6. New options for Cati-call management in Manipula

New options in Blaise III and Manipulus make it possible to increase the amount of control over the interviewing process.

The following situation is not implemented at Statistics Netherlands, but nothing more than an idea for the future. So, what is this idea ? As already mentioned, the system creates and updates history files with dial results during the interview period. If the right tools are there, Manipulus enables you to read the history file after each call, and to control tools to signal if something is out of the ordinary.

Data can be read from the history file. The point here is that Manipulus (using additional tools) can update the day-batch. This can be very helpful when the desired response has not been reached and the number of interviews in the day batch should be increased (or vice versa).

Maniplus can control the tools that can signal the Cotel manager the status quo concerning the reached response. He or she can adapt his planning and take action earlier (for example, changing the schedule for the interviewers). Or, still more advanced, a tool (the algorithm is still unknown) may decide that the number of interview in the day portion should be increased or decreased, and update the day-batch.

If the right tools are there, Maniplus enables you to increase the amount of control over all surveys at once. For example, you could have a quick look at the response figures for all surveys at once, while, in the usual Blaise Overview, you can only look at one survey at the time.

At this moment time these options are only available in mind. Maniplus can help you to realise this type of tools.

CapiPlus, un système intégré pour la réalisation des enquêtes auprès des ménages

Gilles Luciani, INSEE, France

1. Introduction

Le système CapiPlus développé à l'Insee a pour objectif de proposer aux statisticiens une architecture informatique capable d'accueillir les enquêtes CAPI auprès des ménages en réduisant les coûts de développement informatique à ceux de la programmation du questionnaire de collecte et du programme d'apurement, qui sont spécifiques à l'enquête. Ces programmes, une fois testés sur un échantillon de quelques centaines de ménages, doivent s'intégrer dans le système CapiPlus sans qu'il soit nécessaire de réaliser des développements complémentaires. La possibilité de cette intégration repose sur le respect d'un certain nombre de standards communs à toutes les enquêtes :

- une architecture du programme du questionnaire,
- des blocs de programme de questionnement pour la composition du ménage (le "tronc commun"),
- des variables de gestion,
- des interfaces utilisateurs,
- un carnet de tournée,
- une ergonomie,
- un langage.

J'ai présenté rapidement à la dernière conférence des utilisateurs de Blaise à Helsinki en septembre 1994 ces principales caractéristiques du système CapiPlus, qui était à l'époque en cours de réalisation.

Ce système a depuis été mis en oeuvre pour deux enquêtes :

- « Logement », enquête réalisée auprès de 45000 ménages de novembre 1996 à février 1997,
- « Jeunes et Carrières » en mars et avril 1997 auprès de 60 000 personnes.

La prochaine enquête utilisant CapiPlus sera l'enquête " Patrimoine ", en production en octobre 1997.

2. Résumé

On décrit dans un premier temps l'architecture générale du système CapiPlus : les logiciels utilisés, les matériels, la circulation des données entre les différents sites informatiques.

Il est ensuite expliqué au travers de quelques exemples comment l'utilisation de Blaise dans CapiPlus simplifie la *capisation* d'une enquête, tout en maximisant les deux principaux avantages attendus de la technique CAI :

« *faire plus vite..* » dans la réalisation des programmes de collecte et d'apurement, dans la production de fichiers propres,

« *...et mieux* » dans la qualité des données obtenues à l'issue de l'apurement et dans le confort des utilisateurs du système.

3. L'architecture générale

3.1. Les logiciels

CapiPlus utilise actuellement sur les logiciels et outils suivants :

- Le logiciel Blaise 3 (DEP, Manipula, Maniplus, Caméléon) pour la réalisation des questionnaires, les programmes de traitement des données en batch, les interfaces utilisateurs, la production des dictionnaires SAS. Le basculement de la version beta K de Blaise (juin 1996) utilisée pour la production des enquêtes " Logement " et " Jeunes et Carrières " dans la version 1.15 a été réalisé en avril 1997.
- Le logiciel Arbitr (Société Tangram - USA) pour les transmissions des données entre les postes utilisateurs et le site central.
- Des programmes spécifiques écrits en TurboPascal (Société Borland) pour la tabulation des données sur les postes de travail et quelques traitements spécifiques.
- L'utilitaire PkZip (Société PkWare) pour la compression-décompression des données sur les postes de travail.

A l'issue de la collecte, puis de l'apurement des données, le fichier de l'enquête est traité principalement avec le logiciel SAS.

L'applicatif fonctionne sous Dos 5 ou Dos 6, selon les postes.

Les évolutions de cette architecture prévues à court terme sont l'utilisation d'Abacus en batch pour la tabulation des données, l'utilisation de PkUnzip sur site central (IBM, MVS), et l'intégration de CapiPlus sous Windows 3.11 dans une fenêtre Dos. A moyen terme (1998), nous envisageons le remplacement du logiciel arbitr par un logiciel de transmission sous Windows, l'utilisation de Blaise sous Windows.

3.2. Les matériels

CapiPlus est un système informatique réparti sur quatre types de sites informatiques :

Les 900 micros de collecte des enquêteurs : Le parc des micros qui équipent les enquêteurs se répartit actuellement entre des Toshiba T1850 (486 SX RAM4+4 HD80 monochrome) et des Hewlett-Packard Omnibook 5000 CT et Omnibook 800CS (Pentium 100 RAM 16 HD800 couleur). L'objectif est d'homogénéiser dès cette année le parc avec uniquement des Pentium HP.

Une trentaine de postes de travail dans les directions régionales répartis en 18 serveurs.

Les postes de travail du concepteur sur un réseau local unique à la direction générale.

Le suivi de la collecte et l'apurement des données dans les 18 directions régionales qui encadrent les enquêteurs utilise les serveurs sur lesquels sont connectés des micros de bureau de différents types (486 DX et Pentium, RAM16), utilisés pour différentes applications. Toutes les données et programmes Capi sont sur le disque dur du serveur. Les accès concurrents sur les données sont gérés par Blaise. La CPU du serveur n'est pas utilisée.

Enfin le site central par lequel transitent les données entre les enquêteurs et les directions régionales, et sur lequel sont regroupées les données à la fin de l'enquête.

3.3. a circulation des données

On détaille ci-dessous les différentes étapes du cheminement des données depuis la constitution de l'échantillon de l'enquête jusqu'à la restitution des données finales sur le site central.

Avant la collecte :

1. Livraison des échantillons régionaux sur le site central.
2. Intégration de l'enquête sur le serveur régional : importation des programmes.
3. Importation des programmes du poste de collecte, et production de la disquette D1 d'installation de l'enquête sur les postes de collecte.
4. Importation de l'échantillon sur le serveur local en région.
5. Attribution des fiches-adresse (FA) par enquêteur : un échantillon régional avec pour chaque FA un n° d'enquêteur est ainsi constitué.
6. Production de la base des enquêteurs propre à l'enquête. Production de la disquette D2 contenant la base des enquêteurs et permettant la personnalisation des postes de collecte.
7. Eclatement de l'échantillon régional en sous-échantillons " enquêteur ".
8. Conversion de chaque échantillon dans le datamodel de collecte, et production de la disquette D3 des échantillons.
9. Chargement des postes de collecte :
10. - Installation des programmes de l'enquête avec la disquette D1,
11. - Personnalisation du micro de l'enquêteur avec la disquette D2,
12. - Chargement de l'échantillon avec la disquette D3.

Pendant la collecte :

13. Collecte des données par l'enquêteur.
14. Transmission quotidienne via Transpac des questionnaires validés et zippés sur le site central, ou par envoi postal de disquettes pour les enquêteur qui n'ont pas accès à Transpac (3%).
15. Importation quotidienne de ces données sur le serveur de la DR, dézippage et conversion dans le datamodel d'apurement DR.
16. Apurement des données par la DR.
17. Conversion des données dans le datamodel d'apurement Concepteur.
18. Conversion en ASCII et exportation quotidienne des données apurées sur le site central.
19. Importation quotidienne des questionnaires par le concepteur des questionnaires encore erronés.
20. Apurement sur le poste concepteur à la direction générale.
21. Conversion en ASCII et exportation quotidienne des questionnaires apurés par le concepteur sur le site central.

4. L'installation d'une enquête sur les postes de travail

Cette opération est réalisée avec des programmes indépendants de l'enquête. Le gestionnaire indique le code de l'enquête à installer, par exemple PAT_97 pour l'enquête Patrimoine 1997, et l'enregistrement de l'enquête est importé du site central, le fichier des enquêtes disponibles est mis à jour par un manipula ASCII to Blaise.

L'installation de l'enquête, et des éventuelles mises à jour se font par un choix de l'enquêtes grâce à un LOOKUP sur le fichier des enquêtes, puis par l'importation des programmes et fichiers.

Dans l'interface du poste du gestionnaire, le menu primaire regroupe les fonctions communes à toutes les enquêtes, et il y a un menu secondaire pour chaque enquête. Chacun de ces menus sont programmés en Manipus. Le choix du menu de l'enquête que l'on souhaite lancer se fait à toujours par LOOKUP sur le fichier des enquêtes : le code de l'enquête est utilisé comme paramétrage de l'appel de tous les programmes.

Il n'y a pas de limite au nombre d'enquêtes que l'on peut installer sur les postes de travail, sauf celle du disque dur du micro de collecte, ou du serveur.

Le datamodel du fichier des enquêtes :

```
DATAMODEL Enquete
PRIMARY CodEnq
TYPE
    EnqType=(      T1 "Enquête à un seul lot de logements",
                   T2 "Enquête par aires",
                   T3 "Enquête à vagues",
                   T4 "Panels")
FIELDS
    CodEnq          "Code de l'enquête"          : STRING[6]
    NomEnq          "Nom de l'enquête"          : STRING[40]
    DateDeb        "Date de début de l'enquête" : DATETIME
    TypEnq         "Type d'enquête"            : EnqType
    NbQuest        "Nombre de sous-questionnaires" : 1..99
```

L'appel du menu de l'enquête dans le maniplus primaire des postes de travail, en passant les paramètres Code de l'enquête (CodEnq) et numéro de DR (NumDR) :

```
PROCESS AppliDR "Menu primaire CAPI+ - Poste DR"
USES
    Menu          'I:\CAPI\PROGGEN\BLAISE\menu'
    Enquete       'I:\CAPI\PROGGEN\BLAISE\enquete'
INPUTFILE MenuP: Menu ('I:\CAPI\PROGGEN\BLAISE\menudr.asc', ASCII)
                   SETTINGS SEPARATOR='/' CONNECT = NO
INPUTFILE ListEnquete : Enquete(BLAISE3)
                   SETTINGS OPEN=NO ACCESS = SHARED
(...)
DIALOG LookupEnq 'Choix d'une enquête'
    SIZE = (74,19)
    LOOKUP ListEnquete POSITION = (2,1) SIZE = (69,8)
    BUTTON Bouton CAPTION = '~O~K' VALUE = Ok
    BUTTON Bouton CAPTION = '~A~nnuler' VALUE = Nul
(...)
MANIPULATE
(...)
ListEnquete.OPEN('\CAPI\DATAGEN\PARAM\enquete')
    LookupEnq
    CASE Bouton OF
    Ok: CodEnq := ListEnquete.Code
        NomEnquete := ListEnquete.Nom
        DISPLAY ('Vous avez choisi l'enquête: ' + NomEnquete) PAUSE
    ENDCASE
(...)
ListEnquete.CLOSE
ProgN := 'I:\CAPI\PROGGEN\BLAISE\'+CodEnq+'DR /P'+CodEnq+';' + NumDR
Reslt := CALL(ProgN)
```

5. Le menu de l'enquête

Le menu de l'enquête est un setup maniplus qui diffère selon le type de l'enquête. Pour les enquêtes du même type, ces setups ne diffèrent que par les instructions USES qui appellent les datamodels propres aux questionnaires des enquêtes. En effet, le nom du datamodel dans l'instruction USES ne peut être introduit comme paramètre.

Le début du maniplus de l'enquête Logement, et les instructions de lancement du programme d'interview (ordre EDIT) :

```
PROCESS MenuEnquêteur1 "Enquête de type 1 "  
USES  
MenuEnq1      'I:\CAPI\PROGGEN\BLAISE\Menu'  
MetaCollect   'I:\CAPI\LOGT96\PROG\BLAISE\Logt96'  
  
INPUTFILE MenuF : MenuEnq1 ('MENUENQ1.ASC',ASCII)  
                SETTINGS SEPARATOR = '/' CONNECT=NO  
UPDATEFILE     Flux : MetaCollect(BLAISE)  
                SETTINGS OPEN = NO ACCESS = SHARED  
INPUTFILE      FStat : MetaStat(ASCII)  
                SETTINGS OPEN=NO CONNECT = NO          (...)  
MANIPULATE  
  CodEnq := PARAMETER(1) NumDr := PARAMETER(2)  
  PathN:= 'I:\CAPI\'+'CodEnq+'\DATA\  
REPEAT Reslt:= MenuF.MENU('V3.2 04/12/96')  
  CASE Programme OF          (...)  
'Collecte':                  (...)  
  FileN := PathN+CodEnq  
  Flux.OPEN(FileN) Reslt:=Flux.EDIT(FileN) Flux.CLOSE (...)
```

Insistons sur le fait que la seule chose qui diffère entre les programmes des menus des enquêtes de même type est l'instruction USES dans laquelle le nom du fichier du datamodel est en *dur*.

Par conséquent, une fois le datamodel de l'enquête au point, on comprend que son intégration dans un système stabilisé nécessite un minimum d'investissement pour le maintenicien. Il faut tout de même procéder à une compilation de tous les programmes et bien sûr s'assurer par des tests de leur fonctionnement correct.

6. Le chargement de l'échantillon

Dans CapiPlus, le fichier de l'échantillon doit respecter un dessin précis. Dans ce cas, le manipula de conversion de l'échantillon ASCII en échantillon au format Blaise dans la datamodel de l'enquête est toujours le même, avec la même réserve que ci-dessus : le nom du datamodel dans l'instruction USES doit être modifié, et le manipula recompilé.

7. Une qualité accrue dans des délais courts

Dans CapiPlus, les données de collecte sont contrôlées à trois niveaux successifs :

Contrôles de niveau 1 sur le poste de collecte

Le datamodel comprend un grand nombre de contrôles (CHECK et SIGNAL) assurant à la saisie des données correctes. De plus, au cours et à la fin du questionnaire; l'enquêteur valide ou non les parties qu'il vient de saisir. L'état du questionnaire ne pourra être à Bon que si toutes les validations ont été faites, et tous les avertissements supprimés. Quand l'enquêteur décide d'envoyer les questionnaires en DR, un manipula sélectionne tous les questionnaires bons, qui sont zippés dans un fichier. Mais cette technique est insuffisante pour garantir la correction parfaite des données au niveau 1, car après validation, et avant l'envoi,

l'enquêteur peut ouvrir un questionnaire pour le relire (droit au remords), et effacer par erreur des réponses obligatoires, ou rendre incohérent le questionnaire en modifiant des réponses : il suffit pour cela de faire ces modifications en remontant dans le questionnaire, et de quitter par AltX, la sauvegarde étant automatique. C'est pourquoi le manipula de sélection des questionnaires à l'état Bon refait les contrôles grâce à l'instruction CHEKRULES. En cas d'erreurs résiduelles, le questionnaire n'est pas sélectionné pour l'envoi, l'état et la validation finale remis à blanc. Un message prévient l'enquêteur de la dévalidation.

Contrôles de niveau 2 sur le poste de d'apurement en DR

Lorsque les questionnaires reçus en DR sont intégrés dans le fichier d'apurement, ils sont convertis par un manipula dans un autre datamodel (niveau 2), dans lequel on a supprimé des contrôles de niveau 1, rendu plus stricts certains contrôles (fourchettes plus étroites, cohérences plus fines,...), et ajouté de nouveaux contrôles. Afin de ne pas réintroduire à l'apurement des erreurs de niveau 1, il faut être très rigoureux dans la suppression des contrôles de niveau 1, et mettre éventuellement en .SHOW les variables sur lesquelles aucun nouveau contrôle n'est effectué.

Le manipula de conversion dans le niveau 2 fait passer automatiquement à l'état Bon les questionnaires qui n'ont pas d'erreurs de niveau 2.

Le programme produit un fichier des remarques et des questions ouvertes des enquêteurs pour le lot des questionnaires reçus. La production du fichier des remarques oblige à une conversion en ASCII de la totalité du questionnaire, la possibilité n'étant pas offerte aujourd'hui de ne produire que les remarques dans un fichier texte. La consultation de ce fichier permet de se pencher plus aisément sur les questionnaires présentant des problèmes. Les enquêteurs ont pour consigne de motiver systématiquement la suppression d'un avertissement par une remarque.

Le nombre d'avertissements supprimés à la collecte est affiché (SUPPRESSCOUNT). Cet indicateur permet d'attirer l'attention du gestionnaire sur les questionnaires douteux.

Une variable « Arelire » est renseignée à Oui si le nombre d'erreur de niveau 2 est positif.

Manipula EnqToDr

```
USES
  MetaCollect 'I:\CAPI\LOGT96\PROG\BLAISE\Logt96'
  MetaADR     'I:\CAPI\LOGT96\PROG\BLAISE\Logt96F'

INPUTFILE  Collecte : MetaCollect
            ('@I:\CAPI\LOGT96\DATA\ziplist.asc', BLAISE)
OUTPUTFILE ApurtDR   : MetaADR
            ('I:\CAPI\LOGT96\DATA\entree'      , BLAISE)
OUTPUTFILE Rmk      = Collecte ('I:\CAPI\LOGT96\DATA\Remark.asc',
ASCII)
AUXFIELDS  Er1 : INTEGER[4]   Er2 : INTEGER[5]

MANIPULATE
  ApurtDR.MSC := Collecte.SUPPRESSCOUNT
  Er1 := ApurtDR.MSC
  IF Er1 > 0 THEN  ApurtDR.ErrColV1 := Oui
                  ELSE            ApurtDR.ErrColV1 := Non
  ENDIF
  ApurtDR.CHECKRULES
  Er2 := ApurtDR.ERRORCOUNT(SOFT)
  IF ER2 > 0 THEN  ApurtDR.ErrColV2 := Oui
                  ApurtDR.ARelire  := Oui
                  ApurtDR.Etat     := EMPTY
                  ELSE            ApurtDR.ErrColV2 := Non
                  ApurtDR.VALidDR  := Oui
                  ApurtDR.Arelire  := Non
                  ApurtDR.Etat     := B
  ENDIF
  ApurtDR.Comment.Enq := Collecte.Commentaire
  IF Collecte.Commentaire.Comgen <> Empty THEN
    ApurtDR.RmqEnq := '*'
  ENDIF
  ApurtDR.WRITE
  Rmk.WRITE
```

Quand la DR souhaite envoyer les questionnaires apurés sur le site central, les questionnaires sont à nouveau contrôlés dans le manipula de sélection des questionnaires bons, et éventuellement dévalidés, ils sont convertis dans un datamodel de niveau 3, dans lequel sont programmés des contrôles sophistiqués qui sont traités centralement par des agents très qualifiés appartenant à l'équipe du concepteur. Le manipula place dans un fichier ASCII tous les questionnaires apurés par la DR, et dans un fichier Blaise les questionnaires qui relèvent d'un apurement de niveau 3.

On notera dans le programme l'utilisation de la *wildcard* '@I:\CAPI\LOGT96\DATA\ziplist.asc' dans l'instruction INPUTFILE. Cette façon de faire résout le problème de la concaténation des fichiers de collecte des enquêteurs. A leur importation, un programme Pascal les dézippe chacun dans un répertoire particulier, et crée simultanément le fichier *ziplist.asc* dans lequel se trouve la liste des fichiers importés et dézippés, avec leur chemin complet.

Niveau 3 sur le poste concepteur

Lorsque les questionnaires arrivent sur le poste concepteur, ils ont déjà été convertis dans le datamodel de niveau 3 en sortie d'apurement. Les dernières erreurs sur ces questionnaires *difficiles* sont examinées et corrigées.

Un flux rapide

Dans le cas idéal, un questionnaire gravit les trois échelons de ces contrôles en trois jours :

Envoi sur site central par l'enquêteur le soir du jour J de collecte, importation des questionnaires par le programme de transfert qui se déclenche automatiquement la nuit (on évite les encombrements diurnes des transmissions et on gagne le temps de traitement du manipula de conversion),

Apurement en DR le jour J+1 des questionnaires reçus dans la nuit,

Transfert nocturne des questionnaires apurés sur le site central, une partie étant livrée au concepteur pour les corrections de niveau 3,

Apurement le jour J+2 par le concepteur,

Envoi le jour J+3 du questionnaire définitif sur le site central.

