Standardised Survey Management at Statistics Netherlands
An Example of the Object Approach

Frans Kerssemakers, Statistics Netherlands
Marien Lina, Statistics Netherlands
Lon Hofman, Statistics Netherlands

1. Introduction

Data collection for household surveys at Statistics Netherlands has since long been concentrated in one specialised department. With the arrival of laptop-computers for multi-survey use around 1990 this department got an interview administration system (IAS). IAS was centred round a central database for all CAPI surveys, storing samples and interviewers, with automated tools for supporting the allocation of cases to interviewers and providing them with material, keeping book of the cases sent back (by phone), checking on (piecework) payments and reporting on progress and results. For decentralised case management on the interviewer's laptop a dedicated system had been developed in the early nineties called LIPS (see Hofman and Keller, 1993), which took care of receiving and handling cases and preparing them for transmission by phone. It contained questions for the interviewer about how a case was dealt with, mainly concerning home visits and their results. Later on a standard Blaise questionnaire was used for the interviewer's reporting, next to one or more data models for the actual interviewing, all these embedded in a self-processing Manipula program, using Maniplus. Separate from CAPI the centralised CATI facility had its own Blaise-integrated management system for call scheduling, making appointments etc.

Top-down automation of case management for CAPI appeared to be a huge effort at the time. But once the system went ahead, it could serve most surveys till the end of the nineties. Then fighting non-response in particular necessitated more versatile designs such as mixed-mode and longitudinal ones. Although multi-wave surveys had originally been taken into account potentialities were insufficiently used and tested to get at a self-supporting system. In a situation where so many things had to be rearranged at once the system first had to establish itself for the then prevailing one-time surveys. Besides, being relatively inexperienced, it was not the right moment to develop a quality plan for orderly maintenance and expansion of the system to meet future growth. Later on, as a consequence of several reorganisations, adaptability began to suffer from staff turnover and loss of experience-based knowledge. As new researchers often were afraid to jeopardise ongoing data-collection, this concomitantly resulted in a gradual loss of control over the more managerial aspects of surveys. Automation specialists on the other hand often had to take decisions unwillingly, without knowing much of the research background, just to have the system operating.

The functioning of case management became rather critical around 2000 when a longitudinal design was mounted for the Labour Force Survey (LFS), using CAPI the first time and CATI for four subsequent waves. Because the old management system could not meet the needs of the new design, as was the case with several other surveys, a hybrid situation arose in which increasingly survey-specific systems were being developed while at the same time maintenance of the still indispensable old system was becoming more and more cumbersome. From this and under considerable time pressure an urgent demand for redesign arose. In accordance with the bureau's tradition a *standardised approach* for all surveys was again judged the best solution, but starting point this time should be the ability to deal with different waves and different modes in the same survey, so as to permit dedicated designs for specific demands. The question then became which elements were general enough to be standardised and to what extent flexibility should be built in.

Simultaneously plans were evolving for a new fieldwork organisation in which CAPI interviewers were to be regular employees instead of being hired as freelancer. It was intended to have more supervision and coaching and closer working together in local groups. One of the goals was to get better opportunities for *flexible solutions* particularly for interviewer deployment problems and non-response. This would increase the need for exchanging messages electronically and for moving around cases between interviewers. At the same time we wanted the central office to be well-informed on the progress of the fieldwork, enabling the researcher or field staff to adjust in time such things as the workloads assigned to the interviewers, the sampling plan or other parts of the design. It was therefore decided to channel all relevant information to a central database from which the latest info should be readily available to all interested parties. This means, for instance, that a local supervisor can be the one who decides to re-allocate an individual case. The transmission of the case from one interviewer to another, however, should still run through the central system.

2. A pragmatic approach to the new design

Thus we were faced with the necessity to redesign case management in the short term, making it fit for handling more complex mixed-mode and multi-wave designs, in a way to make it easily adaptable (regarding anticipated but partly uncertain future developments), all this within a standard framework.

Putting the information for case management on all surveys in one central database and having this information available for multipurpose use makes the development of standard definitions and routines indispensable. Sharing the same database implies the moulding of information on cases in a common set of variables or fields. Yet not every field has to be filled for every survey. So there is room for adding specific pieces of information for particular surveys. Also, at the front-end things may vary between surveys, for instance the reasons for non-response displayed to the interviewer, while at the back-end a common non-response variable may be derived for comparison.

