Update from "Downunder"; History, plans and functions we've built for CAI and Blaise in Australia

Fred Wensing and the CAI team at ABS

1. History and plans for CAI and Blaise III

1993

In late 1993, the business case was accepted for introduction of computer assisted interviewing (CAI) in the ABS household surveys program subject to a comprehensive test being conducted. Previous considerations of CAI had concluded that systems were not advanced enough and cost savings had also been made through the introduction of optical mark reading (OMR) for household survey forms in the late 1980's.

1994

A review of the available software and an evaluation of the potential of Blaise III led to a decision to adopt Blaise as the data collection tool for CAI. Blaise was seen as suitable for ABS household surveys due to its language characteristics (simple syntax and block structure), its metadata-centered approach (shared by all the tools) and the availability of data and metadata conversions to interface with the ABS environment. The main limitation of Blaise III was (and still is) the absence of coding facilities.

A trial of CAI using a Beta-test version of Blaise III was carried out in July 1994 using the Household Expenditure Survey (HES). The HES was deliberately chosen as it was a large survey questionnaire involving more than 800 questions which needed around 12,000 lines of Blaise. The trial was conducted on a sample of 450 households using 10 interviewers who were each equipped with a 486 notebook computer (monochrome screen). The test also included the transmission of data via E-mail. The trial was successful and supported the claims that CAI would improve data quality and timeliness and lead to some saving in costs. The trial also showed that it was relatively easy to introduce CAI to an interviewer workforce who have had very little computing experience. Respondents were not at all concerned by the use of computers in the field.

1995 and the future for CAI

The ABS has now embarked on a full systems redevelopment project which will see CAI (using Blaise III) used in the Monthly Population Survey program (sample size of 30,000 households, involving 560 interviewers) as from October 1997 to coincide with a major post-census revision of the sample. CAI will also be used in our program of special supplementary surveys commencing with the longitudinal Survey of Employment and Unemployment Patterns (sample size of 10,000 persons) starting

with the second wave in September 1996. Developments also include the introduction of telephone interviewing for the Monthly Population Survey in mid 1996. Between the start of 1995 and full implementation in late 1997, the ABS will conduct more than 10 tests of CAI including one major test of 6 months duration.

2. Household Surveys Facilities being developed

The introduction of CAI provides the opportunity to redevelop all the facilities for household surveys in the ABS. A combined team of programmers and survey developers has been established to carry out the developments over the next three years.

The main features of the facilities to be developed include:

- Blaise (version III) will be used as the main data capture tool
- Survey data is expected to be transferred to the client-server (Oracle-Unix) environment so that it can be integrated with other data stores and systems, particularly the ABS data warehouse and generalised survey processing interface
- Links will also be built to enable Blaise metadata to be transferred to and from the data warehouse
- Interviewer workload management facilities are being developed in Turbo Pascal and are likely to be redeveloped using Maniplus
- Office management facilities are likely to be built using Oracle
- Office based coding facilities will make use of Windows-based computer assisted coding facilities
- Office based processing and estimation will make use of SAS
- Tabulation and output processes will make use of other corporate tools

Some specific functions we have developed for CAI are outlined in the following sections.

3. Keyboard remapping using C

The Blaise data entry program allows the remapping of some of the keys on the keyboard to perform common functions required during a data entry session. Unfortunately it does not allow remapping of some of the most commonly used keys such as PgUp, PgDn and Alt-X.

Some sub-notebook computers due to the small size of their keyboards use the Fn key in combination with other keys to produce these key-presses. This often requires computer users to use two hands to activate a function.

To ease the job of our interviewers (who often perform interviews standing at the door) we have produced a C program to remap these keys to function keys, which can be pressed using one hand

The C program, once run, remains in memory and intercepts user key presses. If the key pressed is one to be remapped, the program converts the keystrokes and outputs the converted key-presses. In this way the program can be used to remap any key on the keyboard to any other.

