The Centralized Survey Experience at National Agricultural Statistics Service

Roger Schou and Emily Caron, National Agricultural Statistics Service, USA

1 Introduction

The National Agricultural Statistics Service (NASS) is an agency of the United States Department of Agriculture. NASS is responsible for collecting, editing, and summarizing agriculture data. We are the sole agency for producing Agriculture Statistics for the United States. NASS primarily uses CATI data collection, but also paper questionnaires for mail and field interviews, CAPI field interviews, and web data collection methods.

NASS is made up of forty-six field offices across the United States, a headquarters (HQ) location in Washington, D.C., a research division in Virginia, and as of recently a National Operations Center (NOC) in St. Louis, MO. The NOC is intended to serve as a primary calling center for NASS with a capacity of up to 150 phone interviewers. Six of our forty-four field offices also serve as Data Collection Centers (DCCs), ranging from a 20 to 60 seat phone interviewer capacity.

When we last presented at IBUC XIII, we reported on the successes and lessons learned from converting our first survey to a Centralized environment: the Mink Survey. Since that time we have converted ten disseminated Blaise instruments and added fifteen new surveys into Centralized solutions all while facing numerous challenges such as an agency-wide upgrade to virtual desktops; adapting to and in some cases assisting with the centralization of other systems which interact with Blaise; updating code to incorporate our new NOC; doubling the number of employees in our section with somewhat “green” programmers who are sitting one time zone away at the new call center; and others.

2 Blaise Programming… Times Two

Eighteen months ago, the NASS Blaise development group was called the CASIC Section and it included six developers and one section head all sitting in HQ. Today, the CASIC Section no longer exists, but there is a Blaise Questionnaire Design and Editing Section (BQDES) in HQ with four developers and one section head. There is also an Enumerative Survey Development Group (ESDG) sitting at the NOC, consisting of seven Blaise developers. Three of the seven at the NOC had never touched Blaise coding and had little or no programming experience in any other language before their arrival in ESDG. A 3½ day training session was conducted for new NOC personnel and group meetings are held when needed over Virtual Teleconferencing equipment. Even though the Blaise group has grown, the overall total number of NASS employees is declining intentionally.

Plans are to add at least another 25 surveys into the Centralized solution during 2012. In a number of cases, there is more involved than just converting existing surveys from decentralized to centralized solutions or introducing new surveys. Some of the surveys are weekly surveys, which we have never attempted in Blaise at NASS. These offer new challenges. For example, some of the weekly surveys are expected to offer a place for revisions for the previous weeks’ data within a given week’s data collection.

Another set of five surveys has a need to be coordinated to allow any given respondent to respond to one or more of these five surveys, depending on the surveys for which they were sampled. A Virtual Survey Coordinator was developed using VB.NET so that a group of surveys could be completed by the same respondent in a single phone call. So even though the size of our group developing Blaise instruments has nearly doubled, the workload and new challenges have more than kept the staff busy.

3 Centralized Blaise Infrastructure

In the past with the decentralized environment, there were challenges with the physical Blaise datasets being located in every field office (FO). We had the six DCCs collecting data and sending completed forms home to the client states via Blaise datasets. It was a stable scenario, but it was very cumbersome with a large number of physical file handoffs. The maintenance rested almost entirely on the CASIC Section.

