Journal of Official Statistics, Vol. 20, No. 2, 2004, pp. 233–264

TADEQ: A Tool for the Documentation and Analysis of Electronic Questionnaires

Jelke Bethlehem and Anco Hundepool[1]

National Statistical Institutes (NSI's), research institutes, and commercial marketing research organisations are more and more using computer-assisted interviewing (CAI) systems for collecting survey data. A computer program that guides respondents through the questionnaire and checks the answers on the spot replaces the traditional paper questionnaire. The growing possibilities of computer hardware and software have made it possible to develop very large, and complex electronic questionnaires. Unfortunately, it also has become more and more difficult for developers, interviewers, supervisors, and managers to keep control of the content and structure of CAI instruments. Within a 4th Framework project of the EU, research has been carried out aimed at developing a tool to make a readable and understandable presentation of an electronic questionnaire. The output of this tool serves to document (on paper, or electronically in hypertext form) an electronic questionnaire in a user-friendly way. It not only provides a useful documentation of the interviewing instrument, but also helps to analyse the questionnaire, and report possible sources of problems in its structure.

Key words: Computer-assisted interviewing; questionnaire; documentation.

1. Computer Assisted Interviewing

Carrying out a survey is a complex, costly and time-consuming process. One of the problems is that data collected by means of paper forms usually contain many errors. Extensive data editing is required to obtain data of acceptable quality. This consumes a substantial part of the total survey budget. Rapid developments of information technology in the last decades made it possible to use microcomputers for computer-assisted interviewing (CAI). A computer program containing the questions to be asked replaces the paper questionnaire. The computer takes control of the interviewing process, and it also checks answers to questions on the spot. Application of computer-assisted data collection has three major advantages:

- It simplifies certain aspects of the work of interviewers. Tasks like decision making involved in determining skip patterns, tailoring questions, and determining if an answer is acceptable, are taken over by the computer. However, it is fair to say that the loss of flexibility associated with paper questionnaires may also complicate the job of the interviewer.
- It is possible to incorporate checks on the answers in the interview software. Through checking and correcting answers during the interview, the quality of the collected

[1] Statistics Netherlands, Statistics and Informatics Department, P.O. Box 4000, 2270 JM Voorburg, The Netherlands. Email: jbtm@cbs.nl and ahnl@cbs.nl

data can be improved. Wherever possible, data editing during the interview is more effective than having to do it afterwards in the office.

- Data are entered in the computer during the interview, resulting in a clean record, so no more subsequent data entry and data editing is necessary. This considerably reduces the time needed to process the survey data, and thus improves the timeliness of the survey results.

More on the benefits of CAI can be found in Couper et al. (1998). Computer assisted interviewing comes in three modes. The first mode implemented was Computer Assisted Telephone Interviewing (CATI). In the eighties the laptop computers arrived, and it became possible to implement Computer Assisted Personal Interviewing (CAPI), which is the electronic form of face-to-face interviewing. Another recent mode of CAI is Computer Assisted Self Interviewing (CASI). CASI is the electronic analogue of mail interviewing. Diskettes are sent to respondents, or they can access the interviewing software via telephone and modem, or via the Internet (also called CAWI, for Computer Assisted Web Interviewing).

The growing possibilities of computer hardware and software have made it possible to develop very large, and very complex, electronic questionnaires. It is not uncommon for electronic questionnaires to have thousands of questions. To protect respondents from having to answer all these questions, routing structures and filter questions see to it that only relevant questions are asked, and irrelevant questions are skipped. Owing to the increasing size and complexity of electronic questionnaires, it has become more and more difficult for developers, users and managers to keep control of the content and structure of questionnaires. It takes a substantial amount of knowledge and experience to understand large and complex questionnaires. It has become more and more difficult to comprehend electronic questionnaires in their entirety, and to understand the process that leads to responses to each of the questions as they ultimately appear in data files. (See e.g., Kent and Willenborg 1997.)

A number of concrete problems have arisen in statistical agencies due to the lack of insight into complex electronic questionnaires:

- Testing of electronic questionnaires, at least for complex surveys, has always been difficult, but with the growing size and complexity of questionnaires it has become harder still. It is no simple problem to test whether every possible person one might encounter in the field will answer only the relevant questions in the correct order. Every possible tool providing insight into this matter will help to avoid problems in the field. A workshop on the problems of questionnaire testing was organized by the U.S. Committee of National Statistics (CNSTAT) in Washington D.C. in 2002 (see Cork et al. 2003).
- Creating textual documentation of an electronic questionnaire is now an enormous task. It is usually a manual task. Therefore, it is error-prone. There is no guarantee that hand-made documentation exactly describes the real instrument. Also, making documentation by hand is a time-consuming task.
- Survey managers have to approve a questionnaire before it goes into the field. In the old days of paper questionnaires, they could base their judgement on the self-documenting paper questionnaire. When it comes to electronic questionnaires, they

have nothing the put their signature on. The printout of the questionnaire specification in the authoring language of the CAI system is not usually easily readable for the nonexpert. So documentation is required that on the one hand is readable, and on the other hand describes as exactly as possible what is going on in the instrument.

- Interviewers carrying out a survey with paper questionnaires can use the questionnaire to get some feeling of where they are in it, of what the next questions are about, and of how close they are to the end. If they use an electronic questionnaire, they lack such an overview. Therefore they often ask for a paper document describing the global content and structure of the questionnaire. They can use this as a tool together with the electronic questionnaire.

In the early times of computer assisted interviewing, when electronic questionnaires were developed for computers of limited memory size and speed, it was still possible to produce instrument documentation by hand. There are ample examples of hand-made flow charts (see e.g., Jabine 1985). And for some CAI systems, the instrument specification (in the authoring language of the system) was more or less self-documenting (see early examples of Blaise). However, recent developments in information technology have made it possible to create such large and complex questionnaires that the cost of creating and maintaining documentation by hand has become prohibitive. An additional problem of hand-made documentation is that it can be a source of error. Also the instrument specification cannot serve any more to document the instrument.

All these problems raise the question of the feasibility of a flexible tool capable of representing the content and logic of an electronic questionnaire in a human-readable way. Such a tool should not only provide a useful documentation, but also help to analyse the questionnaire, and to report possible sources of problems.

2. The TADEQ Project

Consistent and error-free documentation can only be obtained if it is generated automatically. Error-free means here that there are no discrepancies between the instrument and its documentation. Of course, if the instrument contains errors, they will also be part of the documentation. All information about the contents and structure is in essence available in the electronic questionnaire. What is needed is a software tool capable of automatically translating this questionnaire specification into a human-readable format.

There have been a number of initiatives for automatically producing survey documentation. In 1995, the Inter-university Consortium for Political and Social Research (ICPSR) established a committee to develop a definition for an international codebook standard. This is called the Data Documentation Initiative (DDI) (see ICPSR). Originally, the Standard Generalized Markup Language (SGML) was used to define this documentation standard. In 1997, the definition was made compliant with the Extensible Markup Language (XML). Documentation in XML can be made available on the Internet. The DDI concentrates on documenting general survey issues (such as survey population, sampling design, reference period, mode of data collection) and variables in data files containing the collected data. The documentation is typically used for analysis purposes and in data archives.

The Computer Assisted Survey Methods Program of the University of California in Berkeley is developing a set of tools for generating survey documentation. This is the IDOC Project. IDOC stands for Instrument Documentation. (See CSM for more information.) These tools produce documentation that can either be printed or browsed on the Internet. The generated documentation contains general survey information, and detailed information on every survey variable. The tools are designed primarily for use in combination with the CASES system, but also data files produced by other systems can be documented.

The two initiatives mentioned pay little or no attention to documentation of survey data collection instruments. They focus on post-survey data documentation, and not on providing tools to assist in the development and analysis of the operation of the collection instrument. The TADEQ project was set up to develop a tool documenting these instruments. TADEQ stands for Tool for the Analysis and Documentation of Electronic Questionnaires. The project was a European one. It was partly financed by the Esprit Programme of the European Union. Institutes from five countries co-operated in this project: Statistics Netherlands, the Institute of Computer Graphics of the University of Vienna, the Office for National Statistics (UK), Statistics Finland, and the Instituto Nacional de Estatística (Portugal).

One of the objectives of the TADEQ project was to carry out a survey among users of CAI systems to find out what their needs are with respect to survey instrument documentation. A user requirements survey was developed and carried out. Approximately 100 users of CAI systems responded. The results of the survey are described in Kelly and Kuusela (2000). The main conclusions were:

- There is a clear need for automatically generated survey instrument documentation.
- Users need documentation in electronic form, but paper documentation also remains important.
- Since there are various types of users of documentation (e.g., survey designers, supervisors, interviewers, analysts), all with their own specific requirements, a documentation tool should be capable of producing different types of documentation.
- Users want both textual documentation describing questions in detail, and graphical documentation giving information on the global flow of control in the questionnaire.

From the results of the user requirements survey it became clear that a documentation tool for electronic questionnaires should be able to produce human-readable documentation both in paper and electronic form. On the one hand, this tool should be able to show the global structure of the questionnaire, and on the other, it should provide means to focus on the details of parts of the questionnaire.

One of the challenges of the TADEQ project was to display the routing graph of large and complex questionnaires. Owing to the limited size of a sheet of paper and a computer screen, this is not a simple task. It must be accomplished without affecting the readability. It means that a lot of attention has to be paid to layout issues.

Different people involved in the survey process will use the documentation of the questionnaire. Examples are questionnaire developers who want to document their work, survey managers who have to give a formal approval for carrying out the survey, and interviewers who like to have paper documentation of the questionnaire. Different users means different formats of questionnaire documentation. Therefore any documentation tool should be flexible. The users of this tool should have some control over the structure, contents, and format of the documentation.

