Blaise – Alive and kicking for 20 years

Jelke Bethlehem & Lon Hofman (Statistics Netherlands)

1. Introduction

The first version of the Blaise System for computer-assisted data collection was released in 1986. So, the year 2006 marks the 20-th birthday of the system. Although operating systems have changed over time, and new data collection techniques have emerged, the basic ideas and concepts of Blaise have remained the same over the years. The heart of the system, the Blaise language, has proved to be a powerful means to define data collection instruments. Designed in a time where most statistical agencies still used mainframe computers, and the first MS-DOS computers came on the market, Blaise was able to keep up with rapid developments in information technology, and still plays a leading role in the world of data collection for official statistics.

This paper gives an account of the 20 year long history of the Blaise System. It described why it was developed and how it evolved over time.

2. The Data Editing Research Project

It is the task of national statistical offices like Statistics Netherlands to collect data on sources like persons, households, farms and establishments, and to transform these data into relevant, accurate and timely statistical information. There is an ever growing demand for all kinds of statistical information, and this calls for an organised and efficient approach to the entire process of data collection, data analysis and publication. Still, executing a survey and processing the collected information can be expensive and time-consuming.

Data editing is a vital component of the survey process. Data editing activities attempt to detect and correct errors in the individual records, questionnaires or forms. These activities are carried out with the intention to improve the quality of the results of surveys. Since statistical offices always attach much importance to this aspect of the survey process, a large part of human and computer resources are spent on it.

Notwithstanding a long time experience Statistics Netherlands had not much insight in the nature of the data editing activities, back in the early 1980's. Therefore, it started the Data Editing Research Project in 1984. Editing procedures of four surveys were investigated. Objective of this study was to get insight in the different kind of activities carried out during the data editing process, the frequency of occurrence of these activities, the time spent on them, the number and qualifications of the people involved, and the effect on data quality. The data editing research project is described in more detail in Bethlehem (1987).

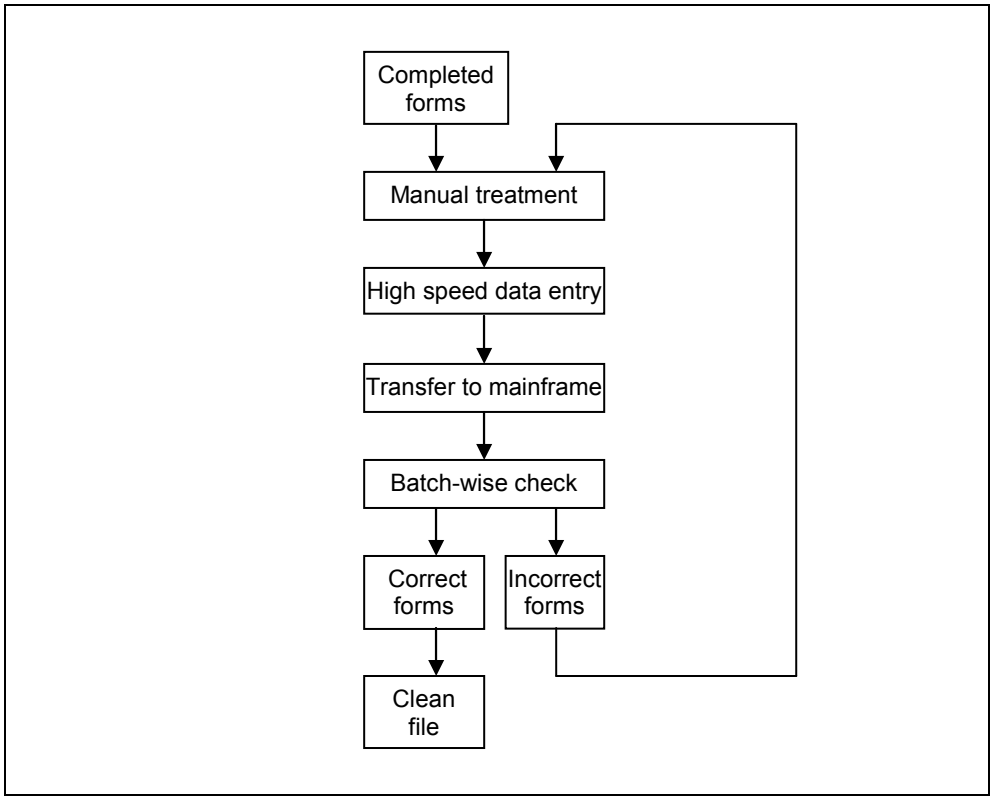
Four surveys were selected in the study: two economic surveys, the Foreign Trade Survey and the Survey on Transport of Goods by Road, and two social surveys, the Labour Force Survey and the Survey on Well-being of the Population. Two surveys (one economic and one social) were very large. For these surveys only an extensive inventory of all activities was made. The two other surveys (one economic and one social) were of moderate size. For these surveys not only a

detailed process analysis was carried out, but, because of their size, it was also possible to carry out a comprehensive analysis of the data produced by the surveys. Considerable differences were observed between the editing procedures of economical and social surveys. In most economical surveys, the questionnaire was straightforward. Questions were answered one after another without complex routing structures. Many range checks and consistency checks were carried out. Totals of individual entries had to be calculated or checked. Often figures of firms were confronted with corresponding figures from previous years. To correct detected errors, it was sometimes possible to contact firms, either by telephone or by an employee of Statistics Netherlands which visited the firm. In short, economical surveys could be characterised by simple questionnaires with comprehensive checking.

Social survey questionnaires tended to be large with complex routing structures. Error checking mainly concerned the route followed through the questionnaire and some range checks. Only a few consistency checks were carried out. To be able to correct detected errors, contact was necessary with the supplier of the information. Since this was hardly ever possible, generally correction resulted in setting a value to 'unknown'.

Although the data editing process differed from survey to survey, still some general characteristics could be observed which held for nearly all surveys. The general scheme of the editing process is summarised in figure 2.1.

Figure 2.1. The traditional approach to data editing



After all forms had been collect, subject-matter specialists checked the forms for completeness. If necessary and possible, skipped questions were answered, and obvious errors were corrected on the forms. Sometimes, the forms were manually copied to a new form to allow for the subsequent step of fast data entry. Next, the forms were transferred to the data entry department. Data typists entered the data in the computer at high speed without error checking. The computer was a dedicated

system for data entry. After data entry, the files were transferred to the mainframe computer system. On the mainframe an error detection program was run. Usually, this was a dedicated COBOL program. This program produced a list of detected errors. This list was sent to the subject-matter department. Specialists investigated the error messages, consulted corresponding forms, and corrected errors on the lists. Corrected forms were sent to the data entry department, and data typists entered the corrections in the data entry computer. The file with corrections was transferred to the mainframe computer. Corrected records and already present correct records were merged. The cycle of batch-wise error detection and manual correction was repeated until the number of detected errors was considered to be sufficiently small.

After the final step of the editing activities, the result was a 'clean' data set, which could be used for tabulation and analysis. Detailed investigation of the data editing procedures for the four selected surveys lead to a number of conclusions. These conclusions are summarised below.

- Various people from different departments were involved. Many people dealt with the information: respondents filled in forms, subject-matter specialists checked forms and corrected errors, data typists entered data in the computer, and programmers from the computer department constructed editing programs. Transfer of material from one person/department to another could be a source of error, misunderstanding and delay.

- Different computer systems were involved. Most data entry was carried out on Philips P7000 minicomputer systems, and data editing programs ran on a CDC Cyber 855 mainframe. Furthermore, there was a variety of desktop (running under MS-DOS) and other systems. About 300 interviewers had been equipped with laptops running under CP/M. Transfer of files from one system to another caused delay, and incorrect specification and documentation could produce errors.

- Not all activities were aimed at quality improvement. A lot of time was spent just on preparing forms for data entry, and not on correcting errors. Subject-matter specialists had to clean up forms to avoid problems during data entry. The most striking example was assignment of a code for 'unknown' to unanswered questions.

- The process was going through *macro cycles*. The whole batch of data was going through cycles: from one department to another, and from one computer system to another. The cycle of data entry, automatic checking and manual correction was in many cases repeated three times or more. Due to these macro cycles, data processing was very time consuming.

