A Different Approach to Blaise Remarks

Peter Stegehuis, Westat

Introduction

Interviewer comments, or remarks in Blaise, are a useful way for field interviewers to record information that may be necessary for the data editing/cleaning stage. Remarks may be made when interviewers are not sure what to do with information provided by the respondent. Also, during long and complicated interviews it may not always be feasible to back up to the question for which the answer may need to be changed, so a comment might be the only way to relate important details.

Triaging all the Blaise remarks can be a very time-consuming and costly task. A Blaise remark is just a text string, so there is no structure or categorization possible. Reading every single remark is the only way to determine whether or not the remark contains actionable information. Prioritizing - or only dealing with - certain categories of comments before reading all remarks is virtually impossible, as categorization cannot happen without reading the remarks in the first place.

This paper shows a solution we have implemented in Blaise 4.8, using the integrated functionality of the DEP and Manipula. This approach retains the very useful elements of Blaise remarks: the interviewer can add or edit a remark at any question, and a paperclip icon shows the existence of a remark. But it adds a new element: the interviewer has to select a category before adding the comment itself. The comments get stored in their own data file, with the comment category as a separate variable. This new variable can subsequently be used as needed during the data editing stage.

Why make remarks?

Ideally, a respondent hears an interviewer's question, remembers all relevant information instantly and answer correctly all the time. Alas, during interviews in real life however it happens frequently that additional and potential answer-altering information gets divulged well after the question has already been answered.

In some cases the interviewer cannot back up to the question for which the answer should now be changed. For instance if an interview starts with eligibility questions to determine who in the household will get the follow-up questions, it is common to disallow backing up into that section. If one of the answers in that section needs to be changed based on information that becomes available after passing that 'wall' then a Blaise remark may be the best option the interviewer has.

In other instances it may simply take too much time for the interviewer to back up multiple screens and change an answer. The risk is always there that a respondent will take the opportunity of this pause in active interviewing to end the cooperation, and with it stop the whole interview. A remark at the current question is much more easily made, with less risk for an untimely end of the interview.

A third reason for adding a remark, especially during a long interview, can be when it is hard to check what has been entered earlier in the interview.

Here is a screenshot of a standard Blaise remark popup:

