Using Blaise to Automate the Clerical Edit and Telephone Follow-up Component of the American Community Survey

by Barbara N. Diskin and Kenneth P. Stulik, United States Bureau of the Census

Summary

The American Community Survey (ACS) conducted by the U.S. Bureau of the Census will ultimately be the largest ongoing survey in the U.S. By 2003 we will sample 3 million households a year to collect housing and population data every month. The ACS is a mail out-mail back survey. Since roughly one-third of all mail returns lack enough critical data, a telephone follow-up (TFU) operation exists to collect this data. The increasing size of the ACS made it imperative to automate the formerly manual TFU step. When we evaluated software that could support the automation effort, Blaise from Statistics Netherlands was the obvious choice due to its features and flexibility.

ACS Basics

The goal of the ACS is to provide current demographic and housing data for small geographic areas. The questionnaire content is almost identical to that of the Decennial Census long form, with a few additional questions on items such as food stamps. The ACS went into production in 1996 in four counties. The initial sampling rate was 15 percent the first year with a decrease in subsequent years.

The Data Collection Cycle

The ACS follows a 3-month cycle. During the first month the sampled addresses receive an initial questionnaire by mail. If they fail to respond, they receive a second questionnaire by mail. If they fail to return the mailed questionnaires, during the second month an interviewer attempts to contact them by telephone. If that too is unsuccessful, an interviewer actually visits a sample of the non-respondents and collects the data on a laptop computer.

Initial Approach to Clerical Edit and Telephone Follow-up

The returned paper questionnaires are often missing data or contain inconsistent data because some of the concepts are difficult for respondents to understand and they make mistakes in completing the forms. For the first 3 years of the survey, clerks manually reviewed each returned paper form to detect deficient forms for telephone follow-up. This was a labor-intensive operation that was itself error-prone because of the difficulty humans have in following algorithms in exactly the same way.

Rationale for Automating TFU

The survey, which started out with 80,000 households in 4 counties the first year, expanded greatly for 2000. It now encompasses 1,239 counties with a sample of 864,000 households. By 1999 it had become apparent that the clerical edit of the mail questionnaires with the accompanying telephone follow-up needed automation to make the operation more consistent and to increase efficiency.

Determining Cases that Go to TFU

We began the automation process by translating the clerical checking into a series of algorithms that would result in either a pass or fail status. If the respondent provides sufficiently consistent data and answers key questions, such as age, for all persons in the household, the questionnaire passes the edit. If not, the questionnaire fails the edit. If there is at least one telephone number for an address with a failed questionnaire, the questionnaire moves to telephone follow-up.

Software Choice and Implementation Strategy

The software for the telephone follow-up operation required the ability to start with reported data and build upon those data. Only Blaise from Statistics Netherlands could do this. With a 9-month window for development and testing, the Census Bureau enlisted the help of Westat for training and consultation. Everyone realized that this was an enormous undertaking for such a short time. The resulting partnership provided the necessary support during these labor-intensive months. Westat had the added advantage of being the U.S. representative for the Blaise software package.

TFU Processing Environment

The TFU processing environment covers several different platforms in two different geographic locations across a wide-area network. Return mail forms are checked into our document control system at Jeffersonville, Indiana, then immediately keyed on a VAX system. Raw keyed data are transferred daily to a Sun server in Suitland, Maryland, where they are processed using SAS to determine the pass/fail cases for TFU. ASCII data representing the failed cases are then daily transferred to a Windows NT server back in Jeffersonville, where they are loaded by automated routines into Blaise format with the ASCIIRELATIONAL method. Case data are then handled by the TFU interviewers using Blaise on NT client stations until the data are resolved in some fashion. Data are output from Blaise daily and ported back to the Sun server in Suitland, where they are further processed and analyzed.

