Experiences with Blaise for Data Editing in the Housing Rental Survey

Harm Melief and Mark Kroon, Statistics Netherlands

1. Introduction

This paper reports on a data editing system designed and built at Statistics Netherlands for use in the housing rental survey. The survey collects data on housing rental, which is used to calculate price indices f or housing rental costs and which is also supplied to the Dutch Government. The data is collected partly from paper forms and partly using electronic sources, all of which have to be corrected manually. Chapter 2 describes the survey in more detail and will explain what features a system should contain to support the survey.

The resulting system is described in Chapter 3. The system supports data import into a relational database, scanning it for mistakes, correcting them, calculating the price indices and exporting the results. All these properties were implemented in the Blaise Suite, using Manipula, Maniplus and Blaise. The resulting system is fairly complicated, the implementation of which presented us with a number of unforeseen problems. In chapter 4 our experiences with using Blaise for data editing are described. Finally, these experiences are discussed in chapter 5, where our final conclusions are given.

2. The housing rental survey

2.1 Description of the housing rental survey

The housing rental survey (huurenquête) is conducted by Statistics Netherlands (CBS) and (partly) commissioned by the Dutch Government. The main output of this survey is the housing rental price index, which is computed for both the normal and harmonized rents. Secondary outputs are the garage rental and service charges price indices.

The housing rental survey is a stratified fixed panel survey of some 14000 (rented) houses from approximately 1300 respondents, i.e. rental agencies. Each year a few hundred houses drop from the panel, usually because they are either sold or demolished. Therefore, the panel is yearly replenished from a group of new rental houses. Moreover, every few years the panel is updated to remove any bias and fill out the strata if needed.

Of the roughly 14000 houses 9000 rentals are collected using paper forms, 4500 are obtained from the administrations of large rental organizations and some 500 are provided as electronic forms (Excel). The paper forms are sent out and received by a special department within Statistics Netherlands. They are sent out in May and usually received over a period of one to two months, starting from the beginning of July. The filled-out paper forms are manually entered into an Microsoft Access database, which is updated daily during the response period.

Approximately 50 variables are measured, most of which are used to determine the housing rental price indices. Some variables, however, are specifically collected for the Ministry of Housing, Spatial Planning and the Environment (VROM), which uses this information for setting housing policy.

All forms are inspected for inconsistencies, omissions and mistakes, after which they are manually corrected. Small and obvious mistakes, or mistakes resulting from the typing out process, are immediately corrected, but for most problems the respondents need to be contacted. Much of this work is done by temporary workers. For optimum efficiency all imperfect records of one respondent are collected and treated in one phone call. Thus only one person is responsible for a given respondent, which guarantees that respondents are not needlessly contacted multiple times. Distributing the work amongst the workers is something of a challenge, as will be explained in chapter 3. Preferably all errors are corrected, but to remain cost effective, some are ignored, especially if the involved variables are not used during the calculation of the price index.

Based on 4 categories, type of housing (2), type of rental agency (2), year of building (4) and geographic region (4), the housing rental price index is stratified into $2 \times 2 \times 4 \times 4 = 64$ strata. For the rental prices the following method is used to calculate the price index. Within each of the cells, the sample mean of all observed and corrected records is calculated. The results of the cells are combined using a weighted average to give the average rent of the country, from which the index is calculated as the ratio of the current and the last period. The weights for this calculation are supplied by VROM, which has information about the complete housing rental sector. For the garage rents and service charges the price indices are calculated through an unweighted average of all respondent and corrected records.

2.2 A supporting system

Discussions with the supervisors of the process helped us define the main features of a support system for the survey. As described above the main purpose of the process is the editing of the respondent data and the calculation of the price index. The task of supporting the data editing consists of three parts. Firstly, a large set of rules is needed to detect any imperfections in the forms. Secondly, an incorrect form needs to be conveniently presented for correction. This means that as much information as possible should be presented in one screen and that related variables should be grouped together. Thirdly, due to the large number of forms that need to be corrected, an efficient system for distributing the work amongst the temps was desired. Calculating the price index is a relatively straightforward set of aggregations, which can be supported by many different tools.

