Using Blaise for Coding Listing Instruments

Malcolm Robert Wallace
Technologies Management Office
Census Bureau

1. Overview

The Census Bureau conducts many listing operations (surveys), using Computer Assisted Interviewing (CAI) instruments (listing instruments), as a means of obtaining sample for many of the surveys that it conducts. Conducting listing surveys is not flashy or exciting. It's a necessity of the survey industry to make sure that a complete sample is taken for the various surveys being conducted. It's a "behind the scenes" operation that doesn't get much appreciation. For the most part, there are no published results from the listing part of a survey. The listing instrument is the "tool" used to gather and, at times, select the sample person or unit. It then passes this information on to other systems or other instruments that will collect the interview data and collect all the "glory" associated with published results.

1.1 Paper Objectives

This paper will explain the differences between a listing "survey" instrument and a household/respondent (interview) survey instrument, review the different listing surveys conducted by the Census Bureau, go over unique functionality required for our listing surveys, discuss some of the challenges the Census Bureau has encountered with using Blaise for coding listing instruments, and review future plans for using Blaise for some of our existing listing applications.

1.2 What is a Listing Instrument and How is it Different from an Interviewing Instrument?

A listing instrument is a CAI instrument that is used as a tool by the Field Representatives (FRs) to collect and sample lists of data in order to generate sample cases for interviewing. These lists of data can be various things, such as lists of permits, addresses, or names.

In the past, a listing instrument was an instrument used by the FRs to only collect "lists" of data. A listing instrument was strictly designed to guide the FR through the listing process, to collect lists of data, and to sample these lists when the listing was complete. The FR did not "interview" anyone with these instruments. Listing instruments were designed this way partly because that's what was required and partly because of the limitations with the software used to create these instruments (CA Clipper).

However, with the movement to use Blaise for coding our listing instruments, the difference between a listing instrument and an interviewing instrument is no longer as obvious. Some of the newer listing surveys now require a respondent to answer some interviewing questions. These surveys still have a listing component that the FR must complete on their own. Regardless, there are some other differences between a listing survey and interviewing survey. Listing instruments (surveys) are typically used as an intermediate step in the overall survey process. The data collected by the listing instrument is used as input into an interviewing instrument or control systems either right after the listing is completed or in the future after headquarters has sampled it.

Another distinction of a listing instrument, is that the FR determines when the case (listing) is complete. With interviewing instruments, the respondent controls whether a case is complete by answering the questions and reaching the end of the interview. Once all questions are answered, the case is considered complete. Since the FR is conducting the listing without a respondent, they must determine when the listing is complete. Lastly, listing instruments tend to contain some unique functionality that is usually not contained in interviewing instruments.

1.3 Listing Instruments in the Past

In the past, the Census Bureau's current surveys used CA Clipper to code listing instruments. This was an excellent application to use for listing operations since it used an underlying database and so allowed for huge listings without pre-defining a maximum limit for the number of lines listed. It also allowed for complex sampling of the data that required sorting the records in the listing (database). Using Object Oriented Programming techniques, the same look and feel was applied to the various listing applications produced at the Census Bureau. The main downside to using CA Clipper was that it did not handle the asking of questions sufficiently. It also didn't produce any type of audit trail file that could be used to troubleshoot challenging field problems.

1.4 Current and Future Listing Instruments

About the same time the Census Bureau started to use Blaise for coding new CAI surveys, two new listing projects were started. It was decided to use Blaise for these listing projects as well. We have since coded a third listing instrument in Blaise and plan to convert three other listing applications in the near future. Although the differences between a listing "survey" instrument and a regular household/respondent survey instrument can be subtle, they do exist.

2. Listing Operations Conducted by the Census Bureau

As mentioned earlier, the Census Bureau conducts many listing operations as a means of obtaining sample for many of the current surveys it conducts. These listing operations vary greatly. Some require the FRs to key in ALL of the listing data, some only require the FRs to key in the sample lines from a paper list, some even require less. The Census Bureau's current surveys area has seven different listing applications (instruments) that are in production or will be in production shortly.

Some of the listing instruments contain interviewing and listing portions, while others are strictly used for listing information.

2.1 Blaise Listing Instrument Surveys

The Census Bureau has conducted 2 production listing surveys with a third going into production in 2005. Each of the Blaise Listing instruments have there own unique functionality. However, where possible, the same "look and feel" is used to give the FRs some consistency.

2.1.1 American Community Survey – Group Quarters (ACS-GQ)

This instrument is scheduled to go into production in 2005. The ACS-GQ instrument is meant to be more of a "tool" for the FRs to use to conduct sampling of the number of units found at a GQ and then allow for the keying of the sample GQ units. This instrument has three main goals – 1. to sample the number of units found at the GQ, 2. to collect the GQ unit information for the sample units, and 3. to update the control systems with the result of the sampling and listing. This is strictly a listing instrument since it doesn't contain any questions that need to be answered by a respondent.

For this instrument, the FRs only need to key GQ information for the sampled GQ units. This is not a complete listing of the GQ. Once the listing is complete, the sampled information will be "handed off" off to the control systems and the FRs will conduct PAPI interviewing of the sample GQ units.

The FRs and Regional Offices (ROs) will then use the control systems to monitor the progress of the paper questionnaires that are handed out to the sample GQ units. The instrument is responsible for the setting of the initial outcome of the paper questionnaires that are needed for each GQ. Non-sample questionnaires are coded appropriately so they are removed from the control systems. By having the automated instrument and control systems, the Regions know exactly how many questionnaires to expect and how many listings are outstanding.