The Blaise instrument

Although not large in scope compared to some CATI instruments, the Data Entry Program (DEP) instrument for ACS TFU is complicated because of a number of non-standard features we felt necessary to build into it. First and foremost, the instrument accepts existing data from the keyed mail forms, a feature that not every CATI software system has. We had to write complicated routines in Manipula for the read-in and read-out of the questionnaire and case header data.

The existence of pre-loaded data in the cases gave us another interesting challenge. It would not do to simply plug through screen after screen in search of an error or omission. We called upon Blaise's navigational flexibility to jump from error to error, often bypassing large sections of the instrument. Interviewers have a choice of "Interview" mode or "Edit" mode to accomplish this. In Interview Mode, an interviewer can efficiently maneuver through sections of the instrument, which may be all or mostly blank. In Edit Mode, the interviewer can use the Show All Errors feature to jump from field to field, often bypassing several screens in the process. Regardless of mode, we were also able to program Blaise in such a way so as to make the majority of the person sections display in a matrix-style format, thus giving our interviewers the flexibility to navigate across persons or across topics (see Figure 1).

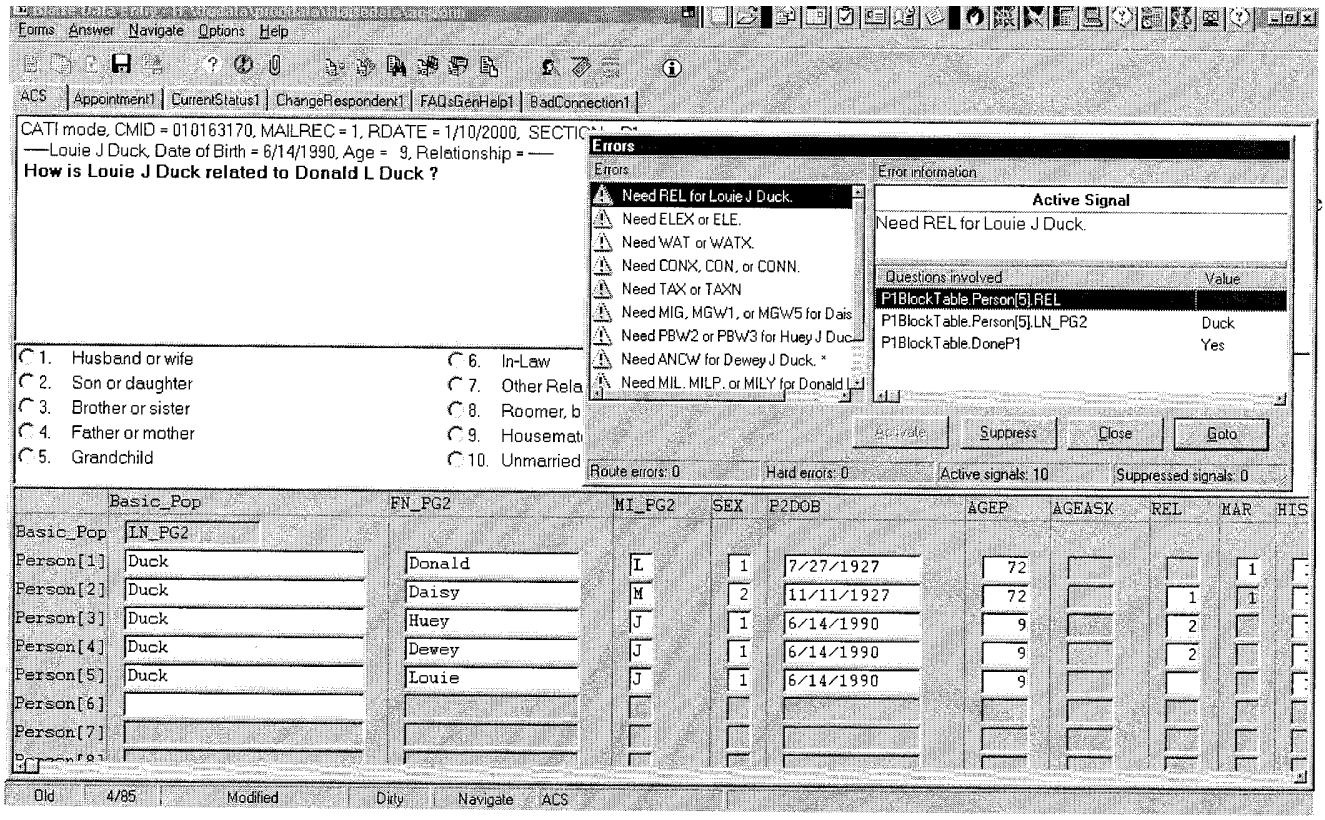

Figure 1: Screen shot of ACS Blaise instrument showing matrix-style format, parallel block tabs, interview (CATI) mode, Show All Errors dialogue, and more. All names depicted are fictitious for confidentiality purposes.