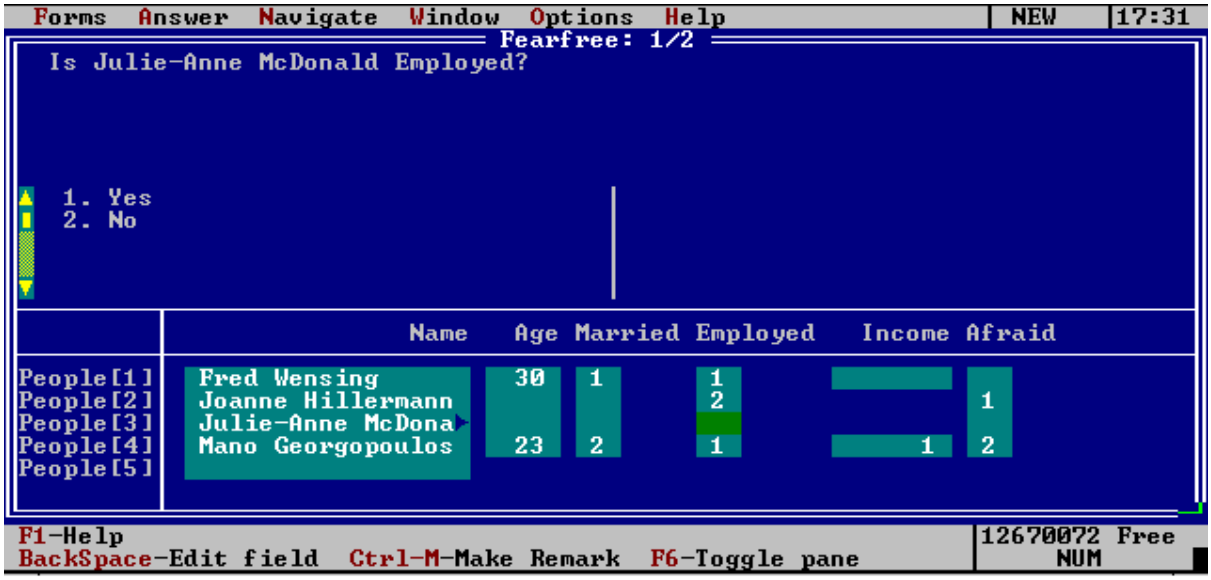
Listing of KEY.C

```
#pragma -mc                          // force compact memory model

#include <dos.h>

void    interrupt int_9();
void    interrupt (*old_int9)(void);
void    main(void);
void    tsr(unsigned size);
void    find_dos_active_flag(void);

// Run only when program is first run
void main(void)
{       disable();                   // Disable interrupts
        old_int9 = getvect(0x09);    // Save old keyboard processing routine
        setvect(0x09,int_9);  // Set new keyboard processing routine
        enable();                    // Enable interrupts
        tsr(1000);                   // Allocate some memory?
}

// Allocate some memory
void tsr(unsigned size)
{       union REGS r;
        r.h.ah = 49;
        r.h.al = 0;
        r.x.dx = size;
        int86(0x21,&r, &r);
}

// Our keyboard routine
void interrupt int_9(void)
{       char far *keybuf = (char far *) 1050;  // head pointer of keyboard
        (*old_int9)();               // use old keyboard routines first
        if(*keybuf != *(keybuf+2))           // if buffer not empty
        {       keybuf += *keybuf-30+5;      // go to character position
                // Key = F7
                if (*keybuf == 65)
                {       // Change key to PGDN
                        *keybuf = 81;
                }
                // Key = F8
                else if(*keybuf == 66)
                {       // Change key to PGUP
                        *keybuf = 73;
                }
                // Key = F3
                else if(*keybuf == 61)
                {       // Change key to ALT-X
                        *keybuf = 45;
                }
                // Key = Ctrl-F3
                else if(*keybuf == 96)
                {       // Change key to F3
                        *keybuf = 61;
                }
        }
}
```

4. Freedom of entry in Blaise tables

In our paper-based household surveys we use a separate household form to collect basic demographic information about all members of a household before collecting information from the individuals. The household form is in a table format, which allows the interviewer to collect information in any order that suits the interview situation. For example, all information about one individual may be collected first, or the same data item may be collected for all members of the household.