2.1.2 GQ Automated Instrument for Listing (GAIL)

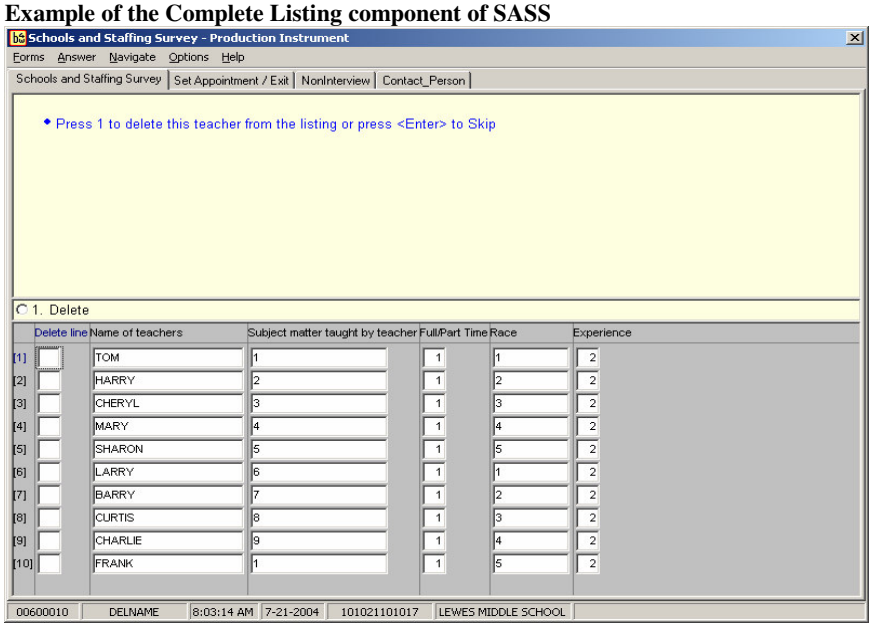
This instrument is currently in production. This is another instrument that collects information about GQs. Although this instrument doesn't do any sampling, it's much more complex than the ACS-GQ instrument since it is used to conduct complete listings at the GQ as well as updating information from a previous listing. The instrument also allows for many listing options.

The GAIL contains both an interviewing piece, where the FRs ask questions of the GQ contact person, and a listing piece - where the FR collects and lists information about the GQ. The information collected by the GAIL instrument is sent back to the Census Bureau and is used to update the GQ sampling frame for the current surveys.

2.1.3 Schools and Staffing Survey (SASS)

This instrument was used in production during the 2003-2004 school year. The instrument contains 3 main components. The first component is a screener portion where the FR makes contact with the school (or address) and interviews a respondent to determine if it's in scope for the survey. The second component of the instrument is a complete listing component where the FR keys in the names of all teachers in the school along with other information about the teacher.

Example of the Complete Listing component of SASS



The third component is a complex sampling component that takes place once the listing is complete.

The instrument then supplies the control systems with the sample teacher name and other listed information for the teachers that fell into sample.

Once the listing is complete, the FRs conduct PAPI interviewing of the sample teachers. They hand out paper questionnaires to the teachers that have fallen into sample. The control systems are used to monitor the progress of the paper questionnaires, as well as the listings.

2.2 CA Clipper Listing Instrument Surveys

The Census Bureau still has a few CA Clipper listing surveys that are in production. These surveys have been in production for years and have had very few problems. All of these surveys will be converted over to Blaise in the near future.

Obviously CA Clipper is out-dated software and these instruments must be converted to a GUI software package. However, using Clipper for coding listing instruments does have some advantages – some of these include:

- Clipper allows you to conduct complex sampling while in the instrument
- Clipper allows you to generate new CAPI cases "on the fly" while still in the listing
- Clipper doesn't require a pre-defined upper limit of listing lines needed for a survey. You can add as many lines as you need for any given case.

Blaise also has advantages for coding listing instruments - some of these include:

- Blaise allows you to easily code questions in the listing instrument. This advantage will allow us to eliminate one of the interview instruments required in SOC by combining it with the listing instrument.
- Blaise has some required functionality (such as the copy from line above) already built-in.
- Blaise generates Audit trail files for help in trouble shooting

2.2.1 Permit Address Listing (PAL)

This is a monthly survey that has been in production since 1996. This instrument collects building permit information for new residential construction. This is a "true" listing instrument. The FRs visit permit offices each month and key in the permit numbers, address information, and geocode information for every residential permit issued that month. This instrument contains no "interviewing" questions. The FR simply enters the listing instrument and starts keying away.

The data collected by the instrument is used by the current surveys to update the new construction sampling frame.

Although it sounds simple, this is a fairly complex instrument behind the scenes. There are many special situations that the instrument must account for. There is also a lot of added functionality to aid the FRs in their listing.

2.2.2 Survey of Inmates and Local Jails/Survey of Prison Inmates (SILJ/SPI)

These two instruments are very similar and alternate in production every couple of years. The instruments are used by the FRs to guide them in obtaining an accurate list of inmates/prisoners, to sample that list based on the number of inmates and sampling information provided, and then to collect the names and IDs of the sample inmates. The instrument then generates the input file needed by the Computer Assisted Personal Interview (CAPI) interviewing instrument.

Once the listing is complete, the FRs conduct CAPI interviews using a separate interviewing instrument. This survey requires additional functionality of being able to copy the new cases to multiple laptops so that one FR can do the listing but, more than one FR can interview at the jail or prison.

2.2.3 Survey of Construction (SOC)

This is a monthly survey that has been in production a long time. This is another instrument that collects permit information from building permit offices. It gathers the permit number, address information, and builder information. It is also a "true" listing instrument that doesn't contain any interview questions. On top of doing a complete listing of the permits for the month, this instrument also conducts complex sampling of these permits and generates input files that are used by another CAPI instrument. Some unique functionality of this instrument is that it generates CAPI cases "on the fly" and allows the FR to call a CAPI interviewing instrument from within the listing instrument.

Once the listing is complete, the FRs conduct CAPI interviewing using a separate interviewing instrument.

3. Unique Functionality of Listing Instruments

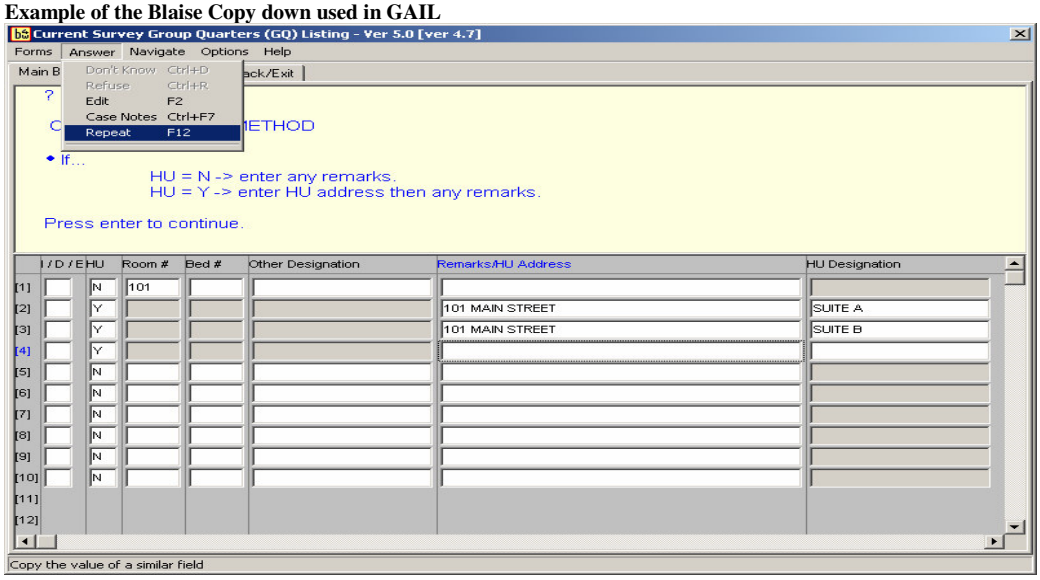
Listing instruments tend to have some unique functionality that isn't normally present within interviewing instruments. This functionality is added to try to help the FR in the listing process and to assure better quality data. Much of this functionality can be handled with Blaise and may be contained in interviewing instruments.

