Understanding the Blaise Selective Checking Mechanism and the Appropriate Use of KEEP Statements

Pamela Ford, Statistics Canada

1. Background

In 1998 Statistics Canada adopted Blaise as the standard development tool for survey collection instruments.

As described by Mayda and Ford (2003), Computer Assisted Personal Interviewing (CAPI) applications at Statistics Canada consist of a combination of Blaise and Visual Basic components. Due to time constraints for the initial implementation of these applications, a pre-existing Visual Basic component was used for the collection of demographic and relationship information. The introduction of full Blaise applications for Computer Assisted Telephone Interviewing (CATI) operations in Regional call centres meant that demographic and relationship components would have to be developed in Blaise. Some of the initial prototypes that were developed and implemented had simpler requirements than the existing CAPI surveys wanted. The more complex prototypes suffered from noticeable performance issues.

With the move toward CAPI CATI Integration of the Labour Force Survey, it became clear that a more robust Blaise membership and demographic component that could be used by this and other surveys would have to be developed. This paper details the development process and the lessons learned.

2. Household Membership - Specification

In birth interviews where no previous information is available, membership data is collected in three rosters. In the first roster, interviewers enter names of those who *usually* live or stay in the dwelling. Next the interviewer is prompted to ask whether there is anybody temporarily staying in the dwelling and if so a second roster is used to collect the names of these *temporary* members. Finally the interviewer is prompted to ask for any persons who might not otherwise have been accounted for. Names of these *other* members are entered in a third roster. To minimize confusion for interviewers, all members listed are shown in all three rosters, giving the impression that only one roster is being manipulated.

In subsequent interviews, the first roster presented is pre-filled with names collected in *previous* collection cycles. A third question on the roster is used to indicate whether the members still reside in the dwelling. Once this data is confirmed and/or updated, the interviewer is prompted to ask for names of *other* members who have moved into the dwelling since the last collection cycle. Some surveys have specified that a special 'ghost' roster be used to display names of members who have lived in the household/dwelling at one time but were not present during the most recent collection cycle.

In both birth and subsequent interviews the member rosters are followed by yet another roster which is used to confirm, update, and/or collect demographic data.

The final step in this process is to determine each member's relationship to every other household member. This information is collected in a matrix format.

Additional complexities include:

- having to display all member names in any or all rosters
- being able to delete members entered erroneously
- being able to correct the spelling of previous members' names while ensuring that they aren't being inappropriately replaced completely,
- being able to copy the last name of a higher row (using the row #)
- being able to do both birth and subsequent interviews within the same application.