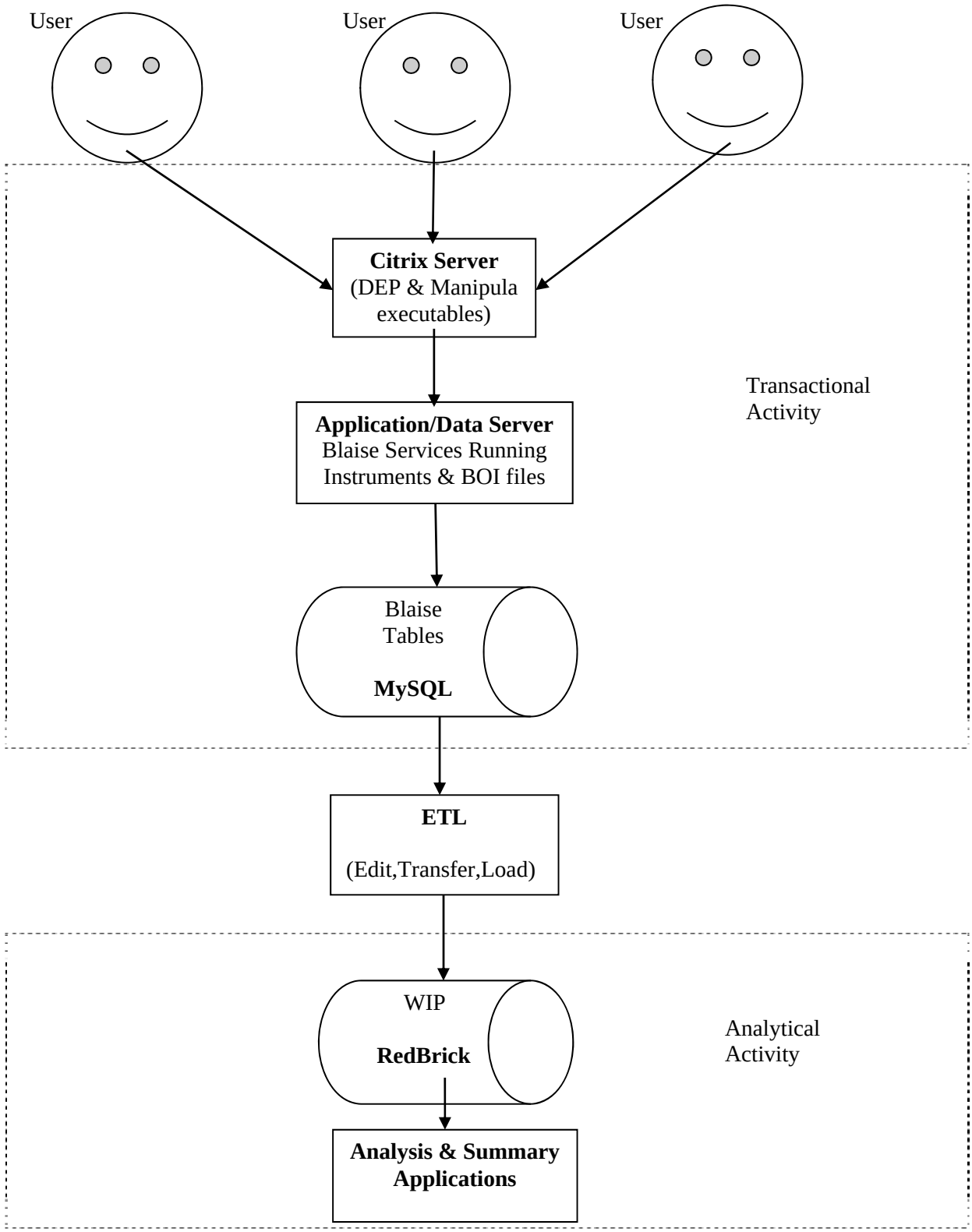
With the centralized environment, there are many more players involved in the maintenance of the systems. Figure 1 shows the many different layers of infrastructure involved in our Centralized Blaise environment.

Figure 1. Infrastructure Table

Component	**Description**
Citrix server	Gateway to all applications. Conversion of all users to Citrix was completed in 2011.
Application servers	Blaise applications and services reside here. We use three different app servers: Development, Beta and Production.
Database servers	The databases themselves reside here. We access three different database servers: Development, Beta and Production.
Storage	There must be physical storage for the databases
Communications	The Wide Area Network must be available for users across NASS to reach the centralized applications
MySQL database	Transactional activity takes place here
ETL to WIP	The ETL (Extract, Transfer, Load) moves data from MySQL to WIP on a timely basis so analysis and summary reflect the latest data
WIP database	RedBrick analytical database - Analysis and summary programs pull data from here. We actively “write” to two different WIP databases: Beta during testing, and Production during live data collection.
Windows AD Table	Each NASS employee that needs to access Blaise must have current information in Windows AD about their ID, location (FO or HQ Div-Branch-Section), and role in order for their rights to the data and menu options to be set correctly.

Figure 2 gives a picture of the system flow. It also identifies the areas of transactional activity on the MySQL database and analytical activity on the RedBrick database.

