Documenting questionnaires

Jean-Pierre Kent and Leon Willenborg, Statistics Netherlands

1. Introduction

The growing possibilities of Blaise III have led designers to build very complex survey instruments. It has become increasingly difficult for users to keep control of the content and structure of questionnaires. Although the Blaise language was designed with self documentation in mind, it takes some knowledge and experience to understand a large questionnaire.

It has become possible to develop and administer data collection instruments so complicated, that researchers, even the original instrument and data producers, have difficulty comprehending them in their entirety. It has become difficult to fully understand the process that leads to responses to each of the items as they ultimately appear on data files.

This rises the question of the feasibility of a tool to represent the content and logic of a questionnaire in a human-readable way. This concern is not limited to the Blaise user community: we read the following in the APDU Newsletter, November, 1993: “how are users going to get access to and develop a thorough understanding of the new survey instrument when it is so complex it requires 6 megabytes or more of memory on a 486 computer to execute?” (quoted by Doyle 1996).

Such concerns are both for the user of the data and for the user of the Blaise system: the former needs to know how the data are structured and how they were collected; the latter needs to keep under control the costs of producing this documentation.

The present paper is primarily intended as a presentation of the state of the problem. It also presents the issues and the costs involved in implementing documentation functionality in Blaise.

2. Historical perspective

When surveys were conducted with PAPI (Paper And Pencil Interviewing) technology, there was no need for separate documentation: the paper form, developed for the interviewer’s use, contained all the information available about the questionnaire. Moreover, routing information was kept simple, in order to allow the interviewer to process branching conditions himself. Routing logic was kept simple, and the questionnaire form *was* the documentation.

The advent of CAI (Computer Assisted Interviewing) brought about two changes in this picture: the paper document was replaced by a computer program, and it became possible to implement more complex branching logic. So users were faced with data collected with a more complex instrument and less documentation to understand it..

In the early times of CAI, when questionnaires were developed for XT computers, in which the operating system, the software and the data had to share 640 KB of memory, instruments were limited to a size that made it possible to document them by hand. We have examples of flow charts created and maintained by hand. The progress of hardware and software technology, however, have opened the way for very large questionnaires: the cost of creating and maintaining the documentation by hand has become prohibitive.

It must also be mentioned that manual documentation can be a source of errors. Whatever is used to express the content and logic of a questionnaire in a way accessible to a broad group of users, whether it is made of flowchart icons, indented paragraphs or natural language references, the information is conveyed in a form essentially different from the Blaise language. Although automatic translation is possible between the two (it should be, otherwise the documentation formalism is poorly designed), the human translator cannot perform the task without extracting the meaning from the Blaise source and re-expressing it in the documentation formalism. This process is subjective and error-prone. It can spawn subtle errors that will usually go by unnoticed, causing the reader to form an erroneous picture. Therefore, automatic documentation is a must.

3. Paper, screen, or both?

In the early times of CAI, developers sometimes asked for the possibility of creating a paper version of the questionnaire. At the beginning, it was not clear why such a document was needed. Interviews were only used in their electronic form. One clearly did not have a multi-mode survey in mind: the intention was not to use CAPI and PAPI side by side.

It eventually became clear that developers needed a document to help them build a picture of the instrument, and to convey that picture to their users. This information contributed to the ideas that led to the creation of the Structure Viewer (hereafter: SV). The SV does more than could be expected from a paper document. It allows the user to picture the data model at various levels of the hierarchy, and to concentrate on the aspects that are relevant for answering any specific question about the structure of the model.

The SV, however, does not respond to the need for a way of documenting the *questionnaire*: the questionnaire is not just the data structure of the model, it is the sum of all features that contribute to making the interview: routing and computations play an important role, and there is nothing in Blaise yet to answer questions relative to these aspects.

In our view, the best response to the expressed documentation needs would be a tool allowing to approach the RULES information in the same way as the SV approaches the FIELDS information. Something that could be called the Route Viewer, or the Rules Viewer (RV). Such a tool should be able to represent the route through the questionnaire as a network of paths, to fold and unfold them on user command, and maybe to create a printed report of the information selected by the user.

In this perspective, screen interaction is dominant: much more relevant information can be obtained by zooming into the area of interest than by visually inspecting a paper document.

It can be argued that the screen document is the only relevant one. Nowadays data are more and more available on line. For on-line access of data, you need to be able to access the metadata on line as well. This type of work implies every user has a computer. In this context, an interactive tool will always be preferred to a passive paper document. We therefore expect that a time will come when paper documentation will no longer be asked for.

In the mean time, we think, the best solution is an interactive tool capable of producing paper output. Although the TELPERION project discussed in this paper (see paragraph 5) was not meant as a documentation tool, it offers an interesting example of what could be achieved.

