Generating Blaise with Blaise

Gerrit de Bolster, Statistics Netherlands, July 26, 2004

1. Introduction

At Statistics Netherlands the first version of the IMPECT information system was implemented in 2001 to support the Business Surveys. This information system includes a questionnaire server linked with a variable server and a classification server. These servers, built with Visual Basic and SQL-server, are used by end-users to define their questionnaires interactively. Once finished, they can use the system to generate questionnaires from the same meta data for different media: paper (using Post Script), Blaise IS and Blaise to be used with the Electronic Data Reporter (EDR) as well as data entry machines for the processing of data from paper forms. The first version only supported the generation of paper questionnaires, the second version was extended to generate also Blaise IS and Blaise questionnaires. This version was limited yet to simple questionnaires like the Short Term Surveys on monthly and quarterly turnover. In this generation process Maniplus set-ups are used to convert the metadata, exported with Visual Basic programs from the SQL-server database as ASCII files, into the final product.

A Maniplus prototype has been developed to generate from this metadata larger Blaise questionnaires as used for the yearly Production Surveys. The questionnaires themselves are basically not complex but extensive. To deal with that the sections defined in the metadata are translated into parallels in the Blaise questionnaire. The questions are considered as rows in a table with a maximum of four answer columns following it. Besides that different symbols are supported for aggregation, subtraction and transport of results to other sections. So called 'table-questions' (e.g. value and quantity of sales per product) are featured too. The generation system supports all combinations of answer columns of different types and symbols by building a question row as a Block in a data model using pre-defined pieces of Blaise language in which parameters are substituted. It adjusts the height of the block on the screen depending on the number of text lines in combination with the size of symbols and answer fields. The explanations are generated as a Windows help file linked to the related questions and sections using the Blaise help facility. Rules are added to sum or subtract different sets of answers to a higher aggregation level, even throughout the different sections. As a special feature enterprise dependant fillings of 'table-questions' in the same type of questionnaire are supported without changing the structure of the data model so the data can be stored in the same data file (.bdb). This paper describes the generation process with all its ins and outs.

2. Questionnaire server

2.1 User-interface

The questionnaire server in the IMPECT system supports a hierarchical (tree) structure for business questionnaires. The metadata is stored in an SQL-server database. The end-users responsible for the maintenance of the questionnaire metadata are given access through a Visual Basic application. This application has a user-interface similar to Outlook. It contains 3 windows: one on the left-hand side of the screen and two on the right-hand side, one above the other. In the left-hand side window a tree-view of the hierarchical structure of the questionnaire is shown. The highest level consists of a list of all the questionnaires available. By clicking on this structure a part of it can be unfolded so detailed levels will be visible. Depending on the focus in the tree the two windows on the right-hand side show different information. In the top window auxiliary and linked information is shown, e.g. the list of possible question texts. By clicking on this information it can be used to define or alter the questionnaire. In the bottom window detailed information about the focused item itself is shown.

2.2 Questionnaire structure

A 4-digit number uniquely identifies a questionnaire within the questionnaire server. This number is presented in the questionnaire preceded by the characters 'VL'. These characters are an abbreviation of the Dutch word for questionnaire: *vragenlijst* and are used to distinguish these questionnaires from those who are manually created by our repro-department. The latter ones start with the characters 'RZ', abbreviation of the Dutch word for repro-department. Besides its identification number a questionnaire has a title and is linked to a certain target group within the population of enterprises. A questionnaire is divided into so-called rubrics. Every rubric has a (capital) letter as identification and a header. A rubric is used to group questions related to one subject expressed in the header. They can be considered as the chapters in a questionnaire. On itself a rubric is divided into questions. The questions are identified by their position within the rubric. The most important element of a question is its question text. This text is taken from a separated table and can be re-used. A question can also be given a sequential number within the scope of the rubric. This sequential number preceded by the rubric identification character is used as a link to explanations. To every question 0 to 4 answer columns can be added. If no answer column is added the question text can be seen as a sub-header within a rubric followed by a number of real questions. On the rubric level each of this columns can be given a header. At every level (questionnaire, rubric, question) an explanation text can be added.

2.3 Variables

For internal identification every answer is linked to a variable. These variables are stored in a different database, the variable server. The variables are identified by their own identification code. Within the questionnaire server each variable corresponds with an answer column of a question. Of course, a variable can be used in different questionnaires. Disseminated questionnaires do not contain these variable identifiers. On the paper questionnaires there is no place for these 13 character long identifications. Besides, it has no use to send out information that is not useful for the respondents as it only increases the volume of the questionnaires. When a statement is received by our input systems the variable identifier is automatically added to the answers for further processing.