We did not start from scratch. Having gained experience with case management for many years we had already developed ideas for improvement. We were also used to standard routines and question blocks for case handling by the interviewer and for standard tasks like listing or rostering of household members. So the new design could partly be based on the problems we had met and partly on tightening up existing practises. But we also had to enlarge the possibilities of the new system to make it ready for the foreseeable future. From this the following starting points were chosen:

- Surveys are to be considered multi-wave surveys and provisions should be made to pass on data from one round or period to the next. Also within a one-time survey there can be successive approaches by using different interviewers or modes (especially regarding non-response).
- It should be possible to use different modes concurrently (as respondents may express a preference for self-completion on paper or through Internet instead of being interviewed).
- Having a standard framework for all surveys while at the same time valuing certain statistically important distinctions can best be handled by explicitly defining these distinctions and have them being parameterised (specifying the characteristics in the context of the case management system).
- As far as the researcher is responsible for certain design decisions this party should also be in charge of the settings and options which effectuate these decisions.
- Not so much as a matter of principle but for practical reasons we give preference to only one data model for a survey. Above all, this should improve the researcher's access to the specs of the data model and through increased transparency simplify maintenance of ever changing questionnaires.

Access and maintenance have proven to be major issues. At the IBUC-conference in 1997 Statistics Netherlands presented an integrative perspective for a Continuous set of Surveys on the Quality of Life, called POLS (Heuvelmans et al, 1997). The characteristic feature of this design is the use of a common basic questionnaire and different data models for specific subjects (next to a separate data model for the interviewer's reporting on the case). Although representing quite a logical structure it soon came out that the dependencies between data models and the concomitant exchange of data put high demands on technical expertise and staffing. As we were already warned of at the same conference this increased 'the reliance on programmers for complex elements of questionnaires, and on non-Blaise software' (Manners and Deacon, 1997) and made the design vulnerable to staff turnover.

For researchers, who are not always Blaise experts after all, it generally became more difficult to get access to parts from other data models which might be relevant to them and, especially in a process of change, to lift out their own question modules for developing purposes.

3. Key elements of the design

3.1. The researcher back in charge

As a reaction to the difficulties encountered we have now chosen to avoid the concurrent use of several data models for the same interview case (leaving aside external data models that only contain a description of data to be read by the questionnaire program). This choice has been made possible by the increased possibilities of both Blaise and hardware. Using one program, the best place to account for the way a household has been approached and to report on the achieved results are usually *parallel blocks* (cf. the use of Non-response- and Appointment-blocks in CATI). The special function of such blocks can be displayed to the interviewer by putting them on tab sheets next to the actual interview part. In this way meta on handling the case can easily be built in the same data model used for interviewing, as we did in a block called *Admin*. Tracking down persons or households that appear to have moved takes place in a similar way in a block called *Moving*. Apart from solving technical and organisational problems, we believed it important to strengthen the researcher's control of parts that provide meta information on a case, in order to allow monitoring how the sample is finished off or studying how distinct approaches and various interviewers produce different results. Because these parts were previously applied in a standard way by specialists outside the project team, the researcher usually did not feel in control. Belonging to the same data model also makes the exchange of data with the interview part easier. This does not mean that every researcher will from now on fully understand what is in the source code. What makes a difference, however, is the notion that all specifications which may effect the outcomes are in the same data model that the researcher has access to, and are written in ordinary Blaise.

3.2. Data exchange with Blaise

Almost by definition every survey has its own distinct questionnaire. Concentrating data-entry entirely in one data model can therefore easily lead to very survey-specific solutions. Standardisation of case management on the other hand asks for standard blocks in the questionnaire to supply the central database with the demanded data. As far as data exchange is concerned, however, it is better to speak of *standard fields*. For the only thing the central database needs to know are the names and the types of the fields from which to read the input variables. The link between the Blaise database and, in our case, the SQL server database is made using the Blaise API as 'communication tool'. In case the standardised blocks are nested in other blocks also the names referring to these blocks are needed. But apart from the standard fields themselves, blocks like *Admin* or *Moving* may well differ between surveys and can be adapted or extended in many respects whenever needed.