- The structure and nature of the data had to be specified in nearly every step of the data editing process. Although essentially the same, the 'language' of this meta-data specification could be completely different for every department or computer system involved. The questionnaire itself was the first specification. The next one was with respect to data entry. Then, automatic checking program required another specification of the data. For tabulation and analysis, e.g. using the statistical package SPSS, again another specification was needed. All specifications came down to a description of variables, valid answers, routing and possibly valid relations.

There are a lot of similarities between statistical production processes as described above and business processes in general. Therefore, approaches to improving business processes can also be applied to the survey process. Two such approaches

are discussed here. One is the theory of Deming on improving quality in industrial production, and the other is the more recent Business Process Redesign (BPR) approach.

Quality control has always been an important issue in industrial production. The theory of Deming (1986) about improving quality and productivity in industry is well-known. Many of his famous 14 points for management also apply to the production of statistical information. One of these points states that one should cease dependence on mass inspection. Inspection of final products to improve quality is too late, ineffective and costly. Quality must be built in at the design stage. These statements particularly apply to data collection and data editing. By trying to detect and correct errors in answer to questions well after they have occurred, one fails to locate the source of the error, and consequently, these errors can not be eliminated.

Some of the problems mentioned by Deming could be solved by introducing *computer-assisted interviewing* techniques. The rapid advent of the microcomputer in the 1980's made it possible to use microcomputers for computer-assisted interviewing. The paper questionnaire was replaced by a computer program asking the relevant questions. The computer took control of the interviewing process. It could perform two important activities:

- *Route control.* The computer program determines which question is to be asked next, and displays that question on the screen. Such a decision may depend on the answers to previous questions. Hence it relieves the interviewer of the task of taking care of the correct route through the questionnaire. As a result, it is not possible anymore to make route errors.

- *Error checking*. The computer program checks the answers to the questions which are entered. If an inconsistency is detected, the program gives a warning, and one or more of the answers concerned can be modified. The program will not proceed to the next question until all detected errors have been corrected.

The concepts of Business Process Redesign (or Re-engineering) are well described in the book by Hammer & Champy (1994). They claim that many of today's companies operate on principles formulated more than two centuries ago by Adam Smith in his book "The Wealth of Nations". These principles worked in an expanding market for mass products and services, but they offer not guarantee to survive in today's quickly changing market conditions and competitive climate. The traditional approach focuses on splitting business processes in simple tasks. The advantage of such a task oriented approach is that the relative simple tasks do not require advanced skills or knowledge. However, simple tasks demand complex processes to knit them all together. For two hundred years, companies have accepted the inconvenience, inefficiencies and costs associated with these complex processes in order to be able to take advantage of the benefits of simple tasks.

Typically, a task oriented process involves different people in different departments. The processes are sequential in the sense that one department cannot start working on a task until another department has finished it. Moreover, some business processes have iterative components. The process goes through a cycle several times before it can be completed. Such sequential and iterative processes are time-consuming and error prone.

Preaching quality is not sufficient to improve business processes. Even if each separate task is carried out efficiently, the process as a whole need not be. Application of the developments of state of the art information technology will not help if it is only used to automate current tasks. More is needed to accomplish

substantial improvements. Business Process Redesign means "the fundamental rethinking and radical redesign of business processes to achieve dramatic improvements in critical contemporary measures of performance, such as cost, quality, service, and speed".

BPR stresses the importance of the creative use of information technology. However, new developments in information technology should not be used just to automate the tasks in the traditional processes. This will only lead to marginal performance improvements. Far more substantial improvements can be obtained by using computer technology in a fundamental new way.

There is a striking degree of similarity between the descriptions of the traditional survey process and that of the traditional business process. Both processes share the same problems: process fragmentation, focusing on tasks instead of processes, involvement of several departments, sequentially and cycling, it is all there. Statistics Netherlands followed a BPR-like approach in redesigning its statistical production processes.

3. Blaise 1: Integrating data entry and data editing

The first phase of the redesign of the survey processes of Statistics Netherlands concentrated on the data editing activities. The idea was to improve the handling of paper questionnaire forms by integrating data entry and data editing tasks. The traditional batch-oriented data editing activities, in which the complete data set was processed as a whole, was replaced by a record-oriented process in which records (forms) were completely dealt with one at a time.

The new group of activities was implemented in a so-called CADI system. *CADI* stands for Computer Assisted Data Input. CADI was an intelligent and interactive system for combined data entry and data editing. The CADI system was designed for use by the workers in the subject-matter departments.

Data could be processed in two ways by this system. In the first approach, subject-matter employees worked through a pile of forms with a microcomputer, processing the forms one by one. They entered the data, and after completion of the form, they activated the check option to test for all kinds of errors. Detected errors were reported and explained on the screen. Errors could be corrected by consulting forms, or by contacting the suppliers of the information. After elimination of all errors, a 'clean' record was written to file. If an employee did not succeed in producing a clean record, he could write the record to a separate file of ‘dirty’ records. A specialist can deal with these hard cases later on, also using a CADI system.

In the second approach, data typists used the CADI system to enter data beforehand without much error checking. After completion, the CADI system checked in a batch run all records, and flagged the incorrect ones. Then subject-matter specialists handled these ‘dirty’ records one by one, and correct the detected errors.

To be able to introduce CADI on a wide scale in the organisation, a new standard package was developed. The name of this standard package was *Blaise*. It derived its name from the famous French theologian and mathematician Blaise Pascal (1623-1662). The basis of the Blaise System was the Blaise language, which was used to create a formal specification of the structure and contents of the questionnaire. The Blaise language had its roots, for a large part, in the programming language Pascal.

The first version of the Blaise System ran on microcomputers (or networks of microcomputers) under MS-DOS. It was intended for use by the people of the subject-matter departments, therefore no computer expert knowledge was needed to use the Blaise system. The design goal of the system was to provide subject-matter experts with a powerful but user-friendly tool that enabled them to input their knowledge about a survey into the system, and to take care of all subsequent data processing steps.

In the Blaise philosophy, the first step in carrying out a survey was to design a questionnaire in the Blaise language. Such a specification of the questionnaire contains more information than a traditional paper questionnaire. It did not only describe questions, possible answers, and conditions on the route through the questionnaire, but also relationships between answers that had to be checked.

The Blaise language had to be powerful enough to be able to deal with large and complex surveys. Since the specification was used to define the survey, the language had to be readable enough. In fact, a Blaise questionnaire had to be self-documenting, i.e. it is the basic description of the survey which could be used by all people involved. Figure 3.1 gives an example of such a simple questionnaire in Blaise.

Figure 3.1. A simple Blaise 1.0 questionnaire specification

```
QUESTIONNAIRE LFS "The Labour Force Survey";

QUEST
    SeqNum  (KEY) "Sequence number of the interview?": 1..1000;
    Age     "What is your age?": 0..99;
    Sex     "Are you male or female?": (Male, Female);
    MarStat "What is your marital status?":
            (Married "Married",
             NotMar  "Not married")
    Job      "Do you have a job?": (Yes, No);
    JobDes   "What kind of job do you have?": STRING[20];
    Income   "What is your yearly income?":
             (Less20  "Less than 20,000",
              Upto40  "Between 20,000 and 40,000",
              More40  "More than 40,000");

 ROUTE
    SeqNum; Age; Sex; MarStat; Job;
    IF Job = Yes THEN
       JobDes; Income
    ENDIF

 CHECK
    IF Age < 15 "Respondent is younger than 15" THEN
       MarStat = NotMar "he/she is too young to be married !"
    ENDIF

 ENDQUESTIONNAIRE.
```

The first part of the questionnaire specification was the *Quest section*, containing the definition of all questions that can be asked. A question consists of an identifying name (for internal use in the questionnaire), the text of the question as presented to the respondents on the paper form, and a specification of valid answers.

The next part of this sample Blaise questionnaire is the *Route section*. It described under which conditions, and in which order the questions had to be asked.

Consistency checks were specified in the *Check section*. They checked whether a specified condition was true for the answers to the questions involved. If the condition was not satisfied, an error message was generated.