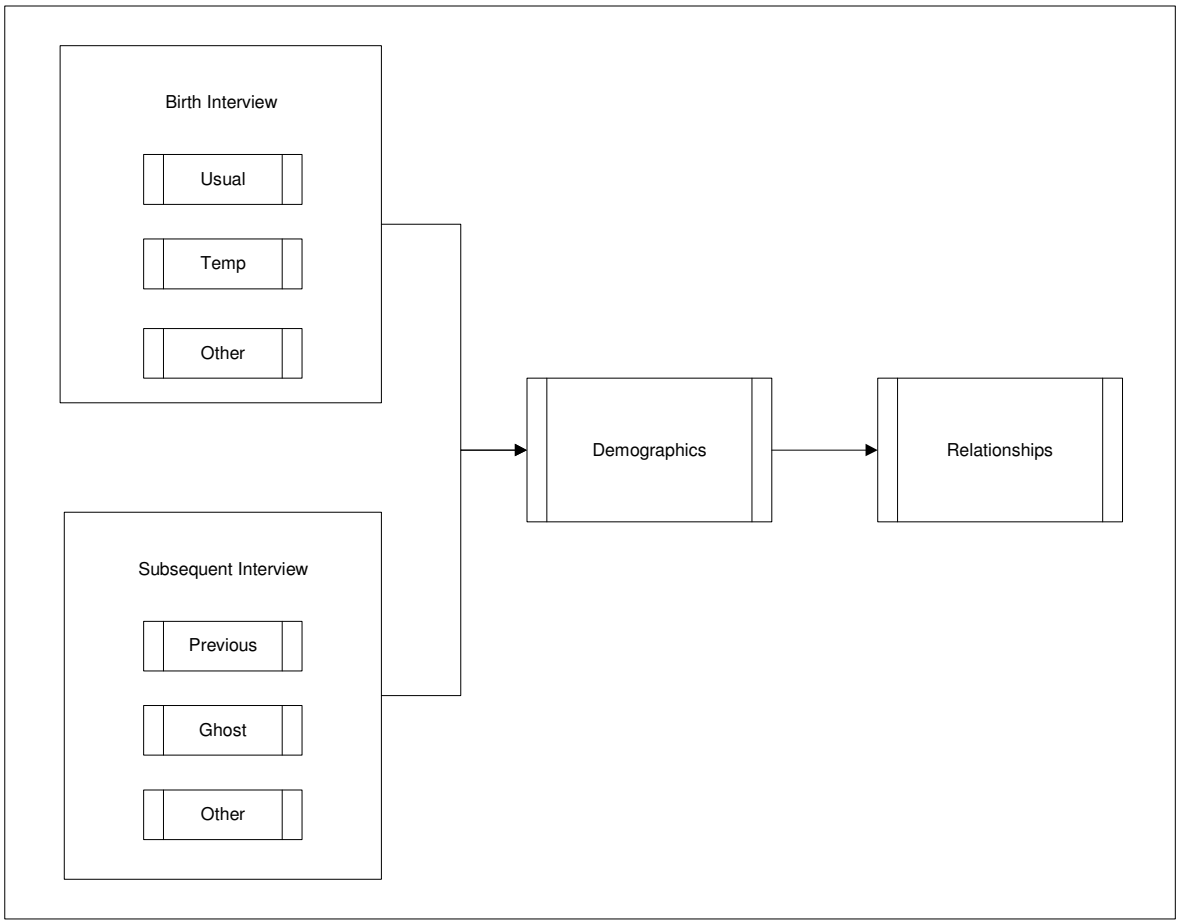

Figure 1 Household Membership Specification

3. Household Membership - Analysis and Development

3.1 Base Design

Having to display all members on every roster led to a number of key design principles.

First, there would have to be a one-to-one relationship across all rosters, whereby a person record would occupy the same line in any roster in which it is shown or asked.

Second, a *Master* roster was conceived to simplify the process of having to update all rosters. That is, data collected in each roster would be copied to the master, and the master would be the sole source for internal updates, including the updating of names in the demographic roster.

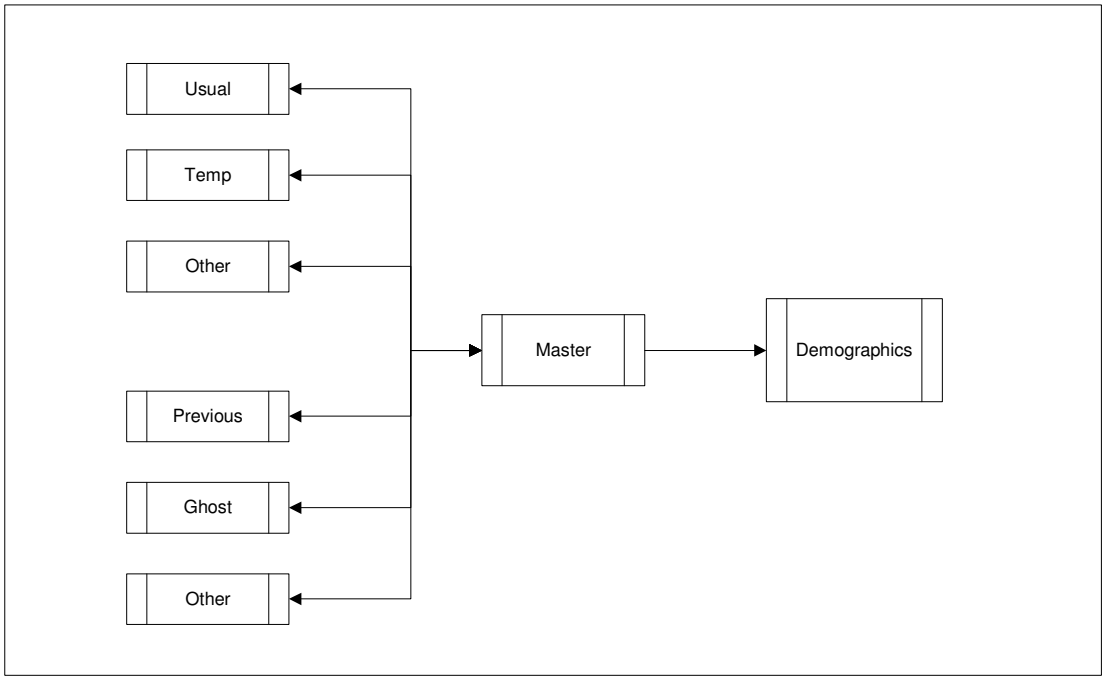

Figure 2 - Master Roster Model

Third, the existence of prompt questions before the Ghost, Temporary, and Other Rosters (Rosters are only on path if 'Yes' is answered to the prompt questions) meant that code had to be included to ensure consistency (If 'Yes' is answered to the prompt, then an edit would ensure that at least one member was enter in the associated roster. If 'No' is answered to the prompt, then any members entered in the associated roster would have to be deleted).

Finally, a 20 household member limit was set and it was necessary to ensure that no further prompts or rosters are presented in the stunningly rare instances where the limit is actually hit.

3.2 The Selective Checking Mechanism and the Placement of KEEP Statements

With the knowledge that Blaise always checks all appropriate rules in an instrument (Pierzchala and Razoux Schultz, 1996), it was anticipated that the continuous updating of the demographic roster could lead to performance degradation. This was managed by including logic which prevents the updating of the demographic roster and relationship matrix until the collection/confirmation of member names is complete.