3.1 Copy Contents from Cell Above

In listing operations, much of the data listed may be the same for multiple lines. For example, when listing permits from a Building Permit Office, many of the new permits may be for houses on the same street. These permits are typically taken out at the same time so they will be listed one after the other. Being able to copy the address from the line above not only saves the FR time, but also helps add consistency to the data in the listing.

The Blaise "copy down" feature is a great feature that will receive plenty of use in listing operations. This feature eliminates the need for coding special functionality for a listing instrument.

Example of the Blaise Copy down used in GAIL



3.2 Allow for a Large Number of Rows in the Table

Some listing applications conduct sampling based on the number of units or people at the facility which means they're sampled before the listing is conducted. These applications only require a finite number of listing lines (rows) and are fairly simple to code. However, other listing applications conduct complete listings of an undetermined number of lines. These listings can be thousands of lines long and create an issue for us with using Blaise.

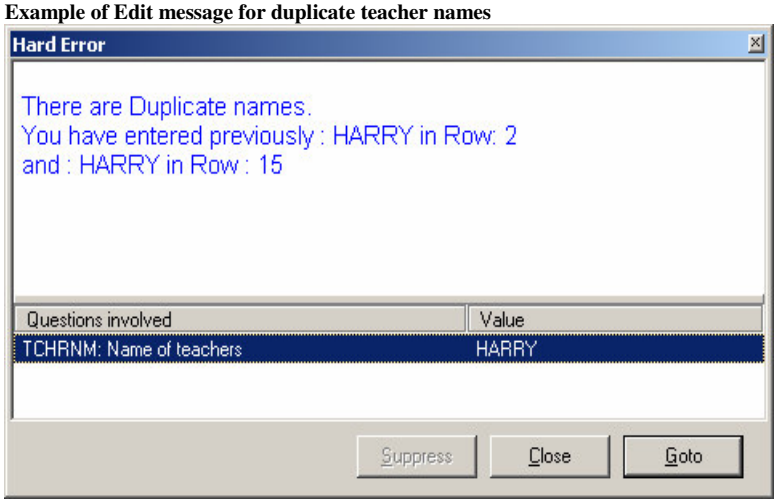
Since one must define an upper limit for a Blaise table, we have to try to determine what the maximum number of listed lines will be for any listing. If we make this number too large, we may create performance issues with the instrument. If we make this number too small, then we risk not being able to list everything for a given assignment. The "maximum number of lines needed" for a listing operation should be supplied by the subject matter experts after they have conducted some research.

3.3 Check for Duplicate Items in the Listing

A fairly common requirement for listing applications is to verify that the FR has not listed any duplicate records. Since listings can be quite large, it is easy for an FR to lose their place and start keying data that has already been keyed. If this is a listing that is going to be sampled by the instrument, many bad things could result.

For example, when creating a complete listing of all of the teachers in a school, we would want to add an edit to verify that each teacher's name entered is not listed on any other line in the listing. Since listings can be quite large, we must be very careful on where this edit is placed or we could create performance problems with the instrument. For SASS, the FR is not allowed to enter in a duplicate name as you can see from the "Hard Edit" below.

Example of Edit message for duplicate teacher names



Example Code used for un-duplication Edit

At the table Level:

```
NumToken : 0..400 3 of rows
Tempstring : String[17200]
Tempstring is defined as an auxfield at table level
Tempstring appendix names delimited by a comma, nutoken is the number of members entered in
tempstring

If Teach[X].TCHRNM <> EMPTY THEN
        Tempstring :=tempstring + Teach[X].TCHRNM + ","
        NumToken := NumToken + 1
Endif
```

At the Row Level:

```
CurrCharPos := 0
PrevCharPos := 1
rowLine := EMPTY
FOR Y := 1 TO NumToken DO
        CurrCharPos := POSITION(',', tempstring, PrevCharPos)
        TempSplit := SUBSTRING(tempstring,PrevCharPos, CurrCharPos-PrevCharPos)
        nametemp[Y] := tempsplit
        PrevCharPos := CurrCharPos + 1
ENDDO
{*** RPV Building a check based on the arrayed stuffed above   ***}



FOR Y := 1 TO NumToken DO
        IF nametemp[Y] = TCHRNM THEN
                rowLine := Y
                ERROR "@L@/ There are Duplicate names. @/
                        You have entered previously : ^ nametemp[Y] in Row: ^rowLine @/
                        and : ^TCHRNM  in Row : ^rownum@L"
        ENDIF
ENDDO
```

If we had allowed the FR to list duplicate teachers, then not only is sampling incorrect, but we could end up with having the same teacher fall into sample twice. That means that we would lose an interview before even trying to obtain it.

3.4 Complex Sampling of the Listing Data:

By automating listing operations, the Mathematical Statisticians (Math Stats) can have the computer do the sampling instead of the FRs. Knowing this, some Math Stats like making the sampling very complex. Not only will they stratify the listing based on the listing data, they'll stratify it four different ways.

Here's a snippet from the SASS sampling specs on how to calculate the Take Every for each stratum:

Compute a take every (TE(h)) for each of the five teacher strata as follows:

Table 3. Take Every Definitions for Teacher Strata

Teacher Stratum	*Take Every*
(A) – Experienced	*Xif SMPSIZAB < (TOTLA + TOTLB), then* $TEA = \frac{TOTLA + (OVRSMP * TOTLB)}{SMPSIZAB}$
	Xif SMPSIZAB = (TOTLA + TOTLB), then TEA = 1
(B) – New	*Xif SMPSIZAB < (TOTLA + TOTLB), then* $TEB = \frac{\frac{TOTLA}{OVRSMP} + TOTLB}{SMPSIZAB}$
	Xif SMPSIZAB = (TOTLA + TOTLB), then TEB = 1
(C) – Bilingual	$TEC = \frac{TOTLC}{SMPSIZC}$
(D) – API	$TED = \frac{TOTLD}{SMPSIZD}$

Since Blaise will not allow you to sort the data efficiently, the complex sampling routine was run in a Manipula script after exiting the DEP. Below is an example of the Manipula script that was used to sort the listing and to sample the first Stratum:

Partial Manipula Code to Sort Listing by Stratum, Subject, and Line Number:

```
FOR Z := 1 TO 4 DO     {//Currently, there are 4 Strata: A thru D}
    currStratum := Z
    currSubj := 0
    FOR J := 1 TO 9 DO   {//Currently, there are 9 subjects}
      currSubj := currSubj + 1
      FOR I := 1 TO Limit DO
            iPntr := I
         IF (UpFile1.Teacher.TeachTbl.Teach[I].TeacStrtum = currStratum) THEN
           SortBySubj(currSubj, UpFile1.Teacher.TeachTbl.Teach[I].SubjSamp,
                                  iPntr, SortPntr)
        ENDIF
      ENDDO
   ENDDO
ENDDO
```

Manipula Code Example for Sampling the First Stratum

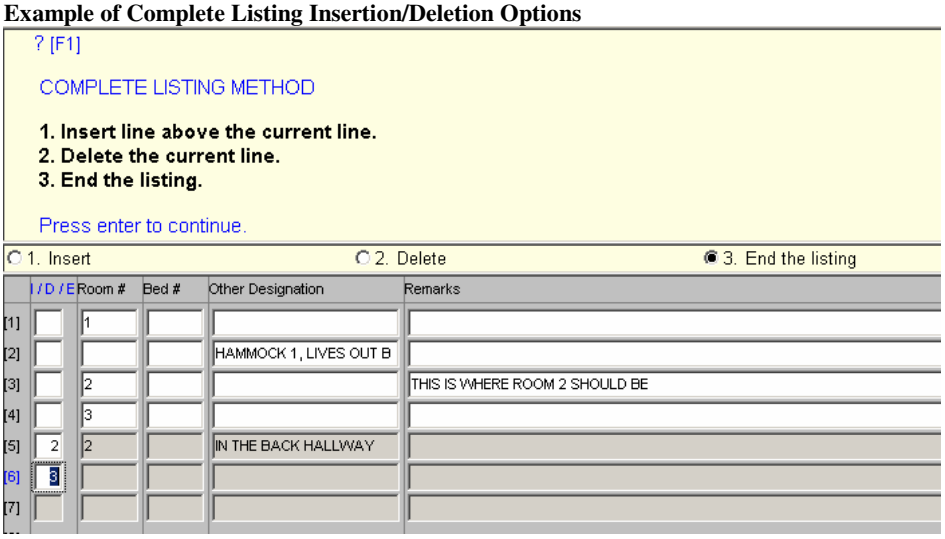
```
{//Initialize the Sampled/Subsampled indicators in the OutTableSampled//}
 FOR I := 1 TO Limit DO
        UpFile1.Teacher.OutTableSampled.Row[I].Sampled := EMPTY
        UpFile1.Teacher.OutTableSampled.Row[I].Subsampled := EMPTY
        IF (StartRowOfStratumC = EMPTY) AND (UpFile1.Teacher.OutTableSampled.Row[I].TStratum = 3) THEN
                StartRowOfStratumC := I
        ENDIF
        IF (StartRowOfStratumD = EMPTY) AND (UpFile1.Teacher.OutTableSampled.Row[I].TStratum = 4) THEN
                StartRowOfStratumD := I
        ENDIF
 ENDDO
 {//Get the raw totals of each stratum.  Remember to skip sampling/subsampling if TOTL(h) = 0!!}
                tTotl_[1] := Upfile1.Teacher.TeachTbl.TotlA
                tTotl_[2] := Upfile1.Teacher.TeachTbl.TotlB
                tTotl_[3] := Upfile1.Teacher.TeachTbl.TotlC
                tTotl_[4] := Upfile1.Teacher.TeachTbl.TotlD
{///Select First Batch from ea. stratum (Step 1)//}
                tSW_[1] := UpFile1.SwA
                tSW_[2] := UpFile1.SwB
                tSW_[3] := UpFile1.SwC
                tSW_[4] := UpFile1.SwD
 FOR stratNum := 1 TO 4 DO
   IF tTotl_[stratNum] > 0 THEN   {//Don't sample if there are no teachers for this stratum (i.e. TOTL(h)=0)!}
       RoundUp(tSW_[stratNum], SampTeacher1st[stratNum])
   ENDIF
 ENDDO
 {//Now do the actual selection for the Step #1 selectees by marking the OutTableSampled table row//}
 FOR stratNum := 1 TO 4 DO
     cntr := 0
     FOR J := 1 TO Limit DO
         IF Upfile1.Teacher.OutTableSampled.Row[J].TStratum = stratNum THEN
            cntr := cntr + 1
            IF cntr = SampTeacher1st[stratNum] THEN
               UpFile1.Teacher.OutTableSampled.Row[J].Sampled := 1
            ENDIF
         ENDIF
     ENDDO
 ENDDO
```

3.5 Insertion and Deletion of Rows

For some listing applications, it is important that the FRs have the ability to insert rows anywhere in the listing. There may be a requirement to have a listing completed in a certain order. For example, we may want units in a GQ to be listed in the order traveled by the FR.

Likewise, it's important to be able to delete a listed line (row). In some cases, we will want this line permanently removed (initial listing) and in some cases we'll want to shade the line (updating a listing) so that we keep a record of it.

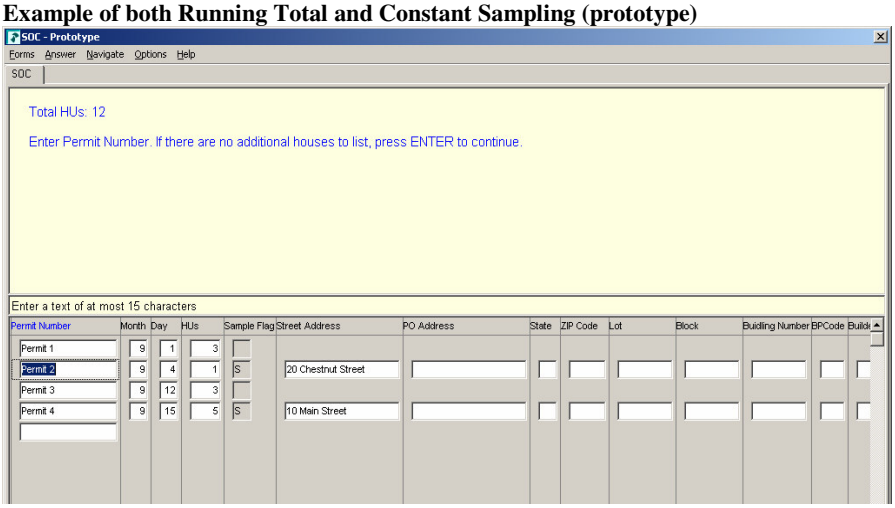
Example of Complete Listing Insertion/Deletion Options



3.6 Running Total of "How Many" Have Been Listed

Some listing operations require a running total to be displayed to the FR. This total is not necessarily the number of lines. Instead, it may be the number of units. For example, a permit may be issued for multiple housing units (HUs). So, the total number of lines listed will most likely not equal the total number of HUs. Total HUs is important information to the FR to help them verify their listing is complete. Again, this functionality is something that can be handled by Blaise.