Figure 3.2. The main screen of Blaise 1.1

Preparation of a CADI program required a number of steps. Figure 3.2 shows the development environment. First, the questionnaire specification was entered using a text editor. The default editor was PC-Write 2.71. Next, the parser was activated to check the questionnaire for syntax errors. If no errors were encountered, the questionnaire specification was transformed in Turbo Pascal source code. Then, the Pascal code was compiled using a Turbo Pascal compiler (Turbo Pascal 3.0), and the result was an executable program. This program was called the *CADI machine*. Figure 3.3 shows how the CADI machine of the specification in figure 3.1 looked like.

Figure 3.3. A Blaise 1.1. CADI machine

The CADI program could be used in three different ways to process survey data:

- Subject-matter specialists entered and simultaneously edited the data. Figure 3.3 shows an example of a screen after a form had been entered. Note that questions Age and MarStat have error counters in front of their input fields. For these questions an inconsistency has been detected.

- Data entry typists entered the data at high speed, without much error checking. After completion, the CADI program carried out an integral check on all records, and flagged the incorrect ones. Then subject-matter specialists corrected the wrong forms in an interactive session.

- If a data file was created and supplied by another agency, the data could be imported in the Blaise system, the CADI program carried out an *integral check*, thereby flagging incorrect records. The subject-matter specialists did the interactive editing of the incorrect records.

Version 1.0 of Blaise was completed and taken into production in 1986. After an initial period in which people had to get used to the new way of working, the new approach was accepted enthusiastically. The Blaise system turned out to be sufficiently flexible for survey processing activities in many departments. Use of Blaise lead to greater speed, lower operating costs, and to an improvement in data quality, even at a time where Statistics Netherlands faced substantial reductions in staff.

The benefits for the people in subject-matter departments were notable. The work gave them much greater scope. They appreciated the increased autonomy and responsibility in organising their work. It lead to reduced specialisation, the acquisition of a wider range of skills, and the possibility to adapt and devise systems for their own purposes. These advantages generally outweighed the need, in some cases, to undertake lengthy periods of training, and a reluctance on the part of some to work with computers for long periods.

4. Blaise 2: Computer-assisted interviewing

After version 1 of Blaise had been in use for a short time, it was realised that such a system could be made much more powerful. The questionnaire specification in the Blaise system contained all knowledge about the questionnaire and the data needed for survey processing. Therefore, Blaise should be capable to manage other data processing applications. As a next step, the system was extended to be able to handle computer assisted interviewing.

In computer assisted interviewing, the paper questionnaire is replaced by a computer program containing the questions to be asked. The computer takes control of the interviewing process. It performs two important activities:

- *Route control*. The computer program determines which question is to be asked next, and displays that question on the screen. Such a decision may depend on the answers to previous questions. Hence it relieves the interviewer of the task of taking care of the correct route through the questionnaire. As a result, it is not possible any more to make route errors.

- *Error checking*. The computer program checks the answers to the questions which are entered. Range checks are carried out immediately after entry and consistency checks after entry of all relevant answers. If an error is detected, the program gives a warning, and one or more of the answers concerned can be

modified. The program will not proceed to the next question until all detected errors have been corrected.

Application of computer assisted data collection has three major advantages. In the first place, it simplifies the work of interviewer (no more route control), in the second place, it improves the quality of the collected data, and in the third place, data is entered in the computer during the interview resulting in a clean record, so no more subsequent data entry and data editing is necessary.

Version 2.0 of Blaise was completed in 1988. It implemented just one form of computer assisted interviewing: CAPI. CAPI stands for Computer Assisted Personal Interviewing. It is a form of face-to-face interviewing in which interviewers use small laptop or notebook computer to ask the questions and to record the answers.

Dutch experiences with CAPI started in 1983. The first experiments showed that respondents had no objections against computer assisted interviewing, and the interviewers developed a positive attitude towards this way of working. Use of computers did not increase nonresponse.

Due to the success of these experiments, Statistics Netherlands started in 1987 with full scale use of CAPI in a regular survey: the continuous Labour Force Survey. More details about these CAPI surveys can be found in Van Bastelaer et al. (1987a, 1987b) and Bemelmans-Spork and Sikkel (1985a, 1985b). The interviewing programs in the early experiments were dedicated, tailor-made programs, and later based on a more generic system (Quest4). But from 1990 on all CAPI surveys were managed by the Blaise system. The most important one was the Labour Force Survey. Each month, about 400 interviewers equipped with laptops visited 12,000 addresses. The laptop was a Toshiba T1000, see figure 4.1.

Figure 4.1. A Toshiba T1000 laptop computer

After a day of interviewing, they returned home and connected their computers to the power supply to recharge the batteries. The laptop computer was also connected to a telephone and modem. At night, the collected data were automatically transmitted to the office, and in return, new addresses were sent to the interviewers. The next morning, they would find a recharged machine with a clean workspace, ready for a new day of interviewing.

Figure 4.2 shows an example of a screen of a CAPI program generated by Blaise 2.0. The screen is divided in two parts. The upper part contains the current question to be answered. As soon as the answer has been entered, this question is replaced by the next question on the route.

Figure 4.2. A Blaise 2.3 CAPI program

Just displaying one question at the time only gives the interviewers a very limited feed-back on where they are in the questionnaire. Therefore, the lower part of the screen displays (in a very compact way) the current page of the questionnaire.

The CAPI program applied dynamic routing. No questions could be skipped that had to be answered, and no questions could be answered that were not on the route. Thus the routing structure of the questionnaire could not be violated.

The CAPI program also applied dynamic error checking. As soon as a consistency error was encountered, an error message appeared on the screen, including a list of all questions involved. The interviewer could jump back to a question in order to correct its answer. It was not possible to continue to a new question until all errors has been corrected. Therefore, when the end of the interview was reached, no more errors remained in the form.

In 1990, another mode of computer assisted interviewing was included: CATI, or Computer Assisted Telephone Interviewing. Thus, version 2.3 of Blaise was born. In the case of CATI, the interviewing program is installed on a desktop computer. The interviewer called respondents from a central unit (call centre), and carried out the interview by telephone. The computer program selected the proper question to be answered, and displayed it on the computer screen. The interviewer read out the question, and entered the answer provided by the respondent. The program checked for all kinds of errors. If an error was detected, the computer warned the interviewer that something is wrong, and corrections could be made. See e.g. Nicholls and Groves (1986a, 1986b) for more details about CATI.

An important component of a CATI survey is the call scheduling system. This system must take care of proper handling of busy numbers, no-answers, appointments, etc. Such a scheduling system was automatically made available by the Blaise system when it created a CATI survey.

In the very early 1990's nearly all household surveys of Statistics Netherlands had become CAPI or CATI surveys. Surveys using paper forms had almost become extinct. Figure 4.1 lists all major and regular household surveys at that time together with their mode of interviewing.

Table 4.1. Household surveys carried out by the CBS

Survey	Mode	Interviews per year
Survey on Quality of Life	CAPI	7,500
Health Survey	CAPI	6,200
Day Recreation Survey	CAPI	36,000
Crime Victimisation Survey	CAPI	8,000
Labour Force Survey	CAPI	150,000
Car Use Panel	CATI	8,500
Consumer Sentiments Survey	CATI	24,000
Social-Economic Panel	CATI	5,500
School Career Survey	CATI	4,500
Mobility Survey	CATI / CADI	20,000
Budget Survey	CADI	2,000

An important new feature in version 2.3 of Blaise was the *Setup Generator*. This tool assisted in preparing data collected with the Blaise for subsequent processing, like tabulation and analysis. Statistics Netherlands had adopted the principle that whenever good software is available, wheels should not be reinvented by developing similar software itself. Particularly, this applied to software for statistical analysis. However, using available software has the drawback that it is usually necessary to put the data, and the description of the data (the metadata) into a different format. Version 2.3 of Blaise took care of that. The Setup Generator was able to use the information in the Blaise questionnaire specification to create setups for the tabulation package Abacus, the statistical packages SPSS, SAS and Stata, and to prepare data for downloading in database packages like Paradox and Oracle.

To prepare data for processing by other software, the Setup Generator used instruction files. Blaise contained instruction files for the software packages mentioned above. For other, not mentioned software, users could generate new instruction files for their own favourite software packages. Thus, Blaise offered a flexible approach to adapt the system for work in a specific environment.