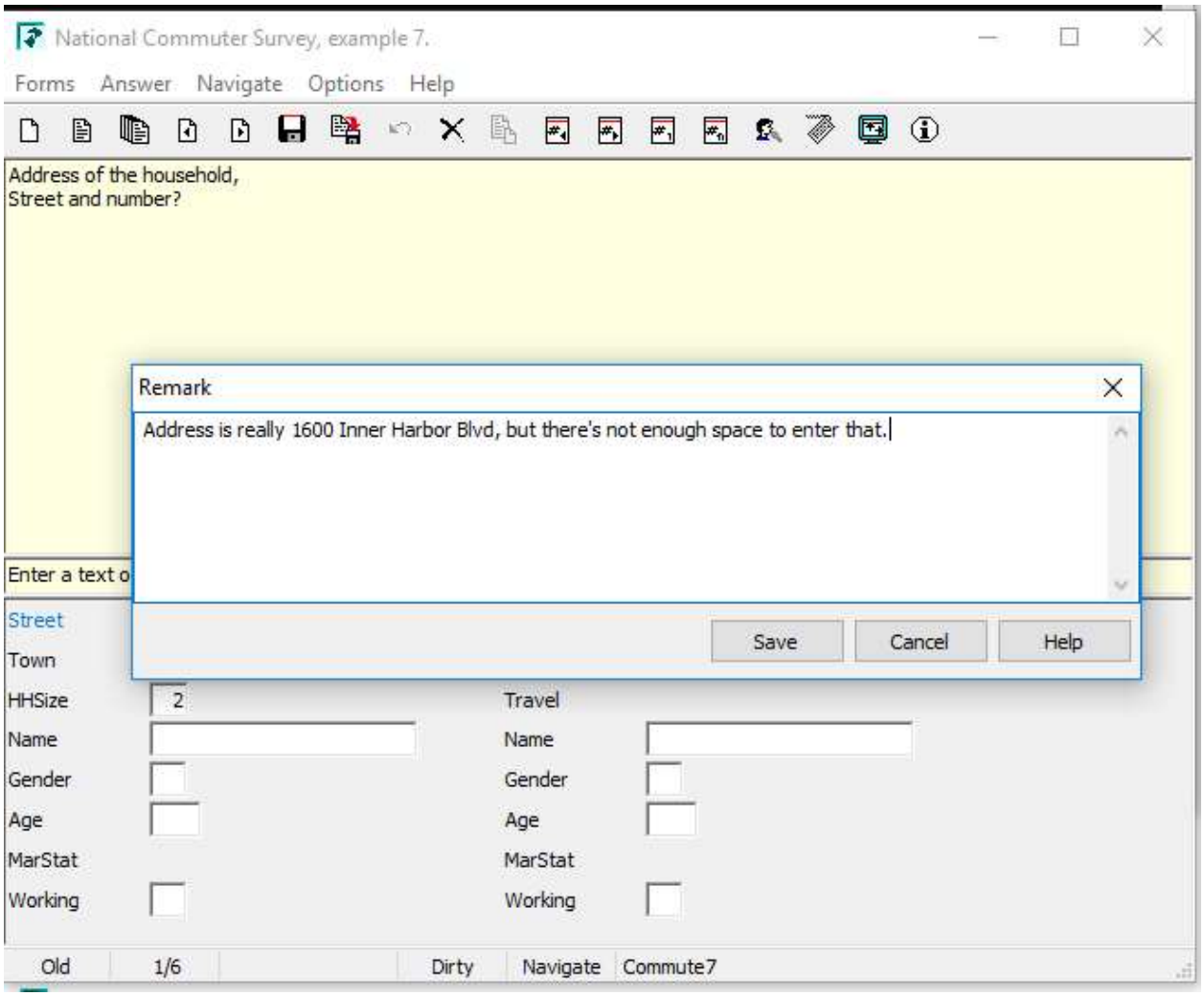

Figure 1. An example of a remark in the Data Entry Program in Blaise 4.8

What is good about remarks?

Remarks can help improve data quality in the phase the data will go through after interviewing, often called the data editing or the data cleaning process. If a remark contains enough specific information the intended data change can be made by the home office staff. Otherwise one or more answers may have to be set to DontKnow, or the remark may have to be ignored.

Blaise has a simple and effective way of showing an interviewer where remarks were made during the interview, with the paperclip icon as shown in Figure 2.