We have developed a method for implementing this freedom of data collection in a Blaise table. This is achieved by allowing all fields in the table to be "Empty" and so allowing the interviewers to move around the form as they like. In order to ensure no fields are accidentally left empty we have implemented an edit mechanism to check that all required data has been entered.



Once the final question on the household form has been answered, various edits are triggered to ensure that all fields on the household form that should have been answered have had values entered.

Listing of FEARFREE.BLA

```
DATAMODEL Fear "The Fear Survey"
    BLOCK BPerson "One person"
         FIELDS
              Name "What is your name?"             : STRING[20], EMPTY
              Age "What is the age of ^Name?"       : 0..120    , EMPTY
              Married "Is ^Name married?"           : (Yes, No) , EMPTY
              Employed "Is ^Name Employed?"         : (Yes, No) , EMPTY
              Income "What is the income of ^Name?" : 0..100000 , EMPTY
              Afraid "Is ^Name ever afraid?"        : (Yes, No) , EMPTY
          RULES
               Name
               IF Name = RESPONSE THEN
                     Age
                     IF CheckTable = RESPONSE THEN
                         CHECK
                         Age = Response "Missing value for age"
                     ENDIF
                     Married
                     IF CheckTable = RESPONSE THEN
                         CHECK
                         Married = Response "Missing value for married"
                     ENDIF
                     IF (Age = RESPONSE) AND (Married = RESPONSE) THEN
                         IF Age < 15 THEN
                              CHECK
                              Married = No "Cannot be married under 15"
                         ENDIF
                     ENDIF
                     Employed
                     IF CheckTable = RESPONSE THEN
                          CHECK
                          Employed = Response "Missing value for employed"
                     ENDIF
                     IF Employed = Yes THEN
                          Income
                          IF CheckTable = RESPONSE THEN
                               CHECK
                               Income = Response "Missing value for income"
                          ENDIF
                     ENDIF
                     Afraid
                     IF CheckTable = RESPONSE THEN
                         CHECK
                         Afraid = Response "Missing value for afraid"
                     ENDIF
                ENDIF
    ENDBLOCK
    TABLE BPeople
         LOCALS
              Count : INTEGER
         FIELDS
              People : ARRAY [1..5] OF BPerson
         RULES
              FOR Count := 1 TO 5 DO
                   People[Count]
              ENDDO
    ENDTABLE
    FIELDS
         People : BPeople
         CheckTable "Press '1' to check table" : (Press1 "Press '1'")
    RULES
         CheckTable.KEEP
         People
         CheckTable
ENDMODEL
```

5. Producing a list of edits from a Blaise program

It is often useful to produce a list of edits implemented in a Blaise program. This report can be used by programmers and subject areas to verify that all specified edits have been implemented.

To do this we have written a C program which parses the source code of a Blaise program and outputs a list of the edits found. The output from this program can be loaded into a spreadsheet and used to produce tabular output showing the edits in the Blaise source.

This program relies on the Blaise coding conventions being used in the ABS to produce the list. These programming conventions require that the "CHECK" or "SIGNAL" keywords must be immediately preceding an edit specification in the program. The program searches though the source of a Blaise questionnaire, for the "CHECK" and "SIGNAL" keywords, and interprets the following statements as the edit condition and message specifications.

This program currently only outputs the condition in the edit. A version of this program is currently being developed which also outputs the scope of the edit from the surrounding IF statements. This new version will be written in Manipula rather than C to integrate better with the Blaise suite of software.