3.3. Standard blocks

Whether different surveys also use the same *standard blocks*, i.e. blocks of the same type, is a matter of policy. We prefer to have common standard blocks for well-defined subjects, but at the same time may allow different surveys to use different parts of it. This forces us to explicitly define on which grounds a survey deserves a different treatment. For instance, we consider the listing of a household from scratch a more demanding task for the interviewer than checking an already existing list for changes. We will consequently use two different blocks within our standard module for household rostering. Standard then refers to common solutions for the same type of problems, such as using the same fields if possible (kinship, marital status etc.) or composing the same type of output for the central database e.g. a standard block with the gender, age, etc. of a person and whether this person is a target, contact or proxy (i.e. the row in the household table or matrix).

3.4. The survey definition

For a differential treatment the standard blocks should know what type of survey they are dealing with. Who can better specify this than the researcher in charge of the survey? It is therefore the researcher who should provide the case management system with specifications of the survey or, as we call it, the *survey definition*. Many characteristics of a survey have to be known to the central system beforehand. For the allocation of cases a fieldwork period has to be set and if there is data from a previous period to be sent to the laptops the system should know. But also whether it is allowed to skip one period of measurement, for instance. So it is rather obvious to define a survey beforehand at the level of the central system, including its features with respect to the way standard blocks are used.

A clear distinction then arises between the researcher who defines the survey and is in charge of the data model for the questionnaire and the IT-staff responsible for the proper functioning of the case management system given the definition of the survey and the relevant standard blocks in the questionnaire. Of course, there should also be an initial sample of cases, whereas an optimal deployment of interviewers has to be based on up-to-date and detailed information on their service contract, training, optimal and maximum workload and availability.

3.5. An object-based approach

Once a case is transmitted to a laptop we deliberately choose to conduct data-entry for this case as much as possible within the questionnaire, irrespective whether it concerns data from interviewing or meta data on the case provided by the interviewer. Where cases from different types of surveys should get a different treatment using the standard blocks, this is pre-specified in the survey definition. This implies that outside the questionnaire data-entry on the laptop should be reduced to a minimum.
As already mentioned in the introduction, Statistics Netherlands is also increasingly facing complex and changing designs. We therefore badly need a more flexible approach towards case handling, moving individual cases from one interviewer or mode to another (instead of assigning more static workloads as we were used to). Questionnaires too have to be better geared to changing demands, for instance when they are used to provide information on gaps in other sources, like registers (the use of which is in our case rapidly increasing). This is where the object-based approach comes in.

As the principles of an object-based approach to case management are extensively dealt with elsewhere (Gray, 1995) it suffices to say that every case can be represented by an independent object. The power we want to use rests on the 'ability to hide internal details from systems that don't need to know them' (Gray and Anderson, 1996). Specific details, like the data collected, are fully encapsulated. This makes it possible to abstract from survey-specific content. An object needs a method that tells the system what to do when the object is activated. From this perspective sample cases can be considered as objects to start questionnaires (and the latter in their turn as objects to install questionnaires). There should also be a 'caption' that makes visible what the interviewer needs to handle the object before it is being activated (e.g. address info and the name of the survey).

3.6. The new LIPS

The assistance our newly developed case management system gives to interviewers for handling objects on the laptop is largely based on the Casebook system from UK's Office for National Statistics. ONS also offered great help with adapting the system to our fieldwork organisation. Their object-based approach shaped our view on how the case management processes should be re-arranged.

Between receiving a case and starting the questionnaire the interviewer should be assisted in adding the object to the workload and keeping an eye on the progress made in dealing with the workload, facilitated by various ways for conveniently arranging the cases from different surveys. When the interviewer wants to take a look at a particular case things like the survey and period and address of a household or name of a person should be displayed. From this screen it should also be possible to start up the questionnaire.