2.4 Classifications

A special type of questions is called "table-question". A table question consists of a set of rows related to a classification, e.g. the value of the purchases per product. Each row is connected to a code within the classification. A filled-in answer is thus identified by the variable identification (the column) in combination with a classification code (the row). The identification of the classification used is linked to the question in the questionnaire server. The text for each row is taken from the classification server. A question can consist of more than one column related to a classification, e.g. the quantity and value of the purchases per product. Within a classification a measurement indicator is available to be used in combination with a quantity. When applied, this indicator appears in a column preceding the answer column for the quantities. If the indicator is empty the answer box is not shown in the questionnaire. This is necessary because the quantities are only asked for a subset of the rows.

In the yearly Production Survey (PS), in the target group of the industry the classification for goods is very extensive (10.000 items). The list is too long to be presented to a respondent in an electronic questionnaire, it is even impossible to print it on a paper questionnaire. For this reason a function is added to select only a subset of the possible codes depending on the enterprise involved. As a consequence individual different questionnaires (as far as it concerns table-questions) are supported by the IMPECT system.

3. Generation process

3.1 The metadata file

The metadata in the questionnaire server in combination with the variable server and the classification server are the source for the questionnaires. (This three-some is called the meta-server.) The metadata is exported from these databases by a query invoked by a Visual Basic program. Its output is an ASCII-file with an XML-like structure. This structure has explicitly been chosen to be flexible and extensible on one hand and to be read easily by a Manipula setup on the other hand. It is hierarchical as the metadata itself is hierarchical. The same file is used to generation the questionnaires for all the possible media: paper (PostScript), EDR (Blaise data model) and Blaise IS/form-based mode (Blaise data model converted finally to HTML). It is also used to generate the Blaise data-entry engines for the paper questionnaires (Blaise data model). It will also be used to generate the questionnaire for Blaise IS/page based mode (Blaise data model).

Every section in the file is starting with a tag between the characters "<" and ">", just like in XML. However, the ending tag is just a line with the characters "<>". The information lines are preceded by a two-character code, mainly two letters in uppercase (A-Z). Within a section several sub-sections can be present and within a sub-section several sub-sub-sections, thus representing the questionnaire structure. The lines are all starting in the first column to make it easier (and faster) for Manipula to read. Its general structure looks like this:

```
<questionnaire>                         *****************
AAquestionnaire information lines                       |
<rubric>                                 *************  |
BBrubric information lines                           |  |
<question>                                ********   |  |
CCquestion information lines                     |   |  |
<column>                                   ***   |   |  |
DDcolumn information lines                   |   |   |  |
<>                                         ***   |   |  |
<column>                                   ***   |   |  |
….                                           |   |   |  |
<>                                         ***   |   |  |
<>                                        ********   |  |
<question>                                ********   |  |
….                                               |   |  |
<column>                                   ***   |   |  |
….                                           |   |   |  |
<>                                         ***   |   |  |
<>                                        ********   |  |
<>                                       *************  |
<rubric>                                 *************  |
….                                                   |  |
<question>                                ********   |  |
….                                               |   |  |
<column>                                   ***   |   |  |
….                                           |   |   |  |
<>                                         ***   |   |  |
<>                                        ********   |  |
<>                                       *************  |
<>                                      *****************
```

3.2 Calculation rules

The metadata stored in the questionnaire server is not rich enough to generate an electronic version of the yearly Production Survey. As the first version of the questionnaire server was only intended to support paper questionnaires, no provision was built in to support calculations of answer values. Even in the second version this was not added, as it was not necessary for the questionnaires of the Short Term Surveys on monthly and quarterly turnover. This second version was extended to support these simple electronic questionnaires. There were not enough resources to upgrade the questionnaire server. It is also understandable that the questionnaire server could not be changed only for a prototype. Therefore a provisional solution was chosen creating some kind of supplement to the questionnaire server. This supplement, consisting of a Blaise database, contains the calculation formulas. After deriving the metadata file from the questionnaire server the calculation rules are added to it enriching the metadata without disturbing its structure. In this way it is very easy to add this supplementary information to the questionnaire server in the future without changing the process. Only the additional step, which adds the calculation rules to the metadata file, must be removed. These rules are based upon the variable identifiers. During the generation process they are substituted by the names of the actual fields in the Blaise data model.

3.3 The process steps

Every step in the generation process is implemented as a Maniplus setup. A main setup (GenEDR.msu) is used to invoke these steps as well as additional programs to parse the generated Blaise data model (B4CPARS.exe) and WinHelp file containing the explanations (HCW.exe). The main setup also takes care of copying the necessary auxiliary files.