The TADEQ project aimed at developing a documentation tool that is at least capable of generating two types of documentation:

- *Textual documentation.* It focuses on giving detailed information on all questionnaire objects (questions, checks, computations, etc). Routing information will be taken care of by attaching a condition to specific parts of the questionnaire. These questionnaire parts will only be executed in situations in which the condition is satisfied. A good example of this approach is the BAD system developed by the Office for National Statistics in the United Kingdom (see Andersen 1997).
- *Graphical documentation.* It focuses on giving detailed information on the routing structure. Owing to space constraints only a limited amount of textual information can be displayed in the graph. Depending on the CAI system used, there is a choice of characteristics: question identification names/numbers, question text (possibly in different languages), specification of the type of accepted answers, etc. The available amount of space in the graph is too limited to display all this information. Moreover the lack of space might affect readability. Therefore a documentation tool must provide the means to select the information shown, and possibly also means to display information in different ways, e.g., in a separate window on the screen. The same problems apply to displaying information about routing decisions. It is important to display the conditions determining transitions from one part of the questionnaire to another. Such conditions may be quite complex. Solutions have to be found to show this information without affecting readability and interpretability. Backer (1996) made a first attempt to generate a flow chart from a questionnaire specification. His prototype showed that the idea might work in practical situations.

A questionnaire documentation tool must be able to generate output in at least two formats:

- *Paper documentation.* This is static documentation. Once printed it is not possible any more to manipulate the information. The challenge of the paper format is that two almost conflicting goals must be achieved. On the one hand, it must be able to show the global structure of the questionnaire in a simple way without the user's being distracted by a wealth of detailed information. On the other hand, the user must be able to find detailed information about a specific question or route instruction.
- *Electronic documentation.* Although the small size of a computer screen causes even more limitations than a piece of paper, electronic documentation has the advantage that it can be dynamic. It provides possibilities like zooming in to specific parts of the questionnaire, clicking on vertices or edges to obtain more information, and using hypertext techniques to navigate through the documentation.

3. Electronic Questionnaires

A questionnaire is composed of a number of elements of various types. The elements of paper questionnaires are rather straightforward: there are questions for respondents, and instructions for interviewers to jump to other questions or to the end of the questionnaire. Sometimes paper questionnaires can also have simple checks contained in *check items*. Electronic questionnaires can be much more complex. They may contain many checks, and each check could deal with answers to many questions. Checks and routing instructions may involve substantial computations. And electronic questionnaires also have the possibility of using data from other sources, like the answers to questions in a previous wave of a panel survey. To be able to develop a tool that can understand all aspects of a questionnaire with all these elements, a formal model was required. *Object orientation* played an important role in this.

Objects are the physical and conceptual things we can find in the universe around us. Objects are thought of as having states. The *state* of an object is the condition of an object, or a set of circumstances describing the object. Often people think of objects as being strictly *static*. That is, the state of the object will not change unless something outside the object requests that it changes its state. Related to the object is its *object type*. It is a template for a set of objects that are structurally the same. All objects generated from the same template are said to be *instances* of the same object type.

In an object-oriented approach, a questionnaire consists of objects and relationships between objects. There are a number of different object types, and each object type has its own state, which is described by means of attributes (properties). Examples of object types are:

- *Question object.* A question object describes a question. Each question has as its attributes an identification (question number or name), a question text (possibly several question texts in different languages), a specification of the type of answer that is expected (text, number, selection from a list, etc), and a field in which the answer is stored. For each type of question, there is a different object type. It has attributes that are specific to that type of question, and it inherits basic attributes from the question object type. So, there is an object hierarchy, in which e.g. an open question object is a specialization of the question object.
- *Check object.* A check object describes a check. A check describes a condition imposed on the answers of a set of questions that must be fulfilled. If the condition is violated, the software processing the questionnaire will usually generate some kind of error message. A check has as its attributes an identification, a logical expression describing a condition that must be fulfilled, and an error message (which is displayed when the condition is not met).
- *Computation object.* A computation object describes a computation that is carried out using the answers to previously asked questions and possibly other information. Each computation may have identification, an arithmetic expression, and a field in which the result must be stored.

This list is by no means exhaustive, nor are all the attributes mentioned. Furthermore, each questionnaire can contain instructions with respect to the flow of control (routing instructions). These instructions describe the order in which the objects are processed, and also under which conditions they are processed. The routing instructions can take several forms. This is illustrated using a simple example of a fragment of a questionnaire. Figure 3.1 shows what this fragment might look like in paper form.

```
1. Are you male or female?
   Male . . . . . . . . . . . . . . . . . . . 1   Go to question 3
   Female . . . . . . . . . . . . . . . . . . 2

2. How many children have you had?          - - children

3. What is your age?                        _ _ years

Interviewer: If younger than 17 then go to question 7

4. What is your marital status?
   Never been married . . . . . . . . . . . . 1   Go to question 6
   Married  . . . . . . . . . . . . . . . . . 2
   Separated  . . . . . . . . . . . . . . . . 3
   Divorced . . . . . . . . . . . . . . . . . 4   Go to question 6
   Widowed  . . . . . . . . . . . . . . . . . 5   Go to question 6

5. What is your spouse's age?               _ _ years

6. Are you working for pay or profit?
   Yes  . . . . . . . . . . . . . . . . . . . 1
   No . . . . . . . . . . . . . . . . . . . . 2

7. Do you regularly listen to the radio?
   Yes  . . . . . . . . . . . . . . . . . . . 1
   No . . . . . . . . . . . . . . . . . . . . 2

END OF QUESTIONNAIRE
```

Fig. 3.1. A simple paper questionnaire

The questionnaire contains two types of routing instructions. In the first place, there are skip instructions attached to answer codes of closed questions. This is the case for Questions 1 and 4. The condition deciding the next question asked only depends on the answer to the current question. In the second place, there are instructions for the interviewer that are included in the questionnaire between questions. These instructions are typically used when the condition deciding the next question depends on the answer to several questions, or on the answer to a question that is not the current question. Figure 3.1 contains an example of such an instruction between Questions 3 and 4. Usually, specification languages of CAI systems (so-called authoring languages) do not contain interviewer instructions. Skip instructions appear in different formats. Figure 3.2 contains a specification of the sample questionnaire of Figure 3.1 in the authoring language of the CASES system.

Routing instructions are goto-oriented in CASES. There are two types (see Walton 1999):

- Skips attached to answer codes are called *unconditional goto's*
- Interviewer instructions are translated into *conditional goto's*.

An example of a CAI system with a different authoring language is the Blaise System developed by Statistics Netherlands (see e.g., Bethlehem 1997). The authoring language of this system uses IF-THEN-ELSE structures to specify routing instructions. Figure 3.3 contains the Blaise code for the sample questionnaire. Note that in the simple example above only question objects have been used. There are no check objects or computation objects.

```
>Sex<
    Are you male or female?
    <1> Male                   [goto Age]
    <2> Female
    @

>Children<
    How many children have you had?
    <0-10>
    @

>Age<
    What is your age?
    <0-120>
    @
[@][if Age lt <17> goto Radio]

>MarStat<
    What is your marital status?
    <1> Never been married    [goto Work]
    <2> Married
    <3> Separated
    <4> Divorced              [goto Work]
    <5> Widowed               [goto Work]
    @

>Spouse<
    What is your spouse's age?
    <0-120>
    @

>Work<
    Are you working for pay or profit?
    <1> Yes
    <2> No
    @

>Radio<
    Do you regularly listen to the radio?
    <1> Yes
    <2> No
    @
```

Fig. 3.2. The sample questionnaire in CASES

The way in which the routing structure is specified is not the only difference between Figures 3.2 and 3.3. The developers of Blaise have considered a clear view on the routing structure so important that routing is specified in a separate section of the specification: the questions are defined in a FIELDS section, and the routing is described in the RULES section.

Several CAI systems offer a modular way of specifying electronic questionnaires. This means the questionnaire is split into a number of subquestionnaires, each with its own question definitions and routing structure. Subquestionnaires can be developed and tested separately. It is possible to incorporate such subquestionnaires as a standard component in several surveys, thereby reducing development time and increasing consistency between surveys.

Also with respect to subquestionnaires there can be routing instructions. Answers to questions in one subquestionnaire may determine whether or not another subquestionnaire is executed. Furthermore, subquestionnaires can be used to implement hierarchical

```
DATAMODEL Example

FIELDS

  Sex      "Are you male or female?": (Male, Female)
  Children "How many children have you had?": 0..10
  Age      "What is your age?: 0..120
  MarStat  "What is your marital status?":
           (NeverMar "Never been married",
            Married  "Married",
            Separate "Separated",
            Divorced "Divorced",
            Widowed  "Widowed")
  Spouse   "What is your spouse's age?": 0..120
  Work     "Are you working for pay or profit?": (Yes, No)
  Radio    "Do you regularly listen to the radio?": (Yes, No)

RULES
  Sex
  IF Sex = Female THEN
     Children
  ENDIF
  Age
  IF Age > 16 THEN
     MarStat
     IF (MarStat = Married) OR (MarStat = Separate) THEN
        Spouse
     ENDIF
     Work
  ENDIF
  Radio

ENDMODEL
```

Fig. 3.3. The sample questionnaire in Blaise

questionnaires. Such questionnaires allow a subquestionnaire to be executed a number of times. A good example of a hierarchical questionnaire is a household questionnaire. There are questions at the household level, and then there is a set of questions (subquestionnaire) that must be repeated for each eligible member of the household.

On the one hand, a subquestionnaire can be seen as one of the objects in a questionnaire. It is part of the routing structure of the questionnaire, and it can be executed just like a question or a check. On the other hand, a subquestionnaire contains a questionnaire of its own. By zooming into a subquestionnaire, its internal part becomes visible, and that is a questionnaire with its various objects (questions, checks) and routing conditions. All this means there are several different ways to look at a questionnaire, and a proper documentation tool should offer these different views.

4. The Routing Structure of a Questionnaire

Whether a questionnaire is designed on paper or with a CAI system, it is always possible to model its routing structure as a graph. (See also Willenborg 1988.)

A *graph* is a pair $G = (V, E)$, where $V = \{v_1, v_2, \ldots, v_n)$ is a finite set of *vertices* (the points of the graph), and $E = (e_1, e_2, \ldots, e_p)$ is a final set of *edges* with each edge e_k being an ordered pair (v_i, v_j) representing a possible transition between vertices v_i and v_j. In the case of a questionnaire graph, the vertices are questionnaire objects, and these objects represent possible actions. Examples of questionnaire objects are questions, route instructions, checks, computations, and subquestionnaires.