A mechanism was built to get direct access to the parallel blocks *Admin* and *Moving* (by pushing the corresponding buttons). For it may be confusing to enter the interview part of the questionnaire form if one just wishes to book some visits (in *Admin*). We also created a facility to display the content of a special field in *Admin*, which may be used for a remark (comparable to the display of a selected field on the dial screen in CATI-management). Finally the screen is used to finish cases. When choosing this option an action is started by which a case will be marked 'dirty' if not completed the proper way. A message will then be displayed indicating what piece of information is still missing, e.g. the final result has not yet been reported (by the interviewer). The checks on completeness, however, are executed in the questionnaire. There the error message is computed and stored as a string. The action to finish a case only reads from the questionnaire whether *'ready = yes'* and, if not, the error message. A reported error can be overruled and the 'dirty' case be moved to the out-tray by the interviewer, putting it ready for transmission to the central office. There its erratic status will be noticed and special attention will be given to the interviewer and the case, as being a possible candidate for re-allocation. Together with the case a progress report is sent to the central office concerning the workload of the interviewer still to be finished. In a similar way the report is based on reading a field in *Admin* in which the interviewer has to report (in a very simple way) on the intermediate results achieved.

A lot of actions are taken in the new LIPS, partially induced by the interviewer making choices what to do next. However, in the new LIPS the interviewer is not answering questions, not even whether a given address can be found, or whether anyone is living there. All this is reported in the questionnaire. The only exception is for extra households that may be found at a given address. This only applies to a first-time approach when we use a sample of addresses to get at a sample of households. Under certain conditions the interviewer can put on stage an extra household, up to three households in total. If permitted to do so a special button will appear on the start-up screen. For case management, however, the assigned case still remains the complete address. To be sent back as 'clean' all households within it must be finished. But if cases should be re-approached in a subsequent period then the households are the sample elements to be followed (as we generally do not use area sampling). The central system will then replace the initial address by one case for each newly identified household.

3.7. Panel management on a flow basis

Once a survey has been defined the system should know what to do in each instance for every case. In longitudinal or multi-wave surveys we strongly believe that the central system should automatically select cases for the next wave or period on the basis of the Blaise data received. That is to say, the Blaise database should preferably contain all necessary information for the system to decide on this issue. Generally, when a refusal has been derived in the questionnaire, reading this result from a standard field will automatically cause the non-continuation of a case, although it may lead to some other action like selection for refusal conversion. In addition to this standard mechanism a researcher may have specific reasons for removing a case from a panel group, for instance, if he wants a complete data set on all targeted persons in a household. Very often these kinds of conditions can be pre-specified by the researcher in the data model and the possible blocking code put ready for reading by the system. We also want data inconsistencies to be solved as much as possible by the interviewer, making conflicting information visible through the questionnaire. From our experience there is little need for data-entry outside the questionnaire form, even with complex panel studies (except for change of address cards). So with panel surveys our central system serves to pass on cases to the next period, mostly together with input data from the previous period that has to be checked for changes.

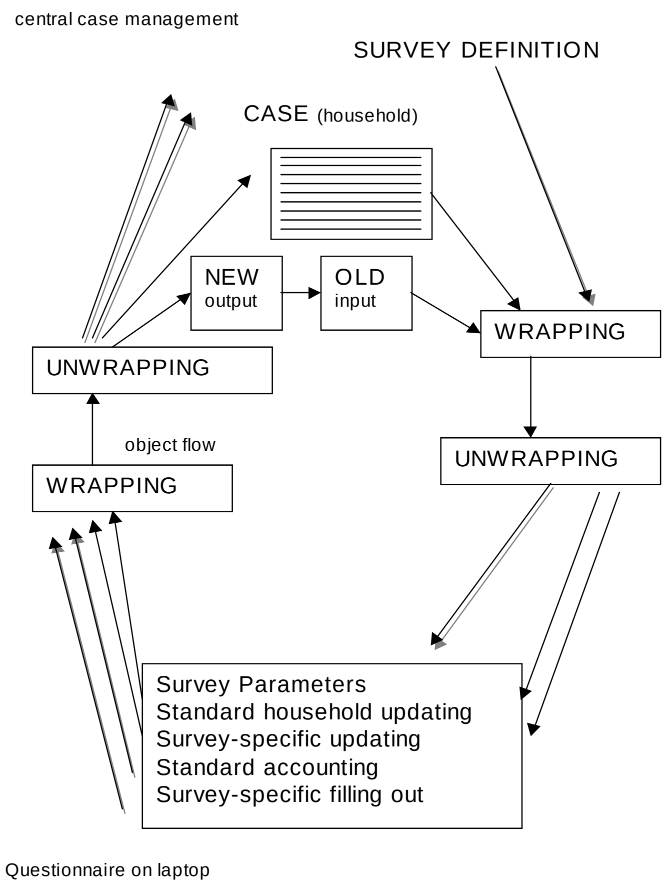
Because our cases are usually households or persons within households we choose to represent each case by way of a matrix in which the rows are used for the household members. This conveniently fits in with the standard matrix we use in the interview form for listing the household members. In terms of variables or fields the row is identical to the block representing a person in the interview form. A standard block for each person can therefore easily be copied both ways. For this we use the Blaise