La rapidité de la circulation des données permet à la DR de se retourner vers l'enquêteur, ou au concepteur vers la DR, en cas de doute sur certains questionnaires, ceux-ci étant encore présents dans la mémoire de l'agent.

En terme de volumes, le bilan des contrôles pour l'enquête Logement est :

	Nombre de questionnaires en %
Sans erreurs à la collecte (niveau 1)	100
Aucun avertissement supprimé à la collecte	60
Aucune erreur en entrée d'apurement DR (niveau 2)	62
Aucune erreur en sortie DR (niveau 3)	95

Le nombre d'avertissements supprimés à la collecte est en moyenne de 0,6 par questionnaire.

An integrated household survey in the UK: The Role of Blaise III in an Experimental Development by the Office For National Statistics

Tony Manners and Kirsty Deacon, Social Survey Division, Office for National Statistics, UK

1. Introduction

This paper examines the role of Blaise III in an experimental development by Social Survey Division (SSD). In 1995/96, SSD developed and tested a survey design which integrated 4 separately designed UK continuous household surveys :

- the ONS's *Family Expenditure Survey* (FES),
- the Department of Social Security's *Family Resources Survey* (FRS),
- the ONS's *General Household Survey* (GHS),
- the Department of the Environment's *Survey of English Housing* (SEH).

The project to investigate the new sample design, and its implications for instrument and fieldwork design, was called the Integrated Household Survey (IHS). When this paper refers to the 4 surveys as components of the IHS, it will call them *sub-surveys*. The sub-surveys ranged in annual responding sample size from the 7,000 households of the FES to the FRS's 25,000 households per year, and summed to 63,000 households for the whole IHS.

The main reasons for considering an integrated design were :

- improvement in the precision of estimates without increasing costs,
- the potential for economies of scale in project and fieldwork management,
- identification - through the variables which were common to the 4 surveys - of larger subsamples of rare populations for follow-up in separate enquiries,
- the potential for weighting subsurvey-specific variables (i.e. measured in part of the integrated sample) by the common variables for the whole IHS sample,

- the potential for surveying continuously some additional topics which at the time were carried out in an ad hoc or repeated way.

Similar projects are under way in a small number of other countries, and there is growing interest internationally. One of the more advanced projects is at Statistics Netherlands, which was visited by SSD to exchange views.

The major issues facing the IHS were the cost-effectiveness of the sample design and the technical feasibility of the fieldwork pattern which it demanded, which implied a heavy training load on interviewers.

The technical feasibility of the IHS was demonstrated in two field trials, discussed in a later section of this paper. It was also shown to be more cost-effective overall than conducting the 4 surveys with their existing separate designs. However, savings varied considerably by survey and by variable. For some key variables, the savings for some of the customer departments were insufficient to justify accepting a range of financial, methodological and strategic risks which were involved in a major change to their survey systems. As a result, the IHS project moved away from integrated sampling and data collection in mid-1996. Work began to investigate the scope for building on harmonisation of concepts and questions to provide a basis for pooling data from separate sources. Nevertheless, much was learned from the IHS project about Blaise III and the improvements it can bring to instrument design and fieldwork, and it is the purpose of this paper to discuss these lessons.

The IHS design made challenging demands on interviewers and therefore on the software for data collection. Social Survey Division has used Blaise in production since 1990. By 1995, virtually all its fieldwork was carried out in CAPI and CATI, mainly using Blaise 2.5, but also beginning to move to Blaise III with preparations for converting the Omnibus and Labour Force Surveys. Conversion of the IHS sub-surveys from Blaise 2.5 to Blaise III was seen as playing a vital part in their successful integration.

2. The IHS design

The form of integration for the IHS sub-surveys involved :

- a single sample design which would lead to improvements in effective sample size for each of the surveys through 'declustering' their current designs. This would be achieved by spreading sampling units for any one subsurvey amongst a large number of clusters; any one cluster which formerly contained sampling units of only one subsurvey, would now have about a quarter of the number of sampling units for that subsurvey. The main benefits of such a design would be the improvements gained in the precision of estimates without increasing costs. Survey sponsors could take the benefits as savings, if they chose,
- as a result of this design, interpenetration of the sub-surveys in primary sampling units which each formed an interviewer workload (i.e.

interviewers were given a *mixed workload* comprising addresses for all 4 sub-surveys),

- a common definition of the household response unit, and of eligible households; these definitions had previously varied between the surveys,
- a set of common questions for all households, covering basic classificatory variables, comprising the primary set of harmonised questions developed in a separate project (see below),
- questions specific to individual IHS sub-surveys,
- technical means to combine questions common to all 4 sub-surveys (or shared by 2 or more of them) with those specific to sub-surveys, in an interview which flowed acceptably for respondents.

Some of the main technical problems to be solved for this design concerned sampling issues such as provision of an optimal set of stratification factors for the 4 sub-surveys. These issues have been reported elsewhere (Elliot and Barton, 1997).

For the purposes of this paper, the crucial point about the sample design was that it required interviewer workloads to be made up of addresses from all 4 sub-surveys. This meant that SSD had to develop training and support systems to ensure that every interviewer could handle the demands of all 4 sub-surveys which would make up each workload. SSD's permanent force of interviewers already contained many who had built up experience of 3 or 4 of the surveys - one at a time - over several years. However, the design of the IHS meant that it would no longer be possible for new interviewers to gain experience of the surveys one at a time: they would have to be fully capable of handling all 4 from the start. The costs of special training measures like dummy workloads comprising a single survey, or allocating new interviewers to the addresses for one survey in several workloads, ruled them out. SSD's solution to this problem relied on facilities provided by Blaise III, and is described in a later section of this paper.

Blaise III was also essential to meeting the other requirements of the design, which were concerned with fast, cheap and reliable production and maintenance of the survey instruments and outputs to customers. Cost effective survey management required integration of the sub-surveys at a deeper level than the sample design, important as that was. Surveys which are commissioned at different times by different customers to meet different needs tend to develop their own idiosyncratic methods, evolved to some degree in ignorance of developments on other survey projects: this common feature of organisations has been called "stovepipe" development (Priest, 1996). For the IHS, it was necessary to find methods which could be applied uniformly to all the sub-surveys (to realise economies of scale) and yet be an improvement on the methods previously used for any one of them. These methods required a more flexible software system than Blaise 2.5.

3. Harmonisation of questions

Before turning to consider how the new demands of the IHS were met by Blaise III, we should note the contribution towards a solution made by another development in UK government social surveys which was taking place at the same time as the IHS project: the project to harmonise concepts, definitions, classifications and questions. This project arose from the increasing demand from users for more coherence in social statistics produced by government. One of the purposes of the creation of the new Office for National Statistics in 1995 was to meet such demands through establishing and maintaining "a central database of key economic and social statistics, drawn from the whole range of statistics produced by Government, and produced to common classifications, definitions and standards". ONS published *Harmonised Questions for Government Social Surveys* at the end of 1995, and an expanded and updated version a year later (GSS, 1996).

Wherever any harmonised questions applied to the IHS sub-surveys, Blaise III source code needed to be written only once. Most such questions were from the *primary set* of harmonised questions which were intended to apply to all government social surveys, covering the demographic details and economic activity of household members and household tenure. There were also some questions from the *secondary set* of harmonised questions, which applied to particular groups of surveys. For example, harmonised questions on housing costs and benefits applied to all 4 IHS sub-surveys (but not to some other government surveys), while harmonised questions on social security benefits applied to only 3 of the 4 IHS sub-surveys, and some very detailed income questions applied only to the FES and FRS.

The harmonisation project had explicitly rejected the idea that harmonised questions should necessarily form a single bloc of questions within a questionnaire. It did not require that topics should be asked in the same order, and even within topics it was permissible to insert additional, more detailed questions if a survey needed them. Part of the IHS project involved designing the optimal route for 4 surveys, which varied widely in the detail that they each sought on particular topics, through all the harmonised and survey-specific questions. Appendix B illustrates through part of the IHS datamodel level RULES paragraph how this was achieved.

4. Development of a flexible software system

The 4 sub-surveys each by themselves presented severe challenges to Blaise 2.5 (Manners and Bennett,1992; Costigan and Sjodin,1992; Manners, Cheesbrough and Diamond, 1993). Each was constrained by system limits such as those on numbers of questions, checks and blocks, and each had circumvented these constraints by using multiple questionnaires linked for a sampling unit by a shell written in non-Blaise software. SSD's case management systems could have handled an IHS which left the 4 sub-surveys as they were in Blaise 2.5, but Blaise III offered some vital opportunities to improve cost effectiveness :

- simplification, by removing constraints which required complicated circumvention, resulting in :

- ◇ source code which was easier to understand, change and maintain without error,
 - ◇ removal of the reliance on programmers for complex elements of questionnaires, and on non-Blaise software,
 - ◇ faster, cheaper, more accurate development, based on simple structures and models and less code,
- increased functionality, allowing :
 - ◇ development of better tools for interviewing - some built-in, like *parallel fields* (see below) and others adapted, like use of the facility for switching languages to provide on-line context sensitive help to interviewers about survey questions,
 - ◇ faster and cheaper output, more readily adapted to customers' requirements, made possible by the integration of Manipula with Blaise III,
 - ◇ development, with the help of Statistics Netherlands, of a utility for automatic documentation which would provide customers with information about the questionnaire, including routing, in a form which they could understand (Anderson, this volume),
 - ◇ tools to implement a strategy for all the surveys in a programme or agency (Pierzchala and Manners, forthcoming).

All versions of Blaise have centred the data collection, processing and delivery system on the metadata provided by the definition of a questionnaire. Blaise III ensures that those metadata are readily and fully available throughout the system. It makes sense, therefore, to illustrate how Blaise III allowed a flexible software system to be built for the IHS by concentrating on the IHS questionnaire. It should also be noted, however, that SSD already had in place by 1996 a case management system, CASEBOOK, which could deal with any kind of survey instrument without imposing any demands of its own (Gray 1995; Gray and Anderson 1996). This removed one major potential source of constraints and freed the IHS development to take place in the knowledge that all the case management issues had been anticipated and dealt with already.

A particular problem which the 4 IHS sub-surveys presented was that they all required concurrent interviewing for the parts of the interview which were asked at the individual level, that is, interviewing more than one person at a time instead of interviewing them sequentially. The longer sub-surveys might need to interview in this mode for an hour or more. SSD interviewers have often commented that the one remaining area of their work where paper questionnaires seemed better than CAI questionnaires was in allowing concurrent interviewing. CAI software tends to require a linear design, so that when a questionnaire has a section of questions which must be answered individually by each household member it is necessary to ask all the questions for the first person before turning to the second and subsequent persons. With paper questionnaires, interviewers often asked a question and took the respective answers from several household members before asking the next question. As may be

imagined, following different routing for several people concurrently was difficult and error-prone on paper, but used with care by experienced interviewers it helped maintain respondent involvement in long interviews and reduced the overall time taken. Concurrent interviewing was so important that it had to be reproduced in Blaise 2.5, despite the programming complexity and the limited form that was possible.

Thus two of the main sources of complexity in the separate surveys in Blaise 2.5 were :

- the need for separate household and individual (or grouped individual) questionnaires, and the difficulty of passing information between them and, if necessary, backing up to change data,
- the limitation of grouping at the individual level, for concurrent interviewing, to no more than 2 persons (otherwise system constraints were broken in these large surveys); this necessitated much code in the individual level questionnaire to keep track, for routing and checks, of which 2 persons in the household were currently the subject of interview.

Blaise III dealt with the first problem by removing software constraints on instrument development. This made it possible to combine household and individual levels in one questionnaire. The IHS took the new possibilities further: the 8 questionnaires of the 4 separate surveys in Blaise 2.5 were combined into 1 IHS questionnaire in Blaise III - it was large (3.2Mb compiled, representing some 3,500 uniquely identified questions, and over 30,000 in total, and 2,500 checks) but the burden of dealing with this was transferred from software to hardware. The hardware requirement will vary with the demands of the Blaise instrument, but the basic laptop computers now on the market (Pentium, 8Mb RAM) are likely to be adequate for most tasks. The IHS described above had acceptable performance on the laptops used in the field trials, which had 8Mb RAM and 486dx processors (75Mhz). The main problem for survey organisations in considering this aspect of Blaise III, for the short term, is if they rely to any significant degree on lower specification laptops which have not reached the ends of their costed working lives and which it would therefore be expensive to replace.

The aspect of performance which gave some cause for concern in the IHS trials was the *initialisation* time. Blaise III must load the Data Entry Program (over 1 Mb) and the questionnaire program (3.2Mb, in the case of the IHS) into memory at the start of an interview. There are ways that this can be done in advance by the interviewer, but they are not practical (without buying a great deal more RAM) for an organisation like SSD whose interviewers might work on several different surveys in a day and so need to keep all of them in memory. For the IHS, the time from switching on the laptop to the correct questionnaire for an address appearing on screen ready for the interview to start was 50 seconds, of which 10 seconds were start-up and case management outside Blaise and 40 seconds were Blaise initialisation. This was considerably longer than interviewers had become used to for surveys in Blaise 2.5. It is particularly a problem because it is a delay at a critical point in the interview when a respondent has just agreed to participate - sometimes after considerable persuasion - and it is vital to start straight away while their interest is engaged. Even if, as in the IHS, respondents did not object, we should be concerned about the additional stress on interviewers from tools which do not respond as quickly as they would like. Persuading people to take part in voluntary surveys can be a stressful activity, and any sources of discouragement are to be avoided.

Having said this, the IHS was an unusually large instrument, comprising some 5 hours-worth of interviewing for an average household if it had received all 4 sub-surveys instead of (as it did) just one of them. When we tested the Blaise initialisation time for one of the largest sub-surveys, the FES, as a separate instrument it was half that of the IHS as a whole. Although the IHS development project used a single questionnaire for the 4 sub-surveys, as an experiment and as an efficient way of developing an integrated instrument in a very short time, it is likely that a production version would have used the new tools in Blaise III to build separate subsurvey instruments out of the blocks of the integrated instrument instead of routing through them.

The second problem of complexity in the Blaise 2.5 instruments which we looked to Blaise III to deal with was the limitation of concurrent interviewing to 2 people at a time. We initially thought that the new facility for *parallel fields* would provide the means to interview at the same time as many people as an interviewer chose. The facility seemed to be an electronic equivalent of the multiple columns (one per person) on a paper questionnaire which allowed an interviewer to ask, for example, the age of each person in turn. However, shifting between parallel fields involves several keystrokes each time and proved too slow in the first field trial for this type of use, for which it was not designed, where rapid movement between respondents is needed. It should be noted, however, that parallel fields are useful in many other contexts, such as those involving subquestionnaires which are not linked in linear fashion to the main questionnaire: for example, a block of questions recording information about each call made at the address, or a set of blocks for subgroups of the sampling unit (e.g. tenancy groups or benefit units in a household).

The solution that we settled on for concurrent interviewing was to build the individual level of the questionnaire as a very simple structure of linked tables with a row for each household member. The Blaise structure of one of these tables is shown in Appendix A. This structure allowed interviewers to interview any combination of household members (up to ten members were allowed for) at once, so enabling individuals to be grouped for interview as they and the interviewer chose. In practice, interviewers quite often found it useful to interview three or four adults at the same time. By careful selection of the table topics and numbers of questions, the questionnaire designer could ensure that interviewers could switch between respondents with the speed required.

This structure had other benefits besides dealing with concurrent interviewing. New versions of software are often praised for the additional functionality they bring. It may be even more important, however, that they allow strategically central tasks to be accomplished more simply than in the past. The ability to adopt a very simple structure, and not to have to worry about its being wasteful in some traditional computing sense, was an important benefit of Blaise III. In this case, an individual appeared in the same row of every table, so that referencing for routing and checks throughout the questionnaire needed only to use the table index rather than searching for respondent characteristics. This made it possible to simplify the code considerably by comparison with Blaise 2.5. The apparent wastefulness, in always allowing for all household members, even when some might be children and so not eligible for most of the

individual level questions in these surveys, was outweighed (since it made no noticeable difference to performance) by the simplicity of the code it allowed and the interviewing flexibility it made possible without additional code.

For example, questions at the individual level about children (such as the monetary assets they hold as individuals, or their visits to the doctor) often present a problem in structures which do not allow for children at the individual level and instead attach them to particular adults. Much code has to be written to ensure that the questions come up at the right time for the right adults about the right children, and considerable inflexibility has to be built into the interview. The interviewer can make these judgements, which involve social understanding, more easily and appropriately for the nuances of a situation than a program. The Blaise program, on the other hand, has a role in ensuring that interviewers are always prompted to make the judgements, so the missing values which arise from interviewer error in paper surveys do not occur. The simple table structure adopted for the IHS was reliable - it ensured that information about children was collected (and only once) - and allowed the interviewers to make decisions which they make better than computer programs. The simpler code that results from this structure is easier for researchers to understand (and to write themselves), and to maintain and to change without errors.

5. Development of computer-assisted training and support systems for interviewers

In proposals for the IHS, an interviewer's set monthly workload was to be made up of 31 addresses: either 13 or 14 FRS, 10 SEH, 4 GHS and either 3 or 4 FES addresses. The sub-surveys make differing demands, which were taken into account in assessing the size of workloads. For example, the FES subsurvey would require multiple visits to each address over more than two weeks, to deal with expenditure diaries; and initial approaches to the FES addresses had to be spaced out over the month. As noted earlier, the new element in the IHS was that interviewers would have to be fully trained and instructed in all 4 surveys and, in particular, newly recruited interviewers would have to be able to handle the combination from the day they started.

It was clear that traditional training methods, which involved a heavy learning burden, could not meet the challenge for new interviewers. Attempts to combine the interviewer briefings for two of the sub-surveys, using traditional briefing methods, had been unsuccessful because there was simply too much for interviewers to absorb at one time. Extending the time available, besides being costly, would not deal with the essential learning problem.

SSD's answer was that a way had to be found to transfer a large part of the learning burden to computer: in effect, to use the computer's hard disk rather than the interviewer's memory to store the huge amount of factual information and instructions about how to handle a wide range of individual situations that these surveys required. Blaise III provided the means, as will be described below. This allowed the training program to concentrate on the higher level interviewing skills required for these

particular surveys, rather than on the minutiae of the subject matter - a much more effective training strategy. The subject matter details had to be available in the interview, but this could be done most reliably by making them available through a context-sensitive help facility.

Blaise III allows an interviewer to switch languages at any point in the interview with a hot key. SSD developed this facility to provide on-line context-sensitive question instructions in the place of a second language. An example is shown in the Blaise code at Appendix A, in the question Working. In the time available for the IHS trials we were able only to transfer existing texts of interviewers' instructions. It is clear that new design work is needed, to present the instructions in a way that interviewers can easily use during an interview. Nevertheless, even in this crude form, such context-sensitive help was found useful by interviewers in the trials and contributed to the successful redesign of the training package.

6. The final field trial : design

The overall purpose of the two IHS field trials (a planned third, large-scale trial was cancelled on business grounds) was to test that interviewers could carry out data collection and editing for such mixed workloads and to judge, as far as the very small pilot sample sizes allowed, if there were any reasons to believe that data quality would suffer. At a technical level, the trials would show if the design of the Blaise III instrument and the new methods of training and briefing interviewers on all of the sub-surveys were effective. This paper concentrates on the second field trial.

The first trial had established the feasibility of meeting IHS fieldwork requirements with experienced interviewers, even if they had not previously worked on all 4 of the IHS sub-surveys. It had also shown the need for some improvements, such as a new form of concurrent interviewing, based on tables rather than parallel fields, which were to be tested in the final trial.

However, the main point of the final trial was to investigate if interviewers with no experience of any of the surveys could be briefed for all 4 together, as demanded by the sample design requirement for them to work on mixed workloads. As noted earlier, there was no cost-effective way for them to build up experience gradually, concentrating on one sub-survey at a time.

This investigation was carried out in its most demanding form, using interviewers who were new to the job as well as to the sub-surveys. The training methods for the IHS assumed no more knowledge or experience than is provided by SSD's basic interviewer training course. Some of the new interviewers had carried out interviews for the SSD Omnibus Survey, but for others the IHS pilot was their first experience of interviewing the public. The basic training course prepares them to do this effectively, using CAPI.

The feasibility of training new interviewers to be effective on four complex surveys immediately after basic training was a key assumption in the plans for managing IHS fieldwork. IHS interviewers might have been managed

within SSD's general field force or SSD might have created a special field force for the IHS, but in either case interviewers would only be able to get experience of any of the sub-surveys by carrying out mixed workloads of all four. Special training measures might have been devised but these would have added to costs.

The new interviewers were supplemented by two experienced interviewers from the first trial, whose role was to comment on the development of the project and make comparisons with current methods and standards on the four sub-surveys. In this report they are always referred to as "the experienced interviewers", in contrast to the "new interviewers" described above.