The most challenging aspect of the development process proved to be the utilization and placement of KEEP statements. In early prototypes, by showing 'block checks' in the watch window, it was obvious that KEEP statements were either excessive or misplaced. Block-level KEEP statements were required to ensure that previous cycle data loaded ahead of time was retained, and to ensure that certain field values were known prior to their being used in rules logic, but how could these be implemented so as to minimize the checking of block rules?

To gain a better understanding of the KEEP functionality of Blaise, a simple datamodel was created and various arrangements of block level KEEP statements were explored. In the course of this research, an anomaly with the use of the <End Key> was uncovered which added further complication until Blaise 4.6 became available.

```
RULES
   {
    Model 1:  Base case, No KEEP Statements
     - Since there are no KEEP statements we risk losing data if the case is re-entered and the
       answer of BlockA.A4.SB1_F1 changed so that blocks B,C, and D are no longer on the path.
       This flow is comparable to the situation where an interviewer re-attempts a case to complete
       an interview and answers 'No' to 'Have you made contact?' in the event of a no-answer or
       busy etc. dial result  We must still KEEP the subject matter component in this situation.
   }

   BlockA
   IF BlockA.A4.SB1_F1 = Yes THEN
      BlockB
      BlockC
      IF BlockC.C4.SB1_F1 = Yes THEN
         BlockD
      ENDIF
   ENDIF
   BlockE
```

Figure 3 No KEEP Statements

```
RULES
   {
    Model 2:  KEEP statements at the beginning
     - The result of the KEEP statements at the beginning is that at times, block rules are
       unnecessarily checked twice for some blocks.
     - N.B. In V453B650, using the endkey can result in blocks/fields being skipped.
   }
   BlockB.KEEP
   BlockC.KEEP
   BlockD.KEEP
   BlockA
   IF BlockA.A4.SB1_F1 = Yes THEN
      BlockB
      BlockC
      IF BlockC.C4.SB1_F1 = Yes THEN
         BlockD
      ENDIF
   ENDIF
   BlockE
```

Figure 4 KEEP Statements at Beginning

```
RULES
   {
    Model 3:  KEEP statements in ELSE sections
     - Here the blocks are only checked once as appropriate.
     - N.B. In V453B650, using the endkey can result in fields being skipped.
   }
   BlockA
   IF BlockA.A4.SB1_F1 = Yes THEN
      BlockB
      BlockC
      IF BlockC.C4.SB1_F1 = Yes THEN
         BlockD
      ELSE
         BlockD.KEEP
      ENDIF
   ELSE
      BlockB.KEEP
      BlockC.KEEP
      BlockD.KEEP
   ENDIF
   BlockE
```

Figure 5 If ASK Else KEEP

```
RULES
   {
    Model 4:  KEEP statements on path only if last question is answered.
     - Here the blocks are only checked once as appropriate.  (N.B. If BlockE.Bye were not
       an auxfield, it would have to be initialized to EMPTY the first time through the rules.)
     - This is a viable work-around for the End-Key issue in V453B650.
   }
   BlockA
   IF BlockA.A4.SB1_F1 = Yes THEN
      BlockB
      BlockC
      IF BlockC.C4.SB1_F1 = Yes THEN
         BlockD
      ENDIF
   ENDIF
   BlockE
   IF BlockE.Bye = Ciao THEN
      BlockB.KEEP
      BlockC.KEEP
      BlockD.KEEP
   ENDIF
```

Figure 6 KEEP if last question answered

From this research, it was obvious that the If-ASK-Else-KEEP approach should be used wherever possible. Though there are several instances where this method was not feasible, the checking of rules within the roster component has been optimized as follows:

- At the highest level, the Roster block is kept only if the 'ASK' is off path.
- A KEEP statement for each of the Usual, Temp, Other, Previous, and Ghost tables is included at the beginning of the Roster rules as much of the block logic is dependant on their contents being known.
- A KEEP statement is applied for the Demographic block, only if the ASK is off path.
- No KEEP statement was required at all for the Relationship block.

4. Lessons Learned

The development of this complex component not only provided further insight into the usage of KEEP statements, but also reinforced existing knowledge of how block parameters should be implemented. Furthermore, it demonstrated how crucial a step prototyping can be in the development process.

It is critical that developers understand the Blaise selective checking mechanism and the impact of both KEEP statements and parameters so that the performance of their instruments will be optimized. The 'Show Block Checks' option in the watch window proved to be an invaluable tool and its use by developers will be encouraged throughout Blaise development areas.

5. Conclusion and Future Direction

A strategic streamlining initiative at Statistics Canada has resulted in the specification and development of standard front and backend templates to be used with all social survey applications. Although the method of multiple rosters was proven to be satisfactory in terms of performance and addressing client needs, it was recommended that the household membership component of the standard templates consist of a single roster for entering member names. This is considered to be a better approach as it will allow for simpler code that can be maintained by a wider range of developers.

6. References

Mayda, J. and Ford, P. (2003). Integration of CAPI and CATI Infrastructures at Statistics Canada. Proceedings of the Eighth International Blaise Users Conference, Denmark

Pierzchala, M. and Razoux Schultz, G. (1996). Optimal Instrument Performance in Blaise III. International BLAISE User Group Newsletter, no 8, pp. 11-18, ONS, London publisher.