API. When it is found during fieldwork that a household has split up and all separate parts have to be approached again then the rows only have to be divided to form new cases. Within each row the status of the corresponding person with regard to the survey can be specified. For instance, a certain child may be the target person, the parents the proxy-persons and one of them may be the one to whom the advance letter should be sent. Linked to the matrix is a survey-independent id-number and info on the name, address or telephone number. In the near future the matrix will probably be filled beforehand by combining information on persons and where they live from the Dutch Population Register. Then the interviewer only has to check this information and adjust it if necessary.

Finally, there will often be survey-specific information to pass on. Again, we want the data to be preloaded in the questionnaire form itself. The required Blaise data set should be in a block. This block, although specific to each survey, should be referred to by a standard name, NEW, to indicate the updated output of the panel wave. The central system will look for the name referring to the block. It does not process the data in any way. It just copies the variables in it to an identical block in the data model for the next period (using the Blaise API, and possibly a conversion tool in case the data model has changed). This block, named OLD, is of the same type as NEW. Whereas NEW is meant to deliver the data for the next period, OLD contains the input data for the new panel wave. So the output is automatically made input, for each case to be re-approached.

Generally conceived, as sketched out in figure 1, we discern two circular motions in case management: One for the updating of standard fields and the other for the updating of survey-specific fields. Besides, every fieldwork period has its own survey definition (although usually not changing very much). This definition is made available to the questionnaire by the new LIPS in the form of external data (which after reading serve as survey parameters). And, of course, there are the one-time outcomes, part of which is used for monitoring and evaluation of the case handling.

Figure 1. Flow of panel data



4. The role of the SQL server database

To reconcile differences between a more rigid standard approach and more flexibility we first had to agree on a minimal set of functional requirements. To the functionality which was already fully realised in the old system the following major extensions were added:

- Comparability on (non-)response results across surveys on the base of a common variable with standardised categories (the score on which is derived from questions displayed to the interviewer on the laptop, which may differ from survey to survey)
- Automatic computation of various standard response measures and the production of standard reports for different purposes (statistical, accounting, checking on performance etc.)
- Storage and accumulation of collected data on reasons for non-participation (refusal, no contact, no opportunity etc.) to be used for non-response analysis
- Quick reallocation within the same fieldwork period of individual cases from one interviewer to another, including the transmission of Blaise-records already partly loaded with data from previous periods or records containing external data
- Easy to handle facilities for shifting a case from one mode to another, e.g. from CAPI to CATI to increase the probability of contacting, and if contacted, e.g. from CATI to CAPI to complete the larger part of the questionnaire; or from interviewing to self-administering a questionnaire form, offering varying degrees of support
- Registering the use of sub-forms in a CAPI- or CATI-survey, such as a self-administered question form, to be collected by the interviewer or sent back directly to the central office

For the central system to know how to handle a case we had to define a number of general entities common to every survey. These include:

- The identification of the survey and the particular fieldwork period for which cases are selected
- The version of the questionnaire to be used when data collection on a case is started
- The definition of the survey, given a standard approach, to allow for a differential treatment; for instance, when handling a first-time sample of addresses versus a persons' sample
- The different tasks within a survey to be discerned, like allocation, re-allocation or changing the mode of data collection, special coding, processing the incoming data and possibly changed address info, putting the eligible cases ready for the next period, wrapping cases in objects, at the same time keeping up-to-date the stage of the process where every case is in
- A standard set of categories to represent the final result and the home visits or phone calls made
- The identification of the case and mostly also of the persons making up a case (see section 3.7)