The system of course needs to support the import of the survey forms, but there are also a number of supporting data sources that need to be imported. At the end of the process the computed results, but also some of the micro-data needs to be exported for use within our organization. Just as for the calculation the I/O can be performed in many different tools. This means that the most distinctive feature of the desired system is support of the data editing. It was deemed that the Data entry Program of Blaise could support this feature, while Manipula should be able to support the I/O and the price index calculation.

3. Support system in Blaise

We have built a system, which supports the process described above, in the Blaise Suite version 4.8.1, build 1447. In this chapter the main parts of the system are described: data I/O (section 3.2), data editing (section 3.3), calculation (section 3.4) and classification management (section 3.5). The basic architecture of the system and the coordinating main menu are described in section 3.1

3.1 General architecture

The system needs to support a large number of different functionalities. To organise the program, a home screen was built in the form of a Maniplus menu. This menu delivers a menu bar, from which the functionalities may be called up. These functionalities are grouped into three main groups: editing, management and miscellaneous purposes. This last group contains both data I/O and the price index calculation. The management section contains both classification management and user rights.

As the most important function of the system is to support the editing (correction) of the survey forms, this forms the basis around which the system has been build. Consequently, the core of the system is determined by three data models:

1. Panel data: This contains the fixed data of the panel houses. Fields include the address and a link to the rental agency.
2. Respondent data: This contains the information of the rental agencies. Variables include the contact information and identifying properties of the agencies.
3. Variable data: This contains all the variables from the survey. This includes all the rents and service charges, as well as the properties of the houses, such as number of rooms and parking facilities.

These three data models are linked through the panel data model, as this model defines the house through its primary identifier, its address. Besides these main data models, a large number of supporting data models were needed for the different parts of the program. In the next sections these different parts of the program are described at which point the supporting data models are described.

Most data is stored in a SQL database. The data models were connected to corresponding database tables through .boi files. Since Blaise only allows joining one record at a time, we chose to combine data from the different data models by creating views in the database. For instance for the calculation information from all three database tables is needed, which was obtained by making a view in which these three tables are joined. These views can again be accessed through a boi file.

3.2 Data I/O

As mentioned in the previous chapter, the survey is collected either as paper forms, from the administration of a large rental agency organisation or from electronic sources. The paper forms come in two formats, introduction forms for new respondents and panel forms for recurring respondents. Including the two types of electronic data, this means that four statistical sources are used. Moreover, some 8 secondary sources are needed to supply supporting data. These typically include a ZIP-code register, information about the values of the properties and others. The sources have different formats, mdb (Access), xls (Excel) and text (csv and fixed columns).

The output takes two forms. A group of output is derived from the panel and respondent tables. These are used for the setting up of the paper forms at the start of a new survey. These outputs are all text files. In addition to this group there are some 12 different outputs required of different aggregates of the three tables. These outputs are delivered to different customers within and outside the statistics bureau. The most important customer is the government (VROM). These files are also all text files.

All these data are imported or exported using Manipula, which supports all of the required file types. The required data models for the input or output files were mostly programmed by hand, but for all mdb or xls files, which are accessed through a boi file, the Oledb workshop could be used.

3.3 Editing the data

The editing of the data consists of three parts, work assignment, main editing window and record based editing (in a DEP). These three approaches work together to form an interactive environment, which supports a group of data editors. Each of these components is treated in a different subsection.

3.3.1 Work assignment

Much of the actual editing work is done by temporary workers. As explained in the previous chapter mistakes or omissions are corrected by calling the respondents. For optimum efficiency all imperfect records of one respondent are collected and treated in one phone call, which means that the work assignment is grouped by respondent. To facilitate this process a list of all respondents with imperfect records is made. This list is presented in a separate window in a lookup, see figure 1. The list contains the name of the respondent as well as the number of houses, number of mistakes and types of mistakes. From this list entries can be selected which can be assigned and selected for editing (see next section) by using the buttons at the top of the screen. A number of filter options are also supplied, in the form of the controls at the bottom of the screen, which allow the temps to select only their own respondents or to prioritize unassigned respondents.