4. A paper documentation generator

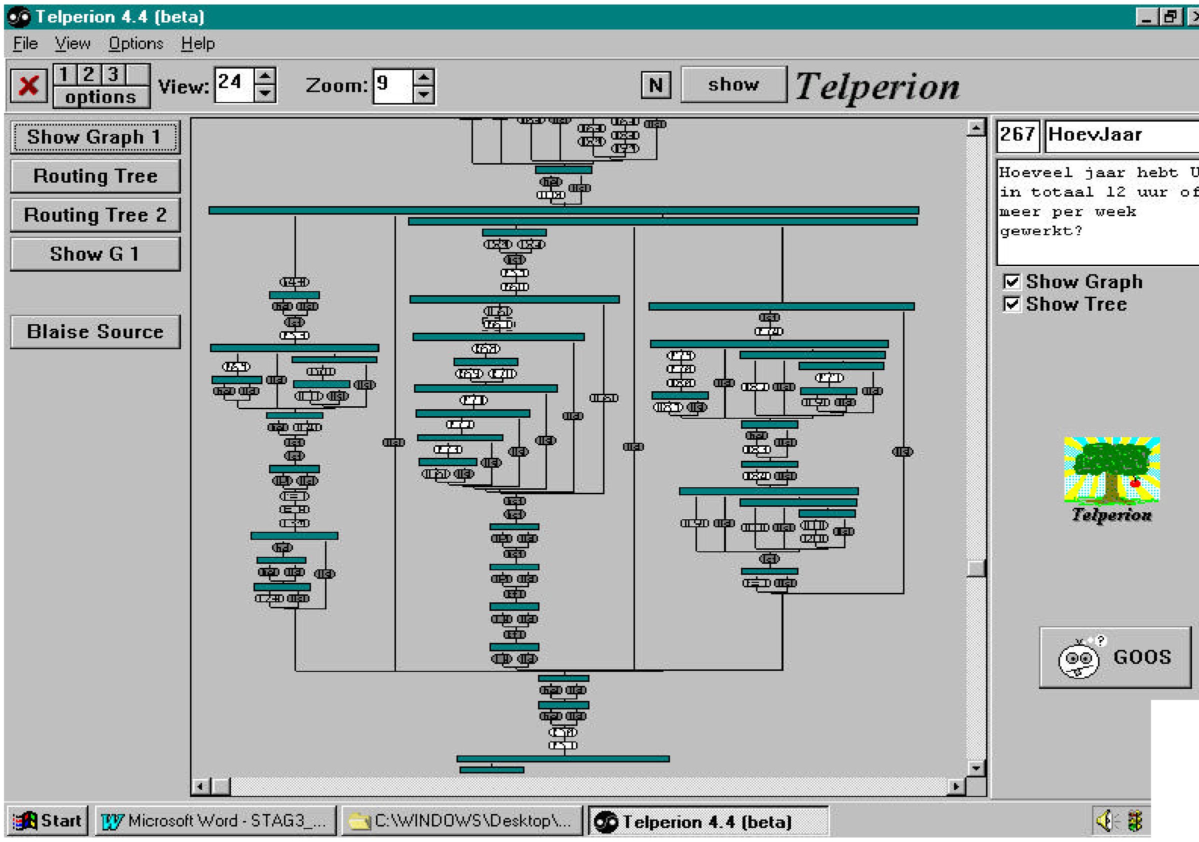
Elsewhere in this volume Steve Anderson presents the project that was developed in the United Kingdom under the acronym BAD (Blaise Automatic Documentation). The reader is referred to Steve's article for a discussion of the problems involved.

BAD is an example of the paper-oriented approach to questionnaire documentation. The following section shows what a screen-oriented tool can look like.

5. An interactive RULES Viewer

TELPERION is a computer program that can be used to plot routing graphs of questionnaires. The currently available version, Telperion I, is only a prototype. Telperion was developed at Statistics Netherlands in 1996 by Dam Backer, then a student of Computer Science of the University of Utrecht, supervised by Leon Willenborg and Goos Kant (University of Utrecht and ORTEC Consultants) and Jan van Leeuwen (University of Utrecht). The initial goal was to develop a program that was able to plot an arbitrary routing graph. Soon after the project was launched, the decision was taken to focus exclusively on Blaise routing graphs. The reasons for this were manifold: time considerations played a role, the fact that Blaise questionnaires were available at Statistics Netherlands and offered concrete examples of routing structures, and also the fact that Blaise questionnaires have a rather simple routing structure, namely series-parallel routing graphs which have the nice property of being planar.