Another feature we built into the instrument was our custom control system, which we were able to effectively integrate with the Blaise call scheduler (BtMana). The challenge was to build in our supplementary CATI control codes into the Blaise framework while still taking advantage of the simple and efficient daybatch concept of Blaise. This was accomplished in

large part by using a series of external lookup files that allowed us to designate our own control codes (primarily case outcome codes and future action queues). There are literally hundreds of combinations of codes and subsequent outcomes based on 6 different case characteristics. The lookup table takes into account these characteristics and assigns our own set of four codes plus the requisite Blaise treatment and routeback. In addition to enjoying all of the features of Blaise CATI management, this approach allowed us to capture all of the internal codes we needed for our own traditional operations analysis while having a coding system that could readily translate to other CATI operations in the Census Bureau.

Yet another feature we wanted to have was a current summary of the pass/fail criteria within the instrument. This allows interviewers to preview the case by looking at summary-level information and rapidly determine whether they need to correct housing errors, correct population errors, add new persons to the rosters, or any combination thereof (see Figure 2). In addition, interviewers can get an up-to-date status of the interview. This could allow them to terminate an interview prematurely, in the event they are talking to a hostile or very reluctant respondent, if the error score is low but not perfect. The parallel block feature of Blaise facilitated this functionality by allowing us to create a separate screen accessible by a keystroke or mouse click from anywhere in the instrument.