The edges of the graph represent possible jumps from one object to the next. The routing structure of the questionnaire imposes restrictions on the structure of the questionnaire graph:

- The graph is *a-cylic*. This prevents loops in the questionnaire. There is one exception. For hierarchical questionnaires it may be allowed to repeat a subquestionnaire or a set of questions a number of times (e.g., for each person in the household). However, a person never answers a question more than once.
- The graph is a *directed* graph. It is only possible to jump in one direction from one object to the next, and not vice versa. Note that most CAI systems allow interviewers to go back and inspect previous questions, but this is not what is meant here. Here we refer to routing instructions in the questionnaire that should prevent interviewers from asking previous questions again.
- There is one vertex that marks the start of the questionnaire graph. This *start vertex* has no incoming edges.
- There are one or more vertices with no outgoing edges. These *end vertices* represent questionnaire objects after which the interview is terminated.
- The graph is a *connected* graph, i.e., every vertex can be reached from the start vertex questionnaire graph, and from each vertex it is possible to reach an end vertex.
- Each object, with the exception of the start vertex, has one or more incoming edges. Each object, with the exception of the end vertices, has at least one outgoing edge. Only route instructions and questions can have more than one outgoing edge. All other objects have just one outgoing edge.

Each *path* through the graph from the start vertex to an end vertex represents a possible route through the questionnaire. Note that it may be possible in ill-constructed questionnaires that certain routes are never taken because of contradicting conditions.

Figure 4.1 contains an example of a routing graph. It represents a small part of the (paper) questionnaire of a labour force survey. It contains 28 questions for people without a job. Note that the number of questions a respondent must answer can vary substantially. The shortest route only consists of three Questions (1, 4, and 5). In the case of the largest route, the respondent has to answer 24 questions. The start vertex is indicated by *B*. Note that here there is only one end vertex, marked *E*.

This routing graph only gives a global idea of the routing structure. Many important details are missing. Particularly, one would like to see more information about the objects themselves, and about the conditions leading to a choice of a subsequent object to be executed. If this kind of information is added to the routing graph, it becomes a flow chart.

A flow chart can be defined as a schematic representation of a formal process, programme, or system. It describes the sequence in which various operations may be carried out, and also shows the logical relationships between these operations.

The flow chart gives more information than the routing graph of Figure 4.1. The boxes represent question objects. The boxes also contain some textual information about the questions. The diamonds represent routing objects. They contain information about the conditions governing jumps to other questions. Here the routing objects are simple.

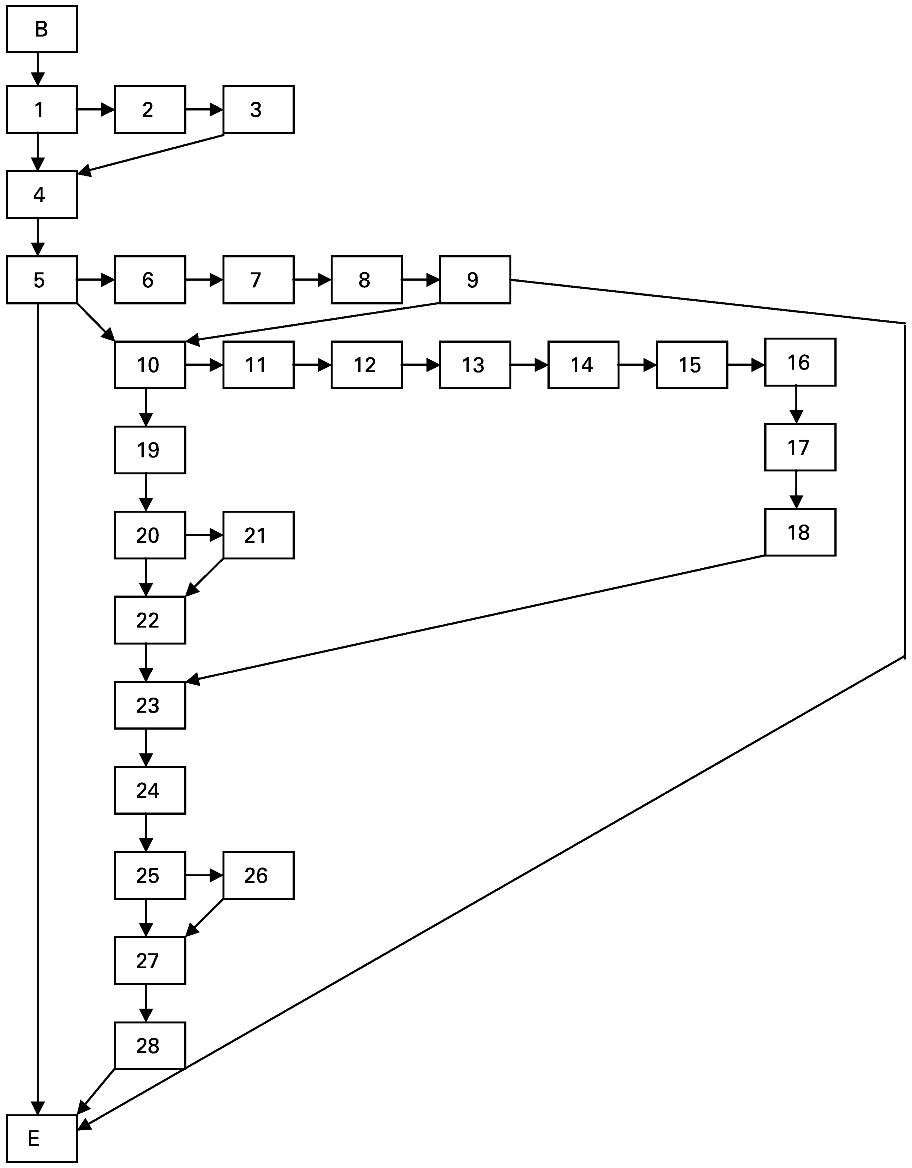


Fig. 4.1. The routing graph of a questionnaire

The condition consists of a question and an answer to the question. If the specific answer has been given, the Yes-branch is followed, otherwise the No-branch in followed.

The use of flow charts is not new in the survey world. For example, Jabine (1985) describes flow charts as a tool to design survey questionnaires. Particularly, flow charts seem to be useful in the early stages of questionnaire development. Sirken (1972) used flow charts to effectively explore alternative structures and sequences for subquestionnaires. He also found that more detailed flow charts, e.g., of the structure of subquestionnaires, could be equally effective. Figure 4.2 contains an example of the type of flow chart used by

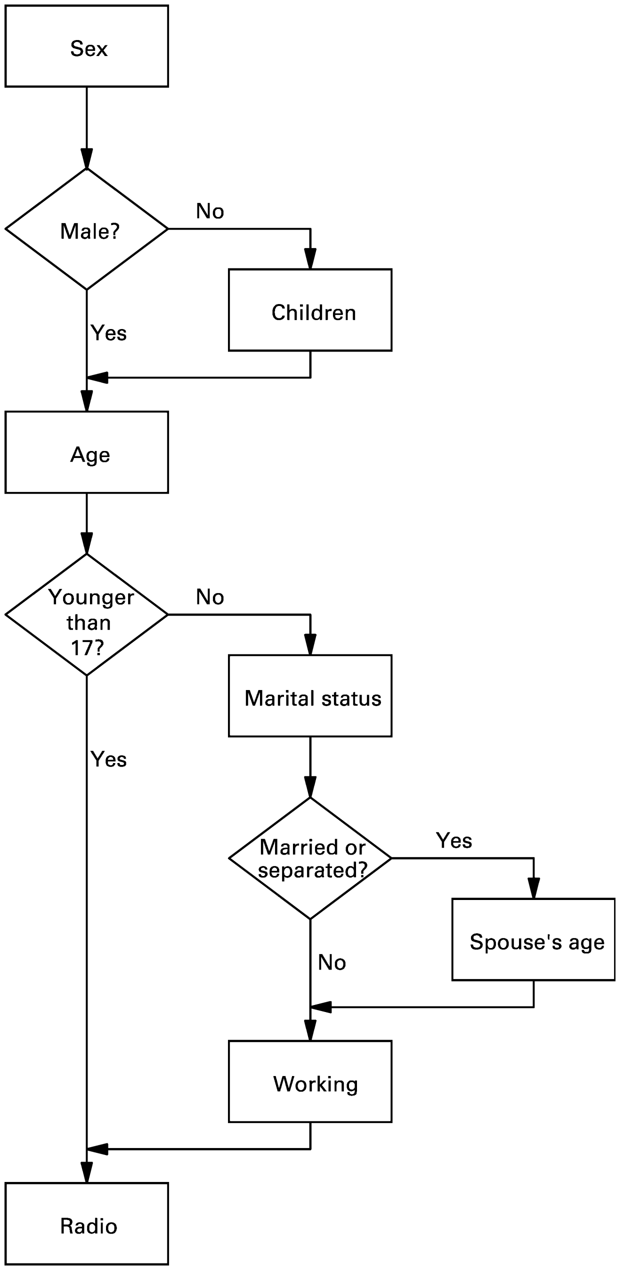


Fig. 4.2. A flow chart of a questionnaire

Sirken (1972). It is the sample questionnaire that has also been used in Section 3. Another, more recent example is the Questionnaire Designer System developed by Katz et al. (1999). This system seems not to have been used yet in a survey production environment. One of its drawbacks is that flow charts have to be produced separately from the written paper specifications of the questionnaire.

A flow chart can also be a useful tool in the documentation of electronic questionnaires. Their strong point is that they can give a clear idea of the routing structure. But they also have the weak point that the amount of textual information that can be displayed about the questionnaire object is limited. Therefore, a flow chart can be a very important component of questionnaire documentation, but it will not be the only component.

5. The Questionnaire Definition Language

The objective of the TADEQ project was to build a documentation and analysis tool that can be used in combination with several CAI systems. This required a generic language able to describe objects in and the routing structure of different authoring languages. To this end, the Questionnaire Definition Language (QDL) was developed. It is based on XML.

XML was once seen as the successor of HTML, the markup language used to define web pages. HTML was originally designed as a means of presenting static information for display purposes only. Over time it developed into a system for developing full-blown interactive web applications. It became clear that HTML had a number of limitations. One of them was the lack of means for describing data structures, another was the lack of extensibility of the language, e.g., to function as an interface to database applications.