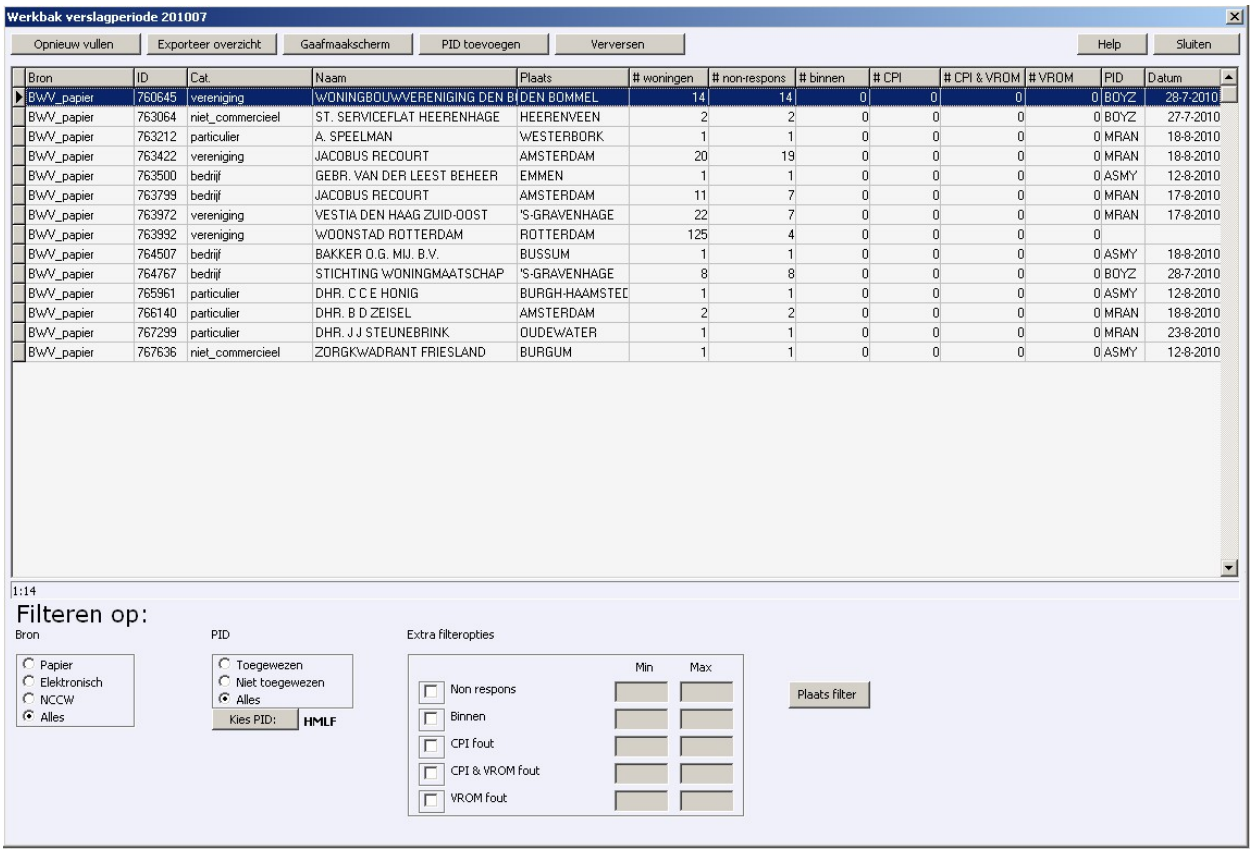

Figure 1: This figure shows the work assignment screen. The buttons at the top of the screen govern refreshing, refilling the list and allow the user to assign or open an entry in the main editing window. The controls at the bottom of the screen govern the different filtering options.

This screen was built in Maniplus. The lookup was found to be an efficient method of presenting large list of data. Moreover it could be integrated with other modules such as the main editing window. This allowed the building of an interactive system. The disadvantage of lookups is that they are difficult to

sort, especially on different categories, which meant that the different filtering options had to be implemented. Moreover, we had some problems with refreshing of the lookups, see next chapter.