Example of both Running Total and Constant Sampling (prototype)



3.7 Conduct Sampling After Each Line has Been Listed

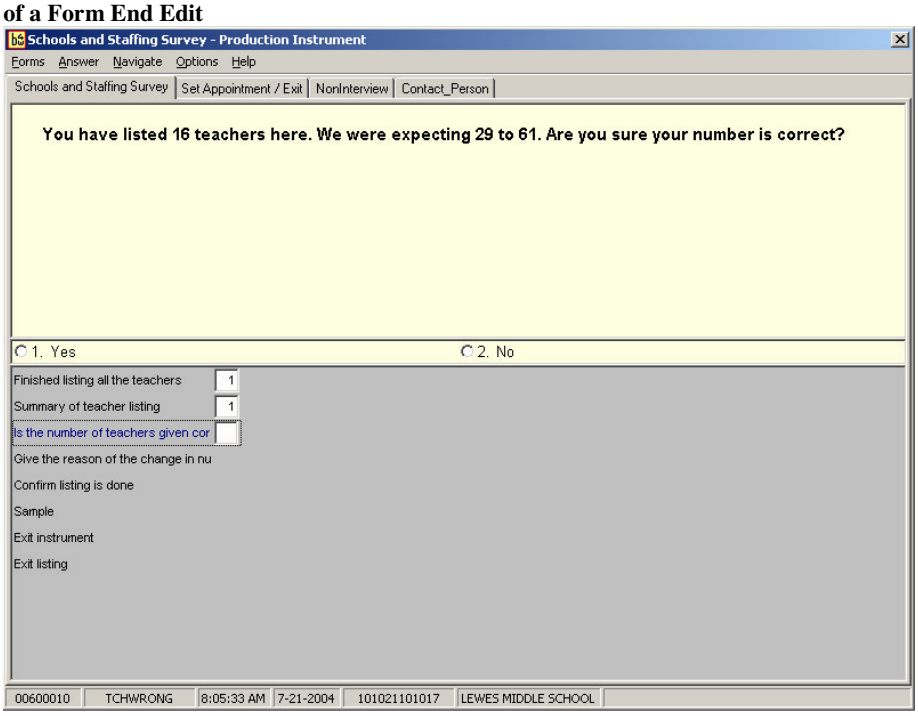
One of our listing applications requires that preliminary sampling be run after every line is entered or modified to indicate to the FR whether that line has a chance to fall into "final sample". This is done so that the FR knows whether to collect additional information from the permit. The FR can collect the information at this time instead of going back through the permits once final sample is run. Blaise handles this functionality as well. In the example above, the "Sample Flag" column is recalculated every time the number of HUs is modified. When the row falls into sample, an "S" is put into the "Sample Flag" column and all of the additional information needed for that permit will come on route.

3.8 Conduct "Form End" Edits

Some listing operations take place in stages. We may collect part of the listing data in one place and finish the listing somewhere else. For example, the FRs may list the addresses for the permits at the building permit office and then finish geocoding each line at home. This means that we have to allow for "incomplete" data in the listing. However, this data has to be completed before the listing is considered "complete". The instrument must be designed so that some edits are only run once the FR says the listing is "complete".

This functionality can be built into a Blaise listing instrument. We just need to have a question to base these edits off of. We would go to an error screen instructing the FR what needs to be "fixed" in order to pass the form end edits.

Example of a Form End Edit



3.9 Exiting a Listing Instrument

Again, the FR must determine when the listing is complete. We allow the FR to exit from the listing in two different ways.

The first way is to F10 to the "Set Appointment/Exit" Screen (see tab in above example). From here, the FR can exit the listing by setting a callback. This sets the outcome of the listing to a partial and the FR must re-enter the listing and complete it. From the "Set Appointment/Exit" screen the FR can also assign a non-interview outcome code to the listing by going to the non-interview screen and selecting one of the options.

The second way the FR can exit the listing, and the only way to assign a complete outcome code to the listing, is to complete the listing, enter "yes" at the "Finished Listing Screen" question, and then resolve all form-end edits. Once that has been done, the listing will wrap up as completed. In the above example, the FR is allowed to complete the listing even though the number of lines listed is outside of the expected range. The FR, however, must explain this discrepancy before exiting the listing.

