A Blaise Editing System at Westat

Rick Dulaney, Westat
Boris Allan, Westat

Introduction

Editing and delivering survey data pose challenges often quite separate from developing Blaise applications for data collection. Editing often requires that design objectives be modified, interviewer intentions inferred, and data reshaped to our clients' purposes. On longitudinal projects, the editing process must serve two masters: the need to deliver clean data in the current round and the need to field clean and consistent data for subsequent rounds of data collection.

We faced these challenges on a recent longitudinal project at Westat, in which we needed to produce data that accommodated statistical and analytic needs for a particular round of data collection, while processing the data for fielding in a subsequent round. Examination and experience showed that the requirement was to represent the data simultaneously in the Blaise "block" format for continued data collection in Blaise and in a "flat" format suitable for downstream processing on the current round. In addition, updates resulting from either activity had to be represented in both formats. Prior experience had also demonstrated that moving the data quickly out of Blaise into a statistical processing package created cross-platform contentions that made it difficult to iterate between the two databases.

In response, we developed a structured methodology that largely relied on the strength of the datamodel approach in Blaise for maintaining the structural integrity of the data during editing, and on the native ability in Blaise to move data between datamodels designed to represent the same data for different purposes. The basic systems components are:

- There are collection, holding, editing, and delivery databases (all in Blaise) to help keep each part of the process distinct.
- We use face sheets containing comments entered by interviewers, and an annotated list of structural edits failed when the data were checked in the collection database. These face sheets act as a physical record of decisions made, and actions taken.
- A tracking system to monitor and organize editors' work – the system has a list of those editors who are allowed to work on a particular project. Software produces a transaction record for each change in each editing session, so that we know what changes were made, and by whom.

This paper discusses our overall approach to editing complex survey data, first outlining some key issues and components that are common across many large-scale longitudinal surveys, and then describing the design and implementation of the editing system on a particular project. We first implemented this Blaise editing system during the editing and delivery of the Early Childhood Longitudinal Study - Kindergarten Cohort (ECLS-K), sponsored by the National Center for Education Statistics (NCES)[1]. The ECLS-K has two major purposes:

- To provide descriptive data on a nationally representative sample of children as they enter kindergarten, transition into first grade, and progress through fifth grade.
- To provide a rich data set that will enable researchers to analyze how a wide range of family, school, community and individual variables affect early success in school

[1] http://www.nces.ed.gov/ecls/

This editing model is now being used by other studies at Westat using Blaise data collection.

Editing Considerations

We use the term "edit" to refer to the review and update of survey data. The project context for editing may involve many considerations; those that we identified as key considerations on ECLS-K include:

Design vs. analytic requirements. In some cases, conscious decisions are made during the instrument design phase to accept inconsistent or ambiguous data rather than to challenge the respondent. However, the delivery plan often requires unambiguous variables for analysis, many of which may be derived from other variables. Examples abound in household enumeration; for example, the analytic requirement for father's level of education requires special editing rules in the case of multiple-father households. Or the Round 2 interviewer may have deleted the biological father entered in Round 1 and entered another biological father -- are they the same person? Which is correct? A second consideration arises when the field encounters situations that were unforeseen during the design and which therefore need to be accommodated during the editing process before the data can be made ready for analysis.

Data quality issues. Most survey systems, including Blaise, allow interviewers to make free-form comments analogous to the marginalia recorded on hard copy instruments. These remarks often contain information not represented in the actual survey data, and which may have an impact on the quality of the data, in the sense that survey data should be updated to reflect the remarks. In this sense also, coding open-ended or "other-specify" data can be considered as data quality issues, because again survey data are generally updated to reflect the content of these text fields. In some surveys, this coding operation may dominate other editing activities.

Structural considerations. A Blaise datamodel used as an instrument is normally optimized for data collection, to take advantage of Blaise system features for performing complex flow. Perhaps the most obvious element here is the use of blocks to modularize the instrument and to instantiate different levels of analysis within the instrument. For instance, a household-level questionnaire that includes an enumeration will normally be implemented using one or more blocks at the household level and at least one block at the person level, with each instance representing one household member. However, many analytic files are flat or "rectangular," with one record per household. Person-level data in this approach is repeated for each person on this single record. A successful editing and delivery system will make this transition as seamless as possible.