Listing of FEAR.BLA

```
DATAMODEL Fear "The Fear Survey"
    FIELDS
         Name "What is your name?"
              : STRING[20]
         Age "What is your age?"
              : 0..120
         Married "Are you married?"
              : (Yes "Yes",
                 No "No")
         Income "What is your monthly income?"
              : 0..100000
         Afraid "Are ^Name ever afraid?"
              : (Yes "Yes",
                 No "No")
         WhatAfraid "WHAT IS ^Name AFRAID OF?"
              : SET OF (Monsters "Monsters",
                        Dark     "The Dark")

    RULES
         Name
         Age
         Income
         SIGNAL
              Income <= 5000
              "Please confirm that income is more than 5000"
         Married
         IF Married = Yes THEN
              CHECK
                   Age >= 15
                   "Must be over 15 years of age to be married"
```

```
        ENDIF
        Afraid
        IF Afraid = Yes THEN WhatAfraid ENDIF
ENDMODEL
```

Output from the Check program:

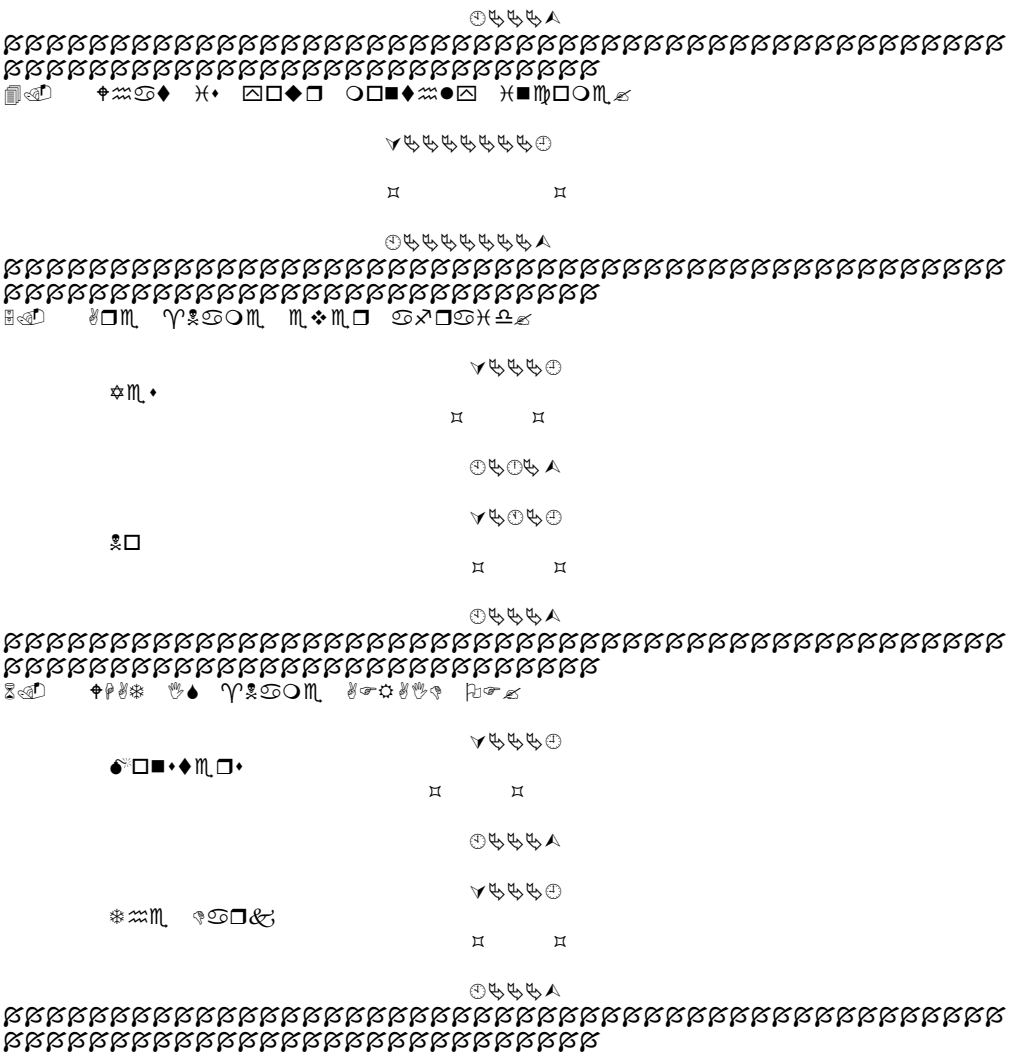
File	Type	Condition	Message
FEAR.BLA	SIGNAL	Income <= 5000	Please confirm that income is more than 5000
FEAR.BLA	CHECK	Age >= 15	Must be over 15 years of age to be married