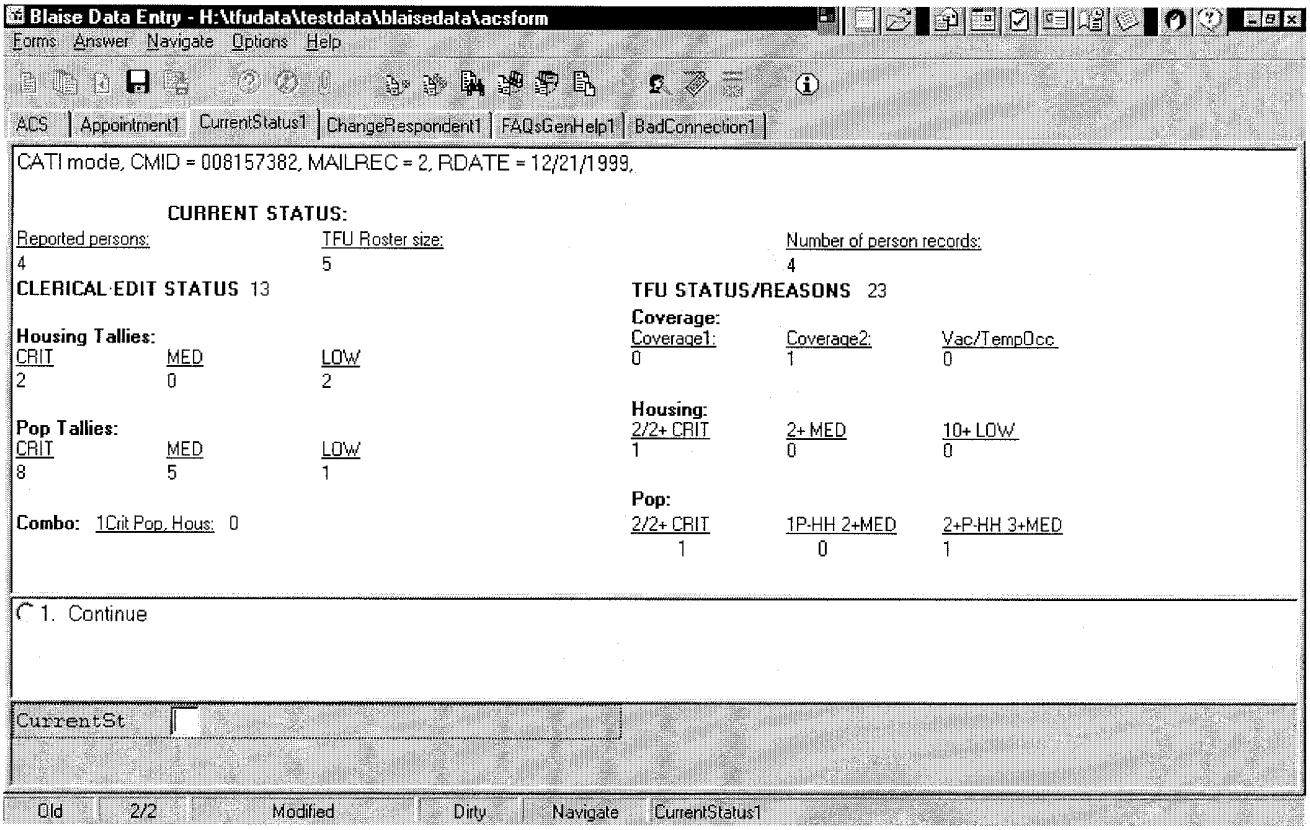

Figure 2: Screen shot of ACS Blaise instrument Current Status parallel block, showing case-level summary of error counts and types.

In designing the TFU operation, we realized we would need to use some of the peripheral components supplied with Blaise in order to make the whole process more efficient. One such feature involved the use of audit trails to track case-level activity

default audit trail dynamic link library (DLL), we decided that its functionality was insufficient for our needs. At our request, Westat modified the DLL such that any activity to a given case creates or appends to an audit trail file (which has as its name

directory and call up the audit trail file with the case ID as its file name.

Another desirable feature external to DEP and
allow the supervisors to have control over the future action of any case, but this would primarily be used to handle special cases that required a human decision.
which call for supervisory decision, obtain summary or detail information about the case, and affect a few key fields such as routeback and future action queue. A logging and reporting feature associated with this utility makes it easier for us to identify

account for these automatically.