Training pack

Before the briefings, interviewers worked through newly developed training packs. These comprised a guide to self instruction - which referred where necessary to detailed interviewer instructions - and case studies for interviewers to work through by completing IHS questionnaires on the laptop. The new documents and practice questionnaires built on SSD's normal practice for advance study before a briefing by providing a specific programme. This was designed to ensure that interviewers understood the broad nature of the task, and in particular its complexity and the need for attention to detail, while also demonstrating to them that they could handle it.

Briefing

Again, as in the first trial, new methods of briefing and preparing interviewers for the IHS were used. Since new interviewers were unfamiliar with the sub-surveys, briefing for the final trial was spread over two days, as opposed to one in the first trial. It consisted of an introduction to the IHS followed by separate briefings from each of the sub-survey teams.

The briefings given by each of the teams differed from conventional briefings in that there was less emphasis on systematic coverage of all the complex questions on the surveys. The underlying idea was that it was better to concentrate on training in when and how to access and use such detailed information than on trying to have interviewers learn it. Success with this idea relied on the usual memory burden being shouldered by greater guidance to what was significant, through the programmed advance study, and on-line context-specific comprehensive information to deal with specific queries from respondents. The briefings were designed to show how the interview for each survey formed a meaningful conversation in which valid information was extracted, and to reinforce in interviewers' minds what it was the survey designers wished them to concentrate on in the core questions and in each sub-survey's distinctive elements.

Field trainers

In the final trial, as is normal practice in SSD, field trainers accompanied the new interviewers when they started their work. On the IHS, trainers

went out with interviewers on the first two days of their quotas and then for a further day later on in the field period.

Harmonised questions

Since SSD had to rewrite the questionnaires for the sub-surveys in Blaise III, it took the opportunity to look forward and incorporate the harmonised questions which were to be introduced by April 1997. Arbitrary differences in the ways that surveys handle topics and even specific questions are particularly difficult for interviewers to cope with. Removing these differences from the IHS facilitated the training and the interviewing task, as it will for other surveys.

The IHS experimented with some minor re-ordering of questions in the sub-surveys, on the lines of a flowchart circulated to customers at an early stage, so that particular harmonised sets of questions came in the same general sequence of topics on each sub-survey.

Sample design and fieldwork

The final trial involved 8 interviewers working with full monthly IHS workloads of 31 addresses, made up of the 4 sub-surveys in the proportions described in the section on the IHS design.

The 8 sample areas (postcode sectors) were purposively selected to give the interviewers for the trial the kind of travelling distances that could be expected in a production IHS with a cluster-based design. The selected sectors covered a range of regions across England, and included urban and rural areas. Within the selected sectors, the sample was drawn by a systematic random method from the Postcode Address File (PAF), as in the current surveys, and allocated at random to the sub-surveys. SSD sent out advance letters as usual, specific to the sub-survey allocated to an address.

Following distance learning spread out over one month, and two days of personal briefing on the IHS, interviewers carried out field work between May 1-31, 1996. The FES involved placing a two week diary with each household, and spreading the initial approaches to households so that the whole field period was represented. As a result, several FES diaries were completed and collected in June.

6. The final field trial : findings

Overall assessment

The interviewers all agreed that the IHS fieldwork design was feasible from their standpoint.

Following the cancellation of the development project, visual inspection of samples of the completed interviews and FES diaries was restricted to checking that there were no obvious signs of poor data quality. The interviews were fully completed, with evidence that details were probed out. There were, for example, considerable amounts of detail about

savings and assets, which are some of the more difficult topics in the financial surveys. The rate of consultation of documents by respondents (an important indicator of quality) appeared to be, if anything, higher than usual. The interviewers appeared to have made notes on the relevant surveys in about the quantities and detail that they are made on the current surveys. Only a few FES diaries were inspected in detail: there was no evidence of poor quality, and some positive evidence that attention had been paid to avoiding deep-rooted quality problems on which the interviewers were particularly briefed.

The interviewers' confidence that the IHS fieldwork design was feasible was made despite their having faced a number of problems during the trial that would not exist in a production survey. In production surveys, questionnaires are thoroughly tested so that all routing errors have been removed. The IHS, however, had to be rewritten in Blaise III in a very short period; as a result, it contained a small number of routing errors (mainly in the IHS/FRS sub-questionnaire), which added inappropriate questions which had to be set to *Don't Know* before the correct route was resumed. The errors were due to some imprecisions in the current paper documentation, restrictions on the time available for testing, and the limited resources it was appropriate to invest in a prototype which might be cancelled. In addition, the experienced interviewers and the field trainers had some initial difficulties in adjusting to intentional changes from the current surveys, such as routing to harmonised questions and a key definition (of who was the Household Reference Person).

Two other questionnaire problems which would have to be remedied for a production version were that :

- the speed at which the program moved from question to question was slightly slower than on the separate sub-surveys as they ran in Blaise 2.5,
- start-up time for the lap tops once they were switched on was slow.

As mentioned earlier, it is likely that a production IHS would have dealt with these problems by managing the subsurveys as separate instruments rather than one huge one. The development timetables for the IHS and the separate sub-surveys had included sufficient time for full questionnaire testing.

Training pack

Originally, it was envisaged that interviewers would work through the training pack at one survey a week, but in the pilot, as time was compressed, interviewers had slightly less than this. One of the 8 interviewers joined the project at a late stage and did not have time to work through all of the training pack before the briefing. As a result, he was initially less confident than the others about the task ahead. All interviewers commented, however, that they found the materials very useful.

Interviewers commented that they would like the training pack to start with the simplest survey first (the SEH), working towards the most difficult at

the end. They preferred training information not to dwell on exceptional circumstances, but to concentrate on the more typical situations they were likely to face. Overall, the IHS self-instruction pack was considered to be excellent - very clearly laid out and easy to follow - by all of the interviewers. They had a number of detailed suggestions which have been followed up.

For a production IHS, study material would have included taped example interviews for interviewers to work through, and taped example survey introductions.

Briefing

The new interviewers felt that two days for face-to-face briefing was not enough for the IHS; three or possibly four days were recommended. This had been allowed for in SSD cost estimates for the production survey. All interviewers agreed that they would have preferred more practical sessions - both in small groups and in the group as a whole - where they could follow the questionnaires through on the laptops, with trainers acting as respondents.

Managing mixed workloads

All of the interviewers felt that it was possible to manage work with a mixed workload comprising 4 different surveys. One experienced interviewer commented that the IHS offered a "greater degree of flexibility" in that if an appointment were missed for one survey on the IHS, an interview for another survey in the IHS workload would be relatively near by and could be carried out instead.

A concern of new interviewers, which was also expressed by the experienced interviewers and SSD project managers, was that it would take them quite a long time to build up experience on the FES and GHS since there were relatively few of these addresses in each of the workloads. On the other hand, they recognised that the regularity of working on the surveys might offset this disadvantage by comparison with the current work patterns.

The interviewers all said that, initially, working on the IHS was "very tiring" because of the extra demands of "keeping four surveys in your mind at one time". But they found this much less of a concern by the end of the month.

The experienced interviewers also expressed a theoretical concern over the possibility of an increased burden of work in the last week of the field period, due to FES diary collections; normally the bulk of the workload on the SEH, FRS and GHS would have been finished by this time. However, this distribution of work had been taken into account in SSD's field planning and cost estimates.

In relation to the survey documents, the colour coding of materials was well received by interviewers, having been adopted on interviewer advice after trial 1. They felt that further colour coding would differentiate between the surveys even more clearly, and it would help with the rapid reorientation and re-focusing needed in moving from one survey to another on the same day.

Total number of days worked during the field period

The trial confirmed that the cost estimates for the IHS had made appropriate provision for the number of days that interviewers would need to work.

Blaise III software

Interviewers found that the new form of concurrent interviewing, developed since the first trial, worked well. The experienced interviewers, who had used the former method (based on the Blaise facility for *parallel fields*) in trial 1, said that the new method was much more suitable for the types of survey in the IHS. The experienced interviewers also said that the flexibility to interview any appropriate number of people concurrently was a significant improvement on Blaise 2. There were several interviews in the trial which were carried out with three or four people concurrently; new as well as experienced interviewers used the method successfully. The importance of displaying the respondent's name in a standard position on the screen at every question, for easy reference, was stressed.

On-line question instructions were used successfully by most of the interviewers but they also wanted to have a paper copy of the instructions during training and for reference at home. They commented that it could be made easier to access the instructions (in the test, a sequence of three keys had to be hit to access them). In response to these comments, Statistics Netherlands has provided the facility on a single key (from version 1.15).

SSD has developed solutions, again partly in consultation with Statistics Netherlands, to the problems of slow start-up and question to question speed in very large questionnaires which were discussed in the previous section.

Harmonisation

As in the first trial, interviewers found that harmonisation of questions (and interviewer instructions and edits) made it easier to deal with a mixed workload since a proportion of the questions were the same across each of the surveys. It was found to be particularly useful on the FES and FRS as the surveys are quite similar in content, and small differences can be the most difficult for an interviewer to deal with correctly and consistently.

Response rates

The purposive selection and the small size of the trial, in terms both of interviews and of interviewers, mean that we should be very cautious about interpreting the ostensible response rates. With these qualifications, the fact that the response rate in the IHS trial for each sub-survey was less than 5 percentage points above or below the response rate on the corresponding main survey in May means that there was no evidence at this stage of a potential response problem in the IHS fieldwork design.

7. Conclusion

Blaise III allowed very rapid development of an integrated survey with a completely new structure from the 4 Blaise 2.5 surveys which comprised it. Nearly all the work was done in two months. When the structure using parallel fields was found not to be suitable for the intended purpose in the

first field trial, a wholly different structure was able to be designed and implemented in a week. The Blaise control centre for instrument design, and the removal of software constraints - permitting simple, redundant structures - were the major factors in allowing rapid development.

The field trials showed that it is possible to use the Blaise III facility for a second language, used to provide context-sensitive help to interviewers about questions, to transfer much of the memory burden for complex surveys to the laptop computer. This makes it possible to redesign interviewer training and briefing to concentrate on interviewing skills and use, rather than memorising, of the detailed information. The trials achieved significant improvements in training, as shown by the effective performance of new interviewers on 4 surveys which it had previously only been possible to introduce one at a time over a lengthy period.

Besides the lessons for Blaise III, the project showed the technical feasibility of an integrated survey and the importance of harmonisation to interviewers (and so to data quality) as well as to users.

REFERENCES :

Anderson, S Automated paper documentation of Blaise III datamodels (*this volume*)

Costigan P, Sjodin S & Thomson K The development of the Family Resources Survey in Blaise. *Essays on Blaise 1992* (1992) Statistics Netherlands, Voorburg, The Netherlands, 1992

Elliot, D & Barton, J Sample design and stratification options for an integrated household survey. *Government Statistical Service Methodology Series*, London, United Kingdom (forthcoming, 1997)

Government Statistical Service *Harmonised concepts and questions for government social surveys*. Office for National Statistics, London, United Kingdom, 1996

Gray, J and Anderson, A The data pipeline : processing survey data on a flow basis. *Survey and Statistical Computing 1996. Proceedings of the Second ASC conference*, London, United Kingdom, 1996

Gray, J An object-based approach to the handling of survey data. *Essays on Blaise 1995* (1995) Statistics Finland, Helsinki, Finland, 1995

Manners, T & Bennett, N Computer assisted survey methods (CASM) at OPCS and some current issues in the use of Blaise for the Labour Force Survey. *Essays on Blaise 1992* (1992) Statistics Netherlands, Voorburg, The Netherlands, 1992

Manners T, Cheesbrough S & Diamond A Integrated field and office editing in Blaise. *Essays on Blaise 1993* (1993) OPCS, United Kingdom

Priest, Gordon E. The Issue of Harmonization of Data from Diverse Sources. *Paper presented at the Eurostat Workshop on Harmonization of Survey Variables, London, 1996.*

APPENDIX A

Example table block in the IHS instrument

The table TIHSILO contains harmonised questions (for UK government social surveys) to establish employment status according to the International Labour Office (ILO) definition of 1982.

SSD standards for Blaise code require the following naming conventions seen in this block. Names start with T (tables), B (blockname of blocks), Q (question name of blocks), L (locals), DM (datamodel level variables). LDMNAMES and LDMINDINTERVIEW are, respectively, datamodel level locals for respondent's name and for switching a particular individual's interview on and off as required. DMDLSUN and DMHsize are datamodel level auxfields for, respectively, the date of the Sunday preceding the interview and for the number of people in the household. YN is a TYPE for Yes and No answers. (HLP<F9>) indicates to the interviewer that context-sensitive help about the question is available on the hot key, F9. In Blaise III, ^ indicates a text substitution, and @/ gives a new line on screen.

TABLE TIHSILO

LOCALS { TIHSILO }

LTlooper : INTEGER

BLOCK BIHSILO

AUXFIELDS {BIHSILO}

Name : STRING[12],EMPTY

FIELDS {BIHSILO}

Wrking "(^LDMNAMES[LTlooper])@/

{question} Did you do any paid work in the 7 days ending Sunday the ^DMDLSUN, either as an employee or as self-employed? HLP<F9>"

{help} " 'Work' means ANY work for pay or profit done in the reference week, including Saturday jobs, casual work (eg baby-sitting, running a mail order club, etc.) and children with a paper round etc, even though they may still be at school.@/ @/

Include self-employed people if they work in their own business, professional practice, or farm for the purpose of earning a profit.@/ @/

In general, you should take the respondent's definition of whether they are in paid work or not. If unsure: @/

a job exists if there is a definite arrangement between an employer and an employee for work on a regular basis, whether work is full or part time. @/@/

Long term absence from work: if total absence exceeds 6 months, a person has a job only if full or partial pay has been received during absence. @/@/

Seasonal workers 'between seasons' (ie not currently working) should be coded 2. (Note, the odd week of sick leave during the working season should be treated the same as in other work, and coded 1."

: YN

.....other questions

RULES {BIHSILO}

Name.SHOW

Wrking

.... other routing

ENDBLOCK {BIHSILO}

FIELDS { TIHSILO }

QIHSILO : ARRAY [1..10] OF BIHSILO

RULES { TIHSILO }

FOR LTlooper := 1 TO DMHsize DO

IF LDMINDINTERVIEW[LTlooper] = 'NOW' THEN

QIHSILO[LTlooper]

QIHSILO[LTlooper].Name := DMNAMES[LTlooper]

ENDIF

ENDDO

ENDTABLE { TIHSILO }

FIELDS { DM }

QTIHSILO : TIHSILO

APPENDIX B

An extract from the RULES paragraph for the IHS instrument, illustrating the combination of blocks from the 4 subsurveys where each asks a different set of questions about rooms in addition to a shared set. These questions are followed by a block of harmonised questions about housing tenure, and this is followed by some more detailed questions on tenure for the SEH.

Block question names start with Q. Each block name is accompanied by a comment giving the name of the include file in which it is found; this name also indicates which sub-survey the block is from (e.g. SEHROOMS.V* is a file containing SEH - Survey of English Housing - questions about rooms which are specific to that survey) or that the block is common to all sub-surveys in the IHS (e.g. IHSROOMS.V* is a file containing questions about rooms which are common to all 4 sub-surveys). At the time of the IHS, SSD standards allowed more than one block per include file, but we now only allow one (for ease of management and re-use).

```
RULES { DM }
```

```
routing.....
```

```
  QIHSRooms { IHSROOMS.V* }
```

```
  IF QAdmin1.Survey = SEH THEN
```

```
    QSEHrms { SEHROOMS.V* } {+ more SEH blocks...}
```

```
  ELSEIF QAdmin1.Survey = GHS THEN
```

```
    QBRooms { GHSROOMS.V* }
```

```
  ELSEIF QAdmin1.Survey = FRS THEN
```

```
    QFRSrooms { FRSROOMS.V* }
```

```
  ENDIF
```

```
  QTenure { IHSTENUR.V* }
```

```
  IF QAdmin1.Survey = SEH THEN
```

```
    QTenur { SEHSHARE.V* }
```

```
    IF QTEthnic.NumFamUnits>1 THEN
```

```
      QSublett { SEHSHARE.V* } {+ more SEH blocks...}
```

```
    ENDIF
```

...more routing.

The Transition from Cases and BLAISE 2 to BLAISE III at the National Agricultural Statistics Service

Asa Manning, National Agricultural Statistics Service, USA

1. A History of the Conversion Effort

NASS was one of the first organization to use Blaise III for production data collection and editing. Blaise 2 had been used, first for interactive editing and then data collection on several smaller surveys beginning in 1992. A pilot project to study the feasibility of using Blaise 2 for our major surveys had been initiated in three states (CO, IN, & WY) in September 93. The goal was to produce both data collection and interactive editing instruments from the same source code. In addition, the interactive editing instrument would not only be used to edit data collected in Blaise, but data keyed from paper questionnaires.

This multiple mode concept was fairly successful in Blaise 2. However, the instruments for our largest applications exceeded the capacity of the software. In addition, NASS had identified a number of enhancements that would be needed in the CATI system before pursuing a larger scale implementation. So NASS was waiting anxiously as the initial versions of the Blaise III 1.0x series were released.

The first production use of Blaise III took place in March 1994 in the original three pilot states. The freedom from the memory limitations of Blaise 2 was an obvious advantage. A number of significant bugs and limitations were encountered, but with some real time adjustments in our procedures, the survey was completed on time. The promise of the software was very enticing, but the amount of work left to be done was daunting.

We communicated the problems that occurred during the survey to Statistics Netherlands. NASS requested that they make a visit to one of our three pilot states, prior to the next survey. To Statistics Netherlands credit, they did. Lon Hofman, Mark Pierzchala, and Asa Manning spent a week in the Indiana SSO during April 1994 repeating the March experience.

Those days were key to the survival of our pilot study in NASS. First, NASS management needed to see this type of commitment from Statistics

Netherlands. Secondly, Lon was able to identify and eliminate many bugs from the software. He also was able to offer us advice on certain issues related to our Novell Local Area Networks (LAN). The spirit of cooperation gained from that visit helped get us through what was to be a long road ahead. Little did we know at the time, but it would be over two years before the release of a version of Blaise III that would finally provide almost all of what NASS needed.

The pilot study continued with each quarterly Agricultural Survey through September 95. The first step to expand into additional states occurred in the July 1995. The CASIC Group spent a week in St. Louis, Missouri, training 13 states (12 new states and one of the original pilot states). Two representatives of each state were trained to manage a Blaise III survey in their office. The agenda included sessions on setting up a Blaise survey on the state LANs, working with the CATI management system, managing the flow of data from data collection to editing, using interactive editing, and managing the overall survey process. Each participant was then charged with returning to their respective office and transferring the training to the other staff.

The states attending that first workshop were states involved in NASS's Cattle on Feed program. This application was slated to be downsized in late 1995, with Blaise III as the centerpiece of data collection and editing. PC-SAS on the LAN was to be used for post-edit analysis and summary. This application was scheduled to change for October 1995, but was delayed until January 1996 due to internal changes to the survey program. Thus, the first survey conducted by states trained during the workshop, was the December Agricultural Survey. Fourteen states used Blaise III. Texas was delayed for internal reasons for several quarters. There were some problems encountered, primarily because this is NASS's largest and most complicated CATI application. Some of those challenges still remain and those will be mentioned later.

Another group of 12 states (including two original pilot states) was trained during November 1995. Three of those new states started using Blaise III in January as the Cattle Survey debuted in Blaise III. This brought our total number of states using Blaise III to 17. The experience on this application was improved as states began to build on their December experience, and benefited from the smaller, less complex nature of the Cattle survey. The other seven new states used the March Agricultural Survey Application, bringing the total number of states with Blaise III experience to 24.

We delayed further expansion until the summer of 1996. This allowed the newly released Blaise 1.1x series to mature. The upgraded software offered NASS many needed enhancements. CATI Management was upgraded with many additional features for the telephone supervisor. Manipula was enhanced with an integral check and other features which allowed NASS to dramatically speed up many processes. The new Manipula software offered NASS the potential to develop a more user friendly interface for the statisticians using interactive editing. Basically the Blaise III 1.1x series included many of the features that were in the original Blaise 2 software.

During the June Agricultural Survey, NASS took a long awaited step, when Blaise III 1.1i was used in two states (MT & KS), while Blaise III Version 1.05 was used in the other 22 states. The enhancements to version 1.1i were well received, but some signs of minor bugs existed. Our user interface for interactive editing was much preferred to the standard system interface, but slow form retrieval was identified as a problem. The positives far outweighed the negatives, so NASS went forward with plans for further expansion.

A third workshop was held for twelve new states during July 1996. This brought the total number of trained states up to 37. Version 1.1j was used at the workshop and shortly thereafter distributed to all states using Blaise III.