Figure 1



267 HoevJaar

```
Hoeveel jaar hebt U
in totaal 12 uur of
meer per week
gewerkt?
```

TELPERION I is able to plot routing graphs on a computer screen, but not on paper. Figure 1 shows a screen dump of such a plot (taken from Backer, 1996). A feature of the program is that it allows the definition of blocks, that can either be plotted as a single point or as a more detailed structure consisting of points, blocks and transitions. This allows one to view a routing structure, globally or locally, at various levels of detail.

The currently available version of TELPERION should be viewed as a first prototype. We will now discuss a few features that could be considered for incorporation in a future version. The following list is not intended to be complete: suggestions and feedback from users are welcome.

Routing structure on paper. The current version of TELPERION only yields images of routing structures on the monitor screen. For the production of publishable documentation, however, printer output could be required. As a short-term solution one could work with screen dumps, but for a proper document one needs to rethink the representation problem. Some of the functionality for screen prints is not needed (e.g. folding and unfolding blocks), and some features should be implemented in a different way – one typical example being the link between a question on the routing graph and a full representation of the information referring to it (name, type and various texts). The use of colour is another dimension that could be investigated.

Functionality of a routing graph tester. The current use of TELPERION is rather passive: displaying in a visually appealing form what is already in the Blaise questionnaire. One could also try to develop it into a tool for testing questionnaires while they are being developed. In order to be able to plot a “questionnaire under construction” one needs a syntactically correct questionnaire, which need not be complete. A questionnaire could be constructed through a process of stepwise refinements, by progressively replacing empty blocks by more elaborate structures, with the possibility of viewing the routing structures at each stage of construction. With a few additional features, TELPERION would allow a questionnaire designer to view a sub-graph between a given starting point and end point, to select a path from the routing and inspect a list of conditions. This would turn it into a rich complement of the SV.

A RULES editor. Originally, the SV was meant to become a structure editor: a tool for the interactive construction of questionnaires. It is not yet clear whether and when it will be possible to assign this project sufficient priority. An alternate possibility would be to integrate TELPERION into the Blaise Control Centre and give it editing functionality: this would make it possible to build blocks interactively by working on the routing structure, instead of taking the data structure as the starting point. The ideal solution, of course, would be to build editing capabilities into both tools.

Graphical representations of survey results. For a completed survey it could be interesting to see the flow through the questionnaire, by plotting the routing graph with edges of varying degrees of thickness: the thicker (or the redder, if colour is permitted) an edge, the more individuals have responded to the corresponding questions. This would give a quick view of the main streams through the questionnaire and also draw attention to the parts where nothing happens. This information would be useful for testing questionnaires. It may require redrawing the graph, in order to plot the main streams as much as possible in the middle of the page.

6. Documenting from the Blaise Control Centre ?

The process of documenting a Blaise data model requires both an interface to the data model, and the capability of presenting the information in the desired form. It also needs a way to communicate with the designer, in order to determine how each element of metadata will be displayed. In the case of an interactive tool with edit functionality, these two aspects need to be integrated. This requirement need not be met by a read-only tool like those mentioned in this paper: it is possible to extract metadata in the first step, and to represent this information in the second step without further accessing the metadata file. This two-step approach is the best way to implement questionnaire documentation as a short-term project. We will concentrate here on the first step: extracting metadata.

6.1. The interface to the metadata

A prepared Blaise data model is available in two different forms: the Blaise source and the metadata file (the .~MI file). The advantage of using the source is evident: the Blaise language is fully documented, writing a parser for it is the only step needed for access to the data model. Implementing such a parser can be done without help from the Blaise team. However, this method could entail a substantial maintenance problem, in the eventuality of future changes in the specification of the Blaise language. Although such changes are excluded for the short term, this consideration has led developers to discard this option.

Another possibility is the use of the Blaise metadata engine library to access the .~MI file. Changes in the format of the metadata file would have a negligible maintenance impact, because the only step needed for upgrading an application using the library is a recompile with the new version.

On the other hand, this is an awkward approach, because the library was exclusively designed for internal use. It does not have a simple API, but a complex family of objects. Use of this library requires some coaching from Statistics Netherlands. Although the two projects mentioned here show that this approach can be successful, we do not wish to recommend it. In our view, the documentation developers should be able to do their work without being dependent on the Blaise team.

The normal way of extracting information from the metadata file is to use CAMELEON. For historical reasons, however, the functionality of CAMELEON has been limited so far to the extraction of information from the FIELDS and TYPE sections of the data model. In spite of the attention that has been paid to the problem of extracting information from the RULES section, this work was never assigned the priority it would need in order to reach completion. We would like to suggest here that if we are asked to offer documentation functionality within the short term, our best answer would be to extend the functionality of CAMELEON to the RULES section.