6. Generating a paper image of a Blaise questionnaire

We consider that a paper version of a questionnaire is a useful component of the CAI environment and have developed a program that generates such a document.

The paper questionnaire generator that we have developed uses Cameleon to access the meta-data of a Blaise questionnaire. Cameleon outputs this meta-data to an intermediate file in a format similar to our current paper questionnaire standards. A Manipula program then processes this intermediate file to make the questionnaire more visually appealing.

Sample output from paper generator using the Fear model from section 5.



This paper questionnaire generator makes no attempt to interpret replacement words in the question text and contains none of the sequencing instructions. Given that future versions of Cameleon may be able to access the "rules" in a Blaise program, these enhancements may be possible later.

7. Interviewer Workload Management System in Turbo Pascal

We have written an Interviewer Workload Management System (IWMS) in Turbo Pascal to enable interviewers to manage their interviews and interact with Blaise.

It allows interviewers to:

- select a survey to enumerate (allows for multiple surveys)
- list the households at which they need to interview
- select which household to view based on completion status;
- sort the list of households by various criteria;
- make interview appointments for households;
- call up the DEP program to perform an interview for a household;

- register calls made to households;
- list summary information about households;
- monitor the progress of their workload;
- call up transmission software to transfer data to the office; and
- access other functions such as training and keyboard familiarisation.

The IWMS was developed in Turbo Pascal so that the look and feel is similar to the Blaise suite. It replaces previous facilities developed using Boreas and may be rewritten in Maniplus when available.

How it works

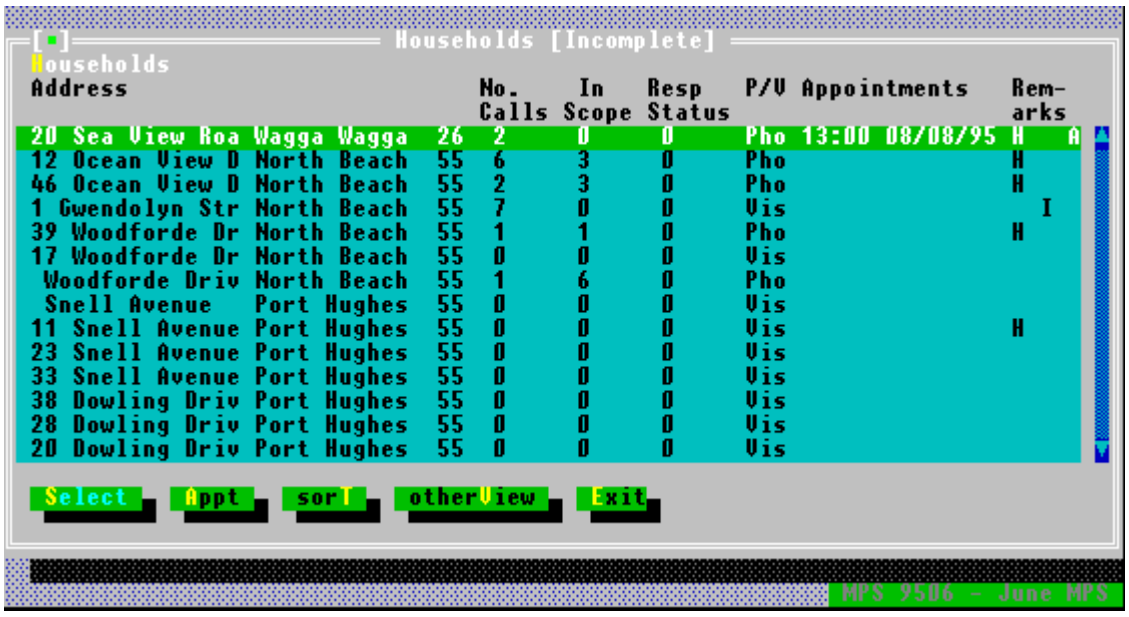
The Blaise files and manipula programs are stored in a separate directory on the notebook for each survey being enumerated. Information used within the IWMS which is not part of the interview process, such as calls and appointments, is stored in text files in the same directory.