All those states used Blaise III on the September Agricultural Survey (including Texas). The improvements over version 1.05 were very popular with the end users. The states finally had a system with almost all of the features they had been looking for.

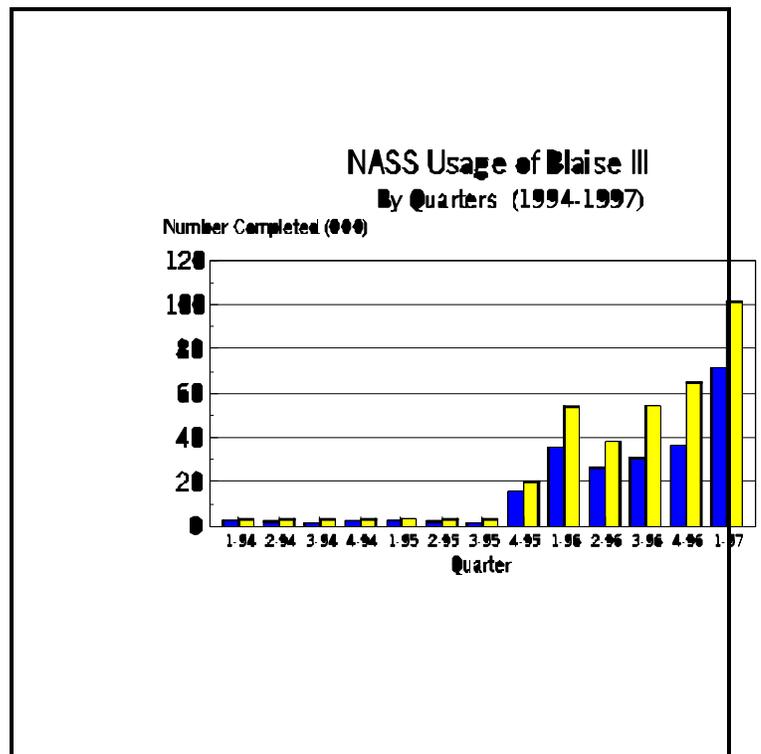
A small workshop was held in November 1996 to train five more states. This left only Nevada untrained. Version 1.12 was used at this workshop. It was distributed to those five states soon thereafter, so they could prepare for the January Cattle and Sheep surveys, when they would use Blaise III operationally for the first time.

The December Survey 1996 was conducted with Blaise in the same 37 states as September. Most states were very pleased with the results. In states with the largest instruments and largest samples, we continued to observe some performance type problems. Texas could not support as many interviewers as needed, without adversely affecting question to question performance. Other states may have seen signs of this as well. The slowness of form retrieval in Maniplus contributed to the staff in Texas being unable to complete all editing in Blaise. The need to "touch" each warning error to suppress it, proved to be too time consuming when samples were large.

During January 1997, the Cattle and Sheep applications (running Blaise III version 1.12) were used in 42 states. The smaller instruments reduced the impact of the performance related issues and resulted in the most successful use of Blaise III thus far.

NASS plans to train Nevada in the next few months. That will bring all the states on-line. During this time frame the last CASES applications are also being replaced.

Figure 1 gives a graphical representation of the quantity of data being collected and edited with Blaise III in NASS.



It has been a much longer road than we expected when we first started this effort over three years ago. There have been occasions when NASS wished it had not been on the "bleeding edge" of the Blaise III software evolution, but we feel the software would not be where it is today without our input. To be honest, if NASS had started using Blaise III later than it did, there would still have been bugs for us to contend with. NASS puts pressure on many pieces of the system and uses the software in some unique ways. Thus other organizations would not have found all the bugs that NASS did. This is proved by the bugs that other organizations found even after NASS had used the software for several surveys. The shakedown cruise for any software is a painful process, and the smaller the number of users the more pain each one will probably feel. It is nice to be looking back on most of this process, at least for the DOS version.

NASS has converted a large number of applications from CASES or Blaise 2 to Blaise III. Data collection and interactive editing instruments are generated from one copy of source code for all these applications, except June Area. The June Area Survey is a large area frame survey where CAPI was tried, but has been discontinued. Each editing instrument must be capable of handling both data collected with Blaise and data entered from paper questionnaires. The following table provides information on each application.

Blaise III Applications In Use At NASS

The above table illustrates the large number of applications (16) that NASS has converted to Blaise III. With a full implementation of Blaise III nearly at hand, the volume of data collected with Blaise could approach 400,000 in 1997. Interactive editing volume could reach about 600,000. Another factor that bears mentioning here is that each of these applications is running on LAN's in the various state offices. This adds significantly to the support demands because there are subtle differences in LANs from location to location and in the staff expertise as well. Because there is hardly a month that goes by without a survey and in some months there may be up to four concurrent surveys, the support demand is virtually constant.

2. An Update on Computer Assisted Personal Interviewing (CAPI)

At past conferences, NASS's Computer Assisted Personal Interviewing research was discussed. During the last couple of years, the research was limited to work with the June Area Survey and major multiple frame surveys (Agricultural Survey, Cattle, Sheep, & Hogs) in Indiana. On the multiple frame surveys the laptops were used for personal interviewing, as well as telephone interviewing from the field enumerators home. NASS referred to this as CATHI (Computer Assisted Telephone Home Interviewing), to distinguish it from CAPI. This effort proved that Blaise was a capable tool for collecting data on a laptop computer in either mode. It was also proven that the technology could support the electronic transmission of data into the state office.

In late 1996 upper management in NASS decided to terminate further CAPI efforts. There were four major reasons.

1) NASS's survey program does not provide the type of surveys that lend themselves to CAPI. Most surveys are of very short duration (less than two weeks) and as documented above are frequent in occurrence. The overhead of supporting CAPI increases significantly per-survey unit in this environment. There ends up being an almost constant support

demand on the state office staff. More surveys with a longer period of data collection would make it easier to justify CAPI/CATHI at NASS.

2) The cost of laptop computers has increased from about \$2000 early in the study to over \$3000 currently. The demand for low end machines was not sufficient to make it economically feasible for vendors to produce them. With potentially between 1000 and 1500 field enumerators to equip with laptop computers, the start-up costs for CAPI are simply too high at this time.

3) NASS has given priority to upgrading its workstations in the state offices for Blaise and other applications. Servers and LAN cabling are out of date and also need improvement. Money is also being invested in a Wide Area Network to connect the field offices and HQ. This will provide the communication backbone for future technologies. These competing initiatives made it even more difficult to find funding for CAPI.

4) Support for CAPI would be more than most field offices could handle. Indiana had a very skilled staff that was determined to make CAPI work. Yet, even in Indiana, there were times when things did not go smoothly and the stress on the staff took its toll. Essentially, if Indiana was finding it a challenge, others would likely be overwhelmed. It might be most efficient to support CAPI from certain regional sites, but this would require some structural changes that simply were not going to be dealt with now.

NASS may revisit CAPI in the future. Changes will have to be made in several areas before this occurs.

3. How Well Is Blaise III Doing ?

As chronicled above in the recent history of Blaise III usage in NASS, there has been much progress in the evolution of the software. We are convinced it is the most flexible, versatile survey processing software in the world. However, there are still areas that need to be addressed by NASS, Statistics Netherlands or both.

It has been NASS's goal since it first starting using Blaise to produce instruments for data collection and editing from one source code. This offers the potential advantage of cutting down total maintenance time. If data can be cleaned in Blaise IE, then other post-edit processing can move to the LAN as well. NASS has been very successful in completely downsizing all of its list frame survey applications. However, we have not been as successful on the more complex multiple frame surveys. The administrative coding is much more complicated on these applications, as is the CATI routing. The extra level of complexity when built into the instrument appears to be leading to some flakiness in the IE instruments. On rare occasions, a situation which should trigger an error in the editing instrument does not. This can not be traced to a logic problem, as the same exact error was caught on many other occasions. Even though this is happening on a small number of records, the impact on subsequent processes can be very severe.

NASS plans to explore the possibility of separating the data collection and interactive editing instruments. This would reduce the size of the data collection instrument about 30 percent and the interactive editing instrument more than 50 percent. By simplifying the logic in many error checks we hope to create a more stable interactive editing instrument. Since the checking mechanism is a major player in performance, we may also see an improvement there as a side benefit. As has been mentioned earlier, slow performance continues to be a problem with the integrated CATI/IE instruments in certain situations.

NASS uses Manipula a great deal in its Blaise applications. Many of the past support calls from the states dealt with Manipula failures. The number of failures increased noticeably when Dell pentium workstations running at 166 mhz began to be used. As it turns out, a bug in the system was creating occasional run time errors on pentium workstations. Our testing on the faster Dell workstations with version 1.15 looks positive. If these workstations prove to be reliable, it will reduce the time wasted when failures occur. Since many of the Manipula processes are potential bottlenecks, running them on the fastest workstation available has obvious advantages. Hopefully, we will be able to give a good report on the March results, when this paper is presented in Paris.

4. Agricultural Census - A Challenging New Application For NASS

An Agricultural Census is conducted in the United States every five years. This survey attempts to contact all farm operations and collect information about acreage, production, and value of sales for crops; production and sales for livestock; and other data relating to agriculture. Survey results are released at national, state, and county levels. Since this survey is close to a complete enumeration of the agriculture sector, NASS uses this data to "true up" its own estimates.

The Census Bureau, an agency of the Department of Commerce, conducted this survey through 1992. In 1995, the Department of Commerce came under scrutiny for elimination or downsizing. Although the Commerce Department survived, funding for future Agricultural Censuses was not included in their appropriations. NASS stepped forward and asked to have the appropriations added to its budget. Congress passed legislation in late 1996 completing the transfer of responsibility for the Agricultural Census to NASS. This brings virtually all surveys of the agricultural sector under NASS's control.

This survey is much larger than any survey that NASS has done before. Initial mailings in December of 1997 will total about 3.5 million questionnaires. NASS does not have time to put the infrastructure in place to handle the 1997 survey. NASS will contract many tasks on the 1997 survey back to the Census Bureau. The Census Bureau will be heavily involved in handling the processing of paper questionnaires, including mailings, checking in, and keying. NASS will handle all the CATI follow-up and editing. All CATI instruments will be developed in Blaise III. The existing interactive editing system developed by the Census Bureau, will continue to be used in 1997.

There will be three Blaise instruments developed for 1997. The main Agricultural Census Instrument will be used to contact large farms that do not return a paper questionnaire and to follow-up in counties where the response rate is inadequate. The Coverage Evaluation Survey instrument will be used to follow-up on operations that did respond and will gather information to identify duplication or operations that might have been misclassified. The Non-response survey instrument will be used to quickly phone non-respondents and determine whether the operation is in or out of scope.

The main Agricultural Census Instrument will present NASS with several challenges. Each is detailed below :

1) The instrument will have up to collect data on up to 900 items and CATI interviews can last over an hour. However, most operations will probably have less than 100 items. To identify the commodities produced by a given operation, NASS plans to use a coding technique to first build a list of commodities. Once the list is complete, detail acreage, production, and value of sales questions will be asked for each commodity in the list. Our goal is to expedite the process of collecting the crop commodity data, while not missing any. NASS has never used the Blaise coding modules, so this will be a learning experience.

2) The instrument will be one of the biggest that NASS has put together. Question to question performance is a concern. It will help that the instrument is interview only. Thus the logic for checks will not be cluttered with coding for interactive editing or paper forms. To help meet this challenge, the instruments will be designed to maximize instrument speed. NASS is also upgrading to state of the art file servers (currently 486) in the field offices during 1997. States still using pre-10BaseT LAN cabling technology will be upgraded to 10BaseT. Additional pentium workstations will also be purchased, so that the bottom end CATI workstation will be a 486 running at 33 mhz.

3) The low response county follow-up will require that NASS complete enough CATI interviews to bring the response rate in those counties to a pre-determined level. This will allow NASS to use the quota feature of Blaise CATI Management for the first time.

4) The typical NASS CATI survey lasts no more than two weeks. The data collection period for this survey will last about four months. Field offices will be able to offer interviewers a longer period of active employment than in the past. NASS must also be aware for the first time that "interviewer burnout" may become a problem. We are also concerned whether the field offices can find a sufficient number of interviewers to meet our needs over such an extended period. The field offices will have to be innovative in their hiring, training, and management of the interviewers.

The CASIC Group in NASS will be kept very busy as we prepare for these applications. We are excited about the unique challenges that they present us.

5. Where To Now For Blaise III ?

As NASS wraps up its conversion from CASES and Blaise 2 to the DOS version of Blaise III, we are basically satisfied. However, there are issues that remain and can hopefully be met as Blaise III moves to Windows. Three major items follow :

1) One of the complaints that we are hearing from the field offices is that it takes too long to edit forms. The primary step that is consistently criticized is the need to touch every soft error to suppress it. NASS would like the option of displaying a listing of all soft checks in a pop-up window. The editor could scroll through the list and click on any message they wanted to review in detail. Once all messages were reviewed, the editor would be able to suppress all warnings with one key stroke from the pop-up window. NASS also needs to continue to minimize the number of micro edits, but that is a cultural change that takes time.

2) NASS would like to see the Blaise software be more open. The proprietary nature of the Blaise format presents barriers to users that need access to the data. Unless they are capable of writing Manipula programs, the data must be converted to another format before they can use it. This conversion itself, should be unnecessary. Blaise III should be able to read and write to other formats. If we could access XBase and SAS files directly, many cumbersome ASCII transactions could be eliminated. We suspect most users of Blaise III could benefit from this ability, and suspect that it is a top priority for many organizations using it.

3) NASS would also like to see the data collection and/or editing software run in a client server mode. The instrument could be located on a centralized platform (server), and users in the field offices would be able to access it. The centralized location of the instrument appeals to NASS, because it reduces the difficulty of supporting applications running in a distributed manner. NASS has taken the first step in this direction by connecting the field offices and headquarters LAN in an Wide Area Network (WAN). Client server is viewed as a longer term goal than the other two.

We have presented a chronology of NASS's experience with Blaise III. From the original pilot study to the planning for the 1997 Agricultural Census, Blaise III has become an integral part of NASS's survey processing methodology. It has not met all of our needs, but it is doubtful that any software could, given NASS's diverse survey program. Hopefully, as Blaise III moves to Windows, the remaining challenges can be met.

SICORE, un outil et une méthode pour le chiffrement automatique à l'INSEE

Eric Meyer et Pascal Rivière, INSEE, France

SICORE (Système Informatique de COdage des Réponses aux Enquêtes) est un système de chiffrement automatique développé par le Département des Applications et des Projets de l'INSEE, qui a fait l'objet de tests sur de nombreuses variables et a déjà été appliqué en production avec succès. Son usage est appelé à se généraliser à l'INSEE, et au sein d'autres services statistiques, français et éventuellement étrangers. Le but de ce document est de présenter la méthodologie SICORE, de décrire le système SICORE actuel, son fonctionnement, et enfin d'explicitier son utilisation en pratique.

1. La méthodologie SICORE

Pour bien comprendre SICORE, nous allons tout d'abord définir ce qu'est le codage, avant d'aborder le principe du codage automatique QUID qui a servi de point de départ pour SICORE. Ce n'est qu'après ce préalable que nous pourrons mettre en évidence ce qui a conduit à l'élaboration de SICORE.

1.1. Le codage

Le codage, également appelé chiffrement, est une opération qui consiste à interpréter le sens d'un libellé, en attribuant à ce dernier un code. Le libellé doit décrire un certain concept statistique, c'est-à-dire une certaine variable, et le code doit appartenir à une certaine nomenclature, que l'on suppose connue. Il se peut aussi que le texte seul ne suffise pas pour le chiffrement, et que l'on ait alors besoin de *variables annexes* pour coder : ainsi, pour le chiffrement de la profession, le statut ou la qualification font partie des variables supplémentaires généralement utiles.

L'opération de codage est indispensable, dans une enquête, dès lors qu'il existe des questions ouvertes dans le questionnaire et que l'on veut les traiter statistiquement : en effet, il est nécessaire de diviser les réponses en plusieurs catégories, et placer une réponse dans une catégorie n'est rien d'autre que coder.

En pratique, le codage peut être réalisé de trois façons :

- le codage manuel dans lequel l'opérateur utilise ses propres connaissances et éventuellement des documents sur papier,
- le codage assisté dans lequel l'opérateur s'aide des fonctionnalités d'un logiciel mis à sa disposition, avant de prendre sa décision finale,
- le codage automatique où l'informatique réalise seule le codage, avec pour conséquence l'existence de rejets de codage qu'il faudra traiter ultérieurement par l'une des deux méthodes précédentes.

Le chiffrage "à la main" a l'inconvénient d'être très coûteux, car il requiert beaucoup de temps ; c'est pourquoi les services statistiques cherchent de plus en plus à réaliser cela soit de façon assistée, avec des logiciels interactifs d'aide au codage, soit avec des programmes de codage automatique.

1.2. QUID

L'INSEE s'est préoccupé de codage automatique depuis longtemps : l'algorithme QUID, mis au point en 1979 par l'unité de recherche de l'INSEE, a donné de bons résultats. Le logiciel qui met en pratique cette méthode a été utilisé sur de nombreuses applications, en particulier le codage de la profession dans l'enquête Emploi, et le codage de la catégorie socio-professionnelle et de la commune dans les déclarations annuelles de données sociales.

La méthode QUID est générale, au sens où elle est indépendante de la variable à coder. Mais pour la mettre en oeuvre, il est nécessaire de disposer d'un fichier de référence reliant les libellés aux codes, appelé *fichier d'apprentissage* : il s'agit en quelque sorte d'une nomenclature fondée sur les réponses effectives des enquêtés passés.

L'algorithme fonctionne alors en deux temps. Dans une première phase, il "apprend" le fichier, l'assimile, et crée une structure synthétique du fichier ainsi digéré, structure appelée *arbre de questionnement*. Dans un deuxième temps, le programme est capable de coder, en parcourant cette structure : l'arbre généré a en effet l'avantage d'être remarquablement adapté au chiffrage, qui se révèle donc très rapide avec cette méthode.

1.3. Les problèmes rencontrés

Le codage automatique ne permet jamais de traiter tous les libellés : il reste une proportion de non-codés, qui n'est pas toujours négligeable. Une analyse de ces échecs devrait permettre d'améliorer petit à petit le fichier d'apprentissage, en y incorporant les intitulés les plus fréquents parmi ceux que l'on n'a pu chiffrer auparavant. Comment et en fonction de quoi le mettre à jour ? Avec quels outils ? Selon quels critères ?

En pratique, cette amélioration régulière et raisonnée du fichier d'apprentissage n'a pas réellement eu lieu avec QUID ; à titre d'exemple, l'enquête Emploi, qui utilise cette méthode, n'a pas changé ce fichier de référence depuis la création de la chaîne de codage automatique, en 1990.

On peut aisément expliquer cette lacune. En fait, l'enrichissement d'un fichier d'apprentissage nécessite une organisation adaptée : des experts, spécialisés dans le chiffrement d'une variable, des logiciels facilitant le travail de mise à jour effectué par les experts (ateliers d'expertise), une centralisation des connaissances sur chaque domaine, et une prise en compte réelle des chiffrements automatiques ayant déjà eu lieu.

Tout cela n'était pas pris en compte par les structures et outils existants : le chiffrement automatique proprement dit fonctionnait bien, mais le travail qui restait à faire était *autour* du chiffrement.

1.4. Le projet SICORE

SICORE a été lancé, vers la fin 1993, pour deux raisons essentielles. En premier lieu, comme nous venons de le voir, il fallait bâtir une organisation et mettre au point des outils autour du codage. Mais le souhait était également de généraliser l'utilisation du codage automatique à l'INSEE : pour cela, il devenait indispensable de rendre cette opération plus facilement accessible aux non-informaticiens.

La volonté a donc été affichée dès le départ de créer un système simple, unificateur, efficace et portable pour le chiffrement automatique, avec un algorithme de codage s'appuyant mathématiquement sur QUID, mais cherchant à l'assouplir et à en généraliser l'approche. De plus, comme l'efficacité n'est jamais de 100%, il fallait préparer le cycle d'enrichissement des connaissances.

Cela passait d'abord par l'écriture d'outils en plus de la codification automatique : visualisation des libellés non-codés, ou bien analyse de fichiers de connaissances sur le codage (en particulier le fichier de référence). Cela nécessitait que les connaissances statistiques existent, et qu'elles évoluent régulièrement sur la base de ces analyses. Enfin, cet objectif ambitieux exigeait également que soient mis en place des repères méthodologiques et organisationnels pour la mise à jour de connaissances.

Cet ensemble "programmes + connaissances statistiques + méthodologie + organisation" constitue ce que nous appellerons un **système** de codage automatique, prenant en compte l'ensemble des tenants et aboutissants.