XML was created to solve the problems with HTML. It should be as simple as HTML, but it should also be able to describe structure, and it should be extensible. XML turned out to be much more powerful than HTML. Users can define their own language elements, and structure is separated from layout. It is becoming more and more clear that XML is not only a powerful language for designing web sites, but is also very useful to define structure of data files. (See Boumphrey et al. 1998, and Morrison et al. 2000.)

XML concentrates on defining structure in documents. For presenting information in a readable format, *stylesheets* are used. Stylesheets define layout for an XML document. One way of doing this is to make use of *Cascading Style Sheets* (CSS). The possibilities of CSS are limited. It comes down to defining the appearance (font family, font size, colour, italics, bold, underline, etc) of the text between tags. Another way of doing this is using XSL (Extended Stylesheet Language). XSL is much more powerful than CSS. It has various possibilities not only to define layout aspects, but also to transform the information in an XML document. TADEQ uses CSS to produce textual documentation in HTML format. Users may adapt layout by defining their own CSS. Currently, TADEQ does not make use of XSL because it was considered to be of limited use.

Formally, XML is a meta-language, i.e., it is a language to define languages. QDL is an XML application. It is a new language that allows for the definition of the different objects one might encounter in an electronic questionnaire. Examples are questions (various types), checks, computations, route instructions, subquestionnaires, etc. Each object has a number of attributes, like a question text (for questions), logical expressions (for route instructions and checks) and arithmetical expressions (for computations). All objects have an attribute in common indicating the next object to be processed. Some objects (closed questions, route instructions) can have several successors. Which successor will be processed depends on the value of logical expressions involved. (For more details about QDL, see Bethlehem and Hundepool 2002.)

XML has also raised interest in the statistical community, in particular in the area of metadata. An example is the ADDSIA project, financed by the Fourth Framework of the European Union (see Bi and Murtagh 1998). XML is used in this project to describe the metadata for the statistics on economic indicators. Another example is the DDI, the Data Documentation Initiative. This is an initiative of the international social science community of researchers and archivists to develop a standard for a codebook describing variables in social survey data files (see the web site www.icpsr.umich.edu/DDI/codebook.html.)

The use of XML described above concentrates on documenting data files, and pays little or no attention to the instrument used to collect the data. TADEQ uses XML to describe the elements and routing structure of an electronic questionnaire.

The advantage of having a questionnaire definition in XML format is that there are various interesting possibilities of processing the information in a fairly straightforward way. One way is to make use of an XML parser offered by Microsoft. This parser comes for free with the Internet Explorer 5.0 browser (and later versions).

6. The Global Structure of TADEQ

TADEQ is a tool that can be used in combination with several CAI systems. All developers of a CAI system can make their own conversion tool to put the questionnaire definition into QDL format. And once it is in this format, it can be processed by TADEQ. A conversion tool has already been developed for the Blaise system. It is a piece of software (DLL) that transforms the information about questionnaire objects, as it is available internally in Blaise into a QDL document. Figure 6.1 gives an example. The top part of the table contains a question object in the Blaise language. The bottom part shows how it is translated into QDL.

Each CAI package to be supported must have an interface that can translate a questionnaire specification in the authoring language of the system into a QDL file. TADEQ reads the QDL file, and checks its syntax. If accepted, the available information is displayed on the screen in the format of a tree view. Certain parts of the questionnaire (or the complete questionnaire) can then be selected. For the selected subquestionnaire, either textual documentation (the text view) or a routing graph (the graph view) can be generated.

6.1. Textual documentation

After reading the QDL file, TADEQ will present the information on the screen in the form of a *questionnaire tree*. Figure 6.2 shows a version of this tree for a larger sample questionnaire. It is a questionnaire used for a survey among listeners to local radio stations. The specification of this survey instrument in the language of the Blaise system can be found in the Appendix.

Each questionnaire object is represented by a small icon and a short text. Icons containing two-headed arrows indicate points at which a condition is checked that determines the route to be followed. An arrow pointing to the lower right, and all indented objects below it, represent the path that is followed if the condition is satisfied. And an arrow pointing to the lower left, and all indented objects below it, represent the path

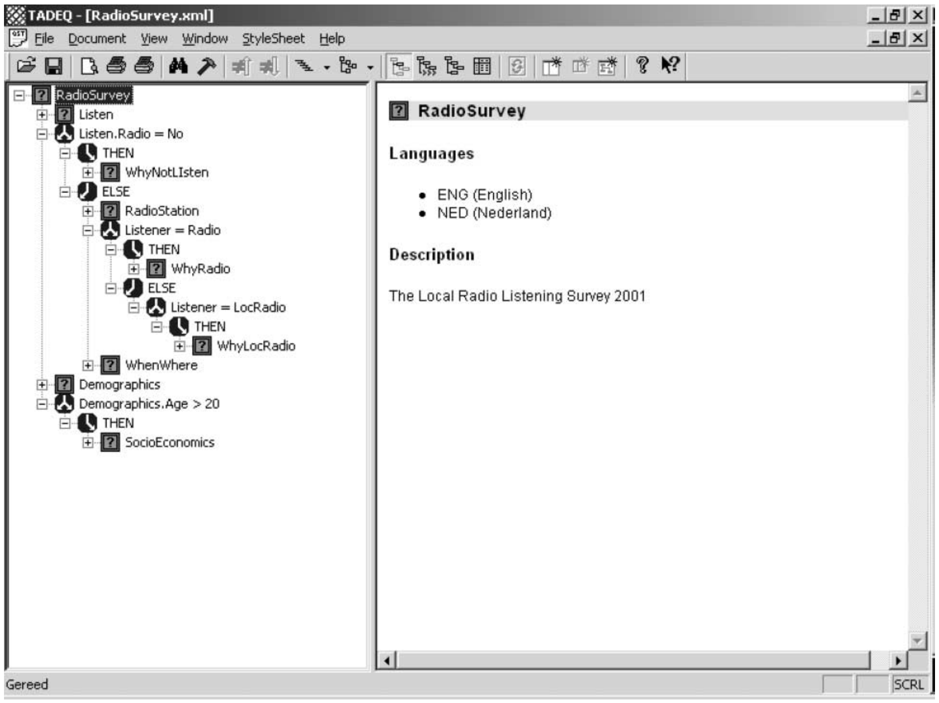
Blaise:

```
WhyNotListen
  "Why do you not listen to the radio? ":
  (NoRadio    "Does not have a radio",
   NoTime     "Does not have time to listen",
   NoInterest "Not interested in radio",
   Other      "Other reason")
```

QDL:

```
<qobject number="3" name="WhyNotListen">
  <question>
    <text>![CDATA[Why do you not listen to the radio?]]></text>
    <closed max="1">
      <item code="1" name="NoRadio">
        <text>![CDATA[Does not have a radio]]></text>
      </item>
      <item code="2" name="NoTime">
        <text>![CDATA[Does not have time to listen]]></text>
      </item>
      <item code="3" name="NoInterest">
        <text>![CDATA[Not interested in radio]]></text>
      </item>
      <item code="4" name="Other">
        <text>![CDATA[Other reason]]></text>
      </item>
    </closed>
    <status>ask</status>
  </question>
</qobject>
```

Fig. 6.1. A questionnaire object in Blaise and QDL



```
RadioSurvey
  Listen
  Listen.Radio = No
    THEN
      WhyNotLIsten
    ELSE
      RadioStation
      Listener = Radio
        THEN
          WhyRadio
        ELSE
          Listener = LocRadio
            THEN
              WhyLocRadio
      WhenWhere
  Demographics
  Demographics.Age > 20
    THEN
      SocioEconomics
```

RadioSurvey

Languages

- ENG (English)
- NED (Nederland)

Description

The Local Radio Listening Survey 2001

Fig. 6.2. A questionnaire tree showing only the highest hierarchical level

that is followed if the condition is not satisfied. So, the various branches of the execution tree are indicated by means of indentation. For example, the object *WhyNotListen* (a subquestionnaire) is only executed if the question *Radio* in the subquestionnaire *Listen* (*Listen.Radio*) is answered by *No*.

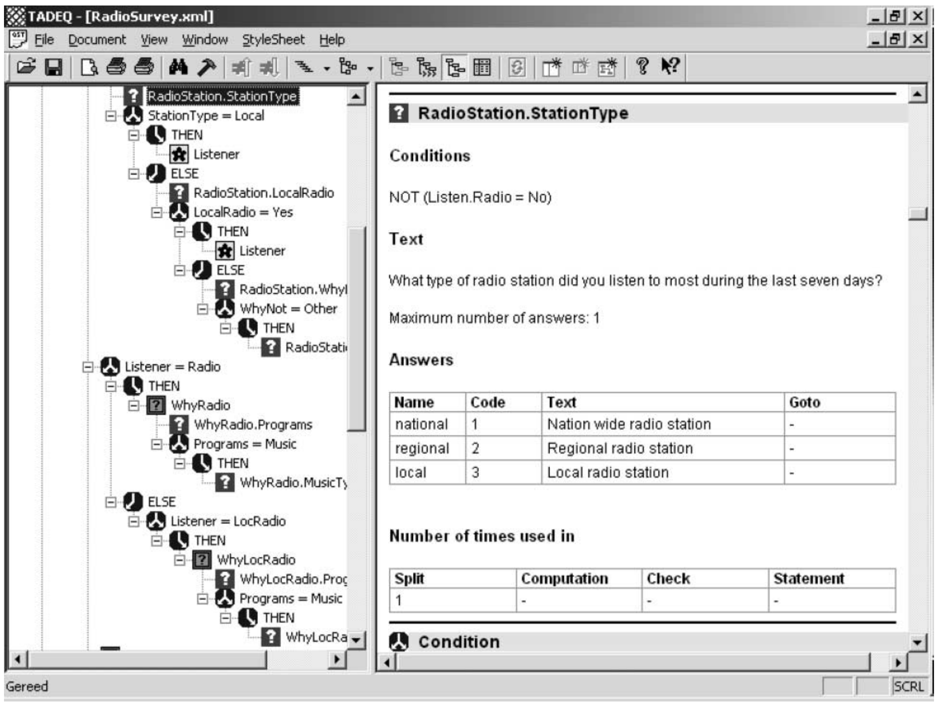
Branches and subbranches may be *collapsed* (i.e., closed so that they are temporarily invisible) or *expanded* (i.e., opened so that they become visible).