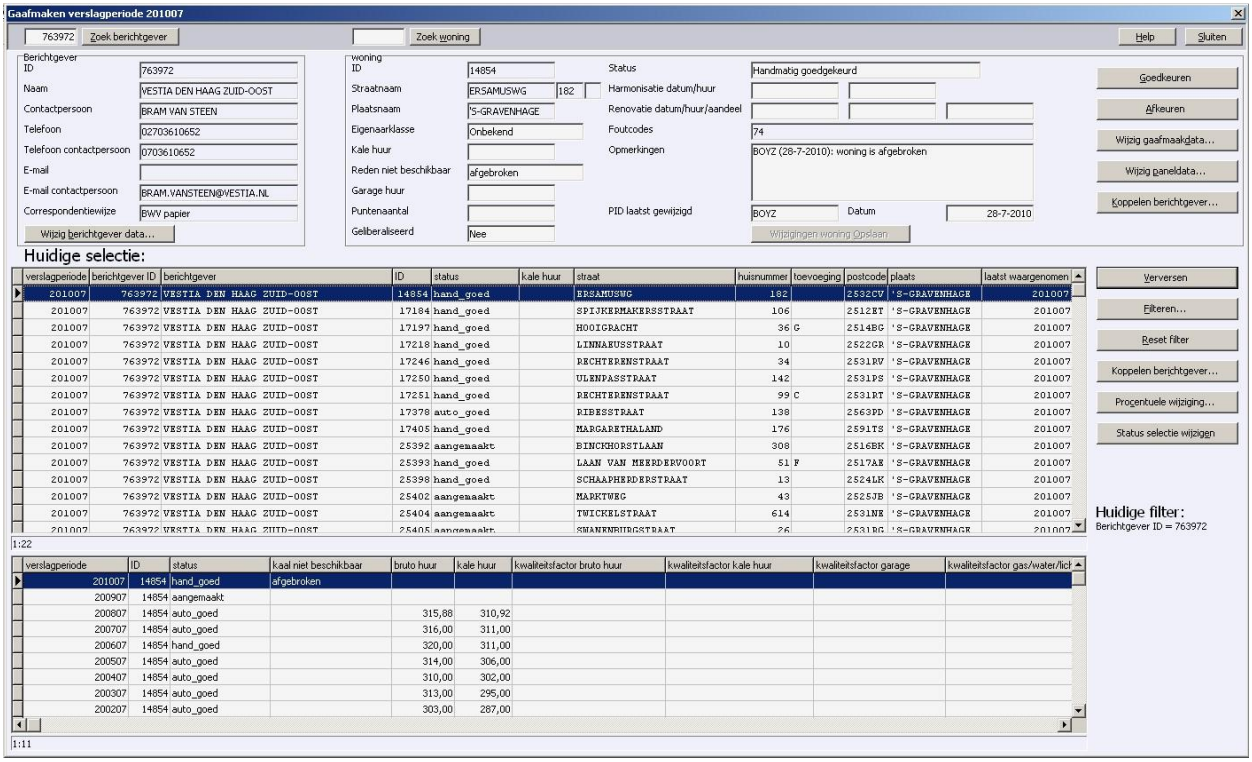

Figure 2: This figure shows the main editing window. This window was set up to edit groups of houses, typically all houses of one respondent. For privacy reasons the rental prices have been removed.

3.3.2 Main editing window

As explained above, the editing is simultaneously performed on all houses of one respondent. Therefore a screen was build which supports the editing of a group of houses, see figure 2. The upper lookup shows the selected group of houses, one of which can be selected. The lower lookup shows a time series of the currently selected house. The buttons to the right of the lookup govern all group-based actions (the first two bullets below). The controls at the top of the screen show detailed information on the currently selected house and the associated respondent. The buttons allow the user to edit some of the detailed information and open a DEP for the panel data, variable data and respondent data for the currently selected house.

In principle the group can be selected based on different criteria. Usually, all the houses of one respondent are selected. A number of actions can be performed in this screen:

- Changing variables of the entire group. Certain variables, such as the rental costs and also the status of a record (whether it is correct or not) may be changed for an entire group. This is governed by the buttons to the right of the lookups.
- Changing the filters to change the group size. As mentioned above, the typical group contains all houses of one respondent, but for larger respondents it often is convenient to further refine the filter, for instance to show only the records with mistakes in it. This is also governed by the buttons to the right of the lookups.
- Viewing and editing currently selected house: The details of the currently selected house are presented in the upper lookup. Double clicking one record of the list triggers a refreshing of the

lower lookup, giving a time series of that house over the last ten surveys. This also updates the controls at the top of the screen. The block on the left shows information about the respondent of the house, which facilitates the contacting of this respondent. The button below the information opens a DEP, which allows the editor to change the information of the respondent. The controls on the right give some of the panel and variable information of the house. The buttons allow the editor to change the status of the house and also to start the DEP's for these two data sources, which is treated in the next section.