Pour arriver à cela, le principe de base du projet a toujours été de "faire d'abord", c'est-à-dire de fabriquer le plus tôt possible des prototypes et de les appliquer. Ceci permet en effet d'anticiper sur les outils, de se donner la possibilité de les ajuster, et de créer la méthodologie et l'organisation de façon naturelle et pragmatique. Il ne fallait pas imposer *a priori* des outils et une méthodologie : ceux-ci devaient se dégager de l'utilisation courante. Il en est de même pour l'organisation : une organisation ne se décrète pas, elle naît de l'action.

2. Le système SICORE

Le projet SICORE est à l'heure actuelle achevé : le système SICORE, au sens proposé plus haut, existe effectivement. En effet, on dispose bien

des 4 composantes préconisées, mais elles ont toutes vocation à évoluer ; les connaissances s'enrichissent pour améliorer l'efficacité et la qualité des futurs chiffrements ; de nouveaux principes méthodologiques émergent d'une utilisation régulière du codage automatique ; enfin, les organisations sont malléables et éphémères, leur existence dépendant de leur utilité réelle et de la volonté des acteurs.

2.1. Les programmes et les connaissances

2.1.1. La séparation programmes — connaissances

C'est là un des apports fondamentaux de SICORE. Les programmes de codage automatique de SICORE sont généraux : ils ne concernent aucune variable particulière ; à eux seuls, ils sont donc incapables de réaliser le moindre chiffrement. Pour fonctionner, il leur faut des *connaissances* sur la variable à coder, complètement indépendantes des programmes généraux.

Le fichier d'apprentissage, que QUID utilisait déjà, est l'une de ces connaissances. Mais SICORE en emploie d'autres, notamment les *règles de normalisation* (synonymes, mots vides, troncatures, caractères à éliminer) et les *règles logiques* (prise en compte des variables annexes dans des règles de décision de type "système expert"), et les *paramètres d'apprentissage* (permettant à l'utilisateur expert en variable d'orienter l'algorithme d'apprentissage comme il le désire).

Toutes ces connaissances dépendent de la variable à coder, mais il s'agit de connaissances générales dans leurs domaines respectifs : ce sont toujours les mêmes, quelle que soit l'enquête. Pour atteindre cette généralité, les règles de décision, qui mettent en jeu les variables annexes, font intervenir la modalité "manquant", ce qui permet de coder les enquêtes pour lesquelles la variable annexe est absente.

Mais il existe également des connaissances spécifiques à l'enquête qui viennent enrichir les connaissances générales pour répondre au mieux aux besoins spécifiques de traitement : dessin de fichier à coder, table de passage entre les variables annexes de l'enquête et celles des règles logiques.

En définitive, **c'est toujours un couple [SICORE — bases de connaissances] qui réalise le codage.** On emploie, par souci de simplification, l'expression "SICORE code ...", mais elle est en toute rigueur impropre. La distinction nette qui est faite entre programmes généraux et connaissances offre au statisticien l'avantage de bien maîtriser ce qu'il fait : en quelque sorte, les connaissances sont des spécifications écrites à l'extérieur des programmes.

2.1.2. Le fonctionnement du codage automatique avec SICORE

Pour coder, SICORE utilise toutes les bases de connaissances dont il dispose, et opère en trois temps.

En premier lieu, il simplifie le libellé, grâce à des règles de normalisation (mots vides, synonymie, ...) : on passe d'un texte à un autre texte, on ne change donc pas d'univers.

Dans un deuxième temps, il faut passer de l'univers des textes à celui des codes : on va transformer notre libellé simplifié en un code, éventuellement imprécis, après apprentissage du fichier de référence. Sur le plan théorique, l'algorithme s'inspire largement de l'algorithme QUID¹², mais en diffère cependant par certains aspects, ce qui a permis non seulement de gagner un temps important (SICORE effectue un apprentissage 40 fois plus vite que le logiciel QUID sur certains essais), mais aussi d'améliorer l'efficacité de codage, et enfin d'obtenir une meilleure stabilité de l'arbre de questionnement.

Si le code obtenu est un code intermédiaire, on lève l'ambiguïté sur le code définitif grâce aux règles logiques de codage intégrant les variables annexes, informations supplémentaires que nous avons déjà évoquées.

Avec SICORE, le codage d'un libellé est très rapide, même si les bases de connaissances sont de taille importante : de l'ordre de quelques millièmes de seconde tant sur site central que sur un PC de type pentium.

2.1.3. SICORE en tant que logiciel

SICORE est un **ensemble de programmes qui sont des briques élémentaires pour le codage** : lecture de connaissances, apprentissage, normalisation, reconnaissance de libellé, traitement de variables annexes, ... Ces briques sont écrites en langage C (norme ANSI), et fonctionnent aussi bien sur site central MVS que sur PC (WINDOWS 3.1, WINDOWS 95, WINDOWS NT 4).

Sur site central, il n'y a rien de plus que les briques élémentaires, qui sont appelables par des programmes.

Sur PC, toutes ces briques sont réunies au sein d'un seul et même logiciel. Outre les programmes de chiffrement automatique, il comporte une interface permettant de créer, d'analyser et de mettre à jour des connaissances. L'objectif majeur de ce logiciel, appelé *l'outil SICORE*, est de **mettre au point des connaissances** ; le codage automatique fait partie des outils qui contribuent à cette mise au point, mais il n'est pas ici une fin en soi. L'outil SICORE a très peu d'utilisateurs (il est utilisé sur 7 postes aujourd'hui) : ce sont les experts de variables.

2.2. Bases de connaissances

SICORE a été appliqué à de nombreuses variables, ce qui a nécessité, dans plusieurs cas, la création de bases de connaissances. A l'heure actuelle, de telles bases existent en langue française pour les variables suivantes : SICAV, communes, départements, nationalités, professions et

¹²Mais pas sur un plan informatique : aucun programme du logiciel QUID n'a été réutilisé.

CS, occupations, lieux de séjour, raisons sociales et adresses d'établissements.

Toutes ont conduit à des tests sur des données réelles : par exemple, le codage de la profession par SICORE a été appliqué à des fichiers de quelques dizaines de milliers de libellés, provenant du Recensement de la Population, des Déclarations Annuelles de Données Sociales, de l'Etat-Civil, de l'enquête Emploi, ou de l'enquête Permanente Conditions de Vie.

Ces bases de connaissances sont de tailles très variables : presque 5 millions de libellés pour le fichier d'apprentissage des établissements au niveau national¹³, de l'ordre de 45000 pour celui des communes¹⁴, un peu plus de 4000 pour les SICAV, quelques centaines pour les départements ; de même, on peut avoir aussi bien quelques synonymes épars (nationalités, SICAV) que plusieurs centaines (occupations), voire plusieurs milliers (2000 pour la profession).

Par ailleurs, des négociations sont en cours pour élaborer des bases de connaissances en vue du codage de la profession dans le recensement américain, ce qui démontrera l'indépendance du logiciel par rapport à la langue de traitement.

2.3. Méthodologie

Pour mettre au point les connaissances dans de bonnes conditions, une méthodologie d'utilisation a été constituée petit à petit. La documentation méthodologique, qui a découlé de ces réflexions, aborde des sujets variés : description détaillée du fonctionnement du chiffrement automatique avec SICORE, principes de contrôle de cohérence, détection d'erreurs dans une base de connaissances, choix des connaissances à modifier, mesures de qualité, mise à jour des connaissances au fur et à mesure du traitement de l'enquête, codage de libellés hétérogènes,

La méthodologie ne doit surtout pas être considérée comme une chose définitive et figée : SICORE a dégagé un certain nombre de principes généraux, mais il est probable que l'on en découvrira de nouveaux, ou que l'on en viendra à amender certaines règles existantes.

2.4. Organisation

SICORE nécessite l'existence d'une organisation pour deux raisons bien distinctes : d'une part, pour faire en sorte que les bases de connaissances soient "vivantes", c'est-à-dire évoluent et s'enrichissent régulièrement ; d'autre part, pour être prêt à assurer la mise en place du chiffrement automatique dans une enquête ou une source quelconque.

¹³Test qui a nécessité un serveur doté de 2 gigaoctets de mémoire centrale.

¹⁴Il y a souvent, en pratique, plusieurs libellés possibles pour la même commune, par exemple "ISSY LES MOULINEAUX" et "ISSY LES MLX".

Pour le premier objectif, SICORE a créé la notion d'*expert de variable*, dont le rôle est de mettre au point les bases de connaissances relatives à la variable en question. Pour cela, chaque expert dispose de *l'outil SICORE*, sur PC. L'ensemble des variables devant faire l'objet de ce travail d'expertise n'est pas fermé : il est destiné à évoluer. Le projet SICORE a commencé avec les variables "Profession" et "Commune".

Mais la présence d'experts ne suffit pas. Pour avancer, ceux-ci doivent pouvoir se nourrir d'autres expériences, provenant d'autres enquêtes que celles qu'ils connaissent, ou bien de l'expérience des personnes de terrain. C'est pourquoi des *groupes de travail par variable* se sont créés, autour des variables Commune et Profession. Ces groupes se réunissent deux à trois fois par an ; il est probable qu'ils évolueront, aussi bien dans leur composition que dans leur périodicité de réunion, ou leurs objectifs.

Enfin, pour qu'une "culture d'expert de variable" se crée, il existe également un *club utilisateurs* de l'outil SICORE. Il ne réunit que les experts (seuls utilisateurs de ce logiciel), et permet de faire le point sur les éventuelles lacunes de l'outil, et donc de faire remonter les spécifications.

Tous ces réseaux (groupes de travail par variable, club utilisateurs) sont animés par *l'expert SICORE*, membre de l'unité de méthodologie de l'INSEE, qui est la plaque tournante de l'ensemble de l'organisation.

Le deuxième objectif, c'est-à-dire la mise en place du chiffrage automatique dans une application donnée, nécessite une organisation complètement différente, à nouveau centrée autour de l'expert SICORE. Il s'agit ici de mettre en correspondance trois entités : l'enquête (représentée par un responsable statistique et un responsable informatique), SICORE (représenté par l'expert SICORE et le responsable informatique de SICORE), et la variable à coder (représentée par l'expert de variable).

La difficulté de cette mise en place varie énormément selon la variable et selon l'enquête considérée : il arrive que cela ne prenne que quelques jours, mais il se peut aussi que ce soit long, notamment quand les bases de connaissances n'existent pas déjà.

3. Utilisation de SICORE en production

3.1. Pratique de SICORE

SICORE n'a rien d'un outil clé en main, et ce serait d'ailleurs impossible pour un outil général : ce ne sont que des *modules élémentaires de codage*, qu'il faut intégrer dans une application, et auxquels il faut joindre les bases de connaissances. Ainsi, le module de codage automatique (proprement dit) reçoit-il, en entrée, un libellé additionné d'éventuelles variables annexes ; en sortie, il renvoie un résultat de codage et un code retour.

Mais ce n'est pas tout : outre l'intégration des modules élémentaires, le codage automatique engendre plusieurs travaux, en amont et en aval, dont on ne doit pas sous-estimer l'ampleur. Ainsi, l'on doit prévoir la saisie

des libellés, dont le coût n'est pas négligeable, quelque soit la sophistication du poste de saisie. Il faut également écrire des logiciels conviviaux pour le traitement des "rejets" du codage automatique, c'est-à-dire des outils de codage assisté. Enfin, la mise en place du codage au niveau statistique nécessite un minimum de préparation ; dans le cas de "nouvelles" variables, cela conduit à la création de bases de connaissances ex-nihilo, ce qui peut s'avérer long (cela dépend en fait de l'existence ou non d'une première nomenclature reconnue).

3.2. Applications de SICORE

SICORE a été appliqué à l'enquête sur les transports dans l'agglomération lyonnaise, appelée SYTRAL, dans laquelle il fallait coder des **communes**, en l'occurrence des points de départ et d'arrivée pour chaque utilisation d'un transport en commun. Sur quelques milliers de libellés, seuls trois n'ont pas été codés par SICORE : le chiffrage assisté n'a donc pas été véritablement nécessaire.

Il est difficile d'en tirer des conclusions sur un plan statistique, car le chiffrage de la commune n'est certes pas le plus difficile. En revanche, informatiquement, on a pu voir que dans ce cas (où il n'y avait pas véritablement d'intégration dans une chaîne de traitements), le temps de travail nécessaire était très minime (un à deux jours).

SICORE a également servi à chiffrer **la CS (variante de la profession) et les lieux de séjour** dans l'enquête Permanente Conditions de Vie de l'INSEE. La situation était idéale pour une évaluation de SICORE, puisqu'un codage manuel avait été fait au préalable.

La codification des CS a porté sur 9992 libellés, celle des lieux de séjour sur 12239. **SICORE a codé automatiquement 76% des CS et 92% des lieux de séjour**¹⁵. Il restait à savoir si les codes SICORE étaient "bons"¹⁶. Pour évaluer cela, une technique s'imposait : rechercher dans combien de cas le code automatique différait du code manuel, puis analyser une par une ces divergences.

Dans les cas des lieux de séjour, lorsqu'on étudie les écarts entre codage automatique et codage manuel, on s'aperçoit qu'il y a très peu de différences : seulement 3% des libellés (315 exactement) codés par SICORE avaient des résultats distincts du codage manuel. En étudiant un par un les libellés en question, on s'aperçoit que **le code SICORE est juste dans 82% des cas, et le code manuel dans les cas restants**.

Parmi les erreurs de SICORE, citons l'exemple de "CANARIES" (qui n'était pas dans le fichier d'apprentissage), codé comme la ville corse "CANARI".

Le codage d'une CS est en revanche une opération plus délicate, ce qui implique qu'il y ait beaucoup plus de divergences entre codage automatique et codage à la main : parmi les intitulés de profession codés par SICORE, plus de 1 sur 6 (18%) divergent du code manuel. La question se pose à nouveau : qui a raison ? Les différences de chiffrage de CS ont donc été analysées : les experts Profession s'en sont chargés.

¹⁵C'est-à-dire lieu de vacances, décrit sous la forme d'une ville, d'un département, d'un pays étranger, d'un circuit, ...

¹⁶En effet, il ne faut pas espérer d'une méthode de codage automatique que *tous* ses chiffrages soient exacts : le seul moyen, pour cela, serait de ne coder que des libellés strictement identiques à ceux du fichier d'apprentissage (et encore, il peut subsister des erreurs dans celui-ci ...).

L'étude des divergences montre une nouvelle fois que c'est plutôt le codage automatique qui donne les meilleurs résultats : dans 62,5% des cas, le code SICORE est le bon code. A l'inverse, le codage manuel a raison dans 17,5% de ces cas de divergence. Enfin, dans les 20% restants, les deux codes peuvent convenir : il y a ambiguïté ; celle-ci est souvent due à la non-utilisation du libellé d'activité en clair.

Bien entendu, les principes de base de SICORE ont été immédiatement mis en pratique : toutes les erreurs de codage ont été réanalysées une par une, de même que les libellés non-codés, ce qui a conduit à améliorer les bases de connaissances en conséquence.

Du côté des applications de SICORE à l'extérieur de l'INSEE, citons le codage de la CS au sein du CEREQ, organisme français chargé de statistiques sur les qualifications, mentionnons également les négociations actuellement en cours avec le bureau du CENSUS américain pour l'emploi de SICORE pour le recensement fédéral.

3.3. L'intégration de SICORE

Une fois réglées toutes les questions de mise au point des *connaissances SICORE* par l'*outil SICORE* sur PC, il faut déterminer de quelle façon l'on souhaite utiliser SICORE.

Pour une réussite optimale, l'enquête utilisatrice de SICORE doit au préalable fixer la méthode d'intégration, ce qui revient en fait à arbitrer le couple "degré d'intégration" "personnalisation du codage automatique".

Les deux méthodes proposées dans l'intégration sont en quelque sorte les deux extrêmes de cet arbitrage :

- l'intégration des briques SICORE constitue un moyen très souple d'intégration car cette méthode permet d'ordonner et de maîtriser les appels à SICORE, par exemple en travaillant simultanément sur plusieurs variables. En revanche, elle exige que l'application ait une forte adhérence à SICORE, notamment de par son implémentation. Il ne faut également pas perdre de vue que les compétences requises sont celles d'un informaticien maîtrisant le langage C¹⁷,
- l'intégration des traitements SICORE, quant à elle, permet de ne pas lier les traitements de SICORE avec ceux de l'applicatif, puisque seul l'enchaînement des traitements est impacté. Cependant, il ne peut s'agir que de traitements prédéfinis dont l'unité indivisible est le fichier, ce qui ne lui confère pas beaucoup de souplesse. La compétence nécessaire est du niveau utilisateur ou utilisateur confirmé, capable d'enchaîner les étapes d'un travail (par exemple grâce à la maîtrise du JCL dans le monde MVS). Il faut alors que les traitements sur les fichiers (normalisation et codage) soient disponibles sur la plate-forme envisagée pour l'exécution.

L'important pour faire le choix est d'avoir en tête les tenants et aboutissants de chaque solution, sachant qu'il est toujours possible de faire un choix pour une partie de l'application et un autre pour une deuxième : tout n'est alors qu'affaire d'organisation.

3.3.1. L'intégration des briques SICORE

Comme nous l'avons vu, il s'agit du cas où l'adhérence de SICORE à l'application est la plus forte : le programmeur à qui il revient de finaliser cette intégration dispose :

- des fichiers contenant les déclarations de toutes les fonctions et constantes symboliques (par exemple pour identifier les codes-retour) nécessaires à l'utilisation de SICORE,
- des fichiers contenant les modules-objet nécessaires pour l'édition de liens de SICORE avec l'application,
- de la documentation d'intégration sous forme HyperText qui le guidera dans les étapes d'intégration.

Les briques correspondent aux fonctions classiques que nous avons déjà rencontrées, à savoir principalement la normalisation et le codage. Cependant, il faut en général se préoccuper du chargement en mémoire des bases de connaissances, chargement qui à l'exécution occasionnera une consommation en temps et en mémoire centrale¹⁸. A titre d'exemple,

¹⁷ Il est possible d'écrire des applications en un autre langage, et qui font appel à des sous-programmes écrits en langage C, avec plus ou moins de difficultés suivant les plate-formes, mais au prix d'une technicité plus pointue.

¹⁸ En effet, pour des raisons de performances, le choix a été fait de faire résider en mémoire toutes les données nécessaires aux traitements, évitant ainsi les lenteurs des périphériques d'entrée-sorties.

le chargement de la base de connaissances complète concernant la commune occupe 11 méga-octets en mémoire et demande 20 secondes sur un pentium 90, alors que ces chiffres sont respectivement de 8 méga-octets et 10 secondes pour la base complète de la profession. On aura donc intérêt à minimiser le nombre des chargements de bases de connaissances, autant que possible.

Il existe cependant un cas dans lequel on peut s'affranchir du chargement de bases de connaissances : dans le prolongement du projet SICORE, l'INSEE a décidé de développer un serveur de codage dans l'environnement du moniteur transactionnel CICS d'IBM. Le fonctionnement en est simple : une transaction serveur effectue le chargement d'une ou plusieurs bases de connaissances une fois pour toutes, les transactions-clientes programmées par le service-utilisateur¹⁹ interrogent directement ce serveur sans avoir à réaliser les chargements de données. Ce serveur n'est pour l'instant disponible que dans le monde MVS, mais permet cependant de répartir les bases de connaissances sur plusieurs noyaux CICS interconnectés, de sorte qu'une transaction-cliente s'exécutant sur un noyau peut très bien interroger une base qui se trouve en fait sur une autre machine MVS, et ce sans le savoir puisque c'est SICORE qui prend intégralement à sa charge le routage de la requête et de la réponse.

¹⁹ Là aussi avec l'aide d'une documentation en ligne.

3.3.2. L'intégration des traitements

Il s'agit dans ce cas de traiter des fichiers de données. Une étape consiste en la normalisation ou le codage d'un fichier pour une variable donnée. Comme précédemment, un chargement de base de connaissances est nécessaire, mais cette fois pour chaque fichier à traiter. L'avantage de cette solution est cependant la simplicité de mise en oeuvre puisqu'il suffit de peu d'informations pour réaliser les traitements (définitions de dessins pour déclarer la position des informations actives). Ces traitements sont en fait constitués de petits programmes appelant les briques SICORE de base. Du fait que ces programmes réalisent des opérations non-portables (prise en compte des paramètres et restitution de statistiques), ils ne sont pas portables (au contraire des briques) : il faut donc les développer pour chacune des plate-formes d'exécution. Cependant, d'une part leur simplicité les rend facile à développer (surtout avec l'aide de la documentation d'intégration), d'autre part il existe des versions déjà écrites dans le monde MVS²⁰.