Figure 6.2 only shows the highest level of the sample questionnaire. The icons with the question marks indicate subquestionnaires. The dot notation is used to indicate questions in sub-questionnaires. For example, *Listen.Radio* indicates the question *Radio* in the sub-questionnaire *Listen*. The *plus* signs indicate that these subquestionnaires are currently collapsed but can be unfolded.

On the one hand, the tree gives a useful, but global overview of what may happen during the execution of a questionnaire. By clicking on a questionnaire object in the execution tree, the user can obtain detailed information about it. Figure 6.3 contains the information displayed after selecting the question *StationType* in the subquestionnaire *RadioStation*. Two new types of icons appear. The question marks in squares without a border indicate questions, and the stars indicate computations to be carried out.

The window on the right contains detailed information about the currently selected object in the tree. In Figure 6.3, the question *StationType* in the subquestionnaire *RadioStation* has been selected.

Under the heading *Conditions* there may be a list of conditions that have to be fulfilled for the question to be executed. Here, there is only one such condition. The question


TADEQ - [RadioSurvey.xml]
File Document View Window StyleSheet Help
Gereed
SCRL

```
RadioStation.StationType
StationType = Local
  THEN
    Listener
  ELSE
    RadioStation.LocalRadio
    LocalRadio = Yes
      THEN
        Listener
      ELSE
        RadioStation.Whyl
        WhyNot = Other
          THEN
            RadioStati
Listener = Radio
  THEN
    WhyRadio
      WhyRadio.Programs
      Programs = Music
        THEN
          WhyRadio.MusicTy
  ELSE
    Listener = LocRadio
      THEN
        WhyLocRadio
          WhyLocRadio.Prog
          Programs = Music
            THEN
              WhyLocRa
```

RadioStation.StationType

Conditions

NOT (Listen.Radio = No)

Text

What type of radio station did you listen to most during the last seven days?

Maximum number of answers: 1

Answers

Name	Code	Text	Goto
national	1	Nation wide radio station	-
regional	2	Regional radio station	-
local	3	Local radio station	-

Number of times used in

Split	Computation	Check	Statement
1	-	-	-

Condition

Fig. 6.3. The unfolded questionnaire tree

StationType will only be asked if the answer to the question *Listen.Radio* is not equal to *No*. If the questionnaire object represents a question, the question text is also displayed. The example is a closed question. Therefore there is a list of possible answers.

The questionnaire tree is also a useful navigation aid. The user can use the tree to navigate through the questionnaire, expand or collapse branches, and focus on specific parts of the questionnaire.

Note that the example in Figure 6.3 does not contain all information that can be given about a question. The settings of the system determine how much and which type of information is displayed.

6.2. The routing graph

To obtain more insight into the routing structure of the questionnaire, TADEQ can produce a routing graph. In this graph branches can also be expanded or collapsed. For the initial state of the routing graph, TADEQ uses the state of the questionnaire tree. Figure 6.4 shows the routing graph corresponding to the tree of Figure 6.2.

The rectangles with double lines at the bottom indicate subquestionnaires that can be expanded. The hexagons indicate points at which different routes can be taken. Depending on the value of the expression in this symbol either the *true*-branch or the *false*-branch is taken.

Figure 6.5 contains part of the routing graph of the completely expanded questionnaire. The rectangles represent questions. As far as there is space available, the question text is displayed. The rectangles with the rounded corners indicate computations. For questionnaires of a substantial size, this routing graph can be very large, and therefore cover dozens of sheets of paper. In the case of a more structured questionnaire, it is probably wiser to first select a subquestionnaire, and to study and print the routing graph of each subquestionnaire separately.

6.3. Analysis and statistics

A final element of TADEQ is the possibility of producing some statistics about the questionnaire. These statistics may provide some insight into the performance of the questionnaire when it is executed in the field. A first table produced contains the frequencies of appearance of the various kinds of questionnaire objects. These frequencies could help in characterising the questionnaire. For example:

- A relatively small number of checks could indicate that the quality of the collected data is low.
- A relatively large number of splits could indicate a (too) complex routing structure.
- Open questions are much more difficult to analyse than closed questions. Therefore, the number of open questions should be as small as possible.
- Too few subquestionnaires compared with the number of questions could be an indication of a less well-structured questionnaire.

An example of a table with object frequencies can be found in Figure 6.6. It is the upper table in the right-hand pane.

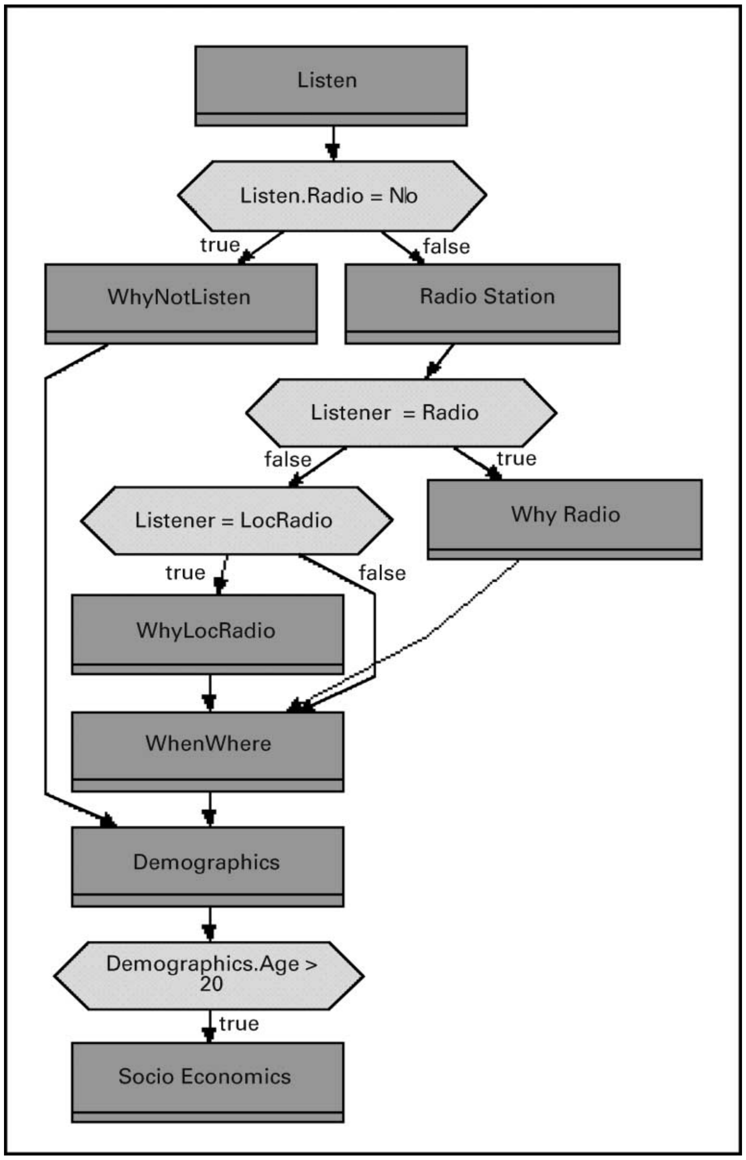


Fig. 6.4. The routing graph corresponding to the tree of figure 6.3

A second table produced by the analysis shows which questions are involved in splits, computations, checks, and statements. Figure 6.6 contains an example of such a table. It is the lower table in the right-hand pane.

Questions involved in splits are vital for a correct routing structure. Therefore even more attention must be paid to these questions. Errors in such questions may cause entire subquestionnaires to be processed incorrectly. It is clear that these questions cannot be removed from the questionnaire without seriously affecting its structure.

Also very critical are the questions involved in checks. They have a large effect on the quality of the collected data. Particular attention must be paid to questions involved in

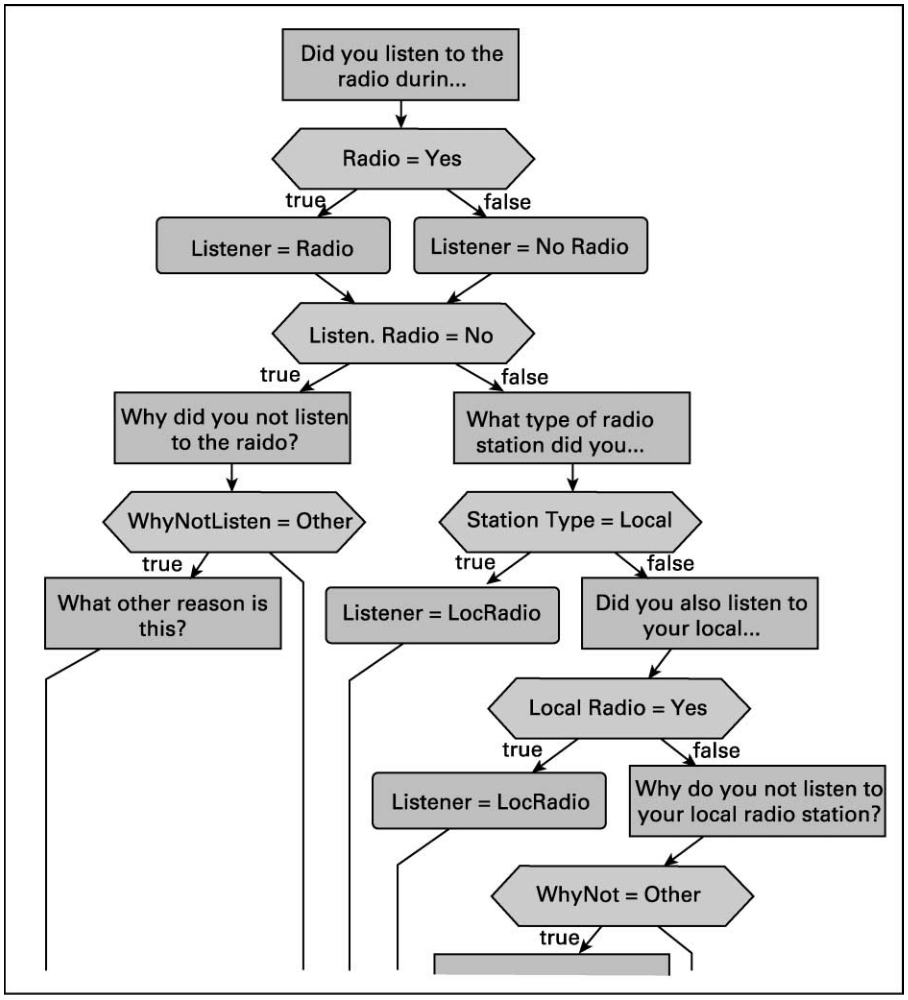
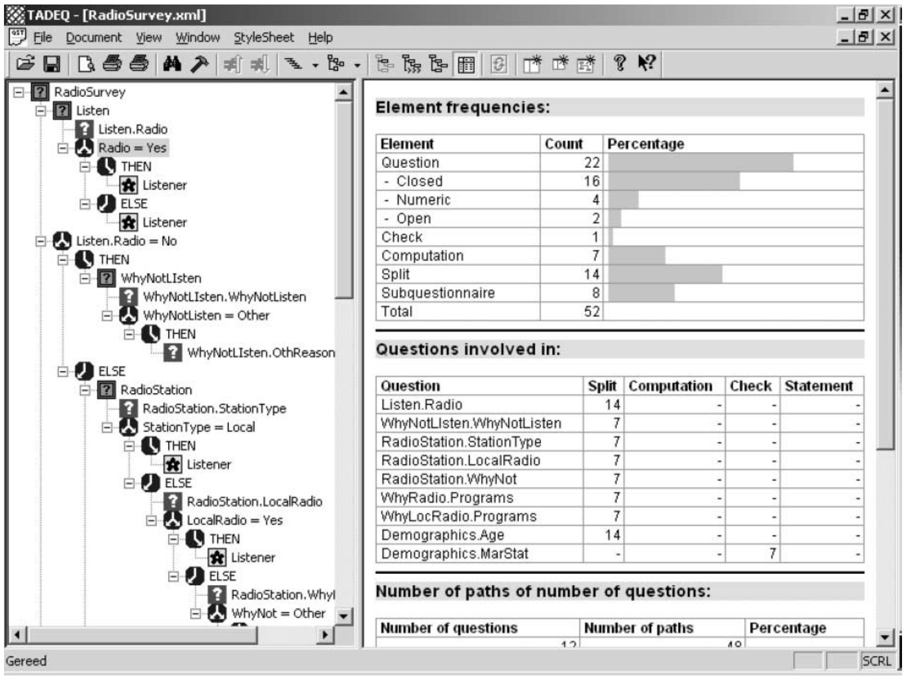