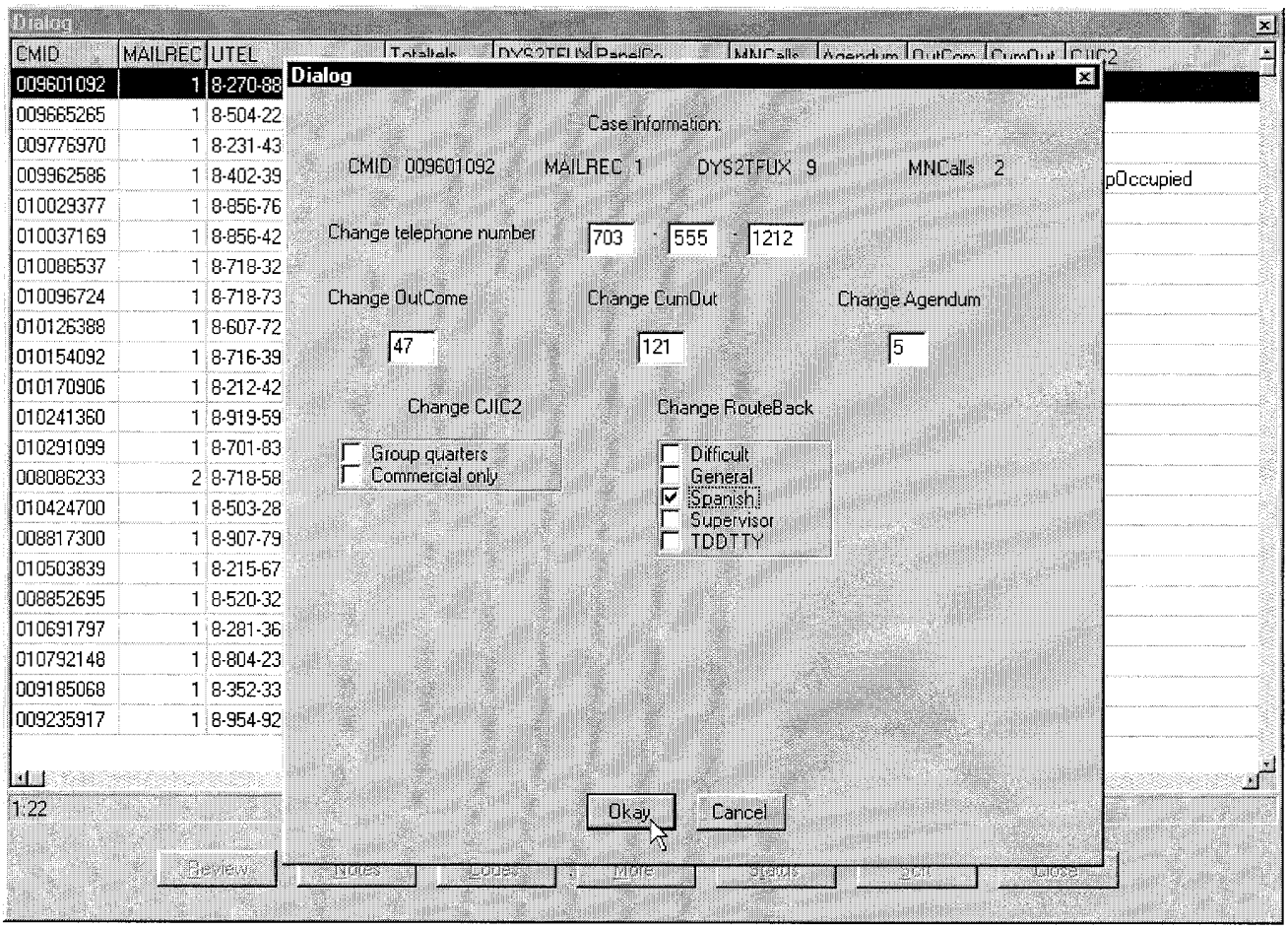

Figure 3: Screen shot of the Maniplus "Supervisory Hold" utility, allowing supervisors case-level control over disposition of individual forms.

One final feature we needed which was not part of the Blaise suite was the ability to monitor an interviewer's computer screen remotely for quality assurance purposes. We chose Stac Software's Reach Out Enterprise because of a recommendation from National Agricultural Statistics Service that it provided excellent functionality as well as peaceful coexistence with Blaise. Today, our supervisors use it continually and it has never been known to cause any problems with the ACS Blaise system.

Moving TFU into Production

Due to the challenges presented by the tight time frame from March 1998 to December 1998, deployment of the Blaise instrument was a harried effort. But a number of things worked in our favor, and the end result was a success. First, the ease of programming in Blaise allowed for the rapid development of the initial instrument, resulting in more time than expected to work out the major problems. While the instrument that was initially deployed in production in December had a host of minor problems, none of them was major and they tended only to affect the navigation of the instrument or handling of cases.