5. The new LIPS: some technicalities

As mentioned in section 3.6 the design of the new LIPS is based on the ONS object approach. We decided to start the development of the new LIPS from scratch just using the ideas from ONS. We invited one expert to visit us for one week. The idea was to try to build a prototype to prove an object-based LIPS satisfying the needs of Statistics Netherlands is possible. The prototype could form the basis for a final implementation. After some discussion and based on the experience with the old LIPS system and the experience from ONS we made some important implementation decisions:

- We decided to use Maniplus as the main tools to build the prototype. The only alternative was Microsoft Visual Basic in combination with the Blaise API. But to keep things simple, both development and laptop installation, we decided to prototype in Maniplus.
- We decided to use only ‘modern’, Windows based technology when we have to go outside Maniplus because of missing functionality. So no use of batch-files, 16-bit DOS tools (like PKZIP) and shelling-out to the operating system (for instance executing a MKDIR command to create a new folder). Experience with the existing system indicated that using such old-fashioned ‘DOS-based’ solutions can introduce instabilities and will lead to less over-all control. A good choice as Maniplus-extender proved to be VB-script. Within VB-script it is possible to use numerous available ActiveX objects to handle specialised functionality. For instance for creating

ZIP-files we selected an ActiveX archiving tool. With VB-script in combination with the ActiveX archiving tool it is possible to have total control over the archiving process.

- We decided to use password protected ZIP-files as physical implementation of the objects. These ZIP-files always contain at least a file called object.ini. This file has a simple so-called ini-structure. Ini-files are very flexible in use and can easily be extended with new sections and new keys without causing compatibility problems. All information on how the object has to behave is present in the ini-file, but the ini-file can also be used by the object to store extra information, like progress information on an assignment of a specific address.
- We decided to make the system completely data model independent. So all object methods implemented are independent of data models used for interviewing. Compared to the old system this means for instance that no specialised, data model dependent Manipula set-ups are used to extract data from the interview or to pre-load data. To make it possible to use data from an interview in LIPS, a way had to be found to extract data from a Blaise data file without knowing the data model while preparing the Maniplus LIPS set-ups. The Manipula function GETVALUE was added to the Blaise system to achieve this goal.
  Because the system does not know the data model, it is also not possible to create a Blaise form run-time in the field by the Maniplus LIPS application. A form can in the new LIPS only be created by the data entry program (by passing the correct command line parameters to the EDIT function in the Maniplus setup) or before hand at headquarters.
  So in the new LIPS all data required by the interview need to be included in the object. There are roughly two possibilities:
  - The data is stored in the object.ini file as ASCII data and made available by LIPS to the interview session as external ASCII file. For instance all survey definition related data is passed on this way.
  - The data is preloaded in a Blaise form at headquarters and included as file inside the object. For instance when panel related data is present a form is created at headquarters.
- In case of an address sample multiple forms (each corresponding with one household) can be required at one address. In this case the Blaise form is created by passing a unique identifier as key via the command line to the interview session. In case the sample contains only identified elements (for instance a person sample based on the Population Register or a panel follow-up) a form is created at Statistics Netherlands with the correct data using the Blaise API.
- As described in previous sections all administrative data collected in the field is part of a standardised parallel block of the data model. To allow the interviewer easy access to this parallel block from within LIPS a new command line option has been added to the data entry program. By mentioning the correct parallel name on the command line the data entry session starts with the focus on that parallel. The name of the parallel to be started is stored in the assignment object.

5.1. Working with the new LIPS

The new LIPS-application uses three so-called trays: The in-tray, the work-tray and the out-tray.

The in-tray

In general objects mailed to the laptop arrive in the in-tray. (LIPS can also handle so-called ‘Auto activate’ objects, for instance for installing new software on the laptop. We will not discuss this any further). The interviewer sees in the in-tray a short description of the contents of each object, for instance ‘New assignments for the Labour Force Survey, period June, 2001’. The only action available for the interviewer in the in-tray is ‘Accept’. This action moves the object from the in-tray to the work-tray.