Timing issues. On longitudinal studies, the editing process is often subject to two constraints: the need to deliver and the need to re-field. If the editing operation discovers data that need updating, but too late to field corrected data in the next round, several problems may result. At the very least, the activity is inefficient, because the same updates will need to be applied after two rounds in separate editing operations. In addition, the incorrect data in the field may affect data collection so that either the wrong data are collected or the data are wrongly collected. Finally, making the same corrections multiple times introduces opportunities for error and therefore the need for reconciliation.

Editing Considerations on the ECLS-K

The ECLS-K began in fall 1998 by visiting approximately 1000 schools and sampling over 20,000 children. Each of these consenting children received an assessment using CAPI, and each cooperating household completed a parent interview conducted over the phone by a field interviewer using CAPI.

Westat conducted subsequent rounds in spring 1999, fall 1999, and spring 2000. Well over 100,000 CAPI interviews have been conducted so far over the four rounds.

The child assessment data is relatively straightforward; after a few screening questions, the child is administered reading, math, and science assessments. Each assessment generally includes a routing section, followed by a high, medium or low version of the assessment. Within the routing section and the assessment itself, there are few skips and no loops. The parent interview, however, is another matter; there is a household enumeration, several sections that are asked twice if twin children are sampled, and various miscellaneous sections that have individual data requirements. The parent interview generally ran from 45-60 minutes. (For completeness' sake, the ECLS-K also collects teacher- and school-level instruments, but these are administered using hard copy.)

After data collection, the data are coded, edited, and simultaneously prepared for re-fielding and extracted for delivery. On the delivery side, the data are weighted, imputed, and masked for public release. In addition, approximately 125 composite variables are created each round from the raw survey data and perhaps other sources such as the frame. This latter is quite important; most of the analytic requirements of the file surface during the development and construction of the composite variables. In many cases, the needs of the composites led to edits that supplemented the checks in the original datamodel. In other cases, the composite specifications supposed that the data were unambiguous; as discussed above, there were many exceptions. Finally, sometimes other activities, such as the coding, yielded data that was now inconsistent for analytic purposes with other survey data; the process of building the composites normally brought these inconsistencies to light.

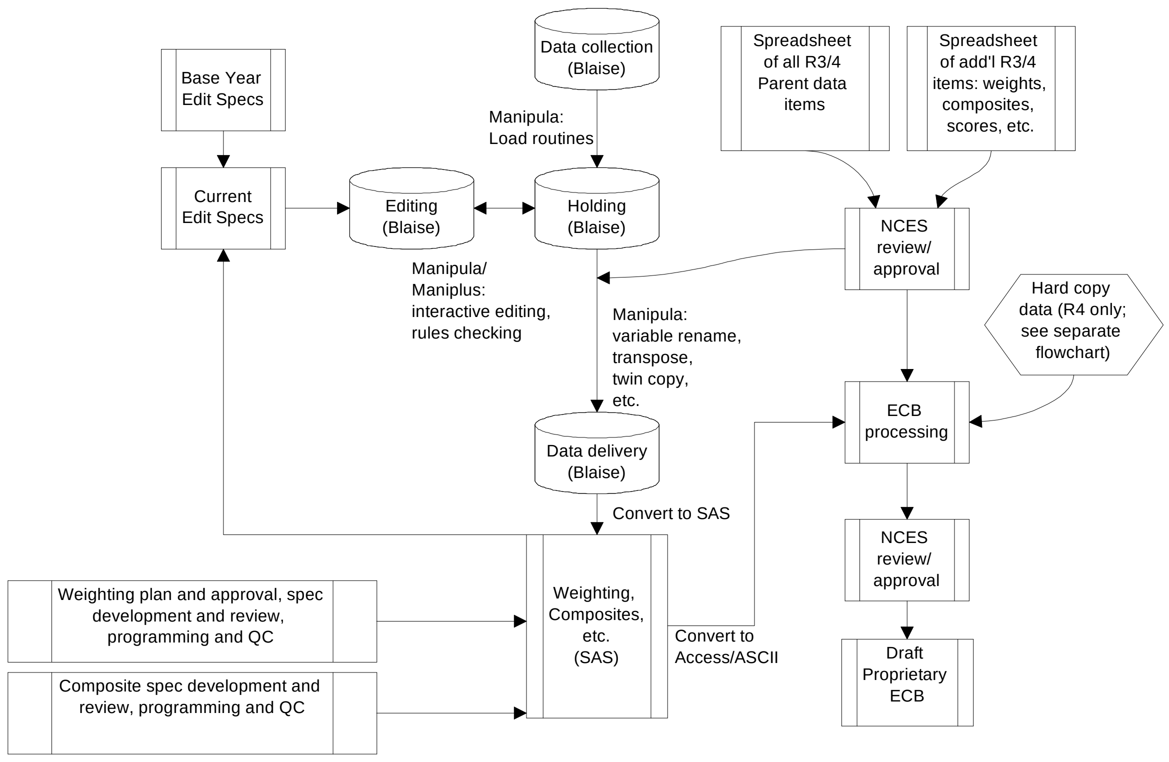
Based on our experience with other studies and the ECLS-K pilot, we identified a sequence of editing steps for the survey:

- **Comments review**
  Comments entered by interviewers during the course of the interview are extracted, converted to an Access database, reports are produced, and any necessary changes are made to interview data.
- **Resolving field problems**
  Field staff (interviewers and supervisors) report issues to the ECLS-K Help Desk, and in some cases this means changes have to be made to the interview after it has arrived at the home office.
- **Other-specify coding**
  The text strings are reviewed to see whether, in fact, the response should have been coded in one of the existing categories, or if a new category should be created.
- **Structural Edits**
  We identified and programmed (using Blaise Manipula) several specific edits. The specifications generally came out of one of the previous steps or from the composite variable process.
- **Prepare File for Delivery**
  Again using Blaise Manipula, we "rectangularize" each component (child and parent) in a new Blaise datamodel. We then use this rectangular datamodel to export the Blaise data to ASCII

The Design of the Blaise Editing System for ECLS-K

While much of the documentation necessarily focuses on data collection, the Blaise system is equally strong when used for data editing. The foundation of the technical design for Blaise editing in ECLS-K is the identification of datamodels that map to survey processing needs, and development of rules for transitioning from one model to another. Not including the original datamodel used to define the actual data collection, we identified three datamodels: a holding datamodel, an interactive editing datamodel, and a delivery datamodel.

The following flowchart shows these datamodels in the overall context of the ECLS-K Year 2 editing process. (Note: ECB stands for electronic codebook.)


Base Year Edit Specs
Current Edit Specs
Data collection (Blaise)
Manipula: Load routines
Editing (Blaise)
Holding (Blaise)
Manipula/ Maniplus: interactive editing, rules checking
Manipula: variable rename, transpose, twin copy, etc.
Data delivery (Blaise)
Convert to SAS
Weighting, Composites, etc. (SAS)
Weighting plan and approval, spec development and review, programming and QC
Composite spec development and review, programming and QC
Convert to Access/ASCII
Spreadsheet of all R3/4 Parent data items
Spreadsheet of add'l R3/4 items: weights, composites, scores, etc.
NCES review/ approval
Hard copy data (R4 only; see separate flowchart)
ECB processing
NCES review/ approval
Draft Proprietary ECB

The holding database serves as a repository for cases as they complete data collection or as they complete the editing. The datamodel is a superset of the data collection datamodel, that is, there are added variables and/or categories to support the different survey processes, but the entire block structure, type library and other features of the data collection datamodel are retained.

The editing datamodel is similar to the holding datamodel in structure at the block level, but some changes have been made to support the editing. For instance, some of our problems in later rounds were due to erroneous prior round data that were loaded before the problems were detected and cleaned. The editing datamodel therefore changed these from KEEP to ASK items so that they were available for editing. This is the database where the editors can go in and change the data on a case-by-case basis, whether editing or coding.