6.2 Authoring with documentation in mind

A questionnaire developer works primarily to produce a data model, and a data entry application to collect the data. If these are the author's sole concerns, an automatic documenting tool is doomed to fail. Whether meta-data are extracted with CAMELEON or any other tool, hints will be needed to guide the program in its choices and to make conditions readable to an user not involved in development or collection. If, for instance, a series of questions falls under the following condition:

```
IF Age >= 65 AND Pension = No THEN
```

the documentation should mention something like "elderly people without a pension". This wording could be different from what is needed in error messages. So, clearly, the documentation needs a text that can be extracted neither from the conditional expression, nor from the error text.

The documentation tool also has to know what parts of the RULES are relevant for output and which are not. What about computations? Most of them should not make it into the documentation, but some are relevant to the meaning of the data. For instance, complex routing conditions can be pre-computed in local variables: these calculations should appear in the documentation.

The solution to both problems like in the use of the multi-language functionality of Blaise. Originally the languages feature was built into the system in order to support multilingual questionnaires. It can, however, be used for the specification of any text. We know projects in which “languages” are defined and used to specify alternate variable names and labels for analysis and tabulation packages.

CAMELEON can be tuned to use any of the defined languages, and to switch in the middle of a setup. In order to use this feature for documentation, it is sufficient to define a documentation “language” in the data model, and to switch to this language in the setup.

We cannot give here an example of a CAMELEON setup, because the syntax for accessing RULES information is not defined yet. The example in Figure 2 shows how a data model can be defined with documentation texts.

7. Literature

Anderson, S., 1997: “Automated Paper Documentation of Blaise III Datamodels” (elsewhere in this volume).

Backer, D., 1996: *Drawing Questionnaire Routing Graphs*, Report, Department of Statistical Methods, Statistics Netherlands, Voorburg.

Doyle, P., 1997: “Documenting cai instruments: a blueprint for the future.” (Paper to be presented at IASSIST, MAY 1997).

Willenborg, L.C.R.J., 1988*: Computational Aspects of Survey Data Processing*, CWI Tract 54, Centre of Mathematics and Computer Science, Amsterdam.

Figure 2

```
👎✌❄✌💣⚐👎☜☹ 👎□♍◆❍♏■⧫♋♌●♏

☹✌☠☝🕆✌☝☜💧 🖬
    ✋❖⬥ ✂✋■⧫♏❒❖♓♏⬥✂📪
    ☜♎⧫ ✂👎♋⧫♋ ♏♎♓⧫♓■♑✂📪
    👎□♍ ✂👎□♍◆❍♏■⧫♋⧫♓□■✂

☞✋☜☹👎💧
    📬📬📬

☼🕆☹☜💧
    ✋☞ ✌♑♏ ✇🖬 ⌛🗄      ✂✡□◆ ♋❒♏ ♈✌♑♏ ⌧♏♋❒⬧ □●♎✂
                        ✂☼♏⬧□□■♎♏■⧫ ♓⬧ □❖♏❒ ⌛🗄📪✂
                        ✂☜●♎♏❒●⍓ □♏□□●♏✂
    ❄☟☜☠
          ✋☞ 🏱♏■⬧♓□■ 🖬 ☠□

          ✂♌◆⧫ ⍓□◆ ♒♋❖♏ ■□ □●♎ ♋♑♏ □♏■⬧♓□■📬✂
                        ✂♌◆⧫ ♒♋⬧ ■□ □●♎ ♋♑♏ □♏■⬧♓□■✂
```

✂⬧♓⧫♒□◆⧫ □●♎ ♋♑♏ ◻♏■⬧♓□■✂

❄🏳☜☠

💧✋☝☠✌☹

🏳⧫♒♏❒✋■♍□❍♏ 🖬 ✡♏⬧

✂✌❒♏ ⍓□◆ ⬧◆❒♏ ⍓□◆ ♒♋❖♏ ■□ □⧫♒♏❒
⬧□◆❒♍♏ □♐ ♓■♍□❍♏✍✂

✂□❒ □⧫♒♏❒ ⬧□◆❒♍♏ □♐ ♓■♍□❍♏📪 ✋⬧ ⧫♒♓⬧
♍□❒❒♏♍⧫✍✂

✂♋❒♏ ♏⌧◻♏♍⧫♏♎ ⧫□ ♒♋❖♏ □⧫♒♏❒ ♓■♍□❍♏
⬧□◆❒♍♏⬧📪✂

☜☠👎✋☞

☜☠👎✋☞

☜☠👎💣🏳👎☜☹