4 Challenges with Listing Instruments

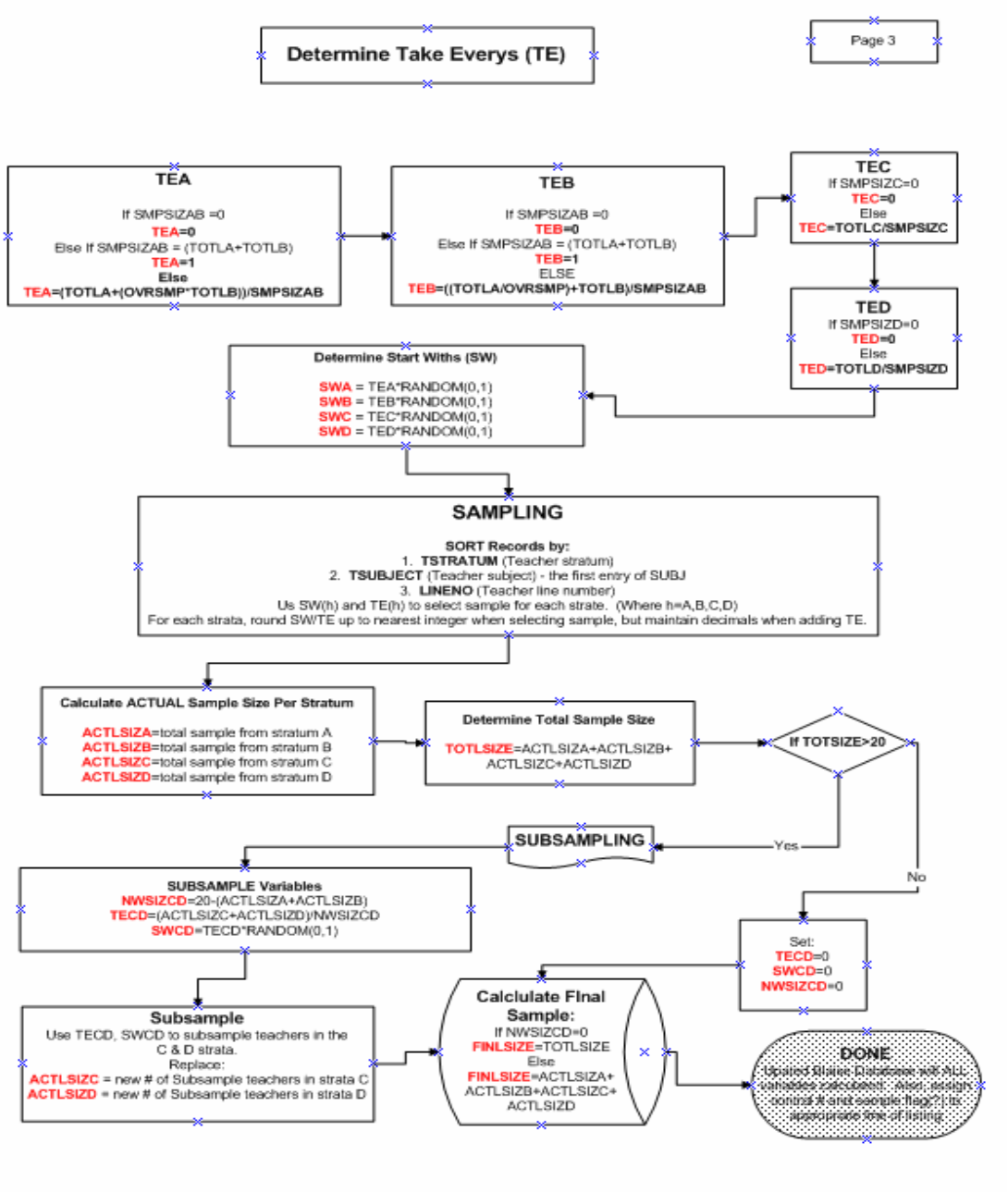
4.1 Coding and Testing Complex Sampling

Regardless of what software package is used, coding complex sampling is not fun. With Blaise it was even more challenging because this had to be handled outside of the instrument. The sampling piece of the instrument is the most important part. If this doesn't work correctly, then it doesn't matter how nice the listing or screener portions worked – the survey is going to be a failure.

The two main challenges with incorporating complex sampling into a Blaise listing instrument were verifying the sample results were correct and trouble shooting if they weren't. With the sampling being so complex, we couldn't easily tell whether the results produced were correct. We just knew that teachers fell into sample. And, with the sampling being done outside the instrument, the watch window was not available for verifying the values assigned to the various variables. It gets very tedious and challenging to verify this is correct. So, what do you do as the programmer?

The first thing one needs to do is flowchart the sampling specifications. This needs to be done to verify that all potential possibilities are covered. This can be very helpful in bringing out the "holes" in the logic. A "hole" in the sampling logic could result in a "fatal error" at the time the sampling program is run in production. Since there are so many possible scenarios, we may not discover this problem during testing.

Example of SASS Sampling flowchart

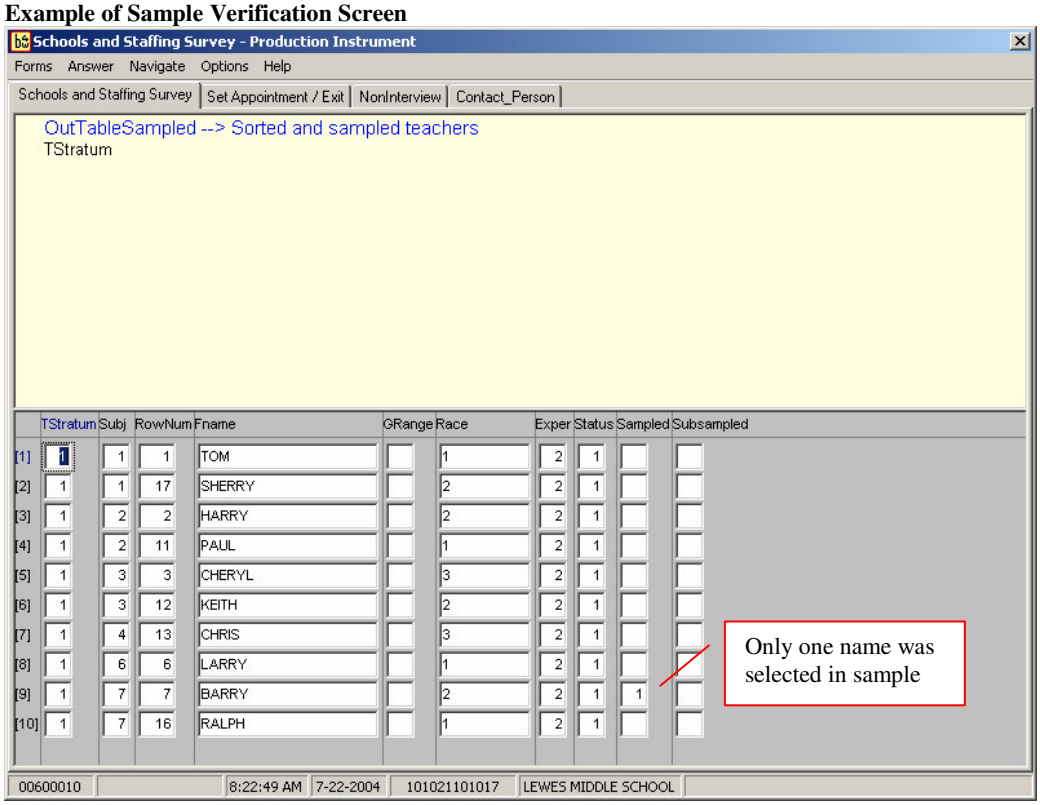


The next step is to program the logic. Since the Manipula code used is somewhat "different" (tricky) than other languages, the code didn't match up nicely to the specs. All of the logic required by the sampling specs couldn't be programmed in a straight forward manor using Manipula. To help verify the sampling specs were understood and that the Manipula code was working as desired, the programmer working on this project decided to code the sampling specs in Visual Basic as well. Results were displayed using XML and compared against the results from the Manipula sampling program. He then verified that the same results were received when testing the same scenario in each language. This not only helped resolve a few issues, but it gave us confidence that the sampling part was working correctly. This made things easier to test as well since the programmer could control the variables in the VB script – he didn't have to worry about the constant re-execution of rules changing the values of sampling variables, such as random number.