Did you listen to the radio durin...
Radio = Yes
true
false
Listener = Radio
Listener = No Radio
Listen. Radio = No
true
false
Why did you not listen to the raido?
What type of radio station did you...
WhyNotListen = Other
Station Type = Local
true
true
false
What other reason is this?
Listener = LocRadio
Did you also listen to your local...
Local Radio = Yes
true
false
Listener = LocRadio
Why do you not listen to your local radio station?
WhyNot = Other
true

Fig. 6.5. The routing graph of the complete questionnaire

more than one check. Wrong construction of such questions may result in serious problems during the fieldwork. In systems like Blaise, the interviewer cannot continue with the interview unless each condition is satisfied.

Figure 6.6 shows that the Radio Survey questionnaire contains only one check. This means that the correction possibilities of the questionnaire are very limited. The question *Radio* in the subquestionnaire *Listen* is involved in two splits. This question determines the route through the questionnaire at two points. Therefore the design of this question must be carefully checked.

Figure 6.7 contains an example of a third table that can be produced. It contains some route statistics. For each possible path through the questionnaire, its length is computed (in terms of number of questions asked), and also how often paths of such lengths occur. The information is summarised at the bottom of the table. It can be seen that the shortest possible route consists of 5 questions, and the longest route of 18 questions. The average length is 13.6 questions. In total there are 304 different ways to go through to the questionnaire.



Element frequencies:

Element	Count	Percentage
Question	22	
- Closed	16	
- Numeric	4	
- Open	2	
Check	1	
Computation	7	
Split	14	
Subquestionnaire	8	
Total	52	

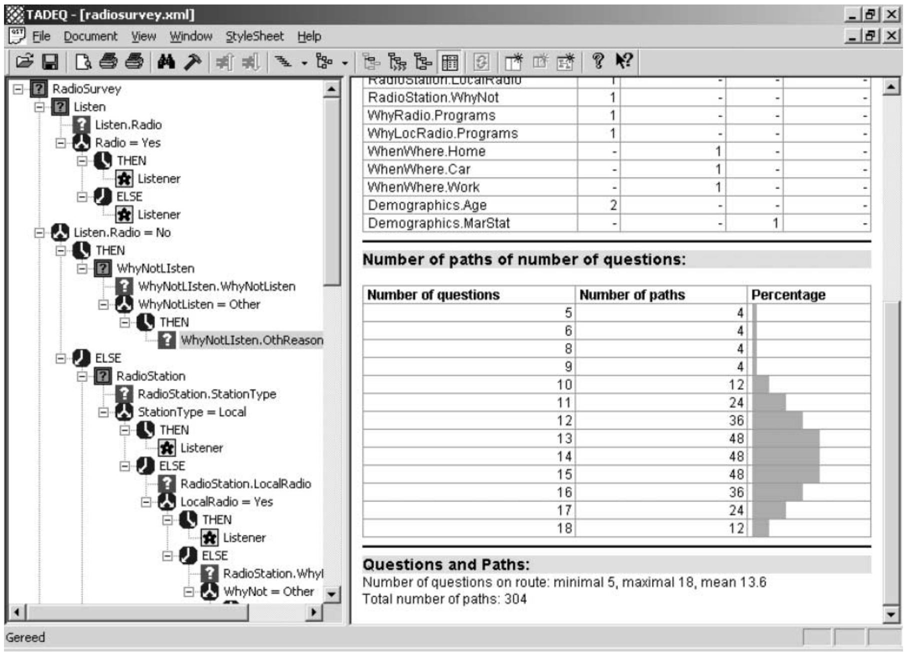
Questions involved in:

Question	Split	Computation	Check	Statement
Listen.Radio	14	-	-	-
WhyNotLIsten.WhyNotListen	7	-	-	-
RadioStation.StationType	7	-	-	-
RadioStation.LocalRadio	7	-	-	-
RadioStation.WhyNot	7	-	-	-
WhyRadio.Programs	7	-	-	-
WhyLocRadio.Programs	7	-	-	-
Demographics.Age	14	-	-	-
Demographics.MarStat	-	-	7	-

Number of paths of number of questions:

Number of questions	Number of paths	Percentage
12	48	

Fig. 6.6. Questionnaire statistics



RadioStation.LocalRadio	1	-	-	-
RadioStation.WhyNot	1	-	-	-
WhyRadio.Programs	1	-	-	-
WhyLocRadio.Programs	1	-	-	-
WhenWhere.Home	-	1	-	-
WhenWhere.Car	-	1	-	-
WhenWhere.Work	-	1	-	-
Demographics.Age	2	-	-	-
Demographics.MarStat	-	-	1	-

Number of paths of number of questions:

Number of questions	Number of paths	Percentage
5	4	
6	4	
8	4	
9	4	
10	12	
11	24	
12	36	
13	48	
14	48	
15	48	
16	36	
17	24	
18	12	

Questions and Paths:
Number of questions on route: minimal 5, maximal 18, mean 13.6
Total number of paths: 304

Fig. 6.7. Route statistics

It should be noted that the current prototype counts the number of logically possible routes. This includes both correct and incorrect routes. An incorrect route is a route that is not possible because of constraints imposed by routing instructions. It would make the tool more interesting if it were able to detect impossible routes through the questionnaire. It could indicate a possible error in the questionnaire specification. Detecting impossible routes would mean completely evaluating all expressions in route conditions. This was beyond the scope of the project. It may be an interesting topic for future research.

Note that these statistics are computed without taking into account the actual contents of the routing conditions. For example, two conditions on one path through the questionnaire may contradict each other, so that it is impossible in practice to complete this path. Such paths are still counted in the statistics.

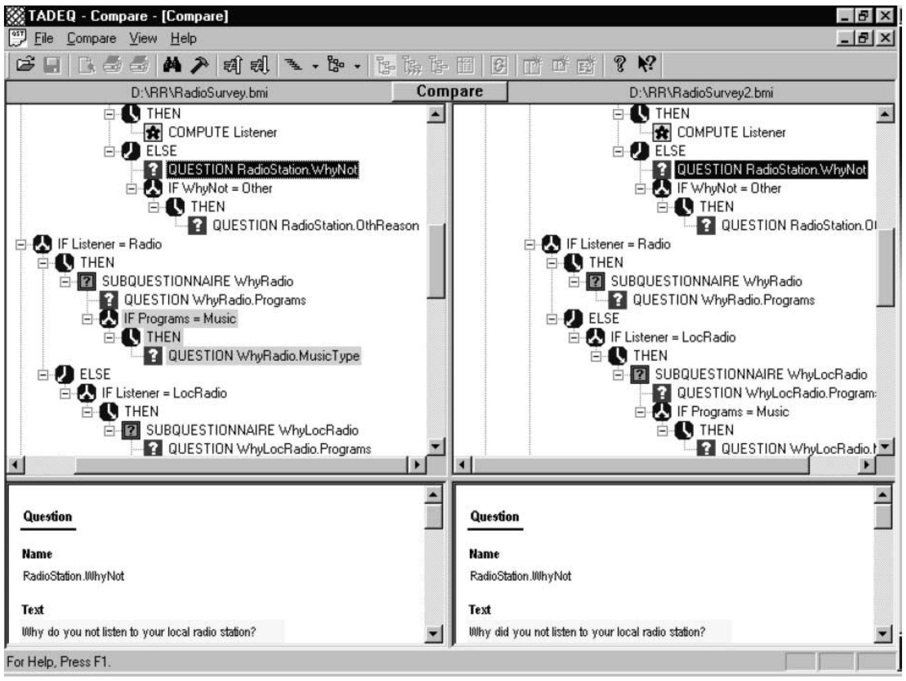
This overview of the main components does not cover all details of TADEQ. More about these details can be found in the TADEQ manual (see Bethlehem and Hundepool 2002).

6.4. Comparing questionnaires

A special feature of TADEQ is the possibility of comparing two different questionnaires. Such a comparison helps in detecting differences between questionnaires. This feature is very useful in comparing different versions of the same questionnaire.