3.4. Mise à disposition de SICORE

Chaque unité d'acquisition de SICORE comprend :

- l'*outil SICORE* (environnement PC),
- la documentation méthodologique détaillant toute la méthodologie SICORE actuelle, disponible à la fois sur papier et sous forme de documentation HyperText,
- le glossaire SICORE reprenant l'ensemble des termes propres à la méthodologie SICORE, disponible à la fois sur papier et sous forme de documentation HyperText,
- la documentation de l'utilisateur de l'outil SICORE pour manipuler correctement l'outil, disponible à la fois sur papier et sous forme de documentation HyperText,
- la documentation d'intégration de SICORE décrivant les modes d'intégration de SICORE dans des applications, disponible uniquement sous forme HyperText.

Sous réserve de négociation complémentaire avec l'INSEE, peuvent être proposés :

- une formation à l'outil SICORE,
- les bases de connaissances déjà réalisées par l'INSEE pour ses propres besoins, mais qui peuvent intéresser d'autres organismes,

²⁰ Sur PC, l'outil SICORE peut très bien réaliser cette tâche. Il n'est cependant pas souhaitable de confondre l'outil SICORE avec un outil de production. L'INSEE s'interroge actuellement sur l'opportunité de développer un outil bridé ne reprenant du poste de l'expert que les fonctionnalités de production.

- une aide à la constitution de bases de connaissances spécifiques à un organisme faisant l'acquisition de l'outil SICORE,
- la fourniture des modules de SICORE pour son intégration dans une application.

The Application of Blaise III to the Israel Labour Force Survey

Edith Noy and Gad Nathan, Central Bureau of Statistics, Jerusalem, Israel

1. Introduction

The Central Bureau of Statistics has only recently started to automate its collection processes. In the last two years the transfer of the telephone collection of monthly industrial establishment data to an automated process was implemented successfully, by means of locally developed software. This was an application to the continuous collection of data, at three regional centres, from a panel of respondents with a fixed questionnaire and it did not present any special problems. Following this success, a decision was reached to extend the automation of the Bureau's collection processes in two different areas. One is the collection, by CATI and CAPI, of price data from outlets for the consumer price index, presently under development. The second, to be described in the following, is the application of automation to the collection processes of the most important and central household survey system of the bureau - the Labour Force Surveys.

The Central Bureau of Statistics has carried out Labour Force Surveys since 1954. These surveys are the source of official government statistics on employment and unemployment. In addition, these surveys also provide data on demographic characteristics, such as age, sex, marital status, educational attainment, family relationship, occupation and industry. Occasionally, additional questions are asked on health, education, income, previous work experience and other topics. The statistics obtained from these questions are used to update similar information collected through the decennial census and from other sources. They are also regarded by government policy makers as important indicators of the nation's economic situation and are used for planning and evaluating many government programs.

A major redesign of the survey was initiated in 1995, primarily to improve the quality of the data derived from the survey. This process is being implemented in two stages:

- 1) Redesigning the questionnaire, primarily aimed to measure the official labour force concepts more precisely and to incorporate definition changes, according to the recommendations of the International Labour Organisation. s persons who had worked for at least one hour during the

determinant week, at any type of work, for pay, profit or other remuneration and Unemployed Persons as those who had not worked for even one hour during the determinant week and who had actively sought work during that week, by registering with the Labour Exchanges of the Employment Service or any other employment office. Finally all persons aged 15 and over, who were neither "employed" nor "unemployed" during the determinant week, are defined as Not Belonging to the Civilian Labour Force.

2. The Labour Force Survey - general description

2.1. Definitions

The sample population includes the permanent (de jure) population of Israel aged 15 and over, including the Jewish residents living in Judea, Samaria and the Gaza Area, potential immigrants and permanent residents living abroad for a period of less than one year. It does not include tourists and temporary residents, unless they have been living in Israel continuously for more than one year.

The Civilian Labour Force is defined as persons aged 15 and over who were "employed" or "unemployed" during the determinant week, Employed Persons are defined a.

Towards the next pilot test, we shall attempt to develop the treatment of partially filled out questionnaires and the transition from the questionnaire of one person to another one, in the same household. In addition, we shall work on developing the mode of interviewing separate household units in the second wave in cases where households have split or new households have joined since the first wave, when only a single unit was interviewed.

Since we have realised that the transition from the dialling stage to the interview was not flexible enough, we decided to change the standard mode suggested by Blaise and to adapt it to our needs.

In general, it is clear that there are significant potential benefits from moving to computer assisted methodologies, in terms of maximising the opportunities for better and faster data collection, and also possibly long-term cost benefits for a large survey, such as the Labour Force Survey. We expect that in the next few years, our office will be able to take full advantage of the CAI.

2.2. Sampling Methods

In the present format, four quarterly surveys are conducted each year, the interviewing for each quarterly survey being spread over the entire quarter. In each survey approximately 12,000 households are sampled and about 22,000 different households are interviewed over the year.

Two types of frames are used to select the sample: (a) a frame of localities, and (b) frames within localities. In urban localities and in some rural localities, the frames are the lists of dwellings in the municipal tax file.

In other localities the frames are lists of households, of dwelling units or of individuals.

The sample is drawn in two stages. In the first stage, localities are sampled from the Bureau's file of localities. In the second stage, in most localities, dwellings from the sampled localities and in some of the rural localities, households are sampled. The sampling is done in such a way that the final probability of being included in the sample is the same for every household in the population (slightly more than 0.7%).

The sample which is selected once a year, is divided into four groups or panels (waves). The interviewing of each of the four panels commences in consecutive quarters. Each panel is investigated four times - two investigations during two consecutive quarters and then, after a break of two quarters, there are two additional investigations during two consecutive quarters.

Thus, in each quarter, the sample is composed of four panels. The method of investigation by panel was devised for the purpose of providing good estimates for the differences between: consecutive years, consecutive quarters and parallel quarters from consecutive years.

2.3 Interviews and questionnaires

The collection is conducted by trained interviewers of the Central Bureau of Statistics who visit every dwelling in the sample. Dwellings not used for residential purposes, empty apartments, businesses etc., are considered as out of scope ("zero cases").

In residential dwellings, one of the household members is interviewed with respect to all members of the household. For households where no one is at home on the first visit, the interviewer usually makes two subsequent visits. If also on these visits no one is at home, special questionnaires are usually left to be filled in and returned by post. For each visit resulting in non-interview, the interviewer indicates the reason in a special questionnaire.

In urban localities the interviewers were asked to obtain the permission of households with telephones to conduct the second and third stage interviews by telephone. In 1995, about 52% of all interviews were conducted by telephone (about 87% of the interviews in the second and third investigations).

Interviewing is carried out continuously during each week for the entire three months of the survey. During each week, about 1/13 of the households included in the survey are interviewed. The "determinant week" in this period always refers to the "previous week," namely the week ending on the Saturday prior to the visit of the interviewer. The data obtained for any period (quarter, year, etc.) are intended to reflect the situation of an "average" week in this period.

The questionnaire used in the current survey is a paper and pencil questionnaire. For each household one questionnaire is completed

containing information pertaining to the entire household and one for each member aged 15 and over.

The questionnaire for an individual includes questions on work in the determinant week, number of work hours in general and in the determinant week, number of work hours less than usual in the determinant week, number of overtime hours, reasons for part-time work, reasons for absence from work during the entire determinant week or a part of it, place of work, geographical mobility of all employed persons, type of work and employment status. In addition, information is obtained on unemployed persons on search for work, search for full-time or part-time work, reasons for unemployment, whether the individual ever worked in Israel and when, and the previous occupation. For employed persons, the amount of time worked during the year and reasons for working only part of the year are also investigated. Those persons not in the civilian labour force but who had worked during the year preceding the interview, are asked about the last type of work they have done. Those not in the civilian labour force and who had not worked during the year preceding their interview, are asked why they did not work.

Apart from details about work, demographic information is also collected: age, sex, marital status, country of birth, period of immigration, level of education (number of years of schooling and type of last school attended). The household questionnaire includes information on the number of persons in the household, the number of rooms in the dwelling, the number of children in the household and the number of hours of work of any paid domestic help.

Besides the regular questions about work, and household and demographic characteristics, the questionnaire occasionally relates to other matters, such as housing conditions, domestic appliances owned by the household and various other subjects.

The inclusion of these subjects in the survey is intended to provide statistical material for a detailed investigation of the connection between work patterns of the household and other areas of behaviour. Also included occasionally in the survey are questions related to specific aspects of work such as seniority at work, labour mobility and so on. They are intended to provide further information on both household and individual behaviour in the labour sphere.

2.5 Processing the data and estimation

The collected data are checked, completed, edited, coded, punched and undergo logic editing - all at a central location.

The data are weighted, by age, sex and type of locality group, based on estimated total population of each group. The annual estimate is the average of the four quarterly estimates.

3. The redesign of the collection process

The preparations for the redesign of the data collection methods, began in 1996 and are presently continuing. The major feature of this redesign is the automation of the collection process. The revised questionnaire is now being designed for a computer assisted interview. The questionnaire, as it is now being designed, will most likely serve for a long period before it is redesigned. Therefore, a large amount of effort is being given to considering alternative questionnaire designs. Issues such as question phrasing, screen layout, computer assisted coding and editing methodology, must all be considered.

The possibility of developing a custom-made software for automated collection was considered, but quickly rejected. Given the late entry into the area of computer assisted interviewing and the high stage of development of ready-made software, development of our own software was considered as too costly and time-consuming. The only potential advantage of self-development was that it could potentially overcome more easily the inherent problems in the use of Hebrew (written from right to left) for interviewing. However after some initial testing of the Blaise system, the transfer to the use of a Semitic language was shown to be feasible, though not without its problems - see further detail below.

No systematic testing of the feasibility of various alternative software packages for computer assisted interviewing was carried out. However after a perusal of the literature and based on personal experience and demonstrations of the alternatives, the Blaise system was selected as a first choice for testing. Since there was no rigid deadline for automation, the perceived advantages of Blaise were considered sufficient to test for feasibility, with the possibility of turning to an alternative should it fail to perform as required.

The major change in collection procedure envisaged is that interviewers will conduct the survey, either in person at the respondent's home, using laptop computers, or by telephone, calling from a centralised location, using personal computers. The Bureau is looking forward to the potential benefits of the computerisation, mentioned in the research literature, which provides information on computer assisted interviewing. Primarily, the move to computer-assisted collection is expected to yield the following advantages.

Currently we are burdened by the time-gap between the collection of the survey data and its release. We expect that the new system will help to reduce this lag considerably, since it simplifies the process of collecting and transmitting data. In addition, work that is currently carried out after the termination of the collection of data will be done during the data collection stage and expedited.

Also, we hope that use of this system will aid in attaining data that is higher in quality through various checks, incorporated into the interview process. Errors detected by the computer at this stage can be corrected with the aid of the respondent. This capability is made possible through built-in editing features, verification procedures and automatic consistency checks.

In addition, we expect the automation of the questionnaire to aid the interviewer in following the correct line of questioning, i.e., skipping when necessary, etc. and moving the respondent smoothly through the questionnaire. We are confident that computerising the questionnaire will aid in reducing misclassification and other response errors. Questions will be tailored automatically to the respondent through insertion of appropriate pronouns and changing verb tenses and also by omitting irrelevant questions on a case-by-case basis.

4. The implementation of the Blaise system

At the beginning of 1996, a four-member team was set up to design and to implement the strategy for developing and testing computer assisted interviewing for the Labour Force Survey. This team consists of one representative from the Labour Division, one representative from the Field Division and two programmers from the Information Systems Division. Its work is guided by a steering committee of senior staff from these divisions, headed by the Bureau's Chief Scientist.

The basic strategy consisted of a continuous programme for testing and implementing computer assisted interviewing for the Labour Force Survey collection, starting with CATI for the portion presently interviewed by telephone. In the process of converting the questionnaires to telephone computer assisted interview, the Blaise case management system for CATI will also be introduced. Once the CATI system becomes operational, the field interviewing will be transferred to CAPI, with experiments continuing in parallel with those for CATI.

The programmers were responsible for designing the Blaise application of the questionnaire and for implementing the CATI Management System. At the start of the implementation, they studied the Blaise system, (from manuals) and functioned with a certain measure of success. When they encountered problems, they sought assistance from Blaise advisors via e

mail. When problems arose more and more, solving them via e-mail became arduous and time consuming. As our timetable was already set for the first test of the Blaise system, we invited a Blaise advisor from Holland to provide staff support in promoting the implementation. The advisor also distributed an updated and more detailed, "Developer's Manual" which aided the programmers greatly in their work in the Blaise system.

Two main types of difficulties were encountered while designing the questionnaire - those of the programmer and those of the interviewer. Following are a few examples of the problems encountered, of each type.

1) Difficulties for the programmer :

Most of the programming difficulties related to the use of Hebrew. Thus, on the question screen, the default position for the code numbers and the text for the possible answers is left aligned (numbers first). In Hebrew, it is necessary to right-align, with the numbers first (on the right). It was not possible to change the position of the code numbers, but we could

manually align the text to the right. This caused the numbers to be at unequal distances from the text, which makes it difficult to choose the right answer.

The error screens, which are in English, had to be translated into Hebrew. To do this, our programmers had to delve into the system codes and change them one by one, which makes for excruciating work.

Many problems were associated with the use of tabular entry. For the interviewer's convenience, we would like to consolidate all the responses to questions on demographic subjects in one table. This way, all questions on each person in the household can be asked and recorded consecutively. Due to the size limitations of the screen, the responses scroll beyond the width of the first screen and it is not clear which response belongs to which household member. To overcome this problem, first names are presented again on the second screen. However, on the second screen, we were unable to determine which responses would appear or to align the names on the left side of the screen. Currently, some of the responses appear on both the first and second screen (before the name).

When a closed question is asked, the answer is numerical. When we wish to present the answers in a table, it is difficult for the interviewer to use the table with no text. If we wish to insert the text into the table, we must create an additional column. We cannot refrain from presenting the numerical answer because of Blaise limitations. This creates a problem because of the limited screen space for the chart.

By Blaise default, each question is in a separate screen. In certain cases, we need to organise more than one question on the same screen. We were told by the Blaise advisor that this could be achieved, but it would be a difficult task. The default layout of the screen does not always meet our demands. In order to adapt it to our needs we must create different layouts, which involves a great deal of work.

Additional difficulties arose in adapting the system to the transfer of data from an external source (the first round of interviewing - carried out by field interview) to the telephone interview, via MANIPULA. The difficulties were mostly technical in nature and related to problems such as missing system files and unexplained error messages.

2) Difficulties for the interviewer :

Again many of the problems are associated with the use of the Hebrew language, as Blaise was created for languages which are written from left to right, while Hebrew is written from right to left. By default, the text is in English. In order to change to Hebrew, Ctrl + Tab must be pressed. Numbers are considered as English text, so every time a number is typed, the program enters English mode. In order to continue typing in Hebrew the interviewer has to switch to Hebrew. The same problem occurs when the "Enter" key is pressed.

If an error in an answer to an open question must be corrected, it is not possible to switch to Hebrew with Ctrl + Tab. Instead, the keys Shift +

Backspace must be pressed. It is complicated for an interviewer to cope with these different functions while conducting the interview. In addition, complications arise when answering an open question in Hebrew, unless one first opens the answers window using the space-bar, which is not necessary in English.

5. The testing program:

In order to test the impact of transferring the Labour Force Survey to computer assisted interviewing, it is proposed to conduct a number of small scale tests prior to a large-scale test. The outcome of these tests, will provide the basis for the decision as to whether a parallel survey is necessary.

Beginning in January 1997, a number of small-scale tests are being conducted using CATI. The tests involve the transfer of the questionnaire onto the CATI instrument and the CATI management. A total of some 80 households is indicated for the tests. These households will be interviewed over a six month period. It is proposed to analyse the results from the test, every two months, to the extent that the system has been implemented at that stage, and based on the conclusions, to continue the process.

During the first wave (carried out at the end of 1996), the interviewers conducted face-to-face interviews using paper-and-pencil questionnaires. During the second wave, the same households will be interviewed again, by telephone interview, using computerised questionnaires. The telephone interview is conducted from one centralised location, using personal computers. The CATI case management has been introduced, but the interviewers still dial the phone numbers by themselves, due to lack of appropriate equipment. The first part of this experiment has been concluded successfully and, though indicating some problems, shows that overall the CATI system works.

A larger-scale operational test, with a sample of approximately 2,500 households, is planned for the end of 1997. This test will be based on a full probability sample, to allow comparison of its results with those of the current paper-and-pencil collection and a fully developed CATI instrument. In 1998-99, an additional test will follow to begin with face-to-face interviews, using laptop computers instead of paper-and-pencil questionnaires.

The larger-scale operational tests will provide the necessary criteria regarding the functionality of the Blaise system according to our requirements and will indicate if the system is easy to use by survey programmers, by developers and by interviewers. Other issues will also be checked, such as potential effects on interviewers and permanent staff, effects on respondents, response effects, data quality, timeliness, costs, etc.

A decision whether to conduct a parallel test will be reached later on. Testing experiences of the US Census Bureau have shown that CAI produces comparable results to paper-and-pencil collected data and the

agency suggests that statistical impacts of CAI alone are not significant. In our case, as the revised questionnaire was introduced prior to CAI, a serious break in series is not expected with the implementation of CAI. Given the likelihood that the impact to be measured would be small and the assessment that a parallel test would be very expensive, probably no parallel test will be conducted.

6. Conclusions

As mentioned before, our Bureau is only at the beginning of the computerisation of the data collection in the Labour Force Survey. Our experience is limited to the design of the questionnaire and the implementation of the CATI management. Although we have already performed a few real interviews, beginning in January 1997, they were done by the planning team and not by the survey interviewers; so that we do not have the interviewers' opinions on using the new system. As the interviews were conducted from the central office, the transmission of the collected data has yet to be checked.

Despite the early stage of our project, we are pleased to have already a basis for the computerised questionnaire, which became possible due to the Blaise system. We are confident that we will be able to overcome problems that will arise in the process of developing our project. Our main concern is the difficulties in the transition from English to Hebrew. We hope that these difficulties will be solved by the adaptation of the Blaise system to Hebrew, or by its introduction into Windows.

In order to partially overcome the problems arising from writing in Hebrew, we plan to reduce as much as possible the writing of text and the use of remarks. Also, there will be no writing in Hebrew and English texts together in one answer. This stage was completed in 1994 and the new version of the questionnaire is fully implemented in the field since January 1995, as an operational paper-and-pencil questionnaire.

2) Redesigning the collection methods, begun in 1996 with experiments in the use of computer assisted interviewing, initially for telephone collection, using Blaise III. This will be described more fully in the following.

Making Blaise 2.5 Do Things It Was Never Meant to Do

James M. O'Reilly, Research Triangle Institute, USA

Abstract

Blaise 2.5's fundamental architecture allows direct access to the most elementary interviewing functions and processes. This allows one to extend 2.5's functionality in important new ways. The paper discusses the Blaise 2.5 architecture and describes how, for a methodology study, we added a keystroke capture mechanism. The system saves every keystroke entered during an interview, both human-readable and special PC keystrokes. In addition to the keystrokes the system save a time stamp to the hundredth of a second each time the [Enter] key is pressed. The keystroke stream is saved to a separate data file periodically.

1. Introduction

Blaise 2.5 has limitations for some applications, and it lacks many of the impressive powers of Blaise III. Still, 2.5 remains widely used. Some are reluctant to change proven legacy applications. Others continue to develop new 2.5 applications because Blaise 2.5 is adequate to the task, because it may more efficient to build on the already established staff knowledge and experience, and other reasons.

Blaise 2.5 also possesses another "secret" feature which may make it useful in ways that Blaise III is less capable. Blaise 2.5's fundamental architecture allows direct access to the most elementary interviewing functions and processes, allowing one to extend 2.5's functionality in important new ways. In the Blaise 2.5 system various standard modules of Turbo Pascal code can be changed by the programmer so that when the application is parsed and compiled entirely new functions can be implemented.

The paper discusses the Blaise 2.5 architecture and describes how, for a methodology study, we added a keystroke capture mechanism. The system saves every keystroke entered during an interview, both human-readable and special PC keystrokes. In addition to the keystrokes the system save a time stamp to the hundredth of a second each time the [Enter] key is pressed. The keystroke stream is saved to a separate data file periodically.

2. Why abandoning Version 2.5 may not be wise

Blaise III is clearly the most powerful, comprehensive, and elegant computer-assisted interviewing system available today. It is a major leap forward from its predecessor Blaise 2.5, not to mention the competitive products which were, and are, inferior to Version 2.5 in many significant ways.