The first setup (AddMeta.msu) is enriching the metadata file with the calculation rules as was described before. Additional it also replaces dotted lines in the question text (…. ….) by a text telling the respondent to add descriptions as a remark (Ctrl-M). These dotted lines are used in paper questionnaires to offer the respondent the opportunity to write down in free text the descriptions of goods not covered by the rows of the classification.

The next setup (GenVertMeta.msu) converts the metadata file into a series of 4 text files. The first text file contains a list of the blocks to be generated including their pattern. A block can represent a rubric-header, a question or a additional line containing e.g. symbols to be displayed. How blocks are constructed including the use of the pattern will be explained in the next chapter (From bricks to blocks). The second one contains the text lines to be used in the blocks. The text lines contain special characters for fonts and other layout features mostly the same as used in Blaise (rich text). The third file contains the calculation rules and the fourth the variable identifiers linked to the names of the Blaise fields in the data model.

The third setup (GenBla.msu) generates the actual data model using the 4 files created by the previous setup. A file containing templates of Blaise and Manipula source lines (GenBla.tfm) is read and, based upon the information in the 4 input files, the right set of lines are selected and the variable parts are replaced using text fills. Besides the data model a Maniplus setup is generated so the questionnaire can be used with the CBS data collection tool “Electronic Data Reporter”. The command line parser from Blaise (B4CPARS.exe) is used to parse the data model and the Maniplus setup.

A fourth setup (GenHlp.msu) uses the file containing the text lines to generate a help file with the explanations linked to the questionnaire, the rubrics and the questions. It can be used through the Blaise help mechanism for questionnaires. The setup creates a Windows help project file as well as an RTF-file according to the format of a Windows help file. To do so it selects the lines with the explanation from the text file in combination with a file containing templates with Project directives and RTF-lines (GenHlp.tfm). After generation of the output files the Windows help parser (HCW.exe) is invoked to create the final help file.

A last, small setup is used to create a XML-file that is used by the Electronic Data reporter. Finally the set of files are packed using the Windows cabinet program (CabArc.exe) to make the package file for Electronic Data reporter (.esf).

4. From bricks to blocks

4.1 The basic structure

The basic structure of the generated Blaise data model is a set of tables, one for each rubric, containing a block for every normal question or a block for each row in a table-question. Every table is defined as a parallel so the rubrics will appear as tabs. The main program appearing as the first tab contains a fixed set of questions related to contact information. This is standard within the IMPECT questionnaires. Each field in a table refers to a block. The fields are not created as an array as each block can be different. In stead, the fields are defined each with its own name: a letter with a 4-digit sequential number (e.g. B0006). The blocks they refer to are defined separately depending on their pattern (see next paragraph). Questions with the same pattern are referring to the same type of block. The blocks themselves do not contain any text, this is past on to them through parameters. In the rules, where the fields are put on the route, the correct question text is added as well as the link to the Windows help file containing the explanations.

Example of fields and rules:

```
FIELDS
R0018 /"·":B_HHHHH1
B0019 /"·":B_TTTTT1
B0020 /"@bD.1*·":B_TTTBT1
B0021 /"@bD.2*·":B_TTTBT1

RULES
R0018('@bBedrijfsopbrengsten','·','·','@b@ebedragen x 1000','·','_rbr')
B0019('@bNetto omzet','·','·','·','·','_vrg')
B0020('Netto omzet uit de hoofdactiviteit van de onderneming','·','·','·','D.1')
B0021('Netto omzet uit overige activiteiten','·','·','·','D.2')
```

4.2 Block pattern

The pattern of a block (and thus of a question) is described as a set of 5 capital letters. Each letter represents a column in the table. The first one is a wide column in which the rubric header or the question text is placed. The other four are the answer columns in which an edit box or a text or a symbol can be placed. These symbols, like a line under a sum of fields, are made using normal characters and displayed as text. Each type of letter is linked to a special part of a block. The letter "T" e.g. causes that a part that shows a text is included, a letter "R" an edit box with a field of the type "real". The 5 parts together form a complete block with a unique pattern: from bricks to blocks. The pattern serves as a part of its name as well.