Figure 4.2 shows an example of setup script for SPSS. Such a script partly consists of fixed text and partly of setup instructions. The fixed text (e.g. `VAR LABELS`) was just copied to the setup file. Instructions were placed between square brackets. They used information that was retrieved from the questionnaire specification file (e.g. `QuestionName`).

The ideas behind the Setup Generator language were similar to those behind XML / Sthyle sheets. However, at that time XML did not yet exist. The Setup Generator was an XML application 'avant la lettre'.

Figure 4.2. Setup generator script for SPSS

```
[SubFileLoop]
  [OutFile := SubFileBlockName + '.SPS']
  []TITLE           '[SubFileBlockName]'.
  []DATA LIST       FILE = '[FileName].[SubFileExtension]' /
  [QuestionLoop]
    [if not FirstQuestion then][/][endif]
    [:16][QuestionName 16:] [FirstPosition:4] -[LastPosition:4][&]
    [if QuestionType=Open or QuestionType=Code then] (A)[&]
      [elseif QuestionType=Subrange and NumberOfDecimals>0
      then] ([NumberOfDecimals])[&]
    [endif]
  [EndQuestionLoop]
  [].

  []VAR LABELS
  [QuestionLoop]
    [If QuestionLabel <> '' Then]
      [if not FirstQuestion then][/][endif]
      [:16][QuestionName 16:]'[QuestionLabel]'[&]
    [EndIf]
  [EndQuestionLoop]
  [].

  []VALUE LABELS[&]
  [TypesLoop]
    [if TypesIndex>1 then]/[endif][&]
    [QuestionLoop]
      [if QuestionTypeNr=TypesIndex
        then][/][:16][QuestionName 8:][&]
      [endif]
    [EndQuestionLoop]
    [AnswerLoop]
      [if AnswerIndex>1 then][/][:24][endif]
      [AnswerCode :8] '[AnswerLabel]'[&]
    [EndAnswerLoop]
  [EndTypesLoop]
  [].

  []LIST            /CASES TO 10.
  []SAVE            /OUTFILE '[SubFileBlockName].SYS'.
[EndSubFileLoop]
```

Finally, version 2.5 of Blaise emerged in 1992. One of the major enhancements included in this version, was *trigram coding*. One of the time-consuming activities of the traditional survey process is the coding of open answers. A typical example is the question about the occupation of the respondent. Questions are easiest to process if a respondent selects one possibility from a list of pre-coded answers. However, for a question like occupation, this set of pre-coded answers would be very long, and thus it would be very hard for the respondent to select the proper answer. This problem is avoided by letting the respondent formulate his own answer, and then literally copying the answer on the form. To enable analysis of this type of information, answers must be classified afterwards. This coding was usually carried out by experienced subject-matter specialists.

The Blaise System already contained a tool for *interactive coding*, thereby providing the possibility of integrating coding either in the data collection phase or in the data entry and data editing phase. The coding module could be used in two different ways. One way was *hierarchical coding*. Coding of an answer starts by entering the first digit of the code by selecting the proper category from a menu. After the user had entered a digit, the program presented a subsequent menu containing a refinement of the previously selected category. So the description became more and more detailed until the final digit was reached. Another way to

code a question was to apply the dictionary approach. This approach simply tried to locate an entered description in an alphabetically ordered list. New in version 2.5 of Blaise was trigram coding. A trigram is a sequence of three subsequent characters appearing in a word. For example, the word ‘coder’ can be decomposed into the trigrams 'cod', 'ode' and 'der'. An entered description is decomposed in the set of all of its trigrams. Also all descriptions in the dictionary are stored in the form of trigram sets. The system now attempts to find a description in the dictionary by locating trigram sets that have a high percentage of trigrams in common. Thus, a description will even be found if spelling errors are made, or if an alternate spelling is used. All these coding techniques provided a better treatment of open questions. They improved data quality, and speeded up the survey process, particularly if coding could be carried as part of the interview.

5. Other developments

During the development of Blaise 1 and Blaise 2 other projects were initiated that were loosely linked to Blaise and played a roll in the use of the data captured with Blaise. Worth mentioning are Manipula, Abacus and Bascula.

5.1. Manipula

The Manipula project started in 1989 during the development of Blaise CATI call management system. Users experienced problems in preparing data files required as input for this system. Programs were available on the mainframe computer to convert and reformat files, but on the PC they were lacking. The first version of Manipula was developed to do this job.

Manipula was designed for manipulating ASCII and Blaise files under MS-DOS. The program could be used to link files, to select fields and records, to do computations, to detect errors, to make print files and for sorting and summarising. Manipula uses a *setup*. In this setup the user gives a description of one or more input files, the output file and the manipulations to be carried out on the data. A setup is specified in the Manipula language. A simple example of a setup can be found in figure 5.1

Figure 5.1. A simple Manipula 1.0 setup

```
INPUTFILE "samplon.dat"
  city, 1, 1, STRING;
  province, 3, 1, STRING;
  sex, 5, 1, STRING;
  age, 7, 2, STRING;
  employed, 10, 1, STRING;
  income, 12, 4, STRING;
OUTPUTFILE "samplon.exl"
  sex, 1, 1, STRING;
  age, 3, 2, STRING;
  income, 6, 4, STRING;

MANIPULATE
  IF ((citys'1') OR (city='2') OR (city='3') AND
     (employed='1')
  THEN
    SELECT
  ENDIF;
```

The setup above describes a selection of fields for certain records in an ACII data set. Manipula was a tool developed independently of Blaise which also had its own way of describing the data.

5.2. Abacus

Blaise 2.0 also contained the tabulation package Abacus. Abacus had a very efficient counting algorithm. This made it possible to produces tables out of very large data files. And the tables produced, were camera-ready.

Abacus was a user-friendly package allowing tables to be specified in an interactive way. This was an innovation. Most tabulation packages at that time used, sometimes complex, specification scripts.

Three-dimensional tables could be specified (rows, columns and layers), allowing for concatenation and nesting of variables in each dimension. Also weight variables could be specified, so weighted totals could be computed (e.g. in order o correct for a nonresponse bias).

Figure 5.2 shows an example of a table specification screen of Abacus. The rows contain the variable *Town* nested in *Province*. The columns contain the two concatenated variables *Job* and *Sex*.

Figure 5.2. Specifying a table in Abacus

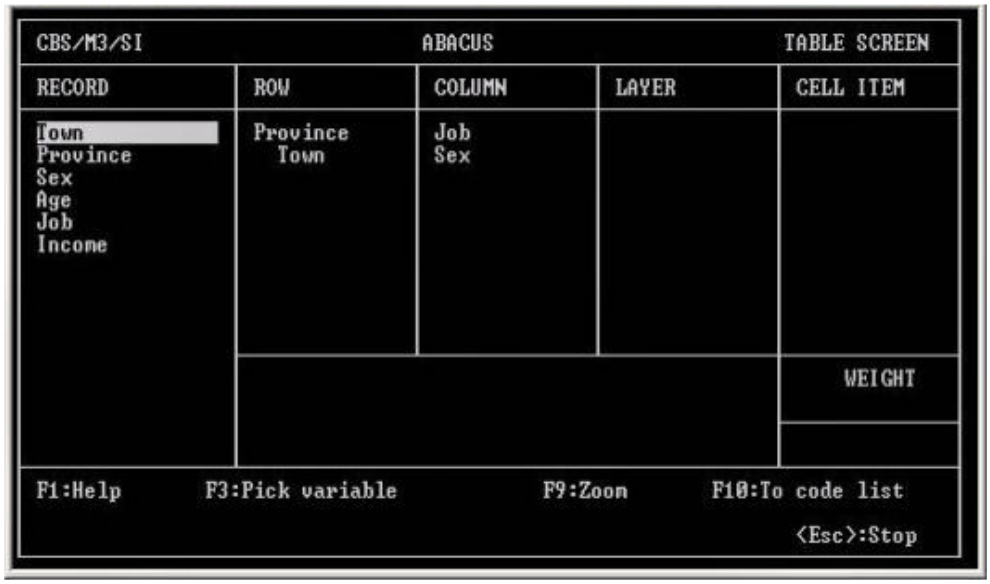


Figure 5.3 shows an example of the type of output that could be generated using Abacus.