Initial information for each interviewer's workload is loaded into a Blaise data file in the office using Manipula. This information includes indicative, address and any data to be included from previous interviews.

In the field, the IWMS extracts this information using Manipula to create the pick list of households (see Screen A).

When a household is selected a more detailed screen of information appears (see Screen B). From this screen the interviewer can call up the Blaise data entry program (DEP) to perform the interview.

On completion of the interview, the IWMS runs Manipula again to extract the household list so that any updated information is reflected within the IWMS.



8. Program to assist with the transfer of Blaise data toRelational databases

We have developed a Cameleon program that converts questionnaire data from a Blaise data structure to a relational database structure.

The structure of the relational database table generated is as follows:

Blaise record key	Blaise field name	Value of field

The generalised table structure has been devised as being suitable for storing of data from any Blaise collected data, without regard to the underlying structure of the Blaise questionnaire. From this basic structure the data can be readily transformed into other table structures using SQL statements prior to further processing.

The system uses Cameleon to extract the meta-data from a Blaise questionnaire, and to produce a Manipula program. The Manipula program when executed extracts data collected using the Blaise questionnaire and writes an SQL program which inserts this data into the relational database table.

Warning: With large Blaise datamodels, it is very easy to produce Manipula programs that are too big to execute.

Listing of BLSE2RDB.CIF

```
[OutFile:= '.\Extract.Man']
USES
    DATAMODEL BSql
```

```
         FIELDS
              OutRec
                   : STRING[[255]]
    ENDMODEL

INPUTFILE IFile : BFile ("input", BLAISE3)

OUTPUTFILE OFile : BSql ("Extract.SQL", ASCII)
SETTINGS
        TRAILINGSPACES = No

MANIPULATE
    DISPLAY (KEY(IFile))
[BlockProc]
    [QuestionLoop]
         [ArrayLoop]
              [SetLoop]
                   [If QuestionType = Block Then]
                        [BlockCall]
                   [Else]
    IF [FullQuestionName] <> EMPTY THEN
         OutRec := 'INSERT INTO [QuestionnaireName] VALUES ('''
                   + KEY(IFile)
                   + ''', ''[FullQuestionName]'', '''
                        [If QuestionType = String Then]
                   + [FullQuestionName][&]
                        [ElseIf QuestionType = Date Then]
                   + DATETOSTR([FullQuestionName])[&]
                        [ElseIf QuestionType = Time Then]
                   + TIMETOSTR([FullQuestionName])[&]
                        [Else]
                   + STR([FullQuestionName])[&]
                        [EndIf]
 + ''');'
         WRITE( OFile )
    ENDIF
                   [EndIf]
              [EndSetLoop]
         [EndArrayLoop]
    [EndQuestionLoop]
[EndBlockProc]
ENDWHILE
```

Note. Copies of the programs and other sample code can be made available on disk or by E-mail (Contact: cai.team@abs.telememo.au or fax +61-6-2525281)