3.3.3 Record based editing (DEP)

From the main editing window data entry programs (DEP's) may be launched to edit records in the three main database tables. Each of these DEP's is constructed from the corresponding data models. The layout of the DEP's has been defined in the layout sections which use modelib and menu files.

The panel and respondent data entry programs are relatively simple. They both contain only one double column with the names and values of the variables. Both sources use only one external data link, to the ZIP code register to check addresses which are entered. This check is also the only rule in these data models.

The variable data has a much more complicated layout, see figure 3. This data source contains most yearly changing variables and as such most editing is done in this screen. The DEP contains some 150 variables and another 50 auxiliary variables. It is spread out over two pages. To check for mistakes and inconsistencies approximately 150 rules have been defined.

The screen is set up in three parts. The buttons govern a number of functions, which are explained later on. The information blocks at the top of the screen contain external data from the other two main data sources. Below this information block the variable data is presented. The third and fourth columns are conditionally interchangeable, depending on whether the record represents a new (introduction) or a panel housing unit. Violations of the rules, which corresponds to errors or omissions in the data, are denoted in the usual manner using a red sign with a white cross in it.

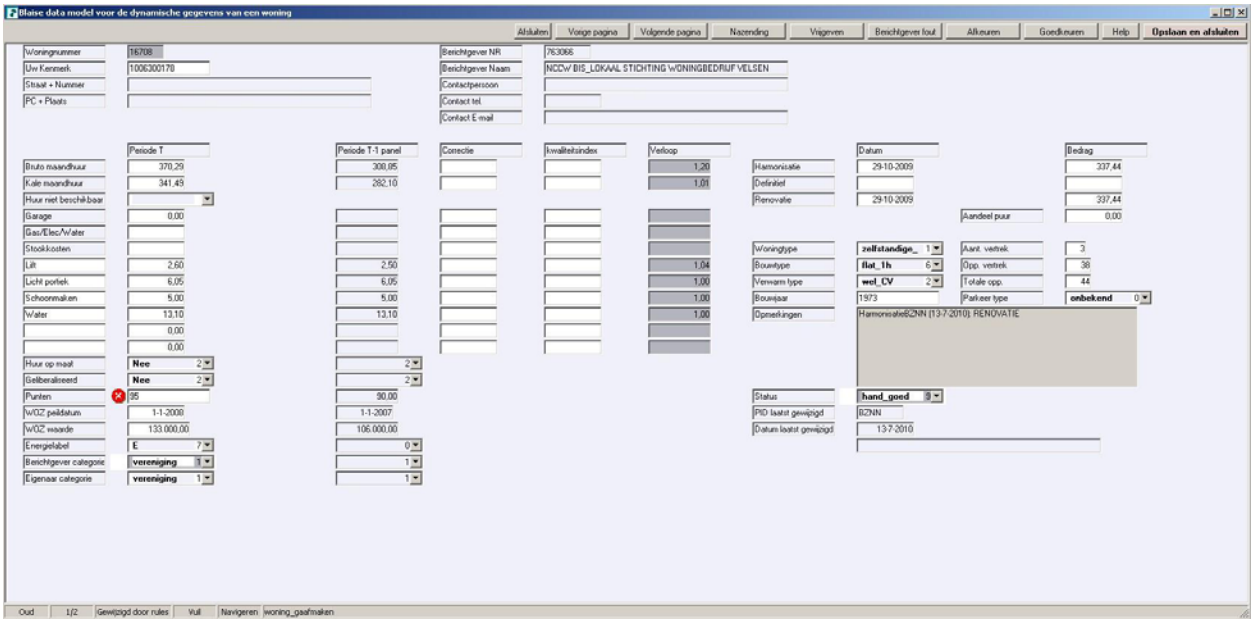

Figure 3: This figure shows the DEP for the variable data. For privacy reasons some of the contact information was removed.

All three DEP's contain a number of buttons defined in the menu file. The variable data has most buttons, which are mainly used for navigation and changing the status of the record (validate/reject). All three DEP's have a close button which starts a Maniplus script, which forces the user to place a comment if the record has been altered. In that case it also records who the user was and at what time the record was changed. This information is used to supervise the temps.