Figure 5.3. A table generated by Abacus

Spec. file : D:\ABACUS\SAMPLON.ABC 15- 3-2006 12:34:16 Page 1

The Population of Samplonia

Number of records	Total	Employment		Sex	
		Job	No job	Male	Female
Total	1,000	341	659	511	489
Agria	293	121	172	145	148
Wheaton	144	60	84	70	74
Greenham	94	38	56	44	50
Newbay	55	23	32	31	24
Induston	707	220	487	366	341
Oakdale	61	26	35	36	25
Crowdon	244	73	171	128	116
Smokeley	147	49	98	80	67
Mudwater	255	72	183	122	133

Source: Statistics Samplonia

Scroll : ↑ ↓ [Ctrl] ← → [Ctrl] Pgup,Pgdn Home,End ESC: Stop

5.3. Bascula

To improve the quality of estimates in sample surveys, some kind of weighting adjustment procedure is often carried out. The reason is that the collected data can not be used for making inference about the population without having made corrections for unequal selection probabilities or selective non-response.

Post-stratification is a well-known correction technique. Every record is assigned some weight, and these weights are computed in such a way that the weighted sample distribution of characteristics like sex, age, marital status, and area reflects the known distribution of these characteristics in the population.

Lack of sufficient population information can make application of post-stratification difficult. Statistics Netherlands carried out research in order to improve weighting techniques. The result was a new general weighting technique called *linear weighting*. This technique computes weights from a linear model that relates the target variables of a survey to auxiliary variables.

Post-stratification is a special case of this method. Because of the generality of the method, different weighting modes can be applied that take advantage of the available population information as much as possible, and at the same time avoid the above mentioned problems.

Multiplicative weighting is another weighting technique that can compute weights for the case of insufficient population information. This technique is sometimes also called raking or iterative proportional fitting. Weights are obtained as the product of weight factors, which are computed in an iterative process.

In the late eighties, Bascula was developed as a tool implementing these weighting techniques. The first version of Bascula ran under MS-DOS, and later versions under Windows.

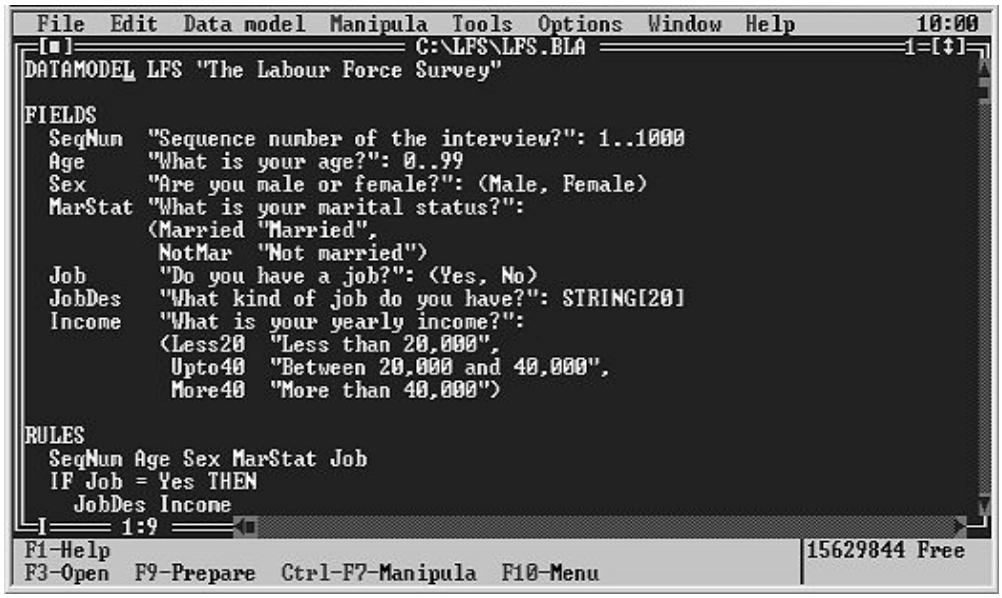
6. Blaise 3: A control centre for survey processing.

Version 2 of Blaise started a process of integrating the software needed to carry out all steps of the survey process. Many computer programs existed that took care of

one or more of these steps (data entry, data editing, tabulation, analysis, etc.), but the ideal package that does everything did not (yet) exist. This was not very surprising. Handling statistical data has so many diverse aspects that it is impossible to cover everything in one package. Moreover, since new ideas have evolved in quick succession both in statistics and informatics, they could never have been implemented at once in one system.

Having to work with all these different programs, each requiring its own data format, its own specification language (metadata format), and its own user interface, is not very efficient. Blaise 3 offered a solution to these problems. It was a system that could manage all software needed in the survey process, from questionnaire design, via data capture, to data editing and subsequent activities. It spoke the language of each program and thus was able to take care of the communication between them. Separate programs became modules in a *statistical control centre*. Using such a control centre was like being a pilot in the cockpit of an aeroplane: There are instruments to control all activities, and to monitor the situation. For more about the idea of a statistical control centre, see Bethlehem (1997).

Figure 6.1. The Blaise III Control Centre



Blaise 3 was a completely redesigned version of Blaise 2. It was finished in 1994. Object oriented programming was used to develop the system. To indicate that it was a completely new package, its name was changed into Blaise III. See figure 6.1 for a sample screen of the survey development environment if Blaise III.

Figure 6.2. A simple Blaise III data model

```
DATAMODEL LFS "The Labour Force Survey";

FIELDS
    SeqNum  "Sequence number of the interview?": 1..1000
    Age     "What is your age?": 0..99
    Sex     "Are you male or female?": (Male, Female)
    MarStat "What is your marital status?":
               (Married "Married",
                NotMar  "Not married")
    Job      "Do you have a job?": (Yes, No)
    JobDes   "What kind of job do you have?": STRING[20]
    Income   "What is your yearly income?":
               (Less20  "Less than 20,000",
                Upto40  "Between 20,000 and 40,000",
                More40  "More than 40,000")

 RULES
    SeqNum Age Sex MarStat Job
    IF Job = Yes THEN
       JobDes Income
    ENDIF

    IF Age < 15 "Respondent is younger than 15" THEN
       MarStat = NotMar "he/she is too young to be married !"
    ENDIF

 ENDMODEL
```

Blaise III was based on the idea that processing survey information is essentially a set of activities for manipulating data and meta-data. The data are the answers to the questions on the questionnaire form, and in the course of the process they are transformed into useful statistical information. This can only be achieved properly if relevant metadata is available: the information about the data. Blaise III implemented the principle that there could be no data without metadata. This principle was applied everywhere in the system.

Carrying out a survey with Blaise III started by specifying a *data model*. This was the new name for the questionnaire specification in the Blaise language. Blaise III introduced some basic changes in the language. Figure 6.2 shows the Blaise III version of the questionnaire in figure 3.1.

Some keywords have been replaced by new ones to reflect the more general nature of the metadata specification. For example, `QUESTIONNAIRE` has become `DATAMODEL`, and `QUEST` has become `FIELDS`. Blaise 1 had separate sections of route and check instructions, but in Blaise III they were combined in one `RULES` section. The semicolon was not necessary any more as a separator between commands.

The data model is a metadata specification. Blaise stored this metadata specification in machine-readable form, making it available to all parts of the system. After that, never again variables had to be specified. The metadata was always there, and it could automatically be transformed into many different formats, depending on the software that needs it.

Blaise III supported hierarchical data models. This is how many survey researchers see the world. There are households, households have members, members have health problems, etc. This is exactly what the language of the Blaise system offered. Blaise III promoted a modular approach to data model design in which hierarchies are implemented by means of nesting simple data structures.