Example of a block:

```
BLOCK B_TTTBB1
PARAMETERS
IMPORT ITekst0 : STRING
IMPORT ITekst1 : STRING
IMPORT ITekst2 : STRING
IMPORT IToelID   : STRING
AUXFIELDS
Tekst0 "^ITekst0" "^IToelID" : STRING[1]
Tekst1 "^ITekst1" "^IToelID" : STRING[1]
Tekst2 "^ITekst2" "^IToelID" : STRING[1]
FIELDS
Vraag3 "^Header @/@b* zie toelichting (Ctrl-F1)" "^IToelID" : -99999999..99999999
Vraag4 "^Header @/@b* zie toelichting (Ctrl-F1)" "^IToelID" : -99999999..99999999
RULES
Tekst0.SHOW
Tekst1.SHOW
Tekst2.SHOW
Vraag3.ASK
Vraag4.ASK
LAYOUT
AT Tekst0 FIELDPANE fptext1
AT Tekst1 FIELDPANE fpcolumn1
AT Tekst2 FIELDPANE fpcolumn1
AT Vraag3 FIELDPANE fpnumber
AT Vraag4 FIELDPANE fpnumber
ENDBLOCK
```

The number at the end of the pattern indicates the height of the layout of fields. The length of the question text determines this height. If it is too long it must be wrapped and the row on the screen must be higher to fit multiple lines. A “1” indicates that only one line is needed, a “2” two lines and so on. The different heights are defined in the standard modelib that is added to this application. This modelib contains a standard set of field layouts, one for each type of field. The calculation of the height needed must still be improved. It depends e.g. on the font size and attributes like “bold”. To complicate the case more: two lines as defined in the modelib can often fit 3 text lines. To conclude: a block is defined by a combination of its pattern and the height of the text lines to be displayed. Generating a block based upon its parts makes it possible to support all type of patterns.

Some times a question can lead to 2 blocks. In the questionnaire server it is possible to define a question column containing an edit box with a line beneath it, to symbolize the aggregation of fields. On a paper questionnaire it can be printed in the same space as any other question. On the screen this is not possible. Blaise does not allow putting an edit box together with text in one table cell. As a consequence in an electronic questionnaire two rows are needed: one for the edit box and one for the aggregation symbol.

5. One size fits all

5.1 Different look, one database

As was mentioned before the system has to support individual different questionnaires as well. These questionnaires are individual in the sense that in one or more table-questions the sub-set of rows is dependant on the enterprise where it is used. A simple solution would be to generate tailor-made data models. However, the IMPECT system demands that questionnaires with the same identification have the same data model. Not only because of the internal processing but also because of the use with the Electronic Data Reporter. Giving them all different identifications would create a chaos. Besides, it is quiet normal that an accountancy office is doing the statistical reporting for several clients. In that case the accountancy office can load the different questionnaires from the clients into its installation of the Electronic Data Reporter. If it is the same questionnaire it is stored once. The different clients are distinguished by their identification as part of the primary key of the data model. Furthermore, it will not be understood that questionnaires would be considered as different because some rows in a table are different. So the problem was to generate different looking questionnaires that were using the same basic data model and thus the same data base (.bdb) to store the data.

5.2 Maximum number of rows

In the questionnaire server in the case of these individual table-questions already a field was available containing a maximum number of rows. This was introduced because the data entry instruments faced the same situation: one instrument per questionnaire. For the instrument it was used to generate a table with "open" rows. "Open" means that the code of the row had to be typed in. As they were printed on the paper questionnaire this was possible. In CASI questionnaires as used in the Electronic Data Reporter such solution is out of the question. The questionnaire should contain a predefined set of rows according to the situation of the responding enterprise. The respondent only has to fill-in the requested values at the right row. To achieve this the maximum number of rows is used to generate a fixed set of blocks for the table-question involved. As the question defines its pattern they all refer to the same block type. The only difference could occur in the height of the text lines. In Blaise a same pattern of fields always produces the same database (.bdb). In other words, doing so individual different questionnaires can use the same database to store the data. The difference in text and number of displayed questions is stored in the rules and thus in the data model (.bmi) and the difference in layout (height) is stored in the layout file (.bdm). By activating the right data model and layout file as another enterprise is selected the problem was solved: *one size fits all*. This has been implemented by replacing the "current" data model and layout file in the working folder by the ones from the respondent stored in another folder. This replacement functionality is included in the generated Maniplus setup that invokes the form using the "edit" statement after the respondent has selected the keys through a dialog. For the use with normal, not individual, questionnaires a setup is generated without this extra functionality. Using the new Blaise feature of directives that are interpreted during preparation time creates the different appearances of the setup. Only by filling in the right directive {$DEFINE …} the matching setup is automatically selected.

6. Concluding remarks

Generating Blaise with Blaise is not difficult. The capabilities of Maniplus to manipulate text files in combination with the structure of Blaise sources offer a wide range of possibilities.

Concerning the process described in this paper: at this moment a pilot is prepared to test the questionnaires generated with the process described in this paper. It will used with Blaise IS, the page based mode. If successful, it will be the first step to on-line interviewing at Statistics Netherlands.