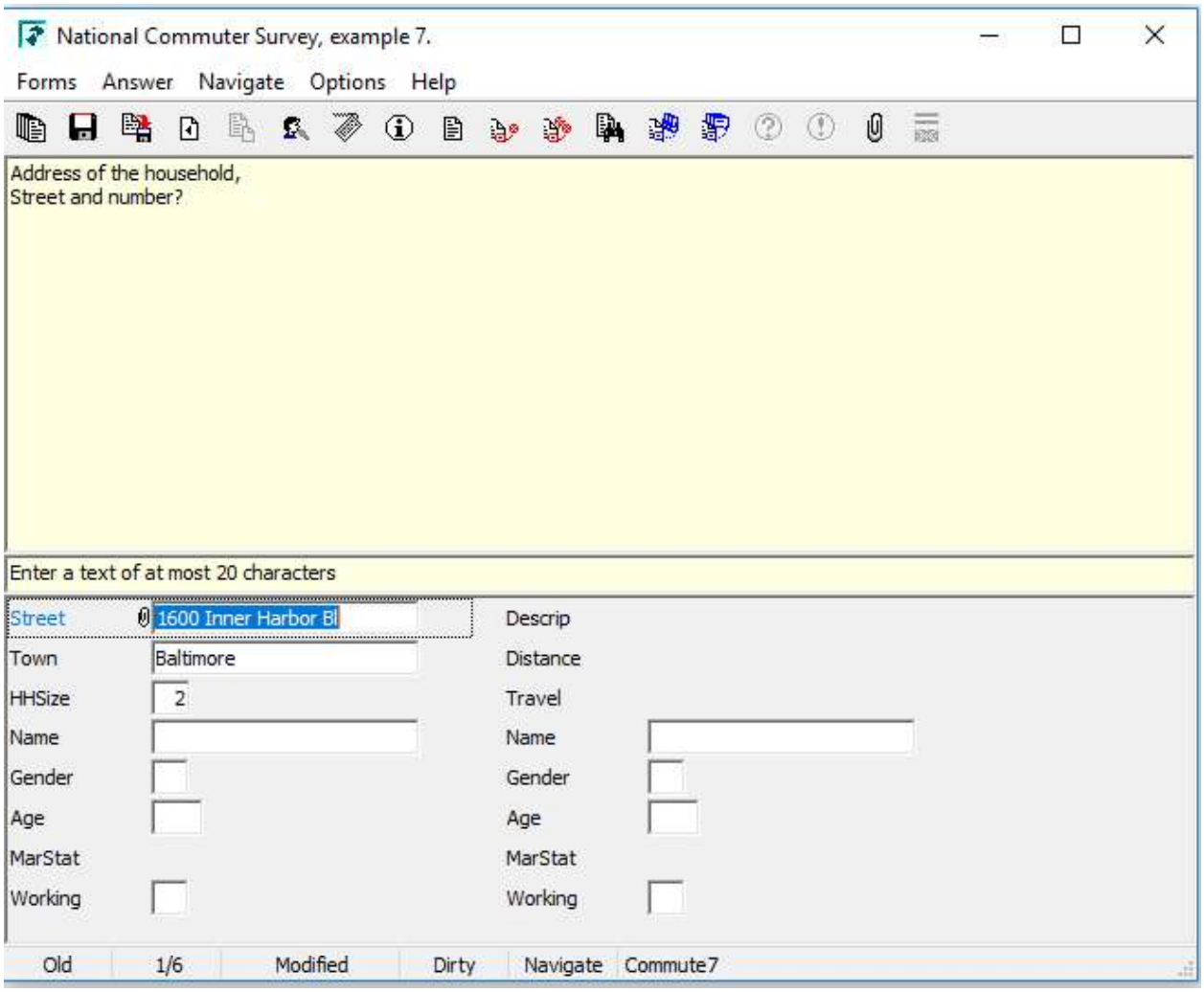

Figure 2. The resulting paperclip at the question where the remark was made

Blaise also allows for easy export of all remarks, per case or for all cases at once, to a separate text file.

Problems with remarks

The problems that arise from interviewers resorting to remarks are twofold:

- The resulting data quality is not likely to be as good as if the change had been made by the interviewer, with the respondent still available
- Going through all the remarks for all interviews in the home office is time-consuming and expensive

Especially if interviewers are resorting to remarks to not break the flow of an interview, the number of remarks to work through for data editing staff can be quite high. There is no easy way to know which

remarks are actionable – contain enough information to make the intended changes to the data – or which ones are more important to data quality compared to others. As remarks are just text strings you really have no option to read all of them before you can prioritize. Depending on constraints in the data delivery schedule, staff availability and overall budget handling all remarks appropriately can become a problem.

One possible course of action is to try to make the interview instrument as user-friendly as possible for the interviewer, in order to reduce the need for making remarks in the first place. We have implemented attempts at doing exactly that, for instance by adding menu options that give access to review screens, and that may even allow quick access to a certain section and automatically return to the question from which the menu item was called.

The focus of this paper is however finding ways to make the handling of remarks at the home office more manageable. In order to make categorization possible before reading all remarks we will show how to replace the standard Blaise remark popup with a Manipula dialog box that can be tailored to the needs of the project at hand.

This starts with the DEP menu option for remarks, which will have to be 'diverted' to a Manipula procedure.