Figure 6.3. Blaise specification of a hierarchical data model

```
DATAMODEL HouseHold "The Household Survey"

  BLOCK BPerson "Demographic data of respondent"
    FIELDS
      Name     "What is your name?": STRING[20]
      Sex      "Are you male of female?": (Male, Female)
      Married  "Are you married?": (Yes, No)
      Age      "What is your age?": 0..120
    RULES
      Name Sex Married Age
      IF (Age < 15) "If age of respondent is less than 15" THEN
        MarStat = NevMarr "you are too young to be married !"
      ENDIF
  ENDBLOCK

  LOCALS
    I: INTEGER

  FIELDS
    Address "What is the address of the household?": STRING[30]
    HHSize "Number of persons in the household?": 1..10
    Person: ARRAY[1..10] OF BPerson

  RULES
    Address HHSize
    FOR I:= 1 TO HHSize DO
      Person[I]
    ENDDO

 ENDMODEL
```

Data models could become large and complex, making it difficult to keep track of both the overall structure, and the details of several thousands of fields. Blaise offered the *block* concept to group metadata in a modular way. A block can be seen from the inside as a group of closely related fields that are independent from all other fields. From the outside a block can be seen as a unit of information that can be nested in other blocks, or repeated a number of times. Figure 6.3 contains an example of such a simple hierarchical data model.

This example contains the definition of one block: *BPerson*. The fields section at the highest level (the household level) introduces an array of 10 occurrences of this block (*Person[1]* to *Person[10]*). All these 10 fields represent a group of fields: the fields in the block BPerson (*Name*, *Sex*, *MarStat* and *Age*).

The Blaise Control Centre offered tools to handle data model specifications in a transparent and user-friendly way. It could visualise a data model by displaying it as a tree structure. Figure 6.4 shows how the data model of figure 6.3 is displayed.

Figure 6.4. A hierarchical data model

```
┬ HOUSEHOLD                          ┬ HOUSEHOLD
├── Address                          ├── Address
├── HHSize                           ├── HHSize
└─+ ARRAY [1..10] OF BPERSON         └─┬ ARRAY [1..10] OF BPERSON
                                       ├── PERSON
                                       ├── Name
                                       ├── Sex
                                       ├── Married
                                       └── Age
```

The data tree on the left-hand side described the data model at the highest level. It gave a good view of the general structure. There is one elementary field (*Address*) and one block field (*BPerson*). The plus sign (+) indicated a block of which the fields were not displayed. Unfolding the block, i.e. making visible all fields in the block, would result in the data tree on the right hand side with all the details.

Previous versions of Blaise could generate either a CADI, CAPI or CATI program. Each of these programs had its own characteristics and behaviour. And all programs had to be compiled using a Turbo Pascal compiler. This all changed in Blaise III. Blaise III had just one program, and it needed not be compiled. So Blaise users did not need purchase the Turbo Pascal compiler any more. This program was called the *Data Entry Program* or DEP for short. Its characteristics and behaviour could be changed in a dynamic way. There was no need any more to decide at the design stage what kind of program was required. Here are two of several aspects that could be controlled:

- *Static or Dynamic routing*. Static routing meant that route instructions were interpreted as checks. The Data Entry Program did not force any questions to be answered or skipped. In dynamic routing mode, the Data Entry Program took control of routing through the fields. Fields were processed in the order prescribed by the rules section.

- *Static or dynamic error handling.* Static error handling meant that a detected problem did not interrupt data entry or data editing activities. Errors were reported as error counts displayed adjacent to the input fields. If the Data Entry Program was in dynamic error handling mode, it would stop immediately if a problem was encountered. It was not possible to continue data entry until the problem has been solved.

By combining a routing mode and an error handling mode, different behaviour of the DEP could be invoked. Figure 6.5 shows the DEP in interviewing mode.

Figure 6.5. The DEP in interviewing mode

After data collection, data and meta-data had been brought together. Then, the set of information is ready for further analysis. What the next step would be depended on the survey at hand. It would often require some kind of manipulation of the data. Blaise III had a special tool for data manipulation. It was called *Manipula*. Manipula was a 'Swiss army knife' for data handling. It could combine or split data

files. It could create new data files by selecting variables and/or records. It could derive new variables by means of complex algorithms. It could re-arrange or sort data files, and it could convert data files from one format to another. It supported several data formats including ASCII, Oracle and Paradox. The principle that there should be no data without meta-data also applied to Manipula. In fact, an input data model and an output data model had to be specified, and Manipula took care of transforming the data in such a way that they fit the output model. Manipula required data and metadata as input, and produced data as output.

Blaise III 1.1 introduced in 1994 a new tool called Maniplus. As the name suggests the basis of this tool is Manipula, so all the functionality which was already present in Manipula was also available in Maniplus. Maniplus was described as 'a powerful tool to build applications involving Blaise'. It gave survey system designers the possibility to combine tools like the DEP, the Data Viewer, Manipula, and a DOS Shell into a user-friendly package controlled by user-defined menu systems and dialog boxes including integrated help.

It turned out that Maniplus filled an important gap in the Blaise system and probably because of that it usage quickly became widespread. Over the years major uses of Maniplus included interactive survey control systems such as a CAPI laptop management system (single-, or multi-survey) and in-office data flow control systems. Figure 6.6 show a screen of a Capi laptop management system designed in Blaise III using Maniplus.

Figure 6.6. Example of a Maniplus application screen

A closer look at traditional survey processing would make clear that metadata specifications can come in many forms. The questionnaire form is just one specification, and a setup for a statistical analysis package like SAS or SPSS is another. One design objective of Blaise III was to confront its users with only one type of metadata specification. This did not mean that he could do without other types of metadata specifications. Particularly, if software was used that was not an integral part of the Blaise system, a tool was needed to translate the metadata specification into a form suitable for processing it by the other software. Blaise III had a tool for manipulating metadata. It was called *Cameleon*. It was the successor of the Setup Generator. It supported many metadata formats. Examples are setups for the statistical packages SAS, SPSS and Stata, instructions for building databases in Paradox and Oracle, and specifications for the (built-in) tabulation package Abacus and the weighting package Bascula. For packages not supported

by Cameleon, users could easily design a metadata specification script themselves, and add it to the list.

Another new feature in Blaise III is the possibility to work with multi-lingual questionnaires. If question texts were specified in several languages, interviewers could change to a different language 'on the fly'. This turned out to be a very useful feature in countries where people speak different languages, like Finland (Finnish and Swedish), or the US (English and Spanish).

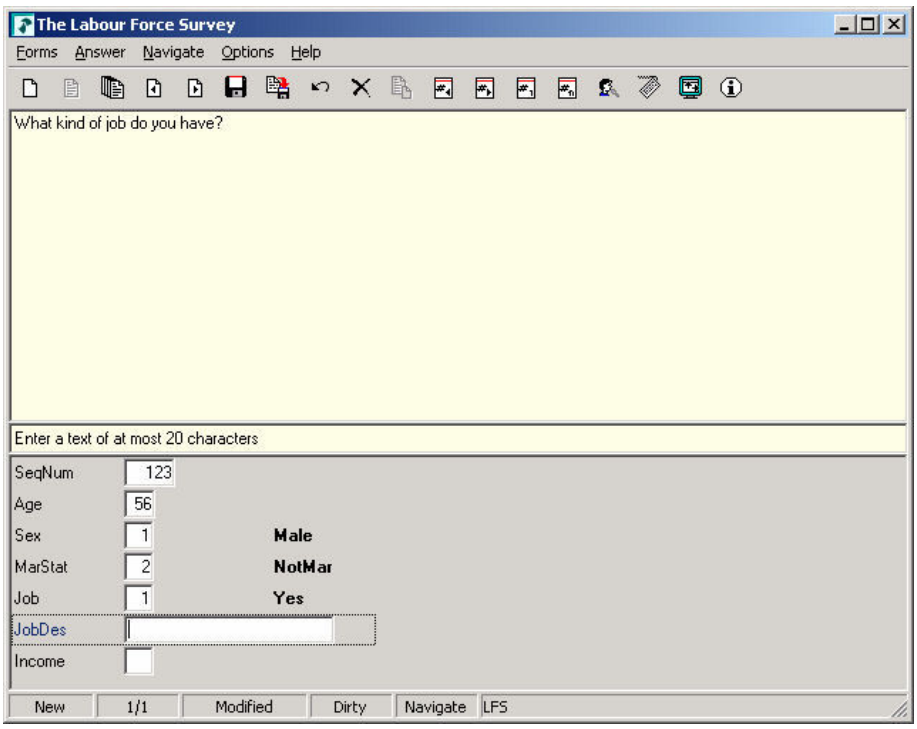
Finally, Blaise III had a tool for translating all texts used by the system into another language. The system was distributed in two languages (English and Dutch), but this option made it possible to make, for example, a French version of the Blaise system. In fact, this translation facility was a kind of version control system for all languages used. It not only helped translating texts, but also simplified the job of updating text files in case a new version of Blaise was distributed.