3.4 Calculation

The calculation was performed in Manipula. As will be explained in section 4.2, a view was made, which joined the three main data sources. This view is used as the input for the Manipula script. The script uses a sort section for aggregation, in combination with two manipulate sections. In the first section each record is assigned to one of the cells mentioned in chapter 2. Also, all records that should not participate in the calculation, either because they were manually rejected or because essential variables are blank, are removed from the calculation. Subsequently, all valid records are aggregated in the sort section, after which the average rental prices for each cell are computed in the second manipulate section.

The results of this calculation are joined to the weights per cell that are supplied by the Ministry. In additional scripts the data are then summed to different aggregates. These include aggregates over provinces, municipalities and a total aggregate for the entire country. For all aggregates the average prices are calculated both for the current and former year. These averages are divided to obtain the price indices for the aggregates. The aggregate for the whole country is called the housing rental price index and is part of the consumer price index. These results are also used for some of the export products, mentioned in section 3.2.

For the service charges a different approach is used. Firstly, some 700 categories of service charges have been recorded, which are grouped into 13 categories (12 real and 1 residual groups). Secondly, unweighted (arithmetic) means for the entire country are used to calculate the average values for the former and current year. The averages are then divided to obtain the prince index for the 13 categories.

3.5 Classification management

Within the survey two types of classifications are in use.

- A large number of categorical variables are used, half of which are yes/no variables. The other half usually have between five and ten categories. These variables are defined using Blaise types, the advantage of which is that Blaise consequently enforces these types in the DEP and in Manipula and Maniplus scripts. A disadvantage is that these types are hard-coded and changing them usually means changing all involved data models. That in turn means that all Manipula/Maniplus scripts and boi files that use these data models have to be recompiled too.
- Lists are also used extensively in the survey. For instance the 700 different service charges mentioned in the last section have to be categorised into 13 groups. These lists have to be maintained which is typically done in the following manner. The list is put in a table, which is accessed through a boi file. A screen is built in Maniplus in which a lookup presents the list. A user can select an item from the list and subsequently edit this item, either directly through a control and buttons or by opening a DEP to edit the entry.

4. Using the Blaise Suite

In the previous chapter we have presented a large system build in the Blaise Suite (Blaise, Manipula and Maniplus). To build this system we have used many parts of the Suite extensively and in some parts reached the boundaries of the Suite. In this chapter our experiences with the main components of the Suite are described.

4.1 Experiences using Blaise

Blaise was used to generate the DEP's for the main data sources as well as a few DEP's for the editing of classifications. In general our experiences with Blaise were positive. Defining data models is rather straightforward and self defined types are a very powerful tool. In particular the property of Blaise (and Maniplus) that values outside the type definition are automatically blocked is very convenient. For the DEP many automatic properties are set and many settings may be refined in the mode library (bml). Finally, the rules, which are integrated in the forms, are a powerful tool of checking the validity of the input. All these properties make Blaise a very powerful tool for building survey forms.

A less attractive property of Blaise is that the layout of the DEP is divided over two sections, the order of appearance is defined in the Rules, while their aspects are defined in the layout section. Moreover, four files are used to define the DEP, the bla, which contains the variables and the aforementioned sections, the modelib, which allows personalization of the variable and page makeup, the menu file (BMF), which allows personalization of the menu bar and the datamodelproperties file, which is used to define some of the properties of categorical variables. This separation of the code makes the development of a complex DEP somewhat of a struggle. In fact for highly customized and interactive designs Blaise can be as time-consuming to use as any normal programming language. This is especially the case if a reasonably small number of variables is used in a complicated design, as was the case in the variable data DEP.

Another problem with Blaise is that including external files into the data model can be rather complicated, a fact that is made more troublesome by the fact that Blaise has no debugger, as opposed to Manipula and Maniplus. Moreover, external tables, which are imported through a boi file into a DEP are completely put into memory even though only one of the external records is needed. For our project this meant that by opening one record in the variable data required opening the full panel and respondent data. This resulted in an unacceptable performance loss. The solution was to gather into a separate table all relevant external data for a record, data which is already opened in the main editing window. This table was then used as an external within the DEP, which improved the performance to an acceptable level.