Figure 2. System Flow



4 Determining Rights in Centralized Blaise

The Blaise software is accessible to users through a VB.NET menu system, written and maintained by the Blaise programming staff. One of the first responsibilities of the menu is to see which user is attempting to access the system and check that individual's information in the Windows AD table. If any pertinent information about the user is missing, or if the user not found, the menu will not allow the user to go any further. However, if all information is present and accounted for, the user will be granted access and the location and role of the user are stored. Role could be statistician, stat assistant, supervisory interviewer, or interviewer.

In order to maintain a common structure across all centralized Blaise instruments, we use Generic BOI files with the in-depth data partition type. This allows us to use one ETL for all surveys to copy the data from the Blaise MySQL tables to the WIP analytical database. In addition to the eight standard Blaise tables that accompany Generic BOI files, we found the need to create a few other flat tables: CASIC_SurveyInfo, CASIC_Management and CASIC_FAT. As other systems in NASS centralize, this same type of information will be needed, so these tables may end up being shared or moved so that many can glean information from them.

The CASIC_SurveyInfo table holds critical information about each survey, such as survey code; year, month and day values; Blaise instrument name; different survey type indicators; data collection dates and other useful information. The CASIC_Management table contains many key Blaise fields on which we often need to sort or limit the datasets. These fields are defined as indexes and used in Record Filters in Manipula. The fact that these fields are indexed allows the queries to the database to be quick and efficient. The CASIC_FAT table (FAT = FIPS Assignment Table) is necessary to track which NASS locations are collecting and/or editing survey data, or simply identifying a state that is being serviced by other states. The acronyms we use for these three office level roles are as follows: DCC = Data Collection Center, EC = Estimation Center, and CS = Client State. HQ staff can interactively assign CSs to DCCs and ECs for each survey using the CASIC_FAT. Once these assignments are created, the resulting data is populated into the CASIC_FAT table and the menu will know which rights to provide to each office. **Every different survey we conduct at NASS has a different mix of DCCs, ECs, and CSs.**

Every survey instrument in the centralized solution has three fields defined in it to store the DCC, EC, and CS responsible for each form. These fields are populated when we initialize the sample for the survey. Once the menus have identified the user's attributes, Manipula can be used to retrieve the appropriate forms by using Record Filters.

The buttons on the VB.NET menu interface will appear and disappear dynamically based on a combination of the roles and location of the person attempting to access them. The access to the centralized database is controlled entirely by the menu. The design of the menu, as seen in Figures 3 and 4, allows whole tabs to be invisible based on the needs of the survey and the roles of the user accessing them. The design also allows group boxes to be made invisible within a certain tab. The goal is to only supply a given user the necessary buttons/functions for a given survey.