7. Blaise 4 Windows

Version 4 of Blaise marked the change form MS-DOS to Windows. The first version of Blaise 4 Windows was released in 1998. The functionality of Blaise 4 was basically the same as that of Blaise III. Of course, the graphical user interface offered much more possibilities for screen layout. Figure 7.1 gives an example of a screen of the Blaise 4 DEP in interviewing mode.

Since the introduction of Windows there are many more ways to communicate between applications. The Blaise developers recognised in the late 1990's that it was vital to open up the Blaise system in order to 'survive' in the ever more demanding world of official statistics. This new approach has two main advantages: it is easier to re-use parts of Blaise in other environments and it becomes easier for others to build extensions to the system.

Figure 7.1. The Blaise 4 DEP in interviewing mode.



It started with OBA, the Open Blaise Architecture. The main objective of OBA was to offer an API for all re-usable parts of the system. An *API* (Application Programming Interface) is a set of definitions of the ways one piece of computer software communicates with another. One of the primary purposes of an API is to provide a set of commonly-used functions. Programmers can then take advantage of the API by making use of its functionality, saving them the task of programming everything from scratch.

The most important API in Blaise is the 'Blaise API'. This API offers:

- Access to all metadata of a data model, like question texts, type definitions, rules definitions, etc.
- Read / write access to the data of one form (record).
- Access to the rules engine of Blaise. This makes it, for instance, possible to access all information on 'errors/edits' or to find out what is the next question on the route given a certain question in the data model.

To make something like the Blaise API possible a lot of technology in the background was needed. Blaise used Microsoft COM technology. This is a software architecture developed by Microsoft to build component-based applications.

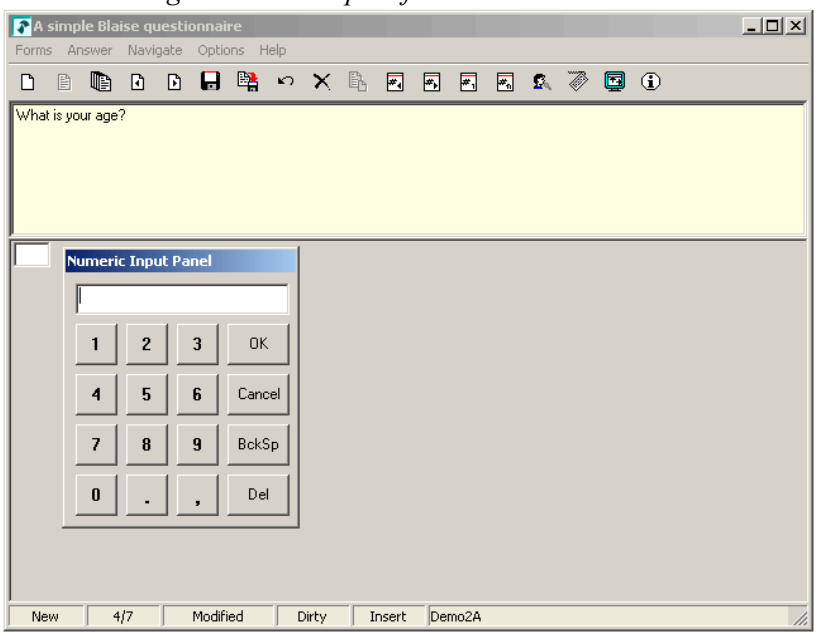
Two examples of the use of the Blaise API are:

- You can make your questionnaire documentation tool, by using the API to extract the relevant metadata from the data model.
- You can build your own Internet application based on Blaise by using the API.

One of the objectives of OBA also was to offer data storage in relational database packages. It meant the collected Blaise data could now directly be stored in many other data formats, for example, in databases systems like Oracle and SQL Server.

At many places the Blaise system offers hooks that can be used to extend its basic functionality. One way to use such hooks is to introduce alternative controls during interviewing. An example of such a control is displayed in figure 7.2. Here a small numeric input panel is displayed that can be used to supply an answer to a numerical question. This can for instance be handy when interviewing is performed using a Tablet PC.

Figure 7.2. Example of an alternative control



OBA was introduced in version 4.5 van Blaise. It was released in 2001. The new set of software tools was called the Blaise Component Pack (BCP).

Now that more and more people are connected to the Internet, web surveys are gaining in popularity amongst researchers as means of data collection. This data collection mode is called computer assisted web interviewing (CAWI) and has many advantages, which include a high response speed, the ability to interact so a respondent can get feedback about questions or possible errors, and leaving it to the respondent when to fill in the questionnaire.

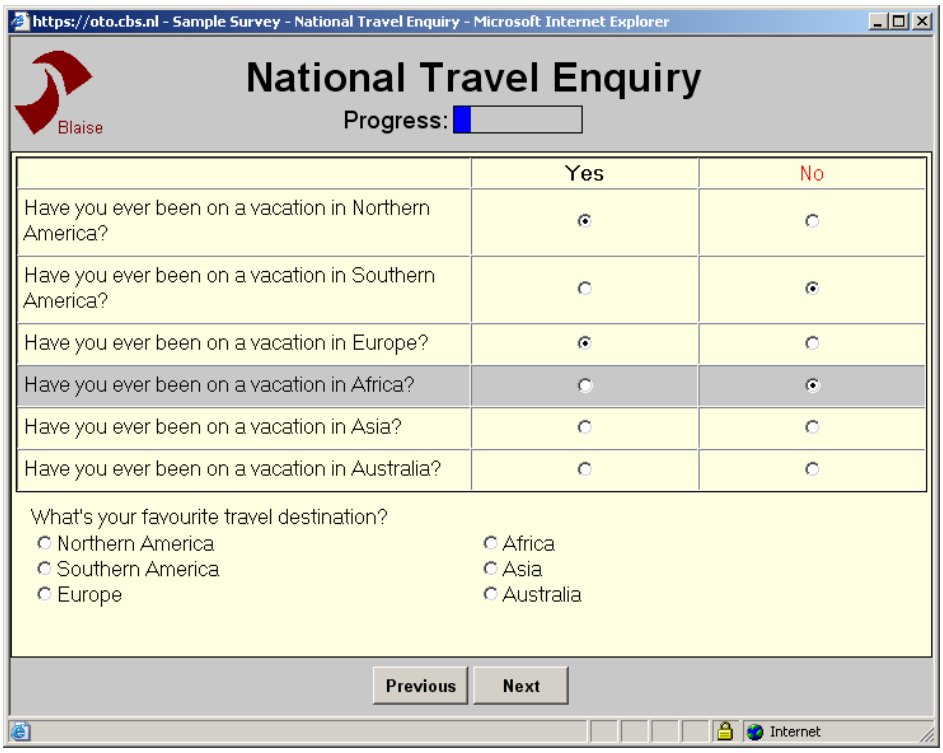
Blaise 4.6 was released in 2003. This version provided a powerful and flexible system for this data collection mode. Blaise Internet lets you choose between two possible approaches for web-based questionnaires. They are called the Interview approach and the Form approach.

The *Interview approach* is typically used for long and complex questionnaires that ask for conditional question routing and checking of questions. They are preferable if skipping or branching patterns are required in the questionnaire, or if controlling the respondent's flow through the questionnaire is important.

A survey using the Internet approach is sometimes called an 'on-line survey'. The reason is that an interview needs (and offers) continuous interaction between the respondent and the internet server. Therefore, it is unavoidable that the respondent is online during the entire interview.

Interviews are page-based, i.e. the questions are grouped into pages. Each page is sent to the respondent as a separate HTML page. After the respondent has answered the questions on the page, the answers are submitted to the internet server. The answers are checked and the applicable rules are interpreted on the server. Then, an adapted HTML page is returned to the respondent based on the answers to the previous questions. See figure 7.3. for an example.

Figure 7.3. A web questionnaire using the Interview approach



The *Form approach* is suitable for short and simple surveys with straightforward data input without question routing. Internet forms are actually paper and pen interviews turned into a version suitable for Internet. The questionnaire comprises one HTML page that can be scrolled up and down to answer the questions.