Finally, if the data model is used in combination with a boi file, as we did, changing rules may invalidate the boi link. For this problem, however, there is a solution: reservechecks may be used as 'false' rules, but they require a strict bookkeeping.

4.2 Experiences using Manipula

Manipula was used in our project for calculating the index, for the data I/O and also for various smaller functions, mostly in cooperation with Maniplus. We found Manipula to be a versatile tool, which can access a wide range of file types, especially through the boi files. Moreover, the property that Manipula automatically links input and output variables of the same name, reduces the number of lines of code. The aggregation (sort section) is unusual as it combines ordering and aggregation of data, but it works and is rather fast.

However, we found that it had three main limitations, all associated with boi files. Firstly, it was found that joining data from two tables, either bdb or boi files, is very slow. For text files this problem doesn't exist, but for the sake of data integrity databases were preferred. In our system data joining was often needed, which made this a big problem. The solution was that we generated a number of views, which performed the necessary joining within the database and which we could access through new boi files. The resulting performance was much better.

The second limitation was that while reading data through a boi file has a decent speed, writing data to a database table is very slow. This problem was acute for the importing of data, which is performed daily and could take up to two hours. We haven't found a workaround to improve the performance. The import is now performed at night through a batch job.

The third problem was that, while the boi files allow version control on records, this capability makes database interaction even slower. This also affected the main editing window and the DEP's, all of which need to interact with the database. In the end, to maintain a decent performance for our system, version control was dropped. We decided to use daily version control by making a backup of the main data tables each night before the new import.

4.3 Experiences using Maniplus

Maniplus was used for defining the main menu and the different editing screens. A Manipula menu defines a menu bar onto the main window. It may adopt different forms, depending on the operational rights of the user. The advantages of using Maniplus are that it can communicate with Manipula and Blaise. Moreover it is relatively simple to set up. The main disadvantages are that it is somewhat limited and doesn't allow a full integration with other screens, like office applications, where the menu bar is always shown in all opened windows..

All screens, except for the DEP screens are in effect graphical user interfaces, which in Maniplus are called Dialogboxes. Four types of objects may be defined within a dialogbox, buttons, texts, controls and lookups. Buttons and texts are functional, but not special when compared to other GUI builders. Controls are ways of presenting and editing variables and as such may take different forms (radiobuttons, dropdowns, editfields). They are a robust way of accessing data and actually force the user to use values within the type definition. The lookup is a good way of presenting list. They automatically align the data and return the primary key of the selected record.

The main problems with Maniplus screens are that most properties of the elements are hardcoded. No relative positioning may be used and no wildcards may be used for the other properties such as colouring. This can make setting up and changing a window rather time consuming. Also, refreshing Maniplus windows is difficult. At the beginning of the project the Evaluateonchange setting did not work. A new build was rolled out by the Blaise team, fixing this problem. However, this property remains somewhat difficult in use and might slow down an application. We found that a good way of refreshing a window is performing a (dummy) writing action on the underlying databases. This triggers a refresh of the values of that source and it can be implemented in a refresh button.

5 Conclusions

In this project we have build a complex data processing tool, which uses the Blaise Suite in an unusual manner. While Blaise has been designed to perform data collections for surveys, we have elected to only use it for the editing of separately collected data. Thus, we did not use one of the primary strengths of Blaise, interactively collecting data, while using rules to force only correct responses. We did find that it is certainly possible to perform data editing in Blaise. However, given the size of our data set and the fact that we had to use relational databases for storing our data. As a consequence the performance of the system was a major issue. Moreover, the complexity of the editing process, especially the work distribution and the classification management meant that we had to build much functionality in Maniplus. It did support these functions, but for this purpose it is not a particularly powerful tool.

Even though we did have some problems using the Blaise Suite, the support team was always very helpful. We usually contacted them through the normal Blaise support e-mail and they usually came back to us within a day. This aided significantly in making this project a success.

Concluding we can say that if most of the data collection is performed in Blaise, the last bit of data editing can very well be done in this tool too. It is a very good package for quickly building those types of tools. However, if the data collection is not done in Blaise and the complexity of the required solution becomes higher, Blaise is not the ideal tool for further data processing.