Blaise III has expanded dramatically the framework for computerized interviewing system to become the first true “survey processing system”. Blaise III’s object-oriented architecture, advanced meta data framework, and integration of the interviewing/data entry process with the surrounding components of survey processing provide survey systems designers and developers with a much richer, more powerful, and more flexible environment.

Blaise III is clearly the system of choice for serious DOS-based survey processing projects. At the same time it is important to point out that there may be situations where Blaise III is not the superior choice over Blaise 2.5. Among these might be :

- where existing Blaise 2.5 applications are performing properly,
- where “legacy” DOS hardware must be used (Blaise III can be quite slow on older x86 processors and on machines no extra random access memory (RAM)), and
- similar situations where the significant investment in software licenses, learning the new system, and retooling existing processes cannot be justified yet.

3. Limitations of 2.5 and user work-arounds

Blaise 2.5, as with any application, was developed to meet the realities of the computer environment of the time it was designed and built. This meant the world of DOS, 640k of programmable memory, Turbo Pascal, character screens, and no doubt many other constraints.

The resulting system has limitations which, as clients have expanded the scope of what they want done in computerized surveys, have become obstacles. Among these are, for a single instrument: a maximum of 2,000 items, 64k limit on the amount of question text, 64k limit in the size of enumerated answer sets, only one language can be used, and others.

One of the great frustrations of many computer experts and gurus is that, in their view, MS-DOS is a profoundly flawed, out-of-date, makeshift crime against all known principles of operating systems design. And it should have been strangled at birth, or at least buried permanently ten years ago. While the technical logic of the gurus is not seriously questioned, the market and the ingenuity of users seeking to solve their immediate problems with the tools and resources available have failed to pay attention to the dicta of the experts.

Countless work-arounds, adaptations, extensions, and other ad hoc strategies have been found to make MS-DOS do things it was never thought to be capable of doing. A similar situation has occurred among Blaise 2.5 users. As limits have been reached, creative solutions have been found to extend the system in new ways. A number of these strategies have been reported in Blaise User Group meetings and in the user group's newsletter.

One example is the 2,000 question limit. A number of studies have successfully bypassed this restriction by converting their instrument into two or more separate sequential Blaise instrument in which the later sections read data from the prior ones using the valuable Blaise 2.5 external file access interface. At Research Triangle Institute we have recently built a survey made up of 13 separate Blaise 2.5. Each respondent actually encountered eight instruments. The first, seventh and eighth modules were done by all, while modules two to seven had two versions to respondents were randomly assigned. A similar multiple Blaise 2.5 architecture was used successfully in the National Survey of Family Growth (Duffer and Moser, 1996; O'Reilly, 1993).

The separate Blaise instruments are executed by a driver process—either a DOS batch file or for more complex requirements a custom application in FoxPro, C, or another language. Of course, work arounds have their costs in terms of the extra effort to structure all the components to work properly together. And Blaise III eliminates the need for many of these work arounds, particularly those arising from Blaise 2.5 memory limitations. Blaise III used DOS extended memory so that most or all of the previous limitations no longer exist.

4. Version 2.5's peculiar architecture

The fundamental strengths of Blaise 2.5 and Blaise III derive from the creative and comprehensive design implemented by the Blaise team at Statistics Netherlands. Each of these two very different systems was designed to incorporate the most advanced and appropriate applications design principles of its day to a clearly conceived target of survey data processing. The power of this approach and the clear thinking, talented technical skills, and overall leadership that has supported it is, to this long-time user, most impressive.

One of Blaise 2.5's fundamental strengths is its speed of execution, even on 286-class PCs running on a floppy disk. When Blaise 2 was conceived execution speed as a key issue for applications. It was not something one could take for granted, as designers do today when they assume that users will have a Pentium-speed processor, 16 megabytes of memory and a very fast and large hard disk.

Blaise 2.5's execution speed is a function of the fact that a Blaise 2.5 application is a DOS executable process. Other systems then and now, including Blaise III, use an standard executable engine which interprets the application meta information and executes the application. Interpreted systems are significantly slower than executables. But, of course, this usually matters only on older, less powerful hardware.

Blaise 2.5's architecture has these components :

1. A parser which reads the Blaise application code, checks for syntax errors and, for an application with no errors, generates a series a Turbo Pascal source files and data files with the application's execution logic, data requirements, and question definitions.
2. A set of standard Turbo Pascal source files which handle various fixed elements of an application—getting keyboard input, managing screen display, handling data, executing the question flow.
3. The Turbo or Borland Pascal compiler which builds the DOS executable system from the application-specific and standard Pascal source files.

5. Extensibility and Version 2.5

The ability to extend a software system to implement specialized functions not in the core development system is an important goal of systems designers especially in recent years. In the Windows environment and web application development a major feature is the ability to utilize flexible and powerful objects using such technology as DLLs, ActiveX, Java, OpenDoc, COBRA. Blaise III includes a DLL interface to provide a capability to extend applications with specialized tools. This DLL interface offers far greater speed and integration with the standard Blaise application than those offered by competitive systems.

Blaise 2.5's open architecture of Pascal source files offers a non-standard, unsupported, and unorthodox avenue for extending survey applications. By altering these Pascal units and procedures one can change how a Blaise 2.5 application works in quite fundamental ways. In fact, this provides the 'backdoor' approach to extensibility with at least some functionality that an application developer in Blaise III cannot do either as well or possibly at all.

6. Caution

It should be clear that anyone doing this type of reverse engineering of Blaise 2.5 is doing so at his or her own risk. Statistics Netherlands is not responsible for anything that might result, and does not support developers who are, uninvited, making changing to the core processes of the system.

Similarly, neither the author nor Research Triangle Institute take any responsibility for any problems which others may encounter in manipulating Blaise 2.5 as described in this paper. Some valuable functionality can be created using these methods. But it should only be attempted by persons with the requisite programming skills and persons who take full responsibility for any results.

7. Reverse engineering Blaise 2.5

The fundamental approach used is to study the standard Pascal source files to understand what the system is doing and where. This approach is aided by fact that the Pascal is a strongly-typed, highly structured language which was designed originally for teaching and learning programming. As a result the code is more “wordy” and amenable to someone else puzzling through the purpose and details of the code than C and other languages with dense and cryptic syntax.

Another aid in learning how the interior of Blaise 2.5 works is the programmers source file comments. While the source code is not extensively commented, many helpful comments are included. Some are in English and some Dutch. Although we have not done it yet, at some point one would find useful a Dutch-English (or French, etc.) dictionary.

The table below shows the names of the Blaise 2.5 Pascal standard source files.

APPOT25.PAS	HELPIX25.PAS	RAUTLX25.PAS
BOMENO25.PAS	INCONO25.PAS	RDCHRX25.PAS
CALLT25.PAS	INFRMD25.PAS	REMRKX25.PAS
CHECKD25.PAS	INITD25.PAS	SCRNX25.PAS
CHECKP25.PAS	INITP25.PAS	SHOWQD25.PAS
CHSTRX25.PAS	INSESD25.PAS	SHOWQP25.PAS
CHUTLD25.PAS	LIPSP25.PAS	STARTP25.PAS
CHUTLP25.PAS	LOCKX25.PAS	STATT25.PAS
CODERX25.PAS	LSTUTX25.PAS	TEXTD25.PAS
CODUTL25.PAS	MARKP25.PAS	TEXTP25.PAS
DATA25.PAS	MEMX25.PAS	TOOLSX25.PAS
DIALT25.PAS	OVL25.PAS	UPDATX25.PAS
DIVET25.PAS	OVL25.PAS	USERPX25.PAS
EDITX25.PAS	PAGEX25.PAS	VIEWRX25.PAS
EDSTRX25.PAS	PARSX25.PAS	WNDWX25.PAS
FILERX25.PAS	QUTLX25.PAS	

From the DOS file name one can make an initial guess about what the file does—CHECK, DATA, EDIT, TOOLS, SCRNX, and so forth. In general, names ending in P25 are for CAPI applications, D25 for CADE, T25 for CATI, and X25 for general functions. Within the ‘X’ files different sections of code may be compiled different for CAPI, CADE, and CATI applications based on preprocessor directives.

For the work described here we have only dealt with the screen and data handling functions (SCRNX25.PAS and DATA25.PAS)

8. Keystroke data

Our most recent extension of Blaise 2.5 to add a function it doesn’t have in native form was to implement a keystroke capture capability. Keystroke files record every keystroke pressed by a user of an application—standard alphanumeric as well as specialized computer keystroke entries such as function keys, backspace, enter, and key-combinations like SHIFT-F4, etc.

Keystroke files are used in software usability testing generally and specifically in survey methodology studies to study what users are actually doing in the applications. Where is on-line Help invoked most often, how often does the user back up, change modes, delete answers and so forth. Usability testing of applications is of increasing importance as studies reveal that there is often a wide gulf between what application designers think they have provided to users and what users are able to successfully make use of.

Usability testing has not been traditionally employed in survey research. One reason is that many survey designers and programmers believe that their users—telephone and field interviewers, coders, and data entry operators--will be thoroughly trained on the system and then gain substantial on-the-job experience. So, the programmers conveniently assume, why worry; interviewers will be able to master the system, even if it lacks some polish and has some clumsy elements.

Yet studies from the Human-Computer Interaction literature suggest that even experienced users fail to use key features of an application, make significant errors, and develop an adversarial attitude to poorly designed applications. (Landauer, 1995; Shneiderman, 1996). Landauer (Chapter 10) cites efficiency differences of from 10 to 200 percent between applications developed using user-centered design methods and those developed with these methods.

Another impetus for usability testing in survey research is the increasing utilization of audio computerized self-interviewing (ACASI) for data collection (O'Reilly, et al, 1994; Tourangeau and Smith, 1996; Turner et al. 1996). In this mode users who may never have used a computer before are introduced to the system and guided through tutorial application. After a few minutes they are expected to manipulate the survey application entirely on their own. Learning where these users may be having trouble based on keystroke patterns and timing can be critical to identifying where the application must be refined.

There are many different techniques for usability testing—laboratory testing, observation, user debriefings, and others. Keystroke records provide an important additional method, one with more detailed information on some activities and, as well, one that may be less expensive and more broadly applicable than laboratory methods.

9. Implementing a keystroke capture capability in Version 2.5

For our keystroke system we wanted to be able to

- record all keys pressed by the user,
- record the timing to the hundredths of a second when the Enter key was pressed,
- identify the current question on the screen associated with the key series,

- save the keystroke series in a text file in a form that facilitated later analysis of the file.

The critical element is having access to the point at which each keystroke code is evaluated by the program. One cannot do this in Blaise III, CASES and most other Computer-Assisted Interviewing packages. Some, such as CASES, allow developers to save a 'trace' file which records every response to questions input by the user. While quite useful for testing, a trace file only save the state of the response when the Enter key is pressed. Backspacing, overwriting a mistake, and resuming cannot be detected along with other user actions.

In Blaise 2.5 adding this capability is not very difficult. In fact it is so simple that we can describe in a few sentences :

1. In DATA25.PAS define the array ABCDE of [1..200] char.
2. In the SCRNX25.PAS there is a GETKEY procedure. In GETKEY is a while-do-end loop in which the PASCAL READKEY function is called and the result is returned in the LETTER. If LETTER is not empty a key has been pressed.
3. Add PASCAL code to save the key code to ABCDE. When the Enter key is pressed add the system time to ABCDE followed by carriage-return/line-feed codes.
4. When ABCDE approaches its maximum size, open the DOS file named KSynnndd.DAT , append ABCDE to it, close it, and ABCDE load keys from the beginning.

The actual implementation has a few additional intricacies to make the file more useful. Still, the total effort was relatively modest, probably less than 25 hours by a skilled Pascal programmer. That, in itself, should suggest that flexibility and power of this approach.

The following section displays an example of the keystroke file for a recent study.

```

{*** 11/13/96 14:30:13.38 CURRSYS:ftest1e CASE_ID:6535 MODE:MI ***}
1{4}14:30:24.64
{7}14:30:25.24
{8}14:30:25.46
1{8}14:30:26.39
{9}14:30:26.89
22{10}14:30:27.82
2 3 43[8][8]74{22}14:31:09.29
1{12}14:30:28.59
[10]{13}14:30:32.55
[[ 6535]]
{*** 11/13/96 14:30:35.79 CURRSYS:ftest7f CASE_ID:6535 MODE:MI
***}
[10]{2}14:30:38.09
{2}14:30:38.70
[[ 6535]]
{*** 11/13/96 14:30:51.11 CURRSYS:ftest1e CASE_ID:6534 MODE:SKIP
***}
1{4}14:30:56.22
{7}14:30:56.44
{8}14:30:56.66
1{8}14:30:57.70
1{9}14:30:58.58
22{10}14:30:59.62
{12}14:30:59.95
1{13}14:31:00.78
1{14}14:31:01.71
{20}14:31:03.25
2{21}14:31:04.24
1{26}14:31:11.65

```

One can see some of the additional refinements we included to make viewing and parsing the file more convenient. These include :

1. A header record identify the date/time the current Blaise instrument was started and information on the application's name, the case ID, and mode. Because we did not know how to determine internally from Blaise all the information we wanted to embed in the keystroke file, we added a section in the SCRNX25 code. The current directory is checked for a file named CURRSYS.TXT. If found, its first line is read and added to the header line following the CURRSYS: tag. This allows us to add-in easily any annotation information we want in the keystroke file by writing to CURRSYS.TXT before calling the application.
2. Following the header, each line contains in order the series of keystrokes entered, followed by the number of the current question field inside of curly braces {}, followed by the system time.
3. Non-printable key-presses such as function keys are displayed in readable form within square brackets. So in the following line :

```
2/3/43[8][8]74{22}14:31:09.29
```

the [8] means the backspace was pressed twice to change the year of birth from 43 to 74.

4. At the end of the keystroke capture for each application the KEY field is identified in double square brackets.

10. Adding keystroke capture to Blaise III

If a keystroke capture feature is of significant value to some users, shouldn't this be implemented to Blaise III? We emphatically think so. As our survey applications become more elaborate and as we venture into new areas such as audio self-interviewing and other potentially valuable new methods, the need to understand what the users are doing or failing to do become much more important. A keystroke capture capability is critical to studying that.

The Blaise III team must be the ones who implement it, adding it to the DEP data entry program. Fortunately or unfortunately, there is no back door into DEP so that a users might do it on their own, as in Blaise 2.5.

11. Conclusion

Our experience is that the unique architecture of Blaise 2.5 permits adding important extensions to the system such as the keystroke capture, and doing so is relatively simple. Of course, other extensions might not work so easily. The ability to access elementary system functions and processes in Blaise 2.5 through its Pascal source files is extremely flexible, powerful, and dangerous. For specialized needs it may provide a solution unavailable by any other means.

12. References

Landauer, Thomas K. 1995. The Trouble with Computers. Cambridge, MA : MIT Press.

O'Reilly, J. M. (1993). 'Lessons Learned Programming a Large, Complex CAPI Instrument.' A paper presented at the International Blaise Users Conference, London, England.

O'Reilly, J., M. Hubbard, J. Lessler, P. Biemer, and C. Turner (1994). 'Audio and Video Computer-Assisted Self-Interviewing: Preliminary Tests of New Technologies for Data Collection.' Journal of Official Statistics , Vol. 10, pp. 197-214.

Shneiderman, Ben. "User Interfaces for Survey Data Collection". Presentation to the International Conference on Computer Assisted Survey Methods. San Antonio TX, December 1996.

Tourangeau, Roger & Tom W. Smith. "Collecting Sensitive Information with Different Modes of Data Collection." Paper presented to the International Conference on Computer Assisted Survey Methods. San Antonio TX, December 1996.

Turner, Charles F. Barbara H. Forsyth, James O'Reilly, Phillip C. Cooley, Timothy K. Smith, Susan M. Rogers, and Heather G. Miller. "Automated Self-Interviewing in surveys: a review and research agenda," Paper

presented to the International Conference on Computer Assisted Survey
Methods. San Antonio TX, December 1996.

Optimal Screen Design in Blaise

Mark Pierzchala, Westat, Inc., USA²¹

1. Abstract

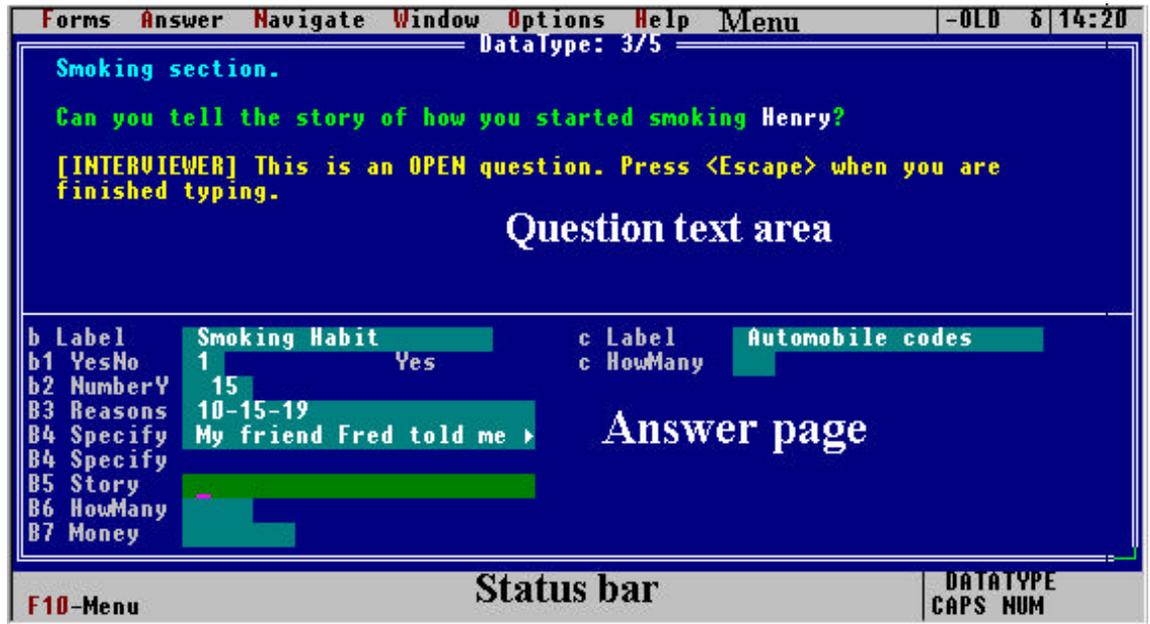
This paper discusses the enhancements that can be made to the default screen presentation of the Blaise system to make it easier for interviewers to do their jobs. Especially important are the conventions that the instrument developer can adopt to ease interviewing navigation. Suggestions that Blaise developers can employ for optimal screen presentation are offered. Examples from a datamodel called *DataType* are used to illustrate the enhancements.

Ce document discute les améliorations qu'on peut faire à la présentation de l'écran d'interviewer dans le system Blaise, esepcialement pour que la navigation se déroule meilleur. Quelques suggestions les programmeurs peuvent employer pour réaliser une présentation optimale de l'écran sont données. Les exemples d'un data model qui s'appelle *DataType* sont utilisées pour illustrer ces améliorations.

2. The split-screen interviewing presentation of Blaise

The default presentation of the Blaise system features four functional areas from top to bottom; a top menu bar, an area for question text on the top half of the screen (called an *Info Pane*), an answer page where data are entered and stored, and a status bar on the bottom as shown below.

²¹ Acknowledgements: Much of this paper is based primarily on experience gained when the author worked for the United States National Agricultural Statistics Service. Several individuals there had key roles in defining presentation standards. These individuals include Asa Manning, Roger Schou, David Knopf, Roberta Pense, Bruce Eklund, Ann Ponti, Mark Schleusener, and others. Mary Laidlaw of Westat, Inc. helped with the tables and the demonstration data model.



The two most important parts of the screen for the interviewer are the question text area and the answer page. The menu bar and the status bar are auxiliary areas of minor importance for the purpose of communicating information to the user. The split-screen approach to the display of questions and answers has long been a trademark of the Blaise system, and has been very successful and popular with interviewers. The default presentation without developer enhancements is usually well received, nevertheless, the presentation can be improved in several important ways.

The split-screen presentation of Blaise is an example of a page-based (or a forms-based) system. This is in contrast to the presentation style of some other systems which are question-based systems. A question-based system displays one question at a time to the interviewer. Experience in switching from a question-based system to Blaise teaches that the interviewers greatly prefer the Blaise page-based interface to the question-based approach. There are several reasons for this, of which a few are mentioned.