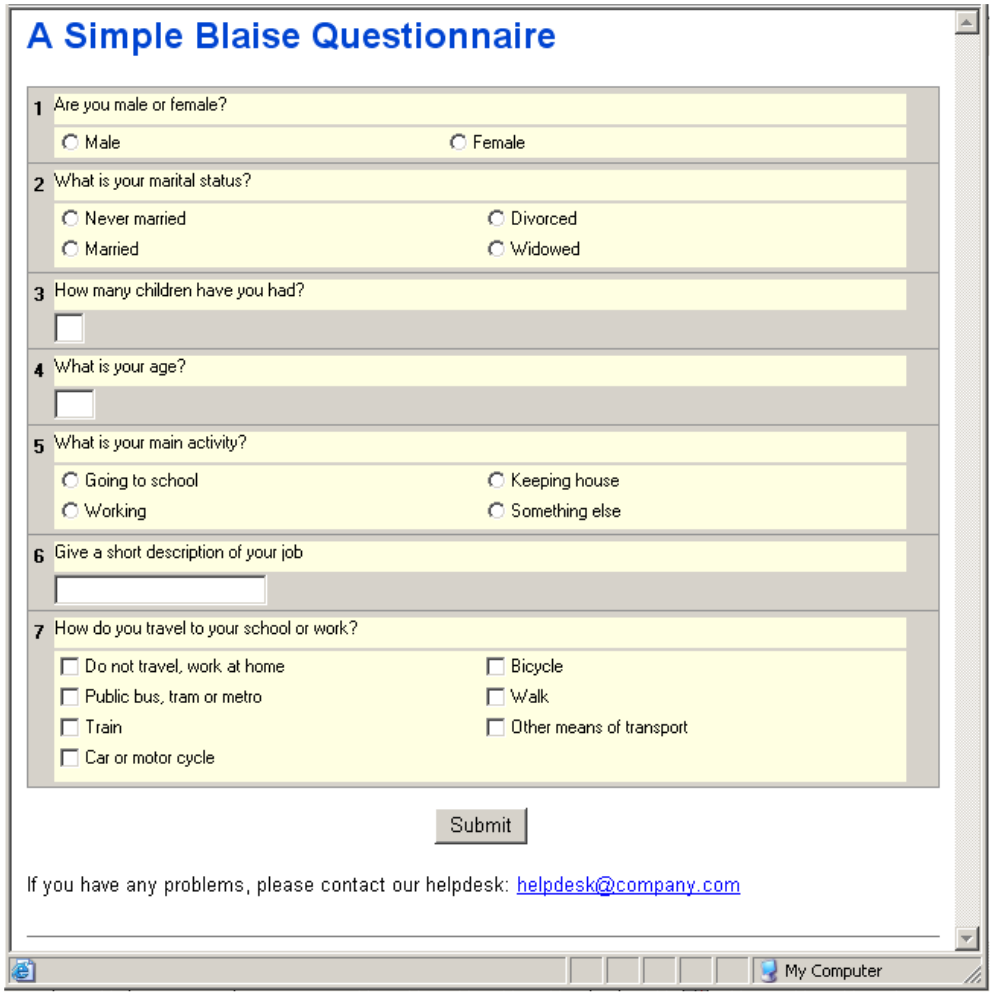
The HTML file may be delivered in any way (for example as an email attachment or as a URL on a website). All questions are presented in a fixed sequence. Respondents can browse through the form and answer questions in any order. They can fill out questions offline, meaning that there is no need for continuous contact between the respondent and the server while they answer questions. However, this doesn't mean the form can not be filled out on-line.

Because there is no contact between the respondent and the server, no routing or checking is possible during the interview. However, the client computer does execute range checks for entered answers.

When the respondent is done and submits his form to the server (after making an internet connection if none is present), the answers are sent to the Blaise Internet server. The server can execute the checking mechanism if desired and stores the data in a Blaise database. The respondent receives an answer from the server.

Figure 7.4. contains an example of a screen of a web survey using the Form approach.

Figure 7.4. A web questionnaire using the Form approach



The layout of an Internet questionnaire is very important. Since no interviewers are available to assist respondents in answering questions and navigating through the questionnaire, everything must be clear for every respondent. The standard Blaise layouts used by the Data Entry Program are not suitable for web surveys.

A web survey is set up with the *Internet Workshop*. This one of the options in the *Tool* menu of the Control Centre of Blaise 4.7. After a Blaise data model has been created, either the Interview approach or the Form approach is selected. The next step is to select an HTML page layout by selecting the proper style sheet. There are several pre-defined style sheets available.

After the survey specification has been completed, the necessary files must be installed on an Internet server. For this, the option *Internet Server Manager* in the Tools menu of the Control Centre can be used.

8. Conclusion

Over the years Blaise has enabled statisticians and others to streamline their statistical production process. Through the common metadata language, and the possibilities to transfer data and metadata from one program to the other, it became easier to process survey data.

Blaise has proved to provide a useful and efficient environment for survey processing. It is not only the standard package for Statistics Netherlands, but it is also in wide scale use at more than 120 other statistical agencies, universities, and research agencies elsewhere in the world. That Blaise is not a simple system for small scale surveys is proved by the fact the system was selected for processing the first agricultural census of the Peoples Republic of China, where approximately 3 million interviewers were to collect data on approximately 200 million farms. Unfortunately the project was stopped due to internal political struggle in China.

It is surprising to see that ideas behind Blaise have survived now for 20 years, even in this quickly changing world of processing statistical information. The first ideas emerged in an era of small, slow stand-alone computers running under MS-DOS, but these ideas still turn out to be valuable in our current Internet era. Blaise was alive and kicking for 20 years and holds a promise to remain doing so in the coming years.

9. References

Van Bastelaer, A.M.L. and Sikkel, D. (1987), “From three to three hundred hand-held computers,” in *CBS select 4, Automation in Survey Processing*, Voorburg: Statistics Netherlands, pp. 13-26.

Van Bastelaer, A.M.L., Kerssemakers, F.A.M. and Sikkel, D. (1987), “A test of the Continuous Labour Force Survey with hand-held computers, interviewer behaviour and data quality,” in *CBS select 4, Automation in Survey Processing*, Voorburg: Statistics Netherlands, pp. 37-54.

Bemelmans-Spork, E.J. and D. Sikkel (1985a): Observation of prices with hand-held computers, Statistical Journal of the United Nations Economic Commission for Europe, vol. 3, no.2.

Bemelmans-Spork, E.J. and D. Sikkel (1985b): Data Collection with hand-held computers, Proceedings of the 45th session, International Statistical Institute, Book III, topic 18.3.

Bethlehem, J.G. (1987), "The Data Editing Research Project of the Netherlands Central Bureau of Statistics," *Proceedings of the Third Annual Research Conference of the US Bureau of the Census*, Washington: Bureau of the Census, pp. 194-203.

Bethlehem, J.G. (1997): Integrated Control Systems for Survey Processing. In: Lyberg, L., Biemer, P., Collins, M., De Leeuw, E., Dippo, C. and Schwarz, N. (Eds), *Survey Measurement and Process Control*, Wiley, New York, pp. 371-392.

Deming, W.E. (1986), *Out of the Crisis*, Cambridge: Cambridge University Press.
Hammer, M. and Champy, J. (1993*), Reengineering the Corporation, A Manifesto for Business Revolution*. London: Nicholas Brealey Publishing.

Hammer, M. and Champy, J. (1993*), Reengineering the Corporation, A Manifesto for Business Revolution*. London: Nicholas Brealey Publishing.

Session 2.1: Standardising

In-house Support for Blaise Application Developers and Users
Janica Cajhen & Pavle Kozjek
(Statistical Office of the Republic of Slovenia, SORS)

Blaise Programming Techniques for Better Documentation
Peter Sparks & James Hagerman
(University of Michigan, USA)

Standardizing and Automating Blaise Output Test Data Delivery (TransSAS)
Latha Srinivasamohan
(U.S. Census Bureau)

Developing an Integrated Household Survey (IHS) Blaise Questionnaire from Four Major Social Surveys
Tim Burrell
(Office For National Statistics, UK)