Figure 3. Data Collection Menu Tab

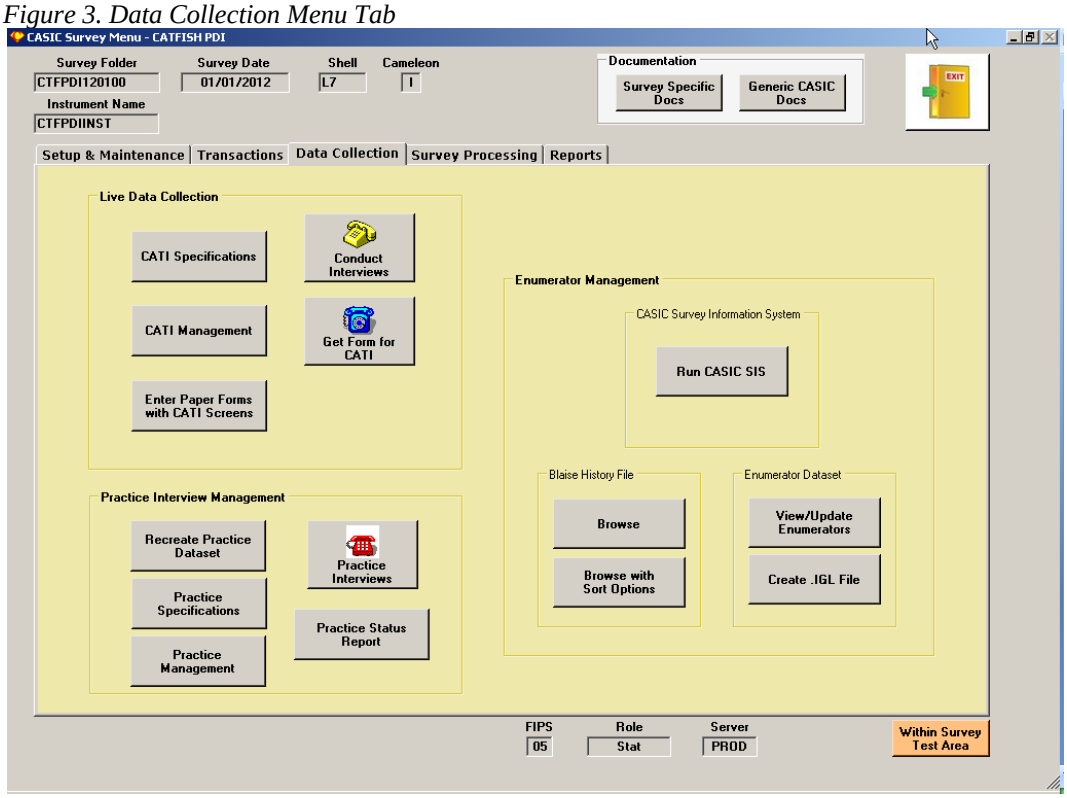
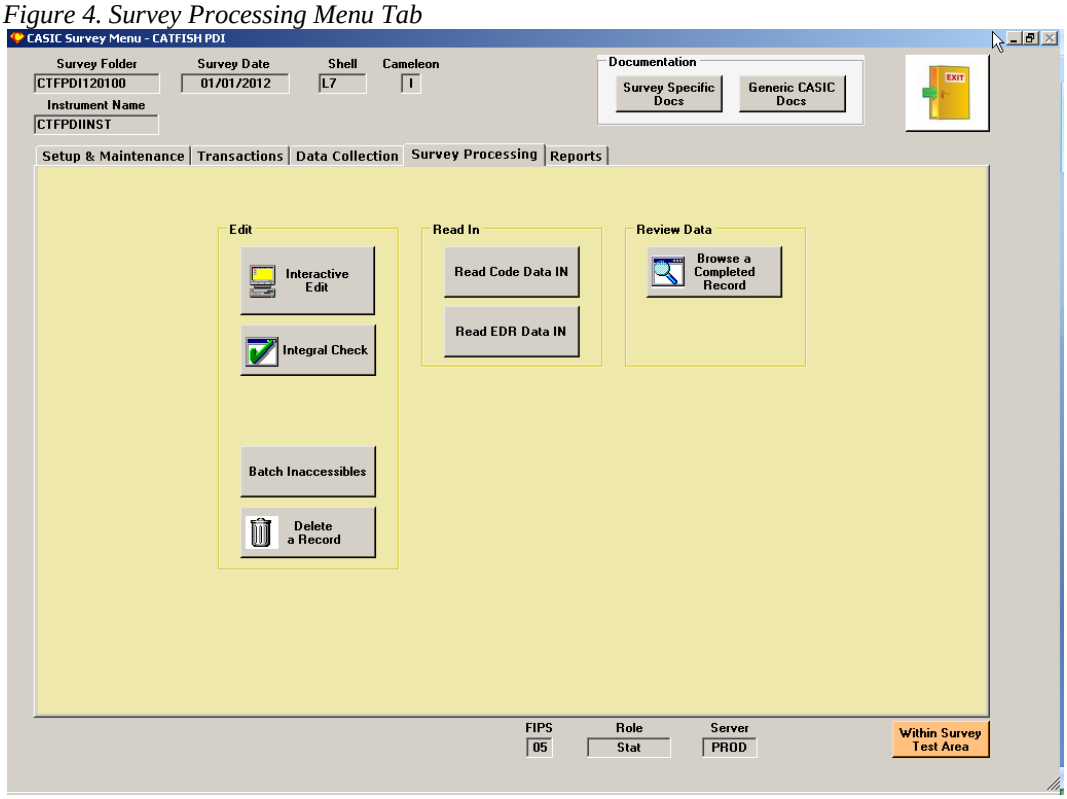