Another factor working in our favor was the support of experienced Blaise consultants at Westat. With their help, we designed a sound instrument, taking advantage of the many features of Blaise described in the previous section. Westat provided professional on-site Blaise training, in addition to periodic classes at their training facility, and this helped to prepare the Census employees that programmed the ACS Blaise instrument.

The deployment of the server and workstations hardware and their integration with the existing network infrastructure in Jeffersonville went off on time and with little trouble. From the first day that we deployed the instrument there, we received excellent technical support and did not need to divert a significant amount of attention to hardware or network issues. There was, however, a learning curve regarding Blaise's interaction with NT with respect to file permissions and user profiles. We also experienced some problems with profiles which at times caused the interviewers difficulty in logging into the correct DEP session (or any session at all for that matter). But in due time we learned the configuration that Blaise needed, and we tracked down the problems with the profiles, so we now enjoy smooth operations from that standpoint.

Of course, our interviewers needed to be trained for their role in the automated TFU. But since they were already familiar with interviewing techniques and were also familiar with the questionnaire, they primarily needed to be trained in using Blaise. While some of the interviewers had little computer experience, others had experience with CATI software in general. And since they were all very motivated individuals, training and ramp-up took less time than we expected.

Problems We Encountered and their Resolution

In deploying an instrument of this magnitude, it is not unusual to encounter technical or software problems, especially when the software is new and non-traditional. So it came as no surprise to us that there were some glitches to iron out during and after the project's initial deployment.

There were a number of bugs in DEP build 269 with which we needed to contend. Some of these bugs, such as the various Access Violations and Sharing Violations that our interviewers received, were easily solvable by changing our NT configuration and allowing read/write (instead of read only) access to key meta-data files. There were also many instances of DEP sessions simply hanging indefinitely, and those likewise were cured by setting the Opportunistic Locking NT setting to OFF. Another DEP bug we encountered was the creation of an invalid key and a null record when routinely entering a form. The problem was, amazingly, found to be triggered by entering the form using the <Enter> key as opposed to clicking 'OK' with the mouse. Statistics Netherlands was able to quickly track down the problem and issue a new DEP build.

We encountered other problems while perfecting our daily routines to load and unload data. The nature of the input data required a complex read-in routine designed to import multiple hierarchical flat files into a Blaise database containing arrayed records of sub-blocks and embedded blocks. It also required us to convert Don't Know/Refused responses into Blaise DK/RF values, and handle field-level comments. While the Blaise ASCIIRELATIONAL read-in method would facilitate this, the need to constantly recompile data models and Manipula routines, coupled with difficulty in handling slight changes to file layouts and handling field-level comments, made the initial development of the read-in/read-out routines cumbersome.

Another minor problem in the daily routine involved a Manipula RUN command occasionally not executing properly, which would bring the whole routine to a stop. Simply resetting and resubmitting the job would usually fix the problem for that day. Eventually, we thought to alter the routine to have the calling batch job execute the DOS command, which fixed the problem. While it was theorized this problem was a result of multiple RUN commands executing simultaneously, causing file locks which prevented multiple simultaneous operations, it was never conclusively determined to be either a Blaise bug or an OS bug.

A more serious failure related to these morning routines occurred when Hospital, designed to detect and repair data corruption (primarily involving secondary keys), began to drop large numbers of cases from the database wholesale. It did not take us long to figure out that this was happening to every case that had been touched the previous day. We were then quickly able to deduce that the handling of an inherent Blaise field left over from a previous build of Blaise was confusing Hospital and causing the cases to be dropped. The immediate solution was to stop using Hospital until the bug could be isolated and fixed. We also were able to reinsert the cases dropped by using a simple Manipula program such that no permanent data loss was suffered. Within a week or so, Statistics Netherlands found the problem and issued a new Hospital program which corrected the bug, and we resumed using Hospital.