Now comes the tough part – testing. How to test and verify the sampling program is working correctly when the sampling takes place outside of the instrument?

A trick we used for testing was to add a "sampling results table" to the instrument that would display the results of the sampling. This table would only be displayed when we re-entered a completed case (this is not allowed in production). This table displayed the stratum assigned to each line, the sort order for sampling, and the sampling and subsampling flags assigned to each listed line. There were also other screens that displayed the values of all of the variables used in sampling. The sponsors and authors simply had to re-enter the case after it was sampled to see the results. This helped the testing process immensely. Note that in the example below, the listing has been resorted based on the sampling requirements. The listing is sorted by stratum, subject, and row number.

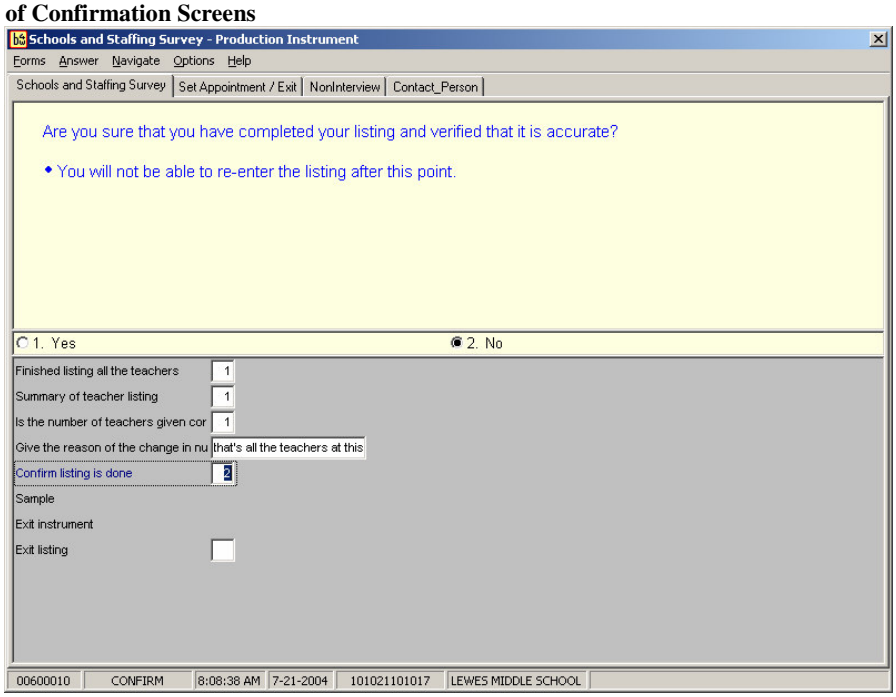
Example of Sample Verification Screen



Although these screens helped us in testing, having complex sampling in a listing instrument makes it very difficult to test. It literally creates thousands of possible scenarios that would need testing in order to verify sampling is working correctly in all of them. We can't possibly test all of them, so we focused on testing the most common ones.

Before running the complex sampling on the listed data, we first had to determine when the listing was complete. We added a series of questions that the FR would answer when they were absolutely sure the listing was done.

Example of Confirmation Screens



Once the FR enters "yes I'm sure it's complete", the instrument exits and the sampling runs. A completed outcome code is assigned to the case and passed out to the Manipula script. Once the Manipula script identifies the case as "complete", it executes the sampling portion of the code.

Manipula Transaction Script Used to Call Sampling

```
IF Upfile.Outcome = '801' THEN  {//This line prevents sampling if case
                                 is incomplete}
       Reslt5 := CALL('subsample /I' + THECASE) {// Susbsample the SASS
                                database's TeacherList table}

{=============================================================================
       Error Checks to verify that Sampling was run & display results
=============================================================================
IF Reslt5 <> 0 THEN
    DISPLAY('Error - Sampling program failed to execute.', WAIT)
  ELSE
    Reslt5             :=             RUN('C:\Progra~1\Intern~2\IEXPLORE.EXE
file:\\fld2\authors\SASS\Sampling\XMLOut.htm' {Sampled.htm}, WAIT)
  ENDIF

=============================================================================}
ENDIF
```

Listing Data from the Blaise database is pulled from the listing table and put into an array where it is stratified, sorted, sampled, and then sub-sampled if necessary. Once the sample lines are identified, the Blaise database is updated with results (assigned sampling and sub-sampling flags). The Blaise database is also updated with the modified sampling variable values used in sampling.

The manipula script then passes out the sample data to the control systems. The control systems are updated with the sampling information so that the FRs can then conduct and monitor the interviews.

Manipula Code used to write out Sampling Information to the Control Systems:

```
IF  (UPFILE1.OUTCOME  <>  '800')  AND  (UPFILE1.OUTCOME  <>  '802')  AND
(UPFILE1.OUTCOME <> '804') THEN         {Completed Case}
 OUTFILE.OutData  := '#ST1CID'
 OUTFILE.WRITE
 OUTFILE.OutData  := UPFILE1.ST1CID
 OUTFILE.WRITE
 OUTFILE.OutData  := '#ST1CNTNUM'
 OUTFILE.WRITE
 OUTFILE.OutData  := UPFILE1.ST1CNTNUM
 OUTFILE.WRITE
 OUTFILE.OutData  := '#TFOUTCOME1'
 OUTFILE.WRITE
 IF UPFILE1.OUTCOME = '801' THEN   {'good' case with data}
    {assign outcome code of the PAPI Questionnaires}
    IF UPFILE1.Teacher.OutTableFinal.TeachSamp[1].TCHRNM = EMPTY THEN
              {Questionnaire not needed - assigns non-sample outcome code}
       UPFILE1.Teacher.OutTableFinal.TeachSamp[1].OutComeCode := '247'
       UPFILE1.WRITE
    ENDIF
       {questionnaire outcome code initialized to 200}
  OUTFILE.OutData := UPFILE1.Teacher.OutTableFinal.TeachSamp[1].OutComeCode
 ELSE        {listing is a Non-Interview}
   OUTFILE.OutData := UPFILE1.SchOutcome
 ENDIF
 OUTFILE.WRITE
 OUTFILE.OutData  := '#STCHRNM1'
 OUTFILE.WRITE
 OUTFILE.OutData  :=upfile1.Teacher.OutTableFinal.TeachSamp[1].TCHRNM
 OUTFILE.WRITE
 OUTFILE.OutData  := '#SUBJ1'
 OUTFILE.WRITE
 OUTFILE.OutData  := STR(UPFILE1.Teacher.OutTableFinal.TeachSamp[1].SUBJ)
 OUTFILE.WRITE
   :

Endif {completed case}
```