Figure 4. Survey Processing Menu Tab



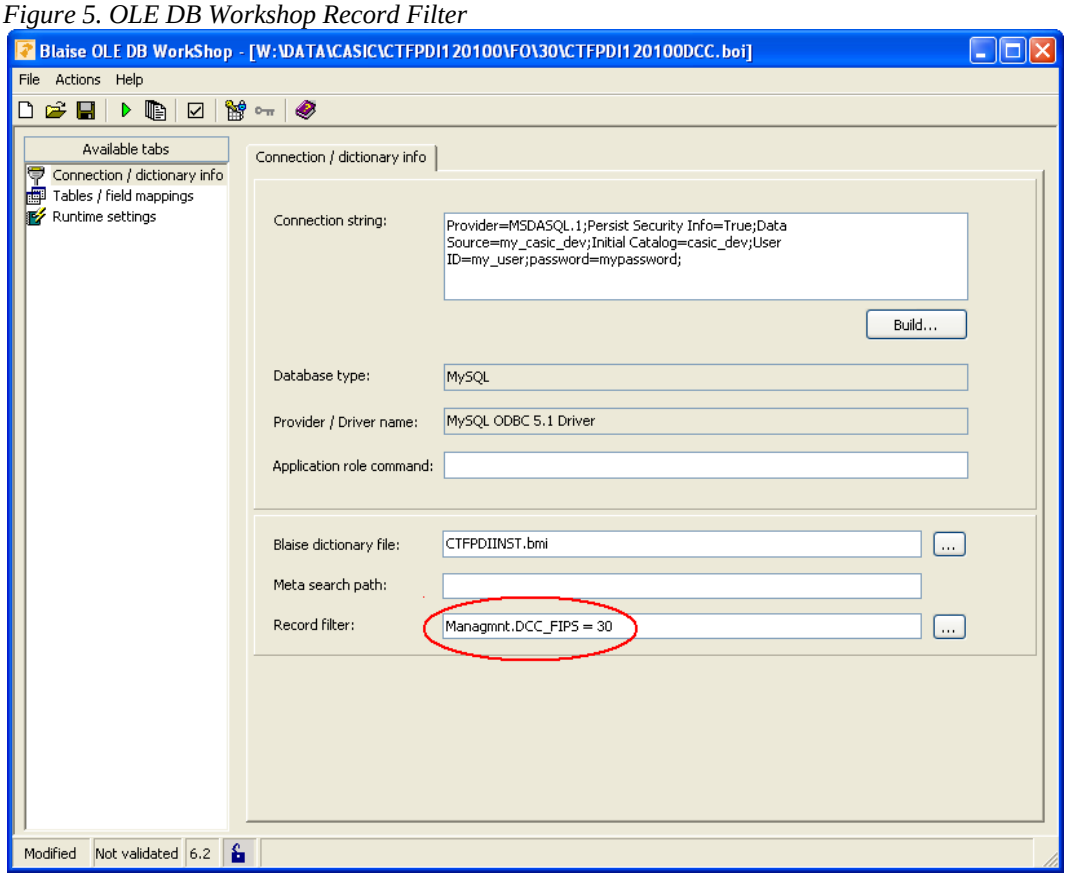
So to summarize, our VB.NET menu needs to react based on where a user is sitting (AD table), the user's role in the office (AD group), the office's role in the survey that is selected (CASIC_FAT table), and the specifics of the survey itself (CASIC_SurveyInfo table). All of this information together serves as a map of the different rights and functionalities we are providing to our users at the

survey level. So each piece of information is critical to ensure security and functionality of our Blaise instruments.

5 Managing CATI

For each survey, we have a main .BOI file defined for the entire sample. However, we need to virtually separate the data for collecting as well as editing the data. With all of the data centrally located in the same MySQL database, there is a need to manage the daybatches for each FO that collects CATI data. We also have to take into account six time zones. We create a folder for each FO involved in the survey for either data collection or editing. In this folder exists a filtered .BOI file that is used for creating daybatches as shown in Figure 5. This limits the daybatches as well as the forms that are accessible within CATI Management to the forms that are assigned to that DCC.

Figure 5. OLE DB Workshop Record Filter



For all other processing like interactive editing, reports, posting transactions, etc. we use the main .BOI file. The reason being, Blaise .BOI files are unaware of locked forms in other .BOI files. So in order to limit any possible collisions between users, we use the main .BOI file (many times with Record Filters) so that Blaise is aware of the potential locked records.