The delivery datamodel is entirely different from any preceding datamodels, in that the data are completely restructured into the units of analysis required for data delivery. In the ECLS-K case, all blocks need to be moved to the child level, meaning that there is one record per child, with no confidential data or operational items in the public-use file. If there are twins in a household, there are two records for that household, with the only differences between the records being data specific to the child. All repeating data (household enumeration, etc.) are repeated "horizontally" as array variables on the same record. In addition, any variables not needed for delivery (identifying text strings, operational "flags", etc.) are dropped. Also, all the variables are renamed as necessary to the names -- approved by the client after the start of data collection -- used in the final delivery file. This file is ready to go into SAS (or any other processing package), generally using the "out of the box" conversion routines that ship with Blaise.

Editing Tools

Several of the data processing steps are not actually tools, in the sense that they are not bundled in a programmer-independent way. Nonetheless, I list them here since they are necessary for the editing process to function, and to serve as a checklist for new projects:

- *Manipula code*: Manipula is used to move cases from data collection to holding, from holding to editing, from editing back to holding, and from holding to delivery. The last one is the most significant of these. Also note that these programs set codes that reflect the current status of the case as it moves through the editing process, so that we can report on how many cases are where. A detailed flowchart is given at the end of this paper showing the specific codes and flow for the ECLS-K editing.
- *Structural edits*: A collection of about 30 edits that clear up problems known to cause problems in the downstream processing. The specs are also reproduced at the end of the document.
- *Face sheets*: Face sheets output from the structural edits help guide the editors. We have found it useful to produce one face sheet for each case that needs attention, and include all data problems. A couple of examples are attached at the end of this document.
- *Rules checking*: When cases come back from editing to the holding database, we run a *rules check* that invokes rules of the holding Blaise datamodel and checks that all the data items are consistent with the skip patterns and other rules.

The tools that the ECLS-K editors use include:

1. **Interactive editing**. This is the heart of the system, controlling and monitoring access to cases and invoking Blaise:

    a. A shell program collects user information and stores it along with the case ID and start time, and invokes Blaise using the editing datamodel for this case.
    b. The editing takes place with static checking, meaning that skip patterns are not enforced, which allows the editor to move through a case and fix data items in the order most appropriate for the individual case.
    c. When the editor exits the case, the shell notes and saves the end time, thus allowing us to report actual editing time per case. (Our final data showed us at about 1.25 minutes per case.)

2. **Coding**. ECLS-K coding happens in two ways.

    - If there are not many actual cases to code (for example a particular other-specify with only a few hundred text responses or with few duplicates), we just use the interactive editing system. The editor jumps down to the question, codes the case by entering the numeric category in the question, and the editing text disappears.
    - Alternatively, if there are a large number of cases or if there is a predefined coding process we want to use, then we export the data into an appropriate database management system (we use Access), perform the coding, and import the data into the editing database using Manipula. We have developed a system that uses Cameleon to write the Blaise

metadata (blocks, arrays, variables, etc.) into Access tables, making the interface to other routines fairly straightforward.

3. **Data Viewer**. It is sometimes useful for the editors just to go in and look at a case, so we have a read-only data viewer that provides access to a copy of the data collection database.

Conclusion

Our experience on ECLS-K with editing the CAPI data using Blaise was very positive. We found it quite straightforward to process updates stemming from the editing considerations described above, while maintaining the full relational integrity of the database. An enormous advantage was our ability to iterate during the entire delivery processing cycle. For instance, if a downstream process such as composite creation or item imputation exposed the need for further editing, we were quickly able to perform the edit in Blaise and retransform the data into rectangular files. A second advantage was the flexibility offered by the multiple datamodel structure; when one piece of the system needed to change (if for instance, a new edit or an additional status code was added) the datamodel was updated and the data transformation routines automatically accommodated the changes.

This approach to data editing – implementing a structured methodology using Blaise – has proven effective in other areas as well. At Westat, we are developing variations of this system to edit hard copy data that has been keyed outside the Blaise system, that is, using Blaise on a project purely to edit data. We also expect the newly released Open Blaise Architecture will allow integration of Blaise editing systems more closely with other technologies.