The work-tray

The work-tray contains all objects the interviewer can activate. A number of different object-types are available (and new object-types can easily be added to the system). The most important object-types

(from the interviewer perspective) are: the ‘Install questionnaire’ object, the ‘Assignments’ object and the ‘Individual assignment’ object. What happens when the object is activated depends on the object-type. For instance, activating the object with the new assignments for the LFS will result in extracting the individual LFS assignments to the work-tray and a self-destruct of the LFS assignments object. Extracting the individual assignments is only possible when the correct LFS questionnaire has been installed on the laptop. In case this questionnaire has not been installed an appropriate error message is displayed.
If the object allows it, the interviewer can move the object from the work-tray to the out-tray. For instance, an ‘Install questionnaire’ object can not be moved, but ‘Individual assignment’ objects can be moved to the out-tray.
The work-tray shows for each object a short description plus the contents of a special entry in the *object.ini* file. This entry can be used for instance to show the progress made with the assignment (for instance ‘No contact’). The contents of the work tray can be sorted based on the contents of some entries in the *object.ini* file. The display order of the objects on the screen is always based on the object type. Because of this objects of the same type are grouped on the screen.

The out-tray

The out-tray contains all objects that need to be mailed to headquarters. The interviewer can move an object back from the out-tray to the work-tray. During data communication all objects present in the out-tray are mailed and, when this is successful, removed from the out-tray. During data communication administrative information is extracted from each object present on the laptop (so from all trays!) and mailed to headquarters. This information can be used to monitor the progress in the fieldwork. The content of this information depends on the object type.

Activating the individual assignment object

When this object type is activated the interview method is activated. In general this is another Maniplus setup that is responsible for handling the assignment. The main functionality offered is:

- Detail information on the assignment (like the full survey name, survey period, full address information, etc)
- The possibility to spawn extra households (if applicable)
- The possibility to start the interview
- The possibility to start the accounting parallels directly
- The possibility to check whether the assignment is completed. This is done by inspecting the contents of one field in the interview form that summarises the result based on collected interview data and data filled out in the accounting parallel block.

6. Some API technology

On the other side of the interview system, records filled with interview data are coming in from CATI or CAPI and must be linked to the newly developed central interview administration system. The management software that handles this is built with the Blaise API. The organisational scheme is that packages are entering the system, containing interview data and a status file for each case that had been put out earlier by LIPS. The system determines the state the received data are in, discriminating between *response*, *partial response* and “*not-opened-at-all*”. After that the system takes several decisions, based on the results in the interview record and the status that is in the guiding object file. It would be too much to explain all the details. For example, the system decides to (whether or not) put the record through to the next wave, to a specific interview mode, to update address information in the administration, to create a new record if only one person of a household moves to another place, and many things more. Only cases with unexpected results, such as data corruption, are copied as received to a reserved place where manual intervention is required. By applying certain standard blocks the system can read values in standard fields to recognise the subsequent steps for each case. The entire system has been developed in Microsoft Visual Basic using the Blaise API. The processing time of the

Blaise API approach may be slightly more compared to the old Manipula approach, but the advantage is that there is no need to re-prepare Manipula set-ups for different data models. When data models change (which happens on a regular basis) then the system does not need to be maintained as long as the called fields in the standard blocks remain the same.

The API makes it possible to read information from different data models with the same API application. In this way the API puts aside the problem of data models being incompatible, as long as the data models use a uniform block structure and fields that are used by the Visual Basic application.

7. Conclusions

It is too early to draw final conclusions because the system is not yet in full production. It will go life in January 2002. But, during the implementation and testing of the new system we already noticed that the approach that has been chosen is flexible and it was very easy to make required changes. We are confident that we have chosen the right approach for the years to come.

References

Gray, J and Anderson, A. The data pipeline: Processing survey data on a flow basis, in: *Survey and Statistical Computing 1996. Proceedings of the Second ASC conference*, London, United Kingdom, 1996

Gray, J. An object-based approach to the handling of survey data, in: *Essays on Blaise 1995* Statistics Finland, Helsinki, Finland, 1995

Heuvelmans, F, Kerssemakers, F and Winkels, J. Integrating Surveys with Blaise III, in *Proceedings of the Fourth International Blaise Users Conference* INSEE, Paris, 1997

Hofman, L and Keller, W. Design and management of Computer Assisted Interviews in the Netherlands, in: *Journal of Official Statistics*, 9, 1993

Manners, T and Deacon, K. An integrated household Survey in the UK, in: *Proceedings of the Fourth International Blaise Users Conference* INSEE, Paris, 1997