Our TFU supervisors did not take long to begin using BtMana to dynamically track daybatch progress, view shift workloads, and reassign cases to specific groups or interviewers as needed. But in so doing, a bug was uncovered that had a serious impact on our database. Soon after the adaptation of BtMana into the daily procedure, we noticed that some cases lost entire blocks of data for no apparent reason. As BtMana was the only new variable introduced at that time, we quickly determined that using it to assign a case to an interviewer was causing the data loss. Again, the bug was quickly located by Statistics Netherlands and a new build of Blaise was released. Upon deployment, that bug did not recur.

At one point, we encountered one of our most severe bugs in a very unusual way. Due to the heavy amount of post-data-collection editing and imputation that is done by the Continuous Measurement Office, another atypical requirement of this instrument is to keep all data, whether on-path or off-path. This, of course, is not a problem for Blaise, as the KEEP statement at the field or block level allows for just that. Our instrument was programmed to maintain all data (including off-path data) from session to session and to future processing operations. But somehow, part of our instrument code was replaced with a version of similar code that lacked only the KEEP statements. This went undetected because, as it turns out, a bug in Blaise build 4.1.0.269 kept the off-path data anyway in those sections of the instrument that had had their KEEP statements removed (imagine, for once, a bug that works in your favor). But when we upgraded to Blaise v. 4.1.1.322, the lack of KEEP statements became apparent. We slowly became aware of the problem of dropping the off-path data, and hundreds of cases became afflicted with this data loss. Luckily, we were able to fix the problem quickly by simply restoring the KEEP statements.

The last major problem involved cases reappearing to interviewers even after they had been resolved or scheduled for an appointment at a different time. When the problem first occurred, it appeared to be epidemic in proportion, but as we began to research it, we discovered that it had probably been happening all along, just at a very low and infrequent level. It turned out that any case outcome was subject to this problem, and that the problem was not isolated to one interviewer, or one computer, or one network, or one instrument, or even one version of Blaise. Once in a great while, in the first year of production deployment, we would get the occasional complaint from the call unit that a closed case would appear again to an interviewer. Tracking of the audit trail would confirm this, but at first we attributed it to a minor glitch in the instrument based on a certain series of responses to the front-end questions. When the problem began occurring dozens of times per day (roughly 5% to 7% of all cases), we soon noticed that there was an inordinate number of "hanging" phone numbers. These occur when there is not a complete set of write operations performed on the dayfile and call history files. As a temporary fix, we were able to stop the problem by simply not clearing hanging phone numbers with BtMana. But this did not address the root of the problem, which Statistics Finland coincidentally discovered to be linked to the Autosave feature. When leaving a case immediately after Autosave was invoked, the case would hang in BtMana. We fixed the problem permanently by turning off Autosave in the modelib.~ml, recompiling the instrument, and reconfiguring our shortcuts to not refer to any dep.diw. A re-issue of version 4.1.2, build 371, was to have allowed for Autosave with no hanging numbers.

We discovered a number of Blaise bugs within the production environment throughout the course of the first year. But this is not unusual when dealing with relatively new software being used in ways which stretch the boundaries of its functionality. One common thread which runs through every problem is that as soon as a bug was identified and found to be replicable in similar environments, Statistics Netherlands was consistently able to identify the source of the bug and issue a new Blaise build within a very short time period. Furthermore, as a result of our close work with Statistics Netherlands in identifying and replicating bugs, we received a number of new Blaise builds containing features that were either never planned or released well ahead of schedule.

The Future

For the ACS, the future of the TFU operation features Blaise. There is still no competing CATI software which has the features and functionality that Blaise for Windows offers. Daily processing of cases through Blaise continues at planned levels, and while there are occasional problems that crop up, the overall effort is a success. The benefits of the system will pay off even more when the ACS reaches planned levels of 3 million mail surveys in 2003. The Blaise TFU operation will at that time process over 1 million cases per year. Because of the functionality of Blaise, our staff will be able to do so in a very efficient manner, allowing us to reach those target levels with a relatively small staff, and thereby greatly improve the ACS data.