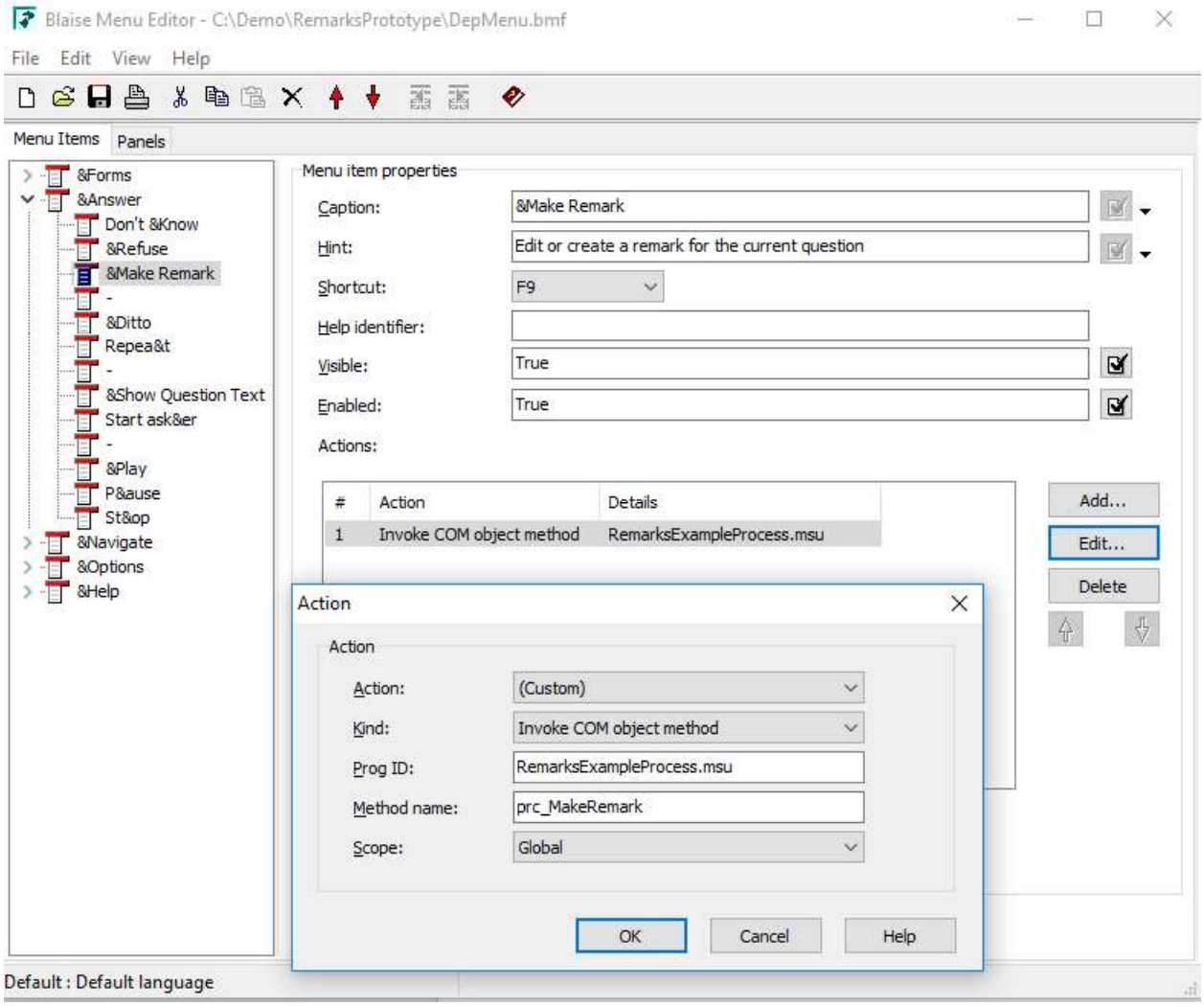

Figure 3. A screenshot of the DEP Menu Editor and the menu item to replace the standard remark popup with a customized version.