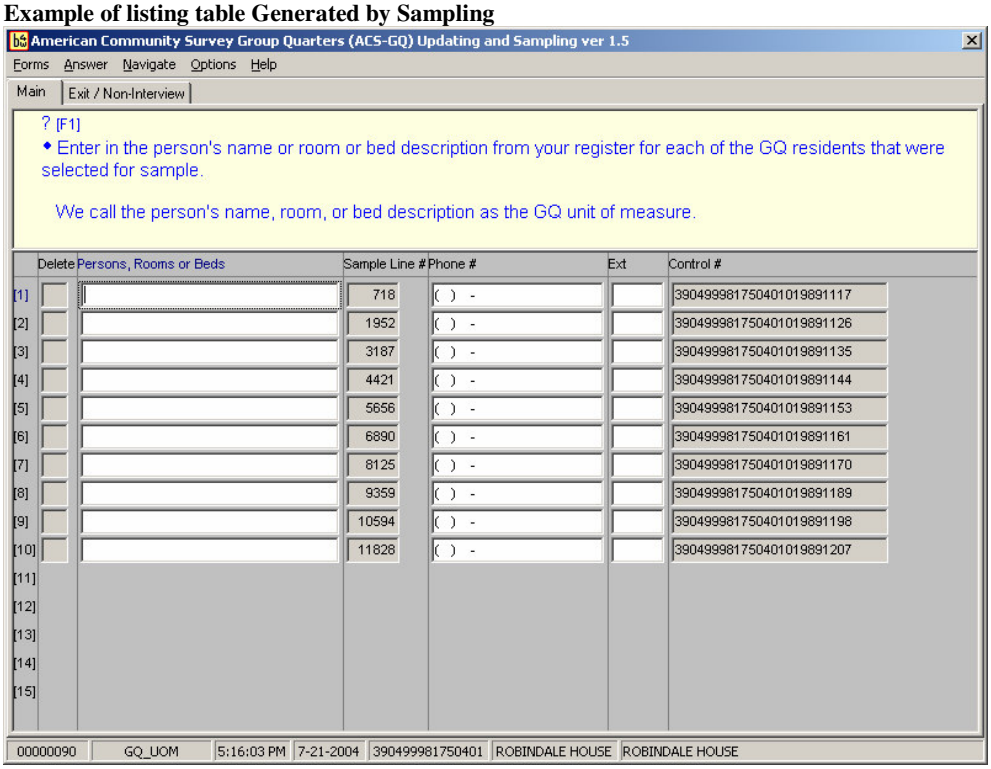
4.2 Don't Allow for Re-Execution of Sampling Component

If the listing instrument contains a sampling component, we must be careful not to allow it to re-execute the sampling piece. Once the FR answers "yes the listing is complete" and sampling is run, we can NOT allow the FR to modify anything that will change the sampling results.

For example, if we were conducting sampling inside of the instrument (in Blaise while the FR is collecting data) based on a number entered into a response and then used those sampling results to generate a table that contains sample line numbers that must be keyed, we can't allow the FR to go back and change that number (correct it) and resample. In other words, if that generated listing table is now partially complete, we can't allow the FR to back up and change the value of the "number" or do anything that would then cause Blaise to re-execute the sampling. If this happens, the listing table will most likely have new sample line numbers assigned which would then correspond to different information than that which was already keyed and would be incorrect (i.e., the wrong information is assigned to the sample rows). In the example below, 10 sample lines were generated from the sampling. The "sample line number" column tells the FR which information to key

from the printout or paper listing. If they had partially completed this table and then backed-up and done something to cause re-sampling, the "sample line numbers" would most likely change causing an incorrect listing.

Example of listing table Generated by Sampling



Depending on the listing, we either have the FRs restart the case and re-sample with the correct number or add in code to the instrument that will "refresh" all of the sample lines whenever the instrument re-samples. This way, we eliminate the possibility of incorrect information being assigned to the sampled lines. We just have to be careful not to "refresh" good data if that's the approach that's chosen.

4.3 Restarting a Case

For most of our interviewing surveys, FRs are allowed to "restart" a case. Restarting a case is basically allowing the FRs a "do-over". When a case is restarted, the current (partially complete) Blaise database and associated files for the case are replaced by the original files sent down for interview. The control systems have to then address any spawns or additional cases that were generated before the case was restarted. The same is true for listing surveys – the control systems have to figure out a way to handle any new cases generated or acted upon by the listing instrument.

For the Census Bureau, this is a little more challenging for listing surveys since the new cases generated by the listing instrument are typically used in a different survey study. If the listing survey generates an input file that is then used to "setup" a new study on the laptop as well as the case management systems, then both the case management system and the newly created study must be "corrected" if you allow for restarts. If the listing survey simply passes out data to the control system then only the case management system needs to be "corrected". This process becomes even more challenging if the FR has already transmitted in completed interviews from cases that were generated by the listing.

4.4 Testing Complex Sampling with the Systems

Just like testing a complex interview instrument, testing a listing instrument with complex sampling is very challenging. First, the author must make sure the sampling script is compiled with the latest instrument and that it's integrated in with the control scripts and Blaise instrument. Before sending off the instrument and sampling program for a systems test, the author should run it "stand alone" and verify that the sampling program is being executed at the appropriate time, the sampling program is producing sampling results, and that the results are in the correct format expected by the control systems.

The most critical piece needed for a good systems test is to have a good realistic input file that has a wide variety of sample cases that cover the most likely scenarios that one can run into during production. If the sampling input data used during a systems test is not realistic then the test is a waste of time.

5. Conclusion

A listing survey instrument is an instrument that is used as a tool by the FRs to collect and sample lists of data in order to generate sample cases for interviewing. These instruments typically don't produce output that is published. This tool has unique functionality that typically isn't required by interviewing instruments. Some of this functionality can be challenging for CAI software packages to handle.

Based on our experience of using Blaise to code a three different listing instruments and to prototype a fourth, we have concluded that Blaise can handle most of the functionality required for listing instruments used at the Census Bureau. The main weaknesses we've identified in using Blaise have to do with having to define an upper limit for our listing tables and the ability to handle complex sampling while running the DEP. Both of these issues can be worked around.

The Census Bureau plans to use Blaise in the conversion of our existing CA Clipper listing applications to a GUI CAI software package. We're confident that Blaise, can handle the required functionality of these applications.

Acknowledgements

I'd like to acknowledge the following authors for their input into this paper and/or work on the various Blaise listing applications coded at the Census Bureau:

Chris Borillo
Alexis Hunt
Roberto Picha