TADEQ is capable of detecting the following differences:

- The routing structure of the two questionnaires is the same. The same type of objects appear at the same positions, but the contents (name, text, expression) of one or more objects are different.



D:\RR\RadioSurvey.bmi

```
        THEN
          COMPUTE Listener
        ELSE
          QUESTION RadioStation.WhyNot
          IF WhyNot = Other
            THEN
              QUESTION RadioStation.OthReason
IF Listener = Radio
  THEN
    SUBQUESTIONNAIRE WhyRadio
      QUESTION WhyRadio.Programs
      IF Programs = Music
        THEN
          QUESTION WhyRadio.MusicType
  ELSE
    IF Listener = LocRadio
      THEN
        SUBQUESTIONNAIRE WhyLocRadio
          QUESTION WhyLocRadio.Programs
```

Question

Name

RadioStation.WhyNot

Text

Why do you not listen to your local radio station?

D:\RR\RadioSurvey2.bmi

```
        THEN
          COMPUTE Listener
        ELSE
          QUESTION RadioStation.WhyNot
          IF WhyNot = Other
            THEN
              QUESTION RadioStation.Ot
IF Listener = Radio
  THEN
    SUBQUESTIONNAIRE WhyRadio
      QUESTION WhyRadio.Programs
  ELSE
    IF Listener = LocRadio
      THEN
        SUBQUESTIONNAIRE WhyLocRadio
          QUESTION WhyLocRadio.Program
          IF Programs = Music
            THEN
              QUESTION WhyLocRadio.
```

Question

Name

RadioStation.WhyNot

Text

Why did you not listen to your local radio station?

Fig. 6.8. Comparing two questionnaire instruments

- A questionnaire object has moved to a different position in the questionnaire tree.
- A questionnaire object has been removed from the questionnaire tree.
- A questionnaire object has been added to the questionnaire tree.

As an example, a different version of the Radio Survey questionnaire has been made. In this new version, the text of the question *WhyNot* has been changed, and the part asking the question *MusicType* has been removed. The results of such a comparison are displayed on the screen in a way such as that illustrated in Figure 6.7.

In Figure 6.8 the cursor is on the first difference encountered. It happened in the question *WhyNot* in the subquestionnaire *RadioStation*. The panes at the lower left and right explain what the differences are. In this case the question on the left contains the word *do* whereas on the right it is replaced by *did*.

The next difference is also already visible in Figure 6.7. Some text on the left has a red background colour (here: grey). This indicates it exists on the left but is not found on the right. It can be seen that it has been removed in the new version of the questionnaire.

The compare function of TADEQ has proved to be very powerful for documenting year-to-year changes in survey questionnaires.

7. Some Future Challenges

The TADEQ prototype has proved to be a useful tool for the documentation of CAI questionnaires. It has several features that make it attractive for use by those involved in the development and execution of surveys that make use of CAI software. Overall, they will already be experienced computer users. It was not expected that the software (features and user-interface) would pose serious problems. The tool has been tested by the users that were involved in the TADEQ project. They were happy with the features and the user-interface.

It should be noted that TADEQ does not solve all documentation problems. Some challenges for the future remain.

In principle, TADEQ supports two types of questionnaire definition languages: those that make use of goto-oriented navigation through the questionnaire, and those that are based on IF-THEN-ELSE constructs. The first type of questionnaire, based on goto's, is much harder to document. It is almost possible to jump from any position to any other position. This may lead to a routing structure that cannot easily be presented in simple, clear planar form. Therefore the graphical part of TADEQ has not yet been completely implemented for goto-oriented routing.

Another problem with goto-oriented questionnaire navigation is the computation of the total number and the length of the possible routes through the questionnaire. This is particularly difficult if there are no restrictions on the types of jump that are allowed.

The possibility of assigning weights to questions, and then computing weighted lengths of routes, is considered important. An application of this feature is to use as weight the (estimated) time required to process a question in an interview. A weighted analysis of path lengths will then provide insight into the variation of total interview length. This information is very useful in fieldwork planning.

TADEQ can be very useful in the development stage of the questionnaire. One feature users would like to have is the possibility of detecting impossible routes. These routes are logically impossible because the various route conditions encountered in following them contradict each other. Impossible routes can only be detected if every route condition encountered is evaluated. This would mean building a complete expression parser. Moreover, the result of these conditions may depend on values of variables that are not available at the time of analysis (e.g., values of variables from previous surveys, or other external data files).

After completion of the survey, it is always useful to evaluate the survey process. A careful analysis of the problems encountered may help to avoid these problems in the future. One way to analyse the routing structure of the questionnaire involves seeing the flow of data through the questionnaire. This can be accomplished by assigning frequencies to questionnaire objects. These frequencies are the number of times respondents encountered the corresponding objects. Route branches with zero (or very low) frequencies may indicate irrelevant or ill-designed parts of the questionnaire. It is also possible to make the flow of data through the questionnaire visible by making the widths of lines connecting objects proportional to the number of times the route was followed by respondents.

Several questionnaire definition languages offer the possibility of using text fills in questionnaire texts. This is a powerful feature that makes it possible to adapt the question text to the situation at hand. However, it is very difficult to document questions using text fills. Since the text to be filled in may vary, and is generally not available at the time of documentation, the actual question text is not available.

An interesting research task could also be to investigate the relationship between structural questionnaire characteristics and the quality of the questionnaire as perceived by interviewers and respondents. If such a relationship could be established, use of TADEQ could lead to better questionnaires in the future.

The TADEQ prototype has been tested in several small and large questionnaires. It can handle large questionnaires. However, the system turned out to be somewhat slow. As a next step in the development of this kind of tool, the Blaise Development Team has decided to build in most parts of TADEQ directly into the Blaise System. In this way, the documentation tool can be more tuned to specific characteristics of the Blaise language, and also processing time can be reduced.