We tend to use the custom Action Kind 'Invoke COM object method' with the Scope set to Global, as the screenshot above shows, instead of selecting the probably more straightforward 'Start Manipula' option in the 'Kind' dropdown menu. This is done on purpose: using 'Start Manipula' will initialize/load the Manipula setup every time it is called. Using the method shown above the setup will be loaded once, at the start of the interview, and kept in memory. For a small Manipula process the speed difference may not be very noticeable, but for more extensive ones it really makes a difference in how fast the procedure gets executed. The lag at the very start of the interview may become fairly long if there are many setup to load, but a pause before the interview starts is a lot more manageable than several pauses later on, during the interview.

The next screenshot shows an alternative remark popup for the same remark as shown in Figure 1. The category at the top has to be selected before the remark text box becomes accessible.

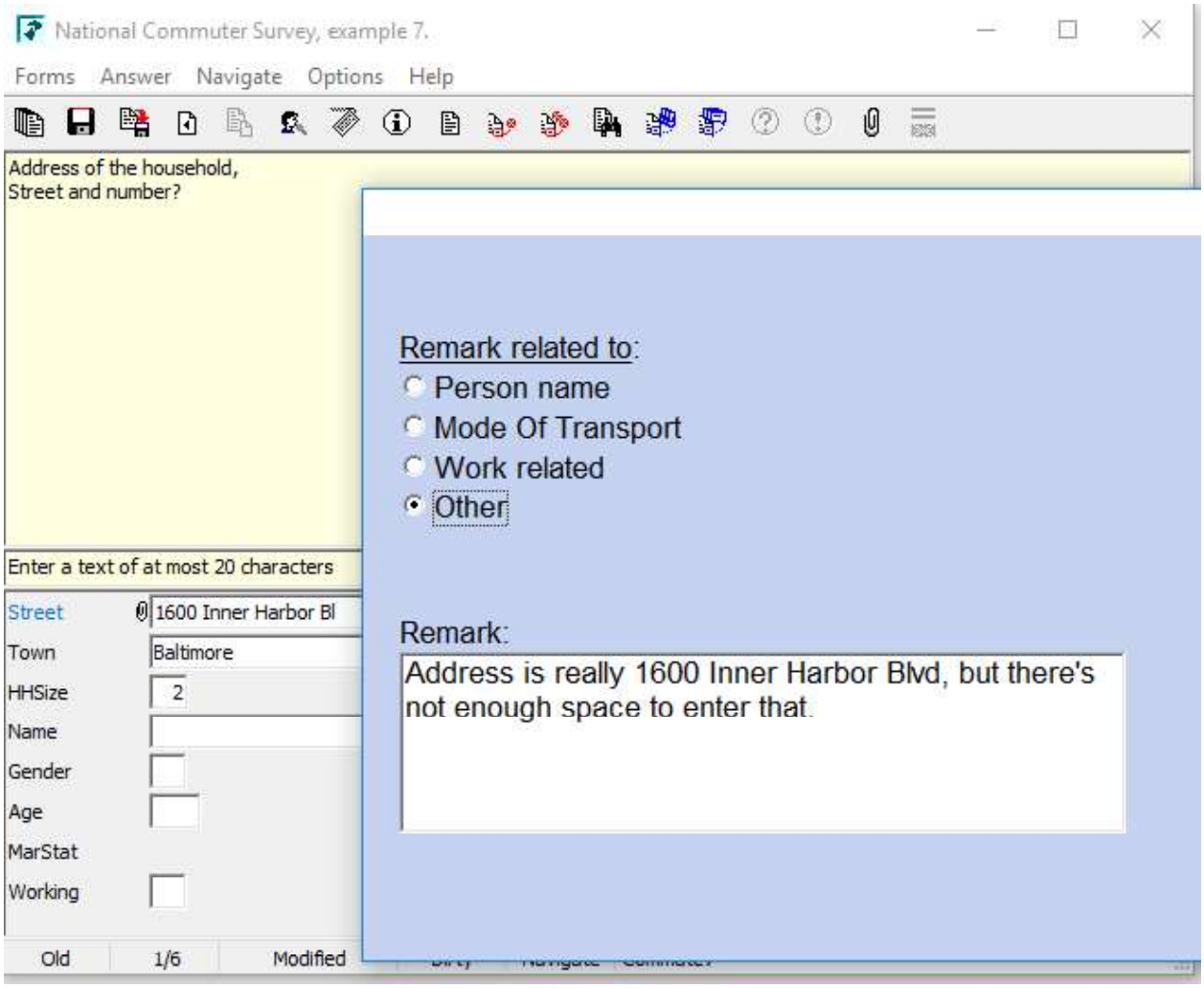

Figure 4. The same remark as in Figure 1, now with a custom remark popup that includes a category

Here is another example of a remark in the same interview:

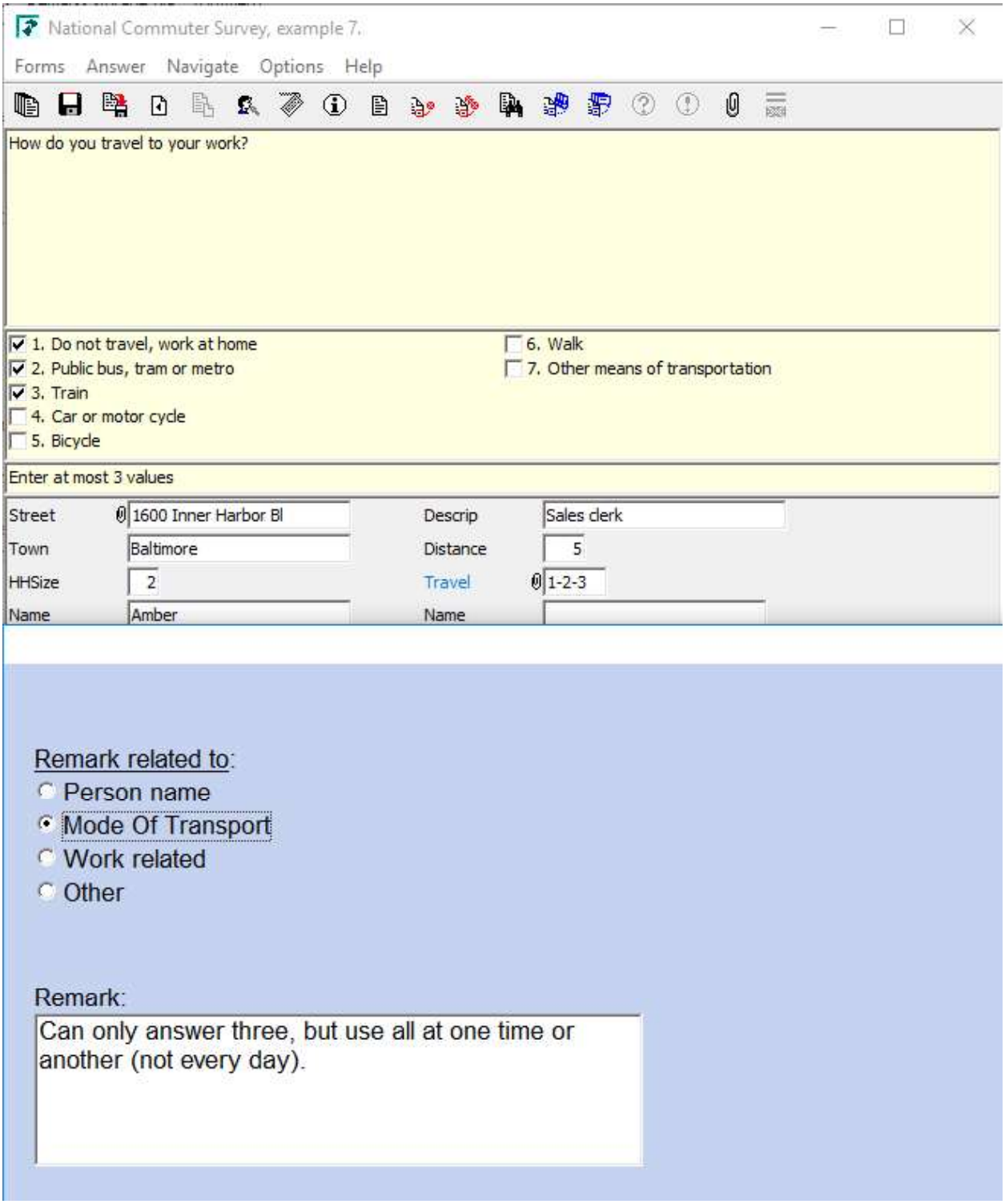

Figure 5. Another example of a categorized remark

And here is a screenshot of the Data Viewer, to show the external storage of the remarks data:

C:\Demo\RemarksPrototype\Commute7_Remarks - Blaise 4.8 Database Browser

CaseI	Rema	FullFieldName	LocalFieldN	Field	RemarkCategory	Remark	DateRemarkCr	TimeRema
6	1	Street	Street		Other	Address is really 1600 Inner	9/25/2018	5:03PM
6	2	Work[1].Travel[1]	Travel		ModeOfTransport	Can only answer three, but us	9/25/2018	5:19PM

Figure 6. Data Viewer screenshot for the external remarks data fil.

Note that the remark field doesn't show the full text here, in order to make the screenshot fit the page without resorting to an unreadable small font size. In the data file the remark is stored in its entirety however.

The interaction between DEP, DEP menu and external data model and data file is all done in the Maniplus process that is referenced in the menu file. It uses some of the extremely powerful capabilities of Manipula in Blaise 4.8:

- having access to the metadata and the data of the current interview, in particular the name of the current question in the interview
- the ability to read from and write to separate data files
- display dialogs that can use external data
- access the internal Blaise remarks (still used for the 'paperclip functionality')

Following in Figure 7. are a few pretty basic lines from the setup code showing the use of a temporary file and access to remarks and metadata information within the Maniplus code:

```
PROCESS RemarksExampleProcess

SETTINGS
  MessageFile = 'AnyRemarkErrors.txt'

USES
  Commute7
  RemarksStorage

UPDATEFILE theRemarks : RemarksStorage