First, in the page-based system, the interviewer can see to some extent what lies ahead. A typical answer page in Blaise can display from 10 to 20 questions depending on allocation of space. This is often enough space to display a whole section, or at least major parts of one at one time. The way the routing is handled in the Blaise system, where answer cells for questions on the route are displayed and where answer cells for questions not on the route are not displayed, gives the interviewer a good understanding of the flow. Second, the interviewer can see answers to several previous responses, can verify that data have been entered correctly, and is also in a better position to catch suspicious answers that may not be caught by programmed edits. Third, the navigation in and between answer pages is very intuitive. If the interviewer is in the right

column and wants to go to a question in the left column, then she or he uses the left arrow key to get there, bypassing intervening questions. As there are from 10 to 20 questions in an answer page, it is very common that a whole interview's responses, even for a long interview, are displayed in a manageable number of pages. The page organisation enables the interviewer to review great parts of the instrument by pressing the page keys.

These and other reasons make the Blaise page-based interface popular and useful for the interviewer. It is possible in Blaise to implement a question-based approach, and a few organisations have done this. However, it is the opinion of this paper that this is a retrograde step that complicates navigation (see below). Another penalty in implementing such an interface in the Blaise system is that one of the instrument files, the **.~DM* file, may be greatly increased in size than a corresponding one for a page-based format. This can use a lot of memory and can slow down the instrument performance because there is overhead in the Blaise system when changing from one answer page to another.

3. The answer page as a question pick-list

A well designed answer page may be compared in many respects to a question pick-list in other systems, only better. In some other systems, a question pick-list can be invoked by the interviewer and appears in a pop-up window. There the interviewer can pick a question to jump to and get there by invoking a jump mechanism. The usefulness of this kind of question pick-list can be limited when the instrument is very large. For example, it can be very difficult to pick out the appropriate question if the question list is long.

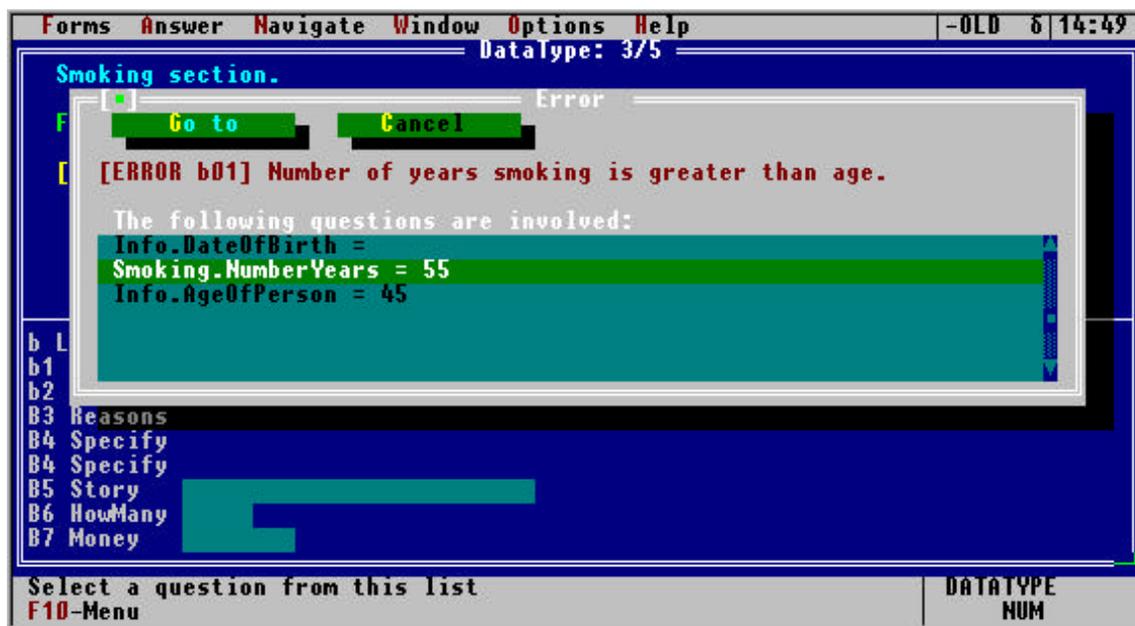
The answer page in Blaise gives a different answer to this navigational need. Each answer page can be viewed as part of an overall question pick-list, only in Blaise, it is well travelled territory. After the interviewer has gone through training and a few interviews (even practice ones), it is easy to back up and find questions. The developer can take a few steps here to ease backing up for the interviewer. The first is to incorporate labels in the page, the second is to display a section heading in the question text area. Both methods are discussed below.

Novice reviewers of the Blaise system often mistake the jumping capability of the Blaise system as the equivalent to the question pick-list in other systems. The jumping capability in Blaise has its place and can be very valuable, but it is of secondary usefulness compared to other methods of navigation.

4. Edit-specific pick-list

One of the nicest navigation features of the Blaise system is the edit-specific pick-list when errors are encountered in the interview. This is

presented in an edit window that is automatically generated by the Blaise system. The pick-list of questions is appropriate to each edit in the instrument. In order to fix an error, the interviewer selects a question from the pick-list, presses <Enter> and jumps to the question to be fixed. There are several things the developer can do to make the edit-specific pick-list more useful. The most important is to use readable question names to identify fields in Blaise. For example, *AgeOfPerson* instead of *PersnAge*. There are other useful enhancements that can be made in the edit window that are shown below.



In the edit window above, the edit message begins with a heading in square brackets which inform the interviewer that this is a hard error, the edit number is *b01*, and a succinct sentence gets the message across to the respondent and the interviewer quickly. It is also possible to include verbatim text and calculated numbers, for example rate of speed, which helps users to understand the situation better.

5. Ten methods of interviewer navigation

There are ten methods of interviewer navigation not counting the use of the mouse or pointing devices. They are listed below. The function key assignments have been changed from the Blaise defaults using the *Depmenu* files.

Navigation Facility	Function
Recording answers in an interview	Normal forward movement, taking into account flow strictures. Blaise takes care of this automatically.
Arrow keys (Up, Down, Left, and Right)	Used for short range navigation within a page or neighbouring pages. Where a page has two or more

	columns, use the left and right arrows to skip past several questions at once (for example, moving from a question in the right column to a question in the left column with the left arrow).
<Page Up> and <Page Down>	Used for medium-range navigation through sections or pages of an instrument. This is a major way to navigate. <Page Up> moves the interviewer back one page. <Page Down> moves forward one page.
<Home>	Used to jump to the first question in the instrument.
<End>	In interview mode , used to jump to the next appropriate question, taking into account changes in route due to changes in answers. In editing mode , used to jump to the last question in the instrument.
Edit-specific pick-list	This list is automatically invoked whenever a response causes an edit to be invoked. The list is in a pop-up window along with an error message and a list of question names. If it is necessary to fix an answer to a question, the interviewer can pick any of the questions from this edit-specific pick-list.
<Ctrl Enter> , Parallel blocks	Used to break the linear progression of the interview for tasks such as making appointments, recording non-response, concurrent household interviewing, or to complete sections by different respondents. After pressing <Ctrl Enter> , the interviewer selects a parallel block, to make an appointment for example, from a list of parallel blocks.
<F8> , jump box	Used to jump to any field with a field tag. It is not a primary way of navigation to individual questions since interviewers cannot be expected to know all question jump tags. It is very effective for jumping to sections with section labels.
<Shift F9> , Remarks lister	Used to review all remarks made in the instrument. You can jump to the field associated with any remark.
<Ctrl F9> , Open field lister.	Used to review all open fields the instrument. You can jump to any open field.

Some of the navigational methods are used in combination. For example, it is common to use the <Home> key to get to the beginning of the instrument and then to page down (forward) to get to an early section of the instrument, and then to use the arrow keys to get to a specific question.

It is possible to use the mouse or other pointing device in Blaise. This can be a great boon to the interviewer provided that he or she is specifically trained in the use of the device. If this training is not provided, it is best to disable the mouse driver for the interviewer.

The usefulness of the methods of navigation depends on the conventions adopted in an organisation. Two areas of concern include the choice between a page-based approach and question-based approach, and in the kinds of enhancements the developers can use to ease navigation.

6. Negative effects of reverting to a question-based presentation

The following table shows how some of the navigation methods are enhanced by the page-based display of the Blaise system.

Method	Page-based effect
Arrow keys (Up, Down, Left, and Right)	Arrowing in an answer page, or in adjacent pages, enables the interviewer to see where he or she is heading, and the arrow keys move the cursor in the direction intended. In a question-based display, the left and right arrow keys are useless, the interviewers cannot see where they are heading.
<Page Up> and <Page Down>	The concentration of answer cells in several or many answer pages, reduces the number of screens required to display data. Thus for medium range navigation, it is possible to review many answer cells with a few key strokes. The use of a question-based interface reduces <Page Up> and <Page Down> to the equivalent of <Up Arrow> and <Down Arrow> respectively.
<Home> and <F8>, jump box	Since these keys are often used in combination with the page and arrow keys, their utility is also diminished when the question-based approach is used. For example, if the interviewer jumps to a section, he or she would still have to navigate one question at time to arrive at the targeted question.
Answer page as a question pick-list	The page-based presentation enables this, the question-based approach destroys this.

The use of arrow keys, the page keys, the edit-specific pick-list, the parallel blocks, and the jump box, can all be directly facilitated by developer-provided enhancements.

7. Answer page enhancements and standards

Given that a paged-based presentation style is adopted, there are several conventions that a developer can adopt to make the answer page more understandable to the interviewer. Page standards developed over the years are shown in the table below.

Type of Text	Example in answer page	Enhancement	Purpose
Section heading	Label	Top of a column of questions in a page, or at the start of a new page.	Aids in providing context to the interviewer and in short- and medium range navigation with the page concept.
Question name	Name, DateOfBirth	To the left of the answer cell	Gives an understandable name to the question that interviewers can use in navigation with arrows or in fixing answers due to edit failure.
Question number (tag)	a, a1, b, b2	To the left of the question name	Gives an alternate name to the question that interviewers can use with the jump facility.
Start new column	Start of a new section	Top of column Done with NEWCOLUMN	This is a way to keep related questions together, and makes the page more recognisable for page-based navigation.
Start new page		Done with NEWPAGE	Usually NEWCOLUMN is better, but there are times when you want a new section to start on a new answer page.
Blank space in columns or holes in table		Done with DUMMY	Not as important as formerly for the columnar format since NEWCOLUMN eliminates the need to use multiple DUMMYs to start a section at the top of a column. However, they are still important for tables with uneven rows, especially in economic collections.

An example of the implementation of these page standards is shown in the figure below. Especially important is the use of labels in the page. These are AUXFIELDS where a brief string is computed. These labels give context and texture to the page, and are important for paging. They enable the interviewer at a glance to know where they are in the instrument. An example of a label in a page is given below.

b	Label	Smoking Habit
b1	YesNo	1 Yes
b2	NumberY	55

The text string `Smoking Habit` is computed in the RULES. In this instrument the interviewer can land on the label. This is done for two reasons; to give an introduction to the section, and to provide a jumping place for the section. The latter use of the label is made possible by the tag that is associated with the label, in this case `b`. If the interviewer wishes to move to section `b`, he or she presses the <F8> key, enters `b`, and presses <Enter> to jump. The code for the label is given below.

AUXFIELDS

```
Label (b) "@/@Y[INTERVIEWER] You are now entering
           the smoking section.
           @/@/Press <Enter> to continue."
           : STRING[20], EMPTY
```

. . .

```
RULES {Many lines later.}
```

```
Label := 'Smoking Habit'

Label

{and more code following}
```

The label is very useful as well for editing mode where the question text is not displayed. It helps orient the data editor very quickly to the screen.

A readable question name is almost always a better identifier for a question than a question number. This is true for the developer, who writes readable code. It is also true for the interviewer who reads the readable question name in the edit-specific pick-list of questions in an edit window, or who is browsing the answer pages with the page keys. It is not necessary to limit the length of the question name to 8 characters in order to conform to limitations of downstream systems. You can either let Cameleon truncate the question name to the first 8 characters in order to generate, say a SAS data step, or you can modify a Cameleon setup in order to read a SAS Var Name from another meta data location. For example, it is possible to use the first 8 characters of the description space to state a SAS Var Name. If a question number is necessary, it is better to record it as a question tag.

8. Text enhancement standards

Text enhancement standards are used in the question part of the screen, the help text in a pop-up window, and in pop-up edit windows. These parts of the interface are used to give a lot of different kinds of information such as the question, possible answers, the name of the section, the name of the respondent, and special instructions. While the question to be posed is most important, it may take up a relatively small percentage of the amount of text that appears in these windows. The enhancements are used to distinguish between types of text. Some standard text enhancements are given in the table below.

Type of Text	Position of Text	Enhancement	Purpose
Section name	Top line of the question text area	Bright blue text on dark blue background @B	Orients interviewer to the location within the instrument. It often corresponds to the label in the answer page.
Question text	Below the section name in the question text area.	Bright green text on dark blue background @G	Indicates the information to be read aloud to the respondent.
Interviewer instructions	Below the question text.	Yellow text on dark blue background @Y	Indicates information to be read by the interviewer only, (not to the respondent).
Enumerated responses that are part of the question text	Below all other text	Green text on dark blue background @G	Indicates that for this question the response categories are to be read as part of the question text. If the enumerated responses are not to be read to the respondent they are left grey.
Fills in question text	As appropriate in question text	White text on dark blue background @W	Indicates text that is inserted into the question text based on a response to a previous question.

Edit label and edit number	In the edit window	[ERROR 101] [WARNING 10] Red text on grey background @R	Distinguishes between hard and soft edits. The edit number is used to communicate problems back to the developer.
Edit banner	After the edit label and number in the edit window	Red text on grey background @R	Gives a short summary of the problem to the interviewer and respondent. Often, this is all that is necessary. Example: "Total does not equal sum of parts."
Edit message	After the edit banner in the edit window	Green text on grey background @H	Verbatim text to be read to the respondent if necessary to fix the problem. Example: "The total you just gave does not equal the sum of the items you just gave. Let us review and fix this."
Fills in edit message	As appropriate in the edit message text	White text on grey background @X	Indicates text that is inserted into the edit message based on a response to a previous question, or on a calculation done by the system.

The use of a particular text enhancement indicator such as @G does not commit the organisation to any particular colour scheme. The indicator @G may mean green for one organisation and yellow for another as Blaise allows this definition to be changed dynamically.

9. Function key assignments

Blaise allows you to map many interviewer and editor tasks to function keys F1 through F10 (F11 and F12 are not recognised by Blaise). The strategy employed is to map the most common tasks to one-stroke function keys. Common function key assignments may be:

Function	Hot key	Description
Help, question-by-question	F1	Display help text. Developers can create question-by-question help for an instrument.
Calculator	F2	Display the calculator. To copy a calculation into the answer cell use the following sequence. <Ctrl Ins>, <Esc> (gets rid of the calculator), <Shift Ins>.
Next language	F3	Switch to the next language in the set of languages available in the instrument.
Previous language	F4	Switch back to the previous language in the set of languages available in the instrument.
Don't Know	F5	Record a <i>Don't Know</i> response for the current question.
Refusal	F6	Record a <i>Refusal</i> response.
Save and Continue	F7	Save data to the hard drive and continue with the interview.
Jump	F8	Jump to a specified question or section tag. If the interviewer knows the question or section tag, he or she can jump to it if it is on the route. This is most usefully used to jump to a section tag that is attached to a label in the answer page.
Make, inspect, or modify a remark	F9	Open remark window at any question. Type a new remark, modify an existing remark, or inspect the remark in the window. Press <Esc> when the remark is complete.
Menu	F10	Move cursor to the menu bar. Developers can disable the menu bar for interviewers.
Navigate through remarks	Shift F9	Review remarks. Blaise displays the remark windows, starting with the first remark. Right arrow displays further remarks, one at a time. Jump to the remark by pressing <Enter>.
Navigate through open questions	Ctrl F9	Review responses for open questions. Similar to navigating remarks above.

10. Minor variations on the page-based presentation

The page-based presentation that Blaise gives works very well for all question types except one. The exception is for enumerated types of many

responses. In this situation, the space in the question text area can be too limited to pose both the question and to list all the answer possibilities. This limitation is due to the way that the Blaise screen uses space. In a standard 25 line DOS window, fully six lines are used by the Blaise system itself, far more than in Blaise 2.5. This leaves only 19 lines to the developer. To make matters worse, only about 76 characters are allowed per line of text instead of 80 that was available in previous versions. This space limitation for the question text area has led several organisations to experiment with the ability in Blaise to modify the default presentation. The most common screen modification is to lower the dividing line between the question text area and the answer page. This is done with the *Modelib* configuration files. The exact mechanism for doing so is beyond the scope of this article. However, it is appropriate to make a few remarks here.

The modification of the *ModeLib* file is a challenging process. On one hand, it gives an organisation tremendous flexibility in screen design. (To see that this is true, read Chapter 6 of the Developer's Guide.) On the other hand, the parameters in the *Modelib.txt* file are not well documented. It can take a great deal of experimentation to arrive at the correct combination of values of parameters. It is best for an organisation to define several standard styles and to limit developers to those possibilities.

The design tool found in *DEMOS\DESIGN* under the Blaise system directory offers limited help. This tool, a combination of a Maniplus menu system and several Blaise data models, is intended to help the developer define different screen and behaviour configurations. However, some parts of the design tool are not well executed. There is undefined and inconsistent jargon that is used in the menuing system and the question text of the data models. The excellent help facilities of the Blaise system and of Maniplus are not used. Three of the data models for specification, *Grids*, *Field Panes*, and *Toggles* are somewhat helpful, in that they help you to understand the *Modelib.txt* ASCII file. The *Toggles* data model does a very good job of explaining the behaviour possibilities of the *Modelib* file.

Since the design tool is of limited value, and since often what is needed is a lot of iteration, it is better to use a text editor on the *Modelib.txt* file and modify the parameters directly. The trouble with this method is that the *Modelib.txt* file is difficult to understand on first (second, third, . . . , nth) glance. However, if you persevere you can get the results you want. One approach is to bring the *Modelib.txt* ASCII file into the control centre and edit it there. Also present in the control centre is a data model. You can make changes to the *Modelib.txt* file then use the menu options *Tools*, *Own program* to execute the command that transforms the *Modelib.txt* file to its binary equivalent. Then re-prepare the data model, invoke it, and see what changes have been wrought. By doing everything in the control centre, it is possible to get 10 to 20 iterations per hour. This is not the thing you want your developers spending a great deal of time on. Thus the organisation would benefit if it assigns this thankless task to an individual, define several options, and let developers choose from these.

To lower the dividing line between the question text area and the answer page, you must adjust two aspects of the default presentation. First increase the area of the so-called *InfoPane*. Then decrease the area of

the *Grid*. If you do the former and not the latter, the answer page extends below the bottom line. The instrument will scroll vertically as it needs to in order to enable responses to be entered. However, when some answer cells are not visible to the interviewers when they navigate by paging, they can become very confused. The only situation where it might be appropriate to extend the answer page below the bottom line is in a repetitive rostering situation not done in a table. For example, say a block has ten questions in it and this block is repeated many times but for different topics. If NEWCOLUMN is used to start each instance of the block, and a label is used to define the topic in each block, then it probably does not matter if some answer cells are below the bottom line. The interviewer will quickly learn the pattern and know where to find things. In this situation, this is similar to the horizontal scrolling found in the table.

11. Changes in the default presentation

If there was one thing that should be changed in the default presentation, it is the elimination of some of the cosmetic display lines used in the Blaise screens. None of the framing white line needs to be there, either top or bottom, right or left. Too much space is taken by this framing line which has no functional value (it makes the screen look nicer until you have to overcome space limitations). The bottom two lines which make up the status bar are also not used. On occasion, some information is displayed there, but few if any users ever notice that information. It would be most helpful if the status bar were to disappear. Given this extra space, many organisations would not have to experiment with the *Modelib* file.

12. Summary

The default Blaise presentation is superb for almost every situation. Where it falls short for the display of enumerated responses, Blaise provides the possibility to alter the presentation style, once different styles are developed. The developer can take several simple steps to enhance the Blaise style to make the job of the interviewer and data editor easier.

ⁱ Bushnell (1995), Computer Assisted Occupation Coding, Essays on Blaise 1995, Third International Blaise Users' Conference, Statistics Finland

ⁱⁱ Dodd, T. (1985). An Assessment of the Efficiency of the Coding of Occupation and Industry by Interviewers. OPCS New Methodology Series, 14.

ⁱⁱⁱ White, P. (1983). The Continuous Manpower Survey - Feasibility Study. OPCS Survey Methodology Bulletin, 15.

-
- iv Martin, J., Bushnell, D. and Campanelli, P. (1996). A comparison of Interviewer and Office Coding of Occupations. Submitted to the Journal of Official Statistics
- v Office of Population Censuses and Surveys (1990). Standard Occupational Classification, Volumes 1 and 2. London: HMSO
- vi Elias, P., Halstead, K. and Prandy, K. Computer Assisted Standard Occupational Coding (1993). HMSO