Appendix

The Blaise sample questionnaire

```
DATAMODEL RadioSurvey "The Local Radio Listening Survey 2001"

LANGUAGES =
  ENG "English",
  NED "Nederland"

TYPE
  TMusic =
    (Pop      "Chart/Top fourty" "Hits / Top 40",
     Dance    "Dance" "Dance",
     Rock     "Rock" "Rock",
     Jazz     "Jazz" "Jazz",
     Country  "Country" "Country",
     Classic  "Classical" "Klassiek",
    Easy     "Easy listening" "Easy listening",
    NewAge   "New age" "New age",
    Other    "Other type of music" "Ander soort muziek")
  TProgram =
    (Music    "Music" "Muziek",
     News     "News and current affairs" "Nieuws en lokale
               informatie",
     Sports   "Sports" "Sport",
     Culture  "Culture" "Cultuur",
     Other    "Other programs" "Andere programma's")

AUXFIELDS
  Listener
    "Type of listener" "Type luisteraar":
   (NoRadio
     "Does not listen to the radio"
     "Luistert nooit naar de radio",
    Radio
      "Listens to the radio, but not to the local radio"
      "Luistert naar de radio, maar niet naar een lokale
       omroep",
    LocRadio
      "Listens to the local Radio"
      "Luistert naar de lokale omroep")

BLOCK TListen
  "Listened last week" "Vorige week geluisterd"
  FIELDS
    Radio
      "Did you listen to the radio during the last seven
       days?"
      "Heeft u de afgelopen zeven dagen naar de radio
       geluisterd?":
      (Yes "Yes" "Ja", No "No" "Nee")
  RULES
    Radio
    IF Radio = Yes "Listened to the radio?" "Luistert naar
    de radio?"
    THEN
      Listener := Radio
    ELSE
      Listener := NoRadio
    ENDIF
ENDBLOCK
```

```
BLOCK TWhyNotListen
  "Reasons for not listening" "Redenen van niet luisteren"
  FIELDS
    WhyNotListen
      "Why did you not listen to the radio?"
      "Waarom heeft u niet naar de radio geluisterd?":
      (NoRadio
        "Doesn't have a radio"
        "Heeft geen radio",
       NoTime
         "Doesn't have time to listen"
         "Heeft geen tijd om te luisteren",
       NoInterest
         "Not interested in radio"
         "Heeft geen belangstelling voor radio",
       Other
         "Other reason"
         "Andere reden")
    OthReason
      "What other reason is this?"
      "Wat is die andere reden?": STRING[40]
  RULES
    WhyNotListen
    IF WhyNotListen = Other
    "Other reason for not listening?" "Andere reden voor niet
     luisteren?"
    THEN
      OthReason
    ENDIF
ENDBLOCK

BLOCK TRadioStation
  "Favorite radio station" "Favoriet radiostation"
  FIELDS
    StationType
      "What type of radio station did you listen to most during the
       last seven days?"
      "Naar welke type omroep heeft u de afgelopen zeven dagen het
       meest geluisterd?":
      (National
         "Nation wide radio station"
         "Landelijke zender",
       Regional
         "Regional radio station"
         "Regionale omroep",
      Local
        "Local radio station"
        "Lokale omroep")
  LocalRadio
    "Did you also listen to your local radio station during the
     last seven days?"
    "Heeft u de afgelopen zeven dagen naar de lokale omroep
     geluisterd?":
    (Yes "Yes" "Ja", No "No" "Nee")
  WhyNot
    "Why did you not listen to your local radio station?"
    "Waarom heeft u niet naar de lokale omroep geluisterd?":
    (NotAware
       "Not aware of existence such a radio station"
       "Wist niet dat er een lokale omroep bestond",
     DontLike
        "Doesn't like local radio station"
        "Houdt niet van lokale omroep",
     Technical
        "Is not able to receive the signal"
        "Kan het signaal niet (goed) ontvangen",
     Other
        "Other reason"
        "Andere reden")
  OthReason
    "What is that other reason?"
    "Welke andere reden?": STRING[40]
RULES
  StationType
  IF StationType = Local
  "Listened most to local radio station?"
  "Luistert meestal naar lokale omroep"
  THEN
    Listener := LocRadio
  ELSE
    LocalRadio
    IF LocalRadio = Yes
    "Listened also to local radio station?"
    "Luistert ook naar lokale omroep?"
    THEN
      Listener := LocRadio
    ELSE
      WhyNot
      IF WhyNot = Other
      "Other reason for not listening to local radio station?"
```

```
       "Andere reden om niet naar lokale omroep te luisteren?"
       THEN
         OthReason
       ENDIF
     ENDIF
  ENDIF
ENDBLOCK

BLOCK TWhyRadio
  "Favorite program" "Favoriet programma"
  FIELDS
    Programs
      "Which type of programs do you listen to most?"
      "Naar welk soort programma’s luistert u het meest?":
       TProgram
    MusicType
      "Which type of music do you listen to most?"
      "Naar welk soort muziek luistert u het meest?": TMusic
  RULES
    Programs
    IF Programs = Music "Listened to music?" "Luisterde naar
    muziek?"
    THEN
      MusicType
    ENDIF
ENDBLOCK

BLOCK TWhyLocRadio
  "Favorite local radio program" "Favoriet programma bij de
   lokale omroep"
   FIELDS
     Programs
       "Which type of programs do you listen to most on your local
        radio station?"
       "Naar welk soort programma’s luistert u het meest?":
        TProgram
     MusicType
       "Which type of music do you listen to most?"
       "Naar welk soort muziek luistert u het meest?": TMusic
   RULES
     Programs
     IF Programs = Music "Listened to music?" "Luisterde naar
     muziek"
     THEN
       MusicType
     ENDIF
```

```
ENDBLOCK;

BLOCK TWhenWhere
  "Listening behaviour" "Luistergedrag"
  LOCALS
    RadioTxt: String[20]
  FIELDS
    Home
      "How many hours did you spend listening to the ^RadioTxt at
       home during the last seven days?"
      "Hoeveel uur heeft u thuis naar de ^RadioTxt geluisterd
       gedurende de laatste zeven dagen?": 0..100
    Car
      "How many hours did you spend listening to the ^RadioTxt in
       your car during the last seven days?"
      "Hoeveel uur heeft u in de auto naar de ^RadioTxt geluisterd
       gedurende de laatste zeven dagen?": 0..100
    Work
      "How many hours did you spend listening to the ^RadioTxt at
       work during the last seven days?"
      "Hoeveel uur heeft u op uw werk naar de ^RadioTxt geluisterd
       gedurende de laatste zeven dagen?": 0..100
    Hours
      "How many hours did you spend listening to the ^RadioTxt in
       total during the last seven days?"
      "Hoeveel uur heeft u in totaal naar de ^RadioTxt geluisterd
       gedurende de laatste zeven dagen?": 0..300
    TypeDay
      "At what type of day you listen most to the ^RadioTxt"
      "Op welke dag van de week luistert u het meest naar de
       ^RadioTxt?":
      (Monday     "Monday"     "Maandag",
       Tuesday    "Tuesday"    "Dinsdag",
       Wednesday  "Wednesday"  "Woensdag",
       Thursday   "Thursday"   "Donderdag",
       Friday     "Friday"     "Vrijdag",
       Saturday   "Saturday"   "Zaterdag",
       Sunday     "Sunday"     "Zondag");
    TimeDay
      "At what time of day you listen most to the
       ^RadioTxt"
      "Op welke tijd van de dag luistert u het meest naar
       de ^RadioTxt?":
```

```
      (Morning   "In the morning (6-12 AM)"   "'s ochtends
                                              (6-12 uur)",
      Afternoon "In the afternoon (12-6 PM)" "'s middags
                                              (12-18 uur)",
      Evening   "In the evening (6-12 PM)"   "'s avonds
                                              (18-24 uur)",
      Night     "In the night (12-6 AM)"     "'s nachts")
  PROCEDURE TotalHours
    PARAMETERS
      Arg1, Arg2, Arg3: INTEGER
      EXPORT Tot: INTEGER
    RULES
      Tot := Arg1 + Arg2 + Arg3
  ENDPROCEDURE

  RULES
    IF Listener = LocRadio
    "Listened to local radio?" "Luisterde naar lokale omroep?"
    THEN
      IF ACTIVELANGUAGE = ENG "English version?"
      "Engelse versie"
      THEN
        RadioTxt := 'local radio station'
      ELSE
        RadioTxt := 'lokale omroep'
      ENDIF
    ELSE
      RadioTxt := 'radio'
    ENDIF
    Home Car Work
    TotalHours(Home, Car, Work, Hours)
    TypeDay TimeDay
ENDBLOCK

BLOCK TDemographics
  "Demographic characteristics" "Demografische kenmerken"
  FIELDS
    Sex
      "What is your sex?"
      "Wat is uw geslacht? ":
      (Male "Male" "Man", Female "Female" "Vrouw")
    Age
      "What is your age?"
      "Wat is uw leeftijd?": 0..120
    MarStat
      "Wat is your marital status?"
      "Wat is uw burgerlijke staat?":
      (NeverMar    "Never married"   "Ongehuwd",
       Married     "Married"         "Gehuwd",
       Divorced    "Divorced"        "Gescheiden",
       Widowed     "Widowed"         "Weduwe/Weduwnaar")
  RULES
    Sex Age MarStat
    IF Age < 15 "Younger than 15?" "Jonger dan 15?"
    THEN
      MarStat = NeverMar "Then never married!" "Dan ongehuwd!"
    ENDIF
ENDBLOCK

BLOCK TSocioEconomics
  "Work and education" "Werk en opleiding"
  FIELDS
    Employed
      "What is your employment status?"
      "Wat is uw werksituatie?":
      (NotEmp     "Not employed"                "Heeftgeenwerk",
       EmpOutside "Employed outside the home"  "Heeft werk
                                                 buitenshuis",
       HomeBased  "Have a home-abasd business""Werkt thuis",
       Retired    "Retired"                    "Met pensioen")
    Income
      "What is your monthly family income (after taxes)?"
      "Wat is het maandelijks netto inkomen van het huishouden":
      (Less1000    "Less than 500 Euro"   "Minder dan 500 Euro",
       To1000      "500 - 1000 Euro"      "500 - 1000 Euro",
       To1500      "1000 - 1500 Euro"     "1000 - 1500 Euro",
       To2000      "1500 - 2000 Euro"     "1500 - 2000 Euro",
       To2500      "2000 - 2500 Euro"     "2000 - 2500 Euro",
       More2500    "2500 Euro or more"    "2500 Euro or more")
    Education
      "What is your highest level of formal education?"
      "Wat is het niveau van de hoogst behaalde opleiding?":
      (PrimEduc  "Primary education"  "Lager onderwijs",
       SecEduc   "Secondary education""LBO/MBO//MAVO/HAVO/VWO",
       TertEduc  "Tertiary education" "HBO",
       UnivEduc  "University education" "Universiteit")
  RULES
    Employed Income Education
ENDBLOCK
```

```
FIELDS
  Listen: TListen
  WhyNotLIsten: TWhyNotListen
  RadioStation: TRadioStation
  WhyRadio: TWhyRadio
  WhyLocRadio: TWhyLocRadio
  WhenWhere: TWhenWhere
  Demographics: TDemographics
  SocioEconomics: TSocioEconomics

RULES
  Listen
  IF Listen.Radio = No
  "Did not listen to the radio?" "Luisterde niet naar de radio"
  THEN
    WhyNotListen
  ELSE
    RadioStation
    IF Listener = Radio "Listened to the radio?" "Luisterde naar
    de radio"
    THEN
      WhyRadio
    ELSEIF Listener = LocRadio
    "Listened to the local radio?" "Luisterde naar lokale
     omroep"
    THEN
      WhyLocRadio
    ENDIF
    WhenWhere
 ENDIF
 Demographics
 IF Demographics.Age > 20 "Older than 20?" "Ouder dan 20?"
 THEN
   SocioEconomics
 ENDIF
ENDMODEL
```

8. References

Anderson, S. (1997). Automated Paper Documentation of Blaise III. Actes de la 4e Conférence Internationale des Utilisateurs de BLAISE, INSEE, Paris, 1–20.

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

Bethlehem, J.G. (1997). A Taste of Blaise. Technical Report. Voorburg: Statistics Netherlands.

Bethlehem, J.G. and Hundepool, A.J. (2002). TADEQ Prototype Version 1.5, User Manual. Technical Report. Voorburg: Statistics Netherlands.

Bi, Y. and Murtagh, F. (1998). The Roles of Statistical Metadata and XML in Structuring and Retrieving Statistical Information. Pre-proceedings of the International Seminar on New Techniques and Technologies for Statistics, Sorrento, Italy, 73–78.

Boumphrey, F., Direnzo, O., Duckett, J., Graf, J., Hollander, D., Houle, P., Jenkins, T., Jones, P., Kingsley-Hughes, A., Kingsley-Hughes, K., McQueen, C., and Mohr, S. (1998). XML Applications. Birmingham, USA: Wrox Press.

Cork, D.L., Cohen, M.L., Groves, R., and Kalsbeek, W. (2003). Survey Automation: Report and Workshop Proceedings. Washington D.C.: National Research Council.

Couper, M.P., Baker, R.P., Bethlehem, J., Clark, C.Z.F., Martin, J., Nicholls, W.L., and O'Reilly, J.M. (1998). Computer Assisted Information Collection. New York: John Wiley and Sons.

CSM, The IDOC Project. http://sda.berkeley.edu:7502/idoc/members.htm

ICPSR, The Data Documentation Initiative. http://www.icpsr.umich.edu/DDI

Jabine, T.P. (1985). Flow Charts: A Tool for Developing and Understanding Survey Questionnaires. Journal of Official Statistics, 1, 189–207.

Katz, I.R., Stinson, L.L., and Conrad, F.G. (1999). Questionnaire Designer versus Instrument Authors: Bottlenecks in the Development of Computer-Administered Questionnaires. Memo, University of Michigan.

Kelly, M. and Kuusela, V. (2000). User Requirements. TADEQ Working Paper 1. Voorburg: Statistics Netherlands.

Kent, J.P. and Willenborg, L.C.R.J. (1997). Documenting Questionnaires Research Paper no. 9708. Voorburg: Department of Statistical Methods, Statistics Netherlands.

Morrison, M., Boumphrey, F., and Brownell, D. (2000). XML Unleashed. Indianapolis: Sams Publishing.

Sirken, M. (1972). Designing Forms for Demographic Surveys. Laboratory for Population Statistics Manual Series, No. 3. Chapel Hill: University of North Carolina.

Walton, C.J. (1999). Personal communication.

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

Received February 2003
Revised December 2003