In the decentralized scenario, there were only two time zones of which Blaise needed to be aware: the time zone of the respondent and the time zone of the interviewer (and incidentally the Blaise dataset which was always the same as the interviewer). With the centralized scenario, Blaise now needs to keep track of three time zones: the time zones of the respondent, the interviewer, and the database. We learned that there are two places in the CATI Specifications where the time zone of the database plays a role. The first is the Crew Times. These must be defined in the time zone of the database, which is not necessarily the same as where the actual crew is sitting. The second is the Time Zone differentials. These must be defined with the time zone of the database as the “home” time zone – the one with a differential of zero.

6 Split-State Scenarios

Roughly a year ago we came across a situation where one of our FOs wanted to send part of its sample to another FO to be collected. We had tossed around the idea of split-state samples before but had never had to deal with them. To complicate matters, the FO needing to take part of the sample wasn't defined in our CASIC_FAT table as having DCC capabilities for that particular survey – yet another FO was defined as collecting data for the receiving FO's own sample!

We found the best solution was to add another column to the CASIC_SurveyInfo table called Add_Func (short for "add functionality"). The FO needing to take part of the other FOs sample for phoning received a value of 1 in this column, to indicate to our VB.NET menu that it needed to have the data collection related buttons activated, in spite of the fact that they were not designated as a DCC. Then all we needed to do was to run a Manipula program to update the field in the dataset to assign that FO as the DCC_Fips for the subsample of records needing to be transitioned. Our Manipula Record Filters took care of the rest.

7 Growing Pains

We would be remiss if we did not mention some of the growing pains that we suffered through to get where we are. The Blaise programming group at NASS was one of the first groups to move toward a centralized environment. So as we moved forward, we had to continually build bridges to the legacy systems so that the overall survey process would still function normally. As other pieces of the process began to centralize, new links to these pieces needed to be developed while maintaining the bridges so that parallel testing was possible. It is our goal that we will eventually no longer be passing data to the analysis and summary systems. The ETL will copy what is needed from the Blaise MySQL tables to WIP, and all post editing systems will retrieve what is needed directly from WIP.

The pre-survey processes are in the beginning stages of being centralized. The NASS Survey Management System (SMS) where the name and address files are loaded to create labels as well as the input files to our Blaise initialize process is being redesigned into a centralized environment. Once this is achieved, our front-end process in Blaise will need to be altered to accept a nationally generated file instead of multiple state-level generated files. Also, it is not impossible to imagine the Blaise database being populated for each survey by directly reading the centralized SMS databases.

We found a major bug in the services part of the Blaise software, which halted most of our forward momentum. The services would lock up and freeze all of the users until the services were restarted. It took us awhile to even diagnose the problem. Once we could reproduce the problem (although sporadically), we had to send Statistics Netherlands a set of files so they could hopefully reproduce the problem. They were able to set up a survey in the closest environment to NASS that has ever been implemented outside of NASS. They were successful in reproducing the freeze, and from that point they were in high gear to solve the bug. Our faith in the Blaise Team at Statistic Netherlands never faltered, but the task was not a trivial one. Several months later, a new beta version was sent in which they could no longer reproduce the error. The testing with this version will begin on the day this paper is due, February 29, 2012. We anticipate that the tests will prove that the system is sound and a production version can be released soon. This will allow us to move our more time critical, high-profile surveys into the centralized solution. Up to this point we have not abandoned our forward progress, but limited it to smaller surveys and surveys with a large data collection window.

As systems become more and more centralized, concepts of databases working together arise. "Why can't these ASCII files be eliminated?" "Why can't this database be aware of this other database?" "This information should be stored in a common table shared by all." All of these exciting questions and observations are valid and usually good ideas. As the overall staffing numbers continue to decline at NASS, and the need to continue the integrity of the on-going survey programs remains, the challenge becomes reigning in those who want to implement these ideas immediately. Our goal is to create a sound survey process utilizing all of the efficiencies possible with enough forethought that the

systems communicate with each other effectively and minimal rewrites are needed. We want to avoid the feeling that we have jumped into a rowboat and started floating only to realize that the oars are still on the shore.