  SETTINGS
    OPEN = NO
    MAKENEWFILE = NO

TEMPORARYFILE DepSession: Commute7
  SETTINGS
    INTERCHANGE = SHARED

{...}

{The procedure called from the DEP menu:}

PROCEDURE prc_MakeRemark
  AuxFullFieldName:= ACTIVEFIELD
  prc_GetRemark
  dlg_MakeRemarkDialog
  CASE OKCancelButton OF
    OK : DepSession.PUTREMARK(ACTIVEFIELD, TRIM(AuxRemark))
         prc_WriteToRemarksFile
  ENDCASE
  prc_ReleaseRemarksFile
  SETALIENROUTERACTION (BLRADEFAULT)
ENDPROCEDURE {prc_MakeRemark}

{And a few other lines showing the use of meta information}
  AuxLocalBlock := DepSession.GETFIELDINFO(ACTIVEFIELD,'BLOCKNAME')
  AuxFileName:= DepSession.GETMETAINFO('DICTIONARYNAME') + '_Remarks.bdb'
  theRemarks.OPEN(AuxFileName)
```

Figure 7. Several code snippets from the Maniplus setup

The complete setup used for this example is a relatively short one, perhaps 150 lines, a good part of which are taken up by the definition of the dialog box. The main point of including some of the lines here is to show that these kinds of changes and additions to standard Blaise DEP functionality are actually very easy to make.

Conclusion

Blaise 4.8 offers ways to use Manipula as a separate but integrated layer of control for the Data Entry Program, opening doors for functionality that was not available before. This functionality can be used to great effect for the interview itself, as has been shown in previous IBUC papers. The small example in this paper shows how it can be used to also positively affect the post-interview stage of data editing.

The hope is that in a current or future version of Blaise 5 changes like these – if necessary at all – can be made as effectively.